Oracle® Warehouse Builder

ユーザーズ・ガイド

10*g* リリース 1（10.1）

部品番号 : B13517-02

2007 年 11 月

ORACLE®

Oracle Warehouse Builder ユーザーズ・ガイド , 10*g* リリース 1（10.1）

部品番号 : B13517-02

原本名 : Oracle Warehouse Builder User's Guide 10*g* Release 1 (10.1)

原本部品番号 : B12146-02

原本協力者 : Shirinne Alison, Kavita Nayar, Michelle Bird, Julia Stein

目次

4 データ定義のインポート

5 データ・オブジェクトの構成

7 マッピングのデバッグ

8 マッピング演算子の使用方法

9 変換の使用方法

10 プロセス・フローの設計

11 ETL オブジェクトの構成

14 配布と実行の監査

第 IV 部　メタデータの管理

15　Metadata Loader（MDL）を使用したインポートおよびエクスポート

16 メタデータ変更管理

17 メタデータの参照およびレポート

18 Warehouse Builder Browser の管理

第 V 部 Warehouse Builder の統合と拡張

19 Warehouse Builder 機能の拡張

20 データの品質 : Name and Address のクレンジング

21 Warehouse Builder での SAP R/3 データの使用

22 その他の BI 製品と Warehouse Builder のメタデータの統合

D Warehouse Builder Public View

E パブリック・オブジェクト

用語集

索引

はじめに

この章では、次のトピックについて説明します。

目的

Oracle Warehouse Builder は、データの移動や変換、ビジネス・インテリジェンス・システムの開発および実装、メタデータの管理、Oracle データベースおよびメタデータの作成と管理を行う専門家のための包括的なツールセットです。このガイドでは、Warehouse Builder を使用して次の操作方法について説明します。

- データ・ウェアハウスの定義の作成
- データ・ウェアハウスの物理インスタンスの定義の構成
- 定義やその構成のセットの検証
- スクリプトのセットの生成による、データ・ウェアハウス・インスタンスの作成および移入
- データ変換スクリプトの生成
- データ・ウェアハウス・インスタンスの配布および初期ロード
- 生成済スクリプトを使用した条件付きのリフレッシュによる、物理インスタンスのメンテナンス
- その他のビジネス・インテリジェンス製品と Warehouse Builder のメタデータの統合
- データ・ウェアハウス分析に使用する Oracle Discoverer エンド・ユーザー・レイヤーと OLAP カタログの移入

対象読者

このガイドは、次のようなデータ・ウェアハウス技術者を対象としています。

- ビジネス・インテリジェンス・アプリケーション開発者
- ウェアハウス設計者および開発者、特に SQL と PL/SQL の開発者
- データ・アナリストや、抽出、変換、ロードのルーチンを開発するユーザー
- データ・ウェアハウスを基盤とした大規模な製品の開発者
- ウェアハウス管理者
- システム管理者
- その他の MIS 専門家

このガイドの情報を利用するには、リレーショナル・データベース管理システムやデータ・ウェアハウスの設計に関する概念を理解する必要があります。データ・ウェアハウスの詳細は、『Oracle データ・ウェアハウス・ガイド』を参照してください。また、Oracle Database、SQL*Plus、SQL*Loader、Oracle Enterprise Manager および Oracle Workflow など、Oracle リレーショナル・データベース・ソフトウェア製品の知識も必要です。

このガイドの構成

Oracle Warehouse Builder ユーザーズ・ガイドは、次の章と付録で構成されています。

- 第 1 章「Oracle Warehouse Builder の概要」では、製品全体の構成および機能と、Warehouse Builder を使用する利点を説明します。また、ログオン方法を説明し、メイン・コンソールに表示されるコンポーネントおよび設定可能な作業環境とオプションを紹介します。
- 第 2 章「Warehouse Builder スタート・ガイド」では、Warehouse Builder の機能を使用してプロジェクトを初めて作成する方法を説明します。

第 I 部「ソース・オブジェクトとターゲット・オブジェクトの定義」では、データを取得してロードするオブジェクトの、メタデータ定義の作成および構成を説明します。この部には、次の章があります。

- 第 3 章「Oracle データ・オブジェクトの定義」では、Warehouse Builder でリレーショナルおよびディメンションのターゲット・オブジェクトの定義を作成する方法を説明します。リレーショナル・オブジェクトには、表、ビュー、マテリアライズド・ビュー、順序、外部表などがあります。ディメンション・オブジェクトには、ディメンションやキューブなどがあります。
- 第 4 章「データ定義のインポート」では、Warehouse Builder のインポート・ウィザードによるデータソースの定義のインポート方法を説明します。この章では、Oracle や Oracle 以外のデータベース・システムや、フラット・ファイル・ソースからの定義のインポート方法を説明します。
- 第 5 章「データ・オブジェクトの構成」では、論理モデルで定義したターゲット・オブジェクトとソース・オブジェクトに、物理プロパティを割り当てる方法を説明します。この章ではまず、Warehouse Builder で定義された表、マテリアライズド・ビュー、ディメンション、キューブなどの設計オブジェクトに対して、索引およびパーティションを作成する方法を説明します。次に、Warehouse Builder のオブジェクト定義に物理プロパティを割り当てる方法を説明します。

第 II 部「ETL オブジェクトの設計」では、データ移動に使用する、抽出、変換、ロードの各プロセスの設計および構成を説明します。ETL オブジェクトには、マッピング、変換、プロセス・フローなどがあります。この部には、次の章があります。

- 第 6 章「マッピングの設計」では、ソースからデータを抽出して変換し、ターゲットにロードするために、マッピングを作成および設計する方法を説明します。
- 第 7 章「マッピングのデバッグ」では、マッピング・エディタ内のデータ・フローに対する Warehouse Builder のデバッグ機能を説明します。
- 第 8 章「マッピング演算子の使用方法」では、データ変換を行う際に、マッピングで複数の演算子を使用する方法を詳しく説明します。この章では、ソースからターゲットにデータを渡す際に、そのデータを操作する式を Expression Builder で作成する方法も説明します。

- 第 9 章「変換の使用方法」では、マッピングで変換をインポート、作成、使用する方法を詳しく説明します。
- 第 10 章「プロセス・フローの設計」では、（電子メール、FTP コマンド、オペレーティング・システムの実行可能ファイルなど）Warehouse Builder 内外のアクティビティとマッピングを相関させるプロセス・フローを作成および設計する方法を説明します。
- 第 11 章「ETL オブジェクトの構成」では、マッピングとプロセス・フローを構成して、特定のデータベース・インスタンスに配布する方法を説明します。また、パフォーマンス向上のためにパーティション交換ロードを使用する利点と、その使用方法を説明します。

第 III 部「配布および実行」では、定義したデータの配布、またデータ移動のために設計したプロセスの実行を説明します。この部には、次の章があります。

- 第 12 章「オブジェクトの検証」では、定義したオブジェクトの整合性を検証する方法と、配布時に起こりうるあらゆる問題またはエラーを検出する方法を説明します。また、生成したスクリプト（DDL や PL/SQL スクリプトなど）の表示方法も説明します。
- 第 13 章「ターゲット・システムの配布」では、ターゲット・データ・システムの物理インスタンスを配布する方法を説明します。ターゲット・システムのアップグレード方法と、再利用できるように配布スクリプトを保存する方法を説明します。また、ターゲット・データ・システムに配布済のマッピングとプロセス・フローのスクリプトを実行する方法も説明します。
- 第 14 章「配布と実行の監査」では、Warehouse Builder の Web ベースの Runtime Audit Browser を使用して、配布と実行に関する情報を表示する方法を説明します。

第 IV 部「メタデータの管理」では、インポートやエクスポート、バックアップとバージョニングに使用するスナップショット、メタデータの参照とレポートなど、メタデータの管理タスクを説明します。この部には、次の章があります。

- 第 15 章「Metadata Loader（MDL）を使用したインポートおよびエクスポート」では、バックアップとアップグレード用にメタデータのインポートとエクスポートを行うユーティリティを説明します。
- 第 16 章「メタデータ変更管理」では、メタデータ・スナップショットを使用して、メタデータの変更を管理する方法を説明します。また、スナップショットを使用して、バックアップ、バージョニング、比較、メタデータのリストアを行う方法を説明します。
- 第 17 章「メタデータの参照およびレポート」では、Warehouse Builder の Web ベースの Design Browser を使用して、メタデータを参照してメタデータ・レポートを表示する方法を説明します。
- 第 18 章「Warehouse Builder Browser の管理」では、Warehouse Builder のポータル・ベースの Design Browser を構成する方法と、カスタム・メタデータ・レポートの作成方法を説明します。

- 第 19 章「Warehouse Builder 機能の拡張」では、Warehouse Builder Security Registration パッケージ（セントラル・スキーマの所有者が Warehouse Builder のユーザーになる見込みあるすべてのユーザーをリポジトリに登録する方法を含みます）、Warehouse Builder Metadata Loader、Warehouse Builder Runtime Platform のプラグイン・インタフェースを説明します。

第 V 部「Warehouse Builder の統合と拡張」では、Oracle Warehouse Builder と他のビジネス・インテリジェンス製品を統合する機能を説明します。この部には、次の章があります。

- 第 20 章「データの品質 : Name and Address のクレンジング」では、Warehouse Builder Name and Adress を購入し、Name and Adress 演算子を使用してカスタマ・データをクレンジングする場合に、解析表をメンテナンスする方法を説明します。
- 第 21 章「Warehouse Builder での SAP R/3 データの使用」では、Warehouse Builder と SAP ソース・システムを統合する方法を説明します。Warehouse Builder のリポジトリに SAP データの定義をインポートし、マッピングに SAP 定義を使用し、ABAP または PL/SQL コードを生成してターゲットに配布できます。また、SAP データをソースシステムから抽出し、ターゲット・システムにロードする方法も説明します。
- 第 22 章「その他の BI 製品と Warehouse Builder のメタデータの統合」では、Warehouse Builder のメタデータを他のビジネス・インテリジェンス製品のメタデータと交換する際に使用するユーティリティを説明します。

第 VI 部「付録」では、それまでの部で参照されたものを扱います。この部には、次の付録があります。

- 付録 A「キーボードのショートカット」では、Warehouse Builder コマンド用のキーボード・ショートカットのリストを紹介します。
- 付録 B「予約語」では、Warehouse Builder 用の予約語のリストを紹介します。
- 付録 C「XML ツールキットの使用」では、ターゲット・スキーマでの XML 文書からのデータの取得および保存方法を説明します。
- 付録 D「Warehouse Builder Public View」では、SQL を使用してアクセスできる Warehouse Builder のリストを紹介します。このビューを使用すると、Warehouse Builder のレポート作成機能を拡張できます。
- 付録 E「パブリック・オブジェクト」では、ナビゲーション・ツリーのファーストクラス、セカンド・クラス、サード・クラス、およびフォース・クラスのオブジェクトを紹介します。ここでは、ユーザー定義のプロパティとメタデータ・スナップショットの動作を詳しく説明します。

10*g* リリース 1（10.1）の新機能

マッピング・エディタの拡張機能 : マッピング・デバッガ

このリリースの Warehouse Builder では、マッピング・エディタ内からのマッピング用に、拡張的なデバッグ機能が提供されます。マッピング・デバッガを使用して、マッピング内の論理設計エラーを見つけます。この新機能では、ブレーク・ポイントやウォッチの設定、テスト・データの対話的変更などの包括的なデバッグ機能を使用して、マッピングのデータ・フローを順を追って確認できます。

複数ターゲットのサポート強化 : 相関コミット

このリリースでは、複数のターゲットを含むマッピングに使用する新しいコミット方法が導入されます。以前のリリースの Warehouse Builder では、独立したコミットが実行されました。つまり、Warehouse Builder では、他のターゲットに関係なく、各ターゲットがコミットおよびロールバックされていました。このオプションに加え、このリリースの Warehouse Builder では相関コミットも実行されます。つまり、すべてのターゲットをひとまとまりと見なして、ターゲット全体で一様にデータをコミットまたはロールバックします。影響を受けるターゲットすべてに対して、ソースのあらゆる行が均一な影響を及ぼすようにすることが重要な場合に、相関コミットを使用します。

ダイレクト・パーティション交換ロード

以前のリリースの Warehouse Builder では、ソース・データの追加処理が必要なマッピング用に一時表がデフォルトで作成され、その後パーティションが交換されていました。このような状態は、マッピングにリモート・ソースまたは結合した複数のソースが含まれた場合に発生しました。このリリースでは、一時表を作成する必要はなく、ソースを直接ターゲットに交換できます。マッピングでダイレクト・パーティション交換ロードを使用して、前に実行されたマッピングでロードしたファクト表を即座にパブリッシュします。

データの品質に関する機能

- **複数の Name and Address ソフトウェア・プロバイダ :** このリリースからは、動作保証された複数の Name and Address ソフトウェア・プロバイダと Warehouse Builder が連携できるようになります。サードパーティ・ベンダーに許諾された Name and Address ソフトウェアを Warehouse Builder で使用できます。これによって、プロジェクトに最適な Name and Address プロバイダを選択できるようになります。
- **Name and Address 演算子ウィザード :** 以前のリリースでは、マッピング・キャンバスと演算子の構成プロパティ・シートを使用して、Name and Address 演算子を定義していました。このリリースの Warehouse Builder では、使い易さを考慮して、ウィザードおよび演算子エディタで Name and Address 演算子を作成および編集できるようになりました。

- **Match-Merge 演算子 :** Oracle Pure Integrate で使用していたデータの品質に関する機能が、Warehouse Builder に組み込まれました。マッピング・エディタで Match-Merge 演算子を使用して、レコードの一致とマージに関するビジネス・ルールを定義できます。Match-Merge 演算子を Name and Address 演算子とともに使用すると、Name and Address データで一意の世帯を識別する世帯検索プロセスがサポートされます。

メタデータ変更管理

以前のリリースでは、OMB Plus スクリプト・ユーティリティを使用してメタデータ変更管理を実行していました。このリリースからは、Warehouse Builder Design Client ユーザー・インタフェースで同様の機能を実行できます。メタデータ変更管理によって、メタデータ・オブジェクトのスナップショットを取得したり、バックアップや履歴管理にスナップショットを使用したりできます。ナビゲーション・ツリーの任意のオブジェクトでスナップショットがサポートされます。スナップショットには、オブジェクト自身の情報（表やモジュールなど）や、オブジェクト内のオブジェクト（モジュール内の表など）の情報が格納されます。

Oracle Warehouse Builder 機能の拡張

- **セキュリティ :** このリリースの Warehouse Builder では、セキュリティ要件に応じて採用するリポジトリ・セキュリティと監査のオプションが拡張されます。拡張されたセキュリティ・オプションは次のとおりです。

  プロアクティブ・セキュリティ : Warehouse Builder では、Warehouse Builder Design Repository に入っているセキュリティ PL/SQL 実装パッケージをカスタマイズしてプラグインすることにより、ユーザー組織の規定するセキュリティ・ルールに従って、ユーザーへのアクセス制御を定めることができます。

  リアクティブ・セキュリティ : Warehouse Builder では、メタデータ履歴に基づいて監査情報を追跡し、このような監査情報によってセキュリティ・ポリシーを決定できます。

  データ管理 : Warehouse Builder では、技術管理者でなく、個人または数名のグループが、メタデータの一部を所有できます。そのため、メタデータのセキュリティ管理において、メタデータの所有が重要な役割を果たすようになりました。

- **RAC のサポート :** リリース 2（9.2）では、RAC 機能のサポートが拡張されました。このリリースの Warehouse Builder では、実行時にネット・サービス名を使用できるようになります。これによって、ランタイム環境を再構成せずに、クラスタ内のノードの保守計画を立てることができます。また、Warehouse Builder では、ランタイム・サービスの可用性も拡張されます。たとえば、サービス・インスタンスまたは関連のノードが失敗するか、サービスが停止すると、別のノードにあるランタイム・サービス・インスタンスをかわりに実行できます。Warehouse Builder Design Repository も RAC クラスタで使用できますが、このリリースでは RAC のフェイル・オーバー機能は利用できません。

フラット・ファイル・サポートの拡張

- **ZONED データ型のサポート :** このリリースの Warehouse Builder では、ゾーン 10 進データを含む固定形式のデータ・ファイルをロードできるようになります。フラット・ファイル・サンプル・ウィザードで、インポートするフラット・ファイルに ZONED データ型を指定します。ゾーン化データの形式は、10 進数の文字列（1 バイトに 1 桁）で、最終バイトに記号が含まれています（COBOL では、SIGN TRAILING フィールドになります）。このフィールドの長さは、指定した精度（桁数）と同じです。小数点以下の桁数であるスケールも指定できます。
- **DECIMAL データ型のサポート :** 10 進データはパック化された 10 進形式で、1 桁と記号を含む最終バイトを除いて、1 バイトに 2 桁ずつ含まれています。DECIMAL データ型には、精度とスケールが含まれているので、端数値が示されます。

データベース接続性の拡張

このリリースでは、パブリック・データベース・リンクを作成して、データベース全体で共有できるようになります。リポジトリの所有者や、`CREATE PUBLIC DATABASE LINK` 権限を持つ他のユーザーは、パブリック・データベース・リンクを作成できます。

HP-UX と AIX でも Warehouse Builder が使用可能に

このリリースからは、UNIX（HP-UX および AIX）プラットフォームで Warehouse Builder が使用可能になります。これは、以前のリリースから使用可能だった UNIX（Solaris および Linux）プラットフォーム、Windows（NT、2000 および XP）プラットフォームに加えて使用可能になりました（MITI Bridge の機能は、Windows プラットフォームでのみ有効になり、Name and Address Server は、Windows と Solaris プラットフォームでのみ使用できることに注意してください）。

Public API

このリリースでは、Public API が Warehouse Builder に組み込まれました。Public API にアクセスするには、次のファイルを解凍し、ローカル・マシン上のフォルダに展開します。

```
<owb home directory>¥owb¥lib¥int¥pubapi_javadoc.jar
```

ファイル `index.html` をダブルクリックします。API の使用方法については、「ヘルプ」リンクを選択します。

10*g* リリース 1（10.1）の追加機能

Oracle Warehouse Builder では、次の新機能が導入されました。

Warehouse Builder コンソールの変更

- **拡張ナビゲーション・ツリー :** Warehouse Builder コンソールに表示されるナビゲーション・ツリーが拡張されたため、プロジェクト間のナビゲーションが改善され、メタデータ・リポジトリ・オブジェクトに直接アクセスしやすくなりました。以前は一度に 1 つのプロジェクトしか表示できませんでしたが、すべてのプロジェクトをツリーで表示できるようになりました。プロジェクト・ノードを開いて、アクティブなプロジェクトの内容を表示できます。モジュール・ツリーは、別のウィンドウに表示されなくなりました。
- **ウィザード、エディタ、プロパティ・シート :** Warehouse Builder のウィザード、エディタおよびプロパティ・シートはすべて、ナビゲーション・ツリーから起動できます。
- **ビジネス・エリアからコレクションに名前変更 :** 以前のリリースでは、ウェアハウス・モジュールでビジネス・エリアを作成して、Warehouse Builder のオブジェクトを編成し、Oracle Discoverer などのツールにメタデータをエクスポートできました。このリリースからは、すべての機能でビジネス・エリアがコレクションに変更され、コレクションへのメタデータのインポート、コレクションからのメタデータのエクスポートなどの機能が拡張されています。
- **ファクト表からキューブに名前変更 :** OLAP の業界標準に対応するために、このリリースからファクトおよびファクト表という用語がキューブに変更されています。
- **論理名からビジネス名に名前変更 :** このリリースから、オブジェクトの論理名はビジネス名に変更されています。
- **Warehouse Builder コンソールのツールバー** : 最も重要な機能を 1 箇所にまとめるために、ユーティリティ・ドローワが削除され、Warehouse Builder コンソールの横および上部のツールバーが上部でマージされました。

配布の拡張

- **配布管理オブジェクトの追加** : このリリースでは、配布のソースおよびターゲットへの接続管理に役立つ 3 つのオブジェクト・タイプが導入されています。ロケーションでは、配布の物理的なロケーションが定義されます。コネクタでは、ロケーション間のリレーションシップが定義されます。ランタイム・リポジトリ接続では、Runtime Repository に関する情報が提供されます。これらのオブジェクトを使用すると、同一のターゲット設計に関して複数の配布ターゲットを作成できます。

- **単一の配布管理インタフェース :** デプロイメント・マネージャでは、単一のインタフェースを使用して、すべてのオブジェクトの配布や、配布されたマッピング、配布された変換、配布されたプロセス・フローの実行を管理できます。また、以前に配布されたオブジェクトの履歴にも直ちにアクセスできます。デプロイメント・マネージャによってこれらすべてのタスクを 1 つのインタフェースから実行できるだけでなく、Warehouse Builder ではランタイム・メタデータの追跡が行われ、以前に配布されたオブジェクトの履歴を確認できます。

Warehouse Builder メタデータ・ブラウザの拡張

- **設計メタデータの参照 :** Warehouse Builder Design Browser が拡張され、外部表、ロケーション、コネクタなど、新しく公開されたオブジェクトがすべて表示されます。また、Design Browser をスタンドアロンの実行可能ファイルで起動できるようになりました。シングル・ユーザーで使用する場合は、Oracle AS をインストールする必要はありません。
- **ランタイム・メタデータの参照 :** Warehouse Builder Runtime Audit Viewer が Runtime Audit Browser に変更され、Web ベースのレポート作成が可能になりました。Runtime Audit Browser を使用すると、以前のリリースに比べ、配布および実行の監査レポートが拡張されます。この監査データは、Runtime Repository に格納されている情報から取得されたもので、配布と実行の両方のデータが含まれています。

Warehouse Builder のプログラムで規定されたアクセスの拡張

- **Warehouse Builder Public API:** このリリースの Warehouse Builder では、Oracle Warehouse Builder の機能にプログラムで規定されたアクセスを行う方法として、アプリケーション・プログラマに一連の Public Java API が提供されるようになります。アプリケーション・プログラマは Public Java API を使用して、自分のアプリケーションに Warehouse Builder の機能およびサービスを埋め込むことができます。
- **Warehouse Builder スクリプト言語 :** Oracle MetaBase（OMB）Scripting Language を使用すると、Warehouse Builder の Graphical User Interface（GUI）を使用せずに、Warehouse Builder のすべての機能にアクセスできます。ユーザーは Warehouse Builder のスクリプト・ユーティリティである OMB Plus を使用して、Warehouse Builder のメタデータおよび機能にアクセスできます。これにより、開発者は、Warehouse Builder をプログラム的に使用し、必要に応じて機能を拡張できるようになります。OMB Scripting Language の詳細は、『Oracle Warehouse Builder スクリプト・リファレンス』を参照してください。

メタデータ管理の拡張

- **セキュリティ :** このリリースの Warehouse Builder では、セキュリティ要件に応じて採用するリポジトリ・セキュリティと監査システムのオプションが提供されます。複数の識別可能なユーザーが同一の Warehouse Builder Design Repository にアクセスできるように、マルチユーザー・アカウント・システムを作成できます。また、Warehouse Builder では、カスタマイズされた PL/SQL のセキュリティ実装パッケージを Warehouse Builder Design Repository にプラグインして、ユーザーの組織で定義されたセキュリティ・ルールに応じたアクセス制御をユーザーに提供できます。
- **メタデータ変更管理（メタデータのスナップショット**このリリースから、メタデータ・オブジェクトのスナップショットを取得したり、バックアップや履歴管理にスナップショットを使用したりできます。ナビゲーション・ツリーの任意のオブジェクトでスナップショットがサポートされます。スナップショットには、オブジェクト自身の情報（表やモジュールなど）や、オブジェクト内のオブジェクト（モジュール内の表など）の情報が格納されます。
- **複数言語サポート（MLS）:** この機能を使用すると、表示されているビジネス名や記述を、リポジトリの基本言語以外の言語で格納できます。ビジネス名や記述のさまざまな翻訳を使用して、ターゲット・ユーザーの言語で EUL に配布できます。
- **ユーザー定義プロパティによる拡張性 :** Warehouse Builder OMB Plus スクリプト・ユーティリティを使用すると、任意の Warehouse Builder オブジェクトに追加のプロパティを定義できます。スクリプトでユーザー定義プロパティを定義した後、ユーザー・インタフェース、Oracle MetaBase（OMB）Scripting Language、Warehouse Builder Public API および Warehouse Builder Design Browser で、これらのプロパティにアクセスできます。これにより、Warehouse Builder がさらに拡張され、他のビジネス・インテリジェンス製品との統合が簡単になります。
- **Metadata Loader（インポートおよびエクスポート）の柔軟性の拡張 :** この領域を拡張するために、2 つの新機能が追加されました。1 つ目は、メタデータをコレクションから直接エクスポートする機能です。2 つ目の機能は、Metadata Loader コマンドライン・ユーティリティで使用できます。これにより、ファースト・クラス・オブジェクトのインポート時に適用するアクションのタイプを柔軟に指定できます。

プロセス・フロー・エディタ

このリリースからは、Warehouse Builder でプロセス・フロー・エディタを使用して、プロセス・フローを作成および定義できます。以前マッピングで定義した外部プロセス演算子は、ユーザー定義プロセスにアップグレードされ、プロセス・フロー・モジュールに含まれます。プロセス・フローは、同じ Warehouse Builder 設計環境に統合されるので、Oracle Workflow 設計クライアントでこれらの機能を実行する必要はありません。Warehouse Builder プロセス・フロー・エディタでは、マッピングのセマンティックが固有の方法で解釈されます。プロセス・フロー・エディタで、FTP、電子メールなどのアクティビティをモデル化できます。

パフォーマンスの改善

- **マッピング・ユーザー・インタフェース :**「マッピング」という事前定義された表示セットが、このリリースで新たに追加されました。この表示セットを選択すると、「マッピング」では、事実上マップまたは使用されている列のみが表示されます。
- **マッピング圧縮 :** この機能では、指定されたマッピング内にある、演算子と属性の間の未使用接続が検出され、リポジトリから削除されます。これにより、大量のデータ・フローを要する大規模なマッピングをロードおよび格納する際のパフォーマンスが劇的に改善されます。
- **Metadata Loader（インポートおよびエクスポート）:** インポートおよびエクスポートの機能では、各マッピングに使用可能な新しい圧縮機能が利用されます。つまり、Metadata Loader では、実際に使用されているマッピング・オブジェクトのみがエクスポートおよびインポートされます。

Oracle Database の統合

- **OLAP の統合 :** Warehouse Builder を使用すると、多次元 OLAP オブジェクトを ROLAP モデルまたは MOLAP モデルとして、さまざまなデータ・ソースから設計、配布およびロードできます。データをロードしたら、BI ツールおよびアプリケーションを使用して、ビジネス上の疑問に対応する複雑な分析問合せを実行できます。Warehouse Builder を使用すると、リレーショナル・オブジェクトと多次元オブジェクトの両方を、同じキューブおよびディメンションの設計から作成および管理できます。
- **アドバンスト・キュー（AQ）の統合 :** Warehouse Builder では、アドバンスト・キューの定義をインポートしたり、データ・ウェアハウスを設計する際のデータ・ソースおよびターゲットとして AQ を使用したりできます。アドバンスト・キュー機能を Messaging Gateway と組み合せることで、Warehouse Builder では、MQ Series および Tibco で、メッセージング・アプリケーションを Warehouse Builder データ・ソースとしてサポートできます。AQ を使用すると、Oracle Database のチェンジ・データ・キャプチャ機能によってキャプチャされた変更データをソース・システムからターゲットへ伝播することもできます。AQ を統合する機能は、将来リアル・タイムのデータ・ウェアハウスを提供する基盤となります。
- **外部表 :** このリリースから、外部表を使用して、非リレーショナルのファイル・ソースのデータを、リレーショナルの読取り専用形式で表現できます。Oracle データベースから既存の外部表をインポートできます。あるいは、フラット・ファイル定義に基づいて、Warehouse Builder で外部表を作成できます。Warehouse Builder では、外部表を Oracle データベースに配布するための DDL が生成されます。
- **Oracle Database マルチ・テーブル・インサート :** Warehouse Builder では、ターゲット・データベースが Oracle Database である場合に、Oracle Database のデータベース機能を利用して、マルチ・テーブル・インサート文が生成されます。これにより、マッピングを最適化して、1 つの操作で複数の表にデータを挿入できます。

- **Oracle Database テーブル・ファンクション :** Warehouse Builder では、ターゲット・システムのロード時にパフォーマンスを改善できるようにテーブル・ファンクション演算子が導入されました。一連の入力行を操作し、可能なかぎり異なるカーディナリティの別の一連の行を返すカスタム・コードを開発する場合に、この演算子を使用します。従来のファンクションとは異なり、テーブル・ファンクションでは物理表のように問合せ可能な一連の行が出力されます。

フラット・ファイル・サポートの拡張

- **ターゲットとしての非バインドのフラット・ファイル :** このリリースでは、マッピングの作成時に、新しい非バインドのフラット・ファイル・オブジェクトを作成できます。Warehouse Builder では、指定のロケーションに、新しいカンマ区切りのシングル・レコード・タイプのフラット・ファイルが作成されます。この機能により、リレーショナル・オブジェクトの内容をフラット・ファイルに簡単にロードできます。
- **フラット・ファイルのアウトバウンド調整 :** アウトバウンド調整により、マッピング・フラット・ファイルから新しいリポジトリ・オブジェクトを作成できます。この結果、フラット・ファイルがリポジトリにとって新規である場合、新しいカンマ区切りファイルが指定のロケーションに作成されます。この機能により、リレーショナル・オブジェクトの内容をフラット・ファイルに即座にダンプできます。
- **デリミタ付きファイルの論理レコード :** フラット・ファイル・サンプル・ウィザードのユーザー・インタフェースが改善され、デリミタ付きファイルの論理レコードを定義できるようになりました。
- **位置ベースのマスター / ディテールのロード :** 追加のマッピング演算子を使用すると、位置ベースのマスター / ディテール・フラット・ファイルを簡単にロードできます。
- **SQL プロパティの拡張 :** Warehouse Builder にインポートするフラット・ファイルの SQL プロパティを指定できます。これにより、フラット・ファイルの各フィールドの SQL プロパティ値を事前に定義できます。したがって、フラット・ファイル・ソースをリレーショナル・ターゲットにマッピングする場合、ターゲットの列は、デフォルトでこれらの事前に定義された SQL プロパティ値になります。これらの値は、リレーショナル・ターゲットの列を構築する場合や、外部表の列を作成する場合に使用されます。

マッピング・エディタの拡張

- **マッピング・ユーザー・インタフェース :** 新しい一連のプロパティ・タブを使用して、マッピング演算子や属性プロパティを迅速に作成および編集できます。
- **ピボット演算子とアンピボット演算子 :** このリリースから、ピボット演算子やアンピボット演算子をマッピングに追加できます。ピボット演算子では、複数の属性が含まれる単一行を複数の行に変換できます。アンピボット演算子では、複数の入力行が 1 つの出力行に変換されます。
- **Name and Address 演算子の拡張 :** Name and Address 演算子が拡張され、新しい入力ロールと出力属性が追加されました。United States Postal Service Code Accuracy Support System（CASS）レポートも、このリリースからサポートされます。

UNIX プラットフォームでも Warehouse Builder が使用可能に

このリリースからは、Windows（NT、2000 および XP）プラットフォームに加え、UNIX（Solaris および Linux）プラットフォームで Warehouse Builder が使用可能になります。これは、Warehouse Builder のすべてのコンポーネントに該当しますが、このリリースでは、Linux 上で Name and Address ライブラリを使用することはできません（OLAP Bridge の機能は、Windows プラットフォームでのみ有効になり、Name and Address Server は、Windows と Solaris プラットフォームでのみ使用できることに注意してください）。

表記規則

このマニュアルでは、Windows NT、Windows 2000 および Windows XP オペレーティング・システムを総称して Windows と記載します。Oracle Database の SQL*Plus インタフェースは、SQL と記載する場合があります。

例では、特に明記しないかぎり、各行末に改行が挿入されています。そのため、入力の行末で [Enter] キーを押す必要があります。

次の表に、コード例で使用される表記規則とその使用例を示します。

表記規則	意味
. . .	垂直の省略記号は、例に直接関連しない複数の行が省略されていることを示します。
...	文またはコマンド中の横の省略記号は、その例に直接関係のない文やコマンドの一部が省略されていることを示します。
太字テキスト	テキスト中の太字は、インタフェースのボタンまたはリンクを示します。太字は、主題を強調する目的で使用します。
`Unicode` テキスト	Unicode テキストは、コード自体、ファイルのディレクトリや名前、リテラル・コマンドを示します。
イタリックの `Unicode` テキスト	イタリックの Unicode テキストは、ユーザーが値を指定するパラメータを示します。
[ ]	大カッコは、カッコ内の項目を任意に選択することを表します。大カッコは、入力しないでください。

関連文書

Warehouse Builder の文書には、次のマニュアルがあります。

- 『Oracle Warehouse Builder ユーザーズ・ガイド』
- 『Oracle Warehouse Builder インストレーションおよび構成ガイド』
- 『Oracle Warehouse Builder トランスフォーメーション・ガイド』
- 『Oracle Warehouse Builder スクリプト・リファレンス』
- 『Oracle Warehouse Builder リリース・ノート』

Warehouse Builder の文書に加えて、『Oracle データ・ウェアハウス・ガイド』も参照してください。

1

Oracle Warehouse Builder の概要

Oracle Warehouse Builder はビジネス・インテリジェンス・ツールで、エンタープライズ・データ・ウェアハウス、データ・マート、ビジネス・インテリジェンス・アプリケーションの設計や配布を行う統合型ソリューションです。これを使用すると、分散しているデータのソースとターゲット間でデータ統合を行う際の複雑な問題を解決できます。これに加えて、Warehouse Builder には、開発するシステムのライフサイクルを保持に必要な機能がすべて用意されています。

この章は次の項に分かれています。

製品のアーキテクチャと機能

Warehouse Builder は、データ・ウェアハウスとビジネス・インテリジェンス・システムの構築と管理を目的とした、高度な設計および実装ツールです。抽出（Extraction）、変換（Transformation）、ロード（Loading）用（ETL）ツールと設計ツールの主要なコンポーネントを組み合せて、1 つの製品を構成しています。これに加えて、Warehouse Builder は Oracle データベース・テクノロジを活用しています。Oracle ビジネス・インテリジェンス・ツール群統合の中心部分であり、非定形問合せツールと OLAP およびリレーショナル・データベース機能が提供されます。

Warehouse Builder のアーキテクチャは、設計環境とランタイム環境という 2 つのコンポーネントで構成されます。各コンポーネントでは、システムの別々の側面が扱われます。設計環境ではメタデータが管理され、ランタイム環境では物理データが扱われます。

設計コンポーネント

Warehouse Builder の設計コンポーネントは、Oracle データベースに格納された、スケーラビリティの高いメタデータ・リポジトリと、Java または HTML で作成されたクライアント設計およびレポート・ツールのセットで構成されます。これらのツールを使用すると、メタデータの表示と操作ができます。

メタデータの作成は設計アクティビティで、クライアント・ツール内のエディタを使用して、オブジェクト、プロセスおよびジョブを設計します。メタデータを作成するこの対話的な方法は、通常は新しいシステムの設計に使用します。Warehouse Builder では、リレーショナル・データベース・スキーマ、多次元スキーマ、ETL プロセス、エンド・ユーザー・ツール環境の設計が、クライアントを通じてサポートされています。

どの ETL ソリューションにおいても、ソース・システムの役割は重要です。Warehouse Builder ではメタデータを手動で作成するのではなく、関連情報をリポジトリにインポートする統合コンポーネントが用意されています。

このアーキテクチャの長所として、ライフサイクル・マネジメントがサポートされているので、ソース・システムの変更に基づいてメタデータを更新できます。また Warehouse Builder では、ETL プロセスとターゲット・システムに対する変更の伝播が簡単になります。

リポジトリ内のメタデータの品質と完全性を確保するため、Warehouse Builder では、リポジトリ内を広範に検証します。検証は、複数のユーザーによって作成された複雑なシステムを、正確で一貫性のある状態に保持するのに役立ちます。

メタデータの開発と評価に役立つように、Web ベースのメタデータ・レポート環境も使用できます。開発者とビジネス・ユーザーは、このレポート環境を使用すると、設計ツールを使用しなくても、システム要素の参照と調査ができます。このレポート環境の重要なコンポーネントが影響分析機能です。これにより、実際に変更を実行する前に、変更がシステム全体に及ぼす影響を確認できます。影響分析レポートを使用すると、高度な変更管理と変更実装計画が可能になります。これと反対の、データ元をトレースする機能は、データ系統レポートと呼ばれます。これも Warehouse Builder の機能です。

ランタイム・コンポーネント

ETL システムを論理レベルで設計したら、それを物理データベース環境に移動する必要があります。その前に、配布用にターゲットを構成する際、データベース環境の情報を論理設計に追加します。構成が完了した後、コードを生成できます。

Warehouse Builder では、ETL プロセス用の抽出固有言語と、データベース・オブジェクト用の SQL 文および DDL 文が生成されます。生成されたコードは、ファイル・システムとデータベースのどちらかに配布されます。

ETL 機能の実行とは、データベースに配布されたコードを実行することです。これは、Warehouse Builder のデプロイメント・マネージャを使用して、または Oracle Enterprise Manager などの外部ツールから実行できます。実行後、ETL プロセスによって、ソース・データがターゲット・データベースに取得されます。ターゲット・データベースとは、ステージング領域、オペレーショナル・データ・ストア、ウェアハウスなどのスキーマです。Oracle Database 外部のコード・セクションは、それぞれの環境で実行されます。たとえば、SAP システムから抽出する ABAP コードは SAP 環境で実行されます。

データ・ロードに関するレポートができるように、Warehouse Builder によって生成されたコードには、監査ルーチンが含まれています。このルーチンによって、Warehouse Builder のランタイム表に現在のロード情報が書き込まれます。コードの実行中に取得される情報には、選択、挿入または更新された行数などがあります。データの変換中またはロード中にエラーが発生した場合、監査ルーチンによって、ランタイム表にエラーがレポートされます。このランタイム情報に簡単にアクセスし、便利なレポートができるようにするため、Warehouse Builder には Runtime Audit Browser が組み込まれています。

依存性管理とスケジューリングは、特定の Oracle ツールと密接に統合して行います。Oracle Enterprise Manager は、Oracle のスケジューリングとデータベース管理を行うツールです。Warehouse Builder では Oracle Enterprise Manager リポジトリにジョブを作成し、他のデータベース・アクティビティとともに、ジョブのスケジューリングと監視ができます。Warehouse Builder のユーザーは、Oracle Workflow（OWF）との対話を通じて、通知などの ETL プロセス間の依存性に対して、本格的なプロセスを作成できます。

Warehouse Builder の目標達成方法

ビジネス・インテリジェンス・アプリケーションの作成手順は複雑です。多数のシステム、リソース、機能領域にまたがる様々なステップとフェーズが関わっています。Warehouse Builder を使用すると、1 つのインタフェースからタスクを管理できるので、この手順が簡単になります。また最新の Oracle データベース・テクノロジが活用されているので、スケーラビリティ、信頼性、柔軟性が確保されます。

Warehouse Builder の主要な機能には次のものがあります。

- ソース・データ定義のインポート
- ターゲット・データベース・スキーマ・オブジェクトの設計と作成
- ソースとターゲット間での、データ移動とデータ変換の定義

- ETL プロセス間の依存性割当て
- ソース定義の管理と更新
- ターゲット・スキーマ・オブジェクトの配布、アップグレードおよび管理
- 非定形問合せツール環境の設計と作成
- OLAP 環境の設計と作成

Warehouse Builder のコンポーネント

Warehouse Builder は、次の主要コンポーネントで構成されます。

Warehouse Builder Design Client アプリケーション

Warehouse Builder Design Client アプリケーションには、操作の簡単なグラフィカル・インタフェースがあり、これでビジネス・インテリジェンス・システムの定義、設計、配布が行えます。多くのコンポーネントはプロセスの各部分に活用されています。コード・ジェネレータとデプロイメント・マネージャはクライアント・アプリケーションのコンポーネントで、ユーザーの作成するシステムの作成と管理を制御します。

コード・ジェネレータ このコンポーネントでは、リポジトリ内のメタデータに基づいて、DDL や PL/SQL などのスクリプトが生成されます。このジェネレータは、Oracle8*i* および Oracle9*i* Database および Oracle Database 10*g* の機能を利用するように設計されています。生成されたスクリプトにより、Oracle Database システムのパフォーマンスが最適化されます。

デプロイメント・マネージャ このコンポーネントにより、配布と配布済オブジェクトのすべての面が管理されます。配布するオブジェクトを選択し、オブジェクトの配布方法を決定できます。次に、配布済オブジェクトを実行できます。また、配布履歴に即座にアクセスして、システムのライフサイクルを管理することもできます。

Warehouse Builder Runtime Platform Service

Runtime Platform Service は、Warehouse Builder ソフトウェアのサーバー側のコンポーネントで、サービスの実行と配布を行います。こうしたサービスを実行できるようにするには、Runtime Platform Service をアクティブにしておく必要があります。

Runtime Platform Service では、マッピングとプロセス・フローの実行が、Warehouse Builder 内から管理され、実行と配布の監査データがすべて、Runtime Repository に格納されます。リモート実行の場合、Oracle Enterprise Manager の Management Server に接続されます。Runtime Platform Service は、データベース・ジョブを通じて呼び出されます。このジョブは、データベースが起動したときに自動的に起動し、データベースがシャットダウンされたときに自動的にシャットダウンされます。

Warehouse Builder Design Repository

Oracle Database にインストールされた Design Repository には、Warehouse Builder で使用されるすべてのオブジェクトのメタデータ定義が格納されます。ここには、作成するターゲット・システムの設計情報がすべて格納されます。クライアント・ユーザー・インタフェースや OMB Plus（Warehouse Builder のスクリプト・ユーティリティ）を使用すると、ここに格納したメタデータにアクセスできます。

Warehouse Builder Runtime Repository

Oracle Database にインストールされたランタイム・リポジトリには、配布データおよび実行したマッピングやプロセス・フローの情報がすべて格納されます。ここには、作成するビジネス・インテリジェンス・システムのターゲット環境情報が格納されます。ここには、すべての配布場所の接続情報が含まれます。

Warehouse Builder Runtime Audit Browser

このレポート・ブラウザを使用すると、Web ベースのアプリケーションから、配布および実行の監査情報を表示できます。レポートはランタイム・リポジトリに格納されたデータに基づいています。

Warehouse Builder Design Browser

このレポート・ブラウザを使用すると、Web ベースのアプリケーションから、設計リポジトリに格納したメタデータを表示できます。また、より多くのユーザーに情報を提供できます。情報は、業務担当ユーザー向けの形式に整理されます。レポートは Warehouse Builder Design Repository に格納されたデータに基づいています。

Warehouse Builder のオブジェクト

Warehouse Builder には、ファーストクラス・オブジェクトと呼ばれるオブジェクトのカタログがあります。これには、インポート、設計および配布が可能なオブジェクトがあります。

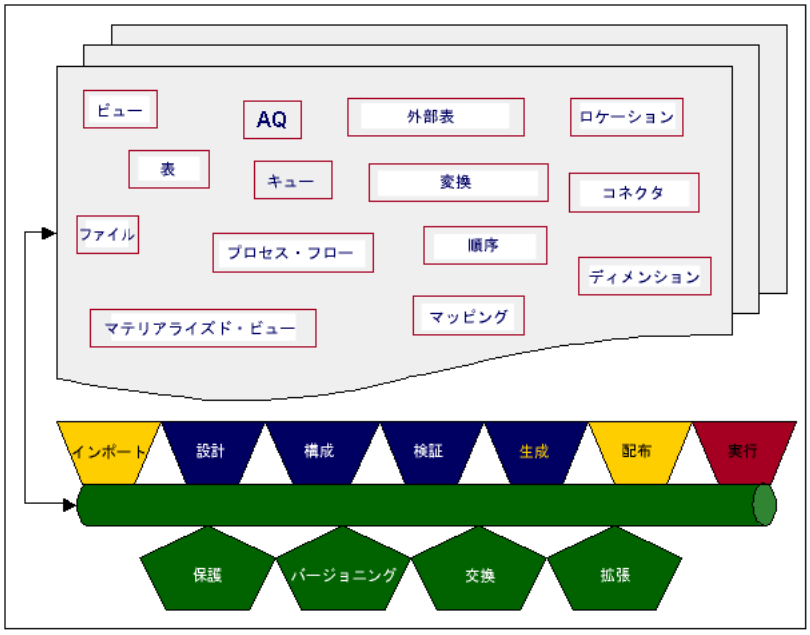
Warehouse Builder のファーストクラス・オブジェクトには、次のものがあります。

- アドバンスト・キュー
- 外部表
- コネクタ
- キューブ
- ディメンション
- ファイル
- ロケーション

- マッピング
- マテリアライズド・ビュー
- プロセス・フロー
- 順序
- 表
- 変換
- ビュー

ファーストクラス・オブジェクトには、Warehouse Builder のサービス経由で実行できるオブジェクトのセットがあります。図 1-1 では、ファーストクラス・オブジェクトがダイアグラムの 1 番上に、Warehouse Builder の使用可能サービスが一番下に示されています。

図 1-1 ファーストクラス・オブジェクトとサービス



Warehouse Builder の配布ターゲット

Warehouse Builder でターゲットを設計する場合、様々な配布ターゲットを選択できます。ビジネス・インテリジェンスのニーズに応じて、設計の配布方法を決定できます。Warehouse Builder を使用すると、次のシステムのいずれにも配布できます。

- Oracle Database
- Oracle OLAP
- Oracle Enterprise Manager
- Oracle Workflow
- Oracle Discoverer
- CWM Bridges

Warehouse Builder を使用する利点

Warehouse Builder を使用すると、データ統合とビジネス・インテリジェンス・システムの配布を行う際に、次のような利点があります。

- **設計の迅速化 :** Warehouse Builder では、操作が簡単なビジュアル・エディタ、ウィザード方式のユーザー・インタフェース、事前定義済変換のライブラリが提供されているため、設計時間を短縮できます。
- **設計の一元化 :** システムの設計情報はすべて、Warehouse Builder の設計リポジトリという、中心となる 1 つの場所に格納および管理されています。これは設計全体の単一ソースで、これにより、メタデータの複製を削減するために不整合が発生するという余分なコストを削減できます。
- **変更の組み込みに要する時間の短縮 :** 拡張ライフサイクル・マネジメントは、単一のリポジトリに基づいているので、メンテナンス処理がスムースに行えます。
- **コードにエラーがない :** コードは 1 箇所に格納された設計から生成されるため、エラーがありません。さらに、再作成、アップグレード、メンテナンスも簡単に行えます。
- **技術投資の活用 :** 変換エンジンとストレージ機能として Oracle データベースを使用すれば、Warehouse Builder では、Oracle のスケーラビリティ、パフォーマンス、セキュリティ、高可用性が活用されます。

2

Warehouse Builder スタート・ガイド

Warehouse Builder には多くのウィザード、エディタ、ツールがあり、ビジネス・インテリジェンス・システムの設計や配布に役立ちます。この章では、Warehouse Builder を使用してターゲット・システムの設計と作成を行う方法を、簡単に説明します。

この章では、次のトピックについて説明します。

ビジネス・インテリジェンス・システム作成の概要

Warehouse Builder では、分散したソースからデータを統合して、ビジネス・インテリジェンス・ソリューションを作成できます。既存のソースからデータ・システムを構築する理由は様々です。多くの場合、ビジネス上の決定を行う際に使用できるようにデータを情報に交換し、統合できます。Warehouse Builder には、使いやすいウィザードとエディタが用意されているので、そのようなシステムの設計プロセスが簡単になります。システム設計後、Warehouse Builder では、データ・ウェアハウス、データ・マート、ビジネス・インテリジェンス・アプリケーションなど、各種の配布ターゲットをサポートしています。

ターゲット・システムを構築するには、次の手順を実行します。

- 手順 1: プロジェクトの作成
- 手順 2: ソース・モジュールとターゲット・モジュールの定義
- 手順 3: データ移動とデータ変換の定義
- 手順 4: 検証と生成
- 手順 5: 配布および実行

手順 1: プロジェクトの作成

ターゲット・システムの設計を開始するには、新しいプロジェクトを作成します。Warehouse Builder のプロジェクトは、Warehouse Builder Design Repository 内で最大の格納オブジェクトです。プロジェクトにより、ターゲット・システムのメタデータ定義が格納および整理されます。これらの定義には、データ・ソース、ターゲット・ウェアハウス・オブジェクト、ソースからターゲットへのマッピング、変換操作、構成パラメータなどがあります。これらの定義は、プロジェクト内のフォルダに整理されます。通常、プロジェクトには関連するメタデータがあります。複数のプロジェクトを作成すると、設計を組織的に作成できます。

手順 2: ソース・モジュールとターゲット・モジュールの定義

ソース・モジュールとターゲット・モジュールを定義し、システムのモデル設計を開始します。モジュールとはプロジェクト内の格納オブジェクトで、ソースとターゲットのオブジェクト整理に役立ちます。ソース・モジュールには、取得元となる既存のソース・システムに入っているメタデータが含まれています。ターゲット・モジュールには、設計対象のメタデータが含まれます。モジュールの作成は、ソースとターゲットのどちらから始めてもかまいません。

ソースの定義

既存のソース・システムからデータを選択して、ソース・モジュールを作成します。ソース・モジュールには、次の 3 種類があります。

- データベース
- ファイル
- アプリケーション

Warehouse Builder でデータ・ソースを定義する場合、ソースとオブジェクトの論理定義をインポートしてモデルに入れます。これらの定義は、Design Repository のプロジェクト内に格納されます。ソース・モジュールの作成手順は、使用しているソースのタイプによって異なります。詳細は、第 4 章「データ定義のインポート」を参照してください。

ターゲットの定義

配布時に作成するターゲット・ウェアハウスのモデルを作成して、ターゲットを定義します。Oracle データベース・ウェアハウス・モジュールを使用して、メタデータ定義を格納します。新規オブジェクトの作成と設計ができます。また、ソース・データのロケーションから定義をインポートすることもできます。詳細は、第 3 章「Oracle データ・オブジェクトの定義」を参照してください。

ロケーションとコネクタの定義

ソースまたはターゲットのモジュールを定義する際に、配布を正しく実行できるロケーションにモジュールを登録する必要があります。正しく登録した場合、ソースまたはターゲットのモジュールのタイプとバージョンがロケーションに定義されます。ロケーションは論理定義として Design Repository に保存されます。配布の実行中に、接続情報が要求され、Runtime Repository に保存されます。

マッピングによって他のモジュールにリンクされているモジュールに、コネクタを定義する必要もあります。コネクタでは、各モジュールのロケーション同士がリンクされます。コネクタ定義は、論理定義として Design Repository に保存されます。コネクタによってリンクされた 2 つのロケーションが、別々のマシン上にある場合、配布中にデータベース・リンクを生成するために、コネクタ定義が使用される場合があります。詳細は、3-7 ページの「コネクタの定義」と 13-3 ページの「Runtime Repository 接続の定義」を参照してください。

構成

ターゲット・モジュール内のオブジェクトを構成し、オブジェクトの物理的特性を定義します。ターゲットの定義に使用できる構成パラメータがいくつもあります。デフォルト値も使用できます。データ・オブジェクト構成の詳細は、第 5 章「データ・オブジェクトの構成」を参照してください。

手順 3: データ移動とデータ変換の定義

ソースとターゲットのモジュールを定義したら、データの移動および変換を行うロジックを作成します。これは ETL ロジックとも呼ばれています。Warehouse Builder で行う設計作業の大部分は、ETL ロジックの定義です。Warehouse Builder におけるデータの移動と変換を定義するには、次のコンポーネントを使用します。

- 変換
- マッピング
- プロセス・フロー

これらのオブジェクトを使用して、ターゲット・システムを作成する際にソース・データを操作する方法を定義します。マッピング・エディタで使用可能なマッピング演算子が大量にあり、変換のライブラリ一式も用意されています。詳細は、第 II 部「ETL オブジェクトの設計」を参照してください。

構成

マッピングとプロセス・フローを構成し、オブジェクトの物理的特性を定義します。構成パラメータを使用し、ターゲット・システムの配布を最適化します。デフォルト値も使用できます。データ移動とデータ変換のオブジェクト構成の詳細は、第 11 章「ETL オブジェクトの構成」を参照してください。

手順 4: 検証と生成

ソースとターゲットのモジュールを定義し、ETL ロジックを定義したら、配布前にコードの検証と生成を行って、エラーがないかチェックできます。

検証

無効または不完全な定義をなくすように、すべてのオブジェクトを検証できます。無効な定義を見つけた場合、エディタを使用して問題を修正します。定義がすべて有効な場合、配布に向けてモジュールとコンポーネントを構成できます。詳細は、12-2 ページの「検証について」を参照してください。

生成

配布の前にスクリプトの生成と表示ができます。生成されるコードのタイプは、ターゲットのタイプに定義された構成パラメータによって異なります。Warehouse Builder の外部からコードを配布する場合、生成したコードを保存することもできます。詳細は、12-8 ページの「生成済スクリプトの表示」を参照してください。

手順 5: 配布および実行

ターゲット・モデルが完成したら、ターゲット環境に設計を配布します。配布の前に、Runtime Repository をインストールし、Runtime Repository 接続を定義する必要があります。Warehouse Builder には、2 つの配布方法があります。メイン・コンソールから直接配布する方法と、デプロイメント・マネージャを使用する方法です。その後、デプロイメント・マネージャからマッピングとプロセス・フローを実行できます。配布の前にオブジェクトの検証と生成もできます。

Runtime Repository 接続の作成

オブジェクトの配布時には、Runtime Repository 接続を選択する必要があります。指定する接続で、配布データを格納する Runtime Repository が定義されます。配布を行う前に、Runtime Repository をインストールし、ナビゲーション・ツリーで Runtime Repository 接続を作成する必要があります。Runtime Repository 接続には、配布の管理に使用する Runtime Repository への接続が記述されます。Runtime Repository の詳細は、『Oracle Warehouse Builder 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。

デプロイメント・マネージャの使用方法

配布および実行のプロセスで、デプロイメント・マネージャを使用すると便利です。デプロイメント・マネージャには、完全な配布プラットフォームがあります。また、配布と将来のアップグレードを管理し、マッピングとプロセス・フローを即座に実行するインテリジェント・コンポーネントもあります。また、デプロイメント・マネージャでは、システムのアップグレードについて決定できるように、以前に配布したオブジェクトのデータを表示することもできます。配布と実行、およびデプロイメント・マネージャの詳細は、第 13 章「ターゲット・システムの配布」を参照してください。

配布と実行の監査レポートの表示

Warehouse Builder Runtime Audit Browser を使用して、マッピングおよびプロセス・フローの配布と実行に関するレポートを表示します。ターゲットにオブジェクトを配布する場合、またはスクリプトを実行する場合、各配布および実行に関するデータは、Runtime Repository に格納されます。Runtime Audit Client ベースのブラウザと、Runtime Audit Portal ベースのブラウザのどちらかを使用して、レポート内でこのデータにアクセスできます。詳細は、第 14 章「配布と実行の監査」を参照してください。

Warehouse Builder Design Repository へのアクセス

Warehouse Builder Design Client にログインするには、Warehouse Builder Design Repository スキーマを、データベース内に新規作成する必要があります。Warehouse Builder Design Repository には、データ・ウェアハウスを設計するために、Warehouse Builder 内に作成またはインポートされたすべてのオブジェクトの定義が格納されます。OWB Repository Assistant でリポジトリを新規作成して、Warehouse Builder Design Client を実行し、Warehouse Builder メタデータにアクセスします。

オプション機能として、Warehouse Builder では、複数のユーザー・スキーマから同じリポジトリにアクセスできます。Warehouse Builder では、次のものを実装すると、リポジトリ・セキュリティと監査システムが用意されます。

- **マルチユーザー・アカウント・システム :** 複数の識別可能なユーザーが、同じ Warehouse Builder Design Repository にアクセスできます。Warehouse Builder Design Repository スキーマの所有者が、ユーザーを設定します。
- **セキュリティ登録サービス :** Warehouse Builder Design Repository に入っているセキュリティ PL/SQL 実装パッケージをカスタマイズしてプラグインできます。Warehouse Builder では、ユーザー組織が PL/SQL パッケージで規定するセキュリティ・ルールに従って、正しいアクセス制御を定めることができます。

Warehouse Builder を使用すると、次の作業が行えます。

- マルチユーザー・アカウント・システムと、セキュリティ登録メカニズムの両方を使用して、Warehouse Builder にセキュリティを実装します。
- マルチユーザー・アカウント・システムのみを使用して、Warehouse Builder の中央リポジトリに複数の識別可能なユーザーがアクセスできるようにします。ユーザーは監査証跡を残しますが、アクセス制御は行われず、全ユーザーが Warehouse Builder のすべての操作を起動できます。
- セキュリティ機能を実装しない、またはマルチユーザー・アカウント・システムを使用しないという選択を行います。ユーザーごとに別々のリポジトリ・スキーマを作成できます。また、複数のユーザーがリポジトリ・ログインとパスワードを共有し、同じリポジトリを使用するようにもできます。

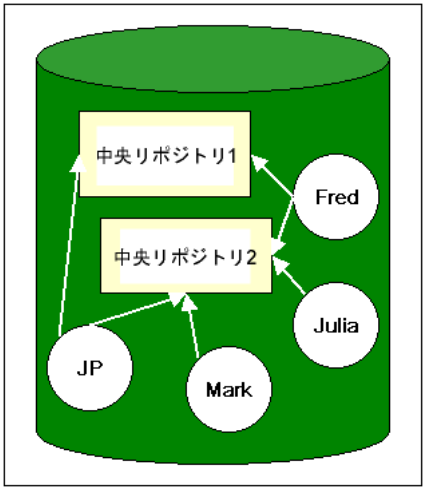
次の項では、Warehouse Builder でマルチユーザー・アカウントとセキュリティ登録メカニズムを設定する方法を説明します。

マルチユーザー・アカウント・システム

マルチユーザー・アカウント・システムを使用すると、複数のユーザー・スキーマが Warehouse Builder Design Client を使用して、Warehouse Builder の中央リポジトリ・スキーマにアクセスできます。これにより、中央スキーマの所有者は、他のユーザーが同じリポジトリに対して行った変更を、監査および追跡できます。ただし、これらのリポジトリ・ユーザーは、SQL*Plus など、他のメソッドを使用して Warehouse Builder Design Repository にアクセスしてはなりません。Warehouse Builder では、これが強制されています。

図 2-1 では、複数のウェアハウス・ユーザー（円の内側）が中央リポジトリ（四角の内側）にアクセスする方法を示しています。

図 2-1 ユーザーが複数いる状態の Warehouse Builder Design Repository



Warehouse Builder の複数のユーザー・スキーマが、Warehouse Builder の中央リポジトリにアクセスできるようにするには、次を実行する必要があります。

- Warehouse Builder の各ユーザーは、Warehouse Builder の中央リポジトリと同じデータベース・インスタンスに、自分専用の Oracle データベース・スキーマを作成しておく必要があります。
- 中央スキーマの所有者は、予想される Warehouse Builder ユーザーをすべてリポジトリに登録する必要があります。2-8 ページの「Warehouse Builder ユーザーの登録」を参照してください。

Warehouse Builder の複数のユーザーが、Warehouse Builder Design Client セッションを開始し、Warehouse Builder の同じ中央リポジトリにアクセスできるようになりました。

ヒント： Warehouse Builder のユーザーは Warehouse Builder Design Client からのみ、Warehouse Builder の中央スキーマにアクセスできます。SQL*Plus からはアクセスできません。Warehouse Builder では、これが強制されています。

Warehouse Builder ユーザーの登録

Warehouse Builder ユーザーの作成と登録を行うには、ユーザーは Warehouse Builder の中央リポジトリと同じデータベース・インスタンスに、スキーマを持っている必要があります。

データベース・ユーザーではない Warehouse Builder ユーザーを登録する手順は次のとおりです。

1. DBA は、SQL*Plus または OEM を使用して、通常のデータベース・ユーザー（User1）を作成する必要があります。
2. DBA は、USER1 のデフォルト・ロールを制限するため、次の SQL 文を実行する必要があります。

   ```
   ALTER USER user1 DEFAULT ROLE NONE;
   ```

   データベース・ユーザーを最初に作成したとき、ユーザーのデフォルト・ロールは ALL に設定されています。マルチユーザー・アカウント・システムでは、ユーザーのデフォルト・ロールを ALL に設定できないので、この設定を変更する必要があります。変更しないと、以降の登録手順が失敗します。
3. リポジトリ所有者は SQL*Plus を使用してリポジトリにログインし、次のコマンドを入力する必要があります。

   ```
   call WBSecurityHelper.registerOWBUser('User1');
   ```

   これで、データベース・ユーザー User1 は Warehouse Builder のユーザーとなり、Warehouse Builder Client を使用して、Warehouse Builder の中央リポジトリにアクセスできるようになりました。

すでにデータベース・ユーザーである Warehouse Builder ユーザーを、Warehouse Builder ユーザーとして登録する手順は次のとおりです。

1. リポジトリ所有者は SQL*Plus を使用してリポジトリにログインし、次のコマンドを入力する必要があります。

   ```
   call WBSecurityHelper.registerOWBUser('user1');
   ```

   これで、データベース・ユーザー User1 は Warehouse Builder のユーザーとなり、Warehouse Builder Client を使用して、Warehouse Builder の中央リポジトリにアクセスできるようになりました。

上記のプロシージャ・コールが次のエラー・メッセージにより失敗した場合、

「user: User1 has the DEFAULT ROLE set to ALL.The database administrator should use ALTER USER statement to limit the default roles of user: User1.Then register the OWB user again.」

データベース管理者は、ALTER USER 文を使用して、ユーザーのデフォルト・ロール（ALL は不可）を制限する必要があります。

その他のユーザー管理ユーティリティ

Warehouse Builder には、次のメンテナンス・タスクを実行するためのユーティリティ・プロシージャが含まれています。

- ロール・パスワードの更新

  リポジトリ所有者は、ロールの保護パスワードを明示的に使用するわけではありませんが、パスワードは頻繁に変更することをお薦めします。

  リポジトリ所有者はリポジトリがあるデータベースに接続し、SQL*Plus から次の文を発行する必要があります。

  ```
  Call WBSecurityHelper.updateRolePwd('mewpwd');
  ```

  'newpwd' は、リポジトリ所有者がロール保護用に選択した新しいパスワードです。変更されたパスワード暗号は、OWB_Role_Info という名前の表に表示されます。

- リポジトリ・ユーザーの登録解除

  リポジトリ・ユーザーを登録解除するには、リポジトリ所有者はリポジトリ・スキーマに接続し、SQL*Plus から次の文を実行する必要があります。

  Call WBSecurityHelper.unregisterOWBUser('username');

- すべてのリポジトリ・ユーザーの表示

  リポジトリ所有者はリポジトリ・スキーマに接続し、次の文を発行する必要があります。

  ```
  Set serveroutput on;
  Call WBSecurityHelper.list OWBUsers();
  ```

  list OWBUsers() では、ユーザー・インタフェースへのデータのダンプ出力に DBMS_OUTPUT.put_line が使用されるため、Set serveroutput on を使用する必要があります。それ以外の場合、DBMS_OUTPUT.put_line は出力を中間のデータ構造にのみダンプ出力します。

マルチユーザー・アカウント・システムの動作

次の手順では、Warehouse Builder 内のマルチユーザー・アカウント・システムの動作を示します。

1. Warehouse Builder Design Client で要求される権限を収集します。
2. Warehouse Builder のインストール中に OWB Repository Assistant を実行します。パスワードで保護された `OWB USER ROLE` というデータベース・ロールが、自動的に作成されます。
3. これらの権限を、新規に作成された `OWB USER ROLE` に付与します。
4. Warehouse Builder ユーザーが新規登録されるたびに、ユーザー登録プロセスを通じて、そのユーザーに `OWB USER ROLE` が付与されます。
5. Warehouse Builder Design Client セッションが起動すると、Warehouse Builder によって `OWB USER ROLE` が有効化され、ユーザーは中央リポジトリ・スキーマにアクセスできるようになります。
6. オプション機能として、Warehouse Builder ユーザーがリポジトリの提供するパブリック・ビューにアクセスする必要がある場合、中央スキーマの所有者は、そのユーザーに `OWB PUBLIC VIEW` というロールを付与できます。たとえば、`"OWBR_"+ repos_name` は、OWB パブリック・ビューのロール名で、リポジトリ名は中央リポジトリ・スキーマの名前です。このロールは、OWB Repository Assistant でも作成されます。

セキュリティ登録サービス

Warehouse Builder にはセキュリティ登録サービスがあり、これを使用すると、Warehouse Builder にアクセス制御機能を持たせるように拡張できます。このサービスを使用すると、ユーザー組織のセキュリティ・ポリシーに従って、セキュリティ・サービス実装をプラグインでき、ユーザーが Warehouse Builder から起動した操作に対して、正しいアクセス制御が行われます。現時点では、READ 権限に制限はありません。

Warehouse Builder には PL/SQL パッケージ・インタフェースがあり、ユーザーが各自のセキュリティ・サービス実装をプラグインする場合に実装できます。この PL/SQL パッケージ・インタフェースとダミー実装は、Warehouse Builder Design Repository にインストールされています。ダミー実装にはインテリジェント機能がないため、Warehouse Builder におけるすべての操作を起動する許可が、すべてのユーザーに付与されます。Warehouse Builder セキュリティを使用する場合、カスタマイズした実装をリポジトリにインストールし、このダミー PL/SQL パッケージ実装を、ユーザー独自の実装に置き換える必要があります。

PL/SQL パッケージ・インタフェース内で定義される主要なプロシージャは、ログイン・ユーザーが起動した操作が可能かどうかを認識する際に、コールアウト・プロシージャとして Warehouse Builder によって使用されます。PL/SQL 実装によって、Warehouse Builder がアクセス制御を決定できます。プラグインの仕組みとして PL/SQL パッケージを使用することには、次の利点があります。

- **セキュリティ :** リポジトリ・スキーマ内の PL/SQL パッケージを作成または更新する権限を持つのはリポジトリ所有者のみなので、他のユーザーはプラグイン・パッケージを置き換えられません。
- **柔軟性 :** この PL/SQL パッケージを実装することで、セキュリティ・ポリシーの定義と適用ができます。
- **利便性 :** SQL*PLUS から PL/SQL パッケージの本体ファイルをリポジトリ・スキーマにインストールすると、セキュリティ・メカニズムが実装されます。

セキュリティ登録サービスの使用方法の詳細は、19-8 ページの「PL/SQL でのセキュリティ管理」を参照してください。

権限

Warehouse Builder の権限には、次の 2 つのタイプがあります。

- **システム権限 :** ユーザーがこのタイプの権限を付与された場合、操作中のオブジェクトが与える影響に関係なく、システム全体でその権限を行使できます。次に示すのは、サービス・タイプ権限とも呼ばれるシステム権限の一覧です。

  DEPLOY
  MDL_IMPORT
  MDL_EXPORT
  BRIDGE_IMPORT
  BRIDGE_EXPORT
  RUNTIME_EXECUTE
  SNAPSHOT_RESTORE
  SOURCE_IMPORT

- **オブジェクト権限 :** この権限は、特定の Warehouse Builder オブジェクトに接続できます。このタイプの権限は、オブジェクト、オブジェクト・タイプ、モジュール・フォルダ（ユーザーは、モジュールの下のすべてのオブジェクトに対して操作を起動できます）、プロジェクト・フォルダの各レベルで、ユーザーに付与されます。次に示すのは、オブジェクト権限の一覧です。

  EDIT
  DELETE
  REFERENCE
  VALIDATE
  GENERATION
  VERSION
  CREATE

オブジェクトと権限の詳細な一覧は、19-8 ページの「PL/SQL でのセキュリティ管理」を参照してください。

注意： 現在のリリースでは、すべてのオブジェクトに対する READ 権限が、すべてのユーザーに付与されています。

インストール後

リポジトリに PL/SQL パッケージをインストールした後、Warehouse Builder では、実装に埋め込んだセキュリティ・ポリシーに従って、正しいアクセス制御が行われます。たとえば、プロジェクトの編集権限がないユーザーがプロジェクトを編集しようとすると、Warehouse Builder によって、そのユーザーの作業が停止され、エラー・メッセージが表示されます。

セキュリティ要件

Warehouse Builder セキュリティに関しては、次の点に注意してください。

- データベース管理者は、一般のユーザーや Warehouse Builder ユーザーに対して `EXECUTE ALL PROCEDURES` 権限を付与しないでください。
- Warehouse Builder ユーザー・スキーマのデフォルト・ロールは、`ALL` に設定しないでください。
- 中央スキーマの所有者や管理者は、Warehouse Builder ユーザーに、Warehouse Builder の中央スキーマにアクセスする権限を付与しないでください。
- セキュリティ・サービス・パッケージの実装は、正しく行う必要があります。

Warehouse Builder の起動

Warehouse Builder を使用するには、既存の Warehouse Builder Design Repository にアクセスできるか、OWB Repository Assistant を使用してリポジトリを新規作成する必要があります。Warehouse Builder を起動する際に、使用可能な接続情報が必要です。Warehouse Builder Design Repository 作成の詳細は、『Oracle Warehouse Builder インストレーションおよび構成ガイド』を参照してください。

Windows NT/2000/XP ユーザー

WindowsNT /2000/XP を使用して Warehouse Builder を起動する手順は次のとおりです。

1. 「スタート」→「プログラム」→「Oracle Database」→「Warehouse Builder」→「OWB Client」を選択します。

   「Warehouse Builder ログイン」ダイアログが表示されます。

2. 「接続情報」をクリックします。

   「接続情報」ダイアログが表示されます。

3. Warehouse Builder Design Repository を含むマシンの接続情報として、データベース・サーバーのホスト名、データベース・インスタンスのポート番号、Oracle サービス名を入力します。

   接続情報を一度指定しておけば、Warehouse Builder を起動するたびに同じ情報が使用されます。別のリポジトリにログインする場合は、このダイアログを開いて接続情報を編集する必要があります。

4. 「OK」をクリックします。

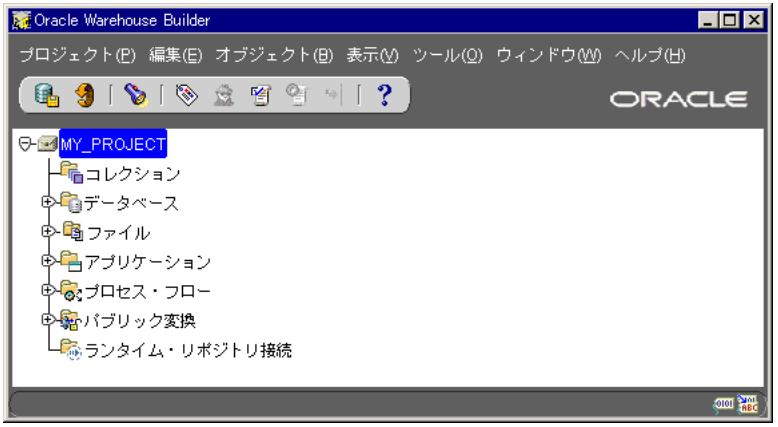
   Warehouse Builder Design Client に接続情報が保存され、ログイン・ダイアログが再表示されます。

5. Warehouse Builder Design Repository のユーザー名とパスワードを入力し、「ログイン」をクリックします。

   Oracle8*i* または 9*i* のデータベース・インスタンスが非アクティブのときは、接続エラー・メッセージが表示されます。

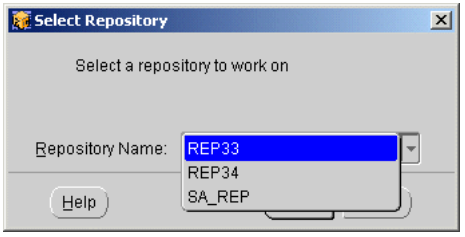
   Warehouse Builder コンソールが起動し、ナビゲーション・ツリーに表示されたリポジトリの内容が表示されます。新規に作成したリポジトリの場合、デフォルト・プロジェクトの MY_PROJECT が表示されます。図 2-2 では、Warehouse Builder コンソールに MY_PROJECT というデフォルト・プロジェクトが表示されています。

図 2-2 Warehouse Builder コンソール



ログイン・ユーザーが Warehouse Builder Design Repository の所有者でない場合、Warehouse Builder では、図 2-3 に示すように、「リポジトリの選択」ダイアログが表示されます。アクセスするリポジトリを選択し、「OK」をクリックします。

図 2-3「リポジトリの選択」ダイアログ



複数のユーザーが使用するリポジトリにアクセスする場合、コンソールを更新してリポジトリに格納されている内容を反映させるため、同期化を定期的に行ってください。詳細は、2-16 ページの「マルチユーザーのサポート」を参照してください。

UNIX ユーザー

UNIX を使用して Warehouse Builder を起動する手順は次のとおりです。

1. シェルを起動します。
2. <OWB ORACLE HOME>/owb/bin/unix に移動（cd）します。

   例：/private/home/OWB904/owb/bin/unix

表 2-1 UNIX

Warehouse Builder のコンポーネント	起動
Warehouse Builder Design Client	owbclient.sh
Warehouse Builder Browser Assistant	browserasst.sh
Warehouse Builder MDL File Upgrade Utility	mdlconvertui.sh
Warehouse Builder Repository Assistant	reposasst.sh
Warehouse Builder Runtime Assistant	runtimeinst.sh
Warehouse Builder OMB Plus	OMBPlus.sh

Warehouse Builder の終了

Warehouse Builder を終了する手順は次のとおりです。

1. 「プロジェクト」メニューから「終了」を選択します。

   変更内容をコミットしなかった場合は、「コミット確認」ダイアログが表示されます。
2. 変更内容をコミットするには、「はい」をクリックします。変更内容を廃棄するには、「いいえ」をクリックします。

   Warehouse Builder が閉じ、セッションが終了します。

設定内容のコミット

Warehouse Builder のセッション中は、いつでも Design Repository に変更をコミットできます。セッション中に変更内容をコミットするには、コンソール・ウィンドウに移動してツールバーの「コミット」アイコンをクリックします。「コミット確認」ダイアログが表示されます。「はい」をクリックして変更内容をコミットします。

変更内容は、セッションを終了するたびにコミットすることもできます。Warehouse Builder を終了する際に変更内容をコミットするには、「コミット確認」ダイアログが表示された時に「はい」をクリックします。

マルチユーザーのサポート

複数のユーザーが同時に Warehouse Builder Design Repository にアクセスできます。Warehouse Builder では、オブジェクトへの書込み権限は 1 人のユーザーのみが保持し、他のユーザーは読取り専用でアクセスします。あるユーザーがオブジェクトに書込みアクセスを行った後は、トランザクション中は、そのオブジェクトは Warehouse Builder によってロックされます。Warehouse Builder がロックを解除するのは、ユーザーが変更内容をコミットするかロールバックを実行して、そのオブジェクトに関連するエディタをすべて閉じた時点です。

> **注意：** Warehouse Builder セッションのどの時点でコードの検証、生成あるいは配布を行っても、マルチユーザー・ロックは、変更内容をコミットするかロールバックするまで有効です。

エディタ、プロパティ・シートまたはオブジェクトのダイアログを開くときはオブジェクトをロックします。Warehouse Builder 階層のオブジェクトにアクセスすると、フォルダ使用中ロックを取得します。このロックによって、他のユーザーが階層内の上位オブジェクトを削除できないようになります。たとえば、あるモジュールでオブジェクトを編集しているとき、他のユーザーはそのモジュールを削除できません。

読込み / 書込みモード

あるオブジェクトに対して読込み / 書込みモードを入力し、そのオブジェクトをロックするには、エディタ、プロパティ・シートまたはダイアログを開きます。デフォルトでは、オブジェクトへのアクセスは読込み / 書込みモードで行います。

図 2-4 に示すように、エディタ、プロパティ・シートおよびダイアログのタイトル・バーでは、オブジェクトが読込み / 書込みモードであることが示されます。

図 2-4 タイトル・バーの読込み / 書込みインジケータ

オブジェクトに対する読込み / 書込みモードを終了し、オブジェクトのロックを他のユーザーのために解除するには、次のいずれかの操作を行います。

- オブジェクトのエディタを閉じて、変更をコミットまたはロールバックします。
- 使用中のプロパティ・シート、エディタ、ウィザードを、変更せずに閉じるか取り消します。
- Warehouse Builder を終了します。

読取り専用モード

他のユーザーによってロックされているオブジェクトを開こうとすると、要求を取り消すか、読取り専用モードでそのオブジェクトにアクセスするよう求めるプロンプトが表示されます。

読取り専用モードの継続を選択すると、タイトルバーに「読取り専用」と表示されます。エディタ・キャンバスでのオブジェクトの移動、オブジェクト・アイコンの展開および縮小を行うことができます。読取り専用モードでは、オブジェクトを編集できません。読取り専用モードでオブジェクトを編集しようとすると、エラー・メッセージのダイアログが表示されます。読取り専用モードで開いたプロパティ・シートでは、「OK」ボタンが無効化されています。

同期

Warehouse Builder のオブジェクトが、あるユーザーによってロックされた場合、変更をコミットした後であれば、他のユーザーは同期化を使用して変更内容を確認できます。

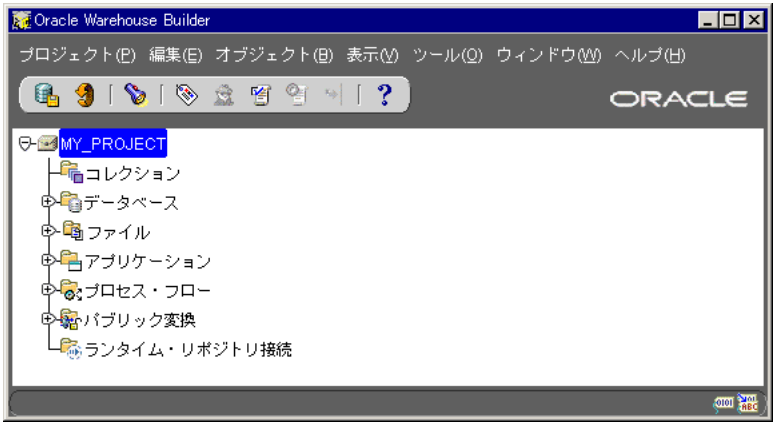
Warehouse Builder では、次のタイプの同期を利用できます。

- **自動同期 :** 読込み / 書込みモードでオブジェクトにアクセスすると、オブジェクトは自動的に同期化されます。
- **コマンドでの同期 :** 「プロジェクト」メニューで「このビューをリポジトリと同期化」を選択するか [F5] を押すと、オブジェクトを同期化できます。この方法では、自動同期では表示されないオブジェクトの変更が示されます。ツールバーの「このビューをリポジトリと同期化」ボタンも使用できます。これはツリーの同期化や、読取り専用モードでの同期化に便利です。

コンソールの概要

Warehouse Builder Design Repository にログオンすると、図 2-5 に示すように、リポジトリの内容がナビゲーション・ツリーとしてコンソールに表示されます。Warehouse Builder コンソールは、複数のコンポーネントで構成されています。

図 2-5 Warehouse Builder コンソール



Warehouse Builder コンソールは、次の主要なコンポーネントで構成されています。

- プロジェクト・ナビゲーション・ツリー
- ツールバー

プロジェクト・ナビゲーション・ツリー

Warehouse Builder では、図 2-6 に示すように、ナビゲーション・ツリー内のプロジェクトに、オブジェクトを整理できます。ナビゲーション・ツリーはファイル・システムと同じように、すべてのオブジェクトが展開可能なフォルダに整理されています。リポジトリ内に複数のプロジェクトが存在する場合、すべてのプロジェクトは、最初はフォルダが閉じた状態で表示されます。一度に開けるプロジェクトは 1 つのみです。プロジェクトを切り替える場合には、コミットもしくはロールバックする必要があります。

図 2-6 ナビゲーション・ツリー

表 2-2 では、ナビゲーション・ツリーの主なフォルダを説明しています。これらのフォルダを開いて内容を表示し、オブジェクトを作成できます。

表 2-2 ナビゲーション・ツリー・フォルダ

フォルダ	説明
コレクション	プロジェクト内にあるオブジェクトのグループ。コレクションを使用してプロジェクトの内容を整理し、Warehouse Builder 転送ウィザードを使用して、他のツールにメタデータをエクスポートします。
データベース	すべてのデータベース・オブジェクトを含みます。これらのオブジェクトはモジュール内に格納されています。モジュールには、ソースとターゲットどちらかのデータベース・オブジェクトが含まれています。一番高いレベルでは、データベースは Oracle と Oracle 以外に分かれます。
ファイル	モジュール内のファイルベースのデータ・オブジェクトが含まれます。
アプリケーション	SAP などのパッケージ・アプリケーションで使用するアプリケーションベースのデータ・オブジェクトが含まれます。オブジェクトはモジュールとしてグループ化されます。
プロセス・フロー	モジュールとしてグループ化されたプロセス・フローが含まれます。
パブリック変換	パブリック変換ライブラリと事前定義変換済ライブラリが含まれます。
ランタイム・リポジトリ接続	Runtime Repository の接続情報が含まれます。

ツールバー

ツールバーは、コンソール上部のメニュー・バーの下にあります。ツールバーには、頻繁に使用される Warehouse Builder タスクへのショートカットがあります。表 2-3 で、ツールバーに表示される順にショートカットを一覧表示します。アイコンをクリックしてタスクを実行します。ツールバー・タスクはすべて、メニューバーからも実行できます。表 2-7 が Warehouse Builder コンソール・ツールバーです。

図 2-7 ツールバー

表 2-3 で、Warehouse Builder コンソール・ツールバーに用意されているアイコンとその機能を説明します。

表 2-3 ツールバー・アイコン

アイコン	タスク	説明
	コミット	データベースの変更をコミットします。このアイコンはいつでも選択できます。
	このビューをリポジトリと同期化	ナビゲーション・ツリーに表示されたオブジェクトと、リポジトリ内のオブジェクトを同期化します。この操作は、マルチユーザー環境での作業に役立ちます。コンソールに表示されたオブジェクトを更新して、他のユーザーがコミットした変更内容を反映させます。このアイコンはいつでも選択できます。
	検索	現在開かれているプロジェクト内でオブジェクトを検索します。オブジェクト検索ダイアログが表示されます。オブジェクトの名前または名前の一部を入力できます。このアイコンはいつでも選択できます。
	プロパティ	ナビゲーション・ツリーでオブジェクトを選択し、このアイコンをクリックすると、そのオブジェクトのプロパティ・シートが表示されます。このアイコンは、選択したオブジェクトにプロパティが存在する場合にのみ使用できます。フォルダまたはラベルを選択した場合はグレー表示されます。
	構成	オブジェクトを選択し、このアイコンをクリックします。構成パラメータ・ダイアログが表示され、パラメータの定義や編集ができます。このアイコンは、選択したオブジェクトに構成パラメータが存在する場合にのみ使用できます。フォルダまたはラベルを選択した場合はグレー表示されます。

表 2-3 ツールバー・アイコン（続き）

アイコン	タスク	説明
	検証	オブジェクトまたはオブジェクト・セットを選択し、このアイコンをクリックします。Warehouse Builder では、オブジェクトが検証され、検証結果が表示されます。このアイコンは、選択したオブジェクトが検証可能な場合にのみ使用できます。
	生成	オブジェクトまたはオブジェクト・セットを選択し、このアイコンをクリックします。Warehouse Builder では、オブジェクトが生成され、生成結果が表示されます。このアイコンは、選択したオブジェクトが生成可能な場合にのみ使用できます。
	配布	オブジェクトまたはオブジェクト・セットを選択し、このアイコンをクリックします。Warehouse Builder では、配布の Runtime Repository 接続を選択していない場合に、選択を求めるプロンプトが表示されます。このアイコンは、選択したオブジェクトが配布可能な場合にのみ使用できます。
	ヘルプ	このアイコンをクリックすると、オンライン・ヘルプ・ナビゲーターに『Oracle Warehouse Builder ユーザーズ・ガイド』が表示されます。このアイコンはいつでも選択できます。

作業環境の設定

Warehouse Builder には、クライアント環境の構成に使用できる作業環境のセットがあります。作業環境を開くには、メニュー・バーの「プロジェクト」から「作業環境」を選択します。

表 2-4 では、「作業環境」ウィンドウに表示される作業環境のタイプを説明します。詳細は、次の項で説明します。

表 2-4 作業環境

作業環境のタイプ	説明
一般	カラー環境および各ウィザードの「ようこそ」ページの作業環境を、確認または定義します。ロケールと MLS 言語の定義にも使用します。
ネーミング	ネーミング作業環境の設定に使用します。
メッセージ・ログ	ファイルのロケーション、サイズおよびログ・ファイルに保存するメッセージのタイプなど、ログ・ファイルのオプション設定に使用します。ログ・ファイルには、設計プロセスに関連するメッセージが格納されています。
ユーティリティ	選択したユーティリティについての情報を、「ツール」メニューに表示するのに使用します。

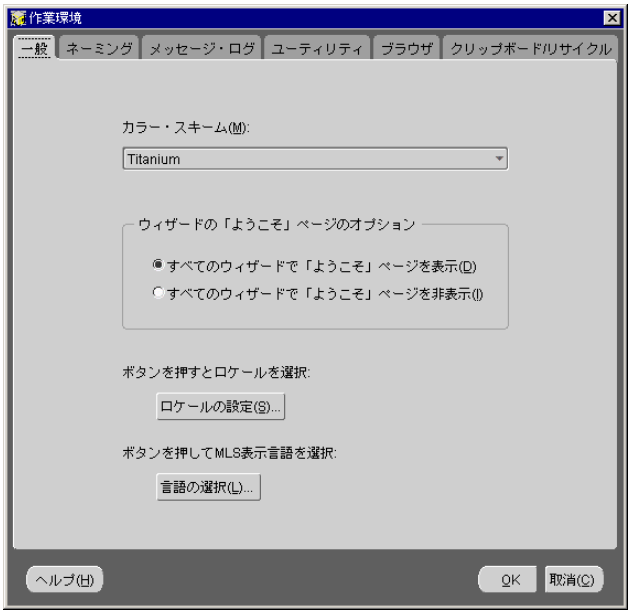
表 2-4 作業環境（続き）

作業環境のタイプ	説明
ブラウザ	Warehouse Builder Design Browser をホスティングしている Web サーバーへの接続定義に使用します。これで、クライアント内からメタデータ・レポートにアクセスできるようになります。
クリップボード / リサイクル	ごみ箱とクリップボードの作業環境設定に使用します。

一般的な作業環境

「一般」タブでは、カラー・スキームを定義し、各ウィザードの「ようこそ」ページの作業環境を定義します。さらに、ロケールの設定と表示言語の選択もできます。

図 2-8「作業環境」の「一般」タブ



ドロップダウン・メニューから Warehouse Builder コンソールのカラー・スキームを選択します。デフォルトはチタン色です。

該当するラジオ・ボタンを選択し、ウィザードの「ようこそ」ページの表示 / 非表示を設定します。Warehouse Builder のウィザードはすべて、そのタスクを完了するまでの手順を要約した「ようこそ」ページで始まります。

