Oracle9*i* Lite for Windows 32

開発者ガイド

リリース 5.0.2.1.0

2003 年 4 月

部品番号 : J07298-01

ORACLE®

Oracle9*i* Lite for Windows 32 開発者ガイド , リリース 5.0.2.1.0

部品番号 : J07298-01

原本名 : Oracle9*i* Lite Developer's Guide, Release 5.0.2.1.0 for Windows 32

原本部品番号 : B10344-01

目次

3 同期

4 パッケージ・ウィザードの使用方法

5 Consolidator

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

図目次

表目次

はじめに

このマニュアルでは、Oracle Lite リレーショナル・データベース管理システム（Relational Database Management System: RDBMS）とそのコンポーネント、およびデータのレプリケーションと同期を行うためのクライアントおよびサーバーの Application Program Interface（API）の概要を説明します。クライアント・マシン上での Oracle Lite データベースを使用した Windows 32 用のモバイル・アプリケーションの開発、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」	Oracle9*i* Lite の概念およびアプリケーション・アーキテクチャについて説明します。Mobile Development Kit の内容および開発手順の概要を説明します。
第 2 章「Oracle Lite RDBMS」	Oracle Lite RDBMS 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。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」	Oracle9*i* Lite のデータ・レプリケーションおよび同期の概念、機能、アーキテクチャについて説明します。この中には、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ます。
第 4 章「パッケージ・ウィザードの使用方法」	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。

第 5 章「Consolidator」	Mobile サーバーおよびその 2 つの主要なコンポーネントである Consolidator と Resource Manager の概念、機能、アーキテクチャを説明します。この中には、カスタマイズ方法などの高度なトピックが含まれます。
第 6 章「Oracle Mobile SQL（MSQL）」	Oracle Mobile SQL（MSQL）コマンドおよび対応するコマンドラインの使用方法について説明します。
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、Oracle9*i* Lite の概要、および Mobile Development Kit とそのコンポーネントを使用するアプリケーション開発プロセス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 により、多数のモバイル・ユーザーを対象としたオフラインのモバイル・データベース・アプリケーションの開発、配置および管理が簡単になります。オフラインのモバイル・アプリケーションとは、サーバーとの常時接続を必要とせずに、モバイル・デバイスで実行できるアプリケーションです。オフラインのデータベース・アプリケーションの場合は、モバイル・デバイス上にローカル・データベースが存在する必要があります。このデータベースの内容は、エンタープライズ・データ・サーバーに格納されているデータのサブセットです。アプリケーションによってローカル・データベースに加えられる変更は、サーバーのデータと一致するように随時調整します。モバイル・データベースとエンタープライズ・データベース間の変更を一致させるために使用するテクノロジは、データ同期と呼ばれます。

オフラインのモバイル・データベース・アプリケーションは、様々な方法で開発できます。モバイル・プラットフォーム専用のネイティブ C または C++ アプリケーションを開発する方法が最も一般的です。これらのアプリケーションは、Open Database Connectivity（ODBC）インタフェース、Active Data Object（ADO）、または ODBC 上に構築された他のインタフェースのいずれかを使用してデータベースにアクセスします。別の方法として、Java および Java Database Connectivity（JDBC）インタフェースを使用する方法もあります。Oracle9*i* Lite は、Web-to-Go というサーブレット・ベースの Web モデルを使用してオフラインのモバイル・データベース・アプリケーションを開発する方法も提供します。

この章では、Windows 32 システムのみを対象としたアプリケーション開発とデータベースの同期について説明します。Windows CE については、『Oracle9*i* Lite for Windows CE/Pocket PC 開発者ガイド』を参照してください。Palm については、『Oracle9*i* Lite Developer's Guide for Palm』を参照してください。Web-to-Go の詳細は、『Oracle9*i* Lite for Web-to-Go 開発者ガイド』を参照してください。

アプリケーション開発の終了後、そのアプリケーションを配置する必要があります。アプリケーションの配置では、エンド・ユーザーがアプリケーションを簡単にインストールして使用できるようにサーバー・システムを設定することが重要です。Oracle9*i* Lite モバイル・アプリケーションが配置される Mobile サーバーは、アプリケーションのサーバー・システムの中枢をなします。配置は、次の 5 つの手順で構成されます。

1. 必要なレベルのパフォーマンス、拡張性、セキュリティ、可用性および接続性を実現するためのサーバー・システムの設計。Oracle9*i* Lite は、データ同期のパフォーマンスをチューニングするためのツール製品（consperf ユーティリティなど）を提供します。また、拡張性を保持するための容量の見積りに使用可能な、ベンチマーク・データも提供します。認証、認可、暗号化などのセキュリティ保護装置は、適切な標準でサポートされます。可用性および拡張性も、Oracle9*i* Application Server（Oracle9*i*AS）および Oracle9*i*（Oracle データベース・サーバー）のロード・バランシング、キャッシング、透過的なスイッチオーバー・テクノロジによってサポートされます。
2. サーバーへのアプリケーションのパブリッシュ。これは、アプリケーションのすべてのコンポーネントを Mobile サーバーにインストールすることを指します。Oracle9*i* Lite には、Mobile サーバーにアプリケーションをパブリッシュする場合に使用可能なパッケージ・ウィザードというツールが含まれています。

3. モバイル・ユーザーへのアプリケーションの提供。これには、使用するデータのサブセット、提供するアプリケーションおよび提供対象のユーザーの決定などが含まれます。Oracle9*i* Lite には、ユーザーの作成、ユーザーへのアプリケーションの実行権限の付与、ユーザーのデータ・サブセットの定義などを行うコントロール・センターと呼ばれるツールが含まれています。Java API を使用してこれらの操作を行うこともできます。
4. 実際の配置環境での機能およびパフォーマンスのテスト。モバイル・アプリケーション・システムは、様々なモバイル・デバイスのクライアント・テクノロジ（オペレーティング・システム、フォーム・ファクタなど）、様々な接続オプション（LAN、無線 LAN、携帯電話、ワイヤレス・データなどのテクノロジ）、様々なサーバー構成オプションなどが関与する複合システムです。ロールアウト前に実際の環境で行うシステムのテストおよびパフォーマンスのチューニングにかわる手段はありません。パフォーマンスの問題のほとんどの原因が、データのサブセッティング問合せのパフォーマンスにあるため、このパフォーマンスのチューニングは特に注意して行う必要があります。
5. モバイル・デバイスでのアプリケーションの初期インストール（アプリケーションの配布）方法の決定。初期インストールには、Oracle9*i* Lite クライアント、アプリケーション・コードおよび初期データベースのインストールが含まれます。初めてモバイル・デバイスにアプリケーションを初期インストールする場合、非常に多くのデータ量が必要になる可能性があります。そのため、モバイル・デバイスとサーバーの間に信頼性が高い高速の接続を使用するか、またはオフライン・インスタンス化と呼ばれる方法を使用する必要があります。オフライン・インスタンス化の場合、モバイル・デバイスにアプリケーションをインストールするために必要なすべてのものが、CD またはフロッピィ・ディスクに収められ、ユーザーに郵送されます。ユーザーは、このメディアを使用して、デスクトップ・マシンを使用してデバイスにアプリケーションをインストールします。Oracle9*i* Lite には、オフライン・インスタンス化用のツールが含まれています。

配置後、拡張または不具合解消のため、アプリケーションとデータ・スキーマの両方が変更される場合があります。アプリケーションの更新およびデータのスキーマ展開は、Oracle9*i* Lite Mobile サーバーが処理します。管理者は、アプリケーションとデータの再パブリッシュのみを行う必要があります。Mobile サーバーは、旧バージョンのアプリケーションまたはデータを持つモバイル・クライアントを自動的に更新します。

Oracle9*i* Lite のインストールには、Mobile サーバーまたは Mobile Development Kit をインストールするオプションが含まれています。アプリケーション開発では、開発に使用するマシンに Mobile Development Kit をインストールする必要があります。ただし、後述の開発手順例では、Mobile サーバーを稼働状態にする必要があります。そのため、サンプルのアプリケーションをシステムで再作成する場合は、Mobile サーバーを別のマシンにインストールすることをお薦めします。Mobile サーバーをインストールする場合は、Oracle データベースのインスタンスを稼働状態にする必要があります。既存のテスト用データベースをこのインスタンスとして使用できます。Mobile サーバーは、メタデータをこのデータベースに格納します。

1.2 Oracle9*i* Lite のアプリケーション・モデルおよびアーキテクチャ

Oracle9*i* Lite のアプリケーション・モデルでは、各アプリケーションは、パブリケーションを使用してデータ要件を定義します。パブリケーションは、データベース・スキーマに類似していて、1 つ以上のパブリケーション・アイテムが含まれています。パブリケーション・アイテムは、パラメータ化されたビュー定義と同様、バインド変数を指定した SQL 問合せを使用してデータのサブセットを定義します。このようなバインド変数は、サブスクリプション・パラメータまたはテンプレート変数と呼ばれます。

サブスクリプションとは、すべてのメンバー・パブリケーション・アイテムのすべてのサブスクリプション・パラメータに値が指定されているパブリケーションです。サブスクリプションは、ユーザーに対してアプリケーションを提供するために必要で、ユーザー用のデータのサブセットを定義します。

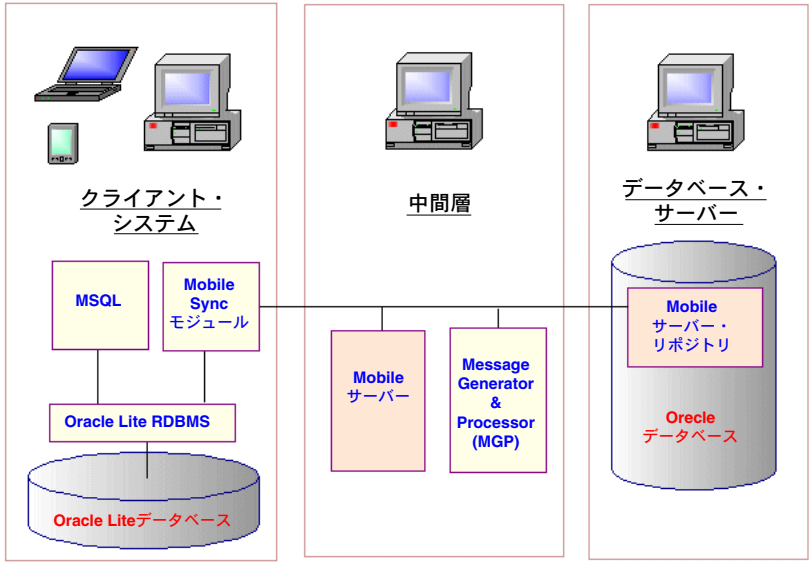
ユーザーが Mobile サーバーに初めてログインすると、Mobile サーバーは、そのユーザーに対して提供されたサブスクリプションごとに、クライアント・マシン上に Oracle Lite データベースを作成します。次に、Mobile サーバーは、サブスクリプションに含まれているパブリケーション・アイテムごとに、このデータベースのスナップショットを作成し、パブリケーション・アイテムに関連付けられた SQL 問合せ（変数をすべてバインド）を実行して、このスナップショットに、サーバー・データベースから取り出したデータを移入します。インストール後、Oracle Lite データベースは、エンド・ユー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のみが必要となります。

ユーザーがアプリケーションを使用している間に Oracle9*i* Lite データベースに加えられた変更は、スナップショットにより取得されます。Mobile サーバーへの接続が使用可能になった時点で、ユーザーは、変更を Mobile サーバーと同期させることができます。Oracle9*i* Lite Mobile Sync アプリケーション（msync）を直接使用するか、またはアプリケーションから Mobile Sync API をプログラムでコールすることにより、同期を開始することができます。Mobile Sync アプリケーションは Mobile サーバーと通信し、クライアント・マシンで行われた変更をアップロードします。次に、Mobile サーバーによってすでに準備されている、クライアントについての変更をダウンロードします。

Message Generator and Processor（MGP）と呼ばれる、Mobile サーバーと同じ層で実行されるバックグラウンド・プロセスは、アップロード済のすべての変更をモバイル・ユーザーから定期的に収集し、それをサーバー・データベースに適用します。次に、MGP は、各モバイル・ユーザーに送信する必要がある変更を準備します。次回モバイル・ユーザーが Mobile サーバーと同期をとる場合、これらの変更をクライアントにダウンロードし、クライアント・データベースに適用することができるため、この手順は非常に重要です。

図 1-1 に、Oracle9*i* Lite アプリケーションの配置アーキテクチャを示します。

図 1-1 Oracle9*i* Lite の配置アーキテクチャ



注意： 図には示されていませんが、Web-to-Go クライアントには、Lightweight HTTP Listener が追加コンポーネントとして存在します。

1.2.1 Oracle Lite DBMS

Oracle Lite RDBMS は、ラップトップ・コンピュータ、ハンドヘルド・コンピュータ、PDA およびその他の情報機器専用に作成された、フットプリントの小さい Java 対応の、安全なリレーショナル・データベースです。Oracle Lite RDBMS は、Windows 98/NT/2000/XP、Windows CE/Pocket PC、Palm Computing、EPOC および Embedded Linux で稼働します。Oracle Lite RDBMS には、JDBC、ODBC および OKAPI プログラミング・インタフェースと呼ばれるオブジェクト指向インタフェースが提供されています。これらのインタフェースを使用すると、Java、C/C++ および Visual Basic 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。このデータベース・アプリケーションは、ユーザーがデータベース・サーバーから切断されている状態でも使用できます。

Mobile Development Kit をインストールすると、Oracle Lite RDBMS および付録 C にリストされているすべてのユーティリティが、開発用マシンにインストールされます。本番システムで、Mobile サーバーにより Oracle9*i* Lite アプリケーションがインストールされると、クライアント・マシンには、RDBMS、Mobile Sync および Mobile SQL アプリケーションのみがインストールされます。

1.2.2 Mobile SQL（MSQL）

MSQL は、ラップトップおよびハンドヘルド・コンピュータ上での Oracle Lite データベースの作成、アクセスおよび操作を可能にするインタラクティブ・ツールです。MSQL を使用すると、次のことができます。

- 表、ビューなどのデータベース・オブジェクトの作成
- 表の表示
- SQL 文の実行

MSQL は、Mobile Development Kit のインストールによってインストールされます。MSQL は、アプリケーション・インストールの一部として Mobile サーバーによってもインストールされます。Windows 32 プラットフォーム用 MSQL は、Oracle SQL*Plus ツールと同様のコマンドライン・ツールです。Windows CE および Palm 用 MSQL は、Graphical User Interface （GUI）をサポートします。

1.2.3 Mobile Sync

Mobile Sync は、モバイル・デバイスに常駐するフットプリントの小さいアプリケーションです。Mobile Sync を使用すると、ハンドヘルド・コンピュータ、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/XP、Windows CE/Pocket PC、Palm Computing、EPOC および Embedded Linux 上で稼働します。

Mobile Sync は、Oracle Lite データベース内のスナップショットを、対応する Oracle データ・サーバー内のデータと同期させます。スナップショットは、Mobile サーバーにより、モバイル・アプリケーションに関連付けられたパブリケーション・アイテムからユーザーごとに作成されます。Mobile サーバーは、同期プロセスの調整も行います。

Mobile Sync アプリケーションは、サポートされているプロトコルのいずれか（HTTP など）を使用して、Mobile サーバーと通信します。Mobile Sync アプリケーションは、モバイル・ユーザーからコールされると、ユーザー情報を収集してから、Mobile サーバーを使用してユーザーを認証します。次に、Oracle Lite データベースに加えられた変更をスナップショットの変更ログから収集し、Mobile サーバーにアップロードします。その後、ユーザーについての変更を Mobile サーバーからダウンロードし、Oracle Lite データベースに適用します。

この基本機能の他に、Mobile Sync アプリケーションには送信データを暗号化、復号化および圧縮する機能もあります。

Mobile Development Kit をインストールすると、Mobile Sync アプリケーションも開発用マシンにインストールされます。また、Mobile Sync は、Mobile サーバーによっても、アプリケーションのインストールの一部として開発用マシンにインストールされます。

スナップショットは、実表およびビューとは異なり、SQL 文を使用して Oracle Lite データベース内に作成することはできません。スナップショットは、アプリケーションに関連付けられたパブリケーション・アイテムから導出されるサブスクリプションに基づき、Mobile サーバーによってのみ作成されます。この機能については、この章の後半および第 4 章で説明します。

1.2.4 Mobile サーバー

Mobile サーバーは、次の機能を提供する中間層サーバーです。

- アプリケーションのパブリッシュ
- アプリケーションの提供
- アプリケーションのインストールおよび更新
- データ同期

Mobile サーバーには、Resource Manager および Consolidator と呼ばれる 2 つの主要モジュールがあります。Resource Manager は、アプリケーションのパブリッシュ、提供およびインストールを行います。Consolidator は、データおよびアプリケーションの同期を行います。

アプリケーションのパブリッシュとは、モバイル・ユーザーに対してアプリケーションを提供できるように、Mobile サーバーにアプリケーションをアップロードすることです。アプリケーションの開発を終了すると、パッケージ・ウィザードと呼ばれる開発ツールを使用して、アプリケーションを Mobile サーバーにパブリッシュできます。

アプリケーションの提供は、ユーザーのサブスクリプションの作成およびアプリケーション実行権限のユーザーへの割当てを行います。アプリケーションの提供には、次の方法も使用できます。

- コントロール・センターと呼ばれる管理ツールを使用すると、ユーザーおよびグループを作成できます。サブスクリプション・パラメータに値を割り当てることにより、ユーザーのサブスクリプションを作成できます。また、アプリケーションを使用する権限をユーザーまたはグループに付与することができます。
- Resource Manager API を使用すると、前述のタスクをプログラムにより実行できます。

エンド・ユーザーは、モバイル・アプリケーションを次の 2 つの手順でインストールします。まず、モバイル・ユーザーとして、Mobile サーバーのセットアップ・ページを参照し、使用するプラットフォーム用のセットアップ・プログラムを選択します。セットアップ・プログラムは、Windows 32 プラットフォームでのみ稼働します。Windows 32 ベースのクライアント・システムの場合は、セットアップ・プログラムを直接 Windows 32 システムにダウンロードして実行すると、Oracle9*i* Lite クライアントをセットアップできます。Windows CE および Palm デバイスの場合、セットアップ・プログラムは、Windows 32 デスクトップにダウンロードしてから実行する必要があります。その後で、ActiveSync（Windows CE の場合）または Hot Sync（Palm の場合）を使用して、デバイスに Oracle9*i* Lite クライアントをインストールします。

次に、モバイル・デバイスで Mobile Sync（msync）コマンドを実行します。このコマンドを実行すると、ユーザー名とパスワードを入力するように要求されます。Mobile Sync アプリケーションは、Mobile サーバーの Consolidator モジュールと通信し、提供されたアプリケーションおよびデータをダウンロードします。

アプリケーションおよびデータのインストールが終了すると、アプリケーションが使用できます。定期的に Mobile Sync またはカスタム・コマンドを使用して、ローカル・データベースをサーバー・データベースと同期させます。この同期により、変更したすべてのアプリケーションも更新されます。

1.2.5 Message Generator and Processor（MGP）

Mobile サーバーの Consolidator モジュールは、変更内容をクライアント・データベースからサーバーにアップロードし、関連するサーバーの変更内容をクライアントにダウンロードします。ただし、Consolidator モジュールは変更内容の調整は行いません。変更内容の調整および変更によって発生した競合の解決は、MGP が行います。MGP は、一定の間隔でサイクルを開始するように制御可能なバックグラウンド・プロセスとして稼働します。

MGP の各サイクルは、適用と構成の 2 つのフェーズで構成されています。

適用フェーズ

適用フェーズで、MGP は、前回の適用フェーズ以降、ユーザーがアップロードした変更を収集し、サーバー・データベースに適用します。変更をアップロードした各ユーザーに対して、各サブスクリプションの変更を単一トランザクションで適用します。トランザクションに失敗すると、ログ・ファイルに理由を書き込み、エラー・ファイルに変更を保存します。

構成フェーズ

適用フェーズが終了すると、MGP は構成フェーズに進み、各クライアントにダウンロードする必要がある変更を準備します。

サーバー・データベースへの変更の適用

データ同期は非同期で行われるため、モバイル・ユーザーは、予期しない結果を受け取る場合があります。その典型的なケースは、ユーザーが、サーバーに接続している他のユーザーも更新しているレコードを更新する場合です。この場合、同期が一巡しても、ユーザーはサーバーの変更内容を受け取らない可能性があります。

この状況は、ユーザーが行う変更とサーバー・データベースに加えられる変更との間で調整が図られていないために発生します。MGP の次のサイクルで、サーバー・データベースとの間で変更が調整され、調整により発生した競合が解決されます。その後、クライアントに変更内容をダウンロードするために、新しいレコードが準備されます。ユーザーは、再度同期をとると（2 回目）、サーバーの変更内容が反映されたレコードを受け取ります。サーバーの変更内容とクライアントの変更内容の間に競合が発生した場合、ユーザーは、競合解決ポリシーの定義に基づいて、サーバーの変更内容またはクライアントの変更内容のいずれかを反映したレコードを受け取ります。

1.2.6 Mobile サーバー・リポジトリ

Mobile サーバー・リポジトリ（リポジトリと略します）には、Mobile サーバーの実行に必要なすべての情報が含まれています。通常、この情報は、アプリケーション・データと同じデータベースに格納されます。アプリケーション・データがリモート・インスタンスに存在し、DBLink を使用するこのリモート・インスタンスに対するシノニムが、Mobile サーバーに定義されている場合のみが例外となります。

MSQL を使用してリポジトリを操作することはできますが、参照にのみ使用してください。リポジトリの更新には、Mobile サーバーのコントロール・センターまたは Mobile サーバーの Resource Manager API のみを使用する必要があります。

1.3 Mobile Development Kit

Oracle9*i* Lite を使用してオフライン・アプリケーションを開発する前に、アプリケーション開発用マシンに Oracle9*i* Lite Mobile Development Kit をインストールする必要があります。Mobile Development Kit をインストールする方法は、『Oracle9*i* Lite for Windows NT/2000/XP 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。

Oracle9*i* Lite Mobile Development Kit には、次のコンポーネントが含まれています。

- Oracle Lite RDBMS －軽量の組込みリレーショナル・データベース管理システム
- パッケージ・ウィザード－アプリケーションを Mobile サーバーにパブリッシュするツール
- Mobile Sync －トランザクション同期アプリケーション

C または C++ 開発ツールを Mobile Development Kit（Windows 用）と組み合せて使用すると、Oracle Lite データベースを対象とする Windows 用モバイル・アプリケーションを開発し、パッケージ・ウィザードを使用してこれらのアプリケーションを Mobile サーバーにパブリッシュできます。Mobile サーバーをインストールする方法は、『Oracle9*i* Lite for Windows NT/2000/XP 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。

アプリケーションを Mobile サーバーにパブリッシュすると、コントロール・センターを使用してモバイル・ユーザーに対してアプリケーションを提供できます。アプリケーションの提供では、特定のユーザーを対象としたアプリケーションで必要なデータのサブセット化に使用するサブスクライブ・パラメータの値を指定する必要があります。アプリケーションの提供がすでに終了しているユーザーは、Mobile サーバーにログインし、Mobile サーバーに、ユーザーのデバイス上でアプリケーションを実行するために必要なすべての設定を行うよう要求できます。

Mobile Development Kit は、<*ORACLE_HOME*>/Mobile/SDK ディレクトリにインストールされます。この Bin ディレクトリには、次のものが含まれます。

- Oracle Lite RDBMS、Mobile SQL（msql.exe）などのコンポーネント（付録 C「データベースのツールとユーティリティ」参照）。Mobile SQL は Java で作成されているため、使用する前に Java Runtime Environment（JRE）1.3 をシステムにインストールする必要があります。JDK 1.3 がインストールされている場合、JRE はすでにマシンにインストールされています。
- Mobile Sync、実行可能ファイル（msync.exe）、COM インタフェースおよび COM インタフェース用 Java ラッパー。
- パッケージ・ウィザード（wtgpack.exe）
- ODBC データ・ソースの作成に使用可能な ODBC Data Source Administrator（odbcad32.exe）。

Examples ディレクトリ <*ORACLE_HOME*>¥Mobile¥SDK¥Examples には、サンプル・アプリケーションがいくつか含まれています。このマニュアルの 2.8 項で、サンプル・プログラムの内容と実行方法を説明します。ソース・コードを調べ、サンプルを実行して、Oracle9*i* Lite の様々な機能を理解しておく必要があります。

<*ORACLE_HOME*>¥Mobile¥SDK¥OLDB40 ディレクトリには、**polite.odb** という初期データベースが含まれています。

Mobile Development Kit をインストールすると、インストーラによって、Mobile Development Kit の Bin ディレクトリを含むように環境変数 PATH が設定されます。Windows 32 マシンでコマンド・プロンプトを使用すると、次の簡単なテストを行うことができます。

コマンド・プロンプトで、次のとおり入力します。

```
msql system/manager@jdbc:polite:polite
...

SQL>create table test (c1 int, c2 int);
Table created
```

```
SQL>insert into test values(1,2)
1 row(s) created
SQL>select * from test;

 C1 | C2
----+----
  1 | 2

SQL>rollback;
Rollback completed
SQL>exit
```

1.4 開発手順

簡単なモバイル・フィールド・サービスの例を使用して、開発手順を示します。モバイル・フィールド・サービス担当者が 1 日に行う作業に関する情報が含まれている TASK 表がサーバーに存在するとします。TASK 表の構成は、TASK(ID number(4) primary key、Description varchar(40) not null、CustName varchar(30) not null、CustPhone varchar(12)、CustStAddr varchar(40) not null、CustCity varchar(40) not null、Notes varchar(100) です。TASK 表の各行には、顧客のサイトで行う作業が記述されています。

また、Tom、Dick、Harry の 3 名のサービス担当者がいるとします。Tom にはクパチーノ市の作業を、Dick にはマウンテンビュー市の作業を、Harry にはパロアルト市の作業を割り当てます。このアプリケーションは、次のように機能すると想定できます。

各サービス担当者はラップトップ・コンピュータを持ち、それを使用して毎朝作業リストを入手します。各担当者は、日中その作業を行い、作業の NOTES 列にその日に行った作業についての情報を入力して更新します。終業時に、変更内容をサーバーにアップロードします。

次のアプリケーション環境を前提とします。

- Mobile サーバーを、`mserver` というマシンにインストールします。
- アプリケーション・データおよび Mobile サーバー・リポジトリを格納するために使用するテスト用 Oracle データベースを、リスナーがポート 1521 でリスニングする `oradbserver` というマシンにインストールします。Oracle データベースのインスタンス名は、`orcl` です。`master` というパスワードを持つユーザー `master` として、データベースにログインするとします。`master` は任意のユーザーに置き換えることができます。ただし、そのユーザーは適切な権限を所有している必要があります。
- Mobile Development Kit は、開発用マシンにすでにインストールされているものとします。

実装プランは次のとおりです。各手順のコマンドの正確なシーケンスについては、後述します。

1. oradbserver に TASK 表を作成し、その表にいくつかの行を挿入します。データベースに TASK と同様の表がすでに含まれている場合、この手順は不要です。
2. パッケージ・ウィザードを使用して、モバイル・フィールド・サービスというアプリケーションを定義します。TASK 表に基づくパブリケーション・アイテムを、アプリケーションに対して 1 つ作成します。アプリケーション（アプリケーション・ファイルなし）を、Mobile サーバーにパブリッシュします。
3. コントロール・センターを使用して、Tom、Dick、Harry というユーザーを Mobile サーバーに作成します。モバイル・フィールド・サービス・アプリケーションを実行する権限をすべてのユーザーに付与し、ユーザーごとにサブスクリプションを作成します。
4. 開発用マシンの個別のディレクトリに、Oracle9*i* Lite クライアントをインストールします（サービス担当者のマシンをエミュレートします）。Mobile Sync アプリケーションを実行して、モバイル・フィールド・サービス・アプリケーション（この時点では、空の状態）とデータをダウンロードします。
5. 開発用マシンで MSQL を使用して、TASK スナップショット内の行を調べ、NOTES 列に注意事項を入力して、行を更新します。
6. Mobile Sync アプリケーションを再度実行して、スナップショットに加えた変更をサーバー・データベースと同期させます。
7. サーバー・データベースに接続し、変更が反映されていることを確認します。クパチーノ市の顧客についての行の DESCRIPTION を変更します。
8. Mobile Sync アプリケーションを再度実行します。サーバーで行った変更が、クライアント・データベース内のスナップショットに反映されます。
9. 次のことを行うために、Oracle Lite データベースを対象とした C または C++ プログラムを作成します。
   - サービス担当者に作業を指示する。
   - サービス担当者に作業を選択させ、作業についての注意事項を入力させる。
10. パッケージ・ウィザードを使用して、前述のプログラムが含まれるようにアプリケーションを更新します。

これで、Mobile サーバーは実際の環境でのテストが実行可能な状態になります。

1.4.1 コマンド・シーケンス

コマンド・シーケンスは次のとおりです。

1.4.1.1 ステップ 1: サーバー・データベースに TASK 表を作成

Oracle9*i* Thin JDBC ドライバを使用して、oradbserver マシンで稼働している Oracle データベースに接続します。Thin JDBC ドライバ・ファイル（<ORACLE_HOME>¥jdbc¥lib¥classes12.zip）が環境変数 CLASSPATH 内に存在することを確認します。`master` というパスワードを持つ `master` として接続します。

```
D:> msql master/master@jdbc:oracle:thin:@oradbserver:1521:orcl
```

このデータベースに TASK 表を作成します。サーバー・データベースを作成し、移入するための SQL スクリプトは、次のディレクトリにあります。
<Oracle_Home>¥mobile¥sdk¥examples¥MFS¥src

```
SQL>create table TASK(

1> ID number(4) primary key,
2> Description varchar(40) not null,
3> CustName varchar(30) not null,
4> CustPhone varchar(12),
5> CustStAddr varchar(40) not null,
6> CustCity varchar(40) not null,
7> Notes varchar(100));
```

次に、この表に 4 つの列を挿入します。

```
SQL> insert into task values(1,'Refrigerator not working','Able','408-999-9999','123
Main St.','Cupertino',null);
SQL> insert into task values(2,'Garbage Disposal broken','Baker','408-888-8888','234
Central Ave','Cupertino',null);
SQL> insert into task values(3,'Refrigerator makes
noise','Choplin','650-777-7777','1 North St.','Mountain View',null);
SQL> insert into task values(4,'Faucet leaks','Dean','650-666-6666','10 University
St.','Palo Alto','Beware of dogs');
SQL> commit;
SQL> exit
```

1.4.1.2 ステップ 2: パブリケーション・アイテムの定義と、アプリケーションへのパブリッシュ

パッケージ・ウィザードを使用して、アプリケーション用のパブリケーション・アイテムを作成します。現行バージョンのパッケージ・ウィザードについて、注意する点がいくつかあります。第一に、パッケージ・ウィザードでは、パブリケーションおよびパブリケーション・アイテムの概念自体は公開されませんが、スナップショットというクライアント・データベースの概念がサポートされています。そのため、パブリケーション・アイテムは、「スナップショット」という画面で作成します。

第二に、パッケージ・ウィザードは、アプリケーション名の入力を要求するプロンプトを表示します。パッケージ・ウィザードは、このアプリケーション名を使用して、アプリケーションを識別し、アプリケーションからパブリケーション名およびパブリケーション・アイテム名を生成します。このウィザードは、スナップショット定義をパブリケーション・アイテム定義として使用します。

第三に、パッケージ・ウィザードは実表からのスナップショット（パブリケーション・アイテム）の作成のみをサポートします。パッケージ・ウィザードを使用して、ビューからスナップショットを作成することはできません。これを行うには、Resource Manager および Consolidator API を使用するプログラムを作成する必要があります。ビューからスナップショットを定義するための Java プログラムのサンプル（ViewSnapshotExample.java）は、<*ORACLE_HOME*>¥Mobile¥SDK¥Examples¥Consolidator¥ViewSnapshot¥src にあります。Oracle9*i* Lite の次回のリリースには、ビューからのスナップショットの作成をサポートするパッケージ・ウィザードの拡張機能が含まれます。

第四に、パッケージ・ウィザードは、複数のプラットフォームを対象としたパブリケーションおよびパブリケーション・アイテムを同時に作成できます。パッケージ・ウィザードにより作成されたパブリケーションおよびパブリケーション・アイテムは、Resource Manager API を使用して検索できます。

パッケージ・ウィザードを使用するには、コマンド・プロンプトで次のとおり入力します。

```
d:¥> wtgpack
```

パッケージ・ウィザードの最初の画面が表示されます。

「新規アプリケーションの作成」オプションを選択し、「**OK**」をクリックします。

次の画面で、「使用可能プラットフォーム」リストから「Win32 ネイティブ」プラットフォームを選択し、下矢印をクリックしてこ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。これにより、Windows 32 アプリケーションが作成されることをパッケージ・ウィザードに伝えます。「**次へ**」ボタンをクリックします。

