

Oracle9i Lite

Windows CE 開発者ガイド

リリース 5.0

2001 年 7 月

部品番号 : J04361-01

ORACLE®

Oracle9i Lite Windows CE 開発者ガイド, リリース 5.0

部品番号 : J04361-01

原本名 : Oracle9i Lite Developer's Guide for Windows CE, Release 5.0

原本部品番号 : A90252-01

Copyright © 2001, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されております。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

* オラクル社とは、Oracle Corporation（米国オラクル）または日本オラクル株式会社（日本オラクル）を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的のみ使用されており、それぞれの所有者の商標または登録商標です。

目次

はじめに	xi
------------	----

1 概要

はじめに	1-2
アプリケーション開発	1-3
ツール	1-3
JDBC	1-4
ODBC	1-4
オブジェクト・カーネル API (OKAPI)	1-5
Mobile Sync API	1-5
Active Data Objects for Windows CE (ADOCE)	1-5
エミュレーション・モード	1-5
インストール	1-7
アンインストール	1-7
開発過程	1-8
開発システムの構成	1-8
パブリケーション項目の作成	1-9
宣言によるパブリケーション項目の作成	1-9
プログラムによるパブリケーション項目の作成	1-10
手動によるデータベースの作成	1-11
Windows CE デバイスの設定	1-11
Windows CE デバイスでのレプリケーション	1-12
Windows CE 用のモバイル・アプリケーションの開発	1-12
デバイスでのテスト	1-12
アプリケーションのパッケージ化	1-12

制限事項	1-13
------------	------

2 同期

概要	2-2
パブリケーションとサブスクリプション	2-2
クライアント・デバイス・データベースの DDL 操作	2-2
データベース・サポート	2-3
ユーザー定義の PL/SQL パッケージのバインド	2-3
ネーミングの柔軟性	2-3
ビューの高速リフレッシュおよび更新操作	2-3
更新可能な親表	2-3
親表のヒントと INSTEAD OF トリガーの使用	2-4
ビューの高速リフレッシュ	2-4
ビューの完全リフレッシュ	2-4
更新可能パブリケーション項目の外部キー制約	2-5
外部キー制約違反の例	2-5
BeforeApply および AfterApply での制約違反の回避	2-5
表の比率を使用した制約違反の回避	2-6
レプリケーション・エラーと競合	2-7
バージョニング	2-7
ウィニング・ルール	2-7
Oracle サーバーとデバイス間でのデータ型のマッピング	2-8
Oracle Lite データ型	2-8
パブリッシュ・サブスクライブ・モデル	2-9
Mobile サーバーへの接続	2-11
Mobile Server Admin API のユーザー関数	2-11
ユーザーの作成	2-11
パスワードの変更	2-11
ユーザーの削除	2-12
パブリケーションの作成	2-12
パブリケーション項目の定義	2-12
データのサブセット化	2-13
パブリケーション項目の作成	2-14
Null Sync コールアウト	2-15
パブリケーション項目名の取得	2-15

パブリケーション項目索引の作成	2-16
クライアント索引の定義	2-16
パブリケーションへのパブリケーション項目の追加	2-17
読取り専用パブリケーション項目	2-18
競合ルールの定義	2-18
表の比率の使用	2-19
シーケンスの作成	2-19
パブリケーションへのユーザーのサブスクライブ	2-20
クライアント・デバイスに対するシーケンスのパーティション化	2-20
パブリケーションに対するクライアント・サブスクリプション・パラメータの定義	2-21
サブスクリプションのインスタンス化	2-22
代替パブリケーション項目を使用したスキーマの発展	2-23
パブリケーション項目の変更	2-23
パブリッシュ・サブスクライブ・メソッド固有の関数	2-24
構成と適用を使用したコールバックのカスタマイズ	2-24
カスタマイズ DML 操作の定義	2-24
仮想主キー	2-25
仮想主キー列の作成	2-25
仮想主キー列の削除	2-25
制限選択条件	2-26
エラー・キューを使用した競合の解決	2-26
トランザクションの実行	2-26
トランザクションのページ	2-27
Mobile Sync API	2-27
ocSessionInit	2-27
ocSessionTerm	2-28
ocSaveUserInfo	2-28
ocDoSynchroize	2-29
ocSetTableSyncFlag	2-29
Mobile Sync API のデータ構造	2-30
ocEnv	2-30
ocTransportEnv	2-32
デバイス上の Mobile クライアントの設定	2-32
ディレクトリ構造	2-32
CESETUP.EXE ファイルの実行	2-33
トランスポートの構成	2-33
NT RAS の構成	2-33

デバイスの RAS User アカウントの作成	2-35
Windows デスクトップの RAS 設定	2-35
Windows デスクトップの設定	2-36
デバイスへのサンプル・アプリケーションのインストール	2-36
同期のテスト	2-36
Mobile サーバーのシステム・カタログ・ビュー	2-38

3 ActiveX Data Objects for Windows CE

概要	3-2
構成	3-2
機能	3-3
ActiveX Data Objects for Windows CE のオブジェクト	3-4
Active Connection オブジェクトのメソッド	3-5
Connect	3-5
Disconnect	3-5
Recordset オブジェクトのメソッド	3-6
AddNew	3-7
CancelUpdate	3-8
Clone	3-9
Close	3-9
Delete	3-10
GetRows	3-11
Move	3-13
MoveFirst、MoveLast、MoveNext、MovePrevious	3-14
Open	3-15
Supports	3-17
Update	3-19
Recordset オブジェクトのプロパティ	3-20
AbsolutePage	3-21
AbsolutePosition	3-22
ActiveConnection	3-23
BOF、EOF	3-23
Bookmark	3-24
CacheSize	3-25
Count	3-25
CursorType	3-26
EditMode	3-26

LockType	3-27
PageCount	3-28
PageSize	3-29
RecordCount	3-29
Source	3-30
Field オブジェクトのプロパティ	3-30
ActualSize	3-31
Attributes	3-31
DefinedSize	3-33
Name	3-33
Type	3-34
UnderlyingValue	3-35
Value	3-36
Recordset オブジェクトのコレクション	3-36
Fields	3-36
SQL およびデータベースの参照	3-37
エラー・メッセージ	3-37
ActiveX Data Objects のエラー	3-38
SQL エラー	3-39

4 パッケージ・ウィザードの使用

パッケージ・ウィザードの概要	4-2
パッケージ・ウィザードの起動	4-2
新規アプリケーションの命名	4-3
プラットフォーム・ファイルの命名	4-5
アプリケーション・ファイルのリスト表示	4-5
ソート	4-6
フィルタ	4-6
データベース情報の入力	4-7
レプリケーション用スナップショットの定義	4-8
新規スナップショットの作成	4-12
スナップショットのインポート	4-13
スナップショットの編集	4-15
アプリケーションの完了	4-16
XML ファイル	4-17
JAR ファイルの作成	4-17

SQL ファイルの作成	4-17
パッケージ・ウィザードの再起動	4-18
アプリケーションのパブリッシュ	4-18
アプリケーションの編集	4-19

A Mobile サーバーのシステム・カタログ・ビュー

"MOBILEADMIN".CV\$ALL_CLIENTS	A-2
"MOBILEADMIN".CV\$ALL_PUBLICATIONS	A-2
"MOBILEADMIN".CV\$ALL_SUBSCRIPTIONS	A-2
"MOBILEADMIN".CV\$ALL_SEQUENCES	A-3
"MOBILEADMIN".CV\$ALL_SEQUENCE_PARTITIONS	A-3
"MOBILEADMIN".CV\$ALL_PUBLICATION_ITEMS_ADDED	A-3
"MOBILEADMIN".CV\$ALL_PUBLICATION_ITEMS	A-4
"MOBILEADMIN".CV\$ALL_PUBLICATION_INDEXES	A-4
"MOBILEADMIN".CV\$ALL_SUBSCRIPTION_PARAMS	A-5

B Oracle Lite ユーティリティ

CREATEDB	B-2
DECRYPDB	B-3
ENCRYPDB	B-4
Mobile SQL	B-6
データベース・アクセス	B-6
Mobile SQL の起動	B-6
ODBINFO	B-7
REMOVEDB	B-9

用語集

索引



1-1	開発インタフェース	1-3
2-1	「Mobile Sync」画面	2-37
4-1	「ようこそ」パネル	4-3
4-2	「アプリケーション」パネル	4-4
4-3	「ファイル」パネル	4-6
4-4	「フィルタ」パネル	4-7
4-5	「データベース」パネル	4-8
4-6	「スナップショット」パネル	4-11
4-7	「新規スナップショット」ウィンドウ	4-12
4-8	「データベースへの接続」ウィンドウ	4-14
4-9	「表」ウィンドウ	4-14
4-10	「スナップショットの編集」ウィンドウ	4-15
4-11	「アプリケーションの定義が完了しました。」ウィンドウ	4-17
4-12	「アプリケーションをパブリッシュします。」ウィンドウ	4-18

表

2-1	Oracle Lite データ型	2-8
2-2	パブリッシュ・サブスクライブ・モデルの要素	2-9
2-3	パブリッシュ / サブスクライブ・モデルを実装する方法	2-10
2-4	ユーザー作成パラメータの例	2-11
2-5	パスワード設定パラメータの例	2-12
2-6	ユーザー削除パラメータの例	2-12
2-7	パブリケーション作成パラメータの例	2-13
2-8	パブリケーション項目作成パラメータの例	2-14
2-9	パブリケーション項目名取得パラメータの例	2-15
2-10	パブリケーション項目索引作成パラメータの例	2-16
2-11	パブリケーション項目追加パラメータの例	2-17
2-12	サブスクリプション作成パラメータの例	2-20
2-13	シーケンス・パーティション作成パラメータの例	2-21
2-14	サブスクリプション・パラメータ設定パラメータの例	2-22
2-15	サブスクリプションのインスタンス化パラメータの例	2-22
2-16	パブリケーション項目変更パラメータの例	2-23
2-17	仮想主キー列作成パラメータ	2-25
2-18	仮想主キー列の削除パラメータ	2-26
2-19	トランザクション実行パラメータ	2-26
2-20	トランザクションのページ・パラメータ	2-27
2-21	Mobile Sync Client のパラメータ	2-36
3-1	ADOCE のオブジェクト型	3-3
3-2	Active Connection オブジェクトのメソッド	3-5
3-3	Connect のパラメータ	3-5
3-4	Recordset オブジェクトのメソッド	3-6
3-5	AddNew のパラメータ	3-7
3-6	Delete のパラメータ	3-10
3-7	GetRows のパラメータ	3-11
3-8	Move のパラメータ	3-13
3-9	Open のパラメータ	3-15
3-10	Supports のパラメータ	3-18
3-11	Update のパラメータ	3-19
3-12	Recordset オブジェクトのプロパティ	3-20
3-13	BOF および EOF のパラメータ	3-24
3-14	Bookmark のパラメータ	3-24
3-15	Count のパラメータ	3-25
3-16	編集モードの戻り値	3-27
3-17	ロック・モードの戻り値	3-28
3-18	Field オブジェクトのプロパティ	3-30
3-19	Attributes の戻り値	3-32
3-20	Type の戻り値	3-34
3-21	ADOCE のシステム表	3-37
3-22	ADOCE のエラー	3-38

3-23	SQL エラー	3-39
4-1	「ようこそ」 パネルのオプション	4-2
4-2	「アプリケーション」 パネルのオプション	4-3
4-3	「ファイル」 パネルのオプション	4-5
4-4	「データベース」 パネルのオプション	4-7
4-5	スナップショット・パラメータ	4-9
4-6	「新規スナップショット」 パネルのオプション	4-13
4-7	「スナップショットの編集」 ウィンドウのオプション	4-15
4-8	「アプリケーションをパブリッシュします。」 ウィンドウのオプション	4-19
A-1	ALL_CLIENTS のパラメータ	A-2
A-2	ALL_PUBLICATIONS のパラメータ	A-2
A-3	ALL_SUBSCRIPTIONS のパラメータ	A-2
A-4	ALL_SEQUENCES のパラメータ	A-3
A-5	ALL_SEQUENCE_PARTITIONS のパラメータ	A-3
A-6	ALL_PUBLICATION_ITEMS_ADDED のパラメータ	A-3
A-7	ALL_PUBLICATION_ITEMS のパラメータ	A-4
A-8	ALL_PUBLICATION_INDEXES のパラメータ	A-4
A-9	ALL_SUBSCRIPTIONS_PARAMS のパラメータ	A-5
B-1	ツールとユーティリティ	B-1
B-2	DECRYPDB のリターン・コード	B-3
B-3	ENCRYPDB のリターン・コード	B-5
B-4	ODBInfo のパラメータ	B-7

はじめに

『Oracle9i Lite Windows CE 開発者ガイド』では、Oracle Lite とそのコンポーネントを紹介
します。このガイドでは、Oracle Lite データベースの開発、配布およびメンテナンスの方法
を説明します。

内容は次のとおりです。

第 1 章「概要」

Oracle Lite データベースおよび Mobile
Development Kit for Windows CE の概要を説明し
ます。

第 2 章「同期」

Mobile Sync を使用したレプリケーションについ
て説明します。Mobile Sync の機能、データ型の
マッピング、トランスポートの構成、パブリッ
シュ / サブスクライブ・モデル、ウィニング・
ルール、索引およびシーケンスなどが含まれてい
ます。

第 3 章「ActiveX Data Objects for Windows CE」

Oracle Lite ActiveX Data Objects for Windows CE
について説明します。

第 4 章「パッケージ・ウィザードの使用」

パッケージ・ウィザードを使用してアプリケー
ションを作成および配布する方法を説明します。

付録 A「Mobile サーバーのシステム・ カタログ・ビュー」

システム・カタログ内のオブジェクト型について
説明します。

付録 B「Oracle Lite ユーティリティ」

Oracle Lite で使用できるデータベースのツールと
ユーティリティについて説明します。

この章では、Mobile Development Kit for Windows CE を使用したアプリケーション開発の概要（要件、製品コンポーネント、インストール手順など）を説明します。内容は次のとおりです。

- [はじめに](#)
- [アプリケーション開発](#)
- [インストール](#)
- [アンインストール](#)
- [開発過程](#)
- [制限事項](#)

はじめに

今回のリリースの Oracle Lite データベースは、Windows CE 2.0（およびそれ以上の）デバイスで稼働します。Oracle Lite を使用すると、標準 SQL を使用したエンタープライズ・アプリケーションを、エンタープライズ・サーバーと自動的にデータを同期しながら、Windows CE 上で実行できます。

Windows CE 上の Oracle Lite データベースには、標準の ODBC インタフェースまたは JDBC インタフェースが提供されます。データ・レプリケーションにより、分散ユーザーは Mobile サーバーと同期した軽量データベースにアクセスできます。Oracle Lite データベース for Windows CE は、副問合せスナップショットを使用して双方向の同期レプリケーションをサポートします。副問合せスナップショットを使用すると、リモート・デバイスや接続が切断されているデバイスごとに必要なデータ・サブセットを簡単に定義できます。副問合せスナップショットの詳細は、[第 2 章「同期」](#)を参照してください。

Oracle Lite データベース for Windows CE は、次に示す転送メカニズムのいずれかを使用して、イーサネット、ワイヤレス、モデム、およびシリアル的各接続を介するレプリケーションをサポートします。

- HTTP（Mobile サーバーとともに使用する場合）。詳細は、[第 2 章「同期」](#)の「[デバイス上の Mobile クライアントの設定](#)」の項を参照してください。
- ファイル・ベース。

Activesync、SQL*Net および Oracle Net8i レプリケーションは現時点ではサポートされていません。Activesync は、コピー・ファイルに使用されることがあります。

注意： Oracle Lite データベース for Windows CE データベース・ファイルは、デスクトップ上の Oracle Lite データベース・ファイル（.ODB ファイル）と互換性があります。この 2 つの環境間でデータベース・ファイルをコピーできます。

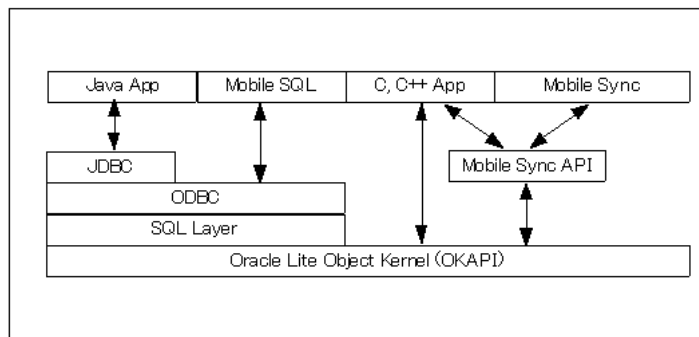
また、Oracle Lite データベース for Windows CE データベース・ファイル（.ODB ファイル）はフラッシュ・メモリーに格納できるので、格納領域を拡大できます。ただし、フラッシュ・メモリーから Oracle Lite データベースへの直接アクセスは、現在はサポートされません。

アプリケーション開発

Windows CE デバイス用のアプリケーションは、最初にデスクトップ上で開発およびテストされ、その後でデバイスにダウンロードされるのが一般的です。Oracle Lite データベース for Windows CE では、次のようなクロス環境開発モデルが使用できます。

- Windows NT ワークステーション上で Windows CE エミュレータを使用できます。これにより、再コンパイルするたびに Windows CE デバイスにダウンロードする必要がないため、開発およびテスト期間を短縮できます。
- アプリケーションを Windows CE デバイス上で直接テストできます。

図 1-1 開発インターフェース



ツール

開発およびデバッグを効率よく行うには、デスクトップから Mobile SQL などのツールを使用します。これにより、Windows CE デバイス上の Oracle Lite データベースへのアクセスが可能になります。この場合は、単に接続文字列として ODBC:CE を使用します。

次に示す、Oracle Lite データベースのその他のツールとユーティリティも Windows CE デバイス上で使用できます。

- CREATEDB
- REMOVEDB
- ODBINFO
- ENCRYPDB
- DECRYPDB

これらのユーティリティの詳細は、[付録 B「Oracle Lite ユーティリティ」](#)を参照してください。

注意： Mobile Development Kit for Windows CE は、Visual C++ 6.0 (Windows CE 対応) に対して認証されています。Mobile Development Kit は、CE Tools 3.0 に対しては認証されていません。

JDBC

Java Database Connectivity (JDBC) インタフェースは、Java アプリケーション用に、ODBC に似た SQL データベース・インタフェースを提供する Java クラス・セットを指定します。Java Development Kit (JDK) の主要部分の一部である JDBC は、リレーショナル・データベースに対するオブジェクト・インタフェースを提供します。Oracle Lite データベースでは、Oracle Lite データベースの JDBC (Type-2) ドライバを通じて JDBC をサポートします。このドライバは JDBC コールを解釈して Oracle Lite データベースに渡します。Mobile Development Kit には JDK 1.1.8 が必要です。

詳細は、`Oracle_Home¥Mobile¥SDK¥Examples¥Java` ディレクトリを参照してください。

ODBC

Windows CE Toolkit は、**SQL.H** などの ODBC ヘッダー・ファイルを Visual C++ for Windows CE のインクルード・ディレクトリにインストールしません。Visual C++ を使用して ODBC アプリケーションをコンパイルするには、ヘッダー・ファイルを Windows CE のインクルード・ディレクトリに手動でコピーする必要があります。

ODBC ドライバ・マネージャは、Windows CE ではサポートされていません。Visual C++ ODBC アプリケーションで Oracle Lite データベースをデータベースとして使用するには、次の手順を実行する必要があります。

- Visual C++ および Windows CE Toolkit for Visual C++ をインストールします。
- ファイル **SQL.H**、**SQLEXT.H**、**SQLTYPES.H** および **SQLUCODE.H** を
`¥program files¥devstudio¥vc¥include`
から
`¥program files¥devstudio¥wce¥include¥wce200` にコピーします。
この例では、Visual C++ は `c:¥DevStudio` ディレクトリにあると想定しています。
- **SQLTYPES.H** ファイルの `#include <wchar.h>` 行を `#include <tchar.h>` に変更します。
- アプリケーションを作成する際、コンパイル時フラグ `-D SQL_NOUNICODEMAP` を指定します。

- `Oracle_Home¥Mobile¥SDK¥wince¥platform¥Lib` ディレクトリにある **olod2040.LIB** ライブラリにアプリケーションをリンクします (*platform* は MIPS、SH-3、SH-4、ARM、x86 です)。

オブジェクト・カーネル API (OKAPI)

OKAPI は、Oracle Lite オブジェクト・カーネルへのアプリケーション・プログラミング・インタフェース (API) です。OKAPI は、次のデータベース機能をサポートします。

- 実行時のクラスの作成とクラス情報へのアクセス
- オブジェクト識別情報とナビゲーションに基づく直接オブジェクト・アクセス
- オブジェクトのクラスタ化とグループ化
- クラスおよびそのサブクラスに対する問合せ
- オブジェクトのネーミングとオブジェクト間の関連
- バイナリ・ラージ・オブジェクト (Binary Large Object: BLOB) データ
- トランザクションおよびクラッシュ・リカバリ

詳細は、『Oracle Lite (C および C++) オブジェクト・カーネル API リファレンス』を参照してください。

Mobile Sync API

この API を使用すると、アプリケーションはレプリケーション・プロセスをプログラムで制御できます。アプリケーションは Mobile Sync API の関数を呼び出して、レプリケーション・プロセスを開始し、Mobile Sync API により生成されるエラー・メッセージを獲得します。詳細は、[第 2 章「同期」](#)の「[Mobile Sync API](#)」の項を参照してください。

Active Data Objects for Windows CE (ADOCE)

ActiveX Data Objects はプログラミング・インタフェースです。これを使用すると、Visual Basic アプリケーションで Oracle Lite データベースの機能にアクセスできます。この開発インタフェースの詳細は、[第 3 章「ActiveX Data Objects for Windows CE」](#)を参照してください。

エミュレーション・モード

Oracle Lite データベース for Windows CE は、デスクトップ上でのエミュレーション・モードをサポートします。エミュレーション・モード用のアプリケーションを作成するには、`Oracle_Home¥bin` ディレクトリにあるデスクトップ版 (32 ビット) の Oracle Lite DLL にリンクします。

たとえば、次の手順に従うと **SAMPLE.EXE** アプリケーションのエミュレーションをサポートできるようになります。手順を開始する前に、「**ODBC**」の項にリスト表示されている手順を完了していることを確認してください。

1. ユーティリティ **CREATEDB** を使用して新規データベースを作成します。このユーティリティの詳細は、付録 B「**Oracle Lite ユーティリティ**」の「**CREATEDB**」の項を参照してください。
2. 「**WCE MFC AppWizard (exe)**」を選択して、新規 **SAMPLE** プロジェクトを **C++** で作成します。アプリケーション・タイプとして **ダイアログ・ベース** を選択して、「終了」をクリックします。
3. プロジェクトをいったん終了し、*Oracle_Home¥Mobile¥SDK¥wince¥samples¥odbc* ディレクトリにある元の **.CPP** ファイルおよび **.H** ファイルを、新規プロジェクト用のディレクトリにコピーします (**StdAfx.*** ファイルはコピーしないでください)。
4. ファイル **dbaccess.cpp** を、ソース・ファイルとして **FileView** の下に追加します。また、ファイル **sampleDlg.cpp** をソース・ファイル群から削除します。
5. ファイル **MFCCE20D.DLL** をディレクトリ
¥program files¥devstudio¥wce¥mfc¥lib¥wce200¥x86em
からディレクトリ
¥program files¥windows ce platform sdk¥wce¥emul¥hpc にコピーします。
6. 新しいコードをデバッグ・モードでコンパイルします。

注意： Windows CE エミュレータではソケットは動作しないため、**Oracle Lite** データベースのレプリケーションはエミュレーション・モードではサポートされていません。

7. Windows CE エミュレータ・データベースに接続するには、既存の Windows POLITE データ・ソース名 (DSN) を使用します。

インストール

Mobile Development Kit for Windows CE のインストールの詳細は、『Oracle9i Lite インストールेशनおよび構成ガイド』を参照してください。

Mobile Development Kit for Windows CE をインストールした後、Windows CE デバイスにファイルをコピーする必要があります。デバイスに接続している間に、デスクトップ上の Oracle_Home¥Mobile¥SDK¥wince ディレクトリにある **CESETUP.EXE** ファイルを実行します。

アンインストール

Windows CE の Remove Programs ツールでは、Oracle Lite for Windows CE はデバイスから完全には削除されません。次の手順を手動で実行する必要があります。

注意： デバイスから Oracle Lite を削除する前に、関連ファイルに依存している他のアプリケーションがないことを確認します。

1. **CESETUP.INI** ファイルのバックアップ・コピーを作成します。次に、**CESETUP.INI** ファイルの内容を次のように置き換えます。

```
[DELETE]
¥ORACE¥odbc.txt
¥ORACE¥polite.txt
¥ORACE¥olite40.msb
¥Windows¥ATLCE211.DLL
¥Windows¥MFCCE211.DLL
¥Windows¥ceansi.dll
¥Windows¥olobj40.dll
¥Windows¥oljdbc40.dll
¥Windows¥olod2040.dll
¥Windows¥olcl2040.dll
¥Windows¥olsql40.dll
¥Windows¥oladoce.dll
¥Windows¥REGSVRCE.EXE
¥ORACE¥createdb.exe
¥ORACE¥removedb.exe
¥ORACE¥decrypdb.exe
¥ORACE¥encrypdb.exe
¥ORACE¥odbinfo.exe
¥ORACE¥remodbc.exe
¥ORACE¥olodbcce.exe
¥ORACE¥msql.exe
¥Windows¥ocapi.dll
```

```
¥Windows¥zlibce.dll
¥ORACE¥mSync.exe
¥ORACE¥oljdkce.jar
¥ORACE¥olitejdbc40.jar
```

次にデスクトップから **CESETUP.EXE** を実行します。

開発過程

この項では、Mobile Development Kit for Windows CE の開発過程の概要を説明します。この処理に含まれる手順は、次のとおりです。

1. [開発システムの構成](#)
2. [パブリケーション項目の作成](#)
3. [Windows CE デバイスの設定](#)
4. [Windows CE デバイスでのレプリケーション](#)
5. [Windows CE 用のモバイル・アプリケーションの開発](#)
6. [デバイスでのテスト](#)
7. [アプリケーションのパッケージ化](#)

開発システムの構成

Mobile Development Kit による開発を開始するには、Windows で実行する開発システムに次のコンポーネントをインストールする必要があります。

- Mobile サーバー
- Mobile Development Kit
- Windows CE 用統合開発環境 (IDE)

サーバーおよび開発環境のインストールおよび構成の詳細は、[第 2 章「同期」の「デバイス上の Mobile クライアントの設定」](#)の項を参照してください。

パブリケーション項目の作成

次の方法を使用すると、パブリケーション項目を作成できます。パブリケーション項目は、同期が発生したときにモバイル・データベース上に自動的に表を作成します。パブリケーション項目の作成には、次の3つの方法があります。

1. **宣言によるパブリケーション項目の作成** – パッケージ・ウィザードを使用してパブリケーション項目を作成します。これがお薦めの方法です。
2. **プログラムによるパブリケーション項目の作成** – Mobile Server Admin API を使用して、パブリケーション項目をプログラムで作成します。
3. **手動によるデータベースの作成** – Mobile データベースを手動で作成します。

宣言によるパブリケーション項目の作成

この方法では、Mobile サーバーに含まれているパッケージ・ウィザードを利用します。アプリケーションを Mobile サーバー・リポジトリにパブリッシュするときにパブリケーション項目が自動的に作成されるため、プログラミングは不要です。開発者は同期処理を実行する必要があります。この処理により、パッケージ・ウィザードに提供されるスナップショット（データ・サブセット）情報に基づいて、Mobile データベースのパブリケーション項目オブジェクト（表や索引など）が自動的に作成されます。パッケージ・ウィザードは、スナップショット定義を含むアプリケーションをパッケージ化して Mobile サーバー・リポジトリにパブリッシュするために使用します。Mobile サーバーは、スナップショット以外のアプリケーション情報が実装されていない場合でも、データの同期に必要なレプリケーション環境を作成します。

このツールの便利な点は、開発者が Mobile データベースを作成する際に、安全でエラーが発生しにくいことです。このツールを使用するには、実際のアプリケーション・プログラミングを開始する前に、次の手順を実行する必要があります。

- アプリケーションのパッケージ化
- Oracle データベース・サーバーでのデータベースの作成
- Mobile サーバー・リポジトリへのアプリケーションのパブリッシュ
- Mobile クライアントの設定
- アプリケーションおよびデータの配布

Mobile サーバーのアーキテクチャは、集中サーバーを使用して Mobile アプリケーションを管理および配布できるように設計されています。使用の手順については、[第4章「パッケージ・ウィザードの使用」](#)で説明しています。

プログラムによるパブリケーション項目の作成

データベースを作成しデータを移入する 2 つ目の方法は、Mobile Server Admin API を使用してプログラムでパブリケーション項目を作成する方法です。パブリケーション項目には、パブリケーションおよびサブスクリプションなどのデータ・レプリケーション・オブジェクトが含まれます。Mobile Server Admin API を起動する前に、データベース・スキーマが必要になります。分散データベース・スキーマを作成するには、次の基本手順が必要です。

- パブリケーションの作成
- パブリケーション項目の作成
- ユーザー ID の作成
- サブスクリプションの作成

パブリケーションの作成

パブリケーションは、表のサブセット化定義や索引などのメタデータを含むテンプレート・グループです。パブリケーションは、Mobile Server Admin API を使用して作成できます。この API には、パブリッシュ・サブスクライブ・モデルを実装する Java 関数が含まれています。これらの関数は、Java プログラムの内部から標準のファンクション・コールとしてコールできます。