- **すべてのウィザードで「ようこそ」ページを表示 :** Warehouse Builder でウィザードを開始したときには、常に「ようこそ」ページを表示します。
- **すべてのウィザードで「ようこそ」ページを非表示 :** Warehouse Builder でウィザードを開始したときには、常に「ようこそ」ページをスキップします。

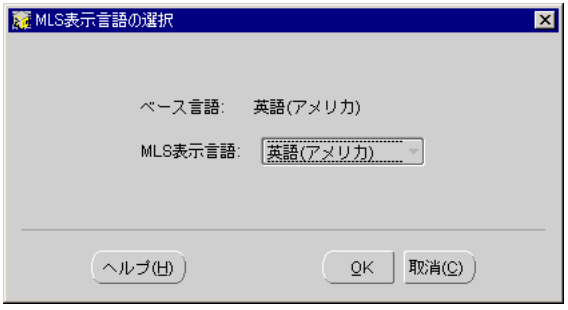
「ロケールの設定」をクリックして、ドロップダウン・リストからクライアントのテキストに表示する言語を選択します。この選択はリポジトリのキャラクタ・セット定義には適用されません。この選択が影響するのは、クライアントのユーザー・インタフェースのテキストおよびメニュー・オプションのみです。リポジトリのキャラクタ・セットはデータベースで定義されます。新しい言語を使用するためにコンピュータの再起動を求めるプロンプトが表示されます。

図 2-9 ロケールの設定

「言語の選択」をクリックして、Warehouse Builder で作成するオブジェクトのビジネス名と説明を編集するときに使用する言語を変更します。ビジネス名は、ビジネス・ネーミング・モードで作成したオブジェクトに割り当てる名前です。MLS 表示言語を変更した後に変更できるのは、既存のオブジェクトのビジネス名と説明のみです。新しいオブジェクトを作成するには、MLS 言語をベース言語に戻す必要があります。

この機能を使用するには、リポジトリのインストール中またはアップグレード中に、サポートされる言語を定義しておく必要があります。詳細は、『Oracle Warehouse Builder 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。

図 2-10 MLS 言語の設定



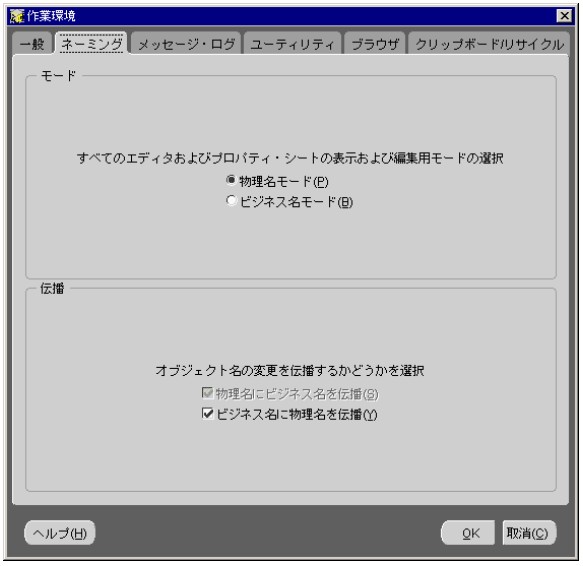
- **ベース言語 :** Warehouse Builder Design Repository の言語です。
- **MLS 表示言語 :** Warehouse Builder 内のオブジェクトのビジネス名と説明を編集または表示するときに使用する言語を選択します。使用できる言語の一覧は、リポジトリの作成またはアップグレード時に選択した言語になります。

MLS の詳細は、『Oracle Warehouse Builder 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。

ネーミング作業環境

このページでは、ビジネス名モードと物理名モードのどちらでオブジェクトを表示するかを選択して、ネーミング作業環境の設定ができます。また、オブジェクト名の変更を伝播する方法も設定できます。

図 2-11「作業環境」の「ネーミング」タブ



ネーミング・モードについて

Warehouse Builder では、リポジトリに格納されているオブジェクトごとにビジネス名と物理名が保持されます。ビジネス名は、オブジェクトの記述的な論理名です。

名前付きオブジェクトの DDL スクリプトを生成するときは、物理名が使用されます。物理名は、『Oracle Database SQL リファレンス』で定義されている基本要素の構文規則に準拠している必要があります。

次に示すように、名前はそれぞれのカテゴリ内で一意にする必要があります。

- モジュール名は、プロジェクト内で一意にする必要があります。
- ウェアハウス・オブジェクト名は、ウェアハウス・モジュール内で一意にする必要があります。これには、表、ディメンション、キューブ、マッピング、マテリアライズド・ビュー、順序、ビュー、索引の名前などがあります。
- 変換名は、変換パッケージ内で一意にする必要があります。

ビジネス名モード Warehouse Builder でビジネス名モードが選択されている場合は、オブジェクトのビジネス名を新規に作成することも、既存のオブジェクトのビジネス名を変更することもできます。ビジネス名モードでは、Warehouse Builder のエディタ、ウィザードおよびプロパティ・シートにオブジェクトのビジネス名が表示されます。

ビジネス名は、次の規則に準拠している必要があります。

- 名前は 4000 文字以内で指定します。
- 名前は、それぞれのカテゴリ内で一意にする必要があります。
- すべてのソース・モジュールは、インポートしたソースの大 / 小文字区別を反映し、『Oracle Database SQL リファレンス』で定義されている二重引用符の規則に従います。
- マッピングにおけるソースからターゲットへのコピー操作では、大 / 小文字区別は伝播されません。

ビジネス名を作成すると、そのビジネス名に類似した有効な物理名が生成されます。既存の物理名と重複するビジネス名を作成すると、一意な名前を作成するために、アンダースコアと番号が追加されます。

物理名モード Warehouse Builder で物理名モードが選択されている場合、オブジェクトの物理名を新規に作成することも、既存のオブジェクトの物理名を変更することもできます。物理名モードでは、Warehouse Builder のエディタ、ウィザードおよびプロパティ・シートにオブジェクトの物理名が表示されます。物理名は大文字に変換されます。

物理名には次の制約があります。

- 30 文字以内で指定します。
- 『Oracle Database SQL リファレンス』で定義されている、スキーマ・オブジェクトの基本的な構文規則に準拠します。

> **注意：** コレクションは 200 文字以内の物理名で指定します。

Warehouse Builder では、無効な物理名の入力が阻止されます。たとえば、重複する名前、文字が多すぎる名前、または予約語の名前は入力できません。

名前モードの設定

オブジェクトのビジネス名を作成または変更するには、ビジネス名モードを選択する必要があります。オブジェクトの物理名を作成または変更するには、物理名モードを選択する必要があります。

Warehouse Builder のデフォルトのネーミング作業環境は、次のとおりです。

- **モード :** 物理名モード
- **伝播 :** ビジネス名に物理名を伝播

名前モードおよび名前伝播設定のアイコンは、エディタの右下隅にあります。これらのアイコンは現在の設定を示しています。

名前モードを設定する手順は次のとおりです。

1. 「ネーミング」タブから、物理名モードまたはビジネス名モードを選択します。

   ネーミング・モードは、セッション中に随時切り替えることができます。

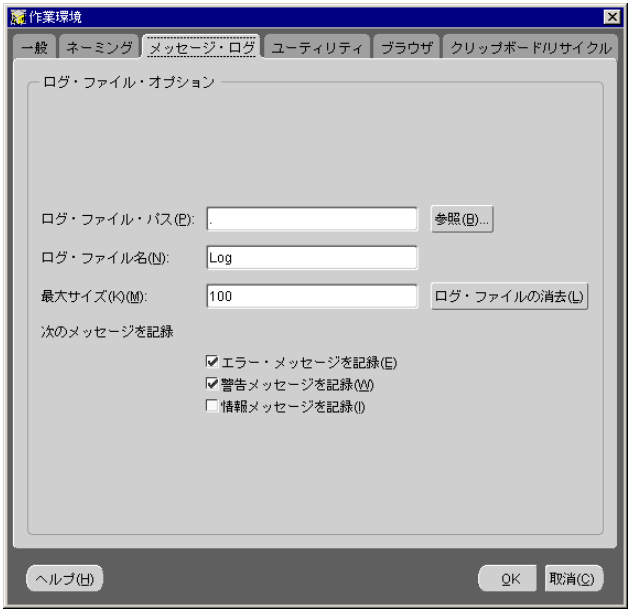
2. 名前の伝播方法を選択します。
3. 「OK」をクリックします。

「ネーミング」タブの設定がセッション全体で保存されます。「ネーミング」タブの設定は、クライアント・ワークステーション上のファイルに格納されます。他のワークステーションから Warehouse Builder を使用すると、異なる設定が表示される場合があります。

メッセージ・ログ作業環境

「メッセージ・ログ」タブでは、ファイルのロケーション、名前、サイズおよびログ・ファイルに保存するメッセージのタイプなど、ログ・ファイルのオプションを設定します。これらのファイルには Warehouse Builder の設計操作とエラーが格納されます。デフォルトでは、メッセージ・ログはデフォルトのロケーションに保存されます。

図 2-12「作業環境」の「メッセージ・ログ」タブ



ログ・ファイル・パス : ログ・ファイルのロケーションを入力するか、「参照」ボタンをクリックしてロケーションを選択します。デフォルトのロケーションは `owbhome¥owb¥bin¥admin` です。

ログ・ファイル名 : ログ・ファイル名を入力します。ファイル拡張子は入力しません。

最大サイズ : ログ・ファイルの最大サイズを指定します。ログ・ファイルは 2 つ作成され、< ログ・ファイル名 >.0 と < ログ・ファイル名 >.1 という名前になります。最初のログ・ファイル < ログ・ファイル名 >.0 が最大サイズに到達すると、2 番目のログ・ファイル < ログ・ファイル名 >.1 への書込みが開始されます。2 番目のログ・ファイルが最大サイズに到達すると、最初のログ・ファイルが上書きされます。

ログ・ファイルの消去 : ログ・ファイルの内容を消去するには、このボタンを使用します。

次のオプションを選択して、取得するメッセージのタイプを指定します。

- **エラー・メッセージを記録 :** ログ・ファイルにすべてのエラー・メッセージを書き込みます。
- **警告メッセージを記録 :** ログ・ファイルにすべての警告メッセージを書き込みます。
- **情報メッセージを記録 :** ログ・ファイルにすべての情報を書き込みます。

ユーティリティ作業環境

「ユーティリティ」タブでは、ユーティリティの追加、更新、削除を行います。ユーティリティは「ツール」メニューから使用します。

図 2-13「作業環境」の「ユーティリティ」タブ

ユーティリティの追加

「ツール」メニューにユーティリティを追加する手順は次のとおりです。

1. 「ユーティリティ」タブを選択して、フィールドに次の情報を入力します。

   ユーティリティの名前。

   ユーティリティのロケーション。「参照」をクリックし、プログラムのフォルダとファイルを検索します。

   ユーティリティのアイコンのロケーション。「参照」をクリックし、アイコンのフォルダとファイルを検索します。

   ユーティリティの説明。

2. 「追加」をクリックします。
3. 「OK」をクリックします。

ユーティリティの更新

ユーティリティの構成を更新する手順は次のとおりです。

1. 「ユーティリティ」シートで、「ユーティリティ・ドローワの内容」リストからユーティリティ名を選択します。

   構成情報が表示されます。

2. 次の構成詳細を変更します。

   ユーティリティの名前。

   ユーティリティのロケーション。「参照」をクリックし、プログラムのフォルダとファイルを検索します。

   ユーティリティのアイコンのロケーション。「参照」をクリックし、アイコンのフォルダとファイルを検索します。

   ユーティリティの説明。

3. 「更新」をクリックします。
4. 「OK」をクリックします。

ユーティリティの削除

「ツール」メニューからユーティリティを削除する手順は次のとおりです。

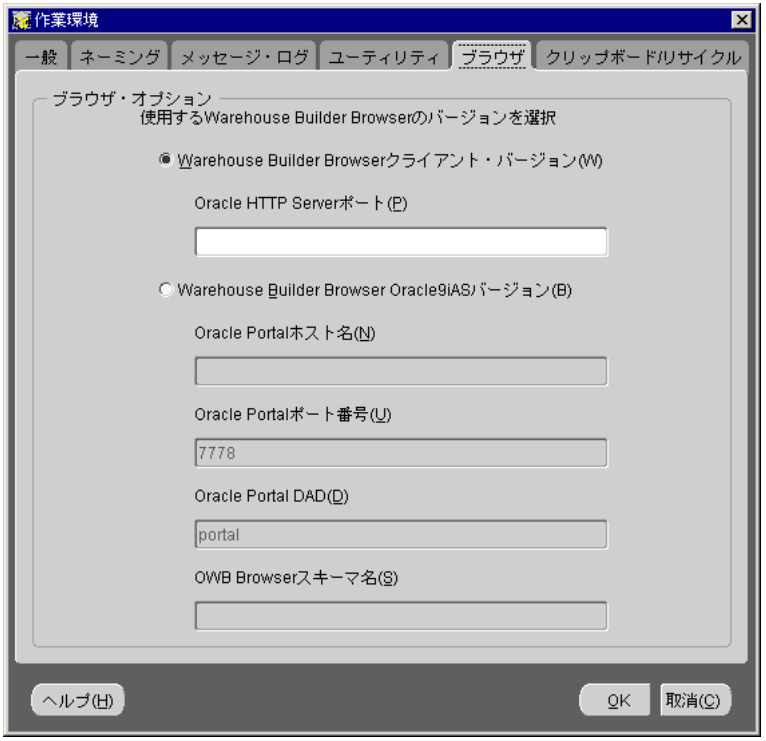
1. 「ユーティリティ」シートで、「ユーティリティ・ドローワの内容」リストからユーティリティ名を選択します。
2. 「削除」をクリックします。
3. 「OK」をクリックします。

   「ツール」メニューからユーティリティが削除されます。

ブラウザ作業環境

「ブラウザ」タブでは、クライアント・コンソール内からメタデータ・レポートを呼び出す際に使用する Warehouse Builder Browser のバージョンを選択します。選択できるバージョンは、クライアントと Oracle9*i*AS です。

図 2-14「作業環境」の「ブラウザ」タブ



クライアント・バージョンを使用する場合

Warehouse Builder Browser のクライアント・バージョンを選択した場合、Oracle HTTP Server Port を指定する必要があります。これには、アクセス先の Warehouse Builder Design Repository と同じマシンで HTTP Server を実行する必要があります。Oracle HTTP Server Port は、DBA に連絡して入手してください。

OracleAS を使用する場合

Oracle Portal バージョンを選択した場合、OracleAS に付属してインストールされている Oracle Portal との接続を確立する必要があります。このオプションを使用するには、次の接続詳細を入力します。

Oracle Portal ホスト名 : Oracle Portal がインストールされているシステムの名前を入力します。

Oracle Portal ポート番号 : デフォルト : 7778。

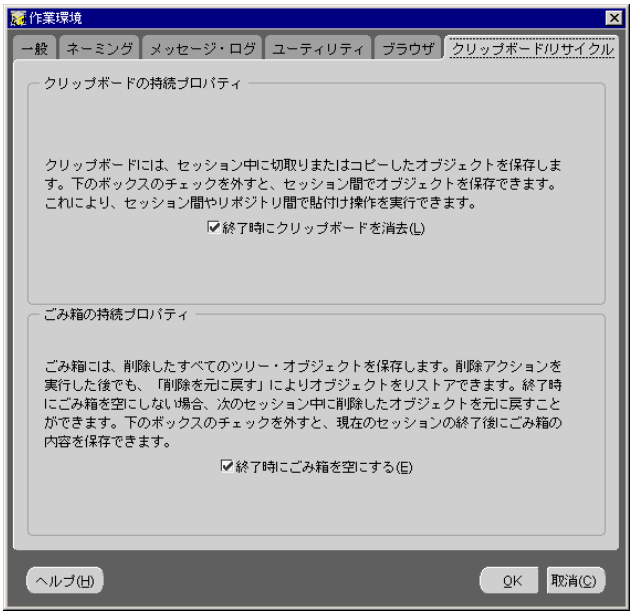
Oracle Portal DAD: Oracle Portal のインストール中に選択されるポータル・データベース・アクセス記述子。

OWB Browser スキーマ名 : Browser Assistant を使用して Warehouse Builder Browser をインストールした際に指定されたユーザー名。

クリップボード / ごみ箱の作業環境

「クリップボード / リサイクル」タブでは、クリップボードおよびごみ箱からオブジェクトが削除されるタイミングを設定します。

図 2-15「作業環境」の「クリップボード / リサイクル」タブ



クリップボードの持続プロパティ

クリップボードのオブジェクトをセッション間で保持するには、「終了時にクリップボードを消去」チェック・ボックスの選択を解除する必要があります。

クリップボードには、Warehouse Builder ナビゲーション・ツリーから最後に切り取ったオブジェクトまたは最後にコピーしたオブジェクトが保存されます。クリップボードは Warehouse Builder Design Client に保存されます。この作業環境を設定して、オブジェクトの格納期間を、現在のセッション中だけにするのか、このクライアント・マシンを使ったセッションすべてにするのかを指定します。デフォルト設定では、セッションをログアウトするたびにクリップボードは空になります。

ゴミ箱の持続プロパティ

ごみ箱のオブジェクトをセッション間で保持するには、「終了時にごみ箱を空にする」チェック・ボックスの選択を解除する必要があります。

ごみ箱には、Warehouse Builder ナビゲーション・ツリーから削除したオブジェクトがすべて保存されます。ごみ箱は Warehouse Builder Design Client に保存されます。この作業環境を設定して、オブジェクトの格納期間を、現在のセッション中だけにするのか、このクライアント・マシンを使ったセッションすべてにするのかを指定します。デフォルト設定では、セッションをログアウトするたびにごみ箱は空になります。

ごみ箱内の削除済オブジェクトが持続するよう選択すると、別のセッション中にそれらのオブジェクトをリポジトリにリストアできます。ごみ箱内のオブジェクトは、「コミット」および「ロールバック」のアクションの影響を受けません。

Warehouse Builder オブジェクトの作成と編集

Warehouse Builder を使用する際、最初に行う作業はプロジェクトの作成です。プロジェクトを作成したら、その他の Warehouse Builder オブジェクトをすべて作成できます。Warehouse Builder には、ウィザード、オブジェクト・エディタ、プロパティ・シート、オブジェクト検索ツールがあり、ビジネス・インテリジェンス・システムの設計に役立ちます。これらのコンポーネントは設計プロセスで役立ちます。

プロジェクトの作成

Warehouse Builder Design Repository をインストールすると、事前定義済のパブリック変換セットとともに、MY_PROJECT というデフォルト・プロジェクトが作成されます。デフォルト・プロジェクトは、最初のログオン・シーケンスに必要です。他のオブジェクトを作成した後、MY_PROJECT を削除できます。Warehouse Builder Design Repository のインストールの詳細は、『Oracle Warehouse Builder 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。

新規プロジェクトの作成

Warehouse Builder の使用を開始する際には、設計作業を管理するために新しいプロジェクトを作成します。

プロジェクトを新規作成する手順は次のとおりです。

1. 「プロジェクト」メニューから「プロジェクトの作成」を選択します。

   「新規プロジェクト・ウィザード: ようこそ」ページが表示されます。

2. 「次へ」をクリックします。

   「名前」ページが表示されます。

3. プロジェクトの名前と説明（オプション）を入力し、「次へ」をクリックします。

   「バージョン・プロパティ」ページが表示されます。

4. バージョンのラベルを入力し（オプション）、「終了」をクリックします。

   新規プロジェクトが作成され、ナビゲーション・ツリーの下部に追加されます。Warehouse Builder では、暗黙的にコミットされ、Design Repository に新規プロジェクトが追加されます。

   これで、プロジェクト内にオブジェクトを作成するために、ウィザードを使用できるようになりました。開始するには、プロジェクトを開いてオブジェクト・タイプを選択します。

プロジェクトの削除

プロジェクトの削除は、Warehouse Builder の他のオブジェクトを削除するより複雑です。プロジェクトは Warehouse Builder の主要な設計コンポーネントなので、プロジェクトを削除しないように、この機能が設計されました。

このとき、次のガイドラインが適用されます。

- 現在アクティブなプロジェクトまたは開かれているプロジェクトは削除できません。
- リポジトリ内に 1 つだけ残ったプロジェクトは削除できません。
- 各 Design Repository には、最低 1 つのプロジェクトが格納されている必要があります。

プロジェクトを削除する手順は次のとおりです。

1. ナビゲーション・ツリー内のプロジェクトのうち、削除しないものを選択して開きます。

   開いたプロジェクトは、現在アクティブなプロジェクトになります。

2. 削除するプロジェクトを選択します。ただし開かないでください。

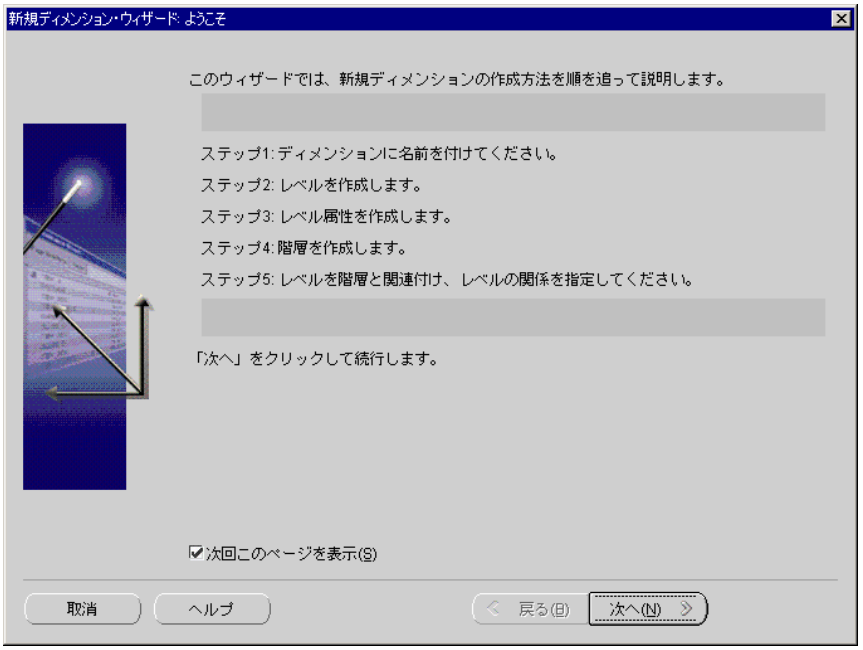
3. 「編集」メニューから「削除」を選択します。プロジェクトを右クリックし、「削除」を選択することもできます。

   「削除確認」ダイアログが表示されます。オブジェクトを削除するか、ごみ箱に入れるか選択できます。プロジェクトの新規作成の場合と違って、プロジェクトの削除には暗黙的コミットがされません。

Warehouse Builder ウィザードの使用

Warehouse Builder では、ウィザードを使用してすべてのオブジェクトを作成できます。ウィザードを起動するには、オブジェクト・タイプを右クリックして「作成」を選択します。表 2-16 に示すように、各ウィザードの最初のページは「ようこそ」ページです。このページには、ウィザードのページ番号が示され、各ページで実行する作業の説明が表示されます。

図 2-16 ウィザードの「ようこそ」ページ



「次回このページを表示」チェック・ボックスの選択を解除すると、「ようこそ」ページをスキップするようウィザードを設定できます。「ようこそ」ページを再び有効にするには、「プロジェクト」メニューで「作業環境」ダイアログを開き、「すべてのウィザードで「ようこそ」ページを表示」ボックスを確認します。

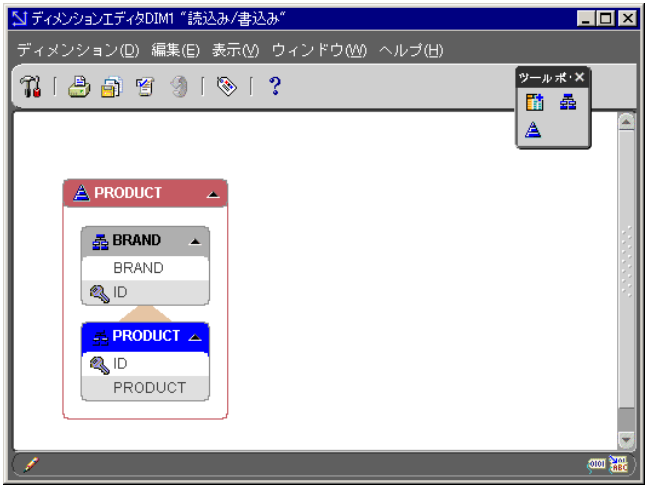
ウィザードのページ間を移動するには、「次へ」および「戻る」ボタンをクリックします。設定内容を保存しないで終了するには、「取消」をクリックします。ページ内を移動するには、マウスまたは [Tab] キーを使用します。長い説明のテキスト・ボックスからカーソルを移動するには、[Ctrl] を押しながら [Tab] を押します。

ウィザードの特定のトピックに関するヘルプを参照するには、ウィザード・ページの下部にある「ヘルプ」をクリックします。

オブジェクト・エディタの使用

モジュールとオブジェクトを作成したら、エディタとプロパティ・シートを使用してこれらのモジュールとオブジェクトを編集できます。オブジェクトのエディタでは、メニュー項目別に独立したウィンドウとオブジェクト編集用のツールボックスを利用できます。エディタを表示するには、オブジェクトを右クリックして「エディタ」を選択します。図 2-17 は、ディメンション・エディタの例を示しています。

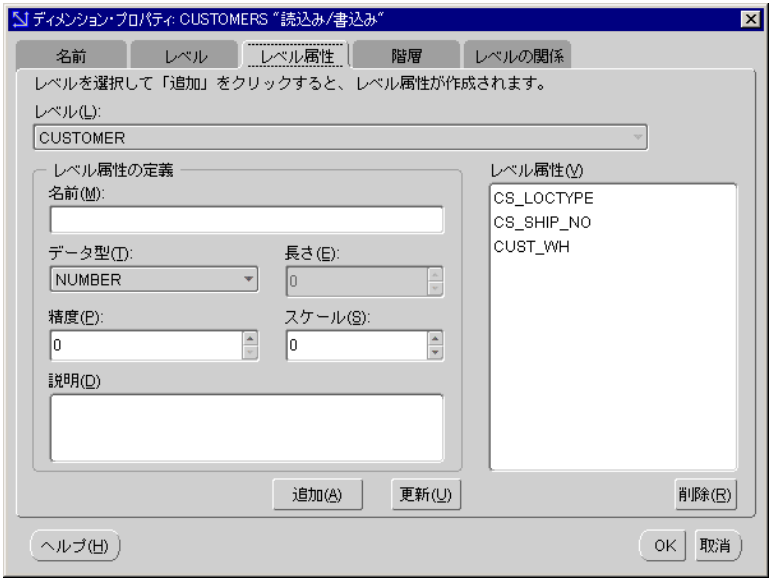
図 2-17 ディメンション・エディタ



プロパティ・シートの使用

コンソール・ツールバーから「プロパティ」アイコンを選択すると、選択したオブジェクトのプロパティ・シートが表示されます。プロジェクトで保存されている各オブジェクトは、そのオブジェクトを定義するプロパティ・シートを備えています。プロパティ・シートを編集して、格納されたオブジェクト定義を更新できます。プロパティ・シートはオブジェクトによって異なります。図 2-18 は、ディメンションのプロパティ・シートの例を示しています。

図 2-18 ディメンションのプロパティ・シート



ユーザー定義プロパティの作成

ユーザー独自のニーズに応じてメタデータを拡張できるように、Warehouse Builder では、オブジェクトにプロパティを追加作成できます。ユーザー定義プロパティは、ナビゲーション・ツリー上のどのオブジェクト・タイプにも作成できます。ユーザー定義プロパティを作成するには、Oracle MetaBase（OMB）Scripting Language、OMB Plus を使用します。これは、Warehouse Builder に付属のスクリプト・ユーティリティです。ユーザー定義プロパティを作成し、オブジェクトに追加するビジネス、設計またはバージョニング情報を格納します。

ユーザー定義プロパティを作成する場合は、UDP_ という接頭辞を付ける必要があります。このネーミング規則により、ユーザーと Warehouse Builder は、コア・プロパティとカスタマイズ・プロパティを区別できます。ネーミング規則に従わないでプロパティを作成しようとすると、エラーメッセージが表示されて、接頭辞 UDP_ を使用するように指示されます。作成したプロパティの表示と移入は、オブジェクトに対応するユーザー・インタフェース内のプロパティ・シートで、または直接 OMB Plus から行えます。

たとえば UDP_BUSINESS_PURPOSE というプロパティを表に追加し、各表のビジネス上の目的を記述できるようにすると、どの表のプロパティ・シートを開いても、プロパティはユーザー・インタフェースに「UDP Business Purpose」と表示されます。プロパティにビジネス上の目的を移入する作業は、ユーザー・インタフェースのプロパティ・シートで、または直接 OMB Plus から行えます。

ヒント： データ・オブジェクトを定義する前に、作成予定のユーザー定義プロパティを完成するようにしてください。こうしておくと、作業に沿ってユーザー定義プロパティを移入できます。後で設計プロセスにさかのぼってオブジェクトを編集する必要はありません。

ユーザー定義プロパティに関連するスクリプト・コマンドと引数は、『Oracle Warehouse Builder スクリプト・リファレンス』を参照してください。

プロジェクト内のオブジェクトの検索

Warehouse Builder Design Repository に格納されているオブジェクトは、ナビゲーション・ツリーによって編成されます。リポジトリ内のオブジェクトを検索するには、「検索」をクリックして、名前または名前の一部を入力します。1 つ以上の文字に一致するワイルド・カード文字としてアスタリスク（*）を使用します。図 2-19 はオブジェクト検索ダイアログです。

図 2-19 オブジェクト検索ダイアログ

メタデータの管理

Warehouse Builder では、設計したターゲット・システムのメタデータはすべて Warehouse Builder Design Repository で格納されます。Warehouse Builder の使用を開始する前に、リポジトリをインストールしておく必要があります。Warehouse Builder を起動すると、Design Repository の内容がナビゲーション・ツリーに表示されます。メタデータを管理できるように、Warehouse Builder では、系統および影響分析などのメタデータに関するレポートの実行、リポジトリ間でのメタデータのエクスポートおよびインポート、ツール間でのメタデータの交換ができます。

メタデータのレポート

Warehouse Builder Design Browser を使用して、Design Repository に格納されたメタデータすべてのレポートにアクセスします。メタデータ・レポートには、Warehouse Builder Design Client 内からアクセスすることも、クライアントをインストールせずに、Oracle Portal からアクセスすることもできます。詳細は、第 17 章「メタデータの参照およびレポート」を参照してください。

メタデータのインポートとエクスポート

Metadata Loader（MDL）を使用して、新規リポジトリへの移入を行います。既存のリポジトリ・メタデータのバックアップの転送、更新、リストアもできます。MDL ユーティリティを使用すると、ナビゲーション・ツリー上のどのタイプのオブジェクトが対象でも、メタデータのインポートとエクスポートができます。MDL には、Warehouse Builder Design Client と OMB Plus スクリプト・インタフェースのどちらからもアクセスできます。詳細は、15-2 ページの「Metadata Loader を使用したメタデータのインポートおよびエクスポート」を参照してください。

メタデータの交換

コレクションを作成し、Warehouse Builder 転送ウィザードを使用して、Warehouse Builder Design Repository 内に格納されたメタデータを交換します。

コレクションの定義

コレクションとはプロジェクト内のオブジェクトのグループで、Warehouse Builder でオブジェクト整理のために定義できます。コレクションを作成する際に、プロジェクト内にすでに存在しているオブジェクトを指すショートカットを作成します。これらのショートカットにより、ベース・オブジェクトにすぐアクセスでき、変更もすばやく行えます。

Warehouse Builder 転送ウィザードの使用

Warehouse Builder 転送ウィザードを使用すると、様々なビジネス・インテリジェンス・ツール内の Design Repository に格納されたメタデータを同期化、統合および使用できます。ブリッジを使用して、メタデータのインポートおよびエクスポートができます。また、メタデータは、OMG ファイル、Oracle Discoverer、Oracle Express、ERwin、Predesignate および Oracle OLAP Server と交換することもできます。転送ウィザードの使用方法とこのトピックの詳細は、第 22 章「その他の BI 製品と Warehouse Builder のメタデータの統合」を参照してください。

第 I 部

ソース・オブジェクトとターゲット・オブジェクトの定義

この部には、次の章があります。

3

Oracle データ・オブジェクトの定義

データ・ウェアハウスまたはデータ・マートの要件収集が完了すると、Warehouse Builder を使用してターゲット・システムを設計できるようになります。ほとんどのターゲット・スキーマ・モデリングは、Oracle ウェアハウス・モジュール内で行われます。この章では、Oracle ウェアハウス・モジュールの作成方法と、そのモジュール内でのデータ・オブジェクトの定義方法について説明します。

この章では、次のトピックについて説明します。

ウェアハウス・モジュールの作成

Warehouse Builder では、ターゲット・スキーマの定義がウェアハウス・モジュールに保存されます。これらの定義を作成するには、Warehouse Builder のウィザードを使用するか、外部のソースから定義をインポートします。この項では、ウェアハウス・モジュールの作成方法について説明します。次の項では、ウェアハウス・モジュールでデータ・オブジェクトの定義を作成およびインポートする方法について説明します。

ウェアハウス・モジュールを作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「データベース」ノードを開きます。
2. Oracle データ・ソースのソース・モジュールを作成するには、「Oracle」ノードを右クリックし、「Oracle モジュールの作成」を選択します。
3. 「新規モジュール・ウィザード : ようこそ」ページが表示されたら、「次へ」をクリックします。

   「新規モジュール・ウィザード : 名前」ページが表示されます。
4. 次の情報を指定します。

   モジュールの名前。

   モジュールのステータス : 選択できるステータスは「開発」、「品質保証」または「製品」です。これは、説明の目的でのみ使用されます。

   モジュール・タイプとして「ウェアハウス・ターゲット」を指定します。

   説明（オプション）。

   「次へ」をクリックします。
5. 「新規モジュール・ウィザード : 接続情報」ページでは、ウェアハウス・モジュールにメタデータをインポートするために必要なデータベース・リンク情報を指定できます。このページをスキップし、メタデータをインポートするときにこの情報を指定することもできます。

   まず、メタデータ・ソースを選択します。

   Oracle データ・ディクショナリ : Oracle データベースからメタデータをインポートする場合に選択します。

   Oracle Designer リポジトリ : Oracle Designer リポジトリからメタデータをインポートする場合に選択します。

   「データベース・リンク」フィールドで、すでに作成されているデータベース・リンクの一覧から選択します。または、次の情報を指定して新しいデータベース・リンクを作成します。

   所有者 : このデータベース・リンクを作成するソース・データベース・ユーザー。

ユーザー名 : データベース・リンクを使用してソース・データベースにアクセスするユーザーの名前。

接続文字列 : ソース・データベースが常駐するシステムの名前。

スキーマ : ソース・データベースが常駐するスキーマの名前。

「次へ」をクリックします。

6. 「新規モジュール・ウィザード : ロケーション」ページでは、ドロップダウン・メニューからロケーションを選択するか、「新規」をクリックして新しいロケーションを作成できます。

   オブジェクトの配布先となるデータベース・スキーマまたはターゲット・ツールについての情報が、ロケーションに定義されます。ロケーションは、Oracle データベース、SAP、フラット・ファイルなど、モジュールのタイプに固有です。詳細は、3-3 ページの「ロケーションの定義」を参照してください。

   この手順はオプションです。このモジュールのロケーションは、後でオブジェクトを配布する際に作成することもできます。

   「次へ」をクリックします。

7. ウィザードの各ページで指定した情報をまとめた「新規モジュール・ウィザード : 終了」ページでは、メタデータ・インポート・ウィザードを直接起動するためのチェックボックスを選択できます。

   「終了」をクリックします。

   新規モジュール・ウィザードによってウェアハウス・モジュールが作成され、その名前がプロジェクトのナビゲーション・ツリーに挿入されます。チェック・ボックスを選択した場合は、メタデータ・インポート・ウィザードが起動します。3-71 ページの「ターゲット・モジュールへのメタデータのインポート」を参照してください。

ロケーションの定義

ロケーションは、特定のデータベース・スキーマとターゲット・ツールを表します。これは、Oracle データベース、Oracle 以外のデータベース、SAP、ファイル・システムなどのモジュール・タイプに固有のもので、ナビゲーション・ツリーのこれらのモジュールの下に整理されます。ロケーションを作成すると、ロケーションのタイプとバージョンを含む論理的な定義が格納されます。ロケーションが登録されると、物理的な接続情報が要求され、それが Runtime Repository に格納されます。

プロジェクト内に定義されている各ロケーションは、各 Runtime Repository 内で別々に登録でき、各登録は異なる物理的情報を参照できます。この方法により、ターゲット・システムを一度設計および構成するだけで、異なる物理的特徴を使用してそれを何度も配布できるようになります。これは、1 つのシステムから、開発用、テスト用、本番用などの複数のバージョンを作成する必要がある場合に役立ちます。

モジュールの作成時には、各モジュールのロケーションを指定する必要があります。1 つのロケーションを複数のモジュールに割り当てることはできますが、配布先の個別のデータベース・スキーマまたはツールごとにロケーションを作成する必要があります。Oracle データベース・モジュールの場合は、参照先のモジュールへのコネクタも定義する必要があります。これらのコネクタにより、配布時に、設計されているデータの移動に対して必要に応じてデータベース・リンクが生成される場合があります。Oracle データベース・モジュール内では、ファイル・システムを参照するコネクタを作成することもできます。この種のコネクタは、データベース・ディレクトリとして生成され、外部表によって使用されます。

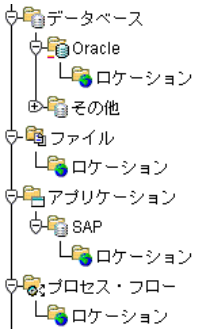
ロケーションの作成

ロケーションを新規作成する手順は次のとおりです。

1. プロジェクトを選択し、ナビゲーション・ツリーを開くとロケーションが表示されます。

   図 3-1 に示すように、ロケーションの主なタイプには、データベース、ファイル・システム、アプリケーションおよびプロセス・フローの 4 つがあります。

図 3-1 ナビゲーション・ツリー上のロケーション



2. ロケーションを作成するモジュール・タイプの下から「ロケーション」ノードを選択します。
3. 「オブジェクト」メニューから「Oracle のロケーションの作成」を選択するか、「ロケーション」を右クリックして「Oracle のロケーションの作成」を選択します。
4. 「新規ロケーション・ウィザード: ようこそ」ページが表示されたら、「次へ」をクリックして続行します。

5. 「新規ロケーション・ウィザード: 名前」ページで、ロケーションについて次の情報を定義します。

   名前: ロケーションの名前を入力します。最大 30 文字です。

   説明（オプション）: ロケーションの説明（オプション）を入力します。最大 400 文字です。

   新しいロケーションに、事前に割り当てられたデフォルト値はありません。

   「次へ」をクリックして続行します。

6. 「新規ロケーション・ウィザード: 詳細」ページで、ロケーションについて次の情報を定義します。

   ロケーション・タイプ: ドロップダウン・リストから、作成するロケーションのタイプを選択します。デフォルトは、ロケーションを作成するモジュールのタイプによって決まります。

   バージョン: 作成するロケーションのタイプのバージョンをドロップダウン・リストから選択します。

   「次へ」をクリックして続行します。

7. 「新規ロケーション・ウィザード: 終了」ページで、新しいロケーションの定義を確認します。このページには、名前、タイプおよびバージョンが一覧表示されています。

   「終了」をクリックして、定義したロケーションを作成します。ロケーションが作成され、該当する「ロケーション」ノードの下に追加されます。

ロケーションの編集

ロケーションを編集するには、ナビゲーション・ツリーでロケーションを右クリックし、「プロパティ」を選択します。図 3-2 に示すように、プロパティ・ウィンドウが表示されます。

図 3-2 ロケーションの「名前」タブ

このページには、選択したロケーションのプロパティが表示されます。次の 2 つのタブを使用して、プロパティを表示および編集します。「OK」をクリックして変更内容を保存するか、「取消」をクリックしてウィンドウを閉じます。

「名前」タブ

「名前」タブには次の情報が表示されます。

- **名前 :** ロケーション名が表示されます。最大 30 文字です。
- **説明 :** ロケーションの説明（オプション）が表示されます。最大 4000 文字です。

「詳細」タブ

「詳細」タブには次の情報が表示されます。

- **ロケーション・タイプ :** 作成するロケーションのタイプが表示されます。デフォルトは、ロケーションを作成するモジュールのタイプによって決まります。
- **バージョン :** 作成するロケーションのタイプのバージョンが表示されます。

コネクタの定義

コネクタは、ナビゲーション・ツリー内の Oracle データベース・モジュールのロケーションとその他の定義済モジュールのロケーションの間の接続を定義します。これらは、ナビゲーション・ツリー内の Oracle データベースの「ロケーション」ノードの下にのみ存在します。コネクタは、あるロケーションから別のロケーションにデータを転送するためのパスが存在することを示します。コネクタは、コネクタを含む Oracle データベース・モジュールによって所有され、別の 1 つのロケーションを参照します。

ナビゲーション・ツリーにコネクタを作成すると、Warehouse Builder Design Repository に論理的な定義が格納されます。コネクタの定義されているロケーション内のオブジェクトが配布されると、必要に応じて、データベース・リンクまたはディレクトリ・オブジェクトが参照される場合があります。2 つの特定のロケーション間の各方向には、コネクタを 1 つしか定義できません。コネクタは、マッピング・エディタ内でもソースとして暗黙的に作成され、ターゲットがキャンバスに配置されます。ソースをキャンバスに配置すると、マッピングのロケーションとソースのロケーションの間にコネクタが存在しない場合、コネクタが自動的に作成されます。このコネクタは、ソースのロケーションからマッピングのロケーションにデータを転送するためのパスが存在することを示します。マッピングのソースにフラット・ファイルを使用した場合、コネクタは自動的に作成されないため、フラット・ファイル・ソースからリレーショナル・ターゲットへのコネクタを、手動で定義する必要があります。

注意： ファイル・システムを参照するコネクタは、データベース・ディレクトリとして実装されます。ディレクトリを作成してドロップするには、「create any directory」権限と「drop any directory」権限が付与されている必要があります。これらは高レベルの権限であり、背後で付与されるものではありません。

コネクタの作成

コネクタを作成する手順は次のとおりです。

1. プロジェクトを選択し、ナビゲーション・ツリーを開くと、Oracle データベース・モジュールがすべて表示されます。
2. 「ロケーション」ノードを開きます。
3. コネクタを作成するロケーションを右クリックします。
4. 「オブジェクト」メニューから「コネクタの作成」を選択するか、ロケーションを右クリックして「コネクタの作成」を選択します。
5. 「新規コネクタ・ウィザード: ようこそ」ページが表示されたら、「次へ」をクリックして続行します。

6. 「新規コネクタ・ウィザード: 名前」ページで、コネクタについて次の情報を定義します。

   名前: コネクタの名前を入力します。最大 30 文字です。

   説明（オプション）: コネクタの説明（オプション）を入力します。最大 400 文字です。

   新しいコネクタに、事前に割り当てられたデフォルト値はありません。

   「次へ」をクリックして続行します。

7. 「新規コネクタ・ウィザード: 詳細」ページで、コネクタの参照ロケーションを指定します。

   データベース: このオプションを選択して、データベースのロケーションにリンクします。ドロップダウン・リストから特定のロケーション名を選択します。

   ファイル・システム: このオプションを選択して、ファイルのロケーションにリンクします。ドロップダウン・リストから特定のロケーション名を選択します。

   アプリケーション: このオプションを選択して、アプリケーションのロケーションにリンクします。ドロップダウン・リストから特定のロケーション名を選択します。

   未指定: このオプションを選択して、未指定タイプのロケーションにリンクします。

   「次へ」をクリックして続行します。

8. 「新規コネクタ・ウィザード: 終了」ページで、新しいコネクタの定義を確認します。このページには、名前と参照ロケーションが一覧表示されています。

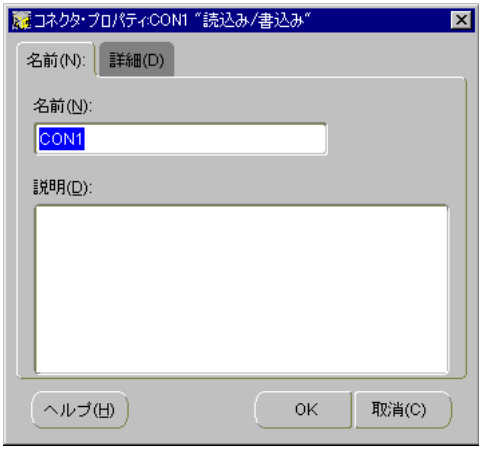
   「終了」をクリックして、定義したコネクタを作成します。

   コネクタが作成され、ロケーションの下に追加されます。必要に応じて、各ロケーションのコネクタ作成を続行します。

コネクタの編集

コネクタを編集するには、ナビゲーション・ツリーでコネクタを右クリックし、「プロパティ」を選択します。図 3-3 に示すように、プロパティ・ウィンドウが表示されます。

図 3-3 コネクタの「名前」タブ



このページには、選択したコネクタのプロパティが表示されます。次の 2 つのタブを使用して、プロパティを表示および編集します。「OK」をクリックして変更内容を保存するか、「取消」をクリックしてウィンドウを閉じます。

「名前」タブ

- **名前 :** ロケーション名が表示されます。最大 30 文字です。
- **説明 :** ロケーションの説明（オプション）が表示されます。最大 400 文字です。

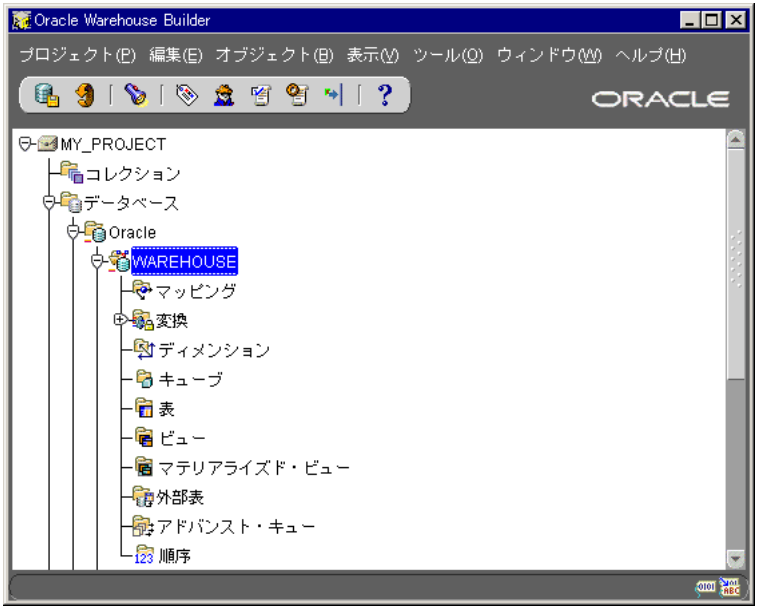
「詳細」タブ

- **データベース :** このオプションを選択して、データベースのロケーションにリンクします。ドロップダウン・リストから特定のロケーション名を選択します。
- **ファイル・システム :** このオプションを選択して、ファイルのロケーションにリンクします。ドロップダウン・リストから特定のロケーション名を選択します。
- **アプリケーション :** このオプションを選択して、アプリケーションのロケーションにリンクします。ドロップダウン・リストから特定のロケーション名を選択します。
- **未指定 :** このオプションを選択して、未指定タイプのロケーションにリンクします。

データ・オブジェクトについて

ウェアハウス・モジュールを作成した後は、「データベース」ノードと「Oracle」ノードを順に開いて、モジュールのロケーションを確認できます。次にウェアハウス・モジュールを開き、図 3-4 に示すように、Warehouse Builder がサポートしているデータ・オブジェクトをすべて表示します。

図 3-4 ウェアハウス・モジュール内のデータ・オブジェクト



Warehouse Builder では、リレーショナル・データ・オブジェクトとディメンション・データ・オブジェクトをサポートしています。リレーショナル・オブジェクトは、リレーショナル・データベースと同様に、表および表から導出されたオブジェクトを使用して、すべてのデータを格納およびリンクします。ユーザーが定義するリレーショナル・オブジェクトは、データベース内の物理的なコンテナで、データの格納に使用されます。ウェアハウス作成後に問合せを実行するのは、このリレーショナル・オブジェクトからです。リレーショナル・オブジェクトには、表、ビュー、マテリアライズド・ビュー、順序などがあります。この章では、各種のリレーショナル・オブジェクトに関する特定の情報を説明し、各オブジェクト・タイプが Warehouse Builder でどのように使用されるかを示します。

ディメンション・オブジェクトには、データを識別し、分類するための追加のメタデータが含まれています。ディメンション・オブジェクトを定義するときには、より構造化した形式でデータを格納しやすくなるように、論理的な関係を記述します。ディメンション・オブジェクトには、ディメンションやキューブなどがあります。この章では、各種のディメンション・オブジェクトに関する特定の情報を説明し、各オブジェクト・タイプが Warehouse Builder でどのように使用されるかを示します。

表 3-1 では、Warehouse Builder で使用できるデータ・オブジェクトのタイプを説明します。

表 3-1 Warehouse Builder 内のデータ・オブジェクト

データ・オブジェクト	タイプ	説明
表	リレーショナル	リレーショナル・データベース管理システムのストレージの基本単位。表を作成した後、有効なデータ行を挿入できます。その後で、表に対して、問合せ、削除、更新を行えます。定義済のビジネス・ルールを表のデータに適用するため、表の整合性制約とトリガーも定義できます。3-12 ページの「表の使用方法」を参照してください。
外部表	リレーショナル	外部表は、非リレーショナルなフラット・ファイルのデータを、リレーショナル形式で表示する表です。外部表は、フラット・ファイル演算子や SQL* Loader のかわりに使用します。3-27 ページの「外部表の使用方法」を参照してください。
ビュー	リレーショナル	ビューは、データの表示を 1 つ以上の表にカスタマイズしたものです。ビューには、データが実際に格納されているわけではありません。データは基となる表から導出されます。制約はいくつかありますが、表と同様、ビューに対しても、問合せ、更新、挿入、削除を行えます。ビューに対して実行した操作はすべて、ビューの実表に影響を与えます。データの表現を簡潔化したり、データへのアクセスを制限するには、ビューを使用します。3-35 ページの「ビューの使用方法」を参照してください。
マテリアライズド・ビュー	リレーショナル	マテリアライズド・ビューは事前に計算された表で、ファクト表、および場合によってはディメンション表から集計または結合されたデータで構成されます。これは、サマリー表や集計表とも呼ばれます。問合せのパフォーマンスを改善するには、マテリアライズド・ビューを使用します。3-40 ページの「マテリアライズド・ビューの使用方法」を参照してください。
順序	リレーショナル	順序は、一意の数字のリストを生成するデータベース・オブジェクトです。一意のサロゲート・キー値を生成するには、順序を使用できます。3-44 ページの「順序の使用方法」を参照してください。

表 3-1 Warehouse Builder 内のデータ・オブジェクト（続き）

データ・オブジェクト	タイプ	説明
アドバンスト・キュー	リレーショナル	アドバンスト・キューにより、アプリケーションの統合に必要なメッセージ管理と通信が可能になります。3-47 ページの「アドバンスト・キューの使用方法」を参照してください。
ディメンション	ディメンション	データ・セットのメンバーの指定に使用する特性を表す一般的な用語です。販売向きのデータ・ウェアハウスにおける最も一般的な 3 つのディメンションは、時間、地理および製品です。ほとんどのディメンションには階層があります。3-55 ページの「ディメンションの使用方法」を参照してください。
キューブ	ディメンション	キューブには、メジャー、および 1 つ以上のディメンション表へのリンクが含まれます。これはファクトとも呼ばれます。3-67 ページの「キューブの使用方法」を参照してください。

表の使用方法

Warehouse Builder では、表はリレーショナル・ストレージ・オブジェクトをメタデータで表したものです。これらの表には、Oracle などのデータベース・システムの表や、SAP システムの表が含まれます。次の項では、Warehouse Builder で表を作成および使用する方法について説明します。

- 表定義の作成（3-12 ページ）
- 表エディタの使用方法（3-16 ページ）
- 表定義の編集（3-17 ページ）

表定義の作成

Warehouse Builder で作成する表には、ターゲット・スキーマのモデル化に使用するメタデータが取り込まれます。この表の定義では、表の制約、索引、および表で使用される列やデータ型に関するメタデータを指定します。この情報は、Warehouse Builder Design Repository に格納されます。後でこれらの定義を使用すると、Warehouse Builder で `.ddl` スクリプトを生成し、ターゲット・データベースに物理表を作成する目的で配布できるようになります。さらに、これらの表には、選択したソース表のデータをロードできます。

表定義を作成するには、新規表ウィザードを使用します。この項では、新規表ウィザードの主なページについて説明します。

注意： マッピング・エディタで表を作成することもできます。

新規表ウィザードを使用した表の作成

表を作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「データベース」ノード、「Oracle」ノードの順に開きます。
2. 表を作成するモジュールを開きます。
3. 「表」を右クリックして、「表の作成」を選択します。

   「新規表ウィザード: ようこそ」ページに、表の作成手順が表示されます。
4. 「次へ」をクリックして、表に名前を付ける作業に進みます。
5. 表の名前を入力します。

   物理ネーミング・モードでは、1 ～ 28 の有効な文字数で構成される一意の名前を入力する必要があります。空白は使用できません。論理モードでは、4000 文字以内の名前を入力できます。空白も使用できます。名前はモジュール内で一意である必要があります。詳細は、2-25 ページの「ネーミング作業環境」を参照してください。

   表に名前を付けたら、ウィザードを引き続き使用して表のプロパティを定義するか、「終了」をクリックして表を作成します。後で、表エディタを使用して表のプロパティを設定または編集できます。
6. 表の説明を入力します。

   説明は 4000 文字以内で指定します。これはオプションです。
7. 「次へ」をクリックし、図 3-5 に示すように、列を定義する作業に進みます。

図 3-5「新規表ウィザード : 列」ページ

8. 「追加」をクリックして列を追加します。

   列名を入力します。Warehouse Builder では、ユーザーが列に入力した順序で、列の位置が生成されます。列順の変更は、3-21 ページの「表内の列の順序変更」を参照してください。

9. 「データ型」ドロップダウン・リストから列のデータ型を選択します。データ型の詳細は、3-18 ページの「列の追加」を参照してください。Warehouse Builder がサポートしている Oracle Database データ型は次のとおりです。

   CHAR

   DATE

   FLOAT

   NUMBER

   VARCHAR

   VARCHAR2

10. 該当する精度、長さまたはスケールを入力します。

    Warehouse Builder では、選択したデータ型に関連する値を入力できます。たとえば、CHAR データ型の場合は長さを指定する必要があります。NUMBER データ型の場合は精度とスケールを指定する必要があります。

11. NULL または NOT NULL を指定します。デフォルトでは、表内のすべての列で NULL が認められています。NOT NULL 制約を指定すると、表内の列に NULL 値を含めることができなくなります。

12. 「説明」フィールドに列の説明を入力します（オプション）。

13. 各列に対して、手順 8 ～ 13 を繰り返します。

14. 「次へ」をクリックし、制約を定義する作業に進みます。

15. 「新規表ウィザード : 制約」ページでは、必要に応じて制約を定義できます。制約の追加の詳細は、3-21 ページの「制約の編集」を参照してください。

16. 「次へ」をクリックします。

    Warehouse Builder では、新規表ウィザードに定義されている表のプロパティがすべて表示されます。名前、説明および様々な表のプロパティを確認します。

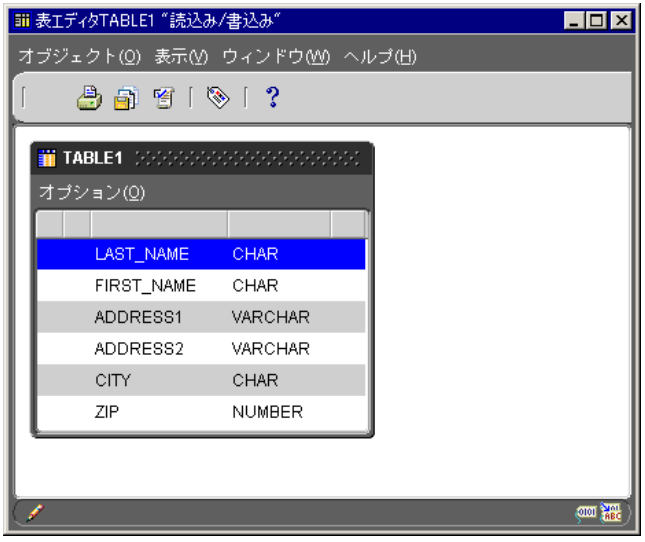
17. 「終了」をクリックします。

    表の定義が作成され、リポジトリに格納されます。また、ナビゲーション・ツリーに新しい表の名前が挿入されます。

表エディタの使用方法

Warehouse Builder で表を作成した後、表エディタを使用すると、表の構造、列および関連する表が表示されるようになります。表エディタを起動するには、表の名前を右クリックし、「エディタ」を選択します。図 3-6 に、表エディタを示します。

図 3-6 表エディタ



プロパティの編集、表の図の印刷、表定義の検証、表定義の同期化、および選択した表のレポートを実行するために Warehouse Builder Browser の呼出しを行うには、表エディタのメニューまたはナビゲーション・バーを使用するか、右クリックで表示されるポップアップ・オプションを使用します。

関連する表の表示

表エディタを起動すると、選択した表の図と、その表に定義されている列が表示されます。この表内の列は、外部キーの関係を通じて別の表内の列を参照する場合があります。参照されている表や関連する表を表示するには、「表示」→「関連オブジェクトの表示」を選択します。表エディタには、関連する表と、関係を示すリンク線が表示されます。表示された表の配置を変更するには、表を選択し、キャンバス上で移動します。

表定義の検証

DDL スクリプトを生成して表を作成する前に、表エディタで表定義を検証できます。「オブジェクト」→「検証」を選択します。「検証結果」ダイアログが表示され、定義が有効かどうかが示されます。定義が有効な場合は、スクリプトを生成して、そのオブジェクトをターゲット・データベースに作成できます。定義の検証とコードの生成の詳細は、第 12 章「オブジェクトの検証」を参照してください。

レポートの表示

表のメタデータ・レポートを表示するには、「表示」→「レポート」を選択します。表の系統レポートおよび影響分析レポートを表示して、表内のデータの系統や、表に対して行われた変更の影響を示すこともできます。これらのレポートのいずれかを表示するには、「表示」を選択してから、「系統」または「影響分析」を選択します。Warehouse Builder Browser のインストールと構成の詳細は、第 17 章「メタデータの参照およびレポート」を参照してください。

表定義の編集

表エディタからは、「表プロパティ」ウィンドウにアクセスし、表の名前、説明、列、制約および属性セットを編集できます。

表のプロパティを編集するには、表エディタのメニューから「オブジェクト」→「プロパティ」を選択するか、ナビゲーション・ツリーで表の名前を右クリックし、「プロパティ」を選択します。「表プロパティ」ウィンドウには、「名前」、「列」、「制約」および「属性セット」という 4 つのタブが表示されます。

これらのタブをクリックして次のタスクを実行します。

- 表の名前の変更
- 列の追加と削除
- 列順の変更
- 制約の追加と削除
- 制約の編集
- 制約の削除
- 属性セットの追加

表の名前の変更

表の説明を編集せずに名前を変更するには、Warehouse Builder ナビゲーション・ツリーで表の名前を右クリックし、「改名」を選択します。

表の名前と説明を編集する手順は次のとおりです。

1. 表の名前を右クリックし、「プロパティ」を選択します。
2. 「表プロパティ」ウィンドウで「名前」タブを選択し、次のプロパティを編集します。

   名前 : 表の新しい名前を入力します。物理ネーミング・モードでは、1 ～ 30 の有効な文字数で構成される一意の名前を入力する必要があります。空白は使用できません。論理ネーミング・モードでは、200 文字以内の名前を入力できます。空白も使用できます。名前はモジュール内で常に一意である必要があります。

   説明 : このフィールドに表の説明を入力するか、フィールド内の説明を変更します。説明は 4000 文字以内で指定します。このフィールドはオプションです。

列の追加

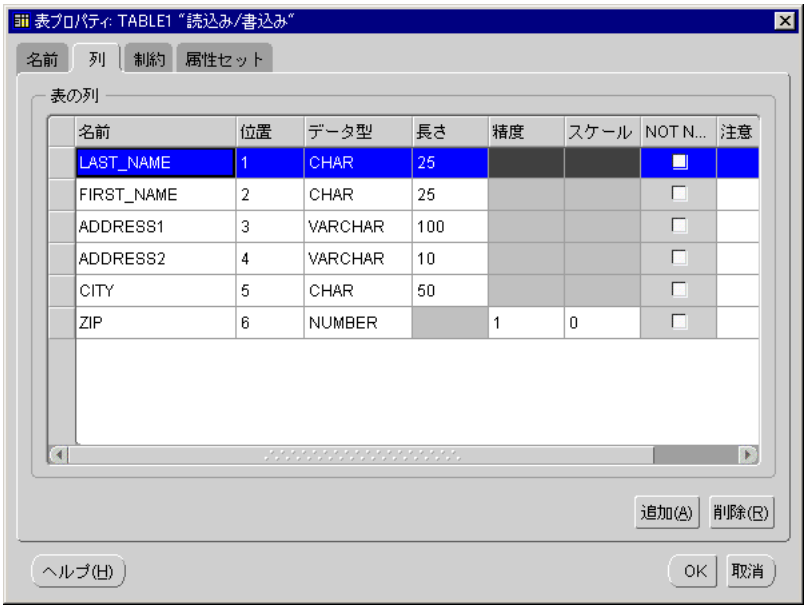
新しい列を追加する手順は次のとおりです。

1. ナビゲーション・ツリーでオブジェクトの名前を右クリックし、「プロパティ」を選択します。

   「表プロパティ」ウィンドウが表示されます。

2. 図 3-7 に示すように、「列」タブを選択します。

図 3-7「表プロパティ」の「列」ページ



3. 「追加」をクリックします。

   列フィールドに空白の行が表示されます。

4. 次の情報を入力し、新しい列を定義します。

   名前 : 列の名前を入力します。物理モードでは、1 ～ 30 の有効な文字数で構成される名前を入力する必要があります。空白は使用できません。論理モードでは、4000 文字以内の一意の名前を入力できます。列名は表内で一意である必要があります。空白も使用できます。

   位置 : デフォルトでは、列は作成順に並べられます。ただし、Warehouse Builder では列順を変更することが可能です。3-21 ページの「表内の列の順序変更」を参照してください。

   データ型 : ドロップダウン・リストから、列のデータ型を選択する必要があります。これは必須のフィールドです。Warehouse Builder では、次のデータ型を使用できます。

   長さ : 列の長さを定義します。長さは文字データ型に対してのみ定義されます。選択したデータ型によって、このフィールドは必須、編集不可、オプションのいずれかになります。

   精度 : 列に使用できる合計桁数を定義します。精度は数値データ型に対してのみ定義されます。選択したデータ型によって、このフィールドは必須、編集不可、オプションのいずれかになります。

スケール : 小数点以下の合計桁数を定義します。スケールは数値データ型に対してのみ定義されます。選択したデータ型によって、このフィールドは必須、編集不可、オプションのいずれかになります。

NOT NULL: デフォルトでは、表内のすべての列で NULL が認められています。NULL は、値がないことを意味します。このフィールドにチェックマークが付いている場合は、NULL 値または空の値を列に含めることができないことを示します。たとえば、NOT NULL 制約を定義して、従業員表の各行の last_name 列に値を入力するように要求できます。これはオプションのフィールドです。

注意 : 4000 文字以内で列の説明を入力します。これはオプションのフィールドです。

選択可能なデータ型

列の作成に使用できるデータ型は次のとおりです。

- CHAR: 固定長の文字データを格納する場合は、データ型に CHAR を選択します。長さは 4000 文字まで指定できます。内部のデータ表現は、データベースのキャラクタ・セットによって異なります。サイズはバイト数または文字数で指定できます。1 つの文字は、使用しているキャラクタ・セットのエンコードに応じて、1 以上のバイトを含みます。
- DATE: 固定長の日時を格納する場合は、データ型に DATE を選択します。この日時には、午前 0 時からの経過秒数が含まれています。日付は現在の月の最初の日、時刻は午前 0 時にデフォルト設定されます。日付関数 SYSDATE は、現在の日時を返します。
- FLOAT: データが単精度浮動小数点数の場合は、データ型に FLOAT を選択します。FLOAT の表現に互換性があり、長さが同じシステム間でのみ、FLOAT に正しい結果をロードできます。
- NUMBER: 固定小数点または浮動小数点の形式の実数を格納する場合は、データ型に NUMBER を選択します。このデータ型を使用する数値は、異なる Oracle プラットフォーム間で移植可能であることが保証され、精度は小数点以下最高 38 桁で表されます。NUMBER 列には、正数と負数のほかに、ゼロも格納できます。
- VARCHAR: VARCHAR フィールドは、長さの値を持つデータ型です。このデータ型は、バイナリ長のサブフィールドに続いて、指定された長さの文字列を含みます。長さはバイトで表現されますが、データファイルに文字の長さセマンティクスが使用されている場合には、文字数で表現されます。
- VARCHAR2: 可変長の文字データを格納する場合は、データ型に VARCHAR2 を選択します。内部のデータ表現は、データベースのキャラクタ・セットによって異なります。VARCHAR2 データ型は、最高 4000 文字までのサイズを指定する必須パラメータを使用します。

表内の列の順序変更

デフォルトでは、表内の列は作成順に表示されます。この順序は、その表を作成するために Warehouse Builder で生成される DDL スクリプトにも反映されます。このデフォルトの順序がアプリケーションのニーズに適していないか、問合せのパフォーマンスをさらに最適化したい場合は、列の順序を変更できます。

列の位置を変更する手順は次のとおりです。

1. ナビゲーション・ツリーで表の名前を右クリックし、「プロパティ」を選択します。

   「プロパティ」ウィンドウが表示されます。

2. 「列」タブを選択します。

   「列」ページに、そのオブジェクトに定義されている列がすべて表示されます。

3. 列名の左側にあるグレーの正方形を選択します。

   行全体が選択されます。カーソルが十字形に変わるまで待ちます。

4. 行を上下にドラッグして新しい位置にドロップします。

   これで列の位置が更新されました。

5. 「OK」をクリックします。

制約の編集

制約は、データベース内の情報と関連付けるビジネス・ルールを適用し、無効なデータが表に挿入されるのを防ぐために使用します。たとえば、従業員表の給与列に、給与 <10,000 という制約を定義した場合、この制約によって、この列の数値が 10,000 を超える行をこの表に入れない、というルールが適用されます。INSERT 文または UPDATE 文によって、この整合性制約への違反が発生する場合は、エラー・メッセージが表示されます。制約を使用すると、ロードのパフォーマンスが遅くなることに注意してください。

表に定義できる制約は次のとおりです。

- **一意キー（UK）:** 一意キーの制約では、1 つの列または列セット（キー）内のあらゆる値が一意であることが要求されます。表内の指定された列または列セットでは、2 つの行が重複する値を持つことはできません。一意キー列の値は NULL でもかまいません。
- **主キー（PK）:** キー（列または列セット）に定義されている値で、これにより、表内の各行を、キー（列または列セット）内の値で一意に識別できるように指定します。表内の指定された列または列セットでは、2 つの行が重複する値を持つことはできません。データベース内の各表には、主キー制約を 1 つしか指定できません。主キー列の値は NULL であってはいけません。
- **外部キー（FK）:** 1 つの表内のキー（列または列セット）に定義されているルールで、これにより、そのキーの値が参照表の主キーまたは一意キー（列または列セット）の値に一致することが保証されます。

- **チェック制約 :** 列（または列セット）にユーザーが定義するルールで、その列（または列セット）に格納されている値に基づいて、行の挿入や更新が制約されます。チェック制約の条件は、挿入または更新する行の値を使用して評価するブール式として指定します。たとえば、注文日 < 発送日という条件は、注文日列の値が発送日列の値よりも常に小さくなっているかどうかを確認します。この条件を満たしていない場合は、表のロード時にエラーが発生し、レコードは拒否されます。チェック制約の条件に、副問合せや順序、または SYSDATE 関数、UID 関数、USER 関数、USERENV SQL 関数を含めることはできません。チェック制約はデータの検証に役立ちますが、ロードのパフォーマンスが遅くなります。

Warehouse Builder での制約の定義

Warehouse Builder では、一意キー、主キー、外部キーおよびチェック制約を定義できます。次のルールに留意してください。

- 一意キー制約を作成した場合は、「表プロパティ」ウィンドウを使用して、制約タイプを後で主キーに変更できます。
- 主キー制約を作成した場合は、「表プロパティ」ウィンドウを使用して、制約タイプを後で一意キーに変更できます。
- 外部キー制約を作成した場合は、制約タイプは変更できません。制約をいったん削除し、「表プロパティ」ウィンドウを使用して、新しい制約を作成する必要があります。
- チェック制約を作成した場合は、制約タイプは変更できません。制約をいったん削除し、「表プロパティ」ウィンドウを使用して、新しい制約を作成する必要があります。
- チェック制約または主キー制約をそれぞれ外部キー制約またはチェック制約に変更するには、制約をいったん削除し、「表プロパティ」ウィンドウを使用して、新しい制約を作成する必要があります。

表に制約を追加する手順は次のとおりです。

1. オブジェクト名を右クリックし、「プロパティ」を選択して、オブジェクトの「プロパティ」ウィンドウを開きます。

   「プロパティ」ウィンドウが表示されます。

2. 図 3-8 に示すように、「制約」タブを選択します。

図 3-8「表プロパティ」の「制約」ページ

3. 「制約」フィールドの横にある「追加」ボタンをクリックします。

   「制約」フィールドに空白の行が表示されます。

4. 次の情報を指定して制約を定義します。

   名前 : 制約の名前を入力します。物理モードでは、1 ～ 30 の有効な文字数で構成される名前を入力する必要があります。空白は使用できません。論理モードでは、4000 文字以内の名前を入力できます。空白も使用できます。制約名はモジュール内で常に一意である必要があります。

   タイプ : ドロップダウン・リストから、チェック制約、外部キー、一意キーまたは主キーのいずれかの制約タイプを選択します。

5. 参照アプリケーションを指定します。

   外部キー制約を作成する場合は、参照先の表を含む参照先モジュールの名前をドロップダウン・リストから選択する必要があります。これは、現在のロケーションと異なるモジュールでもかまいません。このフィールドは外部キー制約には必須ですが、他の制約では入力できません。

6. 参照表を指定します。

   外部キー制約を作成する場合は、参照先のキーを含む参照先の表の名前をドロップダウン・リストから選択できます。このフィールドは外部キー制約には必須ですが、他の制約では入力できません。

7. 参照先のキーを指定します。

   外部キー制約を作成する場合は、参照先のキーの名前をドロップダウン・リストから選択します。このフィールドは外部キー制約には必須ですが、他の制約では入力できません。

8. チェック制約の条件を指定します。

   チェック制約を作成する場合は、このフィールドに条件またはルールを入力する必要があります。たとえば「Status = Active」と入力します。このフィールドを空白のままにすると、検証中にエラーが生成され、この制約に有効なコードを生成することはできません。

   チェック制約の条件で参照される列名は、その表のプロパティ・シートで定義されている物理名と完全に一致する必要があります。Warehouse Builder では、検証中に条件の構文をチェックすることはありません。このため、配布中にエラーが発生する場合があります。エラーが発生した場合は、Runtime Audit Browser で詳細を確認してください。

9. 「OK」をクリックして「プロパティ」ウィンドウを閉じます。

制約の削除

制約を削除する手順は次のとおりです。

1. 表の名前を右クリックして、「プロパティ」を選択します。

   「表プロパティ」ウィンドウが表示されます。

2. 「制約」タブを選択します。

   表に定義されている制約がすべて表示されます。

3. 削除する制約を選択します。

4. 「削除」をクリックします。

   「削除確認」ダイアログが表示されます。

5. 制約を削除するには「OK」をクリックします。

属性セットの追加

Warehouse Builder では、各表の属性セットまたは列のグループを定義できます。属性セットには、指定した順序で並べられた選択済の列が含まれます。属性セットは、マッピングの定義中や、データのインポートまたはエクスポート中に役立ちます。

Warehouse Builder では、表ごとに、その表内のすべての列を含む、事前定義済の属性セットが生成されます。さらに、定義された制約ごとに、事前定義済の属性セットも生成されます。事前定義済の属性セットは、変更も削除もできません。

次のタイプの属性セットを作成できます。

- ユーザー定義 :「表プロパティ」ウィンドウで作成、変更、削除できるオプションの属性セット。
- ブリッジ型 : Oracle Discoverer など、別のアプリケーションにエクスポートして表示できるオプションの属性セット。ブリッジ型属性セットは、「表プロパティ」ウィンドウで作成、変更、削除できます。1 つのオブジェクトで使用できるブリッジ型属性セットは 1 つのみです。

表に属性セットを追加する手順は次のとおりです。

1. ナビゲーション・ツリーで表の名前を右クリックし、「プロパティ」を選択します。

   「プロパティ」ウィンドウが表示されます。

2. 「属性セット」タブを選択します。

   その表で定義されている属性セットが表示されます。

3. 「追加」をクリックします。

   エンティティ・フィールドの属性セットに空白の行が表示されます。

4. 次の情報を指定して属性セットを定義します。

   名前 : 属性セットの名前を入力します。物理モードでは、1 ～ 30 の有効な文字数で構成される名前を入力する必要があります。空白は使用できません。論理モードでは、200 以内の有効な文字を入力できます。空白も使用できます。属性セット名はプロジェクト内で一意である必要があります。

   タイプ : ドロップダウン・リストから、「USER_DEFINED」または「BRIDGE_ TYPE」のどちらかの属性セット・タイプを選択します。

   説明 : 属性セットの説明を入力します。これはオプションのフィールドです。

5. 「選択された属性セットの属性」で、含める属性ごとに「挿入」をクリックします。列を選択した順序が、その属性セットの最初の順序になります。

   属性セット内の列をすべて含めるには「すべて選択」を、属性セットから列をすべて除外するには「すべて選択解除」をクリックします。属性セットから列を削除するには、チェック・ボックスを再びクリックしてチェックマークを外します。

ヒント： 属性セット内の属性の位置を変更するには、属性の左側にあるグレーの正方形をクリックして上下にドラッグし、新しい位置にドロップします。「表プロパティ」の「属性セット」タブを再び開くと、選択されている属性の順序に続いて、選択されていない属性の順序が表示されます。

6. ブリッジ型を選択した場合は、「拡張」をクリックします。

   「拡張された属性セット・プロパティ」ダイアログが表示されます。

7. ブリッジ型属性セットの属性ごとに、次のプロパティを指定します。これらのプロパティによって、Oracle Discoverer でオブジェクトを表示する方法が決定されます。

   Hidden: このチェック・ボックスを選択すると、別のアプリケーションで表を表示するときに、未使用の列または廃止された列が非表示になります。Discoverer Administrator の場合、非表示に設定した列はグレー表示されます。Discoverer Plus の場合、非表示に設定した列は表示されません。

   Aggregation: 数値属性の集計タイプを、「SUM」、「MIN」、「MAX」、「AVG」または「COUNT」の中から選択します。集計しない場合は「DETAIL」を選択します。デフォルトは「SUM」です。

   Position: デフォルトの属性位置を、「DATA POINT」、「PAGE」、「SIDE」、「TOP」または「TOP/SIDE」の中から選択します。デフォルトは「DATA POINT」です。

   Item Class: TRUE の場合はチェックマークを付け、FALSE の場合はチェックマークを外します。デフォルトは FALSE です。

   Heading: ヘッダーとして使用するテキストを入力します。

   Format: フォーマット・フィールドのテキストを入力します。

8. 「OK」をクリックして、「拡張された属性セット・プロパティ」ダイアログを閉じます。
9. 「OK」をクリックして「プロパティ」ウィンドウを閉じます。

外部表の使用方法

外部表は、この Oracle Database 内のデータベース・オブジェクトです。他のデータベース型やこの Oracle Database のリリースより前の Oracle データベースを外部表と併用することはできません。

外部表は、フラット・ファイルのデータをリレーショナル形式で表示した表です。これらは読取り専用の表で、Warehouse Builder では通常のソース表と同じような動作をします。外部表を作成および定義すると、外部表のメタデータが Warehouse Builder Design Repository に保存されます。これらの外部表の定義をマッピングで使用すると、フラット・ファイル・ソースからターゲットにデータを移動して変換する処理を設計できます。

次の項では、外部表について説明します。

- 外部表について
- 新しい外部表定義の作成
- 外部表エディタの使用方法
- 外部表の定義とファイル内のレコードとの調整
- 外部表定義の編集

関連情報は、次のトピックを参照してください。

- フラット・ファイルのソースとターゲットについて（4-28 ページ）
- 『Oracle Database ユーティリティ』マニュアル

外部表について

外部表は、読取り専用の表で、フラット・ファイルなど、外部データ内の 1 つのレコード・タイプに関連付けられています。外部表は、非リレーショナル・ソースのデータをリレーショナル表形式で表したものです。マッピングに外部表を使用するとき、列のプロパティは、フラット・ファイルのインポート時に定義した SQL プロパティに基づきます。フラット・ファイルの SQL プロパティの詳細は、4-47 ページの「SQL プロパティ」を参照してください。

マッピングのソースとして外部表を使用するときは、通常の表と同じように使用できます。Warehouse Builder では、外部表から行を選択するための SQL コードが生成されます。表を経由してファイルに並列アクセスすることもできます。

注意： 外部表はソース表のみです。マッピング・エディタでターゲットとして外部表演算子を接続すると、マッピングを実行したときに検証エラーが返されます。

外部表とフラット・ファイル演算子

フラット・ファイルのデータは、外部表またはフラット・ファイル演算子を経由して、マッピングに使用できます。2 つのオプションのどちらを使用するかは、データの変換方法を考慮して決定します。

外部表を使用すると、Warehouse Builder では SQL コードが生成されます。データを他の表と結合するか、複雑な変換が必要な場合は、外部表を使用します。

フラット・ファイル演算子を使用すると、Warehouse Builder では SQL*Loader コードが生成されます。大容量のデータを抽出する際に変換がほとんど不要な場合には、フラット・ファイル演算子を使用できます。フラット・ファイル演算子からは、必要に応じてステージング表へのデータのロード、索引の追加および変換の実行ができます。フラット・ファイル演算子を使用する場合、データに対して実行できる変換は、SQL*Loader 変換のみに制限されます。

Warehouse Builder では外部表を使用して、単一セットをベースとする SQL DML 文内で、ロードと変換を組み合せることができます。データは、ターゲット表に挿入する前に、一時的にステージングする必要はありません。

外部表と SQL*Loader（Warehouse Builder 内のフラット・ファイル演算子）の違いの詳細は、『Oracle Database ユーティリティ』マニュアルを参照してください。

新しい外部表定義の作成

フラット・ファイル・サンプル・ウィザードを使用してメタデータをインポートした後は、1 つのフラット・ファイル・レコード・タイプに基づいて外部表を作成できます。フラット・ファイル・データのインポートの詳細は、4-31 ページの「フラット・ファイル・サンプル・ウィザードの使用方法」を参照してください。

新しい外部表定義を作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「データベース」ノード、「Oracle」ノードの順に開きます。
2. 外部表を作成するターゲット・モジュールを開きます。
3. 「外部表」ノードを右クリックし、「外部表の作成」を選択します。

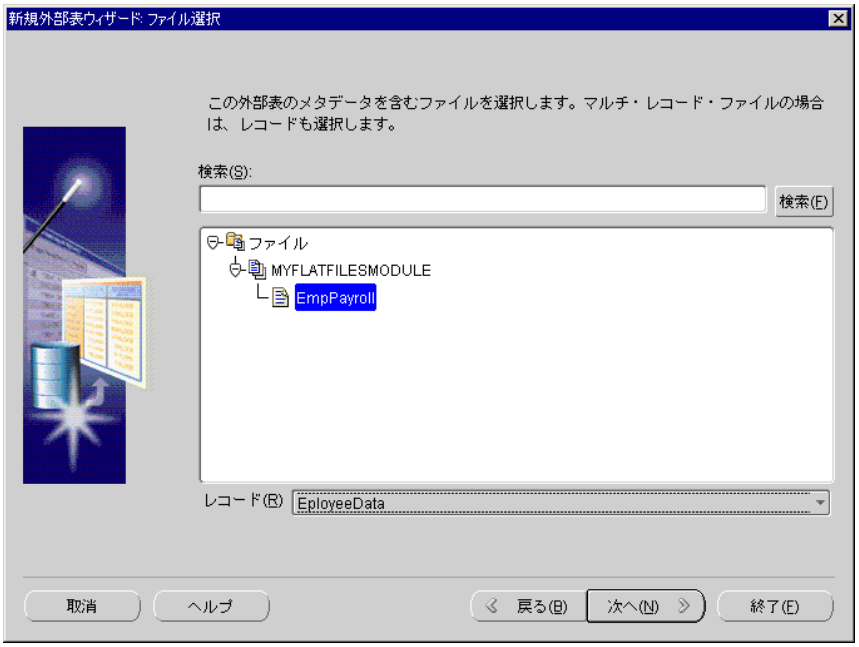
   「新規外部表ウィザード: ようこそ」ページが表示されます。
4. 「次へ」をクリックします。

   「新規外部表ウィザード: 名前」ページが表示されます。
5. 外部表の名前と説明（オプション）を入力します。

6. 「次へ」をクリックします。

   図 3-9 に示すように、「新規外部表ウィザード : ファイル選択」ページが表示されます。

図 3-9「新規外部表ウィザード : ファイル選択」ページ



7. ウィザードには、リポジトリ内の使用可能なフラット・ファイルが一覧表示されます。外部表の基になるファイルを選択します。長いファイル・リストから検索するには、検索するファイル名の最初の数文字を入力して「検索」をクリックします。

   ファイルが見つからない場合は、ファイルのメタデータがインポートされていることを確認してください。フラット・ファイルのインポートの詳細は、4-31 ページの「フラット・ファイル・サンプル・ウィザードの使用方法」を参照してください。

   レコード・タイプが複数あるファイルを選択した場合は、「新規外部表ウィザード : ファイル選択」ページの下部で、レコード・タイプ名も選択する必要があります。外部表には 1 つのレコード・タイプしか表示されません。

8. 「次へ」をクリックします。

   「新規外部表ウィザード: ロケーション」ページが表示されます。

9. ロケーションは、フラット・ファイルに関連付けられているロケーションを一覧表示したドロップダウン・ボックスから選択できます。ロケーションは指定しなくてもかまいません。ウィザードでロケーションを指定しない場合、後から外部表のプロパティ・シートでロケーションを指定できます。詳細は、3-3 ページの「ロケーションの定義」を参照してください。

   ヒント: 外部表を配布するには、フラット・ファイルのロケーションとOracle モジュールのロケーションの間のコネクタを配布する必要があります（このコネクタは、外部表作成時に自動的に作成されます）。

10. 「次へ」をクリックします。

    「新規外部表ウィザード: 終了」ページが表示されます。このページには、ウィザードの各ページで入力した情報の要約が表示されます。情報を確認します。

11. 「終了」をクリックします。

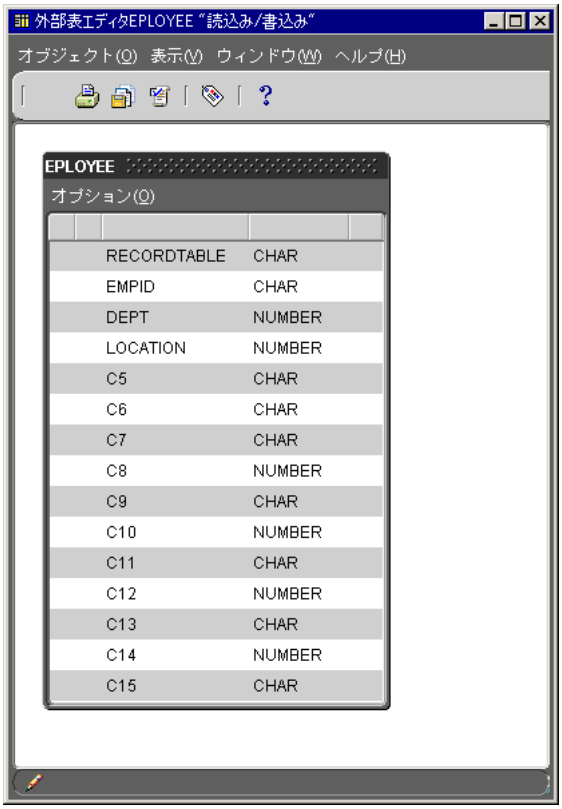
    このウィザードによって外部表が作成され、その名前がナビゲーション・ツリーに挿入されます。

外部表エディタの使用方法

Warehouse Builder には、エディタ・ダイアログと、通常の表に似た右クリックによるポップアップ・オプションが用意されています。

図 3-10 に、外部表エディタを示します。

図 3-10 外部表エディタ



外部表の図の印刷、外部表の定義の検証、外部表の定義の同期化、および選択した外部表のレポートを実行するための Warehouse Builder Browser の起動ができます。表エディタと外部表エディタの両方で使用できるオプションの詳細は、3-16 ページの「表エディタの使用方法」を参照してください。

外部表の定義とファイル内のレコードとの調整

表エディタでは使用できず、外部表エディタでのみ使用できるオプションに、調整オプションがあります。「調整」を使用すると、外部表に関連付けられているファイルに対するメタデータの変更を反映して、外部表の定義が更新されます。

外部表の定義とファイル内のレコードとを調整する手順は次のとおりです。

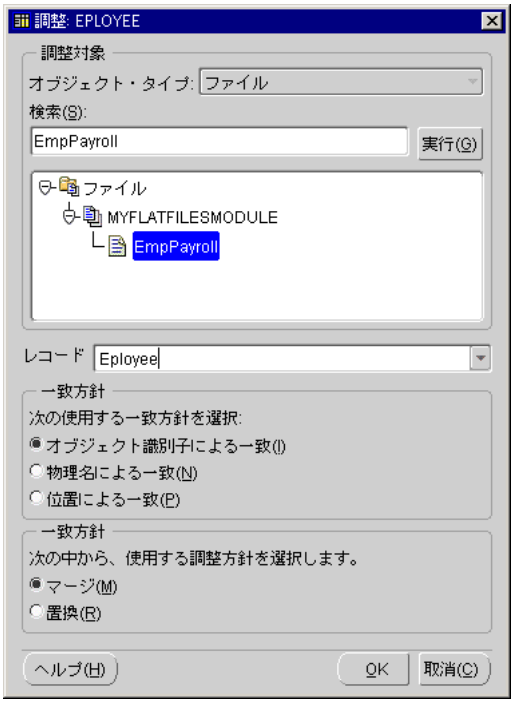
1. ナビゲーション・ツリーで外部表を右クリックして「エディタ」を選択し、外部表エディタを起動します。

