Oracle9*i* Lite

Windows 32 開発者ガイド

リリース 5.0.1

2002 年 4 月

部品番号：J06005-01

ORACLE®

Oracle9*i* Lite Windows 32 開発者ガイド , リリース 5.0.1

部品番号：J06005-01

原本名：Oracle9*i* Lite Developers Guide for Windows 32, Release 5.0.1

原本部品番号：A95912-01

目次

2 Oracle Lite のサンプル・アプリケーションの使用方法

4 パッケージ・ウィザードの使用

A POLITE.INI のデータベース・パラメータ

B システム・カタログ・ビュー

C データベースのツールとユーティリティ

D SQL 問合せの最適化

図

表

はじめに

『Oracle9*i* Lite Windows 32 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 を使用した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 データベースの概要を説明します。プログラミング・インタフェース、スキーマの作成、データベースの移行方法、Oracle Lite データベースの操作および初期データベースの使用方法を説明します。
第 2 章「Oracle Lite のサンプル・アプリケーションの使用方法」	Oracle Lite で提供されているサンプル・アプリケーションについて説明します。
第 3 章「同期」	Consolidator を使用したレプリケーションについて説明します。Consolidator 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、索引および順序などが含まれています。
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たアプリケーションの配布方法を説明します。
付録 A「POLITE.INI のデータベース・パラメータ」	**POLITE.INI** ファイルに設定できるデータベース・パラメータについて説明します。
付録 B「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 C「データベースのツールとユーティリティ」	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。
付録 D「SQL 問合せの最適化」	SQL 問合せのパフォーマンスを向上させるためのヒントを提供します。

付録 E「Oracle Lite ロード・ユーティリティの API」	Oracle Lite ロード・ユーティリティ API について説明します。

1

概要

この章では、Mobile Development Kit およびそのコンポーネントの概要を説明します。内容は次のとおりです。

1.1 概要

Oracle9*i* Lite Mobile Development Kit には、次のものが含まれています。

- Oracle Lite データベース－軽量小型の組込みデータベース
- Mobile Sync －トランザクション・レプリケーション・エンジン

Mobile Development Kit（Windows 用）を使用すると、エンタープライズ・データおよびアプリケーションを分散モバイル・プラットフォームに配布できます。デバイス上のデータは、各プラットフォーム用に設計されたオブジェクト・リレーショナル・データベースの Oracle Lite データベースに格納されます。配布後、Oracle Lite データベースはエンド・ユーザーに対し透過的に機能し、最低限のチューニングと管理しか必要ありません。

注意： この章では、Windows システム上のアプリケーション開発とデータベースの同期についてのみ説明しています。Windows CE に関しては、『Oracle9*i* Windows CE 開発者ガイド』を参照してください。EPOC に関しては、『Oracle9*i* Developer's Guide for EPOC』を参照してください。

1.1.1 Oracle Lite DBMS

Oracle Lite データベースは、ラップトップ・コンピュータ、携帯端末、PDA およびその他の情報機器用に特別に作成された、軽量小型で Java 対応のリレーショナル・データベースです。Oracle Lite データベースは、Windows 95/98/NT/2000、Windows CE/Pocket PC、Palm Computing および EPOC 上で稼働します。Oracle Lite データベースには ODBC および OKAPI プログラミング・インタフェースが提供されているため、C/C++、Visual Basic および Satellite Forms などの様々なプログラミング言語でデータベース・アプリケーションを作成できます。このようなデータベース・アプリケーションは、データベース・サーバーに接続されていない状態でも使用できます。

1.1.2 Mobile Sync

Mobile Sync はモバイル・デバイスに常駐する軽量小型のアプリケーションです。Mobile Sync を使用すると、携帯端末、デスクトップ・コンピュータおよびラップトップ・コンピュータと Oracle データベース間のデータを同期させることができます。Mobile Sync は、Windows 95/98/NT/2000、Windows CE/Pocket PC、Palm Computing および EPOC 上で稼働します。

開発者がアクセスする必要のあるコンポーネントはいくつかあります。

- Oracle データベース・サーバー。
- Mobile サーバー。モバイル・デバイスと Oracle データベース・サーバー間のゲートウェイです。

- Message Generator and Processor（MGP）。Mobile サーバーと Oracle データベース・サーバー間のトランザクションとデータ交換を管理するバックグラウンド・プロセスです。
- Mobile Sync はモバイル・デバイス上に常駐し、任意の通信方式を介して Mobile サーバーと通信します。
- モバイル・デバイスと Mobile サーバー間の通信メカニズム（HTTP など）。

一般に、携帯端末は、中央データベースに含まれているデータのごく一部のサブセットのみを扱います。たとえば、ある地域の営業担当員がアクセスする必要があるのは、その地域内の顧客の連絡先情報のみです。Mobile サーバーは、データとアプリケーションの管理にパブリッシュ・サブスクライブ・モデルを使用します。このモデルでは、Mobile Server Admin API を使用して作成するパブリケーションが、特定のサブスクライバに提供するデータ・サブセットを定義します。サブスクリプションが、ユーザーをパブリケーションに関連付けます。クライアントとサーバーがデータを交換するとき、Mobile サーバーはユーザーが定義した規則を使用してデータ競合を検出し解消します。

1.1.3 Mobile サーバー

Mobile サーバーは、携帯端末上の Oracle Lite データベースと Oracle Server の間のシームレスな同期を実現します。

データの配布にパブリッシュ・サブスクライブ・モデルを使用すると、Mobile サーバーでは個々のクライアント・デバイスでユーザー ID や場所などのパラメータに基づいた様々なデータ・サブセットをサブスクライブできます。Mobile サーバーは、デバイスとサーバー間で HTTP トランスポート・プロトコルをサポートします。

1.1.4 Mobile SQL

Mobile SQL を使用すると、ラップトップおよび携帯端末上で Oracle Lite データベースを作成、アクセスおよび操作できます。Mobile SQL を使用すると、次のことができます。

- データベースの作成
- 表の表示
- SQL 文の実行

1.1.5 サンプル

Mobile Development Kit には、サンプル・アプリケーションが含まれています。サンプルは、Mobile Development Kit のインストール時に自動的にインストールされ、Oracle_Home¥Mobile¥SDK¥Examples にあります。

1.2 開発インタフェース

この項では、開発インタフェースの概要を説明します。内容は次のとおりです。

1.2.1 開発インタフェースの概要

Oracle Lite では、データベース・アプリケーション開発のために次のインタフェースを提供しています。

- リレーショナル・データベース開発用 :
  - JDBC
  - ODBC
- オブジェクト・データベース開発用 :
  - オブジェクト・カーネル API（OKAPI）

モデルは単独でも組み合せても使用できます。たとえば、同一プログラム内に OKAPI コールと JDBC コールの両方を含めることができます。

1.2.1.1 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）の主要コンポーネントである 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、JDBC ドライバ（タイプ 2）経由で 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。Oracle Lite データベースは、Branch Office の下で稼働する Oracle Lite Multi User Service と接続するときに JDBC ドライバ（タイプ 4）を使用します。タイプ 2 ドライバは、リモート・データベースとネイティブ・データベースの両方で使用できます。

JDBC および Oracle Lite の詳細は、『Oracle9*i* Lite Java 開発者ガイド』を参照してください。

1.2.1.2 ODBC

Microsoft 社の Open Database Connectivity（ODBC）インタフェースは、SQL データベースにアクセスするためのプロシージャ型コール・レベル・インタフェースで、多くのデータベース・ベンダーがサポートしています。ODBC は、データベースへの接続、実行時の SQL 文の準備と実行、および問合せ結果の取出しを可能にする一連の関数をアプリケーションに対して提供します。Oracle Lite では、ODBC コールを解釈して Oracle Lite に渡す Oracle Lite ODBC ドライバを介して、レベル 3 準拠の ODBC 2.0 ドライバと ODBC 3.5 ドライバをサポートします。

ODBC の詳細は、次を参照してください。

- Microsoft 社の ODBC のドキュメント。
- Oracle Lite に含まれている ODBC サンプル・アプリケーション。このアプリケーションの場所は、第 2 章「Oracle Lite のサンプル・アプリケーションの使用方法」を参照してください。
- 詳細は、C.7 項「ODBC Administrator と Oracle Lite ODBC ドライバ」を参照してください。

注意： ODBC 3.5 ドライバは、Java ストアド・プロシージャをサポートしていません。

1.2.1.3 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、『Oracle9*i* 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。

1.2.2 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 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。Mobile Sync API の詳細は、3.6 項「OCAPI.DLL を使用したプログラミング」を参照してください。

1.2.3 Oracle Lite ロード・ユーティリティ（OLLOAD）

Oracle Lite ロード・ユーティリティを使用すると、外部ファイルから Oracle Lite データベースの表にデータをロードしたり、Oracle Lite データベースの表から外部ファイルにデータをアンロード（ダンプ）できます。OLLOAD の詳細は、付録 E「Oracle Lite ロード・ユーティリティの API」を参照してください。

1.3 初期データベースの使用

Oracle Lite データベースをインストールすると、ODBC データ・ソース名（DSN）POLITE が作成され、初期データベース **POLITE.ODB** が POLITE 専用に指定されます。DSN POLITE の新規データベースの場所は、*Oracle_Home*¥Mobile¥SDK¥oldb40 に設定されます。

サンプルのインストール中に、SYSTEM という名前のデフォルト・ユーザーが設定されます。SYSTEM にはすべてのデータベース権限が含まれ、パスワードはありません。SYSTEM 用のパスワードは、ALTER USER コマンドを使用して作成できます（次の項でサンプルの構文を説明します）。デフォルトのユーザー名を使用するか、独自のユーザー名を設定できます。

注意： 初期データベースを使用する前に、『Oracle9*i* Lite SQL リファレンス』を参照してください。このマニュアルには、Oracle Lite データベース内の情報の管理に使用する Structured Query Language（SQL）が説明されています。

Mobile SQL などのアプリケーションを使用して、Oracle Lite 初期データベースに接続できます。Mobile SQL はコマンドライン・インタフェースです。POLITE データベースに接続するには、「Connect」と入力し、次に「ユーザー名 / パスワード @DSN 名」を入力します。たとえば、次のようになります。

```
Connect SYSTEM/MANAGER@POLITE
```

SYSTEM には、次のコマンドを入力してパスワードを割り当てられます。

```
ALTER USER SYSTEM IDENTIFIED BY <password>
```

注意： 詳細は、1.5.4 項「パスワードの変更」を参照してください。

ODBC アプリケーションから初期データベースに接続する場合は、デフォルトの ODBC DSN である POLITE を使用します。

1.4 データベースの操作

この項では、Oracle Lite データベースの操作の概要（データベースの作成、データベースへの接続、ユーザーの作成およびデータベースの管理など）を説明します。

1.4.1 新規データベースの作成

POLITE データ・ソース名を使用して新規データベースを作成すると、この新規データベースのファイルは *Oracle_Home*¥Mobile¥SDK¥oldb40 ディレクトリに入れられます。メンテナンスを容易にするために、すべてのデータベースを 1 つのデータベース・ディレクトリに入れることをお薦めします。

注意： 新規に作成されるすべてのデータベースにユーザー SYSTEM が含まれています。パスワードは NULL です。

新規データ・ソース名は、ODBC Administrator を使用して作成できます。詳細は、1.4.2 項「ODBC Administrator を使用したデータ・ソース名の作成」を参照してください。

1.4.2 ODBC Administrator を使用したデータ・ソース名の作成

ODBC Administrator は Microsoft 社が提供するツールです。Windows 95/98/NT および Windows 2000 の **ODBC.INI** ファイルおよびそれに関連付けられたレジストリ・エントリを管理します。このツールを使用すると、データ・ソース名を追加し、そのデータ・ソース名のデフォルトに使用するデータベース・ファイルを指定できます。ODBC Administrator の詳細とこのツールを使用したデータ・ソース名の作成方法の詳細は、C.7 項「ODBC Administrator と Oracle Lite ODBC ドライバ」を参照してください。

1.4.3 コマンドライン・ユーティリティを使用した新規データベースの作成

コマンドラインから新規データベースを作成するには、CREATEDB ユーティリティを使用します。構文は次のとおりです。

```
CREATEDB mydatabase mydbname
```

たとえば、次のようになります。

```
CREATEDB polite newdb
```

mydatabase は DSN 名で、*mydbname* は新規データベース名です。

詳細は、C.2 項「CREATEDB」を参照してください。

1.4.4 新規データベースへの接続

Mobile SQL を使用して新規データベースに接続するには、ユーザー SYSTEM、パスワード MANAGER およびデータ・ソース名で接続します。たとえば、次のようになります。

```
C:¥msql system/manager@ODBC:polite
```

MYDATABASE は事前に定義した ODBC データ・ソース名に置き換えます。

データ・ソース名に複数のデータベースが対応付けられている場合は、次の書式を使用します。

```
ODBC:dsn:dbname
```

dsn は DSN 名で、*dbname* はデータベースの名前です。たとえば、次のようになります。

```
C:¥msql system/manager@ODBC:polite:newdb
```

1.5 複数のユーザーの作成

CREATE USER コマンドを使用すると、Oracle Lite に複数のユーザーを作成できます。ユーザーはスキーマではありません。ユーザーを作成すると、Oracle Lite によりそのユーザーの名前でスキーマが作成され、これがデフォルト・スキーマとしてそのユーザーに自動的に割り当てられます。スキーマ名を接頭辞として指定しなくても、デフォルト・スキーマ内のデータベース・オブジェクトにアクセスできます。

適切な権限を持つユーザーは、CREATE SCHEMA コマンドを使用して追加スキーマを作成できますが、データベースに接続できるのはそのユーザーのみです。スキーマ名を使用してデータベースに接続することはできません。スキーマ名とユーザー名が同じでも、オブジェクトは異なります。

スキーマはスキーマを作成したユーザーにより所有され、スキーマ内のオブジェクトにアクセスするには、スキーマ名を接頭辞として指定する必要があります。

CREATEDB ユーティリティまたは CREATE DATABASE コマンドを使用してデータベースを作成すると、Oracle Lite は SYSTEM という特殊なユーザーを作成します。SYSTEM にはすべてのデータベース権限があり、パスワードは割り当てられていません。必要な場合は、SYSTEM にパスワードを割り当てることができます。独自のユーザー名を設定するまで、SYSTEM をデフォルト・ユーザー名として使用できます。

Oracle Lite では、SYSTEM 以外のユーザーがデータにアクセスしたり、所有していないスキーマ内で操作を実行することはできません。他のユーザーのスキーマ内のデータにアクセスしたり操作を実行するには、特定の権限が必要です。

1.5.1 事前定義のロール

Oracle Lite では、便宜上、事前定義のロールに権限をいくつか組み合せています。多くの場合、ユーザーに事前定義のロールを付与する方が、別のスキーマ内の特定の権限を付与するより容易です。Oracle Lite では、ロールの作成または削除はサポートしていません。Oracle Lite の事前定義ロールの一覧を次に示します。

表 1-1 事前定義のロール

ロール名	ロールに付与されている権限
ADMIN	他のユーザーを作成し、スキーマ内のオブジェクトに対する DBA および ADMIN 以外の次の権限を付与できます。 CREATE SCHEMA、CREATE USER、ALTER USER、DROP USER、DROP SCHEMA、GRANT、REVOKE
DBA	SYSTEM でのみ発行できる次の DDL 文を発行できます。 すべての ADMIN 権限、CREATE TABLE、CREATE ANY TABLE、CREATE VIEW、CREATE ANY VIEW、CREATE INDEX、CREATE ANY INDEX、ALTER TABLE、ALTER VIEW、DROP TABLE、DROP VIEW および DROP INDEX

表 1-1 事前定義のロール（続き）

ロール名	ロールに付与されている権限
RESOURCE	RESOURCE ロールは、DBA ロールと同じ制御レベルを付与しますが、制御範囲はそのユーザー自身のドメイン内のみです。SQL 文で次のコマンドを実行できます。 ユーザー自身のスキーマ内のオブジェクトに対する CREATE TABLE、CREATE VIEW、CREATE INDEX、CREATE CONSTRAINT、ALTER TABLE、ALTER VIEW、ALTER INDEX、ALTER CONSTRAINT、DROP TABLE、DROP VIEW、DROP INDEX、DROP CONSTRAINT、および GRANT または REVOKE 権限

一般的注意： Oracle Server とは異なり、Oracle Lite では明示的に COMMIT コマンドを発行しないかぎりデータ定義言語（DDL）コマンドをコミットしません。

1.5.2 ユーザーの作成

データベースに「SYSTEM」として接続している場合、または ADMIN ロールか DBA ロールが付与されている場合は、ユーザーを作成できます。ユーザーを作成するには、次の文を発行します。

```
CREATE USER user IDENTIFIED BY password
```

user は文字で始まる最大 128 文字の一意のユーザー名で、password は最大 128 文字の文字列です。この文は、ユーザー名と同じ名前のスキーマを作成し、そのスキーマをそのユーザーのデフォルト・スキーマとして割り当てます。

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** ファイルにアクセスする前に、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。

1.5.3 ユーザーの削除

データベースに「SYSTEM」として接続している場合、または ADMIN ロールか DBA ロールが付与されている場合は、ユーザーを削除できます。

ユーザーおよびそのユーザーが所有するスキーマをすべて削除するには、次の構文を使用します。

```
DROP USER user
```

ユーザーを削除する前にユーザーのスキーマ内のオブジェクトをすべて削除するには、次の構文を使用します。

```
DROP USER user CASCADE
```

DROP USER コマンドの詳細は、『Oracle9*i* Lite SQL リファレンス』を参照してください。

1.5.4 パスワードの変更

次のいずれかの条件を満たす場合は、ユーザーのパスワードを変更できます。

- そのユーザーとしてデータベースに接続されている場合。
- SYSTEM としてデータベースに接続されている場合。
- ADMIN、DBA または RESOURCE ロールが付与されている場合。

ユーザーのパスワードを変更するには、次の文を発行します。

```
ALTER USER user IDENTIFIED BY password
```

1.5.5 ロールの付与

ADMIN または DDL ロールをユーザーに付与するには、次の文を発行します。

```
GRANT role TO user_list
```

user_list は 1 人のユーザーか、複数ユーザーをカンマで区切ったリストです。

1.5.6 権限の付与

データベース・オブジェクトに対する権限をユーザーに付与するには、次の文を発行します。

```
GRANT privilege_list ON object_name TO user_list
```

privilege_list は、次に示す権限をカンマで区切ったリストか、ALL と呼ばれる組合せです。

- ALL
- INSERT
- DELETE
- UPDATE (column_list)
- SELECT

object_name は、スキーマ名を接頭辞として指定した表名です。

privilege_list が ALL の場合、ユーザーは表またはビューに対して INSERT、DELETE、UPDATE または SELECT を実行できます。privilege_list が INSERT、DELETE、UPDATE または SELECT のいずれかである場合、ユーザーは表に対しそれぞれの権限を持ちます。

1.5.7 ロールの取消し

ユーザー・ロールを取り消すには、次の文を発行します。

```
REVOKE role FROM user_list
```

1.5.8 権限の取消し

ユーザーからデータベース・オブジェクトに対する権限を取り消すには、次の文を発行します。

```
REVOKE privilege_list ON table_name FROM user_list
```

注意： ODBC インタフェースを使用すると、すべてのアプリケーションで CREATE USER 文、DROP USER 文および ALTER USER 文を発行できます。

1.5.9 デモで使用する表の作成

Oracle Lite には、**POLDEMO.SQL** というスクリプトが含まれています。このスクリプトを使用すると、Oracle Lite のデフォルト初期データベース（POLITE.ODB）に含まれている表と同じ表を作成できます。

1.5.10 Mobile SQL を使用したデータベースの移入

SQL スクリプトを使用すると、表とスキーマを作成し、表にデータを挿入できます。SQL スクリプトは、通常、**.SQL** という拡張子の付いたテキスト・ファイルで、SQL コマンドが含まれています。SQL スクリプトは、Mobile SQL のプロンプトで次のように入力して実行できます。

```
SQL> @Oracle_home¥DBS¥Poldemo.sql
```

または、次のように入力することもできます。

```
SQL> START filename
```

注意： スクリプトを実行するときは、**.SQL** ファイル拡張子を指定する必要はありません。

1.5.11 データベースのバックアップ

Oracle Lite データベースは 1 つのファイルを占有し、それに依存するログ・ファイルがあります。これらのファイルは、別の場所にコピーすることでバックアップできます。ただし、ファイルをコピーする前に、データベース管理者はデータベースをシャットダウンし、ログ・ファイルにある変更をデータベースに適用することを確認する必要があります。この後、***.odb**、***.opw** および ***.plg** ファイルを別のディレクトリにコピーして、データベースのバックアップを作成します。

1.5.12 データベースの暗号化と復号化

ENCRYPDB および DECRYPDB という 2 つのツールを使用すると、Oracle Lite データベースを暗号化および復号化できます。この 2 つのツールでは、パスワードを指定して Oracle Lite データベースを暗号化します。パスワードを使用することで、許可されていないデータベース・アクセスを防止できます。また、データベースを暗号化することで、データベース・ファイルを調べて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します。パスワードは 40 ビットの暗号化キーを導出するために使用されます。Oracle Lite では、CAST5 と呼ばれるデータ暗号化規格（DES）を使用しています。

この 2 つのユーティリティの詳細は、C.4 項「ENCRYPDB」および C.3 項「DECRYPDB」を参照してください。

1.6 Oracle Lite データベースのトランザクション・サポート

アプリケーションは、Oracle Lite データベースに接続すると、データベースとのトランザクションを開始します。Oracle Lite データベース 1 つにつき、最大 32 の接続が可能です。Oracle Lite データベースへの各接続は、それぞれ別々のトランザクションを維持します。

1.6.1 原子性

トランザクションとは、SELECT、UPDATE、DELETE および INSERT などの一連のデータベース操作のことです。すべての操作は、正常に実行されてコミットされるか、ロールバックされるかのいずれかです。これは、トランザクションの原子性プロパティと呼ばれます。

Oracle Lite では、データベース・コミットの時点まで実際のデータベース・ファイルを更新しないことで原子性を実現しています。コミット時には、一時 UNDO ログが作成されてからデータベース・ファイルが更新されます。停電などによりコミットが中断された場合、データベースは次の接続時にこのログからリストアされます。

1.6.2 一貫性

トランザクションはデータベースの一貫性を維持します。トランザクションは、ある一貫した状態から別の一貫した状態にデータベースを変換します。その間、一貫性は必ずしも保持されません。Oracle Lite では、トランザクションが制約に違反した場合、つまり一貫性に違反した場合は、トランザクションのコミットを許可しません。

1.6.3 分離

トランザクションは相互に分離され独立しています。多数のトランザクションが同時に実行されても、特定のトランザクションによる更新は、そのトランザクションがコミットされるまで他のトランザクションからは隠されます。Oracle Lite では、次に示すトランザクションの分離レベルをサポートしています。