パブリケーション項目の作成

パブリケーション項目は、クライアントがアクセスできるデータ・サブセットを指定する SQL の SELECT 文です。パブリケーション項目は、通常クライアント・デバイス上のレプリカ表に対応します。パブリケーション項目は、Mobile Server Admin API を使用して作成できます。この API には、パブリッシュ・サブスクライブ・モデルを実装する Java 関数が含まれています。これらの関数は、Java プログラムの内部から標準のファンクション・コールとしてコールできます。

ユーザー ID の作成

各クライアントはユーザー ID により識別されます。開発の目的上、データ・サブスクリプションを特定ユーザーに対応付けるために、ユーザー ID は Mobile Server Admin API を使用して作成する必要があります。

サブスクリプションの作成

サブスクリプションは、ユーザーをパブリケーションにリンクします。サブスクリプションは、Mobile Server Admin API を使用して作成できます。この API には、パブリッシュ・サブスクライブ・モデルを実装する Java 関数が含まれています。これらの関数は、Java プログラムの内部から標準のファンクション・コールとしてコールできます。Java を使用してパブリケーションおよびサブスクリプションを作成する方法は、[第 2 章「同期」の「パブリッシュ・サブスクライブ・モデル」の項](#)を参照してください。

重要： プログラムによるパブリケーション項目の作成は、Java に関する高度なスキルと Mobile Server Admin API が必要で、労力のかかる作業です。第 4 章「パッケージ・ウィザードの使用」で説明されているパッケージ・ウィザードの使用をお勧めします。これを使用すると手順は多くなりますが、容易に作業できます。

手動によるデータベースの作成

Oracle Lite データベースを作成する 3 つ目の方法は、SQL を使用する方法です。Mobile Development Kit にはいくつかのツールがバンドルされており、これらのツールを使用すると、基本データベースを手動で作成し、なんらかのデータを挿入できます。Mobile SQL は、この目的のために使用できるアプリケーションです。

この方法を使用して開発する場合は、後でパッケージ・ウィザードを使用して指定されるスナップショット定義が、手動で作成された表定義と完全に一致することを確認する必要があります。これには、スナップショット情報を注意深く定義する必要があります。レプリケーション中に作成されたスナップショットが手動で作成された表定義に一致しない場合は、アプリケーション・エラーが発生します。詳細は、付録 B「Oracle Lite ユーティリティ」の「Mobile SQL」の項を参照してください。

Windows CE デバイスの設定

開発過程の 3 番目の手順は、開発システムに Oracle Lite データベースを作成するためのデータをレプリケートします。レプリケーションを実行する前に、Mobile Development Kit for Windows CE デバイスのランタイム・ライブラリをインストールしておく必要があります。

Mobile サーバーと Windows CE デバイスの間のトランスポートを構成した後で、Mobile クライアントをインストールし構成する必要があります。同様に、Mobile サーバーと同期する前に、ユーザーおよび接続パラメータをデバイスに入力して保存する必要があります。

構成およびインストールの詳細は、第 2 章「同期」の「デバイス上の Mobile クライアントの設定」の項を参照してください。

Windows CE デバイスでのレプリケーション

開発過程の 4 番目の手順には最初のレプリケーション・サイクルが含まれ、このサイクルで Oracle のデータベース・サーバーから開発システム上の Oracle Lite データベースにデータをレプリケートします。Windows CE デバイスのランタイム・ライブラリは、Windows マシン上で Windows CE デバイスをシミュレートします。これによって、データ・レプリケーションの監視とテストが容易になります。サンプルの Oracle Lite データベースは、Mobile Development Kit に含まれているサンプル・ファイルをインストールしたときに Windows 開発システム上に作成されます。

サンプル・ファイルのインストールの詳細は、『Oracle9i Lite インストレーションおよび構成ガイド』を参照してください。

Windows CE 用のモバイル・アプリケーションの開発

開発過程の 5 番目の手順はアプリケーションの開発です。デバイス上でサーバー・データを同期した後、Windows CE IDE を使用して Windows 開発システム (PC) 上でアプリケーションを開発します。Windows CE アプリケーションを Windows CE デバイスに配布および実行する前に、そのデバイスを使用してアプリケーションのテストを試みる必要があります。

アプリケーションの開発の詳細は、第 3 章「ActiveX Data Objects for Windows CE」を参照してください。

デバイスでのテスト

開発過程の 6 番目の手順では、実際の Windows CE Computing 携帯端末でアプリケーションをテストします。実際のデバイスで Windows CE アプリケーションをテストするには、開発者が手動でインストールと構成の作業を実行する必要があります。CEServices ツールを使用して、ランタイム・ライブラリとアプリケーション・ライブラリを手動で Windows CE Computing 携帯端末にインストールします。ただし、ランタイム・ライブラリとアプリケーション・ライブラリを簡単にインストールする方法は、『Oracle9i Lite パブリッシュおよびディプロイ・ガイド』で説明されています。

アプリケーションのパッケージ化

開発過程の 7 番目の手順には Windows CE Computing アプリケーションを開発およびテストした後、そのアプリケーションをパッケージ化する必要があります。パッケージ・ウィザードを使用すると、アプリケーション・ファイルおよび Application Deployment Descriptor を含んだ自己完結型のパッケージを作成できます。Mobile サーバー管理者は、このパッケージを使用して Windows CE Computing アプリケーションを Mobile サーバー・リポジトリにパブリッシュします。

パッケージ・ウィザードの使用方法の詳細は、第 4 章「パッケージ・ウィザードの使用」を参照してください。

制限事項

今回のリリースには、次のような制限があります。

- 現時点では、Java のストアド・プロシージャおよびトリガーや、Oracle Lite JAC インタフェースおよび JDBC インタフェースは、Windows CE ではサポートされていません。
- かわりの Windows CE ODBC データ・ソース名 (DSN) を手動で定義する必要があります。
- CREATEDB のような Oracle Lite ツールは、実行中にメッセージ・ボックスを表示し、出力ファイルを **ANSIOUT.TXT** というファイルに格納します。「スタート」メニューの「ファイル名を指定して実行」コマンドを使用して、引数を指定できます。ツールを終了すると、メッセージ・ボックスは自動的に消えます。「OK」ボタンをクリックすると、処理が即時に終了します。
- プログラム **CESETUP.EXE** は、デバイス上の ¥orace サブディレクトリにあるファイル **POLITE.TXT** および **ODBC.TXT** を削除して置き換えます。これらのファイルをあらかじめインストールして上書きされないようにする場合、**CESETUP.EXE** を実行する前に **CESETUP.INI** ファイルの DELETE セクションにあるこれらのファイルへの参照を削除します。インストール・プロセスでは、**POLITE.ODB** ファイルは削除されません。インストール中にこのファイルを削除する必要がある場合は、**CESETUP.INI** ファイルの DELETE セクションに次の行を追加します。

```
¥orace¥polite.odb
```


この章では、**Mobile** サーバーを使用したレプリケーションについて説明します。内容は次のとおりです。

- 概要
- [Oracle](#) サーバーとデバイス間でのデータ型のマッピング
- [パブリッシュ・サブスクライブ・モデル](#)
- [パブリッシュ・サブスクライブ・メソッド固有の関数](#)
- [Mobile Sync API](#)
- デバイス上の **Mobile** クライアントの設定
- [Mobile](#) サーバーのシステム・カタログ・ビュー

概要

Mobile サーバーにより、携帯端末上の新規または既存のアプリケーションやデータを Oracle サーバーとレプリケート、同期および共有できます。デバイス上のデータは、Oracle8i データベース上の Mobile サーバー・エージェントを介して Oracle8i データ・サーバーに直接マップできます。Mobile サーバーでは、モバイル・デバイスのデータ・サブセット化ポリシーを管理するパブリッシュ / サブスクライブ・モデルが使用されます。デバイス上のデータは、HTTP によるトランスポートを介して Oracle8i にマップされます。

MGP (Message Generator and Processor) は、クライアント・デバイスで実行中のアプリケーションからトランザクションをアップロードする Java バックグラウンド・プロセスです。MGP は、Oracle データベースにトランザクションを適用します。クライアント・デバイスがダウンロードする新規の更新内容 (データ) も生成します。MGP の管理の詳細は、『Oracle9i Lite パブリッシュおよびディプロイ・ガイド』を参照してください。

パブリケーションとサブスクリプション

Mobile サーバーでは、パブリケーションとサブスクリプションが使用されます。パブリケーションは、Oracle8i の表またはビューに対して定義された問合せで、オプションでパラメータ化されます。パブリケーションは、1 人以上のユーザーによってサブスクライブされます。Mobile サーバーは、デバイスのユーザーを追跡します。確立されたサブスクリプションを介して、Mobile サーバーは各クライアント・デバイスに対する新規データを準備します。データは、行単位で水平に、または列単位で垂直にパーティション化できます。データのうち必要なサブセットのみが各クライアントにダウンロードされます。

Mobile サーバーのパブリッシュ・サブスクライブ・モデルでは、各パブリケーションに複数のパブリケーション項目を含むことができます。パブリケーション項目は、通常デバイス上の表にマップされます。パブリケーション項目内の表の列は改名できます。各パブリケーション項目には、複数のパラメータを含むことができます。詳細は、「[パブリッシュ・サブスクライブ・モデル](#)」の項を参照してください。

クライアント・デバイス・データベースの DDL 操作

クライアント・デバイスの最初のレプリケート時に、Mobile サーバーは、クライアント・デバイスにデータベース・オブジェクトを自動的に作成します。デフォルトでは、表の主要索引がサーバーから自動的にレプリケートされます。パブリケーション項目を使用して、デバイス上で 2 次索引を作成できます。1 次索引が必要ない場合は、パブリケーション項目から明示的に削除する必要があります。API の詳細は、『Oracle9i Lite Consolidator Admin API リファレンス』参照してください。

データベース・サポート

サーバー側では、Mobile サーバーは Oracle8i に対して動作します。クライアント側では、Mobile Sync は、Oracle Lite によってサポートされる任意のプラットフォーム上の Oracle Lite データベースとのレプリケーションをサポートします。

ユーザー定義の PL/SQL パッケージのバインド

Mobile サーバーの同期プロセスは、多くの方法でカスタマイズできます。アプリケーション・ロジックは、PL/SQL パッケージをパブリケーション項目にバインドすることで、Mobile サーバーにアタッチできます。パッケージは、BeforeCompose、AfterCompose、BeforeApply および AfterApply の各メソッドを公開する必要があります。Mobile サーバーは、次の作業の前後にこれらのメソッドをコールします。

- クライアントの変更内容を Mobile Sync Client にかわってサーバーの表に適用する
- 指定されたパブリケーション項目に対する高速リフレッシュ変更を構成する

Mobile サーバーは、現在の Mobile Sync Client ユーザーをこれらのメソッドに渡します。

ユーザー定義の PL/SQL パッケージは、データのキャッシュや事前計算ができます。外部キー制約違反の問題も解決できます。詳細は、「[更新可能パブリケーション項目の外部キー制約](#)」の項を参照してください。これらのコールの使用の詳細は、「[構成と適用を使用したコールバックのカスタマイズ](#)」の項を参照してください。

ネーミングの柔軟性

クライアント、表、ビューおよび列の名前には、スペースを含む特殊文字がサポートされます。コール側は Mobile サーバーに対して、データベースに格納されているとおりに（多くの場合、大文字を使用する）正確にオブジェクト名を指定する必要があります。

ビューの高速リフレッシュおよび更新操作

Mobile サーバーでは、特定の条件を満たす複合複数表パブリケーションに対する高速リフレッシュ操作と更新操作をサポートしています。

更新可能な親表

ビューを更新可能にするには、親表が必要です。親表には、ビューの任意の実表が格納されています。実表のビューの列リストには主キーが含まれており、これはビューの行セットで一意です。ビューを更新可能にする場合は、ビューでパブリケーション項目を作成する前に、Mobile サーバーに適切なヒントとビューの親表を指定する必要があります。

親表のヒントと INSTEAD OF トリガーの使用

ビュー・ベースのパブリケーション項目を更新可能にするには、次の 2 つの方式を使用する必要があります。

- 親表のヒント
- INSTEAD OF トリガー

親表のヒントは、指定されたビューの親表を定義します。親表のヒントは、Consolidator Admin API の ParentHint 関数を使用して指定します。詳細は、『Oracle9i Lite Consolidator Admin API リファレンス』の「ParentHint」を参照してください。

INSTEAD OF トリガーは、INSTEAD OF INSERT、INSTEAD OF UPDATE または INSTEAD OF DELETE の各コマンドを実行するために使用します。さらに INSTEAD OF トリガーは、ビューの実表に対して実行される操作にこれらの DML コマンドをマップします。INSTEAD OF トリガーは、Oracle8i の機能です。INSTEAD OF トリガーの詳細は、Oracle8i のドキュメントを参照してください。

ビューの高速リフレッシュ

パブリケーション項目は、デフォルトでは高速リフレッシュ用に作成されます。高速リフレッシュでは、増分変更のみがレプリケートされます。高速リフレッシュの利点は、同期セッション間での変更が限られている場合に大量のデータを持つデータ・ストアをレプリケートするときのオーバーヘッドが軽減され速度が向上することです。

ビューが次の条件を満たす場合、Mobile サーバーはビューの高速リフレッシュを実行します。

- 各ビューの実表には主キーが必要です。
- すべての実表の主キーは、すべてビューの列リストに含まれている必要があります。
- 項目がビューで、項目の選択条件に複数の表が含まれている場合、選択条件定義に含まれているすべての表が主キーを持ち、対応するパブリケーション項目を持っている必要があります。

ビューでは、親表に対してのみ一意の主キーが必要です。その他の表の主キーは重複してもかまいません。実表の主キー列ごとに、ビューでその列をコールする方法に関するヒントを Mobile サーバーに提供する必要があります。これは、Consolidator Admin API の PrimaryKeyHint を使用して実現できます。詳細は、『Oracle9i Lite Consolidator Admin API リファレンス』を参照してください。

ビューの完全リフレッシュ

パブリケーション項目は、Consolidator Admin API の CompleteRefresh コールを使用して、完全リフレッシュ用に作成できます。このモードを指定した場合、クライアント・データは同期のたびにサーバーの現在のデータで完全にリフレッシュされます。管理者は、API コールを介して、パブリケーション全体に完全リフレッシュを強制できます。完全リフレッシュ関数は、指定したクライアントに対するパブリケーションの完全リフレッシュを強制的

に実行します。詳細は、『Oracle9i Lite Consolidator Admin API リファレンス』を参照してください。

更新可能パブリケーション項目の外部キー制約

Oracle8i とクライアント・デバイスの間で更新可能モードで表をレプリケートするとき、表に参照整合性制約があると、外部キー制約違反が起きる場合があります。外部キー制約違反が発生した場合、サーバーはクライアント・トランザクションを拒否します。

外部キー制約違反の例

たとえば、2つの表 EMP と DEPT に参照整合性制約があるとします。DEPT 表の DeptNum (部門番号) 属性は、EMP 表の外部キーです。EMP 表の各従業員の DeptNum 値は、DEPT 表の有効な DeptNum 値である必要があります。

Mobile サーバー・ユーザーが DEPT 表に新しい部門を追加し、次に EMP 表でこの部門に新しい従業員を追加します。トランザクションはまず DEPT を更新し、次に EMP 表を更新します。しかし、データベース・アプリケーションでは、これらの操作を実行したシーケンスを保存しません。

ユーザーが Mobile サーバーをレプリケートする際、Mobile サーバーは最初に EMP 表を更新します。この作業で、DeptNum に対して無効な外部キー値を持つ新規レコードを EMP に作成しようとします。Oracle8i により参照整合性違反が検出されます。Mobile サーバーはトランザクションをロールバックし、トランザクション・データを Mobile サーバーのエラー・キューに置きます。この場合、トランザクション内の操作が元と違うシーケンスで実行されたために、外部キー制約違反が発生しました。

BeforeApply および AfterApply での制約違反の回避

PL/SQL を使用すると、DEFERRABLE 制約を BeforeApply 関数および AfterApply 関数とともに使用することで、シーケンスどおりでない操作による外部キー制約違反を回避できます。DEFERRABLE 制約は、INITIALLY IMMEDIATE または INITIALLY DEFERRED のいずれかにできます。DEFERRABLE INITIALLY IMMEDIATE 外部キー制約の動作は、通常の即時制約と同じです。どちらの制約をアプリケーションに適用しても、機能に違いはありません。

Mobile サーバーは、サーバーにクライアント・トランザクションを適用する前に BeforeApply 関数をコールし、トランザクションを適用した後で AfterApply 関数をコールします。BeforeApply 関数を使用して、制約を DEFERRED に設定し、参照整合性検査を遅延できます。トランザクションが適用された後、AfterApply 関数をコールして制約を IMMEDIATE に設定します。この時点で、クライアント・トランザクションが参照整合性に違反している場合は、ロールバックされエラー・キューに移動されます。

DEFERRABLE 制約を使用して外部キー制約違反を回避するには、次の手順に従います。

1. 外部キー制約をすべて削除して DEFERRABLE 制約として作成し直します。
2. ユーザー定義の PL/SQL パッケージを、参照整合性制約を持つ表が含まれたパブリケーションにバインドします。
3. 次のように、BeforeApply 関数で DEFERRED 制約を、AfterApply 関数で IMMEDIATE 制約を定義します。

```
procedure BeforeApply(clientname varchar2) is
cur integer;
begin
  cur := dbms_sql.open_cursor;
  dbms_sql.parse(cur, 'SET CONSTRAINT SAMPLE3.address14_fk
                      DEFERRED', dbms_sql.native);
  dbms_sql.close_cursor(cur);
end;
procedure AfterApply(clientname varchar2) is
cur integer;
begin
  cur := dbms_sql.open_cursor;
  dbms_sql.parse(cur, 'SET CONSTRAINT SAMPLE3.address14_fk
                      IMMEDIATE', dbms_sql.native);
  dbms_sql.close_cursor(cur);
end;
```

表の比率を使用した制約違反の回避

表の比率は、パブリケーションとパブリケーション項目を関連付ける整数プロパティです。Mobile サーバーは、表の比率を使用して、マスター表にクライアント操作を適用する順序を次のように決定します。

1. 最初に、クライアントの INSERT 操作が、表の比率の低いものから高いものへと順に実行されます。
2. 次に、クライアントの DELETE 操作が、表の比率の高いものから低いものへと順に実行されます。
3. 最後に、クライアントの UPDATE 操作が、表の比率の低いものから高いものへと順に実行されます。

「外部キー制約違反の例」の項にリストされた例では、DEPT 表に EMP 表よりも低い比率を割り当てることで、制約違反エラーを解決できます。たとえば、次のようになります。

(DEPT weight=1, EMP weight=2)

レプリケーション・エラーと競合

Mobile サーバーでは、サーバーが行を削除するのと同時にクライアントが更新する場合、Oracle8i のアドバンスド・レプリケーションとの互換性エラーが発生します。NULL の使用違反や外部キー制約違反などのその他のエラーはすべてレプリケーション・エラーです。

Mobile サーバーでは、レプリケーション・エラーは自動的に解決されません。かわりに、Mobile サーバーは対応するトランザクションをロールバックし、トランザクション操作を Mobile サーバーのエラー・キューに移動します。Mobile サーバーのデータベース管理者は、後でこれらのトランザクション操作を変更して再実行したり、エラー・キューからパージできます。

Mobile サーバーのレプリケーション競合は、次の場合に発生します。

- クライアントとサーバーが同じ行を更新する場合。
- クライアントとサーバーが同じ主キー値を持つ行を作成する場合。
- サーバーが更新する行とクライアントが削除する行が同じ場合。

競合解決手法の詳細は、「[エラー・キューを使用した競合の解決](#)」の項を参照してください。

バージョンニング

Mobile サーバーでは、内部バージョンニングを使用してレプリケーションの競合を検出します。バージョン番号は、各サーバー・レコードにかぎらず、各クライアント・レコードに対しても管理されます。クライアントの変更内容がサーバーに適用される際、Mobile サーバーはバージョンの不一致を検出し、ウィニング・ルールに従って競合を解決します。

ウィニング・ルール

Mobile サーバーは、ウィニング・ルールを使用してレプリケーションの競合を自動的に解決します。次のウィニング・ルールが組み込まれています。

- クライアント優先
- サーバー優先

クライアント優先の場合、Mobile サーバーはクライアントの変更内容を自動的にサーバーに適用します。サーバー優先の場合、Mobile サーバーはクライアントに対する変更内容を自動的に構成します。

Mobile サーバーの競合解決方式は、ウィニング・ルールを「クライアント優先」に設定し、データベース表に BEFORE INSERT、BEFORE UPDATE および BEFORE DELETE の各トリガーをアタッチすることで、カスタマイズできます。トリガーにより、新旧の行の値を比較して指定されたとおりにクライアントの変更内容を解決します。

Oracle サーバーとデバイス間でのデータ型のマッピング

Mobile サーバーによって同期される Oracle8i と Oracle9i Lite の表では、互換性のあるデータ型を使用している必要があります。Oracle8i のデータ型は、Oracle9i Lite のデータ型と互換性があります。

Oracle Lite データ型

Oracle9i Lite ベースのスナップショットはすべて、レプリケーション時に Mobile Sync クライアントによって作成されます。Mobile サーバーでは、Oracle8i でのデータ精度に応じて自動的に Oracle9i Lite データ型が選択されます。次の表にデータ変換を示します。Oracle データ型は左側の列に、Oracle9i Lite データ型は最上部行に表示されています。「X」は無条件にサポートされ、「-」はサポートされないことを示します。1B、2B および 4B のデータ型は、OKAPI 専用のデータ型です。詳細は、『Oracle9i Lite (C および C++) オブジェクト・カーネル API リファレンス』を参照してください。

表 2-1 Oracle Lite データ型

	1B	2B	4B	FLOAT	DOUBLE	DATETIME	LONG-VARBINARY	VARCHAR
INTEGER	X	X	X	X	X	-	-	-
VARCHAR2	-	-	-	-	-	-	-	X
VARCHAR	-	-	-	-	-	-	-	X
CHAR	-	-	-	-	-	-	-	X
SMALLINT	X	X	X	X	X	-	-	-
FLOAT	X	X	X	X	X	-	-	-
DOUBLE PRECISION	X	X	X	X	X	-	-	-
NUMBER	X	X	X	X	X	-	-	-
DATE	-	-	-	-	-	X	-	-
LONG RAW	-	-	-	-	-	-	X	-
BLOB	-	-	-	-	-	-	X	-

パブリッシュ・サブスクライブ・モデル

Mobile サーバーではパブリッシュ・サブスクライブ・モデルを使用して、Oracle サーバーと携帯端末間のデータ配分を集中管理します。パブリッシュ・サブスクライブ・モデルは、Mobile Server Admin API および Consolidator Admin API を使用して、特定の専用機能を起動するようプログラムで実装できます。これらの API はどちらも Mobile サーバーの一部です。このモデルの内容は次のとおりです。

表 2-2 パブリッシュ・サブスクライブ・モデルの要素

項目	説明
パブリケーション	パブリケーションは、パブリケーション項目のグループです。
パブリケーション項目	パブリケーション項目は、ユーザーがアクセスできるデータ・サブセットを指定する SQL の Select 文です。パブリケーション項目は、通常クライアント・デバイス上のレプリカ表に対応します。
サブスクリプション	サブスクリプションは、ユーザーをパブリケーションに対応付けます。サブスクリプション・パラメータを含む場合もあります。
ユーザー	<p>ユーザーは、ユーザー名とパスワードで定義されます。Mobile サーバーは、クライアントのサブスクリプションに従ってデータを同期します。</p> <ul style="list-style-type: none">■ ユーザーは単一のユーザー名を使用して、複数のデバイスに格納されたデータを同期できます。ユーザーがデバイスを変更すると、Mobile サーバーにより新規デバイス上でそのユーザーの全サブスクリプションの完全リフレッシュが実行されます。■ ユーザーは、複数のユーザー名を使用して、単一のデバイス上のデータを同期できます。ユーザー名を変更しようとする、Mobile サーバーによりクライアントの更新内容はすべて無視され、クライアントの全サブスクリプションの完全リフレッシュが実行されます。
サブスクリプション・パラメータ	サブスクリプション・パラメータは、名前と文字列値を使用して、個々のパブリケーションに対する個々のクライアントのサブスクリプションを定義します。サブスクリプション・パラメータを使用すると、クライアントはデータのサブセット化を実行し、各クライアントに割り当てられる行数を制限できます。一般的なサブスクリプション・パラメータには、ユーザー名と市外局番を含めることができます。

Mobile サーバーのパブリッシュ / サブスクリブ・モデルを実装するには、次の手順を実行します。

- 1. Mobile サーバーに接続します。
- 2. ユーザーを作成します。
- 3. パブリケーションを作成します。
- 4. パブリケーション項目を作成し、ウィニング・ルールを設定します。
- 5. パブリケーションにパブリケーション項目を追加します。
- 6. 必要に応じて、パブリケーション項目索引を作成します。
- 7. シーケンスを作成します。
- 8. ユーザーをパブリケーションにサブスクリブします。
- 9. クライアントのシーケンスをパーティション化します。
- 10. パブリケーションに対してユーザーのサブスクリプション・パラメータを定義します。
- 11. サブスクリプションをインスタンス化します。

次のいずれかを使用して、Mobile サーバーのパブリッシュ / サブスクリブ・モデルを実装できます。

表 2-3 パブリッシュ / サブスクリブ・モデルを実装する方法

実装メソッド	定義
パッケージ・ウィザード	お薦めする方法です。詳細は、 第 4 章「パッケージ・ウィザードの使用」 を参照してください。
Pure Java メソッド	Consolidator Admin API および Mobile Server Admin API は、Pure Java メソッドを使用して、Java プログラムから実行します。

注意： Consolidator Admin API では、大 / 小文字を区別されます。
Mobile Server Admin API では、大 / 小文字の区別はありません。

Mobile サーバーへの接続

ここでは、Java メソッドを使用して Mobile サーバーに接続する方法を説明します。

Java メソッドの例

```
ResourceManager.openConnection("MOBILEADMIN", "MANAGER");
```

Mobile Server Admin API のユーザー関数

ここでは、Mobile Server Admin API を使用してユーザーの作成、パスワードの変更、ユーザーの削除を行う方法を説明します。

ユーザーの作成

CreateUser() 関数を使用して、Mobile サーバー・ユーザーを作成できます。構文は次のとおりです。

```
public static boolean createUser(String userName, String password, String fullName, String privilege);
```

次の例では、表にリストされたパラメータを使用してユーザー「MOBILE」を作成します。

表 2-4 ユーザー作成パラメータの例

パラメータ	値	定義
userName	"MOBILE"	モバイル・クライアントのユーザー名を指定します。
password	"MOBILE"	ユーザー名 MOBILE のパスワードを指定します。
fullName	"MOBILEUSER"	ユーザー MOBILE のフル・ネームを指定します。
privilege	"C"	このユーザーが Mobile サーバーに接続できることを指定します。

Java メソッドの例

```
ResourceManager.createUser("MOBILE", "MOBILE", "MOBILEUSER", "C");
```

パスワードの変更

Mobile サーバー・ユーザーのパスワードは、SetPassword() 関数を使用して変更できます。構文は次のとおりです。

```
public static void setPassword(String userName, String newpwd);
```

次の例では、ユーザー「MOBILE」のパスワードを変更します。

表 2-5 パスワード設定パラメータの例

パラメータ	値	定義
userName	"MOBILE"	モバイル・クライアントのユーザー名を指定します。
newpwd	"MOBILENEW"	モバイル・クライアントの新しいパスワードを指定します。

Java メソッドの例

```
ResourceManager.setPassword("MOBILE", "MOBILENEW");
```

ユーザーの削除

dropUser() 関数を使用して、既存の Mobile サーバー・ユーザーを削除できます。構文は次のとおりです。

```
public static void dropUser(String userName);
```

次の例では、ユーザー「MOBILE」を削除します。

表 2-6 ユーザー削除パラメータの例

パラメータ	値	定義
userName	"MOBILE"	モバイル・クライアントのユーザー名を指定します。

Java メソッドの例

```
ResourceManager.dropUser("MOBILE");
```

パブリケーションの作成

ここでは、パブリケーションを作成する方法を説明します。

パブリケーション項目の定義

パブリケーション項目名は、26 文字に制限され、すべてのパブリケーションで一意である必要があります。パブリケーション項目は、表とビューの両方に対して定義できます。オブジェクトをパブリッシュするとき、ユーザーはオブジェクトの常駐するパブリケーション項目を指定する必要があります。更新可能な複数表ビューをパブリッシュする場合、特定の制限が適用されます。

- ビューには、主キーの定義された親表が含まれている必要があります。
- ビューの DML 操作に対して INSTEAD OF トリガーを定義する必要があります。
- ビューの実表は、すべてパブリッシュする必要があります。

データのサブセット化

パブリケーション項目を作成するとき、ユーザーは最大 8K の文字制限を持つ、パラメータ化された Select 文を定義できます。サブスクリプション・パラメータはこの時点で指定でき、レプリケーション時に、各クライアントに対してパブリッシュされるデータを制限するために使用されます。文字列代入値は、サブスクライブ時にパラメータ値を置き換えるために使用されます。

Consolidator Admin API の使用方法

CreatePublication 関数を使用して、パブリケーションを作成できます。構文は次のとおりです。

```
public static void CreatePublication(String name, int client_storage_type,
    String client_name_template, String enforce_ri) throws ThrowableConsolidator
```