   外部表エディタが表示されます。

2. メニューから「オブジェクト」→「調整」を選択します。

   図 3-11 に示すように、調整ダイアログが表示されます。調整ダイアログを使用して、フラット・ファイル内のレコードを指定します。

図 3-11 外部表の調整ダイアログ



3. 特定のフラット・ファイルを検索するには、「検索」ボックスにファイル名を入力して「実行」を選択します。または、リストをスクロールしてフラット・ファイルを選択します。

4. ファイルに複数のレコード・タイプがある場合は、「レコード」からレコードを選択します。

   レコードを選択する必要があります。調整対象のレコードを指定するまで、「一致方針」、「調整方針」および「OK」は使用できません。

5. 「一致方針」を設定します。選択内容に基づいて、一致する項目が検索され、フラット・ファイルの情報を反映して、外部表が更新されます。

   オブジェクト識別子による一致 : この方針では、外部表列参照のフィールド ID が、フラット・ファイル内のフィールド ID と比較されます。

   物理名による一致 : この方針では、外部表内の物理名が、フラット・ファイル内の物理名と比較されます。

   位置による一致 : この方針では、物理名や ID にかかわりなく、位置によって項目が照合されます。つまり、外部表の最初の属性はファイル内の最初のレコードによって調整され、2 番目の属性はファイル内の 2 番目のレコードによって調整されます。外部表と新しいレコードを調整するには、この方針を使用します。外部表の属性の方がフラット・ファイルの属性よりも多い場合、余分な属性は外部表から削除されます。

6. 「調整方針」を設定します。既存の外部表の定義と、指定したレコードの間のメタデータの差を、Warehouse Builder で処理する方法を決定するには、これらの設定を使用します。

   マージ : Warehouse Builder 既存の外部表の定義のメタデータと、指定したレコードのメタデータを組み合せます。

   置換 : 外部表の定義のメタデータは、指定したレコードのメタデータと一致しない場合に削除されます。調整後の外部表には、ファイル・レコードのメタデータに一致するメタデータが含まれます。

外部表定義の編集

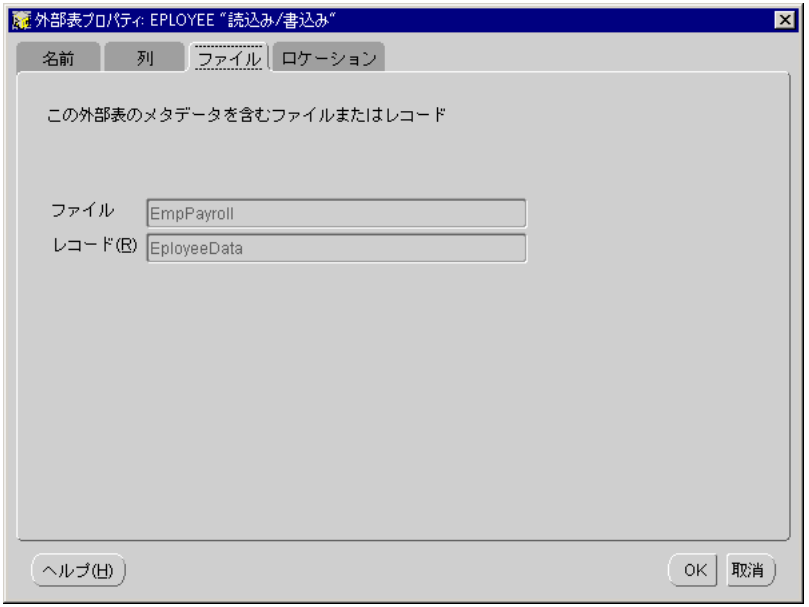
外部表エディタからは、「外部表プロパティ」ウィンドウにアクセスし、表の名前、説明、列、ファイルおよびロケーションを編集できます。

外部表のプロパティを編集するには、外部表エディタのメニューから「オブジェクト」→「プロパティ」を選択するか、ナビゲーション・ツリーで表の名前を右クリックし、「プロパティ」を選択します。「外部表プロパティ」ウィンドウが表示されます。編集可能なタブとプロパティは、リポジトリで外部表がどのように定義されているかによって異なります。

ほとんどの場合、表示される「外部表プロパティ」ウィンドウには、図 3-12 に示すように、次の 4 つのタブが含まれます。

- **名前 :** 外部表の名前を変更するには、「名前」タブを使用します。表の名前を変更するときと同じルールが外部表にも適用されます。詳細は、3-18 ページの「表の名前の変更」を参照してください。
- **列 :** 列を追加または編集するには、「列」タブを使用します。表に列を追加するときと同じルールが外部表にも適用されます。詳細は、3-17 ページの「表定義の編集」を参照してください。
- **ファイル :** 外部表のメタデータが用意されているフラット・ファイルの名前を表示するには、「ファイル」タブを使用します。ソース・フラット・ファイルに複数のレコード・タイプがある場合、「ファイル」タブには外部表に関連付けられているレコード名も表示されます。この関係を更新するか、別のファイルとレコードに変更するには、外部表を調整します。詳細は、3-32 ページの「外部表の定義とファイル内のレコードとの調整」を参照してください。
- **ロケーション :** フラット・ファイルのロケーションを表示または変更するには、「ロケーション」タブを使用します（図 3-12 を参照）。

図 3-12「外部表プロパティ」ウィンドウ



「ファイル」タブは次の条件で表示されます。

- 新規外部表ウィザードを使用して外部表を作成し、ファイル名を指定した場合。
- 新規外部表ウィザードではファイル名を指定していないが、外部表の定義をファイルおよびレコードと調整した場合。

アクセス・パラメータ

アクセス・パラメータでは、フラット・ファイルを読み取る方法を定義します。「外部表プロパティ」ウィンドウに、「ファイル」タブのかわりに「アクセス・パラメータ」タブが表示される場合もあります。

アクセス・パラメータのタブは次の条件で表示されます。

- 外部表を別のリポジトリからインポートした場合。この場合は、アクセス・パラメータの表示と編集ができます。
- 外部表を Oracle データベースに作成し、その定義を Warehouse Builder にインポートした場合。この場合は、アクセス・パラメータの表示と編集ができます。
- 新規外部表ウィザードを使用して外部表を作成し、参照ファイルを指定しなかった場合。アクセス・パラメータは空になります。外部表を生成する前に、外部表の定義をフラット・ファイル・レコードと調整するか、プロパティ・シートにアクセス・パラメータを手動で入力する必要があります。

アクセス・パラメータには、ソース・データファイル内のフィールドを外部表の列としてどのように表現するかを記述します。たとえば、データ型 INTEGER(2) のフィールド `emp_id` がデータファイルに含まれている場合、アクセス・パラメータで、そのフィールドを外部表内の文字列の列に変換するように指定できます。

Warehouse Builder が外部表を生成し、配布する方法に影響を及ぼすアクセス・パラメータを変更することはできますが、Warehouse Builder ではこのような変更は検証されないため、変更しないことをお薦めします。アクセス・パラメータ句の詳細は、『Oracle Database ユーティリティ』マニュアルを参照してください。

ビューの使用方法

Warehouse Builder では、ビューとマテリアライズド・ビューを定義できます。この項ではビューについて説明します。マテリアライズド・ビューの詳細は、3-40 ページの「マテリアライズド・ビューの使用方法」を参照してください。

次の項では、ビューの使用方法について説明します。

ビューについて

データの表現を簡潔化したり、データへのアクセスを制限するには、ビューを使用します。ユーザーが必要とするデータは、多くの列を持つ複数の表にまたがって保存されていることがよくあります。ビューを作成するときは、問合せの作成と格納を行って、関連のあるデータのみ、またはユーザーにアクセス権のあるデータのみを取得するようにします。

Warehouse Builder では、ビューを定義すると、ターゲット・データの問合せをモデル化できます。この問合せ情報は、Warehouse Builder Design Repository に格納されます。後でこれらの定義を使用すると、Warehouse Builder で `.ddl` スクリプトを生成し、ターゲット・システムにビューを作成する目的で配布できるようになります。

ビュー定義の作成

新規ビュー・ウィザードでビューを作成するときには、次の情報の入力が求められます。

- ビューの名前と説明
- 列の別名
- ビューを定義する問合せ（オプション）
- 論理制約（オプション）

ビュー定義を作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「データベース」ノード、「Oracle」ノードの順に開きます。
2. ビューを作成するターゲット・モジュールを開きます。
3. 「ビュー」を右クリックして、「ビューの作成」を選択します。

   「新規ビュー・ウィザード: ようこそ」ページが表示されます。

   注意： マッピング・エディタでビューを定義し、独自の問合せをモデル化することもできます。

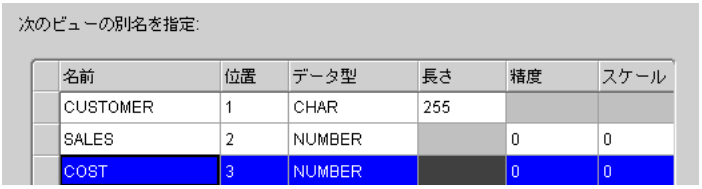
4. 「次へ」をクリックします。

   「新規外部表ウィザード: 名前」ページが表示されます。ビューの名前と説明を入力します。この説明はオプションです。

5. 「次へ」をクリックします。

   図 3-13 に示すように、「新規ビュー・ウィザード: 列」ページが表示されます。

図 3-13「新規ビュー・ウィザード : 列」ページ



6. 列を定義するには、「追加」をクリックします。

   列名を入力し、データ型を選択します。

   各列に対して、この手順を繰り返します。

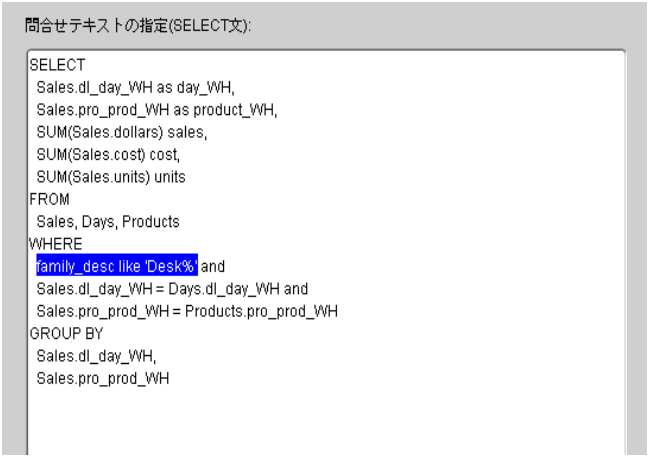
7. 「次へ」をクリックします。

   「新規ビュー・ウィザード : 問合せテキスト」ページが表示されます。問合せの定義を入力するか、「次へ」をクリックして続行します。

   「新規ビュー・ウィザード : 問合せテキスト」ページに問合せの定義を入力する場合は、有効な文を入力するようにしてください。Warehouse Builder では「新規ビュー・ウィザード : 問合せテキスト」ページのテキストは検証せず、構文が無効であってもビューを配布しようとします。

   図 3-14 は、問合せテキストの例を示しています。

図 3-14 問合せテキスト



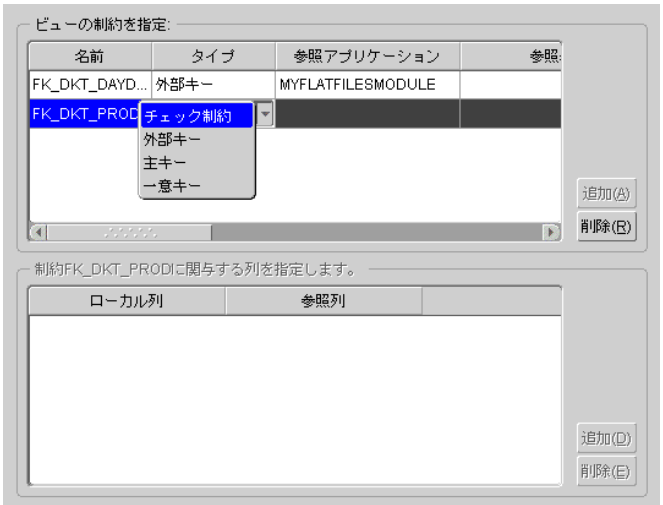
8. 「次へ」をクリックします。

   図 3-15 に示すように、「新規ビュー・ウィザード : 制約」ページが表示されます。

   オプションとしてこのページを使用して、ビューの論理制約を定義します。Warehouse Builder ではビューの DDL を列挙するときにこれらの制約を使用しませんが、これらの制約はビューがマッピングのデータ・ソースとして機能するときに役立ちます。マッピング・エディタは、論理的な外部キー制約を使用して、参照されているディメンションをマッピングの 2 次ソースとして取り込みます。

   制約の一般情報は、3-21 ページの「制約の編集」を参照してください。

図 3-15「新規ビュー・ウィザード : 制約」ページ



9. 「次へ」をクリックします。

   「新規ビュー・ウィザード : 終了」ページが表示されます。定義内容を確認します。定義を変更する場合は「戻る」をクリックします。

10. 「終了」をクリックします。

    ビューの定義が作成され、リポジトリに格納されます。さらに、ナビゲーション・ツリーに、作成した定義の名前が挿入されます。

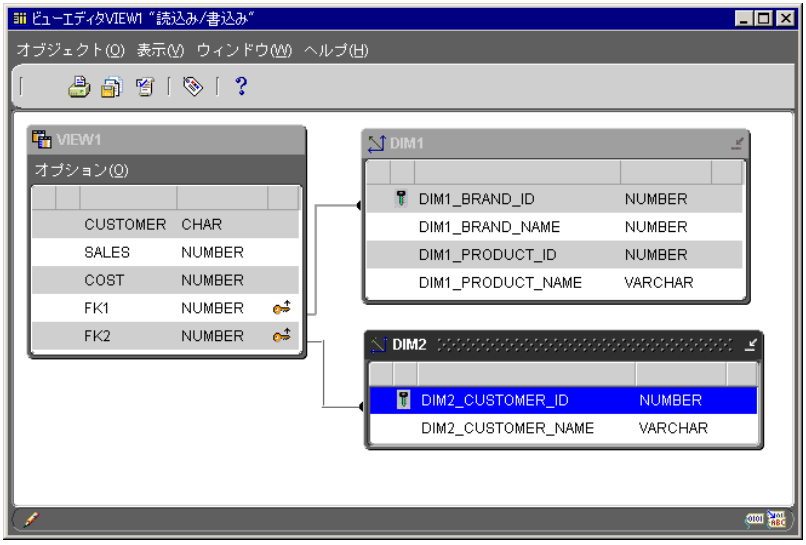
ビュー定義の編集

ビューの名前を変更するには、ビュー名を右クリックして、「改名」を選択します。ハイライト表示されたオブジェクト名の上に新しい名前を入力します。

ビューはビュー・エディタに表示できます。ビューを編集するにはプロパティ・シートを使用します。

ビュー・エディタを開くには、ビューを右クリックして、「編集」を選択します。図 3-16 に示すように、ビューとその参照が表示されます。

図 3-16 ビュー・エディタ



ビューのプロパティ・シートを開くには、目的のビューを右クリックして、「プロパティ」を選択します。ビュー定義を変更するには、プロパティ・シートを編集します。プロパティ・シートの編集例は、3-17 ページの「表定義の編集」を参照してください。

マテリアライズド・ビューの使用方法

Warehouse Builder では、ビューとマテリアライズド・ビューを定義できます。この項ではマテリアライズド・ビューについて説明します。従来型ビューの詳細は、3-55 ページの「ディメンションの使用方法」を参照してください。

次の項では、マテリアライズド・ビューの使用方法について説明します。

マテリアライズド・ビューについて

Warehouse Builder では、マテリアライズド・ビューを作成して問合せのパフォーマンスを改善できます。マテリアライズド・ビューを作成するときは、複数の表からデータを集計または結合する問合せコマンドのセットを作成します。マテリアライズド・ビューには事前に計算されたデータが用意されており、再利用や、リモートのデータ・マートへのレプリケーションが可能です。たとえば、会社の売上に関するデータは、組織全体で必要とされます。

Warehouse Builder でマテリアライズド・ビューを作成するときは、ビューを構成すると、Oracle Database のクエリー・リライト機能や高速リフレッシュ機能を利用できます。クエリー・リライトおよび高速リフレッシュの詳細は、5-25 ページの「マテリアライズド・ビューの高速リフレッシュ」を参照してください。

マテリアライズド・ビュー定義の作成

新規マテリアライズド・ビュー・ウィザードでマテリアライズド・ビューを作成するときには、次の情報の入力が求められます。

- マテリアライズド・ビューの名前
- マテリアライズド・ビューの説明
- 列名
- マテリアライズド・ビューを定義する問合せ
- 論理制約（オプション）

マテリアライズド・ビュー定義を作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「データベース」ノード、「Oracle」ノードの順に開きます。
2. マテリアライズド・ビューを作成するターゲット・モジュールを開きます。
3. 「マテリアライズド・ビュー」を右クリックし、「マテリアライズド・ビューの作成」を選択します。

   「新規マテリアライズド・ビュー・ウィザード: ようこそ」ページが表示されます。

   注意： マッピング・エディタでマテリアライズド・ビューを定義することもできます。

4. 「次へ」をクリックします。

   「新規マテリアライズド・ビュー・ウィザード: 名前」ページが表示されます。
5. ビューの名前と説明を入力します。この説明はオプションです。
6. 「次へ」をクリックします。

   図 3-17 に示すように、「新規マテリアライズド・ビュー・ウィザード: 列」ページが表示されます。

図 3-17「新規マテリアライズド・ビュー・ウィザード: 列」ページ

7. 列を定義するには、「追加」をクリックします。

   列名を入力し、データ型を選択します。各列に対して、この手順を繰り返します。

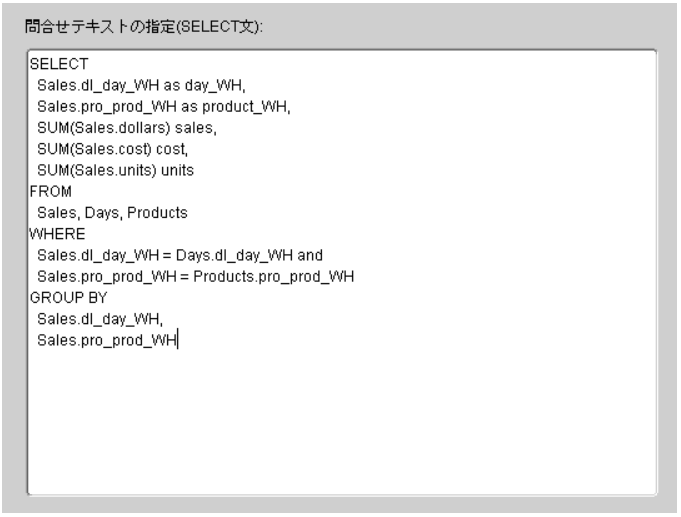
8. 「次へ」をクリックします。

   「新規マテリアライズド・ビュー・ウィザード: 問合せテキスト」ページが表示されます。問合せの定義を入力するか、「次へ」をクリックして「新規マテリアライズド・ビュー・ウィザード: 制約」ページに進みます。

   「新規マテリアライズド・ビュー・ウィザード: 問合せテキスト」ページに問合せの定義を入力する場合は、有効な文を入力するようにしてください。列名には、前の手順で「新規マテリアライズド・ビュー・ウィザード: 列」ページに指定したものと同じ名前を使用してください。「新規マテリアライズド・ビュー・ウィザード: 列」ページで列名を変更する場合は、「新規マテリアライズド・ビュー・ウィザード: 問合せテキスト」ページで、手動で名前を変更する必要があります。Warehouse Builder では、「新規マテリアライズド・ビュー・ウィザード: 問合せテキスト」ページのテキストは検証せず、構文が無効であってもビューを配布しようとします。

   図 3-18 は、問合せテキストの例を示しています。

図 3-18 問合せテキストの例

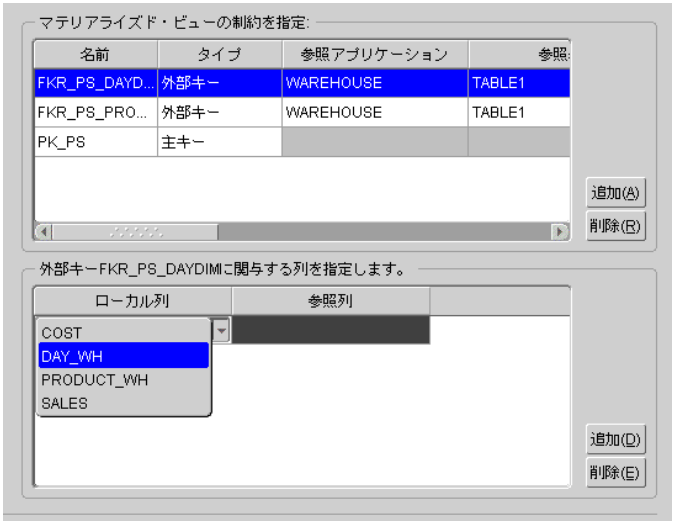


9. 「次へ」をクリックします。

   「新規マテリアライズド・ビュー・ウィザード: 制約」ページが表示されます。オプションとして、図 3-19 に示すように、マテリアライズド・ビューの制約を定義できます。

   これらの制約は、論理設計の目的のみに使用されます。Warehouse Builder では、ビューの DDL を列挙するときにこれらの制約が使用されることはありません。制約の一般情報は、3-21 ページの「制約の編集」を参照してください。

図 3-19「新規マテリアライズド・ビュー・ウィザード: 制約」ページ



10. 「次へ」をクリックします。

    「新規マテリアライズド・ビュー・ウィザード: 終了」ページが表示されます。表示内容を確認し、定義を変更する場合は「戻る」をクリックします。

11. 「終了」をクリックします。

    マテリアライズド・ビューの定義が作成され、データベース・モジュールに保存されます。さらに、ウェアハウス・モジュールのナビゲーション・ツリーに、作成したマテリアライズド・ビューの名前が挿入されます。

マテリアライズド・ビュー定義の編集

マテリアライズド・ビューの名前を変更するには、ビュー名を右クリックして、「改名」を選択します。ハイライト表示されたオブジェクト名の上に新しい名前を入力します。

マテリアライズド・ビューはマテリアライズド・ビュー・エディタに表示できます。マテリアライズド・ビューを編集するにはプロパティ・シートを使用します。

マテリアライズド・ビュー・エディタを起動するには、目的のマテリアライズド・ビューを右クリックして、「エディタ」を選択します。マテリアライズド・ビューとその参照が表示されます。

マテリアライズド・ビューのプロパティ・シートを開くには、目的のマテリアライズド・ビューを右クリックして、「プロパティ」を選択します。ビュー定義を変更するには、プロパティ・シートを編集します。

プロパティ・シートの編集例は、3-17 ページの「表定義の編集」を参照してください。

順序の使用方法

順序は、一意の数値のシリアル・リストを生成するデータベース・オブジェクトです。順序を使用すると、一意の主キーの値が生成され、複数の行や表にわたってキーが調整されます。順序の値は一意であることが保証されます。Warehouse Builder で順序を作成すると、順序の定義が作成されて、リポジトリに保存されます。順序の定義をマッピングに使用すると、データを変換してターゲット・システムに移動する間に一意の数値が生成されます。

次の項では、順序の使用方法について説明します。

順序について

順序は、NEXTVAL 疑似列および CURRVAL 疑似列を使用した SQL 文で参照されます。新しい各順序の数値は、NEXTVAL 疑似列を参照することで増分されます。一方、現在の順序の数値は、CURRVAL 疑似列を使用して参照されます。順序を定義すると、これらの属性が作成されます。

Warehouse Builder では、オブジェクト・インポート・ウィザードを使用して、既存のソース・システムから順序の定義をインポートすることもできます。詳細は、3-67 ページの「ターゲット・モジュールへのメタデータのインポート」を参照してください。

順序定義の作成

新しい順序を作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、ウェアハウス・モジュールのノードを開きます。
2. 「順序」を右クリックし、ポップアップ・メニューから「順序の作成」を選択します。

   新規順序ウィザードが表示されます。
3. 次の情報を入力します。

   名前 : 列の名前を入力します。物理モードでは、1 ～ 30 の有効な文字数で構成される名前を入力する必要があります。空白は使用できません。論理モードでは、4000 文字以内の一意の名前を入力できます。列名は表内で一意である必要があります。空白も使用できます。

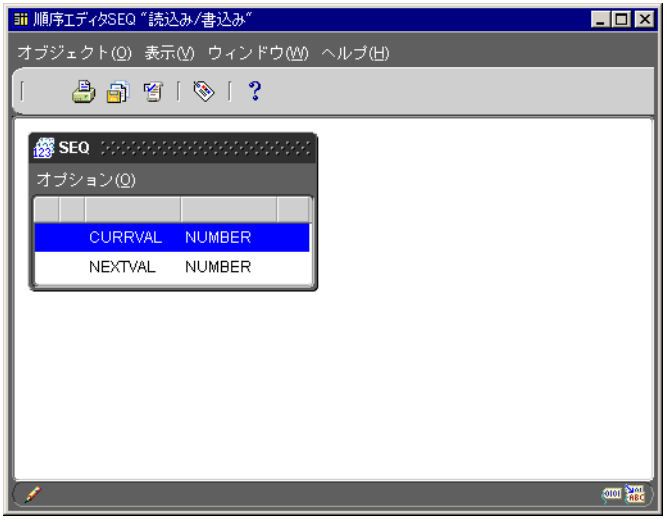
   説明 : 順序の説明を入力します（オプション）。説明は 4000 文字以内にします。
4. 「OK」をクリックします。

   順序の定義が保存され、その名前がナビゲーション・ツリー内に挿入されます。

順序エディタの使用方法

Warehouse Builder で順序を定義した後、順序エディタを使用すると、列が表示されるようになります。順序エディタを起動するには、順序の名前を右クリックし、「エディタ」を選択します。図 3-20 に、順序エディタを示します。

図 3-20 順序エディタ



順序の図の印刷、順序定義の検証、順序定義の同期化、および選択した順序のレポートを実行するために Warehouse Builder Browser の呼出しを行うには、順序エディタのメニューまたはナビゲーション・バーを使用するか、右クリックで表示されるポップアップ・オプションを使用します。

順序のメタデータ・レポートを表示するには、「表示」→「レポート」を選択します。Warehouse Builder Browser のインストールと構成の詳細は、『Oracle Warehouse Builder 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。

順序定義の編集

順序エディタからは、「順序プロパティ」ウィンドウにアクセスし、順序の名前、説明、列および列の説明を編集できます。

順序プロパティを編集するには、ナビゲーション・ツリーで順序の名前を右クリックして「プロパティ」を選択するか、順序の名前をダブルクリックします。「順序プロパティ」ウィンドウには、「一般」と「列」という 2 つのタブが表示されます。

これらのタブをクリックして次のタスクを実行します。

- 順序名の変更
- 順序列の編集

順序名の変更

順序の名前と説明を編集する手順は次のとおりです。

1. 順序の名前を右クリックし、「プロパティ」を選択します。

   「順序プロパティ」ウィンドウが表示されます。

2. 「一般」タブを選択します。

   名前 : 順序の新しい名前を入力します。物理モードでは、1 ～ 30 の有効な文字数で構成される一意の名前を入力する必要があります。空白は使用できません。論理モードでは、200 文字以内の名前を入力できます。空白も使用できます。名前はモジュール内で一意である必要があります。

   説明 : このフィールドに順序の説明を入力するか、フィールド内の説明を変更します。説明は 4000 文字以内で指定します。このフィールドはオプションです。

ヒント : 順序の説明を編集せずに順序名を変更するには、ナビゲーション・ツリーで順序の名前を右クリックし、「改名」を選択します。

順序列の説明の編集

順序の列の説明を編集する手順は次のとおりです。

1. 順序の名前を右クリックし、「プロパティ」を選択します。

   「順序プロパティ」ウィンドウが表示されます。

2. 「列」タブを選択します。

3. 「注意」フィールドまでスクロールし、選択した列の説明を入力するか、既存の説明を変更します。

アドバンスト・キューの使用方法

Oracle Advanced Queuing（AQ）は、データベース統合メッセージ・キュー・システムで、エンタープライズ・データの統合において中心的な役割を果たします。Warehouse Builder では、アドバンスト・キューの定義をインポートし、AQ を使用してソース・システムからターゲット・システムにデータを移動できます。

Oracle Advanced Queuing の詳細は、『Oracle Streams アドバンスト・キューイング・ガイドおよびリファレンス』を参照してください。

Oracle Advanced Queuing（AQ）を Warehouse Builder に統合することにより、次の作業が可能になります。

- データ・ウェアハウス設計で、データ・ソースまたはターゲットとして AQ を使用する。
- MQ Series と TIBCO アプリケーションをデータ・ソースとして統合する。
- AQ を使用してソース・データの変更を伝播する。

この項では、次のトピックについて説明します。

- アドバンスト・キューについて（3-48 ページ）
- アドバンスト・キューの定義の作成（3-49 ページ）
- アドバンスト・キュー・プロパティの表示（3-52 ページ）
- SQL を使用したアドバンスト・キューの作成（3-53 ページ）

関連情報は、次を参照してください。

- アドバンスト・キューの構成（5-21 ページ）
- アドバンスト・キューのマッピング演算子（8-24 ページ）

アドバンスト・キューについて

Web 対応のビジネス・アプリケーション同士が、メッセージ・キューを経由して通信することがよくあります。アドバンスト・キューイングでは、Oracle データベースの機能を利用して、これらのメッセージを永続的に格納し、異なるマシンやデータベースのキューの間で伝播し、Oracle Net Services、HTTP（S）および SMTP を使用して転送します。AQ を使用すると、アプリケーションの統合に必要なメッセージ管理や通信が可能になります。
Warehouse Builder は、ウェアハウス設計のデータ・ソースとして AQ をサポートしています。

Warehouse Builder では、次のタイプの AQ がサポートされています。

- マルチ・コンシューマ・キュー。関連付けられているキュー表は、データベース内にマルチ・コンシューマ・キューとして作成する必要があります。
- 標準キュー（例外キューはサポートされていません）。
- 永続キュー（非永続キューはサポートされていません）。

この項では、AQ に関連する主な概念について説明します。

ペイロード

ペイロードは、キューに格納されているデータです。ペイロードには、データ型 RAW のように構造化されていないものと、構造化されているものがあります。ペイロードを構造化するには、Oracle オブジェクト・タイプ（別名 ADT）や、ユーザー定義タイプを使用します。埋込み属性、コレクション属性、XMLType 属性、SYS.AnyData 属性などの複合オブジェクト・タイプは、Warehouse Builder では現在サポートされていません。

AQ では、ペイロード・タイプが Oracle オブジェクト・タイプと RAW のどちらかである必要があります。Warehouse Builder で使用するには、Warehouse Builder がサポートしているスカラー・データ型の属性を含むオブジェクト・タイプとして、ペイロードを定義する必要があります。RAW データ型はサポートされていません。

メッセージ

メッセージは最小単位の情報で、キューに入力するかキューから取得します。メッセージには制御情報とペイロード・データが含まれています。制御情報には、AQ がメッセージの管理に使用するメッセージ・プロパティまたはメタデータが含まれています。ペイロード・データは、キューに格納される情報です。メッセージは 1 つのキューにしか常駐できません。

アドバンスト・キューの定義の作成

現時点では、AQ は Oracle ソース・スキーマから Warehouse Builder にしかインポートできません。ソース・スキーマに AQ を作成する方法の詳細は、3-53 ページの「SQL を使用したアドバンスト・キューの作成」を参照してください。

AQ 定義のインポート

インポートされた AQ メタデータには、AQ の名前、説明、ペイロードのオブジェクト・タイプ名、ペイロードの構造、オブジェクト・タイプが定義されているスキーマ、および関連付けられているキュー表の名前が含まれています。これらの定義はソース AQ ではなく、エージェント AQ からインポートされます。ペイロードのオブジェクト・タイプは、AQ と同じスキーマに常駐する必要があります。ペイロードのオブジェクト・タイプが AQ と同じスキーマに定義されていない場合、AQ はインポートされますが、ペイロード・タイプはインポートされません。この場合、AQ はどのペイロード・タイプとも関連付けられず、無効になります。この AQ を検証すると、エラーが表示されます。オブジェクト型を作成または削除するには、スクリプトおよび Public Java API を使用します。

Warehouse Builder に AQ をインポートする手順は次のとおりです。

1. ナビゲーション・ツリーでターゲット・モジュールを右クリックし、「インポート」を選択します。

   データベース・リンクを定義していない場合は、「データベース・リンク情報」ダイアログが表示されます。

2. AQ 定義のインポート元であるシステムへの新しいデータベース・リンクを作成するか、以前に作成したデータベース・リンクを選択します。

3. 「OK」をクリックします。

   「メタデータ・インポート・ウィザード: ようこそ」ページが表示されます。

4. 「次へ」をクリックします。

   「メタデータ・インポート・ウィザード: フィルタ情報」ページが表示されます。

5. インポートするオブジェクトのタイプとして「アドバンスト・キュー」を選択します。

6. データ・ディクショナリの検索を次のいずれかの方法に制限します。

   検索パターンを入力します。たとえば、プロジェクト名で始まるオブジェクトをインポートするには、目的のウェアハウス・プロジェクト名に続けてパーセント記号（%）を入力します。複数の文字に一致するワイルド・カードにはパーセント記号（%）を、1 文字に一致するワイルド・カードにはアンダースコア（_）を使用します。

   取得するオブジェクト数の上限を入力します。

7. 「次へ」をクリックします。

   Warehouse Builder によって、フィルタ条件を満たすオブジェクトの名前が表示され、「メタデータ・インポート・ウィザード: オブジェクトの選択」ページが表示されます。

8. 「使用可能なオブジェクト」リストから「アドバンスト・キュー」ノードを開き、インポートするキューを選択します。

9. 選択したキューを「選択したオブジェクト」リストに移動するには、「>」ボタンをクリックします。Warehouse Builder では、永続的なマルチ・コンシューマ・キューのインポートしかサポートされていません。

   定義を再インポートする場合、すでにインポートされているオブジェクトは太字で表示されます。

10. 「次へ」をクリックします。

    「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページが表示されます。このページには、選択した内容の要約がスプレッドシートで表示されます。名前、オブジェクトのタイプ、およびオブジェクトが調整されるか作成されるかどうかが表示されます。このページの内容を確認して、オプションで各オブジェクトの説明を追加します。

11. 「終了」をクリックします。

    Warehouse Builder により、選択されたキューの定義がインポートされ、「インポート結果」ダイアログに結果が表示されます。キューをインポートすると、キューのペイロードも同時にインポートされます。キューのペイロードの属性を表示するには、キュー・ノードを開きます。

    ペイロードのオブジェクト・タイプは、AQ と同じスキーマに常駐する必要があります。ペイロードのオブジェクト・タイプが AQ と同じスキーマに定義されていない場合、AQ はインポートされますが、ペイロード・タイプはインポートされません。この場合、AQ はどのペイロード・タイプとも関連付けられず、無効になります。この AQ を検証すると、エラーが表示されます。

12. 定義が作成されていることを確認し、「OK」をクリックしてインポートを完了します。

    定義をインポートしない場合は、「元に戻す」をクリックします。または、定義をローカル・ドライブに保存する場合は、「保存」をクリックします。インポートされた定義は、ウェアハウス・モジュールのナビゲーション・ツリーの「アドバンスト・キュー」ノードの下に保存されます。

注意: 実行時は、AQ をソースとして使用するために、一時表が使用可能になっている必要があります。AQ を配布すると、表の作成スクリプトが生成されます。

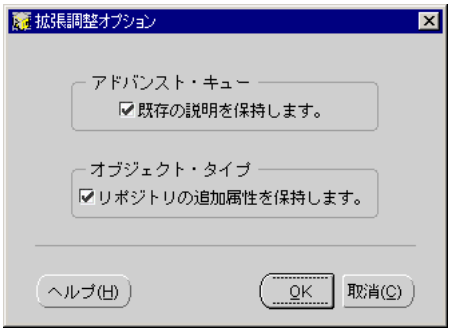
アドバンスト・キュー定義の再インポート

2 つのアドバンスト・キューは、同じペイロード・タイプを共有できます。ペイロードが別のキュー（Q1）とともにすでにインポートされているキュー（Q2）をインポートする場合、Q2 の属性は以前にインポートされた属性と調整されます。

アドバンスト・キューを再インポートする手順は次のとおりです。

1. メタデータ・インポート・ウィザードを使用して、上の手順に従います。「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページでは、「拡張調整オプション」ボタンが有効になっています。

図 3-21 AQ の「拡張調整オプション」



2. 「拡張調整オプション」をクリックして、調整オプションを選択します。

   図 3-21 に示すように、「拡張調整オプション」ダイアログが表示されます。このダイアログを使用して、再インポートされたメタデータを Warehouse Builder Design Repository 内の既存の定義と調整します。リポジトリ内のオブジェクト・タイプ属性と AQ の説明を保持することもできます。

   オブジェクト・タイプを調整する次のオプションを選択します。

   リポジトリの追加属性を保持します。: リポジトリに追加された属性のうち、インポートするオブジェクトに含まれていない属性を保持するには、このチェック・ボックスを選択します。

   AQ を調整する次のオプションを選択します。

   既存の説明を保持します。: リポジトリに以前にインポートした AQ 定義を保持するには、このチェック・ボックスを選択します。

3. オプションを選択したら、「OK」をクリックします。

> **注意：** デフォルトでは、すべてのオプションが選択されています。チェック・ボックスの選択を解除すると、リポジトリ・オブジェクトは保持されず、置き換えられます。

4. 変更を受け入れる場合は「OK」をクリックします。インポートを取り消す場合は「元に戻す」をクリックします。

   インポートされた定義が、ウェアハウス・モジュールのナビゲーション・ツリーの AQ ノードの下に保存されます。

関連情報は、次の項を参照してください。

- アドバンスト・キューのマッピング演算子（8-24 ページ）
- アドバンスト・キューの構成（5-21 ページ）

アドバンスト・キュー・プロパティの表示

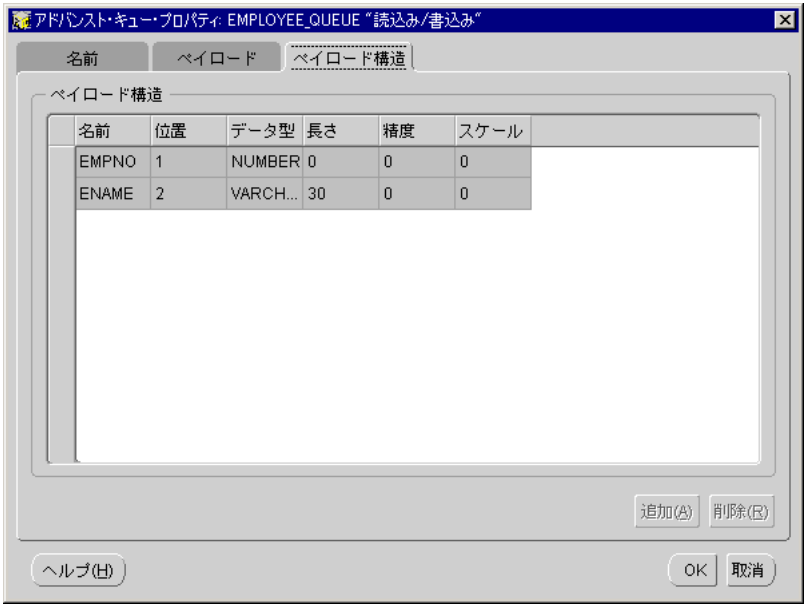
Warehouse Builder に AQ をインポートした後にプロパティを表示するには、AQ 名を右クリックし、ポップアップ・メニューから「プロパティ」を選択します。「アドバンスト・キュー・プロパティ」ウィンドウには、「名前」、「ペイロード」および「ペイロード構造」という 3 つのタブが表示されます。インポートした AQ プロパティは編集できません。

> **注意：** AQ とともにインポートされたオブジェクト・タイプは、クライアントの UI から削除することはできません。これらはスクリプトでしか削除できません。

それぞれのタブをクリックすると、次のプロパティを表示できます。

- **名前：** インポートされた AQ の名前と説明。
- **ペイロード：** AQ とともにインポートされたペイロードの名前と説明。
- **ペイロード構造：**「名前」、「位置」、「データ型」および「長さ」などの、AQ とともにインポートされたペイロードの属性（図 3-22 を参照）。

図 3-22 AQ プロパティの「ペイロード構造」タブ



SQL を使用したアドバンスト・キューの作成

次の手順は、SQL を使用して Oracle ソース・システムで AQ を作成する方法を示したものです。

```
SQL> grant aq_administrator_role, aq_user_role to scott;
```

(AQ の管理や使用を行うユーザーに、必要なロールを付与します。これらのロールは、システム管理者か、SYS または SYSTEM の SYSDBA 権限を持つユーザーによって付与される必要があります)

```
SQL> connect scott/tiger;
```

(権限の付与されているユーザーに、scott/tiger など、ユーザー名とパスワードを使用して接続します)

```
SQL> create type employee as object (empno number, ename
varchar2(30));
```

(従業員など、AQ の基になるオブジェクト構造を作成します)

```
SQL> execute dbms_aqadm.create_queue_table(queue_table => 'EMPLOYEE_
QUEUE_TBL', multiple_consumers=>true queue_payload_type =>
'EMPLOYEE');
```

(AQ 表を作成します。Warehouse Builder は現在、マルチ・コンシューマ・キューしかサポートしていないため、multiple_consumers => true となります)

```
SQL> execute dbms_aqadm.create_queue('EMPLOYEE_QUEUE', 'EMPLOYEE_
QUEUE_TBL');
```

(AQ を作成します)

```
SQL> execute dbms_aqadm.start_queue('EMPLOYEE_QUEUE');
```

(AQ を起動します)

Oracle Enterprise Manager を使用したアドバンスト・キューの作成

次の手順は、Oracle Enterprise Manager を使用して Oracle ソース・システムで AQ を作成する方法を示したものです。

1. AQ_ADMINISTRATOR_ROLE および AQ_USER_ROLE を、AQ を管理または使用するユーザーに付与します。ナビゲーション・ツリーで、「データベース」ノード、「データベース名」、「セキュリティ」、「ユーザー」、「ユーザー名」の順に開き、ロールを付与します (SYSDBA 権限が必要です)。
2. オブジェクト・タイプを作成します。「オブジェクト」メニューから、「作成」→「オブジェクト・タイプ」を選択します。属性を指定し、オブジェクト・タイプを作成します。
3. AQ 表を作成します。ナビゲーション・ツリーで、「データベース」ノード、「データベース名」、「分散」、「アドバンスト・キュー」の順に開き、「キュー表」をダブルクリックして AQ 表を作成します。
4. 「オブジェクト」メニューから、「作成」→「アドバンスト・キュー」を選択し、AQ を作成します。
5. 「データベース」ノード、「データベース名」、「分散」、「アドバンスト・キュー」、「キュー表」、「スキーマ名」、「キュー表名」の順に開き、AQ を起動します。キューを選択し、「オブジェクト」メニューで「キューの起動」を選択します。

ディメンションの使用方法

ディメンションは、スター・スキーマにおけるデータの主要な組織単位です。Warehouse Builder では、ディメンションを使用してキューブのデータを整理し、索引を付けます。一般的なディメンションには、「顧客」、「製品」、「時間」などがあります。

ディメンションを定義するときは、その階層、レベルおよびレベルの関係を定義する必要があります。レベルは集計レベルを表し、階層はレベルのセット間の親子関係を表します。ディメンションの階層は、データの編成方法として、レベルに順序を付けて使用する論理構造です。

ユーザーは既知の階層にドリルダウンしてデータを分析することがよくあるため、ディメンションを使用して問合せのパフォーマンスを向上させることができます。階層の例としては、年、四半期、月、日からなる時間階層などがあります。Oracle Database では、問合せをリライトし、詳細表ではなくサマリーからデータを取得することで、定義済の階層が利用されます。リライトされた問合せのパフォーマンスは向上しています。

典型的なディメンション表には、次のような特徴があります。

- 単一列の主キーに、ウェアハウス・キーと呼ばれる値が移入されます。
- ウェアハウス・キーはディメンションを管理し、ディメンションの履歴を保持する技法をサポートし、キューブのサイズを軽減します。
- ディメンション・オブジェクトとして明示的に定義された階層が 1 つ以上あります。Create Dimension 文で定義された階層は、Oracle Database サーバーによってクエリー・リライトされる回数を最大にします。

ディメンション・オブジェクトのルール

ディメンション定義には、ディメンション・オブジェクトの定義とディメンション表の定義が含まれます。この項では、ディメンション・オブジェクトについての情報を記載します。

表 3-2 は、ディメンションに対するルールをまとめたものです。

表 3-2 Warehouse Builder ディメンション・オブジェクトのルール

ルール	説明
非正規化	生成されるディメンション・オブジェクトは単一の表に定義されます。 Warehouse Builder では、一連の正規化された表に対するディメンション・オブジェクトの定義は現在サポートされていません。
機能的依存性	子の値から親の値が一意に決定される必要があります。 たとえば、市区町村が 1 つのみの県にロールアップされ、月から四半期が、製品からブランドが決定されます。これらの関係は、物理的なデータから取得される必要があります。そうでない場合、問合せは不適切な結果セットを戻すことがあります。

表 3-2 Warehouse Builder ディメンション・オブジェクトのルール（続き）

ルール	説明
一意キーの生成	Warehouse Builder では、階層の一番下のレベルにのみ一意キー制約が実装されます。
外部キー参照	表は階層の（一意キー制約が必ず定義されている）一番下のレベルのみを参照できます。

レベルと階層について

ディメンション・オブジェクトは、一連のレベルと、そのレベルに対して定義された一連の階層からなります。レベルは集計レベルを表します。階層は一連のレベル内での親子関係を表します。

たとえば、一般的なカレンダ・ディメンションは 5 つのレベルからなります。これらのレベルに対して、次のように 2 つの階層を定義できます。

H1: 年レベル > 四半期レベル > 月 レベル > 週 レベル

H2: 年レベル > 四半期レベル > 週 レベル > 日 レベル

階層は親から子へと表されるため、年は四半期の親、四半期は月の親、というようになります。

一意キー制約について

階層の定義を作成すると、Warehouse Builder では階層の各レベルに対して識別子が、最下位レベルに対して一意キー制約が作成されます。Warehouse Builder では、生成フェーズで識別子が使用され、ディメンション・オブジェクトを作成する DDL スクリプトが作成されます。

あるディメンションを指すキューブに外部キー参照を作成する際には、参照する列の候補として一意キー制約とその他の識別子キーが表示されます。キューブで参照できるのは、一意キー制約が設定されている階層の最下位レベルのみです。他のレベルを選択した場合、定義は無効となります。

図 3-23「外部キー」タブが表示されている「キューブ・プロパティ」ダイアログ

複合的な集計レベルについて

アプリケーションによっては、異なる集計レベルから始まる 2 つの階層が必要な場合があります。たとえば、次のような階層が考えられます。

H1: 年レベル > 四半期レベル > 月 レベル > 日 レベル

H2: 年レベル > 週 レベル > 日 レベル

H3: 年レベル > 四半期レベル >MonthLow レベル

この複合的なケースを Warehouse Builder でモデル化するには、次の作業が必要です。

- ディメンション表に、もう 1 つ MonthLow 列を追加する必要があります。
- MonthLow 列に一意の値を移入する必要があります。
- ディメンションに対して、MonthLow レベルを別に定義する必要があります。

この階層セットに対して、Warehouse Builder ではレベル識別子が 6 つと一意キー制約が 2 つ生成されます。1 つの一意制約は Days 列に、もう 1 つは MonthLow 列に定義されます。日 レベルと MonthLow レベルはそれぞれの階層の一番下に位置するため、この 2 つは外部キー参照のターゲットとして使用できます。

Warehouse Builder では、ディメンションは単一の正規化されていない表として生成され、一連のレベルと階層がその表に対して定義されます。各レベルには属性をいくつでも作成できます。

ディメンション定義の作成

ディメンションの定義を作成するには、新規ディメンション・ウィザードを使用します。ディメンションに名前を付け、主キー制約を定義します。各列を定義する際には、一貫性のない結果セットが戻されないようにするため、およびクエリー・リライトされる回数を最大にするために、制約を NOT NULL に設定することをお薦めします。

また、ディメンションの階層とその集計レベルも定義します。表 3-3 はディメンション表の例で、各集計レベル、各レベルの接頭辞、および各レベルに定義された属性を示しています。レベルは親から子の順番で記載されており、class は family の親、family は product の親です。

表 3-3 ディメンション・オブジェクトの例

レベル	接頭辞	属性	データ型	説明
class	cl	class_id	NUMBER	レベル識別子またはキー
		class_desc	VARCHAR (20)	製品クラスの説明
family	fa	family_id	NUMBER	レベル識別子またはキー
		family_desc	VARCHAR (20)	製品ファミリの説明
product	pd	prod_WH	NUMBER	ベース・レベルまたはウェアハウス・キー
		item_desc	VARCHAR (35)	製品の説明
		product_upc	VARCHAR (11)	汎用製品コード（天然キー）
		item_source	VARCHAR (30)	製品のサプライヤ
		packaging	VARCHAR (20)	製品のパッケージング

ディメンション定義を作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「データベース」ノード、「Oracle」ノードの順に開きます。
2. ディメンションを作成するターゲット・モジュールを開きます。
3. 「ディメンション」を右クリックして「ディメンションの作成」を選択します。

   「新規ディメンション・ウィザード: ようこそ」ページが表示されます。

4. 「次へ」をクリックします。

   図 3-24 に示すように、「新規ディメンション・ウィザード : 名前」ページが表示されます。

5. 次の内容を入力します。

   ディメンションの名前

   接頭辞

   接頭辞は、ベース・レベルのキー列に対する一意キー制約に一意の名前を生成するために使用されます。接頭辞に何も設定しない場合は、ディメンション名が使用されます。

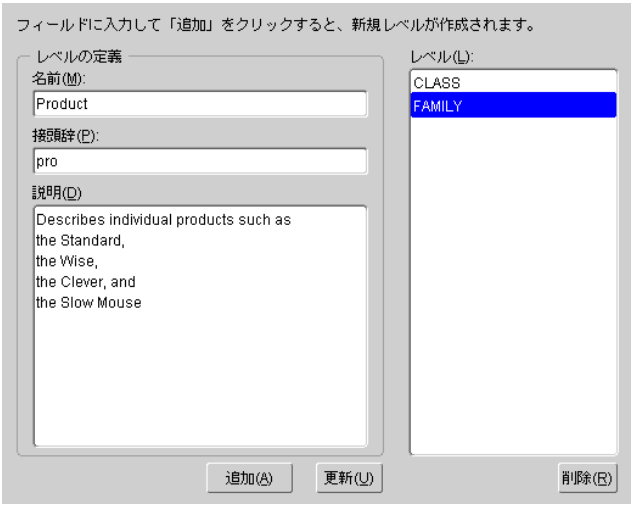
   ディメンションの説明（オプション）

図 3-24「新規ディメンション・ウィザード : 名前」ページ

6. 「次へ」をクリックします。

   「新規ディメンション・ウィザード : レベル」ページが表示されます。ディメンションには、レベルが少なくとも 1 つ必要です。この要件を満たすためにデフォルトのレベルを定義し、必要に応じてレベルを追加できます。図 3-25 に、「新規ディメンション・ウィザード : レベル」ページを示します。

図 3-25「新規ディメンション・ウィザード : レベル」ページ



7. ディメンションに集計レベルを定義します。次の情報を入力します。

   レベルの名前。

   レベルの接頭辞。デフォルトの接頭辞はレベルの名前です。

   レベルの説明。

8. 「追加」をクリックしてレベルを追加します。各集計レベルの定義を完了するまで、この作業を繰り返します。

   接頭辞が便利な理由として、次の 2 点があります。

   手動で入力する必要のある属性の数が削減されます。各レベルの ID 属性はウィザードによって生成され、levelprefix_ID という名前が割り当てられます。

   属性名を再利用できます。属性名の再利用は、より高い集計レベルに対してディメンションを作成する際に頻繁に発生します。

注意 : ディメンション接頭辞は、レベル一意キーの名前の構築に使用されます。一意キーの名前が生成された後でディメンション接頭辞を変更しても、キューブの外部キーが生成済のレベルを参照している可能性があるため、既存レベルの名前は変更されません。接頭辞を変更した後に生成される一意キーの名前には、新しい接頭辞が使用されます。

---

9. 「次へ」をクリックします。

   「新規ディメンション・ウィザード: レベル属性」ページが表示されます。レベルには1つ以上の属性を設定できます。ウィザードによって各レベルにID属性が生成されます。

   あるレベルのID属性は、そのレベルを識別します。ID属性はそのレベルのキー列です。この属性は、ディメンションの作成文でレベルの定義に使用され、定義されたレベルはDETERMINES句でそのレベル内の他の列（依存列）の指定に使用されます。詳細は、『Oracle Database SQL リファレンス』および『Oracle データ・ウェアハウス・ガイド』を参照してください。

10. ドロップダウン・リストから集計レベルを選択します。

11. 属性の名前を入力します。

12. 「データ型」ドロップダウン・リストから属性のデータ型を選択します。Warehouse Builderでは、Oracle Databaseの次のデータ型を使用できます。

    CHAR

    DATE

    FLOAT

    NUMBER

    VARCHAR

    VARCHAR2

13. データ型に応じて、精度、長さまたはスケールを入力します。

14. 属性の説明を入力します。

15. 「追加」をクリックします。

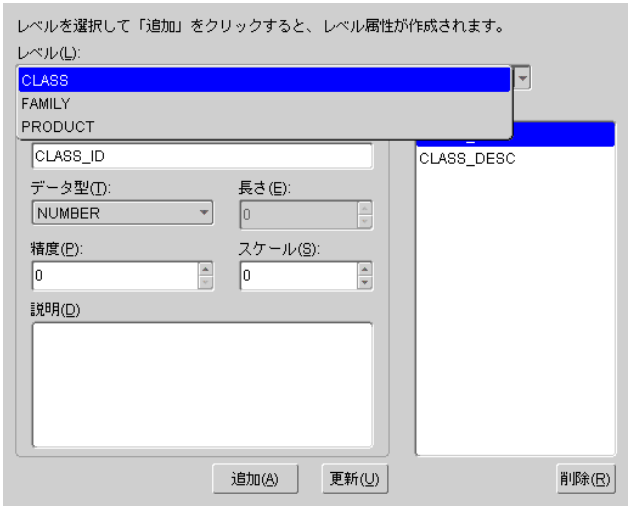
    選択中のレベルに別の属性を定義するか、または別のレベルを選択して、その属性を定義します。図 3-26 に示すように、各レベルの属性をすべて定義するまでこの作業を続けます。

    ID列の名前を変更するには、「レベル属性」テキスト・ボックスの「ID」を選択します。

    「名前」テキスト・ボックスに新しい名前を入力します。

    「更新」をクリックします。

図 3-26「新規ディメンション・ウィザード : レベル属性」ページ



16. 「次へ」をクリックします。

    「新規ディメンション・ウィザード : 階層」ページが表示されます。

17. **階層を定義します。**

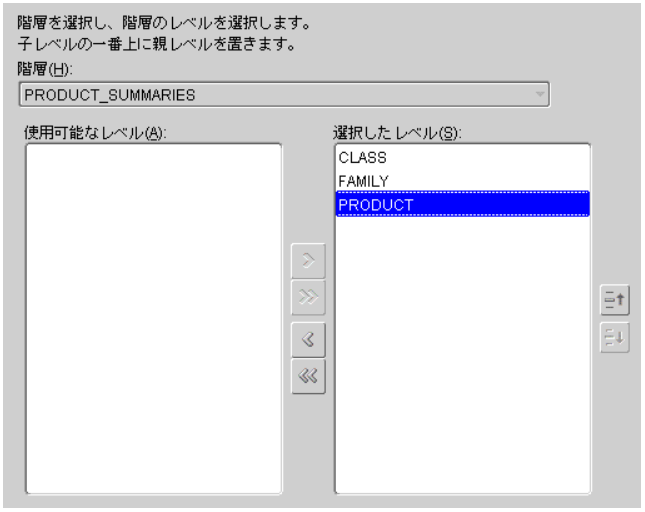
    各階層の名前と接頭辞を入力します。

    階層の説明を入力します。

18. 「次へ」をクリックします。

    図 3-27 に示すように、「新規ディメンション・ウィザード : レベルの関係」ページが表示されます。

図 3-27「新規ディメンション・ウィザード : レベルの関係」ページ



19. 階層内のレベルを定義します。

ドロップダウン・リストから階層を選択します。

選択した階層に対するレベルの名前を、「使用可能なレベル」から「選択したレベル」に移動します。

親から子の順番となるようレベルの位置を変更します。

20. 「次へ」をクリックします。

「新規ディメンション・ウィザード : 終了」ページが表示されます。定義内容を確認します。

21. 「終了」をクリックします。

ディメンションの定義が作成されます。

ウィザードでは、ディメンションのベースの集計レベルに該当する ID 列に対して、ディメンション表の一意キー（UK）制約が生成されます。ディメンションによっては、一意キー制約ではなく主キー（PK）制約が必要なことがよくあります。ディメンションの定義を完了した後で、一意キー制約を主キー制約に変更できます。

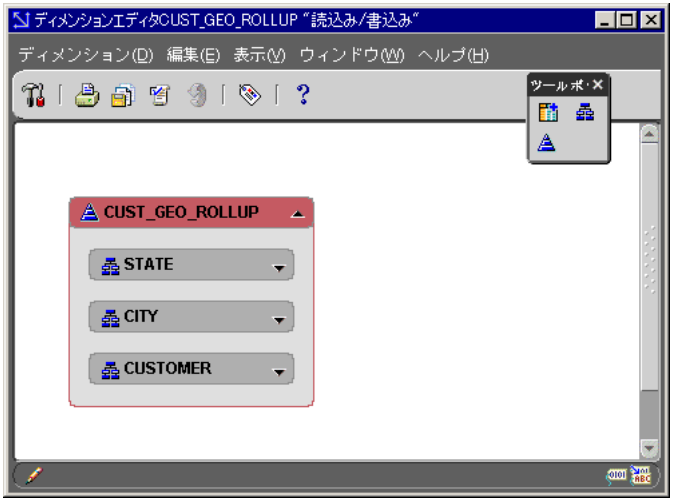
ディメンション定義の編集

ディメンション・オブジェクトの定義を編集するには、ディメンション・エディタを使用するか、ディメンション・プロパティ・シートのエントリを編集します。

ディメンション・エディタの使用

ディメンション・エディタを表示するには、ナビゲーション・ツリーでディメンションを右クリックし、「エディタ」を選択します。図 3-28 にディメンション・エディタを示します。

図 3-28 ディメンション・エディタ



ディメンション・エディタにツールボックスとディメンション・オブジェクトが表示されます。

ディメンション・オブジェクトに要素を追加するには、ツールボックスのアイコンをディメンション要素の上にドラッグ・アンド・ドロップします。

レベルに属性を追加する手順は次のとおりです。

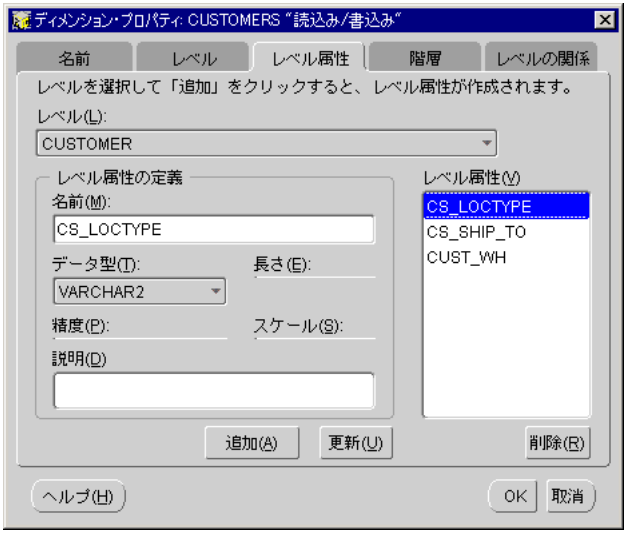
1. 新しい属性を追加するレベルを完全に開きます。
2. 「属性」アイコンをレベルの上にドラッグ・アンド・ドロップします。

   NUMBER 型の属性（属性 1）がレベルに追加されます。
3. 属性の名前を入力します。
4. データ型を変更するには、属性名をダブルクリックします。

   ディメンション・プロパティ・シートが表示されます。

5. 図 3-29 に示すように、「レベル属性」タブを選択します。
6. ドロップダウン・リストからデータ型を選択します。
7. 選択したデータ型に応じて、長さ、スケールまたは精度を変更します。
8. 「更新」をクリックします。

図 3-29 ディメンション・プロパティ・シートの「レベル属性」タブ



図を印刷するには、ディメンション・エディタのツールバーにある「印刷」アイコンをクリックします。

プロパティ・シートの使用方法

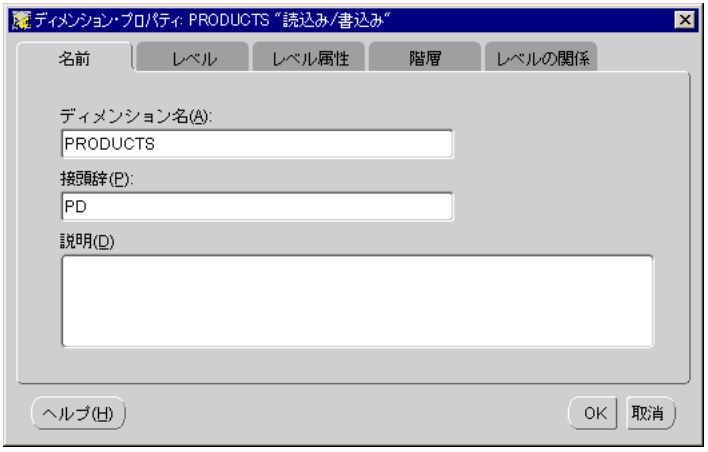
ディメンション・オブジェクトとディメンション表の両方にプロパティ・シートがあります。ディメンション・オブジェクトのプロパティ・シートでは、レベルと階層を編集します。ディメンション表のプロパティ・シートでは、列と制約を編集します。

ディメンション・オブジェクトのプロパティ・シートを表示する手順は次のとおりです。

- ディメンション・エディタで、「編集」メニューから「プロパティ」を選択するか、または「プロパティ」アイコンをクリックします。
- Warehouse Builder ナビゲーション・ツリーでディメンションを右クリックし、「プロパティ」を選択します。

図 3-30 に示すように、ディメンション・オブジェクトに対するプロパティ・シートが表示されます。

図 3-30 ディメンション・オブジェクトのプロパティ・シート



ディメンション・オブジェクトのプロパティ・シートには次のタブがあります。

- 名前
- レベル
- レベル属性
- 階層
- レベルの関係

ディメンション表のプロパティ・シートを表示する手順は次のとおりです。

1. ディメンション・エディタを起動します。
2. 「編集」メニューから「表プロパティ」を選択します。

   ディメンション表のプロパティ・シートが表示されます。表プロパティ・シートの詳細は、3-17 ページの「表定義の編集」を参照してください。

キューブの使用方法

キューブ（別名ファクト）には、メジャーおよび1つ以上のディメンションへのリンクが含まれます。キューブのメジャーのほとんどは、加法的なものです。代表的な加法メジャーには、売上、単位およびコストがあります。

キューブは、外部キー制約によってディメンション表にリンクされます。これらの制約は、データ整合性が最も重要とされるデータ・ウェアハウス環境において不可欠です。制約によって、データ・ウェアハウスを使用する日常業務において参照整合性が維持されます。

ディメンションにウェアハウス・キーが設定されている場合、ウェアハウス・キーは対応する本来のデータより短いため、通常キューブの行は短くなります。その結果、キューブ内で無駄になる記憶領域が少なくなります。

一般的なキューブには、次の要素が含まれます。

- 一連の外部キー参照列、またはデータ・リストの場合は人口キーまたは一連のウェアハウス・キー列に定義された主キー。キューブがデータ・リストの場合、外部キー参照列ではキューブの各行が一意に識別されません。
- 表を対応するディメンションにリンクする一連の外部キー参照列。

キューブの定義を作成する際には、メジャーおよび外部キー参照を定義する必要があります。外部キー参照を定義するには、参照するディメンションの名前とその主キー列を指定します。

キューブ定義の作成

この項では、キューブに対する定義の作成および更新方法を説明します。キューブの定義は新規キューブ・ウィザードを使用して作成し、定義を更新するにはプロパティ・シートを編集します。また、データベース・ソースやOracle Designerリポジトリから表の定義をインポートすることもできます。

キューブの定義を作成するには、新規キューブ・ウィザードを使用します。定義には、外部キー参照、メジャーおよび表のすべての列のデータ型に関する詳細が含まれます。

キューブ定義を作成する手順は次のとおりです。

1. Warehouse Builderナビゲーション・ツリーで、「データベース」ノード、「Oracle」ノードの順に開きます。
2. キューブを作成するターゲット・モジュールを開きます。
3. 「キューブ」を右クリックして「キューブの作成」を選択します。

   「新規キューブ・ウィザード: ようこそ」ページが表示されます。
4. 「次へ」をクリックします。

   「新規キューブ・ウィザード: 名前」ページが表示されます。

5. 次の情報を入力します。

   キューブの名前

   キューブの説明（オプション）

6. 「次へ」をクリックします。

   「新規キューブ・ウィザード: 外部キーの定義」ページが表示されます。

7. 「ディメンション」ドロップダウン・リストからディメンションの名前を選択します。

8. 「レベル」ドロップダウン・リストからベース・キー・レベルを選択します。

9. 「一意キー」ドロップダウン・リストから、ディメンションに定義されている主キー列制約を選択します。

10. 「追加」をクリックします。

    外部キーの一覧が表示されたテキスト・ボックスに、外部キー参照制約が挿入されます。

11. 各外部キー制約に対してこの手順を繰り返します。外部キー参照の対象には、最下位の集計レベルを選択してください。

    生成された外部キーの名前を変更するには、名前を選択して上書きします。名前はプロジェクト内で一意である必要があります。

    注意:

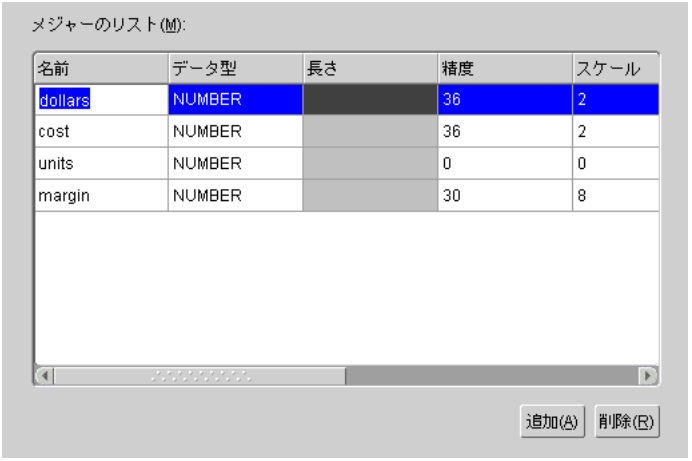
    - 新規キューブ・ウィザードには、レベル列に対する生成済の各主キー制約の名前が表示されます。このうち実際の物理的制約は、最下位レベルの主キー制約のみです。
    - 外部キー参照列の名前やデータ型は変更できません。これらを変更するには、参照先の表の定義を編集する必要があります。
    - あるディメンションの主キー制約または一意キー制約に列を追加した場合には、キューブの外部キー参照も更新する必要があります。

12. 「外部キーからの、セグメント化された一意キーの作成」の横のチェック・ボックスを選択します。

13. 「次へ」をクリックします。

    図 3-31 に示すように、「新規キューブ・ウィザード : メジャーの定義」ページが表示されます。

図 3-31「新規キューブ・ウィザード : メジャーの定義」ページ



14. 「追加」をクリックします。
15. メジャーの名前を入力します。
16. メジャーのデータ型を選択します。
17. キューブの各メジャーに対して、この手順を繰り返します。
18. 「次へ」をクリックします。

    「新規キューブ・ウィザード : 終了」ページが表示されます。このページにはキューブの定義内容が表示されます。変更する項目がある場合は「戻る」をクリックします。

19. 「終了」をクリックします。

    キューブの定義が作成され、ウェアハウス・モジュールに格納されます。さらに、ナビゲーション・ツリーに、作成した定義の名前が挿入されます。

キューブ定義の編集

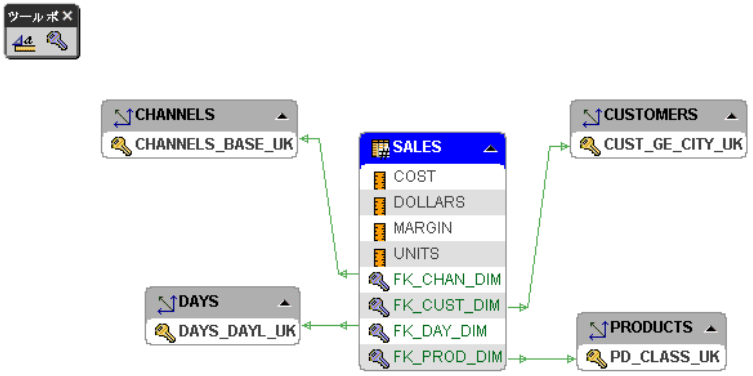
キューブ・オブジェクトには、キューブ・オブジェクト用と表用の 2 つのプロパティ・シートがあります。キューブ・オブジェクトのプロパティは、この 2 つのプロパティ・シートを編集して更新できます。また、キューブ・エディタを使用して外部キー参照またはメジャーをキューブ・オブジェクトに追加できます。キューブ・エディタでは、キューブ・プロパティおよび外部キーとディメンションの関係も変更できます。

キューブ・エディタの使用方法

キューブ・エディタを起動するには、キューブ名を右クリックし、ポップアップ・メニューから「エディタ」を選択します。キューブ・エディタが起動し、ツール・パレットと、キューブおよび関連するディメンションの図が表示されます。

図 3-32 にキューブ・エディタを示します。

図 3-32 キューブ・エディタ



図を印刷するには、キューブ・エディタのツールバーにあるプリンタ・アイコンをクリックします。

キューブ・オブジェクトのプロパティ・シートを表示する手順は次のとおりです。

- 「ファクト」メニューから「キューブ・プロパティ」を選択するか、または「プロパティ」アイコンをクリックします。
- キューブを右クリックし、「プロパティ」を選択します。

プロパティには、オブジェクトの名前と説明、外部キー参照、メジャーおよび属性セットが含まれます。

キューブ・プロパティ・シートでは、次の作業を実行できます。

- オブジェクトの名前および説明の変更。
- 外部キー参照制約の追加または削除。
- 「外部キー」タブには、あるディメンションのベースの集計レベルおよびその上の各集計レベルに定義されている、すべての一意キー制約が表示されます。Warehouse Builder では、ベースの集計レベルに定義された制約に対してのみ DDL が生成されます。
- 外部キー参照制約の名前変更。
- メジャーの追加、削除または編集（名前、データ型および説明）。

ターゲット・モジュールへのメタデータのインポート

Warehouse Builder では、メタデータ・インポート・ウィザードを使用して、ターゲット・モジュールにデータ・オブジェクトの定義をインポートできます。これらの定義は、ターゲット・システムのモデル化に役立ちます。Oracle ターゲット・モジュールの場合は、表、ビュー、外部表、順序、アドバンスト・キューおよび PL/SQL 変換パッケージの定義をインポートできます。SAP システムなど、それ以外のターゲット・モジュールの場合は、表定義をインポートできます。

Oracle ターゲット・モジュールにオブジェクト定義をインポートする手順は次のとおりです。

1. Oracle モジュールの名前を右クリックして、「インポート」を選択します。

   「メタデータ・インポート・ウィザード: ようこそ」ページが表示されます。

2. 「次へ」をクリックします。

   「メタデータ・インポート・ウィザード: フィルタ情報」ページが表示されます。

3. データ・ディクショナリの検索を次のいずれかの方法に制限します。

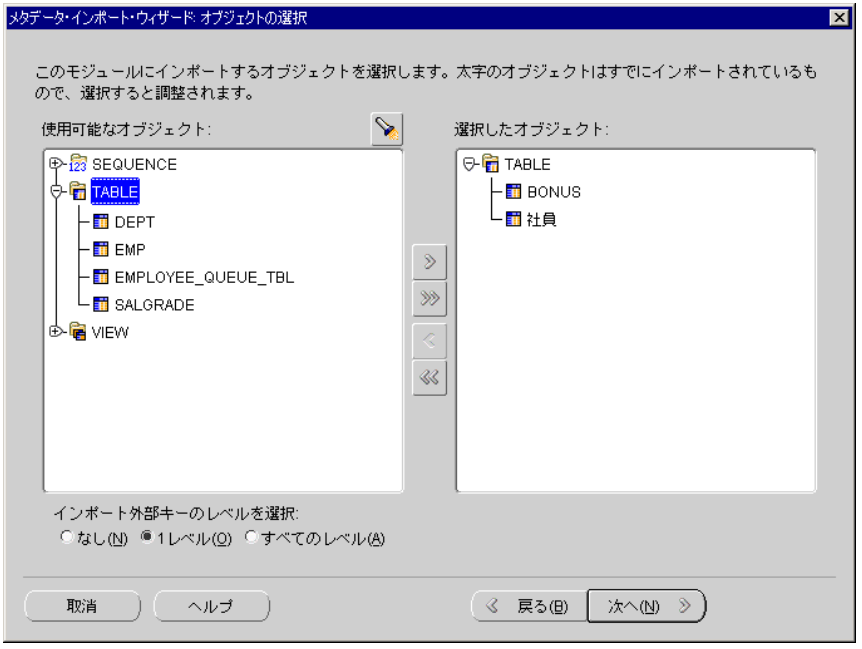
   「表」、「ビュー」、「外部表」、「順序」および「オブジェクト参照にシノニムを使用」を選択します。

   検索パターンを入力します。たとえば、プロジェクト名で始まるオブジェクトをインポートするには、目的のウェアハウス・プロジェクト名に続けて % を入力します。複数の文字に一致するワイルド・カードには % を、1 文字に一致するワイルド・カードには _ を使用します。

4. 「次へ」をクリックします。

   Warehouse Builder によって、フィルタ条件を満たす名前がデータ・ディクショナリから取得され、図 3-33 に示すように、「メタデータ・インポート・ウィザード : オブジェクトの選択」ページが表示されます。

図 3-33「メタデータ・インポート・ウィザード : オブジェクトの選択」ページ



5. インポートする項目を「使用可能なオブジェクト」リストから選択して「>」ボタンをクリックし、「選択したオブジェクト」リストに移動します。

   すべての項目を「選択したオブジェクト」リストに移動するには、「>>」ボタンをクリックします。

   1 つのオブジェクトとそのオブジェクトが参照しているオブジェクトを移動するには、オブジェクトの名前を選択して「1 レベル」を選択します。

   1 つのオブジェクトとそのオブジェクトが直接または間接的に参照しているすべてのオブジェクトを移動するには、オブジェクトの名前を選択して「すべてのレベル」を選択します。

   定義を再インポートする場合、すでにインポートされているオブジェクトは太字で表示されます。

6. 「次へ」をクリックします。

   「メタデータ・インポート・ウィザード : サマリーおよびインポート」ページが表示されます。このページには、選択した内容の要約がスプレッドシートで表示されます。名前、オブジェクトのタイプ、およびオブジェクトが調整されるか作成されるかどうかが表示されます。このページの内容を確認して、各オブジェクトの説明を追加します。

7. 「終了」をクリックします。

   「インポート結果」ページが表示されます。

8. 変更を受け入れる場合は「OK」をクリックします。インポートを取り消す場合は「元に戻す」をクリックします。

   Warehouse Builder によって、定義がターゲット・モジュールに保存されます。

オブジェクト定義を作成またはインポートして、ターゲット・システムをモデル化した後は、これらのオブジェクトを配布用に構成できます。詳細は、第 5 章「データ・オブジェクトの構成」を参照してください。ソース・システムからターゲット・システムに、データを抽出、変換およびロードする方法を定義することもできます。詳細は、第 6 章「マッピングの設計」を参照してください。

4

データ定義のインポート

この章では、Warehouse Builder でソース・モジュールを作成する方法を説明します。また、様々なデータ・ソースからソース・モジュールに定義をインポートする方法も説明します。

この章では、次のトピックについて説明します。

ソース・モジュールの作成

Warehouse Builder では、ソース・モジュールを作成して、異なるソース・システムの定義を格納します。次のタイプのソース・モジュールを作成できます。

- Oracle データベース
- Oracle 以外のデータベース
- フラット・ファイル
- SAP アプリケーション

Warehouse Builder は、ソフトウェア・インテグレータを使用してデータ定義を読み取り、ソース・システムからデータを抽出します。表 4-1 は、アプリケーションのタイプおよび対応するインテグレータをまとめたものです。

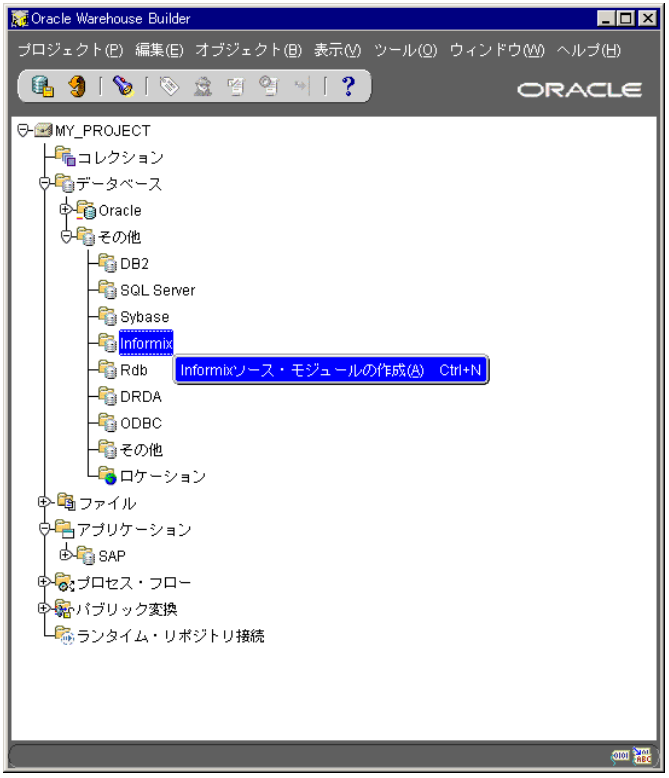
表 4-1 アプリケーションおよび対応するソフトウェア・インテグレータ

ソースのタイプ	アプリケーションのバージョンまたはシステム・タイプ	インテグレータ
Oracle データベース	■ Oracle7 リリース 7.3 ■ Oracle8 リリース 8.0 ■ Oracle8*i* リリース 8.1 ■ Oracle9*i* Database リリース 1 (9.0.1) ■ Oracle9*i* Database リリース 2 (9.2) ■ Oracle Database 10g リリース 1	OWB Integrator for Oracle DB and Apps 3.0
Oracle 以外のデータベース（Sybase、Informix、ODBC など）	Oracle Generic Gateway Connectivity	OWB Integrator for Oracle DB and Apps 3.0
SAP R/3 3.x、4.x	SAP	OWB Integrator for SAP Applications 3.0
フラット・ファイル・システム	ファイル・システム	OWB Integrator for Flat Files

作成するソース・モジュールのタイプの選択

Warehouse Builder でソース・モジュールを作成する場合、新規モジュール・ウィザードにより、選択したソース・タイプに基づいて適切なインテグレータが決定されます。この項では、図 4-1 のように、作成するソース・モジュールのタイプを、Warehouse Builder ナビゲーション・ツリーで選択する方法を説明します。

図 4-1 ソース・モジュールの作成



- **Oracle ソース・モジュール:**「データベース」ノードを開き、「Oracle」ノードを右クリックして「Oracle モジュールの作成」を選択します。詳細は、4-8 ページの「データベース・ソース・モジュールの作成」を参照してください。
- **SAP ソース・モジュール:**「アプリケーション」ノードを開き、「SAP」ノードを右クリックして「SAP R/3 ソース・モジュールの作成」を選択します。21-4 ページの「SAP モジュール定義の作成」を参照してください。
- **フラット・ファイル・モジュール:**「ファイル」ノードを右クリックして「フラット・ファイル・モジュールの作成」を選択します。4-27 ページの「フラット・ファイル・モジュールの作成」を参照してください。
- **DB2 ソース・モジュール:**「データベース」ノードを開いて「その他」ノードを開き、「DB2」ノードを右クリックして「DB2 ソース・モジュールの作成」を選択します。
- **SQL Server ソース・モジュール:**「データベース」ノードを開いて「その他」ノードを開き、「SQL Server」ノードを右クリックして「SQL Server ソース・モジュールの作成」を選択します。
- **Sybase ソース・モジュール:**「データベース」ノードを開いて「その他」ノードを開き、「Sybase」ノードを右クリックして「Sybase ソース・モジュールの作成」を選択します。
- **Informix ソース・モジュール:**「データベース」ノードを開いて「その他」ノードを開き、「Informix」ノードを右クリックして「Informix ソース・モジュールの作成」を選択します。
- **RDB ソース・モジュール:**「データベース」ノードを開いて「その他」ノードを開き、「Rdb」ノードを右クリックして「Rdb ソース・モジュールの作成」を選択します。
- **DRDA ソース・モジュール:**「データベース」ノードを開いて「その他」ノードを開き、「DRDA」ノードを右クリックして「DRDA ソース・モジュールの作成」を選択します。
- **ODBC ソース・モジュール:**「データベース」ノードを開いて「その他」ノードを開き、「ODBC」ノードを右クリックして「ODBC ソース・モジュールの作成」を選択します。
- **その他の Gateway ソース・モジュール:**「データベース」ノードを開いて「その他」ノードを開き、「その他」ノードを右クリックして「その他の Gateway ソース・モジュールの作成」を選択します。次の項「Oracle 異機種間サービス」を参照してください。

Oracle 異機種間サービス

Warehouse Builder は、Oracle 以外のシステムと通信するときに、Oracle Database 異機種間サービスおよびこれを補完するエージェントを使用します。異機種間サービスを利用すると、Oracle 以外のシステムをリモートの Oracle データベース・サーバーとして扱うことができます。エージェントとして使用できるのは、Oracle Transparent Gateway、または Oracle Database に含まれる Generic Connectivity Agent です。

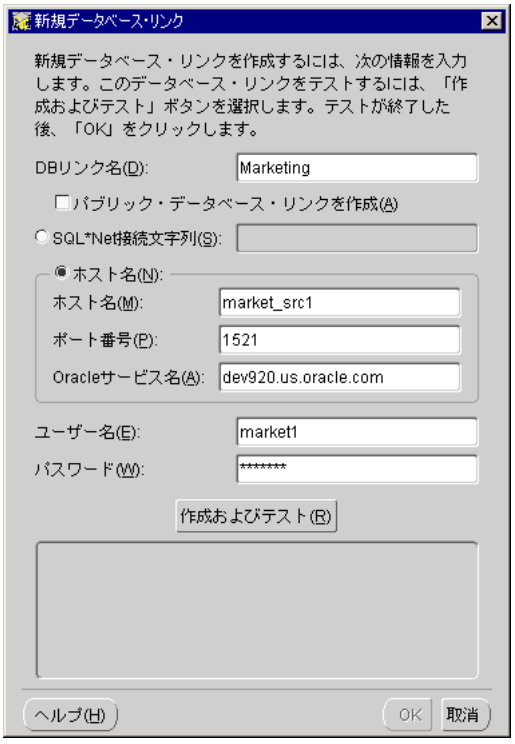
- 透過的ゲートウェイ・エージェントは、システム固有のソースです。たとえば、Sybase のデータ・ソースの場合は、エージェントは Sybase 固有の透過的ゲートウェイです。2 つのシステム間で通信できるよう、このエージェントをインストールして構成する必要があります。
- 汎用接続性の目的は、低コストのデータ統合を実現することです。データの送信は、クライアント・システムにインストールされている ODBC ドライバまたは OLE DB ドライバのルールに従って行われます。この場合、透過的ゲートウェイを購入する必要はなく、Oracle Database サーバーに含まれている Generic Connectivity Agent を使用できます。ただし、使用する Generic Connectivity Agent の初期化ファイルを作成してカスタマイズする必要があります。

データベース・ソースの接続の構成

データベース・ソースに対してソース・モジュールを作成するには、そのソース・システムを指すデータベース・リンクを Warehouse Builder Design Repository に作成するか、リポジトリから選択します。Warehouse Builder は、このリンクを使用してソースのデータ・ディクショナリにアクセスします。

データベース・リンクを指定するには、「新規モジュール・ウィザード : 接続情報」ページを使用します。既存のデータベース・リンクをドロップダウン・リストから選択して、リンクの所有者、ユーザー名および接続文字列を確認するか、図 4-2 に示すように「新規データベース・リンク」ダイアログを使用して、新しいデータベース・リンクを作成します。

図 4-2「新規データベース・リンク」ダイアログ



プライベートまたはパブリックのデータベース・リンクを作成できます。プライベート・データベース・リンクは、そのリンクを作成したユーザーしか使用できません。パブリック・データベース・リンクは、同じデータベースのすべてのリポジトリ・ユーザーが使用できます。同一リポジトリに複数のユーザーが存在する場合、デフォルトでは、CREATE PUBLIC DATABASE LINK 権限が与えられるのは所有者のみです。他のユーザーがパブリック・データベース・リンクを作成するには、この権限が付与されている必要があります。

新規データベース・リンクを作成する手順は次のとおりです。

1. データベース・リンクの名前を指定します。

   データベース・リンク名は 128 バイト以内で指定します。ピリオド（.）とアットマーク（@）を使用できます。

2. PUBLIC スキーマにデータベース・リンクを作成して他のユーザーが使用できるようにするには、「パブリック・データベース・リンクを作成」チェック・ボックスを選択します。

   デフォルトでは、このチェック・ボックスは選択されていません。

3. 「SQL*Net 接続文字列」ラジオ・ボタンまたは「ホスト名」ラジオ・ボタンを選択して、次の接続情報を入力します。

   SQL*Net 接続文字列 : データベース・システム用の接続文字列を、TNSNAMES.ORA ファイルに記載されているとおりに指定します。Oracle 以外のシステムの場合は、connect_data 句の中に「(HS-OK)」を入れて、接続文字列を指定します。