次の画面は、アプリケーション情報を入力するためのものです。アプリケーション名は、「モバイル・フィールド・サービス」とします。このアプリケーションを、Mobile サーバーの /MFS ディレクトリにパブリッシュします。すべてのアプリケーション・ファイルは、開発用マシンの D:¥MFSDEV¥Win32 ディレクトリに格納されます。これには、Windows 32 アプリケーション用の Development ディレクトリの下にある Win32 サブディレクトリを使用する必要があります。パッケージ・ウィザードで使用するディレクトリのネーミング規則については、第 4 章「パッケージ・ウィザードの使用方法」を参照してください。

画面に次の情報を入力します。

アプリケーション名 : モバイル・フィールド・サービス

仮想パス : /MFS

説明 : フィールド・サービス・タスクの割当て

ローカル・アプリケーションのディレクトリ : D:¥MFSDEV（Win32 サブディレクトリは指定できません）

「**次へ**」ボタンをクリックします。

次の画面では、アプリケーションが必要とする実行可能ファイル、イメージ・ファイルなどのすべてのファイルを追加できます。この画面には、D:¥MFSDEV¥Win32 ディレクトリが読み込まれ、このディレクトリに含まれているすべてのファイルが表示されます。ここでは、スナップショットの作成のみを行うため、ファイルの追加は行いません。「**次へ**」ボタンをクリックします。

次の画面を使用して、サーバー・データベースへのログインに使用するユーザー名と、クライアント・データベース用に使用するデータベース名を入力します。次の情報を入力します。

サーバー側

データベースのユーザー名 : master

クライアント側

データベース名 : MFS

「**次へ**」ボタンをクリックします。

次の画面では、パブリケーション・アイテムが定義できます。このパブリケーション・アイテムがクライアント・データベースでのスナップショットになります。作成しているアプリケーションは Win32 用であるため、「プラットフォーム」と書かれたプルダウン・メニューから「WIN32」を選択します。サーバーで定義した TASK 表に基づいてパブリケーション・アイテムを作成します。この項目を作成するには、この表をパッケージ・ウィザードにインポートします。画面下部にある「インポート」ボタンをクリックします。サーバーのログイン情報を入力するように要求されます。次の情報を入力します。

ユーザー名 : `master`

パスワード : `master`

データベース URL: `jdbc:oracle:thin:@oradbserver:1521:orcl`

「**OK**」ボタンをクリックします。

使用可能な表をすべてリストしたダイアログ・ボックスが表示されます。TASK 表を選択し、「**追加**」ボタンをクリックします。「**閉じる**」ボタンをクリックします。「スナップショット」表に TASK 表が表示されます。TASK 表の行を選択し、「**編集**」ボタンをクリックします。

次の画面では、スナップショットのサブセッティング問合せを入力できます。増分リフレッシュ（高速リフレッシュ）できる更新可能なスナップショットを作成します。更新で競合が発生した場合は、サーバーの変更内容を優先させるために次の情報を入力します。

比率の値を 1 に設定し、「Win32」タブをクリックします。比率は、第 5 章「Consolidator」の 5.2.8.3 項「表の比率の使用」を参照してください。

次の画面に、次の情報を入力します。

クライアントで作成 : 選択します。

更新可能 : 選択します。

競合解決 :「サーバー優先」オプションを選択します。

リフレッシュ・タイプ :「高速リフレッシュ」オプションを選択します。

テンプレート :「* from master.task where CustCity = :city」を選択します。

「**OK**」をクリックします。前の画面に戻ります。テンプレート問合せには、「city」という変数（サブスクリプション・パラメータ）が存在します。後で、ユーザーに対してアプリケーションを提供する場合、その値を入力するように要求されます。

「**終了**」をクリックします。ダイアログ・ボックスが表示されます。「現行のアプリケーションのパブリッシュ」オプションを選択し、「**OK**」をクリックします。Mobile サーバーに関する情報を入力するように要求するダイアログ・ボックスが表示されます。

次の情報を入力します。

Mobile サーバーの URL: `mserver`

Mobile サーバーのユーザー名 : `Administrator`

Mobile サーバーのパスワード : `admin`

リポジトリのパス : `/MFS`

「**OK**」をクリックします。「アプリケーションがパブリッシュされました。」というメッセージが表示された場合は、「**OK**」をクリックしてから「**終了**」をクリックします。ファイルがなく、パブリケーション・アイテムを 1 つ持つアプリケーションが、正常にパブリッシュされました。

1.4.1.3 ステップ 3: ユーザーおよびサブスクリプションの作成

Mobile サーバーでユーザーを作成するには、コントロール・センターを使用します。コントロール・センターを使用するには、administrator として Mobile サーバーにログインする必要があります。Mobile サーバーにログインするには、Web ブラウザで次の操作を行います。

アドレス・フィールドに次の URL を入力して、Mobile サーバー・ページを参照します。

```
http://mserver/webtogo
```

（過去の経緯により、URL では、「mobileserver」ではなく「webtogo」を使用します。）

「ログイン」ページが表示されます。次の情報を入力します。

ユーザー名 : Administrator

パスワード : < パスワード > （たとえば、admin）

「**ログオン**」ボタンをクリックします。

次の画面に、現行アプリケーションが表示されます（「アプリケーション」画面が表示されない場合は、画面上部付近の「アプリケーション」タブをクリックします）。「**コントロール・センター**」リンクをクリックします。

「コントロール・センター」画面が表示されます。画面上部付近の「ユーザー」タブをクリックします。「ユーザー」画面が表示されます。この画面を使用して、Tom、Dick、Harry というユーザーを作成します。ここでは、次のコマンドを使用して Tom というユーザーを作成する方法のみを示します。

画面左側にある「**ユーザーの作成**」リンクをクリックします。ダイアログ・ボックスが表示されます。次の情報を入力します。

表示名 : Tom Jones

ユーザー名 : Tom

パスワード : tomjones

パスワードの確認入力 : tomjones

システム権限 : ユーザー

「**保存**」ボタンをクリックします。ユーザーが作成されたことを知らせるメッセージ・ボックスが表示されます。メッセージ・ボックスの「**OK**」ボタンをクリックします。

「ユーザー」画面の左側の表示が変り、このユーザーに対して実行できる機能が表示されます。「**アクセス**」リンクをクリックします。画面の右側に、すべての使用可能なアプリケーションが表示されます。「アクセス」ボックスを選択して、モバイル・フィールド・サービス・アプリケーションを選択します。「**保存**」ボタンをクリックします。メッセージ・ボックスが表示されます。メッセージ・ボックスの「**OK**」ボタンをクリックします。これで、モバイル・フィールド・サービス・アプリケーションを実行する権限が、Tom に付与されます。次に、Tom がアプリケーションをインストールした場合にデータベースに適切なデータのサブセットが作成されるように、Tom のサブスクリプションを定義する必要があります。

画面左側にある「**データ・サブセッティング**」リンクをクリックします。これにより、Tom のサブスクリプションを作成できるようになります。画面右側にある「**Win32**」ボタンをクリックします。画面右側に、2 つの領域が追加されます。「データ・サブセッティング・パラメータ」と書かれた領域では、サブスクリプション・パラメータの値をアプリケーションに関連付けられたパブリケーションに入力できます。モバイル・フィールド・サービス・アプリケーションの場合、パブリケーション・アイテムは 1 つのみで、このパブリケーション・アイテムには「city」というサブスクリプション・パラメータのみ存在します。「city」の値として Cupertino と入力し、「**保存**」ボタンをクリックします。表示されるメッセージ・ボックスの「**OK**」ボタンをクリックします。

これで、Tom というユーザーが作成され、モバイル・フィールド・サービス・アプリケーションを実行する権限が Tom に与えられ、クパチーノ市でのすべての作業が Tom に割り当てられます。

Dick および Harry に対して、前述の手順を繰り返します。

1.4.1.4 ステップ 4: Oracle9*i* Lite クライアント、モバイル・フィールド・サービス・アプリケーションおよびデータのインストール

本番システムでは、Tom、Dick、Harry などのモバイル・ユーザーは、Mobile サーバーのセットアップ・ページにアクセスし、Oracle9*i* Lite の Windows32 クライアントをダウンロードします。次に、Mobile Sync アプリケーションを実行して、モバイル・フィールド・サービス・アプリケーションおよびデータの適切なサブセットをダウンロードします。その後、モバイル・フィールド・サービス・アプリケーションを使用し、随時 Mobile Sync を実行して、データをサーバーと同期させます。

ただし、これは開発段階にあり、実際のモバイル・フィールド・サービス・アプリケーションの開発は完了していません。現段階では、インストールおよび初期の同期により、TASK というスナップショットを持つクライアントの Oracle Lite データベースが作成されるのみです。

開発者は、ユーザー Tom としてインストールし、初期の同期を行って、スナップショットを持つ Oracle Lite データベースを取得します。そして、同期プロセスをテストしてから、アプリケーションの開発に進みます。

Mobile Development Kit がインストールされているマシンを使用して、ブラウザを表示し、Mobile サーバーのセットアップ・ページを参照します。このページの URL は、mserver/webtogo/setup です。ここに、プラットフォームのリストが表示されます。適切なリンクをクリックし、ディレクトリ（たとえば、d:¥MFS）を選択してクライアントをインストールして、Win32 用モバイル・クライアントをダウンロードします。ディレクトリを参照し、その構造を理解してください。コマンド・プロンプトで次のように入力します。

```
C:>D:

D:>cd MFS¥Mobile¥bin

D:¥MFS¥Mobile¥bin>msync
```

これにより、アプリケーション・インストールの一部としてダウンロードされた Mobile Sync アプリケーションが実行されます（¥sdk¥bin ディレクトリにある Mobile Sync アプリケーションも実行できます）。ダイアログ・ボックスが表示されます。次の情報を入力します。

ユーザー名 : Tom

パスワード : tomjones

サーバー : mserver

「**同期**」ボタンをクリックします。同期の進行状況を示すメッセージ・ボックスが表示されます。同期の完了後、Mobile Sync アプリケーションのダイアログ・ボックスにある「**取消**」ボタンをクリックします。

これで、開発用マシンに Oracle Lite データベースがインストールされます。このデータベースには、TASK というスナップショットが存在します。このスナップショットには、行が 2 つありますが、いずれの行の CustCity 列も「Cupertino」です。これらは、クパチーノ市の顧客からのサービス要求で、これらの作業は Tom に割り当てられています。

初期の同期プロセスにより、tom_mfs（ユーザー名の後に、アンダースコア、データベース名が続きます）という ODBC データ・ソース名（DSN）も作成されます。

1.4.1.5 ステップ 5: TASK スナップショットの参照と行の更新

コマンドラインで次のように入力します。

```
D:>MFS¥Mobile¥bin>msql system/manager@jdbc:polite:tom_mfs
SQL> select * from task;
```

2 つの行が表示されます。

```
SQL> update task set Notes ='Replaced the motor:$65' where ID = 1;
```

1 行更新されました

```
SQL> commit;
```

コミットが完了しました

```
SQL> exit
```

TASK スナップショットの 1 行が正常に更新されました。

1.4.1.6 ステップ 6: 変更内容とサーバーの同期

変更内容をサーバーと同期させる前に、MGP プロセスが稼働していることを確認する必要があります。この確認を行うには、ステップ 3 の指示に従って、管理者として Mobile サーバーにログインします。画面上部近くの「サーバー」タブをクリックします。画面左側にある「MGP コントロール」リンクをクリックします。「現在のモード」に「停止」が表示された場合は、「スタート」ボタンをクリックして、MGP プロセスを開始します。

ステップ 4 に説明されているとおり、開発用マシン上で Mobile Sync アプリケーションを実行します。同期が正常に完了すると、サーバー・データベースに変更内容が反映されます。

1.4.1.7 ステップ 7: サーバー変更内容の確認と、サーバー・レコードの変更

サーバー・データベースに接続し、次の SQL 文を発行します。

```
D:> msql master/master@jdbc:oracle:thin:@oradbserver:1521:orcl

SQL> select * from task;
```

変更内容が表に反映されます。次に、この表に変更を加えます。

```
SQL> update task set description = 'Garbage Disposal Leaking',
Notes= 'Urgent:house is getting flooded' where id = 2;

1 行更新されました

SQL> commit;

コミットが完了しました

SQL> exit
```

1.4.1.8 ステップ 8: サーバーの変更内容を取得するための再同期

ステップ 4 に説明されているとおり、開発用マシン上で Mobile Sync アプリケーションを実行します。同期が正常に完了した後、次のように入力します。

```
D:>MFS¥Mobile¥bin>msql system/manager@jdbc:polite:tom_mfs

SQL> select * from task;
```

2 つの行が表示されます。サーバーで行った変更内容は、2 番目の行に表示されます。

1.4.1.9 ステップ 9: Oracle Lite データベースを使用した、モバイル・フィールド・サービス・アプリケーションの開発

Mobile Development Kit には、MFS.exe というサンプルの ODBC プログラムが付属しています。このプログラムは、次のディレクトリにあります。
<ORACLE_HOME>¥Mobile¥Sdk¥Examples¥MFS¥bin

(¥src ディレクトリには、このプログラムのソースと Make ファイルがあります。)

このサンプル・プログラムは、非常に単純で、UI ウィジェットは使用しません。このプログラムは、タスク・リストを表示し、注意事項の入力先となる作業のタスク ID を入力するように要求するプロンプトを表示します。タスク ID を -1 と入力すると、プログラムは終了します。無効なタスク ID を入力すると、注意事項の入力を要求するプロンプトが表示されます。引用符を付けずに、注意事項を入力します。このサンプル・プログラムは、自由に手を加えてください。

このプログラムを Mobile サーバーにパブリッシュするには、mfs.exe ファイルを D:¥MFSDEV¥Win32 ディレクトリにコピーします。

1.4.1.10 ステップ 10: アプリケーション・プログラムを使用した、アプリケーションの再パブリッシュ

パッケージ・ウィザードを使用して、アプリケーションを再パブリッシュします。コマンドラインで次のように入力します。

```
D:>wtgpack
```

最初の画面で、「既存のアプリケーションを編集」オプションを選択します。ドロップダウン・リストから「モバイル・フィールド・サービス」を選択し、「**OK**」ボタンをクリックします。

次の画面で、「**ファイル**」タブをクリックします。「ファイル名」ウィンドウに **MFS.exe** ファイルが表示されます。「**OK**」をクリックします。

次の画面で「現行のアプリケーションのパブリッシュ」オプションを選択し、「**OK**」をクリックします。Mobile サーバーのログイン情報を入力するように要求されます。ログイン情報を入力した後、「**OK**」ボタンをクリックします。メッセージ・ボックスに、Mobile サーバーにこのアプリケーションがすでに存在し、それを上書きするかどうかを確認する警告メッセージが表示されます。「**はい**」ボタンをクリックします。

「アプリケーションがパブリッシュされました。」というメッセージが表示された場合は、「**OK**」をクリックしてから「**終了**」をクリックします。これで、**mfs.exe** というファイルと 1 つのパブリケーション・アイテムを持つアプリケーションが、正常に再パブリッシュされます。

別の Windows 32 マシンを使用して、アプリケーションをテストします。ステップ 4 に従って、Oracle9*i* Lite クライアントおよびモバイル・フィールド・サービス・アプリケーションをインストールします。D:¥MFS¥Mobile¥oldb40¥TOM¥mfs.exe プログラムを実行して、モバイル・フィールド・サービス・アプリケーションを実行します。いずれかの作業についての注意事項を入力します。D:¥MFS¥Mobile¥bin¥msync.exe を実行して、変更内容をサーバーと同期させます。

2

Oracle Lite RDBMS

この章では、Oracle Lite リレーショナル・データベース管理システム（RDBMS）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 RDBMS は、フットプリントが小さく、管理が不要なリレーショナル・データベース管理システムで、ODBC および JDBC インタフェースをサポートしています。SQL92 言語および拡張子の一部をサポートしています。モバイル・クライアントのローカル RDBMS として使用するために設計されています。Oracle Lite データベースに格納されているデータは、Oracle9*i* のような Oracle サーバー・データベースに格納されているデータと同期できます。

2.2 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。

2.2.1 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）

ADO などの ODBC または JDBC データ・ソースをサポートするいずれのインタフェースも、Oracle Lite データベースにアクセスするために使用できます。インタフェース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。

2.2.1.1 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（マルチ・ユーザー・バージョンの場合はタイプ 4）で JDBC をサポートします。このドライバは JDBC コールを解析して、Oracle Lite データベースに渡します。

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

2.2.1.2 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手続き型コール・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、2.8 項「Oracle Lite のサンプルの使用」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。

2.2.1.3 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。

2.2.2 Mobile Sync Client モジュール・アプリケーション・プログラミング・インタフェース（API）

この API を使用すると、アプリケーションはデータ同期プロセスをプログラムで制御できます。アプリケーションは、Mobile Sync API の機能を起動してデータ同期プロセスを開始し、Mobile Sync API によって生成されたエラー・メッセージを取得します。Mobile Sync API の詳細は、第 3 章「同期」の 3.4 項「Mobile Sync アプリケーション・プログラミング・インタフェース（API）」を参照してください。

2.2.3 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、Oracle Lite データベース内の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。OLLOAD の詳細は、付録 C「データベースのツールとユーティリティ」の C.9 項「OLLOAD」を参照してください。

2.3 初期データベースの使用

Mobile Development Kit をインストールすると、ODBC データ・ソース名（DSN）POLITE および **POLITE.ODB** という初期データベースが作成されます。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、コマンド・プロンプトで次のコマンドを使用します。

```
C:>msql system/any@jdbc:polite: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。

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

注意： 詳細は、2.5.4 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。

2.4 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。

2.4.1 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 ディレクトリに格納されます。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・ディレクトリに格納す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、2.4.2 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。

2.4.2 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 98/NT/2000/XP 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、および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、付録 C「データベースのツールとユーティリティ」の C.7 項「ODBC Administrator と Oracle Lite ODBC ドライバ」を参照してください。

2.4.3 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 mydsn mydbname
```

たとえば、次のようになります。

```
CREATEDB polite newdb
```

mydsn は DSN 名で、mydbname は新規データベース名です。

詳細は、付録 C「データベースのツールとユーティリティ」の C.2 項「CREATEDB」を参照してください。

2.4.4 新規データベースへの接続

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

```
C:> msql system/manager@jdbc:polite:mydsn
```

mydsn は事前に定義した ODBC データ・ソース名に置き換えられます。

2.5 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、DBA または ADMIN 権限が必要です。

2.5.1 事前定義済ロール

Oracle Lite では、便宜上、事前定義済ロールに、権限をいくつか組み合せています。多くの場合、別のスキーマ内の特定の権限を付与するより、ユーザーに事前定義済ロールを付与する方が簡単です。Oracle Lite は、ロールの作成および削除はサポートしていません。Oracle Lite の事前定義済ロールの一覧を次に示します。

表 2-1 事前定義済ロール

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
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。

2.5.2 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>
```

ここで、<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 **mydbname.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場合は、**.opw** ファイルおよび **.plg** ファイルを含める必要があります。

2.5.3 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。

ユーザーのスキーマにオブジェクトが含まれていない場合にユーザーを削除するには、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。

2.5.4 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>
```

2.5.5 ロールの付与

ADMIN または DBA 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。

2.5.6 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 のいずれかの場合、ユーザーは表に対して該当権限を持ちます。

2.5.7 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>
```

2.5.8 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>
```

2.5.9 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。

2.5.10 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。

2.5.11 データベースのバックアップ

Oracle Lite データベースには、1 つのファイルおよび複数の依存ログ・ファイルが存在します。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。

2.5.12 データベースの暗号化と復号化

ENCRYPDB および DECRYPDB という 2 つのユーティリティを使用すると、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 つのユーティリティでは、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 ビットの暗号化鍵を導出するために使用します。Oracle Lite では、CAST5 と呼ばれるデータ暗号化規格（DES）が使用されます。

この 2 つのユーティリティの詳細は、付録 C「データベースのツールとユーティリティ」の C.4 項「ENCRYPDB」および C.3 項「DECRYPDB」を参照してください。

2.6 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。

2.6.1 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。

2.6.2 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。

2.6.3 分離

トランザクションは互いに独立しています。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、表 2-2 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 2-2 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バージョンの一時スナップショットを作成することによっては、SELECT 文のロックを取得しません。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、同じ SELECT 文の再実行を行うと、異なるデータ・セットが戻される場合があります。 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。

表 2-2 分離レベル（続き）

分離レベル	説明
REPEATABLE READ	この分離レベルでは、戻されるすべての行に対して、問合せが読込み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、また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込み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較すると、同時実行性が低くなります。 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、このロックは排他ロックになります。排他ロックは、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。この行が更新されずに次の行がフェッチされる場合は、更新ロックが読込みロックに降格し、他のトランザクションがその行を読み込めるようになります。
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。
SINGLE USER	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込み」、「非リピータブル・リード」および「仮読込み」の説明）は、ODBC のドキュメントを参照してください。

2.6.3.1 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。

2.6.3.2 ロック

Oracle Lite では行レベルのロックをサポートします。行が読み込まれると、その行は読込みロックされます。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、読込みロックされた同一の行を読み込むことができます。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。

2.6.3.3 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 の値は、READ COMMITTED では RC、REPEATABLE READ では RR、SERIALIZABLE では SR、SINGLE USER では 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、2.6.3.4 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。

2.6.3.4 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・タイプの組合せを表 2-3 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 2-3 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。

2.6.4 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。設計を行う前に、Oracle Lite で使用可能な各機能を注意深く検討し、要件に最適な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。

2.7 スナップショット定義の作成

オフライン・アプリケーションで操作するデータは、実表またはスナップショットとして Oracle Lite データベースに格納されます。実表は、CREATE TABLE SQL 文を使用して作成できます。実表は、サーバー・データとは独立したデータを格納します。これらに対する変更は、サーバー・データベースとは同期しません。

スナップショットは、サーバー・データのサブセットを格納します。スナップショットに対して行われた変更は、サーバー・データと同期できます。ただし、スナップショットは SQL 文を使用して Oracle Lite データベース内に作成できません。スナップショットは、アプリケーション・インストールの一部として、Mobile サーバーによって作成されます。これらは、Mobile サーバーで定義されたパブリケーション・アイテムに基づき作成されます。パブリケーション・アイテムには、スナップショットに格納する必要がある、サーバー・データのサブセットを定義するパラメータ化された SQL 問合せが含まれます。

ほとんどの場合、アプリケーションで使用するスナップショットを作成する対象の表またはビューは、すでにサーバーに存在しています。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・アイテムを作成できます。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。パブリケーション・アイテムおよび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・アイテムを作成します。これがお薦めの方法です。ただし、パッケージ・ウィザードの現行バージョンでは、ビューからのパブリケーション・アイテムの作成はできません。
2. プログラムによるスナップショット定義の作成ー Consolidator API を使用して、パブリケーション・アイテムをプログラムによって作成します。

2.7.1 宣言によるスナップショット定義の作成

この方法では、Oracle9*i* Lite の GUI ベースのツールであるパッケージ・ウィザードを使用します。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。

- 実表がサーバー・データベースに存在するかどうかを確認します。存在しない場合は作成します。
- パッケージ・ウィザードを使用して、アプリケーションおよびパブリケーション・アイテム（スナップショット定義）を定義します。
- パッケージ・ウィザードを使用して、Mobile サーバーにアプリケーションをパブリッシュします。これにより、アプリケーションに関連するパブリケーション・アイテムが作成されます。
- コントロール・センターを使用して、特定のユーザーのサブスクリプションを作成します。
- 開発用マシンにアプリケーションをインストールします。
- 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。

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

2.7.2 プログラムによるスナップショット定義の作成

スナップショット定義を作成する 2 つ目の方法は、Consolidator API を使用して、プログラムによって Mobile サーバーにパブリケーション・アイテムを作成する方法です。この方法は、より複雑で、Oracle9*i* Lite アプリケーション・モデルの知識を必要としますが、第 5 章「Consolidator」で説明する、ビューからのパブリケーション・アイテムの作成、およびスナップショットを構築するためのコードのカスタマイズなどの製品の全機能を提供します。Consolidator 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。サブスクリプションを作成するには、次の手順を実行する必要があります。

- パブリケーションの作成
- パブリケーション・アイテムの作成およびパブリケーションへの追加
- ユーザーの作成
- パブリケーションに基づくユーザーの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、アプリケーションで必要とされるオブジェクト（パブリケーション・アイテム、それらの索引、プラットフォーム固有の情報など）を整理するために使用する Mobile サーバー・オブジェクトです。パブリケーションは、Consolidator API を使用して作成できます。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・アイテムの作成

パブリケーション・アイテムは、クライアントにレプリケートする親表、ビューまたはシノニムのデータ・サブセットを指定する SQL SELECT 文を含む、Mobile サーバー・オブジェクトです。パブリケーション・アイテムは、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・アイテムは、Consolidator API を使用して作成できます。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、Consolidator API を使用してユーザーを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関連付ける Mobile サーバー・オブジェクトです。サブスクリプションは、Consolidator API を使用して作成できます。クライアント・データベースの作成にサブスクリプションを使用するには、パブリケーションの各パラメータに値を指定する必要があります。Consolidator API の SetSubscriptionParameter メソッドを使用して、各パラメータに値を割り当てます。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 5 章「Consolidator」の 5.1 項「Consolidator API を使用するパブリッシュ・サブスクライブ・モデル」を参照してください。

2.8 Oracle Lite のサンプルの使用

次の項では、Oracle Lite のサンプルの使用方法について説明します。

2.8.1 概要

Oracle Lite のインストールを完了すると、<*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。ツール、サンプルの場所および説明を表 2-4 に示します。

表 2-4 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
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、2.8.2 項「BLOB Manager のサンプルに関する注意」を参照してください。

表 2-4 サンプル・ファイルのディレクトリ（続き）

ツール	サンプル・アプリケーションの場所	説明
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 for 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、2.8.3 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。

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。

2.8.2 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。

このサンプルをアンインストールするには、「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。

サンプルを実行するには、バージョン 3.51.2723.0 以上の MSJET35.dll が必要です。

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。

2.8.3 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 以上を使用）は、Oracle Lite データベースを使用した 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手順では、Oracle Lite と Visual Basic（バージョン 5.0 以上）がすでにインストールされていることが前提となります。

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。

2.8.3.1 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。

2.8.3.2 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:　（1 文字以上を入力します）
   - 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。

2.8.3.3 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 ディレクトリにナビゲートします。**update.mak** を選択し、「オープン」をクリックします。

> **注意：** ファイル **update.mak** 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。

2.8.3.4 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。

2.8.4 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。サンプルを作成するには、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。

2.8.4.1 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について説明します。

2.8.4.1.1 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。

2.8.4.1.2 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。

2.8.4.1.3 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。

2.8.4.1.4 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、結果の列の数を検索します。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、列のスケールなど）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。

2.8.4.1.5 long このサンプルでは、SQL LONG VARCHAR の基本的な読込み / 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、まず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挿入されます。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を移入します。表から SELECT が発行されて行がフェッチされ、表から LONG VARCHAR データが読み込まれます。それぞれの行で、SQL_NO_DATA_FOUND が返されるまで、`SQLGetData` を使用して、データがフレームに読み込まれます。これらのアクションは、ファイル「long.out」に記録されます。

2.9 トレース

Oracle9*i* Lite データベースは、Oracle Forms、SQLJ、Web サーバー、OC4J などの他の製品とともに使用します。ソフトウェアのシステムによって予期しないエラーが通知された場合、エラーの発生箇所と原因を特定する必要があります。エラーの原因は、ユーザーの記述したコード、その他の Oracle のツール製品（Forms、SQLJ、OC4J など）または Oracle9*i* Lite のデータベース・コンポーネントである可能性があります。エラーは、ユーザーのアプリケーションが、JDBC または ODBC ドライバを使用して、直接 Oracle9*i* Lite データベースにアクセスしているような単純な環境でも発生します。原因がユーザーのアプリケーション、JDBC または ODBC ドライバ、あるいはコア・データベースの実行時システムのいずれかにかかわらず、最初は、どのコンポーネントに原因があるかが明確でない場合があります。

オプティマイザがかわりの計画を評価したり、索引統計を収集するために時間がかかる場合、問合せのコンパイルに時間がかかる場合があります。オプティマイザによって選択された実行計画が最適でない場合も、問合せの実行に時間がかかる場合があります。これらの基準に基づいて、トレース機能によって、コンパイル時間および実行計画が提供されます。

この項では、トレース機能の設定方法について説明します。内容は次のとおりです。

- 2.9.1 項「基本機能」
- 2.9.2 項「トレース出力の使用可能化」
- 2.9.3 項「SQL トレース」

2.9.1 基本機能

トレース機能は、**polite.ini** ファイルの構成パラメータで使用可能にできます。トレース機能が使用可能になると、生成された情報は、データベース・プロセスの現行の作業ディレクトリでトレース・ファイル **oldb_trc.txt** にダンプされます。そのファイルが存在する場合は、トレース出力はこのファイルに追加されます。ファイルが存在しない場合は、新しいファイルが自動的に作成されます。データベースがファイルの作成または更新に失敗した場合、トレース機能は実行されません。トレース・ファイルにダンプされる情報は次のとおりです。

1. SQL 文を作成するたびに、そのテキストがトレース・ファイルにダンプされます。テキストは、**Statement Text** というヘッダーで始まります。

2. SQL 文をコンパイルした後、コンパイル時間が **Compilation Time** という行に出力されます。
3. エラーがない場合、実行計画が使用可能になった際に出力されます。実行計画は、WHERE 句を含む文によってのみ生成されます。出力される計画には、副問合せごとの表の実行順序が含まれます。
4. SQL 文にマーカーが含まれる場合、各行にバインド値が出力されます。

2.9.2 トレース出力の使用可能化

トレース出力を使用可能にするには、次の手順を実行します。

polite.ini 構成ファイルに、次の行を含めます。

```
OLITE_SQL_TRACE= yes
```

パラメータ名と値の文字列「yes」では、大 / 小文字は区別されません。たとえば、次の行でもトレース出力は可能です。

```
OLITE_SQL_trace= YeS
```

> **注意：**「yes」以外の値を使用すると、トレース機能は使用禁止になります。パラメータ値は、データベースの起動時に 1 回確認されます。したがって、データベースに接続する前に、この値を設定する必要があります。

2.9.3 SQL トレース

SQL トレースの出力は、データベース・プロセスの現行の作業ディレクトリでトレース・ファイル **oldb_trc.txt** にダンプされます。Windows または Windows NT のデータ・サービス、あるいは Linux プラットフォームの Oracle Lite デーモンでは、現行の作業ディレクトリは、データベース・サービスまたはデーモンの起動時に、**wdir** パラメータで指定されます。トレース機能を実装するには、データベース・プロセスに、現行の作業ディレクトリにトレース・ファイルを作成する権限が必要です。トレース出力は、常にトレース・ファイルに含まれます。トレース・ファイルが存在しない場合は、自動的に作成されます。

SQL トレース機能によってトレース・ファイルにダンプされる情報は次のとおりです。

1. コンパイル後の SQL 文。
2. 最適化を含むコンパイル時間。
3. SQL 文の実行直前に存在するマーカーの値。
4. 『Oracle9*i* Lite SQL リファレンス』の EXPLAIN PLAN 文で説明されている実行計画。

2.9.3.1 SQL 文の出力

各 SQL 文は、先頭に接頭辞 **Statement Text** が付きます。SQL 文自体は書式設定なしで出力されます。SQL 文に含まれる改行文字は、SQL 文の出力にも含まれます。

2.9.3.2 コンパイル時間の出力

SQL 文をコンパイルした後、コンパイル時間が 1 行に出力されます。この行は、**Compilation Time** というタイトルで始まります。

2.9.3.3 バインド値の出力

マーカーの値またはバインド変数は、各行で 1 です。この行は、次の書式で表示されます。

```
Marker [<number>]: <Value>
```

`<number>` はマーカーの番号、`<value>` は実行直前のマーカーの値を示します。

2.9.3.4 実行計画の出力

この出力は、SQL 文 `EXPLAIN PLAN` と同じ書式で出力されます。

3

同期

この章では、Mobile サーバーおよび Mobile Sync クライアント・アプリケーションを使用した Oracle Lite データベースと Oracle データベース間の同期機能について説明します。Mobile Sync クライアント・モジュールのカスタマイズに使用可能なプログラミング・インタフェースの詳細も説明します。内容は次のとおりです。

3.1 概要

Oracle Lite データベースには、Oracle データベースに格納されたデータ・サブセットが含まれます。このサブセットは、Oracle Lite データベースのスナップショットに格納されています。実表と異なり、スナップショットは変更ログ内に変更を記録します。ユーザーは、デバイスが接続されていない状態で、Oracle Lite データベースで変更を行うことができ、それを Oracle データベースと同期できます。

最も一般的な同期方法は、高速リフレッシュです。高速リフレッシュでは、変更はクライアントによりアップロードされ、クライアントの変更はダウンロードされます。同時に、Message Generator and Processor（MGP）と呼ばれるバックグラウンド・プロセスによって、すべてのクライアントによりアップロードされた変更が定期的に収集され、データベース表に適用されます。その後、サブスクリプションと呼ばれる事前定義済パラメータに基づき、次の同期時に各クライアントにダウンロードされる予定の新規データを構成します。