表 1-2 分離レベル

分離レベル	説明
READ COMMITTED	この分離レベルでは、データの最新のコミット済みバージョンの一時スナップショットを作成し、SELECT 文のロックは不要です。READ COMMITTED トランザクションは、他のトランザクションによりブロックされません。他のトランザクションは、このトランザクションが SELECT 文で要求した表に新規データを更新または挿入できます。この結果、同一の SELECT 文を使用して異なるデータ・セットを返すことができます。 FOR UPDATE 句を含む SELECT 文は、REPEATABLE READ 分離レベルで実行されているかのように実行されます。 Oracle Lite では、SELECT 文で Java ストアド・プロシージャを実行できます。Java ストアド・プロシージャを実行するトランザクションが READ COMMITTED 分離レベルであり、Java ストアド・プロシージャがデータベースを更新する場合は、Java ストアド・プロシージャを実行する SELECT 文に FOR UPDATE 句を指定する必要があります。指定しないと、エラーが発生します。
REPEATABLE READ	この分離レベルでは、返されるすべての行に対して問合せが読取りロックを取得します。問合せ自体の複雑さ、表に対して定義されている索引、あるいは問合せオプティマイザにより選択されている実行計画によっては、さらに多くの行が読取りロックされる可能性があります。REPEATABLE READ 分離レベルでは、トランザクションが終了するまでロックが保持されるため、READ COMMITTED 分離レベルのトランザクションと比べて有効性が低くなります。 問合せで指定されている検索条件に適合する追加行をトランザクションが挿入すると、その後同じ問合せを実行した場合に検索条件に適合する行が余分に返されることがあります。 問合せで FOR UPDATE 句を使用した場合は、現在選択されている行に短時間の更新ロックが取得されます。行が更新されると、このロックは排他的ロックになります。排他的ロックは、READ COMMITTED 以外の分離レベルで実行されているトランザクションがこの行にアクセスすることを防止します。この行は更新されずに次の行がフェッチされる場合は、更新ロックが読取りロックに降格し、他のトランザクションがその行を読み取れるようになります。

表 1-2 分離レベル（続き）

分離レベル	説明
SERIALIZABLE	この分離レベルでは、問合せに関与するすべての表に対して共有ロックが取得されます。同一トランザクション内で 1 つの問合せを繰り返し実行すると、同じ行セットが返されます。問合せ対象の表に含まれている行を更新しようとする他のトランザクションは、ブロックされます。
SINGLEUSER	この分離レベルでは、データベースに対して許可される接続は 1 つのみです。トランザクションにはロックがなく、消費メモリーも少なくて済みます。

分離レベルの詳細（具体的にはトランザクションの分離レベルを定義する「内容を保証しない読取り」、「非リピータブル・リード」および「仮読取り」の説明）は、ODBC のドキュメントを参照してください。

1.6.3.1 持続性

トランザクションの持続性が保証されます。つまり、いったんトランザクションがコミットされると、その後システムに障害があった場合でも、トランザクションによる変更はすべてデータベース・ファイルに永続的に保持されます。システムの障害のためにトランザクションがコミット中またはロールバック中に失敗した場合は、データベースを一貫性のある状態にリストアするために UNDO ログ・ファイルが必要になります。

1.6.3.2 ロック

Oracle Lite では行レベルのロックをサポートします。行が読み取られるときは、その行は読取りロックされます。行が更新されるときは、その行は書込みロックされます。異なるトランザクションから、読取りロックされた同一の行を読み取ることができます。ただし、書込みロックされている行は別のトランザクションからはアクセスできません。

1.6.3.3 デフォルトの分離レベルの変更

Oracle Lite では、READ COMMITTED 分離レベルがデフォルトです。

データ・ソース名（DSN）に対するデフォルト分離レベルを変更するには、ODBC Administrator を使用するか、**ODBC.INI** ファイルを手動で編集して次の行を含めます。

```
IsolationLevel = XX
```

XX は、RC、RR、SR または SU です。

トランザクションの分離レベルは、次の SQL 文を使用して設定することもできます。

```
SET TRANSACTION ISOLATION LEVEL <ISOLATION_LEVEL>;
```

ISOLATION_LEVEL は、READ COMMITTED、REPEATABLE READ、SERIALIZABLE または SINGLE USER です。

詳細は、1.6.3.4 項「サポートされている分離レベルとカーソル・タイプの組合せ」を参照してください。

1.6.3.4 サポートされている分離レベルとカーソル・タイプの組合せ

サポートされている分離レベルとカーソル・タイプの組合せを次の表に示します。分離レベルは左に、カーソル・タイプは上に示されています。「S」はサポートされていることを、「U」はサポートされていないことを示します。

表 1-3 サポートされている組合せ

分離レベル	Forward Only	Static	Keyset Driven	Dynamic
READ COMMITTED	S	S	U	U
REPEATABLE READ	S	U	S	S
SERIALIZABLE	S	U	S	S
SINGLE USER	S	S	S	S

サポートされていない組合せでは、エラー・メッセージが生成されます。

1.6.4 アプリケーションのチューニング

アプリケーション設計のチューニングは、アプリケーションの実装を開始する前に行うのが理想です。設計を開始する前に、Oracle Lite で使用可能な各機能を注意深く検討し、要件に最適の機能がどれかを考慮する必要があります。また、Oracle データベース管理者とも相談して、開発対象アプリケーションに対応させるには Oracle マスター・サイトにどのようなチューニングが必要かを判断するようにします。具体的な設計ヒントの概要は、付録 D「SQL 問合せの最適化」に説明されています。

1.7 スナップショット定義の作成

アプリケーションが機能するためにはデータが必要で、アプリケーションは Mobile サーバー経由でデータにアクセスします。アプリケーションはデータベース上の正しい表にダイレクトされる必要があります。これは、クライアント上のスナップショットと Mobile サーバー上のパブリケーション項目を使用して実行されます。スナップショット定義とは、Mobile サーバーが、どの表を使用するかをアプリケーションに指示するためのプロセスです。

ほとんどの場合、アプリケーションで使用されるスナップショットを作成する対象の表はすでに存在しています。次の方法を使用すると、Mobile サーバー上にパブリケーション項目を作成できます。Mobile サーバーは、データベースと同期するときに、クライアント上にスナップショットを自動的に作成します。スナップショット定義の作成オプションは、次のとおりです。

1. 宣言によるスナップショット定義の作成ーパッケージ・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. プログラムによるスナップショット定義の作成ー Consolidator 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。

1.7.1 宣言によるスナップショット定義の作成

この方法では、Mobile サーバーのパッケージ・ウィザードを利用します。この GUI ツールを使用すると、アプリケーションを **.jar** ファイルにパッケージ化することができます。このファイルはアプリケーションのデプロイメント・ディスクリプタで、Mobile サーバーがアプリケーションを管理するために必要な情報が含まれています。この **.jar** ファイルは Mobile サーバー・リポジトリにパブリッシュされます。

パッケージ・ウィザードの 2 番目の機能は、アプリケーションにより使用されるデータベース表を作成するために実行できる SQL スクリプトの生成です。この表は Oracle データベース実表で、Mobile サーバーはこの表と同期を取ります。

このツールの便利な点は、開発者がモバイル・アプリケーションを作成する際に、安全でエラーが発生しにくいことです。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。

- 実表がデータベースにあるかどうかを確認します。ない場合は、次の手順を実行します。
- パッケージ・ウィザードを使用して、スナップショット定義を収集します。
- パッケージ・ウィザードを使用して、**.jar** ファイルと SQL スクリプトを生成します。
- データベースに実表がない場合は、データベースに対して SQL スクリプトを実行して実表を作成します。
- Mobile サーバー・リポジトリに **.jar** ファイルをパブリッシュします。

- Mobile クライアントを設定します。
- Mobile クライアントを Mobile サーバーと同期して、クライアント側スナップショットを作成します。

パッケージ・ウィザード /Mobile サーバーのアーキテクチャを使用してスナップショット定義を作成すると、モバイル・アプリケーションとアプリケーションが必要とするスナップショット定義の管理および配布に集中サーバーを使用できます。この使用手順については、第 4 章「パッケージ・ウィザードの使用」で説明しています。

1.7.2 プログラムによるスナップショット定義の作成

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、第 3 章「同期」で説明されている Consolidator Admin API を使用してプログラムでスナップショット定義を作成する方法です。Mobile サーバーがクライアント・システムと Oracle データベースの両方と通信するため、用語は多少異なります。Mobile サーバーは、パブリケーションに関するすべてを処理します。

パブリケーションには、スナップショット定義と同じ意味のパブリケーション項目などのデータ・レプリケーション・オブジェクトが含まれます。Consolidator Admin API を起動する前に、データベースの実表が必要になります。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、Consolidator 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、同期が発生したときに親データベースのどのデータ・サブセットがクライアントにレプリケートされるかを指定する SQL SELECT 文です。パブリケーション項目は、通常クライアント上のスナップショットに対応します。パブリケーション項目は、Consolidator 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 Consolidator 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、Consolidator 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 3.3 章「パブリッシュ・サブスクライブ・モデル」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 と Consolidator Admin API に関する高度なスキルが必要で、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薦めします。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。

2

Oracle Lite のサンプル・アプリケーションの使用方法

この章では、Oracle Lite データベースに提供されているサンプル・ア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。

2.1 概要

Oracle Lite のインストールを完了した後、*Oracle_Home*¥Mobile¥SDK¥Examples ディレクトリにあるサンプルを使用できます。

注意： サンプル・アプリケーションのほとんどが、データ・ソース名（DSN）として POLITE を使用しています。このデータ・ソース名を削除および再作成する必要がある場合は、REMOVEDB および CREATEDB ユーティリティを使用してください。

表 2-1 サンプル・ファイルのディレクトリ

ツール	サンプル・アプリケーションの場所	説明
AQ Lite	*Oracle_Home*¥Mobile¥SDK¥AQLITE¥AQLSAMP	Mobile クライアントとサーバー・アプリケーション間のメッセージ通信用に AQ Lite を使用する方法を示します。詳細は、『Oracle9*i* Lite AQ Lite 開発者ガイド』を参照してください。
BLOB Manager	*Oracle_Home*¥Mobile¥SDK¥Examples¥BLOB MANAGER	Oracle BLOB データ型の使用方法および Visual Basic の ODBC プログラミング手法とオブジェクト操作を紹介します。詳細は、2.2 項「BLOB Manager のサンプルに関する注意」を参照してください。
Java	*Oracle_Home*¥Mobile¥SDK¥Examples¥Java	JDBC を使用したプログラミング方法を示します。詳細は、『Oracle9*i* Lite Java 開発者ガイド』を参照してください。
ODBC	*Oracle_Home*¥Mobile¥SDK¥Examples¥Odbc	C で作成された ODBC プログラムを提供します。
Visual Basic	*Oracle_Home*¥Mobile¥SDK¥Examples¥Vb	Visual Basic ツールを使用して、Oracle Lite データベースの表に対する問合せを簡単に実行できる例を紹介します。詳細は、2.3 項「Visual Basic サンプル・アプリケーションの実行」を参照してください。

2.2 BLOB Manager のサンプルに関する注意

BLOB Manager のサンプルをインストールするには、*Oracle_Home*¥Lite¥Examples¥BLOB MANAGER の ¥setup フォルダを開き、**setup.exe** を実行します。インストール完了後、「スタート」ボタンをクリックし、「プログラム」メニューから「BLOB Manager」を選択します。

このサンプルを削除するには、「スタート」ボタンをクリックし、「設定」を選択してから「コントロール パネル」を選択します。「アプリケーションの追加と削除」を選択します。**BLOB Manager** を選択し、「追加と削除」ボタンをクリックします。

Visual Basic プロジェクト・ファイルを開いて Visual Basic で実行する前に、前述のように setup.exe と BLOB Manager を「プログラム」メニューから実行してください。「プログラム」メニューからプログラムを実行することで、データベース内に表が自動的に準備されます。

注意： BLOB Manager はデモ用です。デフォルトの ODBC DSN の POLITE を使用してデフォルトのデータベースがインストールされていることが前提です。そうでない場合は、ODBC Administrator を使用して POLITE DSN を作成できます。また、SYSTEM がデータベースの有効なユーザーであることを確認しておく必要があります。

2.3 Visual Basic サンプル・アプリケーションの実行

このサンプル（Visual Basic 5.0 を使用）は、Oracle Lite データベースを使用する Visual Basic アプリケーションの開発方法を示したものです。ODBC DSN の POLITE が使用されます。AddNew、Update および Delete マクロを使用するには、EMP 表に一意の EMPNO 列が必要です。これが、デフォルト・データベースに接続するときのデフォルトの状態です。

Visual Basic サンプル・アプリケーションのインストールと実行に関する説明では、Oracle Lite と Visual Basic（バージョン 4.0 以上）がすでにインストールされていることが前提です。

注意： Visual Basic の ODBC ドライバをインストールしていない場合は、開始の前にインストールする必要があります。

2.3.1 Visual Basic の開始

Visual Basic プログラム・グループにある「Visual Basic」アイコンをダブルクリックして、Visual Basic を開始します。

2.3.2 サンプル・アプリケーションの表とデータの表示

この手順では、Visual Basic 5.0 でのみ使用できる Visual Data Manager を使用します。バージョン 5.0 より前の Visual Basic を使用している場合は、手順 3 に進みます。

1. 「アドイン」メニューの「Visual Data Manager」を選択します。「VisData」ウィンドウで、「ファイル」メニューの「データベースを開く」を選択し、「ODBC」を選択します。
2. 「ODBC Logon」ダイアログ・ボックスで、各フィールドに次のように入力します。
   - DSN: POLITE
   - UID: SYSTEM
   - PW: （少なくとも 1 文字を入力します）
   - Database: POLITE
3. 「OK」をクリックします。「データベース」ウィンドウに Oracle Lite データベース表が表示されます。表をハイライトして右クリックすると、表が開いてレコードが表示されます。

2.3.3 サンプル・アプリケーションの開始

1. サンプル・アプリケーションを開始するには、「ファイル」メニューの「Open Project」を選択します。表示されるダイアログ・ボックスで、*Oracle_Home*¥Mobile¥SDK¥Examples¥Vb ディレクトリに移動します。**MAKEFILE** を選択し、「オープン」をクリックします。

   注意： ファイル **MAKEFILE** が表示されない場合は、ファイル・タイプに *.* を選択してすべてのファイル・タイプを表示します。リストにファイルが表示されます。

2. 「実行」メニューの「開始」を選択してサンプル・アプリケーションを開始し、EMP 表を表示します。

2.3.4 EMP 表内のデータの表示と操作

1. EMP 表のデータを表示する手順は、次のとおりです。
   - 「表示」をクリックすると、EMP 表のデータが表示されます。
   - 「次」をクリックすると、次のレコードが表示されます。
   - 「前」をクリックすると、前のレコードが表示されます。
2. EMP 表のデータを操作するには、「追加」、「更新」および「削除」の各機能を使用します。

2.4 ODBC のサンプル

ODBC のサンプルは、*Oracle_Home*¥Mobile¥Sdk¥Examples¥ODBC にあります。

これらのサンプルは、C++ コンパイラを使用してコンパイルする必要があります。サンプルをビルドするには、コンソールを開き、*Oracle_Home*¥Mobile¥Sdk¥Examples¥ODBC ディレクトリに切り替えて、「nmake」と入力します。

ODBC のサンプルは、「odbctbl」、「odbcview」、「odbcfunc」、「odbctype」および「long」の 5 つあります。これらのサンプルの実行に必要なものは、POLITE データ・ソース名（DSN）のみです。POLITE DSN は Mobile Development Kit のインストール中に自動的に作成されます。

サンプルを実行するには、*Oracle_Home*¥Mobile¥Sdk¥Examples¥ODBC ディレクトリにある **run.bat** を実行します。最初の 4 つのサンプルには、実行されたログを示す独自の出力ウィンドウが含まれています。現在のサンプルのウィンドウを閉じると、次のサンプルが実行されます。サンプルのウィンドウに表示される出力は、ログ・ファイル（**odbctbl.log**、**odbcview.log**、**odbcfunc.log**、**odbctype.log**）にも出力されます。「long」サンプルの出力は、出力ファイル **long.out** に集められます。

2.4.1 サンプル・アプリケーションの内容

ここでは、*Oracle_Home*¥Mobile¥Sdk¥Examples ディレクトリにあるサンプルの機能を説明します。

2.4.1.1 odbctbl

これは、ODBC SQL Table のサンプルです。ODBC API を使用して表を操作する方法を示します。ID、NAME、START_DATE および SALARY という列を含む表 EMP を作成し、この表にデータを移入し、SALARY 列を更新し、行をいくつか削除し、結果の表に対して SELECT を実行し、フェッチ結果を表示します。最後に、EMP 表を削除します。

2.4.1.2 odbcview

これは、ODBC SQL View のサンプルです。ODBC API を使用してビューを操作する方法を示します。表 EMP（前のサンプルと同じ）を作成し、EMP 表からフルネーム（CONCAT スカラー関数を使用）、HIRE_DATE および SALARY を選択するビュー HIGH_PAID_EMP を作成します。次に EMP を移入します。その後で、HIGH_PAID_EMP ビューに対する SELECT 文が発行されて、移入されたデータが表示されます。さらに、EMP 表の SALARY 列を更新し、EMP 表から行をいくつか削除し、HIGH_PAID_EMP に対する SELECT 文を発行して、これらの変更がどのようにビューに反映されているかを表示します。最後に、ビューと表が削除されます。

2.4.1.3 odbcfunc

これは、ODBC SQL スカラー関数のサンプルです。ODBC API でのスカラー関数の使用方法を示します。表 EMP を作成し、この表にデータを移入し、EMP 表に対する ID および FULL_NAME の SELECT を実行します。フルネームの計算には、姓と名を引数として指定した ODBC スカラー関数 CONCAT を使用します。次に、3 文字より小さい ID に関して、ODBC スカラー関数の UCASE と LCASE を使用して、姓を大文字に、名を小文字に変換して、表を更新します。新規データが選択され、再度表示されます。最後に、EMP 表が削除されます。

2.4.1.4 odbctype

これは、ODBC SQL Types のサンプルです。ODBC API を使用して様々なデータ型を操作する方法を示します。このサンプルは、表 EMP を作成し、この表にデータを移入し、行をすべて選択し、結果を表示するサンプルですが、列のバインド方法が他のサンプルとは異なります。まず、SQLNumResultCols をコールして、結果の列の数を見つけます。次に、結果の各列に対して SQLDescribeCol をコールし、各列に関する全情報（列名、列名の長さ、列の型、列の長さ、列の位取りなど）を取得します。この情報を使用して列をバインドします。これは、ODBC API を使用してデータベースからタイプ情報を取得する方法を示したものです。

2.4.1.5 long

このサンプルでは、SQL LONG VARCHAR の基本的な読取り / 書込み関数を実際に使用します。LONG VARCHAR 列を 1 列含む表 LONG_DATA を（最初に削除してから）作成し、この表にデータを挿入します。それぞれの行で、データはフレームに入れられます。各フレームは LONG VARCHAR 型データのバッファ（長さは 4096）を表します。このサンプルでは、SQLParamData と SQLPutData を使用して、フレームを送信して行に移入します。表からの SELECT が発行されて行がフェッチされ、表から LONG VARCHAR データが読み取られます。それぞれの行では、SQL_NO_DATA_FOUND が返されるまで、SQLGetData を使用してデータがフレームとして読み取られます。これらのアクションは、ファイル「long.out」に記録されます。

3

同期

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。

3.1 概要

Mobile サーバーにより、アプリケーションおよびデータを Oracle サーバーとレプリケート、同期および共有できます。データは、Oracle データベースと通信する Mobile サーバー・エージェントを介して Oracle データベース・サーバーに直接マップできます。Mobile サーバーでは、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。データは、HTTP によるトランスポートを介して Oracle データベースにマップされます。

Message Generator and Processor（MGP）は、クライアント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、『Oracle9*i* Lite 管理者ガイド』を参照してください。

3.1.1 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を使用してユーザー操作を処理します。パブリケーションは、Oracle データベース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、通常クライアント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、3.3 項「パブリッシュ・サブスクライブ・モデル」を参照してください。

3.1.2 クライアント・デバイス・データベースの DDL 操作

クライアントが最初に同期をとるときに、Mobile サーバーでは、クライアントにスナップショットの形式でデータベース・オブジェクトを作成できるようになります。デフォルトでは、表の主キー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、クライアント上で 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、『Oracle9*i* Lite Consolidator Admin API リファレンス』を参照してください。

3.1.3 データベース・サポート

サーバー側では、Mobile サーバーは Oracle データベース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。

3.1.4 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 ユーザー名情報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、3.1.7 項「更新可能パブリケーション項目の外部キー制約」を参照してください。これらのコールの使用の詳細は、3.4.3 項「構成と適用を使用したコールバックのカスタマイズ」を参照してください。

3.1.5 データベースのベース・オブジェクトで使用できる特殊文字

データベースのベース・オブジェクトの表、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。ユーザーは 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正確にオブジェクト名を指定する必要があります（多くの場合、大文字を使用しています）。

3.1.6 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。

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。高速リフレッシュ中は増分変更がレプリケートされ、完全リフレッシュ中は現在のデータがすべてリフレッシュされます。

3.1.6.1 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。親表は、ビューの任意の実表です。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。

3.1.6.2 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、『Oracle9*i* 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、Oracle データベースの機能です。INSTEAD OF トリガーの詳細は、Oracle データベースのドキュメントを参照してください。

3.1.6.3 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、ビューの列名に関するヒントを 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、『Oracle9*i* Lite Consolidator Admin API リファレンス』を参照してください。

3.1.6.4 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、『Oracle9*i* Lite Consolidator Admin API リファレンス』を参照してください。

3.1.7 更新可能パブリケーション項目の外部キー制約

Oracle データベースとクライアント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。

3.1.7.1 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順序を保存しません。

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。Oracle データベース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順序で実行されたために、外部キー制約違反が発生しました。

3.1.7.2 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、順序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 DEFFERED 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 制約として作成しなおします。

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. PL/SQL パッケージでは、次のサンプルに示されているように、`BeforeApply` 関数で制約を DEFERRED に設定し、`AfterApply` 関数で IMMEDIATE に設定する必要があります。このサンプルでは、SAMPLE3 という表と `address.14_fk` という制約が使用されています。

```
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;
```