次の例では、表にリストしたパラメータを持つ「T_SAMPE1」という名のパブリケーションを作成します。

表 2-7 パブリケーション作成パラメータの例

パラメータ	値	定義
client_storage_type	Consolidator.OKPI_WINCE	プラットフォーム・タイプを定義する定数。詳細は、『Oracle9i Lite Consolidator Admin API リファレンス』を参照してください。
client_name_template	'%s'	デフォルト。
enforce_ri	NULL	このパラメータは常に NULL です。

Java メソッドの例

```
Consolidator.CreatePublication("T_SAMPLE1", Consolidator.OKPI_WINCE, "%s", null);
```

注意： クライアントの格納タイプとして Oracle Lite データベースを使用する場合、データベースには拡張子は付きません。

パブリケーション項目の作成

ここでは、パブリケーション項目を作成する方法を説明します。

Consolidator Admin API の使用方法

CreatePublicationItem 関数を使用してパブリケーション項目を作成できます。構文は次のとおりです。

```
public static void CreatePublicationItem(String name, String owner,
    String store, String refresh_mode, String select_stmt, String cbk_owner, String
    cbk_name) throws Throwable
```

次の例では、下の表にリストしたパラメータを持つ「P_SAMPLE1」という名のパブリケーション項目を作成します。

表 2-8 パブリケーション項目作成パラメータの例

パラメータ	値	定義
owner	SAMPLE1	SAMPLE1 がベース・オブジェクトの所有者であることを指定します。
store	ADDROLRL4P	ADDROLRL4P がベース・オブジェクト名であることを指定します。
refresh_mode	F または C	リフレッシュ・モードを高速または完全として定義します。詳細は、「 ビューの高速リフレッシュおよび更新操作 」の項を参照してください。
select_stmt	例を参照	sample1.addrolrl4p の指定した列からデータを選択します。
cbk_owner	NULL	コールバック・パッケージの所有者を NULL に指定します。詳細は、「 構成と適用を使用したコールバックのカスタマイズ 」の項を参照してください。
cbk_name	NULL	コールバック・パッケージの所有者名を NULL に指定します。詳細は、「 構成と適用を使用したコールバックのカスタマイズ 」の項を参照してください。

Java メソッドの例

```
Consolidator.CreatePublicationItem("P_SAMPLE1", "SAMPLE1", "ADDROLRL4P", "F" ,
    "SELECT \"LastName\",\"FirstName\", company, phone1, phone2, phone3, phone4,
    phone5, phone1id, phone2id, phone3id, phone4id, phone5id, displayphone,
    address, city, state, zipcode, country, title, custom1, custom2, custom3,
    custom4, note
FROM sample1.addrolrl4p" + " WHERE upper(company) > " + ":COMP", null, null);
```

Null Sync コールアウト

Mobile サーバーは、同期中に、クライアント・デバイスが Null Sync を試行中かどうかを示すコールアウトを作成します。Null Sync とは、クライアントにアップロード対象の変更がないことを言います。このコールアウトは、Mobile サーバーのリポジトリ内に PL/SQL パッケージを作成することで実装できます。パッケージでは、次の指定を行う必要があります。

```
create or replace package CUSTOMIZE as procedure
  NullSync(p_Client IN varchar2, p_NullSync as boolean);
end CUSTOMIZE;
```

パブリケーション項目名の取得

ここでは、パブリケーション項目名を取得する方法を説明します。

Mobile Server Admin API の使用方法

```
public static String getPublicationItemName(String Publication Item, int platform);
```

定数 int platform は、スナップショットの作成対象となったプラットフォームを識別します。定数引数は、「ResourceManager.」で、プラットフォームに応じて WTG、PALM、EPOC または WINCE を後に付けます。詳細は、『Oracle9i Lite Web-to-Go API リファレンス』を参照してください。

次の例では、EPOC アプリケーションに対して DEPT というパブリケーション項目が返されます。

表 2-9 パブリケーション項目名取得パラメータの例

パラメータ	値	定義
PublicationItem	DEPT	スナップショットまたは表を識別する文字列名です。
platform	ResourceManager.WTG	スナップショットの作成対象となったプラットフォーム・タイプを表す定数です。

Java メソッドの例

```
ResourceManager.getPublicationItemName("DEPT",ResourceManager.EPOC);
```

パブリケーション項目索引の作成

Mobile サーバーでは、クライアント・デバイス上の Oracle Lite データベースへの索引の自動配布をサポートしています。Mobile サーバーは、主キー索引をサーバー・データベースから自動的にレプリケートします。Consolidator Admin API では、クライアント・デバイスに一意、標準および主キーの各索引を明示的に配布するコールも同様に提供しています。

Consolidator Admin API の使用方法

CreatePublicationItemIndex 関数を使用して、パブリケーション項目索引を配布できます。構文は次のとおりです。

```
public static void CreatePublicationItemIndex(String name,
                                             String publication_item, String pmode,
                                             String columns) throws Throwable
```

この例では、パブリケーション項目 PSAMPLE1 に INDEX001 という名のパブリケーション項目索引を作成します。パブリケーション項目索引は、表に定義されたように ZIPCODE 列を含む標準索引です。

表 2-10 パブリケーション項目索引作成パラメータの例

パラメータ	値	定義
name	INDX001	作成するパブリケーション項目として INDX001 を定義します。
publication_item	P_SAMPLE1	索引のパブリケーション項目として P_SAMPLE1 を定義します。
pmode	I	索引モードを標準として定義します。
columns	ZIPCODE	索引に ZIPCODE 列を含めます。

Java メソッドの例

```
Consolidator.CreatePublicationItemIndex("INDX001", "P_SAMPLE1", "I", "ZIPCODE");
```

クライアント索引の定義

クライアント側の索引は、既存のパブリケーション項目に対して定義できます。3 種類の索引を指定できます。

- P – 主キー
- U – 一意
- I – 標準

注意： パブリケーション項目に「U」または「P」タイプの索引を定義した場合、サーバー上では重複キーの検査は行われません。パブリケーション項目のベース・オブジェクトに同一の制約がない場合、Mobile Sync は重複キー違反で失敗する可能性があります。詳細は、『Oracle9i Lite Consolidator Admin API リファレンス』を参照してください。

パブリケーションへのパブリケーション項目の追加

パブリケーション項目を作成した後、これをパブリケーションに対応付ける必要があります。定義を変更するには、パブリケーション項目を削除して新しい定義で作成しなおすか、要件に応じてスキーマの発展を使用します。詳細は、『Oracle9i Lite Consolidator Admin API リファレンス』の「DropPublicationItem」と「AlterPublicationItem」をそれぞれ参照してください。

Consolidator Admin API の使用方法

AddPublicationItem 関数を使用して、パブリケーション項目をパブリケーションに追加できます。構文は次のとおりです。

```
public static void AddPublicationItem(String publication, String item,
    String columns, String disabled_dml, String conflict_rule, String
    restricting-predicate, String weight) throws Throwable
```

次の例では、P_SAMPLE1 という名のパブリケーション項目をパブリケーション T_SAMPLE1 に追加します。追加されるパブリケーション項目のパラメータは次のとおりです。

表 2-11 パブリケーション項目追加パラメータの例

パラメータ	値	定義
publication	T_SAMPLE	新規項目を受け取るパブリケーションとして T_SAMPLE を定義します。
item	P_SAMPLE	追加するパブリケーション項目として P_SAMPLE を定義します。
columns	NULL	列の名前を変更しないことを指定します。
disabled_dml	NULL	DML を使用禁止するためのオプションを選択しないことを指定します。その他の値は、『Oracle9i Lite Consolidator Admin API リファレンス』を参照してください。

表 2-11 パブリケーション項目追加パラメータの例（続き）

パラメータ	値	定義
conflict_rule	S	競合解消においてサーバー優先を定義します。その他の値は、「競合ルールの定義」の項を参照してください。
restricting_predicate	NULL	高優先順位モードを示します。制限選択条件は、パブリケーションにパブリケーション項目を追加するときに、パブリケーション項目に割り当てることができます。クライアントが高優先順位モードで同期される場合、選択条件はデバイスにプッシュされるデータを制限するために使用されます。このパラメータは、NULL にできます。このパラメータは、上級者向けです。
weight	NULL	NULL、またはマスター表に対してクライアント操作を実行する優先順位を指定する整数を指定します。詳細は、「表の比率の使用」の項を参照してください。

Java メソッドの例

```
Consolidator.AddPublicationItem("T_SAMPLE1", "P_SAMPLE1", null, null, "S");
```

読取り専用パブリケーション項目

パブリケーション項目は、DML を使用禁止にすることで、読取り専用として定義できます。詳細は、『Oracle9i Lite Consolidator Admin API リファレンス』を参照してください。

競合ルールの定義

パブリケーションにパブリケーション項目を追加する場合、クライアント「C」とサーバー「S」のいずれかを優先してレプリケーションの競合を解決するウィニング・ルールを指定できます。Mobile サーバーのレプリケーション競合は、次のいずれかの状況で検出されます。

- クライアントとサーバーで同一行が更新された場合。
- クライアントとサーバーの両方で同じ主キーを持つ行を作成した場合。
- クライアントが削除した行をサーバーが更新した場合。
- クライアントが更新した行をサーバーが削除した場合、Oracle8i のアドバンスド・レプリケーションとの互換性のためにレプリケーション・エラーと見なされます。
- クライアントのデータが実表に直接適用されない遅延データ処理があるシステム（たとえば、3 層アーキテクチャ）の場合、クライアントが最初に行を挿入し、次に同じ行を更新するときに、サーバーが実表にまだ行を挿入していない状況が発生する可能性があります。この場合、C\$ALL_CONFIG の DEF_APPLY パラメータが TRUE に設定されている場合、UPDATE でなく INSERT 操作が実行されます。結果として生じる PK 競合を

解決するのは、アプリケーション開発者の責任です。ただし、DEF_APPLY が設定されていない場合、「NO DATA FOUND」例外が発生します（レプリケーション・エラーの処理は後述を参照してください）。

- NULL の使用違反や外部キー制約違反などのその他のエラーはすべてレプリケーション・エラーです。
- レプリケーション・エラーが自動的に解決されない場合、対応するトランザクションがロールバックされ、トランザクション操作は Mobile サーバーのエラー・キューに移動されます。Mobile サーバーのデータベース管理者は、後でこれらのトランザクション操作を変更して再実行したり、エラー・キューからパージできます。

詳細は、『Oracle9i Lite Consolidator Admin API リファレンス』を参照してください。

表の比率の使用

表の比率は、パブリケーションとパブリケーション項目を関連付ける整数プロパティです。Mobile サーバーは、表の比率を使用して、マスター表にクライアント操作を適用する順序を次のように決定します。

1. 最初に、クライアントの INSERT 操作が、表の比率の低いものから高いものへと順に実行されます。
2. 次に、クライアントの DELETE 操作が、表の比率の高いものから低いものへと順に実行されます。
3. 最後に、クライアントの UPDATE 操作が、表の比率の低いものから高いものへと順に実行されます。

シーケンスの作成

Mobile サーバーでは、クライアント・デバイスに対してシーケンスのパーティション化とレプリケーションをサポートしています。

Java API の使用方法

CreateSequence 関数を使用してシーケンスを作成できます。構文は次のとおりです。

```
public static void CreateSequence(String name) throws Throwable
```

次の例では、CUSTOM1 という名のシーケンスを作成します。

Java メソッドの例

```
Consolidator.CreateSequence("CUSTOM1");
```

パブリケーションへのユーザーのサブスクライブ

次のいずれかの方法を使用して、パブリケーションに対してユーザー・サブスクリプションを作成できます。

Consolidator Admin API の使用方法

CreateSubscription 関数を使用して、ユーザーをパブリケーションにサブスクライブできます。構文は次のとおりです。

```
public static void CreateSubscription(String publication, String clientid)
    throws Throwable
```

次の例では、下の表にリストしたパラメータを使用して、クライアント DAVIDL をパブリケーション T_SAMPLE1 にサブスクライブします。

表 2-12 サブスクリプション作成パラメータの例

パラメータ	値	定義
publication	T_SAMPLE1	パブリケーションを T_SAMPLE1 として定義します。
clientid	DAVIDL	クライアントを DAVIDL として定義します。

Java メソッドの例

```
Consolidator.CreateSubscription("T_SAMPLE1", "DAVIDL");
```

クライアント・デバイスに対するシーケンスのパーティション化

Mobile サーバー・シーケンスを使用して、クライアント・デバイス上で一意の主キーを生成できます。レプリケート・シーケンスを使用するには、最初にこれを作成し、Mobile Sync Client ユーザー用にパーティション化する必要があります。

Consolidator Admin API の使用方法

CreateSequencePartition 関数を使用して、シーケンス・パーティションを作成できます。構文は次のとおりです。

```
public static void CreateSequencePartition(String name, String clientid,
    long curr_val, long incr) throws Throwable
```


次の例では、表に指定したパラメータを使用して CUSTOM1 シーケンスをパーティション化します。

表 2-13 シーケンス・パーティション作成パラメータの例

パラメータ	値	定義
name	CUSTOM1	パーティション化するシーケンスとして CUSTOM1 を定義します。
clientid	DAVIDL	シーケンスを割り当てるクライアントとして DAVIDL を定義します。
curr_val	1000	シーケンスの初期値として 1000 を定義します。
incr	1	シーケンスの増分値として 1 を定義します。

Java メソッドの例

```
Consolidator.CreateSequencePartition("CUSTOM1", "DAVIDL", 1000, 1);
```

注意： シーケンスが一意であることを保証するには、一意の開始位置を使用し、サーバー上のクライアント数に等しい増分値を設定します。

パブリケーションに対するクライアント・サブスクリプション・パラメータの定義

一部のパブリケーションにはパラメータがあります。パブリケーションにパラメータがある場合、これらのパラメータをパブリケーションのサブスクリプション用に設定する必要があります。

Consolidator Admin API の使用方法

SetSubscriptionParameter 関数を使用して、サブスクリプションのインスタンス化パラメータを定義できます。構文は次のとおりです。

```
public static void SetSubscriptionParameter(String publication, String clientid,  
String param_name, String param_value) throws Throwable
```

次の例では、表にリストしたパラメータを使用して、クライアント DAVIDL をパブリケーション T_SAMPLE1 にサブスクライブします。

表 2-14 サブスクリプション・パラメータ設定パラメータの例

パラメータ	値	定義
publication	T_SAMPLE1	パブリケーション T_SAMPLE1 を定義します。
clientid	DAVIDL	クライアントを DAVIDL として定義します。
param_name	COMP	パラメータ名を COMP として定義します。
param_value	P	パラメータ値を P として定義します。

Java メソッドの例

```
Consolidator.SetSubscriptionParameter("T_SAMPLE1", "DAVIDL", "COMP", "P");
```

サブスクリプションのインスタンス化

サブスクリプションのパブリケーション・パラメータを設定し、サブスクリプションをインスタンス化してサブスクリプション処理を完了します。Mobile サーバーでサブスクリプションをインスタンス化する際、サブスクリプションの完全な内部表現が作成されます。

Consolidator Admin API の使用方法

InstantiateSubscription 関数を使用して、サブスクリプションをインスタンス化できます。構文は次のとおりです。

```
public static void InstantiateSubscription(String publication,
                                         String clientid) throws Throwable
```

次の例では、表に指定した値を使用して、クライアントのサブスクリプションをパブリケーションに対してインスタンス化します。

表 2-15 サブスクリプションのインスタンス化パラメータの例

パラメータ	値	定義
publication	T_SAMPLE1	パブリケーションを T_SAMPLE1 として定義します。
clientid	DAVIDL	クライアントを DAVIDL として定義します。

Java メソッドの例

```
Consolidator.InstantiateSubscription("T_SAMPLE1", "DAVIDL");
```

代替パブリケーション項目を使用したスキーマの発展

既存のパブリケーション項目に列を追加できます。これらの新しい列は、次に同期される時点で、全サブスクライブ・クライアントにプッシュされます。これは変更済み全パブリケーション項目の完全リフレッシュで達成されます。

- 管理者は、複数の列を追加できます。
- この機能は、すべてのクライアント形式に対してサポートされます。
- クライアントは、サーバーにスナップショット情報をアップロードしません。これは、クライアントがクライアント・データベースでスナップショットを直接変更できないことを意味します。たとえば、Windows CE 上で Mobile SQL を使用して表を変更することはできません。
- パブリケーション項目のアップグレードは、高優先順位の同期中は遅延されます。これは、ワイヤレスなどの低帯域幅ネットワークの場合に必要です。パブリケーション項目のアップグレードには常に、変更済みパブリケーション項目の完全リフレッシュが必要なためです。高優先順位フラグが設定されている間、高優先順位のクライアントは古いスナップショット形式を受信し続けます。
- サーバーは、パブリケーション項目のバージョンを最大 2 個サポートする必要があります。

パブリケーション項目の変更

これにより、既存のパブリケーション項目に列を追加できます。

Consolidator Admin API の使用方法

```
public static void AlterPublicationItem(String name, String select_stmt)
    throws Throwable
```

表 2-16 パブリケーション項目変更パラメータの例

パラメータ	値	説明
name	P_SAMPLE1	パブリケーション項目名を指定する文字列です。
select_stmt	select * from EMP	追加列の含まれた新規パブリケーション項目の Select 文です。

Java メソッドの例

```
Consolidator.AlterPublicationItem("P_SAMEPLE1", "select * from EMP");
```

パブリッシュ・サブスクライブ・メソッド固有の関数

次の機能には、ほとんどのアプリケーション設計で必要でない特殊関数が含まれています。

構成と適用を使用したコールバックのカスタマイズ

パブリケーション項目を作成する際、ユーザーは MGP バックグラウンド・プロセスの適用と構成段階でコールされるカスタマイズ可能パッケージを指定できます。これらのプロシージャにより、カスタマイズされたコードをプロセスに取り込むことができます。ユーザー・レベルおよびトランザクション・レベルでのカスタマイズを可能にするために、`clientname` と `tranid` が渡されます。

このプロシージャは、インキューのクライアント・データが適用される前にコールする必要があります。

```
procedure BeforeApply(clientname varchar2)
```

このプロシージャは、クライアントのデータがすべて適用された後でコールする必要があります。

```
procedure AfterApply(clientname varchar2)
```

このプロシージャは、`tranid` を持つクライアントのデータが適用される前にコールする必要があります。

```
procedure BeforeTranApply(tranid number)
```

このプロシージャは、`tranid` を持つクライアントのデータが適用された後にコールする必要があります。

```
procedure AfterTranApply(tranid number)
```

このプロシージャは、アウトキューが構成される前にコールする必要があります。

```
procedure BeforeCompose(clientname varchar2)
```

このプロシージャは、アウトキューが構成された後にコールする必要があります。

```
procedure AfterCompose(clientname varchar2)
```

カスタマイズ DML 操作の定義

パブリケーション項目を作成すると、そのパブリケーション項目の DML 操作のかわりにコールされるカスタマイズ PL/SQL プロシージャを指定できます。パブリケーションまたは `pub_item` あたり、`mobile_dml_procedure` は 1 つしか指定できません。たとえば、次の問合せを持つパブリケーション項目 A に対して、DML プロシージャを 1 つ持つとします。

```
select A,B,C from pub_item_a_table
```

A は主キーです。DML プロシージャのシグネチャは次のようになります。

```
any_schema.any_package.any_name (DML in CHAR(1), A in TYPE, B in TYPE, C in TYPE, A_
OLD in TYPE)
```

このプロシージャは、DML タイプとして「I」、「U」、「D」でコールされます。挿入および削除の場合、A_OLD は NULL になります。更新の場合、更新される行の主キーに設定されます。

仮想主キー

ベース・オブジェクトに主キーが定義されていないパブリケーション項目に対して、仮想主キーを指定できます。仮想主キーの作成および削除には、いくつかの方法があります。

仮想主キー列の作成

これは仮想主キー列を作成します。

Consolidator Admin API の使用方法

これは、主キーが仮想列である更新可能パブリケーション項目を作成するために使用します。

```
public static void CreateVirtualPKColumn(String owner, String store, String column)
throws Throwable
```

表 2-17 仮想主キー列作成パラメータ

パラメータ	値	説明
owner	SAMPLE_1	ベース・オブジェクトの所有者を指定する文字列です。
store	DEPT	ベース・オブジェクトを指定する文字列です。
column	DEPT_ID	主キー列を指定する文字列です。この文字列はカンマで区切ることができます。

Java メソッドの例

```
Consolidator.CreateVirtualPKColumn("SAMPLE1", "DEPT", "DEPT_ID");
```

仮想主キー列の削除

これにより、主キーを削除できます。

Consolidator Admin API の使用方法

```
public static void DropVirtualPKColumn(String owner, String store) throws Throwable
```

表 2-18 仮想主キー列の削除パラメータ

パラメータ	値	説明
owner	SAMPLE_1	ベース・オブジェクトの所有者を指定する文字列です。
store	DEPT	ベース・オブジェクトを指定する文字列です。

Java メソッドの例

```
Consolidator.DropVirtualPKColumn("SAMPLE1", "DEPT");
```

制限選択条件

制限選択条件は、パブリケーションにパブリケーション項目を追加するときに、パブリケーション項目に割り当てることができます。クライアントが高優先順位モードで同期される場合、選択条件はデバイスにプッシュされるデータを制限するために使用されます。このパラメータは、NULL にできます。このパラメータは、上級者向けです。

エラー・キューを使用した競合の解決

作成したパブリケーション項目ごとに、対応するエラー・キューが別個に作成されます。このキューの目的は、未解決の競合が原因で失敗するトランザクションを格納することです。管理者は、エラー・キュー・データまたはサーバーのエラー・キューを変更することで競合の解決を試みることができ、続いて Execute Transaction API コールを介してトランザクションの再適用を試みることができます。管理者は、Purge Transaction API コールを介してエラー・キューのページを試みることもできます。

トランザクションの実行

トランザクション実行関数は、Mobile サーバーのエラー・キュー内のトランザクションを再実行します。

Consolidator Admin API の使用方法

```
public static void ExecuteTransaction(String clientid, long tid)
    throws Throwable
```

表 2-19 トランザクション実行パラメータ

パラメータ	説明
clientid	Mobile Sync Client 名です。
tid	トランザクション ID です。

Java メソッドの例

```
Consolidator.ExecuteTransaction("DAVIDL", 100002);
```

トランザクションのパージ

トランザクションのパージ関数は、Mobile サーバーのエラー・キューからトランザクションをパージします。

Consolidator Admin API の使用方法

```
public static void PurgeTransaction(String clientid, long tid) throws Throwable
```

表 2-20 トランザクションのパージ・パラメータ

パラメータ	値	説明
clientid	DAVIDL	Mobile サーバー・ユーザー名です。
tid	100001	トランザクション ID です。

Java メソッドの例

```
Consolidator.PurgeTransaction("DAVIDL", 100001);
```

Mobile Sync API

Mobile Sync API を使用すると、アプリケーションはデータベースとの同期をクライアント・デバイスから開始および監視でき、Mobile サーバーから起動する必要はありません。デフォルトの転送方法は HTTP ですが、他の転送形式が使用できる場合は、それを指定できます。

Mobile Sync は、5 つのファンクション・コールと 1 つの制御構造で構成されます。

ocSessionInit

同期環境を初期化します。

構文

```
int ocSessionInit( ocEnv *env );
```

パラメータ

Env

戻される同期環境を保持する ocEnv 構造バッファへのポインタ。

コメント

このコールは、ocEnv 構造体を初期化し、最後の ocSaveUserInfo() コールで保存されたユーザー設定をリストアします。OcEnv 構造体は、パラメータとして渡し、コール元で初期化する必要があります。ocSessionInit() コールの後、コール側でユーザー設定情報を上書きする場合は、ocSaveUserInfo() をコールします。

ocSessionTerm

同期環境を解放し、クリーン・アップします。

構文

```
int ocSessionTerm( ocEnv *env );
```

パラメータ

Env

ocSessionInit によって戻された環境構造体へのポインタ。

コメント

ocSessionInit() コールによって作成された構造体とメモリーをすべて消去します。ocSessionInit() と ocSessionTerm() は常に一对でコールする必要があります。

ocSaveUserInfo

ユーザー設定を保存します。

構文

```
int ocSaveUserInfo( ocEnv *env );
```

パラメータ

Env

同期環境へのポインタ。

コメント

この関数は、ユーザー設定をクライアント側のファイルまたはデータベースに保存するか上書きします。環境構造体で指定された次の情報が保存されます。

- Username
- Password
- SavePassword

- NewPassword
- Priority
- Secure
- PushOnly
- SyncApps
- SyncNewPublication

これらのフィールドの使用方法は、「[Mobile Sync API のデータ構造](#)」の項を参照してください。

ocDoSynchronize

同期プロセスを開始します。

構文

```
int ocDoSynchronize( ocEnv *env );
```

パラメータ

Env

同期環境へのポインタ。

コメント

この関数は、同期サイクルを開始します。syncDirection が 0（デフォルト）の場合、ラウンドトリップ同期がアクティブにされます。syncDirection が 1 の場合、アップロード（送信）のみが実行されます。syncDirection が 2 の場合、ダウンロード（受信）のみが実行されます。クライアント・デバイスがサーバーからのデータのダウンロードを必要としない場合、アップロードのみの同期を実行すると有益です。

ocSetTableSyncFlag

選択同期用の表フラグを更新します。

構文

```
ocSetTableSyncFlag(ocEnv *env, const char* publication_name,  
const char* table_name, short sync_flag)
```

パラメータ

Env

同期環境へのポインタ。

publication_name

Consolidator Admin API に含まれている `CreatePublication()` で使用されます。詳細は、『Oracle9i Lite Consolidator Admin API リファレンス』を参照してください。

`publication_name` が NULL の場合、`sync_flag` はパブリケーションのすべての項目に対して適用されます。

table_name

この文字列は、`CreatePublication()` の `client_name_template` と同じです。OKAPI の場合、これは `database_name + '.' + store` になります。ここで `store` は、`CreatePublicationItem()` の 3 番目のパラメータです。`CreatePublication()` および `CreatePublicationItem()` の詳細は、『Oracle9i Lite Consolidator Admin API リファレンス』を参照してください。

sync_flag

1 の場合、同期します。0 の場合、同期しません。`sync_flag` は持続的に格納されているわけではありません。`ocDoSynchronize()` の前に、毎回 `ocSetTableSyncFlag()` をコールする必要があります。

コメント

この関数により、クライアント・アプリケーションは、特定の表の同期方法を選択できます。

個々の表または個々のパブリケーションに対して `sync_flag` を設定します。`sync_flag=0` の場合、表は同期されません。

Mobile Sync API のデータ構造

ocEnv

`ocEnv` は、内部的なメモリー・バッファと状態情報を保持するためにすべての Mobile Sync 関数によって使用されるデータ構造体です。この構造体を使用する前に、アプリケーションはこれを `ocSessionInit` に渡して環境を初期化する必要があります。

環境構造体には、プログラムで Mobile Sync 関数の動作方法を変更するために、コール側が更新できるフィールドも含まれています。

```
typedef struct ocEnv_s {  
    // User infos  
    char username[MAX_USERNAME];    // Mobile Sync Client id  
    char password[MAX_USERNAME];    // Mobile Sync Client password for  
    // authentication during sync  
    char newPassword[MAX_USERNAME]; // resetting Mobile Sync Client password  
    // on server side if this field is not blank
```

```

short savePassword;      // if set to 1, save 'password'
char appRoot[MAX_PATHNAME]; // dir path on client device for deploying files
short priority;          // High priority table only or not
short secure;            // if set to 1, data encrypted over the wire
enum {
    OC_SENDRECEIVE = 0,    // full step of synchronize
    OC_SENDONLY,          // send phase only
    OC_RECEIVEONLY,       // receive phase only

    // For Palm Only
    OC_SENDTOFILE,        // send into local file | pdb
    OC_RECEIVEFROMFILE    // receive from local file | pdb
}syncDirection;         // synchronize direction

enum {
    OC_BUILDIN_HTTP = 0,   // Use build-in Http transport method
    OC_USER_METHOD         // Use user defined transport method
}trType;                 // type of transport

ocErrorrexError;         // extra error code

ocTransportEnvtransportEnv; // transport control information

// GUI related function entry
progressProcfnProgress;   // callback to track progress; this is optional

// Values used for Progress Bar. If 0, progress bar won't show.
longtotalSendDataLen;      // set by Mobile Sync API informing transport total
number of
// bytes to send; set before the first fnSend() is called
long totalReceiveDataLen;  // to be set by transport informing Mobile Sync API
// total number of bytes to receive;
// should be set at first fnReceive() call.
void* userContext;         // user defined context
void* ocContext;           // internal use only
short logged;              // internal use only
long bufferSize;          // send/receive buffer size, default is 0
short pushOnly;           // Push only flag
short syncApps;           // Application deployment flag
} ocEnv;

```

ocTransportEnv

この構造体は、組み込み転送関数を上書きするために使用します。構造体に関数のリストを指定することにより、アプリケーションは同期エンジンによって使用されるトランスポート層に対して独自の実装を定義できます。

```
typedef struct ocTransportEnv_s {
void* ocTrInfo;          // transport internal context
                        // for built-in Http, mapped to ocTrHttp
connectProc fnConnect;   // plug-in callback to establish a connection from
                        // device to server
disconnectProc fnDisconnect; // plug-in callback to dismantle connection from
                        // device to server
sendProc fnSend;         // plug-in callback to send data
receiveProc fnReceive;   // plug-in callback to receive data
}ocTransportEnv;
```