完全リフレッシュという同期方法もあります。完全リフレッシュでは、パブリケーションのすべてのデータがクライアントにダウンロードされます。たとえば、最初の同期セッションでは、クライアントの全データが Oracle データベースからリフレッシュされます。この同期方法は、クライアントがすでに所有している行に関係なく、サブスクリプションに該当するすべての行がクライアント・デバイスに移動されるため、より時間がかかります。

3.1.1 同期の概念

Mobile サーバーと相互に通信するクライアント・デバイスから Mobile Sync クライアント・アプリケーションを呼び出すことにより、オフライン・クライアント内の Oracle Lite データベースと Oracle データベース・インスタンスのデータを同期します。Mobile サーバーは、ユーザー、パブリケーション、パブリケーション・アイテム、サブスクリプションなどの同期オブジェクトを使用して、同期セッション時のクライアント操作を処理します。これにより、Oracle Lite の基礎となる同期のパブリッシュおよびサブスクライブ・モデルが構成されます。

- パブリケーション・アイテムは、一意の名前を持ち、Oracle データベース表、ビューまたはシノニムに対して定義された SQL 問合せを含む Mobile サーバー・オブジェクトです。パブリケーション・アイテム内の問合せは、各ユーザーに対して同期する表、ビューまたはシノニムのデータ・サブセットを判断するために使用する、サブスクリプション・パラメータまたはテンプレート変数と呼ばれるオプション・パラメータを持つ場合があります。
- パブリケーションは、一意の名前を持ち、パブリケーション・アイテムのコンテナとして機能する Mobile サーバー・オブジェクトです。パブリケーションは、0（ゼロ）以上のパブリケーション・アイテムを含み、パブリケーション・アイテムは、0（ゼロ）以上のパブリケーションに含まれます。パブリケーションは、メンバー・パブリケーション・アイテムで使用される、すべてのサブスクリプション・パラメータを記録します。また、パブリケーションは、パブリケーション・アイテムで定義された索引およびプラットフォーム固有の情報（クライアントで作成されるデータベースの種類など）を含みます。

- 各 Oracle9*i* Lite アプリケーションには、アプリケーションで必要なデータを定義する関連アプリケーションが含まれます。
- サブスクリプションは、パブリケーションをユーザーに関連付けます。サブスクリプションは、パブリケーションの各パラメータが値に対して初期化されていない場合は使用できません。ユーザーが Mobile サーバーと同期すると、各サブスクリプションに対して Oracle Lite データベースが作成されます。パブリケーションの各パブリケーション・アイテムは、このデータベースのスナップショットとなります。
- Mobile サーバーは、確立されたサブスクリプションから、各クライアントの新規のデータを準備します。このデータは、クライアントの同期時にダウンロードされます。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。パブリケーションに完全リフレッシュのためのフラグが設定されると、該当するすべてのデータがダウンロードされます。

3.2 同期の例

次の手順では、同期を実行するために必要なコンポーネントおよびプロシージャについて順を追って説明します。Windows システムにクライアントをインストールすることが前提となっています。次の手順を完了することにより、各同期を実行できるようになります。手順 1 および 2 では、管理者への問合せが必要となる場合があります。

msync.exe クライアントをダウンロードして構成し、それを使用して "S11U1" というサンプル・ユーザーに対してローカル Oracle Lite データベースを作成する必要があります。このユーザーは、インストールされたサンプルの一部として存在します。

1. 『Oracle9*i* Lite for Windows NT/2000/XP インストレーションおよび構成ガイド』に説明されている方法で Mobile サーバーのインスタンスをインストールおよび構成します。
2. Mobile サーバー・システムのコマンドラインから、**instdemo.bat**（**instdemo**）を実行して Mobile サーバー・リポジトリにサンプル・アプリケーションを作成します。

Solaris の場合

次のパスを指定して、Mobile サーバー・リポジトリ内にサンプル・アプリケーションを格納します。

<*ORACLE_HOME*>/mobile/server/samples

Windows NT の場合

次のパスを指定して、Mobile サーバー・リポジトリ内にサンプル・アプリケーションを格納します。

<*ORACLE_HOME*>¥mobile¥server¥samples

<*ORACLE_HOME*> を、実際の Oracle ホーム名と置き換えます。

3. ブラウザを使用して、次の URL で Mobile サーバー・インスタンスに接続するための Mobile Sync アプリケーションをインストールします。

   http://<*MOBILE_SERVER*>/webtogo/setup

   <*MOBILE_SERVER*> は、Mobile サーバー・インスタンスのホスト名です。「Windows 32」のクライアントをインストールするためのリンクをクリック後、Mobile Sync アプリケーションのインストール手順に従います。この段階で、インストール・ディレクトリが要求されます。この手順では、<*ORACLE_HOME*> ディレクトリにインストール（推奨）します。

4. <*ORACLE_HOME*>/Mobile/SDK/Bin を開き、**msync.exe** を実行します。

5. この段階で、Mobile Sync 接続画面が表示されます。

   図 3-1 に、Mobile Sync 接続画面を示します。

図 3-1「mSync」接続画面

6. ユーザーは、次の表の該当フィールドに、適切なパラメータを入力する必要があります。

   表 3-1 に、Mobile Sync のパラメータ、対応するフィールド・エントリおよび説明を示します。

表 3-1 Mobile Sync パラメータ

パラメータ	フィールド・エントリ	説明
ユーザー名	s11u1	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	MANAGER	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	非選択	選択した場合、Mobile Sync モジュールに接続する際、パスワードの変更が要求されます。

表 3-1 Mobile Sync パラメータ（続き）

パラメータ	フィールド・エントリ	説明
パスワードを保存	選択	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・インスタンス・ホスト名	Mobile サーバー IP アドレスまたは URL です。
保護	非選択	選択した場合、**msync.exe** は SSL 経由で接続されます。
強制リフレッシュ	非選択	選択した場合、完全リフレッシュが行われます。
プロキシを使用	必要な場合、選択。	この手順については、システム管理者に連絡する必要がある場合もあります。

7. この情報を保存するには「適用」をクリックします。

8. 同期を開始するには、「同期」をクリックします。

構成、送信、受信、処理などの各同期タスクの完了を示すプログレス・バーが表示されます。プログレス・バーは、各タスクの完了までの所要時間も表示します。同期が正常に実行されると、「同期に成功しました」というメッセージが表示されます。このメッセージが表示されると、クライアント・システムの <*ORACLE_HOME*>/Mobile/oldb40/S11U1 ディレクトリに **orders.odb** が作成されます。データベースは、Mobile SQL などの SQL ビューアを使用して参照できます。ORD_MASTER および ORD_DETAIL という 2 つの表が含まれます。

同期に失敗すると、「Sync Failed」メッセージが表示されます。同期に失敗した原因を特定するために、Mobile サーバー管理者は、Mobile サーバーのログ・ファイルのトレース情報を参照できます。

パブリケーション、パブリケーション・アイテムなどの同期オブジェクトの作成に適した方法は、Mobile Development Kit に含まれるパッケージ・ウィザードというツールを使用する方法です。パッケージ・ウィザードの詳細は、第 4 章「パッケージ・ウィザードの使用方法」を参照してください。

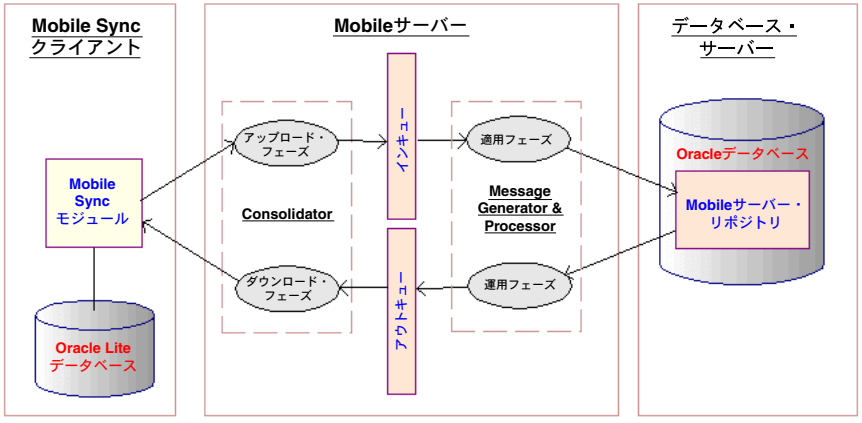
また、同期オブジェクトは、Consolidator API および Resource Manager API を使用してプログラムによって作成できます。詳細は、第 5 章「Consolidator」の 5.1 項「Consolidator API を使用するパブリッシュ・サブスクライブ・モデル」を参照してください。

3.3 同期プロセス

ここまでの手順で、同期を 1 回以上実行したことになります。次に、同期プロセスの詳細を説明します。Oracle Lite では、Oracle Lite データベース・クライアントと Oracle データベース・サーバー間の同期方法として Mobile サーバーを介した非同期方法が使用されます。これは、いずれのコンポーネントも操作の完了に関して相互に依存していないため、Mobile Sync モジュールを MGP とは独立して操作できることを意味します。

図 3-2 に、高速リフレッシュ・プロセスを示します。

図 3-2 高速リフレッシュ同期



3.3.1 高速リフレッシュ同期

デフォルトの同期方法は、表 3-2 に示すとおり、高速リフレッシュ・モードです。高速リフレッシュは、アップロード・フェーズ中に、変更がインキューにアップロードおよび格納される増分リフレッシュです。次に、アウトキューに格納されている変更がクライアントにダウンロードされ、適用されます。その間、MGP はインキューを定期的に参照し、適用フェーズ中に、インキューで検出されたものを取得してデータベースに適用します。対象クライアント、他のクライアント、および Oracle データベースのサーバー側のアプリケーションから生成された変更は構成され、次回クライアントが同期するまでアウトキューに格納されます。

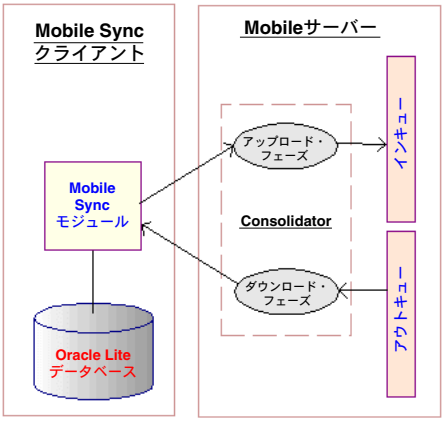
アップロードおよびダウンロード・フェーズは、適用または構成フェーズとは独立して行われます。適用フェーズは、アップロード・フェーズには依存せず、構成フェーズもダウンロード・フェーズには依存しません。いずれの同期セッションにおいても、ダウンロードはアップロード後に発生し、構成は適用後に発生します。

3.3.1.1 クライアントのアップロードおよびダウンロード操作

同期が開始すると、クライアントでは、選択したトランスポート・モードを通して Mobile サーバーへの接続が開かれ、その結果、Mobile サーバーによって Oracle データベース・サーバーへの接続が開かれます。このプロセスを次の図に示します。

図 3-3 に、クライアントのアップロードおよびダウンロード・フェーズを示します。

図 3-3 アップロード / ダウンロード・フェーズ



Oracle Lite データベース・レコードへの変更は蓄積され、行われるデータ操作言語（DML）の種類（挿入、更新または削除など）に対するコードでフラグが設定されます。データは、暗号化および圧縮されて、インキューというオブジェクトを移入するために Mobile サーバーに送信されます。インキューは、同期時に一時的にデータを格納するために作成される永続データベース・オブジェクトです。

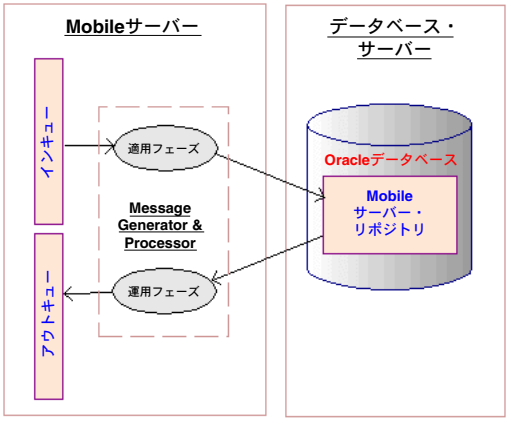
同セッション中、クライアントのスナップショットは、アウトキューから Oracle Lite データベースにデータを適用することで更新されます。アウトキューとインキューの違いは、表ではなく、Oracle データベース実表に含まれるデータへの参照を含むデータ構造にあります。インキューおよびアウトキューを使用した同期プロセスのカスタマイズの詳細は、第 5 章「Consolidator」の 5.4.9 項「レプリケーションをカスタマイズするためのキュー・インタフェース」を参照してください。

3.3.1.2 Mobile サーバーの適用操作

各ユーザーに対して、MGP は、インキューの内容を取得し、Oracle データベースの実表に適用します。この時点で競合が検出され、解決されます。詳細は、第 5 章「Consolidator」の 5.5.3 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。すべてのユーザーによりアップロードされた変更処理が終了すると、適用フェーズは完了です。

図 3-4 に MGP の適用および構成フェーズを示します。

図 3-4 適用 / 構成フェーズ



3.3.1.3 Mobile サーバーの構成操作

適用フェーズ後、MGP は実表を再確認します。MGP は、クライアントにダウンロードするアウトキューへの変更を、構成および格納します。

注意： Message Generator and Processor（MGP）はバックグラウンド・プロセスで、定期的にアクティブとなり、インキューを参照し、Oracle データベース実表に変更を適用します。MGP の構成方法によっては、同期の頻度にかかわらず、クライアントにダウンロードするアウトキューの構成および準備の速度が遅くなる場合があります。変更は、MGP が処理を行うまでインキューに安全に格納され、その後、次の同期プロセスでクライアントにダウンロードされます。

3.3.2 完全リフレッシュ同期

完全リフレッシュ中、クライアントに存在するスナップショットのすべての内容は、Oracle データベース表からリフレッシュされます。すべての内容がリフレッシュされるため MGP には関係しませんが、この同期方法は、時間がかかり、システム・リソースを集中的に使用します。

3.3.3 暗号化されたデータベースの同期

ENCRYPDB というユーティリティを使用して、Oracle Lite データベースを暗号化できます。暗号化されたデータベースの同期には、Oracle Lite による暗号化されたデータベースの管理方法についての理解が必要です。詳細は、付録 C.4 項「ENCRYPDB」を参照してください。

3.4 Mobile Sync アプリケーション・プログラミング・インタフェース（API）

開発者にとって、基盤となる **ocapi.dll** ライブラリを使用して同期を起動するネイティブ・アプリケーションの作成方法は 3 種類あります。これらの API は、Mobile Sync クライアント **msync.exe** によって提供されるものとは異なるアプローチが必要なアプリケーション開発のために提供されています。

- 3.4.1 項「Java インタフェース」
- 3.4.2 項「COM インタフェース」
- 3.4.3 項「C/C++ インタフェース」

注意： 現在 Sun SPARC Solaris 向けのクライアント側同期プログラミング・インタフェースは存在しません。これらのインタフェースでのプログラミングには、Windows オペレーティング・システムの使用をお薦めします。

3.4.1 Java インタフェース

Mobile Sync クライアント側同期のための Java インタフェースを使用することにより、Java で記述されたプログラムで、OCAPI ライブラリによって提供された機能を使用することができます。Java インタフェースは、**oracle.lite.msync** パッケージに含まれています。

Java インタフェースは、次の機能を提供します。

1. ユーザーは、ユーザー名、パスワード、サーバーなどのデータを含むクライアント側ユーザー・プロファイルを設定できます。
2. 同期プロセスを開始します。
3. 同期プロセスの進行状況を追跡できます。

Java インタフェースには、**mSync.jar** および **msync_java.dll** という 2 種類のファイルが存在します。Java インタフェースを使用するには、クラス・パスに **msync.jar** が含まれている必要があります。**mSync.jar** ファイルは、<*ORACLE_HOME*>¥Mobile¥classes ディレクトリにあります。**msync_java.dll** ファイルは、<*ORACLE_HOME*>¥Mobile¥sdk¥bin ディレクトリにあります。

Java インタフェースは、4 つの部分に分かれています。

- 3.4.1.1 項「Sync クラス」
- 3.4.1.2 項「SyncException クラス」
- 3.4.1.3 項「SyncOption クラス」
- 3.4.1.6 項「SyncProgressListener インタフェース」

3.4.1.1 Sync クラス

このクラスは、提供されている同期オプションを使用して同期を開始します。コンストラクタのパラメータを表 3-2 に示します。

コンストラクタ

`Sync(SyncOption option)`

表 3-2 に、Sync クラス・コンストラクタ・パラメータおよびその説明を示します。

表 3-2 Sync クラス・コンストラクタ

パラメータ	説明
`option`	`SyncOption` クラスのインスタンスです。同期を行うために必要なすべてのパラメータが含まれます。詳細は、3.4.1.3 項「SyncOption クラス」を参照してください。

パブリック・メソッド

表 3-3 に、Sync クラス・パブリック・メソッド・パラメータおよびその説明を示します。

表 3-3 Sync クラス・パブリック・メソッド

パラメータ	説明
`SyncProgressListener add(ProgressListener listener)`	同期プロセスの進行状況を監視するために同期オブジェクトに、プログレス・リスナーを追加します。
`void doSync()`	同期セッションを開始し、同期が完了するまでそのスレッドをブロックします。
`void abort()`	同期セッションを中止します。

リスナーは、ProgressListener インタフェースを実装するオブジェクトです。同期オブジェクトは、このオブジェクトの progress() 関数をコールして、同期の進行状況を関数に通知します。

例

次のコードは、デフォルト設定を使用したセッションの開始方法を示します。

```
try
{
  Sync mySync = new Sync( new SyncOption());
  mySync.doSync();
}
catch ( SyncException e)
{
  System.err.println( "Sync Error:"+e.getMessage());
}
```

3.4.1.2 SyncException クラス

このクラスは、同期プロセス中のリカバリ不能なエラーの信号を送ります。SyncException() はクリアなオブジェクトを構成します。コンストラクタのパラメータを次の表に示します。

コンストラクタ

```
SyncException()
SyncException( int errorCode, String errorMessage)
```

表 3-4 に、SyncException コンストラクタ・パラメータおよびその説明を示します。

表 3-4 SyncException コンストラクタ・パラメータ

パラメータ	説明
errorCode	エラーです。使用可能なコードの詳細は、『Oracle9*i* Lite メッセージ・リファレンス』を参照してください。
errorMessage	追加の情報を提供する、読込み可能なテキストです。

パブリック・メソッド

SyncException のメソッドを次の表に示します。

表 3-5 に、SyncException クラス・パブリック・メソッド・パラメータおよびその説明を示します。

表 3-5 SyncException クラス・パブリック・メソッド

パラメータ	説明
`int getErrorCode()`	エラー・コードを取得します。
`String getErrorMessage()`	エラー・メッセージを取得します。

3.4.1.3 SyncOption クラス

SyncOption クラスは、同期プロセスのパラメータを定義するために使用します。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。

コンストラクタ

```
SyncOption()
SyncOption
   ( String user,
     String password,
     String syncParam,
     String transportDriver,
     String transportParam)
```

SyncOption コンストラクタのパラメータを次の表に示します。

表 3-6 に、SyncOption コンストラクタ・パラメータおよびその説明を示します。

表 3-6 SyncOption コンストラクタ・パラメータ

パラメータ	説明
`user`	Mobile サーバーが認証のために使用する名前を含む文字列です。
`password`	ユーザーのパスワードを含む文字列です。
`syncParam`	同期セッションのパラメータのオプション・リストを定義する文字列です。詳細は、3.4.1.4 項「Java インタフェースの SyncParam 設定」を参照してください。
`transportDriver`	トランスポート・ドライバ名を含む文字列です。現在 HTTP のみサポートしています。

表 3-6 SyncOption コンストラクタ・パラメータ（続き）

パラメータ	説明
transportParam	指定されたドライバによる操作に必要な、すべてのパラメータを含む文字列です。詳細は、3.4.1.5 項「Java インタフェースの TransportParam パラメータ」を参照してください。

パブリック・メソッド

次のパブリック・メソッドは、ユーザー・プロファイルをロードおよび保存します。メソッド getPublication は、Web-to-Go のアプリケーション名を使用して、パブリケーション名を戻します。

表 3-7 に、SyncOption パブリック・メソッド・パラメータおよびその説明を示します。

表 3-7 SyncOption のパブリック・メソッド

パラメータ	説明
void load(String username)	このパラメータは、指定したユーザー名のプロファイルをロードします。ユーザー名が NULL の場合、同期の最後のユーザーに対してプロファイルがロードされます。
void save()	このパラメータは、アクティブ・ユーザーのプロファイルの設定を保存します。
void setUser(String username)	このパラメータは、現在のユーザーを取得および設定するために使用します。
String getuser()	
void setPassword(String password)	このパラメータは、パスワードを取得および設定するために使用します。
String getPassword()	
void setSyncParam(String syncParam)	このパラメータは、同期パラメータを取得および設定するために使用します。
String getSyncParam()	
void setTransportDriver(String driverName)	このパラメータは、ドライバ名を取得および設定するために使用します。リリース 5.0.2. は、HTTP ドライバをサポートしています。
String getTransportDriver()	

表 3-7 SyncOption のパブリック・メソッド（続き）

パラメータ	説明
`void setTransportParam(String transportParam)`	このパラメータは、トランスポート・パラメータを取得および設定するために使用します。
`String getTransportParam()`	
`String getPublication(String app_name)`	`String app_name` パラメータは、Web-to-Go のアプリケーション名です。`String` パラメータは、パブリケーション名です。

例

次のコード例は、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。

```
SyncOption opt = new SyncOption
("sam","lion","pushonly","HTTP","server=server1;proxy=www-proxy.us.oracle.com;proxyP
ort=80");
opt.save();
```

3.4.1.4 Java インタフェースの SyncParam 設定

`syncParam` は、`SyncOption` オブジェクトの作成時に渡される文字列です。同期セッションに、補助パラメータを指定できます。文字列は、名前および値のペアで構成されています。

たとえば、次のようになります。

```
"name=value;name2=value2;name3=value3, ...;"
```

大 / 小文字は、名前では区別されませんが、値では区別されます。次の表に、使用可能なフィールド名を示します。

表 3-8 に、Java インタフェースの SyncParam 設定の名前、値および説明を示します。

表 3-8 Java インタフェースの SyncParam 設定

名前	値 / オプション	説明
`"reset"`	なし	残りの設定を適用する前に、環境内のすべてのエントリを消去します。
`"security"`	`"SSL"` `"CAST5"`	SSL または CAST5 ストリーム暗号化から、該当する暗号化を選択します。

表 3-8 Java インタフェースの SyncParam 設定（続き）

名前	値 / オプション	説明
"pushonly"	なし	この設定は、クライアントからサーバーに変更をアップロードする場合のみに使用します。ダウンロードは行いません。データの転送が、クライアントからサーバーへの一方向の場合に有効です。
"noapps"	なし	新規または更新されたアプリケーションをダウンロードしません。遅い接続またはネットワーク上で同期する場合に有効です。
"syncDirection"	"SendOnly'	"SendOnly" は、"pushonly" と同様です。
	"ReceiveOnly"	"ReceiveOnly" では、サーバーに変更を送信できません。
"noNewPubs"	なし	この設定では、最後に行った同期は送信されないため、新規のパブリケーションを作成できず、現行のパブリケーションからのデータのみ同期します。
"tableFlag [Publication.Item]"	"enable" "disable"	enable 設定では、[Publication.Item] を同期できます。disable 設定は、同期を抑止します。
"fullrefresh"	なし	完全リフレッシュを強制実行します。
"clientDBMode"	"EMBEDDED" "CLIENT"	EMBEDDED に設定すると、データベースへのアクセスは従来の ODBC によって行われます。CLIENT に設定すると、複数のクライアントの ODBC によって行われます。

例 1

最初の例では、SSL セキュリティを有効にし、現行の同期セッションのアプリケーション・デプロイメントを無効にします。

```
"security=SSL; noapps;"
```

例 2

2 つ目の例では、前回のすべての設定をリセットし、Dept 表のアップロードのみをアクティブにします。

```
"reset;pushOnly;tableFlag[TestApp.Emp]=disable;tableFlag[TestApp.Dept]=enable;"
```

3.4.1.5 Java インタフェースの TransportParam パラメータ

TransportParam 文字列の形式は、名前および値のペアとなっている文字列を使用して特定のパラメータを設定する場合に使用します。

たとえば、次のようになります。

```
"name=value;name2=value2;name3=value3, ...;"
```

大 / 小文字は、名前では区別されませんが、値では区別されます。次の表に、使用可能なフィールド名を示します。

表 3-9 に、TransportParam パラメータの名前、値および説明を示します。

表 3-9 TransportParam パラメータ

名前	値	説明
"reset"	なし	残りの設定を適用する前に、環境内のすべてのエントリを消去します。
"server"	サーバー・ホスト名	Mobile サーバーのホスト名または IP アドレスです。
"proxy	プロキシ・サーバー・ホスト名	プロキシ・サーバーのホスト名または IP アドレスです。
"proxyPort"	ポート番号	プロキシ・サーバーのポート番号です。
"cookie"	Cookie 文字列	トランスポートに使用する Cookie です。

例

この例では、Mobile Sync エンジンは、ポート 8080 の proxy.oracle.com というプロキシ・サーバーを介して、test.oracle.com のサーバーを使用するように指示されます。

```
"server=test.oracle.com;proxy=proxy.oracle.com;proxyPort=8080;"
```

3.4.1.6 SyncProgressListener インタフェース

SynchProgressListener は、同期中に進行状況の更新をトラップするインタフェースです。

このクラスは、提供されている同期オプションを使用して同期を開始します。このメソッドのパラメータを次の表に示します。

表 3-10 に SyncProgressListener 抽象メソッド・パラメータおよびその説明を示します。

メソッド

```
void progress
   (int progressType,
    int completed);
```

表 3-10 SyncProgressListener 抽象メソッド

パラメータ	説明
progressType	表 3-11 に示す定数の 1 つに設定されます。
completed	特定の progressType の完了の割合（パーセント）です。

同期ステータスをレポートする定数の名前を次の表に示します。

表 3-11 に、SyncProgressListener インタフェースの定数名および進行状況タイプを示します。

表 3-11 SyncProgressListener インタフェースの定数

定数名	進行状況タイプ
PT_INT	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 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。ストリーミングの実装の場合、時間はより短くなります。
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 は現在送信されたバイト数を示します。
PT_RECV	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

次の単純なクラスが、SyncProgressListener を実装します。

```
class myProgressTracker implements SyncProgress Listener;
{
  public void progress
     (int progressType,
     int completed)
    {
```

```
        System.out.println( "Status: "+progressType+"="+ completed+"%" );
      } //progress
}
```

3.4.2 COM インタフェース

COM インタフェースは、同期プロセスを開始し、様々な設定を可能にするアプリケーションのプログラミングに使用します。このインタフェースは、拡張可能なモジュール形式です。ラッパー形式のインタフェースで、**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 Visual Basic で作成できるように設計されていますが、他に COM インタフェースでサポートされているプログラミング手法（VBScript など）も使用可能です。

3.4.2.1 機能およびコンポーネント

COM インタフェースは、次の機能をサポートします。

- ユーザーが同期プロセスを開始できるようにします。
- 同期プロセスの進行状況を追跡できます。
- 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を可能にします。
- 表レベルの同期オプションの割当てが可能です。
- トランスポートの選択ができます。

COM インタフェース API とサンプルは、*<ORACLE_HOME>*¥Mobile¥SDK¥Examples¥msyncCom サブディレクトリにインストールされています。**mSync_com.dll** ライブラリには次のクラスが含まれています。

- 3.4.2.2 項「ISync インタフェース」
- 3.4.2.3 項「ISyncOption インタフェース」
- 3.4.2.6 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場合は、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。

3.4.2.2 ISync インタフェース

ユーザーは、ISync インタフェース MSync.ISync を使用して、同期プロセスを開始できます。ISync インタフェースの形式を、次の表に示します。

表 3-12 に、ISync インタフェースの抽象メソッド名およびその説明を示します。

表 3-12 ISync インタフェースの抽象メソッド

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。
void abort()	現行の同期を中止します。これは、プログレス・リスナー・コールバックからコールできます。
HRESULT setOption(ISyncOption* syncObj)	次の同期で使用するために、ポインタを SyncOption に設定します。doSync() の前にこの関数がコールされない場合は、最後に保存したオプションが使用されます。

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。

3.4.2.3 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。手動で構成できます。また、ユーザー・プロファイルからロードまたは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。ISyncOption クラスのパブリック・メソッドを次の表に示します。

表 3-13 に、ISyncOption のパブリック・メソッド名およびその説明を示します。

表 3-13 ISyncOption のパブリック・メソッド

名前	説明
`void load()`	最後の同期ユーザー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。
`void getPublication (BSTR app_name, BSTR& pub_name)`	Web-to-Go のアプリケーション名を使用して、パブリケーション名を戻します。

ISyncOption クラスのパブリック・プロパティを次の表に示します。

表 3-14 に、ISyncOption のパブリック・プロパティ名およびその説明を示します。

表 3-14 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。
`syncParam`	同期の設定。詳細は、3.4.2.4 項「COM インタフェースの SyncParam 設定」を参照してください。
`transportType`	使用するトランスポートのタイプ。現在「HTTP」タイプのみサポートしています。
`transportParam`	トランスポートのパラメータ。詳細は、3.4.1.5 項「Java インタフェースの TransportParam パラメータ」を参照してください。
`BSTR app_name(in)`	Web-to-Go のアプリケーション名。
`BSTR& pub_name (out)`	パブリケーション名。

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
```

3.4.2.4 COM インタフェースの SyncParam 設定

syncParam は、同期セッションに補助パラメータを指定するための文字列です。文字列は、名前および値のペアで構成されています。

たとえば、次のようになります。

```
"name=value;name2=value2;name3=value3, ...;"
```

大 / 小文字は、名前では区別されませんが、値では区別されます。次の表に、使用可能なフィールド名を示します。

表 3-15 COM インタフェースの SyncParam 設定

名前	値 / オプション	説明
"reset"	なし	残りの設定を適用する前に、環境内のすべてのエントリを消去します。
"security"	"SSL" "CAST5"	SSL または CAST5 ストリーム暗号化から、該当する暗号化を選択します。
"pushonly"	なし	この設定では、ダウンロードはできないため、クライアントからサーバーに変更をアップロードする場合にのみ使用します。データの転送が、クライアントからサーバーへの一方向の場合に有効です。
"noapps"	なし	新規または更新されたアプリケーションをダウンロードしません。遅い接続またはネットワーク上で同期する場合に有効です。
"syncDirection"	"SendOnly' "ReceiveOnly"	"SendOnly" は、"pushonly" と同様です。 "ReceiveOnly" では、サーバーに変更を送信できません。
"noNewPubs"	なし	この設定では、最後に行った同期は送信されないため、新規のパブリケーションを作成できず、現行のパブリケーションからのデータのみ同期します。
"tableFlag [Publication.Item]"	"enable" "disable"	enable 設定では、[Publication.Item] を同期できます。disable は、同期を抑止します。

表 3-15 COM インタフェースの SyncParam 設定（続き）

名前	値 / オプション	説明
"fullrefresh"	なし	完全リフレッシュを強制実行します。
"clientDBMode"	"EMBEDDED" "CLIENT"	EMBEDDED に設定すると、データベースへのアクセスは従来の ODBC によって行われます。CLIENT に設定すると、複数のクライアントの ODBC によって行われます。

例 1

最初の例では、SSL セキュリティを有効にし、現行の同期セッションのアプリケーション・デプロイメントを無効にします。

```
"security=SSL; noapps;"
```

例 2

2 つ目の例では、前回のすべての設定をリセットし、Dept 表のアップロードのみをアクティブにします。

```
"reset;pushOnly;tableFlag[TestApp.Emp]=disable;tableFlag[TestApp.Dept]=enable;"
```

3.4.2.5 COM インタフェースの TransportParam パラメータ

TransportParam 文字列の形式は、名前および値のペアとなっている文字列を使用して特定のパラメータを設定する場合に使用します。

たとえば、次のようになります。

```
"name=value;name2=value2;name3=value3, ...;"
```

大 / 小文字は、名前では区別されませんが、値では区別されます。次の表に、使用可能なフィールド名を示します。

表 3-16 に、COM インタフェースの TransportParam パラメータの名前、値および説明を示します。

表 3-16 COM インタフェースの TransportParam パラメータ

名前	値	説明
"reset"	なし	残りの設定を適用する前に、環境内のすべてのエントリを消去します。
"server"	サーバー・ホスト名	Mobile サーバーのホスト名または IP アドレスです。
"proxy	プロキシ・サーバー・ホスト名	プロキシ・サーバーのホスト名または IP アドレスです。

表 3-16 COM インタフェースの TransportParam パラメータ（続き）

名前	値	説明
"proxyPort"	ポート番号	プロキシ・サーバーのポート番号です。
"cookie"	Cookie 文字列	トランスポートに使用する Cookie です。

例

次の例では、Mobile Sync エンジンは、ポート 8080 の proxy.oracle.com というプロキシ・サーバーを通して test.oracle.com のサーバーを使用するように指示されます。