3.1.7.3 表の比率を使用した制約違反の回避

Mobile サーバーでは、表の比率を使用して、クライアント操作をマスター表に適用する順番を判断します。表の比率は整数として表され、次のように実装されます。

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。

3.1.7.1 項「外部キー制約違反の例」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)
```

3.1.8 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、Oracle データベースのアドバンスト・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・エラーは自動的には解決されません。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、3.4.7 項「エラー・キューを使用した競合の解決」を参照してください。

3.1.8.1 バージョニング

Mobile サーバーでは、内部バージョ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。

3.1.8.2 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・ルールがサポートされています。

- 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。

3.2 Oracle サーバーとクライアント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 Oracle データベースと Oracle9*i* 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。Oracle データベースのデータ型は、Oracle9*i* Lite のデータ型と互換性があります。

3.2.1 Oracle Lite データ型

Oracle9*i* 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 によって作成されます。Mobile サーバーでは、Oracle データベースでのデータ精度に応じて自動的に Oracle9*i* 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、Oracle9*i* 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 データ型です。詳細は、『Oracle9*i* 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 3-1 Oracle Lite データ型

Oracle データベースのデータ型	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	-
LONG	-	-	-	-	-	-	-	X
BLOB	-	-	-	-	-	-	X	-
CLOB	-	-	-	-	-	-	-	-

3.3 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 サーバーとクライアント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、特定の専用機能を起動するように、プログラムで Web-to-Go API および Consolidator Admin API を使用して実装できます。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 3-2 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	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。 ■ ユーザーは単一のユーザー名を使用して、複数のクライアント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、単一のクライアント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と、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、ユーザー名とアプリケーション固有の値（社員番号や市外局番など）を含めることができます。

Mobile サーバーのパブリッシュ・サブスクライブ・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. 順序を作成します。
8. ユーザーをパブリケーションにサブスクライブします。
9. クライアントの順序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・サブスクライブ・モデルを実装できます。

表 3-3 パブリッシュ・サブスクライブ・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 Web-to-Go 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。このメソッドを使用するには、**Consolidator.jar** および **webtogo.jar** が開発システムの CLASSPATH 文の中に含まれている必要があります。

3.3.1 API リファレンス

以降の項で説明するメソッドには、詳細が記載されている API ドキュメントの参照先のラベルが付いています。Resource Manager パッケージを指すエントリは、Web-to-Go API に含まれている Mobile Admin クラスの子です。Consolidator クラスを指すエントリは、Consolidator Admin API に含まれています。

index.htm ドキュメントからリンクを辿ることも、*Oracle_home* ¥Mobile¥Doc¥javadoc フォルダを参照することもできます。

注意： Consolidator Admin API では、大 / 小文字が区別されます。Web-to-Go API では、大 / 小文字の区別はありません。

3.3.2 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");
```

3.3.3 Resource Manager クラスのユーザー関数

ここでは、Web-to-Go 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。

3.3.3.1 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 3-4 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");
```

3.3.3.2 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 3-5 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");
```

3.3.3.3 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 3-6 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");
```

3.3.4 パブリケーションの作成

ここでは、Consolidator クラスを使用してパブリケーションを作成する方法を説明します。

3.3.4.1 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。

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。
- パブリケーション項目を作成する前に、レコードはすべてベース・オブジェクトに挿入する必要があります。

3.3.4.2 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 クラスの使用方法

`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_SAMPLE1」という名のパブリケーションを作成します。

表 3-7 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_WIN32	プラットフォーム・タイプを定義する定数。詳細は、『Oracle9*i* Lite Consolidator Admin API リファレンス』を参照してください。

表 3-7 パブリケーション作成パラメータの例（続き）

パラメータ	値	定義
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_WIN32, "%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。

3.3.5 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 クラスの使用方法

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 3-8 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
name	P_SAMPLE1	パブリケーションを指定します。この文字列の長さは最大 26 文字です。
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、3.1.6 項「ビューの高速リフレッシュおよび更新操作」を参照してください。

表 3-8 パブリケーション項目作成パラメータの例（続き）

パラメータ	値	定義
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、3.4.3 項「構成と適用を使用したコールバックのカスタマイズ」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、3.4.3 項「構成と適用を使用したコールバックのカスタマイズ」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);
```

3.3.5.1 Null Sync コールアウト

Mobile サーバーは、同期中に、クライアントが 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;
```

3.3.6 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。

Resource Manager クラスの使用方法

```
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、『Oracle9*i* 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 3-9 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);
```

3.3.7 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 クラスの使用方法

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 P_SAMPLE1 に INDX001 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 3-10 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索引として 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。

表 3-10 パブリケーション項目索引作成パラメータの例（続き）

パラメータ	値	定義
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");
```

3.3.7.1 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、『Oracle9*i* Lite Consolidator Admin API リファレンス』を参照してください。

3.3.8 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、要件に応じて、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、スキーマの発展を使用します。詳細は、『Oracle9*i* 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 クラスの使用方法

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 3-11 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、『Oracle9*i* Lite Consolidator Admin API リファレンス』を参照してください。
conflict_rule	S	競合解決においてサーバー優先を定義します。その他の値は、3.3.8.2 項「競合ルールの定義」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、クライアントにプッシュされるデータを制限するために選択条件が使用されます。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、3.3.8.3 項「表の比率の使用」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", null,
null);
```

3.3.8.1 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、『Oracle9*i* Lite Consolidator Admin API リファレンス』を参照してください。

3.3.8.2 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。Oracle データベースのアドバンスト・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・キューからトランザクションをパージできます。

詳細は、『Oracle9*i* Lite Consolidator Admin API リファレンス』を参照してください。

3.3.8.3 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。

3.3.9 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 クラスの使用方法

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 3-12 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");
```

3.3.10 順序の作成

Mobile サーバーでは、クライアントに対して順序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。

Consolidator クラスの使用方法

CreateSequence 関数を使用して順序を作成できます。構文は次のとおりです。

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

次の例では、CUSTOM1 という名の順序を作成します。

Java メソッドの例

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

C/C++ または Java で作成されたネイティブ・アプリケーションのシーケンス・サポートと Web-to-Go を使用して作成された Web ベース・アプリケーションのシーケンス・サポートは、多少異なります。違いは、CreateSequence() または CreateSequencePartition

関数によりシーケンス・オブジェクトは作成されませんが、順序定義が C$ALL_SEQUENCE_PARTITIONS 表に作成されるということです。次の手順に従います。

1. パッケージ・ウィザードを使用してネイティブ・アプリケーションをパッケージ化します。
2. このネイティブ・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。
3. Consolidator Admin API 関数の `CreateSequence` または `CreateSequencePartition` を使用します。
4. 同期をとります。これにより、**conscli.odb** データベースに C$ALL_SEQUENCE_PARTITIONS 表が作成されます。
5. ネイティブ・アプリケーションが conscli.odb データベースの C$ALL_SEQUENCE_PARTITIONS 表にアクセスし、CURR_VAL と INCR を取り出して、新しい数値を計算します。この数値は、アプリケーションにより C$ALL_SEQUENCE_PARTITIONS 表に格納されます。
6. 同期をとります。これで、C$ALL_SEQUENCE_PARTITIONS 表が同期されます。

3.3.11 クライアントのパーティション化の順序

Mobile サーバー・シーケンスを使用して、クライアント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 ユーザー用にパーティション化する必要があります。

Consolidator クラスの使用方法

`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 3-13 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順序として CUSTOM1 を定義します。
`clientid`	DAVIDL	順序を割り当てるクライアントとして DAVIDL を定義します。

表 3-13 シーケンス・パーティション作成パラメータの例（続き）

パラメータ	値	定義
curr_val	1000	順序の初期値として 1000 を定義します。
incr	1	順序の増分値として 1 を定義します。

Java メソッドの例

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

注意： 順序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。

3.3.12 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 クラスの使用方法

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 3-14 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。

表 3-14 サブスクリプション・パラメータ設定パラメータの例（続き）

パラメータ	値	定義
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'");
```

> **注意：** このメソッドを使用できるのは、Consolidator Admin API を使用して作成されたパブリケーションに対してのみです。

3.3.13 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 クラスの使用方法

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、表に指定されている値を使用して、パブリケーションに対するクライアントのサブスクリプションをインスタンス化します。

表 3-15 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");
```

3.3.14 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、EPOC 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パブリケーション項目形式を受信し続けます。
- サーバーは、変更されている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。

3.3.14.1 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 3-16 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");
```

3.4 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。

3.4.1 パブリケーション項目の問合せのキャッシング

この機能により、複雑なパブリケーション項目の問合せをキャッシュできます。これは、Oracle 問合せエンジンで最適化できない問合せに適用されます。問合せを一時表にキャッシュすることにより、Consolidator テンプレートがスナップショットをより効率的に結合できます。

データを一時表に格納すると MGP 操作に追加オーバーヘッドが発生するため、一時表に格納するのは、最初にパブリケーション項目の問合せが Consolidator テンプレート内でうまく実行されるように問合せを最適化した後のみです。この方法で問合せを最適化できない場合に、キャッシュする方法を使用します。

次の SQL の例には、無効になったためにクライアントから削除する必要のあるクライアント・レコードを識別するために、MGP が構成段階で使用するテンプレートが含まれています。

```
UPDATE cmp$pub_item map
SET delte = true
WHERE client = <clientid>
AND NOT EXISTS (SELECT 'EXISTS' FROM
    (<publication item query>) snapshot
     WHERE map.pk = snapshot.pk);
```

この例では、<publication item query> が複雑になりすぎると、ネストした複数の副問合せ、UNION、仮想列、CONNECT BY 句およびその他の複雑な関数を含んでいるので、問合せオプティマイザが許容可能な計画を判断できなくなります。この結果、MGP 構成段階でパフォーマンスに重大な影響が発生する可能性があります。パブリケーション項目の問合せキャッシング機能を使用してパブリケーション項目の問合せを一時表にキャッシュすることにより、問合せの構造が単純になり、テンプレートで問合せを効果的に結合できます。

3.4.1.1 パブリケーション項目の問合せキャッシングの有効化

次の API により、パブリケーション項目の問合せキャッシングが有効になります。

Consolidator クラスの使用方法

```
public static void EnablePublicationItemQueryCache(String name)
      throws java.lang.Throwable
```

表 3-17 パブリケーション項目の問合せキャッシングの有効化パラメータ

パラメータ	説明
name	パブリケーション項目名を指定する文字列です。

Java メソッドの例

```
Consolidator.EnablePublicationItemQueryCache(
     "P_SAMPLE1");
```

3.4.1.2 パブリケーション項目の問合せキャッシングの無効化

次の API により、パブリケーション項目の問合せキャッシングが無効になります。

Consolidator クラスの使用方法

```
public static void DisablePublicationItemQueryCache(String name)
      throws java.lang.Throwable
```

表 3-18 パブリケーション項目の問合せキャッシングの無効化パラメータ

パラメータ	説明
name	パブリケーション項目名を指定する文字列です。

Java メソッドの例

```
Consolidator.DisablePublicationItemQueryCache(
     "P_SAMPLE1");
```

3.4.2 選択的同期

この機能により、モバイル・アプリケーションが、特定の表の同期方法を選択できます。モバイル・アプリケーション（C/C++、Visual Basic または Java を使用して作成されたネイティブ・アプリケーション）では、Mobile Sync API の **ocSetTableSyncFlag()** 関数を使用して、同期の必要なパブリケーションとパブリケーション項目を判断します。アプリケーションはこの関数をコールし、表ごとに **sync_flag** パラメータを 1 または 0 に設定します。したがって、表のリストはランタイム中に動的に変更できて、アプリケーション開発者は選択的同期をプログラムで制御できます。詳細は、3.6 項「OCAPI.DLL を使用したプログラミング」を参照してください。

3.4.3 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。クライアント・データは MGP に処理される前にインキューに蓄積されます。データは MGP に処理された後、アウトキューに蓄積されてから、Mobile Sync によりクライアントにプルされます。

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。

```
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)
```

3.4.4 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した後は、Java を使用して Mobile サーバー・リポジトリに格納されているカスタマイズ PL/SQL プロシージャを指定できます。PL/SQL プロシージャはそのパブリケーション項目に対するすべての DML 操作にかわってコールされます。各パブリケーション項目に対して、モバイル DML プロシージャは 1 つしか指定できません。このプロシージャは次の構造で作成する必要があります。

```
AnySchema.AnyPackage.AnyName(DML in CHAR(1), COL1 in TYPE, COL2 in TYPE, COLn.., PK1
in TYPE, PK2 in TYPE, PKn..)
```

表 3-19 モバイル DML 操作のパラメータ

パラメータ	説明
DML	各行に対する DML 操作。値は、DELETE の「D」、INSERT の「I」または UPDATE の「U」のいずれかです。
COL1 ... COLn	パブリケーション項目に定義されている列のリスト。列名は、パブリケーション項目の問合せに指定される順序と同じ順序で指定する必要があります。パブリケーション項目が「SELECT * FROM example」を使用して作成されている場合、列の順序は表「example」に指定されている順序と同じである必要があります。
PK1 ... PKn	主キー列のリスト。列名は、実表または親表に指定されている順序と同じ順序で指定する必要があります。

たとえば、次の問合せにより定義されているパブリケーション項目「example」に対して、DML プロシージャが必要になるとします。

```
select A,B,C from publication_item_example_table
```

「A」が「example」の主キー列とすると、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)
```

実行時に、このプロシージャは、DML タイプの「I」、「U」または「D」でコールされます。挿入および削除操作の場合、A_OLD は NULL になります。更新の場合は、更新される行の主キーに設定されます。PL/SQL プロシージャを定義した後は、次の API コールを使用してプロシージャをパブリケーション項目に連結できます。

```
Consolidator.AddMobileDmlProcedure("PUB_example","example","any_schema.any_
package.any_name")
```

`example` はパブリケーション項目名で、`PUB_example` はパブリケーション名です。

この API コールの詳細は、『Oracle9*i* Lite Consolidator Admin API リファレンス』を参照してください。

3.4.4.1 PL/SQL コードの例

次の PL/SQL コード（一部）は、サンプル・パブリケーションのパブリケーション項目に対する実際の DML プロシージャを定義します。次に説明された ORD_MASTER 表を使用して、問合せは次のように定義されています。

SQL 文

```
SELECT * FROM ord_master", where ord_master has a single column primary key on "ID"
```

ord_master 表

```
SQL> desc ord_master
Name                                      Null?    Type
----------------------------------------- -------- -------------
ID                                        NOT NULL NUMBER(9)
DDATE                                              DATE
STATUS                                             NUMBER(9)
NAME                                               VARCHAR2(20)
DESCRIPTION                                        VARCHAR2(20)
```

コード例

```
CREATE OR REPLACE  PACKAGE "SAMPLE11"."ORD_UPDATE_PKG"  AS
 procedure  UPDATE_ORD_MASTER(DML CHAR,ID NUMBER,DDATE DATE,STATUS
NUMBER,NAME VARCHAR2,DESCRIPTION VARCHAR2, ID_OLD NUMBER);
END ORD_UPDATE_PKG;
/
CREATE OR REPLACE  PACKAGE BODY "SAMPLE11"."ORD_UPDATE_PKG" as
  procedure  UPDATE_ORD_MASTER(DML CHAR,ID NUMBER,DDATE DATE,STATUS
NUMBER,NAME VARCHAR2,DESCRIPTION VARCHAR2, ID_OLD NUMBER) is
  begin
    if DML = 'U' then
     execute immediate 'update ord_master set id = :id, ddate = :ddate,
status = :status, name = :name, description = '||''''||'from
ord_update_pkg'||''''||' where id = :id_old'
      using id,ddate,status,name,id_old;
    end if;
    if DML = 'I' then
 begin
      execute immediate 'insert into ord_master values(:id, :ddate,
:status, :name, '||''''||'from ord_update_pkg'||''''||')'
        using id,ddate,status,name;
 exception
  when others then
   null;
 end;
    end if;
    if DML = 'D' then
     execute immediate 'delete from ord_master where id = :id'
      using id;
    end if;
  end UPDATE_ORD_MASTER;
end ORD_UPDATE_PKG;
/
```

この DML プロシージャを追加する API コールは、次のとおりです。

```
Consolidator.AddMobileDMLProcedure("T_SAMPLE11","P_SAMPLE11-M","SAMPLE11.ORD_UPDATE_
PKG.UPDATE_ORD_MASTER")
```

T_SAMPLE11 はパブリケーション名で、P_SAMPLE11-M はパブリケーション項目名です。

3.4.5 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。仮想主キーは 1 つ以上の列に対して作成できますが、仮想主キーを割り当てるそれぞれの列に対して API を個別にコールする必要があります。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。

3.4.5.1 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 クラスの使用方法

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 3-20 仮想主キー列作成パラメータ

パラメータ	値	説明
owner	SAMPLE1	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	主キー列を示す文字列です。

Java メソッドの例

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

3.4.5.2 仮想主キー列の削除

次の方法で、仮想主キーを削除できます。

Consolidator クラスの使用方法

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

表 3-21 仮想主キー列の削除パラメータ

パラメータ	値	説明
owner	SAMPLE1	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");
```

3.4.6 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、クライアントにプッシュされるデータを制限するために選択条件が使用されます。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。

3.4.7 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・キューのパージを試みることもできます。

3.4.7.1 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 クラスの使用方法

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

表 3-22 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。これはエラー・キューに示される生成済み文字列です。

Java メソッドの例

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

3.4.7.2 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 クラスの使用方法

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

表 3-23 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。これはエラー・キューに示される生成済み文字列です。

Java メソッドの例

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

3.5 Mobile Sync for Windows の設定

Oracle データベースと Windows 上で稼働する Mobile サーバーとの間のトランスポートを構成した後、Mobile サーバーとクライアント間の Mobile Sync をインストールし構成する必要があります。

3.5.1 トランスポートの構成

Mobile クライアントは、Mobile サーバーからデータをダウンロードして書式化します。サーバーと Windows の Mobile クライアント間のデータを同期するには、TCP/IP ネットワーク接続を構成する必要があります。

3.5.2 同期のテスト

Oracle_Home¥Mobile¥SDK¥Bin ディレクトリに移り、**msync.exe** を起動して、Mobile クライアント・アプリケーションを起動します。

Mobile Sync では、同期のために次のパラメータが必要です。

表 3-24 Mobile Sync 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。
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。
http://<mobile server>	Mobile サーバーの IP アドレスです。
プロキシを使用	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、『Oracle9*i* 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.exe**」アイコンを実行します。「mSync」画面が表示されます。

図 3-1「mSync」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 IP アドレスが含まれています。
3. 「適用」ボタンをクリックします。
4. 「同期」ボタンをクリックします。

構成、送信、受信および処理の各同期タスクの完了を示す進行状況バーが表示されます。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 サーバーのログ・ファイルで追跡情報を表示できます。

3.6 OCAPI.DLL を使用したプログラミング

ocapi.dll を使用するアプリケーションを Visual Basic または C/C++ を使用して作成する方法は、次の 2 通りがあります。

- 3.6.1 項「Mobile Sync COM API」
- 3.6.2 項「Mobile Sync API」

3.6.1 Mobile Sync COM API

Mobile Sync COM API（MSync COM API）はアプリケーション・プログラミングのインタフェースで、同期プロセスを開始し、様々な設定を有効にします。このインタフェースはモジュール形式で拡張可能であり、ラッパー形式のインタフェースを介して **ocapi.dll** を使用します。このインタフェースは、アプリケーションを主に Visual Basic で作成できるように設計されていますが、他に VBScript、PowerBuilder などのプログラミング手法も COM インタフェースでサポートされています。

3.6.1.1 機能およびコンポーネント

MSync COM API は、次の機能をサポートします。

- 同期プロセスの開始
- 同期プロセスの進行状況の追跡
- ユーザー名、パスワードおよびサーバーなどのデータを含むクライアント側ユーザー・プロファイルの設定
- 表レベルの同期オプションの割当て
- トランスポートの選択

MSync COM API は、Mobile サーバーと Mobile Development Kit がインストールされているシステム上に次の手順でインストールします。

1. ディレクトリを *Oracle_Home*¥Mobile¥SDK¥bin ディレクトリに変更します。
2. 次のコマンドを実行します。**regsvr32/s mSync_com.dll**

これで、MSync COM API とサンプルが *Oracle_Home*¥Mobile¥SDK¥Examples¥msyncCom サブディレクトリにインストールされます。**mSync_com.dll** ライブラリには次のクラスが含まれています。

- 3.6.1.2 項「ISync インタフェース」
- 3.6.1.3 項「ISyncOption インタフェース」
- 3.6.1.4 項「ISyncProgressListener インタフェース」

インタフェースは MSync ライブラリに含まれています。ISync インタフェースを使用するときは、インタフェース名として **MSync.ISync** を使用する必要があります。

3.6.1.2 ISync インタフェース

ISync インタフェース MSync.ISync を使用して、ユーザーは同期プロセスをインスタンス化できます。

表 3-25 ISync インタフェースのパラメータ

名前	説明
HRESULT doSync()	同期プロセスを開始します。同期プロセスが完了するまで、アクセスはブロックされます。

例

次の Visual Basic コードは、デフォルト設定を使用した同期セッションの開始方法を示します。

```
Dim sync As Msync.sync
Set sync = CreateObject("MSync.Sync")
sync.DoSync
```

SyncOption を使用しない場合、インタフェースは最後に保存された情報をロードして同期を実行します。

3.6.1.3 ISyncOption インタフェース

ISyncOption クラス MSync.SyncOption は、同期プロセスのパラメータを定義します。これは手動で構成することも、ロードされたデータまたはユーザー・プロファイルに保存されているデータを使用して構成することもできます。

表 3-26 ISyncOption のパブリック・メソッド

名前	説明
void load()	最後のユーザー同期プロセスのプロファイルをロードします。
void save()	設定をユーザー・プロファイルに保存します。

表 3-27 ISyncOption のパブリック・プロパティ

名前	説明
username	ユーザーの名前。
Password	ユーザーのパスワード。
transportType	使用するトランスポートのタイプ。現時点でサポートされているのは「HTTP」タイプのみです。
transportParam	トランスポートのパラメータ。

例

次の Visual Basic コードは、デフォルト設定を使用した同期セッションの開始方法を示します。