デバイス上の Mobile クライアントの設定

Oracle8i と Windows 間のトランスポートを構成した後、Mobile Sync をデバイス上にインストールし構成する必要があります。Windows CE 上で Mobile クライアントを設定するとき、次の 2 つの手順を実行します。

- Windows 開発システムから CESETUP.EXE を実行して、Mobile Client for Windows CE のライブラリをインストールします。
- Windows CE デバイスと Mobile サーバーの間の HTTP トランスポートを構成します。

ディレクトリ構造

Mobile Development Kit for Windows CE は、Oracle_Home¥Mobile¥SDK¥wince ディレクトリにインストールされます。このディレクトリ内には、samples、wce200、wce211、wce212 および wce300 というラベル付きフォルダが 5 つあります。

次の表に、このディレクトリの内容とその説明のリストを示します。

ディレクトリ	説明
samples	次のフォルダがあります。 <ul style="list-style-type: none">■ ADOCE: ActiveX Data Objects for Windows CE のサンプル・コード。■ JDBC: JDBC のサンプル・コード。■ ODBC: ODBC のサンプル・コード。■ OKDEMO: OKdemo のリソース・ファイル。■ Consolidator: Consolidator Admin API のサンプル・コード。

ディレクトリ	説明
wce200	CESETUP、および Windows CE 2.0 に必要なその他の .dll ファイルおよび .exe ファイルがあります。このフォルダには、他にも、このバージョンで、MIPS および SH3 の各プロセッサを実行するために必要なライブラリ・ファイルおよび実行可能ファイルのフォルダがあります。
wce211	CESETUP、および Windows CE 2.1.1 に必要なその他の .dll ファイルおよび .exe ファイルがあります。このフォルダには、他にも、このバージョンで ARM、MIPS、SH3 および SH4 の各プロセッサを実行するために必要なライブラリ・ファイルおよび実行可能ファイルのフォルダがあります。
wce212	CESETUP、および Windows CE 2.1.2 に必要なその他の .dll ファイルおよび .exe ファイルがあります。このフォルダには、他にも、このバージョンで x86 プロセッサを実行するために必要なライブラリ・ファイルおよび実行可能ファイルのフォルダがあります。
wce300	CESETUP、および Windows CE 3.0 に必要なその他の .dll ファイルおよび .exe ファイルがあります。このフォルダには、他にも、このバージョンで ARM、MIPS、SH3 および SH4 の各プロセッサを実行するために必要なライブラリ・ファイルおよび実行可能ファイルのフォルダがあります。

CESETUP.EXE ファイルの実行

使用するプロセッサとオペレーティング・システムの組合せに合わせて

`Oracle_Home¥Mobile¥SDK¥wince` ディレクトリにある適切な **CESETUP.EXE** ファイルを実行します。このプログラムは、Windows CE デバイスに必要なファイルをコピーし、Windows CE のデータ・ソース名 (DSN) を構成し、デバイスの環境を設定します。変更しないかぎり、デバイスの `¥oracle` ディレクトリにあるデフォルトのデータベース・ファイルである **POLITE.ODB** にインストールされます。セットアップ初期化ファイルの **CESETUP.INI** は、デバイスにロードされるファイルをリスト表示します。

トランスポートの構成

HTTP を使用する Windows CE デバイスのレプリケートには、TCP/IP 通信プロトコルが必要です。ここでは、Windows NT RAS を使用して Windows CE デバイスの TCP/IP 通信を可能にする方法を示します。

NT RAS の構成

Windows NT RAS を使用して Windows CE デバイスと Windows NT サーバーの間の TCP/IP 通信を可能にするには、次の手順を実行します。

1. Windows の「コントロール パネル」で、「ネットワーク」のアイコンをダブルクリックします。
2. 「ネットワーク」ウィンドウの「サービス」タブで、「追加」ボタンをクリックします。

3. 「リモート アクセス サービス」を選択して「OK」ボタンをクリックします。
4. Windows NT の CD-ROM をディスク・ドライブに挿入して、セットアップ・プログラムが既存の Windows NT RAS ファイルを検索する場所を指定します。
5. 「続行」ボタンをクリックします。セットアップ・プログラムが、Windows NT RAS のファイルを適切なディレクトリにコピーします。セットアップ・プログラムがモデムを検出できない場合は、モデムの追加を要求するプロンプトを表示します。
6. 「はい」ボタンをクリックして、モデムをインストールするために必要な情報を入力します。「RAS デバイスの追加」ウィンドウが表示されます。
7. ドロップダウン・リストから「COM1- ダイヤルアップ ネットワーク シリアル ケーブル」を選択し、「OK」ボタンをクリックします。「リモート アクセス セットアップ」ウィンドウが表示されます。
8. 「ポート」の下で、COM1 を選択して「構成」ボタンをクリックします。「ポート使用の構成」ウィンドウが表示されます。
9. 「着信のみ」ラジオ・ボタンを選択して「OK」ボタンをクリックします。
10. 「リモート アクセス セットアップ」ウィンドウで「ネットワーク」ボタンをクリックします。「ネットワークの構成」ウィンドウが表示されます。
11. 「TCP/IP」を選択して「OK」ボタンをクリックします。「RAS サーバー TCP/IP の構成」ウィンドウが表示されます。
12. 「ネットワーク全体」および「静的アドレス プールを使う」を選択します。
13. 複数デバイスの場合、TCP/IP アドレスの範囲を指定できます。範囲を開始する TCP/IP アドレスを「開始アドレス」フィールドに、終了する TCP/IP アドレスを「終了アドレス」フィールドに入力します。範囲は、クライアント数 +1 にしてください。たとえば、50 台のデバイスがある場合、次の範囲を入力します。

開始アドレス : 10.1.0.1

終了アドレス : 10.1.0.51

重要： 同一ネットワーク上の他のコンピュータによって、選択した範囲の TCP/IP アドレスが使用されていないことを確認する必要があります。

14. 「OK」ボタンをクリックします。
15. 「ネットワークの構成」ウィンドウで、「クリア テキストを含む任意の認証を許可する」を選択して「OK」ボタンをクリックします。「続行」ボタンをクリックします。「セットアップの内部メッセージ」ウィンドウが表示されます。「OK」ボタンをクリックします。

16. 「ネットワーク」ウィンドウで、「閉じる」ボタンをクリックします。「ネットワーク設定の変更」ウィンドウが表示されます。「はい」ボタンをクリックします。
17. システムを再起動してから、Windows の「コントロールパネル」で「サービス」アイコンをクリックします。「サービス」ウィンドウが表示されます。
18. 「Remote Access Server」を選択して「開始」ボタンをクリックします。
19. 「自動」を選択して「OK」ボタンをクリックします。「閉じる」ボタンをクリックします。
20. Windows の「スタート」メニューで「プログラム」→「管理ツール」→「ユーザー マネージャ」を選択します。「ユーザー マネージャ」ウィンドウが表示されます。

デバイスの RAS User アカウントの作成

1. 「ユーザー」メニューで「新しいユーザー」を選択します。「新しいユーザー」ウィンドウが表示されます。
2. 必要なフィールドに、ユーザー名、パスワードおよびパスワードの確認を入力します。
3. 「パスワードを無期限にする」を選択します。
4. 「ダイヤルイン」ボタンをクリックします。「ダイヤルイン情報」ダイアログが表示されます。
5. 「ユーザーにダイヤルインの許可を与える」を選択します。「OK」ボタンをクリックします。
6. 「OK」ボタンをクリックすると、「新しいユーザー」ダイアログが終了します。
7. 「ユーザー マネージャ」画面を終了します。
8. Windows の「スタート」メニューで「プログラム」→「管理ツール」→「リモート アクセス管理」を選択します。「リモート アクセス管理」ウィンドウが表示されます。デバイスにサンプル・アプリケーションをインストールします。
9. 新規 RAS ユーザーにリモート・アクセス権限が付与されたかどうかを確認して、「OK」ボタンをクリックします。

Windows デスクトップの RAS 設定

RAS を構成するときには、次のことを確認する必要があります。

- モデムの設定が「ダイヤルアウトと着信」になっていること。
- Windows CE デバイスがドメインの任意のコンピュータに接続できるオプションを選択していること。
- Windows CE デバイスの静的 IP アドレスが、そのデバイスに接続しているデスクトップと同じサブネット内にあること。同じサブネット内ない場合は HTTP 接続を作成できないため、レプリケーションが失敗します。

- Windows CE デバイスの終了 IP アドレスがネットワークで使用されていないこと。
- ドメイン・ログオン・サーバーが Windows CE デバイスに対して使用可能であること。
- Mobile Sync Client アプリケーションを実行する前に、Windows CE デバイスの IP アドレスが正しく設定されていること。

Windows デスクトップの設定

Windows デスクトップの設定を構成するときには、次のことを確認する必要があります。

- 「コントロール パネル」のモデムおよびポートの構成で、ボー・レートが正しく設定されていること。また、Windows CE デバイスのボー・レートがデスクトップのボー・レートと一致していること。
- Windows CE サービスをインストールした後で、最新の Windows NT サービス・パックを再インストールしていること。

デバイスへのサンプル・アプリケーションのインストール

Mobile Development Kit には、「Sample Orders」というサンプル・アプリケーションがあります。CEServices を使用してインストールできます。CEServices を実行して、Windows CE デバイスの %orace フォルダに、使用しているプロセッサに対応する Oracle_Home¥Mobile¥SDK¥wince¥samples フォルダから **okdemo** ファイルを選択します。

同期のテスト

Oracle_Home¥Moble¥SDK¥wince¥ 中の Windows CE のバージョンと使用しているデバイスのプロセッサがあるフォルダをオープンして Mobile クライアント・アプリケーションを開始し、そのフォルダから **mSync.exe** を開始します。Windows CE のバージョンやデバイスのプロセッサを判断する方法の詳細は、『Oracle9i Lite インストレーションおよび構成ガイド』を参照してください。

Mobile Sync では、同期のために次のパラメータが必要です。

表 2-21 Mobile Sync Client のパラメータ

パラメータ	説明
ユーザー名	Mobile クライアント・ユーザー名です。このフィールドは大 / 小文字を区別しません。
パスワード	Mobile クライアント・パスワードです。このフィールドは大 / 小文字を区別します。
変更	このボックスは空白のままにします。

表 2-21 Mobile Sync Client のパラメータ（続き）

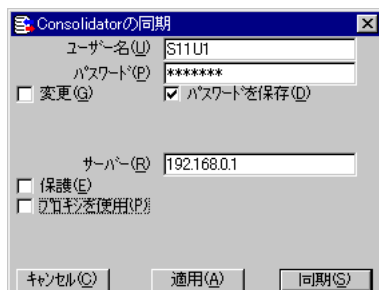
パラメータ	説明
パスワードを保存	パスワードを保存するには、このチェック・ボックスを選択します。
サーバー	Mobile サーバーの IP アドレスです。
プロシキを使用	必要であれば、選択します。

重要： レプリケーションのデモを行うために、この章では **Sample Orders** デモが使用されています。このサンプル・デモは、別個にインストールする必要があります。**Sample Orders** デモ・アプリケーションのインストールに関する情報は、『**Oracle9i Lite** インストレーションおよび構成ガイド』を参照してください。

SAMPLE11 サンプル・アプリケーションを利用するには、次のユーザー名とパスワードを使用します。ユーザー作成アプリケーションおよびユーザー・スナップショットには、別個のユーザー名とパスワードを指定する必要があります。

- ユーザー名 = S11U1
 - パスワード = MANAGER
1. Mobile Sync Client を起動するには、「**mSYNC**」アイコンをクリックします。「Oracle Mobile Sync」画面が表示されます。

図 2-1 「Mobile Sync」画面



2. 必要なフィールドに情報を入力します。「パスワードを保存」チェック・ボックスを忘れずに選択してください。「サーバー」フィールドには、画面名または IP アドレスを含めることができます。
3. 「適用」ボタンを選択します。
4. 「同期」ボタンを選択します。

進行状況バーが表示され、各同期作業の進捗状況（構成中 ...、送信中 ...、受信 ...、処理中 ...）が表示されます。進行状況バーには、各作業の完了に要する時間も表示されます。同期が正常に実行されると、同期成功画面が表示されます。同期に失敗すると、同期が失敗したというメッセージが表示されます。同期に失敗した原因を判断するには、サーバー管理者は Mobile サーバー・ログファイルで追跡情報を表示できます。

Mobile サーバーのシステム・カタログ・ビュー

Consolidator には、Consolidator のコンポーネントを検索するためのシステム・カタログ・ビューが含まれています。次の項目を検索するには、Consolidator のシステム・カタログ・ビューを使用します。

- Mobile サーバー・ユーザー
- パブリケーション
- サブスクリプション
- シーケンス
- シーケンス・パーティション
- スタンドアロン・パブリケーション項目
- パブリケーションに追加されたパブリケーション項目
- パブリケーション項目索引
- サブスクリプション・パラメータ

詳細は、[付録 A 「Mobile サーバーのシステム・カタログ・ビュー」](#) を参照してください。

ActiveX Data Objects for Windows CE

この章では、Oracle Lite データベースの ActiveX Data Objects for Windows CE について説明します。内容は次のとおりです。

- 概要
- [Active Connection](#) オブジェクトのメソッド
- [Recordset](#) オブジェクトのメソッド
- [Recordset](#) オブジェクトのプロパティ
- [Field](#) オブジェクトのプロパティ
- [Recordset](#) オブジェクトのコレクション
- [SQL](#) およびデータベースの参照
- エラー・メッセージ

概要

ActiveX Data Objects はプログラミング・インタフェースです。これを使用すると、Visual Basic アプリケーションで Oracle Lite データベースの機能にアクセスできます。これは COM インタフェース標準を基にしたオブジェクト指向プログラミング・インタフェースで、基礎となるデータベース・エンジンにアクセスする機能を実現します。ActiveX Data Objects for Windows CE には、Windows CE 環境で実行できるように設計された ActiveX Data Objects 機能のサブセットが含まれます。

ActiveX Data Objects for Windows CE インタフェースをサポートするために、Oracle Lite データベースはまったく同じインタフェースを実現するモジュールを実装しました。これによって、Visual Basic プログラマは、Oracle Lite データベースに対して同じ Microsoft 社の ActiveX Data Objects for Windows CE インタフェースを使用してアプリケーションを開発できます。

Oracle Lite データベースの ActiveX Data Objects for Windows CE は、Microsoft Windows CE ActiveX Data Objects v2.0 SDK のガイドラインに準拠しています。このガイドラインは、ActiveX Data Objects 1.0 のデスクトップ・バージョンに若干の修正を加えたものです。

ほとんどすべてのプログラミング言語で、Oracle Lite データベースの ActiveX Data Objects for Windows CE にアクセスできますが、Oracle Lite データベースの実装は Visual Basic 環境にあわせて調整されています。

Oracle Lite データベースの ActiveX Data Objects for Windows CE は、Oracle Lite データベースの ODBC インタフェースの上位に位置し、アプリケーション・プログラムに対する COM インタフェースを実現します。

構成

ActiveX Data Objects for Windows CE をインストールするには、次の手順を実行します。

1. ディレクトリ `Oracle_Home¥Mobile¥SDK¥wince` に移動して、CE デバイスに Oracle Lite データベースの主要部分をインストールするために `CESETUP` を最初に実行します。
2. CE デバイスのルートにある `ORACE` ディレクトリに、`OLADOCE.DLL` をコピーします。
3. 次のように入力して、`OLADOCE.DLL` を登録します。

```
regsvrce ¥ORACE¥OLADOCE.DLL
```

機能

ActiveX Data Objects for Windows CE インタフェースはオブジェクト指向インタフェースで、様々な COM インタフェース（クラス）へのアクセスを介してその機能を提供します。

次の表に、ActiveX Data Objects for Windows CE コントロールのオブジェクト、メソッドおよびプロパティのリストを示します。

表 3-1 ADOCE のオブジェクト型

インタフェース	メソッド	プロパティ
Active Connection	Connect	DSN
	Disconnect	
Recordset オブジェクト	AddNew	AbsolutePage***
	CancelUpdate	AbsolutePosition
	Clone***	ActiveConnection
	Close	BOF
	Delete	Bookmark***
	GetRows***	CacheSize***
	Move	CursorType
	MoveFirst	EditMode
	MoveLast	EOF
	MoveNext	LockType
	MovePrevious	PageCount***
	Open	PageSize***
	Supports	RecordCount
	Update	Source
Field オブジェクト	(メソッドなし)	ActualSize
		Attributes
		DefinedSize
		Name
		Type

表 3-1 ADOCE のオブジェクト型 (続き)

インタフェース	メソッド	プロパティ
		UnderlyingValue
		Value
Field コレクション	(メソッドなし)	Count

*** は未実装を示します。

ActiveX Data Objects for Windows CE の各オブジェクトには、一連のメソッドと一連のプロパティがあります。メソッドは、アプリケーションからコールできる関数です。プロパティはデータ・メンバーで、これを設定または取り出すことで機能の特定の部分を変更できます。

ActiveX Data Objects for Windows CE のオブジェクト

ActiveX Data Objects for Windows CE コントロールには、Active Connection、Recordset および Field という 3 つのオブジェクトがあります。Active Connection オブジェクトを作成し、ODBC.txt に含まれる DSN エントリに基づく接続をアクティブにするには、Create Object を使用します。新規の Recordset オブジェクトのみを作成する場合は、デフォルトの "polite" DSN がコールされ、そのエントリのデータベースに接続します。新規の Recordset オブジェクトを作成するには、CreateObject を使用します。Field オブジェクトは、直接作成できません。既存のレコードセットのコンテキストにのみ存在するためです。特定の Field オブジェクトを参照するには、Set を使用します。たとえば、次のようになります。

```
Dim rstCustomers, fldName
Set rc = CreateObject("oladoce.activeconnection")
rc.connect("polite")
Set rstCustomers = CreateObject("oladoce.recordset")
rstCustomers.Open "customers"
Set fldName = rstCustomers.Fields("Name")
MsgBox fldName.Value
rstCustomers.Close
rs.Disconnect
Set fldName = Nothing
Set rstCustomers = Nothing
```

Active Connection オブジェクトのメソッド

次の表に、Active Connection オブジェクトがサポートするメソッドのリストを示します。

表 3-2 Active Connection オブジェクトのメソッド

メソッド	説明
Connect	接続をオープンします。
Disconnect	接続をクローズします。

Connect

Active Connection オブジェクトへの接続を作成します。

構文

`ActiveConnection.Connect DSNName`

表 3-3 Connect のパラメータ

パラメータ	説明
DSNName	単一の DSN を識別する文字列。

Disconnect

Active Connection オブジェクトの接続をクローズします。

構文

`ActiveConnection.Disconnect`

Recordset オブジェクトのメソッド

次の表に、Recordset オブジェクトがサポートするメソッドのリストを示します。

表 3-4 Recordset オブジェクトのメソッド

メソッド	説明
AddNew	レコードセットに新規の行を 1 行挿入します。
CancelUpdate	メモリーに保持されている変更を取り消します。
Clone***	レコードセットを複製します。
Close	レコードセットをクローズします。
Delete	レコードセットから 1 行を削除します。
GetRows***	レコードセットに格納されているデータを返します。
Move	レコードセットのアクティブ行にポインタを変更します。
MoveFirst	最初の行をアクティブにします。
MoveLast	最後の行をアクティブにします。
MoveNext	レコードセットのアクティブ行ポインタを次の行に移動します。
MovePrevious	レコードセットのアクティブ行ポインタを前の行に移動します。
Open	レコードセットを定義してオープンします。SQL コマンドを実行します。
Update	メモリーに保持されている変更をコミットし、実際の表を更新します。
Supports	レコードセットが特定の機能をサポートするかどうかを判断します。

ActiveX Data Objects for Windows CE データベースは単一ユーザーによるアクセスを想定しているので、バッチ更新モードはありません。ActiveX Data Objects for Windows CE では、次のメソッドをサポートしません。

- CancelBatch
- NextRecordset
- Requery
- Resync
- UpdateBatch

AddNew

更新可能な Recordset オブジェクトに新規レコードを作成します。

構文

```
recordset.AddNew Fields, Values
```

表 3-5 AddNew のパラメータ

パラメータ	説明
Fields	オプションです。新規レコードの単一のフィールド名、または複数のフィールド名やフィールドの位置を示す順序数を表す配列です。
Values	オプションです。新規レコードのフィールドの単一の値、または複数の値を表す配列です。

コメント

新規レコードを作成および初期化するには、AddNew メソッドを使用します。現在の Recordset オブジェクトにレコードを追加できるかどうかを検証するには、Supports メソッド (Const AdAddNew = &H01000400) を使用します。デフォルトの LockType (Const AdLockReadOnly = 1) でレコードセットをオープンしている場合は、レコードを追加できません。

AddNew メソッドをコールすると、新規レコードがカレント・レコードになります。Update メソッドをコールした後もカレント・レコードのままになります。新規レコードは Recordset の最後に追加されます。

カレント・レコードの編集時または新規レコードの追加時に AddNew をコールすると、ActiveX Data Objects for Windows CE は Update メソッドをコールしてすべての変更を保存した後で新規レコードを作成します。

Fields が配列の場合、Values も同じメンバー数の配列である必要があります。それ以外の場合はエラーになります。両方の配列で、フィールド名の順序はフィールド値の順序と一致する必要があります。

引数なしで AddNew メソッドをコールすると EditMode プロパティが 2 (AdEditAdd) に設定され、ActiveX Data Objects for Windows CE はすべてのフィールド値の変更をローカルにキャッシュします。Update メソッドをコールすると新規レコードがデータベースに転送され、EditMode プロパティが 0 (AdEditNone) に設定されます。Fields 引数および Values 引数を渡すと、ActiveX Data Objects for Windows CE は即時に新規レコードをデータベースに転送します。Update のコールは必要ありません。EditMode プロパティ値は 0 (AdEditNone) から変更されません。

ActiveX Data Objects for Windows CE の更新は、常に即時モードです。バッチ・モードはサポートされていません。

例

```
Dim rs, i, L1
Set rs = CreateObject("oladoce.recordset")
rs.open "create table newtable (f1 varchar) "
rs.open "newtable", "", 1, 3
rs.addnew "f1", "a"
rs.addnew
rs.fields("f1") = "b"
rs.Update
rs.Close
rs.open "newtable"
Set rs = Nothing
```

CancelUpdate

Update メソッドをコールする前に、カレント・レコードまたは新規レコードに対して加えられたすべての変更を取り消します。

構文

```
recordset.CancelUpdate
```

コメント

カレント・レコードに対するすべての変更を取り消すか、新規に追加したレコードを廃棄するには、CancelUpdate メソッドを使用します。Update メソッドをコールした後では、カレント・レコードまたは新規レコードに対する変更を取り消せません。

新規レコードの追加中に CancelUpdate メソッドをコールした場合は、AddNew をコールする前にカレントだったレコードが再びカレント・レコードになります。EditMode プロパティは 0 にリセットされます。

カレント・レコードの変更または新規レコードの追加をしていない場合、CancelUpdate メソッドはエラーになります。変更があった場合、レコードセットの EditMode プロパティは 0 以外の値になります。

例

```
Dim rs
Set rs = CreateObject("oladoce.recordset")
rs.Open "Table1", "", 1, 3
rs.AddNew 'Changes the EditMode to 2
If rs.EditMode <> 0 Then rs.CancelUpdate
MsgBox rs.EditMode
rs.Close
Set rs = Nothing
```

Clone

既存の Recordset オブジェクトから複製の Recordset オブジェクトを作成します。

構文

```
recordset.Clone
```

コメント

複製の Recordset オブジェクトを作成する場合、特に、指定した一連のレコードの中で複数のカレント・レコードを維持する必要がある場合に、Clone メソッドを使用します。元のオブジェクトと同じ定義で新規の Recordset オブジェクトを作成してオープンするより、Clone メソッドを使用するほうが効率的です。

新規に作成された複製のカレント・レコードは、最初のレコードに設定されます。

ある Recordset オブジェクトに加えた変更は、カーソル・タイプにかかわらず、すべての複製で参照できます。

元の Recordset をクローズしても複製はクローズされません。複製をクローズしても元の Recordset や他の複製はクローズされません。

複製できるのは、ブックマークをサポートする (Const AdBookmark=8192) Recordset オブジェクトのみです。ブックマークの値は交換可能です。つまり、ある Recordset オブジェクトで参照しているブックマークはどの複製でも同じレコードを参照します。

例

```
Dim rs1, rs2, f
Set rs1 = CreateObject("adoce.recordset")
rs1.open "MSysTables"
Set f = rs1.Fields(0)
Msgbox f.Value, ,f.Name
Set rs2 = rs1.Clone
Set f = rs2.Fields(0)
Msgbox f.Value, ,f.Name
```

Close

オープンしているオブジェクトと依存オブジェクト（ある場合はすべて）をクローズします。

構文

```
recordset.Close
```

コメント

Recordset オブジェクトをクローズしてそのオブジェクトに関連するすべてのシステム・リソースを解放するには、Close メソッドを使用します。ただし、オブジェクトをクローズしてもメモリーからは削除されません。プロパティの設定を変更して、後で再びオープンできます。メモリーからオブジェクトを完全に除去するには、Recordset オブジェクト変数を Nothing に設定します。

Recordset オブジェクトをクローズすると、この特定の Recordset オブジェクトを介して取得した関連データは解放されますが、その元になるデータベースには影響しません。後で Open メソッドをコールして、そのレコードセットを同じ属性または変更された属性で再オープンできます。基礎となるデータベースの構造を変更する前に、レコードセットをクローズする必要があります。

Recordset オブジェクトがクローズされている間は、現在行を要求するメソッドはすべてエラーになります。また、クローズされているレコードセットを再度クローズしようすると、エラーになります。

編集集中に Close メソッドをコールすると、エラーになります。最初に Update メソッドまたは CancelUpdate メソッドをコールします。

オープンされている Recordset オブジェクトの複製を Clone メソッドを使用して作成した場合、複製や元の Recordset をクローズしても他の複製には影響しません。

Delete

オープンしている Recordset オブジェクトのカレント・レコードを削除します。

構文

recordset.Delete *AffectRecords*

表 3-6 Delete のパラメータ

パラメータ	説明
AffectRecords	オプションです。AffectRecords の有効な値は、1 (AdAffectCurrent) のみです。他の値では、ランタイム・エラーになります。この機能は、デスクトップ・コンピュータの ActiveX Data Objects との互換性のために提供されています。

コメント

Delete メソッドを使用すると、データベースからカレント・レコードが削除されます。Recordset オブジェクトがレコードの削除を許可していない場合はエラーになります。削除されたレコードからフィールドの値を取り出そうすると、エラーになります。

削除されたレコードがアクセス不可になった後でも、別のレコードに移動するまでは削除されたレコードがカレント・レコードです。レコードの削除に失敗すると、ランタイム・エラーになります。

GetRows

Recordset の複数のレコードを配列に取り出します。

構文

```
array = recordset.GetRows(Rows, Start, Fields)
```

表 3-7 GetRows のパラメータ

パラメータ	説明
Array	返されたデータが格納される VARIANT 型変数。
Rows	オプションです。LONG 型の式で、取り出すレコードの数を示します。デフォルト値は AdGetRowsRest または -1 です。
Start	オプションです。STRING 型または VARIANT 型で、GetRows 操作の開始レコードに対するブックマークに評価されます。
Fields	オプションです。VARIANT 型で、単一のフィールド名または位置を示す順序数、あるいは複数のフィールド名やフィールドの位置を示す順序数の配列を表します。ADOCE はここで指定したフィールドのデータのみを返します。

コメント

GetRows メソッドは、Recordset から 2 次元配列にレコードをコピーするために使用します。最初の添字はフィールドを示し、2 番目の添字はレコード番号を示します。GetRows メソッドがデータを返すときに、Array 変数のディメンションが自動的に設定されます。

Rows 引数の値を指定しないと、GetRows メソッドは自動的に Recordset オブジェクトからすべてのレコードを取り出します。使用可能なレコード数を超えて要求すると、GetRows は使用可能な数のレコードのみを返します。

Recordset オブジェクトがブックマークをサポートしている場合、どのレコードからデータの取出しを開始するかを、レコードの Bookmark プロパティの値を GetRows メソッドに渡して指定できます。

GetRows コールから返されるフィールドを制限する必要がある場合、単一のフィールド名またはフィールド数、あるいはフィールド名またはフィールド数の配列を Fields 引数として渡せます。

GetRows をコールした後は、まだ読み取られていない次のレコードがカレント・レコードになりますが、それ以上レコードがない場合は EOF プロパティが TRUE に設定されます。

例

```
Dim rstEmployees, strMessage
Dim intRows, avarRecords
Dim intRecord, intField
Dim intRecCount, intFieldCount

Set rstEmployees = CreateObject("adoce.recordset")
rstEmployees.Open "mytable"
strMessage = "Enter number of rows to retrieve."
intRows = CInt(InputBox(strMessage))

If intRows <= 0 Then intRows = 1

' If GetRowsOK is successful, print the results,
' noting if the end of the file was reached.
If GetRowsOK(rstEmployees, intRows, avarRecords) Then
    If intRows > UBound(avarRecords, 2) + 1 Then
        MsgBox "Less than " & intRows & " rows in recordset."
    End If
    intRecCount = UBound(avarRecords, 2) + 1
    intFieldCount = UBound(avarRecords, 1) + 1
    MsgBox intRecCount & " records found."
    MsgBox intFieldCount & " fields found."
    For intRecord = 0 To intRecCount - 1
        strMessage = ""
        For intField = 0 To intFieldCount - 1
            strMessage = strMessage & _
                avarRecords(intField, intRecord) & vbCrLf
        Next
        MsgBox strMessage
    Next
Else
    MsgBox "GetRows failed!"
End If

rstEmployees.Close
Set rstEmployees = Nothing

Public Function GetRowsOK(rstTemp, intNumber, avarData)
    ' Store results of GetRows method in array.
    avarData = rstTemp.GetRows(intNumber)
    ' Return False only if fewer than the desired
    ' number of rows were returned, but not because the
    ' end of the Recordset was reached.
    If intNumber > UBound(avarData, 2) + 1 And Not rstTemp.EOF Then
        GetRowsOK = False
    Else
```

```
GetRowsOK = True
End If
End Function
```