```
"server=test.oracle.com;proxy=proxy.oracle.com;proxyPort=8080;"
```

3.4.2.6 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、同期ステータスの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。ISyncProgressListener の抽象メソッドを次の表に示します。

表 3-17 に、ISyncProgressListener の抽象メソッド名およびその説明を示します。

表 3-17 ISyncProgressListener の抽象メソッド

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。詳細は、3.4.1.6 項を参照してください。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。

ISynchProgressListener は、同期中に進行状況の更新をトラップするインタフェースです。同期進行状況をレポートする定数の名前を次の表に示します。

表 3-18 に、IsyncProgressListener の定数および進行状況タイプの説明を示します。

表 3-18 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。
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
```

3.4.3 C/C++ インタフェース

C/C++ インタフェースは、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、<*ORACLE_HOME*>¥Mobile¥bin ディレクトリに格納されている **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・アプリケーションから開始および監視でき、Mobile Sync アプリケーション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。

C++ のプログラム例、MAKE ファイルおよび依存ファイルは、<ORACLE_HOME>¥Mobile¥SDK¥Examples¥msync¥src ディレクトリにあります。このインタフェースの使用方法については、SimpleSync.cpp 内のソース・コードを調べてください。実行可能ファイル「SimpleSync.exe」は、<ORACLE_HOME>¥Mobile¥SDK¥Examples¥msync¥bin ディレクトリにあります。

3.4.3.1 ocSessionInit

同期環境を初期化するために使用します。

構文

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

`ocSessionInit` 関数のパラメータを次の表に示します。

表 3-19 に、ocSessionInit のパラメータおよびその説明を示します。

表 3-19 ocSessionInit のパラメータ

名前	説明
`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、これをコール元で割り当てる必要があります。`ocSessionInit()` コールの後、コール元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。コール元が、`ocEnv` 構造体のメモリーを割り当てる必要があります。

3.4.3.2 ocSessionTerm

同期環境を消去し、クリーンアップします。

構文

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

ocSessionTerm 関数のパラメータを次の表に示します。

表 3-20 に、ocSessionTerm のパラメータおよびその説明を示します。

表 3-20 ocSessionTerm のパラメータ

名前	説明
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。ユーザーは、これらがペアでコールされることを確認する必要があります。

3.4.3.3 ocSaveUserInfo

ユーザー設定を **conscli.odb** データベース・ファイルに保存します。

構文

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

ocSaveUserInfo 関数のパラメータを次の表に示します。

表 3-21 に、ocSaveUserInfo のパラメータおよびその説明を示します。

表 3-21 ocSaveUserInfo のパラメータ

名前	説明
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。

1. Username
2. Password
3. SavePassword

4. NewPassword
5. Priority
6. Secure
7. PushOnly
8. SyncApps
9. SyncNewPublication

フィールドの使用方法の詳細は、3.4.3.7 項「C/C++ データ構造」を参照してください。

3.4.3.4 ocDoSynchronize

同期プロセスを開始します。

構文

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

ocDoSynchronize 関数のパラメータを次の表に示します。

表 3-22 に、ocDoSynchronize のパラメータ名およびその説明を示します。

表 3-22 ocDoSynchronize のパラメータ

名前	説明
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 OC_SENDRECEIVE（デフォルト）の場合は、ラウンドトリップ同期がアクティブになります。syncDirection が OC_SENDONLY 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 OC_RECEIVEONLY 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。

戻り値 0 は、関数が正常に実行されたことを示します。その他の場合、値はエラー・コードです。

3.4.3.5 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。

このオプションは、ocDoSynchronize の前に使用する必要があります。

ocSetTableSyncFlag のデフォルトの sync_flag 設定は、すべての表で TRUE（1）です。デフォルトでは、すべての表にフラグが設定され、同期されます。特定の表を選択して同期する場合は、最初にすべての表を同期するデフォルト設定を無効にする必要があります。その後、同期したい特定の表の選択同期を有効にします。

構文

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

ocSetTableSyncFlag 関数のパラメータを次の表に示します。

表 3-23 に、ocSetTableSyncFlag のパラメータ名およびその説明を示します。

表 3-23 ocSetTableSyncFlag のパラメータ

名前	説明
env	同期環境へのポインタ。
publication_name	同期が行われるパブリケーションの名前です。publication_name の値が NULL の場合は、すべてのパブリケーションがデータベースにあることを表します。文字列は Consolidator の CreatePublication メソッドの client_name_template パラメータと同じです。多くの場合、このパラメータには NULL を使用します。詳細は、第 5 章「Consolidator」の 5.2.4 項「パブリケーションの作成」を参照してください。
table_name	スナップショットの名前です。CreatePublicationItem() の 3 番目のパラメータである store の名前と同じです。詳細は、第 5 章「Consolidator」の 5.2.5 項「パブリケーション・アイテムの作成」を参照してください。
sync_flag	sync_flag が 1 に設定されている場合は、パブリケーションを同期する必要があります。sync_flag が 0 に設定されている場合は、同期しません。sync_flag の値は、永続的には格納されません。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。

特定の表のみを同期する場合、次の手順を実行します。

1. すべての表が 1（TRUE）に設定されているデフォルト設定を無効にします。

   例：

   ```
   ocSetTableSyncFlag(&env, <publication_name>,null,0)
   ```

   <publication_name> は、パブリケーションの実際の名前に置き換える必要があります。また、null 値は、例外なくパブリケーションのすべての表を指定します。

2. 特定の表の選択同期を有効にします。

   例：

   ```
   ocSetTableSyncFlag(&env, <publication_name>,<table_name>,1)
   ```

3.4.3.6 ocGetPublication

この関数は、Web-to-Go のアプリケーション名からクライアントのパブリケーション名を取得します。アプリケーションをパブリッシュする前に、パッケージ・ウィザードを使用してアプリケーションをパッケージする場合、Web-to-Go のユーザーはアプリケーション名のみを認識します。

構文

```
ocError ocGetPublication(ocEnv* env, const char* application_name,
char* buf, int buf_len);
```

ocGetPublication 関数のパラメータを表 3-24 に示します。この表に、ocGetPublication のパラメータ名およびその説明を示します。

表 3-24 ocGetPublication パラメータ

名前	説明
ocEnv* 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。
const char* application_name（in）	アプリケーションの名前です。
char* buf（out）	パブリケーション名が格納されるバッファです。
int buf_len（in）	バッファ長です。32 バイト以上である必要があります。

コメント

戻り値 0 は、関数が正常に実行されたことを示します。その他の値は、エラー・コードです。

この関数を使用すると、Web-to-Go のアプリケーション名からパブリケーション名が取得され、バッファに格納されます。

例

次のコード例は、パブリケーション名の取得方法を示します。

```
void sync()
{
        ocEnv env;
        int rc;

        // Clean up ocenv
        memset(&env 0, sizeof(env) );

        // init OCAPI
        rc = ocSessionInit(&env);

        strcpy(env.username, "john");
        strcpy(env.password, "john");

        // We use transportEnv as HTTP paramters
        ocTrHttp* http_params = (ocTrHttp*)(env.transportEnv.ocTrInfo);
        strcpy(http_params->url, "your_host");

        // Do not sync webtogo applicaton "Sample3"
        char  buf[32];
        rc = ocGetPublication(&env, "Sample3", buf, sizeof(buf));
        rc = ocSetTableSyncFlag(&env, buf, NULL, 0);

        // call sync
        rc = ocDoSynchronize(&env);
        if (rc < 0)
               fprintf(stderr, "ocDoSynchronize failed with %d:%d¥n",
                   rc, env.exError);
        else
               printf("Sync compeleted¥n");

        // close OCAPI session
        rc = ocSessionTerm(&env);
        return 0;
}
```

3.4.3.7 C/C++ データ構造

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

ocEnv

ocEnv は、内部メモリー・バッファと状態情報を保持するためにすべての Mobile Sync モジュール関数によって使用されるデータ構造です。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。構造体のパラメータを次の表に示します。

表 3-25 に、ocEnv 構造体のフィールド・パラメータのフィールド名、型、使用方法および説明を示します。

表 3-25 ocEnv 構造体のフィールド・パラメータ

フィールド	型	使用方法	説明
Username	char[MAX_USERNAME]	コール元は、ocSessionInit をコールする前に、これらのフィールドを設定する必要があります。	認証するユーザーの名前。
Password	char[MAX_USERNAME]	コール元は、ocSessionInit をコールする前に、これらのフィールドを設定する必要があります。	ユーザー・パスワード（クリア・テキスト）。
NewPassword	char[MAX_USERNAME]	コール元は、ocSessionInit をコールした後、これらのフィールドをオプションで設定できます。	文字列の最初の文字が NULL ではなく、（char）0 の場合、この文字列は、ユーザーのパスワードの変更を要求するためにサーバーに送られます。変更されたパスワードは、次の同期セッションで有効になります。
SavePassword	Short	コール元は、ocSessionInit をコールした後、これらのフィールドをオプションで設定できます。	1 に設定すると、password フィールドのパスワードはローカルに保存され、次回の ocSessionInit コール時にロードされます。
AppRoot	char[MAX_USERNAME]	コール元は、ocSessionInit をコールした後、これらのフィールドをオプションで設定できます。	アプリケーションのコピー先のディレクトリ。最初の文字が NULL の場合、デフォルトのディレクトリが使用されます。

表 3-25 ocEnv 構造体のフィールド・パラメータ（続き）

フィールド	型	使用方法	説明
Priority	Short	コール元は、ocSessionInit をコールした後、これらのフィールドをオプションで設定できます。	予約済。
Secure	Short	コール元は、ocSessionInit をコールした後、これらのフィールドをオプションで設定できます。	0 に設定すると、トランスポートでセキュリティが実行されません。OC_DATA_ENCRYPTION に設定すると、CAST5 同期を使用します。OC_SSL_ENCRYPTION に設定すると、SSL 同期を使用します（Win32 のみ）。
SyncDirection	Enum	コール元は、ocSessionInit をコールした後、これらのフィールドをオプションで設定できます。	0（OC_SENDRECEIVE）に設定すると、同期は双方向になります（デフォルト）。OC_SENDONLY に設定すると、変更はサーバーのみにプッシュされます。これは、ローカルでの変更を収集して送信した後、同期を停止するために行われます。エンジンによって（フロッピー・ベースなどの）異なる段階を分離する必要がある同期の場合に有効です。OC_RECEIVEONLY に設定すると、変更は送信されず、サーバーからの更新のみが受信されます。受信のみが行われ、ローカル・データベースに変更が反映されます。
TrType	Enum	ocSessionInit のコールの前に設定する必要があります。	0（OC_BUILDIN_HTTP）に設定すると、内蔵 HTTP トランスポート・ドライバを使用します。OC_USER_METHOD に設定すると、ユーザー提供のトランスポート・ファンクションを使用します。
ExError	ocError	OCAPI によって更新された読込み専用情報。	拡張エラー・コード - OS または OKAPI エラー・コード。
transportEnv	ocTransportEnv		トランスポート・バッファ。3.4.3.7.1 項「ocTransportEnv」を参照してください。
ProgressProc	FnProgress	コール元は、ocSessionInit をコールした後、これらのフィールドをオプションで設定できます。	NULL ではない場合、プログレス・リスナーのコールバックを指します。

表 3-25 ocEnv 構造体のフィールド・パラメータ（続き）

フィールド	型	使用方法	説明
TotalSendDataLen	Long	OCAPI によって更新された読込み専用情報。	OCAPI によって設定される、トランスポートの合計送信バイト数です。最初の `fnSend()` がコールされる前に設定されます。
TotalRecieveDataLen	Long	OCAPI によって更新された読込み専用情報。	OCAPI によって設定される、トランスポートの合計受信バイト数です。最初の `fnReceive()` コール時に設定されます。
UserContext	Void*	コール元は、`ocSessionInit` をコールした後、これらのフィールドをオプションで設定できます。	コール元によって（処理する進行状況ダイアログ、レンダラ・オブジェクト・ポインタなどの）コンテキスト情報として、どれに対しても設定できます。
OcContext	Void*		予約済。
Logged	Short		予約済。
BufferSize	Long		予約済（ワイヤレス /NetTech のみ）。
PushOnly	Short	コール元は、`ocSessionInit` をコールした後、これらのフィールドをオプションで設定できます。	1 に設定すると、変更はサーバーのみにプッシュされます。
SyncApps	Short	コール元は、`ocSessionInit` をコールした後、これらのフィールドをオプションで設定できます。	1 に設定すると（デフォルト）、アプリケーションのデプロイメントが実行されます。 0 に設定すると、サーバーからはいずれのアプリケーションも受信されません。
SyncNewPublications	Short	コール元は、`ocSessionInit` をコールした後、これらのフィールドをオプションで設定できます。	1 に設定すると（デフォルト）、最後の同期以降サーバーから作成された新規のパブリケーションを受信します。 0 に設定すると、既存のパブリケーションのみ同期します（ワイヤレスなどの遅い転送に有効です）。
ClientDbMode	Enum	コール元は、`ocSessionInit` をコールした後、これらのフィールドをオプションで設定できます。	`OC_DBMODE_EMBEDDED` に設定すると（デフォルト）、ローカルの Oracle Lite ODBC ドライバが使用されます。 `OC_DBMODE_CLIENT` に設定すると、Branch Office ドライバが使用されます。

表 3-25 ocEnv 構造体のフィールド・パラメータ（続き）

フィールド	型	使用方法	説明
SyncTimeLog	Short	コール元は、ocSessionInit をコールした後、これらのフィールドをオプションで設定できます。	1 に設定すると、ログ同期開始時間が conscli.odb ファイルに記録されます。
UpdateLog	Short	コール元は、ocSessionInit をコールした後、これらのフィールドをオプションで設定できます。	デバッグのみ。 1 に設定すると、サーバー側の挿入および更新された行の情報がパブリケーションの odb に記録されます。
Options	Short	コール元は、ocSessionInit をコールした後、これらのフィールドをオプションで設定できます。	デバッグのみ。次のフラグのビット・セット。 ■ OCAPI_OPT_SENDMETADATA メタ情報をサーバーへ送信します。 ■ OCAPI_OPT_DEBUG デバッグ・メッセージを使用可能にします。 ■ OCAPI_OPT_DEBUG_F デバッグのために送信および受信されたすべてのバイトを保存します。 ■ OCAPI_OPT_NOCOMP 圧縮を使用禁止にします。 ■ OCAPI_OPT_ABORT 設定すると、OCAPI によって現在の同期セッションの強制終了が試みられます。 ■ OCAPI_OPT_FULLREFRESH OCAPI に対して既存のすべてのデータをパージし、完全リフレッシュを行うように強制します。

環境構造には、Mobile Sync モジュール関数の動作方法を変更するためにコール元によって更新が可能なフィールドも含まれています。

```
typedef struct ocEnv_s {
    // User infos
    char    username[MAX_USERNAME];  // consolidator client id
    char    password[MAX_USERNAME];  // consolidator client password for
                                     // authentication during synchronization
    char    newPassword[MAX_USERNAME]; // resetting consolidator client password
                                     // on server side if this field is not blank
    short   savePassword;            // if set to 1, save 'password' as preference data on client device
    char    appRoot[MAX_PATHNAME];   // directory path on client device for deploying files
    short   priority;                // reserved
    short   secure;                  // if set to 1, data encrypted over the wire
    enum {
        OC_SENDRECEIVE = 0,          // full step of synchronize
        OC_SENDONLY,                 // send phase only
        OC_RECEIVEONLY,              // receive phase only

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

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

    ocError exError;                 // extra error code

    ocTransportEnv  transportEnv;    // transport control information

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

    // Values used for Progress Bar. If 0, progress bar won't show.
    long        totalSendDataLen;    // set by OCAPI informing transport total number
                                     // of bytes to send; set before the first fnSend() is called
    long        totalReceiveDataLen; // to be set by transport informing OCAPI
                                     // total number of bytes to receive;
                                     //should be set at first fnReceive() call.

    void*   userContext;             // user defined context

    void*   ocContext;               // internal use only
    short   logged;                  // internal use only

    long    bufferSize;              // send/receive buffer size, default is 0
    short   pushOnly;                // Push only flag, default is 0
```

```
    short   syncApps;               // Application deployment flag, default is 1
    short   syncNewPublication;     // Sync New publication flag, dafault is 1

    enum {                          // WIN32 only,
        OC_DBMODE_EMBEDDED = 0,     // Client database is accessed by conventional ODBC
        OC_DBMODE_CLIENT            // Client database is accessed by multi client ODBC
    } clientDBMode;                 // Default is OC_DBMODE_EMBEDDED.
    short   syncTimeLog;            // If 1, Log sync start time and end time to conscli.odb.
    short   updateLog;           // If 1, Log serverside insert/update row info to user odb. Dafault is 0.
    //short sendMetaData;           // If 1, Client send Table Meta data. Default is 0.
    unsigned short  options;        // refer OCPI_OPT
    short   cancel;                 // Cacel button is on or off
} ocEnv;
```

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

3.4.4 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・アプリケーションは、C/C++ インタフェース API の ocSetTableSyncFlag 関数を使用して、同期が必要なパブリケーションおよびパブリケーション・アイテムを判断します。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。

ユーザーは、Oracle Lite データベースで定義された整合性制約が、選択同期によって違反されていないことを確認する必要があります。

4

パッケージ・ウィザードの使用方法

この章では、Mobile Development Kit のパッケージ・ウィザードツール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、次の作業に使用可能なグラフィカル・ツールです。

1. 新規モバイル・アプリケーションの作成
2. 既存のモバイルアプリケーションの編集
3. Mobile サーバーへのアプリケーションのパブリッシュ

新規モバイル・アプリケーションの作成時には、そのコンポーネントとファイルを定義します。既存のモバイル・アプリケーションのコンポーネントの定義を編集する場合もあります。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。また、パッケージ・ウィザードを使用すると、Oracle データベースで実表を作成するために実行可能な SQL スクリプトを作成できます。

4.2 パッケージ・ウィザードの起動

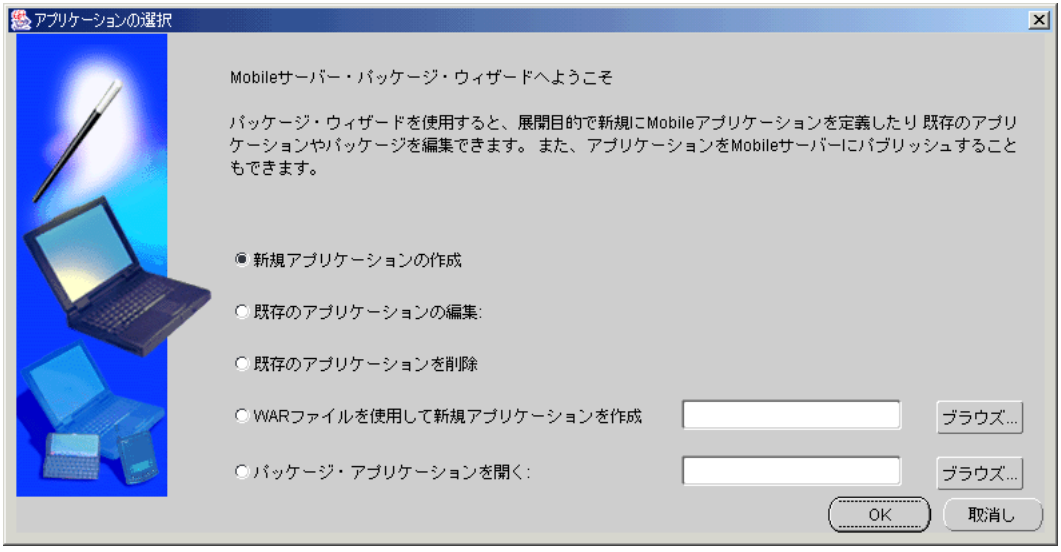
パッケージ・ウィザードを起動するには、DOS プロンプト・ウィンドウで次のように入力します。

wtgpack

パッケージ・ウィザードが表示され、「アプリケーションの選択」パネルが表示されます。「アプリケーションの選択」パネルでは、次の図に示す機能を使用して、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 に、パッケージ・ウィザードの「アプリケーションの選択」パネルを示します。

図 4-1「アプリケーションの選択」パネル



「アプリケーションの選択」パネルで使用可能なオプションを次の表に示します。

表 4-1 に、「アプリケーションの選択」パネルのタスク・オプションを示します。

表 4-1「アプリケーションの選択」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。アプリケーションが以前にパブリッシュされている場合は、アプリケーションを Mobile サーバー・リポジトリから削除しません。Mobile サーバー・コントロール・センターを使用して、Mobile サーバー・リポジトリからアプリケーションを削除できます。
WAR ファイルを使用して新規アプリケーションを作成	WAR ファイルを使用する Web ベースのアプリケーションは、Web-to-Go に対してのみパッケージ化されます。WAR ファイルのサポートの詳細は、『Oracle9*i* Lite for Web-to-Go 開発者ガイド』を参照してください。

表 4-1「アプリケーションの選択」パネルのオプション（続き）

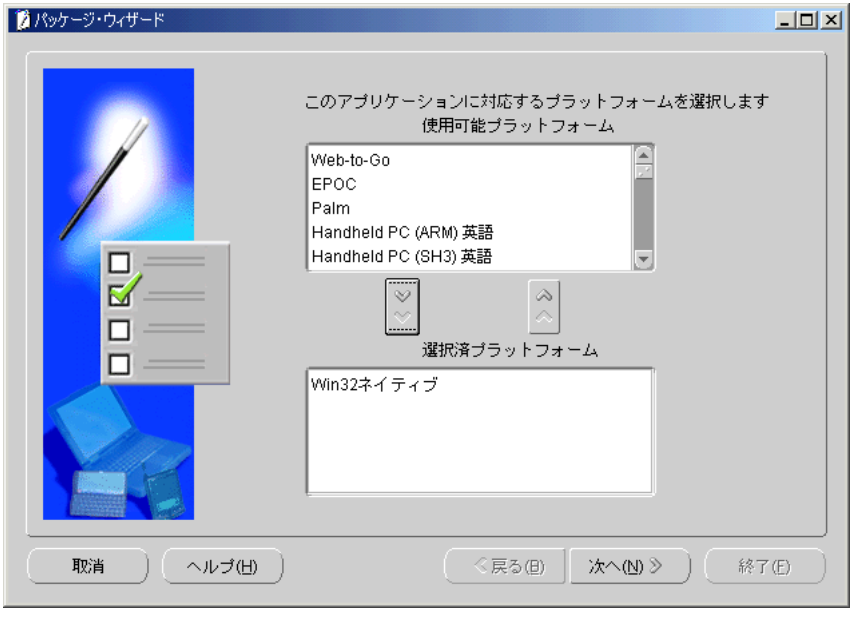
機能	説明
パッケージ・アプリケーションを開く	このオプションを選択すると、**.jar** ファイルとしてパッケージ化されているアプリケーションを選択できます。隣にあるフィールドにパッケージ・アプリケーションの名前を入力するか、「ブラウズ」ボタンを使用して編集対象のアプリケーションを検索します。

4.3 プラットフォームの選択

「プラットフォーム」ウィンドウを使用して、アプリケーションをパッケージ化するプラットフォームを選択できます。1 つ以上のパッケージ・プラットフォームを選択する必要があります。アプリケーションが 2 つ以上のクライアント・タイプで実行される場合、他のパッケージ・プラットフォームも選択できます。次の図に示すとおり、「使用可能プラットフォーム」リストからプラットフォームを選択し、右にある下矢印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 に、「プラットフォーム」パネルを示します。

図 4-2「プラットフォーム」パネル



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、モバイル・アプリケーションに名前を付けて、そのモバイル・アプリケーションの Mobile サーバー上の格納場所を指定するために使用します。次の図に示すとおり、パネルにはフィールドが含まれます。

図 4-3 に、「アプリケーション」パネルを示します。

図 4-3「アプリケーション」パネル

「アプリケーション」パネルのオプションを次の表に示します。

表 4-2 に、「アプリケーション」パネルのフィールド名およびその説明を示します。

表 4-2「アプリケーション」パネルのオプション

フィールド	説明	必須
仮想パス	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。仮想パスの定義にはスラッシュ（/）を使用します。 例 : /MySamples/Sample3	○
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。わかりやすい名前を選択します。アプリケーション名の指定には、空白を使用できます。	○
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。このディレクトリで必要な形式の詳細は、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

プラットフォームの検索には、ローカル・アプリケーション・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格納します。

- 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格納します。
- 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格納します。
- 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格納します。
- 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格納します。

特定のデバイス用のディレクトリに格納できないファイルは、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・レベルにも格納できます。

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パブリッシュし、Mobile サーバー・リポジトリで管理できます。これは、各実装が 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ため、同じアプリケーションに複数の実装が存在することを意味します。たとえば、Windows 32 と Windows CE 両用の C/C++ アプリケーションを使用することもできます。C/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 Windows CE 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。それぞれが同一のデータベース表を使用するため、同一アプリケーションに 2 つのバージョンが存在する場合があります。ユーザーは、アプリケーション・ファイルが専用サブディレクトリに格納され、個別の名前を持っていることを確認する必要があります。ローカル・アプリケーション・ディレクトリには、異なるアプリケーション・バージョンが格納されます。パッケージ・ウィザードは、このアプリケーションのルート・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。次の例に、ローカル・アプリケーション・ディレクトリのパスのリストおよび説明を示します。

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

Windows CE（ARM プロセッサの PPC）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 －「ローカル・アプリケーションのディレクトリ」フィールドにこの文字列を入力する必要があります。固有プラットフォームで使用されるサブディレクトリ以外のサブディレクトリをこのフィールド内に追加すると、パッケージ・プロセスが正常に実行されません。

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、Windows 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リストについては、必要なプラットフォームに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、ローカル・マシン上の場所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。パネルはプラットフォーム固有です。表示されるタブは、それぞれのプラットフォームによって異なります。Web-to-Go アプリケーションのパネルには、次の図に示すとおり、すべてのタブが含まれます。

図 4-4 に、「ファイル」パネルを示します。

図 4-4「ファイル」パネル

「ファイル」パネルのオプションを次の表に示します。

表 4-3 に、「ファイル」パネルのフィールド名およびその説明を示します。

表 4-3「ファイル」パネルのオプション

フィールド	説明	必須
「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」オプションを選択します。

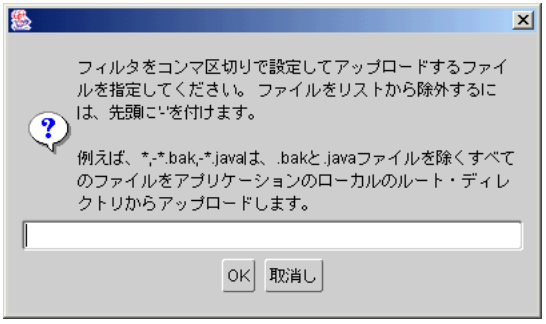
4.5.2 フィルタ

「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。図 4-5 に示すように、「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）フィルタのリストの作成に使用します。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 に、「フィルタ」パネルを示します。

図 4-5「フィルタ」パネル

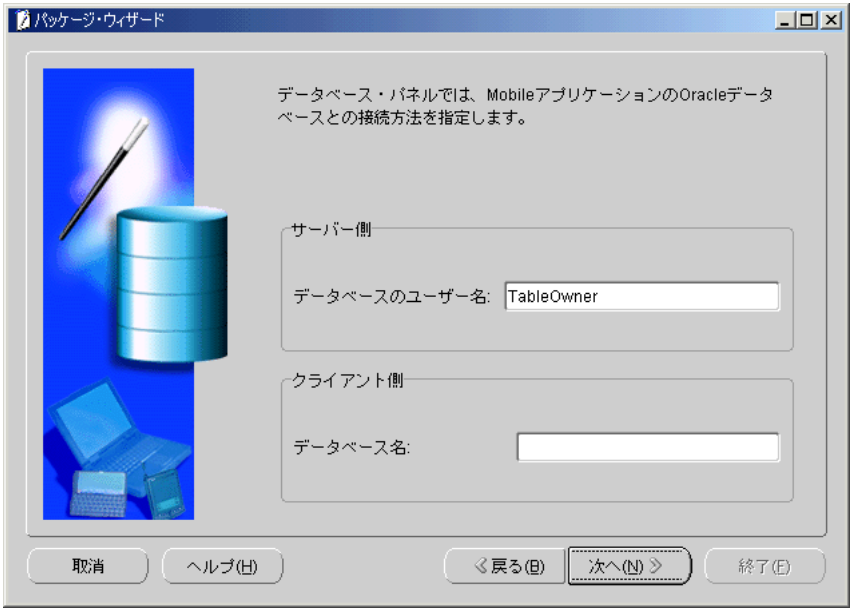


4.6 データベース情報の入力

次の図に示すとおり、「データベース」パネルを使用して、Oracle サーバーでデータ同期が必要なデータベースを指定します。

図 4-6 に、「データベース」パネルを示します。

図 4-6「データベース」パネル



「データベース」パネルのフィールドを次の表に示します。

表 4-4 に、「データベース」パネルのフィールド名およびその説明を示します。

表 4-4「データベース」パネルのオプション

フィールド	説明	必須
データベースのユーザー名	データの同期をとるためにアプリケーションによって使用される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。パッケージ・ウィザードは、スナップショット定義からパブリケーション・アイテムを作成します。パブリケーション・アイテムの名前を生成し、スナップショットに指定された名前をパブリケーション・アイテムのストア名として使用します。ストア名は、パブリケーション・アイテムが Oracle9*i* Lite でインスタンス化された場合、スナップショット名として使用されます。

また、パッケージ・ウィザードは、各プラットフォームのパブリケーションを作成し、そのパブリケーションの名前も生成します。このため、パッケージ・ウィザードではパブリケーションおよびパブリケーション・アイテムは表示されません。

図 4-7 に、「スナップショット」パネルを示します。

図 4-7「スナップショット」パネル

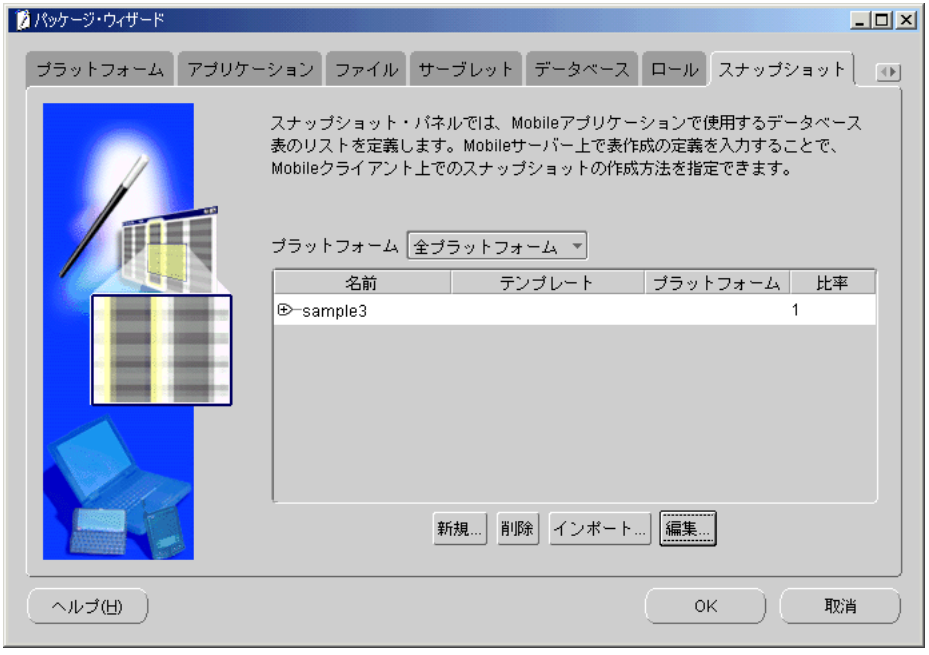


表 4-5 に、「スナップショット」パネルのフィールド名およびその説明を示します。

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 ■ 全プラットフォーム 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、選択した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表示されます。	×
名前	モバイル・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で、Oracle Lite データベース上で作成されたスナップショット名である必要があります。	○
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、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、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。