```
Set syncOpt = CreateObject("MSync.SyncOption")
' Load last sync info
syncOpt.Load
' Change user name to Sam
syncOpt.usename = "Sam"
Set sync = CreateObject("MSync.Sync")
' Tell ISync to use this option
sync.setOptionObject (syncOpt)
' Do sync
sync.DoSync
```

3.6.1.4 ISyncProgressListener インタフェース

ISync では接続ポイント・コンテナを実装して、同期進行状況情報を追跡できるようにしています。ISyncProgressListener は、ISync インタフェースから更新を返すように実装する必要があります。

表 3-28 ISyncProgressListener 抽象メソッド

名前	説明
HRESULT progress([in] int *progressType*, int *param1*, int *param2*	新規進行状況情報が使用可能になったときに同期エンジンによりコールされます。progressType は、ISyncProgressListener 定数表に定義されている進行状況タイプ定数のいずれかに設定されます。current は現在までの完了数で、total は最大値です。current が total に等しくなると、その段階が完了します。total および current の単位は、progressType に応じて異なります。

表 3-29 ISyncProgressListener の定数

名前	進行状況タイプ
PT_INIT	同期エンジンが初期化段階であることを示します。current と total の数は両方とも 0 に設定されます。
PT_PREPARE_SEND	同期エンジンが、サーバーに送信するローカル・データを準備中であることを示します。これには、ローカルに変更されたデータの取得が含まれます。ストリーミング実装の場合、これはもっと短くなります。

表 3-29 ISyncProgressListener の定数（続き）

名前	進行状況タイプ
PT_SEND	同期エンジンがネットワークにデータを送信中であることを示します。 total は送信対象のバイト数を示し、current は現在までに送信されたバイト数を示します。
PT_RECEIVE	同期エンジンがサーバーからデータを受信中であることを示します。 total は受信対象のバイト数を示し、current は現在までに受信されたバイト数を示します。
PT_PROCESS_RECV	同期エンジンが、サーバーから新しく受信したデータをローカル・データ・ストアに適用中であることを示します。
PT_COMPLETE	同期エンジンが同期プロセスを完了したことを示します。

例

次の Visual Basic コードは、イベントの報告方法を示します。

```
' Define the ISync object with events
Dim WithEvents sync As MSync.sync

' Create the callback.
' The name of the call back is the name of the ISync object (not the class), and
' underscore and then the function name - progress
Private Sub sync_progress(ByVal progressType As Long, ByVal param1 As Long, ByVal
param2 As Long)
    Desc = ""
    ' Decipher the progressType
    Select Case progressType
        Case PT_SEND
            Desc = "Sending data..."
        Case PT_RECV
            Desc = "Receiving..."
    End Select
End Sub
```

3.6.2 Mobile Sync API

Mobile Sync API は、5 つの C/C++ 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。これらの定義は **ocapi.h** および **ocapi.dll** にあります。この 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から開始および監視でき、Mobile サーバーから開始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。

3.6.2.1 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 構造体を指すポインタがパラメータとして渡されるので、これをコール側で割り当てる必要があります。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。コール側が、ocEnv 構造体のメモリーを割り当てる必要があります。

3.6.2.2 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 により戻された環境構造体へのポインタ。

コメント

ocSessionInit() コールにより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() は常に一対でコールする必要があります。

3.6.2.3 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または上書きします。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、3.6.3 項「Mobile Sync API のデータ構造」を参照してください。

3.6.2.4 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 の場合は、アップロード（送信）操作のみが実行されます。syncDirection が 2 の場合は、ダウンロード（受信）操作のみが実行されます。クライアント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便利です。

3.6.2.5 ocSetTableSyncFlag

選択同期用の表フラグを更新します。表のそれぞれに対してこれをコールし、次のセッションで表を同期する（1）かしない（0）かを指定します。このオプションを使用する場合は、ocDoSync の前に使用する必要があります。

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 の一部である CreatePublication() で使用されます。詳細は、『Oracle9*i* 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 の用語では、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、『Oracle9*i* 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。

3.6.3 Mobile Sync API のデータ構造

Mobile Sync API に含まれるデータ構造体には、ocEnv と ocTransportEnv の 2 つがあります。

3.6.3.1 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、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 {
                        // ユーザー情報
char username[MAX_USERNAME];      // Mobile Sync クライアント ID
char password[MAX_USERNAME];      // 同期中の認証のための
                        // Mobile Sync クライアント・パスワード
char newPassword[MAX_USERNAME];      // サーバー側での Mobile Sync クライアント・パスワードの
                        // 再設定（フィールドが空白の場合）
short savePassword;       // 1 に設定した場合、' パスワード ' を保存
char appRoot[MAX_PATHNAME];      // クライアント・デバイス上の、ファイル配布のためのディレクトリ・パス
short priority;      // 高優先順位表のみかどうか
short secure;     // 1 に設定した場合、ワイヤ上でデータを暗号化
enum {
OC_SENDRECEIVE = 0,      // 同期の全ステップ
OC_SENDONLY,     // 送信フェーズのみ
OC_RECEIVEONLY,      // 受信フェーズのみ

                        // Palm のみ
OC_SENDTOFILE,      // ローカル・ファイル | pdb への送信
OC_RECEIVEFROMFILE      // ローカル・ファイル | pdb からの受信
}syncDirection;      // 同期方向

enum {
OC_BUILDIN_HTTP = 0,      // 組込みの HTTP 転送方法を使用
OC_USER_METHOD      // ユーザー定義の転送方法を使用
}trType;            // 転送のタイプ
```

```
ocError exError;      // 特別なエラー・コード

ocTransportEnv transportEnv;      // 転送制御情報

                        // GUI 関連機能のエントリ
progressProc fnProgress;      // 追跡の進捗状況へのコールバック。これはオプションです。

                  // 進捗状況バーに使用される値。0 の場合、進捗状況バーは表示されません。
long totalSendDataLen;       // Mobile Sync API で設定され、トランスポートに送信の総バイト数を
                        // 通知します。最初の fnSend() がコールされる前に設定します。
long totalReceiveDataLen;      // トランスポートで設定され、Mobile Sync API に受信の
                        // 総バイト数を通知します。
                        // 最初の fnReceive() のコールで設定します。
void* userContext;      // ユーザー定義のコンテキスト
void* ocContext;      // 内部使用専用
short logged;      // 内部使用専用
long bufferSize;      // 送信 / 受信のバッファ・サイズ。デフォルトは 0
short pushOnly;      // プッシュのみのフラグ
short syncApps;      // アプリケーション配布のフラグ
} ocEnv;
```

3.6.3.1.1 ocTransportEnv この構造体は、組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;             // 転送の内部コンテキスト
                            // 組込み HTTP 用、ocTrHttp にマップ
connectProc fnConnect;      // デバイスからサーバーへの接続を確立する
                            // プラグイン・コールバック
disconnectProc fnDisconnect;      // デバイスからサーバーへの接続を切断する
                            //  プラグイン・コールバック
sendProc fnSend;            // データ送信のプラグイン・コールバック
receiveProc fnReceive;      // データ受信のプラグイン・コールバック
}ocTransportEnv;
```

3.7 Mobile サーバーのシステム・カタログ・ビュー

Mobile サーバーには、Mobile サーバー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。

- 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
- 順序
- 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

Mobile サーバーは、Oracle データベース・システムのシステム・カタログの表を使用して、前述の項目を定義します。ユーザー、開発者および管理者は、これらの表を手動で変更しないでください。適切な API を使用して変更するようにします。システム・カタログ・ビューは、データの表示用に提供されています。

詳細は、B 項「システム・カタログ・ビュー」を参照してください。

4

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。

4.1 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。

- 新規 Mobile サーバー・アプリケーションの作成
- 既存の Mobile サーバー・アプリケーションの編集
- 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

新規 Mobile アプリケーションを作成するときは、そのコンポーネントとファイルを定義します。場合によっては、既存の Mobile 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。このファイルはコントロール・センターを使用してパブリッシュできます。パッケージ・ウィザードの 2 番目の用途は、実表を作成するために Oracle データベースに対して実行する SQL スクリプトの作成です。

4.2 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。

図 4-1「ようこそ」パネル

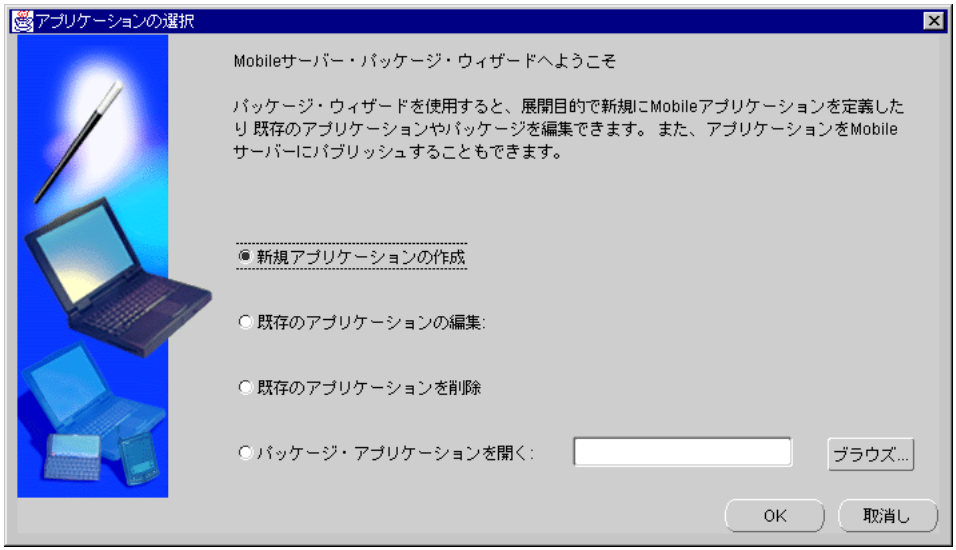


表 4-1「ようこそ」パネルのオプション

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。
既存のアプリケーションを削除	このオプションは、指定されたアプリケーションへの参照をファイルからすべて削除します。以前にパブリッシュされたアプリケーションの場合は、Mobile サーバー・リポジトリから削除しません。これは、コントロール・センターを使用して行う必要があります。
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」ボタンを使用して編集対象のアプリケーションを検索します。

4.3 プラットフォームの選択

この画面を使用して、アプリケーションをパッケージ化する対象プラットフォームを選択できます。プラットフォームは最低 1 つ選択する必要があります。アプリケーションが 2 種類以上のクライアントで実行される場合は、複数を選択できます。上のリストの「使用可能プラットフォーム」で選択するプラットフォームをハイライトし、左にある下向き矢印ボタンを使用してそのプラットフォームを「選択済プラットフォーム」に移動します。

図 4-2 プラットフォームの選択

4.4 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。

図 4-3「アプリケーション」パネル

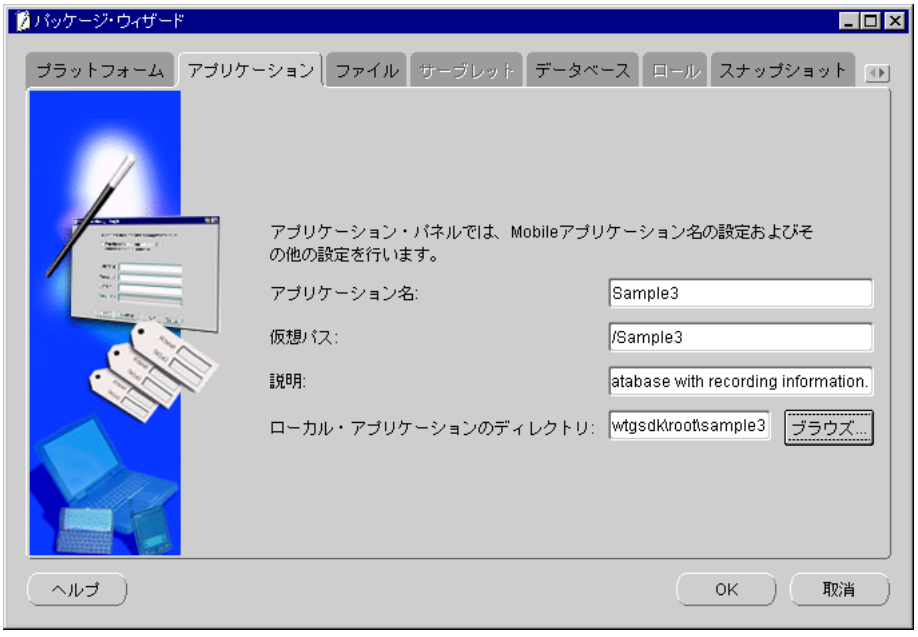


表 4-2「アプリケーション」パネルのオプション

フィールド	説明	必須
仮想パス	これは、アプリケーションに対して一意の ID を提供します。Mobile サーバー・リポジトリのルート・ディレクトリからアプリケーション自体の場所にマップされるパスです。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。	○
アプリケーション名	Mobile サーバーにログインするときに表示されるアプリケーションの名前。	○

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
説明	Windows アプリケーションの簡単な説明。	○
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。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。このディレクトリに必要な形式は、4.4.1 項「プラットフォーム・ファイルの検索」を参照してください。	○

4.4.1 プラットフォーム・ファイルの検索

ローカル・アプリケーションのディレクトリが必要です。アプリケーションに Win32、Palm、EPOC または Windows CE のファイルが含まれている場合は、ローカル・アプリケーション・ディレクトリの次のサブディレクトリにそのファイルを入れます。

- Windows 32 アプリケーションのファイルは、「win32」というサブディレクトリに入れます。
- Windows CE アプリケーションのファイルは、「wince」というサブディレクトリに入れます。
- Palm アプリケーションのファイルは、「palm」というサブディレクトリに入れます。
- EPOC アプリケーションのファイルは、「epoc」というサブディレクトリに入れます。

特定のデバイス用のディレクトリに入れられないファイルは、Web-to-Go アプリケーションに使用されるものと想定されます。Web-to-Go ファイルには特定のディレクトリは不要です。このファイルはローカル・アプリケーション・ディレクトリのルート・レベルにも入れられます。

Mobile サーバーでは、同一アプリケーションの複数のバージョンを Mobile サーバー・ディレクトリにパブリッシュおよび管理できます。これは、同じアプリケーションに複数の実装が存在し、それぞれが Oracle データベース・サーバー内の同一のアプリケーション表にアクセスすることを意味します。たとえば、Windows 32 と Compaq iPAQ 両用の C/C++ アプリケーションを持つこともできます。C++ ソース・コードはその一部またはすべてを再利用できる場合がありますが、Windows 32 用と iPAQ 用にそれぞれファイルを再コンパイルし、異なる実行可能ファイルを作成する必要があります。同一アプリケーションに 2 つのバージョンが存在し、それぞれが同一のデータベース表を使用します。アプリケーション・ファイルは、個別の名前を持つ専用サブディレクトリに格納することが重要です。ローカル・アプリケーション・ディレクトリは Windows 開発システム上のディレクトリで、アプリケーションの複数のバージョンが格納される場所です。パッケージ・ウィザードは、このアプリケーション（ルート）・ディレクトリの下にあるアプリケーション・ファイルを再帰的に読み込みます。

例

'Applications' というローカル・アプリケーション・ディレクトリに、アプリケーションの複数のバージョンが格納されます。

C:¥Applications

Windows 32 用の実行可能ファイルは、**¥win32** サブディレクトリの下に格納する必要があります。

C:¥Applications¥win32

iPAQ 用の実行可能ファイルは、¥wince¥Pocket_PC¥us¥arm サブディレクトリの下に格納する必要があります。

C:¥Applications¥wince¥Pocket_PC¥us¥arm

ローカル・アプリケーション・ディレクトリは **C:¥Applications** ですが、これにはサブディレクトリが 2 つあります。

C:¥Applications －これは、「ローカル・アプリケーション・ディレクトリ」フィールドに入力する必要がある文字列です。このフィールドにサブディレクトリを追加すると、パッケージ・プロセスが失敗します。

C:¥Applications¥win32 －これは、Windows 32 バージョンのアプリケーション・サブディレクトリです。

C:¥Applications¥wince¥Pocket_PC¥us¥arm －これは、CE の StrongArm バージョンのアプリケーション・サブディレクトリです。

Windows 32 アプリケーションの場合、必要なサブディレクトリは **¥win32** のみです。他のプラットフォーム用のディレクトリの一覧は、それぞれの該当する開発者ガイドを参照してください。

4.5 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。

図 4-4「ファイル」パネル

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
「ファイル名」のエントリ	各 Mobile サーバー・アプリケーション・ファイルの絶対パス。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。	○
ファイルのソート	■ 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。	

「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、パッケージ・ウィザードがファイルを自動的に分析し、「ファイル」パネルの表示でローカル・ディレクトリにリストされているファイルのみがロードされます。パッケージ・ウィザードに認識されるのは、適切なサブディレクトリ（たとえば、**¥win32**）に配置されているファイルのみです。他の方法で追加されたファイルではエラー・メッセージが生成されます。

4.5.1 ソート

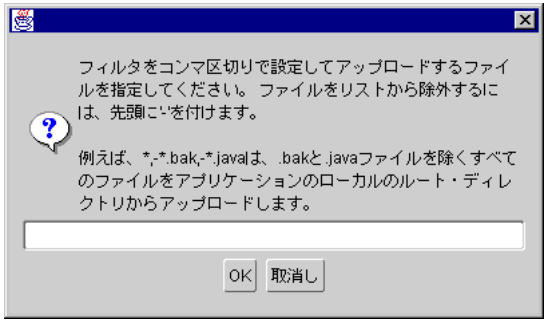
ファイルは、拡張子または含まれているディレクトリごとにソートできます。ファイルをソートするには、「拡張子順」または「ディレクトリ順」ラジオ・ボタンをクリックします。

4.5.2 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、**.bak** および **.jar** 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。

,-.bak, -*.jar

図 4-5「フィルタ」パネル



4.6 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のレプリケート元のデータベースを指定するために使用します。

図 4-6「データベース」パネル

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データベースのユーザー名。	○
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。空白のまま何も指定しないと、自動的に名前が生成されます。	○

4.7 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・スナップショット定義を作成するために使用します。スナップショット名は、全アプリケーション間で一意にする必要があります。

このパネルに含まれるフィールドは次のとおりです。

図 4-7「スナップショット」パネル

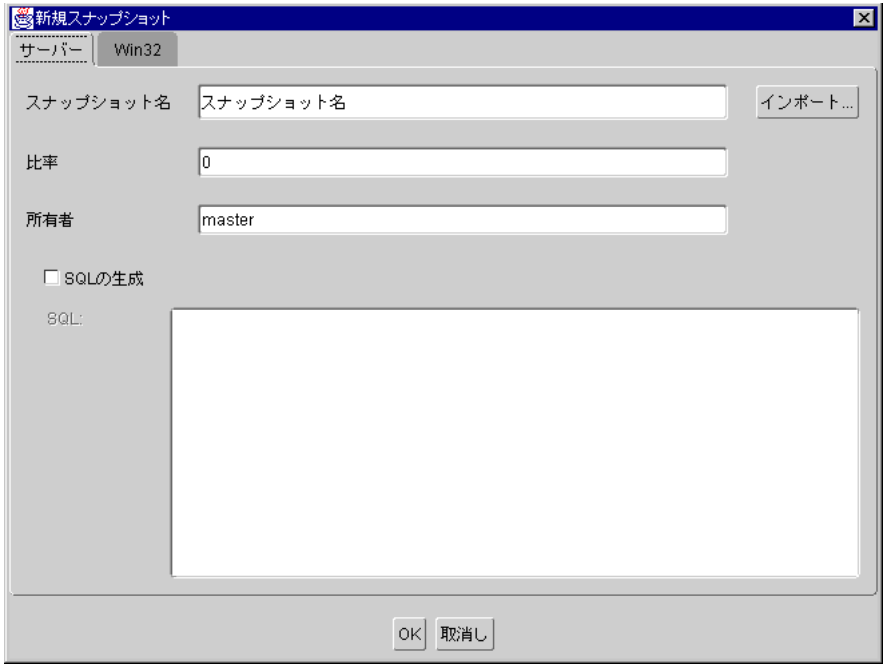


表 4-5「スナップショット」パネルのパラメータ

フィールド	説明	必須
全プラットフォーム	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。 ■ Win32 ■ Palm ■ EPOC ■ Windows CE 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。	×
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定義の名前です。この名前は、既存のデータベース表と同じ名前か、Oracle Lite データベース上に作成する必要があります。	○
テンプレート	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・センターでスナップショット定義テンプレートを変更することはできません。	○
プラットフォーム	スナップショット定義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定義を作成できます。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定義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 / ディテール関係を持つ表の場合、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。	○

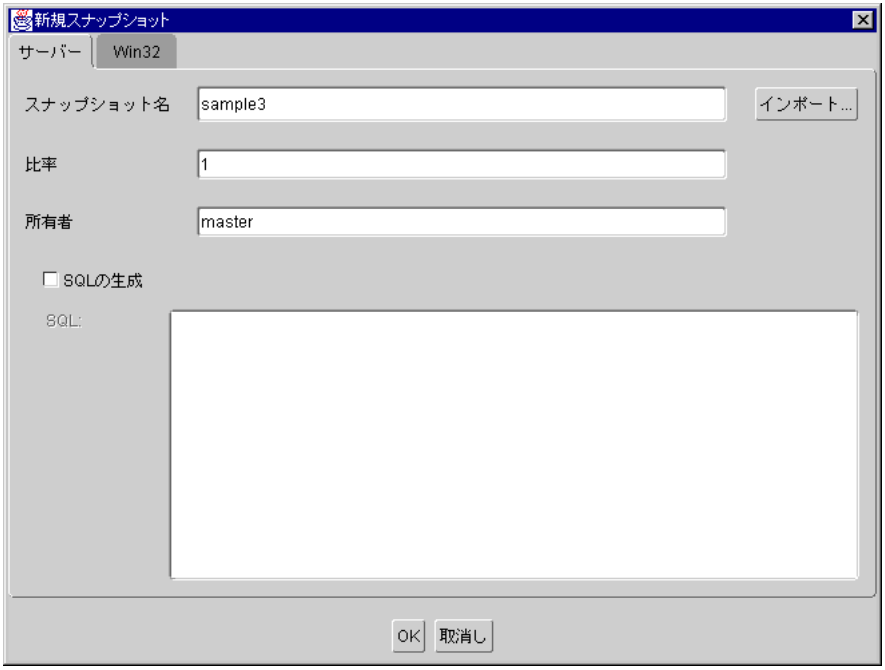
「スナップショット」パネルで「新規」ボタンまたは「削除」ボタンをクリックすると、「スナップショット」パネルにスナップショットを追加または削除できます。スナップショットはインポートまたは編集することもできます。

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。

4.7.1 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」画面が表示されます。

図 4-8「新規スナップショット」ウィンドウ



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

表 4-6「新規スナップショット」ウィンドウのオプション

機能	説明
プラットフォーム	タブには、「プラットフォーム」画面での選択に基づくプラットフォームが表示されます。
スナップショット名	スナップショット定義の基になるデータベース・サーバー表の名前です。
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 SQL スクリプトへの出力情報が収集されます。この SQL スクリプトは、Mobile サーバーに関連付けられているデータベース上にデータベース表を作成するために使用できます。データベースに実表が存在し、SQL スクリプトを使用して実表を作成する必要がない場合は、このチェックボックスの選択を解除します。
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、3.3.8.3 項「表の比率の使用」を参照してください。
SQL	名前付きの表を定義する SQL の CREATE TABLE 文を表示します。この文は変更できます。「SQL の生成」ボックスが選択されている場合は、作成される SQL スクリプトにこの SQL 文が含められます。

4.7.2 スナップショットのインポート

Oracle 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。Oracle データベースと Oracle Lite データベースを切り替える必要があり、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 4-9「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 Oracle 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。「OK」をクリックして続行します。「表」ウィンドウが表示されます。

注意： Oracle 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。

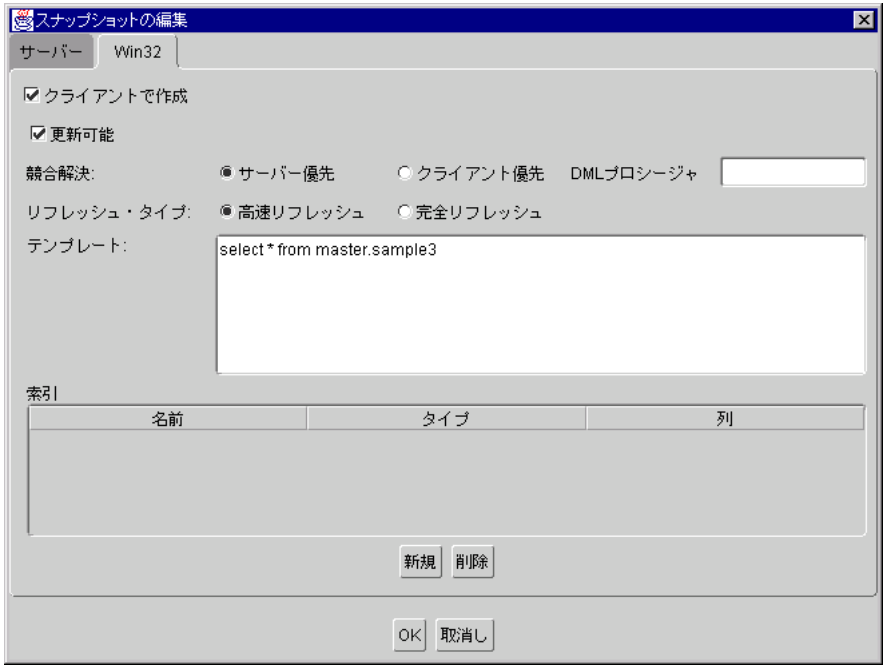
図 4-10「表」ウィンドウ

表のインポート元のパブリケーション項目を選択してから、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。

4.7.3 スナップショットの編集

スナップショット定義を編集するには、「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。最初に選択したプラットフォームがタブに表示されます。スナップショットがまったく同じ場合でも、プラットフォームごとにタブを使用してスナップショットを定義する必要があります。

図 4-11「スナップショットの編集」ウィンドウ



スナップショット定義を編集するには、「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

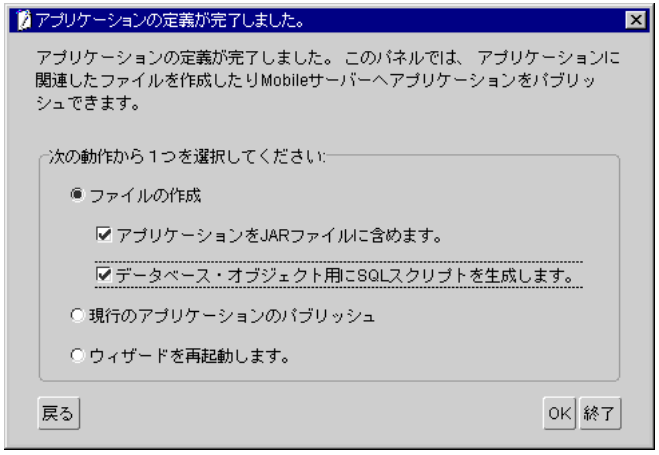
機能	説明
クライアントで作成	このチェックボックスが選択されていると、次のことを実行できます。 ■ 更新可能スナップショットの作成。 ■ スナップショット・テンプレートの作成。管理者は、Mobile サーバー・コントロール・センターを使用して、複数の異なるユーザー用変数をこのテンプレートに対してインスタンス化できます。
更新可能	このチェックボックスは、更新可能として作成されるスナップショットを定義します。
競合解決	このオプションは、すべての競合解決でサーバーが優先するかクライアントが優先するかを定義します。デフォルト設定は「サーバー優先」です。競合解決の詳細は、3.1.8 項「レプリケーション・エラーと競合」を参照してください。
DML プロシージャ	このフィールドは、次の形式で DML プロシージャを指定するために使用できます。 `AnySchema.AnyPackage.AnyName` DML プロシージャを追加すると、「競合解決」オプションの選択が無効になります。 DML プロシージャの作成方法の詳細は、3.4.4 項「カスタマイズ DML 操作の定義」を参照してください。
リフレッシュ・タイプ	このオプションには次の 2 つの選択肢があります。 ■ 高速リフレッシューデフォルトです。変更されたデータのみが転送されます。 ■ 完全リフレッシューデータはすべてリフレッシュされます。
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、4.7 項「レプリケーション用スナップショットの定義」を参照してください。

4.8 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、次のオプションを含む「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 4-12「アプリケーションの定義が完了しました。」ウィンドウ



4.8.1 アプリケーション・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ファイルに自動的に保存します。パッケージ・ウィザードはローカル・マシン上にこのファイルを保持します。さらに、Mobile サーバーにこの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。アプリケーション・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。

4.8.2 JAR ファイルの作成

アプリケーションをパッケージ化した後は、パッケージ・ウィザードにより **.jar** ファイルが作成されます。Mobile サーバー・インスタンスに対する管理権限を持つユーザーなら、だれでも、コントロール・センターを使用して Mobile サーバー・リポジトリにパブリッシュできます。

「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。

4.8.3 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・ディレクトリの下の SQL サブディレクトリ内に入れられます。SQL スクリプトは、スナップショットおよび順序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。

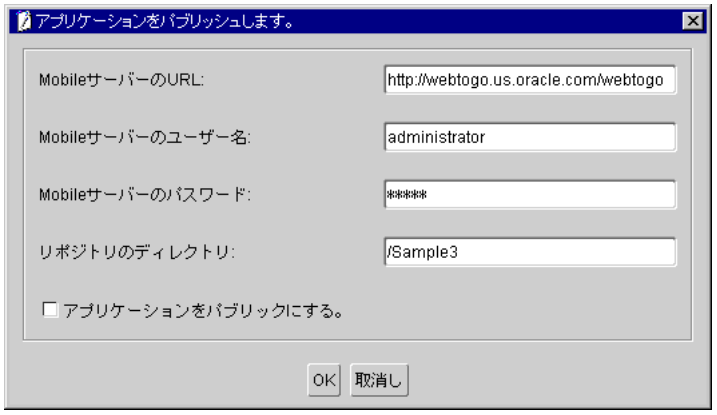
4.8.4 パッケージ・ウィザードの再起動

「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。

4.8.5 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 4-13「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。 http://*mobileserver*:*port*/webtogo *mobileserver* は Mobile サーバーのホスト名で、*port* は TCP/IP ポートです。デフォルト・ポートはポート 80 です。	○
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。	○
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。	○
リポジトリのディレクトリ	Mobile サーバー・リポジトリの宛先ディレクトリです。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。	○
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。	×

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。

4.9 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。

A

POLITE.INI のデータベース・パラメータ

POLITE.INI ファイルに定義されているデータベース・パラメータ値を変更すると、Oracle Lite データベースをカスタマイズできます。この章では、**POLITE.INI** ファイルとその関連パラメータについて説明します。内容は次のとおりです。

A.1 POLITE.INI ファイルの概要

POLITE.INI ファイルにより、データベース・ボリューム ID の割当てが一元化され、システム上の全データベースのパラメータが定義されます。Oracle Lite をインストールすると、**POLITE.INI** ファイルが Windows 95/98/NT または Windows 2000 のホーム・ディレクトリに作成されます。

POLITE.INI ファイルのパラメータは、インストール時に自動設定されますが、それらを変更して、製品動作をカスタマイズできます。**POLITE.INI** ファイルの変更には、ASCII テキスト・エディタを使用します。

A.2 POLITE.INI のパラメータ

次の項以降で、**POLITE.INI** ファイルの *All Databases* セクションに指定するパラメータを示します。

A.2.1 CacheSize

オブジェクト・キャッシュのサイズを KB 単位で指定します。最小値は 128 です。指定がない場合は、デフォルトの 4096（4MB）が使用されます。

A.2.2 DatabaseID

CREATE DATABASE SQL コマンドが次に割り当てるデータベース・ボリューム ID 番号を定義します。システム上の各データベース・ファイルに対するデータベース・ボリューム ID 番号は、一意のものである必要があります。

A.2.3 DbCharEncoding

Oracle Lite により実行される UTF 変換を指定します。NATIVE に設定すると、UTF 変換は実行されません。UTF8 に設定すると、UTF 変換が実行されます。このパラメータを指定しない場合、デフォルトは UTF8 です。これは、Java プログラムにのみ当てはまります。

A.2.4 MAXINDEXCOLUMNS

索引作成文で使用される列数を定義します。詳細は、『Oracle9*i* Lite SQL リファレンス』の「索引作成オプション」を参照してください。

A.2.5 NLS_Date_Format

Oracle Lite のデフォルト以外の日付書式を使用できるようになります。日付値が必要な場所にリテラル文字列が指定されると、Oracle Lite データベースはその文字列をテストして、その文字列が、Oracle または SQL-92 の書式あるいは **POLITE.INI** ファイルでこのパラメータに対して指定されている値の書式に一致するかどうかを調べます。このパラメータを設定すると、TO_CHAR または TO_DATE 関数で他の書式文字列が指定されていない場合に使用されるデフォルト書式も定義されます。

Oracle ではデフォルトは dd-mon-yy または dd-mon-yyyy で、SQL-92 ではデフォルトは yy-mm-dd または yyyy-mm-dd です。

書式に「RR」を使用すると、49 以下の 2 桁の年表記は 21 世紀（2000 ～ 2049）と解釈され、50 以上の年表記は 20 世紀（1950 ～ 1999）と解釈されます。すべての 2 桁年エントリのデフォルトとして RR 書式を設定すると、西暦 2000 年対応になります。たとえば、次のようになります。