   ホスト名 : TNSNAMES.ORA ファイル内に接続文字列が存在しない場合は、「ホスト名」、「ポート番号」、および「Oracle サービス名」を入力します。

4. 接続先データベースにアクセスできるユーザー名とパスワードを入力し、「作成およびテスト」をクリックします。

   大文字と小文字を区別する場合は、ユーザー名およびパスワードを二重引用符で囲みます。BIS アプリケーションの場合は、APPS アカウントのユーザー名を入力します。

   入力した接続情報がテストされ、メッセージが表示されます。

5. 作業が終了したら、「OK」をクリックします。

   データベース・リンクのプロパティは、Warehouse Builder によってリポジトリに保存されます。データベース・リンクを作成した後は、モジュールのプロパティ・シートでリンクを編集できます。データベース・リンクおよび接続文字列の詳細は、『Oracle Database SQL リファレンス』を参照してください。

データベース・ソースについて

この項では、データベース・システムに基づくアプリケーションに接続するソース・モジュールの作成方法を説明します。ソース・モジュールは、データベース・システムからインポートされるデータ定義のコンテナです。

次の項では、データベース・ソース・システムの操作について説明します。

- データベース・ソース・モジュールの作成（4-8 ページ）
- データベースからの定義のインポート（4-11 ページ）
- Oracle データベースからの定義の再インポート（4-14 ページ）
- Oracle データベースのソース定義の更新（4-19 ページ）

関連情報は、次の項を参照してください。

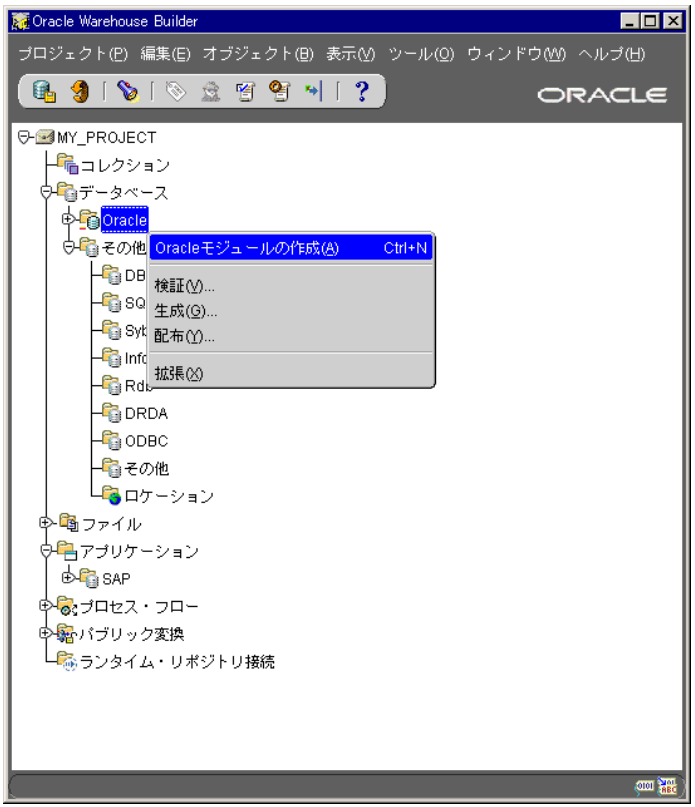
- データベース・ソースの接続の構成（4-5 ページ）
- Oracle Designer 6i/9i のソースの使用方法（4-22 ページ）

データベース・ソース・モジュールの作成

データベース・ソース・モジュールを作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「データベース」ノードを開きます。
2. Oracle データ・ソースのソース・モジュールを作成するには、図 4-3 に示すように、「Oracle」ノードを右クリックし、「Oracle モジュールの作成」を選択します。

図 4-3 Oracle ソース・モジュールの作成



新規モジュール・ウィザードにより、ウィザードを起動したナビゲーション・ツリーのノードに応じてソース・タイプが決定されます。たとえば、Informix ソース・モジュールを作成するには、「Informix」を右クリックし、「Informix ソース・モジュールの作成」を選択します。ウィザードによって、このデータ・ソースに適した Warehouse Builder インテグレータが自動的に決定されます。

「新規モジュール・ウィザード: ようこそ」ページが表示されます。

3. 「次へ」をクリックします。

   「新規モジュール・ウィザード: 名前」ページが表示されます。

4. 「新規モジュール・ウィザード: 名前」ページで、次の情報を入力します。

   モジュールの名前

   モジュールのステータス

   ステータスには、「開発」、「品質保証」または「製品」のいずれかを指定します。このステータスは、説明の目的でのみ使用されます。

   データ・ソース（モジュール・タイプ）

   説明（オプション）

5. 「次へ」をクリックします。

   「新規モジュール・ウィザード: データ・ソース情報」ページが表示されます。

6. 次の情報を選択します。

   汎用 Oracle データベース・アプリケーション**（アプリケーション）**

   アプリケーションのバージョンまたはシステム・タイプ

   選択した情報を基に、ウィザードによって適切なインテグレータが決定されます。Oracle 以外のデータベースの場合は、データベースのバージョンとして「Oracle 汎用ゲートウェイ接続性」を選択してください。

7. 「次へ」をクリックします。

   「新規モジュール・ウィザード: 接続情報」ページが表示されます。

8. メタデータのソースとして、「Oracle データ・ディクショナリ」または「Oracle Designer リポジトリ」を選択します。

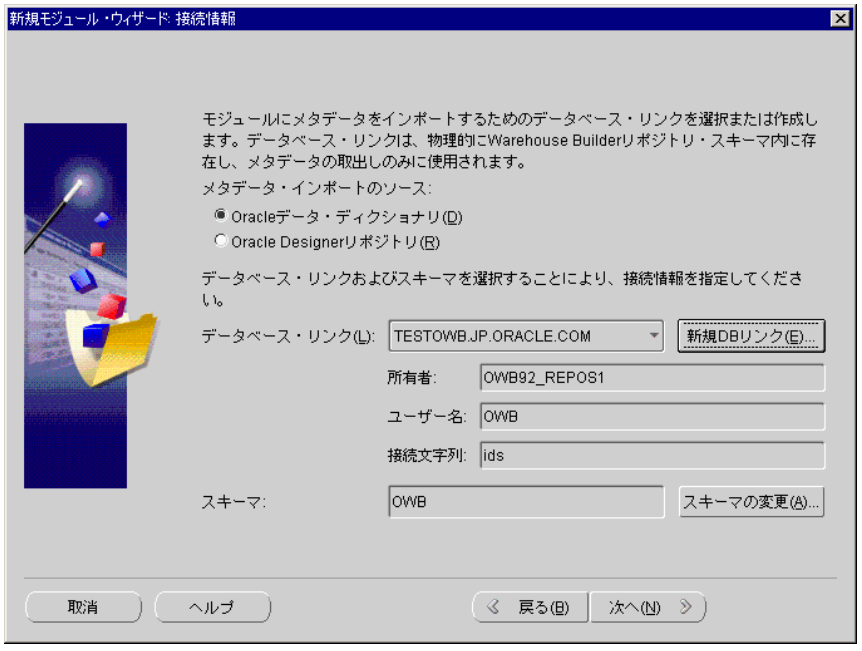
9. データベース・リンクの名前をドロップダウン・リストから選択します。

   リンクが Warehouse Builder Design Repository に存在しない場合は、「新規 DB リンク」をクリックして作成します。詳細は、4-5 ページの「データベース・ソースの接続の構成」を参照してください。

   データベース・リンクが異機種間サービス・エージェントを指していて、Oracle 以外のデータベース（ODBC、OLE DB または Oracle Transparent Gateway を経由してアクセスするシステムなど）からインポートを実行する場合には、ゲートウェイ・エージェントを指定するフィールドがこのページに表示されます。

   図 4-4 に示すように、リンク情報と、スキーマの所有者名が表示されます。

図 4-4「新規モジュール・ウィザード : 接続情報」ページ



10. スキーマ名を変更するには、「スキーマの変更」をクリックします。ユーザーの一覧が表示されます。

11. スキーマを選択して、「OK」をクリックします。

    「新規モジュール・ウィザード : 接続情報」ページのスキーマの所有者名が更新されます。

12. 「次へ」をクリックします。

    「新規モジュール・ウィザード: ロケーション」ページが表示されます。

    オブジェクトの配布先となるデータベース・スキーマまたはターゲット・ツールについての情報が、ロケーションに定義されます。ロケーションは、Oracle データベース、SAP、フラット・ファイルなど、モジュールのタイプに固有です。ロケーションを作成すると、ロケーションのタイプとバージョンを含む論理的な定義が格納されます。ロケーションが配布されると、配布先のランタイム・インスタンスから物理的な接続情報が取得され、その情報が Runtime Repository に格納されます。

13. このページを使用して、モジュールのロケーションを指定します。「新規」をクリックして新しいロケーションを作成するか、ドロップダウン・メニューであらかじめ定義したロケーションのリストから選択します。この手順はオプションです。このモジュールのロケーションは、後でオブジェクトを配布する際に作成することもできます。

14. 「次へ」をクリックします。

    「新規モジュール・ウィザード: 終了」ページには、ウィザードの各ページで入力した情報の要約が表示されます。メタデータ・インポート・ウィザードをすぐに起動する場合は、チェック・ボックスを選択します。

15. 「終了」をクリックします。

    新規モジュール・ウィザードによってソース・モジュールが作成され、その名前がプロジェクトのナビゲーション・ツリーに挿入されます。チェック・ボックスを選択した場合は、メタデータ・インポート・ウィザードが起動します。

データベースからの定義のインポート

データベースからモジュールにメタデータをインポートするには、メタデータ・インポート・ウィザードを使用します。メタデータは、Oracle データベース、Oracle 以外のデータベースまたは Designer リポジトリからインポートできます。

Oracle データ・ディクショナリから定義をインポートする手順は次のとおりです。

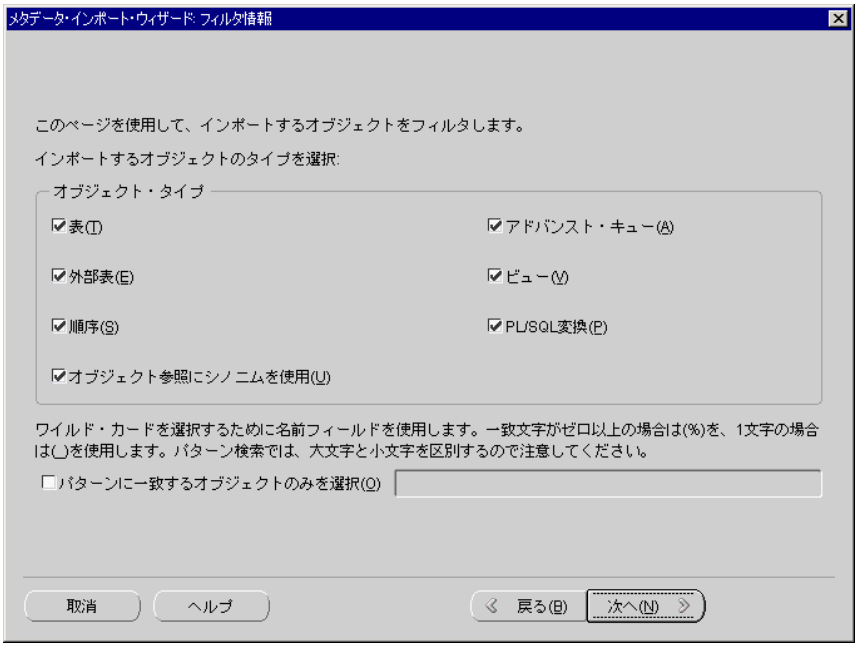
1. データ・ソース・モジュール名を右クリックして、「インポート」を選択します。

   「メタデータ・インポート・ウィザード: ようこそ」ページが表示されます。

2. 「次へ」をクリックします。

   図 4-5 に示すように、「メタデータ・インポート・ウィザード: フィルタ情報」ページが表示されます。

図 4-5「メタデータ・インポート・ウィザード : フィルタ情報」ページ



3. データ・ディクショナリの検索を次のいずれかの方法に制限します。

   「表」、「ビュー」、「外部表」、「順序」および「オブジェクト参照にシノニムを使用」を選択します。

   検索パターンを入力します。たとえば、プロジェクト名で始まるオブジェクトをインポートするには、目的のウェアハウス・プロジェクト名に続けて % を入力します。複数の文字に一致するワイルド・カードには % を、1 文字に一致するワイルド・カードには _ を使用します。

4. 「次へ」をクリックします。

   Warehouse Builder によって、フィルタ条件を満たす名前がデータ・ディクショナリから取得され、図 4-6 に示すように、「メタデータ・インポート・ウィザード : オブジェクトの選択」ページが表示されます。

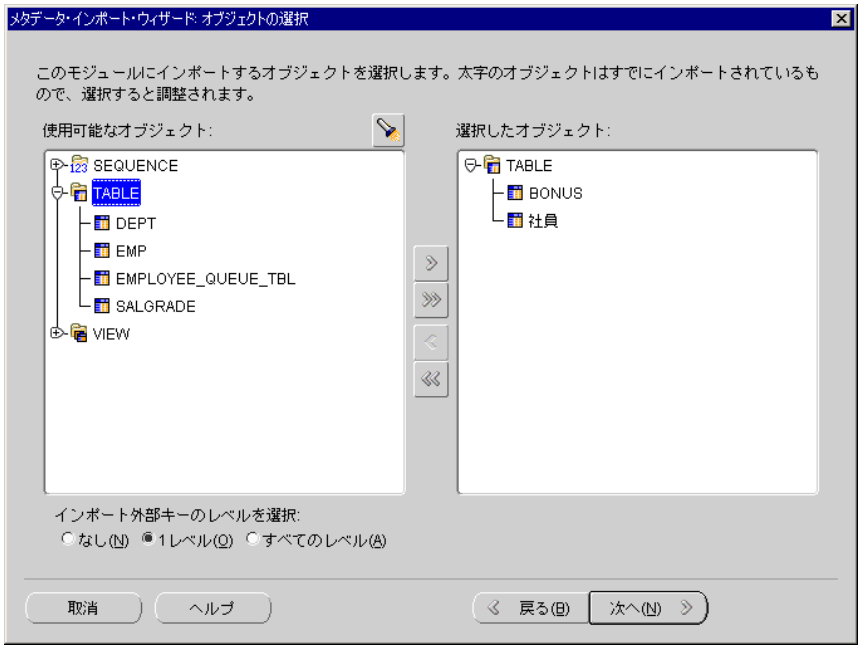
5. インポートする項目を「使用可能なオブジェクト」リストから選択して「>」ボタンをクリックし、「選択したオブジェクト」リストに移動します。

   すべての項目を「選択したオブジェクト」リストに移動するには、「>>」ボタンをクリックします。

   1 つのオブジェクトとそのオブジェクトが参照しているオブジェクトを移動するには、オブジェクトの名前を選択して「1 レベル」を選択します。

   1 つのオブジェクトとそのオブジェクトが直接または間接的に参照しているすべてのオブジェクトを移動するには、オブジェクトの名前を選択して「すべてのレベル」を選択します。

図 4-6「メタデータ・インポート・ウィザード : オブジェクトの選択」ページ



定義を再インポートする場合、すでにインポートされているオブジェクトは太字で表示されます。

6. 「次へ」をクリックします。

   「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページが表示されます。このページには、選択した内容の要約がスプレッドシートで表示されます。名前、オブジェクトのタイプ、およびオブジェクトが調整されるか作成されるかどうかが表示されます。このページの内容を確認して、各オブジェクトの説明を追加します。

7. 「終了」をクリックします。

   「インポート結果」ページが表示されます。

8. 変更を受け入れる場合は「OK」をクリックします。インポートを取り消す場合は「元に戻す」をクリックします。

   Warehouse Builder によって、定義がソース・モジュールに保存されます。

関連情報は、次の項を参照してください。

- Oracle データベースからの定義の再インポート（4-14 ページ）
- データベース・ソース・モジュールの作成（4-8 ページ）
- Oracle Designer 6i/9i のソースの使用方法（4-22 ページ）

Oracle データベースからの定義の再インポート

ソース・データベースの定義を再インポートすると、前回のインポート以降にソース・メタデータに加えられた変更をインポートできます。元の定義をリポジトリから削除する必要はありません。Warehouse Builder には、前回のインポート以降に定義に対して行った変更をすべて保持するオプションも用意されています。このような変更には、Warehouse Builder で作成した新しいオブジェクト、外部キー、リレーションシップ、説明などがあります。

定義を再インポートする手順は次のとおりです。

1. データ・ソース・モジュール名を右クリックして、「インポート」を選択します。

   「メタデータ・インポート・ウィザード: ようこそ」ページが表示されます。

2. 「次へ」をクリックします。

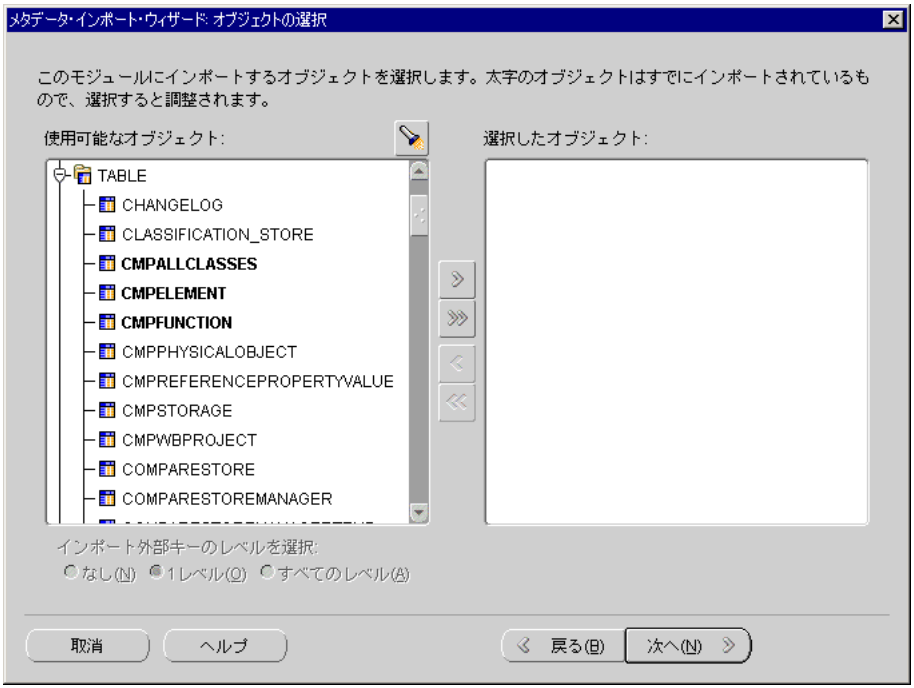
   「メタデータ・インポート・ウィザード: フィルタ情報」ページが表示されます。

3. 再インポートするオブジェクト・タイプを選択します。同じオブジェクトが確実に再インポートされるようにするために、前回インポートしたときと同じ設定を選択してください。

4. 「次へ」をクリックします。

   「メタデータ・インポート・ウィザード: オブジェクトの選択」ページが表示されます。図 4-7 に示すように、すでにインポートされているオブジェクトは太字で表示されます。

図 4-7「メタデータ・インポート・ウィザード : オブジェクトの選択」ダイアログ



5. インポートされているオブジェクトを選択して矢印ボタンをクリックし、「選択したオブジェクト」リストに移動します。

6. 「次へ」をクリックします。

   図 4-8 に示すように、「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページが表示されます。再インポートするオブジェクトのアクションは、「調整」と表示されます。

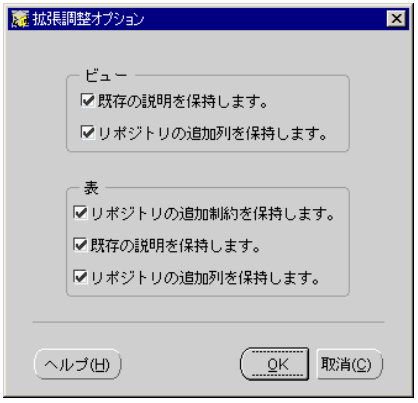
   再インポートするオブジェクトに関連付けられた新しいオブジェクトがソースに存在する場合は、新しいオブジェクトも同時にインポートする必要があります。このようなオブジェクトのアクションは、「作成」と表示されます。

図 4-8「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページに表示される調整アクション

7. 「拡張調整オプション」をクリックして、拡張調整オプションを選択します。

   図 4-9 に示すように、「拡張調整オプション」ダイアログが表示されます。

図 4-9「拡張調整オプション」ダイアログ



このダイアログを使用すると、Warehouse Builder Design Repository でオブジェクト定義に行った編集と追加の内容をすべて保持できます。

ビューを調整する次のオプションを選択します。

既存の説明を保持します。: リポジトリに保存された説明が保持されます。

リポジトリの追加列を保持します。: Warehouse Builder に追加した列が保持されます。

表を調整する次のオプションを選択します。

リポジトリの追加制約を保持します。: Warehouse Builder の表に追加した制約が保持されます。

既存の説明を保持します。: リポジトリに保存された説明が保持されます。

リポジトリの追加列を保持します。: Warehouse Builder に追加した列が保持されます。

デフォルトでは、すべてのオプションが選択されています。チェック・ボックスの選択を解除すると、リポジトリ・オブジェクトは保持されず、置き換えられます。

たとえば、表やビューを初めてインポートした後は、その表やビューの定義に手動で説明を追加します。表やビューの定義を再インポートする際にこのような説明が上書きされないようにするには、「既存の説明を保持します」オプションを選択する必要があります。これにより、説明は上書きされなくなります。

8. オプションを選択したら、「OK」をクリックします。
9. 「終了」をクリックします。

   Warehouse Builder によって、オブジェクトが調整および作成されます。この処理が完了すると、図 4-10 に示すように、「インポート結果」ダイアログが表示されます。

図 4-10「インポート結果」ダイアログ

このダイアログには、各オブジェクトに対して Warehouse Builder が実行したアクションが表示されます。

レポートを保存するには、「保存」をクリックします。必ず再インポート処理に固有のネーミング規則を使用してください。

10. 次に進むには、「OK」をクリックします。

    リポジトリに対する変更をすべて元に戻すには、「元に戻す」をクリックします。

Oracle データベースのソース定義の更新

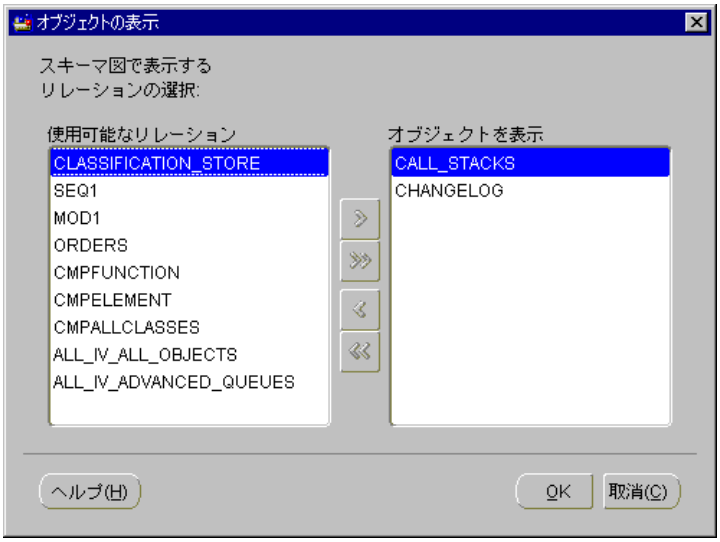
Oracle ソース・モジュール・エディタを使用すると、Oracle ソース・モジュール内のスキーマを表示または印刷したり、オブジェクトの系統レポートや影響分析レポートを実行できます。

Oracle ソース・モジュール・エディタを表示する手順は次のとおりです。

1. ソース・モジュールの名前をダブルクリックします。

   図 4-11 に示すように、「オブジェクトの表示」ダイアログが表示されます。

図 4-11「オブジェクトの表示」ダイアログ

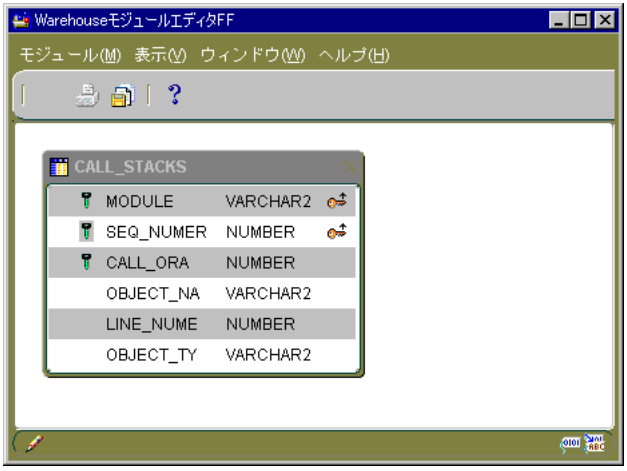


2. Oracle ソース・モジュール・エディタのキャンバスに、図で表示するオブジェクトを選択します。

3. 「OK」をクリックします。

   図 4-12 に示すように、Oracle ソース・モジュール・エディタが表示されます。

図 4-12 ソース・モジュール・エディタ



ソース定義の図を印刷するには、エディタのツールバーにある「印刷」アイコンをクリックします。

ソース定義の更新

ソース定義のプロパティを更新するには、プロパティ・シートを編集します。プロパティ・シートを表示するには、ナビゲーション・ツリーでモジュール名を右クリックし、ポップアップ・メニューから「プロパティ」を選択します。

接続の更新

データ・ソースの接続情報を更新するには、「モジュール・プロパティ」ウィンドウの「接続」タブを選択します。図 4-13 に示すように、ドロップダウン・リストから別のデータベース・リンクを選択できます。

図 4-13 モジュールのプロパティ・シートの「接続」タブ

接続情報を変更すると、ソース・モジュールの既存の定義を損なう可能性があることを示す Warehouse Builder の警告メッセージが表示されます。接続を変更するには、「OK」をクリックします。

ロケーションの更新

データ・ソースのロケーションを更新するには、「モジュール・プロパティ」ウィンドウの「ロケーション」タブを選択します。ソース・モジュールに対して複数のロケーションを定義している場合は、ドロップダウン・リストから別のロケーションを選択できます。

ロケーションの詳細は、13-3 ページの「Runtime Repository 接続の定義」を参照してください。

Oracle Designer 6*i*/9*i* のソースの使用方法

Warehouse Builder では、Oracle Designer リポジトリに接続するソース・モジュールを作成できます。アプリケーションの定義が Oracle Designer リポジトリで保存および管理されている場合は、そのアプリケーションへの接続に必要な時間を短縮できます。

Designer 6*i*/9*i* のリポジトリは、ワークエリアを使用してオブジェクトのバージョンを制御します。リポジトリ・オブジェクトのバージョンを指定するには、ワークエリアを選択します。Designer 6*i*/9*i* では、ワークエリア内でオブジェクトをグループ化して、アプリケーション・システムを作成することもできます。アプリケーション・システムには、名前空間の定義と、オブジェクトの所有権情報が含まれています。アプリケーション・システムを利用すると、別のユーザーが所有するオブジェクトも参照できます。Designer 6*i*/9*i* のアプリケーション・システムはワークエリアによって制御されるため、バージョン管理の対象となります。ワークエリアとアプリケーション・システムの詳細は、Designer 6*i*/9*i* のドキュメントを参照してください。

Designer 6*i*/9*i* のワークエリアまたはアプリケーション・システムで参照可能なオブジェクトは、すべて Warehouse Builder でデータ・ソースとして使用できます。Designer 6*i*/9*i* のオブジェクトを Warehouse Builder のソースとして選択するには、次のいずれかを行います。

- ワークエリアの指定
- ワークエリア内のアプリケーション・システムの指定

新規モジュール・ウィザードによって、データベース・リンクで使用可能な Designer のバージョンが検出されます。Designer 6*i*/9*i* が検出された場合は、「新規モジュール・ウィザード : 接続情報」ページが変更され、「ワークエリア」フィールドと「コンテナ要素」フィールドおよび各フィールドの変更ボタンが表示されます。

「変更」をクリックすると、新規モジュール・ウィザードの画面に選択リストが表示されます。このリストから、ワークエリアまたはアプリケーション・システムを選択します。インポート可能なリポジトリ・オブジェクトのリストを決定する基準は次のとおりです。

- Warehouse Builder でオブジェクト・タイプがサポートされていること（表、ビュー、順序、シノニム）。
- 指定されたワークエリア内でオブジェクトがアクセス可能であること。これにより、アクセスするオブジェクトのバージョンが決まります。
- 指定されたアプリケーション・システムでオブジェクトが参照可能であること。リストには、指定されたアプリケーション・システムが所有するオブジェクトと、そのアプリケーション・システムが所有していないが共有しているその他のオブジェクトが表示されます。

Designer 6*i*/9*i* のソースから定義をインポートするには、データベース・ソースからの定義のインポートで説明されている手順（4-11 ページ）に従う必要があります。

Designer 6*i*/9*i* をメタデータ・ソースとして使用する方法

Designer 6*i*/9*i* のソース・モジュールを作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「データベース」ノードを開きます。
2. 「Oracle」ノードを右クリックして「Oracle モジュールの作成」を選択します。

   「新規モジュール・ウィザード : ようこそ」ページが表示されます。
3. 「次へ」をクリックします。

   「新規モジュール・ウィザード : 名前」ページが表示されます。
4. 「新規モジュール・ウィザード : 名前」ページで、次の情報を入力します。

   モジュールの名前

   モジュールのステータス

   ステータスには、「開発」、「品質保証」または「製品」のいずれかを指定します。このステータスは、説明の目的でのみ使用されます。

   データ・ソース（モジュール・タイプ）

   説明（オプション）
5. 「次へ」をクリックします。

   「新規モジュール・ウィザード : データ・ソース情報」ページが表示されます。ウィザードを起動したナビゲーション・ツリーのノードに基づいて、適切なインテグレータが決定されます。
6. 「次へ」をクリックします。

   図 4-14 に示すように、「新規モジュール・ウィザード : 接続情報」ページが表示されます。
7. メタデータのインポートのソースとして「Oracle Designer リポジトリ」を選択します。
8. Designer 6*i* リポジトリへのデータベース・リンクの名前をリストから選択します。

   リンクが Warehouse Builder Design Repository に存在しない場合は、「新規 DB リンク」をクリックして作成します。詳細は、4-5 ページの「データベース・ソースの接続の構成」を参照してください。

   Designer 6*i*/9*i* のリポジトリに接続する場合は、その Designer リポジトリの所有者として接続する必要があります。

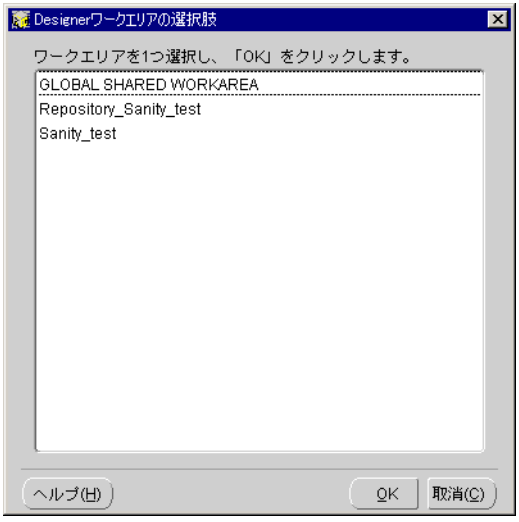
図 4-14「新規モジュール・ウィザード : 接続情報」ページ

新規モジュール・ウィザードによって Designer 6*i* のリポジトリが検出された場合は、「ワークエリア」フィールド、「コンテナ要素」フィールドおよび「変更」ボタンが使用可能になります。

Warehouse Builder で Designer 6*i* リポジトリのオブジェクトを選択するには、ワークエリアを指定する必要があります。

9. ワークエリアを指定するには、「変更」をクリックして、図 4-15 に示すように、選択リストからワークエリアを指定します。

図 4-15 ワークエリア選択リスト



10. アプリケーション・システムを指定するには、「変更」をクリックして、図 4-16 に示すように、選択リストからアプリケーション・システムを選択します。

図 4-16 アプリケーション・システム選択リスト

11. 「次へ」をクリックします。

    「新規モジュール・ウィザード: 終了」ページが表示されます。このページには、ウィザードの各ページで入力した情報の要約が表示されます。情報を確認します。

12. 「終了」をクリックします。

    新規モジュール・ウィザードによってソース・モジュールが作成され、その名前がプロジェクトのナビゲーション・ツリーに挿入されます。

    Designer 6*i* のソースからこのソース・モジュールに定義をインポートするには、データベース・ソースからの定義のインポートで説明されている手順（4-11 ページ）に従う必要があります。

関連情報は、次の項を参照してください。

- データベースからの定義のインポート（4-11 ページ）
- Oracle データベースからの定義の再インポート（4-14 ページ）

フラット・ファイル・モジュールについて

プロジェクトによっては、フラット・ファイルからデータを抽出したり、フラット・ファイルにデータを書き込むことが必要になります。フラット・ファイルには、直接、または外部表演算子を使用してアクセスできます。この項では、フラット・ファイル・モジュールの基本的な作成方法を中心に説明します。この情報は、Warehouse Builder から外部表を作成する際の基礎知識にもなります。外部表の詳細は、3-27 ページの「外部表の使用方法」を参照してください。

フラット・ファイルを使用する場合は、フラット・ファイルと外部表の両方を理解しておくと、どちらの機能を使用するかを判断する際に役立ちます。大量のデータをロードする場合、フラット・ファイルへのロードでは、優れたパフォーマンスを実現するダイレクト・パス・ロード・オプションを使用できます。大量のデータをロードしない場合は、外部表に適用できる多数のリレーショナル変換を使用するほうが適しています。詳細は、3-28 ページの「外部表とフラット・ファイル演算子」を参照してください。

データ・ファイルに直接アクセスする場合で、Warehouse Builder Design Client とデータ・ファイルが、異なるタイプのオペレーティング・システム上に常駐する場合は、システム管理者に連絡して、NFS などのネットワーク・プロトコル経由で必要な接続を確立できるようにしてください。Warehouse Builder Design Client とデータ・ファイルが、Windows オペレーティング・システム上に常駐する場合は、Warehouse Builder Design Client のマシンからローカルにアクセスできるドライブにデータ・ファイルを格納してください。

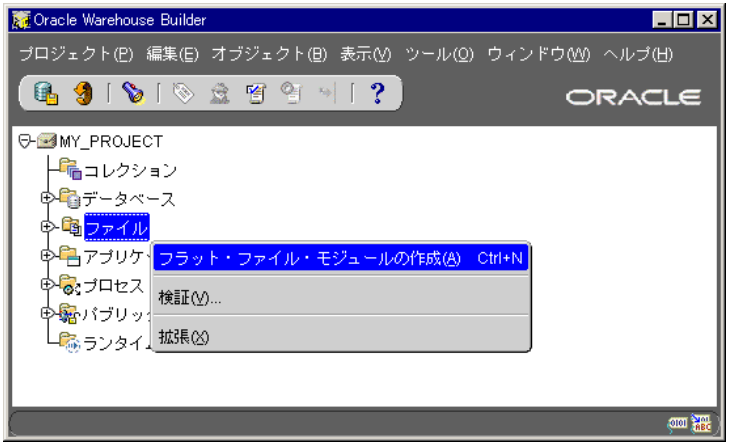
フラット・ファイル・モジュールの作成

フラット・ファイル・モジュールを作成した後は、Warehouse Builder にフラット・ファイルの定義をインポートできます。Warehouse Builder へのフラット・ファイルのインポートの詳細は、4-28 ページの「フラット・ファイルのソースとターゲットについて」を参照してください。

フラット・ファイル・モジュールを作成する手順は次のとおりです。

1. 図 4-17 に示すように、Warehouse Builder コンソールで「ファイル」ノードを右クリックして、「フラット・ファイル・モジュールの作成」を選択します。

図 4-17 フラット・ファイル・モジュールの作成



「新規モジュール・ウィザード: ようこそ」ページが表示されます。

2. 「次へ」をクリックします。

「新規モジュール・ウィザード: 名前」ページが表示されます。

3. 「新規モジュール・ウィザード: 名前」ページでモジュールの名前を入力し、モジュールのステータスを指示します。ステータスには、「開発」、「品質保証」または「製品」のいずれかを指定します。このステータスは、説明の目的でのみ使用されます。モジュールの説明を入力します（オプション）。

4. 「次へ」をクリックします。

「新規モジュール・ウィザード: 接続情報」ページが表示されます。

5. ファイルが格納されたディレクトリの完全修飾名を入力します。必要に応じてドライブの文字も入力します。

6. 「次へ」をクリックします。

   「新規モジュール・ウィザード: ロケーション」ページが表示されます。

   オブジェクトの配布先となるデータベース・スキーマまたはターゲット・ツールについての情報が、ロケーションに定義されます。ロケーションは、Oracle データベース、SAP、フラット・ファイルなど、モジュールのタイプに固有です。ロケーションを作成すると、ロケーションのタイプとバージョンを含む論理的な定義が格納されます。ロケーションが配布されると、配布先のランタイム・インスタンスから物理的な接続情報が取得され、その情報が Runtime Repository に格納されます。

7. このページを使用して、モジュールのロケーションを指定します。「新規」をクリックして新しいロケーションを作成するか、ドロップダウン・メニューであらかじめ定義したロケーションのリストから選択します。この手順はオプションです。このモジュールのロケーションは、後でオブジェクトを配布する際に作成することもできます。

8. 「次へ」をクリックします。

   「新規モジュール・ウィザード: 終了」ページには、ウィザードの各ページで入力した情報の要約が表示されます。メタデータ・インポート・ウィザードをすぐに起動する場合は、チェック・ボックスを選択します。

9. 「終了」をクリックします。

   新規モジュール・ウィザードによってフラット・ファイル・モジュールが作成され、その名前がプロジェクトのナビゲーション・ツリーに挿入されます。チェック・ボックスを選択した場合は、メタデータ・インポート・ウィザードが起動します。

フラット・ファイルのソースとターゲットについて

フラット・ファイルの定義を作成する前に、まずファイル・モジュールとファイルのロケーションを作成する必要があります。フラット・ファイルの定義を作成するには、次のウィザードを使用します。

- **メタデータ・インポート・ウィザード。**インポートするフラット・ファイルの選択には、メタデータ・インポート・ウィザードを使用します。
- **フラット・ファイル・サンプル・ウィザード。**メタデータ・インポート・ウィザードでフラット・ファイルの選択が終了すると、メタデータ・インポート・ウィザードによって、フラット・ファイル・サンプル・ウィザードが起動します。フラット・ファイル・サンプル・ウィザードは、フラット・ファイルのサンプルを表示し、レコード編成とファイル・プロパティを定義するために使用します。

ファイル定義は、Warehouse Builder にインポートした後でも更新できます。また、Warehouse Builder にフラット・ファイルの構造をインポートした後も、ソースまたはターゲットとして、そのフラット・ファイルのデータを使用できます。

この項では、次のトピックについて説明します。

- メタデータ・インポート・ウィザードのフラット・ファイル関連機能の使用方法（4-29 ページ）
- フラット・ファイル・サンプル・ウィザードの使用方法（4-31 ページ）

関連情報は、次の項を参照してください。

- フラット・ファイル・モジュールの作成（4-27 ページ）
- ファイル定義の更新（4-49 ページ）
- フラット・ファイルのマッピング演算子（8-28 ページ）
- 演算子とリポジトリ・オブジェクトの調整（6-41 ページ）

メタデータ・インポート・ウィザードのフラット・ファイル関連機能の使用方法

次の項では、メタデータ・インポート・ウィザードを使用してフラット・ファイルの定義を作成する方法を説明します。

フラット・ファイルをインポートする手順は次のとおりです。

1. ファイル・モジュールを右クリックして、「インポート」を選択します。

   「メタデータ・インポート・ウィザード: ようこそ」ページが表示されます。

2. 「次へ」をクリックします。

   「メタデータ・インポート・ウィザード: フィルタ情報」ページが表示されます。このページには、ファイル名をフィルタするオプションがあります。

   すべてのデータ・ファイル: このオプションを選択すると、フラット・ファイル・モジュールに指定したディレクトリで使用できるデータ・ファイルが、すべて返されます。

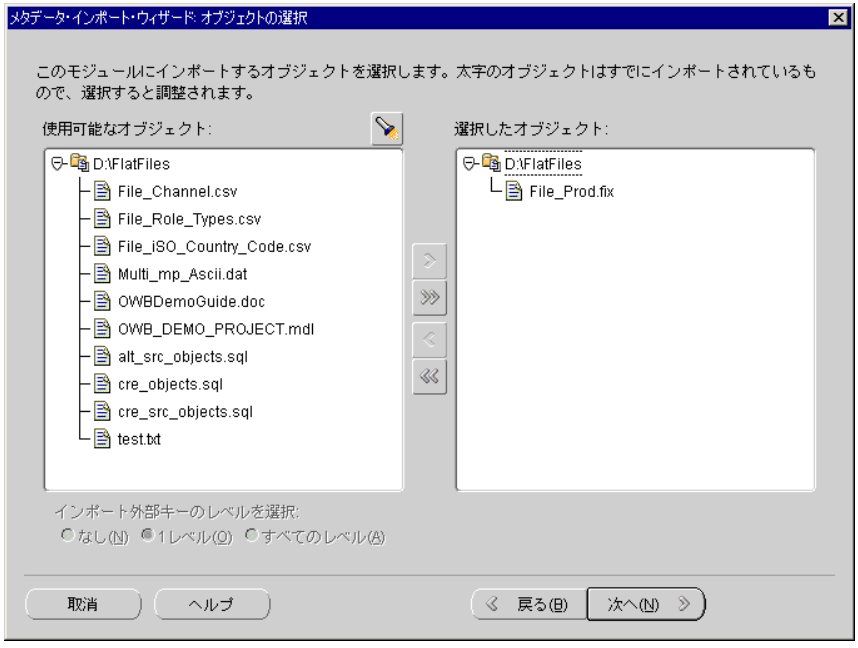
   次のパターンに一致するデータ・ファイル: このオプションを使用すると、入力したパターンに一致するデータ・ファイルのみが選択されます。たとえば、このオプションを選択して (*.dat) と入力すると、ファイル拡張子が .dat のファイルのみが、ウィザードの次のページに表示されます。フィルタ文字列の一部に % を入力すると、複数の文字に一致するワイルドカードと解釈されます。フィルタ文字列の一部に _ を入力すると、1 つの文字に一致するワイルドカードと解釈されます。

3. 「次へ」をクリックします。

   図 4-18 に示すように、「メタデータ・インポート・ウィザード : オブジェクトの選択」ページが表示されます。

   Warehouse Builder では、フラット・ファイルのインバウンド調整は実行されないため、他のオブジェクトが再インポートされた場合と違って、使用可能なオブジェクトは太字では表示されません。フラット・ファイルを再インポートする場合は、そのフラット・ファイルのオブジェクトをもう一度サンプリングする必要があります。

図 4-18「メタデータ・インポート・ウィザード : オブジェクトの選択」ページ



4. 定義するファイルの名前を、左側の「使用可能なオブジェクト」から右側の「選択したオブジェクト」ウィンドウ・ペインに移動します。

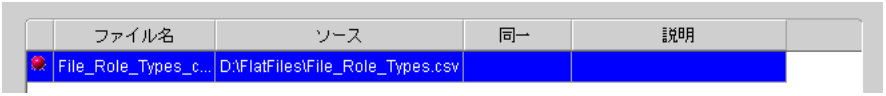
5. 「次へ」をクリックします。

   「メタデータ・インポート・ウィザード : サマリーおよびインポート」ページが表示されます。このページの左端の列には、図 4-19 に示すように、Warehouse Builder にそのファイルのメタデータが存在するかどうかのステータスを表す丸印があります。

   ステータスを表す丸印が赤の場合、Warehouse Builder にはそのメタデータが存在しません。次の手順に進んでください。

6. 「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページの各ファイルについて、「サンプル」をクリックするか、フォーマットの一致するファイルを「同一」リスト・ボックスから選択します。ファイル・フォーマットが、「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページの別のファイル・フォーマットと一致する場合は、「同一」リスト・ボックスからそのファイル名を選択できます。

図 4-19「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページに表示されるファイル・ステータス



7. 赤い丸印が表示されているファイルを選択して、「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページの下部にある「サンプル」をクリックします。

   「フラット・ファイル・サンプル・ウィザード: ようこそ」ページが表示されます。フラット・ファイル・サンプル・ウィザードを完了します。フラット・ファイル・サンプル・ウィザードの詳細は、「フラット・ファイル・サンプル・ウィザードの使用方法」を参照してください。

8. フラット・ファイル・サンプル・ウィザードで 1 つのファイルの入力が完了すると、「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページに戻ります。これで、サンプルしたファイルのステータスを表す丸印が緑色になります。

9. インポートするすべてのファイルでステータスを表す丸印が緑色になったら、「終了」をクリックします。Warehouse Builder によってファイルの定義が作成され、ソース・モジュールに保存されます。また、フォーマットの名前がソース・モジュールのナビゲーション・ツリーに挿入されます。

フラット・ファイル・サンプル・ウィザードの使用方法

メタデータ・インポート・ウィザードを使用して、既存のフラット・ファイルからデータをサンプリングするたびに、メタデータ・インポート・ウィザードによって、フラット・ファイル・サンプル・ウィザードが起動します。フラット・ファイル・サンプル・ウィザードは、フラット・ファイルのメタデータを定義する際の補助として使用します。フラット・ファイル・サンプル・ウィザードは、定義したメタデータを Warehouse Builder Design Repository に保存します。

サンプルされたフラット・ファイルのオブジェクトをマッピングで利用するには、新規外部表ウィザードを実行し、このフラット・ファイルに基づいて外部表を作成する必要があります。この新しい外部表は、その後外部表演算子としてマッピングに使用できます。外部表の使用方法とフラット・ファイル演算子の使用方法の詳細は、3-28 ページの「外部表とフラット・ファイル演算子」を参照してください。

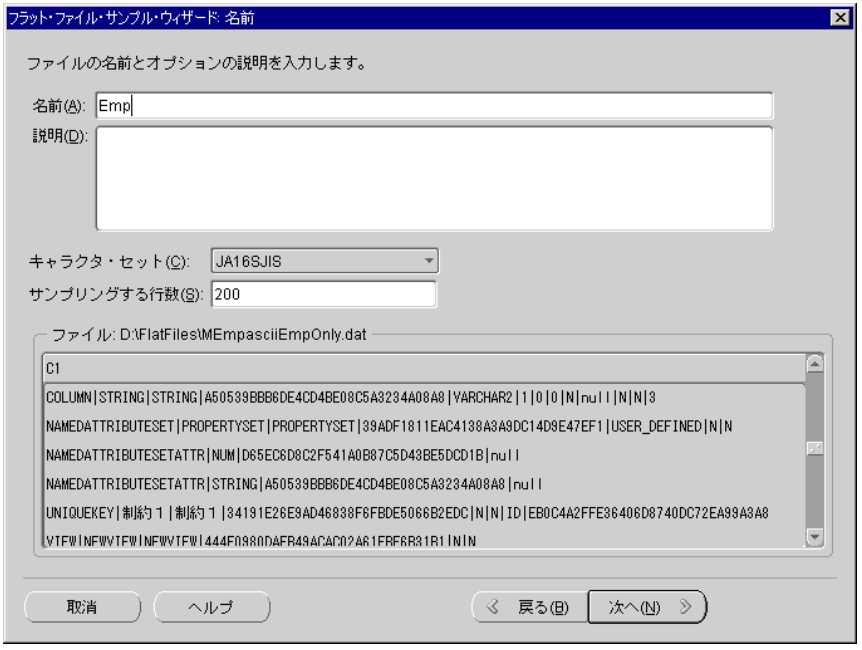
フラット・ファイル・サンプル・ウィザードに従って作業すると、次の手順が完了します。

- フラット・ファイルの記述
- レコード編成の選択
- ファイル・レイアウトの選択
- ファイル・フォーマットの指定
- レコード・タイプの選択（マルチ・レコード・タイプのファイルのみ）
- フィールド長の指定（固定長ファイルのみ）
- フィールド・プロパティの指定

フラット・ファイルの記述

サンプリングするフラット・ファイルを記述するには、図 4-20 に示す「フラット・ファイル・サンプル・ウィザード : 名前」ページを使用します。

図 4-20「フラット・ファイル・サンプル・ウィザード : 名前」ページ



- **名前 :** ファイルのインポート後に、Warehouse Builder がそのファイルの参照に使用する名前です。デフォルトでは、ソース・ファイルの名前に基づき、このウィザードによって名前が作成されます。このページでファイル名を変更できます。

  ファイルの名前を変更する場合は、名前にスペースや記号は使用しないでください。アンダースコアは使用できます。大文字と小文字を使用できます。名前の先頭に数字を使用しないでください。Warehouse Builder の予約語は使用しないでください。予約語の一覧は、付録 B「予約語」を参照してください。

- **説明 :** ファイルの説明を入力できます（オプション）。

- **キャラクタ・セット :** キャラクタ・セットは、データベース・オブジェクトとファイルで表現できる言語を決定します。Warehouse Builder Design Client では、デフォルトのグローバリゼーション・サポートまたは National Language Support（NLS）キャラクタ・セットは Warehouse Builder のホスト・マシンに定義されたキャラクタ・セットと同じです。このキャラクタ・セットがソース・ファイルのキャラクタ・セットと異なる場合は、データのサンプルが判読不可能になることがあります。データ・サンプルをソースに固有なキャラクタ・セットで表示するには、ドロップダウン・リストからそのキャラクタ・セットを選択します。NLS キャラクタ・セットの詳細は、『Oracle Database グローバリゼーション・サポート・ガイド』を参照してください。

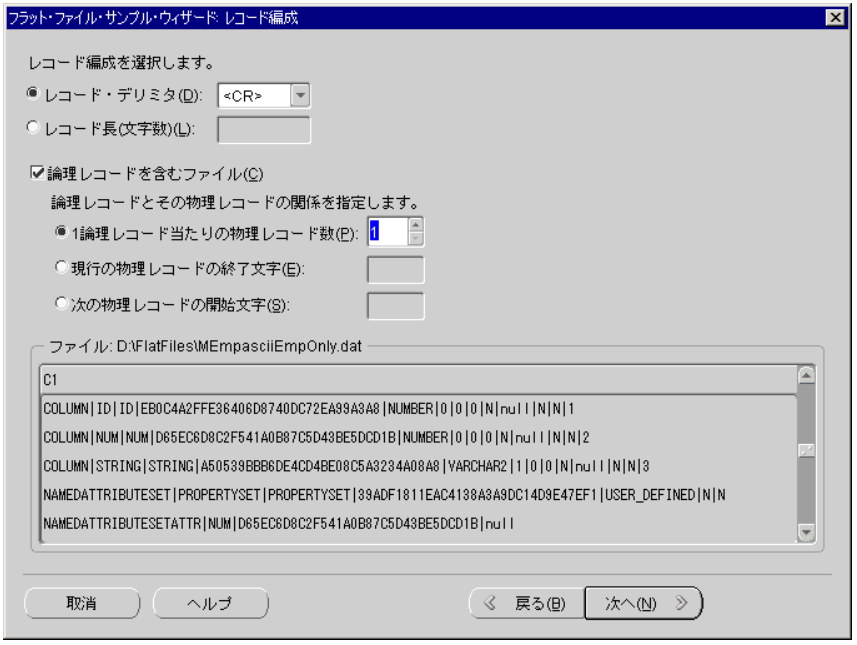
- **サンプリングする行数 :** データ・ファイルからサンプリングする行数を指定できます。デフォルトでは、最初の 200 行がサンプリングされます。このフィールドに最適な値を決定するには、4-38 ページの「例 : レコード・タイプが複数あるフラット・ファイル」を参照してください。

ファイルの設定情報の入力が完了したら、「次へ」をクリックしてウィザードを続行します。

レコード編成の選択

サンプリングするファイルのレコード編成を指定するには、図 4-21 に示す「フラット・ファイル・サンプル・ウィザード : レコード編成」ページを使用します。

図 4-21「フラット・ファイル・サンプル・ウィザード : レコード編成」ページ



ファイル内の各レコードの長さを決定する方法を指定するには、次の 2 つのオプションのいずれかを選択します。

- **レコード・デリミタ :** このオプションは、各レコードの終了がデリミタで指定される場合に選択してください。次に、レコード・デリミタを指定します。デフォルトのレコード・デリミタである改行（<CR>）をそのまま使用するか、新しい値を入力できます。
- **レコード長 (文字数):** このオプションは、ファイル内の各レコードの長さが同一の場合に選択してください。次に、各レコードの文字数を指定します。マルチバイト・キャラクタのファイルでは、マルチバイト・キャラクタを 1 文字として計算します。

論理レコードの指定

フラット・ファイル・サンプル・ウィザードでは、複数の物理レコードに対応する論理レコードで構成されたファイルをサンプリングできます。ファイルに論理レコードが含まれている場合は、「論理レコードを含むファイル」をクリックします。次に、ファイルを記述するオプションを 1 つ選択します。

下部のパネルの論理レコードの表示が更新され、選択内容が反映されます。デフォルトでは、1 レコード当たり 1 物理レコードのオプションが選択されています。

論理レコード情報の入力が完了したら、「次へ」をクリックしてウィザードを続行します。

- **1 論理レコード当たりの物理レコード数 :** データ・ファイルには、論理レコードごとに固定された数の物理レコードが含まれます。

  ```
  PHYSICAL_RECORD1
  PHYSICAL_RECORD2
  PHYSICAL_RECORD3
  PHYSICAL_RECORD4
  ```

  前述の例で、論理レコード当たりの物理レコード数が 2 の場合は、1 つの論理レコードに PHYSICAL_RECORD1 と PHYSICAL_RECORD2 が 1 つの論理レコードを形成し、PHYSICAL_RECORD3 と PHYSICAL_RECORD4 が 2 番目の論理レコードを形成します。

- **現行の物理レコードの終了文字 :** データ・ファイルには、数が可変の物理レコードが含まれています。さらに、各物理レコードの末尾には、そのレコードが次の物理レコードと関連することを表す継続文字が付加されています。

  次の例では、継続文字はレコード末尾のパーセント記号（%）です。

  ```
  PHYSICAL_RECORD1%
  PHYSICAL_RECORD2       end log rec 1
  PHYSICAL_RECORD3%
  PHYSICAL_RECORD4       end log rec 2
  ```

- **次の物理レコードの開始文字 :** データ・ファイルには、数が可変の物理レコードが含まれています。さらに、各物理レコードの先頭には、そのレコードが前の物理レコードと関連することを表す継続文字が付加されています。

  次の例は、レコードの先頭に継続文字を持つ 2 つの論理レコードを示しています。

  ```
  PHYSICAL_RECORD1
  %PHYSICAL_RECORD2      end log rec1
  PHYSICAL_RECORD3
  %PHYSICAL_RECORD4      end log rec 2
  ```

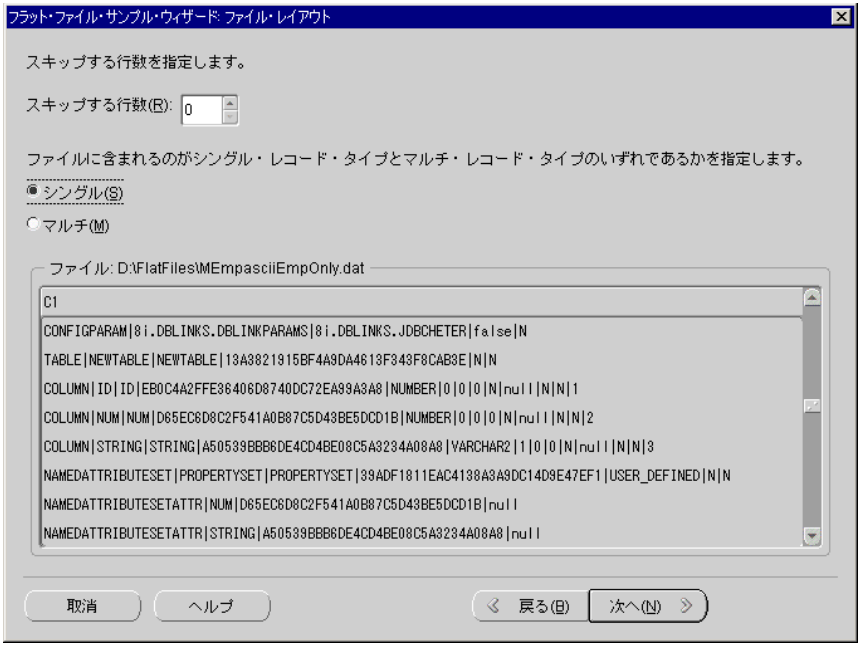
この方法を使用すると、3 つ以上のレコードを結合できます。次の例では、レコード先頭の継続文字を使用した、論理レコード当たり 4 つの物理レコードを示しています。

```
PHYSICAL_RECORD1
%PHYSICAL_RECORD2
%PHYSICAL_RECORD25
%PHYSICAL_RECORD26   (end log record 1)
PHYSICAL_RECORD3
%PHYSICAL_RECORD4
%PHYSICAL_RECORD45
%PHYSICAL_RECORD46 (end log record 2)
```

ファイル・レイアウトの選択

スキップする行数を指定し、シングル・レコード・タイプかマルチ・レコード・タイプかを選択するには、図 4-22 に示す「フラット・ファイル・サンプル・ウィザード: ファイル・レイアウト」ページを使用します。

図 4-22「フラット・ファイル・サンプル・ウィザード : ファイル・レイアウト」ページ



「スキップする行数」に、スキップするレコードの数を入力します。この操作は不要なヘッダー情報のスキップに役立ちます。レコードの 1 つにフィールド名が含まれている場合は、その前にあるヘッダー・レコードをスキップして、フィールド名を含むそのレコードがファイルの先頭になるようにします。シングル・レコードのファイル・タイプを定義している場合は、後の「フラット・ファイル・サンプル・ウィザード : フィールド・プロパティ」ページで、そのレコードをフィールド名として使用するように指示できます。

ファイルに含まれるのがシングル・レコード・タイプとマルチ・レコード・タイプのいずれであるかを指定します。

- ファイルに 1 つのレコード・タイプしか含まれない場合は、「シングル」を選択します。
- ファイルに複数のレコード・タイプが含まれる場合は、「マルチ」を選択します。後からこのウィザードで、ファイルをスキャンしてレコード・タイプを検索するように指示できます。マルチ・レコード・タイプの詳細は、4-38 ページの「レコード・タイプの選択（マルチ・レコード・タイプのファイルのみ）」を参照してください。

ファイル・フォーマットの指定

ファイル・フォーマットに「固定長」または「デリミタ」を選択するには、図 4-23 に示す「フラット・ファイル・サンプル・ウィザード : ファイル・フォーマット」ページを使用します。

図 4-23「フラット・ファイル・サンプル・ウィザード : ファイル・フォーマット」ページ

ファイル・フォーマットを選択すると、ウィザードのページの下部に表示されるサンプルが更新されます。サンプル・データ内を移動するには、スクロール・バーを使用します。

デリミタ付きファイルの場合は、次のプロパティを指定します。

- **フィールド・デリミタ:** フィールド・デリミタは、1 つのフィールドの終了位置と別のフィールドの開始位置を示します。フィールド・デリミタは、入力するか、ドロップダウン・リストから選択できます。ドロップダウン・リストには、一般的なフィールド・デリミタが表示されます。ただし、囲み文字以外であれば、どのような文字でもデリミタとして入力できます。デフォルトはカンマ（,）です。
- **囲み（左および右）:** 一部のデリミタ付きファイルには、フィールド内のテキスト文字列を表す囲み文字が含まれます。ファイルに囲み文字が含まれる場合は、囲み文字をテキスト・ボックスに入力することも、ドロップダウン・リストから選択することもできます。ドロップダウン・リストには、一般的な囲み文字が表示されます。ただし、入力する場合はどのような文字でも入力できます。囲みのデフォルトは、左右いずれも二重引用符（"）です。

レコード・タイプの選択（マルチ・レコード・タイプのファイルのみ）

フラット・ファイルのスキャンによるレコード・タイプの検索、レコード・タイプの追加と削除、およびレコード・タイプへのタイプ値の割当てには、「フラット・ファイル・サンプル・ウィザード: レコード・タイプ」ページを使用します。

注意: この手順は、シングル・レコード・タイプのファイルでは不要です。データ・ファイルがシングル・レコード・タイプで、ファイル・フォーマットが固定長の場合は、4-42 ページの「フィールド長の指定（固定長ファイルのみ）」に進んでください。データ・ファイルがシングル・レコード・タイプで、ファイル・フォーマットがデリミタ付きの場合は、4-45 ページの「フィールド・プロパティの指定」に進んでください。

例: レコード・タイプが複数あるフラット・ファイル

レコード・タイプが複数あるファイルでは、フィールドの 1 つによって、レコード・タイプが区別されます。図 4-24 は、2 つのレコード・タイプ、「E」と「P」を持つカンマ区切りのファイルの例を示しています。フラット・ファイル・サンプル・ウィザードを使用する場合は、各レコードの指定したフィールドをスキャンして、レコード・タイプの値を検索するように指示します。この場合は、最初のフィールドをスキャンするように指示します。タイプ値として、「E」と「P」が返されます。

図 4-24 レコード・タイプが複数あるファイルの例

```
"E],003715,4,153,09061987,0140000.00,"IRENE HIRSH    ],1,085.00,2,066.00,3,088.00,4,125.00
"P],003715,01152000,01162000,00101,0005000.00,0007000.00,150.00,200.00,133.00,075.00,055.00,066.00,077.00
"P],003715,02152000,02162000,00102,0003000.00,0008000.00,120.00,180.00,120.00,065.00,044.00,075.00,055.00
"P],003715,03152000,03162000,00103,0005000.00,0009000.00,130.00,170.00,110.00,055.00,033.00,065.00,066.00
"P],003715,04152000,04162000,00104,0003000.00,0010000.00,140.00,160.00,100.00,045.00,056.00,075.00,065.00
"E],003941,2,165,03111959,0167000.00,"ANNE FAHEY     ],1,099.00,2,066.00,3,088.00,4,125.00
"P],003941,01152000,01162000,00105,0003000.00,0010000.00,140.00,160.00,100.00,045.00,056.00,075.00,065.00
"P],003941,02152000,02162000,00106,0003000.00,0010000.00,140.00,160.00,100.00,045.00,056.00,075.00,065.00
"P],003941,03152000,03162000,00107,0003000.00,0010000.00,140,00.160.00,100.00,045.00,056.00,075.00,065.00
"E],001939,2,265,09281988,0213000.00,"EMILY WELLMET  ],1,077.00,2,066.00,3,088.00,4,125.00
"P],001939,01152000,01162000,00108,0003000.00,0010000.00,140.00,160.00,100.00,045.00,056.00,075.00,065.00
"P],001939,02152000,02162000,00109,0003000.00,0010000.00,140.00,160.00,100.00,045.00,056.00,075.00,065.00
```

このウィザードを使用して、レコード・タイプが複数あるファイルをサンプリングする場合は、「フラット・ファイル・サンプル・ウィザード: 名前」ページで指定したサンプルのサイズが、各タイプのレコードを 1 つ以上含むことのできる大きさになるようにしてください。

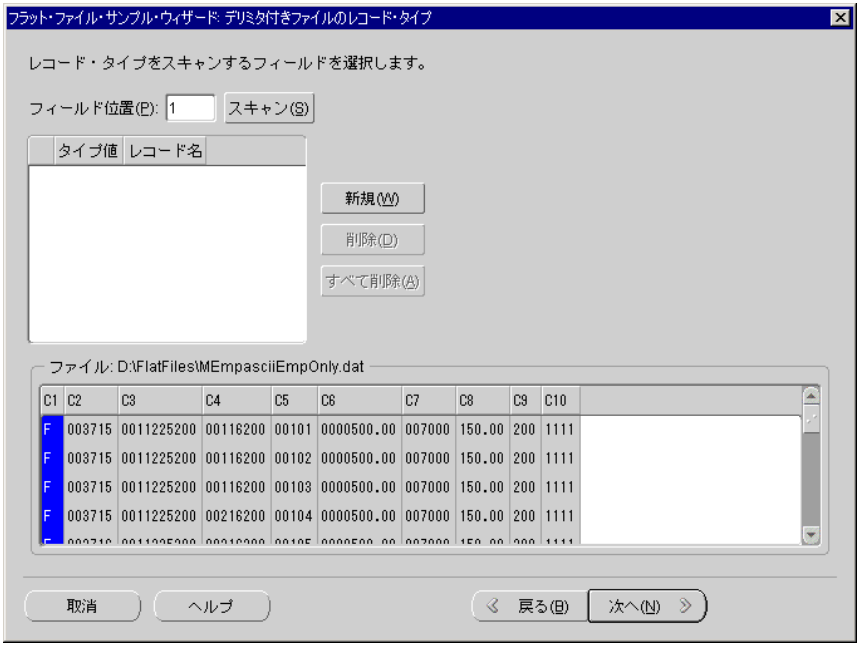
サンプリングは、一度開始すると取消しができないため、最適なパフォーマンスが得られる適度な行数を選択してください。適度な行数内にレコード・タイプがすべて表示できない場合は、マスター・ファイルの異なる部分から行を選択して、仮のサンプル・ファイルを作成し、データ・セットの代理とすることができます。データの内容が不明の場合は、ファイル全体をサンプリングすることもできます。データの内容がよくわかっている場合は、代理のサンプルをスキャンしてから、手動で新しいレコード・タイプを追加できます。

デリミタ付きファイルに対する複数のレコード編成の定義

1 つのデリミタ付きフラット・ファイルに複数の異なるレコード・タイプがある場合は、フラット・ファイル・サンプル・ウィザードのスキャン機能を使用してレコード・タイプを検索し、ラベルを付けることができます。

図 4-25 では、複数のレコード・タイプを検索するためにスキャンする位置として、第 1 フィールドが選択されています。

図 4-25「フラット・ファイル・サンプル・ウィザード : デリミタ付きファイルのレコード・タイプ」ページ



「フラット・ファイル・サンプル・ウィザード : デリミタ付きファイルのレコード・タイプ」ページの作業を完了する手順は次のとおりです。

1. ファイル内でレコード・タイプを特定するフィールドを 1 つ選択します。

   ページ下部のパネルに、サンプル内のフィールドがすべて表示されます。サンプル・ボックスからフィールドを選択します。または、「フィールド位置」に、サンプルに表示されているとおりの位置を入力します。特に指定を行わない場合、デフォルトは第 1 フィールドになります。

   ファイルがスキャンされてフィールドが検索され、タイプ値が表示されます。各タイプ値に、デフォルトのレコード名（RECORD1、RECORD2...）が割り当てられます。

2. レコード名を編集できます。

   レコード名をクリックして名前を変更するか、ドロップダウン・リストから別のレコード名を選択します。1 つのレコード名に、複数のレコード・タイプの値を関連付けることができます。タイプ値は、「新規」ボタンで追加、または「削除」ボタンで削除することもできます。

3. 「次へ」をクリックしてウィザードを続行します。

固定長ファイルに対する複数のレコード編成の定義

1 つの固定長フラット・ファイルに複数の異なるレコード・タイプがある場合は、フラット・ファイル・サンプル・ウィザードのスキャン機能を使用してレコード・タイプを検索し、各レコード・タイプにタイプ値を割り当てることができます。

図 4-26 は、第 1 フィールド位置を基に、レコード・タイプを検索するためにスキャンした結果を示しています。

図 4-26「フラット・ファイル・サンプル・ウィザード : 固定長ファイルのレコード・タイプ」ページ

「フラット・ファイル・サンプル・ウィザード : 固定長ファイルのレコード・タイプ」ページの作業を完了する手順は次のとおりです。

1. ファイル内でレコード・タイプを特定するフィールドを 1 つ指定します。「開始位置」と「終了位置」に値を入力します。第 1 フィールドを基にスキャンしてレコードを検索する場合は、「開始位置」に 0 を入力します。

   ページ下部のパネルにあるファイル・サンプルのルーラーに赤いチェックマークが付き、選択されたフィールドが示されます。

2. 「スキャン」をクリックします。

   ファイルのフィールドがスキャンされ、タイプ値が表示されます。各タイプ値に、デフォルトのレコード名（RECORD1、RECORD2...）が割り当てられます。

3. レコード名を編集できます。

   レコード名をクリックして名前を変更するか、ドロップダウン・リストから別のレコード名を選択します。1 つのレコード名に、複数のレコード・タイプの値を関連付けることができます。タイプ値は、「新規」ボタンで追加、または「削除」ボタンで削除することもできます。

4. 「次へ」をクリックしてウィザードを続行します。

フィールド長の指定（固定長ファイルのみ）

フラット・ファイル・サンプル・ウィザードを使用して固定長のフラット・ファイルを定義する場合は、ファイルの各フィールドの長さも定義する必要があります。

注意 : この手順は、デリミタ付きファイルでは不要です。4-45 ページの「フィールド・プロパティの指定」に進んでください。

フィールド長は、フィールド長を入力するか、ルーラーを使用して定義できます。

図 4-27 は、「フラット・ファイル・サンプル・ウィザード : フィールド長」ページを示しています。

図 4-27「フラット・ファイル・サンプル・ウィザード : フィールド長」ページ

各フィールドの長さがわかっている場合は、「フィールド長」にフィールド長を入力します。それぞれの長さはカンマで区切ります。ウィザード・ページの下部に、サンプルの変更内容が表示されます。

ルーラーを使用するには、ルーラー上の任意の数字かハッシュ・マークをクリックします。ルーラーの上に赤いチェックマークが表示され、境界を示す赤い線が表示されます。指定を間違えた場合は、マーカーをダブルクリックして削除するか、別の位置にマーカーを移動します。ルーラーを使用して、ファイル内の各フィールドにマーカーを作成します。

ルーラーを使用してフィールド長を指定する場合は、このチェックマーカーによって、各フィールドの開始と終了の境界が示されます。Warehouse Builder では、この情報を基に、各フィールドの占める位置が決定されます。たとえば、3 文字からなるフィールドが位置 6、7 および 8 を占めている場合、このフィールドの開始と終了の値は、内部的には「5,8」として特定されます。

OMB Plus でファイルのフィールド位置を定義して、後から同じファイルのプロパティを、Warehouse Builder のユーザー・インタフェースである「フラット・ファイルのプロパティ」画面で表示すると、フィールド定義の表示内容は異なります。OMB Plus で開始値および終了値を「6,8」と定義した同じ 3 文字のレコードが、「フラット・ファイルのプロパティ」画面では「5,8」という位置で表示されます。

マルチ・レコード・ファイルのファイル長の指定

レコード・タイプは、「レコード名」の名前で選択できます。また、ウィザード・ページの右下隅にある「次のレコード・タイプ」でも、レコード・タイプを選択できます。ウィザード・ページの左下隅には、フィールド長が指定されていないレコードの数が表示されます。

フラット・ファイルにレコード・タイプが複数存在する場合は、操作を続行する前に、レコード・タイプごとにフィールド長を指定するように求められます。

図 4-28 は、レコード・タイプが複数ある固定長ファイルの「フラット・ファイル・サンプル・ウィザード: フィールド長」ページを示しています。

図 4-28「フラット・ファイル・サンプル・ウィザード: マルチ・レコード・ファイルのフィールド長」ページ

フィールド・プロパティの指定

各フィールドのプロパティを定義するには、「フラット・ファイル・サンプル・ウィザード: フィールド・プロパティ」ページを使用します。

所定のデータ型に適用されないプロパティは非アクティブになります。たとえば、CHAR の長さは編集できますが、精度とスケールは使用できません。アクティブになっていないプロパティはグレー表示されます。

図 4-29 は、「フラット・ファイル・サンプル・ウィザード: フィールド・プロパティ」ページを示しています。

図 4-29「フラット・ファイル・サンプル・ウィザード : フィールド・プロパティ」ページ

フィールドごとに、次の 2 セットのプロパティが表示されます。

- **SQL*Loader のフィールド・プロパティ :** SQL*Loader プロパティとは、SQL*Loader ユーティリティを使用するフラット・ファイルでサポートされるデータ型です。Warehouse Builder では、ソースとしてフラット・ファイルを使用するあらゆるマッピングのことです。ウィザードでは、このプロパティ・セットが最初に表示されます。SQL*Loader プロパティには、「名前」、「タイプ」、「マスク」、「NULLIF」、「DEFAULTIF」、「開始」、「長さ」があります。「精度」と「スケール」は、将来のために予約されているフィールドです。
- **SQL プロパティ :** SQL プロパティとは、所定のフラット・ファイル・フィールドに関連付けられたリレーショナル・データ型のデフォルトです。デフォルトのデータ型の値を 1 つ選択すると、フラット・ファイルまたは SQL*Loader のデータ型に変更があった場合、自動的に更新されて変更内容が反映されます。また、これらのデフォルトは、任意の値で上書きできます。上書き後の値は、フラット・ファイルのデータ型とは関係なくなります。バインドされていない表にフラット・ファイルをマップしたり、外部表を作成すると、これらのプロパティがデフォルトの列属性になります。ウィザードでは、このプロパティ・セットが 2 番目に表示されます。SQL プロパティには、「SQL タイプ」、「SQL 長」、「SQL 精度」、「SQL スケール」があります。

注意 : 「フラット・ファイル・サンプル・ウィザード : フィールド・プロパティ」ページの作業が完了したら、「フラット・ファイル・サンプル・ウィザード : サマリー」ページで選択内容を確認し、「終了」を選択します。フラット・ファイル・サンプル・ウィザードからメタデータ・インポート・ウィザードに戻ります。サンプリングするファイルをさらに選択するか、「終了」を選択してインポートを開始します。ファイルのサンプリングの続行方法の詳細については、4-29 ページの「メタデータ・インポート・ウィザードのフラット・ファイル関連機能の使用方法」の手順 8 を参照してください。

SQL*Loader プロパティ

デフォルトでは、これらのプロパティは、ソース・ファイルに表示されるとおりに表示されます。SQL*Loader でフィールドを処理する方法を指定するには、これらのプロパティを編集するか、デフォルトをそのまま使用します。

次の SQL* Loader プロパティを編集できます。

名前 各フィールドには、ウィザードによって名前が割り当てられます。最初のフィールドには「C1」、次のフィールドには「C2」というように割り当てられます。フィールドの名前を変更するには、そのフィールドをクリックして新しい名前を入力します。

ファイル・タイプがシングル・レコードの場合は、ファイル内の最初のレコードを使用してフィールドに名前を付けるように指示できます。この指示を行うには、「フィールド名として最初のレコードを使用」チェック・ボックスを選択します。このオプションを選択すると、すべてのフィールドのデータ型属性が、CHAR にデフォルト設定されます。

タイプ SQL*Loader でのフィールドのデータ型を記述します。フラット・ファイル・サンプル・ウィザードを使用してインポートできるデータ型は、CHAR、DATE、DECIMAL EXTERNAL、FLOAT EXTERNAL、INTEGER EXTERNAL、ZONED、ZONED EXTERNAL です。SQL*Loader のフィールドおよびデータ型の詳細は、『Oracle Database ユーティリティ』を参照してください。現時点でサポートされているのは、移植可能なデータ型のみです。

マスク SQL*Loader のデフォルトの日付マスクは dd-mon-yy です。このデフォルトを無効にするには、ファイルを定義するときに有効な日付マスクを入力します。たとえば、入力データで SQL*Loader のデフォルトではなく DD-Mon-YYYY という書式が使用されている場合は、実際の書式をマスクとして入力します。

NULLIF SQL*Loader のデフォルトのアクションを無効にするには、フィールドに NULLIF 条件を指定します。たとえば、文字フィールドの内容が空白の場合に、空白文字を保存するのではなく、その列に NULL のマークを付けるよう指定することができます。このフィールドで有効な構文として、=BLANKS、='quoted string'、=X'ff' (hex value)、!= (not equal) を使用できます。

DEFAULTIF SQL*Loader のデフォルトのアクションを無効にするには、フィールドに DEFAULTIF 条件を指定します。たとえば、数値または DATE 型のフィールドが空白の場合は、レコード全体が拒否されます。このアクションを無効にするには、「DEFAULTIF」プロパティに「BLANKS」と入力します。SQL*Loader は、この条件を評価するとき、該当する数値フィールドにゼロを設定してレコードをロードします。このフィールドで有効な構文として、=BLANKS、='quoted string'、=X'ff' (hex value)、!= (not equal) を使用できます。

開始 固定長ファイルで、フィールドの開始位置を示します。

長さ SQL* Loader で使用するフィールドの長さを指定します。デリミタ付きファイルでは、フィールド長に値は入りませんが、フィールドの最大長がわかっている場合には手動で編集してもかまいません。

精度 データ型 DECIMAL および ZONED の精度を定義します。これは、将来のために予約されているフィールドです。

SQL プロパティ

フラット・ファイル・サンプル・ウィザードは、対応する SQL*Loader プロパティに基づいて、これらのプロパティにデフォルト値を割り当てます。Warehouse Builder では、次のいずれかの方法でこの SQL プロパティを使用できます。

- **外部表 :** フラット・ファイル・サンプル・ウィザードを使用してメタデータをインポートした後は、1 つのフラット・ファイル・レコード・タイプに基づいて外部表を作成できます。マッピングに外部表を使用するとき、列のプロパティは、フラット・ファイルに定義した SQL プロパティに基づきます。外部表の詳細は、3-27 ページの「外部表の使用方法」を参照してください。

- **空のマッピング表へのデータの移入 :** フラット・ファイル・サンプル・ウィザードを使用してメタデータをインポートした後は、空のリレーショナル表にメタデータを移入できます。この表は、フラット・ファイルのソースに定義した SQL プロパティを継承します。
- **フラット・ファイル・ターゲット :** フラット・ファイル・サンプル・ウィザードを使用してメタデータをインポートした後は、マッピングのターゲットとしてフラット・ファイルを使用できます。フラット・ファイル・ターゲットは SQL プロパティを継承しません。すべてのフィールドは文字データ型にデフォルト設定されます。このオプションは、ここでは使用されていません。フラット・ファイル演算子をターゲットとして使用する方法の詳細は、8-28 ページの「フラット・ファイルのマッピング演算子」を参照してください。

SQL* Loader のデータ型を持つフィールドを、SQL データ型の列にマップする方法を指定するには、これらの SQL プロパティを編集するか、デフォルトをそのまま使用します。デフォルト値は SQL* Loader の値に調整されますが、ユーザーが独自に変更した値は、SQL* Loader のデータ型の値とは無関係な定数として残ります。これらの SQL プロパティは、マッピング、検証および生成に使用されます。

SQL タイプ SQL データ型を記述します。Warehouse Builder がサポートしている移植可能な SQL データ型は次のとおりです。

- CHAR
- DATE
- FLOAT
- NUMBER
- VARCHAR
- VARCHAR2

SQL 長 SQL で使用するフィールド長を定義します。

SQL 精度 データ型 NUMBER および FLOAT に SQL で使用するフィールドの精度を定義します。

SQL スケール データ型 NUMBER および FLOAT に SQL で使用するフィールドのスケールを定義します。

ファイル定義の更新

ファイル・フォーマットの定義を更新するには、プロパティ・シートを編集します。

ファイル定義を更新する手順は次のとおりです。

1. ナビゲーション・ツリーで、ファイル定義を選択します。
2. ファイル名を右クリックして、「プロパティ」を選択します。

   Warehouse Builder では、フラット・ファイルのプロパティ・シートに、次のタブが表示されます。

   一般 : このタブを使用して、定義の名前と説明を編集します。また、グローバル・プロパティ（物理レコード・サイズ、1 論理レコード当たりの物理レコード数、デリミタ文字および囲み文字など）を変更することもできます。

   レコード : このタブは、レコード・タイプが複数あるフラット・ファイルにのみ使用できます。このタブを使用して、フィールドの再定義やレコード・タイプの追加、削除および編集を行います。

   構造 : このタブを使用して、フィールドのレベル属性や SQL*Loader および SQL のプロパティを編集します。

「一般」タブ

このタブを使用して、定義の名前と説明を編集します。また、グローバル・プロパティ（物理レコード・サイズ、1 論理レコード当たりの物理レコード数、デリミタ文字および囲み文字など）を変更することもできます。

デリミタ付きレコードの場合、「一般」タブには次のフィールドがあります。

- **レコード・デリミタ :** このオプションは、各レコードの終了がデリミタで指定される場合に選択してください。次に、レコード・デリミタを指定します。デフォルトのレコード・デリミタである改行（<CR>）をそのまま使用するか、新しい値を入力できます。
- **レコード長 (文字数):** このオプションは、ファイル内の各レコードの長さが同一の場合に選択してください。次に、各レコードの文字数を指定します。マルチバイト・キャラクタのファイルでは、マルチバイト・キャラクタを 1 文字として計算します。

ファイルに論理レコードが含まれている場合は、「論理レコードを含むファイル」をクリックします。次に、ファイルを定義するためのオプションを、次の中から 1 つ選択します。

- **1 論理レコード当たりの物理レコード数 :** データ・ファイルには、論理レコードごとに固定された数の物理レコードが含まれます。
- **現行の物理レコードの終了文字 :** データ・ファイルには、数が可変の物理レコードが含まれています。さらに、各物理レコードの末尾には、そのレコードが次の物理レコードと関連することを表す継続文字が付加されています。

- **次の物理レコードの開始文字:** データ・ファイルには、数が可変の物理レコードが含まれています。さらに、各物理レコードの先頭には、そのレコードが前の物理レコードと関連することを表す継続文字が付加されています。

「レコード」タブ

レコード・タイプが複数あるファイルを選択した場合は、「レコード名」フィールドから各レコード・タイプを選択できます。Warehouse Builder ではレコード・シートが表示されるので、レコード・タイプ情報を編集できます。

レコード・タイプは次の列番号にあります: このフィールドには、レコード・タイプのインジケータを含む列が表示されます。この値は変更可能です。たとえば、2 つのレコード・タイプを持つフラット・ファイルで、次の一覧に示すように、3 番目の列の最初の文字によって各レコード・タイプが区別される場合は、このフィールドの値は 3 になります。

- レコード・タイプ 1: `2002 0115 E 4564564`
- レコード・タイプ 2: `2003 1231 D 659871 Q HKLIH`

レコード・タイプの値: この表には各レコード・タイプ、レコード・タイプを他のレコード・タイプと区別する値、およびレコード・タイプに付けた名前が表示されます。表 4-2 は、前述の 2 つのサンプル・レコードについて、レコード・タイプの値の例を示しています。

表 4-2 レコード・タイプの値の例

タイプ値	レコード名
E	Employee
D	Department

- レコード・タイプを追加するには、「新規」をクリックして、タイプ値、およびレコード・タイプを説明するレコード名を入力します。
- レコード・タイプを削除するには、削除する各レコード・タイプの左側にあるチェック・ボックスを選択して、「削除」をクリックします。

ターゲット・システムのソースの特定と定義が終わったら、ターゲット・スキーマをモデル化する準備は完了です。

「構造」タブ

「構造」タブを使用して、フィールド名、データ型、マスクおよび SQL プロパティを編集します。フィールドを追加または削除できます。また、フィールド・マスク、NULLIF 条件または DEFAULTIF 条件も追加できます。

レコード・タイプが複数あるファイルを選択した場合は、「レコード名」フィールドから各レコード・タイプを選択できます。Warehouse Builder ではレコード・シートが表示されるので、レコード・タイプ情報を編集できます。

所定のデータ型に適用されないプロパティは非アクティブになります。たとえば、CHAR の長さは編集できますが、精度とスケールは使用できません。

フィールドごとに、次の 2 セットのプロパティが表示されます。

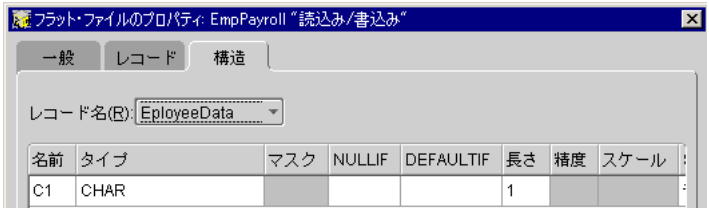
- **SQL* Loader プロパティ :** ウィザードでは、このプロパティ・セットが最初に表示されます。SQL*Loader プロパティには、「名前」、「タイプ」、「マスク」、「NULLIF」、「DEFAULTIF」、「開始」、「長さ」があります。「精度」と「スケール」は、将来のために予約されているフィールドです。
- **SQL プロパティ :** ウィザードでは、このプロパティ・セットが 2 番目に表示されます。SQL プロパティには、「SQL タイプ」、「SQL 長」、「SQL 精度」、「SQL スケール」があります。

図 4-30 は、シングル・レコード・タイプのデータ・ファイルの「構造」タブを示しています。

図 4-30 シングル・レコード・タイプのデータ・ファイルの「構造」タブ

図 4-31 は、マルチ・レコード・タイプのデータ・ファイルの「構造」タブを示しています。

図 4-31 マルチ・レコード・タイプのデータ・ファイルの「構造」タブ



5

データ・オブジェクトの構成

設計フェーズでは、Warehouse Builder の設計オブジェクトを使用してターゲット・システムの論理モデルを定義しました。この章では、このような設計オブジェクトに物理プロパティを割り当てる方法を説明します。

この章ではまず、Warehouse Builder で定義された表、マテリアライズド・ビュー、ディメンション、キューブなどの設計オブジェクトに対して、索引およびパーティションを作成する方法を説明します。次に、Warehouse Builder のオブジェクト定義に物理プロパティを割り当てる方法を説明します。

この章では、次のトピックについて説明します。

索引とパーティションの作成

Warehouse Builder では、表、マテリアライズド・ビュー、ディメンションおよびキューブに、索引とパーティションを作成できます。索引とパーティションは、データ・ウェアハウスの問合せパフォーマンスを高めるために作成します。

この項では、次のトピックについて説明します。

索引について

索引は、ウェアハウス内で処理されるデータに迅速にアクセスして、問合せを高速化するうえで重要です。表の 1 つ以上の列に索引を作成すると、その表に対する SQL 文の実行を高速化できます。索引には、次のような特徴があります。

- 索引列の値は、事前ソートされて保存されます。
- 索引はデータベースの別の領域に保存されるため、基礎となる表に影響を与えず、いつでも作成または削除ができます。
- データが削除、追加または更新されると、索引は自動的にメンテナンスされます。索引は表のデータから独立しています。

B* ツリー索引およびビットマップ索引は、データ・ウェアハウスでは特に便利です。索引と索引作成計画の詳細は、『Oracle データ・ウェアハウス・ガイド』を参照してください。

Warehouse Builder では、表、マテリアライズド・ビュー、ディメンション、キューブなどのデータ・オブジェクトを定義した後で、ターゲット・システムの要件に従って、データ・オブジェクトに索引を作成できます。索引を作成した後は、物理プロパティでこれらの索引を構成して、配布可能にする必要があります。次の項では、Warehouse Builder で索引を作成および構成する方法を説明します。