Move

Recordset オブジェクトのカレント・レコードの位置を移動します。

構文

recordset.Move NumRecords, Start

パラメータ

表 3-8 Move のパラメータ

パラメータ	説明
NumRecords	移動するカレント・レコードの位置をレコード数で指定する符号付 LONG 型の式。
Start	オプションです。STRING 型または VARIANT 型で、ブックマークに評価されます。次に説明する値のいずれかです。 <ul style="list-style-type: none">AdBookmarkCurrent — 値は 0。これがデフォルトです。カレント・レコードから開始します。Start に値を渡さないことと、論理的に等価です。AdBookmarkFirst — 値は 1。最初のレコードから開始します。AdBookmarkLast — 値は 2。最後のレコードから開始します。

コメント

NumRecords 引数が正の値の場合、カレント・レコードの位置は順方向にレコードセットの最後に向かって移動します。NumRecords が負の値の場合、カレント・レコードの位置は逆方向にレコードセットの最初に向かって移動します。

Move コールでカレント・レコードの位置を最初のレコードより前に移動しようとする、ActiveX Data Objects for Windows CE はカレント・レコードを最初のレコードより前に移動し、BOF が TRUE になります。BOF プロパティがすでに TRUE のときにさらに前に移動しようすると、エラーになります。

Move コールでカレント・レコードの位置を最後のレコードより後に移動しようとする、ActiveX Data Objects for Windows CE はカレント・レコードを最後のレコードより後に移動し、EOF が TRUE になります。EOF プロパティがすでに TRUE のときにさらに後に移動しようすると、エラーになります。

空の Recordset オブジェクトで Move メソッドをコールすると、エラーになります。

Start 引数を渡すと、(Recordset オブジェクトがブックマークをサポートしている場合) このブックマークがあるレコードに対して相対的に移動します。指定されない場合は、カレント・レコードに対して相対的に移動します。

次のコード例は、Move メソッド用の定数を設定しています。

```
Const adBookmarkCurrent = 0
Const adBookmarkFirst = 1
Const adBookmarkLast = 2
```

MoveFirst、MoveLast、MoveNext、MovePrevious

これらのメソッドは、それぞれ、指定された Recordset オブジェクトの中で最初のレコード、最後のレコード、次のレコードまたは前のレコードに移動し、移動先のレコードをカレント・レコードにします。

構文

```
recordset.MoveFirst
recordset.MoveLast
recordset.MoveNext
recordset.MovePrevious
```

コメント

MoveFirst メソッドは、カレント・レコードの位置をレコードセットの最初のレコードに移動するために使用します。

MoveLast メソッドは、カレント・レコードの位置をレコードセットの最後のレコードに移動するために使用します。

MoveNext メソッドは、カレント・レコードの位置をレコードセットの次のレコードに（最後に向かって順方向に 1 つ）移動するために使用します。最後のレコードがカレント・レコードで MoveNext メソッドをコールした場合、ActiveX Data Objects for Windows CE はカレント・レコードをレコードセットの最後のレコードより後に移動し、EOF が TRUE になります。EOF プロパティがすでに TRUE のときにさらに後に移動しようとすると、エラーになります。

MovePrevious メソッドは、カレント・レコードの位置をレコードセットの前のレコードに（最初に向かって逆方向に 1 つ）移動するために使用します。最初のレコードがカレント・レコードで MovePrevious メソッドをコールした場合、ActiveX Data Objects for Windows CE はカレント・レコードをレコードセットの最初のレコードより前に移動し、BOF が TRUE になります。BOF プロパティがすでに TRUE のときにさらに前に移動しようとすると、エラーになります。

Open

このメソッドはカーソルをオープンします。デフォルトは読取り専用です。

構文

`recordset.Open Source, ActiveConnection, CursorType, LockType, Options`

パラメータ

表 3-9 Open のパラメータ

パラメータ	説明
Source	必須です。VARIANT 型で、表の名前または SQL 文に評価されます。
ActiveConnection	オプションです。長さが 0 の文字列 ("") で H/PC オブジェクト・ストアを表します。ActiveX Data Objects for Windows CE 1.0 ではこれが唯一の有効なオプションです。
CursorType	オプションです。レコードセットで許可されている移動、および基礎となるデータベースに対する更新でレコードセットに反映されるものを判断します。内容は次のとおりです。 <ul style="list-style-type: none">■ AdOpenForwardOnly — 値は 0。デフォルトです。Forward-only カーソルです。静的カーソルと同じですが、レコードを順方向でしかスクロールできない点が異なります。デスクトップ・コンピュータの ActiveX Data Objects との互換性のためのものです。■ AdOpenKeyset — 値は 1。Keyset カーソルです。他のユーザーによる追加、変更および削除は参照できません。レコードセット内のすべての種類の移動が許可されます。 他の値を使用すると、CursorType は 1 (AdOpenKeyset) になります。動的カーソルおよび静的カーソルは、ActiveX Data Objects for Windows CE では使用できません。

表 3-9 Open のパラメータ（続き）

パラメータ	説明
LockType	<p>オプションです。Recordset をオープンするときにプロバイダが使用する必要がある、ロック（並行性）のタイプを判断します。値は次のとおりです。</p> <ul style="list-style-type: none">■ AdLockReadOnly — 値は 1。これがデフォルトです。レコードを追加、削除または変更することはできません。■ AdLockOptimistic — 値は 3。レコードを追加、削除および変更できます。 <p>他の値を使用すると、LockType は 3（AdLockOptimistic）になりますが、表自体が読取り専用の場合は別です。その場合、LockType は 1（AdLockReadOnly）になります。デスクトップ・コンピュータの ActiveX Data Objects との互換性のために、デフォルトの LockType は 1（AdLockReadOnly）になっています。即時バッチ・ロックおよび最適バッチ・ロックは ActiveX Data Objects for Windows CE では使用できません。</p>
Options	<p>Options の内容は次のとおりです。</p> <ul style="list-style-type: none">■ AdCmdText — 値は 1。Source を SQL 文として評価します。■ AdCmdTable — 値は 2。Source を MSysTables の表の名前として評価します。■ AdCmdStoredProc — 値は 4。Source を MSysProcs のストアド・プロシージャとして評価します。■ AdCmdUnknown — 値は 8。これがデフォルトです。Source プロパティ内のコマンドのタイプは不明です。

コメント

ActiveX Data Objects for Windows CE には Command オブジェクトはありません。したがって、Source 引数は文字列である必要があります。

ActiveX Data Objects for Windows CE には Connection オブジェクトはありません。オブジェクト・ストアが唯一可能な接続です。ActiveConnection 引数が指定されている場合は、長さが 0 の文字列 ("") である必要があります。

使用しているコマンドのタイプがわかっている場合、Options 引数を設定すると ActiveX Data Objects for Windows CE が関連コードに直接移動します。Options 引数が Source 引数のコマンドのタイプと一致しない場合、Open メソッドをコールしたときにエラーになります。

CREATE TABLE のような行を返さない SQL コマンドを設定して Open メソッドがコールされると、レコードセットは返されません。また、レコードセットの状態はクローズのままです。

ActiveX Data Objects for Windows CE データベースが主にシングル・ユーザーであっても、他のアプリケーションで同じデータベースのレコードセットをオープンできます。ActiveX Data Objects for Windows CE は、オープンする各レコードセットから参照されるすべてのデータベース行へのポインタを保持するキーセットを生成します。マルチユーザーの場合は、基礎となるデータベースの変更によってこのキーセットが動的に更新されないため、レコードセットを最初にオープンした後に他のアプリケーションによって削除されたレコードへのポインタを保持し続ける可能性があります。これらの削除済みレコードにアクセスしようとすると、エラーになります。現在オープンしているレコードセットを使用してデータベースに行が追加されると、そのレコードセットに対するキーセットも更新されます。

マルチユーザーの場合、別のプログラムによってデータベースに行が追加されたときに、そのデータベースを参照している他のレコードセットに対するキーセットは動的に更新されないため、別のプログラムによって追加された行は参照できません。キーセットが認識している行を別のアプリケーションが変更した場合は、その行に対するポインタは有効であるため参照できます。新規のキーセットを生成して別のプログラムによる追加と削除を参照するには、レコードセットをクローズして再オープンします。複製されたレコードセットは共通のキーセットを共有します。

例

次のコード例は、Open メソッド用の CursorType 定数と LockType 定数を設定しています。

```
Const adOpenKeyset = 1
Const adLockOptimistic = 3
Dim rstLocked, rstUpdateable
Set rstLocked = CreateObject("oladoce.recordset")
Set rstUpdateable = CreateObject("oladoce.recordset")
'The default is to open a read-only, forward-only recordset
rstLocked.Open "table1"
'You must specify other parameters to make a recordset 'updateable
rstUpdateable.Open "table2","", adOpenKeyset, adLockOptimistic
```

Supports

指定された Recordset オブジェクトが特定のタイプの機能をサポートするかどうかを判断します。

構文

```
Boolean = recordset.Supports(CursorOptions)
```

表 3-10 Supports のパラメータ

パラメータ	説明
CursorOptions	LONG 型の式。次の値のいずれか 1 つまたは値の組合せです。 <ul style="list-style-type: none">■ AdAddNew — 値は 16778240。AddNew メソッドをサポートします。■ AdApproxPosition — 値は 16384。AbsolutePosition プロパティおよび AbsolutePage プロパティをサポートします。■ AdBookmark — 値は 8192。Bookmark プロパティをサポートします。■ AdDelete — 値は 16779264。Delete メソッドをサポートします。■ AdMovePrevious — 値は 512。MovePrevious メソッドと、Move メソッドによるカレント・レコードの位置の逆方向移動をサポートします。■ AdUpdate — 値は 16809984。Update メソッドをサポートします。■ AdHoldRecords — 値は 256。サポートされません。■ AdResync — 値は 131072。サポートされません。■ AdUpdateBatch — 値は 65536。サポートされません。

コメント

Supports メソッドは、Recordset オブジェクトがサポートする機能のタイプを判断するために使用します。CursorOptions に含まれている定数に対応する機能を Recordset オブジェクトがサポートする場合、Supports メソッドは TRUE を返します。それ以外の場合は、FALSE を返します。

定数 AdHoldRecords、AdResync および AdUpdateBatch の場合、ActiveX Data Objects for Windows CE は常に FALSE を返します。他の定数に対しては、TRUE を返す可能性があります。

例

次のコード例は、Supports メソッド用の定数を設定しています。

```
Const adHoldRecords = &H00000100
Const adMovePrevious = &H00000200
Const adAddNew = &H01000400
Const adDelete = &H01000800
Const adUpdate = &H01008000
Const adBookmark = &H00002000
Const adApproxPosition = &H00004000
Const adUpdateBatch = &H00010000
Const adResync = &H00020000
Const adNotify = &H00040000
```

Update

Recordset オブジェクトのカレント・レコードに対するすべての変更を保存します。

構文

`recordset.Update Fields, Values`

表 3-11 Update のパラメータ

パラメータ	説明
Fields	オプションです。変更するフィールドの単一フィールド名、または複数のフィールド名やフィールドの位置を示す順序数を表す配列です。
Values	オプションです。新規レコードのフィールドの単一の値、または複数の値を表す配列です。

コメント

Update メソッドは、AddNew メソッドのコール後、または既存レコードのフィールド値の変更後の、Recordset オブジェクトのカレント・レコードに対するすべての変更を保存するために使用します。Recordset オブジェクトは更新をサポートする必要があります。フィールドの値を設定するには、次のいずれかの方法を実行します。

- Field オブジェクトの Value プロパティに値を割り当てて Update メソッドをコールします。
- フィールド名と値を Update コールの引数として渡します。
- フィールド名の配列と値の配列を Update コールに渡します。

フィールド名の配列と値の配列を使用する場合は、両方の要素数が同一であることが必要です。また、フィールド名の順序はフィールド値の順序と一致する必要があります。フィールド名と値の数および順序が一致しない場合、エラーになります。

Update メソッドをコールする前に追加または編集中のレコードから移動すると、ActiveX Data Objects for Windows CE は自動的に Update をコールして変更を保存します。カレント・レコードに対するすべての変更を取り消す、または新規に追加したレコードを廃棄する場合は、CancelUpdate メソッドをコールする必要があります。

カレント・レコードは、Update メソッドのコール後もカレントのままです。

Recordset オブジェクトのプロパティ

次の表に、Recordset オブジェクトがサポートするプロパティのリストを示します。

表 3-12 Recordset オブジェクトのプロパティ

プロパティ	説明
AbsolutePage***	新規のカレント・レコードに対して移動するページを指定します。
AbsolutePosition	Recordset オブジェクト内でのカレント・レコードの位置を示す順序数を指定します。
ActiveConnection	現在のデータベース接続を設定します。ActiveX Data Objects for Windows CE 1.0 では、常に長さが 0 の文字列 ("") です。
BOF	カレント・レコードの位置が Recordset オブジェクトの最初のレコードより前かどうかを示します。
EOF	カレント・レコードの位置が Recordset オブジェクトの最後のレコードより後であることを示します。
Bookmark***	Recordset オブジェクト内でレコードを一意に識別するブックマークを指定します。
CacheSize***	Recordset オブジェクトからローカルのメモリーにキャッシュされるレコード数を指定します。
Count	レコードセット内のフィールド数を示します。
CursorType	Recordset オブジェクト内で使用するカーソルのタイプを示します。
EditMode	カレント・レコードの編集ステータスを示します。
LockType	編集中のレコードに対するロックのタイプを示します。
PageCount***	Recordset オブジェクトに含まれるデータのページ数を示します。
PageSize***	Recordset の 1 ページを構成するレコード数を示します。
RecordCount	Recordset オブジェクトのカレント・レコード数を示す LONG 型の値を返します。
Source	Recordset オブジェクト、SQL 文または表の名前に含まれるデータのソースを示します。
ActualSize	フィールド値の実際の長さをバイト数で示します。
Attributes	Field オブジェクトの 1 つまたは複数の特性を示します。
DefinedSize	Field オブジェクトのデータ容量を判断するために使用します。
Name	Field オブジェクトの名前を返します。

表 3-12 Recordset オブジェクトのプロパティ（続き）

プロパティ	説明
Type	Field オブジェクトのデータ型を示します。
Underlying Value	Field オブジェクトの現在の値を示します。
Value	Field オブジェクトの値を示します。

レコードセットは仮想データベース表で、そのフィールドおよび行は H/PC の実際のデータベース表のフィールドおよび行のサブセットと対応しています。レコードセットの行に含まれる情報を追加、削除または変更すると、基礎となる表の対応している部分にその変更内容を渡せます。レコードセットのデータを変更するとその変更内容はメモリーに格納されるので、基礎となるデータベースの更新前に変更を取り消すことができます。ActiveX Data Objects for Windows CE では、バッチ更新をサポートしていません。データが変更されても基礎となるデータベースがコミットされていない行は、同時に 1 つしか存在できません。

レコードセット表の構造を変更すると、基礎となるデータベース表に即時に反映されます。

AbsolutePage

新規のカレント・レコードが常駐するページを指定します。

構文

```
var = recordset.AbsolutePage
```

戻り値

1 から Recordset オブジェクトに含まれるページ数 (PageCount) までの LONG 型の値を返します。

コメント

AbsolutePage プロパティは、カレント・レコードが置かれたページの番号を識別するために使用します。PageSize プロパティは、Recordset オブジェクトを一連のページに論理的に分割するために使用します。各ページには PageSize と同じレコード数が含まれますが、最後のページのレコード数は少ない場合があります。

AbsolutePosition プロパティと同様に、AbsolutePage は 1 から数えられ、カレント・レコードが Recordset の最初のレコードの場合に 1 になります。このプロパティは、特定のページの最初のレコードに移動するときに設定します。PageCount プロパティからページ数の合計を取得します。

AbsolutePage は 3 つの特殊な値、-1 (AdPosUnknown)、-2 (AdPosBOF) および -3 (AdPosEOF) をサポートします。

次のコード例は、**AbsolutePage** プロパティ用の定数を設定しています。

```
Const adPosUnknown = -1
Const adPosBOF = -2
Const adPosEOF = -3
```

AbsolutePosition

Recordset オブジェクト内でのカレント・レコードの位置を示す順序数を指定します。

構文

```
var = recordset.AbsolutePosition
```

戻り値

1 から Recordset オブジェクトに含まれるレコード数 (**RecordCount**) までの LONG 型の値を返します。

コメント

AbsolutePosition プロパティは、Recordset オブジェクト内でのレコードの位置を示す順序数を基にして移動するために使用します。

AbsolutePage プロパティと同様に、**AbsolutePosition** は 1 から数えられ、カレント・レコードが Recordset の最初のレコードの場合に 1 になります。**RecordCount** プロパティから、Recordset オブジェクトに含まれるレコード数の合計を取得できます。

AbsolutePosition プロパティを、レコード番号の代用として使用しないでください。指定されたレコードの位置は、先行するレコードを削除したときに変更されます。また、Recordset オブジェクトが再オープンされた場合に、指定されたレコードが同じ **AbsolutePosition** を持つという保証はありません。指定した位置を保持してそこに戻る方法としては、ブックマークをお勧めします。これは、すべてのタイプの Recordset オブジェクトで通用する唯一の方法です。

AbsolutePosition は 3 つの特殊な値、-1 (**AdPosUnknown**)、-2 (**AdPosBOF**) および -3 (**AdPosEOF**) をサポートします。

AbsolutePosition に 0 または負の値を設定すると、エラーになります。

ActiveConnection

現在のデータベース接続を設定します。

構文

```
var = recordset.ActiveConnection
```

コメント

ActiveConnection プロパティは、次の理由により書き込み専用となっています。

- このプロパティはデスクトップ・コンピュータの ActiveX Data Objects では Connection オブジェクトを返しますが、ActiveX Data Objects for Windows CE には Connection オブジェクトがありません。
- ActiveX Data Objects for Windows CE 1.0 は、オブジェクト・ストアしか操作しません。
- このプロパティは、常に長さが 0 の文字列 ("") です。

BOF、EOF

Recordset オブジェクトで、BOF はカレント・レコードの位置が最初のレコードより前であることを示します。EOF はカレント・レコードの位置が最後のレコードより後であることを示します。

構文

```
object.BOF  
object.EOF
```

戻り値

BOF プロパティおよび EOF プロパティの戻り値はブール値です。

BOF プロパティは、カレント・レコードの位置が最初のレコードより前にあるときに TRUE を返し、カレント・レコードの位置が最初のレコードまたはそれより後にあるときは FALSE を返します。

EOF プロパティは、カレント・レコードの位置が最後のレコードより後にあるときに TRUE を返し、カレント・レコードの位置が最後のレコードまたはその前にあるときは FALSE を返します。

BOF プロパティまたは EOF プロパティが TRUE であるとき、カレント・レコードはありません。

コメント

BOF プロパティおよび EOF プロパティは、Recordset オブジェクトにレコードが含まれているか、あるいはレコードからレコードへの移動時に Recordset オブジェクトの外側に出たかどうかを判断するために使用します。

レコードがない Recordset オブジェクトをオープンした場合、BOF および EOF の両プロパティが TRUE に設定され、Recordset オブジェクトの RecordCount プロパティは 0 に設定されます。少なくともレコードを 1 つ含んでいる Recordset オブジェクトをオープンすると、最初のレコードがカレント・レコードになり、BOF プロパティおよび EOF プロパティは FALSE になります。

Delete メソッドをコールすると、Recordset の最後のレコードを削除した場合でも BOF プロパティまたは EOF プロパティの設定は変更されません。

次の表に、様々な Move メソッドをコールして正しくレコードを再配置できなかったときに、BOF プロパティおよび EOF プロパティがどのように設定されるかを示します。

表 3-13 BOF および EOF のパラメータ

メソッド	BOF	EOF
MoveFirst、MoveLast	TRUE	TRUE
Move=0	変更なし	変更なし
MovePrevious、Move<0	TRUE	変更なし
MoveNext、Move>0	変更なし	TRUE

Bookmark

Recordset オブジェクト内でカレント・レコードを一意に識別するブックマークを返します。または、Recordset オブジェクト内のカレント・レコードを、有効なブックマークにより識別されるレコードに設定します。

構文

`object.Bookmark [= value]`

表 3-14 Bookmark のパラメータ

パラメータ	説明
object	オブジェクトに評価されるオブジェクト式。
value	ブックマークに評価される VARIANT 式。

戻り値

有効なブックマークに評価される、VARIANT 式を返します。

コメント

Bookmark プロパティは、カレント・レコードの位置を保存して、いつでもそのレコードに戻るために使用します。

Recordset オブジェクトをオープンすると、その中の各レコードが一意のブックマークを持ちます。カレント・レコードのブックマークを保存するには、Bookmark プロパティの値を変数に割り当てます。別のレコードに移動した後でそのレコードに戻るには、Recordset オブジェクトの Bookmark プロパティにその変数の値を設定します。

Clone メソッドを使用して Recordset オブジェクトの複製を作成した場合は、元の Recordset オブジェクトと複製 Recordset オブジェクトの Bookmark プロパティは同一に設定されるので、どちらも同じように使用できます。ただし、異なる Recordset オブジェクトのブックマークを同じものとして使用することはできません。それらが同じソースまたはコマンドによって作成された場合であっても同様です。

CacheSize

Recordset オブジェクトからローカルのメモリーにキャッシュされるレコード数を指定します。

構文

```
recordset.CacheSize
```

コメント

このプロパティは ActiveX Data Objects for Windows CE では使用されません。常に 1 を返します。他の値を設定しようとしても無視されます。

Count

レコードセット内のフィールド数を示します。

構文

```
fields.Count
```

表 3-15 Count のパラメータ

パラメータ	説明
fields	オープンしている Recordset の Fields コレクションを指定します。

例

```
Dim rs
Set rs = CreateObject("adoce.recordset")
rs.Open "MSysIndexes"
MsgBox rs.Fields.Count
rs.Close
Set rs = Nothing
```

CursorType

Recordset オブジェクト内で使用するカーソルのタイプを示します。

構文

```
recordset.CursorType
```

戻り値

レコードセットで許可されている移動、および基礎となるデータベースに対する更新でレコードセットに反映されるものを示します。0 (AdOpenForwardOnly) または 1 (AdOpenKeyset) のどちらかです。

コメント

CursorType プロパティは、Recordset オブジェクトをオープンするときに使用するカーソルのタイプを指定するために使用します。CursorType プロパティは、レコードセットがクローズしているときは読取り / 書込み可能で、オープンしているときは読取り専用です。

Recordset オブジェクトがオープンしているときは、使用している実際のカーソルのタイプに一致するように CursorType プロパティが変更される可能性があります。返されたカーソルの特定の機能を検証するには、Supports メソッドを使用します。

ActiveX Data Objects for Windows CE が実装しているのは AdOpenForwardOnly および AdOpenKeyset のみなので、その他の値はすべて AdOpenKeyset にマップされます。

EditMode

カレント・レコードの編集ステータスを示します。

構文

```
object.EditMode
```

戻り値

次の表で説明する値のいずれかが返されます。

表 3-16 編集モードの戻り値

EditMode	値	説明
AdEditNone	0	進行中の編集作業はありません。
AdEditInProgress	1	カレント・レコードのデータが変更され、まだ保存されていません。
AdEditAdd	2	AddNew メソッドが起動されていて、コピー・バッファにあるカレント・レコードはまだデータベースに保存されていない新規レコードです。

コメント

EditMode プロパティは、カレント・レコードの編集ステータスを判断するために使用します。編集プロセスが中断されて Update メソッドまたは CancelUpdate メソッドを実行する必要があるかどうかを判断する場合に、保留中の変更があるかどうかをテストできます。

異なる編集条件での EditMode プロパティの詳細は、AddNew メソッドを参照してください。

次のコード例は、EditMode プロパティ用の定数を設定しています。

```
Const adEditNone = 0
Const adEditInProgress = 1
Const adEditAdd = 2
```

LockType

編集中のレコードに対するロックのタイプを示します。

構文

```
recordset.LockType
```

戻り値

次の表で説明する値のいずれかが返されます。

表 3-17 ロック・モードの戻り値

LockType	値	説明
AdLockReadOnly	1	デフォルトです。レコードを追加、削除または変更することはできません。
AdLockOptimistic	3	レコードを追加、削除および変更できます。

コメント

LockType プロパティは、Recordset オブジェクトをオープンするときにプロバイダが使用する必要があるロックのタイプを判断するため、または Recordset オブジェクトをオープンしたときに使用したロックのタイプを返すために使用します。LockType プロパティは、Recordset がクローズしているときは読取り / 書き込み可能で、オープンしているときは読取り専用です。

ActiveX Data Objects for Windows CE は AdLockPessimistic または AdLockBatchOptimistic をサポートしません。通告なしで AdLockOptimistic に置き換えます。

PageCount

Recordset オブジェクトに含まれるデータのページ数を示します。

構文

`recordset.PageCount`

コメント

PageCount プロパティは、Recordset オブジェクトに含まれるデータのページ数を判断するために使用します。ページはレコードのグループで、PageSize プロパティの設定と同じサイズです。最後のページが、レコード数が PageSize の値より少ないためにページとして未完成の場合でも、PageCount の値としてはフルサイズのページとして扱われます。Recordset のサイズが判断できないときは、-1 (AdUnknown) に設定されます。

ページ機能の詳細は、PageSize プロパティおよび AbsolutePage プロパティを参照してください。

PageSize

Recordset の 1 ページを構成するレコード数を示します。

構文

```
var = recordset.PageSize
```

戻り値

1 ページに含まれるレコード数を示す LONG 型の値を返します。デフォルトは 10 です。

コメント

PageSize プロパティは、データの論理ページを構成するレコード数を判断するために使用します。ページのサイズを確定すると、AbsolutePage プロパティを使用して特定のページの最初のレコードに移動できます。ユーザーがデータ全体をページングして一度に特定の数のレコードを参照できるようにするときに、有効です。

このプロパティはいつでも設定でき、値は特定ページの最初のレコード位置を計算するために使用されます。

RecordCount

Recordset オブジェクト内の現在のレコード数を示します。

構文

```
var = object.RecordCount
```

戻り値

LONG 型の値を返します。

コメント

RecordCount プロパティは、Recordset オブジェクトに含まれるレコード数を判断するために使用します。ActiveX Data Objects for Windows CE がレコード数を判断できないときは、-1 (AdUnknown) を返します。

クローズしている Recordset オブジェクトの RecordCount プロパティを読み取ろうとすると、エラーになります。

Source

Recordset オブジェクト、SQL 文または表の名前に含まれるデータのソースを示します。

構文

```
var = recordset.Source
```

戻り値

STRING 型の値を返します。

コメント

Source プロパティは、Recordset オブジェクトのデータ・ソースを指定するために使用します。

Source 引数は Open 文で必須であり、ActiveX Data Objects for Windows CE には Execute メソッドと Command オブジェクトがないので、このプロパティを設定しても影響がありません。

Field オブジェクトのプロパティ

次の表に、Field オブジェクトがサポートするプロパティのリストを示します。

表 3-18 Field オブジェクトのプロパティ

メソッド	説明
ActualSize	フィールド値の実際の長さを、バイト数で示します。
Attributes	Field オブジェクトの 1 つまたは複数の特性を示す値を返します。このプロパティは、読取り専用です。
DefinedSize	Field オブジェクトのデータ容量を判断するために使用します。定義されたフィールドのサイズを、文字数で返します。サイズをバイト数で返す ActualSize と比較してください。
Name	フィールドの名前を返します。このプロパティは、読取り専用です。
Type	Field オブジェクトのデータ型を示します。Type プロパティは、読取り専用です。
UnderlyingValue	データベース内の、Field オブジェクトの現在の値を示します。
Value (デフォルト)	レコードセット内の、Field オブジェクトの現在の値を示します。

備考

Field オブジェクトにはメソッドもイベントもありません。Value を除いてすべてのプロパティが読取り専用です。次のコード例は、Field オブジェクトの使用方法を示しています。

```
Dim rs, f
Set rs = CreateObject("oladoce.recordset")
rs.open "MyTables"
Set f = rs.Fields(0)
Msgbox f.Value, f.Name
```

ActualSize

フィールド値の実際の長さを、バイト数で示します。

構文

```
var = field.ActualSize
```

戻り値

LONG 型の値を戻します。

コメント

ActualSize は、Field オブジェクトの値の実際の長さをバイト数で返すために使用します。すべてのフィールドに対して、ActualSize プロパティは読取り専用です。ActiveX Data Objects for Windows CE がフィールド値の長さを判断できない場合、ActualSize プロパティは -1 (AdUnknown) を返します。