```
NLS_DATE_FORMAT='RR-MM-DD'
```

日付書式は、ALTER SESSION コマンドを使用して変更することもできます。詳細は、『Oracle9*i* Lite SQL リファレンス』を参照してください。

A.2.5.1 日付書式

日付書式には、次の表にリストされている要素が 1 つ以上含まれています。同じような情報を表す要素を組み合せることはできません。たとえば、「SYYYY」と「BC」を同一のフォーマット文字列で使用することはできません。

表 A-1 日付書式

書式	説明
AM または A.M.	正午標識。ピリオドはオプションです。
PM または P.M.	正午標識。ピリオドはオプションです。
CC または SCC	世紀。「S」によって紀元前の日付の先頭に「-」が付けられます。
D	曜日（1 ～ 7）。
Day	曜日。9 文字になるまでブランクが埋め込まれます。
DD	月間通算日（1 ～ 31）。
DDD	年間通算日（1 ～ 366）。
DY	曜日の省略形。
IW	ISO 規格に基づく年間通算週（1 ～ 52 または 1 ～ 53）。
IYY、IY または I	ISO 年表記の最後の 3 桁、2 桁または 1 桁。

表 A-1 日付書式（続き）

書式	説明
IYYY	ISO 規格に基づく 4 桁の年表記。
HH または HH12	時（1 ～ 12）。
HH24	時（0 ～ 23）。
MI	分（0 ～ 59）。
MM	月（01 ～ 12。たとえば、JAN=01）。
MONTH	月の名前。9 文字になるまでブランクが埋め込まれます。
MON	月の名前の省略形。
Q	四半期（1、2、3、4。たとえば、JAN ～ MAR=1）。
RR	年号の最後の 2 桁。外国での年号。書式に「RR」を使用すると、49 以下の 2 桁の年表記は 21 世紀（2000 ～ 2049）と解釈され、50 以上の年表記は 20 世紀（1950 ～ 1999）と解釈されます。
WW	年間通算週（1 ～ 53）。第 1 週は、その年の 1 月 1 日で始まり、1 月 7 日で終了します。
SS	秒（0 ～ 59）。
SSSSS	午前 0 時以降の秒数（0 ～ 86399）。
Y,YYY	指定した位置にカンマの付いた年表記。
YEAR または SYEAR	綴りで表した年表記。「S」によって紀元前の日付の先頭に「-」が付けられます。
YYYY または SYYYY	4 桁の年表記。「S」によって紀元前の日付の先頭に「-」が付けられます。
YYY、YY または Y	年表記の最後の 3 桁、2 桁または 1 桁。

A.2.5.2 日付書式の例

次のリストは、NLS_DATE_FORMAT パラメータの例です。

```
NLS_DATE_FORMAT= format
```

YYYY-MONTH-DAY:HH24:MI:P.M.
YYYY/MONTH/DD, HH24:MI A.M.
YYYY-MONTH-DAY:HH24:MI:P.M.
MM D, YYY, HH:MI A.M.
MM, WW, RR, HH:MI A.M.
MM, IW, RR, HH:MI A.M.
MM, DY, RR, HH:MI A.M.
MM; DY; IYY, HH:MI A.M.
MON WW, RR, HH:MI A.M.
MONTH.DD, SYYYY, HH:MI A.M.
MONTH/DD, YYYY, HH:MI A.M.
MONTH|DD, YYYY, HH:MI A.M.
MONTH DD, YYYY, HH:SSSSS:MI A.M.
MONTH DD, HH:SS:MI CC
MONTH DD, HH:SS:MI SCC
MONTH W, YYYY, HH:MI A.M.
MONTH WW, YYYY, HH:MI A.M.
MONTH WW, RR, HH:MI A.M.
MONTH WW, Q, HH:MI A.M.
MONTH WW, RR, HH:MI A.M.

A.2.6 NLS_Locale

Oracle Lite の言語依存の動作を指定するには、**POLITE.INI** ファイルで NLS_Locale パラメータを定義します。NLS_Locale の値は、次の書式です。

```
language_territory.codepage
```

territory と *codepage* はオプションです。指定しないと、デフォルト値が使用されます。

たとえば、次のように指定します。

```
NLS_LOCALE=FRENCH_FRANCE
```

この例の場合、言語が日本語で、地域が日本となります。

月や日の名前、その省略形や数値による書式は、言語に基づいてカスタマイズされます。サポートされている言語は、次のとおりです。

AMERICAN	FINNISH	MEXICAN SPANISH
BRAZILIAN PORTUGUESE	FRENCH	NORWEGIAN
BULGARIAN	GERMAN	POLISH
CANADIAN FRENCH	GREEK	PORTUGUESE
CATALAN	HEBREW	ROMANIAN
CROATIAN	HUNGARIAN	RUSSIAN
CZECH	ICELANDIC	SLOVAK
DANISH	ITALIAN	SLOVENIAN
DUTCH	JAPANESE	SPANISH
EGYPTIAN	LATIN AMERICAN SPANISH	SWEDISH
ENGLISH	LITHUANIAN	TURKISH
ESTONIAN	MALAY	UKRAINIAN

A.2.7 NLS_SORT

このパラメータは、Oracle Lite のインスタンスに対して作成されたデータベースの照合順序を定義するために使用できます。照合とは、言語に受入れ可能な順序で文字列を並べることです。照合順序とは、アルファベットの全照合要素の最小から最大までの順序です。次のパラメータを使用したとします。

```
NLS_SORT=[collation sequence]
```

このとき、CREATEDB コマンドライン・ユーティリティで作成されたデータベースまたは Mobile サーバーからレプリケートされたデータベースでは、ユーティリティの使用時に別の照合順序が指定されないかぎり、すべてこの照合順序が有効になります。現時点でサポートされている言語は、BINARY（デフォルト）、FRENCH、GERMAN、CZECH および XCZECH です。

注意： サポートされている照合順序に対応した言語ソートをすべてのデータベースで有効にする必要がある場合を除き、NLS_SORT <collation sequence> パラメータを指定して CREATEDB ユーティリティを使用することをお薦めします。このパラメータは **polite.ini** のパラメータをオーバーライドします。**polite.ini** ファイルを使用した NLS_SORT の設定は、指定された照合順序がすべてのデータベースで有効になっていることを意味します。現時点で、データベースの照合順序を切り替える方法はありません。

この機能の詳細は、C.2 項「CREATEDB」を参照してください。

A.2.8 ReverseJoinOrder

問合せで表を結合する順序を指定します。オプションは TRUE または FALSE です。TRUE の場合は、FROM 句における順序とは反対の順序で表が結合されます。FALSE の場合は、FROM 句における順序と同じ順序で表が結合されます。指定されていない場合は、Oracle Lite の問合せオプティマイザにより、最適な結合順序が決定されます。このオプションにより、すべての問合せに対して JOIN 最適化が使用禁止になるため、上級ユーザーによる使用をお薦めします。1 つの問合せを最適化する場合は、このかわりに HINTS の使用をお薦めします。詳細は、『Oracle9*i* Lite SQL リファレンス』を参照してください。

A.2.9 SharedAddress

Oracle Lite では、アプリケーションにまたがって必要なデータは、共有メモリーを使用して管理します。Oracle Lite は、プロセス・メモリーの特定の位置に共有メモリーを連結します。ごくまれに、この位置が他のツールによりすでに使用中のことがあり、その場合はエラーが発生します。この問題に対応するために、Oracle Lite では次の規則をサポートして、共有メモリー用に連結するメモリー・アドレスを判断します。

共有メモリーを連結する前に、Oracle Lite は SharedAddress 変数と 16 進の 32 ビット・アドレス（たとえば、18000000）を指定する SuggestedSharedAddress 変数を調べます。Oracle Lite は、見つかった最初の値を使用します。いずれの変数も設定されていない場合、Oracle Lite はアドレス 30000000 を最初に試します。この値は、ほとんどのアプリケーションで使用されている範囲より上位にあります。

Oracle Lite クライアントがすでに実行中の場合、2 番目のプロセスが同じ共有メモリー・アドレスを取得できないと、このクライアントはエラーで失敗します。ただし、2 番目のプロセスで使用可能なアドレスを SuggestedSharedAddress として **POLITE.INI** ファイルに書き込みます。ここでユーザーが Oracle Lite クライアントをすべて終了し、同じアプリケーション・セットを実行した場合、問題は再発しません。

自動競合解決に失敗した場合は、問題が解決されるまでの間、SharedAddress 変数を変更する必要があります。たとえば、値の間隔を 256MB に設定し、20000000、24000000、28000000 などで試してみます。

A.2.10 SuggestedSharedAddress

説明は、A.2.9 項「SharedAddress」の項を参照してください。

A.2.11 SQLCompatibility

Oracle Lite は、Oracle SQL と SQL-92 の両方の機能をサポートします。Oracle SQL と SQL-92 の詳細は、『Oracle9*i* Lite SQL リファレンス』を参照してください。

Oracle SQL と SQL-92 の間に矛盾がある場合、SQLCompatibility フラグが参照されます。このパラメータに ORACLE を指定すると Oracle SQL が優先され、SQL92 を指定すると SQL-92 が優先されます。このパラメータを **POLITE.INI** に含めないと、デフォルトで Oracle SQL が優先されます。

A.2.12 TempDB

一時データベースは、デフォルトではメイン・メモリー内に作成されます。これにより、一時表の使用が必要な問合せのパフォーマンスが向上します。一時データベースを意図的にファイル・システム内に作成する場合を除いて、*poltempx*.**odb** ファイルは作成されません。セーブポイント情報の格納に使用されることのある ***.slx** ファイルも作成されません。大規模な結果セットを作成する場合は、結果を保持する十分なスワップ領域を用意するか、一時データベースにファイル・オプションを選択する必要があります。

このオプションを含めるには、**POLITE.INI** ファイルに次の構文を使用します。

```
TempDB=<path temporary_database_name>
```

たとえば、次のようになります。

```
TempDB=c:¥temp¥olite_
```

この例のように設定すると、Oracle Lite では次の一時データベースが作成されます。

```
c:¥temp¥olite_0.odb, c:¥temp¥olite_1.odb, ...
```

A.2.13 TempDir

一時データベース **POLTEMP.ODB** が作成されるディレクトリを指定します。設定されていない場合は、各環境で定義されている TEMP、TMP または WINDIR の設定がデフォルトとして使用されます。

A.3 POLITE.INI ファイルのサンプル

次に、**POLITE.INI** ファイルのサンプルを示します。