索引の作成

キューブ、ディメンション、表およびマテリアライズド・ビューに、索引を作成および構成できます。

索引を作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで該当する設計オブジェクトを右クリックし、「構成」を選択します。

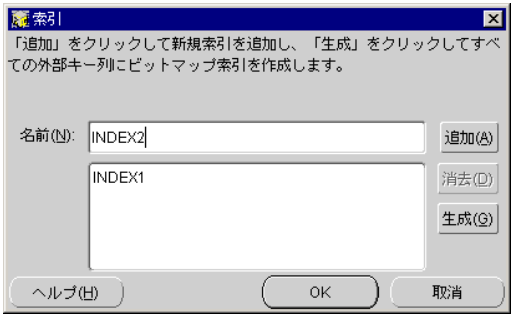
   「構成プロパティ」ダイアログが表示されます。

2. 「構成プロパティ」ダイアログから、「索引」の右側にあるフィールドを選択します。

3. 「...」ボタンをクリックします。

   図 5-1 に示す「索引」ダイアログが表示されます。

図 5-1「索引」ダイアログ



4. 「名前」フィールドに索引の名前を入力して「追加」をクリックします。

   作成する各索引に対して、この手順を繰り返します。

5. 索引が正しければ「OK」をクリックします。正しくない場合は「取消」をクリックします。

   これで、索引を構成して配布する準備が整いました。

ビットマップ索引の作成

Warehouse Builder では、Oracle データベースのビットマップ索引作成機能を利用して、表内で特定のキー値を含む行へのポインタを提供します。データ・ウェアハウスでは、ビットマップはスター・クエリー変換を可能にするために作成されます。スター型変換は、コストベースの問合せ変換で、スター・クエリーの効率的な実行を目的としています。スター型変換を実行するには、前提条件として、キューブ（複数可）の外部キー列それぞれに、ビットマップ索引が作成されている必要があります。詳細は、『Oracle データ・ウェアハウス・ガイド』を参照してください。

ビットマップ索引を作成する手順は次のとおりです。

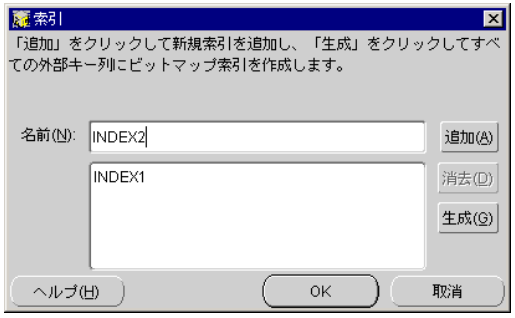
1. Warehouse Builder ナビゲーション・ツリーで該当する設計オブジェクトを右クリックし、「構成」を選択します。

   「構成プロパティ」ダイアログが表示されます。

2. 「構成プロパティ」ダイアログから、「索引」の右側にあるフィールドを選択します。
3. 「...」ボタンをクリックします。

   図 5-2 に示す「索引」ダイアログが表示されます。

図 5-2「索引」ダイアログ



4. 「名前」フィールドに索引の名前を入力して「追加」をクリックします。

   作成する各索引に対して、この手順を繰り返します。

5. ビットマップ索引を作成する場合は、「生成」をクリックします。

   すべての外部キー列にビットマップ索引が作成されます。ビットマップ索引の生成前に、Warehouse Builder は表の外部キーをチェックします。次のような場合、Warehouse Builder はビットマップ索引を生成しません。

   外部キーが存在しない : 表に外部キーが存在しません。この場合、Warehouse Builder によって、表に外部キーが存在しないために索引が作成できないことが示されます。

   列が存在しない : 表に外部キーが存在していますが、それらに対して列が定義されていません。

   索引が存在する : 表に外部キーが存在しており、すでに索引を持つものが存在します。ビットマップ索引がいずれかの列にすでに作成されている場合、「影響レポート」のダイアログにはその列の名前と既存の索引の名前が表示され、これらの列には新規でビットマップ索引を作成できないというメッセージも表示されます。

   「ビットマップ索引作成の影響レポート」ダイアログが表示されます。

6. 索引が正しければ「OK」をクリックします。正しくない場合は「取消」をクリックします。

次の項の手順に従って、索引を配布できるように構成します。

索引の構成

索引を作成した後は、「構成プロパティ」ダイアログで索引にパラメータを定義する必要があります。

索引を構成する手順は次のとおりです。

1. 「構成プロパティ」ダイアログで「索引」ノードを開き、構成パラメータを表示します。
2. 索引に次のパラメータを構成します。

   ロギング・モード : 索引の作成ログを REDO ログ・ファイルに作成するかどうかを指定します。

   パラレル : このパラメータが PARALLEL に設定されている場合は、表の作成時にパラレル処理が有効になります。

   表領域 : 索引を作成する表領域を指定します。

   一意でない B* ツリー索引の場合は、「索引タイプ」パラメータを BITMAP,UNIQUE に設定するか、空白にしてください。ビットマップ索引を構成する場合は、ビットマップ・オプションが自動的に選択されます。

   ローカル索引 : ローカル索引は、基礎となる表の構造を反映します。このパラメータを true に設定すると、索引が、基礎となる表と同じ列に、同じパーティション数で、同じパーティション・バウンドでパーティション化されるように指定されます。

配布可能 : この索引を配布する場合は、`true` を設定します。

索引列 :「索引列」ダイアログを表示するには、「...」ボタンをクリックします。「選択肢」フィールドで、索引に含める列を選択します。右矢印をクリックして、選択した列を「選択済み」リストに移動します。「OK」をクリックします。

3. 「構成プロパティ」ダイアログを閉じます。

索引の名前の変更

索引の名前を変更する手順は次のとおりです。

1. 「構成プロパティ」ダイアログから「索引」を開きます。
2. 名前を変更する索引の右側にある列をクリックして、「...」ボタンをクリックします。

   「索引」ダイアログが表示されます。
3. 「改名」フィールドに索引の新しい名前を入力して、「OK」をクリックします。
4. 「構成プロパティ」ダイアログの「索引」の下に、新しい索引名が表示されます。

パーティションについて

パーティションを使用すると、非常に大きな表や索引を、管理しやすい小さな部分に分割して効率的に管理できます。パーティションを作成すると、問合せやロードのパフォーマンスが向上し、物理記憶域の管理が容易になります。パーティションは個別に管理でき、他のパーティションから独立して操作できるため、パフォーマンス・チューニングのしやすい構造になります。DML 文を使用すると、表や索引の全体ではなく、個々のパーティションにアクセスして操作できます。

Warehouse Builder では、次の 2 つのタイプのパーティションを作成できます。

- **レンジ :** レンジ・パーティション化は、パーティションごとに設定するパーティション・キーの値の範囲に基づいています。各パーティションに格納されるのは、特定の値セットを持つデータのみです。これは最も一般的なタイプのパーティション化であり、多くの場合、日付とともに使用されます。たとえば、販売データを月別のパーティションに分けることができます。
- **ハッシュ :** ハッシュ・パーティション化を実装するには、パーティション化用のキーとパーティション数を選択します。ハッシュ・パーティション化では、全パーティションに均等にデータが分割されます。固定数のパーティションに対する物理的なデータ配置を制御する場合は、ハッシュ・パーティション化を選択します。ハッシュ・パーティション化は、データが履歴ではなく、列や列のリストが明らかでない場合に役立ちます。

パーティションは、表、マテリアライズド・ビュー、キューブおよびディメンションに対して作成してから構成できます。パーティションを作成した後は、物理プロパティでこれらのパーティションを構成して、配布可能にする必要があります。次の項では、Warehouse Builder でパーティションを作成および構成する方法を説明します。

パーティションの作成

Warehouse Builder でパーティションを作成するには、次の手順を実行します。

1. 作成するパーティションのタイプを、「ハッシュ」と「レンジ」のどちらかに決定します。
2. パーティション・キーのエントリを作成し、作成するパーティションのタイプを指定します。パーティション・キーには、表をパーティション化する方法を決定する列のセットが含まれます。5-7 ページの「パーティション・キーのエントリの作成」を参照してください。
3. レンジ・パーティションまたはハッシュ・パーティションを作成します。5-8 ページの「ハッシュ・パーティションの作成」および 5-8 ページの「レンジ・パーティションの作成」を参照してください。
4. 「索引」の「ローカル索引」パラメータを構成し、索引をパーティション化するかどうかを指定します。5-9 ページの「索引のパーティション化」を参照してください。

次の項では、これらの手順を詳細に説明します。

パーティション・キーのエントリの作成

パーティション・キーのエントリを作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで該当する設計オブジェクトを右クリックし、「構成」を選択します。

   「構成プロパティ」ダイアログが表示されます。

2. 「構成プロパティ」ダイアログから、「パーティション・キー」の右側にあるフィールドを選択します。

   「パーティション・キー」ダイアログが表示されます。

3. 「選択肢」フィールドで、含める列を選択します。
4. 右矢印ボタンをクリックして、選択した列を「選択済み」フィールドに移動します。
5. 「OK」をクリックします。
6. 「構成プロパティ」ダイアログから「パーティション・キー」ノードを開き、パーティション・キーの名前を開きます。
7. 作成するパーティションのタイプを、「RANGE」と「HASH」から選択します。

   レンジ・パーティション情報は、常にハッシュ・パーティション情報を上書きします。

   ハッシュ・パーティションを作成する場合は、この項の次に説明する手順を実行します。レンジ・パーティションを作成する場合は、5-8 ページの「レンジ・パーティションの作成」を参照してください。

ハッシュ・パーティションの作成

ハッシュ・パーティションを作成する手順は次のとおりです。

1. 「構成プロパティ」ダイアログから「ハッシュ・パーティション・パラメータ」ノードを開きます。
2. 「ハッシュ・サブパーティション番号」フィールドに、ハッシュ・サブパーティションの数を指定します。
3. 「ハッシュ・パーティション表領域」フィールドに、パーティションの表領域の名前を入力します。

レンジ・パーティションの作成

レンジ・パーティションを作成する手順は次のとおりです。

1. 「構成プロパティ」ダイアログで「レンジ・パーティション」ノードを開きます。
2. 「...」ボタンをクリックして、レンジ・パーティションを作成します。

   「レンジ・パーティション」ダイアログが表示されます。
3. 「名前」フィールドにパーティション名を入力して「追加」をクリックします。

   リストからパーティション名を削除するには、パーティション名を選択して「削除」をクリックします。
4. 「OK」をクリックします。

   「構成プロパティ」ダイアログの「レンジ・パーティション」ノードの下に、レンジ・パーティションが表示されます。
5. レンジ・パーティション名を開いて、次のパラメータを構成します。

   上限値 : 現在のパーティションの非包含的な上限を入力します。このエントリは、partition_by_range 句の column_list に対応したリテラル値の配列されたリストです。value_list のすべてのリテラルに、キーワード MAXVALUE を代入できます。MAXVALUE は、NULL を含むどの値よりも大きい値としてソートされる最大値を指定します。

   Warehouse Builder は、パーティション名と上限値プロパティを使用して表をパーティション化する DDL スクリプトを生成します。上限値プロパティによって、パーティションの内容が定義されます。

   表領域 : パーティションに関連付けられた表領域の物理属性の名前を入力します。この値を指定しない場合、このパーティションの表領域には、表で指定された表領域のデフォルト値が使用されます。

   配布可能 : このパーティションを配布する場合は、true を選択します。Warehouse Builder では、配布可能とマークされたパーティションのみにスクリプトが生成されます。

レンジ・パーティションの改名

レンジ・パーティションの名前を変更する手順は次のとおりです。

1. 該当するデータ・オブジェクトの「構成プロパティ」ダイアログを開きます。
2. 「レンジ・パーティション」ノードを開きます。
3. 編集するパーティション名の横にあるフィールドをクリックして、「...」ボタンをクリックします。

   「レンジ・パーティション」ダイアログが表示されます。
4. 「改名」フィールドの値をハイライトし、新しい名前を入力します。
5. 「OK」をクリックします。

索引のパーティション化

ローカル索引は、基礎となる表の構造を反映するように作成します。ローカル索引は、基礎となる表と同じ列でパーティション化され、基礎となる表の対応するパーティションと同じ数のパーティションまたはサブパーティション、およびパーティション・バウンドが作成されます。

ローカル索引を作成する手順は次のとおりです。

1. 「構成プロパティ」ダイアログから「索引」ノードを開きます。
2. 「索引タイプ」ノードを開きます。
3. パーティション化された索引を作成する場合は、「ローカル索引」パラメータを「true」に設定します。

Warehouse Builder の設計オブジェクトの構成

このフェーズでは、表領域、パーティション、その他の識別パラメータなどのプロパティを構成することにより、Warehouse Builder で作成したオブジェクト定義に、配布の物理プロパティを割り当てます。また、ジョブ名やランタイム・ディレクトリなどのランタイム・パラメータも構成します。

このような物理プロパティは、「構成プロパティ」ウィンドウを使用して設定します。プロパティは、ターゲット・モジュールに、また表、ディメンション、ビュー、マッピングなど、個々の設計オブジェクトごとに設定できます。次の項では、論理設計モデルに物理プロパティを割り当てる方法を説明します。

ターゲット・モジュールの構成

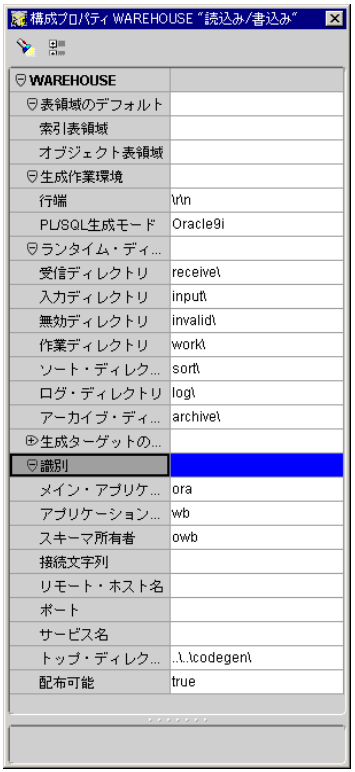
各ターゲット・モジュールでは、そのモジュールに含まれるすべてのオブジェクトに、トップレベルの構成オプションが用意されています。

ターゲット・モジュールを構成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで「データベース」を開き、「Oracle」を開きます。ターゲット・モジュール名を右クリックして「構成」を選択します。

   図 5-3 に示すように、「構成プロパティ」ダイアログが表示されます。

図 5-3 モジュールの「構成プロパティ」ウィンドウ



2. 構成するパラメータを選択し、パラメータ名の右側にある空欄をクリックして値を編集します。

   各パラメータでリストからオプションを選択するか、値を入力することができます。または、「...」をクリックして別のプロパティ・ダイアログを表示します。

3. 表領域のデフォルトについて構成します。

   索引表領域 : 索引を作成する各表領域の名前を定義します。デフォルトは NULL です。オブジェクト・レベルではなくターゲット・モジュール・レベルで索引表領域を構成する場合、Warehouse Builder では、コード生成時に、ターゲット・モジュール・レベルで構成される表領域の値が使用されます。索引ごとの表領域をオブジェクト・レベルで構成する場合、Warehouse Builder では、ターゲット・モジュール・レベルで構成された表領域の値が上書きされます。

   オブジェクト表領域 : 表、ビュー、マテリアライズド・ビューなどのオブジェクトを作成する各表領域の名前を定義します。デフォルトは NULL です。個々のオブジェクト・レベルではなくターゲット・モジュール・レベルでオブジェクト用の表領域を構成する場合、Warehouse Builder では、コード生成時に、ターゲット・モジュール・レベルで構成される値が使用されます。個々のオブジェクトごとに表領域を構成する場合、Warehouse Builder では、ターゲット・モジュール・レベルで構成された表領域の値が上書きされます。

4. 次の生成作業環境を構成します。

   行端 : フラット・ファイルの行端マーカーを定義します。これは、ウェアハウスの配布先のプラットフォームによって異なります。UNIX では `\n` を使用し、NT では `¥r¥n` を使用します。

   PL/SQL 生成モード : ターゲット・データベース・タイプを定義します。コード生成は、このフィールドの選択内容に基づいて実行されます。たとえば、「Oracle Database」を選択すると Oracle Database のコードが作成され、「Oracle8*i*」を選択すると Oracle8*i* のコードが生成されます。

   Oracle Warehouse Builder では、Oracle9*i* 以降のデータベースのみで使用できる新しい機能が導入されました。「PL/SQL 生成モード」で「Oracle8*i*」を選択すると、テーブル・ファンクションや外部表など、Oracle9*i* Warehouse Builder の機能の一部が使用できません。Oracle Warehouse Builder の機能のうち、Oracle9*i* より古いリリースと互換性のないものは、『Oracle Warehouse Builder リリース・ノート』を参照してください。

5. ランタイム・ディレクトリに次の作業環境を構成します。

   受信ディレクトリ : 現在は使用されていません。デフォルトは receive¥ です。

   入力ディレクトリ : 現在は使用されていません。デフォルトは input¥ です。

   無効ディレクトリ : LOADER エラーと拒否されたレコードのディレクトリです。デフォルトは invalid¥ です。

   作業ディレクトリ : 現在は使用されていません。デフォルトは work¥ です。

   ソート・ディレクトリ : 現在は使用されていません。デフォルトは sort¥ です。

   ログ・ディレクトリ : SQL*Loader のログ・ディレクトリ。デフォルトは log¥ です。

   アーカイブ・ディレクトリ : 現在は使用されていません。デフォルトは archive¥ です。

6. 次の生成ターゲットのディレクトリを構成します。

   DDL ディレクトリ : ターゲット・スキーマに、データベース・オブジェクトを作成するスクリプトのロケーションを入力します。デフォルトは `ddl¥` です。

   DDL 拡張子 : DDL スクリプトのファイル名拡張子を入力します。デフォルトは `.ddl` です。

   DDL スプール・ディレクトリ : スクリプトの生成処理中に DDL スクリプトがバッファされるロケーションを入力します。デフォルトは `ddl¥log` です。

   LIB ディレクトリ : Oracle のファンクションとプロシージャを生成するスクリプトのロケーションを入力します。デフォルトは `lib¥` です。

   LIB 拡張子 : マッピング名に追加される接尾辞を入力します。デフォルトは `.lib` です。

   LIB スプール・ディレクトリ : ユーザー定義のファンクションとプロシージャを生成するスクリプトのロケーションを入力します。デフォルトは `lib¥log¥` です。

   PL/SQL ディレクトリ : PL/SQL スクリプトの場所を入力します。デフォルトは `pls¥` です。

   PL/SQL 拡張子 : PL/SQL スクリプトのファイル名拡張子を入力します。デフォルトは `.pls` です。

PL/SQL 実行パラメータ・ファイル : PL/SQL ジョブ内のパラメータ・スクリプトの接尾辞を入力します。デフォルトは _run.ini です。

PL/SQL スプール・ディレクトリ : スクリプトの生成処理中に PL/SQL スクリプトがバッファされるロケーションを入力します。デフォルトは pls¥log¥ です。

ABAP ディレクトリ : SAP 表に関連するすべての ABAP の構成は、第 21 章「Warehouse Builder での SAP R/3 データの使用」を参照してください。

ABAP 拡張子 : ABAP スクリプトのファイル名拡張子です。デフォルトは .abap です。

ABAP 実行パラメータ・ファイル : ABAP ジョブ内のパラメータ・スクリプトの接尾辞です。デフォルトは _run.ini です。

ABAP スプール・ディレクトリ : スクリプトの生成処理中に ABAP スクリプトがバッファされる場所です。

LOADER ディレクトリ : 制御ファイルのロケーションを入力します。デフォルトは ctl¥ です。

LOADER 拡張子 : ローダー・スクリプトの接尾辞を入力します。デフォルトは .ctl です。

LOADER 実行パラメータ・ファイル : パラメータ初期化ファイルの接尾辞を入力します。デフォルトは _run.ini です。

7. 次の識別パラメータを構成します。

   メイン・アプリケーションの短縮名

   アプリケーションの短縮名

   スキーマ所有者

   接続文字列

   リモート・ホスト名

   ポート

   サービス名

   トップ・ディレクトリ

   配布可能

表の構成

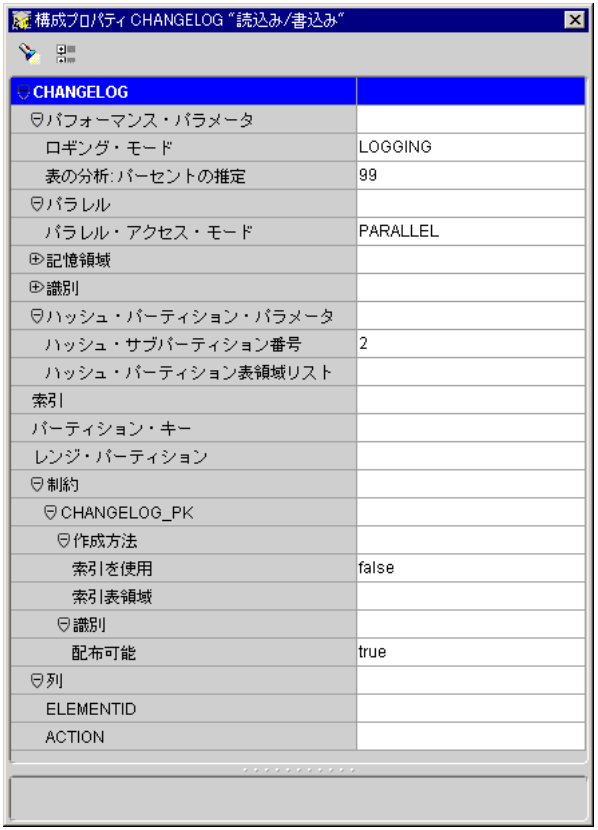
Warehouse Builder により、ターゲット・モジュールで定義された表ごとに DDL スクリプトが生成されます。表を構成するには、次の手順に従います。

表の物理プロパティを構成する手順は次のとおりです。

1. 表の名前を右クリックし、「構成」を選択します。

   図 5-4 に示すように、「構成プロパティ」ダイアログが表示されます。

図 5-4 表の構成プロパティ



パフォーマンス・パラメータ

- **ロギング・モード :** DML アクションを REDO ログ・ファイルに書き込むかどうかを指定します。パフォーマンスを向上するには、このパラメータを NOLOGGING に設定します。デフォルトは LOGGING です。
- **表の分析 : パーセントの推定 :** サンプル・サイズを行全体に対する割合で表した値です。ゼロ以外の値を設定した場合、表の分析を行う DDL スクリプトが Warehouse Builder によって生成されます。

パラレル

- **パラレル :** 表の作成時にパラレル処理を有効にします。デフォルトは PARALLEL です。

記憶領域

- **表領域 :** 表を作成する各表領域の名前を定義します。デフォルト値は NULL です。デフォルト値 NULL をそのまま使用する場合、Warehouse Builder では、ターゲット・モジュールの構成プロパティに設定されている表領域の値に基づいて表が生成されます。個々のオブジェクトに表領域を構成する場合、Warehouse Builder では、ターゲット・モジュールに構成された表領域の値が上書きされます。

識別

- **配布可能 :** この表制約を配布する場合は、true を選択します。Warehouse Builder では、配布可能とマークされた表制約のみにスクリプトが生成されます。

索引

5-16 ページの「外部表の構成」および 5-5 ページの「索引の構成」で説明されているように、索引を作成および構成します。

制約

- **索引を使用 :** 既存の索引を使用して制約を作成します。
- **配布可能 :** この表制約を配布する場合は、true を選択します。Warehouse Builder では、配布可能とマークされた表制約のみにスクリプトが生成されます。

外部表の構成

外部表に次のプロパティを構成します。

- アクセス指定
- 拒否
- データ特性
- パラレル
- フィールド編集
- 識別
- データ・ファイル

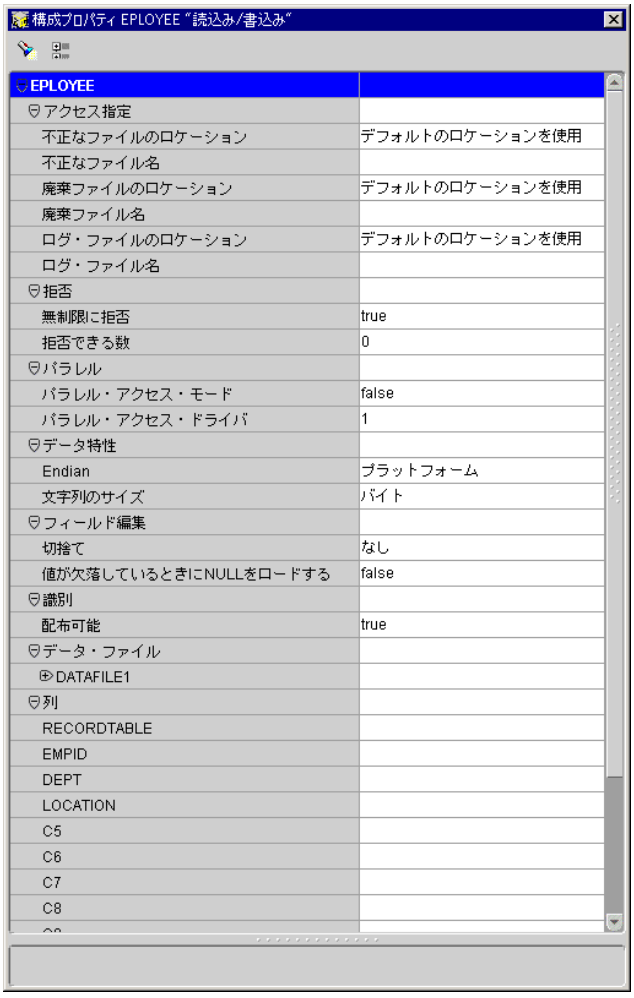
注意： 外部表をリポジトリにインポートしたり、外部表のアクセス・パラメータを手動で定義する場合、外部表の一部の構成プロパティは、「外部表プロパティ」ウィンドウの「アクセス・パラメータ」タブの設定によって無効になります。

外部表の物理プロパティを構成する手順は次のとおりです。

1. ナビゲーション・ツリーから外部表を選択します。
2. 「編集」メニューから「構成」を選択します。ツールバーから「構成」アイコンをクリックして選択することもできます。

   図 5-5 に示すように、「構成プロパティ」ウィンドウが表示されます。

図 5-5 外部表の構成プロパティ



3. プロパティを構成するには、空白部分をクリックしてドロップダウン・ボックスからプロパティを選択します。

アクセス指定

外部表をリポジトリにインポートしたり、ソース・ファイルを指定せずに外部表を作成した場合は、これらのプロパティを構成しないでください。アクセス指定のプロパティは、「外部表プロパティ」ウィンドウの「アクセス・パラメータ」タブの設定によって無効になります。

「アクセス指定」では、Warehouse Builder が SQL*Loader 経由で外部表をロードする際に使用する次のファイルの名前とロケーションを指定できます。

- **不正なファイル:** 不正なファイルの名前とロケーションを指定すると、Warehouse Builder の指示により、Oracle Database は、エラーのためロードされなかったすべてのレコードを、このファイルに書き込みます。たとえば、不正なファイルに書き込まれるレコードには、フィールドを外部表の列に変換する際に、データ型エラーが原因でロードされなかったレコードなどがあります。すでに存在する不正なファイルを指定すると、そのファイルは上書きされます。
- **廃棄ファイル:** 廃棄ファイルの名前とロケーションを指定すると、Warehouse Builder の指示により、Oracle Database は、ファイルに設定された SQL *Loader のロード条件に基づいてロードされなかったすべてのレコードを、このファイルに書き込みます。すでに存在する廃棄ファイルを指定すると、そのファイルは上書きされます。
- **ログ・ファイル:** ログ・ファイルの名前とロケーションを指定すると、Warehouse Builder の指示により、Oracle Database は、外部表に関連するメッセージのログを、このファイルに書き込みます。すでに存在するログ・ファイルを指定すると、新しいメッセージが追加されます。

この 3 種類のファイルそれぞれについて、ファイルの名前とロケーションを指定するか、「未使用」または「デフォルトのロケーションを使用」を選択できます。

拒否

「拒否」では、拒否できる行の数を指定できます。デフォルトでは、拒否できる行の数は無限です。「無制限に拒否」を false に設定する場合は、「拒否できる数」に数値を入力します。

パラレル

パラレル : パラレル処理を有効にします。単一のシステムを使用している場合、パフォーマンスを向上するには、この値を NONPARALLEL に設定します。複数のシステムを使用している場合、デフォルトの PARALLEL をそのまま使用します。データ・ファイルはアクセス・ドライバによって、個別に処理可能なチャンクに分割されます。次のファイル、レコード、およびデータ特性では、ファイルのパラレル処理ができません。

- シーケンシャルなデータ・ソース（テープ・ドライブやパイプなど）。
- マルチバイト・キャラクタ・セット内にあり、文字列中で任意のバイトから始まっているために文字境界を決定できないデータ。この制限は、レコード当たりのバイト数が固定されているデータ・ファイルには適用されません。
- VAR フォーマットのレコード

データ特性

外部表をリポジトリにインポートしたり、ソース・ファイルを指定せずに外部表を作成した場合は、これらのプロパティを構成しないでください。データ特性のプロパティは、「外部表プロパティ」ウィンドウの「アクセス・パラメータ」タブの設定によって無効になります。

「データ特性」では、次のプロパティを設定できます。

- **Endian:**「Endian」プロパティのデフォルトはプラットフォームです。この指定により、Warehouse Builder は、フラット・ファイルのエンディアンが、そのファイルの常駐するプラットフォームのエンディアンに一致するものと見なします。ファイルが Windows プラットフォーム上に常駐する場合、データはリトル・エンディアンとして処理されます。ファイルが Sun Solaris または IBM MVS に常駐する場合、データはビッグ・エンディアンとして処理されます。フラット・ファイルのエンディアン値がわかっている場合は、ビッグとリトルどちらかのエンディアンを選択できます。ファイルが UTF16 で、エンディアンを示すマークがファイルの先頭に含まれている場合、Warehouse Builder はそのエンディアンを使用します。
- **文字列のサイズ :** このプロパティは、UTF16 など、マルチバイト・キャラクタ・セットのデータを Warehouse Builder で処理する方法を指定します。デフォルトでは、Warehouse Builder はデータ・ファイル内の文字列の長さをバイト単位と見なします。選択内容を変更して、文字列のサイズが文字数で指定されるようにすることもできます。

フィールド編集

外部表をリポジトリにインポートしたり、ソース・ファイルを指定せずに外部表を作成した場合は、これらのプロパティを構成しないでください。フィールド編集のプロパティは、「外部表プロパティ」ウィンドウの「アクセス・パラメータ」タブの設定によって無効になります。

「フィールド編集」では、データ・ファイル内の文字フィールドで実行される余白切捨てのタイプを指定できます。Warehouse Builder のデフォルト設定では、切捨ては実行されません。これ以外の切捨てオプションを指定すると、パフォーマンスが低下する場合があります。また、切捨てオプションの設定により、文字フィールドの左側、右側、およびその両方の空白を切り捨てることもできます。

また、SQL*Loader の切捨て機能に従って切捨てを実行するという別オプションの設定もできます。SQL*Loader の切捨てを選択すると、固定長ファイルの場合は右側が切り捨てられます。囲みを持つように指定されたデリミタ付きファイルの場合は、フィールドに囲みがない場合にのみ左側が切り捨てられます。

レコード内の欠落フィールドの処理方法を指示できます。「値が欠落しているときに NULL をロードする」オプションを true に設定すると、値の欠落したフィールドは NULL に設定されます。このプロパティを false に設定すると、値の欠落したフィールドは拒否され、指定した不正なファイルに送られます。

識別

詳細は、5-15 ページの「識別」を参照してください。

データ・ファイル

外部表と複数のフラット・ファイルを関連付けるには、外部表に少なくとも 1 つのデータ・ファイルを追加する必要があります。

データ・ファイルを追加する手順は次のとおりです。

1. 「データ・ファイル」の右側のフィールドを選択し、「...」ボタンをクリックします。

   「データ・ファイル」ダイアログが表示されます。

2. フラット・ファイルの名前を入力し、「OK」を選択します。

   「データ・ファイル」プロパティの下に、そのフラット・ファイルが表示されます。

3. 新しく追加したフラット・ファイルを開き、次のプロパティを構成します。

   データ・ファイルのロケーション: フラット・ファイルのロケーション。

   データ・ファイル名: フラット・ファイルの名前。

アドバンスト・キューの構成

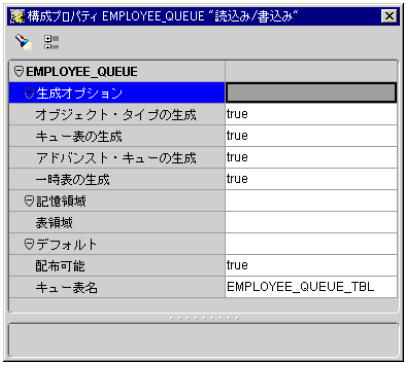
Warehouse Builder でアドバンスト・キュー（AQ）を構成するには、次の手順に従います。

アドバンスト・キューを構成する手順は次のとおりです。

1. Warehouse Builder コンソールで、アドバンスト・キューの名前を選択します。
2. 「オブジェクト」メニューから「構成」を選択します。

   図 5-6 に示すように、「構成プロパティ」ダイアログが表示されます。

図 5-6 AQ の「構成プロパティ」ウィンドウ



3. AQ に次の「記憶領域」パラメータを構成します。

   表領域 : AQ とそれに対応する表を作成する表領域の名前。

4. 次のパラメータを追加構成します。

   配布可能 : スクリプトを生成してこの AQ を配布する場合は、true を選択します。Warehouse Builder では、配布可能とマークされた AQ のみにスクリプトが生成されます。

   キュー表名 : AQ 内にメッセージを持続するために使用する AQ 表の名前を入力します。

5. 次の生成オプションを構成します。

   オブジェクト・タイプの生成 : AQ に関連付けられたオブジェクト・タイプを配布するかどうかを指定します。

   キュー表の生成 : AQ に関連付けられたキュー表を配布するかどうかを指定します。

   同じオブジェクト・タイプまたはキュー表を他の AQ と共有する第 2 の AQ を配布する場合は、これらのプロパティの生成オプションを false に設定する必要があります。このように設定しないと、AQ の配布が失敗します。

アドバンスト・キューの生成 : AQ を配布するかどうかを指定します。

一時表の生成 : AQ に関連付けられた一時表を配布するかどうかを指定します。ターゲット・システムにすでに AQ が存在している場合でも、このオプションを true に設定して、一時表を配布する必要があります。

キュー表、AQ および一時表の配布状況は、Runtime Audit Browser の「アドバンスト・キュー」で確認できます。

ディメンションの構成

ディメンションを構成するときは、ディメンションと基礎になる表の両方を構成します。

ディメンションの物理プロパティを構成する手順は次のとおりです。

1. ナビゲーション・ツリーでディメンション名を右クリックし、ポップアップ・メニューから「構成」を選択します。

   「構成プロパティ」ウィンドウが表示されます。

2. 表に列挙された構成に関するガイドラインに従ってください。詳細は、5-14 ページの「表の構成」を参照してください。

3. ディメンションに次の生成オプションを構成します。これらのパラメータを使用して、ディメンションとその基礎になる表を生成するかどうかを選択できます。

   表の生成 : true または false の設定には、ドロップダウン・リストを使用できます。基礎となる表を生成する場合は、true に設定します。基礎となる表を生成しない場合は、false に設定します。

   ディメンションの生成 : true または false の設定には、ドロップダウン・リストを使用できます。ディメンション・オブジェクトを生成する場合は、true に設定します。ディメンション・オブジェクトを生成しない場合は、false に設定します。

ディメンションの詳細は、次の項を参照してください。

- 「ディメンション定義の作成」(3-58 ページ)

キューブの構成

キューブを構成するときは、キューブと基礎になる表の両方を構成します。

キューブの物理プロパティを構成する手順は次のとおりです。

1. ナビゲーション・ツリーでキューブ名を右クリックし、ポップアップ・メニューから「構成」を選択します。

   「構成プロパティ」ウィンドウが表示されます。

2. 表に列挙された構成に関するガイドラインに従ってください。詳細は、5-14 ページの「表の構成」を参照してください。

キューブには追加の構成パラメータ・セットはありませんが、キューブを構成するためのガイドラインを次にいくつか示します。

- どのディメンションにも外部キー制約が存在する。
- どの外部キー列についても、その列が参照するディメンションにビットマップ索引が生成されている。

キューブの詳細は、次の項を参照してください。

- 「キューブの使用方法」（3-67 ページ）
- 「キューブ定義の作成」（3-67 ページ）

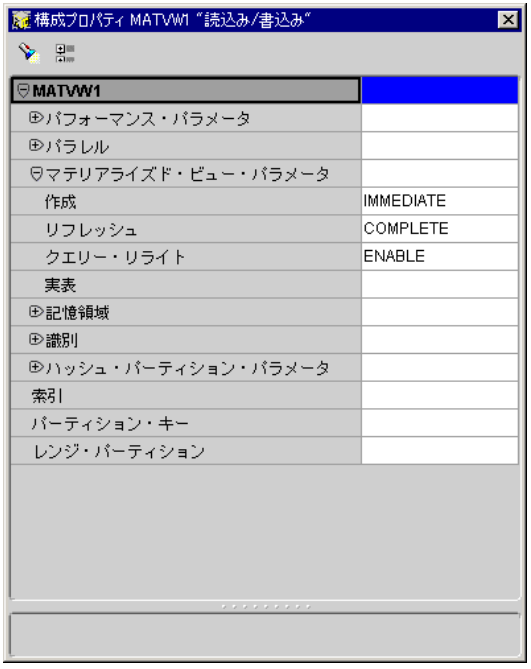
マテリアライズド・ビューの構成

マテリアライズド・ビューの物理プロパティを構成する手順は次のとおりです。

1. ナビゲーション・ツリーでマテリアライズド・ビューの名前を右クリックし、「構成」を選択します。

   図 5-7 に示すように、「構成プロパティ」ウィンドウが表示されます。

図 5-7 マテリアライズド・ビューの構成プロパティ



2. 表に列挙された構成に関するガイドラインに従ってください。詳細は、5-14 ページの「表の構成」を参照してください。
3. 次の項のマテリアライズド・ビュー・パラメータを構成します。

マテリアライズド・ビュー・パラメータ

マテリアライズド・ビューのパラメータを次に示します。

作成

- **IMMEDIATE:** （デフォルト）作成時に、マテリアライズド・ビューにデータを移入します。
- **DEFERRED:** 次のリフレッシュ操作まで、マテリアライズド・ビューへの移入を遅らせます。マテリアライズド・ビューの設計中、実表のメタデータは正しいがデータは正しくない場合に、このオプションを選択できます。

リフレッシュ

- **COMPLETE:** （デフォルト）Oracle Database は、リフレッシュ時にマテリアライズド・ビューを切り捨て、問合せを再実行します。
- **FAST:** 実表のデータに変更のみを適用するために、マテリアライズド・ビューを使用します。高速リフレッシュを正常に実行するには、多くの要件があります。詳細は、5-25 ページの「マテリアライズド・ビューの高速リフレッシュ」を参照してください。
- **FORCE:** Oracle Database は、高速モードを使用してリフレッシュを実行します。高速モードでリフレッシュできない場合、Oracle Database はリフレッシュ時に問合せを再実行します。

クエリー・リライト

- **ENABLE:** （デフォルト）クエリー・リライトを有効にします。その他のクエリー・リライト要件は、5-25 ページの「マテリアライズド・ビューの高速リフレッシュ」を参照してください。
- **DISABLE:** クエリー・リライトを無効にします。クエリー・リライトは、マテリアライズド・ビュー内のデータが失効していることがわかっていたり、問合せ文を変更する場合に無効にできます。

実表

「実表」を構成するには、マテリアライズド・ビューが参照する表の名前を入力する必要があります。それぞれの表名はカンマで区切ります。表名が大文字でない場合は、名前を二重引用符で囲みます。デフォルトでは、このフィールドは空です。

関連情報は、3-40 ページの「マテリアライズド・ビューの使用方法」を参照してください。

マテリアライズド・ビューの高速リフレッシュ

Warehouse Builder では、マテリアライズド・ビューは、増分リフレッシュを実行するように構成できます。マテリアライズド・ビューの実表を更新すると、データベースによって、マテリアライズド・ビューのログに、更新されたレコードのポインタが格納されます。ログ表内の変更内容は、関連付けられたマテリアライズド・ビューのリフレッシュに使用されます。

Warehouse Builder でマテリアライズド・ビューを確実に増分リフレッシュするには、次の条件を確認します。

- 「リフレッシュ」パラメータが FAST に設定され、「実表」パラメータにすべての実表が列挙されている必要があります。
- それぞれの実表に主キー制約が定義されている必要があります。Warehouse Builder は、この主キー制約に基づいて CREATE 文を生成し、そのログを使用して、その表に依存するマテリアライズド・ビューをリフレッシュします。

- マテリアライズド・ビューには、SYSDATE や ROWNUM のような繰り返しの存在しない式への参照や、繰り返しが許可されない PL/SQL ファンクションへの参照を含めることはできません。
- マテリアライズド・ビューには、データ型 RAW および LONG RAW への参照を含めることはできません。
- 結合、集計、および UNION 文が含まれているマテリアライズド・ビューには、これ以外にも制約があります。追加の制約の詳細は、『Oracle データ・ウェアハウス・ガイド』を参照してください。

ビューの構成

Warehouse Builder により、ターゲット・モジュールで定義された各ビューに対するスクリプトが生成されます。「配布可能」パラメータを true または false に設定することにより、特定のビューを配布するかどうかを構成できます。

ビューの詳細は、次の項を参照してください。

- 「ビューについて」（3-36 ページ）
- 「ビュー定義の作成」（3-36 ページ）

順序の構成

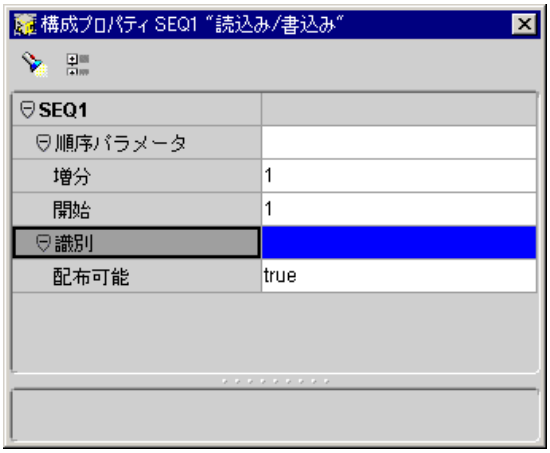
Warehouse Builder では、各順序オブジェクトに対するスクリプトを生成します。順序オブジェクトには、開始および増分パラメータがあります。このパラメータは両方とも数値です。

順序の物理プロパティを構成する手順は次のとおりです。

1. 順序の名前を右クリックし、「構成」を選択します。

   図 5-8 に示すように、「構成プロパティ」ダイアログが表示されます。

図 5-8 順序の「構成プロパティ」ウィンドウ



2. 順序に次のプロパティを構成します。

   増分 : 順序の増分に当たる数値。

   開始 : 順序を開始する数値。

3. 次の識別パラメータを構成します。

   配布可能 : この順序を配布する場合は、true を選択します。Warehouse Builder では、配布可能とマークされた順序のみにスクリプトが生成されます。

関連情報は、次の項を参照してください。

- 「順序について」(3-44 ページ)
- 「順序定義の作成」(3-45 ページ)

第 II 部

ETL オブジェクトの設計

この部には、次の章があります。

6

マッピングの設計

Warehouse Builder でデータ・オブジェクトの定義の作成およびインポートが完了したら、データをソースからターゲットに移動する抽出、変換およびロード（ETL）の各処理を定義できます。Warehouse Builder では、マッピングを使用してこれらの処理を設計します。

この章では、マッピングの作成、編集および使用方法に関する次のトピックについて説明します。

マッピングについて

マッピングには、ソースからデータを抽出し、そのデータを変換してターゲットにロードするという一連の処理が記述されます。これにより、データ・フローとデータに対して実行される処理が視覚的に表されます。

Warehouse Builder でマッピングを設計するときは、マッピング・エディタ・インタフェースを使用します。Warehouse Builder のスクリプト・インタフェースである OMB Plus を使用して、マッピングの作成と定義を行うこともできます。OMB Plus でマッピングを作成および定義する方法の詳細は、『Oracle Warehouse Builder スクリプト・リファレンス』を参照してください。

Oracle ウェアハウス・モジュールについて

マッピングの設計を始めるには、その前に Oracle ウェアハウス・モジュールを定義する必要があります。Warehouse Builder では数種類のモジュールを用意していますが、マッピング・ロジックを格納するモジュールは、Oracle ウェアハウス・モジュールのみです。Oracle ウェアハウス・モジュールの詳細は、3-2 ページの「ウェアハウス・モジュールの作成」を参照してください。

マッピングを定義する手順

マッピングを定義するときは、ETL ロジックを定義した演算子を保持するコンテナを作成します。マッピングを定義するには、次の手順を実行します。

1. マッピングの作成（6-3 ページ）
2. 演算子の追加（6-9 ページ）
3. 演算子の編集（6-13 ページ）
4. 演算子の接続（6-21 ページ）
5. 演算子のプロパティの設定（6-29 ページ）
6. マッピング構成のリファレンス（11-2 ページ）
7. 演算子とリポジトリ・オブジェクトの調整（6-41 ページ）

マッピングを定義した後は、マッピングに使用したコードを検証できます。詳細は、第 12 章「オブジェクトの検証」を参照してください。プロセス・フローを作成して、マッピングを相互に関連付けることもできます。詳細は、第 10 章「プロセス・フローの設計」を参照してください。

マッピングの作成

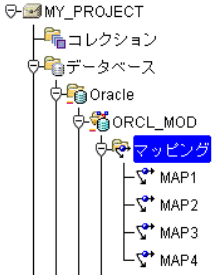
マッピングの定義における最初の手順では、マッピング・ウィザードを使用してマッピングします。

マッピングを作成する手順は次のとおりです。

1. プロジェクトの「マッピング」ノードに移動します。「マッピング」ノードは、「データベース」ノードの下の Oracle フォルダ内の、Oracle ウェアハウス・モジュールにあります。

   図 6-1 は、マップ MAP1、MAP2、MAP3、MAP4 が含まれる「マッピング」ノードを示しています。この例の Oracle ウェアハウス・モジュールの名前は、ORCL_MOD です。

図 6-1 ナビゲーション・ツリーの「マッピング」ノード



2. 「マッピング」を右クリックして、「マッピングの作成」を選択します。

   新規マッピング・ウィザードが表示されます。

3. 「次へ」をクリックします。

   「新規マッピング・ウィザード : 名前」ページが表示されます。

4. 新しいマッピングの名前と説明を入力します。

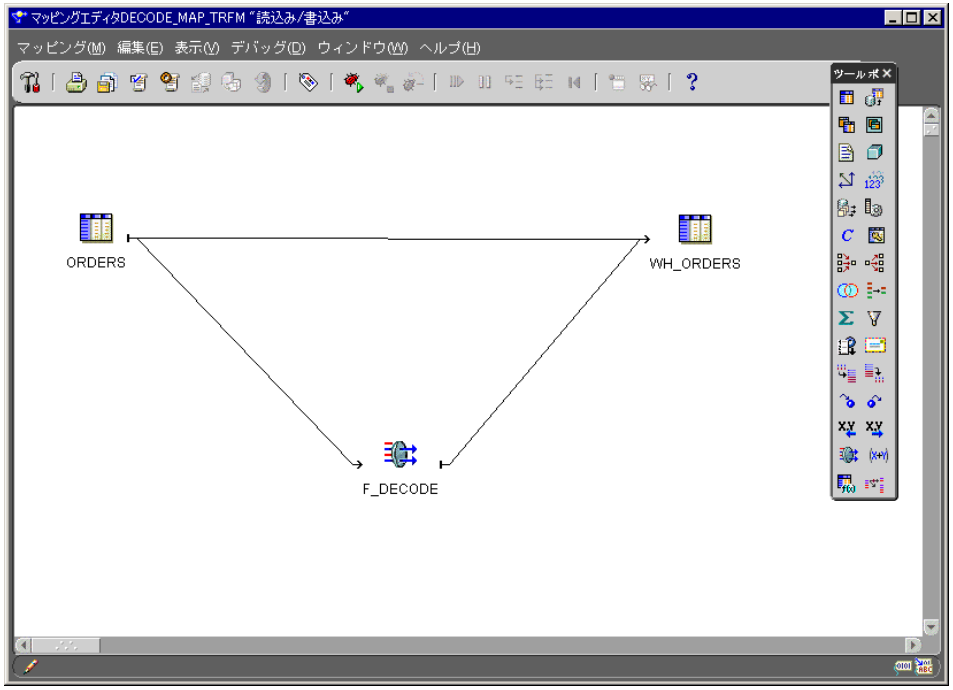
5. 「OK」をクリックします。

   マッピングの定義が保存され、その名前がウェアハウス・モジュールのナビゲーション・ツリー内に挿入されます。作成したマッピング用にマッピング・エディタが起動します。マッピング・エディタのタイトル・バーには、作成したマッピングの名前が表示されます。

マッピング・エディタについて

マッピング・エディタは、マッピングの設計や編集に使用するインタフェースです。マッピング・エディタには、マッピングの設計時に追加または接続できる様々な演算子が用意されています。図 6-2 は、3 つの演算子が接続されているマッピング・エディタを示しています。

図 6-2 マッピング・エディタのキャンバス



マッピング・エディタは、次のコンポーネントで構成されています。

- **メニュー・バー :** マッピング・エディタのコマンドを実行できます。
- **ツールバー :** 頻繁に使用するコマンドを実行できます。
- **ツールボックス :** 演算子のアイコンが表示されます。マッピングに演算子を追加するには、演算子のアイコンをマッピング・エディタのキャンバスにドラッグします。
- **マッピング・エディタのキャンバス :** マッピングの作成および変更を行う作業領域です。

マッピング・エディタを起動する手順は次のとおりです。

1. ナビゲーション・ツリーで Oracle ウェアハウス・モジュールを見つけます。

   これらのモジュールは、6-3 ページの図 6-1 に示すように、プロジェクトまたはコレクションの Oracle データベース・フォルダの下にあります。プロジェクトまたはコレクションにウェアハウス・モジュールがない場合は、3-2 ページの「ウェアハウス・モジュールの作成」の手順を参照して作成してください。

2. 「マッピング」ノードを開きます。
3. 次のいずれかの操作で、マッピング・エディタを起動します。
   - マッピングをダブルクリックします。
   - マッピングを選択し、「オブジェクト」メニュー「エディタ」を選択します。
   - マッピングを選択し、[Ctrl] を押しながら [O] を押します。
   - マッピングを右クリックし、「エディタ」を選択します。

演算子について

マッピングの基本となる設計要素は演算子です。Warehouse Builder でマッピングを設計するときは、マッピング・エディタのツールボックスから演算子を選択して、キャンバスにドラッグします。

Warehouse Builder には、次のタイプの演算子が用意されています。

- **ソース演算子とターゲット演算子**: ソース演算子とターゲット演算子は、マッピングにおいて、リレーショナル・データベースのオブジェクトとフラット・ファイルのオブジェクトを表します。SAP ソース・データの使用に関する詳細は、「Warehouse Builder での SAP R/3 データの使用」を参照してください。
- **データ・フロー演算子**: データ・フロー演算子はデータを変換します。

ソース演算子とターゲット演算子

リレーショナル・データベースのオブジェクトとフラット・ファイルのオブジェクトを表すには、ソース演算子とターゲット演算子を使用します。表 6-1 は、ソース演算子およびターゲット演算子の簡単な説明と、マッピング・エディタに表示されるアイコンを示しています。

表 6-1 ソース演算子とターゲット演算子

アイコン	演算子	説明
	アドバンスト・キューのマッピング	インポート済のアドバンスト・キューを表します。
	キューブのマッピング	定義済のキューブを表します。
	ディメンションのマッピング	定義済のディメンションを表します。
	外部表のマッピング	定義済またはインポート済の外部表を表します。
	フラット・ファイルのマッピング	定義済またはインポート済のフラット・ファイルを表します。
	マテリアライズド・ビューのマッピング	定義済のマテリアライズド・ビューを表します。
	表のマッピング	定義済またはインポート済の表を表します。
	ビューのマッピング	定義済またはインポート済のビューを表します。

データ・フロー演算子

マッピング内のデータを変換するには、データ・フロー演算子を使用します。表 6-2 は、データ・フロー演算子の簡単な説明と、マッピング・エディタに表示されるアイコンを示しています。演算子の詳細は、第 8 章「マッピング演算子の使用方法」を参照してください。

表 6-2 データ・フロー演算子

アイコン	演算子	説明
	集計演算子	SUM や AVG などのデータ集計を実行し、集計されたデータを使用して行セットを出力します。
	定数演算子	1 つの出力グループが作成されます。このグループには、1 つ以上の定数属性を含めることができます。
	データ・ジェネレータ演算子	レコード番号、システムの日付、順序の値などの情報を表示します。
	デュプリケータ解除演算子	マッピングで生成されたコードの SELECT 文に DISTINCT 句を挿入して、ソース内の重複するデータを削除します。
	式演算子	この演算子の 1 つの出力パラメータについて、非プロシージャ・アルゴリズムを定義する SQL 式を記述できます。式では、入力パラメータ名、変数名およびライブラリ・ファンクションの組合せを使用できます。
	フィルタ演算子	行セットの行を特定の条件に基づいてフィルタリングします。
	結合子演算子	カーディナリティの異なる複数のソースから取り込んだ複数の行セットを結合し、1 つの出力行セットを生成します。
	キー検索演算子	表、ビュー、キューブ、ディメンションなどの参照オブジェクトからデータを検索します。
	入力パラメータのマッピング演算子	パラメータ値をマッピングに渡します。
	出力パラメータのマッピング演算子	マッピングの外部に値を渡します。
	順序のマッピング演算子	1 行ごとに増分する連続番号を生成します。
	Name and Address 演算子	Name and Address のソース・データにおけるエラーと非一貫性を識別し修正します。

表 6-2 データ・フロー演算子（続き）

アイコン	演算子	説明
	ピボット演算子	複数の属性が含まれる単一行を複数の行に変換します。複数の行ではなく、複数の属性間に含まれるデータを変換する場合に、この演算子を使用します。
	マッピング後プロセス演算子	マッピングを実行した後に、ファンクションまたはプロシージャをコールします。
	ピボット演算子	マッピングを実行する前に、ファンクションまたはプロシージャをコールします。
	データ・ジェネレータ演算子	マッピングで UNION、UNION ALL、INTERSECT および MINUS の演算を行います。
	ソーター演算子	属性を昇順または降順にソートします。
	スプリッタ演算子	ブール分割条件に基づいて、1 つの入力行セットを複数の出力行セットに分割します。
	マッピング後プロセス演算子	独自のコードを開発して入力行セットを操作し、同じカーディナリティまたは異なるカーディナリティから、物理表などの問合せ可能な出力行セットを返すことができます。
	変換演算子	PL/SQL ファンクションまたはプロシージャに基づいて、行セット内の行の属性値データを変換します。
	アンピボット演算子	複数の入力行を 1 つの出力行に変換します。これにより、ソース・データ内の属性ごとにグループ化されているソース行セットからソースをいったん抽出し、そこから 1 つの行を作成できるようになります。

演算子の追加

Warehouse Builder では、追加したすべてのソース演算子とターゲット演算子に対して、そのオブジェクトの Warehouse Builder Design Repository 用のバージョンと、マッピング・エディタ用のバージョンが個々に作成され、維持されます。たとえば、表をマッピングに追加した場合は、その表のコピーがリポジトリに作成され、維持されます。このコピーをリポジトリ表と言い、マッピングに追加した表をマッピング表といいます。

Warehouse Builder では、ソース演算子とターゲット演算子にリポジトリ・オブジェクトとマッピング・オブジェクトが個別に維持されるのみではなく、次のデータ・フロー演算子についても、リポジトリ・オブジェクトが個別に維持されます。

- キー参照
- 入力パラメータのマッピング
- 出力パラメータのマッピング
- 順序のマッピング
- 変換

Warehouse Builder では、リポジトリ・オブジェクトが個別に維持されるので、オブジェクトの定義の変更を調整することができます。たとえば、マッピング表を変更した場合は、対応するリポジトリ表にその変更を伝播する必要があります。反対に、リポジトリ表に新しいメタデータ定義をインポートした場合は、その変更をマッピング表に伝播する必要があります。これらのタスクは、調整と呼ばれる処理により実現します。調整を行うときは、一方のオブジェクト定義を他方のオブジェクト定義にバインドします。オブジェクトのバインドの詳細は、6-41 ページの「演算子とリポジトリ・オブジェクトの調整」を参照してください。

マッピング演算子をリポジトリ・オブジェクトにバインドするという概念は、演算子をマッピングに追加する方法に影響するため、理解しておく必要があります。

演算子をマッピングに追加する手順は次のとおりです。

1. マッピング・エディタを起動します。
2. 「マッピング」メニューから「追加」を選択し、演算子を選択します。または、ツールボックスにある演算子のアイコンを、マッピング・エディタのキャンバスにドラッグします。

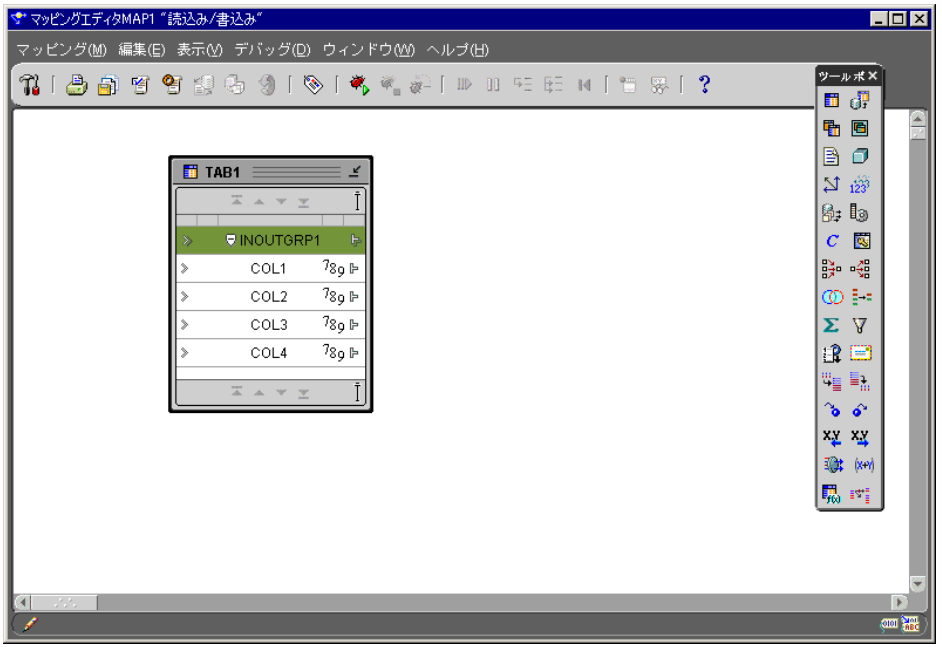
   リポジトリ・オブジェクトにバインドできる演算子を選択すると、マッピング・エディタに「マッピング < 演算子名 > の追加」ダイアログが表示されます。このダイアログの使用方法の詳細は、6-11 ページの「バインド可能な演算子の追加」を参照してください。

   リポジトリ・オブジェクトにバインドできない演算子を選択した場合は、演算子を作成するためのウィザードまたはダイアログが表示されます。詳細は、第 8 章「マッピング演算子の使用方法」を参照してください。この章では、各演算子について、マッピングを設計する際の使用方法を説明しています。

3. Warehouse Builder に表示される指示に従い、「OK」をクリックします。

   図 6-3 に示すように、マッピング・エディタのキャンバスに、最大化された演算子が表示されます。ここでは、各属性名とデータ型を確認できます。演算子名は左上隅に表示されます。アイコンを最小化する場合は、右上隅の矢印をクリックします。

図 6-3 マッピング・エディタに表示された表のマッピング演算子のソース



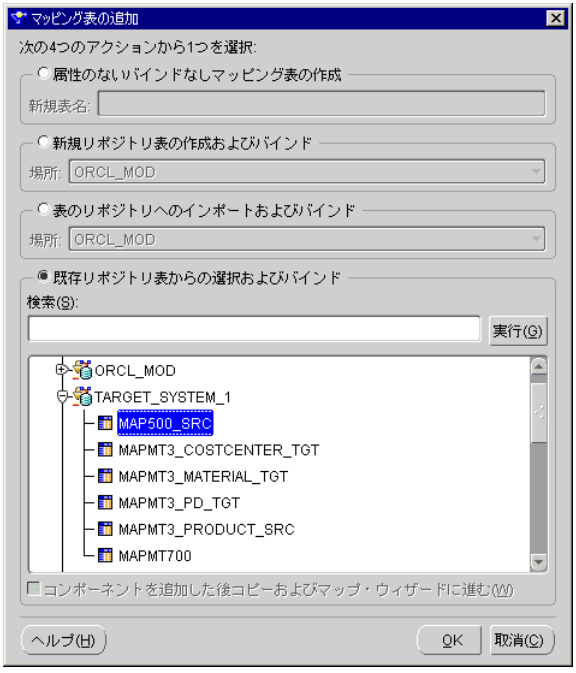
バインド可能な演算子の追加

バインド可能な演算子をマッピングに追加するときは、この項を参照してください。バインド可能な演算子には、次のものがあります。

- 表のマッピング
- 外部表のマッピング
- フラット・ファイルのマッピング
- ディメンションのマッピング
- アドバンスト・キューのマッピング
- キューブのマッピング
- ビューのマッピング
- マテリアライズド・ビューのマッピング
- キー参照
- 順序のマッピング
- 変換
- マッピング前プロセス
- マッピング後プロセス

リポジトリ・オブジェクトにバインドできる演算子を追加すると、マッピング・エディタに「マッピング < 演算子名 > の追加」ダイアログが表示されます。図 6-4 は、表のマッピング演算子を追加した場合に表示されるダイアログです。

図 6-4「マッピング表の追加」ダイアログ



次の 4 つのいずれかのオプションを選択します。選択した演算子のタイプによっては、グレー表示されるオプションもあります。

- 属性のないバインドなしマッピング表の作成
- 新規リポジトリ表の作成およびバインド
- 表のリポジトリへのインポートおよびバインド
- 既存リポジトリ表からの選択およびバインド

属性のないバインドなしマッピング表の作成

このオプションは、マッピング・エディタを使用して新しいオブジェクトを定義する場合に選択します。「属性のないバインドなしマッピング表の作成」を選択した後、新しいオブジェクトの名前を入力します。演算子がキャンバスに表示されます。ただし、属性は表示されません。バインドのない表演算子と「自動マッピング」ダイアログを使用して、すばやくステージング表を作成する方法の例については、6-23 ページの「例 : マッピング・エディタによるステージング領域表の作成」を参照してください。

新規リポジトリ表の作成およびバインド

このオプションは、ウィザードを使用して新しいオブジェクトを作成する場合に選択します。最初にターゲット・モジュールを選択します。Warehouse Builder に、新しい演算子の作成に使用するウィザードが表示されます。ウィザードを終了すると、作成した演算子のコピーがリポジトリに保存されます。たとえば、マッピング表を追加するときにこのオプションを選択すると、新規表ウィザードが起動し、マッピング表とリポジトリ表が作成され、マッピング表がリポジトリ表にバインドされます。

表のリポジトリへのインポートおよびバインド

このオプションは、リポジトリにインポートできるオブジェクトに基づいて、演算子を作成する場合に選択します。オブジェクトのインポート先のモジュールを選択します。

データベース・リンクが未定義のモジュールを選択すると、「データベース・リンク情報」ダイアログが表示されます。新規データベース・リンクの定義方法の詳細は、4-5 ページの「データベース・ソースの接続の構成」を参照してください。

データベース・リンクが定義済のモジュールを選択すると、インポート・ウィザードが表示されます。適切な情報をこのダイアログに入力します。

既存リポジトリ表からの選択およびバインド

このオプションは、リポジトリ内の定義済またはインポート済のオブジェクトに基づいて、マッピングに演算子を追加する場合に選択します。

接頭辞を入力してオブジェクトを検索するか、選択したモジュール内のオブジェクトの表示リストからオブジェクトを選択します。

複数のオブジェクトを選択するには、[Ctrl] キーを押しながら各オブジェクトをクリックします。連続したオブジェクトのグループを選択するには、選択範囲内の最初のオブジェクトをクリックし、[Shift] キーを押しながら最後のオブジェクトをクリックします。

演算子は、マッピングと同じモジュールのリポジトリ・オブジェクトからも、別のモジュールのリポジトリ・オブジェクトからも追加できます。別のモジュールのリポジトリ・オブジェクトを選択した場合に、マッピング・エディタにそのオブジェクトとのコネクタが存在しないときは、コネクタが作成されます。このコネクタにより、マッピングのロケーションとリポジトリ・オブジェクトのロケーション間のデータの移動パスが確立します。ロケーションとコネクタの詳細は、3-3 ページの「ロケーションの定義」を参照してください。

演算子の編集

演算子には、それぞれに関連付けられたエディタがあります。その演算子エディタを使用して、演算子、グループおよび属性に関する一般的な情報と構造的な情報を指定します。演算子エディタでは、グループと属性の追加、削除および名前の変更を行うことができます。演算子の名前も変更できます。

ロードのプロパティと条件付きの動作を指定するには、プロパティ・ウィンドウを使用します。プロパティ・ウィンドウの詳細は、6-17 ページの「マッピングでのネーミング規則」を参照してください。

演算子、グループまたは属性を編集する手順は次のとおりです。

1. マッピング・エディタのキャンバスから演算子を選択します。

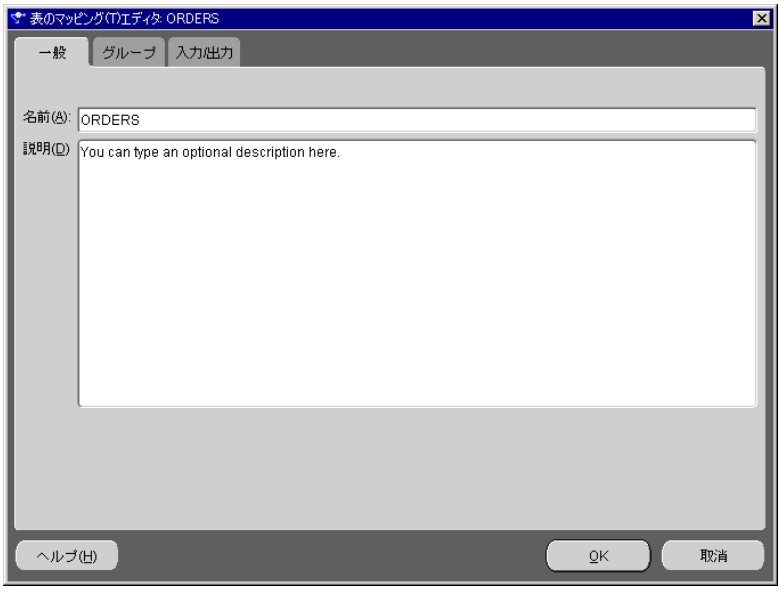
   あるいは、演算子内のグループまたは属性を選択します。

2. 選択したアイテムを右クリックし、「編集」を選択します。

   マッピング・エディタ内で演算子エディタが起動し、「一般」タブ、「グループ」タブ、および演算子内の各グループ・タイプのタブが表示されます。図 6-5 は、演算子エディタの「一般」タブを示しています。

   「一般」タブには、演算子名と説明（オプション）が表示されます。ここでは、演算子の名前を変更したり、説明を追加することができます。演算子に名前を付けるときは、6-17 ページの「演算子のネーミング規則」に記載されている規則に従ってください。

図 6-5 演算子エディタの「一般」タブ



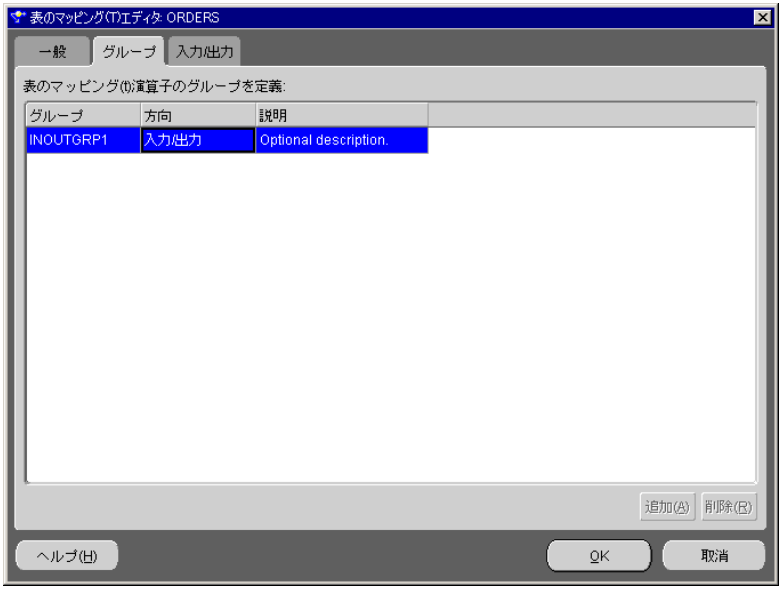
1. 図 6-6 に示すように、「グループ」タブでグループ情報を編集します。

   各グループには、名前、方向および説明（オプション）が表示されます。グループの名前は変更できますが、グループの方向は変更できません。グループの方向には、「入力」、「出力」、「入力 / 出力」のいずれかを指定できます。

   演算子によっては、「グループ」タブでグループを追加および削除できる場合があります。たとえば、入力グループを「結合子」に、出力グループを「スプリッタ」に追加できます。

   演算子エディタには、「グループ」タブに表示されているグループのタイプごとに、タブが 1 つずつ表示されます。これらのタブには、属性名、データ型、長さ、精度、スケールおよび説明（オプション）が表示されます。

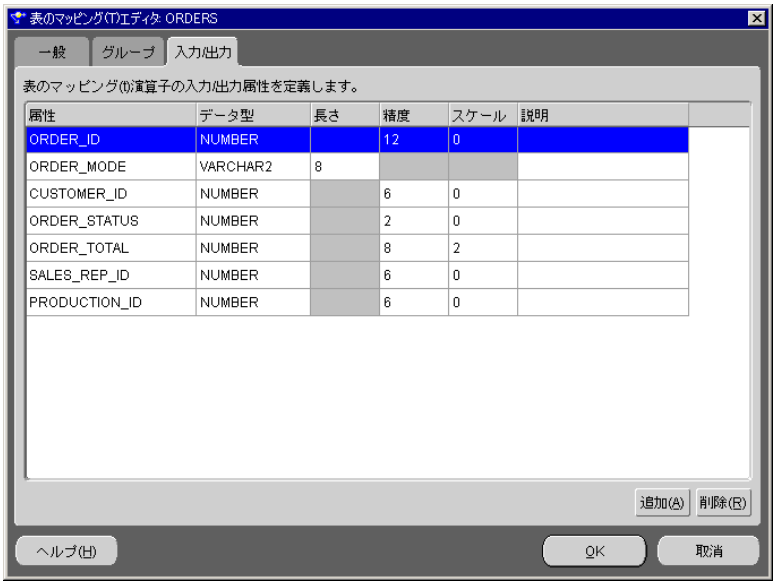
図 6-6 演算子エディタの「グループ」タブ



2. 残りのそれぞれのタブで、属性情報を編集します。

   図 6-7 は、演算子エディタの「入力 / 出力」タブを示しています。この例では、演算子として表を選択しているので、「入力 / 出力」タブのみが表示されます。他の演算子では、「入力」タブや「出力」タブが表示される可能性もあります。

図 6-7 演算子エディタの「入力 / 出力」タブ



属性は追加、削除および編集できます。マッピング・エディタでは、編集できないプロパティはグレー表示になります。たとえば、データ型が NUMBER の場合、精度とスケールは編集できますが、長さは編集できません。

属性のデータ型、長さ、精度およびスケールに正しい値を設定するには、PL/SQL ルールに従います。演算子を調整すると、Warehouse Builder では SQL ルールに基づいて属性がチェックされます。

マッピングでのネーミング規則

マッピング・エディタでオブジェクトに名前を付けたり、名前を変更したりするときは、次のネーミング規則に従います。

属性およびグループのネーミング規則

属性名とグループ名は論理名です。通常、オブジェクトの属性名は演算子の属性名に一致しますが、そのプロパティはバインド先の演算子の属性から独立した状態で維持されます。これにより、属性を演算子内で操作した場合に、属性の表現または使用が損なわれるのを防ぐことができます。ソースに依存しないグループおよび属性は名前を変更できます。

演算子のネーミング規則

演算子をマッピングに追加すると、マッピング・エディタにその演算子の物理名またはビジネス名が表示されます。どちらが表示されるかは、プロジェクトの「作業環境」ウィンドウで選択したネーミング・モードによります。ネーミング・モードの指定方法は、2-25 ページの「ネーミング作業環境」を参照してください。

プロジェクトの「作業環境」ウィンドウでビジネス・ネーミング・モードを選択した場合は、演算子名がビジネス名として表示されます。ビジネス・ネーミング・モードでマッピングに演算子を追加すると、その演算子にデフォルトのビジネス名が生成されます。ビジネス名は任意の名前に変更できますが、次の要件を満たしている必要があります。

- 使用できる文字数は 200 文字以内です。名前にスペースが含まれる場合は、二重引用符で囲む必要があります。
- 属性グループ内、属性内、および親に対する表示セット・レベル内で一意である必要があります。

プロジェクトの「作業環境」ウィンドウで物理ネーミング・モードを選択した場合は、演算子名が物理名として表示されます。Warehouse Builder の OMB Scripting Language を使用している場合は、物理名に基づいて演算子間を移動します。物理名はスクリプトの生成時に使用されるため、次の要件を満たしている必要があります。

- グループ内、属性内、および親に対する表示セット・レベル内で一意である必要があります。
- 使用できる文字数は 28 文字以内です。2 文字は Warehouse Builder によって使用されます。
- 『Oracle Database SQL リファレンス』で定義されている基本要素の構文規則に準拠している必要があります。

演算子には、物理名とビジネス名のほかにバウンド名もあります。バウンド名は、バインドできる演算子に対して、論理プロパティおよび物理プロパティとして使用できます。バウンド名は、コードの生成時にオブジェクトの参照に使用され、次のような特性を持ちます。

- バウンド名は一意である必要はありません。
- バウンド名は、一般的な Warehouse Builder の物理ネーミング規則に準拠している必要があります。
- 一般に、バウンド名は直接変更せず、アウトバウンド調整で変更します。
- 演算子または属性の論理名を変更した場合は、アウトバウンド調整を行ったときに、新しい論理名がバウンド名として伝播されます。ただし、論理名は 200 文字まで使用できますが、バウンド名は 30 文字に制限されているため、Warehouse Builder では、論理名の最初の 30 文字がバウンド名として使用されます。

表示セットの使用

表示セットは、属性のサブセットの視覚的な表現です。表示セットを使用すると、演算子内に表示する属性の数を制限して、複雑なマッピングを簡略化することができます。

デフォルトでは、演算子には「すべて」という表示セットがあり、演算子で表されるリポジトリ・オブジェクト内のすべての属性が表示されます。

また、演算子には「MAPPED」というデフォルトの表示セットもあります。この表示セットには、グループ内の属性でマッピング内の他の演算子に接続されている属性のみが表示されます。マッピング・エディタで属性の接続を変更したときは、「MAPPED」表示セットは自動的に更新されます。表 6-3 は、デフォルトの表示セットを示しています。

表 6-3 デフォルトの表示セット

表示セット	説明
すべて	演算子内の属性がすべて含まれます。
MAPPED	演算子内の属性で、他の演算子に接続されている属性のみが含まれます。
階層	ディメンション内の階層ごとに、その階層のすべてのレベル属性が表示セットに含まれます。

属性セットの詳細は、3-25 ページの「属性セットの追加」を参照してください。演算子の属性とグループの詳細は、6-13 ページの「演算子の編集」を参照してください。

表示セットの定義

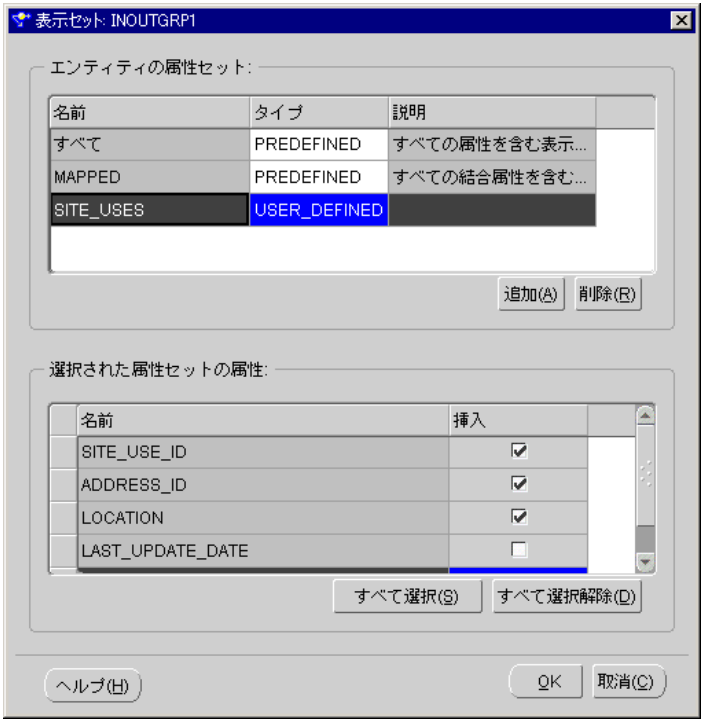
マッピング内の任意の演算子に対して、表示セットを定義できます。

表示セットを定義する手順は次のとおりです。

1. 演算子内のグループを右クリックして、「表示セット」を選択します。

   図 6-8 に示すように、「表示セット」ダイアログが表示されます。

図 6-8「表示セット」ダイアログ



2. 「追加」をクリックします。
3. 「名前」列に表示セットの名前を入力し、[Enter] を押します。

   「選択された属性セットの属性」フィールドに、演算子に使用できるすべての属性が表示されます。「タイプ」列に USER_DEFINED が自動的に設定されます。

   定義済の属性セットは編集および削除できません。

4. 「挿入」列で、表示セットに含める属性を選択します。

   すべての属性を含めるには「すべて選択」を、すべての属性を除外するには「すべて選択解除」をクリックします。

5. 「OK」をクリックします。

   演算子のグループには、表示用に選択した属性セットに含まれる属性のみが一覧表示されます。

表示セットの選択

1 つのグループに複数の表示セットがある場合は、「表示」メニューのリストから異なる表示セットを選択できます。

表示セットを選択する手順は次のとおりです。

1. 演算子内のグループを右クリックします。
2. 「表示セットの使用」を選択して、表示セットを選択します。

マッピングの圧縮

ソース演算子またはターゲット演算子をマッピングに追加すると、物理ソースまたはターゲットのすべての列が属性として演算子に追加されます。これらの属性には、マッピングに不要なものもあります。マッピングを圧縮すると、他のいずれの属性にも接続されていないすべてのソース・マッピング属性およびターゲット・マッピング属性がマッピングから削除されます。これにより、マッピングの表示が簡略化され、MDL ファイルのインポートおよびエクスポートのパフォーマンスも向上します。

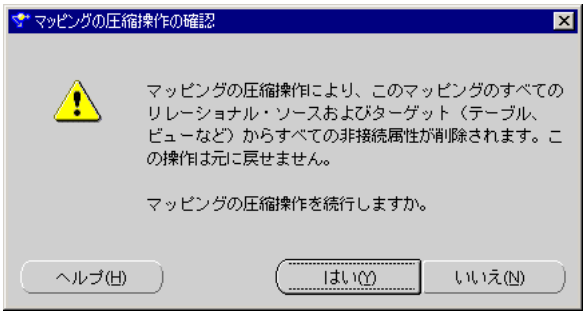
マッピングを圧縮した後に圧縮を元に戻すことができるのは、インバウンド調整のみです。アウトバウンド調整を行うと、圧縮したマッピング上のオブジェクトがリポジトリに保存されるため、ロール・バックすることはできません。

マッピングを圧縮する手順は次のとおりです。

1. マッピング・エディタでマッピングを開きます。
2. 「マッピング」メニューから「マッピングの圧縮」を選択します。

   図 6-9 に示すように、未接続の Oracle リレーショナル・ソース属性およびターゲット属性が削除されることを確認する警告メッセージが表示されます。

図 6-9「マッピングの圧縮操作の確認」ダイアログ



3. 続行する場合は、「はい」をクリックします。他のいずれの属性にも接続されていないすべてのソース・マッピング属性およびターゲット・マッピング属性がマッピングから削除されます。ただし、未接続属性でも、更新条件または削除条件およびフラット・ファイル・ソースの一部である属性は削除されません。

演算子の接続

マッピング・ソース演算子、データ・フロー演算子およびターゲット演算子の選択が完了すると、これらの演算子を接続できます。データ・フロー・コネクションは、ソースから演算子を通過してターゲットに至るデータ・フローの状態を視覚的に表します。

演算子は、次のいずれかの方法で接続できます。

- **属性の接続**: 個々の演算子属性を相互に接続します。
- **グループの接続**: グループを接続するときは、接続する属性を「自動マッピング」ダイアログを使用して制御できます。

マッピングを開始する属性の位置とマウス・ボタンを離す位置によって、マッピングで作成するデータ・フロー・コネクションのタイプが決まります。

属性の接続

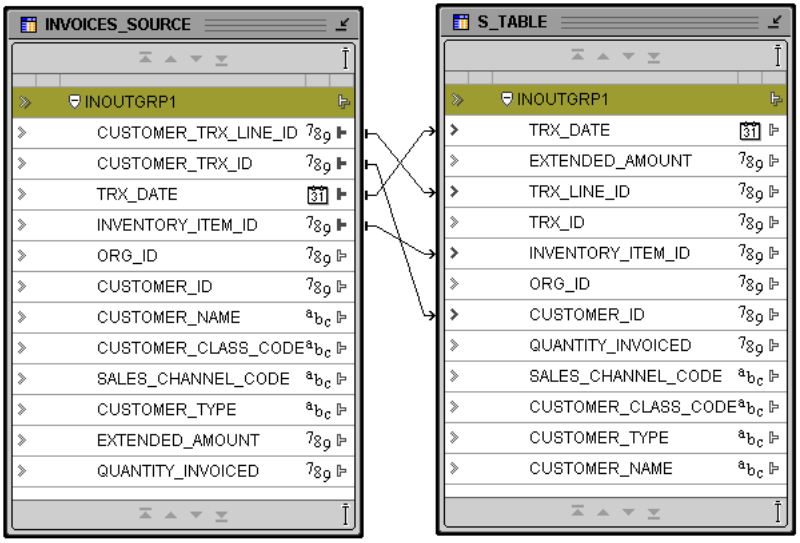
一方の演算子の 1 つの出力属性から、他方の演算子の 1 つの入力属性に線を引きます。

属性を接続する手順は次のとおりです。

1. マウス・ボタンをクリックしたまま、ポインタを出力属性の上に置きます。
2. 出力属性からデータ・フロー先の入力属性にマウスをドラッグします。

   マウスをドラッグすると、図 6-10 に示すように、接続を示す線がマッピング・エディタのキャンバスに描かれます。
3. 入力属性上でマウスを離します。

図 6-10 マッピングで接続した演算子



手順 1 ～ 3 を繰り返し、状況に応じて適切なデータ・フロー・コネクションをすべて作成します。

属性を接続するときは、次のルールに留意します。

- 同じ入力属性には二度接続できない。
- 同じ演算子内の属性には接続できない。
- 入力専用の属性からは接続できない。
- 出力専用の属性には接続できない。
- 確立されているカーディナリティに矛盾するマッピング線は作成できない。かわりに結合演算子を使用する。

グループの接続

グループを接続すると、自動的に属性がコピーされるか、または「自動マッピング」ダイアログに詳しい情報を入力するよう指示されます。「自動マッピング」ダイアログの詳細は、6-26 ページの「「自動マッピング」ダイアログの使用」を参照してください。

既存の属性を持たないターゲット・グループに接続すると、自動的に属性がコピーされ、マッピング線が作成されます。これは、「例 : マッピング・エディタによるステージング領域表の作成」に示すようなマッピングを設計する場合に効果的です。

例 : マッピング・エディタによるステージング領域表の作成

マッピング・エディタでバインドのない表演算子を使用すると、ステージング領域表をすばやく作成できます。

次の手順は、既存のソース表に基づいてステージング表を作成する方法を示しています。この手順は、ビュー、マテリアライズド・ビュー、フラット・ファイルおよび変換の作成にも使用できます。

ソース表をステージング表にマップする手順は次のとおりです。

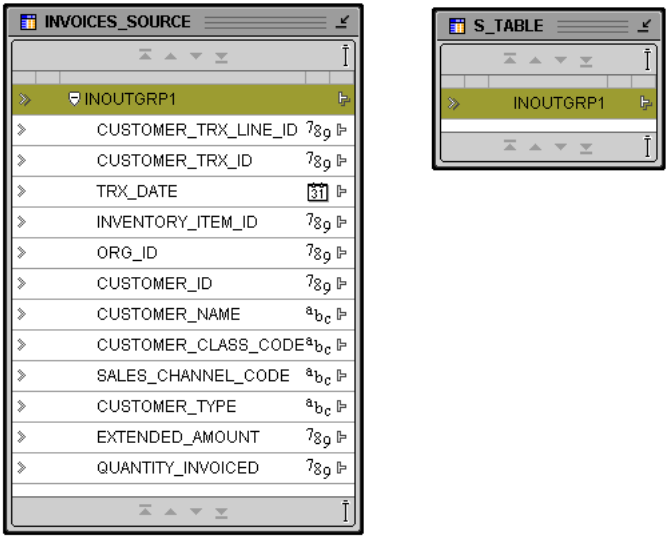
1. マッピング・エディタでソース表を追加します。

   メニュー・バーから「マッピング」、「追加」の順に選択します。「マッピング表の追加」ダイアログを使用して、マッピング内のソース表演算子を選択し、バインドします。

2. 新しいバインドのない表演算子を追加します。

   メニュー・バーから「マッピング」、「追加」の順に選択します。「マッピングの追加」ダイアログで、「属性のないバインドなしマッピング表の作成」を選択します。マッピングには、図 6-11 に示すように、1 つのソース表と、属性のない 1 つのステージング領域表が作成されます。

図 6-11 属性のないバインドなしステージング表とソース表



3. ソース表の演算子内のグループにマウス・ポインタを置きます。
4. マウス・ボタンを押しながら、ステージング領域表グループまでマウスをドラッグします。

   ソースの属性がステージング領域表にコピーされ、対応するマッピング線が作成されます。
5. マッピング・エディタで、マッピングに追加したバインドなし表を選択します。それを右クリックして、「アウトバウンドの調整」を選択します。図 6-12 に示すように、「アウトバウンド調整演算子」ダイアログが表示されます。