文字データ型では、最大許容サイズを判断するには DefinedSize を使用するほうが便利な場合があります。

Attributes

Field オブジェクトの 1 つまたは複数の特性を示します。このプロパティは、読取り専用です。

構文

```
var = field.Attribute
```

戻り値

Field オブジェクトでは、この値がフィールドの特性を示します。次の表に示す 1 つまたは複数の値の合計をとることができます。

表 3-19 Attributes の戻り値

戻り値	値	説明
AdFldMayDefer ##	2	フィールドが遅延処理され、明示的にアクセスしたときを除いて、フィールドの値がレコード全体とともにデータ・ソースから取り出されるわけではないことを示します。
AdFldUpdatable	4	フィールドに書き込めることを示します。
AdFldUnknownUpdatable ##	8	フィールドに書き込めるかどうか、プロバイダが判断できないことを示します。
AdFldFixed	16	フィールドに固定長データが含まれていることを示します。すべてのデータ型で使用できますが、AdVarWChar、AdLongVarWChar、AdVarBinary および AdLongVarbinary では使用できません。
AdFldIsNullable	32	フィールドが NULL 値を受け入れることを示します。
AdFldMayBeNull	64	フィールドから NULL 値を読み取れることを示します。
AdFldLong ##	128	フィールドがロング・バイナリ型のフィールドであることを示します。または、AppendChunk メソッドおよび GetChunk メソッドを使用できることを示します。
AdFldRowID ##	256	フィールドにレコード番号、一意の識別子などのレコード識別子が含まれていることを示します。

ActiveX Data Objects for Windows CE では、AdFldRowVersion または AdFldCacheDeferred の値を返しません。

例

次のコード例は、Attributes プロパティ用の定数を設定しています。

```
Const adFldMayDefer = 2
Const adFldUpdatable = 4
Const adFldUnknownUpdatable = 8
Const adFldFixed = 16
Const adFldIsNullable = 32
Const adFldMayBeNull = 64
Const adFldLong = 128
Const adFldRowID = 256

Dim rs, n
```

```
Set rs = CreateObject ("oladoce.recordset")
rs.Open "table1"
For n = 0 to rs.Fields.Count -1
MsgBox rs.Fields(n).Attributes
Next
```

DefinedSize

Field オブジェクトのデータ容量を判断するために使用します。定義されたフィールドのサイズを、文字数で返します。サイズをバイト数で返す **ActualSize** と比較できます。

構文

```
var = field.DefinedSize
```

戻り値

フィールドの最大長を指定する数を返します。

コメント

ActiveX Data Objects for Windows CE は、256 文字より少ない文字列と最大 32,733 の Unicode 文字をサポートできるメモ文字列の、2 種類の文字列型をサポートします。長さは表が作成されたときに設定され、データが設定されたときに施行されます。256 文字より少ない文字列の場合、**DefinedSize** は表が作成されたときに指定された長さを返します。長さが定義されていないテキスト・フィールドの場合、**DefinedSize** はフィールドに保持できる最大 Unicode 文字数である 32,733 を返します。

DefinedSize プロパティと **ActualSize** プロパティは異なります。たとえば、データ型 202 (AdVarChar) で最大長 50 文字として宣言された Field オブジェクトの場合に返される **DefinedSize** プロパティの値は 50 ですが、**ActualSize** プロパティの値にはカレント・レコードのそのフィールドに格納されているデータのバイト数が返されます。AdVarChar は 1 文字当たり 2 バイトを使用するため、定義されたサイズ (文字数) より格納データのバイト数のほうが大きい場合もあります。

Name

フィールドの名前を返します。このプロパティは、読み取り専用です。

構文

```
var =field.Name
```

戻り値

名前を指定する STRING 型の値を返します。

コメント

フィールド名は 64 文字以下です。

例

```
Dim rs, n
Set rs = CreateObject ("oladoce.recordset")
rs.Open "table1"
For n = 0 to rs.Fields.Count -1
MsgBox rs.Fields(n).Name
Next
```

Type

Field オブジェクトのデータ型を示します。このプロパティは、読取り専用です。

構文

```
var = field.Type
```

戻り値

次の表で説明する値のいずれかが返されます。

表 3-20 Type の戻り値

定数	値	説明
AdVarChar	202	256 文字より少ない、NULL で終了する Unicode 文字の文字列。
AdLongVarChar	203	NULL で終了する Unicode 文字の文字列。
AdVarBinary	204	256 文字より少ないバイナリ値。
AdLongVarBinary	205	65533 (4096*16 - 3) バイト以下のバイナリ値。
AdInteger	3	4 バイトの符号付き整数。
AdSmallInt	2	2 バイトの符号付き整数。
AdDouble	5	倍精度浮動小数点値。
AdDate	7	DATE 型の値。
AdUnsignedSmallInt	18	2 バイトの符号なし整数。
AdUnsignedInt	19	4 バイトの符号なし整数。
AdBoolean	11	ブール型の TRUE/FALSE 値。
AdDouble	5	倍精度浮動小数点数。

デスクトップ・コンピュータの ActiveX Data Objects では Type に他の値も使用できますが、ActiveX Data Objects for Windows CE ではサポートされず戻り値としても使用されません。

例

次のコード例は、Type プロパティ用の定数を設定しています。

```
Const adVarChar = 202
Const adLongVarChar = 203
Const adVarBinary = 204
Const adLongVarBinary = 205
Const adInteger = 3
Const adSmallInt = 2
Const adDouble = 5
Const adDate = 7
Const adUnsignedSmallInt = 18
Const adUnsignedInt = 19
Const adBoolean = 11
Const adDouble = 5

Dim rs, n
Set rs = CreateObject ("oladoce.recordset")
rs.Open "table1"
For n = 0 to rs.Fields.Count -1
MsgBox rs.Fields(n).Type
Next
```

UnderlyingValue

データベース内の、Field オブジェクトの現在の値を示します。

構文

```
var = field.UnderlyingValue
```

戻り値

VARIANT 型の値を返します。

コメント

UnderlyingValue プロパティは、データベースからフィールドの現在の値を返すために使用します。

OriginalValue プロパティは ActiveX Data Objects for Windows CE ではサポートされません。

Value

Field オブジェクトに割り当てられた値を示します。

構文

```
var = field.Value
```

戻り値

VARIANT 型の値を返します。デフォルト値は、Type プロパティに依存します。

コメント

Value プロパティは、Field オブジェクトにデータを設定するかデータを返すために使用します。

ActiveX Data Objects for Windows CE では、Value プロパティでロング・バイナリ値を設定または返せます。

Recordset オブジェクトのコレクション

Recordset オブジェクトには、コレクションが 1 つあります。Fields コレクションです。索引には、対応するコレクションはありません。

Fields

Fields コレクションには、Recordset の各列の Field オブジェクトが 1 つずつ含まれます。名前または索引によって、特定のフィールドを参照できます。Fields コレクションは Count プロパティをサポートします。

次のコード例に、Fields コレクションを使用して MyTable 表からすべてのフィールド名を取得する方法を示します。

```
Dim rs, n
Set rs = CreateObject("oladoce.recordset")
rs.open "MyTable"
For n = 0 to rs.Fields.Count -1
Msgbox rs.Fields(n).Name
Next
```

SQL およびデータベースの参照

ActiveX Data Objects for Windows CE は Oracle Lite データベース for Windows CE の SQL 構文をすべてサポートしますが、次の 2 つはサポートしません。

- 文の最後のセミコロン。
- commit や rollback などの単一語からなる文。

次の表に Oracle Lite データベースの ActiveX Data Objects for Windows CE でサポートされない ActiveX Data Objects for Windows CE のシステム表のリストを示します。

表 3-21 ADOCE のシステム表

表	説明
MsysTables	ActiveX Data Objects for Windows CE が認識するすべての表とその特性。
MsysIndexes	ActiveX Data Objects for Windows CE が認識するすべての表に対する索引。
MsysFields	ActiveX Data Objects for Windows CE が認識するすべての表のフィールド。
MsysProcs	名前で実行できるストアド SQL 文。

エラー・メッセージ

Oracle Lite データベース ActiveX Data Objects for Windows CE は ActiveX Data Objects for Windows CE で発生する可能性があるエラーをマップして、独自に使用します。

ActiveX Data Objects for Windows CE は、ActiveX Data Objects エラーと SQL エラーを生成できます。ActiveX Data Objects エラーは、コントロールのメソッドおよびプロパティが適切に使用されなかったときに発生します。SQL エラーは、Open メソッドの中で使用された SQL 文に問題があるときに生成されます。

エラーには、番号と関連するエラー・テキストがあります。テキストは、Err オブジェクトの Description プロパティを使用して取り出されます。Helpfile 情報または HelpContext 情報はありません。

エラーは、インライン・エラー・トラップで On Error Resume Next 文を使用して獲得されます。インラインでエラーをトラップするには、操作の完了後に Err オブジェクトをチェックして、Err.Number が 0 以外の値かどうかを確認します。Err.Number に返されるエラー・コードを使用すると、それが SQL エラーであっても ActiveX Data Objects エラーであっても適切に処理できます。

ActiveX Data Objects のエラー

次の表に、ActiveX Data Objects のエラー値のリストを示します。

表 3-22 ADOCE のエラー

エラー	値	説明
AdErrInvalidArgument	3001	型が誤っている、許容域を超えている、または互いに競合している引数をアプリケーションが使用しています。
AdErrNoCurrentRecord	3021	BOF または EOF が TRUE であるか、カレント・レコードが削除されています。アプリケーションが要求している操作には、カレント・レコードが必要です。
AdErrIllegalOperation	3219	アプリケーションが要求している操作は、このコンテキストではできません。
AdErrFeatureNotAvailable	3251	アプリケーションが要求している操作は、プロバイダによってサポートされていません。
AdErrItemNotFound	3265	ActiveX Data Objects は、アプリケーションが要求した名前または序数による参照に対応するオブジェクトを、コレクションの中で検索できませんでした。
AdErrObjectNotSet	3420	アプリケーションによるオブジェクト参照が、有効なオブジェクトを指していません。
AdErrDataConversion	3421	アプリケーションが現在の操作に使用している値の型が誤っています。
AdErrObjectClosed	3704	アプリケーションが要求している操作は、オブジェクトがクローズしているときは実行できません。
AdErrObjectOpen	3705	アプリケーションが要求している操作は、オブジェクトがオープンしているときは実行できません。
AdErrProviderNotFound	3706	ActiveX Data Objects は、指定されたプロバイダを見つけられませんでした。
AdErrInvalidConnection	3709	指定した接続文字列が無効です。

SQL エラー

次の表に、発生する可能性がある SQL エラーのリストを示します。表に示したエラー値は 16 進表現で、コード中では Hex(Err.Number) を使用して表示できます。

表 3-23 SQL エラー

エラー	値	内容	説明
DB_E_CANTCONVERTVALUE	80040E07	型を変換できません "constant"。	コマンド・テキスト中のリテラル値が、(データ・オーバーフロー以外の理由で) 関連する列の型に変換できませんでした。エラー文字列には問題の定数が含まれています。
DB_E_DATAOVERFLOW	80040E57	定数値 "constant" がオーバーフローしました。	コマンド・テキスト中のリテラル値が、関連する列によって指定されている型をオーバーフローしました。エラー文字列には問題の定数が含まれています。
DB_E_ERRORSINCOMMAND	80040E14	"token" の近くに不正な構文があります。	コマンド・テキストに 1 つまたは複数のエラーがありました。一般には、構文エラーまたは予想外のキーワードです。エラー文字列には予想外のトークンが含まれています。
E_OUTOFMEMORY	8007000E	メモリー不足です。	メモリー不足です。
DB_E_NOTABLE	80040E37	表 "table" が存在しません。	指定された表が存在しません。エラー文字列にはエラーになった表の名前が含まれています。
DB_E_BADCOLUMNID	80040E11	"field" フィールドが存在しません。	指定された列が存在しませんでした。エラー文字列にはエラーになったフィールドの名前が含まれています。
DB_E_DUPLICATETABLEID	80040E3F	表 "table" はすでに存在します。	指定された表は、すでに現在のデータ・ソースに存在しています。
DB_E_DUPLICATEINDEXID	80040E34	索引 "index" はすでに存在します。	指定された索引は、すでに現在のデータ・ソース・オブジェクトに存在しています。

表 3-23 SQL エラー（続き）

エラー	値	内容	説明
DB_E_NOINDEX	80040 E35	索引 "index" が存在しません。	指定された索引が現在のデータ・ソースに存在しないか、指定された表に適用されませんでした。
DB_E_DUPLICATECOLUMNID	80040 E3E	"field" フィールドはすでに存在します。	複数の要素のフィールド名が同じでした。
DB_E_NOCOMMAND	80040 E0C	未対応	コマンドが設定されていません。
DB_E_BADBOOKMARK	80040 E0E	未対応	ブックマークが無効です。
DB_E_DELETEDROW	80040 E23	未対応	行が削除されています。
DB_E_CANTFETCHBACKWARDS	80040 E24	未対応	forward-only カーソルは逆方向には読み取れません。
DB_E_FIELDDIFFERENT	80040 E41	"%1!s" の近くに無効なフィールド比較があります。	型が異なる 2 つのフィールドを比較しようとしてしました。
DB_E_FIELDMAXEXCEED	80040 E42	表の "%1!s" の近くで最大列数を超過しました。	表当たりの最大数を超える列を使用しようとしてしました。

パッケージ・ウィザードの使用

この章では、Mobile Development Kit のパッケージ・ウィザード・ユーティリティについて説明します。内容は次のとおりです。

- [パッケージ・ウィザードの概要](#)
- [パッケージ・ウィザードの起動](#)
- [新規アプリケーションの命名](#)
- [アプリケーション・ファイルのリスト表示](#)
- [データベース情報の入力](#)
- [レプリケーション用スナップショットの定義](#)
- [アプリケーションの完了](#)

パッケージ・ウィザードの概要

パッケージ・ウィザードは、次の目的に使用できるグラフィカル・ツールです。

- Windows プラットフォーム用の新規 Mobile サーバー・アプリケーションを作成する。
- 既存の Mobile サーバー・アプリケーションを編集する。

新規 Mobile サーバー・アプリケーションを作成するときは、コンポーネントを定義し Mobile サーバー・リポジトリにパブリッシュします。場合によっては、既存の Mobile サーバー・アプリケーションのコンポーネントの定義を編集することがあります。たとえば、アプリケーションの新規バージョンを開発する場合は、パッケージ・ウィザードを使用してアプリケーション定義を更新します。パッケージ・ウィザードを使用すると、アプリケーション・コンポーネントを **.jar** ファイルにパッケージ化することもできます。

パッケージ・ウィザードの起動

パッケージ・ウィザードを起動するには、DOS プロンプトで次のように入力します。

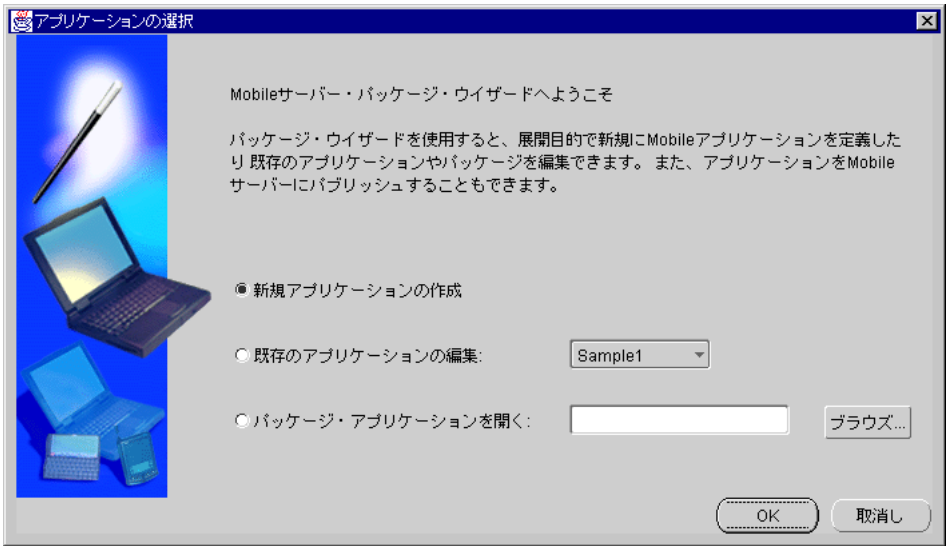
```
wtgpack
```

パッケージ・ウィザードが表示され、デフォルトで「ようこそ」パネルが表示されます。「ようこそ」パネルでは、次の機能を使用して、パッケージ化されたアプリケーションの作成、編集またはオープンができます。

表 4-1 「ようこそ」パネルのオプション

機能	説明
新規アプリケーションの作成	このオプションを選択すると、新規アプリケーションを定義できます。
既存のアプリケーションの編集	このオプションを選択すると、既存のアプリケーションを編集できます。隣にあるドロップダウン・リストから既存のアプリケーションを選択できます。
パッケージ・アプリケーションを開く	このオプションを選択すると、 .jar ファイルとしてパッケージ化されているアプリケーションを選択できます。隣にあるフィールドにパッケージ・アプリケーションの名前を入力するか、「ブラウズ」ボタンを使用してアプリケーションを選択します。

図 4-1 「ようこそ」 パネル



新規アプリケーションの命名

「アプリケーション」パネルは、Mobile サーバー・アプリケーションに名前を付けて、このアプリケーションを Mobile サーバー上のどこに格納するかを指定するために使用します。このパネルに含まれるフィールドは次のとおりです。

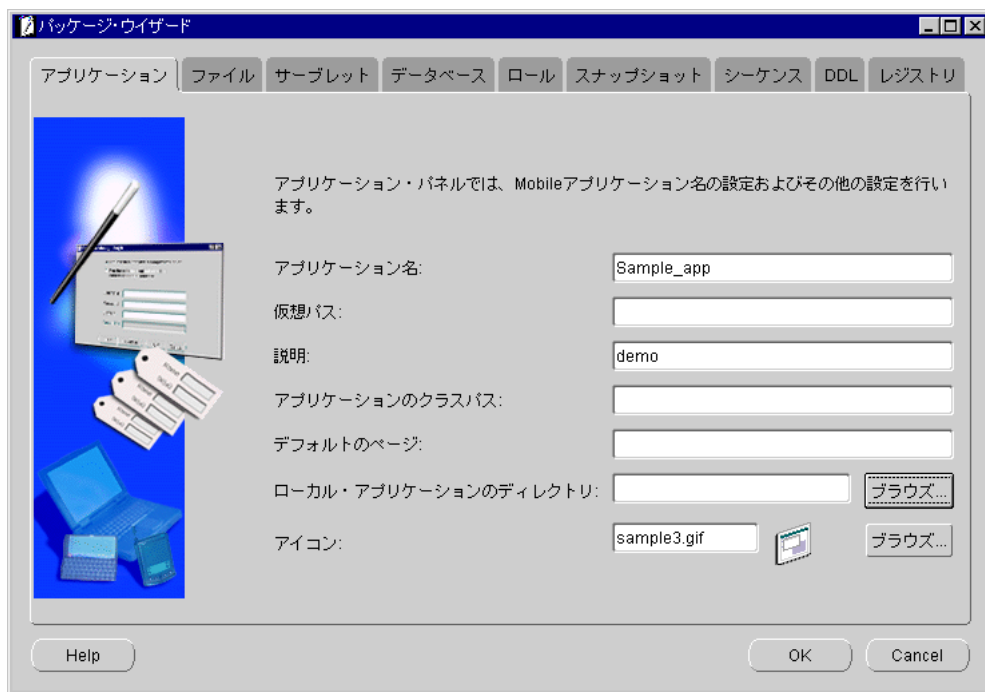
表 4-2 「アプリケーション」パネルのオプション

フィールド	説明	必須
アプリケーション名	Windows アプリケーションの名前。	○
仮想パス	サーバー・リポジトリのルート・ディレクトリから Web-to-Go アプリケーション自体にマップされるパス。仮想パスにより、アプリケーションのディレクトリ構造全体を参照する必要がなくなります。アプリケーションに対して一意の ID も提供します。	○
説明	Windows アプリケーションの概要。	○
アプリケーションのクラスパス	セミコロン (;) で区切られたクラス・ディレクトリ。Web-to-Go はここに指定された順番でアプリケーション・クラスを検索します。	×

表 4-2 「アプリケーション」パネルのオプション（続き）

フィールド	説明	必須
デフォルトのページ	Web-to-Go アプリケーションのエントリ・ポイントとして機能する Web ページのサーバー位置。これは、リポジトリのディレクトリに対する相対パスです。たとえば、サーバー・ディレクトリが /apps で、デフォルトのページが index.html の場合、デフォルトのページは /apps/index.htm になります。デフォルト・ページはサブレットにできます。ユーザーがデフォルト・ページを指定しない場合は、汎用ページが発行されます。	×
ローカル・アプリケーションのディレクトリ	ローカル・マシン上でこのアプリケーションの全コンポーネントが含まれているディレクトリです。この場所は、入力するかまたは「ブラウズ」ボタンをクリックして選択します。	×
アイコン	Mobile サーバー・ワークスペースで Mobile サーバー・アプリケーションのアイコンとして使用される GIF イメージです。隣にあるフィールドにアイコンの名前を入力するか、「ブラウズ」ボタンを使用してアイコン・ファイルを選択します。	×

図 4-2 「アプリケーション」パネル



プラットフォーム・ファイルの命名

ローカル・アプリケーションのディレクトリが必要です。Windows のアプリケーション・ファイルは、「win32」というサブディレクトリに入れる必要があります。

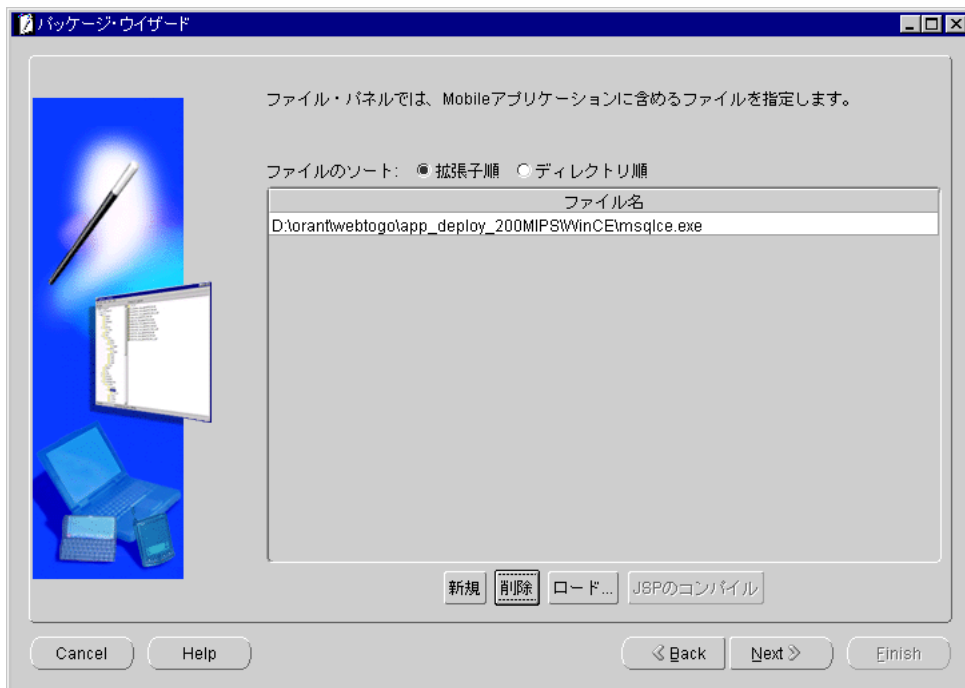
アプリケーション・ファイルのリスト表示

「ファイル」パネルは、アプリケーション・ファイルを表示し、ファイルがローカル・マシン上のどこにあるかを示すために使用します。パッケージ・ウィザードはローカル・アプリケーションのディレクトリの内容を分析し、各ファイルのローカル・パスを表示します。このパネルに含まれるフィールドは次のとおりです。

表 4-3 「ファイル」パネルのオプション

フィールド	説明	必須
ローカルのパス	各 Mobile サーバー・アプリケーション・ファイルの絶対パスです。リスト内の各エントリには、各ファイルまたはディレクトリの完全なパスが含まれています。	<input type="radio"/>
ファイルのソート	<ul style="list-style-type: none">■ 拡張子順 — ファイルを拡張子ごとにアルファベット順に表示します。■ ディレクトリ順 — ファイルをディレクトリごとにアルファベット順に表示します。	

図 4-3 「ファイル」 パネル



「ファイル」パネルにリストされているファイルは、すべて追加、削除またはロードできます。新規アプリケーションを作成する場合、「ファイル」パネルに進むと、ローカル・ディレクトリにリストされているすべてのファイルが自動的に分析されロードされます。既存アプリケーションを編集する場合は、「ロード」ボタンを使用して個々のファイルをロードできます。

ソート

ファイルは、拡張子または含まれているディレクトリごとにソートできます。ファイルをソートするには、「拡張子順」または「ディレクトリ順」ラジオ・ボタンをクリックします。

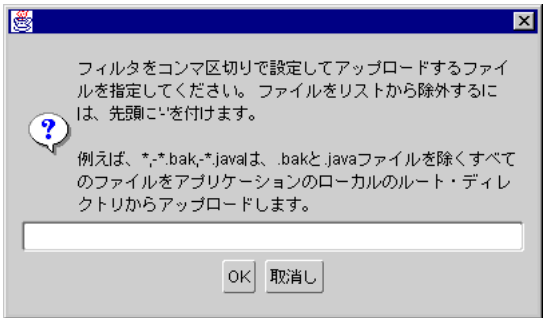
フィルタ

「ロード」ボタンをクリックすると「入力」ダイアログ・ボックスが表示されます。「入力」ダイアログ・ボックスは、アップロード・プロセスからのアプリケーション・ファイルを含めるか除外するかを指定する（カンマで区切られた）フィルタのリストを作成するために使用します。ファイルを除外するには、ファイル名の前に負符号（-）を付けます。たとえば、

.dll および .exe 拡張子の付いたファイルを除くすべてのファイルをロードするには、次のように入力します。

```
*,-*.dll, -*.exe
```

図 4-4 「フィルタ」 パネル



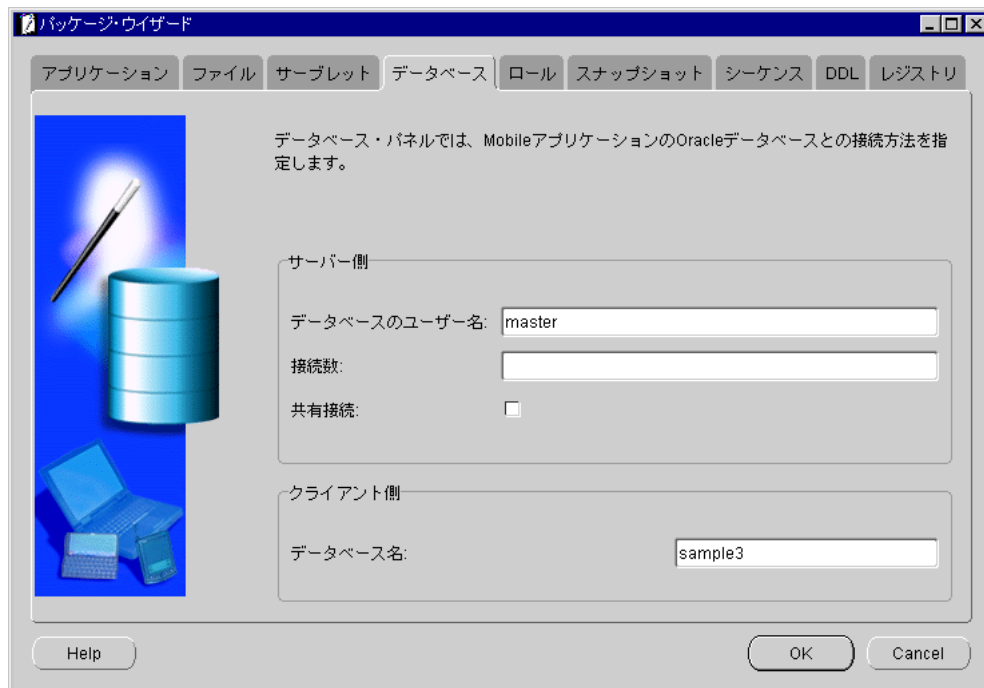
データベース情報の入力

「データベース」 パネルは、Oracle サーバー上のデータベースを指定するために使用します。このパネルに含まれるフィールドとチェックボックスは次のとおりです。

表 4-4 「データベース」 パネルのオプション

フィールド	説明	必須
データベースのユーザー名	データの同期をとるためにアプリケーションによって使用される Mobile サーバー・データベースのユーザー名。デフォルトは「master」です。	<input type="radio"/>
接続数	Web-to-Go アプリケーションから Mobile サーバーに対して持つことのできる同時接続数。	<input checked="" type="checkbox"/>
共有接続	チェックした場合、複数のサブレットで同一のデータベース接続を共有できます。複数のサブレットで接続を共有することにより、Oracle の使用可能な最大同時接続数を超えないようにできます。ただし、1 つのサブレットがコミットを実行すると、その接続を共有する他のすべてのサブレットに自動的に影響します。	<input checked="" type="checkbox"/>
データベース名	Mobile クライアント・デバイス上で接続中のデータベースの名前。アプリケーションで使用される名前である必要があります。	<input type="radio"/>

図 4-5 「データベース」 パネル



レプリケーション用スナップショットの定義

「スナップショット」パネルは、アプリケーションのレプリケーション・スナップショットを作成するために使用します。スナップショットは、全アプリケーション間で一意にする必要があります。これを実現する方法の1つは、スナップショット名の前にアプリケーション名を付けて変更することです。パッケージ・ウィザードでは、次のプラットフォームに対してスナップショットを作成できます。

- Win32
- Palm
- EPOC
- Windows CE

このパネルに含まれるフィールドは次のとおりです。

表 4-5 スナップショット・パラメータ

フィールド	説明	必須
名前	Mobile サーバー・アプリケーションに関連付けられているスナップショット（複数も可）の名前（複数も可）です。この名前は、 Oracle Lite データベース上のデータベース表と同じである必要があります。	○
テンプレート	利用可能なスナップショット・テンプレートのリストです。テンプレートとは、スナップショットの作成に使用される SQL 文です。テンプレートには変数を含められます。テンプレートを Mobile サーバーにパブリッシュした後は、Mobile サーバー・コントロール・センターを使用して、ユーザー固有のテンプレート変数を指定できます。ただし、Mobile サーバー・コントロール・センターではスナップショット・テンプレートは変更できません。	○
プラットフォーム	スナップショットのプラットフォームです。ユーザーは異なるプラットフォームに対してスナップショットを作成できます。クライアントのデータと同期をとった場合、クライアント・アプリケーションを実行中のプラットフォームに適したスナップショットのみが取得されます。	○
比率	これは、データベース表の同期順序を決定する正の整数です。「マスター」「ディテール」関連を持つ表の場合、マスター表はディテール表より先にレプリケートされるように、低い比率を持つ必要があります。詳細は、 第 2 章「同期」の「表の比率の使用」の項 を参照してください。	○

フィールド	説明	必須
全プラットフォーム	<p>現在のスナップショットのプラットフォームのドロップダウン・リストです。ドロップダウン・リストには、次のプラットフォームをすべて含めることができます。</p> <ul style="list-style-type: none">■ Win32■ Palm■ EPOC■ Windows CE <p>ドロップダウン・リストからプラットフォームを選択すると、そのプラットフォーム用のスナップショットのみが「スナップショット」パネルに表示されます。たとえば、「全プラットフォーム」ドロップダウン・リストから「Win32」を選択すると、Win32 ベースのスナップショットのみが表示されます。ドロップダウンから「全プラットフォーム」オプションを選択すると、現在使用中のプラットフォームごとにすべてのスナップショットが表示されます。ユーザーが新しいスナップショットを追加した場合、ドロップダウン・リストには追加のプラットフォームがリスト表示されます。</p>	×

図 4-6 「スナップショット」パネル



「スナップショット」パネルで「新規」ボタンまたは「削除」ボタンをクリックすると、「スナップショット」パネルにスナップショットを追加または削除できます。スナップショットはインポートまたは編集することもできます。

注意：「スナップショット」パネルから複数のスナップショットをインポートできますが、「新規表」ダイアログ・ボックスから新規表を作成するときにインポートできるスナップショットは1つのみです。

新規スナップショットの作成

新規スナップショットを作成するには、「新規」ボタンをクリックします。「新規スナップショット」表が表示されます。

図 4-7 「新規スナップショット」ウィンドウ

The screenshot shows a Windows-style dialog box titled "新規スナップショット" (New Snapshot). It features four tabs: "WIN32", "PALM", "EPOC", and "WINCE". The "名前" (Name) field is populated with "RECORDINGS", and an "インポート..." (Import...) button is located to its right. The "比率" (Ratio) field is set to "1". A checkbox labeled "SQLの生成" (Generate SQL) is present and unchecked. Below this is a large text area for "SQL:". At the bottom of the dialog are "OK" and "取消し" (Cancel) buttons.