```
[All Databases]
DatabaseID=128
DBCharEncoding=NATIVE
SuggestedSharedAddress=10270000
CacheSize=4096
MAXINDEXCOLUMNS=5
SQLCompatibility=SQL92
NLS_Date_Format=RR/MM/DD H24,MI,SS
NLS_Locale=ENGLISH
TempDB=c:¥temp¥olite_
TempDir=D:¥TMP
```

B

システム・カタログ・ビュー

この付録では、Oracle Lite データベースのシステム・カタログ・ビューを解説します。次に示すカタログ・ビューについて説明します。

B.1 Oracle Lite データベースのカタログ・ビュー

Oracle Lite データベースのシステム・カタログでは、次のビューを使用できます。

- ALL_COL_COMMENTS
- ALL_CONSTRAINTS
- ALL_CONS_COLUMNS
- ALL_INDEXES
- ALL_IND_COLUMNS
- ALL_OBJECTS
- ALL_SEQUENCES
- ALL_SYNONYMS
- ALL_TABLES
- ALL_TAB_COLUMNS
- ALL_TAB_COMMENTS
- ALL_USERS
- ALL_VIEWS
- CAT
- COLUMN_PRIVILEGES
- DUAL
- DATABASE_PARAMETERS
- SNAPSHOTS
- TABLE_PRIVILEGES
- USER_OBJECTS

注意： 以降に示す表で、アスタリスクの付いた列は Oracle Lite では使用されませんが、Oracle データベースと互換性があり、一般に NULL またはデフォルト値を返します。

B.1.1 ALL_COL_COMMENTS

このビューは、表の列に対するユーザーのコメントをリストします。

表 B-1 ALL_COL_COMMENTS のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)	NOT NULL	表の所有者。
TABLE_NAME	VARCHAR2(128)	NOT NULL	オブジェクトの名前。
COLUMN_NAME	VARCHAR2(128)	NOT NULL	列の名前。
COMMENTS	VARCHAR2(4096)		列のコメントのテキスト。

B.1.2 ALL_CONSTRAINTS

このビューは、アクセス可能な表に対する制約の定義について次の情報を提供します。

表 B-2 ALL_CONSTRAINTS のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)	NOT NULL	制約定義の所有者。
CONSTRAINT_NAME	VARCHAR2(128)	NOT NULL	制約定義に対応付けられた名前。
CONSTRAINT_TYPE	VARCHAR2(128)	NOT NULL	制約定義のタイプ : C（表に対するチェック制約） P（主キー） U（一意キー） R（参照整合性） V（ビューに対するチェック・オプション）
TABLE_NAME	VARCHAR2(128)	NOT NULL	制約定義を持つ表の名前。
SEARCH_CONDITION	VARCHAR2(1000)		表検査のための検索条件のテキスト。
R_OWNER	VARCHAR2(128)		参照制約で使用される表の所有者。
R_CONSTRAINT_NAME	VARCHAR2(128)		参照される表に対する一意制約定義の名前。

表 B-2 ALL_CONSTRAINTS のパラメータ（続き）

列	データ型	NULL	説明
DELETE_RULE	VARCHAR2(128)		参照制約の削除規則：「NO ACTION」
STATUS	VARCHAR2(20)	NOT NULL	制約のステータス：「ENABLED」または「DISABLED」

B.1.3 ALL_CONS_COLUMNS

このビューは、制約定義内のアクセス可能な列について次の情報を提供します。

表 B-3 ALL_CONS_COLUMNS のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)		制約定義の所有者のユーザー名。
CONSTRAINT_NAME	VARCHAR2(128)		制約定義に対応付けられた名前。
TABLE_NAME	VARCHAR2(128)		制約定義を持つ表の名前。
COLUMN_NAME	VARCHAR2(128)		制約定義に指定されている列に対応付けられた名前。
POSITION	NUMBER(10)		定義内での列の元の位置。

B.1.4 ALL_INDEXES

このビューには、表に定義されているすべての索引の説明が含まれています。

表 B-4 ALL_INDEXES のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)	NOT NULL	INDEX 定義の所有者。
INDEX_NAME	VARCHAR2(128)	NOT NULL	INDEX 定義に対応付けられた名前。
TABLE_OWNER	VARCHAR2(128)	NOT NULL	INDEX が定義されている表の所有者。
TABLE_NAME	VARCHAR2(128)	NOT NULL	INDEX 定義を持つ表の名前。

表 B-4 ALL_INDEXES のパラメータ（続き）

列	データ型	NULL	説明
TABLE_TYPE	VARCHAR2(10)		オブジェクトの型。
UNIQUENESS	VARCHAR2(128)	NOT NULL	「UNIQUE」または「NONUNIQUE」を含む文字列。

B.1.5 ALL_IND_COLUMNS

このビューは、データベース内のすべての索引に対する索引キー列をリストします。

表 B-5 ALL_IND_COLUMNS のパラメータ

列	データ型	NULL	説明
INDEX_OWNER	VARCHAR2(128)	NOT NULL	INDEX 定義の所有者。
INDEX_NAME	VARCHAR2(128)	NOT NULL	INDEX 定義に対応付けられた名前。
TABLE_OWNER	VARCHAR2(128)	NOT NULL	INDEX が定義されている表の所有者。
TABLE_NAME	VARCHAR2(128)	NOT NULL	INDEX 定義を持つ表の名前。
COLUMN_NAME	VARCHAR2(128)	NOT NULL	INDEX 定義に指定されている列に対応付けられた名前。
COLUMN_POSITION	NUMBER(10)	NOT NULL	索引定義内での列の位置。

B.1.6 ALL_OBJECTS

このビューには、オブジェクト（表、ビュー、シノニム、索引および順序）の説明が含まれています。

表 B-6 ALL_OBJECTS のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)	NOT NULL	OBJECTS 定義の所有者。
OBJECT_NAME	VARCHAR2(128)	NOT NULL	OBJECTS 定義に対応付けられた名前。
OBJECT_TYPE	VARCHAR2(128)		オブジェクトの型 : TABLE、VIEW、INDEX、SEQUENCE、SYNONYM

表 B-6 ALL_OBJECTS のパラメータ（続き）

列	データ型	NULL	説明
CREATED	DATE		OBJECTS の作成タイムスタンプ。
STATUS	VARCHAR2(128)		OBJECTS のステータス：VALID、INVALID または N/A（常に有効）

B.1.7 ALL_SEQUENCES

このビューは、データベース内のすべての順序の説明をリストします。

表 B-7 ALL_SEQUENCES のパラメータ

列	データ型	NULL	説明
SEQUENCE_OWNER	VARCHAR2(128)	NOT NULL	SEQUENCES 定義の所有者。
SEQUENCE_NAME	VARCHAR2(128)	NOT NULL	SEQUENCES 定義に対応付けられた名前。
MIN_VALUE	NUMBER(10)	NOT NULL	順序の最小値。
MAX_VALUE	NUMBER(10)	NOT NULL	順序の最大値。
INCREMENT_BY	NUMBER(10)	NOT NULL	順序の増分値。

B.1.8 ALL_SYNONYMS

このビューは、データベース内のすべてのシノニムをリストします。

表 B-8 ALL_SYNONYMS のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)		SYNONYMS 定義の所有者。
SYNONYM_NAME	VARCHAR2(128)		SYNONYMS 定義に対応付けられた名前。
TABLE_OWNER	VARCHAR2(128)		SYNONYMS が定義されている表の所有者。
TABLE_NAME	VARCHAR2(128)		SYNONYMS 定義を持つ表の名前。
DB_LINK	VARCHAR2(128)		予約済み。

B.1.9 ALL_TABLES

このビューは、ユーザーがアクセスできる表について次の情報を提供します。

表 B-9 ALL_TABLES パラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)	NOT NULL	表の所有者のユーザー名。
TABLE_NAME	VARCHAR2(128)	NOT NULL	表の名前。
TABLESPACE_NAME	VARCHAR2(128)		この表を含むカタログまたはデータベース・ファイルの名前。
CLUSTER_NAME*	VARCHAR2(128)		この表が属するクラスタの名前（クラスタがある場合）。
PCT_FREE*	NUMBER(10)		ブロック内の空き領域の最小値（パーセント）。
PCT_USED*	NUMBER(10)		ブロック内の使用済み領域の最小値（パーセント）。
INI_TRANS*	NUMBER(10)		トランザクション数の初期値。
MAX_TRANS*	NUMBER(10)		トランザクション数の最大値。
INITIAL_EXTENT*	NUMBER(10)		初期エクステントのサイズ(バイト単位)。
NEXT_EXTENT*	NUMBER(10)		2 次エクステントのサイズ(バイト単位)。
MIN_EXTENTS*	NUMBER(10)		セグメント内で使用できる最小エクステント数。
MAX_EXTENTS*	NUMBER(10)		セグメント内で使用できる最大エクステント数。
PCT_INCREASE*	NUMBER(10)		エクステント・サイズの増分パーセント。
BACKED_UP*	VARCHAR2(1)		最後の変更以降に表がバックアップされているかどうか。
NUM_ROWS*	NUMBER(10)		表の中の行数。
BLOCKS*	NUMBER(10)		表に割り当てられているデータ・ブロック数。

表 B-9 ALL_TABLES パラメータ（続き）

列	データ型	NULL	説明
EMPTY_BLOCKS*	NUMBER(10)		表に割り当てられているデータ・ブロックの中でデータを含まないブロック数。
AVG_SPACE*	NUMBER(10)		表に割り当てられているデータ・ブロック内の平均空き領域（バイト単位）。
CHAIN_CNT*	NUMBER(10)		表の中で、あるデータ・ブロックから別のデータ・ブロックに連鎖されている行の数、または新規ブロックに移行された行で、古い ROWID を保持するためにリンクが必要な行の数。
AVG_ROW_LEN*	NUMBER(10)		表の行の平均長（バイト単位）。

B.1.10 ALL_TAB_COLUMNS

このビューは、ユーザーがアクセスできる表、ビューおよびクラスタの列について次の情報を提供します。

表 B-10 ALL_TAB_COLUMNS のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)	NOT NULL	表、ビューまたはクラスタの所有者のユーザー名。
TABLE_NAME	VARCHAR2(128)	NOT NULL	表、ビューまたはクラスタの名前。
COLUMN_NAME	VARCHAR2(128)	NOT NULL	列の名前。
DATA_TYPE	VARCHAR2(30)		列のデータ型。
DATA_LENGTH	NUMBER(10)		列の長さ（バイト単位）。
DATA_PRECISION	NUMBER(10)		NUMERIC および DECIMAL データ型の場合は 10 進精度、FLOAT、REAL および DOUBLE データ型の場合はバイナリ精度、その他すべてのデータ型では NULL。

表 B-10 ALL_TAB_COLUMNS のパラメータ（続き）

列	データ型	NULL	説明
DATA_SCALE	NUMBER(10)		NUMERIC または DECIMAL データ型での小数点以下の桁数。
NULLABLE	VARCHAR2(1)		列で NULL を使用できるかどうかを示します。列に NOT NULL 制約が指定されている場合、または列が主キーの一部である場合、値は N です。
COLUMN_ID	NUMBER(10)	NOT NULL	作成された時点での列の順序番号。
DEFAULT_LENGTH	NUMBER(10)		列のデフォルト値の長さ。
DATA_DEFAULT	VARCHAR2(4096)		列のデフォルト値。
NUM_DISTINCT*	NUMBER(10)		表の各列内の個別値の数。
LOW_VALUE*	NUMBER(10)		*HIGH_VALUE* の説明を参照してください。
HIGH_VALUE*	NUMBER(10)		4 行以上の表の場合は、列内で 2 番目に低い値および 2 番目に高い値。3 行以下の表の場合は、1 番低い値および 1 番高い値。この統計値は、値の最初の 32 バイトの内部表記の 16 進表記で表されます。

B.1.11 ALL_TAB_COMMENTS

このビューは、ユーザーが表およびビューに対して入力したコメントをリストします。

表 B-11 ALL_TAB_COMMENTS のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)	NOT NULL	TAB_COMMENTS 定義の所有者。
TABLE_NAME	VARCHAR2(128)	NOT NULL	TAB_COMMENTS 定義を持つ表の名前。
TABLE_TYPE	VARCHAR2(128)	NOT NULL	オブジェクトの型。
COMMENTS	VARCHAR2(4096)	NOT NULL	コメントのテキスト。

B.1.12 ALL_USERS

このビューは、接続済みデータベースに作成されているすべてのスキーマについて次の情報を提供します。

表 B-12 ALL_USERS のパラメータ

列	データ型	NULL	説明
USERNAME	VARCHAR2(30)	NOT NULL	ユーザーの名前。
USER_ID*	NUMBER	NOT NULL	ユーザーの ID 番号。
CREATED	DATE	NOT NULL	ユーザー作成日付。

B.1.13 ALL_VIEWS

このビューは、ユーザーがアクセスできるビューについて次の情報を提供します。

表 B-13 ALL_VIEWS のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)	NOT NULL	ビューの所有者のユーザー名。
VIEW_NAME	VARCHAR2(128)	NOT NULL	ビューの名前。
TEXT_LENGTH	NUMBER(10)	NOT NULL	ビューのテキストの長さ。
TEXT	VARCHAR2(1000)	NOT NULL	ビューのテキスト。

B.1.14 CAT

このビューは、ユーザーがアクセスできる表およびビューについて次の情報を提供します。

表 B-14 CAT のパラメータ

列	データ型	NULL	説明
TABLE_NAME	VARCHAR2(128)	NOT NULL	オブジェクトの名前。
TABLE_TYPE	VARCHAR2(128)	NOT NULL	オブジェクトの型 : TABLE または VIEW

B.1.15 COLUMN_PRIVILEGES

このビューは、ユーザーが権限付与者、権限受領者または所有者である場合、あるいは PUBLIC が権限受領者である場合の、列に対する権限付与について次の情報を提供します。

表 B-15 COLUMN_PRIVILEGES のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)	NOT NULL	オブジェクトの所有者のユーザー名。
TABLE_NAME	VARCHAR2(128)	NOT NULL	オブジェクトの名前。
COLUMN_NAME	VARCHAR2(128)		列の名前。
GRANTOR	VARCHAR2(128)		権限付与を実行したユーザーの名前。
GRANTEE	VARCHAR2(128)		アクセス権が付与されたユーザーの名前。
GRANT_TYPE	VARCHAR2(128)	NOT NULL	オブジェクトに対する権限。値は、SELECT、INSERT または DELETE です。
GRANTABLE	VARCHAR2(128)		GRANT OPTION を指定して権限が付与されている場合は YES、それ以外の場合は NO です。

B.1.16 DATABASE_PARAMETERS

このビューは、照合順序を制御する NLS_SORT パラメータの値をリストします。

表 B-16 DATABASE_PARAMETERS のパラメータ

列	データ型	NULL	説明
PARAMETER	VARCHAR2(30)	NOT NULL	NLS_SORT
VALUE	VARCHAR2(128)		照合順序の文字列定数。値は、BINARY、FRENCH、GERMAN、CZECH または XCZECH のいずれかです。

B.1.17 DUAL

このビューは、単一行を返す問合せで使用できるダミー表です。たとえば、CURRENT_TIMESTAMP の選択に DUAL を使用できます。

表 B-17 DUAL のパラメータ

列	データ型	NULL	説明
DUMMY	VARCHAR2(1)	NOT NULL	常に「X」。

B.1.18 SNAPSHOTS

このビューはデフォルトでは存在せず、レプリケーション中にのみ作成されます。ユーザーがアクセスできるスナップショットについて次の情報を提供します。

表 B-18 SNAPSHOTS のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(30)		スナップショットの所有者。
NAME	VARCHAR2(30)		スナップショットを表示するためにユーザーおよびアプリケーションが使用するビューの名前。
TABLE_NAME	VARCHAR2(30)		スナップショットが格納される表。
MASTER_VIEW	VARCHAR2(30)		スナップショット・マスター・ビューの名前。
MASTER_OWNER	VARCHAR2(30)		マスター表の所有者。

表 B-18 SNAPSHOTS のパラメータ（続き）

列	データ型	NULL	説明
MASTER	VARCHAR2(30)		このスナップショットのコピー元のマスター表の名前。
MASTER_LINK	VARCHAR2(128)		マスター・サイトへのデータベース・リンクの名前。
MASTER_ROLLBACK	VARCHAR2(30)		マスター・サイトで使用するロールバック・セグメント。
CAN_USE_LOG	VARCHAR2(3)		スナップショットでスナップショット・ログを使用できるかどうかを指定します。このスナップショットがスナップショット・ログを使用できる場合は YES、このスナップショットが複雑すぎてログを使用できない場合は NO です。
UPDATABLE	VARCHAR2(3)		スナップショットが更新可能かどうかを示します。更新可能の場合は YES、更新可能でない場合は NO です。NO の場合、スナップショットは読取り専用です。
SUBQUERY	VARCHAR2(3)		スナップショット問合せに副問合せが含まれているかどうかを示します。含まれている場合は YES、含まれていない場合は NO です。
KEYTYPE	VARCHAR2(4)		Oracle7 の場合に、スナップショットが ROWID スナップショットか主キー・スナップショットかを示します。値は、ROWID の場合は R、主キーの場合は P です。
SNAPSHOT_ID	DATE		Oracle7 の場合に、スナップショットの一意 ID を示します。
SNAPSHOT_ID8	NUMBER		Oracle8 の場合に、スナップショットの一意 ID を示します。

表 B-18 SNAPSHOTS のパラメータ（続き）

列	データ型	NULL	説明
SNAPTYPE	NUMBER		Oracle8 の場合に、スナップショットが ROWID スナップショットか主キー・スナップショットかを示します。値は、ROWID の場合は R、主キーの場合は P です。
REFMETHOD	NUMBER		リフレッシュ・タイプを示します。値は、COMPLETE、FAST、OPTIMUM（FORCE）です。この列は Oracle の内部使用専用です。
LAST_REFRESH	DATE		マスター・サイトでの最後のリフレッシュ日時。
TYPE	VARCHAR2(8)		スナップショット・タイプを示します。値は、複合の場合は C、単純の場合は S です。
NEXT	VARCHAR2(200)		次のリフレッシュ日付を計算する日付関数。
START_WITH	DATE		最初のリフレッシュ日付を計算する日付関数。
REFRESH_GROUP	NUMBER		リフレッシュ・グループの ID を示します。
UPDATE_TRIG	VARCHAR2(30)		UPDATE_LOG にデータを挿入するトリガーの名前。
UPDATE_LOG	VARCHAR2(30)		更新可能スナップショットに加えられた変更を記録する表。
STATUS	VARCHAR2(8)		実行時のリフレッシュ・ステータスを示します。
PKCOLS	VARCHAR2(1056)		主キー列を格納します。
TABLE_COUNT	NUMBER		副問合せスナップショットの表の数を示します。

表 B-18 SNAPSHOTS のパラメータ（続き）

列	データ型	NULL	説明
SCHEMA_CHANGED	CHAR(1)		マスター・スキーマが変更されたかどうかを示します。変更された場合は YES、変更されていない場合は NO です。
HIDDEN_COLUMNS	VARCHAR2(1056)		非表示列を格納します。
QUERY	LONG		スナップショットを定義する SQL 問合せ。

B.1.19 TABLE_PRIVILEGES

このビューは、オブジェクトに対する権限付与について、ユーザーまたは PUBLIC が権限受領者である場合の情報を提供します。

表 B-19 TABLE_PRIVILEGES のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)	NOT NULL	オブジェクトの所有者のユーザー名。
TABLE_NAME	VARCHAR2(128)	NOT NULL	オブジェクトの名前。
GRANTOR	VARCHAR2(128)		権限付与を実行したユーザーの名前。
GRANTEE	VARCHAR2(128)		アクセス権が付与されているユーザーの名前。
GRANT_TYPE	VARCHAR2(128)	NOT NULL	オブジェクトに対する権限。値は、SELECT、INSERT または DELETE のいずれかです。
GRANTABLE	VARCHAR2(128)		GRANT OPTION を指定して権限が付与されている場合は YES、それ以外の場合は NO です。

B.1.20 USER_OBJECTS

このビューは、ユーザーがアクセスできるオブジェクトについて次の情報を提供します。

表 B-20 USER_OBJECTS のパラメータ

列	データ型	NULL	説明
OWNER	VARCHAR2(128)	NOT NULL	オブジェクトの所有者のユーザー名。
OBJECT_NAME	VARCHAR2(128)	NOT NULL	オブジェクトの名前。
OBJECT_ID	NUMBER(10)	NOT NULL	オブジェクトのオブジェクト識別子。
OBJECT_TYPE	VARCHAR2(128)		オブジェクトの型：TABLE、VIEW、INDEX、SEQUENCE、SYNONYM
CREATED	DATE		オブジェクトの作成タイムスタンプ。
LAST_DDL_TIME	DATE		DDL コマンド（GRANT および REVOKE を含む）の結果、オブジェクトが最後に変更されたタイムスタンプ。
CREATED_TIME	VARCHAR2(128)		オブジェクト（文字データ）の作成タイムスタンプ。
STATUS*	VARCHAR2(128)		オブジェクトのステータス：VALID、INVALID または N/A（常に有効）

C

データベースのツールとユーティリティ

この付録では、次のデータベース・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 C-1 ツールとユーティリティ

ユーティリティ	説明
言語ソートのサポート	言語ソート機能を有効にしてデータベースを作成できます。
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。
dropjava	Oracle Lite データベースから Java クラスを削除するために使用できるコマンドライン・ユーティリティです。詳細は、『Oracle9*i* Lite Java 開発者ガイド』を参照してください。
ENCRYPDB	Oracle Lite データベースの暗号化に使用します。
loadjava	Oracle Lite に Java クラスをロードするために使用できるコマンドライン・ユーティリティです。詳細は、『Oracle9*i* Lite Java 開発者ガイド』を参照してください。
MIGRATE	以前のリリースから Oracle Lite に移行するために使用します。
Mobile SQL	Mobile SQL は、Oracle Lite データベースに接続できるコマンドライン・インタフェースです。
ODBC Administrator と Oracle Lite ODBC ドライバ	ODBC 接続の管理に使用します。Oracle Lite ODBC ドライバ経由でアクセスする Oracle Lite データベースに ODBC ドライバを関連付けるデータ・ソース名（DSN）を作成します。
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。

表 C-1 ツールとユーティリティ（続き）

ユーティリティ	説明
OLLOAD	このコマンドライン・ツールは、Oracle Lite データベースにある表に外部ファイルからデータをロードしたり、Oracle Lite データベースの表から外部ファイルにデータをアンロード（ダンプ）す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。
VALIDATEDB	Oracle Lite データベースの構造の検査に使用します。

C.1 言語ソートのサポート

言語ソートは、Oracle Lite データベースの ASCII バージョンの新機能です。指定された言語または照合順序で、言語に受入れ可能な順序に並べられた文字列を生成します。ASCII バージョンは、シングルバイトの 8 ビット・コード体系により定義されたコード・ページをいくつかサポートします。これらのコード・ページは、それぞれが 7 ビット ASCII のスーパー・セットで、ヨーロッパ言語グループのサポートに必要な追加のアクセント文字が上位 128 バイトに含まれています。新規の文字列比較方法が提供されています。この方法では、文字列の各照合要素を、サポートされているコード・ページの対応する 8 ビット値にマップすることにより、言語的に正しい順序で文字列を生成します。

C.1.1 言語ソート対応データベースの作成

<collation sequence> を使用可能に指定した CREATEDB コマンドライン・ユーティリティを使用してデータベースを作成するときは、言語ソート機能を使用可能にする必要があります。

ORDER_BY 句および WHERE 条件の動作は、NLS_SORT パラメータがどのように実装されているかにより決まります。バイナリ・ソートがデフォルト設定で、言語ソートの順序ルールを使用するように collation_sequence パラメータが設定されていないかぎり、これが使用されます。

Oracle Lite データベースの現在のバージョンでは、Unicode および NLSRT はサポートされていません。したがって、NCHAR データ型および照合順序のカスタマイズはまだ使用できません。

C.1.2 照合の動作

照合とは、言語に受入れ可能な順序で文字列を並べることです。照合順序とは、アルファベットの全照合要素の最小から最大までの順序です。照合順序が指定された後は、同一アルファベットで構成されるすべての文字列の順序が固定されます。このように、照合順序は、照合に関する言語要件をコード化したものです。照合要素とは、2 つの文字列の順序を決定するために比較関数で使用される最小単位のサブ文字列です。

C.1.3 照合要素の例

通常、照合要素は 1 文字です。バイナリ・ソートでは、1 つのプロパティ（文字を表すコード値）のみが使用されます。しかし、言語ソートでは、通常、3 つのプロパティが使用されます。1 次相違レベルは基本文字です。2 次相違レベルは、基本文字に対する発音区別記号です。3 次相違レベルは、文字の大 / 小文字区別です。句読点を 4 次相違レベルとすることもできますが、句読点の比較は最後に発生するもので、言語レベルではなくバイナリ・レベルで行われます。これらの相違レベルが言語要素のそれぞれに対して使用されます。次の項では、ソート優先順位を示す例を説明します。

C.1.3.1 通常の文字のソート

例 1: 'a' < 'b'。この 2 つには文字レベルで 1 次的な相違があります。

例 2: 'À' > 'a'。この相違は 2 次レベルで発生しています。'À' と 'a' は、1 次レベルでは「等価」とみなされることに注意してください。

例 3: FRENCH では 'À' < 'à'、GERMAN では 'À' > 'à'。この相違は 3 次レベルで発生しています。'À' と 'à' は、1 次レベルと 2 次レベルでは「等価」とみなされることに注意してください。また、大 / 小文字区別の規則は、言語によって異なる可能性があることにも注意してください。

例 4: 'às' < 'at'。これは 1 次レベルの相違です。この例は、それぞれの相違レベルの役割を示しています。1 次レベルの相違が文字列内のどこかにある場合、下位レベルの相違は無視されます。

例 5: '+data' < '-data' < 'data' < 'data-'。文字列が比較され、1 次、2 次または 3 次の各レベルでいずれも相違がない場合は、句読点が比較されます。

C.1.3.2 フランス語のアクセント記号の逆ソート

一部の言語（特にフランス語）では、最後のアクセントの違いに従って、2 次レベルで単語に順序を付けることが必要になります。この動作は、フランス語の 2 次ソートまたはフランス語のアクセント順序と呼ばれます。

例 6: FRENCH では 'côte' < 'coté'、GERMAN では 'coté' < 'côte'。'e' と 'é' の 2 次レベルの相違が、'ô' と 'o' の相違よりも後に発生していることに注意してください。

C.1.3.3 短縮文字のソート

グループ化された 2 つ以上の文字が 1 つの照合要素として機能する特殊な場合がいくつかあります。このような照合要素は短縮文字またはグループ文字と呼ばれます。この場合、このような文字プロパティのそれぞれに適切な値が割り当てられます。

例 7: XCZECH での 'h' < 'ch' < 'i'。ここで、'ch' には 1 次プロパティ値が割り当てられ、これにより 'h' および 'i' とは区別されます。このため、'h'<'ch'<'i' になります。'ch' は 1 つの文字として扱われることに注意してください。

C.1.3.4 拡張文字のソート

1 文字が 2 文字以上のつながりとしてソートされる場合、これを拡張文字と呼びます。たとえば、ドイツ語の「ß」は、他の文字と比較するときに、2 文字「ss」の文字列であるかのように扱われます。

C.1.3.5 数値文字のソート

現在サポートされているのは、1 桁の数字 '0' から '9' までのソートのみです。サポートされているヨーロッパ言語の場合、数字は常にアルファベット文字よりも上位にソートされます。その他の言語では、そうとはかぎりません。ローマ数字や、「one」、「two」、「three」というような数える順番を表すその他の数値文字は、現時点ではサポートされていません。

例 8: すべてのヨーロッパ言語では '1' > 'z'、ラトビア語では '1' < 'a'。この相違は 1 次レベルで発生しています。

C.2 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] DATABASE_SIZE] EXTENT_SIZE]
[collation sequence]
```