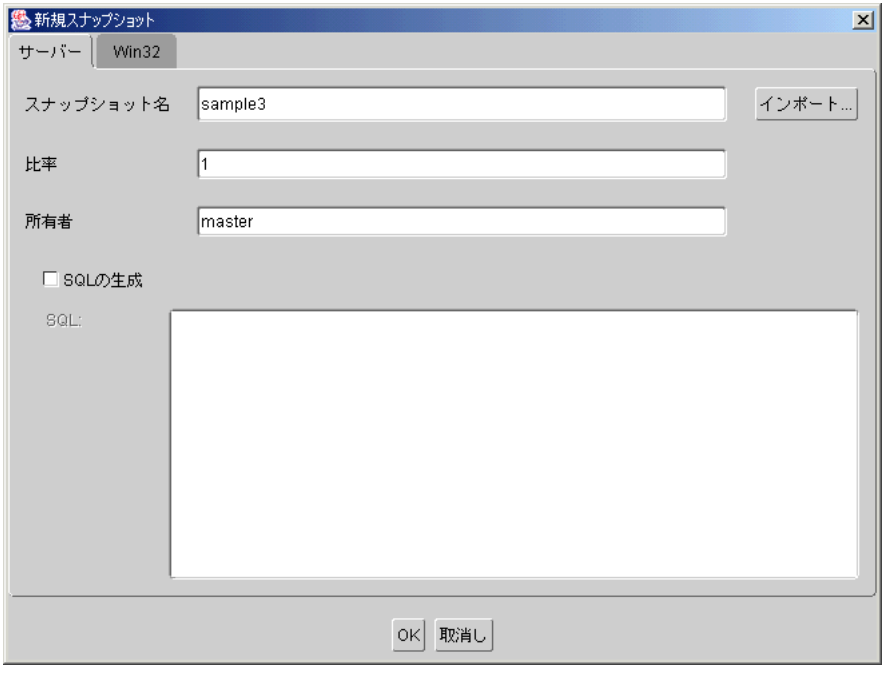
注意： 「スナップショット」パネルから複数のスナップショットをインポートできますが、「新規表」ダイアログ・ボックスから新規表を作成するときにインポートできるスナップショットは 1 つのみです。

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」画面は、次の図のとおり表示されます。

図 4-8 に、「新規スナップショット」パネルを示します。

図 4-8「新規スナップショット」ウィンドウ



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

表 4-6 に、「新規スナップショット」パネルのフィールド名およびその説明を示します。

表 4-6「新規スナップショット」パネルのオプション

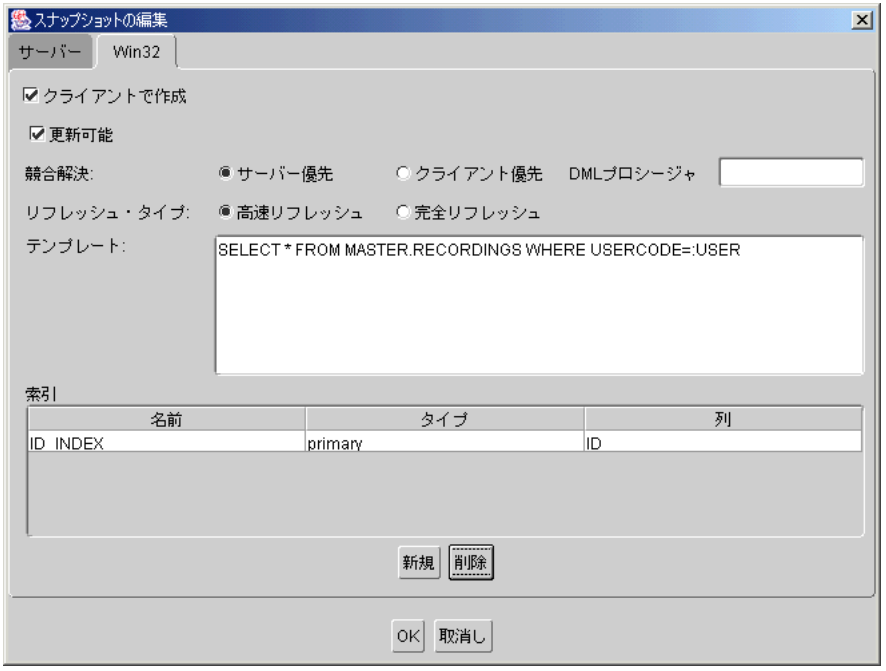
機能	説明
プラットフォーム	図 4-2 に示すとおり、タブには、「プラットフォーム」画面での選択に基づくプラットフォームが表示されます。
スナップショット名	スナップショット定義の基になるデータベース・サーバー表の名前です。
比率	この表に対する表の比率を設定できます。表の比率は、同期時の競合を解決するために使用されます。詳細は、第 5 章「Consolidator」の 5.2.8.3 項「表の比率の使用」を参照してください。
所有者	実表が属するスキーマの名前です。
SQL の生成	この機能を選択した場合、パッケージ・ウィザードによって、SQL スクリプトへの出力情報が収集されます。この SQL スクリプトを使用すると、Mobile サーバーと関連するデータベースに、データベース表を作成できます。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。
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。

4.7.1.1 スナップショットの索引の作成

パッケージ・ウィザードを使用してスナップショットの索引を作成するには、次の手順を実行します。

1. 「スナップショット」パネル（図 4-7 を参照）で「編集」ボタンを選択して、既存のスナップショットから索引を作成します。スナップショットおよび索引を新規作成する場合は、「新規」ボタンを選択します。

図 4-9 索引の作成



2. 図 4-9 に示すとおり、パネルから「プラットフォーム」タブを選択します（例 : Win 32）。「テンプレート」フィールドに、スナップショットを定義する SQL 文が表示されます。新規の索引を作成するには、表の下にある「新規」ボタンを選択します。
3. 「索引」表には、次の 3 列があります。
   - 名前 - 索引の名前です。
   - タイプ - 「通常」、「プライマリ」、「一意」のいずれかのタイプを選択します。表示されるリストから該当するオプションを選択します。
   - 列 - 索引で使用する列名を入力します。

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。すでに接続が確立されているが、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、再度 **wtgpack.exe** を実行します。

図 4-10 に、「データベースへの接続」パネルを示します。

図 4-10「データベースへの接続」パネル

図 4-10 に示すとおり、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 を入力する場合は、次の書式を使用します。

```
jdbc:oracle:thin:@<databasehostname>:<port>:<SID>
```

たとえば、jdbc:oracle:thin:@<HOSTNAME>:1521:orcl と指定します。Oracle Lite の場合は、jdbc:polite:webtogo を使用します。

図 4-11 に、「表」パネルを示します。

図 4-11「表」パネル

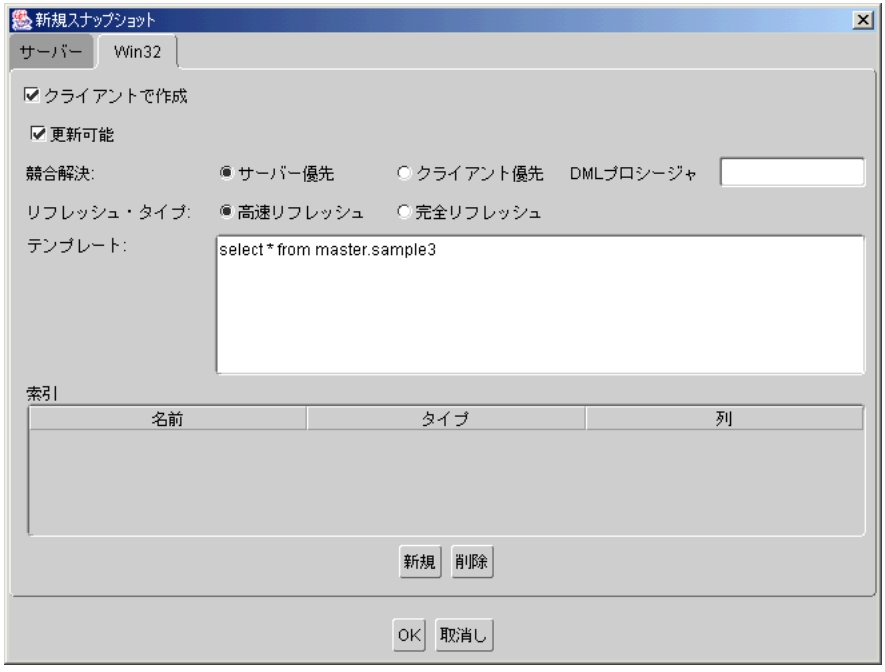
図 4-11 に示すとおり、表のインポート元のスキーマを選択した後、表を選択します。「追加」をクリックしてから「閉じる」をクリックします。パッケージ・ウィザードの「スナップショット」パネルに表が表示されます。

4.7.3 スナップショットの編集

図 4-12 に示すとおり、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 4-12 に、「スナップショットの編集」パネルを示します。

図 4-12「スナップショットの編集」パネル



スナップショット定義を編集するには、次の表に示す「スナップショットの編集」パネルのパラメータを変更します。

表 4-7 に、「スナップショットの編集」パネルのフィールド名およびその説明を示します。

表 4-7「スナップショットの編集」パネルのオプション

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。

表 4-7「スナップショットの編集」パネルのオプション（続き）

機能	説明
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、第 5 章「Consolidator」の 5.5 項「同期エラーと競合」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、第 5 章「Consolidator」の 5.4.13 項「DML 操作のコールバックのカスタマイズ」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。テンプレート変数は、指定された表からデータのサブセット化を行い、クライアント用に構成されたデータを特殊化するために使用します。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。

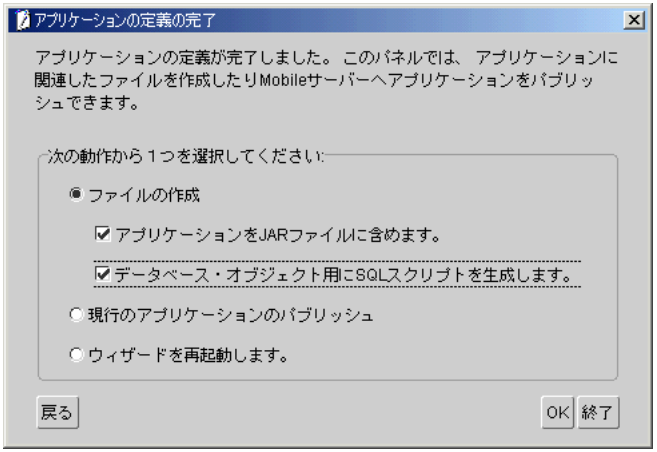
4.8 アプリケーションの完了

次の図に示すとおり、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の完了」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 4-13 に、「アプリケーションの定義の完了」ウィンドウを示します。

図 4-13「アプリケーションの定義の完了」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は、コントロール・センターを使用して Mobile サーバー・リポジトリにアプリケーションを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、ローカル・アプリケーションのルート・ディレクトリの下の SQL サブディレクトリ内に格納されます。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

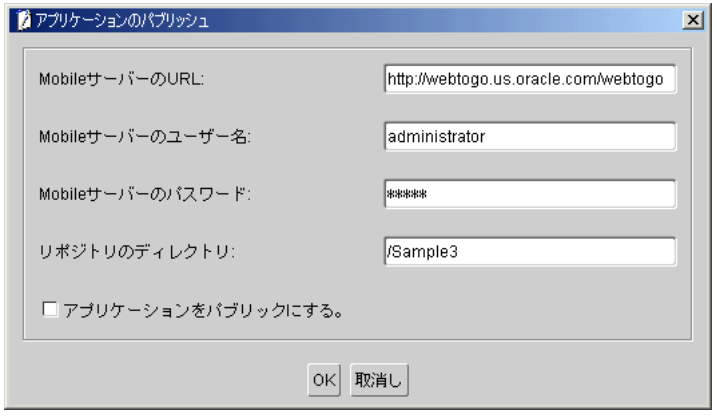
「ウィザードを再起動します。」オプションを使用すると、パッケージ・ウィザードを再起動できます。このオプションを使用すると、パッケージ・ウィザードの「アプリケーションの選択」パネルに戻ります。パッケージ・ウィザードを再起動するには、「ウィザードを再起動します。」をクリックしてから「OK」をクリックします。

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。次の図に示すとおり、「アプリケーションのパブリッシュ」ウィンドウが表示されます。

図 4-14 に、「アプリケーションのパブリッシュ」ウィンドウを示します。

図 4-14「アプリケーションのパブリッシュ」ウィンドウ



次の表に示すとおり、「アプリケーションのパブリッシュ」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 に、「アプリケーションのパブリッシュ」ウィンドウのフィールド名およびその説明を示します。

表 4-8「アプリケーションのパブリッシュ」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。

4.8.6 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、「既存のアプリケーションを編集」オプションを選択して、アプリケーションを編集できます。

5
Consolidator

この章では、Consolidator のパブリッシュ・サブスクライブ・モデルについて説明し、Consolidator および Resource Manager API を使用してアプリケーションをカスタマイズする方法を説明します。次の項では、デフォルトの同期プロセスに対して実行可能な変更について説明します。この章の内容を理解するには、Java プログラミング言語の十分な知識が必要です。内容は次のとおりです。

5.1 Consolidator API を使用するパブリッシュ・サブスクライブ・モデル

Mobile サーバーは、パブリッシュ・サブスクライブ・モデルを使用して、Oracle データベース・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。パブリケーション・アイテム、パブリケーションなどの基本機能は、パッケージ・ウィザードを使用すると簡単に実装できます。これらの機能は、Consolidator API および Resource Manager API を使用しても実行できます。その場合は、Java プログラムを作成して、必要に応じて機能をカスタマイズします。拡張機能をプログラムにより使用可能にするには、Consolidator API および Resource Manager API を使用する必要があります。

パブリッシュ・サブスクライブ・モデルでは、次の表で説明するとおり、Mobile サーバー・オブジェクトを使用します。

表 5-1 に、パブリッシュ・サブスクライブ・モデルの要素項目およびその説明を示します。

表 5-1 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・アイテム	パブリケーション・アイテムは、Oracle Lite データベース内にスナップショットを作成するために必要な情報を追跡するオブジェクトです。パブリケーション・アイテムの主要な属性には、一意の名前、ユーザーがアクセス可能なデータを指定するバインド変数（サブスクリプション・パラメータ）を持つ SQL SELECT 文、Oracle Lite データベース内に作成されるスナップショットの名前などがあります。パブリケーション・アイテムは、クライアント上のスナップショットの構造を定義します。サブスクリプション・パラメータに特定のユーザーまたはグループの値をバインドし、サーバー・データベース上で問合せを実行することで、対応するサーバー表、ビューまたはスナップショットからスナップショットの行が取り出されます。したがって、スナップショットは、サーバー上の表のスナップショット定義とみなすことができます。
パブリケーション	パブリケーションは、クライアント・データベースを作成するために必要な情報を追跡するオブジェクトです。パブリケーションの主要な属性には、パブリケーション名、パブリケーションに属する一連のパブリケーション・アイテム、クライアント・マシン上に作成されるデータベースの名前、パブリケーションの設計対象となるクライアント・プラットフォームなどがあります[1]。パブリケーション・オブジェクトは、メンバー・パブリケーション・アイテムに定義されたすべてのサブスクリプション・パラメータを収集し、これらのパラメータにサブスクリプションの作成を可能にする値を設定する 1 つの方法を提示します。

表 5-1 パブリッシュ・サブスクライブ・モデルの要素（続き）

項目	説明
スナップショット	スナップショットは、Oracle Lite 表で、この表は Oracle データベースのマテリアライズド・ビューです。スナップショットは、通常の表の挿入、削除、更新などのすべての操作をサポートします。また、スナップショット自身に加えられた変更も追跡します。スナップショットの作成、変更ログの取得またはリセットに、SQL 文は使用できません。Mobile Sync API（OCAPI）はスナップショットを作成し、同期プロセスの一環として、スナップショットの変更内容をサーバー・データベースと同期させます。
サブスクリプション	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。パブリケーションのすべてのサブスクリプション・パラメータに値が指定されていないかぎり、サブスクリプションは無効になります。サブスクリプション・パラメータは、クライアントがサブスクライブされるパブリケーション内のすべてのパブリケーション・アイテムに対して設定されます。
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、パブリケーション・アイテムの問合せをサブセット化する場合に使用されるバインド変数です。このパラメータにより、サーバーは、各クライアントに割り当てられた行数に応じてデータのサブセット化を実行できます。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。
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、単一クライアントのデータを同期させることができます。 ■ ユーザーは、単一のユーザー名を使用して、複数のデバイスに格納されたデータを同期させることができます。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。

[1] Oracle9*i* Lite の将来のリリースでは、パブリケーションはプラットフォーム依存ではなくフォーム・ファクタ依存になります。そのため、1 つのパブリケーションにより、プラットフォームに関係なく、同一フォーム・ファクタのすべての PDA に対応できるようになります。

次のいずれかの方法で、パブリッシュ・サブスクライブ・モデルを実装できます。

- 宣言によって、パッケージ・ウィザードを使用して、アプリケーションをパッケージ化およびパブリッシュします。これがお薦めの方法です。この方法の詳細は、第 4 章「パッケージ・ウィザードの使用方法」に説明されています。

- Resource Manager API および Consolidator API を使用して、特定の拡張機能を起動するか、または実装をカスタマイズすることもできます。この方法は、特殊化した機能を必要とする上級者にお薦めします。

5.1.1 Java ストアド・プロシージャおよび SQL スクリプトの配置

Consolidator の同期機能を使用して、モバイル・クライアントに Java ストアド・プロシージャおよび SQL スクリプトを配置できます。『Consolidator Admin API Specification』に記載されている Consolidator API を使用すると、ご使用のパブリケーションに SQL スクリプトおよび Java リソースを追加できます。

パブリケーションに、制約（NULL など）およびオブジェクト（INDEX など）に関する SQL 文を含む SQL スクリプトを追加できます。

同期クライアントによって SQL スクリプトが実行される状況は、次のとおりです。

- SQL スクリプトが初めて配置される場合
- 新しいスクリプトが追加される場合
- 同期されたパブリケーション表に変更を追加する場合

SQL スクリプトの実行中に例外が発生した場合は、例外メッセージが **sync_errors.txt** に記録されます。ただし、同期トランザクションはロールバックされません。

また、Windows 32 クライアントは、初めて同期する場合、新しいリソースが追加される場合、または既存のリソースが変更される場合に、Oracle Lite データベースに Java ストアド・プロシージャをロードします。Java リソース（クラス・ファイルまたは JAR ファイル）をパブリッシュするには、リソースがファイル・システムに格納されている必要があります。

詳細は、『Consolidator Admin API Specification』を参照してください。

5.1.2 パブリッシュ・サブスクライブ・モデルの使用方法

パブリッシュ・サブスクライブ・モデルは、Consolidator API および Resource Manager API を使用してプログラムによりカスタマイズできます。Consolidator を起動して、パブリッシュ・サブスクライブ・モデルを実装するための基本手順は、次のとおりです。

1. データベース表を作成します。
2. Mobile サーバーに接続します。
3. パブリケーションを作成します。
4. パブリケーション・アイテムを作成します。
5. 必要に応じて、パブリケーション・アイテム索引を作成します。
6. ユーザーを作成します。
7. パブリケーションにパブリケーション・アイテムを追加します。

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. 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。
10. サブスクリプションをインスタンス化します。

5.1.3 Consolidator メソッドの機能比較

パッケージ・ウィザードおよびコントロール・センターには、パブリッシュ・サブスクライブ・モデルの使用頻度の高い機能を実行する機能、アプリケーションをパッケージ化してパブリッシュする機能、ユーザーを作成または削除する機能、サブスクリプションを作成または削除する機能などが用意されています。さらに高度な機能は、Consolidator API および Resource Manager API により提供されます。基本機能の比較を、次の表で説明します。

表 5-2 に、Consolidator とパッケージ・ウィザードでの基本機能の比較を示します。

表 5-2 Consolidator の基本機能の比較

機能	パッケージ・ウィザード	コントロール・センター	API
接続のオープン	×	×	○
ユーザーの作成	×	○	○
ユーザーの削除	×	○	○
パブリケーションの作成	○（暗黙的）	×	○
パブリケーション・アイテムの作成	○（暗黙的）（表のみ）	×	○
パブリケーション・アイテム索引の作成	○	×	○
パブリケーションの削除	○（暗黙的）	×	○
パブリケーション・アイテムの削除	○（暗黙的）	×	○
パブリケーション・アイテム索引の削除	○（暗黙的）	×	○
シーケンスの作成	○（Web アプリケーションのみ）	×	○
シーケンス・パーティションの作成	○（Web アプリケーションのみ）	×	○
シーケンスの削除	○（Web アプリケーションのみ）	×	○
シーケンス・パーティションの削除	○（Web アプリケーションのみ）	×	○
パブリケーション・アイテムの追加	○	×	○

表 5-2 Consolidator の基本機能の比較（続き）

機能	パッケージ・ウィザード	コントロール・センター	API
パブリケーション・アイテムの削除	○（暗黙的）	×	○
サブスクリプションの作成	×	○	○
サブスクリプションのインスタンス化の解除	×	×	○
サブスクリプション・パラメータの設定	×	○	○
サブスクリプションの削除	×	○	○
トランザクションのコミット	×	×	○
トランザクションのロールバック	×	×	○
接続のクローズ	×	×	○

Consolidator の拡張機能は、Consolidator API および Resource Manager API により使用できます。これらの拡張機能を、次の表で比較します。

表 5-3 に、Consolidator とパッケージ・ウィザードでの拡張機能の比較を示します。

表 5-3 Consolidator の拡張機能の比較

機能	パッケージ・ウィザード	コントロール・センター	API
仮想主キー列の作成	×	×	○
仮想主キー列の削除	×	×	○
モバイル DML プロシージャの追加	○	×	○
モバイル DML プロシージャの削除	○	×	○
パブリケーション・アイテムのインスタンス化の解除	×	×	○
親のヒント	×	×	○
依存性のヒント	×	×	○
依存性のヒントの削除	×	×	○
パブリケーション・アイテムの問合せキャッシュの有効化	×	×	○
パブリケーション・アイテムの問合せキャッシュの無効化	×	×	○

表 5-3 Consolidator の拡張機能の比較（続き）

機能	パッケージ・ウィザード	コントロール・センター	API
主キーのヒント	×	×	○
トランザクションのパージ	×	×	○
トランザクションの実行	×	×	○
完全リフレッシュ	×	○	○
文の実行	×	×	○
メタデータの生成	×	×	○
キャッシュのリセット	×	×	○
キャッシュの依存性	×	×	○
キャッシュの依存性の削除	×	×	○
現在の時刻の取得	×	×	○
認証	×	○	○
制限選択条件の設定	×	×	○
パブリケーションの変更	×	×	○

5.2 Consolidator を使用した Sample11.java の定義

これらの API を使用して Consolidator を定義する方法を示すために、次の項では Oracle9*i* Lite に付属している **sample11.java** という Java プログラムのサンプルを使用します。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 API に含まれています。

サンプルのエントリは、次の場所にあります。

Solaris の場合

<*ORACLE_HOME*>/mobile/server/samples

Windows NT の場合

<*ORACLE_HOME*>¥mobile¥server¥samples

5.2.1 Sample11.java

sample11.java プログラムのソース・コードを、次に示します。

```
import java.sql.SQLException;
import java.sql.*;

import oracle.lite.sync.Consolidator;

public class sample11
{

    static String CONS_SCHEMA;
static String DEFAULT_PASSWORD;

  public static void main(String argv[]) throws Throwable
  {
////////////////////////////////////////////////////////////////////////////
//SAMPLE11
////////////////////////////////////////////////////////////////////////////
    if(argv.length != 2)
    {
        System.out.println("Syntax: java sample11 <Schema> <Password>");
return;
    }
    CONS_SCHEMA = argv[0] ;
    DEFAULT_PASSWORD = argv[1] ;

    //標準 JDBC を使用した必須表の作成
        DriverManager.registerDriver ((Driver)Class.forName
("oracle.jdbc.driver.OracleDriver").newInstance ());
        Connection c = null;
        Statement s = null;
        try
        {
            c = DriverManager.getConnection ("jdbc:oracle:oci8:@WEBTOGO.WORLD",
"MASTER", "MASTER" );
            s = c.createStatement ();
    s.executeUpdate("create table MASTER.ORD_MASTER("
        + "ID number(9),"
        + "DDATE DATE default TO_DATE('1990-01-01 15:35:40', 'YYYY-MM-DD
HH24:MI:SS'),"
        + "STATUS number(9),"
        + "NAME varchar2(20),"
        + "DESCRIPTION varchar2(20)"
        + ")");

    s.executeUpdate("alter table MASTER.ORD_MASTER add constraint"
```

```
        +" orders_pk primary key(ID)");

    s.execute("GRANT ALL ON MASTER.ORD_MASTER to " + CONS_SCHEMA + " WITH GRANT
OPTION");

    s.executeUpdate("create table MASTER.ORD_DETAIL("
        + "ID number(9),"
        + "KEY number(9),"
        + "DDATE DATE default TO_DATE('1995-01-01 15:35:40', 'YYYY-MM-DD
HH24:MI:SS'),"
        + "DESCRIPTION varchar2(20),"
        + "QTYORDERED number(9),"
        + "QTYSHIPPED number(9),"
        + "QTYRECEIVED number(9),"
        + "COST number(9)"
        + ")");

    s.executeUpdate("alter table MASTER.ORD_DETAIL add constraint"
        +" items_pk primary key(ID, KEY)");

            s.execute("GRANT ALL ON MASTER.ORD_DETAIL to " + CONS_SCHEMA + " WITH
GRANT OPTION");
            c.commit ();
        }
        catch (SQLException ee)
        {
            ee.printStackTrace ();
        }
        finally
        {
            if (s!= null) try {s.close ();}catch (SQLException e1){}
            if (c!= null) try {c.close ();}catch (SQLException e2){}
        }

        //Mobile サーバーへの接続
    oracle.mobile.admin.ResourceManager.openConnection(CONS_SCHEMA, DEFAULT_
PASSWORD);
        //パブリケーションの作成
try {
Consolidator.DropPublication("T_SAMPLE11");
} catch (Throwable e) {
//e.printStackTrace(); ignore error
}
Consolidator.CreatePublication("T_SAMPLE11", Consolidator.OKPI_CREATOR_ID,
"OrdersODB.%s", null);
```

```
      // パブリケーション・アイテムの作成
try {
Consolidator.DropPublicationItem("P_SAMPLE11-M");
} catch (Throwable e) {
//e.printStackTrace(); ignore error
}
        try
        {
    Consolidator.CreatePublicationItem("P_SAMPLE11-M","MASTER","ORD_MASTER", "F",
"SELECT * FROM MASTER.ORD_MASTER", null, null);
        } catch (Throwable e) {
e.printStackTrace();
}

try {
Consolidator.DropPublicationItem("P_SAMPLE11-D");
} catch (Throwable e) {
//e.printStackTrace();
}
        try
        {
Consolidator.CreatePublicationItem("P_SAMPLE11-D","MASTER","ORD_DETAIL", "F",
"SELECT * FROM MASTER.ORD_DETAIL", null, null);


     // パブリケーション・アイテム索引の作成

Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I1", "P_SAMPLE11-M", "I",
"DDATE");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I2", "P_SAMPLE11-M", "I",
"STATUS");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I3", "P_SAMPLE11-M", "I",
"NAME");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11D-I2", "P_SAMPLE11-D", "I",
"KEY");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11D-I3", "P_SAMPLE11-D", "I",
"DESCRIPTION");

     // パブリケーションへのパブリケーション・アイテムの追加

Consolidator.AddPublicationItem("T_SAMPLE11", "P_SAMPLE11-M", null, null, "S", null,
null);
Consolidator.AddPublicationItem("T_SAMPLE11", "P_SAMPLE11-D", null, null, "S", null,
null);
        }
        catch (Throwable e)
        {
```

```
            e.printStackTrace ();
        }

        // ユーザーの作成
try {
oracle.mobile.admin.ResourceManager.Example("S11U1");
} catch (Throwable e) {
//e.printStackTrace(); ignore error
}


oracle.mobile.admin.ResourceManager.Example("S11U1","manager","S11U1","C");

        // サブスクリプションのインスタンス化
Consolidator.Example("T_SAMPLE11", "S11U1");
Consolidator.InstantiateSubscription("T_SAMPLE11", "S11U1");

oracle.mobile.admin.ResourceManager.commitTransaction();
oracle.mobile.admin.ResourceManager.closeConnection();

  }
}
```

5.2.2 標準 JDBC を使用した必須表の作成

このプログラムの最初のセクションは、データベース MASTER との JDBC 接続を取得し、データベース内に実表 ORD_MASTER および ORD_DETAIL を作成します。この部分のプロセスには、SQL も使用できます。第 3 章「同期」の 3.2 項「同期の例」に説明されている手順を完了すると、Mobile サーバー・リポジトリおよびクライアントにこれらの表が作成されます。

5.2.3 Mobile サーバーへの接続

次の式は、Mobile サーバーへの接続を行います。

5.2.3.1 openConnection

例

```
ResourceManager.openConnection(<USER_NAME>, <PASSWORD>);
oracle.mobile.admin.ResourceManager.openConnection
   (CONS_SCHEMA,
    DEFAULT_PASSWORD);
```

この例では、<USER_NAME> は S11U1 で、<PASSWORD> は MANAGER です。

5.2.4 パブリケーションの作成

次の手順では、Consolidator クラスを使用してパブリケーションを作成します。パブリケーションは、基本的にデータベース・スキーマで、パブリケーション・アイテム（基本的には表ですが、厳密にはスナップショット定義）を含みます。Sample11.java により、2 つのパブリケーションが作成されます。最初に、DropPublication コマンドを使用して、作成するパブリケーションが存在していないことを確認します。

パブリケーション名には、空白などの特殊文字が使用できます。

5.2.4.1 CreatePublication

CreatePublication の構文は、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 Throwable
```

例

Sample11.java では、T_SAMPLE11 というパブリケーションを作成しています。

```
Consolidator.CreatePublication("T_SAMPLE11", Consolidator.OKPI_CREATOR_ID,
"OrdersODB.%s", null);
```

CreatePublication のパラメータを次の表に示します。

表 5-4 に、CreatePublication のパラメータおよびその説明を示します。

表 5-4 CreatePublication のパラメータ

パラメータ	定義
name	作成しているパブリケーションの名前です。
client_storage_type	プラットフォーム・タイプを定義する定数。詳細は、表 5-5「クライアントの格納タイプ」を参照してください。

表 5-4 CreatePublication のパラメータ（続き）

パラメータ	定義
client_name_template	クライアント・デバイスでのパブリケーション・アイテム名のテンプレートで、次の選択肢があります。 ■ %s - これはデフォルト設定で、パブリケーション・アイテムをデフォルトのデータベースである **conscli.odb** に格納します。 ■ <DATABASE>.%s - このオプションは、パブリケーション・アイテムを <DATABASE> というデータベースに格納します。このオプションは、ファイル名拡張子をサポートしません。 ■ テンプレートのかわりに、1 つのパブリケーション・アイテムを含むパブリケーションに特定の値を使用することもできます。たとえば、「AddressBook」を Palm OS の Address Book アプリケーションに使用できます。
enforce_ri	これは、将来の拡張用に確保されているパラメータで、常に NULL です。

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。

表 5-5 に、クライアントの格納タイプのパラメータ値およびその説明を示します。

表 5-5 クライアントの格納タイプ

パラメータ値	定義
Consolidator.DFLT_CREATOR_ID	OKAPI（Oracle Kernal API）は、すべてのデバイスでのデフォルトです。OKAPI は、クライアント上の Oracle Lite データベースの書式を定義します。この格納タイプを選択すると、データベースは、OKAPI と互換性のあるすべてのデバイスで稼働します。
Consolidator.OKPI_CREATOR_ID	OKAPI の使用方法を指定します。この格納タイプを選択すると、データベースは、OKAPI と互換性のあるすべてのデバイスで稼働します。
Consolidator.OKAPI_PALM	格納タイプを Palm OS 専用として定義します。
Consolidator.OKAPI_EPOC	格納タイプを EPOC 専用として定義します。
Consolidator.OKAPI_WIN32	格納タイプを Windows 専用として定義します。

表 5-5 クライアントの格納タイプ（続き）

パラメータ値	定義
Consolidator.OKAPI_WINCE	格納タイプを Windows CE 専用として定義します。
Consolidator.PALMDB_EXPENSE	格納タイプを Palm OS の Expense アプリケーション専用として定義します。
Consolidator.PALMDB_DATEBOOK	格納タイプを Palm OS の DateBook（Calendar）アプリケーション専用として定義します。
Consolidator.PALMDB_EMAIL	格納タイプを Palm OS の電子メール・アプリケーション専用として定義します。
Consolidator.PALMDB_ADDRESSBOOK	格納タイプを Palm OS の AddressBook アプリケーション専用として定義します。
Consolidator.PALMDB_TODO	格納タイプを Palm OS の To Do アプリケーション専用として定義します。
Consolidator.PALMDB_MEMO	格納タイプを Palm OS の Memo アプリケーション専用として定義します。

5.2.5 パブリケーション・アイテムの作成

パブリケーションを作成した後、パブリケーション・アイテムを作成する必要があります。パブリケーション・アイテムは、Oracle Lite データベースにダウンロードされる実表のスナップショットを定義します。パブリケーション・アイテムは、作成時にリフレッシュ・モードが指定されるため、高速リフレッシュまたは完全リフレッシュに対応するように事前構成されます。データ要件について細部まで制御できるように、パブリケーション・アイテムの作成時に、特定のユーザーに対してデータ・サブセッティング・パラメータを設定することもできます。

パブリケーション・アイテム名は、26 文字に制限され、パブリケーション・アイテム全体で一意である必要があります。次の例では、P_SAMPLE11-M というパブリケーション・アイテムを作成します。このパブリケーション・アイテムを作成する前に、DropPublicationItem を使用して、名前が重複している既存のすべてのパブリケーション・アイテムをクリーンアップします。

5.2.5.1 CreatePublicationItem

CreatePublicationItem の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
```