「新規スナップショット」画面の次の機能を変更して、新規スナップショットを作成します。

表 4-6 「新規スナップショット」パネルのオプション

機能	説明
名前	サーバー上の新規表の名前です。
プラットフォーム	<p>タブに、スナップショットのデバイス・プラットフォームが表示されます。現在、Mobile サーバーでは次のプラットフォームがサポートされます。</p> <ul style="list-style-type: none"> ■ Win32 ■ Palm ■ EPOC ■ Windows CE
SQL の生成	この機能を選択すると、パッケージ・ウィザードにより、データベース表の作成を可能にする SQL スクリプトへの出力情報が収集されます。
比率	この表に対する表の比率を設定できます。表の比率は、同期時の競合を解決するために使用されます。詳細は、 第2章「同期」 の「 表の比率の使用 」の項を参照してください。
SQL	名前付きの表を定義する SQL 文を表示します。この文は変更できます。

スナップショットのインポート

Oracle8i データベースまたは Oracle Lite データベースからスナップショットをインポートするには、「インポート」ボタンをクリックします。接続を指定していない場合は、データベース接続ウィンドウが表示されます。

注意： 一度指定したデータベース接続は、パッケージ・ウィザードの残りの部分でも使用されます。Oracle8i と Oracle Lite データベースを切り替える必要があるが、すでに接続が確立されている場合は、パッケージ・ウィザード・アプリケーションを完全に終了して、再度 **wtgpack.exe** を実行します。

図 4-8 「データベースへの接続」ウィンドウ



スナップショットのインポート元の Oracle8i データベースのユーザー名、パスワードおよびデータベース URL を入力します。「表」ウィンドウが表示されます。

注意： Oracle8i データベースのデータベース URL を入力するときは、次の書式を使用します。

`jdbc:oracle:thin:@o8host:o8 port:SID`

たとえば、`jdbc:oracle:thin:@o8-db:1521:orcl` と指定します。

Oracle Lite の場合は、`jdbc:polite:webtogo` を使用します。

図 4-9 「表」ウィンドウ

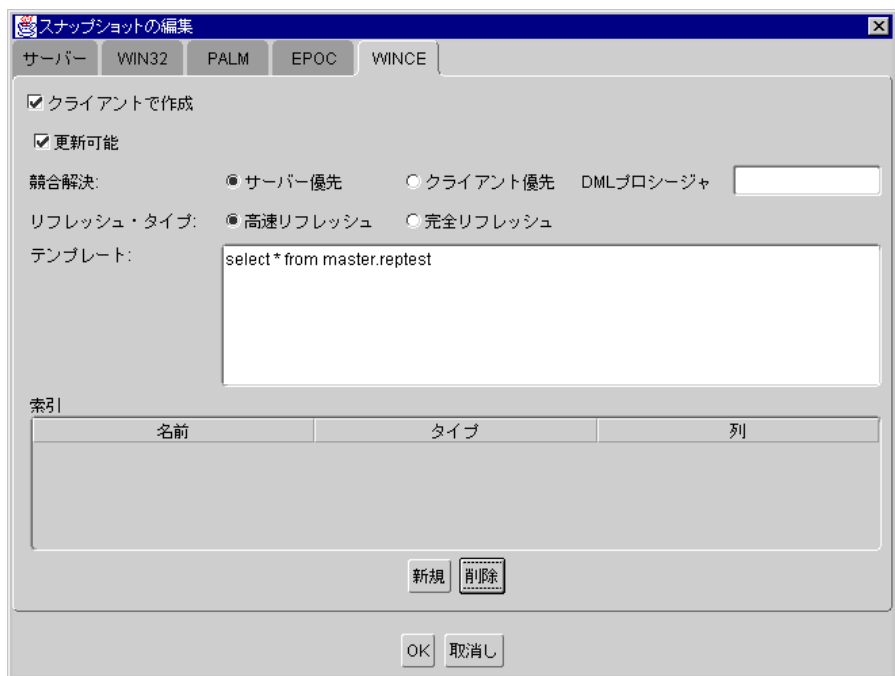


表のインポート元のスキーマを選択してから、表を選択します。「追加」をクリックしてから「閉じる」をクリックします。パッケージ・ウィザードの「スナップショット」パネルに表が表示されます。

スナップショットの編集

スナップショットを編集するには、「スナップショット」パネルからスナップショットを選択し、「編集」をクリックします。「スナップショットの編集」ウィンドウが表示されます。

図 4-10 「スナップショットの編集」ウィンドウ



スナップショットを編集するには、「スナップショットの編集」ウィンドウの次の機能を変更します。

表 4-7 「スナップショットの編集」ウィンドウのオプション

機能	説明
プラットフォーム	<p>タブに、スナップショットのデバイス・プラットフォームが表示されます。</p> <ul style="list-style-type: none"> ■ Win32 ■ Palm ■ EPOC ■ Windows CE

表 4-7 「スナップショットの編集」ウィンドウのオプション（続き）

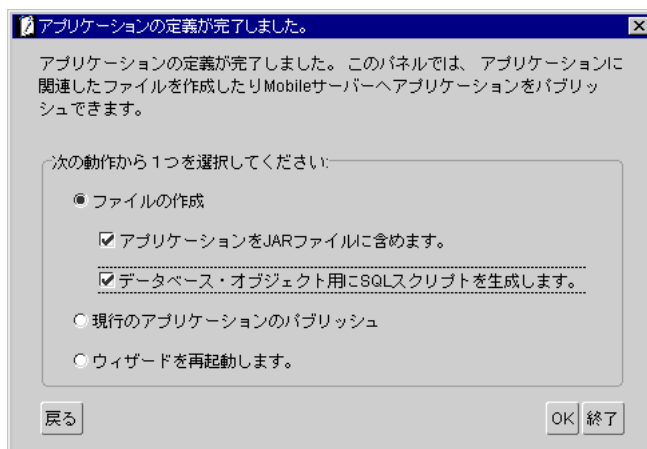
機能	説明
クライアントで作成	このチェックボックスが選択されていると、次のことを実行できます。 <ul style="list-style-type: none">■ 更新可能スナップショットの作成。■ スナップショット・テンプレートの作成。管理者は、Mobile サーバー・コントロール・センターを使用して、複数の異なるユーザー用変数をこのテンプレートに対してインスタンス化できます。
更新可能	このチェックボックスが選択されていると、名前付きの表の更新可能スナップショットが Mobile サーバーの表に作成されます。
テンプレート	名前付きの表のスナップショット・テンプレートを表示します。スナップショット・テンプレートは変更できます。管理者は、 Mobile サーバー・コントロール・センターを使用して、複数の異なるユーザー用変数をこのテンプレートに対してインスタンス化できます。テンプレート変数の詳細は、 「レプリケーション用スナップショットの定義」 の項を参照してください。

アプリケーションの完了

パッケージ・ウィザードの全パネルを完了すると、次のオプションの含まれた「アプリケーションの定義が完了しました。」ウィンドウが表示されます。

- ファイルの作成
- 現行のアプリケーションのパブリッシュ
- ウィザードを再起動します。

図 4-11 「アプリケーションの定義が完了しました。」ウィンドウ



XML ファイル

パッケージ・ウィザードは、アプリケーション情報のすべてを XML ファイルに自動的に出力します。パッケージ・ウィザードはローカル・マシン上で XML ファイルを保持します。さらに、Mobile サーバーに XML ファイルをパブリッシュするオプションも提供しています。XML ファイルを Mobile サーバーにパブリッシュできるのは、サーバーが実行されているときのみです。

JAR ファイルの作成

「ファイルの作成」オプションを使用すると、アプリケーション・コンポーネントを **.jar** ファイルにパッケージ化できます。アプリケーション・コンポーネントを **.jar** ファイルにパッケージ化するには、「ファイルの作成」をクリックしてから「アプリケーションを JAR ファイルに含めます。」をクリックします。**.jar** ファイルの位置を指定するよう要求されます。

SQL ファイルの作成

SQL スクリプトを生成するには、「ファイルの作成」をクリックしてから「データベース・オブジェクト用に SQL スクリプトを生成します」をクリックします。生成されたスクリプトは、アプリケーションのローカル・ルート・ディレクトリの下に SQL サブディレクトリ内に入れます。SQL スクリプトは、サーバー側の表、シーケンスおよび DDL に関して指定した情報を使用します。この SQL スクリプトをデータベースに対して実行して、これらのデータベース・オブジェクトを作成できます。

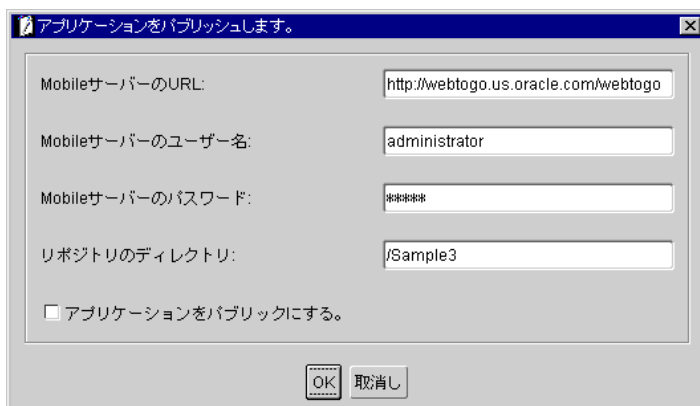
パッケージ・ウィザードの再起動

「ウィザードを再起動します。」オプションを使用すると、パッケージ・ウィザードを再起動できます。このオプションを使用すると、パッケージ・ウィザードの「ようこそ」パネルに戻ります。パッケージ・ウィザードを再起動するには、「ウィザードを再起動します」をクリックしてから「OK」をクリックします。

アプリケーションのパブリッシュ

「現行のアプリケーションのパブリッシュ」オプションを使用すると、パッケージ・ウィザードで作成し定義したアプリケーションをパブリッシュできます。Mobile サーバー・アプリケーションをパブリッシュするには、「現行のアプリケーションのパブリッシュ」チェックボックスを選択してから「OK」をクリックします。「アプリケーションをパブリッシュします。」ウィンドウが表示されます。

図 4-12 「アプリケーションをパブリッシュします。」ウィンドウ



「アプリケーションをパブリッシュします。」ウィンドウの各フィールドに必要な情報を入力します。

表 4-8 「アプリケーションをパブリッシュします。」ウィンドウのオプション

フィールド	説明	必須
Mobile サーバーの URL	サーバー名とポート番号を含む、Mobile サーバーの URL です。サーバー名とポート番号は、次の書式で指定します。 <code>http://mobileserver/webtogo/:port</code> 。ポート 80 はデフォルト・ポートです。	<input type="radio"/>
Mobile サーバーのユーザー名	Mobile サーバー・ユーザーの名前です。	<input type="radio"/>
Mobile サーバーのパスワード	Mobile サーバー・ユーザーのパスワードです。	<input type="radio"/>
リポジトリのディレクトリ	Mobile サーバー上の Mobile サーバー・リポジトリの場所です。パッケージ・ウィザードはアプリケーション・ファイルをこのディレクトリにパブリッシュし、ローカル・アプリケーションのディレクトリ上でディレクトリ構造をメンテナンスします。	<input type="radio"/>
アプリケーションをパブリックにする。	このアプリケーションをパブリック・アプリケーションとしてパブリッシュするには、これを選択します。パブリック・アプリケーションに対しては、すべてのユーザーがアクセス権を持ちます。	<input type="radio"/>

注意： アプリケーションを Mobile サーバーにパブリッシュするには、publish 権限が必要です。Mobile サーバー管理者は Mobile サーバー・コントロール・センターを使用して権限を割り当てます。

アプリケーションの編集

パッケージ・ウィザードを起動して「既存のアプリケーションを編集」を選択すると、アプリケーションを編集できます。Mobile サーバーの DTD ファイルに準拠する XML 文書を作成または変更すると、アプリケーションを手動で作成または編集できますが、アプリケーションの作成または変更ではパッケージ・ウィザードの使用が最善の方法です。

Mobile サーバーのシステム・カタログ・ビュー

この章は、Mobile Admin スキーマの一部である Mobile サーバーのシステム・カタログ・ビューの参考資料です。Mobile Admin スキーマは Mobile サーバーの一部としてインストールされます。Mobile Admin スキーマは Mobile Development Kit の一部ではありません。

システム・カタログ・ビューは読み取り専用なので、変更できません。ビューには次のものが含まれています。

- ["MOBILEADMIN".CV\\$ALL_CLIENTS](#)
- ["MOBILEADMIN".CV\\$ALL_PUBLICATIONS](#)
- ["MOBILEADMIN".CV\\$ALL_SUBSCRIPTIONS](#)
- ["MOBILEADMIN".CV\\$ALL_SEQUENCES](#)
- ["MOBILEADMIN".CV\\$ALL_SEQUENCE_PARTITIONS](#)
- ["MOBILEADMIN".CV\\$ALL_PUBLICATION_ITEMS_ADDED](#)
- ["MOBILEADMIN".CV\\$ALL_PUBLICATION_ITEMS](#)
- ["MOBILEADMIN".CV\\$ALL_PUBLICATION_INDEXES](#)
- ["MOBILEADMIN".CV\\$ALL_SUBSCRIPTION_PARAMS](#)

"MOBILEADMIN".CV\$ALL_CLIENTS

このビューには、Mobile サーバーのクライアントに関する情報が表示されます。列の内容は次のとおりです。

表 A-1 ALL_CLIENTS のパラメータ

列	データ型	NULL	説明
CLIENT	VARCHAR2(30)		Mobile サーバーのクライアント。
LASTREFRESH_STARTTIME	VARCHAR2(19)		最終リフレッシュの開始時間。
LASTREFRESH_ENDTIME	VARCHAR2(19)		最終リフレッシュの終了時間。

"MOBILEADMIN".CV\$ALL_PUBLICATIONS

このビューには、Mobile サーバーのパブリケーションに関する情報が表示されます。列の内容は次のとおりです。

表 A-2 ALL_PUBLICATIONS のパラメータ

列	データ型	NULL	説明
NAME	VARCHAR2(30)		パブリケーション名。
TYPE	VARCHAR2(40)		パブリケーション・タイプ。
NAME_TEMPLATE	VARCHAR2(30)		スナップショット名テンプレート。
ENFORCE_RI	CHAR(1)	NOT NULL	予約済み。

"MOBILEADMIN".CV\$ALL_SUBSCRIPTIONS

このビューには、Mobile サーバーのサブスクリプションに関する情報が表示されます。列の内容は次のとおりです。

表 A-3 ALL_SUBSCRIPTIONS のパラメータ

列	データ型	NULL	説明
CLIENT	VARCHAR2(30)		サブスクリプションのクライアント。
PUBLICATION	VARCHAR2(30)		サブスクリプションのパブリケーション。
INSTANTIATED	CHAR(1)		サブスクリプションがインスタンス化されているかどうかを示すブール値。

"MOBILEADMIN".CV\$ALL_SEQUENCES

このビューには、Mobile サーバーのシーケンスに関する情報が表示されます。列の内容は次のとおりです。

表 A-4 ALL_SEQUENCES のパラメータ

列	データ型	NULL	説明
NAME	VARCHAR2(30)		シーケンス名。

"MOBILEADMIN".CV\$ALL_SEQUENCE_PARTITIONS

このビューには、Mobile サーバーのシーケンス・パーティションに関する情報が表示されます。列の内容は次のとおりです。

表 A-5 ALL_SEQUENCE_PARTITIONS のパラメータ

列	データ型	NULL	説明
CLIENT	VARCHAR2(30)		シーケンスを割り当てるクライアント。
NAME	VARCHAR2(30)		シーケンス名。
CURR_VALUE	NUMBER(38)		現在のシーケンス値。
INCREMENT_BY	NUMBER(38)		シーケンスの増分値。シーケンスはこの数値で増加します。

"MOBILEADMIN".CV\$ALL_PUBLICATION_ITEMS_ADDED

このビューには、Mobile サーバーのパブリケーションに追加されたパブリケーション項目に関する情報が表示されます。列の内容は次のとおりです。

表 A-6 ALL_PUBLICATION_ITEMS_ADDED のパラメータ

列	データ型	NULL	説明
PUB_NAME	VARCHAR2(30)		パブリケーション名。
ITEM_NAME	VARCHAR2(30)		パブリケーション項目名。
OWNER	VARCHAR2(30)	NOT NULL	ベース・オブジェクトの所有者。
OBJECT_NAME	VARCHAR2(30)	NOT NULL	ベース・オブジェクト名。
TEXT	VARCHAR2(2048)	NOT NULL	選択文。
UPDATABLE	VARCHAR2(1)		更新可能オプション。

表 A-6 ALL_PUBLICATION_ITEMS_ADDED のパラメータ (続き)

列	データ型	NULL	説明
REFRESH_METHOD	CHAR(1)	NOT NULL	リフレッシュ方法。オプションには、高速リフレッシュと完全リフレッシュがあります。
WINNING_RULE	VARCHAR2(30)		レプリケーションの競合を解決するためのウィニング・ルール・オプション。オプションには、「クライアント優先」と「サーバー優先」があります。

"MOBILEADMIN".CV\$ALL_PUBLICATION_ITEMS

このビューには、Mobile サーバーのパブリケーション項目に関する情報が表示されます。列の内容は次のとおりです。

表 A-7 ALL_PUBLICATION_ITEMS のパラメータ

列	データ型	NULL	説明
NAME	VARCHAR2(30)		パブリケーション項目名。
OWNER	VARCHAR2(30)	NOT NULL	パブリケーション項目のベース・オブジェクトの所有者。
OBJECT_NAME	VARCHAR2(30)	NOT NULL	ベース・オブジェクト名。
TEXT	VARCHAR2(2048)	NOT NULL	選択文。
REFRESH_METHOD	CHAR(1)	NOT NULL	リフレッシュ方法。オプションには、高速リフレッシュと完全リフレッシュがあります。

"MOBILEADMIN".CV\$ALL_PUBLICATION_INDEXES

このビューには、Mobile サーバーのパブリケーション索引に関する情報が表示されます。列の内容は次のとおりです。

表 A-8 ALL_PUBLICATION_INDEXES のパラメータ

列	データ型	NULL	説明
NAME	VARCHAR2(30)		索引名。
PUB_ITEM	VARCHAR2(30)	NOT NULL	パブリケーション項目名。

表 A-8 ALL_PUBLICATION_INDEXES のパラメータ (続き)

列	データ型	NULL	説明
INDX_TYPE	CHAR(1)	NOT NULL	索引タイプ。
COLUMN_LIST	VARCHAR2(2048)	NOT NULL	列リスト。

"MOBILEADMIN".CV\$ALL_SUBSCRIPTION_PARAMS

このビューには、Mobile サーバーのサブスクリプション・パラメータに関する情報が表示されます。列の内容は次のとおりです。

表 A-9 ALL_SUBSCRIPTIONS_PARAMS のパラメータ

列	データ型	NULL	説明
NAME	VARCHAR2(30)		パブリケーション名。
CLIENT	VARCHAR2(30)		クライアント。
PARAM_NAME	VARCHAR2(30)		パラメータ名。
PARAM_VALUE	VARCHAR2(30)		パラメータ値。

Oracle Lite ユーティリティ

この付録では、次に示した Oracle Lite データベースのユーティリティの使用方法を説明します。ユーティリティの名前はアルファベット順に並んでいます。

表 B-1 ツールとユーティリティ

ユーティリティ	説明
CREATEDB	Oracle Lite データベースの作成に使用します。
DECRYPDB	Oracle Lite データベースの復号化に使用します。
ENCRYPDB	Oracle Lite データベースの暗号化に使用します。
Mobile SQL	Mobile SQL は、Oracle Lite データベースに接続するための GUI インタフェースです。
ODBINFO	このユーティリティは、Oracle Lite データベースのバージョン・ナンバーとボリューム ID を見つけるために使用します。
REMOVEDB	Oracle Lite データベースの削除に使用します。

CREATEDB

説明

データベースを作成するためのユーティリティです。

構文

```
CREATEDB DataSourceName DatabaseName [VolID]
```

キーワードとパラメータ

DataSourceName

データ・ソース名で、デフォルトのデータベース・ディレクトリの **ODBC.INI** ファイルを検索するために使用します。

注意： 無効な DSN を指定すると、Oracle Lite はその DSN を無視し、カレント・ディレクトリにデータベースを作成します。このデータベースを ODBC 経由でアクセスするには、データベースが存在するディレクトリを指す DSN を作成する必要があります。

DatabaseName

作成するデータベース名です。フルパス名またはデータベース名のみを指定できます。データベース名のみを指定すると、データベースは、**ODBC.INI** ファイルで指定されたデータ・ソース名のデータ・ディレクトリの下に作成されます。データベース名の拡張子は、常に **.ODB** です。指定した名前に **.ODB** が付いていない場合は、**.ODB** が付加されます。

VolID

VolID を指定すると、**POLITE.INI** ファイルに指定されているデータベース ID のかわりに VolID がデータベース ID として使用されます。ID はデータベースごとに一意にする必要があります。

例

```
createdb polite db1
createdb polite c:\testdir\db2.odb 300
```

DECRYPDB

説明

このツールを使用すると、暗号化された Oracle Lite データベースを復号化できます。詳細は、「[ENCRYPDB](#)」を参照してください。

構文

```
DECRYPDB DSN | NONE DBName [Password
```

キーワードとパラメータ

DSN

復号化する Oracle Lite データベースのデータ・ソース名です。NONE を指定すると、DBName には（.ODB 拡張子を除く）フルパス名で入力する必要があります。

DBName

復号化するデータベース名です。DSN に NONE が指定されていると、DBName はフルパス名で入力する必要があります。

Password

オプションです。Oracle Lite データベースの暗号化に使用されたパスワードです。パスワードを指定しないと、DECRYPDB により入力を要求するプロンプトが表示されます。

コメント

Oracle Lite データベースに対してオープンされている接続がある場合は、データベースを復号化できません。

このユーティリティを別のアプリケーションからコールすると、結果は次のようになります。

表 B-2 DECRYPDB のリターン・コード

リターン・コード	説明
EXIT_SUCCESS	成功。
EXIT_USAGE	コマンドライン引数の使用方法が適切でないか、引数にエラーがあります。
EXIT_PATH_TOO_LONG	パスが長すぎます。
EXIT_SYSCALL	復号化された新規コピーをディスク上に作成中に I/O エラーが発生しました。
EXIT_BAD_PASSWD	不適切なパスワードが指定されました。

詳細は、「[ENCRYPDB](#)」のコメントを参照してください。

ENCRYPDB

説明

このツールにより、パスワードを使用して Oracle Lite データベースを暗号化したり、データベースのパスワードを変更できます。パスワードを使用すると、データベースへの許可されていないアクセスを防止し、データベースを暗号化できるので、データベース・ファイル内に格納されているデータを解釈できないようにすることが可能です。詳細は、[「DECrypDB」](#)を参照してください。

構文

```
ENCRYPDB DSN | NONE DBName [New_Passwdor [Old_Passwdor]]
```

キーワードとパラメータ

DSN

暗号化する Oracle Lite データベースのデータ・ソース名です。NONE を指定すると、DBName には（.ODB 拡張子を除く）完全修飾されたデータベース名を入力する必要があります。DSN に NONE 以外の値を指定する場合、ODBC.INI ファイル内のデータ・ソース名を指定する必要があります。

DBName

暗号化するデータベース名です。DSN に NONE が指定されていると、DBName はフルパス名で入力する必要があります。

New_Passwd と Old_Passwd

オプションです。データベースを暗号化する（または暗号化のために以前に使用された）パスワードです。このパスワードの長さは最大 128 文字です。パスワードを指定しないと、ENCRYPDB により入力を要求するプロンプトが表示されます。どちらのパスワードもオプションであるため、ユーティリティを起動するときのコマンドラインは次の 3 つの書式になります。

- パスワードなし。データベースがすでに暗号化されている場合は、ENCRYPDB はユーザーがデータベースのパスワードを変更しようとしていると解釈します。古いパスワードの入力を 1 回、新しいパスワードの入力を 2 回要求され、新しいパスワードを使用してデータベースが暗号化されます。データベースが暗号化されていない場合、ENCRYPDB によって、新しいパスワードの入力が 2 回要求され、新しいパスワードを使用してデータベースが暗号化されます。
- 片方のパスワードを指定。このパスワードは新しいパスワードとして解釈されます。データベースがすでに暗号化されている場合、ENCRYPDB によって、古いパスワードの入力が要求され、新しいパスワードを使用してデータベースが暗号化されます。
- 両方のパスワードを指定。ENCRYPDB では、最初のパスワードが新しいパスワードで、2 番目のパスワードが古いパスワードであると解釈します。

コメント

このユーティリティを別のアプリケーションからコールすると、結果は次のようになります。

表 B-3 ENCRYPDB のリターン・コード

リターン・コード	説明
EXIT_SUCCESS	成功。
EXIT_USAGE	コマンドライン引数の使用方法が適切でないか、引数にエラーがあります。
EXIT_PATH_TOO_LONG	パスが長すぎます。
EXIT_SYSCALL	暗号化された新規コピーをディスク上に作成中に I/O エラーが発生しました。
EXIT_BAD_PASSWD	不適切なパスワードが指定されました。

デフォルトの Oracle Lite データベース (**POLITE.ODB**) は暗号化されていません。Oracle Lite データベースを暗号化すると、暗号化した Oracle Lite データベースに対して接続を確立しようとするユーザーはすべて、有効なパスワードを提供する必要があります。パスワードが提供されない場合、Oracle Lite はエラーを返します。Oracle Lite データベースに対してオープンされている接続がある場合は、データベースを暗号化できません。