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。DSN を追加する方法は、C.7.1 項「ODBC Administrator を使用した DSN の追加」を参照してください。

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。

DATABASE_SIZE
バイト単位のデータベース・サイズ。

EXTENT_SIZE
データベース・ファイル内のページ数の増分量。データベースで現在のファイルのページが足りなくなった場合、ファイルのページはこの数を単位として拡張されます。

collation sequence
このパラメータは文字列定数で、デフォルト以外の値が使用されたときに、言語ソート対応のデータベースが作成されます。ここで指定される照合順序は、NLS_SORT [collation sequence] パラメータを使用して **polite.ini** ファイルに設定される照合順序をオーバーライドします。この文字列には、次の表に示されているオプションのいずれかを使用することもできます。

表 C-2 照合順序の値

照合順序	説明
BINARY	デフォルト。2 つの文字列が 1 文字ずつ比較され、文字はそれぞれのバイナリ・コード値を使用して比較されます。
FRENCH	2 つの文字列がフランス語の照合順序に従って比較されます。ISO 8859-1 または IBM-1252 によりサポートされています。
GERMAN	2 つの文字列がドイツ語の照合順序に従って比較されます。ISO 8859-1 または IBM-1252 によりサポートされています。
CZECH	2 つの文字列がチェコ語の照合順序に従って比較されます。ISO 8859-2 または IBM-1250 によりサポートされています。
XCZECH	2 つの文字列が Xczech（拡張チェコ語）の照合順序に従って比較されます。ISO 8859-2 または IBM-1250 によりサポートされています。

注意： データベースの作成後に照合順序を変更する方法はありません。

例

```
createdb polite db1
createdb polite c:¥testdir¥db2.odb 300
createdb polite polite french
```

C.3 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、C.4 項「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 C-3 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、C.4 項「ENCRYPDB」のコメントを参照してください。

C.4 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、C.3 項「DECRYPDB」を参照してください。

構文

```
ENCRYPDB DSN | NONE DBName [New_Password [Old_Password]]
```

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_Password と Old_Password

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 C-4 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 アプリケーションの実行中は、`ENCRYPDB` は実行しません。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、実行時に入力が要求されます。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。
- DECRYPDB と ENCRYPDB 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、DECRYPDB 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 ENCRYPDB 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。

C.5 MIGRATE

説明

Oracle Lite データベースを前のバージョンからこのリリースに移行するユーティリティです。このユーティリティは、Oracle Lite 3.6 のデータベースを移行し、**.36** の拡張子を付けたバックアップ・コピーを作成します。以前のリリースの Oracle Lite がある場合、詳細は、『Oracle9*i* 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。

このユーティリティを使用する前に、Oracle Lite の現行リリースをインストールしておく必要があります。また、データベースが暗号化してある場合は、このユーティリティを使用する前に復号化しておく必要があります。

構文

```
MIGRATE DSN DBName
```

DBName は、データベース名またはデータベース名とパス名のいずれでもかまいません。

キーワードとパラメータ

DSN

移行するデータベースのデータ・ソース名です。これは、**ODBC.INI** ファイルにあるデフォルト・データベース・ディレクトリを参照して、*DBName* に指定されているデータベース名を探すために使用されます。DSN に NONE を指定する場合、*DBName* にはデータベース・ファイルの完全なパス名を指定する必要があります。

DBName
移行するデータベース名、またはパスとデータベース名です。データベース名のみを指定した場合は、ODBC.INI ファイルの（データ・ソース名の下の）DataDirectory パラメータに指定されているディレクトリにデータベース・ファイルが必要です。

コメント
この項で説明したとおり、このユーティリティを使用する前に Oracle Lite をインストールする必要があります。

MIGRATE ユーティリティにより生成されるメッセージは、画面上のコマンド・ウィンドウに表示されます。

このユーティリティを使用すると、既存の Oracle Lite データベースを圧縮できます。

例

```
MIGRATE polite db1
MIGRATE none c:¥testdir¥db1.odb
```

C.6 Mobile SQL

Mobile SQL は、コマンドライン・インタフェースとして実行されるアプリケーションです。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。

C.6.1 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ます。

C.6.2 Mobile SQL の起動

Mobile SQL を起動するには、Oracle_Home¥Mobile¥SDK¥Bin ディレクトリを開き、**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コマンドライン・インタフェースが起動されます。詳細は、『Oracle9*i* Lite SQL リファレンス』を参照してください。

C.7 ODBC Administrator と Oracle Lite ODBC ドライバ

データ・ソース名（DSN）により、Oracle Lite ODBC ドライバと、そのドライバを介してアクセスする Oracle Lite データベースが関連付けられます。Oracle Lite のインストール・プロセスにより、デフォルト DSN の POLITE が Oracle Lite データベース用に作成されます。追加作成する Oracle Lite データベースには、追加 DSN を作成することもできます。

ODBC Administrator は Microsoft 社が提供するツールです。Windows 95/98/NT および Windows 2000 の ODBC.INI ファイルおよびそれに関連付けられたレジストリ・エントリを管理します。ODBC.INI ファイルと Windows レジストリには、ODBC Administrator によりキャプチャされる DSN エントリが格納されます。ODBC Administrator を使用すると、DSN を Oracle Lite ODBC ドライバに関連付けられます。

注意： このマニュアルでは、ODBC Administrator の使用方法については説明していません。この情報は、オンライン・ヘルプを参照してください。

ODBC Administrator では DSN 以外にも次のパラメータを指定します。

表 C-5 ODBC Administrator の DSN パラメータ

DSN パラメータ	説明
Data Description	データ・ソースのオプションの記述。
Database Directory	データベースが存在するデータ・ディレクトリのパス。これは既存のパスです。
Database	作成する Oracle Lite データベースの名前。**.ODB** 拡張子は付けないでください。
Default Isolation Level	異なるトランザクション間で、どの程度操作をお互いに参照できるようにするかを決定します。サポートされている分離レベルの詳細は、1.6.3 項「分離」を参照してください。デフォルト・レベルは「READ COMMITTED」です。
Autocommit	トランザクション内のデータベース更新操作が実行されるたびに、更新操作をコミットします。自動コミットの値は、「Off」または「On」です。デフォルト値は「Off」です。

表 C-5 ODBC Administrator の DSN パラメータ（続き）

DSN パラメータ	説明
Default Cursor Type	■ *Forward Only*: デフォルト。スクロールできないカーソルで、結果セットを前方にのみ進めます。後方には進みません。この結果、このカーソルは前にフェッチされた行には戻れません。 ■ *Dynamic*: カーソルをオープンした後、結果セットのメンバー、順序または値に変更があった場合に検出できます。動的カーソルが行をフェッチし、その後、別アプリケーションによりその行が削除または更新された場合、動的カーソルは再フェッチしたときにこれらの変更を検出します。 ■ *Keyset Driven*: 結果セットのメンバーまたは順序に対する変更は検出しませんが、結果セットの行の値に対する変更は検出します。 ■ *Static*: カーソルをオープンした後、結果セットのメンバー、順序または値に対する変更を検出しません。静的カーソルが行をフェッチした後、別のアプリケーションによりその行が更新された場合、このカーソルは行を再フェッチした場合でもこの変更を検出しません。 サポートされている組合せを示す表は、1.6.3.4 項「サポートされている分離レベルとカーソル・タイプの組合せ」を参照してください。

たとえば、**ODBC.INI** ファイルの POLITE の DSN エントリには次のような行が含まれます。

```
[POLITE]
Description=Oracle Lite Data Source
DataDirectory=C:¥ORANT¥OLDB40
Database=POLITE
IsolationLevel=Repeatable Read
CursorType=Dynamic
```

C.7.1 ODBC Administrator を使用した DSN の追加

ODBC Administrator を使用して DSN を追加するには、次を実行します。

1. ODBC Administrator を起動します。これには、「Oracle9*i* Lite」プログラム・グループにある ODBC Administrator のアイコンを選択するか、DOS プロンプトで次をタイプします。

   ```
   C:¥>ODBCAD32
   ```

2. 「追加」をクリックします。
3. インストール済みの ODBC ドライバのリストから、「Oracle Lite *nn* ODBC ドライバ」（*nn* はリリース番号）をダブルクリックします。
4. 次に、DSN 名を追加し、ODBC ドライバのセットアップ用ダイアログ・ボックスにパラメータを定義します。パラメータの定義については、前述の表を参照してください。

C.7.2 読取り専用メディア（CD-ROM）を指す DSN の追加

1. C.7.1 項「ODBC Administrator を使用した DSN の追加」で説明されている方法で DSN を作成します。
2. **ODBC.INI** ファイルの新規 DSN に次の行を追加します。

   ```
   ReadOnly = TRUE
   ```

注意： CD-ROM 上のファイルを指す DSN を定義できます。DSN で CD-ROM ドライブおよびディレクトリを指すようにし、データベース・ファイルのファイル名を指定します。次に、**ODBC.INI** ファイルのデータ・ソース定義に「ReadOnly = TRUE」という行を追加します。ODBC プログラマは、（**ODBC.INI** ファイルに行を追加するかわりに）データベースをオープンする前に次をコールして、この機能を使用可能にできます。

```
SQLSetConnectOption( hdbc, SQL_ACCESS_MODE, SQL_MODE_READ_ONLY )
```

データベース・ファイルを読取り専用に設定すると、ログ・ファイルは作成されなくなります。更新、挿入、削除またはコミットは、表のメモリー内イメージに対して操作されているように見えますが、コミットしても、これらの変更はデータベース・ファイルに書き込まれません。アプリケーションを終了し、再接続し、問合せを発行すると、元のデータが表示されます。

C.8 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。

パラメータ

パラメータを設定または消去するには、DSN または NONE の前に「+」または「-」パラメータ引数を 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
```

ODBINFO では次のパラメータを使用できます。

表 C-6 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・ページが *filename*.**plg** にバックアップされます。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。
fsync	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。 `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。 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。
reuseoid	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と、「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。 `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。 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 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。

表 C-6 ODBINFO のパラメータ（続き）

パラメータ	説明
compress	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。 ■ 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、表が断片化されます。 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。

C.9 OLLOAD

Oracle Lite ロード・ユーティリティの API の詳細は、付録 E「Oracle Lite ロード・ユーティリティの API」を参照してください。

説明

このコマンドライン・ツールを使用すると、Oracle Lite データベースにある表に外部ファイルからデータをロードしたり、Oracle Lite データベースの表から外部ファイルにデータをアンロード（ダンプ）できます。SQL*Loader とは異なり、OLLOAD では制御ファイルを使用しません。データ・パラメータと書式情報はすべてコマンドラインに指定します。

データのロード時、OLLOAD は、フィールドがセパレータ文字で区切られたレコードが、1 行に 1 レコードずつ含まれている入力ファイルを取ります。デフォルトのフィールド・セパレータはカンマ（,）です。レコードには、引用符のついた文字列を値として持つフィールドも含めることができます。デフォルトは一重引用符（'）です。データ解析の詳細は、「コメント」を参照してください。

構文

データ・ファイルをロードするには、次のように指定します。

```
olload [options] -load dbpath tbl [col1 col2 ...] [<datafile]
```

出力ファイルにアンロード（ダンプ）するには、次のように指定します。

olload [options] -dump *dbpath tbl* [*col1 col2* ...][>*outfile*]

キーワードとパラメータ

[options]
オプションのリストは、「オプション」を参照してください。

-load
ロード・ユーティリティを使用します。

-dump
アンロード（ダンプ）ユーティリティを使用します。

dbpath
Oracle Lite データベース・ファイル（**.ODB** ファイル）へのパスです。

tbl
表の名前です。OLLOAD は、まずユーザーが指定した大文字 / 小文字の形式で表の名前を見つけようとします。これに失敗した場合、ユーザーが指定した名前を大文字にして検索します。

注意： デフォルト・ユーザーは「SYSTEM」です。別のユーザー名の表に対して OLLOAD 操作を指定する場合は、*tbl* パラメータの前にユーザー名とピリオド（.）を接頭辞として指定します。

---

col1 col2
列名です。OLLOAD は、まずユーザーが指定した大文字 / 小文字の形式で列の名前を見つけようとします。これに失敗した場合、ユーザーが指定した名前を大文字にして検索します。

[datafile] [outfile]
ロードまたはアンロード（ダンプ）操作の対象となるソース・ファイルまたは宛先ファイルです。datafile または outfile を指定しないと、OLLOAD は出力を画面に表示します。

オプション

-sep *character*
フィールド・セパレータです。このオプションを指定しないと、セパレータ文字はカンマ（,）であると解釈されます。

-quote *character*
引用符文字です。このオプションを指定しないと、引用符文字は一重引用符（'）であると解釈されます。

-file *filename*
このオプションは、データのロードおよびアンロード時にソース・ファイル名または宛先ファイル名を指定するために使用します。データのロード時には、*filename* は Oracle Lite データベースにロードされるソース・ファイルを示します。データのアンロード（ダンプ）時には、アンロードされるデータの宛先ファイルを示します。

重要： Oracle Lite データベースからデータをアンロードし、このデータを別の Oracle Lite データベースにロードする（つまりパイプ処理する）場合は、このオプションにファイル名を指定しないでください。構文の例は、「例」の項にある 2 番目の例を参照してください。

-log *logfile*
このオプションは、OLLOAD がロード中に挿入できなかった行をリストしたログ・ファイルを生成する場合に指定します。ログ・ファイルを指定しないと、ロードは最初のエラーで停止します。

-passwd *passwd*
暗号化されているデータベースの接続パスワードです。ロードとアンロードを実行するには、このパスワードを指定する必要があります。

-nosingle
このオプションは、シングル・ユーザー・モードを使用しない場合に指定します。このオプションを指定するとパフォーマンスが低下しますが、データベースに対して他の接続が可能になります。

-readonly
このオプションは、Oracle Lite の読取り専用データベース（たとえば CD-ROM 上のデータベース）からデータをアンロードするときに指定します。

-commit *count*
このオプションは、指定された数の行を処理した後に OLLOAD でコミットを実行する場合に使用します。デフォルトは 10000 です。OLLOAD は、指定された数の行をコミットするたびに、画面にアスタリスク（*）を出力します。コミット操作を使用禁止にするには、「0」を指定します。

-mark *count*
このオプションは、指定された数のレコードを処理した後に OLLOAD で画面にドットを出力する場合に使用します。デフォルトは 1000 です。この機能を使用禁止にするには、「0」を指定します。

コメント

データ解析

OLLOAD のデータ解析の例を次に示します。

表 C-7 データ解析の例

入力	データ	説明
'Redwood Shores, CA'	Redwood Shores, CA	エスケープ・シーケンス（2 個の一重引用符）で 1 個の一重引用符を表します。
'O''Brien'	O'Brien	入力文字列を引用符で囲むと、文字列内の空白と句読点が保持されます。
fire fly	firefly	引用符で囲まれていないデータ内の空白は無視されます。
,	NULL,NULL	空のフィールドは NULL です。
1,,3,	1,NULL,3,NULL	空のフィールドは NULL です。
	[行が挿入されていない]	完全に空の行は無視されます。

データベースの列数を超えた列の値は無視されます。行の終わりに値がない場合は、NULL に設定されます。

OLLOAD ユーティリティの制約

OLLOAD は、タブで区切られた入力ファイルと LONG データ型はサポートしません。

例

```
olload -quote ¥" -file p_kakaku.csv -load c:¥orant¥oldb40¥polite.odb skkm01

olload -dump c:¥orant¥oldb40¥polite.odb emp empno ename | olload -load myfile.odb
myemp
```

C.10 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。DSN は none などのダミー引数に指定できますが、その場合、データベース名は完全修飾のファイル名を指定する必要があります。

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。

例

```
removedb polite db1
removedb none c:¥testdir¥db2.odb
```

C.11 VALIDATEDB

説明

このコマンドライン・ツールはデータベース・ファイル内の構造を検査し、データベース構造が破壊されていることがわかった場合は、検出されたエラーをユーザー指定のファイルにリストします。このツールは次の項目をチェックします。

- オブジェクトーデータベース・オブジェクトのヘッダー情報。オブジェクトが移動または圧縮された場合は、フラグの整合性がチェックされます。オブジェクト長が有効範囲と照合されます。BLOB オブジェクトの場合は、オブジェクトのフレームがボリューム・ページ・ビットマップと照合されます。
- 索引ページ・エントリー索引ページ・エントリを作成した結果、正しい数のノードまたは正しいオブジェクト識別子リストが作成されたことをチェックします。
- 索引ページーページ上のキー値がすべてソートされていることをチェックします。ページに含まれているすべてのオブジェクトが検査されます。ページ記述子情報（オブジェクト数、未使用バイト数、エントリ数など）が、ページ上の実際のオブジェクトと照合されます。

- グループ－各ページの検査時に、グループ記述子情報が実際のページ数およびオブジェクト数と照合されます。
- 索引－すべてのページが B ツリーに照らして検査されます。このツールは、ページ・ポインタもすべて検査します。キー値が全体としてソートされているかどうかを確認するために、B ツリーのレベルをすべてチェックします。B ツリーのリーフ要素に関しては、リーフ・ページ・エントリのすべての OID が実際のグループ・オブジェクトと整合性があるかどうかがチェックされます。

構文

```
validatedb DSName DBName [-p password] [-t schemaname.tablename] -file
outputfilename
```

キーワードとパラメータ

DSName
データ・ソース名。DSN がない場合は、**NONE** に指定できます。

DBName
DSN が存在し、データベース・ファイル名と DSN のデフォルト・ファイル名が異なる場合、これはデータベース・ファイル名（.odb 拡張子なし）です。DSN がない場合、フルパスが指定されていないかぎり、VALIDATEDB はカレント・ディレクトリを使用します。データベース・ファイルと同じディレクトリ内にログ・ファイルがある場合は、ログ・ファイルも検査されます。

Password
暗号化されているデータベースのパスワード。

schemaname
オプションのスキーマ名。これを指定しないとデフォルト・スキーマ名が使用されます。

tablename
オプションの表名。指定された表とその索引のすべてが検査されます。表名を指定しないと、データベース全体が検査されます。

outputfilename
VALIDATEDB により検出されたすべてのエラーとその他の関連情報が保存されるテキスト・ファイルのファイル名（オプション）。デフォルトは **stdout** です。

例

```
validatedb polite polite -t emp -file out.txt
```

D

SQL 問合せの最適化

この付録では、SQL 問合せのパフォーマンスを向上させるためのヒントを示します。内容は次のとおりです。

- D.1 項「単一表問合せの最適化」
- D.2 項「結合問合せの最適化」
- D.3 項「ORDER BY および GROUP BY 句による最適化」

例では、次のデータベース・スキーマを使用します。

表 D-1 データベース・スキーマの例

表	列	主キー	外部キー
LOCATION	LOC#	LOC#	
	LOC_NAME		
EMP	SS#	SS#	
	NAME		
	JOB_TITLE		
	WORKS_IN		WORKS_IN は DEPT（DEPT#）を参照
DEPT	DEPT#	DEPT#	
	NAME		
	BUDGET		
	LOC		LOC は LOCATION（LOC#）を参照
	MGR		MGR は EMP（SS#）を参照

D.1 単一表問合せの最適化

特定の列の値を基に表の行を選択する問合せのパフォーマンスを向上するには、その列の索引を作成します。たとえば、次の問合せは、EMP 表の NAME 列に索引が作成されている方が、パフォーマンスが良くなります。

```
SELECT *
FROM EMP
WHERE NAME = 'Smith';
```

D.2 結合問合せの最適化

次のようにすると、結合問合せ（FROM 句に複数の表参照が使用されている問合せ）のパフォーマンスが向上します。

D.2.1 内部表の結合列に対する索引の作成

次の例では、結合問合せの内部表は DEPT で、DEPT の結合列は DEPT# です。DEPT.DEPT# の索引により、問合せのパフォーマンスが向上します。この例では、DEPT# が DEPT の主キーであるため、その索引は暗黙的に作成されています。オプティマイザは索引があることを検出し、内部表として DEPT を使用することに決定します。EMP.WORKS_IN 列にも索引がある場合、オプティマイザは DEPT の次に EMP（この場合は EMP が内部表）を実行する場合と、EMP の次に DEPT（この場合は DEPT が内部表）を実行する場合のコストを評価し、コストの低い実行計画を採用します。