CreatePublicationItem のパラメータを次の表に示します。

表 5-6 に、CreatePublicationItem のパラメータの例およびその定義を示します。

表 5-6 CreatePublicationItem のパラメータの例

パラメータ	定義
name	パブリケーション・アイテム名を指定します。
owner	ベース・オブジェクト・スキーマの所有者を指定します。たとえば、MASTER がベース・オブジェクト ORD_MASTER の所有者です。
store	Oracle データベース内の実表名またはビュー名を指定します。定義済スナップショットにもこの名前が割り当てられます。
refresh_mode	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、5.4.5 項「複数表パブリケーション（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。
select_stmt	データベース表内の、指定された列のデータを識別する SQL SELECT 文です。この文には、CustCity = :city などのように、コロン（:）の後にバインド変数（サブスクリプション・パラメータ）を挿入できます。特定のユーザーまたはグループに対してサブスクリプションを作成する場合は、パブリケーションの各サブスクリプション・パラメータに値を割り当てる必要があります。
cbk_owner	コールバック・パッケージの所有者を指定します。詳細は、5.4.12 項「構成および適用の前後に行うコールバックのカスタマイズ」を参照してください。
cbk_name	コールバック・パッケージ名を指定します。詳細は、5.4.12 項「構成および適用の前後に行うコールバックのカスタマイズ」を参照してください。

例

Sample11.java プログラムでは、次のコマンドにより、データベース表 ORD_MASTER および ORD_DETAIL の P_SAMPLE-M および P_SAMPLE-D というスナップショット定義（パブリケーション・アイテム）が作成されます。これらのデータベース表は、前述の手順でリポジトリ内に作成されたものです。

```
Consolidator.CreatePublicationItem("P_SAMPLE11-M","MASTER","ORD_MASTER", "F",
"SELECT * FROM MASTER.ORD_MASTER", null, null);
Consolidator.CreatePublicationItem("P_SAMPLE11-D","MASTER","ORD_DETAIL", "F",
"SELECT * FROM MASTER.ORD_DETAIL", null, null);
```

5.2.5.2 更新可能な複数表ビューに対するパブリケーション・アイテムの定義

パブリケーション・アイテムは、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。親表は、主キーがビューの主キーを兼ねているビュー定義に関連するいずれかの表です。したがって、次のことに注意してください。
  - 親表のすべての主キー列が、ビューで選択されている必要があります。
  - いずれの主キー列にも、NULL を割り当てることはできません。
  - ビュー内の 2 つの行は、主キー列のどの値とも一致しません。

  ビューの主キーを定義するには、`PrimaryKeyHint` メソッドを使用する必要があります。
- ビューが更新可能でない場合、またはビューの更新セマンティクスをカスタマイズする場合は、ビューでのデータ操作言語（Data Manipulation Language: DML）の操作に INSTEAD OF トリガーを定義する必要があります。詳細は、5.4.5 項「複数表パブリケーション（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。
- 結果として生成されるビューのすべての実表に、パブリケーション・アイテムを関連付ける必要があります。

5.2.5.3 データのサブセット化

パブリケーション・アイテムを作成する場合、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、パブリケーション・アイテムの作成時に定義する必要があるため、サブスクリプションの作成時に値を指定します。これらのパラメータは、特定のユーザーにダウンロードされたデータを制御するために、同期中に使用されます。

データ・サブセットの作成例

```
Consolidator.CreatePublicationItem("CORP_DIR1", "DIRECTORY1", "ADDRLRL4P", "F" ,
   "SELECT LastName, FirstName, company, phone1, phone2, phone3, phone4,
    phone5, phone1id, phone2id, phone3id, displayphone, address, city, state,
    zipcode, country, title, custom1, custom2, custom3,note
    FROM directory1.addrlrl4p WHERE company > :COMPANY", null, null);
```

このサンプル文では、CORP_DIR1 というパブリケーションからデータが取り出され、企業に基づきサブセットが作成されます。

注意： SELECT 文内では、データ・サブセットのパラメータ名の前に、コロンを接頭辞として指定する必要があります。たとえば、: COMPANY のように指定します。

5.2.6 パブリケーションに対するユーザー・サブスクリプション・パラメータの定義

パブリケーションでデータ・サブセッティング・パラメータを使用する場合は、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。5.2.5.3 項「データのサブセット化」に説明されている「COMPANY」はパラメータの一例です。

5.2.6.1 SetSubscriptionParameter

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

SetSubscriptionParameter のパラメータを次の表に示します。

表 5-7 に、SetSubscriptionParameter のパラメータの例およびその説明を示します。

表 5-7 SetSubscriptionParameter のパラメータの例

パラメータ	定義
publication	サブセットの導出元となるパブリケーションを定義します。
user_name	サブセット・データの対象となるユーザーを定義します。
param_name	パブリケーションのパブリケーション・アイテムで使用するパラメータ名を定義します。
param_value	渡されるパラメータ値を定義します。この値により、このパラメータを使用した場合にパブリケーション・アイテムの問合せから返されるデータが決定されます。

例

この例では、次のパラメータを使用して、ユーザー DAVIDL をパブリケーション CORP_DIR1 にサブスクライブします。

```
Consolidator.SetSubscriptionParameter("CORP_DIR1", "DAVIDL", "COMPANY", "'DAVECO'");
```

注意： このメソッドは、Consolidator 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使用してください。パッケージ・ウィザードを使用して、テンプレート変数（データ・サブセットのもう 1 つのフォーム）を作成する方法も同様に使用できます。

5.2.7 パブリケーション・アイテム索引の作成

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作成がサポートされています。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 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作成するコールも提供しています。

5.2.7.1 CreatePublicationItemIndex

CreatePublicationItemIndex の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
```

CreatePublicationItemIndex のパラメータを次の表に示します。

表 5-8 に、CreatePublicationItemIndex のパラメータおよびその説明を示します。

表 5-8 CreatePublicationItemIndex のパラメータ

パラメータ	定義
name	作成する索引の名前を定義します。
publication_item	索引のパブリケーション・アイテムを定義します。
pmode	索引モード（I - 標準モード、U - 一意モード、P - 主キー・モード）を定義します。詳細は、5.2.7.2 項「クライアント索引の定義」を参照してください。
columns	索引に含める列の名前を定義します。それぞれの文に複数の列を含める場合は、列のリストをカンマで区切る必要があります（空白は挿入しないでください）。

例 1

sample11.java のサンプル・コードに基づき、このプログラムは次の書式をとります。

```
Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I1", "P_SAMPLE11-M", "I",
"DDATE");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I2", "P_SAMPLE11-M", "I",
"STATUS");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11M-I3", "P_SAMPLE11-M", "I",
"NAME");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11D-I2", "P_SAMPLE11-D", "I",
"KEY");
Consolidator.CreatePublicationItemIndex("P_SAMPLE11D-I3", "P_SAMPLE11-D", "I",
"DESCRIPTION");
```

Sample11.java では、5 つの索引が作成されます。P_SAMPLE-M パブリケーション・アイテムの「DDATE」、「STATUS」および「NAME」列および P_SAMPLE-D パブリケーション・アイテムの「KEY」および「DESCRIPTION」列に対して標準索引が設定されます。1 つの索引に、1 つ以上の列を含めることができます。また、次のように、複数列を持つ索引を定義することもできます。

例 2

```
Consolidator.CreatePublicationItemIndex("P_SAMPLE11D-I1", "P_SAMPLE11-D", "I",
"KEY,DESCRIPTION");
```

5.2.7.2 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・アイテム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・アイテムに「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・アイテムのベース・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、『Consolidator Admin API Specification』を参照してください。

5.2.8 パブリケーションへのパブリケーション・アイテムの追加

パブリケーション・アイテムを作成した後は、これをサブスクリプションで使用されるパブリケーションに関連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・アイテム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、『Consolidator Admin API Specification』の「DropPublicationItem」および「AlterPublicationItem」を参照してください。

5.2.8.1 AddPublicationItem

AddPublicationItem の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・アイテムをパブリケーション T_SAMPLE1 に追加します。AddPublicationItem のパラメータを次の表に示します。

表 5-9 に、AddPublicationItem のパラメータおよびその定義を示します。

表 5-9 AddPublicationItem のパラメータ

パラメータ	定義
publication	新規項目を受け取るパブリケーションを定義します。
item	追加するパブリケーション・アイテムを定義します。
columns	パブリケーション・アイテム列の新しい名前を指定します。列の名前を変更しない場合は、NULL を指定します。パブリケーション・アイテムの問合せ内のすべての列は、次のいずれかの適切な順序で指定する必要があります。 ■ パブリケーション・アイテムの SELECT 文に指定されている順序です。 ■ 「SELECT * FROM ...」を指定した文を使用する場合、列名の順序は、実表またはビューの順序と同一である必要があります。

表 5-9 AddPublicationItem のパラメータ（続き）

パラメータ	定義
disabled_dml	DML を使用禁止するためのオプションを指定します。指定可能な値は、次のとおりです。 ■ Y - 完全に更新可能なパブリケーション・アイテムを定義します。 ■ N - 読込み専用パブリケーション・アイテムを定義します。読込み専用パブリケーション・アイテムは、「IUD」オプションを使用しても定義できます。 ■ I - 個々の挿入操作の伝播を無効にします。 ■ U - 個々の更新操作の伝播を無効にします。 ■ D - 個々の削除操作の伝播を無効にします。 ■ 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。
conflict_rule	競合解決で優先する側を定義します。クライアントが優先する場合は「C」、サーバーが優先する場合は「S」を指定します。詳細は、5.2.8.2 項「競合ルールの定義」を参照してください。
restricting_predicate	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・アイテムを追加するときに、パブリケーション・アイテム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、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、5.2.8.3 項「表の比率の使用」を参照してください。この値には、1 ～ 1023 の整数を使用する必要があります。

例

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

5.2.8.2 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・アイテムを追加する場合、クライアント「C」またはサーバー「S」のいずれかを優先して同期の競合を解決するウィニング・ルールを指定できます。Mobile サーバーの同期競合は、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に関連する同期エラーとみなされます。
- 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、行を挿入した後、自分自身でその行を更新すると、実表にまだ行が挿入されていない状況が発生する可能性があります。この場合、C$ALL_CONFIG の DEF_APPLY パラメータが TRUE に設定されている場合、UPDATE でなく INSERT 操作が実行されます。結果として発生する主キー競合を解決するのは、アプリケーション開発者の責任です。ただし、DEF_APPLY が設定されていない場合、「NO DATA FOUND」例外が発生します（同期エラーの処理については後述の内容を参照してください）。
- NULL の使用違反、外部キー制約違反などのその他のエラーはすべて同期エラーです。
- 同期エラーが自動的に解決されない場合、対応するトランザクションがロールバックされ、トランザクション操作は C$EQ 内の Mobile サーバーのエラー・キューに移動されます。データは CEQ$ に格納されます。Mobile サーバーのデータベース管理者は、これらのトランザクション操作を変更して、エラー・キューからトランザクションを再実行するか、またはパージできます。

5.2.8.3 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・アイテム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、各パブリケーション内で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。
4. この値には、1 ～ 1023 の整数を割り当てる必要があります。

表の比率は、特定のパブリケーション内のパブリケーション・アイテムに適用されます。たとえば、1 つのパブリケーションに、複数の比率 2 のパブリケーション・アイテムを含めることができます。この場合、同じパブリケーション内の、比率の低いすべてのパブリケーション・アイテムに対して、INSERT 操作が実行されます。

5.2.9 ユーザーの作成

Sample11 を使用すると、新しいユーザーを作成する前に、dropUser() を使用してユーザーを削除できます。この操作によって、新しいユーザー ID を作成する前に、すべての不要なユーザー ID を消去できます。詳細は、5.2.10 項「ユーザーの削除」を参照してください。この関数のパラメータでは、大 / 小文字は区別されません。

5.2.9.1 createUser

createUser の構文は次のとおりです。

```
public static boolean createUser
    (String user_name,
     String password,
     String fullName,
     String privilege) throws Throwable;
```

createUser のパラメータを次の表に示します。

表 5-10 に、createUser のパラメータの例およびその説明を示します。

表 5-10 createUser のパラメータの例

パラメータ	定義
user_name	モバイル・ユーザーのユーザー名を定義します。
password	ユーザー名に対応するパスワードを定義します。
fullName	オプションです。ユーザーのフル・ネームを指定します。たとえば、John Smith のように指定します。
privilege	このパラメータは、Mobile サーバーユーザーの権限を定義します。値は、次のいずれかになります。 ■ O - アプリケーションをパブリッシュします。 ■ U - Mobile サーバーに接続します。 ■ A - Mobile サーバーを管理します。 ■ NULL - 権限なしを表します。

次の例では、前述の表にリストされたパラメータでユーザー「S11U1」を作成します。

例

```
oracle.mobile.admin.ResourceManager.createUser("S11U1","manager","John Smith","C")
```

5.2.10 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。この関数のパラメータでは、大 / 小文字は区別されません。

5.2.10.1 dropUser

dropUser の構文は次のとおりです。

次の例では、ユーザー「S11U1」を削除します。

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

dropUser のパラメータを次の表に示します。

表 5-11 に、dropUser のパラメータの例およびその説明を示します。

表 5-11 dropUser のパラメータの例

パラメータ	定義
user_name	モバイル・ユーザーのユーザー名を指定します。

例

```
oracle.mobile.admin.ResourceManager.dropUser("S11U1");
```

5.2.11 パブリケーションへのユーザーのサブスクライブ

CreateSubscription 関数を使用すると、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。

5.2.11.1 CreateSubscription

CreateSubscription の構文は次のとおりです。

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

次の例では、表 5-12 に示すパラメータを使用して、ユーザー S11U1 をパブリケーション T_SAMPLE11 にサブスクライブします。

表 5-12 に、CreateSubscription のパラメータの例およびその定義を示します。

表 5-12 CreateSubscription のパラメータの例

パラメータ	定義
publication	サブスクライブ先となるパブリケーションを指定します。
user_name	パブリケーションにサブスクライブするユーザーを指定します。

例

```
Consolidator.CreateSubscription("T_SAMPLE11", "S11U1");
```

5.2.12 サブスクリプションのインスタンス化

ユーザーをパブリケーションにサブスクライブした後、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。

> **注意：** データのサブセット化に必要なサブスクリプション・パラメータを設定する必要がある場合は、サブスクリプションをインスタンス化する前に、この設定処理を完了する必要があります。詳細は、5.2.5.3 項「データのサブセット化」を参照してください。

5.2.12.1 InstantiateSubscription

InstantiateSubscription の構文は次のとおりです。

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

InstantiateSubscription のパラメータを次の表に示します。

表 5-13 に、InstantiateSubscription のパラメータの例およびその説明を示します。

表 5-13 InstantiateSubscription のパラメータの例

パラメータ	定義
publication	サブスクライブ先となるパブリケーションを指定します。
user_name	パブリケーションにサブスクライブするユーザーを指定します。

次の例では、前述の表に示されている値を使用して、パブリケーションに対するユーザーのサブスクリプションをインスタンス化します。

例

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

5.3 Consolidator のその他の標準機能

5.1 項「Consolidator API を使用するパブリッシュ・サブスクライブ・モデル」で使用している API コールは、パブリケーション、パブリケーション・アイテムおよびサブスクリプションをプログラムにより作成する場合に必要な API コールです。この項の内容は、使用する機会は多くありませんが、重要です。

- 5.3.1 項「クライアント・デバイス・データベースの DDL 操作」
- 5.3.2 項「パスワードの変更」
- 5.3.3 項「順序の作成」
- 5.3.4 項「クライアントのパーティション化の順序」
- 5.3.5 項「リモート・データベース・リンクのサポート」

5.3.1 クライアント・デバイス・データベースの DDL 操作

クライアントが初めて同期をとる場合、Mobile サーバーは、スナップショットの形式で、クライアントにデータベース・オブジェクトを自動的に作成します。デフォルトでは、表の主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・アイテムを使用して、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・アイテムから明示的に削除する必要があります。API 情報の詳細は、『Consolidator Admin API Specification』を参照してください。

5.3.2 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。

5.3.2.1 setPassword

`setPassword` の構文は次のとおりです。

```
public static void setPassword
   (String user_name,
    String newpwd) throws Throwable
```

`setPassword` のパラメータを次の表に示します。

表 5-14 に、setPassword のパラメータの例およびその定義を示します。

表 5-14 setPassword のパラメータの例

パラメータ	定義
user_name	モバイル・ユーザーのユーザー名を指定します。
newpwd	モバイル・ユーザーの新しいパスワードを指定します。

次の例では、ユーザー「MOBILE」のパスワードを変更します。

例

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

5.3.3 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および同期をサポートしています。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。

5.3.3.1 CreateSequence

createSequence の構文は次のとおりです。

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

ここで、name は、順序名を指定する値です。次の例では、CUSTOM1 という名の順序を作成します。

例

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

コメント

シーケンス・サポートは、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・アプリケーションでは異なります。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ず、順序定義が C$ALL_SEQUENCE_PARTITIONS 表に作成されることです。この順序定義の作成は、パッケージ・ウィザードと Consolidator API の両方を使用して、宣言とプログラムによる方法を組み合せて行う必要があります。

順序定義を作成するには、次の手順を実行します。

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 API 関数の `CreateSequence` および `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。

5.3.4 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。シーケンスを使用するには、シーケンスを作成した後で、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。このシーケンスは、本当のシーケンス・オブジェクトではなく、実際は、表内のレコードです。そのため、`curr_val` で示される現在の値が最新値に更新されているかどうかは、開発者が責任を持って確認してください。

5.3.4.1 CreateSequencePartition

`CreateSequencePartition` の構文は次のとおりです。

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

`CreateSequencePartition` のパラメータを次の表に示します。

表 5-15 に、CreateSequencePartition のパラメータの例およびその定義を示します。

表 5-15 CreateSequencePartition のパラメータの例

パラメータ	定義
`name`	パーティション化する順序名を定義します。
`user_name`	順序の割当て先となるユーザーを定義します。
`curr_val`	順序の初期値を定義します。
`incr`	順序の増分値を定義します。

次の例では、CUSTOM1 という名の順序をパーティション化します。

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。

5.3.4.1.1 クライアント上での手動による現在の値の更新

クライアント上でパーティションを作成する場合、開発者は、SQL を使用してパーティションの curr_val を手動で更新して、一意の値が使用されていることを確認する必要があります。たとえば、次のようになります。

```
UPDATE C$ALL_SEQUENCE_PARTITIONS
SET curr_val=curr_val+incr
WHERE name = 'CUSTOM1';
```

このプロセスは、新規に現在の値（curr_val）が必要になるたび、またはこの順序を使用して新しい主キーが生成されるたびにクライアント上で実行する必要があります。

5.3.5 リモート・データベース・リンクのサポート

Mobile サーバー・リポジトリの外部にあるリモート・データベース・インスタンスに存在するデータベース・オブジェクトに対して、パブリケーション・アイテムを定義できます。リモート・オブジェクトのローカル・プライベート・シノニムは、Oracle データベース内に作成する必要があります。Consolidator のロギング・オブジェクトを作成するには、<*ORACLE_HOME*>¥Mobile¥server¥admin¥consolidator_rmt.sql にある次の SQL スクリプトをリモート・スキーマに対して実行します。

次に、CreatePublicationItem API を使用して、シノニムをパブリッシュします。リモート・オブジェクトが、更新可能モードまたは高速リフレッシュ・モード（あるいはその両方）でパブリッシュする必要があるビューの場合は、リモート親表もローカルでパブリッシュする必要があります。ローカルの更新可能なビューまたは高速リフレッシュ可能なビュー（あるいはその両方）に使用されるシノニムと同様のリモート・ビューのシノニムに、親のヒントを提供する必要があります。

リモート・オブジェクトをパブリッシュすることにより発生する不明確な依存性を処理するために、DependencyHint と RemoveDependencyHint という 2 つの API が追加されています。

リモート・リンク・プロシージャを試行する前に、Oracle データベースとのリモート・リンクを確立する必要があります。詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

注意： リモート・データベースから行う同期のパフォーマンスは、ネットワーク・スループットとリモート問合せ処理のパフォーマンスの制約を受けます。このようなネットワークの制約があるため、リモート・データの同期は、データが限定されている単純なビューまたは表で使用することをお薦めします。

5.3.5.1 CreatePublicationItem を使用したリモート・オブジェクトのシノニムのパブリッシュ

次のパラメータとともに使用する CreatePublicationItem API は、新しいスタンドアロンのパブリケーション・アイテムを、リモート・データベース・オブジェクトとして作成します。

構文

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

または

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

CreatePublicationItem を使用するシノニム作成のパラメータを次の表に示します。

表 5-16 に、リモート・データベース・リンク用の CreatePublicationItem のパラメータおよびその説明を示します。

表 5-16 リモート・データベース・リンク用の CreatePublicationItem のパラメータ

パラメータ	説明
rmt_jdbc_url	リモート・データベース・インスタンスの JDBC URL を指定する文字列です。
rmt_jdbc_conn	リモート・インスタンスが存在する Oracle データベースへの接続です。
name	新しいパブリケーション・アイテム名を定義する文字列です。
owner	シノニムの所有者を指定する文字列です。
store	シノニム名を指定する文字列です。 **注意**: リモート・オブジェクトをパブリッシュするには、そのオブジェクトのプライベート・シノニムを作成する必要があります。
refresh_mode	リフレッシュ・モードを指定する文字列です。「F」は高速リフレッシュ、「C」は完全リフレッシュを表します。デフォルトは高速リフレッシュです。
select_stmt	新しいパブリケーションに対して SELECT 文を指定する文字列です。この文はパラメータ化できます。次に示す例では、このパラメータは :CAP と指定され、パラメータ名の前にコロンが付いています。
cbk_owner	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、5.4.12 項「構成および適用の前後に行うコールバックのカスタマイズ」を参照してください。
cbk_name	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、5.4.12 項「構成および適用の前後に行うコールバックのカスタマイズ」を参照してください。

URL 文字列を使用すると、リモート接続が確立され、自動的にクローズします。接続が NULL か、または接続が確立できない場合は、例外がスローされます。リモート接続情報は、リンクされたデータベース上にロギング・オブジェクトを作成したり、メタデータを抽出する場合に使用されます。

例

```
Consolidator.CreatePublicationItem(
     "jdbc:oracle:oci8:@oracle.world",
     "P_SAMPLE1",
     "SAMPLE1",
     "PAYROLL_SYN",
     "F"
     "SELECT * FROM sample1.PAYROLL_SYN"+"WHERE SALARY >:CAP", null, null);
```

注意： SELECT 文内では、データ・サブセットのパラメータ名の前に、コロンを接頭辞として指定する必要があります。たとえば、:CAP のように指定します。

5.3.5.2 依存性のヒントの作成

次の構文により、不明確な依存性についてのヒントが作成されます。

構文

```
public static void DependencyHint
   (String owner,
    Sting store,
    String owner_d,
    String store_d) throws Throwable
```

CreateDependencyHint のパラメータを次の表に示します。

表 5-17 に、CreateDependencyHint のパラメータおよびその説明を示します。

表 5-17 CreateDependencyHint のパラメータ

パラメータ	説明
owner	ビューの所有者を指定する文字列です。
store	ビューの名前を指定する文字列です。
owner_d	実表またはビューの所有者を指定する文字列です。
store_d	実表またはビューの名前を指定する文字列です。

例

```
Given remote view definition
        create payroll_view as
        select p.pid, e.name
        from payroll p, emp e
        where p.emp_id = e.emp_id;

Execute locally
        create synonym v_payroll_syn for payroll_view@<remote_link_address>;
        create synonym t_emp_syn for emp@<remote_link_address>;
```

<remote_link_address> は、Oracle データベースに確立されたリンクです。DependencyHint は、ローカル・シノニム v_payroll_syn がローカル・シノニム t_emp_syn に依存することを示すために使用します。

```
Consolidator.DependencyHint("SAMPLE1","V_PAYROLL_SYN","SAMPLE1","T_EMP_SYN");
```

5.3.5.3 依存性のヒントの削除

次の構文により、不明確な依存性についてのヒントが削除されます。

構文

```
public static void RemoveDependencyHint
   (String owner,
    Sting store,
    String owner_d,
    String store_d) throws Throwable
```

RemoveDependencyHint のパラメータを次の表に示します。

表 5-18 に、RemoveDependencyHint のパラメータおよびその説明を示します。

表 5-18 RemoveDependencyHint のパラメータ

パラメータ	説明
owner	ビューの所有者を指定する文字列です。
store	ビューの名前を指定する文字列です。
owner_d	ベース・オブジェクトの所有者を指定する文字列です。
store_d	ベース・オブジェクト名を指定する文字列です。

5.4 Consolidator をカスタマイズするための拡張機能

次に示す機能には、ほとんどのアプリケーション設計には不要な、特殊な機能が含まれています。このような機能を使用するには、Java および操作するデータベースの設計についての十分な知識（問合せの構成、表の並べ方、適用される依存性など）が要求されます。内容は次のとおりです。

- 5.4.1 項「MyCompose を使用した構成フェーズのカスタマイズ」
- 5.4.2 項「Sync Discovery API」
- 5.4.3 項「マップ表のパーティション API」
- 5.4.4 項「AlterPublicationItem を使用したパブリケーション・アイテムの変更」
- 5.4.5 項「複数表パブリケーション（ビュー）の高速リフレッシュおよび更新操作」

5.4.1 MyCompose を使用した構成フェーズのカスタマイズ

構成フェーズでは、サーバー側の 1 つ以上の実表についての問合せが必要になります。パブリケーション・アイテムに対して生成された DML 操作は、問合せで指定されているように、クライアントにダウンロードするアウトキューに格納されます。Consolidator は、サーバー側の実表についての DML 物理ログを使用して、これらの DML 操作を「一般的な」方法で管理します。DML 操作が複雑な場合、このプロセスはリソース集中型になる可能性があります。たとえば、データのサブセット化に複雑な問合せが使用されている場合などです。このプロセスをカスタマイズするツール製品には、オーバーライド可能な compose メソッドを持つ拡張可能な MyCompose や、カスタマイズされたクラスを登録およびロードするための Consolidator API などがあります。

5.4.1.1 ユーザー定義サブクラスとしての MyCompose の拡張

MyCompose は、ユーザー定義サブクラスを作成するためのスーパークラスとして機能する抽象クラスです。次に例を示します。

ItemACompose

```
public class ItemACompose extends oracle.lite.sync.MyCompose
{
...
}
```

ユーザー定義クラスは、実表の DML ログを解析することにより、クライアントに送信されるパブリケーション・アイテムの DML 操作を生成します。拡張 MyCompose サブクラスは、パブリケーション・アイテムに登録され、選択したパブリケーション・アイテムに関する構成フェーズのすべての操作を継承します。拡張 MyCompose クラスは、十分な汎用性を持つ場合は、複数のパブリケーション・アイテムに登録できます。ただし、内部的には、パブリケーション・アイテムごとに拡張クラスの一意インスタンスが存在します。

5.4.1.2 MyCompose のプライマリ・メソッド

MyCompose クラスは、needCompose、doCompose、init および destroy の 4 つのメソッドを使用して構成フェーズをカスタマイズします。カスタマイズしたサブクラスで、これらのメソッドの 1 つ以上をオーバーライドし、構成フェーズの操作をカスタマイズすることができます。一度に 1 クライアントの構成フェーズをカスタマイズする場合は、doCompose および needCompose のみを使用して実行できます。init および destroy メソッドは、個別のクライアント処理の前または後に、すべてのクライアントに対してあるプロセスを実行する必要がある場合に使用します。これ以外にも、5.4.1.3 項「MyCompose の補助メソッド」で説明されているメソッドがいくつかあります。5.4.1.3 項では、これらの 4 つのメソッドの使用方法についての有効な情報が提供されています。

5.4.1.2.1 needCompose メソッド

このメソッドは、ダウンロードする特定のパブリケーション・アイテムに対して変更を行ったクライアントの識別に使用します。このメソッドは、doCompose をトリガーする手段として使用する場合に有効です。

構文

```
public int needCompose(Connection conn,
  String user_name) throws Throwable
```

needCompose のパラメータを次の表に示します。

表 5-19 に、needCompose のパラメータおよびその説明を示します。

表 5-19 needCompose のパラメータ

パラメータ	定義
conn	Mobile サーバー・リポジトリへのデータベース接続です。
user_name	データベースに接続するユーザーを指定します。

次の例では、クライアントの実表の変更を調べます。この例では、「使用済み」レコードが存在します。変更がある場合、このメソッドは doCompose メソッドをトリガーする MyCompose.YES を返します。

例

```
public int needCompose(String user_name) throws Throwable{
    boolean baseDirty = false;
    String [][] baseTables = this.getBaseTables();

    for(int i = 0; i < baseTables.length; i++){
        if(this.baseTableDirty(baseTables[i][0], baseTables[i][1])){
            baseDirty = true;
```

```
                break;
            }
        }

        if(baseDirty){
            return MyCompose.YES;
        }else{
            return MyCompose.NO;
        }
    }
```

このサンプル・コードは、needCompose メソッドをオーバーライドし、5.4.1.3 項「MyCompose の補助メソッド」に示されている補助メソッドを使用して、パブリケーション・アイテム内のいずれかの表に、クライアントに送信する必要がある変更内容が含まれているかどうかを確認します。この例では、実表を取り出し、変更（「使用済み」レコード）の有無を確認します。このテストの結果が True の場合、doCompose のコールをトリガーする値「Yes」が返されます。

5.4.1.2.2 doCompose メソッド

このメソッドは、クライアントのサブスクライブ先となる特定のパブリケーション・アイテムについての DML ログ表を移入します。

構文

```
public int doCompose(Connection conn,
   String user_name) throws Throwable
```

doCompose のパラメータを次の表に示します。

表 5-20 に、doCompose のパラメータおよびその説明を示します。

表 5-20 doCompose のパラメータ

パラメータ	定義
conn	Mobile サーバー・リポジトリへのデータベース接続です。
user_name	データベースに接続するユーザーを指定します。

次の例には、1 つのみの実表を持つパブリケーション・アイテムが含まれています。このパブリケーション・アイテムで、実表を対象とした DML（挿入、更新または削除）操作が実行されます。このメソッドは、パブリケーション・アイテムにサブスクライブされているユーザーごとにコールされます。

例

```
    public int doCompose(Connection conn, String user_name) throws Throwable {
        int rowCount = 0;

        String [][] baseTables = this.getBaseTables();
        String baseTableDMLLogName =
            this.getBaseTableDMLLogName(baseTables[0][0], baseTables[0][1]);
        String baseTablePK =
            this.getBaseTablePK(baseTables[0][0],baseTables[0][1]);
        String pubItemDMLTableName = this.getPubItemDMLTableName();

        String sql = "INSERT INTO " + pubItemDMLTableName
            + " SELECT " +  baseTablePK + ", DMLTYPE$$ FROM " +
 baseTableDMLLogName;

        Statement st = conn.createStatement();
        rowCount = st.executeUpdate(sql);
        st.close();
        return rowCount;
    }
```

このサンプル・コードは、doCompose メソッドをオーバーライドし、5.4.1.3 項「MyCompose の補助メソッド」に示されている補助メソッドを使用して、SQL 文を作成します。このサンプルを使用し、適切な get メソッドを使用して、MyCompose により、実表、実表の主キー、実表の DML ログ名およびパブリケーション・アイテムの DML 表名を取り出します。これらのメソッドにより返された表名およびその他の情報を使用して、実表の DML ログから、実表の主キーおよび DML 操作の内容に含まれているパブリケーション・アイテムの DML 表への挿入を実行する動的 SQL 文（「sql」）を作成できます。

5.4.1.2.3 init メソッド

このメソッドは、ユーザー作成による構成準備プロセスのフレームワークを提供します。init メソッドは、個々のユーザーの構成フェーズより前に、すべてのユーザーに対して 1 回コールされます。デフォルトの実装には影響しません。

構文

```
public void init(Connection conn)
```

init のパラメータを次の表に示します。

表 5-21 に、init のパラメータ名およびその定義を示します。

表 5-21 init のパラメータ

パラメータ	定義
conn	Mobile サーバー・リポジトリへのデータベース接続です。

5.4.1.2.4 destroy メソッド

このメソッドは、ユーザー作成によるクリーンアップ・プロセスのフレームワークを提供します。destroy メソッドは、個々のユーザーの構成フェーズの後で、すべてのユーザーに対して 1 回コールされます。デフォルトの実装には影響しません。

構文

```
public void destroy(Connection conn)
```

destroy のパラメータを次の表に示します。

表 5-22 に、destroy のパラメータ名およびその定義を示します。

表 5-22 destroy のパラメータ

パラメータ	定義
conn	Mobile サーバー・リポジトリへのデータベース接続です。

5.4.1.3 MyCompose の補助メソッド

次のメソッドは、MyCompose のプライマリ・メソッドで使用する情報を返します。

5.4.1.3.1 getPublication

このメソッドは、パブリケーション名を返します。

構文

```
public String getPublication()
```

5.4.1.3.2 getPublicationItem

このメソッドは、パブリケーション・アイテム名を返します。

構文