Oracle Lite データベースを暗号化および復号化する場合、次の点を考慮する必要があります。

- 暗号化されたデータベースは、パスワードなしでは復号化できません。データベースは暗号化する前に、必ず安全な場所にバックアップを作成します。
- 他の Oracle Lite アプリケーションを実行中は、ENCRYPDB を実行しません。データベース・ファイルに他のアプリケーションが接続している場合、エラーが表示されます。
- データベースを暗号化した後、そのデータベースに接続するには、接続文字列にパスワードを含める必要があります。
- パスワードによってデータベース全体が暗号化されます。ユーザー固有のパスワードではありません。
- データベースを暗号化しても、サード・パーティが Oracle Lite データベースを削除するのを防ぐことはできません。つまり、removedb と rmdb は、パスワードをチェックせずにデータベースを削除します。許可されていないユーザーがファイル・システムを操作できないように保護するツールを使用します。
- 暗号化された Oracle Lite データベースに接続する ODBC アプリケーションは、有効なパスワードを指定する必要があります。パスワードは通常、アプリケーション内でコーディングされずに、実行時に入力が必要されます。ほとんどの ODBC アプリケーションでは、Oracle Lite の ODBC ドライバが実行時にパスワードの入力を要求する場合、

SQLConnect 関数ではなく、SQLDriverConnect 関数で DRIVER= オプションを指定できます。

- Oracle Lite のこのリリースに付属しているサンプル・アプリケーションはすべて、暗号化されていないデータベースに対して実行できます。
- DECRYPTDB と ENCRYPTDB を使用して（この順番で）、データベースのパスワードを変更できます。ただし、DECRYPTDB によって Oracle Lite データベースが最初にプレーン・テキストで作成され、次に ENCRYPTDB によって暗号化されます。プレーン・テキスト形式のデータベースが一時的に作成されるため、この方法はお薦めできません。
- 暗号化されたデータベースの場合、ユーザー名とパスワードはすべて **DSN.OPW** というファイルに書き込まれます。その後、各ユーザーは **.ODB** ファイルをアクセスする前に、パスワードを鍵として使用して **.OPW** ファイルのロックを解除できます。データベースをコピーまたはバックアップするときは、**.OPW** ファイルを含める必要があります。

Mobile SQL

Mobile SQL はアプリケーションで、ユーザーはこれを使用してローカル・データベースに対して SQL 文を実行できます。Mobile SQL は開発者用のツールで、コード例も含まれています。Mobile SQL を使用すると、基になる Oracle Lite データベース・エンジンの ODBC インタフェースと Oracle Lite OKAPI インタフェースにより提供されている関数にアクセスできます。

データベース・アクセス

Mobile SQL は、ODBC および OKAPI の両インタフェースを介してデータベースにアクセスします。関数のほとんどは ODBC を介して実行されますが、ODBC が処理できない関数は、OKAPI ファンクション・コールを使用して実装されています。

Mobile SQL の起動

Mobile SQL を起動するには、`Oracle_Home¥Mobile¥SDK¥wince` をオープンし、Windows CE バージョンを表すフォルダを選択し、次に使用するデバイスのプロセッサを選択します。**mSQL.exe** ファイルをダブルクリックします。これで、標準 SQL コマンドを受け入れる GUI インタフェースが起動されます。詳細は、『Oracle9i Lite SQL リファレンス』を参照してください。

ODBINFO

説明

ODBINFO は、Oracle Lite データベースのバージョン・ナンバーとボリューム ID を見つけるために使用できます。また、ODBINFO はパラメータをいくつか表示し設定することもできます。

構文

変更を加えずに現在の情報を表示するには、次の構文を使用します。

```
odbinfo [-p passwd] DSN DBName
```

また、次の構文も使用できます。

```
odbinfo [-p passwd] NONE dbpath%dbname.odb
```

たとえば、次のようになります。

```
odbinfo -p tiger polite polite
odbinfo NONE c:%orant%oldb40%polite.odb
```

データベースを暗号化してある場合は、パスワードを含める必要があります。

オプションを設定または消去するには、DSN または NONE の前に「+」または「-」引数を 1 つ以上使用します。たとえば、次のようになります。

```
odbinfo +reuseoid -pagelog -fsync polite polite
```

パラメータ

ODBINFO では次のパラメータを使用できます。

表 B-4 ODBInfo のパラメータ

パラメータ	説明
pagelog	デフォルトでは、実際に変更が filename.ODB に書き込まれる前に、コミットによって変更済みデータベース・ページのバックアップが filename.plg に作成されます。コミット中にアプリケーションまたはオペレーティング・システムに障害が発生した場合、トランザクションは次の接続時にクリーンな状態にロールバックされます。-pagelog を指定した場合、バックアップは作成されないため、障害が発生するとデータベースが破損する可能性があります。

表 B-4 ODBInfo のパラメータ（続き）

パラメータ	説明
fsync	<p>Oracle Lite は、通常、データベースに関連付けられている変更済みバッファをコミット時にすべてディスクに書き込むように、オペレーティング・システムに対して強制します。このオプションが使用禁止になっていると（-fsync）、オペレーティング・システムは変更を後までメモリー内に保持しておくことができます。バッファがフラッシュされる前に（アプリケーションではなく）システムがクラッシュした場合、データベースも破損する可能性があります。</p> <p>odbinfo -fsync -pagelog を使用すると、多数の（自動コミットがオンに指定された）小規模なトランザクションや大規模な更新を伴うトランザクションを使用するアプリケーションのパフォーマンスが向上します。ただしデータベースが破損した場合、簡単にデータベースを修復したりデータをリカバリする方法はありません。このため、この 2 つのオプションは、(1) .ODB が定期的にバックアップされる場合、または (2) データベース内のデータが他のソースからリカバリできる場合にのみ、データベースの初期ロード中に消去するようにします。</p> <p>このオプションを使用しても、データベースをあまり更新しないアプリケーションには何の影響もありません。この場合は、トランザクション分離レベルを SINGLE USER に設定する方が効果があります。</p>
reuseoid	<p>Oracle Lite は、デフォルトでは、表にあるどの行の ROWID も表が削除されるまで再利用しません。削除されたオブジェクトにアクセスしようとする、「スロットが削除されました」エラーが返されます。この場合、削除されたオブジェクトごとに 2 バイトのストレージが使用されるため、行が頻繁に挿入および削除される場合は、時間が経つとパフォーマンスが低下し使用ディスク領域が増えます。</p> <p>odbinfo +reuseoid を使用すると、削除された行の ROWID を新しい行で再利用できます。ただし、これでは削除されたオブジェクトが多数ある表の領域をすべて解放できないことがあります。最善の方法は、データベースの作成直後にこのオプションを設定することです。</p> <p>このオプションは、純粋にリレーショナルなアプリケーションの場合は安全です。ただし、ROWID を使用する SQL アプリケーションやオブジェクト間で直接ポインタを使用する JAC/OKAPI アプリケーションの場合は、オブジェクトが削除される前に、そのオブジェクトに対する参照がすべて NULL に設定されることを確認する必要があります。そうでないと、参照先のない参照が別の無関係のオブジェクトを指してしまう場合があります。</p>

表 B-4 ODBInfo のパラメータ（続き）

パラメータ	説明
compress	<p>このオプション（デフォルトは「on」）は、オブジェクトの実行長の圧縮を可能にします。実行長の圧縮には CPU 時間がほとんどかからないため、このオプションの選択を解除する（-compress）のは、次の場合のみです。</p> <ul style="list-style-type: none">■ オペレーティング・システム・レベルのファイル圧縮（ドライブスペースや NTFS 圧縮属性など）が使用される場合。この場合、同一データを 2 回圧縮しないので圧縮率が上がります。■ データベース内のほとんどのオブジェクトが高度に圧縮可能な状態（たとえば列がすべて NULL に設定されるなど）に頻繁に更新され、（ランダム・データを指定したバイナリ列のように）データがうまく圧縮できない場合。このような場合は、このオプション（+compress）を指定すると、表が非常に断片化される可能性があります。 <p>このオプションを変更しても、データベース内の既存のオブジェクトは圧縮も圧縮解除もされません。</p>

REMOVEDB

説明

データベースを削除するためのユーティリティです。

構文

REMOVEDB DataSourceName DatabaseName

キーワードとパラメータ

DataSourceName

削除するデータベースのデータ・ソース名です。

DatabaseName

削除するデータベースの名前です。フルパス名またはデータベース名のみを指定できます。データベース名のみを指定すると、データベースは、ODBC.INI ファイルで指定されたデータ・ソース名のデータ・ディレクトリから削除されます。

コメント

REMOVEDB で削除されたデータベースは、引き続き Oracle8i Navigator に表示されます。Oracle8i Navigator に表示される削除されたデータベースに接続しようとすると、かわりにデフォルトのデータベースに接続されます。Oracle8i Navigator を使用してデータベースを削除することをお薦めします。

例

```
removedb polite db1  
removedb polite c:¥testdir¥db2.odp
```

用語集

3 層 Web モデル (Three-Tier Web Model)

クライアント、中間層およびサーバーを含むインターネット・データベース構成。
Web-to-Go アーキテクチャは 3 層 Web モデルに準拠しています。

Apache Server

National Center for Supercomputing Applications (NCSA) から発表されたパブリック・ドメインの HTTP サーバー。

Java Servlet Development Kit

Java サーブレットの開発のために JavaSoft 社が提供しているツール。

Java Web Server Development Kit

Java Web Server Development Kit 1.0.1 は、JavaServer Pages (JSP) と Java サーブレットの開発のために JavaSoft 社が提供しているツールです。

JavaServer Pages

JavaServer Pages (JSP) とは、開発者がページの基になるコンテンツを変更せずにページのレイアウトを変更できるようにするテクノロジーです。JSP は HTML と Java コードを使用し、動的コンテンツとビジネス・ロジックを結び付けたプレゼンテーションを可能にします。

Java アプレット (Java Applets)

ブラウザで実行される小規模なアプリケーションで、動的コンテンツを追加することにより HTML ページの機能を拡張します。

Java サーブレット (Java Servlets)

Java で作成されているプロトコルで、プラットフォームに依存しないサーバー側コンポーネント。Java サーブレットは Java 対応のサーバーを動的に拡張し、要求 - 応答方式を使用して作成されたサービスのための汎用フレームワークを提供します。

JDBC

Java Database Connectivity (JDBC) は Java クラスの標準セットで、リレーショナル・データに対してベンダーに依存しないアクセスを提供します。JDBC クラスは ODBC をモデルにしたもので、複数データベースへの同時接続、トランザクション管理、単純問合せ、バインド変数によるコンパイル済文の操作、ストアド・プロシージャへのコールなどの標準機能を提供します。JDBC では、静的 SQL と動的 SQL の両方がサポートされます。

LEAPFROG シーケンス (Leapfrog Sequence)

Web-to-Go がサポートする 2 つのシーケンスのうちの 1 つで、オフライン・モードの Web-to-Go の Mobile クライアントに対して一意の主キー値を提供するために使用されるもの。LEAPFROG シーケンスの初期値はクライアントごとに異なり、各シーケンスの増分は最大クライアント数より大きい値に設定されます。

MIME

Multipurpose Internet Mail Extensions (MIME) とは、メッセージの内容を記述するためにインターネット上で使用されるメッセージ形式です。MIME は、HTTP サーバーが配布対象ファイルのタイプを記述するために使用します。

MIME タイプ (MIME Type)

Multipurpose Internet Mail Extensions (MIME) により定義されているファイル形式。

Mobile Development Kit for Web-to-Go

Mobile Development Kit (Web-to-Go 用) を使用すると、アプリケーション開発者は、Java サブレット、JavaServer Pages (JSP) または Java アプレットで構成される Web-to-Go アプリケーションの開発とデバッグを行えます。

Mobile サーバー

Mobile サーバーは、Web-to-Go3 層モデルのアプリケーション・サーバー層に常駐し、Web-to-Go の Mobile クライアントからのデータ変更要求を処理してデータベース・サーバー内のデータを変更します。Mobile サーバーは、Oracle HTTP Server、Apache Server およびスタンドアロンの Mobile サーバーとともに実行されるように構成できます。

Mobile サーバー・リポジトリ (Mobile Server Repository)

Mobile サーバー・リポジトリとは、Oracle8i に常駐する仮想ファイル・システムです。このリポジトリは、すべてのアプリケーション・ファイルとアプリケーション定義を含む持続リソース・リポジトリです。

ODBC

Open Database Connectivity (ODBC) は Microsoft 社の標準で、様々なプラットフォーム上のデータベース・アクセスを可能にします。Web-to-Go の Mobile クライアント上では、トラブルシューティング用に ODBC サポートを使用可能にします。ODBC サポートを使用すると、ローカルな Oracle Lite データベースに格納されているクライアントのデータを表示できます。この情報を表示するには、SQL*Plus を使用します。

Oracle8i

Oracle8i は、Mobile サーバーのデータベース・コンポーネントです。Web-to-Go の Mobile クライアントがオンライン・モードのときは、アプリケーションとデータは Oracle8i に格納されます。

Oracle Lite

Oracle Lite は、Web-to-Go の Mobile クライアントのデータベース・コンポーネントです。クライアントがオフライン・モードのときは、アプリケーションとデータは Oracle Lite に格納されます。

SQL

Structured Query Language (SQL) は、リレーショナル・データベース・エンジンのほとんどで使用する非手続き型データベース・アクセス言語です。SQL 文はデータ・セットに対して実行される操作を記述します。SQL 文がデータベースに送られると、データベース・エンジンは指定されたタスクを実行するプロシージャを自動的に生成します。

Web-to-Go

Oracle Web-to-Go は、Web ベースのモバイル・データベース・アプリケーションを作成および配置するためのフレームワークです。Web-to-Go には、Web-to-Go の Mobile クライアント、Mobile サーバーおよび Oracle8i で構成される 3 層データベース・アーキテクチャが含まれます。サーバーから一元管理され、Web-to-Go アプリケーションは Web-to-Go がサーバーに接続されたとき（オンライン）またはサーバーから切断されたとき（オフライン）に実行できます。オフラインのときは Web-to-Go はデータをローカルにキャッシュし、オンラインに戻ったときにそのデータをサーバーと同期します。

Web-to-Go の Mobile クライアント

Web-to-Go の Mobile クライアントは、Web-to-Go の 3 層 Web モデルのクライアント層です。この層には、Mobile サーバーと Oracle Lite データベースが含まれます。Web-to-Go は、ユーザーがオフライン・モードに切り替えたときにユーザー・アプリケーションとデータを Oracle Lite にレプリケートします。ユーザーが元のオンライン・モードに切り替えると、Web-to-Go はすべてのデータ変更を Oracle8i にレプリケートします。

WINDOW シーケンス (Window Sequence)

Web-to-Go がサポートする 2 つのシーケンスのうちの 1 つで、オフライン・モードの Web-to-Go の Mobile クライアントに対して一意の主キー値を提供するために使用されるもの。WINDOW シーケンスには、一意の値範囲が含まれます。他のクライアントと値の範囲は重複しません。クライアントがシーケンスの範囲内の値をすべて使用すると、Web-to-Go は新しい一意の値範囲を持つシーケンスを再び作成します。

一意キー (Unique key)

表の一意キーは、表の各列での一意の列または列グループです。UNIQUE KEY 制約を満たすには、一意キーの値が表の複数の行に出現することはできません。ただし、PRIMARY KEY 制約とは異なり、単一列からなる一意キーは NULL 値を含むことができます。

位置づけ DELETE (Positioned DELETE)

位置づけ DELETE 文により、カーソルの現在行が削除されます。書式は次のとおりです。

```
DELETE FROM table
      WHERE CURRENT OF cursor_name
```

位置づけ UPDATE (Positioned UPDATE)

位置づけ UPDATE 文により、カーソルの現在行が更新されます。書式は次のとおりです。

```
UPDATE table SET set_list
      WHERE CURRENT OF cursor_name
```

オフライン・モード (Offline Mode)

Web-to-Go の Mobile クライアントが Mobile サーバーから切断されている状態。オフライン・モードでは、クライアント・アプリケーションはローカルに実行され、データは Oracle Lite でアクセスおよび格納されます。「[オンライン・モード](#)」も参照。

オンライン・モード (Online Mode)

Web-to-Go の Mobile クライアントが Mobile サーバーに接続されている状態。「[オフライン・モード](#)」も参照。

外部キー (Foreign Key)

外部キーとは表またはビューに存在する列または列グループのことで、その値は別の表またはビューに存在する行を参照します。外部キーには、一般に、別の表の主キー値と一致する値が含まれます。「[主キー](#)」も参照。

結合 (Join)

2 つの異なる表またはビューに存在するキー（主キーと外部キーの両方）の間に確立された関係。結合は、リレーショナル・データベース内の重複したデータを排除するために正規化された表のリンクに使用します。結合リンクの一般的なものとしては、1 つの表の主キーを別の表の外部キーにリンクして、マスター・ディテール関連を確立するものがあります。結合は SQL 文の WHERE 句条件に対応します。

コントロール・センター (Control Center)

Mobile サーバー・コントロール・センターはブラウザ内で実行される Web ベースのアプリケーションで、これを使用すると Web-to-Go アプリケーションとそのユーザーの管理が容易になります。管理者はコントロール・センターを使用して、ユーザーまたはグループに対するアクセス権の付与と取消し、スナップショット・テンプレート変数の変更、Web-to-Go からのアプリケーションの削除などの機能を実行します。

サイト (Site)

Web-to-Go は、Web-to-Go の Mobile クライアント上の各ユーザーに対してデータベースを作成します。このデータベースはサイトと呼ばれます。1 つのクライアントに複数のサイトを

含められますが、サイトは1人のユーザーに1つのみ可能です。ユーザーは、異なるクライアント上に複数のサイトを所有できます。

索引 (Index)

表内のそれぞれの行に対する高速アクセスを提供するデータベース・オブジェクト。索引を作成すると、表のデータに対して実行される問合せおよびソート操作を高速化できます。また、索引を使用して、UNIQUE KEY 制約やPRIMARY KEY 制約などの制約を表に対して規定することもできます。

索引はいったん作成されると自動的にメンテナンスされ、データベース・エンジンにより可能なかぎりデータ・アクセスのために使用されます。

参照整合性 (Referential Integrity)

参照整合性は、レコードが追加、修正または削除されたときにメンテナンスされるマスター・ディテール関連内の表間のリンクの精度として定義されます。

マスター・ディテール・リレーションを注意深く定義しておくことにより、参照整合性が高まります。データベース内の制約によって、データベース（クライアント / サーバー環境でのサーバー）レベルの参照整合性が規定されます。

参照整合性の目的は、孤立したレコード（マスター・レコードとの有効なリンクを持たないディテール・レコード）が作成されないようにすることです。参照整合性を規定する規則により、結果として孤立したレコードを作成するような、マスター・レコードの削除や更新、またはディテール・レコードの挿入や更新を予防できます。

シーケンス (Sequence)

順次数を生成するスキーマ・オブジェクト。シーケンスを作成した後は、これを使用してトランザクション処理用の一意の順次数を生成できます。これらの一意の整数には、主キー値を含むことができます。トランザクションで順序番号が生成される場合、トランザクションをコミットしたかロールバックしたかにかかわらずシーケンスが即時増分されます。

[「WINDOW シーケンス」](#) および [「LEAPFROG シーケンス」](#) も参照。

実表 (Base Table)

ビューの基になるデータのソースで、表またはビューのいずれか。ビュー内のデータにアクセスするとき、実際は実表のデータにアクセスしています。

シノニム (Synonym)

表、ビュー、シーケンス、スナップショットまたは別のシノニムに対する代替名（エイリアス）。

主キー (Primary Key)

表の主キーは、表内の各行を一意に識別するのに使用される1つの列または列グループです。主キーを使用すると表のレコードにすばやくアクセスできます。また主キーは2つの表またはビューの間の結合の基礎として頻繁に使用されます。それぞれの表に対して主キーは1つしか定義できません。

PRIMARY KEY 制約を満たすには、主キー値が表の 2 つ以上の列で使用されたり、主キーの一部の列に NULL 値が含まれないようにします。

スキーマ (Schema)

表、ビュー、索引、シーケンスなどを含む、名前の付いたデータベース・オブジェクトの集まり。

スナップショット (Snapshot)

スナップショットとは Web-to-Go が Oracle データベースからリアルタイムで取得するアプリケーション・データのコピーで、オフラインになる前にクライアントにダウンロードされます。スナップショットは、データベース表全体のコピー、または表の行のサブセットのコピーです。ユーザーが初めてオンラインからオフラインに切り替えるとき、Web-to-Go はクライアント・マシン上にスナップショットを自動的に作成します。その後、オンラインまたはオフラインに切り替えるたびに、Web-to-Go はスナップショットの複雑さに応じて、スナップショットを最新のデータでリフレッシュするか、全体を再作成します。

整合性制約 (Integrity Constraint)

表の 1 つ以上の列に入力できる値を制限する規則。

接続 (Connected)

サーバーに接続されているユーザー、アプリケーションまたはデバイスを指す一般的な用語。Web-to-Go の Mobile クライアントは、オンライン・モードのときに接続されています。

切断 (Disconnected)

サーバーに接続されていないユーザー、アプリケーションまたはデバイスを指す一般的な用語。Web-to-Go の Mobile クライアントは、オフライン・モードのときに切断されています。

データベース・オブジェクト (Database Object)

データベース・オブジェクトとは、表、ビュー、シーケンス、索引、スナップショットまたはシノニムなどの名前の付けられたデータベース構造体です。

データベース・サーバー (Database Server)

Web-to-Go の 3 層 Web モデルの 3 番目の層。アプリケーション・データを格納します。

同期 (Synchronization)

Web-to-Go が Web-to-Go の Mobile クライアントと Oracle8i の間でデータをレプリケートするために使用するプロセス。Web-to-Go は、ユーザーがオフライン・モードに切り替えたときにユーザーのアプリケーションおよびデータを Oracle Lite にレプリケートします。ユーザーが元のオンライン・モードに切り替えると、Web-to-Go はすべてのデータ変更を Oracle8i にレプリケートします。

トランザクション (Transaction)

リレーショナル・データベース内の選択されたデータに対して加えられる一連の変更。トランザクションは通常、ADD、UPDATE、DELETE などの SQL 文を使用して実行します。トランザクションは、コミットされた（変更が永続的になる）とき、またはロールバックされた（変更が破棄された）ときに完了します。

トランザクションの前に問合せが実行されることがよくあります。問合せを使用して、変更対象の特定のレコードをデータベースから選択しておきます。「SQL」も参照。

パッケージ・ウィザード (Packaging Wizard)

パッケージ・ウィザードを使用すると、管理者が Web-to-Go アプリケーションを Mobile サーバー・リポジトリにパブリッシュできます。管理者は、パッケージ・ウィザードを使用して新しい Web-to-Go アプリケーションを作成したり、既存のアプリケーション定義を編集できます。

パブリケーション項目 (Publication Item)

パブリケーション項目は、クライアントがアクセスできるデータ・サブセットを指定する SQL 選択文です。パブリケーション項目は、通常クライアント・デバイス上のレプリカ表に対応します。パブリケーション項目は、Mobile Server Admin API を使用して作成できます。この API には、パブリッシュ / サブスクライブ・モデルを実装する Java 関数が含まれています。これらの関数は、Java プログラムの内部から標準のファンクション・コールとしてコールできます。

ビュー (View)

1 つ以上の表（または他のビュー）から選択されたデータをカスタマイズして表したもの。ビューは「仮想的な表」のようなもので、複数の表（実表と呼ばれます）およびビューからのデータを関連させ、組み合わせることができます。ビューは表示されるデータの選択条件を指定できるため、一種の「格納された問合せ」といえます。

ビューは、表のように、行と列に編成されます。ただし、ビューには、データそのものは含まれません。ビューを使用すると、複数の表またはビューを 1 つのデータベース・オブジェクトとして扱うことができます。

表 (Table)

行と列に編成されたデータを格納するデータベース・オブジェクト。上手に設計されたデータベースでは、各表に単一のトピックに関する情報（たとえば従業員や顧客の住所など）が格納されます。

マスター・ディテール・リレーション (Master-Detail Relationship)

1 つの表またはビュー（ディテール表またはビュー）の複数行が、別の表またはビュー（マスター表またはビュー）の単一のマスター行に関連付けられている場合に、マスター・ディテール・リレーションがデータベース内の表またはビューの間に存在すると言います。

マスター行およびディテール行は通常、ディテール表またはビュー内の外部キー列と一致するマスター表またはビュー内の主キー列により結合されます。

主キーの値を変更した場合、アプリケーションでは、外部キーの値が主キーの値と一致するように一連の新しいディテール・レコードを問い合わせる必要があります。たとえば、EMP 表内のディテール・レコードが、DEPT 表内のマスター・レコードと同期される場合、DEPT 内の主キーは DEPTNO で、EMP 内の外部キーは DEPTNO にします。「主キー」および「外部キー」も参照。

モードの切替え (Switching Modes)

Web-to-Go の Mobile クライアントがオフラインに切り替えたりオンラインに戻るために使用するプロセス。クライアントがオフライン・モードに切り替わると、オフラインで作業するために必要なすべてのアプリケーションとデータが Oracle Lite にダウンロードされます。クライアントがオンラインに戻ったときに、Oracle Lite に対するデータ変更を Oracle8i と同期します。

レジストリ (Registry)

レジストリには、Web-to-Go の一意の名前と値のペアが含まれます。レジストリの名前はすべて一意である必要があります。

レプリケーション (Replication)

分散データベース・システムを構成する複数のデータベース内で、データベース・オブジェクトをコピーしメンテナンスするプロセス。1つのサイトに適用された変更が取得されローカルに格納されてから、各リモート・サイトに転送され適用されます。レプリケーションは、共有データに対する高速のローカル・アクセスをユーザーに提供し、データ・アクセスの代替オプションを提供してアプリケーションの使用を保護します。1つのサイトが使用不能になっても、残りのサイトに対して問い合わせたり更新できます。

レプリケーションの競合 (Replication Conflict)

レプリケーションの競合は、同一のデータに対して矛盾する変更が加えられたときに発生します。Web-to-Go では、切断されているクライアント用のシーケンス値を使用して、レプリケーションの競合を回避します。

ワークスペース (Workspace)

Mobile サーバーのワークスペースとは、Web-to-Go アプリケーションに対するアクセスをユーザーに提供する Web ページです。Web-to-Go は、ユーザーが Web-to-Go にログインした後に、ユーザーのブラウザ内にワークスペースを生成します。ワークスペースは、ユーザーが使用できるすべてのアプリケーションのアイコン、リンクおよび説明を表示します。アプリケーションを使用できるようになるのは、管理者がアプリケーションを Web-to-Go システムにパブリッシュし、ユーザーに対してアクセス権を付与した後です。

索引

C

CREATEDB
使用方法, B-2

D

DDL 操作, 2-2

I

INSTEAD OF トリガー, 2-4

J

jar ファイル
パッケージ・ウィザードによるパッケージ化, 4-17
JDBC ドライバ, 1-4

M

Message Generator and Processor, 2-2
Mobile Sync Client
概要, 2-2

P

PL/SQL, 2-3

R

REMOVEDB
使用方法, B-9

X

XML ファイル, 4-17

あ

アプリケーション
パッケージ・ウィザードによる編集, 4-19
パッケージ・ウィザードによる命名, 4-3
パブリッシュ, 4-18
ファイルのリスト表示, 4-5
「アプリケーション」パネル
パッケージ・ウィザード, 4-3

い

インストール, 1-8

う

ウィニング・ルール, 2-7

え

エラー, 2-7

お

オブジェクト・カーネル API (OKAPI), 1-5
オブジェクトのネーミング, 2-3
親表
INSTEAD OF トリガー, 2-4
更新可能, 2-3
ヒント, 2-4

か

開発過程, 1-8
外部キー制約, 2-5
違反, 2-5

き

競合, 2-7

く

クライアント
パブリケーションへのサブスクリाइブ, 2-20
クライアント・データベース
作成, 2-2

こ

高速リフレッシュと更新, 2-3

さ

索引
パブリケーション項目用の作成, 2-16
サブスクリプション, 2-2, 2-9
インスタンス化, 2-22
サブスクリプション・パラメータ, 2-9
定義, 2-21

し

システム・カタログ・ビュー, A-1
主キー索引, 2-2

す

スナップショット
インポート, 4-13
パッケージ・ウィザードによる作成, 4-12
編集, 4-15
「スナップショット」パネル
パッケージ・ウィザード, 4-8

て

データ型
Oracle Lite, 2-8
マッピング, 2-8
データベース・サポート, 2-3
「データベース」パネル
パッケージ・ウィザード, 4-7
データベース・ユーティリティ
CREATEDB, B-2
REMOVEDB, B-9
テスト
デバイスでの, 1-12

と

トランザクション
実行, 2-26
ページ, 2-27

は

ページ、トランザクション, 2-27
バージョン設定, 2-7
パッケージ・ウィザード
jar ファイルのパッケージ化, 4-17
「アプリケーション」パネル, 4-3
起動, 4-2
スナップショットのインポート, 4-13
スナップショットの編集, 4-15
「スナップショット」パネル, 4-8, 4-12
「データベース」パネル, 4-7
ファイルのソート, 4-6
「ファイル」パネル, 4-5
パブリケーション, 2-2, 2-9
クライアントのサブスクリाइブ, 2-20
作成, 2-12
パブリケーション項目, 2-9
索引の作成, 2-16
作成, 2-14
パブリケーションへの追加, 2-17
パブリッシュ, 2-24
パブリッシュ・サブスクリाइブ・モデル, 2-2, 2-9

ひ

ビュー

高速リフレッシュと更新, 2-3

ヒント, 2-4

ふ

「ファイル」パネル

パッケージ・ウィザード, 4-5

れ

レプリケーション

エミュレータによる, 1-12

エラー, 2-7

競合, 2-7

レプリケート・シーケンス

クライアントごとのパーティション化, 2-20

作成, 2-19

レプリケート・シーケンスのパーティション化, 2-20