```
SELECT e.SS#, e.NAME, d.BUDGET
FROM EMP e, DEPT d
WHERE e.WORKS_IN = DEPT.DEPT#
AND e.JOB_TITLE = 'Manager';
```

D.2.2 問合せオプティマイザのバイパス

通常、オプティマイザが最善の実行計画（結合対象の表の最適の実行順序）を選択します。オプティマイザで適切な実行計画が作成されない場合は、SQL の HINTS 機能を使用して実行順序を制御できます。詳細は、『Oracle9*i* Lite SQL リファレンス』を参照してください。

たとえば、各部門とその管理者の名前を選択する場合は、次の 2 通りの問合せを作成できます。次に示す最初の例では、ヒント /++ordered++/ が、FROM 句内に示される表の順序で結合を実行することを示し、結合順序の最適化が試みられます。

```
SELECT /++ordered++/ d.NAME, e.NAME
FROM DEPT d, EMP e
WHERE d.MGR = e.SS#
```

または

```
SELECT /++ordered++/ d.NAME, e.NAME
FROM EMP e, DEPT d
WHERE d.MGR = e.SS#
```

部門数が 10、従業員数が 1000 名で、各問合せの内部表の結合列には索引が作成されているとします。最初の問合せでは、最初の表から 10 行の該当行が生成されます（この場合は表全体）。2 番目の問合せでは、最初の表から 1000 行の該当行が生成されます。最初の問合せは、EMP 表に 10 回アクセスし、DEPT 表を 1 回スキャンします。2 番目の問合せは、EMP 表を 1 回スキャンしますが、DEPT 表には 1000 回アクセスします。したがって、最初の問合せの方がパフォーマンスが高くなります。経験則として、表の並べ方は、有効行数が一番少ない表を最初に、有効行数の一番多い表を最後に並べます。問合せでの表の有効行数は、その表のみで解決される論理条件を適用すると取得できます。

別の例として、ニューヨークなど、特定の場所における社会保障番号と従業員名を取り出すための問合せを考えてみます。このサンプルのスキーマによると、問合せでは FROM 句に 3 種類の表参照があります。これら 3 種類の表は、6 通りの順序に設定できます。どの順序を選んでも結果は同じになりますが、パフォーマンスに大きな違いが生じる可能性があります。

LOCATION 表の有効行サイズが小さいとします。たとえば、`select count(*) from LOCATION where LOC_NAME = 'New York'` が小さいセットであるとします。前述の規則を基にすると、LOCATION 表が FROM 句の最初の表になります。LOCATION.LOC_NAME に対する索引が必要です。LOCATION は DEPT と結合する必要があるため、DEPT は 2 番目の表で、DEPT の LOC 列に索引が必要です。同様に、3 番目の表は EMP で、`EMP#` に索引が必要です。この問合せは次のように作成できます。

```
SELECT /++ordered++/ e.SS#, e.NAME
FROM LOCATION l, DEPT d, EMP e
WHERE l.LOC_NAME = 'New York' AND
l.LOC# = d.LOC AND
d.DEPT# = e.WORKS_IN;
```

D.3 ORDER BY および GROUP BY 句による最適化

SELECT 文の実行速度の向上とメモリー・キャッシュの消費量の低減を目標に、様々なパフォーマンス改善が行われてきました。GROUP BY および ORDER BY 句は、適切な索引が利用できる場合は、ソートを回避しようとします。

D.3.1 IN 副問合せ変換

副問合せ内の選択リストに一意の索引が作成されるときは、IN 副問合せを結合文に変換します。

たとえば、次の IN 副問合せ文は対応する結合文に変換されます。ここで、c1 は表 t2 の主キーであるとします。

```
SELECT c2 FROM t1 WHERE
c2 IN (SELECT c1 FROM t2);
```

これは、次のようになります。

```
SELECT c2 FROM t1, t2 WHERE t1.c2 = t2.c1;
```

D.3.2 GROUP BY を使用しない ORDER BY 最適化

次の条件がすべて満たされた場合、SELECT 文内の ORDER BY 句に対するソート手順が不要になります。

1. すべての ORDER BY 列が昇順または降順に並んでいる。
2. ORDER BY 句に列のみが指定されている。つまり、ORDER BY 句に式が使用されていない。
3. ORDER BY 列がベース表索引の接頭辞である。
4. 索引によるアクセスの方が、結果セットのソートよりも安価である。

D.3.3 ORDER BY を使用しない GROUP BY 最適化

GROUP BY 列がベース表索引の接頭辞である場合、グループ設定のソート手順が不要になります。

D.3.4 GROUP BY を使用した ORDER BY 最適化

ORDER BY 列が GROUP BY 列の接頭辞で、すべての列が昇順または降順に並んでいる場合、問合せ結果のソート手順が不要になります。GROUP BY 列がベース表索引の接頭辞である場合、グループ設定のソート手順も不要になります。

D.3.5 副問合せ結果のキャッシュ

副問合せで返される行数が少なく、問合せが相関していないとオプティマイザが判断した場合、パフォーマンスを向上させるために問合せ結果はメモリーにキャッシュされます。現在、行数は 2000 に設定されています。たとえば、次のとおりです。

```
select * from t1 where
t1.c1 = (select sum(salary)
from t2 where t2.deptno = 100);
```

E

Oracle Lite ロード・ユーティリティの API

この付録では、Oracle Lite ロード・ユーティリティについて説明します。各項で異なるトピックを取り上げています。内容は次のとおりです。

E.1 概要

Oracle Lite ロード・ユーティリティを使用すると、外部ファイルから Oracle Lite データベースの表にデータをロードしたり、Oracle Lite データベースの表から外部ファイルにデータをアンロード（ダンプ）できます。コマンドライン・ツール OLLOAD の使用方法の詳細は、C.9 項「OLLOAD」を参照してください。

E.2 Oracle Lite ロード・ユーティリティの API

Oracle Lite ロード・ユーティリティには、次の API が含まれています。

- E.2.1 項「データベースへの接続 : olConnect」
- E.2.2 項「データベースからの切断 : olDisconnect」
- E.2.3 項「表のすべての行の削除 : olTruncate」
- E.2.4 項「ロード操作とダンプ操作のパラメータの設定 : olSet」
- E.2.5 項「データのロード : olLoad」
- E.2.6 項「データのダンプ : olDump」

表をロードおよびアンロードする一般的な方式は、次のとおりです。

1. ローカル変数 DBHandle を宣言します。
2. olConnect を使用してデータベースに接続します。
3. オプションとして、ロードまたはアンロードのパラメータを設定します。
4. olDump または olLoad を使用してデータをダンプまたはロードします。オプションとして、olTruncate をコールして表の行をすべて削除することもできます。
5. olDisconnect を使用してデータベースから切断します。

E.2.1 データベースへの接続 : olConnect

この API を使用して、データベースに接続します。これが、最初にコールする API です。後の API で使用されるロードおよびアンロードのコンテキストを作成し、ロードおよびアンロードの動作を制御します。この API は、初期化されたデータベース・ハンドル DBHandle を返します。

構文

```
olError olConnect (char *database_path, char *password, DBHandle &dbh);
```

引数

表 E-1 olConnect の引数

引数	説明
database_path	データベース・ファイルへのフルパス（ディレクトリ・パスとファイル名）。
Password	暗号化されたデータベースに使用するパスワード。その他のデータベースの場合、パスワードは NULL です。
dbh	現在のデータベース接続のアプリケーション・ハンドル。これにより、1 つのアプリケーション・スレッドで複数のデータベース接続を使用できます（各接続は別々のハンドルを持ちます）。

戻り値

（short）整数エラー・コード

値 -1 ～ -8999 はデータベースにより返されるエラー・コードとして使用され、-9000 以下は OLLOAD 固有のエラー・コードとして使用されます。

E.2.2 データベースからの切断 : olDisconnect

データベースとの接続を切断します。

構文

```
olError olDisconnect (DBHandle dbh);
```

引数

表 E-2 olDisconnect の引数

引数	説明
dbh	現在のアプリケーション・ハンドル。

戻り値

（short）整数エラー・コード

E.2.3 表のすべての行の削除 : olTruncate

この API を使用すると、既存の表の行をすべて削除できます。

構文

```
olError olTruncate (DBHandle  dbh, char* table );
```

引数

表 E-3 olTruncate の引数

引数	説明
dbh	現在のアプリケーション・ハンドル。
tablename	次の形式で表される表の名前： owner_name.table_name owner_name は、表の所有者の名前です。

戻り値

（short）整数エラー・コード

E.2.4 ロード操作とダンプ操作のパラメータの設定 : olSet

これはオプションの API です。この API は、ロードおよびアンロードのオプションのパラメータを設定します。

構文

```
olError olSet (DBHandle  dbh, char * parameter_name, char *parameter_value);
```

引数

表 E-4 olSet の引数

引数	説明
dbh	現在のアプリケーション・ハンドル。
parameter_name	指定されたパラメータの名前。大 / 小文字は区別されません。パラメータ名の一覧とパラメータのデフォルト値は、E.2.9 項「パラメータ」を参照してください。
parameter_value	設定値。ほとんどのパラメータで大 / 小文字は区別されません。

戻り値

（short）整数エラー・コード

E.2.5 データのロード : olLoad

olLoad は、現在のパラメータ設定を使用して、ファイルから表にデータをロードします。

構文

```
olError olLoad (DBHandle dbh, char *table, char *file);
```

引数

表 E-5 olLoad の引数

引数	説明
dbh	現在のアプリケーション・ハンドル。
table	次の形式で表した表の情報： owner_name.table_name(col1,col2,…) col1,col2,… は、ロードする列名のリストです。 これにより、表全体でなく特定の列をロードおよびダンプできます。表全体をダンプする場合は、列リストを指定する必要はありません。
file	ロード元のファイルへのパス。

注意： table が NULL の場合、olLoad ではファイル・ヘッダー内で表の説明を見つけようとします。

戻り値

（short）整数エラー・コード

E.2.6 データのダンプ : olDump

olDump は、現在のパラメータ設定を使用して、表からファイルにデータをダンプします。

構文

```
olError olDump (DBHandle dbh, char *table, char *file);
```

引数

表 E-6 olDump の引数

引数	説明
dbh	現在のアプリケーション・ハンドル。
table	olLoad と同じ形式の表の情報。
file	ダンプ・データの書込み先のファイル。

戻り値

（short）整数エラー・コード

E.2.7 コンパイル

DBHandle、パラメータの定数とフラグおよびエラー・メッセージ・コードの宣言は、Oracle_Home¥Mobile¥SDK¥include ディレクトリのファイル **olloader.h** にあります。コンパイルの場合、olloader.h をメイン・ソース・ファイルに含めます。

E.2.8 リンク

リンクでは、ファイル **olloader40.dll** とライブラリ・ファイル **olloader40.lib** を使用します。これらのファイルをプロジェクト設定に含めます。

E.2.9 パラメータ

パラメータ名は大 / 小文字を区別しません。

表 E-7 パラメータ

パラメータ	説明
FILEFORMAT	入力および出力ファイル形式。次の形式がサポートされています。 ■ FixedASCII －各データ型について固定幅フィールドのテキスト・ファイル。 ■ CSV －カンマで区切られた値の形式。 ■ BINARY －バイナリ・ファイル形式。 キーワード値の大 / 小文字は区別されません。
SEPARATOR	値を区切るセパレータ。1 文字で、デフォルトはカンマです。
QUOTECHAR	ファイル内の STRING データ型の値の引用符文字。デフォルトは一重引用符（'）です。
LOGFILE	ログ・ファイルの名前。デフォルトは NULL です（ログ・ファイルは生成されず、ロードは最初のエラーで停止します）。
NOSINGLE	シングル・ユーザー・モードの場合は FALSE（デフォルト）、シングル・ユーザー以外のモードの場合は TRUE です。
READONLY	FALSE（デフォルト）。読取り専用データベース（CD-ROM など）からデータをダンプする場合は、TRUE を指定します。
COMMITCOUNT	olLoad、olDump または olTruncate がコミットされる前に処理される行数。デフォルト値は -1 で、コミットしません。値 0 の場合は操作の最後にコミットし、1 以上の値の場合は、指定された行数が処理された後にコミットします。
HEADER	FALSE（デフォルト）。olDump の実行中にファイルの先頭にヘッダーを作成する場合は、TRUE を指定します。
BITARRAY	バイナリ形式で NULL の書込みおよび読取りをサポートする場合は、TRUE（デフォルト）に指定します。ダンプ中に、NULL 情報を含むビット配列が各行の前にダンプされます。FALSE に指定すると、バイナリ形式で NULL を書き込もうとしたときにエラーが発生します。
NONULL	TRUE（デフォルト）。NULL の読取りまたは書込みをしようとすると、olLoad および olDump がエラーを返します。このフラグを FALSE に設定すると、NULL がサポートされます。デフォルトの BITARRAY 値が TRUE のため、バイナリ形式の NULL もサポートされます。

表 E-7 パラメータ（続き）

パラメータ	説明
DATEFORMAT	日付列およびタイムスタンプ列を FIXED ASCII または CSV 形式のファイルに対して読取りおよび書込みをするための文字列。「YYYYMMDD」、「YYYY-MM-DD」および「YYYY/MM/DD」などの形式がサポートされています。デフォルト値は空の文字列（これは NULL を使用して設定することもできます）で、デフォルトの日付書式は「YYYY-MM-DD」です。（Oracle モードでは、日付はタイムスタンプと同じとして扱われるため、日付書式はデフォルトのタイムスタンプ書式「YYYY-MM-DD HH:MM:SS.SSSSSS」です。）

E.3 ファイル形式

Oracle Lite ロード・ユーティリティでは、FIXEDASCII、BINARY および CSV という 3 つの形式をサポートしています。各ファイルには、オプションのヘッダーとそれに続くゼロ以上のデータ行が含まれます。

E.3.1 ヘッダー形式

ヘッダーの形式は次のとおりです（説明は太字で示してあります）。

```
$$OL_BH$$ [ 開始ヘッダー ]
VERSION=xx.xx.xx.xx   [ バージョン番号 ]
TABLE=T1(C1, C2, ...)... [ 表名およびダンプされた列名のリスト ]
FILEFORMAT=FIXEDASCII
SEPARATOR=,
[ パラメータ・リストのパラメータをここにリストできます。 ]
$$OL_EH$$ [ 終了ヘッダー ]
```

ヘッダーの例を次に示します。

```
$$OL_BH$$
VERSION=01.01.01.01
TABLE=T1(EMPNO,SALARY)
FILEFORMAT=BINARY
BITARRAY=TRUE
HEADER=TRUE
RDONLY=FALSE
LOGFILE=
COMMITCOUNT=-1
NOSINGLE=TRUE
$$OL_EH$$
```

ヘッダーの行はどのような順序でもかまいません。$$OL_BH$$ および $$OL_EH$$ 以外の行はすべてオプションとみなされます。ただし、ダンプ時に HEADER フラグがオンになっている場合は、表の情報とすべてのパラメータ設定がヘッダー内にダンプされます。

ロードの実行時に、ヘッダー内のパラメータ情報が現在のパラメータ設定を上書きします。olLoad の table 引数が NULL の場合は、ヘッダー内にある表の名前と列のリストが優先します。それ以外の場合は、olLoad の table 引数がヘッダーより優先します。

E.3.2 データ形式

E.3.2.1 CSV 形式

表の各行は、ファイル内で個別の行として表されます。各行は、Windows プラットフォーム上の改行および LF（line feed）により区切られます。行内の値は、それぞれセパレータ文字（デフォルトはカンマ）で区切られます。また、各値は引用符文字で囲まれます。NULL は引用符で囲まれた空の文字列（' '）として表されます。ファイル内の引用符付き文字列の数は、表の列数と同じである必要があります。同じでない場合、エラーが発生します。

E.3.2.2 FixedAscii 形式

表の各行は、ファイル内で個別の行として表されます。各行は、Windows プラットフォーム上の改行および LF（line feed）により区切られます。各行が同じサイズです。列のデータ型が、ファイル内の列の書式または表記を決定します。NULL は、*n* '¥0'（NULL）文字の文字列として表されます。*n* は、フィールドの固定サイズです。各データ型のデータ表記を、次に示します。ファイル内の各行の合計レコード長は、各列のフィールド長（精度）の合計と同じである必要があります。同じでない場合、エラーが発生します。

表 E-8 データ型

データ型	説明
CHAR(n)	フィールドの長さ（*n* 文字単位）。データは左に揃えられ、右は空白で埋められます。
VARCHAR(n)	フィールドの長さ（*n* 文字単位）。データは左に揃えられます。NULL バイト（'¥0'）で埋められます。

表 E-8 データ型（続き）

データ型	説明
NUMERIC(p,s)	デフォルト・モードでは、位取りの s がゼロまたは指定されていない場合、フィールドの長さは p+1 です。それ以外の場合は、フィールドの長さは（p+2）文字になります。値は出力フィールドでは右揃えになります。書式は、オプションの負の符号、ゼロ（必要な場合）、有効桁の順に続きます。負の符号がない場合は、かわりに「0」になります。たとえば、Number(5,2) は次のようになります。 12.3 -> ‘ 012.30’ -12.3 -> ‘-012.30’ 1.23 -> ‘ 001.23’ -1.23 -> ‘-001.23’ カスタム・モードでは、フィールドの長さは 1 減り、位取りがない場合は p、それ以外の場合はゼロおよび p+1 になります。ファイルに実際に格納される数値は、NUMERIC(p-1,s) 型です。NUMERIC(p,s) の範囲内で NUMERIC(p-1,s) の範囲外の数値を挿入しようとすると、olDump によりエラーが発生します。したがって、NUMERIC フィールドの最初の文字は、「0」または「-」に指定する必要があります。指定しない場合は、olLoad によりエラーが発生します。
DECIMAL(p,s)	NUMERIC(p,s) と同じ。
INTEGER	フィールドの長さは 11 文字です。負の符号または空白の後に 10 桁が続きます。 先行桁はゼロで埋められます。
SMALLINT	フィールドの長さは 6 文字です。負の符号または空白の後に 5 桁が続きます。
FLOAT	フィールドの長さは 23 文字です。Oracle モードでは、負の符号または空白、先行ゼロ、いくつかの桁、ピリオド、いくつかの桁の順に続きます。たとえば、次のようになります。 0 -> ‘ 00000000000000000000’ -12.34 -> ‘-000000000000000012.34’ SQL92 モードでは、E（指数）が常に存在し、小数点の前にあるのは 1 桁のみです。たとえば、次のようになります。 0 -> ‘ 00000000000000000000E0’ -12.34 -> ‘-000000000000001.234E10’
REAL	DOUBLE PRECISION の場合と同じ書式です。ただし、フィールド長は 23 文字ではなく 16 文字のみです。

表 E-8 データ型（続き）

データ型	説明
DOUBLE PRECISION	フィールドの長さは 23 文字です。負の符号または空白、22 桁の文字、ピリオドまたは E、浮動小数点数と E、指数桁の順に続きます。Oracle モードでは、数値が指数を使用しなくてもフィールドに収まるくらい小さい場合は、E は使用されません。SQL92 モードでは、E が常に使用されます。浮動小数点の前には有意の桁（0 以外）が常に 1 つあります。 たとえば、SQL92 モードでは次のようになります。 0 -> ‘ 00000000000000000000E0’ -1.79E10 -> ‘-00000000000000001.79E10’ 12 -> ‘ 00000000000000001.2E10’ たとえば、Oracle モードでは次のようになります。 1.2E75 -> ‘ 00000000000000001.2E75’ -1.33333 -> ‘-00000000000000001.33333’ -1.79E10 -> ‘-000000000017900000000’
DATE	SQL92 モードでは、YYYY-MM-DD で 10 文字の長さになります。たとえば、次のようになります。 October 1, 1999 -> 1999-10-01 Oracle モードでは、日付はタイムスタンプとしてダンプされます。 デフォルトの日付書式パラメータでない場合、日付書式は、指定されている日付書式文字列に対応します。たとえば、次のようになります。 DATEFORMAT = "YYYYMMDD" October 1, 1999 -> 19991001
TIME	HH:MM:SS で 8 文字の長さになります。たとえば、次のようになります。 午後 5 時 1 分 58 秒は、17:01:58 です。
TIMESTAMP	日付書式、空白、時間書式、ピリオド、および 6 桁（マイクロ秒の桁数）で、合計 26 文字になります。 YYYY-MM-DD HH:MM:SS.SSSSSS デフォルトの日付書式パラメータでない場合、タイムスタンプ書式は、指定されている日付書式文字列に対応します。日付書式文字列に時間が指定されていない場合は、ファイルにダンプされるときに、タイムスタンプの時間情報が省略されます。

E.4 制限事項

現在、OLLOAD では次の機能をサポートしていません。

- データ型が INTERVAL の列、タイム・ゾーンを使用した時間、タイム・ゾーンを使用したタイムスタンプ、BLOB および CLOB。
- バイナリ・データはサポートされていません。
- サポートされている「var」型は VARCHAR のみです。

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（JSP）

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。

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（Web-to-Go 用）（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 サーブレット、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 Server）

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 用 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 サーバーとともに稼働す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、Oracle データベース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永続リソース・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でのデータベース・アクセスを可能にします。Web-to-Go 用 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、Mobile SQL を使用します。

Oracle データベース

Oracle データベースは、Mobile サーバーのデータベース・コンポーネントです。
Web-to-Go 用 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 Oracle データベースに格納されます。

Oracle Lite

Oracle Lite は、Web-to-Go 用 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され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 用 Mobile クライアント、Mobile サーバーおよび Oracle データ・サーバーで構成される 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 用 Mobile クライアント（Mobile Client for Web-to-Go）

Web-to-Go 用 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 Oracle データベース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 つの順序のうちの 1 つで、オフライン・モードの Web-to-Go 用 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。クライアントが順序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順序を再び作成します。

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。

位置付け DELETE（Positioned DELETE）

位置付け 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
```

位置付け UPDATE（Positioned UPDATE）

位置付け 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 用 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（Online Mode）」も参照。

オンライン・モード（Online Mode）

Web-to-Go 用 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（Offline Mode）」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（Primary Key）」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 用 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。

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、順序、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。

順序（Sequence）

順次数を生成するスキーマ・オブジェクト。順序を作成した後は、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順序が即時増分されます。「WINDOW シーケンス（Window Sequence）」も参照。

スキーマ（Schema）

表、ビュー、索引、順序などを含む、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 用 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 用 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、順序、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 用 Mobile クライアントと Oracle データベース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 Oracle データベース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、INSERT、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 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。

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（Primary Key）」および「外部キー（Foreign Key）」も参照。

モードの切替え（Switching Modes）

Web-to-Go 用 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 Oracle データベース 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順序値を使用して、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

N

O

P

R

S

T

U

V

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

て

と

は

ひ

ふ

ゆ

れ