```
public String getPublicationItem()
```

5.4.1.3.3 getPubItemDMLTableName

このメソッドは、doCompose または init が挿入されることになっている DML 表または DML 表ビューの名前（スキーマ名など）を返します。

構文

```
public String getPubItemDMLTableName()
```

戻り値は、動的 SQL 文に埋め込むことができます。表またはビューの構造は次のとおりです。

```
<PubItem PK> DMLTYPE$$
```

getPubItemDMLTableName のパラメータを次の表に示します。

表 5-23 に、getPubItemDMLTableName の View Structure のパラメータおよびその定義を示します。

表 5-23 getPubItemDMLTableName の View Structure のパラメータ

パラメータ	定義
PubItemPK	getPubItemPK() により返される値です。
DMLTYPE$$	値は、挿入を表す「I」、削除を表す「D」、または更新を表す「U」のいずれかです。

5.4.1.3.4 getPubItemPK

このメソッドは、リストされたパブリケーションの主キーを、カンマで区切られた形式（<col1>,<col2>,<col3>）で返します。

構文

```
public String getPubItemPK() throws Throwable
```

5.4.1.3.5 getBaseTables

このメソッドは、パブリケーション・アイテムに対するすべての実表を、2 つの文字列で構成された配列で返します。2 つの文字列で構成される各配列には、実表のスキーマおよび名前が含まれます。親表は常に最初に返される実表（baseTables[0]）です。

構文

```
public string [] [] getBaseTables() throws Throwable
```

5.4.1.3.6 getBaseTablePK

このメソッドは、リストされた実表の主キーを、カンマで区切られた形式（<col1>,<col2>,<col3>）で返します。

構文

```
public String getBaseTablePK
(String owner,
String baseTable) throws Throwable
```

getBaseTablePK のパラメータを次の表に示します。

表 5-24 に、getBaseTablePK のパラメータおよびその定義を示します。

表 5-24 getBaseTablePK のパラメータ

パラメータ	定義
owner	実表の所有者のスキーマ名です。
baseTable	実表の名前です。

5.4.1.3.7 baseTableDirty

このメソッドは、実表に同期を必要とする変更があるかどうかを示すブール値を返します。

構文

```
public boolean baseTableDirty(String owner, String store)
```

baseTableDirty のパラメータを次の表に示します。

表 5-25 に、baseTableDirty のパラメータおよびその定義を示します。

表 5-25 baseTableDirty のパラメータ

パラメータ	定義
owner	実表のスキーマ名です。
store	実表の名前です。

5.4.1.3.8 getBaseTableDMLLogName

このメソッドは、DML 物理ログ表または実表の DML ログ表ビューの名前を返します。

構文

```
public string getBaseTableDMLLogName(String owner, String baseTable)
```

getBaseTableDMLLogName のパラメータを次の表に示します。

表 5-26 に、getBaseTableDMLLogName のパラメータおよびその定義を示します。

表 5-26 getBaseTableDMLLogName のパラメータ

パラメータ	定義
owner	実表の所有者のスキーマ名です。
baseTable	実表の名前です。

戻り値は、動的 SQL 文に埋め込むことができます。パブリケーション・アイテムに複数の実表がある場合は、複数の物理ログが存在する可能性があります。親実表の物理主キーは、パブリケーション・アイテムの主キーに対応します。ログの構造は次のとおりです。

```
<Base Table PK> DMLTYPE$$
```

getBaseTableDMLLogName の View Structure のパラメータを次の表に示します。

表 5-27 に、getBaseTableDMLLogName の View Structure のパラメータおよびその定義を示します。

表 5-27 getBaseTableDMLLogName の View Structure のパラメータ

パラメータ	定義
Base Table PK	親実表の主キーです。
DMLTYPE$$	値は、挿入を表す「I」、削除を表す「D」、または更新を表す「U」のいずれかです。

5.4.1.3.9 getMapView()

このメソッドは、動的 SQL 文で使用できるマップ表のビューを返します。このマップ表には、各クライアント・デバイスの主キーのリストが含まれています。このビューはインライン・ビューになる場合もあります。

構文

```
public String getMapView() throws Throwable
```

マップ表のビューの構造は次のとおりです。

```
CLID$$CS <Pub Item PK> DMLTYPE$$
```

マップ表ビューのパラメータを次の表に示します。

表 5-28 に、getMapView の View Structure のパラメータおよびその定義を示します。

表 5-28 getMapView の View Structure のパラメータ

パラメータ	定義
CLID$$CS	クライアント ID 列です。
Base Table PK	パブリケーション・アイテムの主キー列です。
DMLTYPE$$	値は、挿入を表す「I」、削除を表す「D」、または更新を表す「U」のいずれかです。

5.4.1.4 MyCompose サブクラスを登録するための Consolidator API メソッド

サブクラスは、作成後、パブリケーションに登録する必要があります。パブリケーション・アイテムからサブクラスの追加および削除を実行可能にするために、RegisterMyCompose と DeRegisterMyCompose という 2 つのメソッドが Consolidator API に追加されています。

5.4.1.4.1 RegisterMyCompose メソッド

RegisterMyCompose メソッドは、サブクラスを登録し、クラスのバイトコードを含め、それを Mobile サーバー・リポジトリにロードします。コードをリポジトリにロードすることにより、実行時にロードすることなくサブクラスを使用できます。

構文

```
public static void RegisterMyCompose
  ( String publication,
    String pubItem,
    String className,
    boolean reloadBytecode) throws Throwable
```

RegisterMyCompose のパラメータを次の表に示します。

表 5-29 に、RegisterMyCompose のパラメータおよびその定義を示します。

表 5-29 RegisterMyCompose のパラメータ

パラメータ	定義
publication	パブリケーション・アイテムが属するパブリケーションの名前です。
pubItem	サブクラスの登録先となるパブリケーション・アイテムの名前です。
className	カスタマイズされた MyCompose サブクラスの名前です。
reloadedBytecode	この値が True の場合、Mobile サーバー・リポジトリ内にあるクラスの既存のバイトコードが上書きされます。

5.4.1.4.2 DeRegisterMyCompose

DeRegisterMyCompose メソッドは、Mobile サーバー・リポジトリからサブクラスを削除します。

構文

```
public static void DeRegisterMyCompose
  ( String publication,
    String pubItem,
```

```
    boolean removeBytecode) throws Throwable
```

DeRegisterMyCompose のパラメータを次の表に示します。

表 5-30 に、DeRegisterMyCompose のパラメータおよびその定義を示します。

表 5-30 DeRegisterMyCompose のパラメータ

パラメータ	定義
publication	パブリケーション・アイテムが属するパブリケーションの名前です。
pubItem	サブクラスの登録先となるパブリケーション・アイテムの名前です。
removeBytecode	この値が True の場合、Mobile サーバー・リポジトリ内にあるクラスの既存のバイトコードが削除されます。バイトコードが削除されると、このクラスに登録されているすべてのパブリケーション・アイテムの登録が削除されます。

5.4.2 Sync Discovery API

Sync Discovery 機能は、特定のクライアントを対象としたダウンロードのサイズを履歴データに基づいて見積もるように要求する場合に使用します。履歴データのメンテナンスを行うために、次の統計値が収集されます。

- パブリケーション・アイテムごとに送信された行の合計数
- これらの行のデータの合計サイズ
- これらの行の圧縮データのサイズ

5.4.2.1 getDownloadInfo メソッド

API は、DownloadInfo オブジェクトを返す getDownloadInfo メソッドで構成されます。DownloadInfo オブジェクトには、一連の PublicationSize オブジェクトとアクセス方法が含まれています。PublicationSize オブジェクトは、パブリケーション・アイテムのサイズ情報を扱います。Iterator iterator() メソッドは、DownloadInfo オブジェクト内の各 PublicationSize オブジェクトの表示に使用できます。

構文

```
public DownloadInfo getDownloadInfo
   (String user_name,
    boolean uncompressed,
    boolean completeRefresh)
```

getDownloadInfo のパラメータを次の表に示します。

表 5-31 に、getDownloadInfo のパラメータおよびその説明を示します。

表 5-31 getDownloadInfo のパラメータ

パラメータ	説明
user_name	ユーザー名です。
uncompressed	True に設定すると、データ・オブジェクトの実際のサイズが返されます。False に設定すると、圧縮後のデータ・オブジェクトのサイズが返されます。
completeRefresh	True に設定すると、リフレッシュ・モードと関係なく、完全リフレッシュ時に同期がとられるすべての行のサイズが返されます。

例

```
DownloadInfo dl = Consolidator.getDownloadInfo("S11U1", true, true);
```

5.4.2.2 DownloadInfo クラスの access メソッド

DownloadInfo クラスにより提供される access メソッドを次の表に示します。

表 5-32 に、DownloadInfo クラスの access メソッドおよびその定義を示します。

表 5-32 DownloadInfo クラスの access メソッド

メソッド	定義
public Iterator iterator ()	このメソッドは、ユーザーが DownloadInfo オブジェクト内に含まれているすべての PublicationSize オブジェクトを横断できるように、イテレータ・オブジェクトを返します。
public long getTotalSize ()	このメソッドは、すべての PublicationSize オブジェクトのサイズ情報を、バイト単位で返します。また、ユーザーがサブスクライブするすべてのパブリケーション・アイテムのサイズを、拡張子順に返します。パブリケーション・アイテムに履歴情報がない場合は、「-1」が返されます。
public long getPubSize (String pubName)	このメソッドは、文字列 pubName により参照されるパブリケーションに属するすべてのパブリケーション・アイテムのサイズを返します。パブリケーション・アイテムに履歴情報がない場合は、「-1」が返されます。

表 5-32 DownloadInfo クラスの access メソッド（続き）

メソッド	定義
`public long getPubRecCount (String pubName)`	このメソッドは、文字列 pubName により参照されるパブリケーションに属するすべてのパブリケーション・アイテムのレコードの合計数を返します。このレコードの合計数の同期は、次回の同期時に行われます。
`public long getPubItemSize (String pubItemName)`	このメソッドは、pubItemName により参照される特定のパブリケーション・アイテムのサイズを返します。このメソッドは、次の規則に従います。 1. パブリケーション・アイテムが空の場合は、「0」を返します。 2. パブリケーション・アイテムに履歴情報がない場合は、「-1」が返されます。
`public long getPubItemRecCount (String pubItemName)`	このメソッドは、pubItemName により参照される特定のパブリケーション・アイテムのレコード数を返します。レコード数の同期は、次回の同期時に行われます。

5.4.2.3 PublicationSize クラス

PublicationSize クラスにより提供される access メソッドを次の表に示します。

表 5-33 に、PublicationSize クラスの access メソッドのパラメータおよびその定義を示します。

表 5-33 PublicationSize クラスの access メソッド

パラメータ	定義
`public String getPubName ()`	パブリケーション・アイテムを含むパブリケーションの名前を返します。
`public String getPubItemName ()`	PublicationSize オブジェクトにより参照されるパブリケーション・アイテムの名前を返します。
`public long getSize ()`	PublicationSize オブジェクトにより参照されるパブリケーション・アイテムの合計サイズを返します。
`public long getNumOfRows()`	次回の同期時に同期されるパブリケーション・アイテムの行数を返します。

サンプル・コード

```
import   java.sql.*;
import    java.util.Iterator;
import    java.util.HashSet;

import   oracle.lite.sync.ConsolidatorManager;
import   oracle.lite.sync.DownloadInfo;
import   oracle.lite.sync.PublicationSize;

public class TestGetDownloadInfo
{

   public static void main(String argv[]) throws Throwable
   {
// Open Consolidator connection
      try
      {
// Create a ConsolidatorManager object
         ConsolidatorManager cm = new ConsolidatorManager ();
// Open a Consolidator connection
         cm.OpenConnection ("MOBILEADMIN", "MANAGER",
                     "jdbc:oracle:thin:@server:1521:orcl", System.out);
// Call getDownloadInfo
         DownloadInfo dlInfo = cm.getDownloadInfo ("S11U1", true, true);
// Call iterator for the Iterator object and then we can use that to transverse
// through the set of PublicationSize objects.
         Iterator it = dlInfo.iterator ();
// A temporary holder for the PublicationSize object.
         PublicationSize ps = null;
// A temporary holder for the name of all the Publications in a HashSet object.
         HashSet pubNames = new HashSet ();
// A temporary holder for the name of all the Publication Items in a HashSet
// object.
         HashSet pubItemNames = new HashSet ();
// Traverse through the set.
         while (it.hasNext ())
         {
// Obtain the next PublicationSize object by calling next ().
            ps = (PublicationSize)it.next ();

// Obtain the name of the Publication this PublicationSize object is associated
// with by calling getPubName ().
            pubName = ps.getPubName ();
            System.out.println ("Publication: " + pubName);

// We save pubName for later use.
            pubNames.add (pubName);
```

```
// Obtain the Publication name of it by calling getPubName ().
            pubItemName = ps.getPubItemName ();
            System.out.println ("Publication Item Name: " + pubItemName);

// We save pubItemName for later use.
            pubItemNames.add (pubItemName);

// Obtain the size of it by calling getSize ().
            size = ps.getSize ();
            System.out.println ("Size of the Publication: " + size);

// Obtain the number of rows by calling getNumOfRows ().
            numOfRows = ps.getNumOfRows ();
            System.out.println ("Number of rows in the Publication: "
                                + numOfRows);
         }

// Obtain the size of all the Publications contained in the
// DownloadInfo objects.
         long totalSize = dlInfo.getTotalSize ();
         System.out.println ("Total size of all Publications: " + totalSize);

// A temporary holder for the Publication size.
         long pubSize = 0;

// A temporary holder for the Publication number of rows.
         long pubRecCount = 0;

// A temporary holder for the name of the Publication.
         String tmpPubName = null;

// Transverse through the Publication names that we saved earlier.
         it = pubNames.iterator ();
         while (it.hasNext ())
         {
// Obtain the saved name.
            tmpPubName = (String) it.next ();

// Obtain the size of the Publication.
            pubSize = dlInfo.getPubSize (tmpPubName);
            System.out.println ("Size of " + tmpPubName + ": " + pubSize);

// Obtain the number of rows of the Publication.
            pubRecCount = dlInfo.getPubRecCount (tmpPubName);
            System.out.println ("Number of rows in " + tmpPubName + ": "
                                + pubRecCount);
```

```
        }

// A temporary holder for the Publication Item size.
        long pubItemSize = 0;

// A temporary holder for the Publication Item number of rows.
        long pubItemRecCount = 0;

// A temporary holder for the name of the Publication Item.
        String tmpPubItemName = null;

// Traverse through the Publication Item names that we saved earlier.
        it = pubItemNames.iterator ();
        while (it.hasNext ())
        {
// Obtain the saved name.
           tmpPubItemName = (String) it.next ();

// Obtain the size of the Publication Item.
           pubItemSize = dlInfo.getPubItemSize (tmpPubItemName);
           System.out.println ("Size of " + pubItemSize + ": " + pubItemSize);

// Obtain the number of rows of the Publication Item.
           pubItemRecCount = dlInfo.getPubItemRecCount (tmpPubItemName);
           System.out.println ("Number of rows in " + tmpPubItemName + ": "
                               + pubItemRecCount);
        }
        System.out.println ();

// Close the connection
        cm.CloseConnection ();
      }
      catch (Exception e)
      {
         e.printStackTrace();
      }
   }
}
```

5.4.3 マップ表のパーティション API

マップ表と呼ばれる Consolidator データベース・オブジェクトは、各 Mobile クライアントの状態のメンテナンスに使用します。ユーザー数が多く、各ユーザーが大量のデータをサブスクライブする場合、マップ表のサイズが非常に大きくなり、拡張性の問題が発生する可能性があります。次の API を使用すると、マップ表をユーザー名別にパーティション化して簡単に管理できます。

この API では、マップ表のパーティションの作成、パーティションの追加、1 つまたはすべてのパーティションの削除、マップ表のパーティションのマージが可能です。マップ表のパーティションは、ALL_PARTITIONS データベース・カタログ・ビューを使用して監視できます。

注意： このようなパーティション化は、Oracle サーバーが提供するパーティション機能とは関係なく、Oracle9*i* Lite でのみ使用されます。

5.4.3.1 マップ表のパーティション作成 API

参照パブリケーション・アイテムのマップ表に対応するパーティションを作成します。マップ表にデータが存在する場合、そのデータは作成するパーティションに転送されます。パーティションが正常に作成された後、SQL コマンド TRUNCATE TABLE を使用して、マップ表を切り捨て、冗長なデータを削除できます。

構文

```
public static void PartitionMap
   (String pub_item,
    int num_parts,
    String storage,
    String ind_storage) throws Throwable
```

PartitionMap のパラメータを次の表に示します。

表 5-34 に、PartitionMap のパラメータおよびその定義を示します。

表 5-34 PartitionMap のパラメータ

パラメータ	定義
pub_item	マップ表がパーティション化されるパブリケーション・アイテムです。
num_parts	パーティションの数です。
storage	記憶域パラメータを指定する文字列です。このパラメータには、SQL コマンド CREATE TABLE と同じ構文を使用する必要があります。詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

表 5-34 PartitionMap のパラメータ（続き）

パラメータ	定義
ind_storage	パーティションに索引の記憶域パラメータを指定する文字列です。このパラメータには、SQL コマンド CREATE INDEX と同じ構文を使用する必要があります。詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

例

```
Consolidator.PartitionMap("P_SAMPLE1", "5", "tablespace mobileadmin", "initrans 10
pctfree 70");
```

5.4.3.2 マップ表のパーティション追加 API

参照パブリケーション・アイテムのマップ表に対応するパーティションを追加します。マップ表にデータが存在する場合、そのデータは作成するパーティションに転送されます。SQL コマンド TRUNCATE TABLE を使用すると、パーティションが正常に作成された後、マップ表を切り捨て、冗長なデータを削除できます。

構文

```
public static void AddMapPartition
   ( String pub_item,
    int num_parts,
    String storage,
    String ind_storage) throws Throwable
```

AddMapPartition のパラメータを次の表に示します。

表 5-35 に、AddMapPartition のパラメータおよびその定義を示します。

表 5-35 AddMapPartitions のパラメータ

パラメータ	定義
pub_item	マップ表がパーティション化されるパブリケーション・アイテムです。
num_parts	パーティションの数です。
storage	記憶域パラメータを指定する文字列です。このパラメータには、SQL コマンド CREATE TABLE と同じ構文を使用する必要があります。詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

表 5-35 AddMapPartitions のパラメータ（続き）

パラメータ	定義
ind_storage	パーティションに索引の記憶域パラメータを指定する文字列です。このパラメータには、SQL コマンド CREATE INDEX と同じ構文を使用する必要があります。詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

例

```
Consolidator.PartitionMap("P_SAMPLE1", "5", "tablespace mobileadmin", "intitrans 10
pctfree 40");
```

5.4.3.3 マップ表のパーティション削除 API

このパラメータを使用すると、指定したパーティションを削除することができます。次の例のパラメータ partition は、パーティションの名前です。パーティション名は Consolidator が付けるため、ALL_PARTITIONS 表ビュー CV$ALL_PARTITIONS の問合せによりパーティション名を取得する必要があります。

構文

```
public static void DropMapPartition( String partition) throws Throwable
```

例

```
Consolidator.DropMapPartition("MAP101_1");
```

5.4.3.4 マップ表の全パーティション削除 API

指定したパブリケーション・アイテムに対応するマップ表のすべてのパーティションを削除します。

構文

```
public static void DropAllMapPartitions( String pub_item) throws Throwable
```

例

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

5.4.3.5 マップ表のパーティションのマージ API

データをパーティション間でマージします。パーティション名は Consolidator が付けるため、ALL_PARTITIONS 表ビュー CV$ALL_PARTITIONS の問合せにより、パーティション名を取得する必要があります。

構文

```
public static void MergeMapPartitions
   ( String from_partition,
    String to_partiton) throws Throwable
```

例

```
Consolidator.MergeMapPartition(""MAP101_1", "MAP101_2");
```

5.4.4 AlterPublicationItem を使用したパブリケーション・アイテムの変更

既存のパブリケーション・アイテム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。これは変更済全パブリケーション・アイテム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。つまり、クライアントはクライアント・データベース上でスナップショットを直接変更することはできません。たとえば、EPOC 上では 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・アイテム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・アイテムのアップグレードには常に、変更済パブリケーション・アイテム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・アイテム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・アイテムのバージョンを最大 2 個サポートする必要があります。

5.4.4.1 パブリケーション・アイテムの変更

これにより、既存のパブリケーション・アイテムに列を追加できます。WHERE 句は変更できますが、サブスクリプション・パラメータは追加できません。

構文

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

AlterPublicationItem のパラメータを次の表に示します。

表 5-36 に、AlterPublicationItem のパラメータおよびその定義を示します。

表 5-36 AlterPublicationItem のパラメータ

パラメータ	説明
name	パブリケーション・アイテム名を指定する文字列です。
select_stmt	追加列の含まれた新規パブリケーション・アイテムの Select 文です。

例

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

5.4.5 複数表パブリケーション（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ビューと呼ばれる複合複数表パブリケーション・アイテム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の同期がとられ、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。リフレッシュ・モードは、CreatePublicationItem API コールを使用してパブリケーション・アイテムを作成する際に設定されます。リフレッシュ・モードを変更するには、パブリケーション・アイテムを削除してから、適切なモードを使用してパブリケーション・アイテムを再作成する必要があります。

5.4.5.1 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・アイテムを作成する前に、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。

5.4.5.2 親表のヒントと INSTEAD OF トリガーの使用

ビューに基づいたパブリケーション・アイテム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 トリガーまたは DML プロシージャのコールアウト

5.4.5.2.1 親のヒントの作成

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、ParentHint 関数を使用して指定します。この関数の構文は、次のとおりです。

```
public static void ParentHint
   (String owner,
    Sting store,
    String owner_d,
    String store_d) throws Throwable
```

ParentHint のパラメータを次の表に示します。

表 5-37 に、ParentHint のパラメータおよびその説明を示します。

表 5-37 ParentHint のパラメータ

パラメータ	説明
owner	ビューの所有者を指定する文字列です。
store	ビューの名前を指定する文字列です。
owner_d	ベース・オブジェクトの所有者を指定する文字列です。
store_d	ベース・オブジェクト名を指定する文字列です。

例

```
Consolidator.ParentHint("SAMPLE3","ADDROLRL4P","SAMPLE3","ADDRESS");
```

5.4.5.2.2 INSTEAD OF トリガー

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。

5.4.5.3 ビューの高速リフレッシュ

パブリケーション・アイテムは、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・アイテム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、PrimaryKeyHint を使用して実現できます。

5.4.5.3.1 PrimaryKeyHint

PrimaryKeyHint の構文は次のとおりです。

```
public static void PrimaryKeyHint
   (String publication_item,
    String column,
    String b_owner,
    String b_store,
    String b_column) throws Throwable
```

PrimaryKeyHint のパラメータを次の表に示します。

表 5-38 に、PrimaryKeyHint のパラメータおよびその説明を示します。

表 5-38 PrimaryKeyHint のパラメータ

パラメータ	説明
publication_item	主キーのヒントのマップ先となるパブリケーション・アイテムの名前です。
owner	ビューの所有者を指定する文字列です。
store	ビューの名前を指定する文字列です。
store_d	ベース・オブジェクトの所有者を指定する文字列です。
b_column	ヒントで使用される実表列の名前です。

例

```
Consolidator.ParentHint("SAMPLE3","ADDROLRL4P","SAMPLE3","ADDRESS");
```

5.4.5.4 ビューの完全リフレッシュ

パブリケーション・アイテムは、Consolidator API の Complete Refresh 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。

5.4.5.4.1 CompleteRefresh

CompleteRefresh の構文は次のとおりです。

```
public static void CompleteRefresh
   (String user_name,
    String publication) throws Throwable
```

CompleteRefresh のパラメータを次の表に示します。

表 5-39 に、CompleteRefresh のパラメータおよびその説明を示します。

表 5-39 CompleteRefresh のパラメータ

パラメータ	説明
user_name	Consolidator のユーザー名です。
publication	リフレッシュするパブリケーションの名前です。

5.4.6 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・アイテムに対して、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。仮想主キーの作成方法と削除方法を次に示します。

5.4.6.1 仮想主キー列の作成

これは仮想主キー列を作成します。

構文

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

CreateVirtualPKColumn のパラメータを次の表に示します。

表 5-40 に、CreateVirtualPKColumn のパラメータおよびその説明を示します。

表 5-40 CreateVirtualPKColumn のパラメータ

パラメータ	説明
owner	実表またはビューの所有者を指定する文字列です。
store	実表またはビューを指定する文字列です。
column	主キー列を示す文字列です。

例

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

5.4.6.2 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。

構文

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

DropVirtualPKColumn のパラメータを次の表に示します。

表 5-41 に、DropVirtualPKColumn のパラメータおよびその定義を示します。

表 5-41 DropVirtualPKColumn のパラメータ

パラメータ	説明
owner	実表またはビューの所有者を指定する文字列です。
store	実表またはビューを指定する文字列です。

例

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

5.4.7 パブリケーション・アイテム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・アイテム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 テンプレートはスナップショットとより効率的に結合します。

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・アイテムの問合せが 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。

次の例は、MGP が構成フェーズで使用するテンプレートです。MGP は、このテンプレートを使用して、無効になったためにクライアントから削除する必要があるクライアント・レコードを識別します。

```
UPDATE pub_item_map map
SET delete = true
WHERE client = <user_name>
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。その結果、MGP 構成フェーズで、パフォーマンスに重大な影響を与える可能性があります。パブリケーション・アイテム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・アイテム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。

5.4.7.1 パブリケーション・アイテム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・アイテムの問合せキャッシングが有効になります。

構文

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

EnablePublicationItemQueryCache のパラメータを次の表に示します。

表 5-42 に、EnablePublicationItemQueryCache のパラメータおよびその説明を示します。

表 5-42 EnablePublicationItemQueryCache のパラメータ

パラメータ	説明
name	パブリケーション・アイテム名を指定する文字列です。

例

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

5.4.7.2 パブリケーション・アイテム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・アイテムの問合せキャッシングが無効になります。

構文

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

`DisablePublicationItemQueryCache` のパラメータを次の表に示します。

表 5-43 に、DisablePublicationItemQueryCache のパラメータおよびその定義を示します。

表 5-43 DisablePublicationItemQueryCache のパラメータ

パラメータ	説明
`name`	パブリケーション・アイテム名を指定する文字列です。

例

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

5.4.8 ユーザー定義の PL/SQL プロシージャのバインド

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 プロシージャをパブリケーション・アイテムにバインドすることで、Mobile サーバーにアタッチできます。プロシージャは、Consolidator API の `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・アイテム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 プロシージャは、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、5.4.11 項「更新可能パブリケーション・アイテムの外部キー制約」を参照してください。これらのコールの使用方法の詳細は、5.4.12 項「構成および適用の前後に行うコールバックのカスタマイズ」を参照してください。

5.4.9 レプリケーションをカスタマイズするためのキュー・インタフェース

アプリケーション開発者は、CreateQueuePublicationItem API を使用して、プログラムによりレプリケーション・プロセスを管理できます。通常、MGP によりインキューとアウトキューの両方が管理されるため、この API によって、アプリケーション開発者は、5.4.9.3 項「キュー・インタフェースの PL/SQL プロシージャ」に示す PL/SQL パッケージを使用し、キューそのものを作成することで、同期セッション中にキュー操作を管理できます。

5.4.9.1 キュー・インタフェース操作

クライアントからデータが到着すると、パブリケーション・アイテム・インキューに格納されます。データがコミットされると、Consolidator は、UPLOAD_COMPLETE をコールします。現行同期セッション内のすべてのレコードには、同一トランザクション識別子が付けられます。Consolidator には、このトランザクション識別子を使用して新しいトランザクションを受け取ったパブリケーション・アイテム・インキューを示すキュー制御表（C$INQ+ 名前）があります。この表を参照して、処理が必要なキューを判断できます。

Consolidator は、同期セッションのダウンロード・フェーズを開始する前に、DOWNLOAD_INIT をコールします。このプロシージャでは、クライアントに送信するデータを判断するために設定または変更する必要がある設定を、カスタマイズすることができます。Consolidator は、クライアントのサブスクリプションに基づいてダウンロード可能なパブリケーション・アイテムのリストを検索します。パブリケーション・アイテムのリストとリフレッシュ・モード（完全リフレッシュの場合は「Y」、高速リフレッシュの場合は「N」）が一時表（C$PUB_LIST_Q）に挿入されます。Consolidator は C$PUB_LIST_Q を参照して、クライアントにダウンロードする項目を判断するため、この一時表内で項目の削除またはリフレッシュ・ステータスの変更を実行できます。

インキュー同様、アウトキュー内のすべてのレコードにトランザクション識別子（TRANID$$）を関連付ける必要があります。Consolidator は、クライアントにより正常に適用された最新のトランザクションを示す `last_tran` パラメータを渡します。クライアントにダウンロードされていないアウトキュー内の新しいレコードには、`curr_tran` パラメータの値をマークする必要があります。`curr_tran` の値は、`last_tran` の値よりも常に大きくなります。ただし、連続している必要はありません。Consolidator は、TRANID$$ の値が last_tran よりも大きい場合にのみ、アウトキューからレコードをダウンロードします。データがダウンロードされると、Consolidator は、DOWNLOAD_COMPLETE をコールします。

5.4.9.2 キューの作成

SQL を使用して、Mobile サーバー・リポジトリ内にアウトキューを手動で作成する必要があります。インキューを作成することもできます。Consolidator は、アウトキューおよびインキューが存在しない場合は、これらのキューを自動的に作成します。リポジトリに接続し、次の文を実行して、次の構造を持つインキューとアウトキューを作成します。

アウトキュー

```
'CTM$'+name
(
CLID$$CS   VARCHAR2 (30),
..
publication_item_store_columns (c1..cN),
..
TRANID$$   NUMBER (10),
DMLTYPE$$  CHAR (1) CHECK (DMLTYPE$$  IN ('I','U','D'),
)
```

インキュー

```
'CFM$'+name
(
CLID$$CS   VARCHAR2 (30),
TRANID$$   NUMBER (10),
SEQNO$$    NUMBER (10),

DMLTYPE$$  CHAR (1) CHECK (DMLTYPE$$  IN ('I','U','D'),
..
publication_item_store_columns (c1..cN),
..
)
```

Consolidator は、キュー制御表 C$INQ および一時表 C$PUB_LIST_Q を作成します。キュー制御表を調べて、どのパブリケーション・アイテムが新しいトランザクションを受け取ったかを判断することができます。

キュー制御表

```
'C$INQ'+name
(
CLIENTID   VARCHAR2 (30),
TRANID$$   NUMBER,
STORE      VARCHAR2 (30),

)
```

一時表

```
'C$PUB_LIST_Q'
(
NAME   VARCHAR2 (30),
COMP_REF   CHAR(1),
CHECK(COMP_REF IN('Y','N'))

)
```

手動により作成されるキューのパラメータを次の表に示します。

表 5-44 に、キュー・インタフェース作成のパラメータおよびその説明を示します。

表 5-44 キュー・インタフェース作成のパラメータ

パラメータ	説明
CLID$$CS	クライアントを識別する一意の文字列です。
TRANID$$	トランザクションを識別する一意の番号です。
SEQNO$$	インキュー（CFM$）のみにある、トランザクションごとの DML 言語操作を表す一意の数値です。
DMLTYPE$$	DML 文のタイプを確認します。 ■ T - 挿入 ■ D - 削除 ■ U - 更新 アウトキューのみ。
STORE	キュー制御表（C$INQ）のみにあるパブリケーション・アイテム名を表します。
NAME	一時表（C$PUB_LIST_Q）のみにあるパブリケーション・アイテム名です。
COMP_REF	パブリケーション・アイテムのリフレッシュ・モードを判断する場合に使用するフラグで、値は「Y」（はい）か「N」（いいえ）のいずれかです。

5.4.9.3 キュー・インタフェースの PL/SQL プロシージャ

次の PL/SQL パッケージ仕様により、キュー・インタフェースに必要なコールアウトが定義されます。

サンプル・コード

```
CREATE OR REPLACE PACKAGE CONS_QPKG AS
/*
 *     notifies that inq has new transaction
*/
PROCEDURE UPLOAD_COMPLETE(
     CLIENTID      IN     VARCHAR2,
     TRAN_ID       IN     NUMBER     -- IN queue tranid
     );
/*
 *     init data for download
*/
PROCEDURE DOWNLOAD_INIT(
     CLIENTID      IN     VARCHAR2,
     LAST_TRAN     IN     NUMBER,
     CURR_TRAN     IN     NUMBER,
     HIGH_PRTY     IN     VARCHAR2
     );
/*
 *  notifies when all the client's data is sent
*/
PROCEDURE DOWNLOAD_COMPLETE(
     CLIENTID     IN     VARCHAR2
     );

END CONS_QPKG;
/
```

5.4.9.4 CreateQueuePublicationItem API

この API コールは、パブリケーション・アイテムをキューの形式で作成します。この API コールは、パブリケーション・アイテムを登録し、CFM$name 表が存在しない場合は、この表をインキューとして作成します。