図 6-12「アウトバウンド調整演算子」ダイアログ

6. 「新規表の作成」を選択し、表を作成するターゲット・モジュールを指定します。

   指定したターゲット・モジュールに新しい表が作成されます。

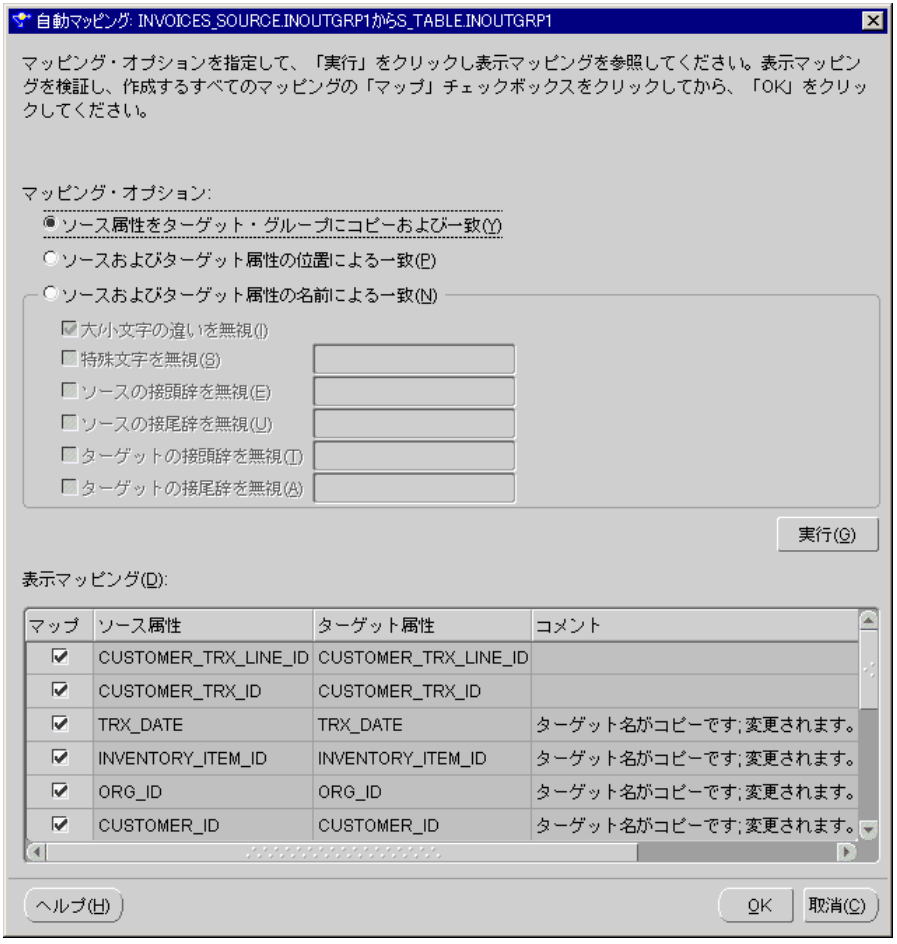
「自動マッピング」ダイアログの使用

属性を持つターゲットに接続した場合は、図 6-13 に示すように、「自動マッピング」ダイアログが起動します。

次のルールのいずれかを選択して、マッピング線をコピーまたは作成します。

- ソース属性をターゲット・グループにコピーおよび一致
- ソースおよびターゲット属性の位置による一致
- ソースおよびターゲット属性の名前による一致

図 6-13 マップされる属性が表示された「自動マッピング」ダイアログ



3 つのオプションのいずれかを選択したら、「実行」を選択します。図 6-14 に示すように、マップされるソース属性とターゲット属性の一覧が「自動マッピング」ダイアログに表示されます。

図 6-14 表示マッピング

属性の選択を解除するには、「マップ」チェック・ボックスの選択を解除します。「コメント」列に、選択の結果が表示されます。コメントの詳細は、6-29 ページの「「自動マッピング」ダイアログの「コメント」列について」を参照してください。

「OK」を選択すると、ソース属性がターゲット・グループにコピーされ、ソースとターゲット・グループ間にマッピング線が作成されます。

ソース属性をターゲット・グループにコピーおよび一致

このオプションは、すでに属性が設定されているターゲット・グループにソース属性をコピーする場合に選択します。「自動マッピング」ダイアログで行った選択内容に基づいて、ソース属性から新しいターゲット属性にマッピング線が作成されます。この処理は、ディメンション・ターゲット演算子やキューブ・ターゲット演算子など、新しい入力属性を受け入れないターゲット・グループには実行されません。

ソースおよびターゲット属性の位置による一致

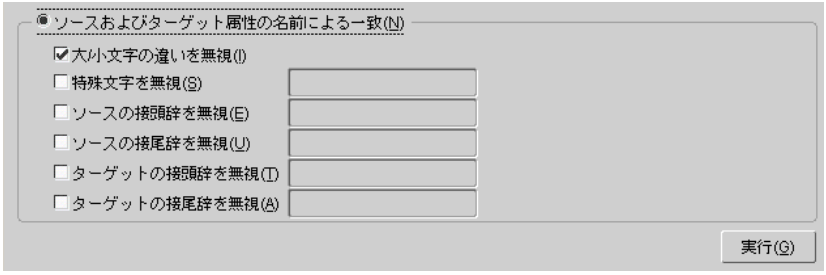
このオプションは、各グループ内の属性の位置に基づいて、既存の属性間にマッピング線を作成する場合に選択します。ターゲットのすべての属性が一致しているかぎり、ソース属性とターゲット属性の順序は一致します。ソース演算子にターゲットよりも多くの属性が含まれている場合、残りのソース属性はターゲットにマップされません。

ソースおよびターゲット属性の名前による一致

このオプションは、図 6-15 に示すように、名前が一致する既存の属性間にマッピング線を作成する場合に選択します。オプションのリストから選択することにより、正確に一致しない名前間で自動マッピングを指定できます。次のオプションを組み合せることができます。

- **大 / 小文字の違いを無視 :** 小文字と大文字を一致させることができます。たとえば、属性 FIRST_NAME と First_Name が一致します。
- **特殊文字を無視 :** 一致プロセス中に無視する文字を指定できます。このオプションの右にあるテキスト・フィールドに文字を入力します。たとえば、ハイフンとアンダースコアを指定すると、属性 FIRST_NAME、FIRST-NAME、FIRSTNAME がすべて一致します。
- **ソースの接頭辞を無視、ソースの接尾辞を無視、ターゲットの接頭辞を無視、ターゲットの接尾辞を無視 :** 一致プロセスで無視する接頭辞と接尾辞を指定できます。たとえば、「ターゲットの接尾辞を無視」にチェックマークを付けてテキスト・フィールドに USER_ を入力すると、ソース属性 USER_FIRST_NAME はターゲット属性 FIRST_NAME と一致します。

図 6-15「ソースおよびターゲット属性の名前による一致」の一致オプション



一致基準を設定したら、「実行」をクリックします。

属性を持たないソース・グループをマップしようとしたり、名前による一致を実行したときに一致が見つからない場合は、エラー・ダイアログが表示されます。

属性が 1 つでも一致すると、「表示マッピング」フィールドにそれらの一致が表示されます。マッピング線は、実装する前に確認して、選択を解除することができます。

「自動マッピング」ダイアログの「コメント」列について

「自動マッピング」ダイアログの「表示マッピング」領域には、選択した一致基準の結果を表示する「コメント」列があります。

マッピングによって属性が重複した場合は、次のメッセージが表示されます。

- **ターゲットはすでにマップされています。:** 1 つのターゲット属性は、1 つのマッピングにのみ使用できます。このターゲット属性は、すでにマッピングに使用されています。
- **ターゲットが二重にマップされています。:** 名前マッチングで、1 つのターゲット属性に対する複数の一致が見つかった場合は、1 つのマッピングでのみ作成できます。そのターゲット属性に対する最初のマッピングだけが選択され、他のマッピングは選択されません。属性を 1 つ選択すると、他の属性は選択解除されます。
- **ソースが二重にマップされています。:** 1 つのソース属性は、異なるターゲット属性にマップできます。名前マッチングで 1 つのソース属性から複数の一致が見つかった場合は、これらのマッピングをすべて作成できます。マップしないターゲット属性のチェック・ボックスをクリックすると、これらのターゲット属性を選択解除できます。

ソース・グループの属性がターゲット・グループの属性よりも多い場合、またはターゲット・グループの属性がソース・グループの属性よりも多い場合は、次のメッセージが表示されます。

- **ソースがマップされません。:** ソースに対してターゲット属性を使用できません。
- **ターゲットがマップされません。:** ターゲットに対してソース属性を使用できません。

マッピング・エディタが読取り専用モードのときに「自動マッピング」ダイアログを開くと、エラー・メッセージが表示されます。自動マッピングを使用するには、読込み / 書込み権限が必要です。読込み / 書込み権限の詳細は、2-16 ページの「マルチユーザーのサポート」を参照してください。

演算子のプロパティの設定

この項では、マッピング内の次の演算子のプロパティを設定する方法について説明します。

- ソース演算子とターゲット演算子のプロパティ（6-31 ページ）
- 属性のプロパティ（6-37 ページ）
- フラット・ファイル演算子のプロパティ（6-40 ページ）

他のタイプの演算子を設定する方法の詳細は、次の項を参照してください。

- 第 21 章「Warehouse Builder での SAP R/3 データの使用」
- 第 8 章「マッピング演算子の使用方法」

演算子、グループ、属性には、それぞれに関連付けられたプロパティ・ウィンドウがあります。プロパティ・ウィンドウを使用して、ロードの設定や、その他の条件設定を行います。表示および設定できるプロパティのタイプは、次のとおりです。

- **演算子のプロパティ :** 演算子全体に影響するプロパティ。設定できるプロパティは、演算子のタイプによって異なります。
- **グループのプロパティ :** 属性のグループに影響するプロパティ。ほとんどの演算子には、グループに関するプロパティはありません。グループのプロパティを持つ演算子には、スプリッタ演算子やデュプリケータなどがあります。グループのプロパティを設定する方法の詳細については、第 8 章「マッピング演算子の使用方法」で、これらの演算子の名前を検索してください。
- 属性のプロパティ : ソース演算子とターゲット演算子の属性に関するプロパティ。属性のプロパティには、データ型、精度、スケールなどがあります。

  属性を追加、削除、名前変更するには、演算子エディタを使用します。演算子エディタの詳細は、6-13 ページの「演算子の編集」を参照してください。

プロパティ・ウィンドウを開く手順は次のとおりです。

1. いずれかのプロパティをクリックします。

   プロパティ・ウィンドウのタイトル・バーに、選択したプロパティの名前が表示されます。

   プロパティ・ウィンドウの下部には、選択したプロパティの説明が表示されます。説明が一部しか表示されない場合は、ウィンドウを拡張してください。

2. オブジェクトに多数のプロパティが含まれる場合は、左上隅にある懐中電灯のアイコンをクリックすると、「検索」フィールドが表示され、プロパティ名を入力して検索できます。

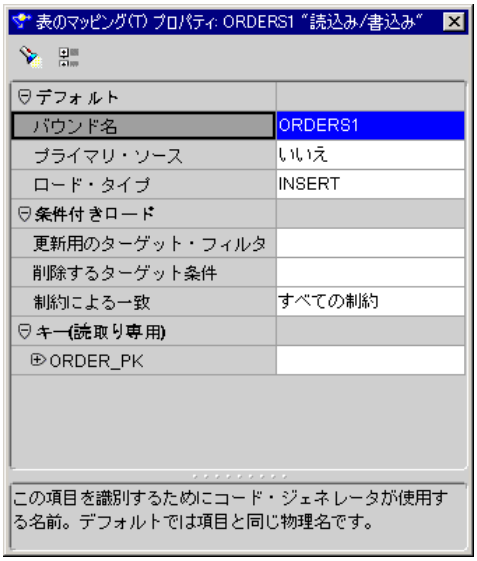
3. リストをアルファベット順にソートするには、左上隅のプラス・アイコン（+）をクリックします。

   デフォルトでは、プロパティ・ウィンドウにはカテゴリ別にプロパティが一覧表示されます。

4. マッピング・エディタで演算子を選択し、ダブルクリックするか、右クリックして「演算子のプロパティ」を選択します。

   マッピング・エディタに、選択したオブジェクトのプロパティ・ウィンドウが表示されます。図 6-16 は、表のマッピング演算子のプロパティ・ウィンドウを示しています。

図 6-16 表のマッピング演算子のプロパティ・ウィンドウ



ソース演算子とターゲット演算子のプロパティ

プロパティ・ウィンドウには、ソース演算子とターゲット演算子の次のパラメータ・カテゴリが表示されます。

- **演算子名 :** 演算子名の下には、「バウンド名」が表示されます。また、「プライマリ・ソース」インジケータと「ロード・タイプ」を設定できます。
- **条件付きロード :** 「更新用のターゲット・フィルタ」、「削除するターゲット・フィルタ」および「制約による一致」を設定できます。
- **キー（読取り専用）:** 「キー名」、「キー・タイプ」および「参照キー」を確認できます。演算子がソースとして機能する場合、キー設定は結合演算子とともに使用されます。演算子がターゲットとして機能する場合、キー設定は制約による一致パラメータとともに使用されます。

この演算子のプロパティ・ウィンドウは、図 6-17 のように表示されます。

図 6-17 表のマッピング演算子のプロパティ・ウィンドウ

バウンド名

コード・ジェネレータで使用する名前です。演算子が現在バインドされて調整されている場合、このプロパティは読取り専用になります。演算子がまだバインドされていない場合は、リポジトリ・オブジェクトに対して調整を行う前に、マッピング・エディタ内でバウンド名を編集できます。

プライマリ・ソース

Oracle Application Embedded Data Warehouse（EDW）を使用している場合は、EDW のドキュメントを参照してください。他の製品を使用している場合は、このパラメータは無視してください。

Oracle ターゲット演算子のロード・タイプ

各ターゲット演算子のロード・タイプを選択します。

- **INSERT:** 受信行セットがデータ・ターゲットに挿入されます。同じ主キーまたは一意キーに行がすでに存在している場合、挿入は失敗します。
- **UPDATE:** 受信行セットが、データ・ターゲットの既存の行の更新に使用されます。指定した一致条件の行が存在しない場合は、更新は行われません。
- **INSERT/UPDATE:** 各受信行について、最初に挿入操作が実行されます。挿入が失敗した場合、更新操作が実行されます。更新する一致レコードがない場合は、挿入が実行されます。「INSERT/UPDATE」を選択したときに、デフォルト・モードが行ベースに設定されている場合は、ターゲットに一意の制約を設定する必要があります。「INSERT/UPDATE」を選択したときに、デフォルトのオペレーティング・モードがセット・ベースに設定されていて、ターゲット・モジュールの「PL/SQL 生成モード」構成パラメータが Oracle Database に設定されている場合は、MERGE 文が生成されます。

  UPDATE/INSERT: 各受信行について、最初に更新操作が実行されます。「UPDATE/INSERT」を選択したときに、ターゲットのデフォルトのオペレーティング・モードがセット・ベースに設定されていて、ターゲット・モジュールの「PL/SQL 生成モード」構成パラメータが Oracle Database に設定されている場合は、MERGE 文が生成されます。
- **DELETE:** 受信行セットが、ターゲットで削除する行を決定するために使用されます。
- **TRUNCATE/INSERT:** 受信行セットがデータ・ターゲットに挿入される前に、データ・ターゲットが切り捨てられます。このオプションを選択すると、マッピングの実行に失敗した場合でもプロシージャはロール・バックできません。
- **DELETE/INSERT:** 受信行セットがデータ・ターゲットに挿入される前に、データ・ターゲット内の行が削除されます。
- **CHECK/INSERT:** データ・ターゲットに行が格納されているかどうかがチェックされます。行が格納されていない場合、受信行セットがデータ・ターゲットに挿入されます。
- **NONE:** データ・ターゲット上では何も操作が行われません。この設定はテスト時に有効です。抽出と変換が実行されますが、ターゲットへの影響はありません。

フラット・ファイル・ターゲットのロード・タイプ

SQL*Loader のパラメータを構成して、マッピング用の SQL*Loader オプションを定義します。構成時に選択された値は、生成された SQL*Loader の内容やランタイム制御ファイルに直接影響します。SQL*Loader では、データをロードする方法は 2 通りあります。

- **従来型パスによるロード:** SQL INSERT 文を実行して Oracle データベースに表を移入します。
- **ダイレクト・パス・ロード:** Oracle のデータ・ブロックを書式化して、データベース・ファイルにデータ・ブロックを直接書き込むことにより、Oracle データベースのオーバーヘッドを大幅に削減します。ダイレクト・ロードではデータベースのリソースについて他のユーザーと競合することがないため、通常、ディスク速度とほぼ同じ速さでデータをロードします。

  データベース・ファイルへの各アクセス方法には、制約、セキュリティおよびバックアップなどの特定の考慮事項があります。詳細は、『Oracle Database ユーティリティ』を参照してください。

  SQL*Loader を使用してフラット・ファイルからデータを抽出するマッピングを設計して実装するときは、生成された SQL*Loader スクリプトに影響する別のプロパティを構成できます。マッピング内の各ロード演算子は、「ロード・タイプ」とよばれる演算子プロパティを備えています。このプロパティによって格納されている値は、そのロード演算子に関する SQL*Loader の INTO TABLE 句の生成方法に影響を及ぼします。SQL*Loader を使用すると、データの追加、挿入、置換、切捨てができますが、処理中にデータを更新することはできません。表 6-4 は、各ロード・タイプに関連付けられた INTO TABLE 句と、既存のターゲット内のデータに与える影響を示しています。

表 6-4 ロード・タイプと INTO TABLE の関連

ロード・タイプ	INTO TABLE 句	既存データを持つターゲットへの影響
INSERT/UPDATE	APPEND	新しいデータがターゲットに追加される。
DELETE/INSERT	REPLACE	既存のデータが削除され、新しいデータに置換される（DELETE トリガーが発生）。
TRUNCATE/INSERT	TRUNCATE	既存のデータが削除され、新しいデータに置換される（DELETE トリガーが発生）。
CHECK/INSERT	INSERT	ターゲット表が空であると仮定される。
NONE	INSERT	ターゲット表が空であると仮定される。

更新用のターゲット・フィルタ

条件が true と評価された場合、その行は更新ロード操作に含まれます。

削除するターゲット・フィルタ

true と評価された場合、その行は削除ロード操作に含まれます。

制約による一致

ターゲット演算子を UPDATE 条件または DELETE 条件付きでロードする場合は、一致基準を指定できます。一致基準とロード基準は手動で設定することも、組込みのオプションから選択することもできます。「制約による一致」ダイアログでは、ターゲットの一意キーまたは主キーの情報を使用して、属性に手動で設定されている一致基準とロード基準を上書きするかどうかを指定します。「制約による一致」プロパティをクリックすると、図 6-18 に示すように、ドロップダウン・リストに、その演算子に定義されている制約と組込みのロード・オプションが表示されます。

図 6-18 演算子の「制約による一致」のオプション

「すべての制約」を選択すると、手動で設定した属性のロード設定がすべて無効になり、データは、ターゲット属性のロードおよび一致プロパティが表 6-5 の値に設定されているものとしてロードされます。

表 6-5「すべての制約」を選択した場合のターゲットのロード設定

ロード設定	キー属性	その他すべての属性
行の更新中に列をロードする	いいえ	はい
行の更新中に列を一致させる	はい	いいえ
行の削除中に列を一致させる	はい	いいえ

「制約なし」を選択すると、手動で設定したロード設定が優先され、データはその設定に従ってロードされます。

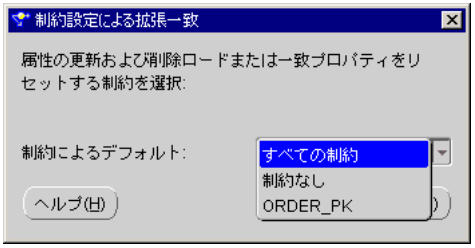
その演算子にすでに指定されている制約を選択した場合は、手動で設定した属性のロード設定がすべて無効になり、データは、ターゲット属性のロードおよび一致プロパティが表 6-7 の値に設定されているものとしてロードされます。

表 6-6 指定済の制約を選択した場合のターゲットのロード設定

ロード設定	選択済キー属性	その他すべての属性
行の更新中に列をロードする	いいえ	はい
行の更新中に列を一致させる	はい	いいえ
行の削除中に列を一致させる	はい	いいえ

属性レベルで変更を行った後で、すべての設定をデフォルト設定に戻すには、「拡張」をクリックします。図 6-19 に示すように、ドロップダウン・リストにロード・オプションが表示されます。Warehouse Builder では、選択した制約に基づいて設定がデフォルト値にリセットされます。

図 6-19「制約による一致」の詳細設定



たとえば、すべてのキー属性の一致プロパティをリセットする場合は、「拡張」をクリックしてから「制約なし」を選択し、「OK」をクリックします。手動で設定したロード設定が上書きされ、表 6-7 の設定に基づいてデータがロードされます。

表 6-7「拡張」と「制約なし」を選択した場合のデフォルトのロード設定

ロード設定	すべてのキー属性	その他すべての属性
行の更新中に列をロードする	はい	はい
行の更新中に列を一致させる	いいえ	いいえ
行の削除中に列を一致させる	いいえ	いいえ

また、「拡張」をクリックしてから「すべての制約」を選択した場合は、手動で設定したロード設定が上書きされ、表 6-8 の設定に基づいてデータがロードされます。

表 6-8「拡張」と「すべての制約」を選択した場合のデフォルトのロード設定

ロード設定	すべてのキー属性	その他すべての属性
行の更新中に列をロードする	いいえ	はい
行の更新中に列を一致させる	はい	いいえ
行の削除中に列を一致させる	はい	いいえ

キー名
主キー、外部キーまたは一意キーの名前です。

キー列
このキーを定義するローカル列です。演算子に複数のキー列が格納されている場合、各キー列はカンマで区切られます。

キー・タイプ
キーのタイプです。PRIMARY、FOREIGN、UNIQUE のいずれかになります。

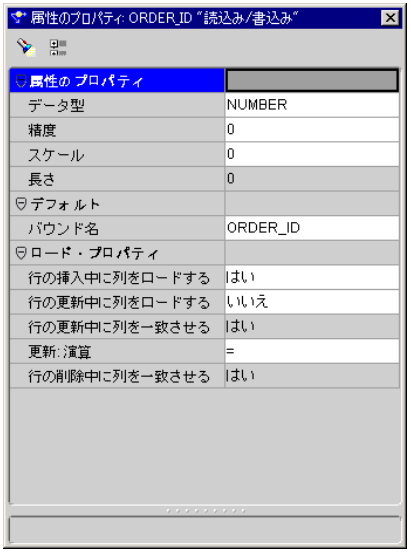
参照キー
演算子に外部キーが格納されている場合、「参照キー」には参照オブジェクトによって使用される主キーまたは一意キーが表示されます。

属性のプロパティ
ソースおよびターゲット演算子の各属性のパラメータは、次のタイプに分類されます。

- **属性のプロパティ :**「属性のプロパティ」の下には「バウンド名」が表示されます。また、「データ型」、「精度」、「スケール」および「長さ」を設定できます。
- **ロード・プロパティ :** 表のマッピング、ディメンションのマッピング、キューブのマッピング、ビューのマッピングおよびマテリアライズド・ビューのマッピングの演算子には、「ロード・プロパティ」のカテゴリがあります。このカテゴリには次の設定が含まれています。「行の挿入中に列をロードする」、「行の更新中に列をロードする」、「行の更新中に列を一致させる」、「更新 : 演算」および「行の削除中に列を一致させる」。

図 6-20 は、表のマッピング、ディメンションのマッピング、ビューのマッピングおよびマテリアライズド・ビューのマッピングに使用される属性パラメータを示しています。

図 6-20 演算子の「属性のプロパティ」ウィンドウ



バウンド名

コード・ジェネレータがこの項目を識別するために使用する名前。デフォルトでは項目名と同じです。演算子がバインドなしの場合、読取り専用になります。

データ型

属性のデータ型。

精度

この属性のデータ型が NUMBER または FLOAT である場合の最大桁数。これは読取り専用の設定です。

スケール

小数点以下の桁数。数値属性にのみ適用されます。

長さ

CHAR、VARCHAR または VARCHAR2 属性の最大長。

行の挿入中に列をロードする

この設定では、データがターゲットに移動するようにマップされた場合でも移動を禁止します。「はい」（デフォルト）を選択すると、データはマップされたターゲットに到達します。

行の更新中に列をロードする

この設定では、選択された属性データがターゲットに移動するようにマップされた場合でも移動を禁止します。「はい」（デフォルト）を選択すると、データはマップされたターゲット属性に到達します。一意キーのすべての列がマップされていない場合、一意キーは一致条件の構成に使用されません。一意キーのどの列もマップされていない場合、Warehouse Builder によってエラーが表示されます。列（キー列以外）がマップされていない場合、ロードでは使用されません。

行の更新中に列を一致させる

この設定では、ソース属性とマップされたターゲット属性が一致した場合のみデータ・ターゲット行を更新します。一致が見つかると、行が更新されます。このプロパティを「はい」（デフォルト）に設定すると、その属性は一致属性として使用されます。この設定を使用する場合、すべてのキー列をマップする必要があります。ターゲットのエンティティ上で定義された一意キーが 1 つのみである場合、制約を使用してこの設定を上書きします。

更新 : 演算

一致する行を検索してターゲットを更新するときに実行する更新の演算を指定できます。更新の演算は、ソース属性のデータを使用してターゲット属性に実行されます。表 6-9 は、指定できる更新の演算と、そのロジックを示しています。

表 6-9 更新の演算

演算	例	ソース値が 5 でターゲット値が 10 のときの結果
=	TARGET = SOURCE	TARGET = 5
+=	TARGET = SOURCE + TARGET	TARGET = 15 （5 + 10）
-=	TARGET = TARGET - SOURCE	TARGET = 5 （10 ñ 5）
=-	TARGET = SOURCE - TARGET	TARGET = - 5 （5 ñ 10）
||=	TARGET = TARGET || SOURCE	TARGET = 105 （10 に 5 を連結）
=||	TARGET = SOURCE || TARGET	TARGET = 510 （5 に 10 を連結）

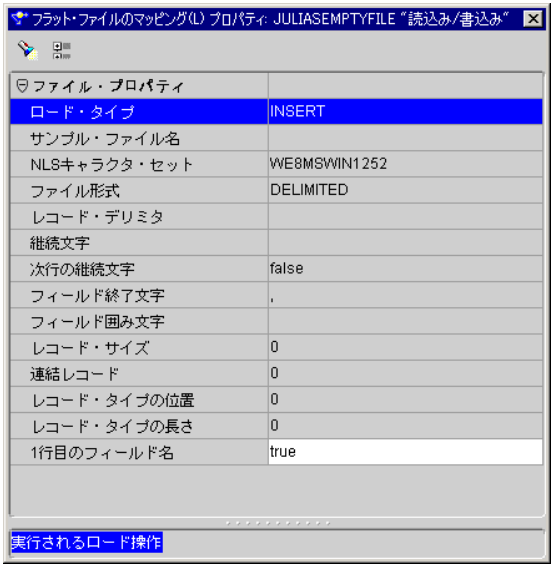
行の削除中に列を一致させる

ソース属性とマップされたターゲット属性が一致した場合のみデータ・ターゲット行を削除します。一致が見つかると、行が削除されます。このプロパティを「はい」(デフォルト)に設定すると、その属性は一致属性として使用されます。この設定は制約によって上書きできます。

フラット・ファイル演算子のプロパティ

フラット・ファイル演算子のプロパティは、ソースまたはターゲットとして設定できます。このプロパティには、「ロード・タイプ」と「1 行目のフィールド名」を設定できます。他の設定はすべて読取り専用であり、フラット・ファイルのインポート方法によって異なります。この演算子のプロパティ・ウィンドウは、図 6-21 のように表示されます。

図 6-21 フラット・ファイルのマッピング演算子のプロパティ・ウィンドウ



ロード・タイプ

ドロップダウン・リストからロード・タイプを選択します。

- **INSERT:** 新しいターゲット・ファイルを作成します。ターゲット・ファイルがすでに存在している場合、新しいターゲット・ファイルで置換されます。
- **UPDATE:** 新しいターゲット・ファイルを作成します。ターゲット・ファイルがすでに存在している場合は追加されます。
- **NONE:** データ・ターゲット上では何も操作が行われません。この設定はテスト実行時に有効です。変換と抽出はすべて実行されますが、ターゲットには影響しないためです。

1 行目のフィールド名

演算子の 1 行目にフィールド名を書き込む場合は、このプロパティを「**true**」に、そうでない場合は「**false**」に設定します。

演算子とリポジトリ・オブジェクトの調整

マッピングで定義する各演算子には、それに対応する定義が Warehouse Builder Design Repository に存在している場合があります。マッピング内の演算子に変更を加えた場合は、対応するリポジトリの定義も変更する必要があります。

演算子を調整する場合は、演算子が対応するリポジトリ・オブジェクトと一致していることを確認してください。調整の方法には、次の 2 通りがあります。

- **インバウンド調整 :** 指定したリポジトリ・オブジェクトの定義を使用して、演算子を更新または調整します。
- **アウトバウンド調整 :** マッピングの演算子に行った変更が反映されるよう、選択したリポジトリ・オブジェクトを更新します。

調整は同期とは異なります。同期とは、マルチユーザー環境において、別のユーザーが行った変更にあわせて最新状態を保つことです。調整とは、対応するリポジトリ・オブジェクトにあわせて演算子を更新することです。

インバウンド調整

インバウンド調整では、指定したリポジトリ・オブジェクトの定義を使用して、演算子が更新または調整します。また、バインドなし演算子をリポジトリ・オブジェクトにバインドするためにインバウンド調整を使用することもできます。演算子の調整は、名前、位置、オブジェクト識別子の一致、またはこれらの方法の組合せによって行われます。

インバウンド調整を行うのは次のような理由によります。

- **変更の伝播 :** インバウンド調整を使用すると、リポジトリ・オブジェクトで行った変更を対応する演算子に伝播できます。リポジトリ・オブジェクトに対する変更には、構造的な変更、属性名の変更、属性のデータ型の変更などがあります。
- **新しいリポジトリ・オブジェクトに基づく演算子の更新 :** 新しいリポジトリ・オブジェクトに演算子を関連付けることができます。たとえば、マッピングを新しいバージョンのデータ・ウェアハウスへ移行する際に、各バージョンについて異なるオブジェクト定義を維持する場合や、基礎となるソース・データへのアクセスを、実表に対して直接行うのではなく、ビューを通して行うようにする場合などです。
- **表によるマッピングのプロトタイプ作成 :** 目的の変換ロジックを得ることができたら、本番のマッピング用の他のオブジェクト・タイプ（ビュー、マテリアライズド・ビュー、キューブなど）に切り替えることができます。

属性を削除しないかぎり、インバウンド調整は、マッピングにおける他の演算子とリポジトリ・オブジェクト間の依存関係には影響しません。Warehouse Builder では、これらの依存関係が維持されます。

外部表演算子にインバウンド調整を行った場合は、対応するフラット・ファイルではなく、リポジトリの外部表のみに基づいて演算子が更新されます。対応するフラット・ファイルに基づいて外部表を更新するには、3-32 ページの「外部表の定義とファイル内のレコードとの調整」を参照してください。

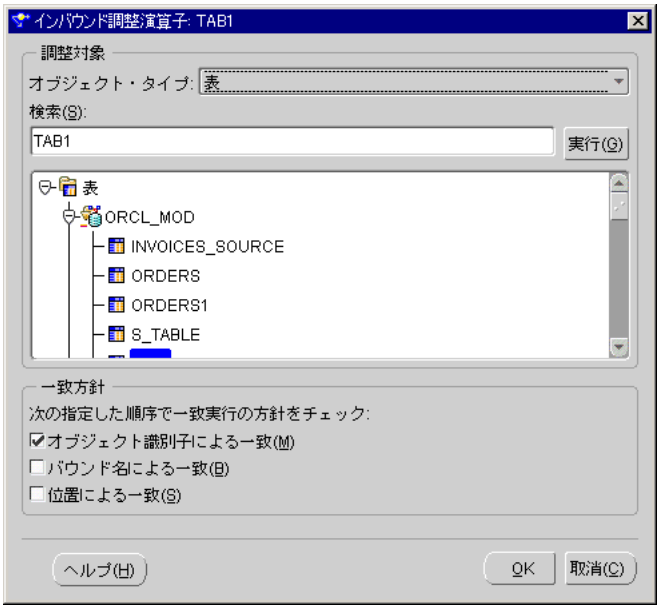
調整を行うと演算子の付加的な論理プロパティも更新されます。たとえば、フラット・ファイル演算子のマッピングでは、キャラクタ・セットとファイル名に関する情報もリフレッシュされます。

演算子にインバウンド調整を行う手順は次のとおりです。

1. マッピング・エディタのキャンバスで演算子を選択します。
2. 「編集」メニューから「インバウンドの調整」を選択するか、演算子のヘッダーを右クリックして「インバウンドの調整」を選択します。

   図 6-22 に示すように、「インバウンド調整演算子」ダイアログが表示されます。

図 6-22「インバウンド調整演算子」ダイアログ



3. 「調整対象」で、演算子の更新に使用するリポジトリ・オブジェクトのタイプを選択します。選択できるオブジェクトのタイプは、演算子のタイプによって異なります。選択できるオブジェクトの一覧は、6-44 ページの表 6-10 を参照してください。

4. 「一致方針」を設定します。

   オブジェクト識別子による一致 : この方針は、演算子をバインド済リポジトリ・オブジェクトに対する変更に一致させる場合に使用します。別のリポジトリ・オブジェクトを調整する場合、オブジェクト識別子による一致は使用できません。

   バウンド名による一致 : この方針は、演算子のバウンド名とオブジェクトの物理名を一致させる場合に使用します。演算子の構造を変える変更が存在する場合、別のリポジトリ・オブジェクトでこの方針を使用することもできます。

   位置による一致 : 演算子属性のビジネス名を維持する場合、この方針を使用して別のリポジトリ・オブジェクトとの調整を行います。この方針は、リポジトリ・オブジェクトに対する変更が、オブジェクトの末尾に列、フィールドまたはパラメータを追加しただけの場合に最も効果的です。

   詳細は、6-47 ページの「一致方針」を参照してください。

5. 「OK」をクリックします。

リポジトリ・オブジェクトに基づく演算子の調整

表 6-10 は、インバウンド調整で演算子を更新する場合に使用できるリポジトリ・オブジェクトの一覧です。

表 6-10 リポジトリ・オブジェクトに基づいて調整される演算子

演算子オブジェクト	調整に使用できるリポジトリ・オブジェクト
表のマッピング	表、外部表、ビュー、マテリアライズド・ビュー、フラット・ファイル、ディメンションおよびキューブ
外部表のマッピング	表、外部表、ビュー、マテリアライズド・ビュー、フラット・ファイル、ディメンションおよびキューブ
ビューのマッピング	表、外部表、ビュー、マテリアライズド・ビュー、ファイル、ディメンションおよびキューブ
マテリアライズド・ビューのマッピング	表、外部表、ビュー、マテリアライズド・ビュー、ファイル、ディメンションおよびキューブ
順序のマッピング	順序のみ
フラット・ファイルのマッピング	表、外部表、ビュー、マテリアライズド・ビュー、フラット・ファイル、ディメンションおよびキューブ
ディメンションのマッピング	表、外部表、ビュー、マテリアライズド・ビュー、フラット・ファイル、ディメンションおよびキューブ
キューブのマッピング	表、ビュー、マテリアライズド・ビュー、フラット・ファイル、ディメンションおよびキューブ
キー検索演算子	表のみ
マッピング前プロセス演算子	変換のみ
マッピング後プロセス演算子	変換のみ
変換のマッピング演算子	変換のみ

アウトバウンド調整

アウトバウンド調整では、演算子の変更が反映されるように、選択したリポジトリ・オブジェクトを更新します。アウトバウンド調整は、表、ビュー、マテリアライズド・ビュー、変換およびフラット・ファイルの各演算子に対して実行できます。アウトバウンド調整を行うのは次のような理由によります。

- **新しいリポジトリ・オブジェクトの作成 :** アウトバウンド調整を使用すると、ターゲット・モジュールに新しいリポジトリ・オブジェクトを作成できます。新しいリポジトリ・オブジェクトは、既存のものと同じタイプであることが必要です。たとえば、アウトバウンド調整を使用する場合、既存の表に基づいて新しい表を作成することはできますが、新しいビューを作成することはできません。
- **変更の伝播 :** アウトバウンド調整を使用すると、演算子に行った変更を対応するリポジトリ・オブジェクトに伝播できます。演算子または属性の論理名を変更した場合は、論理名の最初の 30 文字がバウンド名として伝播されます。
- **リポジトリ・オブジェクトの置換 :** アウトバウンド調整を使用すると、既存のリポジトリ・オブジェクトを置換できます。

アウトバウンド調整は、マッピングにおける他の演算子とリポジトリ・オブジェクト間の依存関係に影響しません。表 6-11 は、アウトバウンド調整に適した演算子の一覧です。

表 6-11 アウトバウンド調整の演算子

マッピング・オブジェクト	リポジトリ・オブジェクトの作成	変更の伝播	リポジトリ・オブジェクトの置換	注意
外部表	可能	可能	可能	リポジトリの外部表のみが更新され、その外部表に対応するフラット・ファイルは更新されません。3-32 ページの「外部表の定義とファイル内のレコードとの調整」を参照してください。
フラット・ファイル	可能	可能	×	シングル・レコード・タイプのフラット・ファイルに対してのみ、カンマで区切られたデリミタ付きの新しいフラット・ファイルが作成されます。既存のファイルの置換はできません。
入力パラメータのマッピング	可能	可能	可能	入力属性とデータ型が入力パラメータとしてコピーされます。
出力パラメータのマッピング	可能	可能	可能	出力属性とデータ型がファンクションの戻り指定としてコピーされます。
マテリアライズド・ビュー	可能	可能	可能	属性とデータ型が列としてコピーされます。

表 6-11 アウトバウンド調整の演算子（続き）

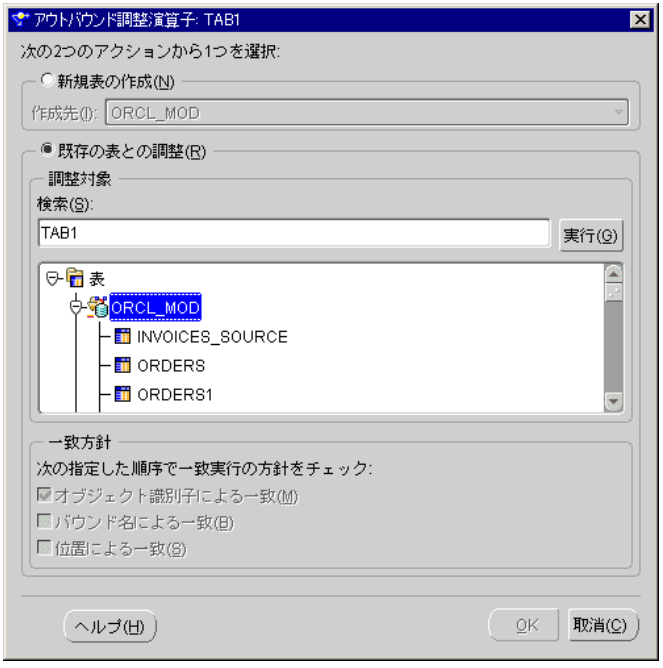
マッピング・オブジェクト	リポジトリ・オブジェクトの作成	変更の伝播	リポジトリ・オブジェクトの置換	注意
表	可能	可能	可能	属性とデータ型が列としてコピーされます。制約プロパティはコピーされません。
変換	可能	可能	可能	
ビュー	可能	可能	可能	属性とデータ型が列としてコピーされます。

既存の演算子にアウトバウンド調整を行う手順は次のとおりです。

1. キャンバス上で演算子を選択します。
2. 「編集」メニューから「アウトバウンドの調整」を選択するか、演算子のヘッダーを右クリックして「アウトバウンドの調整」を選択します。

   図 6-23 に示すように、「アウトバウンド調整演算子」ダイアログが表示されます。

図 6-23「アウトバウンド調整演算子」ダイアログ（表演算子の場合）



3. 「新規 < オブジェクト・タイプ > の作成」または「既存の < オブジェクト・タイプ > との調整」を選択します。

   「新規 < オブジェクト・タイプ > の作成」を選択した場合は、指定したターゲット・モジュールにオブジェクトが作成されます。このオプションは、データ・ウェアハウスのステージング表をすばやく作成する場合に効果的です。アウトバウンド調整と「自動マッピング」ダイアログを使用してステージング表を作成する方法は、6-23 ページの「例 : マッピング・エディタによるステージング領域表の作成」を参照してください。

   「既存の < オブジェクト・タイプ > との調整」を選択した場合は、オブジェクトを更新または置換するためのオプションを選択できます。

   リポジトリ・オブジェクトを更新するには、「調整対象」リストから対応する演算子を選択します。たとえば、S_TABLE リポジトリ・オブジェクトを更新する場合は、S_TABLE 演算子を選択します。演算子からリポジトリ・オブジェクトに伝播される変更のタイプには、演算子のビジネス名と物理名の変更、属性のデータ型の変更などがあります。

   リポジトリ・オブジェクトを置換するには、「調整対象」リストから別の演算子を選択します。たとえば、S_TABLE1 リポジトリ・オブジェクトを置換する場合は、S_TABLE2 演算子を選択します。置換できるリポジトリ・オブジェクトは、同じタイプのものだけです。たとえば、アウトバウンド調整で TABLE_A を TABLE_B に置換することはできても、VIEW_B に置換することはできません。

4. 「一致方針」を設定します。詳細は、6-47 ページの「一致方針」を参照してください。

5. 「OK」をクリックします。

一致方針

マッピングのオブジェクトを調整する場合は、次の方針を指定できます。

- オブジェクト識別子による一致
- バウンド名による一致
- 位置による一致

複数の調整方針を指定することもできます。複数の方針を選択した場合、Warehouse Builder では、次の順序で一致処理が行われます。

1. オブジェクト識別子による一致
2. 位置による一致
3. 物理（バウンド）名による一致

オブジェクト識別子による一致

演算子をリポジトリ・オブジェクトへの変更と一致させる場合や、演算子の属性に対して、リポジトリ・オブジェクト内で行われた物理名の変更とは関係なく個別のビジネス名を維持する場合に、この方針を使用します。タイプの異なるリポジトリ・オブジェクトを調整する場合、オブジェクト識別子による一致は使用できません。

この方針では、一意のオブジェクト識別子を使用して、演算子属性と選択したリポジトリ・オブジェクト属性の相関関係を決定します。リポジトリ・オブジェクト内の属性に対応しない属性は、演算子から削除されます。これは、演算子に属性が追加されたが調整されない場合や、調整の前にリポジトリ・オブジェクトから列が削除された場合などに該当します。属性が削除された場合、その属性に接続されたマッピング線もキャンバスから削除されます。選択したリポジトリ・オブジェクトの属性で、演算子の属性と一致できないものは、演算子の末尾に新しい属性として追加されます。一致した属性のマッピング線は保持されます。

バウンド名による一致

この方針は、演算子内のバウンド名をオブジェクトの物理名と一致させる場合に使用します。リポジトリ・オブジェクト列の名前を変更すると、その列は削除されて新しい列が挿入されたと解釈されます。名前が変更された属性のマッピング線は削除されます。リポジトリ・オブジェクトに演算子の構造を変える変更が存在する場合、別のリポジトリ・オブジェクトでこの方針を使用することもできます。

この方針では、演算子属性のバウンド名とリポジトリ・オブジェクト属性の物理名との一致を使用します。一致については大文字と小文字が区別されます。インバウンド調整では、リポジトリ・オブジェクトの属性と一致できない演算子の属性は削除されます。属性が削除された場合、その属性に接続されたマッピング線もキャンバスから削除されます。選択したリポジトリ・オブジェクトの属性で、演算子の属性と一致できないものは、演算子に新しい属性として追加されます。一致した属性のマッピング線は保持されます。演算子をリポジトリ・オブジェクトにバインドするとバウンド名は読取り専用になるため、インバウンド調整ではバウンド名を操作して別の一致結果を得ることはできません。

位置による一致

この方針では、演算子属性と選択したリポジトリ・オブジェクトの列、フィールドまたはパラメータを位置によって一致させます。つまり、演算子の最初の属性はリポジトリ・オブジェクトの最初の属性によって調整され、2 番目の属性はリポジトリ・オブジェクトの 2 番目の属性によって調整されます。演算子の属性の方がリポジトリ・オブジェクトの属性よりも多い場合、余分な属性は演算子から削除されます。属性が削除された場合、その属性に接続されたマッピング線もキャンバスから削除されます。リポジトリ・オブジェクトの属性の方が演算子の属性よりも多い場合、余分な属性は演算子の末尾に新しい属性として追加されます。演算子における既存の属性のマッピング線は保持されます。演算子属性のビジネス名を維持する場合、この方針を使用して別のリポジトリ・オブジェクトとの調整を行います。この方針は、リポジトリ・オブジェクトに対する変更が、オブジェクトの末尾に列、フィールドまたはパラメータを追加しただけの場合に最も効果的です。

7

マッピングのデバッグ

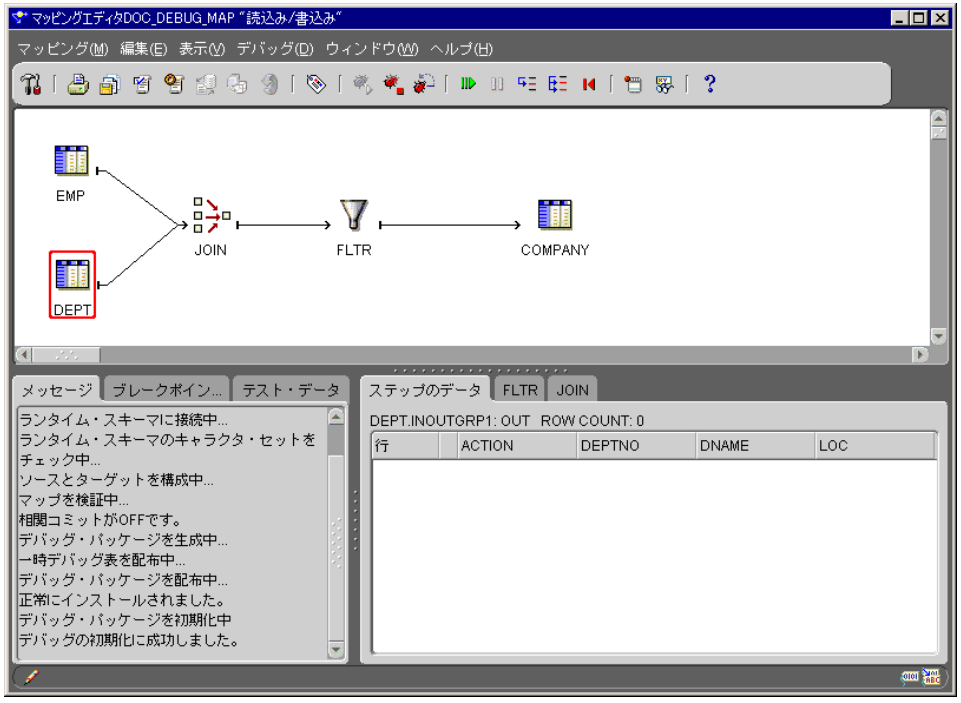
Warehouse Builder には、マッピング・エディタでデータ・フローをテストするデバッグ機能が用意されています。これらのデバッグ機能は、マッピング・エディタのツールバーとデバッグ・メニューから使用できますが、実行するには、デバッグ・セッションを開始して、有効なターゲット・スキーマに接続する必要があります。その後、定義済の一連のテスト・データを使用してデバッグ・モードでマッピングを実行できます。データの抽出から、変換、ロードまでのフローに従って、設計したデータ・フローが期待どおりに動作するかどうかを確認します。問題が見つかった場合は、それらを修正し、配布する前に再びデバッグ・セッションを開始して、問題が修正されたことを確認できます。

次の各トピックでは、デバッグ・プロセスと、マッピング・エディタでのデバッグの使用方法について説明します。

マッピング・エディタのデバッグ・モードについて

デバッグ・セッションを起動すると、マッピング・エディタ内に、デバッグ・セッションの詳細を表示する新しいパネルが表示されます。デバッグ・セッションを開始すると、ツールバーのデバッグ用ボタンと「デバッグ」メニューが有効になります。さらに、マッピング・エディタの下部にあるステータス・バーには、デバッグ・セッションのステータスが表示されます。

図 7-1 デバッグ・モードのマッピング・エディタ



左下のパネルには、次のタブがあります。

- **メッセージ:** デバッガのすべての処理メッセージが表示されます。デバッグ・セッションのステータスは、これらのメッセージで確認できます。メッセージには、デバッグ・モードでマッピングを実行しているときに生成されるエラー・メッセージも含まれます。
- **ブレークポイント:** マッピングに設定したすべてのブレーク・ポイントの一覧が表示されます。チェック・ボックスを使用して、ブレーク・ポイントのアクティブ化と非アクティブ化を切り替えられます。
- **テスト・データ:** マッピングに使用されるすべてのデータ・オブジェクトの一覧が表示されます。この一覧により、テスト・データが定義されているデータ・オブジェクトも確認できます。

右下のパネルには、「ステップのデータ」タブと、デバッグされる演算子の入出力情報を示すウォッチ・ポイント・タブが表示されます。「ステップのデータ」タブには、デバッグ・セッションの現在のステップに関する情報が表示されます。ウォッチ・タブは、設定するウォッチごとに追加できます。これらのウォッチ・タブを使用すると、デバッグ・セッションで現在アクティブな演算子に関係なく、演算子を介して渡される（または渡された）データを追跡および確認できます。複数の入力グループまたは出力グループを持つ演算子には、グループを指定するためのドロップダウン・リストが表示されます。

図 7-2 に示すツールバーのデバッグ・ボタンと「デバッグ」メニュー・オプションは、デバッグ・セッションを開始した後にのみ有効になります。表 7-1 に、各ボタンとその機能を示します。

図 7-2 マッピング・エディタのデバッグ・ツール

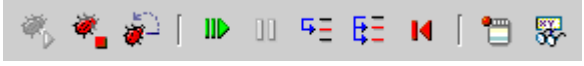


表 7-1 マッピング・エディタのデバッグ・ツール

アイコン	アクション	説明
	デバッグの開始	デバッグ・セッションを開始します。
	デバッグの終了	デバッグ・セッションを終了します。
	再初期化	デバッグ・セッションを再初期化します。再初期化すると、デバッグ・コードが再配布され、デバッグ・セッションがリセットされます。
	再開	デバッグ・セッションを再開します。デバッグ・セッションが次のブレーク・ポイントに達するまで実行されます。ブレーク・ポイントが設定されていない場合は、マッピングのデバッグが終了します。
	一時停止	デバッグ中のマッピングを一時停止します。この機能は、現在のリリースでは使用できません。
	ステップ	1 つの演算子を 1 行ずつデバッグします。
	スキップ	1 つの演算子をスキップしながらデバッグします。指定した演算子の行はすべて処理されます。
	リセット	デバッグ・セッションをリセットし、デバッグを先頭から開始します。
	ブレークポイントの設定	選択した演算子にブレーク・ポイントを設定します。その演算子にすでにブレーク・ポイントが設定されている場合は、このボタンをクリックするとそのブレーク・ポイントが削除されます。
	ウォッチポイントの設定	選択した演算子にウォッチを設定します。その演算子にすでにウォッチ・ポイントが設定されている場合は、このボタンをクリックするとそのウォッチ・ポイントが削除されます。

デバッグ・セッションの開始

デバッグ・セッションを開始する手順は次のとおりです。

1. マッピング・エディタで、「デバッグ」の「開始」を選択します。ツールバーの「デバッグの開始」ボタンをクリックして開始することもできます。

   マッピング・エディタがデバッグ・モードに切り替わってエディタの下部にデバッグ・パネルが追加され、「接続情報」ダイアログが表示されます。

2. デバッグの実行に使用するターゲット・スキーマの接続情報を指定します。

   生成されたコードは、ここで指定するスキーマに配布されます。デバッグ・セッション中に別のターゲット・スキーマに接続するには、「デバッグ」メニューの「再接続」を選択します。

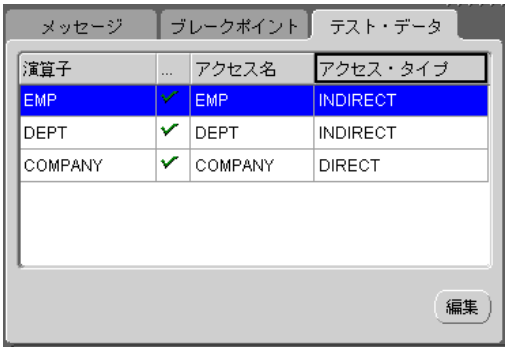
接続が確立されると、テスト・データを定義するよう指示するメッセージが表示されます。テスト・データがすでに定義されている場合は、初期化を行うかどうか確認するメッセージが表示されます。マッピングをデバッグするには、各ソース演算子またはターゲット演算子をデータベース・オブジェクトにバインドし、そのデータベース・オブジェクトに対してテスト・データを定義する必要があります。デフォルトでは、マッピング内の演算子は、一致したオブジェクト名に基づいて、ローカル・ソースおよびターゲットに接続されます。

テスト・データの定義

マッピング内のソース演算子またはターゲット演算子はすべて、左下のパネルの「テスト・データ」タブに一覧表示されます。その他、このタブには、オブジェクト・タイプ、ソース、およびデータベース・オブジェクトがすでにソース演算子またはターゲット演算子にバインドされていることを示すチェックマークが表示されます。タブに一覧表示されるオブジェクト・タイプは、選択したデータ・ソース内の列が、マッピング演算子内の列と一致するかどうかによって決まります。表示されるタイプには、「DIRECT」と「INDIRECT」があります。完全に一致する列があれば、タイプには「DIRECT」と表示されます。マッピング演算子の列と一致しない列が含まれるデータ・ソースを選択する場合は、列のマップ方法を選択できます。この場合、オブジェクトはマッピングの実行時にビューとして配布されるため、タイプには「INDIRECT」と表示されます。

バインド済データベース・オブジェクトで、テスト・データだけでなく、演算子のバインドを追加または変更するには、「編集」ボタンを使用します。デバッグ・モードでマッピングを実行するには、一覧されている各ソース演算子またはターゲット演算子がバインドされ、チェックマークが付けられている必要があります。バインド済データベース・オブジェクトにテスト・データを定義して有効にする必要があるかどうかは、デバッグ・セッションを実行する際に、データ・フローのどの部分に焦点を置くかによって決まります。一般に、すべてのソース演算子には、テスト・データが必要です。ターゲット演算子のテスト・データは、通常、更新またはターゲットの制約を伴うロード・シナリオをデバッグする場合に必要です。

図 7-3「テスト・データ」タブ



テスト・データを定義または編集する手順は次のとおりです。

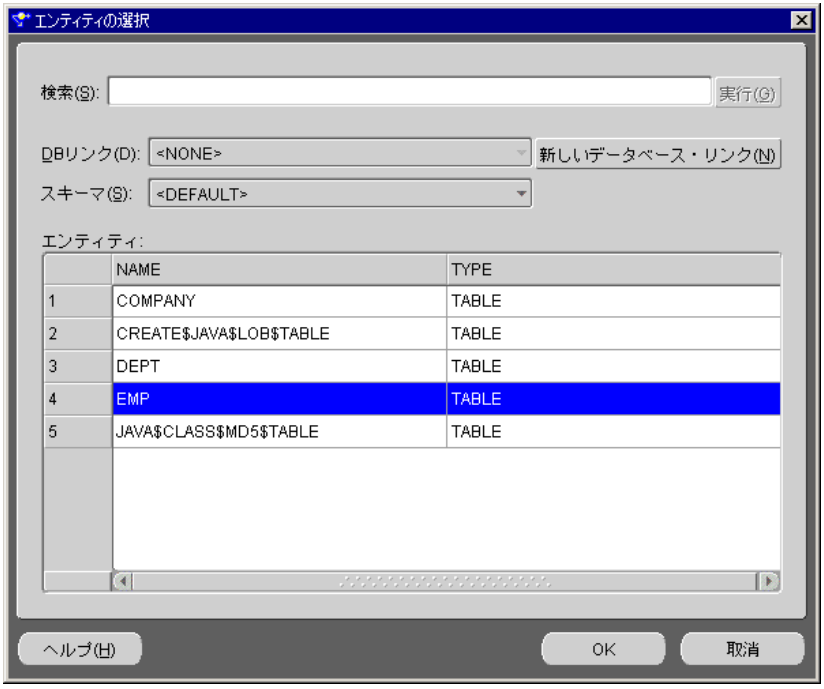
1. マッピング・エディタの「テスト・データ」タブで、リストから演算子を選択し、「編集」ボタンをクリックします。

   図 7-5 に示すように、「テスト・データの定義」ダイアログが表示されます。

2. オブジェクトの「アクセス名」を指定します。

   既存のデータベース・オブジェクトをバインドするには、そのオブジェクトの名前を「アクセス名」フィールドに入力するか、「参照」ボタンをクリックして、使用する既存のデータベースを検索します。図 7-4 に示す「エンティティの選択」ダイアログが表示されます。このダイアログで、データベース・リンクとスキーマを作成または選択します。その後、「エンティティ」リストからデータベース・オブジェクトを選択し、「OK」をクリックします。

図 7-4「エンティティの選択」ダイアログ



新しい表の作成とバインド、またはテスト・データのスプレッドシートに表示される既存のテスト・データ・セットのコピーを行うには、「テスト・データの定義」ダイアログの「新規表の作成」ボタンをクリックします。詳細は、7-8 ページの「新しい表の作成」を参照してください。

3. データを編集するには、「ユーザー・データの編集モード」チェック・ボックスを選択します。

   これにより、バインド済データベース・オブジェクトに適切なデータが存在しない特定のシナリオをデバッグするために必要なテスト・データを作成できます。テスト・データ・パネルの各セルをクリックすると、その中の値を編集できます。

4. テスト・データとして選択できる行の数を制限するには、「行カウント」チェック・ボックスを選択して、行数を設定します。デフォルトでは 100 行に設定されています。

   これにより、デバッグ・セッションでのデータの量を制限できます。

図 7-5「テスト・データの定義」ダイアログ

新しい表の作成

「テスト・データの定義」ダイアログの「新規表の作成」ボタンを使用して新しい表を作成すると、その表には接頭辞 DBG_ とデータ演算子名を組み合せた名前が付けられます。この名前がすでにターゲットに存在する場合は、一意のオブジェクト名となるよう、名前の末尾に順序番号が付けられます。表は、デバッグを開始したときに指定したターゲット・スキーマに作成されます。デバッガによって表が自動的に削除されることはないので、他のセッションでも同じ表を再利用できます。制約は新しい表には継承されません。

テスト・データは、「テスト・データの定義」ダイアログでいつでも編集できます。演算子のバインドを別のデータベース・オブジェクトに変更した場合は、デバッグ・セッションを再初期化して変更を実装した上で、デバッグ・モードでのマッピングを再実行する必要があります。

注意： ターゲットの定義にロードされたデータは、暗黙的にコミットされます。ターゲット・オブジェクトを更新しないようにするには、「新規表の作成」ボタンを使用して、ターゲット・オブジェクトのコピーを作成する必要があります。

ブレーク・ポイントの設定

特定の演算子のデータ処理を確認したい場合は、その演算子に、デバッグ・セッションを中断させるブレーク・ポイントを設定します。これにより、データ・フロー内のすべての演算子を順番に処理しなくても、目的の演算子にすばやく移動できます。デバッグ・セッションがブレーク・ポイントに到達したら、目的の演算子で順を追ってデータを実行し、期待どおりに機能するかどうかを確認します。ブレーク・ポイントの設定は、マッピングの終了時に保存されません。

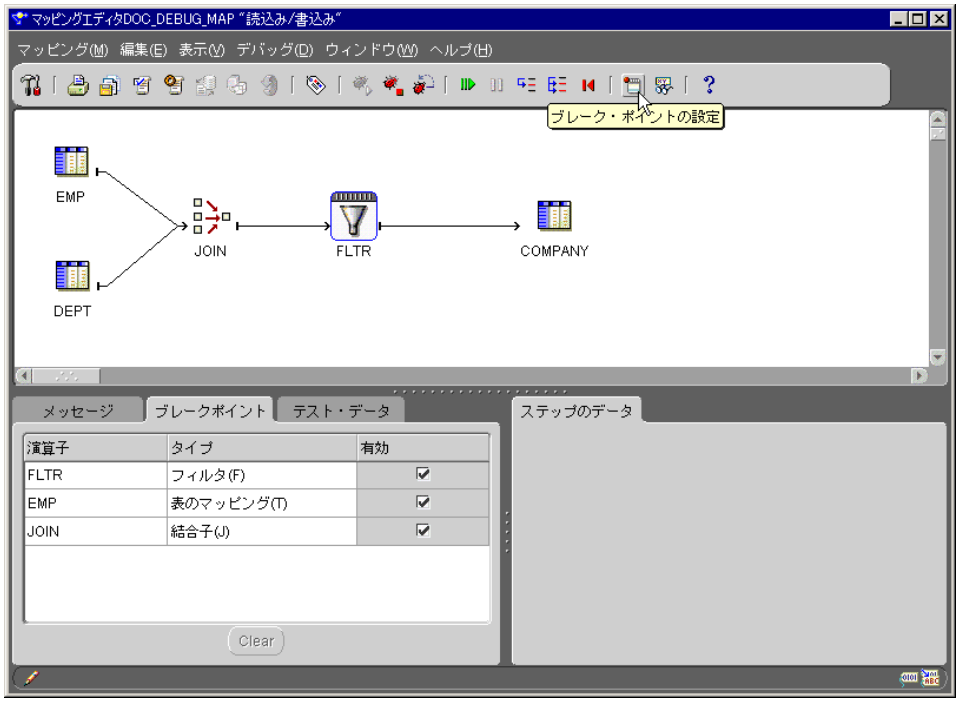
ブレーク・ポイントを設定または削除する手順は次のとおりです。

1. マッピング・エディタで演算子をクリックし、「デバッグ」メニューの「ブレーク・ポイントの設定」を選択します。または、ツールバーの「ブレーク・ポイントの設定」ボタンをクリックして、現在選択している演算子のブレーク・ポイントのオン / オフを切り替えることもできます。

   ブレーク・ポイントを設定すると、ブレーク・ポイントとして設定された演算子の名前が、左下のパネルの「ブレーク・ポイント」タブに一覧表示されます。ブレーク・ポイントを削除すると、その演算子の名前も削除されます。ブレーク・ポイントを削除するには、「ブレーク・ポイント」タブの「消去」ボタンを使用します。

2. ブレーク・ポイントの有効 / 無効を切り替えるには、「ブレーク・ポイント」タブの「有効」を選択または選択解除します。

図 7-6 デバッグ・モードのブレーク・ポイント



ウォッチの設定

右下のパネルにある「ステップのデータ」タブには、常に現在の演算子のデータが表示されます。アクティブな演算子に関係なく、他の演算子を通過したデータを追跡するには、ウォッチを設定します。

ウォッチを使用すると、演算子を通過したデータや、ソースやターゲットの場合は、バインド済データベース・オブジェクトに現在存在しているデータの追跡が可能になります。デバッグが目的の演算子ですでに実行されていても、演算子にウォッチ・ポイントを設定すれば、データ・フローをさかのぼって、データがその演算子によってどのように処理されたかを確認できます。

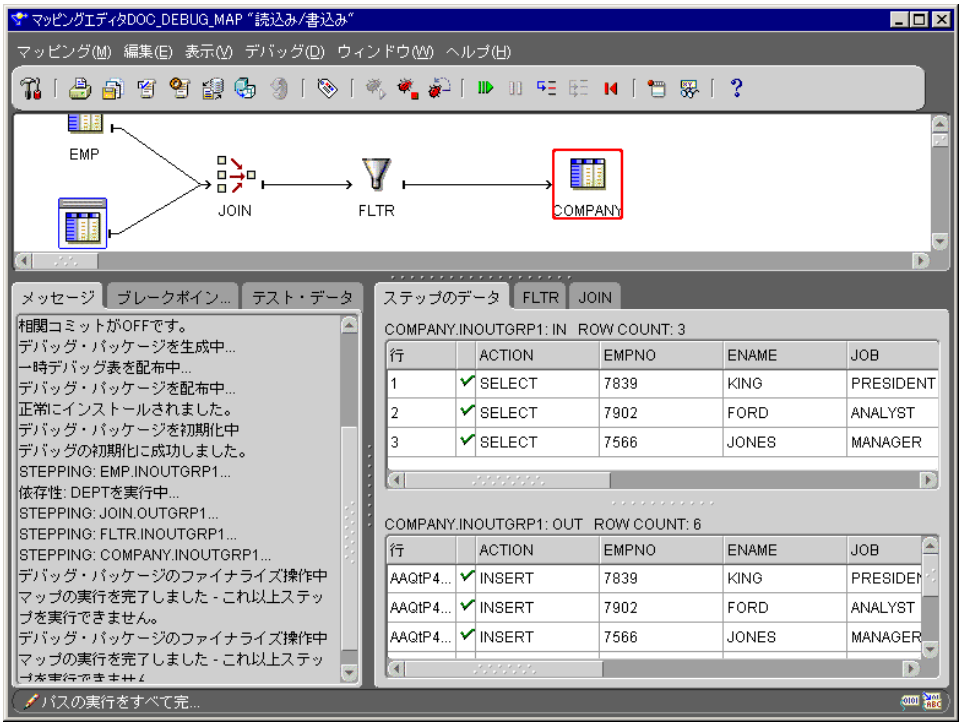
ウォッチを設定する手順は次のとおりです。

1. マッピング・エディタで演算子をクリックし、「デバッグ」メニューの「ウォッチの設定」を選択します。または、ツールバーの「ウォッチの設定」ボタンをクリックして、ウォッチのオン / オフを切り替えることもできます。

   図 7-7 に示すように、右下のパネルの下部に、選択した演算子の名前が新しいタブとして現れ、その演算子の入出力グループが表示されます。

2. ウォッチを削除するには、再び演算子を選択し、ツールバーのウォッチ・ボタンをクリックするか、「デバッグ」メニューの「ウォッチの設定」オプションを選択します。または、マウスを右クリックして、「デバッグ・ウォッチの切替え」を選択します。

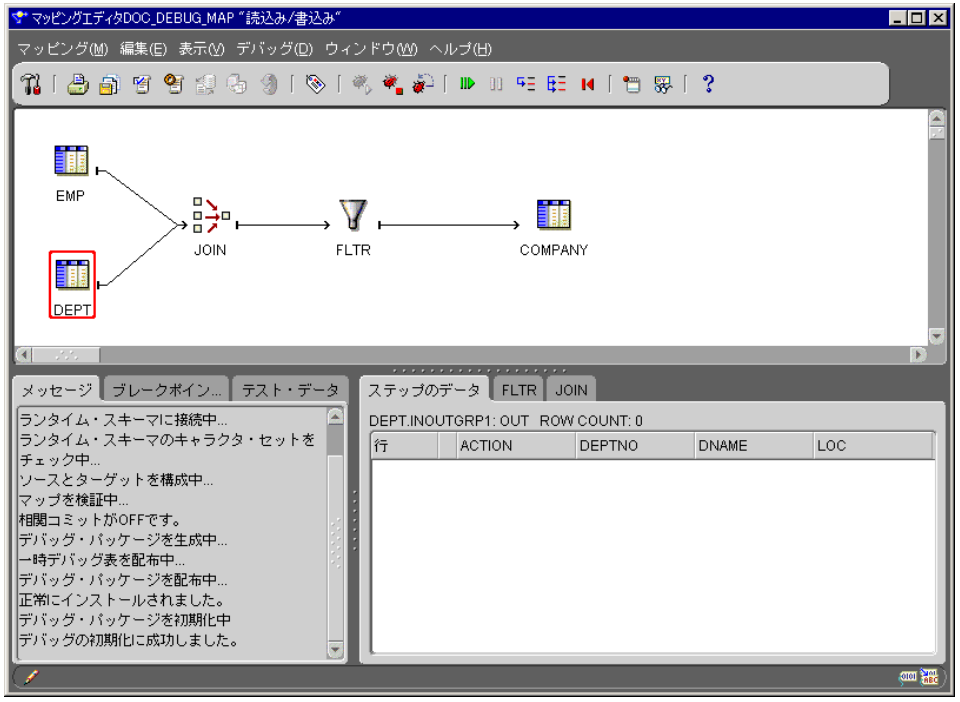
図 7-7 右下のパネルに個別のタブとして表示されたウォッチ



マッピングの実行

各データ演算子に対するテスト・データの接続を定義したら、「デバッグ」メニューの「再初期化」を選択するか、またはツールバーの「再初期化」ボタンをクリックして、初期デバッグ・コードを生成できます。Warehouse Builder でデバッグ・コードが生成され、指定したターゲット・スキーマにパッケージが配布されます。図 7-8 は、デバッグ用に初期化されたマッピングを示しています。

図 7-8 デバッグ用に初期化されたマッピング



デバッグ・セッションは、次のいずれかのモードで実行されるよう選択できます。

- ツールバーの「再開」ボタンまたは対応するメニュー項目を使用して、次のブレーク・ポイントまで、あるいはデバッグの実行が終了されるまで処理を続行する。
- ツールバーの「ステップ」ボタンまたは対応するメニュー項目を使用して、1 行ずつ処理する。
- ツールバーの「スキップ」ボタンまたは対応するメニュー項目を使用して、現在の演算子の残りの行をすべて処理する。
- 「リセット」ボタンまたは「デバッグ」メニューの対応する項目を使用して、デバッグの実行をリセットし、最初に戻る。

優先するソースとパスの選択

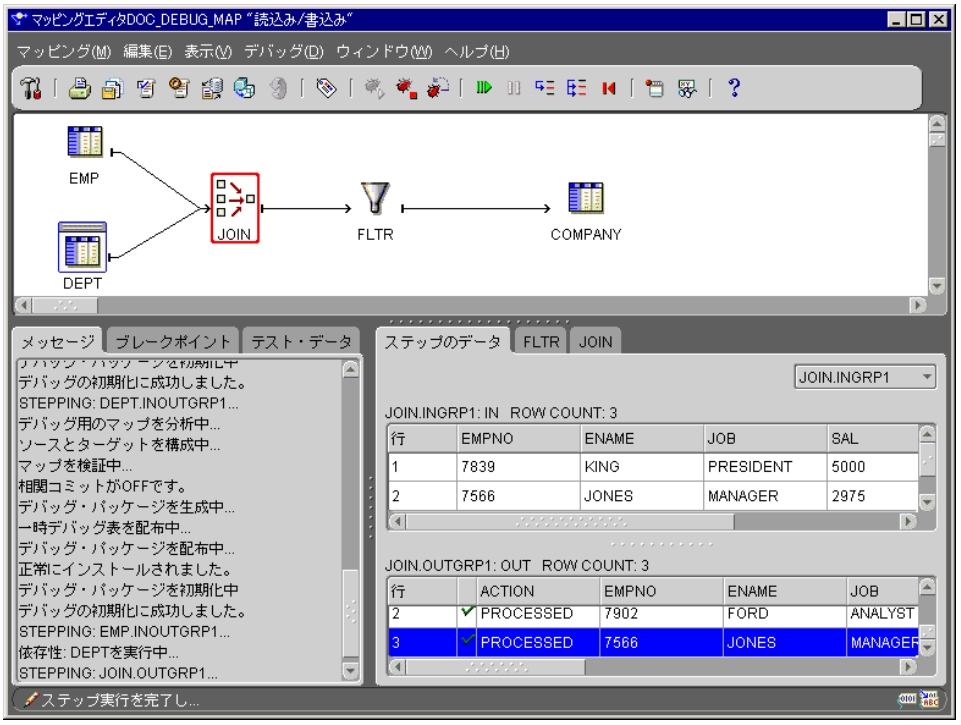
マッピングには、デバッグするソースとパスが複数含まれる場合があります。マッピングに複数のソースが含まれる場合は、最初にデバッグを開始するソースを指定するよう要求されます。たとえば、2 つの表が 1 つの結合子にマップされている場合は、優先してデバッグするソース表を選択する必要があります。

パスの場合も、デバッガが 1 つのパスを終了した後で通過できる複数のパスが存在することがあります。たとえば、スプリッタを使用する場合などです。このような場合は、1 つのパスが終了すると、他のパスも処理するかどうかを尋ねるメッセージが表示されます。

マッピングは、すべてのターゲット演算子が処理されるか、または発生したエラー数がそのマッピングに設定されている最大数に達すると終了します。デバッグ接続とテスト・データの定義は、Warehouse Builder Design Repository に変更をコミットする際に保存されます。ブレーク・ポイントとウォッチの設定は保存されないため、マッピングを開くたびに再設定する必要があります。

デバッガの実行に伴って、適宜デバッグ・メッセージが表示されます。演算子を介するデータ・フローに従うことができます。アクティブな演算子は、図 7-9 に示すように、赤のボックスで囲まれます。

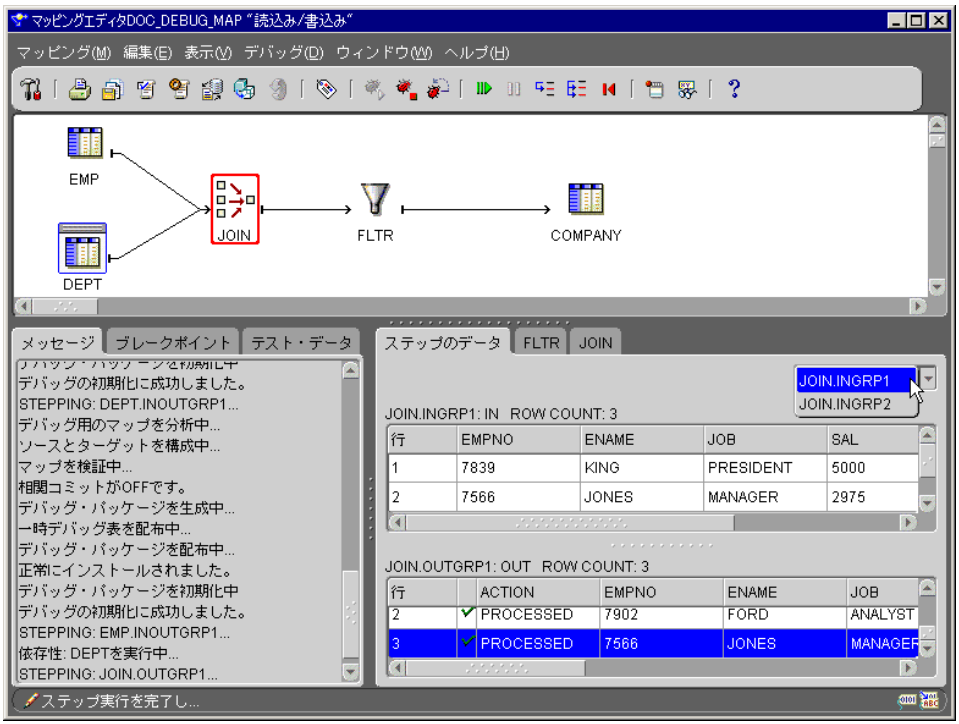
図 7-9 マッピング・デバッガ



「ステップのデータ」パネル

1 つの演算子に複数の入出力グループが含まれる場合は、右上隅にドロップダウン・リストが表示されます。このドロップダウン・リストを使用して、目的のグループを選択します。これは「ステップのデータ」タブとウォッチ・タブの両方に共通しています。図 7-10 は、結合演算子の例を示しています。

図 7-10 結合演算子に含まれる複数の入力グループ



「相関コミット」を使用したマッピングのデバッグ

「相関コミット」パラメータが「true」に設定されているマッピングのデバッグ・セッションを開始すると、マッピングのデバッグにはパスが使用されません。「相関コミット」が「false」に設定されている場合は、1 回に 1 つずつパスが使用されます。この場合、残りのすべてのパスを実行し、他のターゲットをロードするには、1 つのパスが終了した後でステップを逆戻りし、マッピング内の別のパスを実行する必要があります。しかし、「相関コミット」を「true」に設定すると、指定したパスに関係なく、マッピングの最初のステップですべてのパスが実行され、ターゲットもすべてロードされます。また、いずれかのターゲットにステップの制約違反があった場合、そのステップではどのターゲットもロードされません。

たとえば、S1 というソースを持つマッピングがあり、T1、T2 という 2 つのターゲットに分かれるスプリッタに接続しているとします。

「相関コミット」を「false」に設定すると、マッピングは S1 からデバッグされます。その後、T1 に進むパスまたは T2 に進むパスを選択できます。T1 に進むパスを選択すると、T1 に渡されるデータが処理および表示されて、ターゲット T1 がロードされます。T1 のロードが完了すると、最初に戻り、もう 1 つのパスを実行してターゲット T2 をロードするかどうかを選択できます。

「相関コミット」を「true」に設定した場合も、マッピングは S1 からデバッグされ、パスを選択するオプションが表示されます。ただし、この選択は、データの処理に伴ってマッピング・エディタに表示するパスを決めるためだけのもので、パスはすべて同時に実行されます。「相関コミット」を使用したマッピングは、配布可能なコードを実行すると、このように実行されます。

デバッグ・セッションの再初期化

マッピングに変更を加えたり、ソースまたはターゲット演算子を別のデータベース・オブジェクトにバインドした場合、新しい設定でマッピングのデバッグを続行するには、デバッグ・セッションを再初期化する必要があります。デバッグ・コードの再生成と再配布には、ツールバーの「再初期化」ボタンまたは「デバッグ」メニューの「再初期化」を使用します。マッピングのデバッグ・セッションは、再初期化の後、先頭から開始されます。

スケーラビリティ

マッピングをデバッグする際、「ステップのデータ」パネルに表示される列数と、渡されるデータ量の両方にはスケーラビリティがあります。「テスト・データの定義」ダイアログには、マッピング全体のデータ量を制限できる、行を制限するオプションがあります。また、独自の表を作成してレコードを手動で操作することにより、独自のデータ・セットを定義することもできます。

「ステップのデータ」ウィンドウやウォッチ・タブに表示される列数を制限するには、表示セットを使用します。デフォルトでは、すべての演算子に表示セット「すべて」が設定されています。マップされている属性のみを表示するには、表示セット「MAPPED」を使用します。マッピング・エディタを直接使用すると、表示セットをソースに手動で追加できます。入出力グループでマウスを右クリックして、「表示セットの使用」を選択し、表示セットを選択します。

制限事項

ここでは、マッピング・デバッガに装備されていながらまだ使用することのできない機能、および今後のリリースで追加される予定の機能について説明します。

1. マッピング・エディタでデバッグ・モードで実行されるマッピングは、デバッグの目的のみに使用できます。マッピング・エディタから実行されるマッピングは、デプロイメント・マネージャを使用して実行されるマッピングほどパフォーマンスが高くありません。これは、デバッグ機能のサポートに必要な一時的なオブジェクトが設定されているためです。マッピングを実行するには、デプロイメント・マネージャを使用してください。
2. ツールバーの「一時停止」ボタン、または「デバッグ」メニューの対応するメニュー項目を使用してアクティブなデバッグを一時停止することはできません。
3. マッピングの統計機能が今後のリリースで追加される予定です。統計は、左下のパネルに新しいタブとして表示されます。
4. Runtime Audit Browser を使用して、デバッグ・モードでのマッピングの実行結果を確認することはできません。
5. ブレーク・ポイントおよびウォッチの設定は、デバッグ・セッション間で維持されません。
6. 現行では、PL/SQL パッケージとして実装されるマッピングのみ、デバッグ・モードで実行できます。

7. 次のマッピング演算子は、デバッグ・モードでマッピングを実行する際に使用できません。
   - アドバンスト・キュー
   - フラット・ファイル
   - ABAP アプリケーションから使用するソース表

8

マッピング演算子の使用方法

この章では、演算子をマッピングに使用してデータを変換する方法と、Expression Builder を使用して式を作成する方法について説明します。

この章では、次のトピックについて説明します。

Expression Builder の使用方法

この章で説明する一部のデータ・フロー演算子では、式を作成する必要があります。式とはデータを変換したり、制約を指定したりするための文または句を意味します。これらの式を組み合せることにより、SQL 文をインラインで構成できます。各式のタイプはデータ・フロー演算子のロールによって異なります。式を作成するには、Expression Builder を使用するか、あるいは「演算子のプロパティ」ウィンドウまたは「属性のプロパティ」ウィンドウの「式」フィールドに式を入力します。

Expression Builder の起動

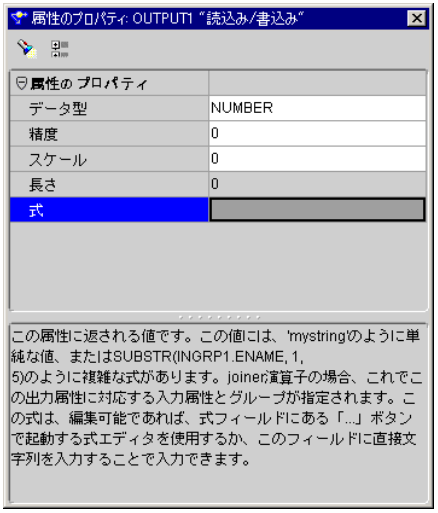
フィルタ、結合子、スプリッタおよび集計演算子の場合は、「演算子のプロパティ」ウィンドウから Expression Builder を起動できます。

式、データ・ジェネレータおよび定数演算子の場合は、「属性のプロパティ」ウィンドウから Expression Builder を起動できます。

Expression Builder を起動する手順は次のとおりです。

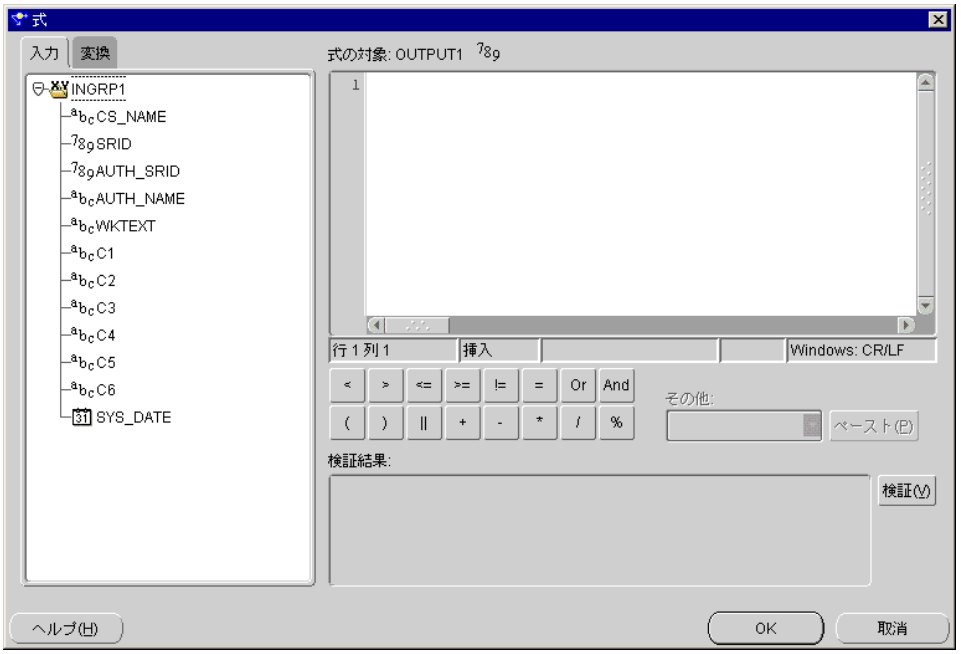
1. 「演算子のプロパティ」ウィンドウまたは「属性のプロパティ」ウィンドウで、「式」フィールドの「...」ボタンをクリックします。図 8-1 は、「属性のプロパティ」ウィンドウです。

図 8-1「属性のプロパティ」ウィンドウ



Expression Builder は、図 8-2 のように表示されます。

図 8-2 Expression Builder のインタフェース



2. 次のいずれかの方法で式を作成します。
   - 「式」フィールドにテキストを入力します。
   - 左側のパネルの「入力」タブおよび「変換」タブからアイテムをドラッグして、右側の「式」フィールドにドロップします。
   - 左側のパネルの「入力」タブおよび「変換」タブでアイテムをダブルクリックします。
   - 「式」フィールドの下にある算術演算子ボタンをクリックします。
3. 「検証」をクリックします。

   式の構文に間違いがないかどうか検証されます。
4. 「OK」をクリックして式を保存し、Expression Builder を終了します。

Expression Builder のユーザー・インタフェース

Expression Builder の構成は、次のとおりです。

- 左側のパネルのナビゲーション・ツリーには次の 2 つのタブが表示されます。
  - **「入力」タブ :** 入力パラメータのリストです。
  - **「変換」タブ :** Oracle 変換ライブラリ、グローバル共有ライブラリ、またはカスタムの変換ライブラリに保存されている定義済のファンクションとプロシージャのリストです。詳細は、『Oracle Warehouse Builder トランスフォーメーション・ガイド』を参照してください。
- **「式」フィールド :** 右側のパネルの上部には、「式」フィールドがあります。このフィールドを使用して、式の入力や編集を行います。
- **算術演算子ボタン :** 「式」フィールドの下には、算術演算子ボタンがあります。このボタンを使用すると、手動で入力しなくても式を作成できます。使用できる算術演算子は、アクティブなデータ・フロー演算子のタイプによって異なります。
- **その他 :** このドロップダウン・リストには、現在の式で使用できる SQL 句が一覧表示されます。

  Oracle9*i* Database からは、DECODE ファンクションのかわりに CASE ファンクションの使用が推奨されています。これは、DECODE ファンクションでは SQL しか生成できませんが、CASE ファンクションでは SQL と PL/SQL の両方を生成できるためです。Warehouse Builder で、式内に DECODE ファンクションを使用すると、コード生成時には CASE ファンクションに変換されます。つまり、すべてのオペレーティング・モード（セットベース、行ベースなど）で DECODE ファンクションを配布でき、Oracle データベースのリリース（8.1、9.0 以上）を意識しなくても済みます。

  たとえば、Warehouse Builder で次のファンクションを使用するとします。

  ```
  DECODE (T1.A, 1, 'ABC', 2, 'DEF', 3, 'GHI', 'JKL')
  ```

  このファンクションは次のように変換されます。

  ```
  CASE T1.A WHEN 1 THEN 'ABC'
  WHEN 2 THEN 'DEF'
  WHEN 3 THEN 'GHI'
  ELSE 'JKL'
  ```

- **「検証結果」フィールド :** 右側のパネルの下部には、「検証結果」フィールドがあります。このフィールドの右側にある「検証」ボタンを選択すると、検証結果が表示されます。