構文

```
public static void CreateQueuePublicationItem
   ( String name,
    String owner,
     String store,
     String select_stmt,
```

```
String pk_columns,
String cbk_owner,
String cbk_name) throws Throwable
```

CreateQueuePublicationItem のパラメータを次の表に示します。

表 5-45 に、CreateQueuePublicationItem のパラメータおよびその説明を示します。

表 5-45 CreateQueuePublicationItem のパラメータ

パラメータ	説明
name	新しいパブリケーション・アイテムおよびキュー名を定義します。
owner	実表またはビューの所有者です。
store	実表またはビューの名前を指定する値です。
select_stmt	新しいパブリケーション・アイテムに対する SELECT 文を指定する文字列です。この文には、サブスクリプション・パラメータを含めることができます。
pk_columns	カンマで区切られた、仮想主キーを作成するリストです。
cbk_owner	コールバック・パッケージの所有者を指定します。詳細は、5.4.12 項「構成および適用の前後に行うコールバックのカスタマイズ」を参照してください。これは拡張機能です。
cbk_name	コールバック・パッケージ名を指定します。詳細は、5.4.12 項「構成および適用の前後に行うコールバックのカスタマイズ」を参照してください。これは拡張機能です。

更新可能または高速リフレッシュ可能なキューを作成するには、owner.store の主キーを Consolidator に指定する必要があります。store に主キーが存在しない場合は、pk_columns パラメータに主キーを指定できます。pk_columns が NULL の場合、Consolidator は store の主キーを使用します。

5.4.9.5 リポジトリの外部での PL/SQL パッケージの定義

PL/SQL パッケージは、Mobile サーバー・リポジトリの外部で定義することができます。ただし、PL/SQL パッケージを機能させるには、必要に応じて、リポジトリ内で定義されているインキュー、アウトキュー、キュー制御表および一時表を参照させる必要があります。次の API コールは、プロシージャ名の取得、プロシージャの登録または削除に使用します。

5.4.9.5.1 RegisterQueuePkg

この API は、文字列「pkg」を現行プロシージャとして登録します。

構文

```
public String RegisterQueuePkg(String pkg) throws SQLException
```

例

```
Consolidator.RegisterQueuePkg("ASL.QUEUES_PKG");
```

5.4.9.5.2 GetQueuePkg

この API コールは、現在登録されているプロシージャの名前を返します。

構文

```
public String GetQueuePkg() throws SQLException
```

5.4.9.5.3 UnRegisterQueuePkg

この API コールは、現在登録されているプロシージャを削除します。

構文

```
public String UnRegisterQueuePkg() throws SQLException
```

5.4.10 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 プロシージャを作成することで実装できます。プロシージャには、次の仕様を含める必要があります。

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

5.4.11 更新可能パブリケーション・アイテムの外部キー制約

Oracle データベースとクライアントの間で更新可能モードにより表をレプリケートする場合、表に参照整合性制約が存在すると、外部キー制約違反が発生する可能性があります。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。

5.4.11.1 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。

5.4.11.2 BeforeApply および AfterApply での制約違反の回避

DEFERRABLE 制約を、`BeforeApply` および `AfterApply` 関数と組み合せて使用することにより、PL/SQL プロシージャを使用し、順序どおりでは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、次の手順を実行します。

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 プロシージャを、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 プロシージャでは、次のサンプルに示されているとおり、`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;
```

5.4.11.3 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。

5.4.11.1 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)
```

5.4.12 構成および適用の前後に行うコールバックのカスタマイズ

パブリケーション・アイテムの作成時に、ユーザーは、MGP バックグラウンド・プロセスの適用と構成フェーズ中に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)
```

5.4.13 DML 操作のコールバックのカスタマイズ

パブリケーション・アイテムを作成した後は、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・アイテムに対するすべての DML 操作にかわってコールされます。各パブリケーション・アイテムに対して、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..)
```

DML 操作をカスタマイズするためのパラメータを次の表に示します。

表 5-46 に、モバイル DML 操作のパラメータおよびその定義を示します。

表 5-46 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。

表 5-46 モバイル DML 操作のパラメータ（続き）

パラメータ	説明
COL1 ... COLn	パブリケーション・アイテムに定義されている列のリスト。列名は、パブリケーション・アイテム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・アイテムが「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るパブリケーション・アイテム「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・アイテムに連結できます。

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

example はパブリケーション・アイテム名で、PUB_example はパブリケーション名です。

この API コールの詳細は、『Consolidator Admin API Specification』を参照してください。

5.4.13.1 DML プロシージャの例

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・アイテムに対する実際の 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・アイテム名です。

5.4.14 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・アイテムを追加するときに、パブリケーション・アイテム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、クライアントにダウンロード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。

5.5 同期エラーと競合

Mobile サーバーでは、サーバーが削除した行を、同時にクライアントが更新した場合、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、その他のエラーはすべて同期エラーです。

Mobile サーバーでは、同期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 サーバーの同期競合が発生する理由は、次のとおりです。

- 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、5.5.3 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。

5.5.1 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同期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。

5.5.2 ウィニング・ルール

Mobile サーバーは、ウィニング・ルールを使用して同期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。

5.5.3 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・アイテムごとに、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 `ExecuteTransaction` 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、`PurgeTransaction` 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。Mobile サーバーのエラー・キューは C$EQ で、データは C$EQ に格納されます。

5.5.3.1 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。

構文

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

`ExecuteTransaction` のパラメータを次の表に示します。

表 5-47 に、ExecuteTransaction のパラメータおよびその定義を示します。

表 5-47 ExecuteTransaction のパラメータ

パラメータ	説明
user_name	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文字列です。

例

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

5.5.3.2 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。

構文

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

PurgeTransaction のパラメータを次の表に示します。

表 5-48 に、PurgeTransaction のパラメータおよびその定義を示します。

表 5-48 PurgeTransaction のパラメータ

パラメータ	説明
user_name	Mobile サーバー・ユーザー名です。
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文字列です。

例

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

5.6 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 データベースおよび 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。

5.6.1 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、同期時に Mobile Sync によって作成されます。Mobile サーバーは、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 データ型を選択します。表 5-49 に、データ変換値を示します。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。「」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 5-49 Oracle Lite データ型

Oracle データ型	1B	2B	4B	FLOAT	DOUBLE	DATETIME	LONG-VARBINARY	VARCHAR
INTEGER						×	×	×
VARCHAR2	×	×	×	×	×	×	×	
VARCHAR	×	×	×	×	×	×	×	
CHAR	×	×	×	×	×	×	×	
SMALLINT						×	×	×
FLOAT						×	×	×
DOUBLE PRECISION						×	×	×
NUMBER						×	×	×
NUMBER WITHOUT PRECISION	×	×	×	×		×	×	×
DATE	×	×	×	×	×		×	×
LONG RAW	×	×	×	×	×	×		×
LONG	×	×	×	×	×	×	×	
BLOB	×	×	×	×	×	×		×
CLOB	×	×	×	×	×	×	×	×

6

Oracle Mobile SQL（MSQL）

この章では、Oracle Mobile SQL （MSQL）コマンドおよび対応するコマンドラインの使用方法について説明します。内容は次のとおりです。

6.1 概要

Mobile SQL（MSQL）は、Windows 32 プラットフォーム上で稼働するアプリケーションです。これによって、MSQL 機能を実装するコマンドライン・インタフェースが提供されます。MSQL を使用すると、Oracle データベースに対するデータの格納と取得、ローカル・データベースに対する SQL コマンドの実行、およびその他のタスクの実行が可能です。MSQL を使用すると、次のタスクを実行できます。

- SQL コマンドの入力、編集、格納、取得および実行
- 表の列定義の表示
- データベース管理の実行

コマンド・プロンプトに次の文を入力して、サンプル MSQL スクリプトを実行します。

```
SQL> @<ORACLE_HOME>¥Mobile¥DBS¥Poldemo.sql
```

パスの <ORACLE_HOME> を、ご使用のマシンの ORACLE_HOME 変数の実際の名前と置き換えます。

この章で説明するコマンドラインの使用方法は、Oracle Lite でサポートされているすべての SQL コマンドに適用されます。詳細は、『Oracle9*i* Lite SQL リファレンス』を参照してください。

6.2 基本概念

次に、MSQL の基本概念の定義を説明します。詳細は、『Oracle9*i* Lite SQL リファレンス』を参照してください。

コマンド

特定のタスクを実行するための、オペレーティング・システムまたはソフトウェア（MSQL、Oracle など）への命令。

SQL コマンド

有効な SQL 文。

MSQL コマンド

MSQL の動作を制御するためのコマンド。

表

Oracle Lite データベースの記憶域の基本単位。

問合せ

1 つ以上の表から情報を取得する、読込み専用の SQL SELECT コマンド。

問合せ結果

問合せで取得されるデータ。

6.3 データベースへの接続

MSQL を使用して起動するには、最初にデータベースに接続する必要があります。データベースに接続するには、コマンド・プロンプトで次のコマンドを入力します。

```
SQL> msql <user_name>/<password>@<jdbc_url>
```

次に、<jdbc_url> 変数の例を示します。

```
jdbc:polite:ordersdb
```

JDBC およびその他の Java 関連の項目については、『Oracle9*i* Lite for Java 開発者ガイド』を参照してください。

6.4 MSQL でサポートされる SQL*Plus のコマンド

この項では、MSQL でサポートされる SQL*Plus コマンドについて説明します。ほとんどの MSQL コマンドは、SQL*Plus コマンド構文に基づきます。MSQL コマンドは、Oracle Lite でサポートされるすべての SQL コマンドをサポートします。次の項では、各 MSQL コマンドおよびその説明を示します。

6.4.1 CONN[ECT]

このコマンドを使用すると、適切なユーザー名およびパスワードを使用して、モバイル・データベースに接続できます。

例

```
connect <username>/<password>@jdbc:polite:<dsn>
```

6.4.2 DISC[ONNECT]

このコマンドを使用すると、既存の接続を切断できます。

6.4.3 SPOOL

このコマンドを使用すると、オペレーティング・システム・ファイルに問合せ結果を格納できます。

使用方法

- SPOOL を単独で使用すると、スプーラの現在の状態を取得できます。

  ```
  SPOOL
  ```

  スプーラの現在の状態が表示されます。

- SPOOL にファイル名を指定すると、スプール・ファイルを設定できます。

  ```
  SPOOL <file_name>
  ```

  スプール・ファイルが設定されます。

- SPOOL を使用して、スプーラをクローズします。

  ```
  SPOOL OFF
  ```

  スプーラがクローズされます。

6.4.4 START

このコマンドを使用すると、サポートされる SQL ファイルを起動または実行できます。オプションで、@ コマンドも使用できます。また、MSQL は、@@ コマンドをサポートして、相対パスで SQL ファイルを実行できます。

SQL*Plus とは異なり、SQL ファイルの exit コマンドは、特定のファイルのコンテキスト内でのみ有効です。

たとえば、exit コマンドが含まれる **t2.sql** ファイルを **t1.sql** ファイルに含める場合、**t2.sql** ファイルの処理が完了するまで、MSQL はデータベース接続を終了または切断しません。

6.4.5 DESCRIBE

このコマンドを使用すると、表やビューの定義を表示できます。

6.4.6 SET

このコマンドを使用すると、システム変数を設定して、現行のセッションの MSQL 環境を変更できます。MSQL がサポートするコマンドは、次のとおりです。

6.4.6.1 SET echo ON/OFF

このコマンドを使用すると、必要に応じてエコーをオンまたはオフに設定できます。

6.4.6.2 SET sqlp[rompt] <string>

このコマンドを使用すると、SQL プロンプトを与えられた文字列に設定できます。

6.4.6.3 SET hea[ding] ON/OFF

このコマンドを使用すると、必要に応じて見出しをオンまたはオフに設定できます。

6.4.6.4 SET auto[commit] ON/OFF

このコマンドを使用すると、必要に応じて自動コミット機能をオンまたはオフに設定できます。

6.4.6.5 SET TRANSACTION ISOLATION LEVEL [READ COMMITTED] [READ WRITE]

このコマンドを使用すると、現行のセッションの分離レベルを確立できます。SET TRANSACTION 文によって実行される操作は、現行のセッションのみに影響し、その他のユーザーやセッションには影響しません。セッションは、COMMIT または ROLLBACK 文を発行すると終了します。

6.4.6.6 SET long width

このコマンドを使用すると、LONG データの表示長を設定できます。

6.4.6.7 SET depth val

このコマンドを使用すると、指定した @filename.sql の再帰的な深度の値を設定できます。

デフォルト値

20

6.4.6.8 SET SCAN {ON/OFF}

このコマンドは、置換変数およびパラメータの有無を確認するスキャンを制御します。OFF を指定すると、置換変数およびパラメータの処理が抑制され、ON を指定すると通常の処理を実行できます。

構文は次のとおりです。

```
SET SCAN {ON/OFF}
```

6.4.7 WHENEVER

このコマンドは、Java 開発では未対応のため、WHENEVER OSERROR コマンド以外の場合は、部分的にサポートされます。

ただし、MSQL は終了、コミット、ロールバック、継続などの機能に対して WHENEVER SQLERROR コマンドをサポートします。

6.4.8 SQL*Plus 形式のコメント

MSQL は、REM コメントおよび -- コメントをサポートします。

6.4.9 SQL*Plus 形式での実行の繰返し

/ またはセミコロンを使用すると、前回実行された SQL 文を繰り返し実行できます。適切な構文の SQL コマンドのみを繰り返すことができます。

6.4.10 DEFINE/UNDEFINE

このコマンドを使用すると、パラメータとそれに対応する値の定義および定義の削除ができます。ユーザーは変数名を入力して、その値を決定する必要があります。MSQL は、DEFINE コマンドの演算子をサポートします。

たとえば、DEF NAME = 1 + 2 は有効で、MSQL で使用できます。

使用方法

```
DEF[INE] var_name = var_value

UNDEF[INE] var_name
```

6.4.11 パラメータ

MSQL は、コマンドライン・パラメータおよび定義済パラメータのパラメータ置換をサポートします。

6.4.12 CLEAR または CLS

このコマンドを使用すると、特定のオプションの現在の値および設定をリセットまたは消去できます。

使用方法

CLS - 画面を消去します。

CLE[AR] - 画面を消去します。

CLE[EAR] - 内部 SQL バッファを消去します。

6.4.13 PAUSE

このコマンドを使用すると、空白行を 1 行とそれに続いてテキストを含む行を 1 行、または空白行を 2 行表示できます。

6.4.14 LIST

このコマンドを使用すると、SQL バッファの現在の SQL 文を表示できます。

6.4.15 HOST

このコマンドは、WIN32 でサポートされ、システム固有のコマンドを実行します。このコマンドを使用すると、Windows NT 環境から **cmd.exe** を起動し、**cmd.exe** が完了するまで待機します。

6.4.16 CLOCK

このコマンドを使用すると、パフォーマンス・クロックを起動または停止できます。クロックを停止した場合にのみ、経過時間が表示されます。

例

次のコマンドは、経過時間をミリ秒単位で出力します。

```
CLOCK start
CLOCK stop
```

6.4.17 REPEAT UNTIL

このコマンドを使用すると、指定した式の条件が満たされるまで、コマンドのブロックを繰り返し実行できます。

例

```
clock start
     define N = 0
     repeat
        create table test%N% (c1 number, c2 varchar2 (256));
        insert into test%N% values ('%N%', 'abcde');
        define N = N + 1
     until N = 10

     define N = 0
     repeat
        select c1, c2 from test%N%;
```

```
   define N = N + 1
until N = 10
clock stop
```

MSQL は、ネストされた REPEAT UNTIL ループもサポートします。

注意： Define コマンドは、「++」、「--」などの単項演算子をサポートしません。

6.4.18 DIR または LS

このコマンドを使用すると、すべての表名と所有者名を出力できます。「DIR TABLE_NAME」などのオブジェクト名と使用すると、コマンド「SELECT * FROM TABLE_NAME」と同じ処理が実行されます。

例

DIR

DIR <TABLE_NAME>/<VIEW_NAME> - <TABLE_NAME> または <VIEW_NAME> の内容を表示します。

DIR VIEW - すべてのビュー名と所有者を表示します。

DIR SEQUENCE - すべての順序名と所有者を表示します。

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 98/NT/2000/XP 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。同様の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。日付書式を表 A-1 に示します。

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
```

書式は次のいずれかです。

- 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	ICELANDIC
BRAZILIAN PORTUGUESE	ITALIAN
BULGARIAN	JAPANESE
CANADIAN FRENCH	LATIN AMERICAN SPANISH
CATALAN	LITHUANIAN
CROATIAN	MALAY
CZECH	MEXICAN SPANISH
DANISH	NORWEGIAN
DUTCH	POLISH
EGYPTIAN	PORTUGUESE
ENGLISH	ROMANIAN
ESTONIAN	RUSSIAN
FINNISH	SLOVAK
FRENCH	SLOVENIAN
GERMAN	SPANISH
GREEK	SWEDISH
HEBREW	TURKISH
HUNGARIAN	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、付録 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.2.14 Olite_Sql_Trace

SQL 文のテキスト、コンパイル時間、実行計画およびバインド値を生成します。

次に例を示します。

```
OLITE_SQL_TRACE= yes
```

SQL トレースの出力は、データベース・プロセスの現行の作業ディレクトリでトレース・ファイル **oldb_trc.txt** にダンプされます。Windows または Windows NT のデータベース・サービス、あるいは Linux プラットフォームの Oracle Lite デーモンでは、現行の作業ディレクトリは、データベース・サービスまたはデーモンの起動時に、**wdir** パラメータで指定されます。埋込み接続を使用してデータベースに接続するアプリケーションには、作業ディレクトリが含まれます。この作業ディレクトリが、アプリケーションの作業ディレクトリです。トレース機能を実装するには、データベース・プロセスに、現行の作業ディレクトリにトレース・ファイルを作成する権限が必要です。トレース出力は、常にトレース・ファイルに含まれます。トレース・ファイルが存在しない場合は、自動的に作成されます。

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。

注意： 次に示す表で、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。このビューのパラメータを表 B-1 に示します。

表 B-1 ALL_COL_COMMENTS のパラメータ

列	データ型	NULL ALLOWED	説明
OWNER	VARCHAR2(128)	×	表の所有者。
TABLE_NAME	VARCHAR2(128)	×	オブジェクトの名前。
COLUMN_NAME	VARCHAR2(128)	×	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。このビューのパラメータを表 B-2 に示します。

表 B-2 ALL_CONSTRAINTS のパラメータ

列	データ型	NULL ALLOWED	説明
OWNER	VARCHAR2(128)	×	制約定義の所有者。
CONSTRAINT_NAME	VARCHAR2(128)	×	制約定義に対応付けられた名前。
CONSTRAINT_TYPE	VARCHAR2(128)	×	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)	×	制約定義を持つ表の名前。

表 B-2 ALL_CONSTRAINTS のパラメータ（続き）

列	データ型	NULL ALLOWED	説明
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。
DELETE_RULE	VARCHAR2(128)	○	参照制約の削除規則 : 「NO ACTION」
STATUS	VARCHAR2(20)	×	「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。このビューのパラメータを表 B-3 に示します。

表 B-3 ALL_CONS_COLUMNS のパラメータ

列	データ型	NULL ALLOWED	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。このビューのパラメータを表 B-4 に示します。

表 B-4 ALL_INDEXES のパラメータ

列	データ型	NULL ALLOWED	説明
OWNER	VARCHAR2(128)	×	INDEX 定義の所有者。
INDEX_NAME	VARCHAR2(128)	×	INDEX 定義に対応付けられた名前。
TABLE_OWNER	VARCHAR2(128)	×	INDEX が定義されている表の所有者。
TABLE_NAME	VARCHAR2(128)	×	INDEX 定義を持つ表の名前。
TABLE_TYPE	VARCHAR2(10)	○	オブジェクトの型。
UNIQUENESS	VARCHAR2(128)	×	「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。このビューのパラメータを表 B-5 に示します。

表 B-5 ALL_IND_COLUMNS のパラメータ

列	データ型	NULL ALLOWED	説明
INDEX_OWNER	VARCHAR2(128)	×	INDEX 定義の所有者。
INDEX_NAME	VARCHAR2(128)	×	INDEX 定義に対応付けられた名前。
TABLE_OWNER	VARCHAR2(128)	×	INDEX が定義されている表の所有者。
TABLE_NAME	VARCHAR2(128)	×	INDEX 定義を持つ表の名前。
COLUMN_NAME	VARCHAR2(128)	×	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)	×	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。このビューのパラメータを表 B-6 に示します。

表 B-6 ALL_OBJECTS のパラメータ

列	データ型	NULL ALLOWED	説明
OWNER	VARCHAR2(128)	×	OBJECTS 定義の所有者。
OBJECT_NAME	VARCHAR2(128)	×	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
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_PARTITIONS

このビューは、すべてのマップ表パーティション、パーティションの削除またはマージに使用するパーティション ID およびパーティション間のクライアントの配分を表示します。このビューのパラメータを表 B-7 に示します。

表 B-7 ALL_VIEWS のパラメータ

列	データ型	NULL ALLOWED	説明
PUB_ITEM	VARCHAR2(30)	×	パブリケーション・アイテム名。
PARTITION_ID	NUMBER	×	パーティション ID。
CLID$$CS	VARCHAR2(30)	×	クライアント ID。

注意 : このパーティション化形式は、Oracle サーバーによって提供されているパーティション機能とは関係なく、Oracle9*i* Lite 専用です。

B.1.8 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。このビューのパラメータを表 B-8 に示します。

表 B-8 ALL_SEQUENCES のパラメータ

列	データ型	NULL ALLOWED	説明
SEQUENCE_OWNER	VARCHAR2(128)	×	SEQUENCES 定義の所有者。
SEQUENCE_NAME	VARCHAR2(128)	×	SEQUENCES 定義に対応付けられた名前。
MIN_VALUE	NUMBER(10)	×	順序の最小値。
MAX_VALUE	NUMBER(10)	×	順序の最大値。
INCREMENT_BY	NUMBER(10)	×	順序の増分値。

B.1.9 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。このビューのパラメータを表 B-9 に示します。

表 B-9 ALL_SYNONYMS のパラメータ

列	データ型	NULL ALLOWED	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。

B.1.10 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。このビューのパラメータを表 B-10 に示します。

表 B-10 ALL_TABLES パラメータ

列	データ型	NULL ALLOWED	説明
OWNER	VARCHAR2(128)	×	表の所有者のユーザー名。
TABLE_NAME	VARCHAR2(128)	×	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。

表 B-10 ALL_TABLES パラメータ（続き）

列	データ型	NULL ALLOWED	説明
BLOCKS*	NUMBER(10)	○	表に割り当てられているデータ・ブロック数。
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.1.11 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。このビューのパラメータを表 B-11 に示します。

表 B-11 ALL_TAB_COLUMNS のパラメータ

列	データ型	NULL ALLOWED	説明
OWNER	VARCHAR2(128)	×	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)	×	表、ビューまたはクラスタの名前。
COLUMN_NAME	VARCHAR2(128)	×	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）。

表 B-11 ALL_TAB_COLUMNS のパラメータ（続き）

列	データ型	NULL ALLOWED	説明
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。
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)	×	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.1.12 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。このビューのパラメータを表 B-12 に示します。

表 B-12 ALL_TAB_COMMENTS のパラメータ

列	データ型	NULL ALLOWED	説明
OWNER	VARCHAR2(128)	×	TAB_COMMENTS 定義の所有者。
TABLE_NAME	VARCHAR2(128)	×	TAB_COMMENTS 定義を持つ表の名前。
TABLE_TYPE	VARCHAR2(128)	×	オブジェクトの型。
COMMENTS	VARCHAR2(4096)	×	コメントのテキスト。

B.1.13 ALL_USERS

このビューは、接続済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。このビューのパラメータを表 B-13 に示します。

表 B-13 ALL_USERS のパラメータ

列	データ型	NULL ALLOWED	説明
USERNAME	VARCHAR2(30)	×	ユーザーの名前。
USER_ID*	NUMBER	×	ユーザーの ID 番号。
CREATED	DATE	×	ユーザー作成日付。

B.1.14 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。このビューのパラメータを表 B-14 に示します。

表 B-14 ALL_VIEWS のパラメータ

列	データ型	NULL ALLOWED	説明
OWNER	VARCHAR2(128)	×	ビューの所有者のユーザー名。
VIEW_NAME	VARCHAR2(128)	×	ビューの名前。
TEXT_LENGTH	NUMBER(10)	×	ビューのテキストの長さ。
TEXT	VARCHAR2(1000)	×	ビューのテキスト。

B.1.15 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。このビューのパラメータを表 B-15 に示します。

表 B-15 CAT のパラメータ

列	データ型	NULL ALLOWED	説明
TABLE_NAME	VARCHAR2(128)	×	オブジェクトの名前。
TABLE_TYPE	VARCHAR2(128)	×	オブジェクトの型 : TABLE または VIEW

B.1.16 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。このビューのパラメータを表 B-16 に示します。

表 B-16 COLUMN_PRIVILEGES のパラメータ

列	データ型	NULL ALLOWED	説明
OWNER	VARCHAR2(128)	×	オブジェクトの所有者のユーザー名。
TABLE_NAME	VARCHAR2(128)	×	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)	×	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.1.17 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。このビューのパラメータを表 B-17 に示します。

表 B-17 DATABASE_PARAMETERS のパラメータ

列	データ型	NULL ALLOWED	説明
PARAMETER	VARCHAR2(30)	×	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.1.18 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。このビューのパラメータを表 B-18 に示します。

表 B-18 DUAL のパラメータ

列	データ型	NULL ALLOWED	説明
DUMMY	VARCHAR2(1)	×	常に「X」。

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。このビューのパラメータを表 B-19 に示します。

表 B-19 TABLE_PRIVILEGES のパラメータ

列	データ型	NULL ALLOWED	説明
OWNER	VARCHAR2(128)	×	オブジェクトの所有者のユーザー名。
TABLE_NAME	VARCHAR2(128)	×	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。

表 B-19 TABLE_PRIVILEGES のパラメータ（続き）

列	データ型	NULL ALLOWED	説明
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)	×	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。このビューのパラメータを表 B-20 に示します。

表 B-20 USER_OBJECTS のパラメータ

列	データ型	NULL ALLOWED	説明
OWNER	VARCHAR2(128)	×	オブジェクトの所有者のユーザー名。
OBJECT_NAME	VARCHAR2(128)	×	オブジェクトの名前。
OBJECT_ID	NUMBER(10)	×	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。

表 B-20 USER_OBJECTS のパラメータ（続き）

列	データ型	NULL ALLOWED	説明
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。表 C-1 のユーティリティの名前はアルファベット順です。

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 for 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 for 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。**polite.ini** を使用して照合順序を有効にする方法の詳細は、A.2.7 項を参照してください。

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、表 C-2 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、結果は表 C-3 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。

ENCRYPDB は、128 ビット DES 準拠の暗号化体系である CAST5 暗号化を使用しています。

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かった場合、ENCRYPDB が入力を要求します。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、結果は表 C-4 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。同じデータベースの別のユーザーは、パスワードを紛失したユーザー用に新規のユーザー名でコピーを作成できます。パスワードを紛失した場合、これ以外の方法では、データベースのリカバリを行うことができません。
- 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、第三者が Oracle Lite データベースを削除することを防ぐことはできません。つまり、removedb と rmdb は、パスワードを確認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.4.1 暗号化された Oracle Lite データベースとの同期

暗号化された Oracle Lite データベースと同期するには、次の手順を実行する必要があります。

1. Mobile サーバー・リポジトリからユーザー・パスワードを取得します。
2. パスワードを大文字に変換します。たとえば、「manager」は「MANAGER」に変換します。
3. Mobile Sync（**msync.exe**）を起動して、同期を実行します。ユーザー名、パスワードおよび Mobile サーバーの URL を入力します。「適用」を選択後、「同期」を選択します。これにより、暗号化されていない Oracle Lite データベースが作成されます。
4. MANAGER などの変換した大文字のパスワードを使用して ENCRYPDB ユーティリティを使用し、Oracle Lite データベースを暗号化します。
5. 同期を続行します。

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 を使用している場合の詳細は、『Oracle9*i* Lite for Windows NT/2000/XP 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。

このユーティリティは、Java ストアド・プロシージャの移行をサポートしていません。

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。

Microsoft 社では、Windows 98/NT/2000/XP 内の ODBC.INI ファイルおよび関連するレジストリ・エントリを管理するためのツールである ODBC Administrator を提供しています。ODBC.INI ファイルおよび Windows レジストリには、ODBC Administrator で取得した 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 は、DSN 以外にも表 C-5 に示す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、第 2 章「Oracle Lite RDBMS」の 2.6.3 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。
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。

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 読込み専用メディア（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・ファイルを読込み専用に設定すると、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 ユーティリティで、表 C-6 に示す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。

表 C-6 ODBINFO のパラメータ（続き）

パラメータ	説明
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 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（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 の読込み専用データベース（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場合に指定します。

-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表 C-7 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証します。実行時にエラーが検出されなかった場合は 0、エラーが検出された場合は 1 に戻します。データベース構造が破損している場合は、検出されたエラーが表示され、そのエラーはユーザーが指定したファイルに格納されます。このツールは次の項目を確認します。

- 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、フラグの整合性が確認されます。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確認します。
- 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確認します。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。
- グループー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。
- 索引ーすべてのページが 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確認します。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確認されます。

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、表 D-1 に示すデータベース・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 API について説明します。この付録の各項では、異なるトピックについて説明します。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、Oracle Lite データベース内の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。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。ここで説明する API コールは、ユーティリティの機能をカスタマイズするために使用できます。

E.2 Oracle Lite ロード・ユーティリティ API

Oracle Lite ロード・ユーティリティ API の説明は、次のとおりです。

- 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、データベースとの接続を切断します。

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);
```

`olConnect` の引数を表 E-1 に示します。

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);
```

olDisconnect の引数を表 E-2 に示します。

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 );
```

olTruncate の引数を表 E-3 に示します。

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);
```

olSet の引数を表 E-4 に示します。

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.3.2 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);
```

olLoad の引数を表 E-5 に示します。

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,...) ここで、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 は、ファイル・ヘッダー内で表の説明の検索を試みます。

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);
```

olDump の引数を表 E-6 に示します。

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** に存在します。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.3 ファイル形式

Oracle Lite ロード・ユーティリティは、FIXEDASCII、BINARY および CSV という 3 つのファイル形式をサポートします。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）。

```
$$OL_BH$$ [begins header]
VERSION=xx.xx.xx.xx   [version number]
TABLE=T1(C1, C2, ...)... [table name with list of column names dumped]
FILEFORMAT=FIXEDASCII
SEPARATOR=,
[any other parameters in the parameter list can be listed here]
$$OL_EH$$ [ends header]
```

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 パラメータ

表 E-7 に示すヘッダー・ファイル・パラメータは、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）。読込み専用データベース（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。

表 E-7 パラメータ（続き）

パラメータ	説明
BITARRAY	バイナリ形式で、NULL の書込みおよび読込み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	NULL の読込みまたは書込みを試行した場合に olLoad および olDump がエラーを返すように、TRUE（デフォルト）に設定します。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。
DATEFORMAT	日付列およびタイムスタンプ列を、FIXED ASCII または CSV 形式で、ファイルに対して読込みおよび書込みするための文字列です。「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.3.3 データ形式

データ形式は、カンマで区切られた値（CSV）、FixedASCII またはバイナリです。次の場合が該当します。

E.3.3.1 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.3.3.2 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。各データ型の説明を、表 E-8 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。
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。

表 E-8 データ型（続き）

データ型	説明
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 -> ‘-00000000000000000012.34‘ 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。
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 -> ‘-0000000000017900000000‘
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

表 E-8 データ型（続き）

データ型	説明
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 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-Go の 3 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 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。

Oracle データベース（Oracle Database）

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。

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 データベースで構成される 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。Mobile サーバーと Oracle Lite データベースが含まれます。オフライン・モードに切り替わると、Web-to-Go は、ユーザーの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。オンラインに戻ると、Web-to-Go はデータの変更を 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。一意キー制約を満たすには、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、主キー制約とは異なり、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」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」も参照。

外部キー（Foreign Key）

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

結合（Join）

2 つの異なる表またはビューに存在するキー（主キーと外部キーの両方）の間に確立された関係。結合は、リレーショナル・データベース内の重複したデータを排除するために正規化された表のリンクに使用します。結合リンクの一般的なものとしては、1 つの表の主キーを別の表の外部キーにリンクして、マスター・ディテール・リレーション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、一意キー制約や主キー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・リレーション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。

主キー制約を満たすには、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）

連続した番号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序番号を生成できます。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。
「Web-to-Go」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 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 Web-to-Go が使用するプロセス。Web-to-Go は、ユーザーがオフライン・モードに切り替わったときに、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。オンラインに戻ると、Web-to-Go はデータの変更を 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。「Oracle データベース」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・アイテム（Publication Item）

パブリケーション・アイテムは、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・アイテムは、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・アイテムは、Mobile サーバー 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いいます。

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」および「外部キー」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わっ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可になっても、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

と

は

ひ

ふ

ま

り

れ