- **「検証」ボタン :** このボタンを使用して、Expression Builder で作成した現在の式を検証します。式を検証すると、その式で参照しているすべてのマッピング・オブジェクトにリポジトリ・オブジェクトが関連付けられていることを確認できます。Expression Builder で作成できる式は、演算子の入力とそのプロジェクトで実行可能な変換のみです。この制限によって、演算子の外部の要素が変更されたときの式の無効化を防止できます。配布データベースが Design Repository と異なる場合、データベースが式を受け付けないことがあります。式が有効でも、データベースに対して不適切である場合にこの現象が発生します。このような式エラーは配布時にのみ検出されます。

集計演算子

集計演算子は、合計値や平均値などのデータ集計を計算し、集計されたデータを使用して行セットを出力します。

各集計演算子は GROUP BY 句と HAVING 句を共有するため、出力グループのすべての属性のカーディナリティが一致します。出力行セットの行数は、入力行数と同じか、それより少なくなります。

集計演算子では、1 つの入力グループと 1 つの出力グループを使用します。ソースを入力グループに接続すると、それに対応する集計行セットが出力グループに作成されます。結果出力には、GROUP BY 句または HAVING 句で使用した列が含まれます。

集計演算子では次のプロパティを指定します。

- **Group By 句 :** 入力行セットをグループ化し、グループごとに 1 つの集計行を出力する方法を定義します。グループ化の方法は、入力グループの順序付き属性リストで指定します。GROUP BY 句に属性を追加すると、集計式は NONE にデフォルト設定されます。
- **Having 句 :** 出力グループの行グループを制限するブール条件です。この条件が true の行グループのみに出力グループとして返されます。Having 句を指定しない場合は、すべてのグループのすべての集計行が出力グループとして返されます。
- **式 :** 属性に対して実行する集計関数を定義します。グループ化されていない各出力属性に対して、集計式の結果を DISTINCT にするか ALL にするかを選択します。ALL がデフォルトです。次に例を示します。
  - ALL: `Select AVG(ALL sal) from emp;`
  - DISTINCT: `Select AVG(DISTINCT sal) from emp;`

  DISTINCT 集計では、結果を算出する前にすべての重複行が削除されます。上の例では、正しい平均値が計算されます。

  ALL 集計では、すべての行の平均値が結果として返されます。

  集計関数が不要である場合は、NONE を指定します。属性の集計に NONE を指定すると、その属性は結果としての GROUP BY 関数に自動的に追加されます。

マッピングで集計演算子を使用する手順は次のとおりです。

1. 集計演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. キャンバスで、ソース属性を集計演算子の入力グループに接続します。
3. 集計演算子を右クリックして、「編集」を選択します。

   演算子エディタが表示されます。
4. 「出力属性」タブで「追加」を選択して、出力属性を追加します。
5. 「OK」をクリックして演算子エディタを閉じます。
6. 「属性のプロパティ」ウィンドウで、各出力属性について式を定義します。この手順の詳細は、8-6 ページの「集計演算子での出力属性の定義」を参照してください。
7. 演算子について、Group By 句を定義します。この手順の詳細は、8-6 ページの「集計演算子での出力属性の定義」を参照してください。

集計演算子での出力属性の定義

出力属性について式を定義する手順は次のとおりです。

1. マッピング・キャンバスの集計演算子で、出力属性を右クリックして「属性のプロパティ」を選択します。
2. 「式」プロパティの右側の「...」ボタンをクリックします。

   図 8-3 に示すように、「式」ダイアログが表示されます。

図 8-3「式」ダイアログ

3. 「ファンクション」、「すべて / 個別」および「パラメータ」の各ドロップダウン・リストから、それぞれの値を選択します。
4. 「OK」をクリックします。

Group By 句を定義する手順は次のとおりです。

1. マッピング・キャンバスの集計演算子で、出力属性を右クリックして「属性のプロパティ」を選択します。
2. 「Group By 句」プロパティの右側の「...」ボタンをクリックします。

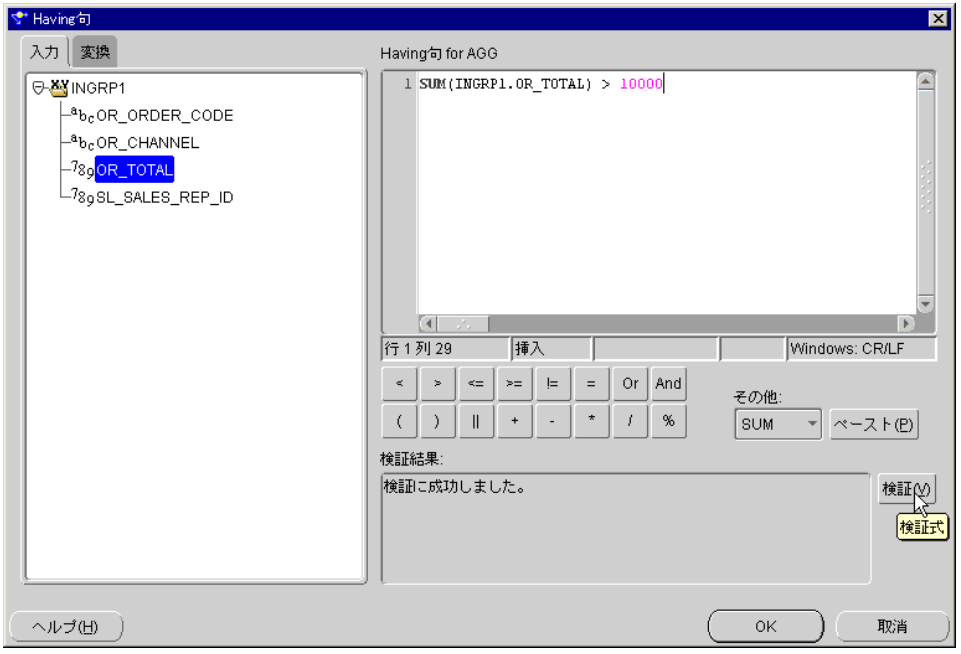
   図 8-4 に示すように、「Group By 句」ダイアログが表示されます。

図 8-4「Group By 句」ダイアログ

3. 「使用可能な属性」リストから「GROUP BY 属性」リストに属性を移動します。
4. 「OK」をクリックします。
5. 「...」ボタンをクリックし、Expression Builder を使用して HAVING 句を定義します。

6. 図 8-5 に示すように、sum(INGRP1.OR_TOTAL) > 10000 などの式を作成します。

図 8-5 Sum 文が定義された Expression Builder



7. 集計演算子の出力グループ内で、編集した属性をターゲットの属性にマッピングします。

定数演算子

定数を定義するには定数演算子を使用します。定数は、マッピングの実行の開始時に初期化されます。定数値はマッピング内の任意の場所で使用できます。

定数演算子では、1 つの出力グループが作成されます。このグループには、1 つ以上の定数属性を含めることができます。定義した定数の各データ型に対して、出力式は同じデータ型の値を返す有効な SQL 式である必要があります。VARCHAR、CHAR または VARCHAR2 データ型の場合は、定数の文字列リテラルを単一引用符で囲みます（例：'my_string'）。

定数演算子には次のプロパティがあります。

- **式 :** その属性によって表される定数値を定義します。

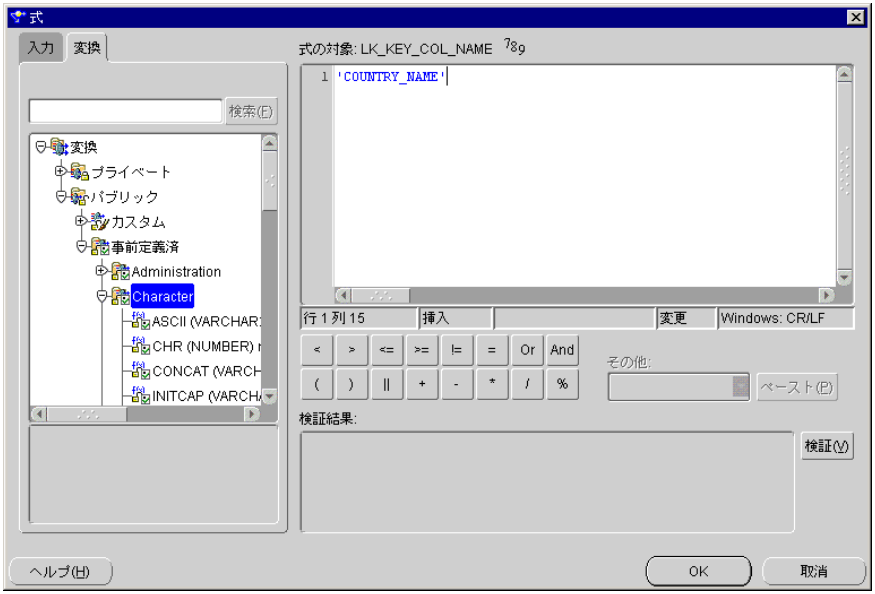
マッピングで定数演算子を使用する手順は次のとおりです。

1. 定数演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. 演算子を右クリックして、「編集」を選択します。

   演算子エディタが表示されます。
3. 「出力属性」タブで「追加」を選択して、出力属性を作成します。
4. 「OK」をクリックして演算子エディタを閉じます。
5. マッピング・キャンバスの演算子で、属性を右クリックして「属性のプロパティ」を選択します。

   「属性のプロパティ」ウィンドウが表示されます。
6. 図 8-6 に示すように、「式」フィールドに式を入力します。または、「...」をクリックし、Expression Builder で式を定義します。定数演算子の属性に割り当てる長さ、精度およびスケールのプロパティは、マッピングで定義した式により返される実際の値と一致する必要があります。

図 8-6 定数が表示された Expression Builder



7. 「OK」をクリックします。
8. 出力属性を適切なターゲット属性に接続します。

データ・ジェネレータ演算子

レコード番号、システムの日付、順序の値などを取り込むにはデータ・ジェネレータ演算子を使用します。

推奨事項： PL/SQL マッピングの場合は、データ・ジェネレータのかわりに定数演算子または順序のマッピング演算子を使用します。

フラット・ファイル・ソースを含むマッピングの場合、データ・ジェネレータ演算子には、定数情報を入力する場所も含まれます。フラット・ファイル・ソースとフラット・ファイル・ターゲットを含むマッピングの場合、データ・ジェネレータ演算子によってマッピングが SQL*Loader に接続され、データベース・レコードに保存されているデータが生成されます。

次の関数を使用できます。

- RECNUM
- SYSDATE
- SEQUENCE

Warehouse Builder でデータを生成する際にフィールド指定として使用できるのは、順序、レコード番号、システム日付および定数のみです。SQL*Loader は、LOAD キーワードで指定された数のレコードを挿入します。

データ・ファイルのレコード番号を列に設定

レコードのロード元のレコード数を属性に設定するには、RECNUM キーワードを使用します。レコードは、最初のデータ・ファイルの先頭から順にカウントされます。最初のレコード番号は 1 です。RECNUM は論理レコードが作成されるたびに増分します。廃棄、スキップ、拒否、またはロードされたレコードが増分対象となります。たとえば、SKIP=10 というオプションを指定した場合、ロードされる最初のレコードの RECNUM は 11 になります。

現在の日付を列に設定

SYSDATE を指定した列には、現在のシステムの日付が取り込まれます。この日付は、SQL 言語の SYSDATE 関数で定義されています。

ターゲット列は、CHAR 型または DATE 型である必要があります。ターゲット列が CHAR 型の場合、日付は dd-mon-yy 形式でロードされます。ロードした日付には、この形式でのみアクセスできます。システムの日付を DATE 列にロードした場合は、様々な時刻と日付の形式でアクセスできます。従来型パスによるロードに挿入された各レコード配列、およびダイレクト・パス・ロード時にロードされた各レコード・ブロックで新しいシステムの日付 / 時刻を使用できます。

一意な順序番号を列に設定

列に一意な値を入力するには SEQUENCE キーワードを使用します。ロードまたは拒否された各レコードについて、SEQUENCE 列の値が増分します。レコードが廃棄された場合またはスキップされた場合は、SEQUENCE 列の値は増分しません。

列名と SEQUENCE 関数の組合せが完全な列指定となります。表 8-1 は、順序値として使用できるオプションを示しています。

表 8-1 順序値のオプション

値	説明
column_name	データベース内で、順序を割り当てる列の名前です。
SEQUENCE	列の値を指定します。
integer	最初の順序番号を指定します。
COUNT	表内の現在のレコード数に増分値を加えた値で順序が開始します。
MAX	列の現在の最大値に増分値を加えた値で順序が開始します。
incr	レコードがロードまたは拒否された後、順序番号が増分する値です。

書式エラーまたは Oracle エラーによってレコードが拒否された場合、これをマスクするために、生成された順序番号が入れ替えられることはありません。たとえば、4 つの行のある列に順序番号 10、12、14、16 が割り当てられており、順序番号 12 の行が拒否された場合、挿入された 3 つの行には、10、12、14 ではなく、10、14、16 が割り当てられます。拒否されたデータを修正して再挿入する際、順序にあわせて列を手動で設定できます。

データ・ジェネレータ演算子では、出力グループを 1 つのみ使用しますが、レコード番号、システムの日付、および一般的な順序に対応した属性が事前定義されています。これらの属性は変更しないでください。かわりに、新しい属性を作成できます。データ・ジェネレータ演算子は、SQL*Loader のマッピングでのみ有効です。

注意： 1 つのマッピングで使用できるデータ・ジェネレータ演算子は 1 つのみです。

データ・ジェネレータ属性では次のプロパティを指定します。

- **式:** この属性のマッピング時に使用する式です。式に値を入力するときは、有効な SQL*Loader 構文を使用してください。

マッピングでデータ・ジェネレータを使用する手順は次のとおりです。

1. データ・ジェネレータ演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. 演算子を右クリックして、「編集」を選択します。

   演算子エディタが表示されます。
3. 「出力属性」タブを選択します。

   事前定義された出力属性 RECNUM、SYS_DATE および SEQUENCE が表示されます。
4. 「出力属性」タブでプロパティを定義し、事前定義された出力属性に説明（オプション）を入力します。
5. 「OK」をクリックして演算子エディタを閉じます。
6. マッピング・キャンバスの演算子で、RECUM 属性を右クリックして「属性のプロパティ」を選択します。

   「属性のプロパティ」ウィンドウが表示されます。
7. 「式」フィールドで「...」をクリックして Expression Builder を起動し、式を定義します。
8. SEQUENCE 属性について手順 6 と 7 を繰り返します。

デュプリケータ解除演算子

ソース内の重複するデータを削除するには「デュプリケータ解除」を使用します。この演算子は、生成されたコードの SELECT 文に DISTINCT 句を挿入します。

重複データを削除する手順は次のとおりです。

1. デュプリケータ解除演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. ソース演算子の属性を、デュプリケータ解除演算子のグループ入力 / 出力に接続します。
3. デュプリケータ解除演算子グループの属性をターゲット演算子の属性に接続します。

式演算子

式演算子では、この演算子の 1 つの出力パラメータについて、非プロシージャ・アルゴリズムを定義する SQL 式を記述できます。

式では、入力パラメータ名、変数名およびライブラリ・ファンクションの組合せを使用できます。式演算子を使用すると、入力行セットのカーディナリティを維持したまま、PL/SQL タイプの式に基づいて行セット内の行の列値データを変換できます。これらの式を作成するには、出力属性の「属性のプロパティ」ウィンドウを開き、Expression Builder を起動します。

式演算子では、1 つの入力グループと 1 つの出力グループのみを使用します。これらのグループは、演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップするときに自動的に作成されます。

この演算子の出力式では集計関数を使用できません。集計関数を実行する場合は、集計演算子を使用してください。8-5 ページの「集計演算子」を参照してください。

式演算子の各出力属性に、次のプロパティを指定します。

- **データ型 :** 属性のデータ型。
- **精度 :** 属性の精度。数値型の属性でのみ使用します。
- **スケール :** 属性のスケール。数値型の属性でのみ使用します。
- **長さ :** 属性の長さ。キャラクタ型の属性でのみ使用します。
- **式 :** 出力属性の式テンプレート。コード生成時には、式テンプレートの入力属性名が入力属性として使用されます。

マッピングで式演算子を使用する手順は次のとおりです。

1. 式演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. 適切なソース属性を、式入力グループに接続します。

   入力属性が演算子にコピーされます。
3. マッピング・キャンバスで演算子を右クリックし、「編集」を選択します。

   演算子エディタが表示されます。
4. 「出力属性」タブで「追加」を選択し、属性名、データ型などのプロパティを指定します。
5. 「OK」をクリックして演算子エディタを閉じます。
6. 式演算子で出力属性を右クリックし、「属性のプロパティ」を選択します。
7. 「式」プロパティの右側のフィールドをクリックして、フィルタ条件式を入力します。または、「...」をクリックして Expression Builder を起動し、式を定義します。
8. 「属性のプロパティ」ウィンドウを閉じます。
9. 式演算子の出力属性を適切なターゲット属性に接続します。

フィルタ演算子

行を特定の条件に基づいてフィルタリングするには、フィルタ演算子を使用します。

フィルタ演算子は、マッピングの生成コードに WHERE 句を挿入することによって、ソースからターゲットに渡されるデータをフィルタリングします。ソース演算子をフィルタ演算子に接続し、フィルタ条件を適用します。次に、行のサブセットを次の演算子に渡します。

フィルタ演算子では、入力 / 出力グループを 1 つのみ使用します。このグループは、ソース行セットとターゲット行セットの両方に接続できます。結果として出力される行セットは、ブール・フィルタ条件式に基づいてフィルタリングされたソース行セットのサブセットです。

フィルタ演算子では次のプロパティを指定します。

- **フィルタ条件 :** 出力行セットに渡す行を指定するためのブール条件。

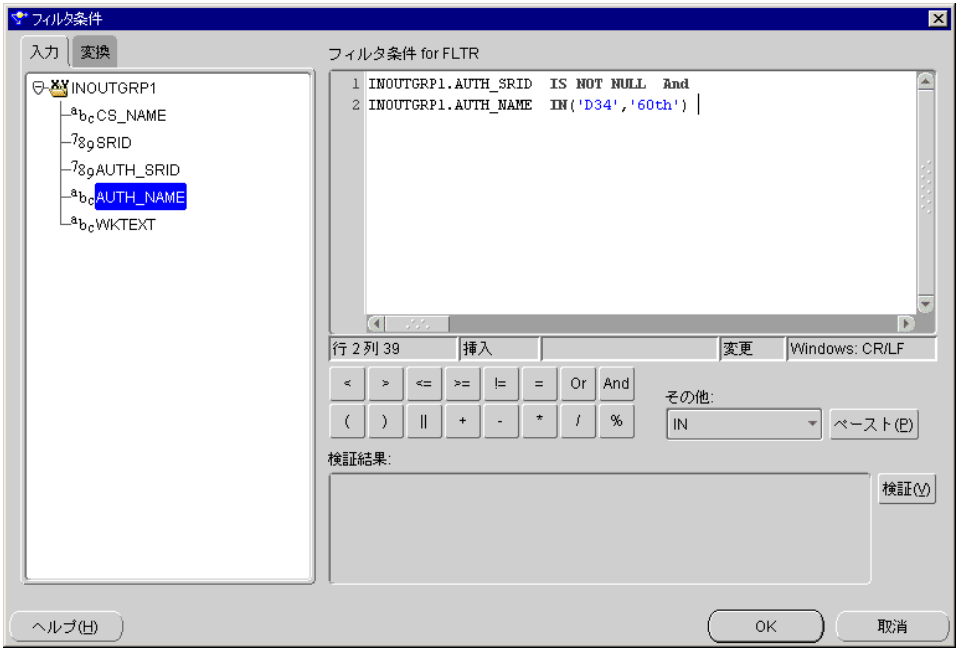
フィルタ演算子を含むマッピングを生成する場合、コード・ビューアでは、フィルタ条件式が、セットベース・ビュー・モードの WHERE 句として表示されます。元のフィルタ条件のフィルタ入力名は、ソース表の実際の列名に置換されます。この列名は、ソース表の別名によって修飾されます。

マッピングでフィルタ演算子を使用する手順は次のとおりです。

1. フィルタ演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. ソース属性をフィルタ演算子の入力属性に接続します。
3. フィルタ演算子のヘッダーを右クリックし、「演算子のプロパティ」を選択します。

   「フィルタ (F) プロパティ」ウィンドウが表示されます。
4. 「フィルタ条件」プロパティの右側のフィールドをクリックして、フィルタ条件式を入力します。または、「...」をクリックし、図 8-7 に示すように Expression Builder を使用してフィルタ条件を定義します。

図 8-7 フィルタ条件が表示された Expression Builder



5. Expression Builder で「OK」をクリックし、「フィルタ (F) プロパティ」ウィンドウを閉じます。
6. フィルタ演算子の出力をターゲットの入力 / 出力グループに接続します。

結合子演算子

結合子演算子を使用して、カーディナリティの異なる複数のソースから取り込んだ複数の行セットを結合し、1 つの出力行セットを生成できます。

結合子演算子では、ブール条件を使用して、各ソース行セットの列値を 1 つ以上の他の行セットに関連付けます。

注意： データ・ソースと結合子の間に他の演算子を配置すると、複雑な SQL または PL/SQL が生成されます。

入力行セットが外部キーによって関連付けられている場合は、その関係に基づいてデフォルトの結合条件が作成されます。このデフォルトの条件をそのまま使用するか、または変更できます。ソースが外部キーによって関連付けられていない場合は、結合条件を定義する必要があります。

デフォルトの外部キーにより WHERE 句が重複して作成される場合は、結合子演算子により重複句が削除されます。これは、結合条件で複数の外部キーが参照される場合に生じます。たとえば、表 T1 に表 T2 の一意キー UK1 を指す外部キー FK1 があり、表 T2 に表 T1 の一意キー UK2 を指す外部キー FK2 がある場合、結果として次のような結合条件が作成されます。

```
T1.A = T2.A AND T1.B = T2.B /*All instances of FK1 -> UK1 are reduced to one where
clause*/ AND
T2.B = T1.B AND T2.C = T1.C /*All instances of FK2 -> UK2 are reduced to one where
clause*/
```

この結合条件は、次の結合子演算子により生成されます。

```
T2.A = T2.A AND T1.B = T2.B AND T2.C = T1.C
```

結合子演算子では次のプロパティを指定します。

- **結合条件 :** 結合条件のテキスト式テンプレート。コード生成時には、入力属性がソース列に置き換えられます。この式は、WHERE 句で使用できる有効な SQL 式です。

注意： 結合条件は PL/SQL のコンテキストで定義されます。Warehouse Builder では、SAP ソースの場合、PL/SQL の結合条件を ABAP コンテキストで解釈することにより ABAP コードを生成できます。ABAP は、定義された外部キーの関係でのみ結合できます。

結合子演算子の属性では次のプロパティを指定します。

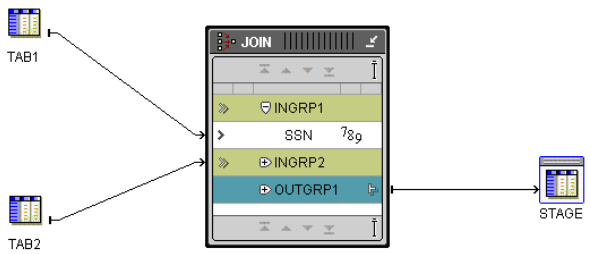
- **データ型 :** 属性のデータ型。
- **精度 :** 属性の精度。数値型の属性でのみ使用します。
- **スケール :** 属性のスケール。数値型の属性でのみ使用します。
- **長さ :** 属性の長さ。文字列型の属性でのみ使用します。

マッピングで結合子演算子を使用する手順は次のとおりです。

1. 結合子演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. 最初のソースの出力グループを、結合子入力グループに接続します。

   対応する入力データと同じデータ型の出力属性が作成されます。
3. 2 番目のソース演算子のグループを結合子演算子の INGRP2 グループに接続します。

図 8-8 マッピングに使用した結合子演算子



4. 結合子演算子のヘッダーを右クリックし、「演算子のプロパティ」を選択します。

   「結合子 (J) プロパティ」ウィンドウが表示されます。
5. 「結合条件」フィールドに結合条件を入力します。または、「...」をクリックし、Expression Builder を使用して式を定義します。
6. 「結合子 (J) プロパティ」ウィンドウを閉じます。

結合子の制約

結合条件式では、SUM などの集計関数を使用できません。集計関数を使用した場合、そのマッピングの生成コードを配布するときにコンパイル・エラーが発生します。結合子演算子の場合、入力グループの数に制限はありませんが、出力グループは 1 つのみです。

入力グループの順序は、結合順序として使用されます。ANSI 結合と Oracle 結合との大きな違いは、ANSI 結合では結合順序を明確に指定する必要がある点です。Oracle 結合では結合順序を指定する必要がありません。

フィルタ条件は、結合の後に適用されます。たとえば、次のような結合が考えられます。

```
Input1.c --- +
Input2.c --- +---> Joiner
Input3.c --- +
```

これに次のような条件があるとします。

- **条件 1:** Input1.c (+) = Input2.c (+)
- **条件 2:** Input2.c = Input3.c
- **条件 3:** Input1.c is null

最初の 2 つの条件は true の結合で、3 番目の条件はフィルタ条件です。ANSI コードが生成される場合、Warehouse Builder では、次のように解釈されます。

```
select ...
from Input1 full outer join Input2 on (Input1.c = Input2.c)
join Input3 on (Input2.c = Input3.c)
where Input1.c is not null;
```

完全外部結合の指定

ターゲット・ウェアハウスに Oracle9*i* 以上のバージョンが使用されている場合、Warehouse Builder 結合子は完全外部結合もサポートします。完全外部結合条件を指定するには、関係演算子の両側に（+）記号を付ける必要があります。次に例を示します。

```
T1.A (+) = T2.B (+)
```

完全外部結合を使用すると、結果は次のようになります。

- ソース T1 および T2 からの行は条件 T1.A = T2.B を満たす。
- ソース T1 からの行は条件を満たさない。T2 に対応する列には NULL が移入される。
- ソース T2 からの行は条件を満たさない。T1 に対応する列には NULL が移入される。

注意： 関係演算子は等号のみではありません。>、<、!=、>=、<= などの演算子も使用できます。

T1.A = T2.B (+) などの部分外部結合に Oracle SQL 構文を使用する場合、関係演算子の両側に (+) 記号を付けると、無効な Oracle SQL 構文になります。ただし、Warehouse Builder では、(+) 記号が 2 つ付いた条件は ANSI SQL 構文に変換されます。次に例を示します。

```
SELECT ...
FROM T1 FULL OUTER JOIN T2 ON (T1.A = T2.B);
```

- 完全外部結合を使用する場合は、次の事項に注意してください。
- Oracle9*i* Database 以外のターゲット・システムに対して完全外部結合条件を指定すると、検証エラーが発生してコードは生成されません。
- ANSI 結合構文が生成されるのは、結合子に完全外部結合条件を指定した場合のみです。指定しない場合は、次のように Oracle 独自の結合構文が生成されます。

  ```
  SELECT ...
  FROM T1, T2
  WHERE T1.A = T2.B;
  ```

- 入力グループの順序は、ANSI 結合の順序として使用されます。3 つ以上の入力グループを結合する場合、結合順序は入力グループの順序によって決定されます。Oracle 独自の結合構文とは異なり、ANSI 結合構文では結合順序を明示的に指定する必要があります。
- 結合子の作成時には、結合する順序で、入力グループを作成する必要があります。たとえば、3 つの入力グループを T1、T2、T3 の順序で作成すると、結合条件は次のとおりになります。

  ```
  T1.A (+) = T2.A (+) and T2.A = T3.A
  ```

  Warehouse Builder では次のように生成されます。

```
SELECT ...
      FROM T1 FULL OUTER JOIN T2 ON (T1.A=T2.A)
                          JOIN T3 ON (T2.A=T3.A);
```

入力グループを T1、T3、T2 のような別の順序で作成すると、Warehouse Builder では次のように生成されます。

```
SELECT ...
FROM T1 JOIN T3 ON (1=1)
        JOIN T2 ON (T1.A=T2.A and T2.A=T3.A);
```

T1 と T3 の結合には、指定された結合条件はありません。Warehouse Builder では条件 1=1（必然的にブール値が true）が実行され、指定した 2 つの条件が T2 の結合に使用されます。

- 完全外部結合と結合条件はどちらも同じ結合子で指定できます。ただし、両方の条件が同じソースで指定されている場合は、結合の強いタイプがコードの生成に使用されます。たとえば、次のように指定するとします。

  ```
  T1.A(+) = T2.A(+) and T1.B = T2.B
  ```

  この場合、Warehouse Builder では完全外部結合ではなく、結合文が生成されます。これは、`T1.B = T2.B` のほうが、T1 と T2 間の完全外部結合条件よりも強いためです。

- 完全外部結合と部分外部結合条件は同じ結合子で指定できません。完全外部結合を指定した場合は、結合条件内のどこにも部分外部結合を指定できません。たとえば、`T1.A (+) = T2.A (+) and T2.B = T3.B (+)` を指定すると、検証エラーが発生してコードは生成されません。

完全外部結合条件の作成

等価結合では、2 つの表のキー値が一致する必要があります。完全外部結合の場合、キー値が突き合され、一致しないキー値についてはその表示に NULL が作成されます。左側外部結合または右側外部結合では、指定した表のすべての行が保持されます。

Oracle8*i* の場合、SQL で外部結合を作成するには結合条件変数（+）を使用します。

```
SELECT ...
FROM A, B
WHERE A.key = B.key (+);
```

この例は左側外部結合です。表 B のいずれの行も表 A の行と一致しない場合でも、表 A の行は結合結果に含まれます。Oracle8*i* で完全外部結合を作成するには、複数の SQL 文を使用する必要があります。

Expression Builder では、次の構文を使用して完全外部結合を作成できます。

```
TABLE1.COL1 (+) = TABLE2.COL2 (+)
```

Oracle8*i* はこの構造をサポートしていません。Oracle Database は ANSI SQL 1999 に準拠しています。ANSI SQL 1999 標準では、完全外部結合を実行するための式の構文が定義されています。コード・ジェネレータは、前述の式を ANSI SQL 1999 準拠の完全外部結合文に変換します。たとえば、次のようになります。

```
SELECT ...
FROM table1 FULL OUTER JOIN table2 ON (table1.col1 = table2.col2)
```

この完全外部結合文は ANSI SQL 1999 に準拠しているため、生成されたコードは Oracle Database でのみ有効となります。Oracle8*i* データベースに対して完全外部結合を指定すると、検証エラーが発生します。

1 つの SQL 文内で完全外部結合と部分外部結合を一緒に使用することは可能ですが、AND または AND/OR 条件である必要があります。完全外部結合と部分外部結合を OR 条件で使用すると、予期しない AND 条件に解析されます。次に例を示します。

```
SELECT ...
FROM table1 FULL OUTER JOIN table2 ON (A = B or C = D)
```

この文は、Oracle Server では `A (+) = B (+) AND C = D` と評価されます。

マッピングで完全外部結合を使用する手順は次のとおりです。

1. 8-17 ページの手順 1 ～ 4 に従って、結合子演算子を追加します。
2. 「結合条件」フィールドに完全外部結合文を入力します。または、「...」をクリックし、Expression Builder を使用して式を定義します。
3. 「結合子 (J) プロパティ」ウィンドウを閉じます。

キー検索演算子

キー検索演算子を使用して、表、ビュー、キューブまたはディメンションからデータを検索します。

たとえば、キューブをロードするマッピングを定義する場合や、ディメンションにサロゲート・キーを定義する場合に、キー検索演算子を使用できます。この例では、ディメンション表でサロゲート・キーを参照し、対応する元のレコードを返すキー検索演算子を作成して、外部キーの関係を作成します。

表、ビュー、キューブまたはディメンションはキー検索演算子にバインドされます。1 つのマッピング内で複数のキー検索演算子を使用できます。

すべての一意の値を検索キーとして使用できます。検索キーは、RDBMS で定義されている主キーや一意キーである必要はありません。キー検索演算子は、ユーザーが入力したキーに基づいて参照表からデータを読み取り、一致する行を検索します。この演算子は、1 つの入力キーにつき 1 行を返します。

キー検索演算子の出力は、参照オブジェクトの列に対応します。参照表の複数の行がキー入力と一致する場合、出力のカーディナリティが入力と異なり、ターゲット演算子に渡されるデータとの整合性が損なわれ、実行時にエラーが発生します。各入力キーについて 1 つの行のみが参照されるように、一致条件に基づいたキーを使用してください。

キー検索の出力ではインバウンド調整を使用できます。アウトバウンド調整は使用できません。詳細は、6-41 ページ以降の「演算子とリポジトリ・オブジェクトの調整」を参照してください。

キー検索の各出力属性には、DEFAULT VALUE というプロパティがあります。入力値と一致する値が参照表内にない場合、出力行セットでは、NULL のかわりに DEFAULT VALUE プロパティが使用されます。生成されたコードでは NVL 関数が使用されます。キー検索を使用する場合は常に外部結合文が生成されます。

この演算子の検証を実行すると、次の点がチェックされます。

- 参照表内の完全な一意キーを使用して条件を指定していない場合は、警告が表示されます。一意キーを使用しないと、1 つの入力行に対して複数の行が検索される可能性があります。
- 条件内で、データ型の異なる複数の属性を "=" 比較演算子で結んでいる場合は、警告が表示されます。この場合、Oracle データベースによって暗黙的変換が実行されますが、ランタイム・エラーが発生する場合があります。

マッピングでキー検索演算子を使用する手順は次のとおりです。

1. キー検索演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。

   「マッピングキー (K) 参照の追加」ダイアログが表示されます。

2. 「マッピングキー (K) 参照の追加」ダイアログを使用して、1 つ以上の表を選択します。

   マッピング・キャンバスには、表を 1 つ選択するたびにキー検索演算子が 1 つ追加されます。「マッピングキー (K) 参照の追加」ダイアログの使用方法の詳細は、6-11 ページの「バインド可能な演算子の追加」を参照してください。

3. ソース属性を、キー検索演算子の入力グループに接続します。
4. キー検索演算子のヘッダーを右クリックし、「演算子のプロパティ」を選択します。

   「マッピングキー (K) 参照の追加」ウィンドウが表示されます。

5. 「参照条件」の右側のフィールドをクリックし、「...」ボタンをクリックします。

   図 8-9 に示すように、「参照条件」ダイアログが表示されます。

図 8-9「参照条件」ダイアログ

6. 図 8-10 に示すように、「参照表の列またはキー」ドロップダウン・リストから、参照エンティティ属性を選択します。

   選択した参照表の列に対応する属性を選択してください。

   コンポジット・キー以外の場合は、ドロップダウン・リストから入力属性またはキーを選択します。「リストに追加」をクリックします。「参照条件」ダイアログの下部の表に、列または入力属性の組合せが追加されます。

   コンポジット・キーの場合は、「リストに追加」をクリックします。「キーの入力へのキー列の一致」ダイアログが表示されます。「入力属性」ドロップダウン・リストからキー入力属性を選択します。「OK」をクリックします。

図 8-10「キーの入力へのキー列の一致」ダイアログ

7. 「OK」をクリックして、「参照条件」ダイアログを閉じます。

アドバンスト・キューのマッピング演算子

キュー内のメッセージをソース・システムからターゲット・システムに伝播するには、アドバンスト・キュー（AQ）のマッピング演算子を使用します。

Warehouse Builder では、AQ をソースまたはターゲットとして使用できます。AQ ソース・オブジェクトをターゲット表またはステージング表にマッピングし、そのマッピングをターゲット・データベースに配布できます。AQ マッピングを実行すると、その変更内容が AQ からウェアハウス内のターゲット表に伝播されます。マッピングが正常に実行された後、同じマッピングを起動しても、すでに処理されているメッセージは表示されません。マッピングに失敗した場合、ソース AQ からメッセージはデキューされません。

Warehouse Builder では、AQ をウェアハウス内のターゲットとして使用できます。リポジトリ内のソース AQ は、異なるメッセージング・システムやアプリケーションからのデータを統合する中心的な AQ を意味します。Warehouse Builder では、このソース AQ をターゲット AQ にマッピングし、あるタイプのメッセージング・システムのメッセージを別のタイプのメッセージング・システムに伝播できます。

詳細は、次を参照してください。

- アドバンスト・キューの使用方法（3-47 ページ）
- アドバンスト・キューについて（3-48 ページ）
- アドバンスト・キューの定義の作成（3-49 ページ）

AQ マッピングの作成

マッピングでアドバンスト・キューのマッピング演算子を使用する手順は次のとおりです。

1. アドバンスト・キューのマッピング演算子をマッピング・キャンバスにドラッグ・アンド・ドロップします。

   「マッピングアドバンスト・キュー」ダイアログが表示されます。

2. 新しい AQ 定義をリポジトリにインポートするか、すでにリポジトリにインポートされている AQ から選択できます。

   マッピングで AQ 演算子にバインドされている AQ はすべて、そのマッピングと同じウェアハウス・モジュールに属している必要があります。同じウェアハウス・モジュールに属していない場合は、マッピングの検証時にエラーが発生します。

   「マッピングアドバンスト・キュー」ダイアログの使用方法の詳細は、6-11 ページの「バインド可能な演算子の追加」を参照してください。

3. 「OK」をクリックします。

4. AQ のマッピング演算子の出力属性を、ターゲット演算子の入力グループに接続します。

マッピングに AQ を使用する例

ある会社では、顧客サービス担当者を地域ごとのすべての顧客に割り当てています。顧客数が増加すると、その会社では新しいサービス担当者を雇用して、新規顧客や既存顧客を割り当てます。新しいサービス担当者に対する顧客の割当てを、キューを使用して追跡してみます。AQ を作成して顧客表の変更内容をメッセージで捕捉できます。

この AQ は、Warehouse Builder にインポートして、マッピング時のデータ・ソースとして使用できます。マッピング・エディタで AQ ソースを顧客ターゲット表に接続します。Warehouse Builder でこのマッピングの PL/SQL スクリプトを生成し、そのスクリプトをターゲット・ウェアハウスに配布できます。マッピングを実行すると、メッセージはデキューされ、ソース顧客表のサービス担当者の新しい割当てがターゲット・データベースの顧客表に伝播されます。

各マッピングを AQ ごとの個別のサブスクライバとして Warehouse Builder に登録すると、異なるマッピングが同じ AQ を参照しても、相互に干渉することはありません。どのマッピングも、すべてのサブスクライバに公開されている AQ 内のメッセージをすべて参照します。

アドバンスト・キューの調整

AQ 演算子にインバウンド調整を実行すると、バインド先である AQ オブジェクトのリポジトリ定義の変更内容で AQ 演算子を更新できます。詳細は、6-42 ページの「インバウンド調整」を参照してください。Warehouse Builder では現在、AQ 演算子にアウトバウンド調整を実行できません。

アドバンスト・キュー演算子のプロパティ

マッピングで使用する AQ のマッピング演算子に次のプロパティを構成できます。

- マッピング・エディタで AQ 演算子を右クリックして、「演算子のプロパティ」を選択します。

  「バウンド名」プロパティを含む「アドバンスト・キューのマッピングプロパティ」ダイアログが表示されます。コード生成時には、この名前で AQ が識別されます。

- マッピング・エディタで AQ 演算子を右クリックして、「編集」を選択します。

  「一般」タブ、「グループ」タブ、「入力 / 出力」タブを含むアドバンスト・キューのマッピングエディタが表示されます。

一般

AQ のマッピング演算子の名前を変更するには、既存の名前を選択して、そこに上書きします。AQ のマッピング演算子に説明（オプション）を入力します。

グループ

AQ のマッピング演算子グループの名前を変更するには、既存の名前を選択して、そこに上書きします。どの AQ 演算子にも 1 つの INOUT グループがあります。これは読取り専用フィールドです。AQ のマッピング演算子グループに説明（オプション）を入力します。

入力 / 出力

AQ 演算子の INOUT グループには属性を追加できません。属性名、データ型、長さ、精度、スケールおよび説明（オプション）を表示し、編集することはできます。

マッピング実行における AQ の前提条件

Warehouse Builder から配布された AQ は、リモート・システムの AQ ソースを表すプロキシ AQ です。Oracle ではメッセージのリモート・デキューがサポートされるため、最初にローカル・スキーマにソース AQ のエージェントを配布し、ローカルに配布されたプロキシ AQ をソース・システムの AQ のサブスクライバとして登録する必要があります。
Warehouse Builder で AQ マッピングを正常に実行するには、ソース・システムおよびウェアハウス・システムで次の手順を実行する必要があります。

- 次の権限を AQ ソース・システム・ユーザーに付与します。

```
execute on dbms_aq
execute on dbms_aqadm
execute on aq$_agent
```

- 次の権限をウェアハウス・システム・ユーザーに付与します。

  ```
  execute on dbms_aq
  execute on dbms_aqadm
  execute on aq$_agent
  ```

- AQ ソース・システムから、AQ ターゲット・システムへのデータベース・リンクを作成します。

  これで、ターゲット・ユーザーはターゲット・プロキシ AQ をターゲット・インスタンスに配布できます。

- 配布用の AQ を構成する場合、AQ オブジェクトと、この AQ オブジェクトにバインドされる AQ 演算子があるマッピングが、どちらも同じロケーションに配布されるようにしてください。
- マッピングを配布する前に、AQ のマッピング演算子のバインド先である AQ オブジェクトに関連付けられた一時表を適切に配布する必要があります。
- プロキシ AQ をターゲット・システムに配布した後は、ローカルで配布したプロキシ AQ をソース・システムの AQ のサブスクライバとして登録する必要があります。8-27 ページの「サブスクライバとしての AQ の登録」を参照してください。
- AQ ソース演算子を使用して同じマッピングを同時に実行する操作はサポートされていません。
- AQ がマッピングのターゲットである場合は、マッピングを実行する前に、そのターゲット AQ のサブスクライバが 1 つ以上あることを確認してください。サブスクライバがないと、マッピング実行時にランタイム・エラーが発生します。
- AQ の最大サブスクライバ数は 1,024 です。AQ q1 の場合、q1 にバインドされたソース AQ ステージング・コンポーネントを持つマッピングの数は 1,024 未満である必要があります。
- AQ が含まれるマッピングのアップグレードを行うと、アップグレードの前に AQ に含まれていたデータを失う可能性があります。これは、アップグレード時にプロキシ AQ のサブスクライバが削除され、再び作成されるためです。

サブスクライバとしての AQ の登録

AQ がソース・システムに定義されている場合、次の手動手順に従って、配布されたプロキシ AQ をソース AQ のサブスクライバとして登録し、AQ の伝播をターゲット・ウェアハウスにスケジュールする必要があります。

1. ソース・システムからデータ・ウェアハウスへのデータベース・リンクを作成します。
2. AQ をすべてのメッセージへのサブスクライバとして登録します。
3. ソース・システムからターゲット・システムへの AQ の伝播をスケジュールします。

次の例では、AQ をサブスクライバとして登録し、その伝播をターゲット・システムにスケジュールする方法を示します。

```
declare
subscriber sys.aq$_agent;
begin
subscriber := sys.aq$_agent('subscribername','schema.queue2',0); --
schema.queue2 refers to the queue that is subscribing
dbms_aqadm.add_subscribe(queue_name=>'QUEUE1',subscriber=>subscriber,rule=>'
',transformation=>''); -- QUEUE1 refers to the queue that 'queue2 is
subscribing to
dbms_aqadm.schedule_propagation('SCHEMA.QUEUE1',NULL,SYSDATE,'','','60'); --
SCHEMA.QUEUE1 are the schema and queuename of queue1. This command starts
the message propagation from queue1 to queue2 with a latency of 60 seconds
end;
```

フラット・ファイルのマッピング演算子

フラット・ファイルのマッピング演算子はソースまたはターゲットのどちらにも使用できます。

ただし、この 2 つは同じマッピング内でも相互に排他的です。フラット・ファイルのソースとターゲットでは、コード生成言語が異なります。また、マッピングには、フラット・ファイル・リレーショナル・オブジェクトおよび変換を混在させることはできますが、この後で説明するように制約もあります。

フラット・ファイル・ソース演算子

ソースとしてのフラット・ファイルのマッピング演算子は、SQL*Loader ユーティリティを使用してフラット・ファイルの読込みを行う行セット・ジェネレータとして機能します。ただし、マッピング先がフラット・ファイル・ターゲットまたは外部表の場合は、フラット・ファイル・ソース演算子を使用できません。フラット・ファイル・ソースを含むマッピングを設計する場合は、次の演算子を使用できます。

- フィルタ演算子（8-14 ページ）
- 定数演算子（8-8 ページ）
- データ・ジェネレータ演算子（8-10 ページ）
- 順序のマッピング演算子（8-48 ページ）
- 式演算子（8-13 ページ）
- 変換演算子（8-110 ページ）
- 外部表演算子を除くその他のリレーショナル・ターゲット・オブジェクト

注意： 順序演算子、式演算子または変換演算子を使用する場合は、構成パラメータとして SQL*Loader のダイレクト・ロード設定を使用できません。

フラット・ファイル・ターゲット演算子

フラット・ファイル・ターゲットを含むマッピングでは、PL/SQL パッケージが生成されます。PL/SQL パッケージは、データを表の行にロードせず、フラット・ファイルにロードします。設計時には既存のフラット・ファイルを使用するか、新しいフラット・ファイルをターゲットとして識別し、アウトバウンド調整を使用してそのファイルを定義します。6-45 ページの「アウトバウンド調整」を参照してください。

注意： マッピングでは、一度に最大 50 のフラット・ファイル・ターゲット演算子を含むことができます。

フラット・ファイルをターゲットとして使用する場合は、フラット・ファイルと外部表の両方を理解しておくと、どちらの機能を使用するかを判断する際に役立ちます。大量のデータをロードする場合、フラット・ファイルへのロードでは、優れたパフォーマンスを実現するダイレクト・パス・ロード・オプションを使用できます。大量のデータをロードしない場合は、外部表機能で多数のリレーショナル変換を使用するほうが適しています。詳細は、3-28 ページの「外部表とフラット・ファイル演算子」を参照してください。

複数レコード・タイプのフラット・ファイルをターゲットとして使用する場合は、その中の 1 つのレコード・タイプにのみマッピングできます。同じソースのフラット・ファイルにあるすべてのレコード・タイプをロードする場合は、同じフラット・ファイルを再びターゲットとしてマッピングにドラッグ・アンド・ドロップしてから、異なるレコード・タイプにマッピングするか、ロードするレコード・タイプごとに別のマッピングを作成できます。

フラット・ファイルのマッピング演算子では、次のオプションを使用できます。

- インポート済フラット・ファイルの使用
- マッピングへの新しいフラット・ファイルのインポートとバインド
- マッピングでの新規フラット・ファイル・ソースまたはターゲットの定義

インポート済フラット・ファイルの使用

このシナリオでは、すでにインポートおよびサンプルされているフラット・ファイル・オブジェクトの使用方法について説明します。フラット・ファイル・オブジェクトのインポートとサンプルの詳細は、第 4 章「データ定義のインポート」を参照してください。

インポート済フラット・ファイルをソースまたはターゲットとして使用する手順は次のとおりです。

1. フラット・ファイルのマッピング演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。

   「マッピングファイルの追加」ダイアログが表示されます。

2. 「既存リポジトリ・ファイルからの選択およびバインド」をクリックします。
3. 使用するフラット・ファイル・オブジェクトを選択します。
4. 「OK」をクリックします。
5. 配布前に、11-13 ページの「フラット・ファイル演算子の構成」の手順および 11-2 ページの「マッピング構成のリファレンス」の手順を実行する必要があります。

マッピングへの新しいフラット・ファイルのインポートとバインド

このオプションでは、フラット・ファイルをソースまたはターゲットとして使用しますが、フラット・ファイルはまだインポートまたはサンプルされていません。フラット・ファイルはマッピングの設計時にインポートします。

新しいフラット・ファイルをマッピングにインポートおよびバインドする手順は次のとおりです。

1. フラット・ファイルのマッピング演算子をマッピング・キャンバスにドラッグ・アンド・ドロップします。

   「マッピングファイルの追加」ダイアログが表示されます。

2. 「ファイルのリポジトリへのインポートおよびバインド」をクリックします。
3. フラット・ファイルのインポート元のモジュールを選択します。
4. 「OK」をクリックします。

   「オブジェクト・インポート・ウィザード : ようこそ」ページが表示されます。フラット・ファイルのインポートの詳細は、4-29 ページの「メタデータ・インポート・ウィザードのフラット・ファイル関連機能の使用方法」を参照してください。

5. 「オブジェクト・インポート・ウィザード : ようこそ」ページで「次へ」をクリックします。

   「オブジェクト・インポート・ウィザード : オブジェクトの選択」ページが表示されます。

6. ロケーションに移動して、フラット・ファイルを選択します。
7. 「次へ」をクリックします。

   「オブジェクト・インポート・ウィザード: サマリーおよびインポート」ページが表示されます。ファイルの隣にある赤い丸印は、ファイル構造に関するメタデータ情報がないことを示しています。

   - フラット・ファイルの構造が、すでにインポートおよびサンプルされている別のファイルの構造と同じである場合は、「同一」フィールドをクリックして、ドロップダウン・リストから同じファイルを選択します。
   - このファイル構造と同じファイルがインポートまたはサンプルされていない場合は、「サンプル」をクリックし、4-31 ページの「フラット・ファイル・サンプル・ウィザードの使用方法」の手順を実行します。
8. 「終了」をクリックします。

   フラット・ファイルのマッピング演算子がマッピング・キャンバスに表示されます。
9. 配布前に、11-13 ページの「フラット・ファイル演算子の構成」の手順および11-2 ページの「マッピング構成のリファレンス」の手順を実行する必要があります。

マッピングでの新規フラット・ファイル・ソースまたはターゲットの定義

マッピングの作成時に、新規フラット・ファイル・オブジェクトを作成するには、「属性のないバインドなしマッピング・ファイルの作成」オプションを選択します。この方法を使用した場合に作成できるのは、カンマで区切られたデリミタ付きのシングル・レコード・タイプのフラット・ファイル演算子に限られます。フラット・ファイルをバインドなしにすることも、リポジトリにアウトバウンド調整を行うこともできます。

注意： アウトバウンド調整を行えるのは、フラット・ファイルがそのリポジトリで新規の場合のみです。調整の詳細は、6-41 ページの「演算子とリポジトリ・オブジェクトの調整」を参照してください。

新規のバインドなしのフラット・ファイル演算子をマッピングに使用する手順は次のとおりです。

1. フラット・ファイルのマッピング演算子をマッピング・キャンバスにドラッグ・アンド・ドロップします。

   「マッピングファイルの追加」ダイアログが表示されます。
2. 「属性のないバインドなしマッピング・ファイルの作成」をクリックします。
3. 新しいファイル名を入力します。
4. 「OK」をクリックします。

属性のないフラット・ファイルのマッピング演算子がマッピングに表示されます。

5. 新規フラット・ファイル演算子の属性を定義するには、演算子を右クリックして「編集」を選択します。「入力 / 出力」タブに属性を手動で入力するか、別の演算子からの自動マッピングを実行します。

   属性の定義方法は、6-13 ページの「演算子の編集」を参照してください。

6. フラット・ファイル演算子を右クリックし、「アウトバウンドの調整」を選択して、新規のリポジトリ・フラット・ファイルを作成します。

7. 新規フラット・ファイルを作成するフラット・ファイル・モジュールを選択します。

8. 「OK」をクリックします。

   新規のカンマ区切りフラット・ファイルがリポジトリに作成されます。

9. 配布前に、11-13 ページの「フラット・ファイル演算子の構成」の手順および 11-2 ページの「マッピング構成のリファレンス」の手順を実行する必要があります。

フラット・ファイルからのマスター / ディテール構造の抽出

マスター / ディテール構造を持つ複数レコード・タイプのファイルからデータを抽出して表にマッピングする場合、順序のマッピング演算子をマッピングに追加して、マスター・レコードとディテール・レコード間の関係をサロゲート主キーまたは外部キーの関係で保持します。マスター / ディテール・ファイル構造とは、マスター・レコードにディテール・レコードが従属している構造です。例 8-1 では、"E" で始まるレコードは従業員（Employee）情報が保存されているマスター・レコードです。"P" で始まるレコードは従業員の給与（Payroll）情報が保存されているディテール・レコードです。

例 8-1 マスター / ディテール構造を持つ複数レコード・タイプのフラット・ファイル

```
E 003715 4 153 09061987 014000000 "IRENE HIRSH" 1 08500
P 01152000 01162000 00101 000500000 000700000
P 02152000 02162000 00102 000300000 000800000
E 003941 2 165 03111959 016700000 "ANNE FAHEY" 1 09900
P 03152000 03162000 00107 000300000 001000000
E 001939 2 265 09281988 021300000 "EMILY WELLMET" 1 07700
P 01152000 01162000 00108 000300000 001000000
P 02152000 02162000 00109 000300000 001000000
```

例 8-1 では、マスター・レコードとディテール・レコード間の関係は、物理的なレコードの順序のみが継承され、給与レコードは従属している従業員レコードに関連付けられます。ただし、これがディテール・レコードをマスター・レコードに関連付ける唯一の方法である場合には、Warehouse Builder によって各レコードがターゲット表にロードされるときに、この関係が失われます。

マスター・レコードとディテール・レコード間の関係の維持

マスター・レコードとディテール・レコードが共通フィールドを共有する場合、その両レコード間の関係が維持されます。たとえば、例 8-1 の「従業員 ID」フィールドが従業員レコードと給与レコードの両方に存在する場合は、このフィールドを従業員表の主キー、給与表の外部キーとして使用すると、給与レコードを適切な従業員レコードに関連付けることができます。

ただし、共通フィールドがファイルに存在せず、マスター・レコードとディテール・レコードを結合できない場合は、順序列をマスター・ターゲットとディテール・ターゲットの両方に追加して、マスター・レコードとディテール・レコード間の関係を維持する必要があります（表 8-2 および表 8-3 を参照）。この追加する値を生成するには、順序のマッピング演算子を使用します。

表 8-2 は、8-32 ページの例 8-1 のファイルからマスター・レコードを抽出したターゲット表を示しています。このマスター・レコードのターゲット表には、従業員情報が格納されています。列 E1 ～ E10 にはフラット・ファイルから抽出したデータが格納されています。列 E11 は、マスター順序番号を格納するために追加された列です。この順序番号は従業員 1 人につき 1 ずつ増加します。

表 8-2 マスター・レコードが格納されたターゲット表

E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11
E	003715	4	153	09061987	014000000	"IRENE	HIRSH"	1	08500	1
E	003941	2	165	03111959	016700000	"ANNE	FAHEY"	1	09900	2
E	001939	2	265	09281988	021300000	"EMILY	WELSH"	1	07700	3

表 8-3 は、8-32 ページの例 8-1 のファイルからディテール・レコードを抽出したターゲット表を示しています。このディテール・レコードのターゲット表には、給与情報が格納されており、従業員 1 人につき 1 つ以上の給与レコードが存在します。列 P1 ～ P6 には、フラット・ファイルから抽出したデータが格納されています。列 P7 は、ディテール順序番号を格納するために追加された列です。各給与レコードの順序番号は、表 8-2 の対応する従業員レコードの順序番号と一致します。

表 8-3 ディテール・レコードが格納されたターゲット表

P1	P2	P3	P4	P5	P6	P7
P	01152000	01162000	00101	000500000	000700000	1
P	02152000	02162000	00102	000300000	000800000	1
P	03152000	03162000	00107	000300000	001000000	2
P	01152000	01162000	00108	000300000	001000000	3
P	02152000	02162000	00109	000300000	001000000	3

マスター / ディテール・レコードの抽出とロード

この項では、マスター / ディテール・フラット・ファイルからレコードを抽出し、そのレコードを 2 つの異なる表にロードするマッピングを作成する方法を説明します。1 つのターゲット表にはマスター・レコードを格納し、もう 1 つのターゲット表にはフラット・ファイルからのディテール・レコードを格納します。2 つの表間のマスター / ディテール関係を維持するには、順序のマッピングを使用します。

注意： この説明は、従来型パスによるロードを対象にしています。マスター / ディテール・レコードにダイレクト・パス・ロードを使用する手順は、**8-39 ページ**の「パフォーマンスを向上させるダイレクト・パス・ロード」を参照してください。

この手順では、このようなマッピングを作成するための一般的な方法を説明します。次に示す詳細な手順も参照してください。

- フラット・ファイル・ソースのインポート方法の詳細は、4-29 ページの「メタデータ・インポート・ウィザードのフラット・ファイル関連機能の使用方法」を参照してください。
- ソースとしてフラット・ファイルのマッピングを使用する方法の詳細は、8-28 ページの「フラット・ファイル・ソース演算子」を参照してください。
- 表のマッピングの使用方法の詳細は、6-11 ページの「バインド可能な演算子の追加」を参照してください。
- 順序のマッピングの使用方法の詳細は、8-48 ページの「順序のマッピング演算子」を参照してください。
- マッピングの構成方法の詳細は、11-2 ページの「マッピング構成のリファレンス」を参照してください。

マスター / ディテール・フラット・ファイルから抽出し、マスター / ディテール関係を維持する手順は次のとおりです。

1. マスター・レコードとディテール・レコードで構成されるフラット・ファイル・ソースをインポートして、サンプルします。

   ファイルのサンプル時にレコード・タイプに名前を付ける場合、図 8-11 に示すように、マスター・レコードとディテール・レコードに記述的な名前を割り当てます。これにより、今後これらのレコードを容易に識別できます。

   図 8-11 は、部門と従業員情報が含まれた複数レコード・タイプのフラット・ファイルの場合のフラット・ファイル・サンプル・ウィザードを示しています。マスター・レコード・タイプ（従業員レコード）には EmployeeMaster という名前が付き、ディテール・レコード・タイプ（給与情報）には PayrollDetail という名前が付けられています。

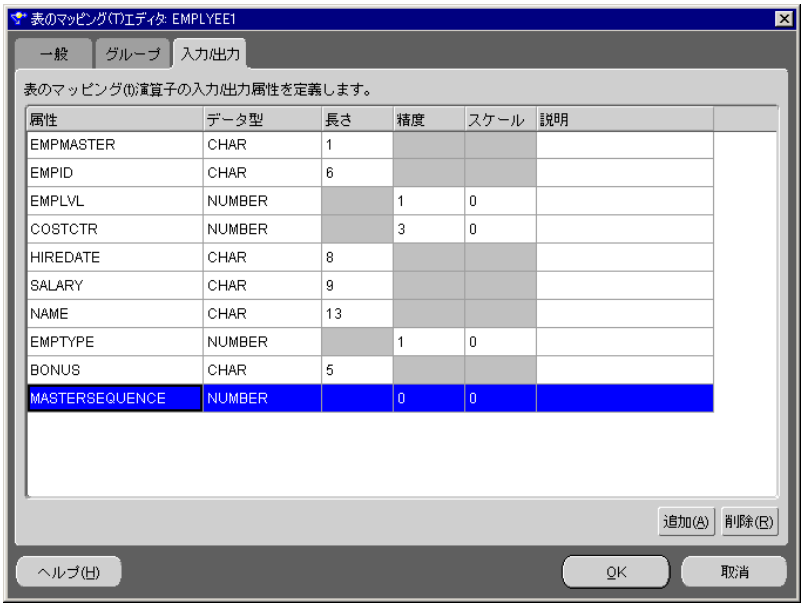
図 8-11 フラット・ファイルのマスター・レコード・タイプおよびディテール・レコード・タイプの名前

2. フラット・ファイルのマッピング演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップし、データの抽出元のマスター / ディテール・ファイルを指定します。
3. 順序のマッピング演算子をマッピング・キャンバスにドラッグ・アンド・ドロップします。
4. マスター・レコードに対する表のマッピング演算子をマッピング・キャンバスにドラッグ・アンド・ドロップします。

   以前作成した既存のリポジトリ表を選択するか、属性を指定せずにバインドなしで新しいマッピング表を作成できます。次に、ファイル演算子のマスター・レコードから必要なフィールドをすべてマスター表演算子にマッピングまたはコピーし（列も作成）、アウトバウンド調整を実行して後で表を定義できるようにします。

   図 8-12 に示すように、表には、ロードするマスター・フィールドに必要なすべての列と、順序の値をロードするための新しい数値列が含まれる必要があります。

図 8-12 マスターおよびディテール・ターゲット表への順序列の追加



5. ディテール・レコードに対する表のマッピング演算子をマッピング・キャンバスにドラッグ・アンド・ドロップします。

   以前作成した既存のリポジトリ表を選択するか、属性を指定せずにバインドなしで新しいマッピング表を作成できます。次に、ファイル演算子のマスター・レコードから必要なフィールドをすべてマスター表演算子にマッピングまたはコピーし（列も作成）、アウトバウンド調整を実行して後で表を定義できるようにします。

   表には、ロードするディテール・フィールドに必要なすべての列と、順序の値をロードするための新しい数値列が含まれる必要があります。

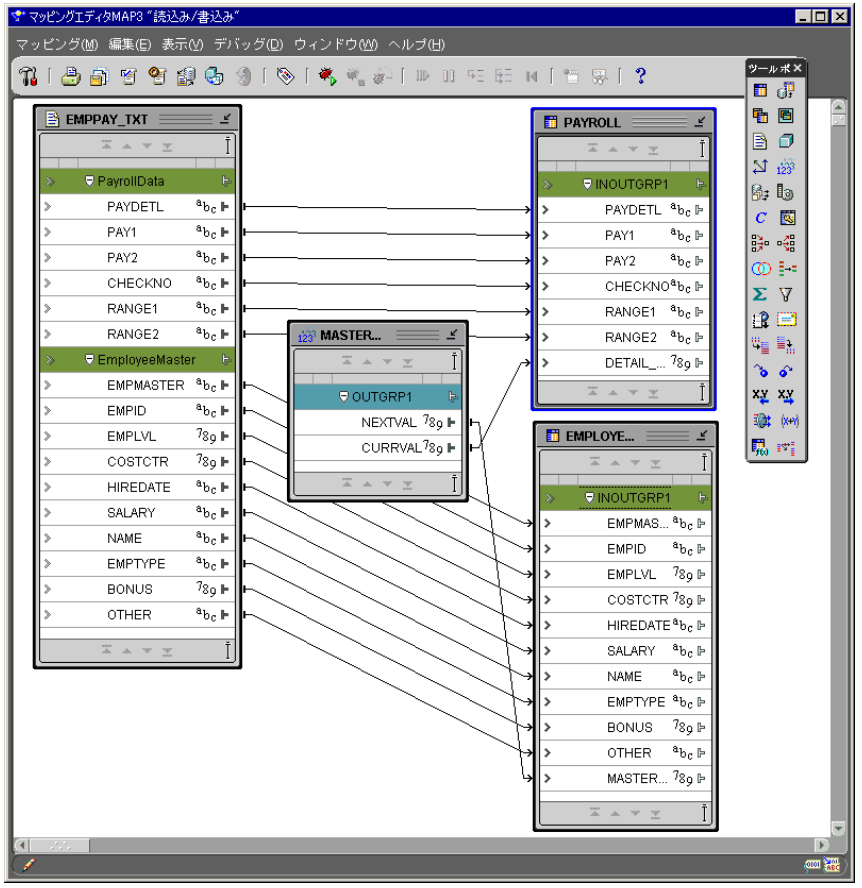
6. 図 8-13 に示すように、フラット・ファイルの必要なマスター・フィールドをすべてマスター表に、必要なディテール・フィールドをすべてディテール表にマッピングします。

7. 図 8-13 に示すように、順序のマッピングの NEXTVAL 属性をマスター表の新しい順序列にマッピングします。

8. 図 8-13 に示すように、順序のマッピングの CURRVAL 属性をディテール表の新しい順序列にマッピングします。

   図 8-13 は、マッピングが完成した状態を示しています。フラット・ファイルのマスター・フィールドはマスター・ターゲット表にマッピングされ、ディテール・フィールドはディテール・ターゲット表にマッピングされ、順序のマッピングの NEXTVAL 属性と CURRVAL 属性はマスター・ターゲット表とディテール・ターゲット表にそれぞれマッピングされています。

図 8-13 マスター / ディテール・フラット・ファイルから 2 つのターゲット表へのマッピング



9. 次のパラメータを使用して、マッピングを構成します。

ダイレクト・モード : False

エラー許可 : 0

行 : 1

後続 NULL 列 : True（すべての表）

エラーの対処方法

この項では、ファイルのエラー数に応じたエラーの対処方法について説明します。

データ・ファイルにほとんどエラーがない場合は次のように対処します。

1. 順序演算子でマッピングを作成します（8-48 ページの「順序のマッピング演算子」を参照）。
2. 次のパラメータを使用して、マッピングを構成します。

   ダイレクト・モード = False

   行 =1

   エラー許可 = 0
3. コードを生成し、SQL Loader スクリプトを実行します。

   データ・ファイルにエラーが含まれる場合、最初のエラーが発生した時点でロードが停止します。
4. データ・ファイルを修正し、次の構成値を使用して制御ファイルを再び実行します。

   CONTINUE_LOAD=TRUE

   SKIP= ロード済のレコード数

データ・ファイルにエラーがいくつか存在する場合は次のように対処します。

1. `seq_nextval` 列に基づいてマスター・レコードに主キー（PK）を作成します。
2. マスター表 PK を参照する `seq_currval` 列に基づいてディテール・レコードに外部キー（FK）を作成します。

   この場合、エラーがあるマスター・レコードは、対応するすべてのディテール・レコードで拒否されます。次の手順に従って、これらのレコードを修復します。
3. マスター・レコードを持たない、失敗したディテール・レコードをすべて削除します。
4. 不正なファイルのエラーを修正し、これらのレコードのみ再びロードします。
5. エラーが非常に少ない場合は、残りのレコードをロードし、正しい順序番号を使用して表を手動で更新できます。

6. ログ・ファイルでは、失敗したレコードをエラーで識別できます。これらのエラーは整合性制約に違反しています。次に、エラーのあるログ・ファイル・レコードの例を示します。

```
Record 9: Rejected - Error on table "MASTER_T", column "C3".
ORA-01722: invalid number
Record 10: Rejected - Error on table "DETAIL1_T".
ORA-02291: integrity constraint (SCOTT.FK_SEQ) violated - parent key not found
Record 11: Rejected - Error on table "DETAIL1_T".
ORA-02291: integrity constraint (SCOTT.FK_SEQ) violated - parent key not found
Record 21: Rejected - Error on table "DETAIL2_T".
ORA-02291: invalid number
```

データ・ファイルに常に多数のエラーが存在する場合は次のように対処します。

1. 順序のマッピング演算子を使用しないで、すべてのレコードをロードします。

   レコードを独立した表にロードします。ダイレクト・モードでデータをロードする際には、次のパラメータを使用すると、データのロードを高速化できます。

   行 >1

   エラー許可 =MAX

2. 拒否されたすべてのレコードを修正します。
3. 順序演算子を使用してファイルを再びロードします（8-48 ページの「順序のマッピング演算子」を参照）。

パフォーマンスを向上させるダイレクト・パス・ロード

マスター・レコードに一意のフィールドがある（または複数のフィールドを連結した結果、一意の識別子がある）マスター / ディテール・フラット・ファイルを使用している場合、オプションのダイレクト・パス・ロードを使用すると、ロードを高速化できます。

ダイレクト・パス・ロードの場合、各レコードのレコード番号（RECNUM）はマスター表およびディテール表に格納されます。ロード後の手順では、RECNUM を使用して、各ディテール行を、対応するマスター行の一意の識別子で更新します。

この手順では、このようなマッピングを作成するための一般的な方法を説明します。次に示す詳細な手順も参照してください。

- フラット・ファイル・ソースのインポート方法の詳細は、4-29 ページの「メタデータ・インポート・ウィザードのフラット・ファイル関連機能の使用方法」を参照してください。
- ソースとしてフラット・ファイルのマッピングを使用する方法の詳細は、8-28 ページの「フラット・ファイル・ソース演算子」を参照してください。
- 表のマッピングの使用方法の詳細は、6-11 ページの「バインド可能な演算子の追加」を参照してください。

- データ・ジェネレータ演算子の使用方法の詳細は、8-10 ページの「データ・ジェネレータ演算子」を参照してください。
- 定数演算子の使用方法の詳細は、8-10 ページの「定数演算子」を参照してください。
- マッピングの構成方法の詳細は、11-2 ページの「マッピング構成のリファレンス」を参照してください。

ダイレクト・パス・ロードを使用してマスター / ディテール・フラット・ファイルから抽出し、マスター / ディテール関係を維持する手順は次のとおりです。

1. マスター・レコードとディテール・レコードで構成されるフラット・ファイル・ソースをインポートして、サンプルします。

   ファイルのサンプル時にレコード・タイプに名前を付ける場合、8-35 ページの図 8-11 に示すように、マスター・レコードとディテール・レコードに記述的な名前を割り当てます。これにより、今後これらのレコードを容易に識別できます。

2. フラット・ファイルのマッピング演算子をマッピング・キャンバスにドラッグ・アンド・ドロップし、データの抽出元のマスター / ディテール・ファイルを指定します。
3. データ・ジェネレータ演算子および定数演算子をマッピング・キャンバスにドラッグ・アンド・ドロップします。
4. マスター・レコードに対する表のマッピング演算子をマッピング・キャンバスにドラッグ・アンド・ドロップします。

   以前作成した既存のリポジトリ表を選択するか、属性を指定せずにバインドなしで新しいマッピング表を作成し、アウトバウンド調整を実行して後で表を定義できるようにします。

   表には、ロードするマスター・フィールドに必要なすべての列と、RECNUM 値をロードするための新しい数値列が含まれている必要があります。

5. ディテール・レコードに対する表のマッピングをマッピング・キャンバスにドラッグ・アンド・ドロップします。

   以前作成した既存のリポジトリ表を選択するか、属性を指定せずにバインドなしで新しいマッピング表を作成し、アウトバウンド調整を実行して後で表を定義できるようにします。

   表には、ロードするディテール・フィールドに必要なすべての列と、RECNUM 値をロードするための新しい数値列、および対応するマスター表の行の一意の識別子で更新する列が含まれている必要があります。

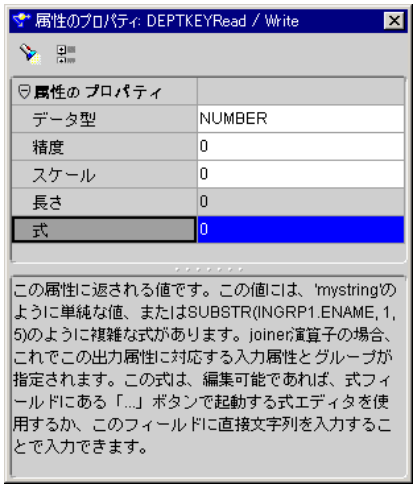
6. 8-42 ページの図 8-15 に示すように、フラット・ファイルの必要なマスター・フィールドをすべてマスター表に、必要なディテール・フィールドをすべてディテール表にマッピングします。
7. 8-42 ページの図 8-15 に示すように、データ・ジェネレータ演算子の RECNUM 属性を、マスター表およびディテール表の RECNUM 列にマッピングします。

8. 定数演算子の定数属性を追加します。

   マスター行の一意の識別子列が CHAR データ型である場合は、式 '*' を使用して定数属性を CHAR 型にします。

   マスター行の一意の識別子列が数字の場合は、式 '0' を使用して定数属性を NUMBER にします。図 8-14 は、定数属性が '0' に設定された式プロパティを示しています。この定数により、すべてのデータ行はロード済としてマークされます。

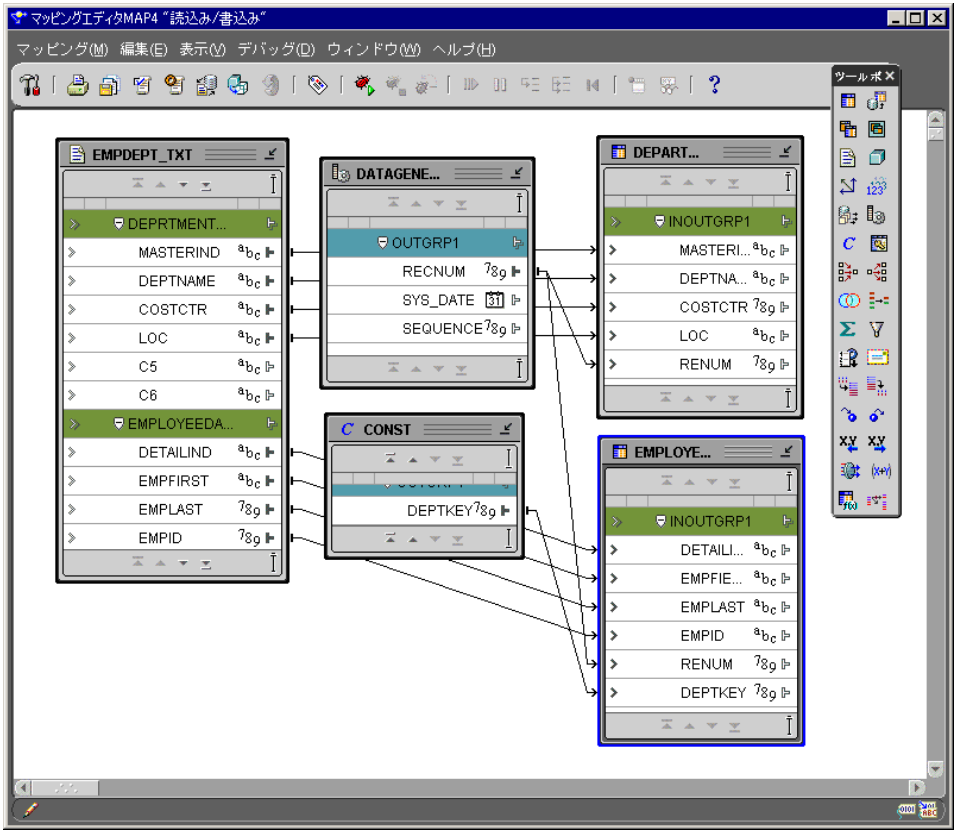
図 8-14 定数演算子のプロパティ



9. 定数演算子の定数属性をディテール表列にマッピングします。この列には、後で対応するマスター表レコードの一意の識別子が格納されます。

   図 8-15 は、マッピングが完成した状態を示しています。フラット・ファイルのマスター・フィールドはマスター・ターゲット表にマッピングされ、ディテール・フィールドはディテール・ターゲット表にマッピングされ、データ・ジェネレータ演算子の RECNUM 属性はマスター・ターゲット表とディテール・ターゲット表にそれぞれマッピングされ、定数属性はディテール・ターゲット表にマッピングされています。

図 8-15 ダイレクト・パス・ロードを使用したマスター / ディテール・フラット・ファイルからのマッピング



10. 次のパラメータを使用して、マッピングを構成します。

    ダイレクト・モード : True

    エラー許可 : 0

    後続 NULL 列 : True（すべての表）

11. マッピングを検証し、SQL*Loader スクリプトを生成した後、更新後の PL/SQL プロシージャを作成し、それを Warehouse Builder ライブラリに追加します。
12. SQL*Loader スクリプトを実行します。
13. 更新後の PL/SQL プロシージャを実行するか、スクリプトを手動で実行して、UPDATE SQL 文を実行します。

次に、生成された SQL*Loader 制御ファイルのスクリプトの例を示します。

```
OPTIONS ( DIRECT=TRUE,PARALLEL=FALSE, ERRORS=0, BINDSIZE=50000, ROWS=200,
READSIZE=65536)
LOAD DATA
CHARACTERSET WE8MSWIN1252
  INFILE 'g:¥FFAS¥DMR2.dat'
  READBUFFERS 4
  INTO TABLE "MATER_TABLE"
  APPEND
  REENABLE DISABLED_CONSTRAINTS
       WHEN
      "REC_TYPE"='P'
  FIELDS
    TERMINATED BY ','
    OPTIONALLY ENCLOSED BY '"'
    TRAILING NULLCOLS

  (
  "REC_TYPE" POSITION (1) CHAR ,
  "EMP_ID" CHAR ,
  "ENAME" CHAR ,
  "REC_NUM" RECNUM
  )

INTO TABLE "DETAIL_TABLE"
  APPEND
  REENABLE DISABLED_CONSTRAINTS
       WHEN
      "REC_TYPE"='E'
  FIELDS
    TERMINATED BY ','
    OPTIONALLY ENCLOSED BY '"'
    TRAILING NULLCOLS
     (
  "REC_TYPE" POSITION (1) CHAR ,
  "C1" CHAR ,
  "C2" CHAR ,
  "C3" CHAR ,
  "EMP_ID" CONSTANT '*',
 "REC_NUM" RECNUM
```

次に、更新後の PL/SQL プロシージャの例を示します。

```
  create or replace procedure wb_md_post_update(
     master_table varchar2
    ,master_recnum_column varchar2
    ,master_unique_column varchar2
    ,detail_table varchar2
    ,detail_recnum_column varchar2
    ,detail_masterunique_column varchar2
    ,detail_just_load_condition varchar2)
  IS
     v_SqlStmt VARCHAR2(1000);
  BEGIN
     v_SqlStmt := 'UPDATE '||detail_table||' l '||
                  ' SET l.'||detail_masterunique_column||' = (select i.'||master_
unique_column||
                  ' from '||master_table||' i '||
                  ' where i.'||master_recnum_column||' IN '||
                  ' (select max(ii.'||master_recnum_column||') '||
                  ' from '||master_table||' ii '||
                  ' where ii.'||master_recnum_column||' < l.'||detail_recnum_
column||') '||
                  ' ) '||
                  ' WHERE l.'||detail_masterunique_column||' = '||''''||detail_just_
load_condition||'''';
     dbms_output.put_line(v_sqlStmt);
     EXECUTE IMMEDIATE  v_SqlStmt;
  END;
  /
```

事後操作

マスター表およびディテール表での最初のロード後、ロードした順序値を使用して、さらにマスター表データとディテール表データを変換、更新またはマージできます。たとえば、マスター・レコードに一意の識別子（たとえば、従業員 ID）として機能する列があり、その列をキーに使用してマスター行とディテール行を結合する場合（この目的で追加した順序フィールドを使用しない場合）には、ディテール表を更新する際に、その一意の列を使用できます。次に最初のロード用に作成した順序列をドラッグ・アンド・ドロップできます。集計、フィルタ、Match-Merge などの演算子は、このような後で行う変換に役立ちます。

入力パラメータのマッピング演算子

Warehouse Builder の外部の情報を入力としてマッピングに使用するには、入力パラメータのマッピングを使用します。たとえば、入力パラメータ演算子を使用して、SYSDATE をマッピングに渡してデータをステージング領域にロードできます。同じ入力パラメータを使用して、タイムスタンプを別のマッピングに渡し、そのデータをターゲットにロードすることもできます。

マッピングを生成するときには、Warehouse Builder によって PL/SQL パッケージが作成されます。入力パラメータのマッピングは、パッケージの main プロシージャのシグネチャの一部になります。

入力パラメータのマッピングは、カーディナリティが 1 です。これにより、1 つの行セットが作成されます。この行セットを他の行セットと結合して次の演算子への入力として使用できます。

入力属性の名前は、マッピング出力属性の名前となります。マッピング・エディタで、入力パラメータのマッピング演算子の属性を接続するにはパラメータを使用します。1 つのマッピングで使用できる入力パラメータのマッピングは 1 つのみです。

入力パラメータのマッピングでは次のプロパティを指定します。

- **デフォルト値 :** 生成されたコードで、指定した属性のデフォルト値として使用される文字列値（指定した場合）。たとえば、値に '1-JUN-2001' を入力すると、DEFAULT '1-JUN-2001' を含むコードが生成されます。

  生成された PL/SQL パッケージでは、ファンクションのパラメータ宣言に続く DEFAULT 句に、入力パラメータのマッピングのデフォルト値が表示されます。たとえば、データ型が VARCHAR2 であるパラメータのマッピング param1 のデフォルト値を 'HELLO' に設定した場合、生成される PL/SQL パッケージの main ファンクションは次のようになります。

```
param1 IN VARCHAR2 DEFAULT 'HELLO'
```

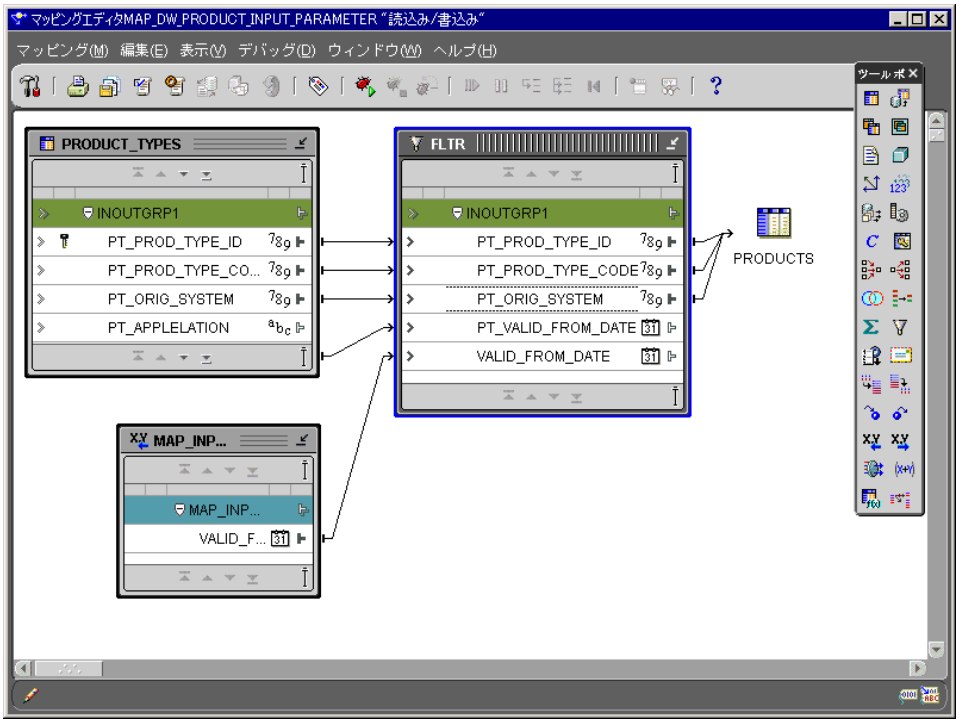
- **データ型 :** この入力パラメータのデータ型を指定します。

マッピングで入力パラメータのマッピング演算子を使用する手順は次のとおりです。

1. 入力パラメータのマッピング演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. 入力パラメータのマッピング演算子を右クリックして、「編集」を選択します。
3. 「出力」タブを選択し、「追加」をクリックして出力属性を追加します。

   属性の名前を変更して、データ型などの属性プロパティを定義できます。
4. 図 8-16 に示すように、入力パラメータのマッピング演算子の MAP_INPUT 属性を、ターゲット演算子のグループに接続します。

図 8-16 入力パラメータのマッピングが表示されたマッピング・エディタ



出力パラメータのマッピング演算子

マッピングの値を Warehouse Builder の外部のアプリケーションに渡すには、出力パラメータのマッピング演算子を使用します。マッピングを生成するときには、Warehouse Builder によって PL/SQL パッケージが作成されます。出力パラメータのマッピングは、パッケージの main プロシージャのシグネチャの一部になります。

出力パラメータのマッピングでは、入力グループを 1 つのみ使用します。1 つのマッピングで使用できる出力パラメータのマッピングは 1 つのみです。行セットに関連付けられていない属性のみを、出力パラメータのマッピングにマッピングできます。たとえば、定数、入力パラメータ、マッピング前プロセスからの出力、マッピング後プロセスからの出力には、行セットに関連付けられていない属性が含まれています。出力パラメータのマッピングは、SQL*Loader マッピングでは無効となります。

生成された PL/SQL パッケージでは、ファンクションのパラメータ宣言に続く DEFAULT 句に、出力パラメータのマッピングのデフォルト値が表示されます。たとえば、データ型が VARCHAR2 であるパラメータのマッピング `param1` のデフォルト値を `'HELLO'` に設定した場合、生成される PL/SQL パッケージの main ファンクションは次のようになります。

```
param1 OUT VARCHAR2 DEFAULT 'HELLO'
```

出力パラメータのマッピング `param1` のデータ型が VARCHAR2 である場合、生成される PL/SQL パッケージの main ファンクションは次のようになります。

```
param1 OUT VARCHAR2
```

出力パラメータのマッピングでは次のプロパティを指定します。

- **データ型 :** この出力パラメータのデータ型を指定します。
- **バウンド名 :** この出力パラメータの物理名を指定します。

注意 : 出力パラメータのマッピングは、すべての演算子にマッピングできるわけではありません。定数、入力パラメータのマッピング、マッピング前プロセスからの出力、マッピング後プロセスからの出力（戻り値を含む）をこの演算子にマッピングできます。

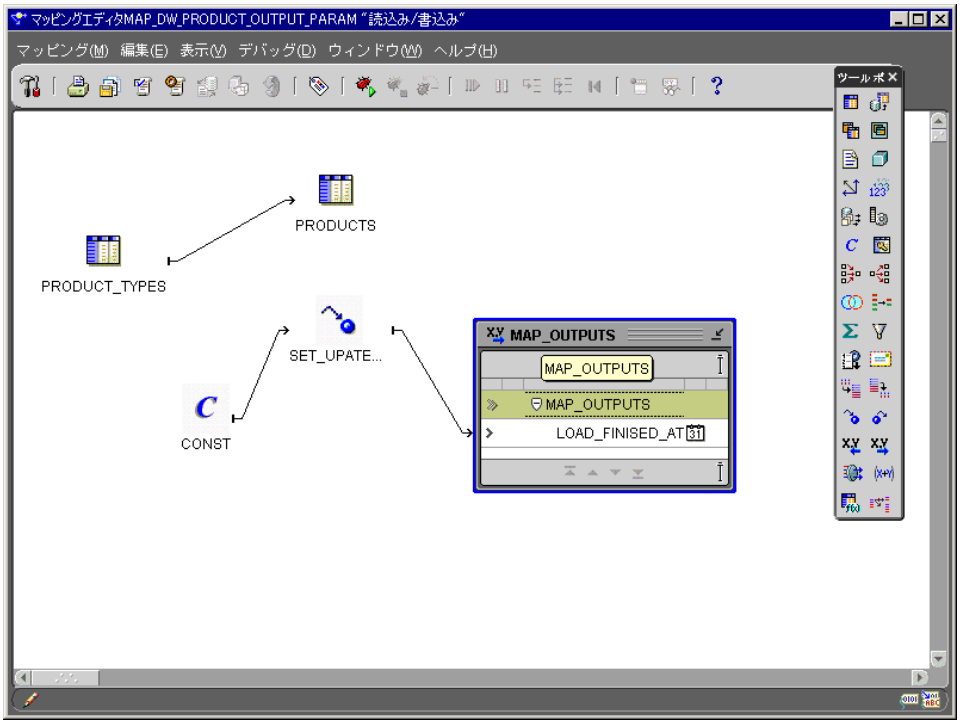
マッピングで出力パラメータのマッピング演算子を使用する手順は次のとおりです。

1. 出力パラメータのマッピング演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. 出力パラメータのマッピング演算子を右クリックして、「編集」を選択します。
3. 「入力属性」タブを選択し、「追加」をクリックして入力属性を追加します。

   属性の名前を変更して、データ型などの属性プロパティを定義できます。

   図 8-17 は、マッピングでの出力パラメータのマッピングの例を示します。

図 8-17 出力パラメータのマッピング演算子が表示されたマッピング・エディタ



順序のマッピング演算子

順序のマッピング演算子は、1 行ごとに値が増分する連続番号を生成します。たとえば、データをディメンション表にロードするときに、順序演算子を使用して、サロゲート・キーを作成できます。

順序のマッピング演算子は、ターゲット演算子の入力または他のタイプの演算子の入力に接続できます。また、他の演算子の出力と順序出力を結合することもできます。

この演算子には、次の出力属性を持つ出力グループを使用できます。

- **CURRVAL:** 現在の値から生成します。
- **NEXTVAL:** 次の値で始まり一定値ずつ増分する行セットを生成します。

順序のマッピングはいずれかのモジュールのリポジトリ順序にバインドおよび調整できます。コード・パッケージを生成する際のエラーを回避するには、順序のマッピングを含むマッピングを配布する前に、リポジトリ順序を生成して配布する必要があります。詳細は、6-11 ページの「バインド可能な演算子の追加」を参照してください。

順序のマッピングは行ベース・モードで生成します。行が選択されていない場合でも、順序の値は増分します。最後の番号から順序を開始する場合は、セット・ベース・モードまたはセット・ベース（フェイルオーバー）モードで SQL パッケージを実行しないでください。モード設定の詳細は、11-4 ページの「ランタイム・パラメータ・リファレンス」を参照してください。

順序のマッピング演算子では、次のプロパティを指定できます。

- **バウンド名 :** 生成済コード内で使用する順序データベース・オブジェクトの名前。リポジトリ・コンポーネントにあわせて順序が調整されている場合は、そのリポジトリ順序の物理名を「バウンド名」に指定します。

マッピングで順序のマッピング演算子を使用する手順は次のとおりです。

1. 順序のマッピング演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。

   「マッピング順序の追加」ダイアログが表示されます。

2. 「マッピング順序の追加」ダイアログで、順序を作成または選択できます。これらのオプションの詳細は、6-11 ページの「バインド可能な演算子の追加」を参照してください。

3. 順序をターゲット属性に接続します。

Match-Merge 演算子

Match-Merge 演算子は、データの品質に関する演算子です。これを使用すると、最初に一致したデータをマージできます。

レコードを照合する場合は、ビジネス・ルールを使用して、表内のどのレコードが同じデータを参照しているかを判別します。レコードをマージする場合は、一致したレコードのデータを 1 件のレコードに統合します。

この項では、マッピングに Match-Merge 演算子を使用する方法とその例を説明します。Match-Merge 演算子を Name and Address 演算子とともに使用すると、Name and Address データで一意の世帯を識別する世帯検索プロセスがサポートされます。

例 : 顧客データの一致およびマージ

Match-Merge 演算子を使用して、顧客のメーリング・リストを管理する方法を考慮してみます。まず一致を使用して、10,000 行ある顧客データの表から同じ人物を参照しているレコードを検索します。この場合、同じような姓と名を持つレコードを選別する一致ルールを定義できます。一致によって、同じ人物を参照している行を 5 つ検出できたとします。これらのレコードを 1 件の新しいレコードにマージできます。たとえば、一致した 5 件のレコードから住所が最も長いレコードの値を保持するマージ・ルールを作成できます。これで、新しくマージした表には、顧客 1 人にレコードが 1 件ずつ表示されます。

表 8-4 では、同じ人物を参照している複数のレコードを示します（Match-Merge 演算子の使用前）。

表 8-4 サンプル・レコード

行	名	姓	SSN	アドレス	住居	郵便番号
1	Jane	Doe	NULL	123 Main Street	NULL	22222
2	Jane	Doe	111111111	NULL	NULL	22222
3	J.	Doe	NULL	123 Main Street	Apt 4	22222
4	NULL	Smith	111111111	123 Main Street	Apt 4	22222
5	Jane	Smith-Doe	111111111	NULL	NULL	22222

表 8-5 では、Match-Merge 演算子を使用した結果作成された Jane Doe の 1 件のレコードを示します。新規レコードは、サンプル・レコードの複数の行からデータが取得されています。

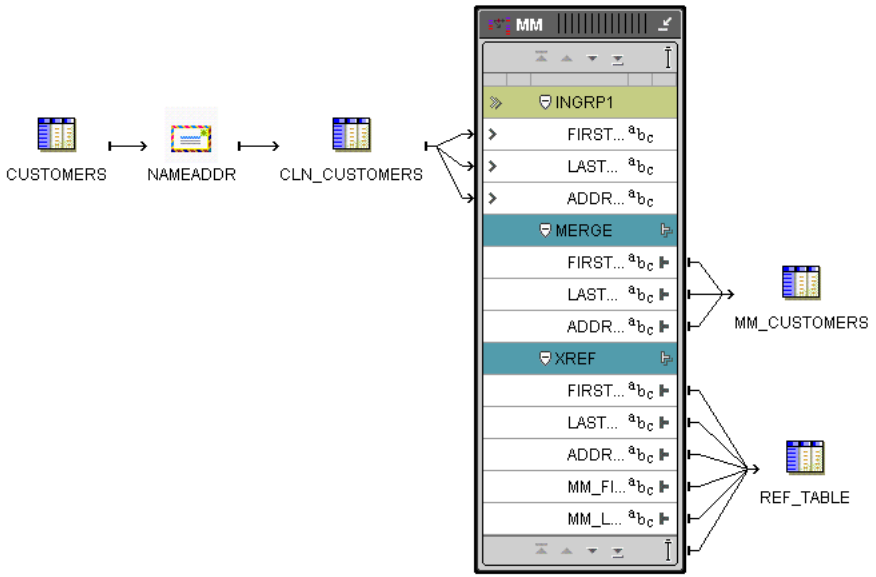
表 8-5 Match-Merge の結果

名	姓	SSN	アドレス	住居	郵便番号
Jane	Doe	111111111	123 Main Street	Apt 4	22222

Match-Merge 演算子を使用したマッピングの設計

図 8-18 では、Match-Merge 演算子を使用して設計できるマッピングを示します。この図からわかるように、この Match-Merge 演算子の前には、Name and Address 演算子である NAMEADDR とステージング表である CLN_CUSTOMERS が使用されています。マッピングは、Name and Address 演算子を使用してもしなくても設計できます。時間がかかる Match-Merge 演算子の実行前に、データをクリーンで標準化された状態にする場合は、Match-Merge 演算子の前に Name and Address 演算子を使用することをお薦めします。

図 8-18　Match-Merge 演算子を使用したマッピング



Name and Address 演算子を使用する場合も使用しない場合も、マッピングを設計する際には次の事項に注意してください。

- **PL/SQL 出力 :** Match-Merge 演算子では 2 つの出力を生成できますが、どちらも PL/SQL 出力のみです。MERGE グループには、マージされたデータが含まれます。XREF グループは、マージ・プロセスを記録するために設計できるオプションのグループです。
- **行ベースのオペレーティング・モード :** Match-Merge 演算子でレコードを照合する場合は、ソースの各行が他の行と 1 つずつ比較され、行ベースのコードのみが生成されます。そのため、これらのマッピングは行ベース・モードでのみ実行されます。

- **Match-Merge の前に使用する SQL ベースの演算子 :** Match-Merge 演算子では、PL/SQL 出力のみが生成されます。SQL コードのみを生成する演算子を使用する場合は、マッピングの設計時にその演算子を Match-Merge 演算子よりも前に使用する必要があります。たとえば、結合、キー参照、集合などの演算子は Match-Merge 演算子よりも前に使用する必要があります。Match-Merge 演算子の後にセット・ベースのコードを生成する演算子を使用して設計すると、そのマッピングは無効になり、Warehouse Builder でコードは生成されません。
- **SQL 入力 :** 例外が 1 つありますが、Match-Merge 演算子では SQL 入力が必要です。Match-Merge の前に、Name and Address 演算子などの PL/SQL 出力のみを生成する演算子を使用する場合は、あらかじめそのデータをステージング表にロードしておく必要があります。
- **Match-Merge 演算子からのデータの詳細化 :** データを詳細化するには、Match-Merge 演算子からの XREF 出力を別の Match-Merge 演算子にマッピングします。この使用例は、Match-Merge 演算子の SQL 入力ルールに対する 1 つの例外です。別の設計要素がある場合には、2 番目の Match-Merge 演算子が PL/SQL を受け入れます。詳細は、8-70 ページの「Match-Merge 演算子からのデータの詳細化」を参照してください。

Match-Merge 演算子の使用方法

Match-Merge 演算子を使用する際には次のオプションがあります。

- **新しい Match-Merge 演算子を定義する :** ツールボックスの Match-Merge 演算子をマッピングにドラッグ・アンド・ドロップします。マッピング・エディタでウィザードが起動します。
- **既存の Match-Merge 演算子を編集する :** 演算子を右クリックして、「編集」を選択します。マッピング・エディタで Match-Merge エディタが起動します。

演算子ウィザードまたは演算子エディタのどちらを使用する場合も、次の各ページで指定します。

- 一般
- グループ
- 接続の入力
- 入力属性
- マージ出力
- 相互参照出力
- 一致 bin
- 一致ルール
- マージ・ルール

一般

「一般」ページで、演算子の名前と説明（オプション）を指定します。ウィザードでは、Match-Merge 演算子に "MM" というデフォルト名が付きます。

グループ

定義上、Match-Merge 演算子には 1 つの入力グループと 2 つの出力グループがあります。このグループの名前を変更して説明（オプション）を追加できますが、Match-Merge 演算子のグループを追加または削除することはできません。入力グループのデフォルト名は INGRP1 です。出力グループのデフォルト名は MERGE と XREF です。

「接続の入力」ページで INGRP1 に属性を割り当て、次にこれらの属性を「入力属性」ページで編集します。「マージ出力」ページでは MERGE グループの属性を定義します。オプションで、「相互参照出力」ページで XREF グループに属性を割り当てることもできます。

接続の入力

「接続の入力」ページで、属性を選択し、演算子にコピーおよびマッピングします。

演算子の「接続の入力」ページで指定する手順は次のとおりです。

1. 左側のパネルで、グループ全体または個別の属性を選択します。

   名前を使用して特定の属性またはグループを検索するには、「検索」にテキストを入力し、「実行」をクリックします。次の一致文字列を検索するには、「実行」を再度クリックします。

   複数のグループまたは属性を選択するには、[Shift] キーを押しながら選択します。異なるグループの属性を選択する場合は、結合子演算子または集合演算子を使用してグループを結合する必要があります。

2. ページ中央にある「>」ボタンを使用して、選択した項目をウィザードの右側に移動します。

   入力接続リストからグループまたは属性を移動するには「<」ボタンを使用します。Warehouse Builder では、選択した項目が入力グループから削除され、ソース演算子と現行の演算子間のマッピング線も削除されます。

入力属性

「入力属性」ページで、「接続の入力」タブまたはウィザード・ページで選択した属性を変更します。

Match-Merge の「入力属性」ページでは、次のタスクを実行できます。

- **属性のプロパティを変更する :** 属性名、データ型、長さ、精度、スケールを変更できます。
- **説明（オプション）を追加する :** 入力属性の説明を入力します。
- **属性を追加する :** 「追加」ボタンを使用して、「接続の入力」タブまたはウィザード・ページで選択していない入力属性を追加します。

マージ出力

「マージ出力」タブで、出力 MERGE グループの属性を指定します。MERGE グループは、選択した属性を使用して、統合されたレコードを生成します。

相互参照出力

「相互参照出力」ページで、XREF グループの属性をオプションで選択します。Match-Merge 演算子では、デフォルトで XREF グループが作成されますが、オプションでそのグループに属性を追加したり、空にしたりできます。

XREF グループは、マージ・プロセスを記録するために定義できるオプションのグループです。これにより、元のデータ・セットと新しくマージしたデータ・セット間の外部キー関係を作成できます。XREF グループの属性は、マージした各行に対応するソース行を記録する表に渡すことができます。

INGRP1 の各行は XREF グループの行に対応します。XREF グループを設計するには、左側の「ソース属性」から元の属性値とマージした属性を選択します。Warehouse Builder では、「ソース属性」にデフォルトの接頭辞 "MM_" が付いたマージ済属性が表示されます。接頭辞を変更するには、ページの左下隅にある「接頭辞の設定」を使用します。

一致 bin

「一致 bin」ページで、一致する件数を適度な数に制限します。Warehouse Builder で行を照合するときには、同じグループ内で各行が他の行と 1 つずつ比較されていきます。したがって、Warehouse Builder では全体のデータ・セットではなく、グループ内のみで一致するデータが検索されるため、パフォーマンスが大幅に向上します。

「一致 bin」の定義で、レコードを適度な数のグループに分離できますが、一致するはずのレコードまでが分離されるようなことは避ける必要があります。グループに選択する属性は、データによって異なります。たとえば、100 万行の顧客アドレスの表がある場合には、部分的な町村名、都市名および郵便番号でデータをグループ化できます。

理想的には、グループごとのレコード数を 2000 件未満にとどめておきます。Warehouse Builder で実行する比較の件数は、次の計算式に基づきます。

```
n=(b*(b-1))/2
```

n は比較の件数を示し、b は bin 内のレコード数を示します。

5 件のレコードを照合するには、Warehouse Builder で 10 回比較を行う必要があります。50 件のレコードを照合するには、Warehouse Builder で 1,225 回比較を行う必要があります。500 件のレコードを照合するには、Warehouse Builder で 124,750 回比較を行う必要があります。

一致ルール

演算子の 1 つの属性または複数の属性に対して一致ルールを定義できます。「一致ルール」タブの上部で、一致ルールを作成します。「一致ルール」タブの下部では、各一致ルールの詳細を指定します。

複数の一致ルールを作成すると、2 行がこれらの一致ルールのいずれかを満たす場合に、2 行の一致と判断されます。つまり、Warehouse Builder では OR 論理を使用して複数の一致ルールが評価されます。この OR 論理は、「一致ルール」タブの左端の列にある「OR」アイコンによって示されます。詳細は、8-58 ページの「一致の概念について」を参照してください。

Warehouse Builder では、追加した一致ルールごとに、位置番号が割り当てられ、MM_MA_0 などのデフォルトの名前が作成されます。このルールは編集して、ルール・リストの新しい位置にドラッグできます。一致ルールは、アクティブ一致ルールまたは受動一致ルールとして指定できます。Warehouse Builder で受動ルールが実行されるのは、受動ルールをカスタム一致ルールでコールした場合のみです。8-58 ページの表 8-7 に示すような、ルール・タイプを割り当てます。ルール・タイプを選択すると、「一致ルール」タブの下部がアクティブになるので、一致ルールの詳細を入力できます。

条件付きルール・タイプの詳細を追加する場合は、1 つ以上の属性の詳細を追加するように求められます。複数の属性の詳細を追加すると、左端の列に「AND」アイコンが表示されます。これは、すべての条件の詳細が満たされた場合にのみ、Warehouse Builder で行を一致させることを意味します。

アクティブ一致ルール

Warehouse Builder では、アクティブとして指定した場合に一致ルールが実行されます。複数の一致ルールがアクティブな場合は、一致が検出されるか、すべてのルールが評価されるまで、一致ルールが順番に評価されていきます。一致が検出されると、そのレコードは一致とみなされます。

受動一致ルール

Warehouse Builder では、受動一致ルールは直接実行されません。そのかわり、アクティブなカスタム一致ルールを作成して、受動一致ルールをコールできます。定義する各一致ルールで、対応するファンクションが一致ルールと同じ名前で作成されます。カスタム・ルールは、このファンクションを使用して受動一致ルールをコールできます。

カスタム一致ルール

一致ルールのカスタム比較アルゴリズムを作成するには、このエディタを使用します。「編集」を選択して、カスタム一致ルール・エディタを起動します。左側のナビゲーション・ツリーで必要なファンクションとパラメータをダブルクリックするか、右側の実装エディタまでドラッグ・アンド・ドロップします。アクティブ一致ルール、受動一致ルール、ファンクション、および THIS_、THAT_ などのパラメータを参照するカスタム一致ルールを記述できます。この THIS_、THAT_ は、比較する INGRP1 の 2 件のレコードを表します。

カスタム・ルールを検証するには、カスタム一致ルール・エディタで「テスト」メニューより「検証」を選択します。

マージ・ルール

「マージ・ルール」タブで、マージしたレコードに属性の値を選択します。

「マージ・ルール」タブの上部で、マージ・ルールを作成します。「マージ・ルール」タブの下部では、各マージ・ルールの詳細を指定します。

Warehouse Builder では、追加したマージ・ルールごとに、位置番号が割り当てられ、MM_ME_0 などのデフォルトの名前が作成されます。Warehouse Builder では、この位置番号の順序に従って、マージ・ルールが実行されます。このルールは編集して、ルール・リストの新しい位置にドラッグできます。表 8-6 に示すような、ルール・タイプを割り当てます。

表 8-6 マージ・ルールのタイプ

マージ・ルールのタイプ	単一または複数の属性の選択	説明
すべて	単一	属性を選択すると、その属性に最初の空白以外の値が割り当てられます。
任意のレコード	複数	タブの下部で複数の属性を選択すると、一致した行に基づいてその属性が割り当てられます。
コピー	単一	属性を選択すると、別のマージ済属性の値が割り当てられます。
カスタム	単一	属性を選択すると、記述した PL/SQL コードに基づいて値が割り当てられます。詳細は、カスタム・マージ・ルールを参照してください。
カスタム・レコード	複数	タブの下部で複数の属性を選択します。Warehouse Builder では、記述した PL/SQL コードに基づいて値が割り当てられます。詳細は、カスタム・マージ・ルールを参照してください。
一致 ID	単一	Match-Merge 演算子からの XREF グループを、別の Match-Merge 演算子の入力にマッピングする場合に、このルールを使用します。タブの下部で順序を選択し、一致 ID を生成します。

表 8-6 マージ・ルールのタイプ（続き）

マージ・ルールのタイプ	単一または複数の属性の選択	説明
最小 / 最大レコード	複数	タブの下部で複数の属性を選択します。Warehouse Builder では、選択する属性の相対値が含まれる一致レコードからマージ属性値が割り当てられます。
最小 / 最大	単一	選択した属性の順序に基づいて一致候補から最初の値を選択します。
ランク	単一	Warehouse Builder では、定義したランク式に基づいてレコードに値が割り当てられます。属性のグループへの移入に使用するレコードを選択します。このレコードを選択すると、一致したレコード・セットにあるすべてのレコードのどの属性にもアクセスできます。
レコードのランク	複数	タブの下部で、複数の属性を選択します。Warehouse Builder では、定義したランク式に基づいて値が割り当てられます。
順序	単一	タブの下部で、プロジェクトに定義された順序から順序を選択します。

ルール・タイプを選択すると、「マージ・ルール」タブの下部がアクティブになるので、マージ・ルールの詳細を入力できます。

マージ・ルールを定義する場合は、マージしたレコードのすべての属性に 1 つのルールを定義するか、属性ごとにルールを定義できます。たとえば、マージしたレコードが顧客レコードの場合、そのレコードには ADDRESS1、ADDRESS2、CITY、STATE、ZIP などの属性があります。その場合は、属性ごとに 1 つずつ、5 つのルールを記述できます。または、全部で 5 つの属性を同じ一致行から取り出せるように、1 つのルールを記述できます。

1 つの属性にルールを 1 つずつ記述する場合は、ページの上部にある「属性」リスト・ボックスで属性を指定します。ルールの詳細が必要な場合は、タブの下部で指定します。

複数の属性に 1 つのルールを記述する場合は、ページの上部にある「属性」リスト・ボックスは使用できません。タブの下部で詳細を定義する必要があります。

カスタム・マージ・ルール

記述した PL/SQL コードに基づいて値が割り当てられ、選択している属性のタイプが返されるカスタム・マージ・ルールを作成するには、カスタム・マージ・ルール・エディタを使用します。

「編集」を選択して、カスタム・マージ・ルール・エディタを起動します。左側のナビゲーション・ツリーで必要なファンクションと属性をダブルクリックするか、右側の実装エディタまでドラッグ・アンド・ドロップします。ソース属性、その他のマージ属性およびファンクションを参照するカスタム・マージ・ルールを記述できます。

カスタム・ルールを検証するには、カスタム・マージ・ルール・エディタで「テスト」メニューより「検証」を選択します。

次に、属性のカスタム・マージ・ルールの例を示します。

```
BEGIN
RETURN M_MATCHES(1)."TAXID";
END;
```

次に、レコードのカスタム・マージ・ルールの例を示します。

```
BEGIN
RETURN M_MATCHES(1);
END;
```

一致の概念について

Warehouse Builder でレコードを照合する場合は、1 つまたは複数の一致ルールを定義できます。複数の一致ルールを作成すると、2 行がこれらの一致ルールのいずれかを満たす場合に、2 行の一致と判断されます。つまり、Warehouse Builder では OR 論理を使用して複数の一致ルールが評価されます。表 8-7 では、指定できる一致ルールを示します。

表 8-7 一致ルールのタイプ

一致ルール	説明
アドレス	郵便アドレスに基づいてレコードを照合します。アドレスを構成する属性を指定します。各属性に入力ロールを割り当てます。詳細は、8-61 ページの「アドレス一致ルール」を参照してください。
すべて一致	一致 bin 内のすべての行を照合します。
条件付き	選択したアルゴリズムに基づいて行を照合します。Warehouse Builder では、すべての条件の詳細が満たされる場合にのみ、行が一致します。詳細は、8-64 ページの「条件付き一致ルール」を参照してください。
カスタム	カスタム比較アルゴリズムを作成します。「編集」を選択して、カスタム一致ルール・エディタを起動します。詳細は、「カスタム一致ルール」を参照してください。

表 8-7 一致ルールのタイプ（続き）

一致ルール	説明
会社	ビジネス名に基づいてレコードを照合します。会社名を構成する属性を指定します。各属性に入力ロールを割り当てます。詳細は、8-66 ページの「会社一致ルール」を参照してください。
一致なし	一致 bin 内で行を照合しないことを指定します。
人名	人名に基づいてレコードを照合します。人名を構成する属性を指定します。各属性に入力ロールを割り当てます。詳細は、8-66 ページの「人名一致ルール」を参照してください。
重み	属性に割り当てるスコアに基づいて行を照合します。Warehouse Builder では、類似度のアルゴリズムを使用して各属性を比較します。類似度のアルゴリズムは、行間の類似度を表す 0 ～ 100 のスコアを返します。2 行が一致すると判断されるには、合計した数が指定した合計スコアよりも大きい必要があります。詳細は、8-69 ページの「重み一致ルール」を参照してください。

複数の一致ルールの例

次の例では、OR 論理を使用して複数の一致ルールが評価される方法を説明します。

「一致ルール」タブの上部で、表 8-8 に示すように、2 つの一致ルールを作成します。

表 8-8 2 つの一致ルール

名前	位置	ルール・タイプ	使用方法	説明
Rule_1	1	条件付き	アクティブ	SSN を照合
Rule _2	2	条件付き	アクティブ	姓と PHN を照合

タブの下部で、表 8-9 に示すように、Rule_1 に詳細を割り当てます。

表 8-9 Rule_1 の詳細

属性	位置	アルゴリズム	類似度のスコア	空白の一致
SSN	1	完全	0	いずれかが空白の場合は一致しない

Rule_2 には、表 8-10 に示すように、詳細を割り当てます。

表 8-10 Rule_2 の詳細

属性	位置	アルゴリズム	類似度のスコア	空白の一致
姓	1	完全	0	いずれかが空白の場合は一致しない
PHN	2	完全	0	いずれかが空白の場合は一致しない

表 8-11 に示すようなデータがあるとします。

表 8-11 データ例

行	名	姓	PHN	SSN
A	John	Doe	650-123-1111	NULL
B	Jonathan	Doe	650-123-1111	555-55-5555
C	John	Dough	650-123-1111	555-55-5555

Rule_1 に従うと、行 B と C が一致します。Rule_2 に従うと、行 A と B が一致します。Warehouse Builder では、OR 論理を使用して一致ルールが処理されるため、3 つのレコードはすべて一致することになります。

推移一致の例

一般のルールでは、A が B と一致し、かつ B が C と一致する場合は、A は C と一致します。ここで、表 8-12 に示すような類似度に基づいて、条件付き一致ルールを割り当てます。

表 8-12 条件付き一致ルール

属性	位置	アルゴリズム	類似度のスコア	空白の一致
姓	1	類似度	80	いずれかが空白の場合は一致しない

表 8-13 に示すようなデータがあるとします。

表 8-13 サンプル・データ

行	名	姓	PHN	SSN
A	John	Jones	650-123-1111	NULL
B	Jonathan	James	650-123-1111	555-55-5555
C	John	Jamos	650-123-1111	555-55-5555

Jones は類似度 80 で James と一致し、James は類似度 80 で Jamos と一致します。Jones と Jamos は類似度が 80 より小さい 60 であるため一致しません。しかし、Jones は James と一致し、James は Jamos と一致するため、これらの 3 つのレコードはすべて一致することになります。

アドレス一致ルール

アドレス一致ルールを使用して、郵便アドレスに基づいてレコードを照合します。アドレスによる照合が最も有効なのは、Match-Merge 演算子の前に Name and Address 演算子を使用してアドレス・データを修正する場合です。Name and Address 演算子は、郵便照合データベースに存在するアドレスを識別し、そのレコードに「検出済」フラグを付けます。Match-Merge 演算子では、「検出済」ロールの付いたアドレスであれば、速く処理できます。これは、フラグが付いているデータは構文的に間違いがなく、違反がなく、存在しているデータであると認識できるからです。

アドレス一致ルールを定義する手順は次のとおりです。

1. 「一致ルール」タブで、「ルール・タイプ」に「アドレス」を選択します。

   ページの下部には、「アドレス属性」タブと「詳細」タブが表示されます。

2. 「アドレス属性」タブの左側のパネルで、第 1 アドレスを示す属性を選択し、「>」ボタンをクリックします。

3. 「ロール」を右クリックし、その属性を「第 1 アドレス」に指定します。

   この手順は必ず実行してください。「第 1 アドレス」ロールを割り当てない場合、一致ルールは無効になり、「詳細」タブにアクセスできません。

4. 必要に応じて他の属性を追加して、そのロールを指定します。Name and Address 演算子を使用してアドレス・データをクレンジングした場合は、「検出済」ロールを該当する属性に割り当てます。割り当てるロールのタイプは、表 8-14 を参照してください。

5. 「詳細」タブを選択して、表 8-15 に示すようなオプションを選択します。

表 8-14 では、アドレス一致ルールに割り当てるアドレス・ロールを示します。

表 8-14 アドレス・ロール

ロール	説明
第 1 アドレス	一致ルールを有効にするには、このロールを 1 つの属性に割り当てる必要があります。第 1 アドレスには 100 Main Street のような町村や、PO Box 100 のような私書箱を使用できます。
区画番号	第 1 アドレスを照合するアドレスの場合、演算子は部屋番号、階数、アパート番号などの区画番号を比較します。区画番号が空白の場合、演算子はそれを一致とみなします。1 つの区画番号のみが空白の場合は、「空白の第 2 アドレスの一致」オプションを選択しないかぎり、一致とみなされません。
私書箱	第 1 アドレスの一部が私書箱の場合、演算子はその私書箱番号を比較します。第 1 アドレスが町村である場合は、私書箱番号は空白です。
二重第 1 アドレス	第 1 アドレスを照合するアドレスの場合、演算子は二重第 1 アドレスを比較します。二重第 1 アドレスは、町村と私書箱の両方が含まれるアドレスです。
二重区画番号	「二重第 1 アドレス」ロールを割り当てる場合にのみ、このロールも割り当てます。演算子は、一方のレコードの二重区画番号と、他方のレコードの区画番号と二重区画番号を比較します。区画番号は、一方が空白の場合でも、両方が空白の場合でも一致とみなされます。
二重私書箱	「二重第 1 アドレス」ロールを割り当てる場合にのみ、このロールも割り当てます。演算子は、一方のレコードの二重私書箱と、他方のレコードの私書箱と二重私書箱を比較します。
市区町村	「州（都道府県）」ロールを割り当てる場合にのみ、このロールも割り当てます。Name and Address 演算子を使用してアドレス・データをクレンジングしたかどうかに応じて、このロールの照合動作が異なります。修正されていないアドレス・データでは、演算子は市区町村を比較します。修正されたアドレスでは、演算子は、郵便番号が一致しない場合にのみ市区町村を比較します。市区町村と州が一致する場合、演算子はアドレス・ロールを比較します。市区町村は、両方が空白の場合に一致、一方のみが空白の場合には不一致とみなされます。
州（都道府県）	「市区町村」ロールを割り当てる場合にのみ、このロールも割り当てます。Name and Address 演算子を使用してアドレス・データをクレンジングしたかどうかに応じて、このロールの照合動作が異なります。修正されていないアドレス・データでは、演算子は州を比較します。修正されたアドレスでは、演算子は、郵便番号が一致しない場合にのみ州を比較します。市区町村と州が一致する場合、演算子はアドレス・ロールを比較します。州は、一方のみが空白ではなく、両方が空白の場合に一致とみなされます。

表 8-14 アドレス・ロール（続き）

ロール	説明
郵便番号	Name and Address 演算子を使用してアドレス・データをクレンジングしたかどうかに応じて、このロールの照合動作が異なります。修正されていないアドレス・データでは、演算子は郵便番号を使用しません。修正されたアドレスでは、演算子は各郵便番号のみを比較します。郵便番号が一致する場合、演算子はアドレス・ロールを比較します。郵便番号が一致しない場合、演算子は市区町村と州を比較し、第 1 アドレスなどのアドレス・ロールを比較する必要があるかどうかを判断します。
検出済	このロールは、Name and Address 演算子を使用してクレンジングと標準化を行ったアドレス・データに割り当てます。Name and Address 演算子は、国内郵便照合データベースの一部としてアドレスを識別した場合に、そのレコードに「検出済」フラグを付けます。

表 8-15 では、アドレス・ロールに割り当てるオプションを示します。

表 8-15 アドレス・ロールのオプション

オプション	説明
異なる第 2 アドレスの許可	区画番号が異なっても、アドレスは一致とみなされます。
空白の第 2 アドレスの一致	1 つの区画番号が空白でも、アドレスは一致とみなされます。
町村または私書箱の一致	演算子は、町村または私書箱のいずれかが一致した場合に、レコードを一致とみなします。
アドレス行の類似度	演算子はアドレス行にある英数字以外の文字や空白をすべて無視し、類似度を計算します。割り当てたスコア以上の類似度を持つレコードは、一致とみなされます。
最終行の類似度	演算子は市区町村と州の英数字以外の文字や空白をすべて無視し、類似度を計算します。割り当てたスコア以上の類似度を持つレコードは、一致とみなされます。

条件付き一致ルール

複数の属性比較を 1 つの複合ルールに結合するには、条件付き一致ルールを使用します。複数の属性を比較して、レコードを一致させるには、すべての比較において true である必要があります。

条件付き一致ルールを定義する手順は次のとおりです。

1. 「一致ルール」タブの上部で、「ルール・タイプ」に「条件付き」を選択します。

   タブの下部に「詳細」セクションが表示されます。

2. 「追加」をクリックし、属性を追加して選択します。
3. 「アルゴリズム」で、表 8-16 に示したオプションのいずれかを選択します。
4. 「類似度」または「標準化された類似度」を選択する場合は、類似度のスコアを指定します。
5. 「空白の一致」では、演算子が空白値を処理する方法を指定します。

表 8-16 一致ルールのアルゴリズム

アルゴリズム	説明
完全	属性の値が完全に一致する場合に、属性が一致します。たとえば、"Dog" と "dog!" は一致しません。これは、2 つ目の文字列の先頭が大文字でなく、余分な文字も含まれているからです。文字列以外のデータ型の場合、これは比較が可能な唯一のタイプです。
標準化された完全一致	演算子は、完全一致を比較する前に、属性の値を標準化します。標準化によって、比較の際に大 / 小文字の区別、空白、英数字以外の文字が無視されます。このアルゴリズムでは、"Dog" と "dog!" は一致します。
類似度	類似度のスコアを 0 ～ 100 の範囲で入力します。2 つの属性の類似度がそのスコア以上の場合、属性値は一致とみなされます。類似度アルゴリズムは、2 つの文字列間の編集差異を計算します。編集差異とは、1 つの文字列を別の文字列に変換する際に必要となる削除、挿入または置換の数を意味します。類似度 100 は、2 つの値が同じであることを示します。ゼロの類似度は、まったく類似性がないことを示します。たとえば、文字列 "tootle" と文字列 "tootles" を比較した場合の編集差異は 1 です。"tootles" の文字列の長さは 7 です。したがって、類似度値は、6/7*100、つまり 85 になります。
標準化された類似度	演算子は、類似度を決める類似度アルゴリズムを使用する前に、属性の値を標準化します。標準化によって、比較の際に大 / 小文字の区別、空白、英数字以外の文字が無視されます。
Soundex	演算子は、データを Soundex 表現に変換し、テキスト文字列を比較します。Soundex 表現が一致する場合、2 つの属性値は一致します。

表 8-16 一致ルールのアルゴリズム（続き）

アルゴリズム	説明
不完全な名前	文字列属性の値が一致とみなされるのは、一方の属性が他方の属性の中に含まれており、しかも最初の単語から一致する場合です。たとえば、"Midtown Power" は "Midtown Power and Light" と一致しますが、"Northern Midtown Power" とは一致とみなされません。比較の際には大 / 小文字の区別、英数字以外の文字が無視されます。このアルゴリズムでは、名前の一部を照合する前に、文字列全体に「標準化された完全一致」比較が実行されます。
略称	文字列の属性値が一致とみなされるのは、一方の文字列に他方の文字列の略語が含まれている場合です。略語を検出する前に、文字列全体に「標準化された完全一致」比較が実行されます。比較の際には大 / 小文字の区別、英数字以外の文字が無視されます。 略語は、単語単位に次の方式で検出されます。単語全体が長いほうに、単語全体が短いほうの文字がすべて含まれ、その文字が短い単語と同じ順序で出現する場合に一致とみなされます。たとえば、"Intl. Business Products" は "International Bus Prd" と一致します。
頭文字	文字列属性の値が一致とみなされるのは、一方の文字列が他方の文字列の頭文字である場合です。頭文字を識別する前に、文字列全体に「標準化された完全一致」比較が実行されます。一致が検出されない場合は、一方の文字列の各単語が、他方の文字列の対応する単語と比較されます。どの単語も一致しない場合、一方の文字列にある単語の各文字が、他方の文字列の各単語の先頭の文字と比較されます。文字が同じ場合は、一致とみなされます。たとえば、"Chase Manhattan Bank NA" と "CMB North America" は一致します。比較の際には大 / 小文字の区別、英数字以外の文字が無視されます。

会社一致ルール

ビジネス名でレコードを照合するには、会社一致ルールを使用します。ビジネス名による照合が最も有効なのは、Match-Merge 演算子の前に Name and Address 演算子を使用してアドレス・データを修正する場合です。

会社一致ルールを定義する手順は次のとおりです。

1. 「一致ルール」タブで、「ルール・タイプ」に「会社」を選択します。

   ページの下部には、「会社属性」タブと「詳細」タブが表示されます。

2. 「会社属性」タブの左側のパネルで、会社名を示す属性を選択し、「>」ボタンをクリックします。

3. 「ロール」を右クリックし、その属性を「会社 1」に指定します。

   デフォルトでは、演算子は「会社 1」の値で完全一致を比較します。このデフォルトの動作は、「詳細」タブで変更できます。

4. 必要に応じて別の属性を追加し、そのロールを「会社 2」に指定します。

   一致ルールを有効にするには、少なくとも 1 つの属性を「会社 1」または「会社 2」に割り当てる必要があります。

5. 「詳細」タブを選択し、必要なオプションを選択します。

   「ストライプ・ノイズ・ワード」を選択すると、ビジネス名に含まれる "the" や "and" は無視されます。「会社 1 と会社 2 のクロス一致」を選択すると、会社 1 のビジネス名と会社 2 のビジネス名が比較されます。

   「詳細」タブのその他のオプションの詳細は、8-64 ページの表 8-16 を参照してください。

人名一致ルール

人名に基づいてレコードを照合するには、人名一致ルールを使用します。人名による照合が最も有効なのは、Match-Merge 演算子の前に Name and Address 演算子を使用してアドレス・データを修正する場合です。

人名一致ルールを定義する手順は次のとおりです。

1. 「一致ルール」タブで、「ルール・タイプ」に「名前」を選択します。

   ページの下部には、「人名属性」タブと「詳細」タブが表示されます。

2. 「人名属性」タブの左側のパネルで、姓を示す属性を選択し、「>」ボタンをクリックします。

3. 「ロール」を右クリックし、その属性を「姓」に指定します。

4. 必要に応じて他の属性を追加して、そのロールを指定します。この一致ルールを有効にするには、「姓」または「標準化された名前」のいずれかを定義する必要があります。割り当てるロールのタイプは、表 8-17 を参照してください。

5. 「詳細」タブを選択して、表 8-18 に示すようなオプションを選択します。

表 8-17 では、名前一致ルールに割り当てる名前ロールを示します。

表 8-17 名前ロール

ロール	説明
プリネーム	演算子がプリネームを比較するのは、一方のレコードの「標準化された名前」が空白であり、「Mrs」オプションが選択され、「姓」ロールとミドル・ネームのいずれかのロールが一致する場合のみです。この基準では、たとえば、レコードの "Mrs. William Webster" と "Mrs. Webster" が一致します。
標準化された名前	両方とも空白の場合に名前が一致します。「プリネーム」ロールが指定されず、「"Mrs" の一致」オプションが設定されていないかぎり、一方の名前が空白で、他方の名前が空白でない場合は一致しません。
標準化されたミドル・ネーム、標準化されたミドル・ネーム 2、標準化されたミドル・ネーム 3	ミドル・ネームが指定されている場合、演算子はそのミドル・ネームを比較および相互比較します。デフォルトでは、ミドル・ネームは完全に一致する必要があります。ミドル・ネームは、一方または両方が空白の場合でも一致します。ミドル・ネームのロールを割り当てるには、「標準化された名前」ロールも割り当てる必要があります。
姓	両方の姓が空白の場合に一致、一方のみが空白の場合は不一致とみなされます。
役職名	これは、"Jr." や "Sr." などのポストネームと同じです。両方の値が完全に同じか、一方が空白の場合に一致とみなされます。

表 8-18 では、名前一致ルールの「詳細」タブで選択できるオプションを示します。

表 8-18 名前ロールのオプション

オプション	説明
イニシャルの一致	"R." と "Robert" のようにイニシャルと名前が照合されます。名前とミドル・ネームのロールに、このオプションを選択できます。
部分文字列の一致	"Rob" と "Robert" のように部分文字列と名前が照合されます。名前とミドル・ネームのロールに、このオプションを選択できます。
類似度	表 8-16 を参照してください。
Soundex	表 8-16 を参照してください。
複合名の検出	"De Anne" と "Deanne" のように複合名と名前が照合されます。名前のロールに、このオプションを選択できます。
"Mrs" の一致	'Mrs. Washington" と "George Washington" のようにプリネームと姓名が照合されます。プリネームのロールに、このオプションを選択できます。
ハイフンで連結された一致名	"Reese-Jones" と "Reese" のようにハイフンで連結された姓とハイフンなしの姓が照合されます。姓のロールに、このオプションを選択できます。
欠落したハイフンの検出	たとえば、"Hillary Rodham Clinton" と "Hillary Rodham-Clinton" を照合する場合に、欠落したハイフンが検出されます。姓のロールに、このオプションを選択できます。
姓名の順序の変更を検出	たとえば、"Elmer Fudd" と "Fudd Elmer" を照合する場合に、姓名の順序の変更が検出されます。「人名属性」タブで属性に「名」ロールと「姓」ロールを選択した場合に、このオプションを選択できます。

重み一致ルール

割り当てた重み値に基づいて行を照合するには、重み一致ルールを使用します。重み一致ルールが最も有効なのは、多数の属性を比較し、不一致とみなされるような異なる属性が 1 つもない場合です。条件付きルールの場合には、異なる属性によって不一致とみなされます。

重み一致ルールを使用する手順は次のとおりです。

1. 「一致ルール」タブで、「ルール・タイプ」に「重み」を選択します。

   ページの下部には、「詳細」タブが表示されます。

2. ページの下部にある「追加」を選択して、ルールに属性を追加します。

   「最大スコア」で、比較対象に加える各属性に重みを指定します。Warehouse Builder では、類似度のアルゴリズムを使用して各属性を比較します。類似度のアルゴリズムは、行間の類似度を表す 0 ～ 100 のスコアを返します。類似度 100 は、2 つの値が同じであることを示します。ゼロの類似度は、まったく類似性がないことを示します。

3. 「一致のための必須スコア」で、一致のための合計スコアを指定します。

   2 行が一致すると判断されるには、合計した数が指定した必須スコアよりも大きい必要があります。

重み一致ルールの例

表 8-19 のデータに重み一致ルールを適用するとします。

表 8-19 照合するレコードの例

レコード番号	Attr_1	Attr_2
Rec_1	CA	QQ
Rec_2	CA	QQ
Rec_3	CA	QR

「最大スコア」には、Attr_1 と Attr_2 の両方に値 50 を指定します。「一致のための必須スコア」には、値 80 を指定します。結果は次のようになります。

- Rec_1 は駆動レコードです。演算子はこれを最初に読み取ります。
- Rec_2 の Attr_1 の値は CA です。この値は、駆動レコードである Rec_1 の値との類似度が 100 です。Attr_1 の重み値は 50 であるため、そのスコアは 50（50 の 100%）です。
- Rec_2 の Attr_2 の値は QQ で、類似度は 100 です。Attr_2 の重み値も 50 であるため、スコアも 50（50 の 100%）です。これで、合計した最大スコアは 100（50 + 50）になります。これは、「一致のための必須スコア」の値以上であるため、Rec_2 と Rec_1 は一致とみなされます。

- Rec_3 の Attr_1 は CA で、Rec_1 との類似度は 100 です。Attr_1 の重み値は 50 であるため、その重みスコアは 50（50 の 100%）になります。
- Rec_3 の Attr_2 の値は QR で、類似度は 50 です。Attr_2 の最大値は 50 であるため、そのスコアは 25（50 の 50%）です。これで、合計した重みスコアは 75（50+25）になります。これは、「一致のための必須スコア」の値未満です。そのため、Rec_3 と Rec_1 は不一致とみなされます。

Match-Merge 演算子からのデータの詳細化

Match-Merge 演算子を使用してデータを渡した後、データの詳細化が必要になる場合があります。たとえば、Name and Address データで世帯検索を行う場合、最初にアドレスのデータをマージし、次に名前のデータをマージする必要があります。MERGE 出力をターゲット表にマッピングする場合は、XREF グループをステージング表か別の Match-Merge 演算子にマッピングできます。ステージング表へのマッピングは、設計が比較的容易ですが、パフォーマンスが著しく低下する場合があります。XREF グループを直接別の Match-Merge 演算子にマッピングすると、パフォーマンスの低下が避けられます。

図 8-19 では、Match-Merge 演算子である MM と MM_1 間にステージング表を使用したマッピングを示しています。

図 8-19 世帯検索 : ステージング表を使用したマッピング

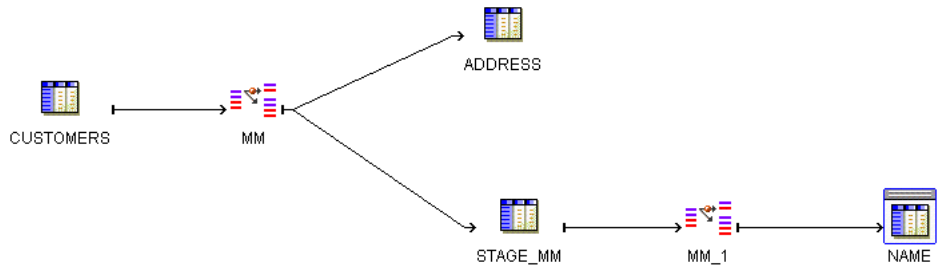
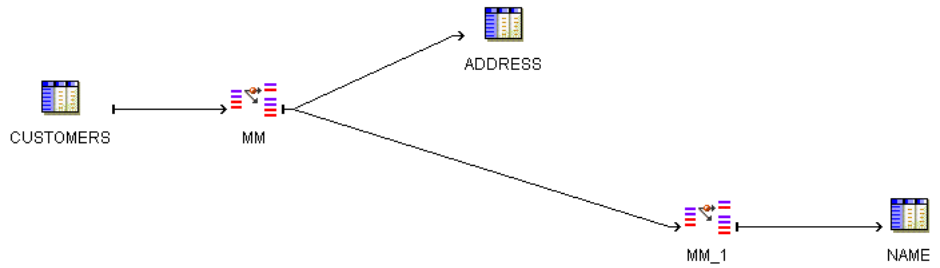


図 8-20 では、同じ結果でもパフォーマンスを向上できるマッピングを示しています。MM からの XREF グループは、直接 MM_1 にマッピングされています。このマッピングを有効にするには、最初の XREF グループで生成された一致 ID を 2 番目の Match-Merge 演算子の一致 bin ルールに指定する必要があります。

図 8-20 世帯検索 : 2 番目の Match-Merge 演算子にマッピングされた XREF グループ



Name and Address 演算子

この項では、次のトピックについて説明します。

- Name and Address 演算子について（8-72 ページ）
- 例 : Name and Address 演算子を使用したレコードの操作（8-73 ページ）
- マッピングでの Name and Address 演算子の使用方法（8-77 ページ）
- 郵便レポート（8-87 ページ）

この項では、Name and Address 演算子の概要とその使用手順について説明します。Name and Address 演算子の詳細は、第 20 章「データの品質 : Name and Address のクレンジング」を参照してください。

Name and Address 演算子について

Oracle Warehouse Builder では、Name and Address 演算子を使用して、データで Name and Address のクレンジングを実行できます。Name and Address 演算子では、サードパーティの Name and Address クレンジング・ソフトウェア・ベンダーから供給されたデータ・ライブラリと入力データを比較して、Name and Address のソース・データにあるエラーや非一貫性を識別し、修正します。データ・ライブラリは、これらのベンダーから直接購入できます。

注意： Name and Address 演算子を使用するには、別途ライセンスを購入し、追加のインストール手順を実行する必要があります。詳細は、『Oracle Warehouse Builder 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。

Name and Address 演算子によって修正されるエラーや非一貫性には、アドレス・フォーマットの相違、略語の使用、スペルの間違い、古くなった情報、一貫性のないデータ、名前の順序変更などがあります。Name and Address 演算子は、これらのエラーと非一貫性を次のように修正します。

- Name and Address の入力データを個々の要素に解析または分割します。
- Name and Address のデータを標準化します。たとえば、一般的なニックネームやビジネス名、あるいは各国の郵政公社で承認されているアドレスの標準的な略称が使用されます。名前やアドレスを標準化すると、照合や世帯検索が容易になり、顧客を単一のビューで表示できます。
- 市区町村名などのアドレス情報を修正します。不適切なアドレスや郵送不可能なアドレスを除外しておくと、マーケティング・キャンペーンなどでコストの節約につながります。
- 性別、ZIP+4、国コード、区画 ID、取引 / 顧客 ID などの詳細データを名前とアドレスに追加します。この情報以外にも、調査ジオコーディングなどのアドレス情報を加えて、地域に基づくマーケティング・キャンペーンに使用できます。

  アドレスに地理的情報を追加しておくと、地域別のマーケティング活動が容易になります。たとえば、大都市圏内（都市圏から半径 *n* マイル以内）の顧客に限定したマーケティングや、会社の店舗の商圏内（店舗から半径 x マイル以内）の顧客に限定したマーケティングなどがあります。この機能には、Oracle Database のオプションである Oracle Spatial や Oracle Database に同梱されている Oracle Locator 製品もあわせて使用できます。

また、Name and Address 演算子を使用すると、アドレス修正と郵便照合がサポートされている国の郵便レポートを生成できます。この機能がサポートされている国の郵便レポートを使用すると、郵便割引を受けられる場合もあります。詳細は、8-87 ページの「郵便レポート」を参照してください。

例 : Name and Address 演算子を使用したレコードの操作

この例では、Name and Address 演算子を使用したマッピングでレコードを操作する方法を説明します。このマッピングでは、データの品質エラーを処理する際の推奨方法として、スプリッタ演算子を使用する方法も説明します。Name and Address 演算子によるデータの処理方法の詳細は、第 20 章「データの品質 : Name and Address のクレンジング」を参照してください。

入力例

この例では、ソース・データの顧客表に、表 8-20 に示すデータの行が含まれています。

表 8-20 Name and Address 演算子へのサンプル入力

アドレス列	アドレス・コンポーネント
名前	Joe Smith
番地	8500 Normandale Lake Suite 710
市区町村	Bloomington
郵便番号	55437

このデータにはニックネーム、姓、および郵送先の一部が含まれますが、顧客のフル・ネーム、完全な番地、州などの情報がありません。また、このデータには、トラック輸送の距離計算に使用できる、緯度や経度などの地理的な情報も含まれていません。こうした Name and Address データを完全にするには、Name and Address 演算子を使用できます。

手順の例

この例のマッピングでは、Name and Address 演算子の後にスプリッタ演算子を使用します。Name and Address のレコードをクレンジングし、その解析結果に応じて、レコードを別々のターゲットにロードします。この項では、このようなマッピングの設計に必要な一般手順について説明します。各演算子の詳細は、この章の該当する項を参照してください。

サンプル・レコードを修正する手順は次のとおりです。

1. マッピング・エディタで、次の演算子をキャンバスに追加します。

   レコードの抽出元の CUSTOMERS 表。これは、データ・ソースです。このソースには、8-73 ページの「入力例」にあるデータが含まれています。

   Name and Address 演算子。これをキャンバスに追加すると、Name and Address ウィザードが起動します。8-77 ページの「マッピングでの Name and Address 演算子の使用方法」の説明に従って、このウィザードに指定します。

   スプリッタ演算子。この演算子の使用方法は、8-101 ページの「スプリッタ演算子」を参照してください。

   次の各レコードを 3 つのターゲット演算子にロードします。

   解析に成功したレコード

   解析エラーがあったレコード

   アドレスは解析されたが、郵便照合ソフトウェアに存在しなかったレコード

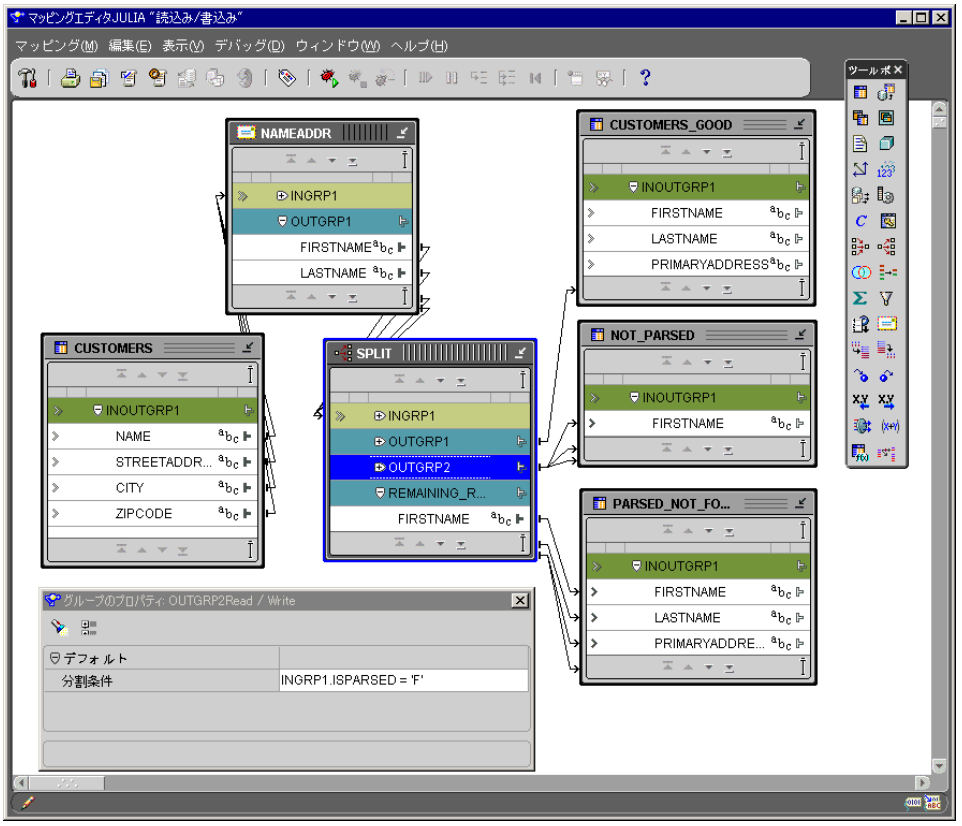
2. CUSTOMERS 表の属性を Name and Address 演算子イングループにマッピングします。Name and Address 演算子アウトグループの属性をスプリッタ演算子イングループにマッピングします。

   Name and Address 演算子とともにスプリッタ演算子を使用する必要はありませんが、組み合せて使用すると、Name and Address のエラー処理機能をよく理解できます。

3. スプリッタ演算子の各アウトグループの分割条件を定義し、アウトグループをターゲットにマッピングします。

図 8-21 では、この例で設計するマッピングを示します。データは、ソース表から Name and Address 演算子にマッピングされ、次にスプリッタ演算子にマッピングされます。スプリッタ演算子により、解析に成功したレコードと、解析時にエラーがあったレコードが分割されます。OUTGRP1 の出力は、CUSTOMERS_GOOD ターゲットにマッピングされます。OUTGRP2 の分割条件は画面の下部に表示され、「解析済」フラグが False のレコードは NOT_PARSED ターゲットにロードされます。REMAINING_RECORDS グループのレコードは解析に成功していますが、そのアドレスは郵便照合ソフトウェア内で検出されません。これらのレコードは、PARSED_NOT_FOUND ターゲットにロードされます。

図 8-21 Name and Address 演算子とともにスプリッタ演算子を使用したマッピング



出力例

この例で設計したマッピングを実行すると、Name and Address 演算子によって、ソース表にあるアドレス・データが標準化および修正され、完全なデータになります。この例では、ターゲット表に、表 8-21 に示すアドレス・データが含まれています（8-73 ページ）の表 8-20 の入力レコードと比較してください)。

表 8-21 Name and Address 演算子の出力サンプル

アドレス列	アドレス・コンポーネント
標準化された名前	JOSEPH
姓	SMITH
第 1 アドレス	8500 NORMANDALE LAKE BLVD
第 2 アドレス	STE 710
市区町村	BLOOMINGTON
州（都道府県）	MN
郵便番号	55437-3813
緯度	44.849194
経度	-093.356352
解析済	このフィールドは、レコードの解析に成功したかどうかを示し、エラー処理に追加されます。この値に基づいて、レコードは、解析に成功したレコード用のターゲット演算子か、エラーがあったレコード用のターゲット演算子にロードされます。
適切な名前	このフィールドは、エラー処理の詳細に追加されます。
名前警告	このフィールドは、エラーを手動で修正する際に役立つように、追加されます。
適切なアドレス	このフィールドは、エラー処理の詳細に追加されます。
検出済	このフィールドは、郵便照合ソフトウェアがアドレスを検出したかどうかを示します。解析に成功したアドレスでも False になる場合があります。このフィールドにより、これらのレコードは別のターゲットに移動します。
町村警告	このフィールドは、エラーを手動で修正する際に役立つように、追加されます。
市町村警告	このフィールドは、エラーを手動で修正する際に役立つように、追加されます。

この例では、入力データが次のように修正されています。

- Joe Smith が「標準化された名前」および「姓」に対応する 2 列に分割されています。
- Joe が JOSEPH に標準化され、Suite が STE に標準化されています。
- Normandale Lake が Normandale Lake BLVD に修正されています。
- 郵便番号の最初の 55437 に ZIP+4 コードが追加され、55437-3813 になっています。
- 緯度および経度情報が追加されています。
- 解析に成功したレコードとエラーのあったレコードを区別するために、フラグが追加されています。エラーのあったレコードは別のターゲットにロードされています。

マッピングでの Name and Address 演算子の使用方法

Name and Address 演算子を使用する際には次のオプションがあります。

- **新しい Name and Address 演算子を定義する :** ツールボックスの Name and Address 演算子をマッピングにドラッグ・アンド・ドロップします。マッピング・エディタでウィザードが起動します。
- **既存の Name and Address 演算子を編集する :** 演算子を右クリックして、「編集」を選択します。マッピング・エディタで Name and Address エディタが起動します。

演算子ウィザードまたは演算子エディタのどちらを使用する場合も、次の各ページで指定します。

- 一般
- 定義
- グループ
- 接続の入力
- 入力属性
- 出力属性
- 郵便レポート

一般

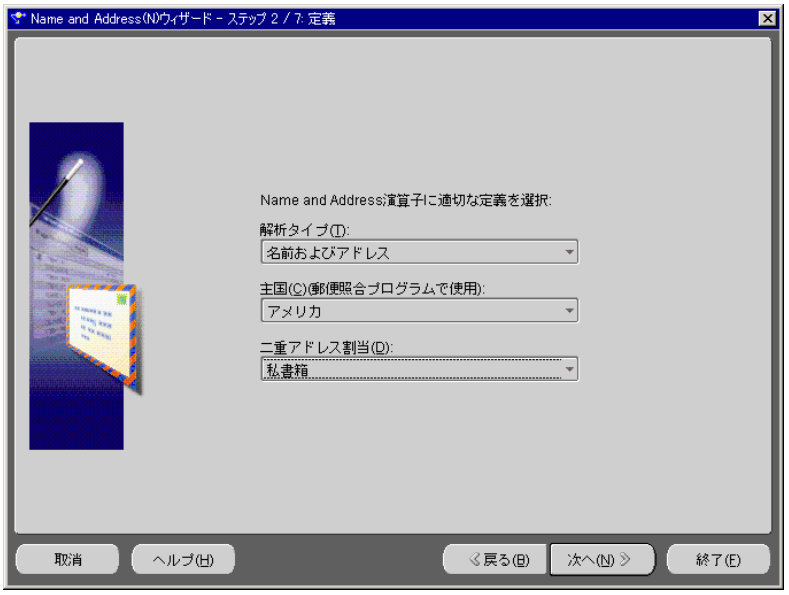
「一般」ページで、演算子の名前と説明（オプション）を指定します。このウィザードでは、Name and Address 演算子に "NAMEADDR" というデフォルト名が付きます。

定義

この Name and Address 演算子に一般的な定義を指定して、入力データを特徴付けます。

図 8-22 の「定義」ページには、8-73 ページの「例 : Name and Address 演算子を使用したレコードの操作」に記載されたデータのサンプル値が示されています。「解析タイプ」は「'Name and Address'」に設定され、「主国」は「アメリカ」に設定され、「二重アドレス割当」は「私書箱」に設定されています。

図 8-22　Name and Address 演算子の「定義」ページ



解析タイプ　ドロップダウン・リストから次のいずれかの解析タイプを選択します。

注意：「解析タイプ」を指定できるのは、マッピングに Name and Address 演算子を追加した場合のみです。演算子エディタで Name and Address 演算子を編集する場合は、「解析タイプ」を変更できません。

- **名前のみ：**入力データに名前データのみが含まれている場合に、この方法を選択します。個人名とビジネス名の両方が対象となります。より包括的な「名前およびアドレス」オプションではなく、このオプションを選択した場合は、パフォーマンスと精度が向上します。
- **アドレスのみ：**入力データがアドレス・データのみで構成され、名前データが含まれていない場合にこの方法を選択します。より包括的な「名前およびアドレス」オプションではなく、このオプションを選択した場合は、パフォーマンスと精度が向上します。
- **名前およびアドレス：**入力データに名前データとアドレス・データの両方が含まれている場合に、この方法を選択します。入力データに名前データのみ、またはアドレス・データのみが含まれている場合は、そのオプションを選択するとパフォーマンスと精度が向上します。

主国 データの中で最も頻出する国を選択します。主国は、レコードの最初の解析に使用する適切なパーサーまたは解析ルールの手がかりとして、Name and Address クレンジング・ソフトウェアの一部のプロバイダによって使用されています。それ以外の Name and Address サービス・プロバイダの場合は、インストールの外部構成によってこの動作が制御されています。

二重アドレス割当 二重アドレスには、同じアドレス・レコードの私書箱と町村の両方が含まれています。二重アドレスを持つレコードの場合は、標準アドレスになるアドレスと、二重アドレスになるアドレスを選択します。二重アドレスの例は、次のとおりです。

PO Box 2589
4439 Mormon Coulee Rd
La Crosse WI 54601-8231

「二重アドレス割当」での選択によって、郵便番号の修正時に指定される郵便コードが決まります。これは、町村の郵便番号と私書箱の郵便番号が異なる場合があるためです。

- **Street Assignment:** 町村が標準アドレスとみなされ、私書箱が二重アドレスとみなされます。これは、アドレス・コンポーネントに町村が割り当てられることを意味します。前述の例では、4439 MORMON COULEE RD が割り当てられています。そのため、郵便番号は 54601-8220 に修正されます。
- **PO Box Assignment:** 私書箱が標準アドレスとみなされ、町村が二重アドレスとみなされます。これは、アドレス・コンポーネントに私書箱が割り当てられることを意味します。前述の例では、PO BOX 2589 が割り当てられています。そのため、郵便番号は 54602-2589 に修正されます。
- **最終行に最も近い :** 最終行に最も近いアドレスが標準アドレスとみなされ、離れているアドレスが二重アドレスとみなされます。これは、最終行に最も近いアドレス行がアドレス・コンポーネントに割り当てられることを意味します。前述の例では、町村の 4439 MORMON COULEE RD が割り当てられています。そのため、郵便番号は 54601-8220 に修正されます。

このオプションは、町村または私書箱のいずれかのみを持つレコードには影響しません。このオプションは、すべての Name and Address クレンジング・ソフトウェア・プロバイダによってサポートされているわけではありません。

グループ

定義上、Name and Address 演算子には 1 つの入力グループと 1 つの出力グループがあります。Name and Address 演算子のグループを編集、追加または削除することはできません。入力グループには INGRP1、出力グループには OUTGRP1 の名前が付けられています。これらの名前を編集することは可能です。入力グループに複数のグループが必要な場合は、グループごとに別の Name and Address 演算子を作成してください。

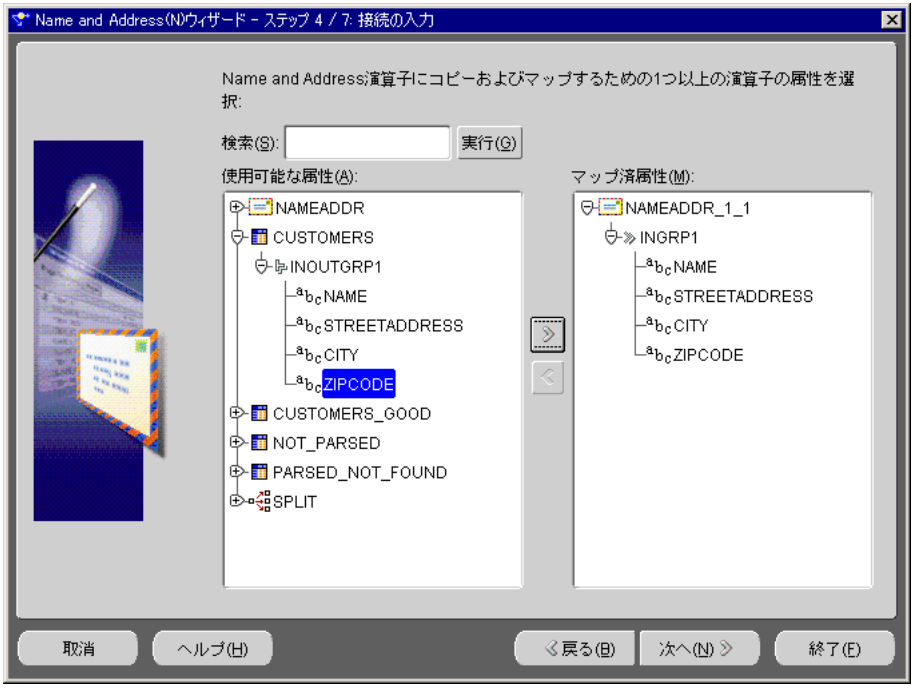
「接続の入力」ページで INGROUP に属性を割り当て、次にこれらの属性を「入力属性」ページで編集します。「出力属性」ページでは OUTGRP1 グループに属性を割り当てます。

接続の入力

「接続の入力」ページで、マッピングの演算子から属性を選択し、それを演算子にコピーおよびマッピングします。「使用可能な属性」パネルでは、マッピング内の任意の演算子から属性を選択できます。「マップ済属性」パネルには、Name and Address 演算子が表示されます。属性を左のパネルから右のパネルに移動すると、その属性が演算子にマッピングされます。

図 8-23 の「接続の入力」ページには、8-73 ページの「例 : Name and Address 演算子を使用したレコードの操作」に記載された例のサンプル値が示されています。ただし、CUSTOMERS 表の列は、入力属性としてマッピングされています。

図 8-23　Name and Address 演算子の「接続の入力」ページ



演算子の「接続の入力」ページで指定する手順は次のとおりです。

1. 「使用可能な属性」パネルで、グループ全体または個別の属性を選択します。「使用可能な属性」パネルでは、マッピング内の任意の演算子から属性を選択できます。

   名前を使用して特定の属性またはグループを検索するには、「検索」にテキストを入力し、「実行」をクリックします。次の一致文字列を検索するには、「実行」を再度クリックします。

   複数のグループまたは属性を選択するには、[Shift] キーを押しながら選択します。異なるグループの属性を選択する場合は、結合子演算子または集合演算子を使用してグループを結合する必要があります。

2. 2 つのパネルの間にある「>」ボタンを使用して、選択内容を「マップ済属性」パネルに移動します。

入力属性

「入力属性」ページで、「接続の入力」ページで選択した属性をさらに変更し、入力ロールを各入力属性に割り当てます。

Name and Address 演算子の「入力属性」ページでは、次のタスクを実行できます。

- **属性を追加する :** 「追加」ボタンを使用して入力属性を追加します。
- **属性のプロパティを変更する :** 属性名、入力ロール、長さを変更できます。データ型は VARCHAR2 のままにします。精度やスケールは変更できません。入力ロールはすべての入力属性に割り当てる必要があります。

  入力ロールは、どのような種類の名前およびアドレス情報がデータ行にあるかを示します。各属性で、ソース属性に含まれるデータに最も近い入力ロールを選択します。入力ロールの一覧とその説明は、20-2 ページの表 20-1 を参照してください。

  自由形式データには目的別に分割されていない（行指向の）入力ロールを選択でき、特定の入力属性には目的別に分割されたロール（名前、第 1 アドレス、市区町村など）を選択できます。入力ロールには可能なかぎり、目的別に分割されていないロール（「行 1」など）ではなく、目的別に分割されたロール（「人名」など）を使用する必要があります。分割されたロールでは、Name and Address 演算子に対して、ソース属性の内容に関する情報がより多く与えられます。
- **説明（オプション）を追加する :** 入力属性の説明を入力します。

図 8-24 の「入力属性」ページには、8-73 ページの「例 : Name and Address 演算子を使用したレコードの操作」に記載されたデータのサンプル値が示されています。この例では、ソース表にある NAME 列は 1 つだけで、姓名（Joe Smith など）が格納されています。したがって、この列には「人名」入力ロールが割り当てられています。また、STREETADDRESS 列には郵送先の町村の部分が格納されているため、この列には「アドレス」入力ロールが割り当てられています。

図 8-24 Name and Address 演算子の「入力属性」ページ

出力属性

「出力属性」ページで出力属性を定義します。出力属性によって、Name and Address での解析済データの処理方法が決定されます。具体的には、出力属性プロパティによって、パーサー出力から抽出されたデータが特徴付けられます。

出力属性コレクションは、最初は空です。属性は次のように作成して、編集できます。

- 出力属性を作成するには、「追加」を選択します。
- 属性名を編集するには、該当するセルをクリックし、デフォルト名に上書きします。

  出力コンポーネントはすべての出力属性に指定する必要があります。この場合、各属性の右側にある「...」ボタンをクリックし、「出力コンポーネント」ダイアログを開いて、コンポーネントを割り当てます。出力コンポーネントの一覧およびその機能の説明は、20-4 ページの表 20-2 を参照してください。
- フィールド長は、出力属性をマッピングするターゲット属性の長さにあわせて調整できます。ターゲット属性にあわせて長さを調整すると、コード生成時のデータ切捨て警告や、実行時のエラーを回避できます。

- データ型は変更できません。

図 8-25 の「出力属性」ページには、8-73 ページの「例 : Name and Address 演算子を使用したレコードの操作」に記載されたデータのサンプル値が示されています。

図 8-25 Name and Address 演算子の「出力属性」ページ

この図では、すべての出力属性に出力コンポーネントが割り当てられています。たとえば、「名」属性には、「標準化された名前」コンポーネントが割り当てられています。また、「緯度」と「経度」属性が追加され、アドレス情報が増えています。

さらに、「解析済」、「適切な名前」、「適切なアドレス」などのエラー処理フラグが付けられています。これらのフラグをスプリッタ演算子に使用すると、解析に成功したレコードを、エラーのあったレコードから分離して、別のターゲットにロードできます。

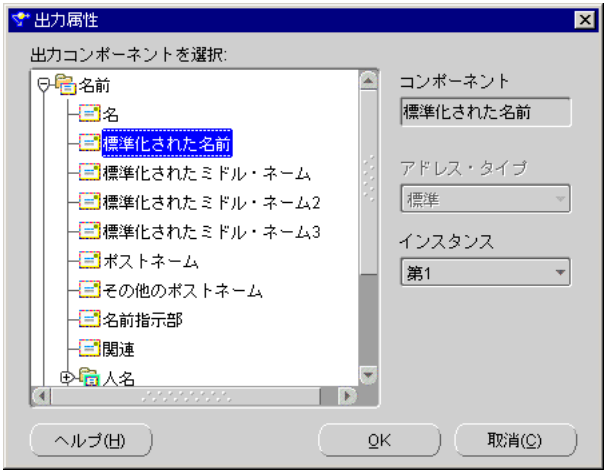
出力コンポーネント「出力属性」コンポーネント・ダイアログで、作成する出力属性のコンポーネントを定義します。出力コンポーネントは、タイトル、標準化された名前、番地、町村名、町村タイプなど、目的別に分割された名前またはアドレス・エンティティを表し、属性が構成される名前やアドレスのコンポーネントを示します。選択したコンポーネントは出力属性に割り当てられます。

- **出力コンポーネント :**「出力属性」コンポーネント・ダイアログの左側にあるコンポーネント・ツリーを使用して、コンポーネントが含まれたカテゴリを開き、該当する属性に割り当てる名前またはアドレス・コンポーネントを選択します。ツリー上で緑色の縁の封筒アイコンが付いているノードはどれでも選択できます。そのコンポーネントが開いて他のノードが表示されていても選択できます。これらのコンポーネントの詳細は、20-4 ページの表 20-2「Name and Address 演算子の出力コンポーネント」を参照してください。
- **アドレス・タイプ :**「二重アドレス割当」のように、このオプションはすべての Name and Address クレンジング・ソフトウェア・プロバイダによってサポートされているわけではありません。このオプションは、町村と私書箱または町村とルートボックス・アドレスが含まれた二重アドレスの場合にのみ適用されます。「定義」ページで指定した「二重アドレス割当」オプションによって、町村と私書箱のどちらを二重アドレスとして使用するかが決まります。標準アドレスまたは二重アドレスを選択します。二重アドレスの詳細は、8-79 ページの「二重アドレス割当」を参照してください。
- **インスタンス :** 1 つのレコードに同じ属性が複数出現した場合に、どの出力コンポーネントのインスタンスを使用するかを指定します。インスタンス制御は、すべての名前コンポーネントと、「他のアドレス」や「複合」のようないくつかのアドレス・コンポーネントに適用されます。インスタンスにより、同じ性質の属性をいくつでも抽出することができます。

  たとえば、John と Jane Doe が含まれる入力レコードには、John Doe と Jane Doe の 2 つの名前が出現します。任意の名前コンポーネントによって最初の人物を抽出するには、インスタンス 1 をそのコンポーネントに割り当てます。同様に、任意の名前コンポーネントによって 2 番目の人物を抽出するには、インスタンス 2 を割り当てます。各種のコンポーネントに可能なインスタンス数は、使用している Name and Address クレンジング・ソフトウェア・ベンダーによって異なります。「他のアドレス」では、電子メール・アドレスと電話番号の両方が存在する場合に、複数のインスタンスを持つことができます。

図 8-26 の「出力属性」ページでは、8-73 ページの「例 : Name and Address 演算子を使用したレコードの操作」で使用した最初のサンプル出力属性のコンポーネントが選択されています。

図 8-26　Name and Address 演算子の「出力属性」コンポーネント・ダイアログ



8-73 ページの「例 : Name and Address 演算子を使用したレコードの操作」では、アドレス・タイプは「標準」です。この例で必要な出力コンポーネントは、「標準化された名前」、「姓」、「第 1 アドレス」、「第 2 アドレス」、「市区町村」、「州（都道府県）」、「郵便番号」、「緯度」、「経度」、「解析済」、「適切な名前」、「名前警告」、「適切なアドレス」、「検出済」、「町村警告」、および「市区町村警告」です。

郵便レポート

郵便レポートは、アドレス修正と郵便照合がサポートされている国にのみ適用されます。国の認証は、Name and Address クレンジング・ソフトウェア・ベンダーによって異なります。最も一般的な国の認証は、アメリカ、カナダおよびオーストラリアです。このプロセスは、郵便利用業者にとって、アドレス照合ソフトウェアの品質を測定する共通プラットフォームとなり、あらゆる郵便に適用される郵便番号（アメリカの場合、5 桁の郵便番号と ZIP+4 コード）、デリバリ・ポイント・コードおよび配達ルート・コードの正確さについて検証します。Name and Address クレンジング・ソフトウェア・ベンダーによっては、これらのパラメータが無視され、郵便レポートを生成する外部のセットアップが必要になる場合もあります。詳細は、8-87 ページの「郵便レポート」を参照してください。

- **郵便レポート :**「郵便レポート」に「はい」を選択した場合、「定義」ページで選択した「主国」によって郵便レポートを生成する国が決まります。郵便レポートは 1 つのみアクティブにできます。
- **プロセッサ名 :** Name and Address クレンジング・ソフトウェア・ベンダーによって、このフィールドを使用するか使用しないかが決まります。通常、ここに指定する値はアメリカ CASS レポートに表示されます。

- **リスト名 :** リスト名は、アメリカ・レポートおよびイギリス・レポートの「リスト名」セクションに表示されるオプションの参照フィールドで、他のレポートには出力されません。リスト名には、'July 2003 Promotional Campaign' のように、複数の郵便レポートを追跡するための参照を指定します。
- **プロセッサ・アドレス行 :** 各種の郵便レポートに 4 つのアドレス行が表示される場合があります。Name and Address クレンジング・ソフトウェア・ベンダーによってこれらのフィールドの使用方法が異なります。これらの行には、会社の詳細アドレスを指定するのが一般的です。

図 8-27 は、サンプル値が含まれた「郵便レポート」ページを示しています。

図 8-27 Name and Address 演算子の「郵便レポート」ページ

Name and Address ウィザードを終了するには、「終了」をクリックします。Name and Address エディタを終了するには、「OK」をクリックします。

郵便レポート

Name and Address 演算子エディタまたはウィザードの「郵便レポート」ページで、郵便レポートを指定できます。Name and Address 演算子が含まれるマッピングを実行すると、郵便レポートが生成されます。

郵便レポートは、アドレス修正と郵便照合がサポートされている国にのみ適用されます。これらの国は、Name and Address のクレンジング・ソフトウェア・ベンダーによって異なります。最も一般的な認証は、米国、カナダおよびオーストラリアです。

このプロセスは、郵便利用業者にとって、アドレス照合ソフトウェアの品質を測定する共通プラットフォームとなり、あらゆる郵便に適用される郵便番号（アメリカの場合、5 桁の郵便番号と ZIP+4 コード）、デリバリ・ポイント・コードおよび配達ルート・コードの正確さについて検証します。

オートメーション料金の郵便物の生成に使用されるアドレス一覧はすべて、郵便レポート認定ソフトウェアにより照合される必要があります。Oracle Warehouse Builder Name and Address は、Name and Address クレンジングを専門とするサードパーティ・ソフトウェア・ベンダーで提供する Name and Address ソフトウェアとデータに基づいて構築されています。そのため、認証はベンダーごとに異なります。次のような認証があります。

- **アメリカ :** 米国郵政公社による Coding Accuracy Support System（CASS）認証。CASS レポートは USPS 指定のテキスト・ファイルで、Oracle Warehouse Builder Name and Address により生成されます。USPS 要件を満たすには、送信者は CASS レポートをオリジナルのフォームで USPS に送信する必要があります。
- **カナダ :** カナダ郵便局による Software Evaluation and Recognition Program（SERP）認証。インセンティブ・レターメール、広告付きアドメール、刊行物メールなどを利用する顧客は、アドレス照合プログラムの要件を満たしている必要があります。顧客は、使用しているデータベースとカナダ郵便局のアドレス・データを比較することで、Statement of Accuracy を取得することができます。
- **オーストラリア :** オーストラリア郵便局による Address Matching Approval System（AMAS®）認証。事前区分けサービス料金では、利用者が、最新バージョンの Postal Address File（PAF）で有効なデリバリ・ポイント ID（DPID）を持つ AMAS 承認ソフトウェアを使用する必要があります。

郵便レポート・ファイルへのアクセス

郵便レポートにアクセスするには、Name and Address Server が実装されたファイル・システムにアクセスする必要があります。レポートは Name and Address Server によって処理され、Warehouse Builder サーバー側インストール時に指定した Oracle ホーム・パス上の `owb/bin/admin/reports` フォルダに書き込まれます。インストール・パラメータの詳細は、『Oracle Warehouse Builder 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。

Warehouse Builder では、どのレポートにも、国コード、グループ名、およびファイルの作成日時を使用して一意のファイル名が作成されます。
`p_CAN_TESTGROUP1_20021219_0130.txt`. このネーミング規則は、郵便照合ソフトウェアのすべてのベンダーに適用されるとは限りません。ファイルのネーミングがベンダー・インストールの外部構成によって制御される場合もあります。

国際データの郵便レポートの制限

郵便レポートが有効なマッピングでは、Name and Address ウィザードの「定義」ページで「主国」に指定された国のデータのみが処理されます。ソースに国際データが含まれ、そのソース・レコードに国コードが含まれる場合は、その「国コード」ソース列を Name and Address 演算子の入力グループ属性にマッピングします。そして、「国コード」入力ロールをその属性に割り当てます。

ピボット演算子

ピボット演算子では、複数の属性が含まれる単一行を複数の行に変換できます。この演算子は、マッピングで複数の行ではなく、複数の属性間に含まれるデータを変換する際に使用します。たとえば、クロス集計フォーマットのデータなど、非リレーショナル・データ・ソースからデータを抽出する場合がこれに相当します。

例 : 販売データのピボット

図 8-28 に示す外部表 `SALES_DAT` には、フラット・ファイルからのデータが含まれています。行には販売担当者が格納され、月ごとに列があります。外部表の詳細は、3-27 ページの「外部表の使用方法」を参照してください。

図 8-28 SALES_DAT

ID	Reg	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
0675	4	10.5	11.4	9.5	8.7	7.4	7.5	7.8	9.7				
0676	3	9.5	10.5	10.3	7.6	8.0	7.8	8.7	8.9				
0679	3	8.7	7.4	7.5	7.8	9.7	10.3	7.6	8.0				
0683	2	9.5	10.5	10.3	9.5	8.7	7.4	7.8	8.7				
0684	1	11.4	9.5	8.7	7.4	7.5	10.3	9.5	8.7				
0687	1	9.5	8.7	7.4	7.8	8.7	7.4	7.5	7.8				
0690	1	8.7	7.4	7.8	8.7	11.4	9.5	8.7	7.4				

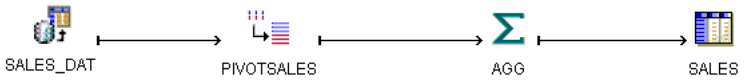
表 8-22 では、Warehouse Builder でピボット操作を実行した後のデータの例を示します。以前は複数の列（M1、M2、M3...）に格納されていたデータが、この表では 1 つの属性（Monthly_Sales）に格納されています。`SALES_DAT` にあった単一の ID 行が、ピボットされたデータの 12 行に対応しています。

表 8-22 ピボットされたデータ

REP	MONTH	MONTHLY_ SALES	REGION
0675	Jan	10.5	4
0675	Feb	11.4	4
0675	Mar	9.5	4
0675	Apr	8.7	4
0675	May	7.4	4
0675	Jun	7.5	4
0675	Jul	7.8	4
0675	Aug	9.7	4
0675	Sep	NULL	4
0675	Oct	NULL	4
0675	Nov	NULL	4
0675	Dec	NULL	4

この例のピボット変換を実行するには、図 8-29 に示すようなマッピングを作成します。

図 8-29 マッピングに使用したピボット演算子



このマッピングでは、Warehouse Builder は、外部表のデータを一度読み取った後、データをピボットし、データを集計してからセット・ベース・モードでターゲットに書き込みます。ピボット後にデータを直接ターゲットにロードする必要はありません。ピボット演算子は、データをターゲット演算子に渡す前後の一連の演算子の中で使用できます。ピボット演算子の前には、フィルタ、結合子、集合演算などの演算子を配置できます。Warehouse Builder でピボットされたデータは、行対行の操作ではないため、セット・ベース・モードでマッピングを実行することもできます。

行ロケータ

ピボット演算子では、行ロケータは、ソースの繰返しデータ・セットに対応させるために作成する出力属性です。ピボット演算子を使用すると、1 つの入力属性が複数の行に変換され、行ロケータの値が生成されます。この例では、ソースに各月の属性が含まれるため、'MONTH' という出力属性を作成し、それを行ロケータとして指定できます。これにより、SALES_DAT の各行から、データがピボットされた 12 行が出力されます。

表 8-21 では、SALES_DAT の最初の行のデータを、行インジケータ 'MONTH' を使用してピボットした結果を示します。

表 8-23 ピボットされたデータ

REP	MONTH	MONTHLY_ SALES	REGION
0675	Jan	10.5	4
0675	Feb	11.4	4
0675	Mar	9.5	4
0675	Apr	8.7	4
0675	May	7.4	4
0675	Jun	7.5	4
0675	Jul	7.8	4
0675	Aug	9.7	4
0675	Sep	NULL	4
0675	Oct	NULL	4
0675	Nov	NULL	4
0675	Dec	NULL	4

ピボット演算子の使用方法

ピボット演算子を使用する際には次のオプションがあります。

- **新しいピボット演算子を定義する :** ピボット・ウィザードを使用して、新しいピボット演算子をマッピングに追加します。ツールボックスのピボット演算子をマッピングにドラッグ・アンド・ドロップします。マッピング・エディタでピボット・ウィザードが起動します。
- **既存のピボット演算子を編集する :** ピボット・エディタを使用して、以前に作成したピボット演算子を編集します。演算子を右クリックして、「編集」を選択します。マッピング・エディタでピボット・エディタが起動します。

ピボット演算子ウィザードまたはピボット・エディタのどちらを使用する場合も、次の各ページで指定します。

- 一般
- グループ
- 接続の入力
- 入力属性
- 出力属性
- ピボット変換

一般

「一般」ページで、ピボット演算子の名前と説明（オプション）を指定します。このウィザードでは、演算子に "Pivot" というデフォルト名が付きます。

グループ

「グループ」ページで、1 つの入力グループと 1 つの出力グループを指定します。

ピボット演算子では、入力グループは、複数の属性に含まれるソースのデータを示します。出力グループは、行に変換されたデータを示します。

入力グループや出力グループの名前を変更したり、説明を追加したりできます。各ピボット演算子には、1 つの入力グループと 1 つの出力グループが必要であるため、ウィザードでグループを追加または削除したり、グループの方向を変更したりできません。

接続の入力

「接続の入力」ページで、属性をピボット演算子にコピーおよびマッピングします。選択した属性は、ピボット入力グループにマッピングされます。ページの左側には、マッピング内のすべての演算子の一覧が表示されます。図 8-30 では、ピボット演算子の入力として選択された外部表 SALES_DAT からのグループを示します。

図 8-30 ピボット演算子の「接続の入力」ページ

ピボット演算子の「接続の入力」ページで指定する手順は次のとおりです。

1. 左側のパネルで、グループ全体または個別の属性を選択します。

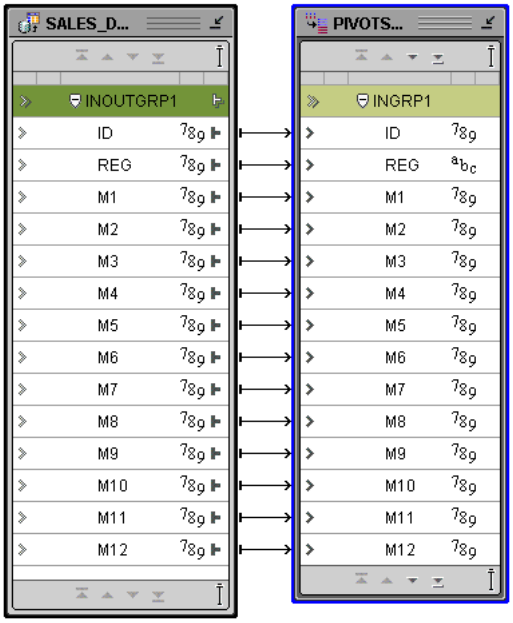
   名前を使用して特定の属性またはグループを検索するには、「検索」にテキストを入力し、「実行」を選択します。次の一致文字列を検索するには、「実行」を再度選択します。

   複数の属性を選択するには、[Shift] キーを押しながら選択します。異なるグループの属性を選択する場合は、結合子演算子または集合演算子を使用してグループを結合する必要があります。

2. ページ中央にある「>」ボタンを使用して、選択した項目をウィザードの右側に移動します。

   接続の入力の一覧からグループまたは属性を削除するには、「<」ボタンを使用します。これにより、入力グループから選択した項目が削除され、ソース演算子とピボット演算子間のマッピング線も削除されます。図 8-31 では、SALES_DAT のグループが PIVOTSALES 演算子にコピーおよびマッピングされています。

図 8-31 ピボット入力グループにコピーおよびマッピングされた属性



入力属性

「入力属性」ページで、「接続の入力」タブまたはウィザード・ページで選択した属性を変更します。

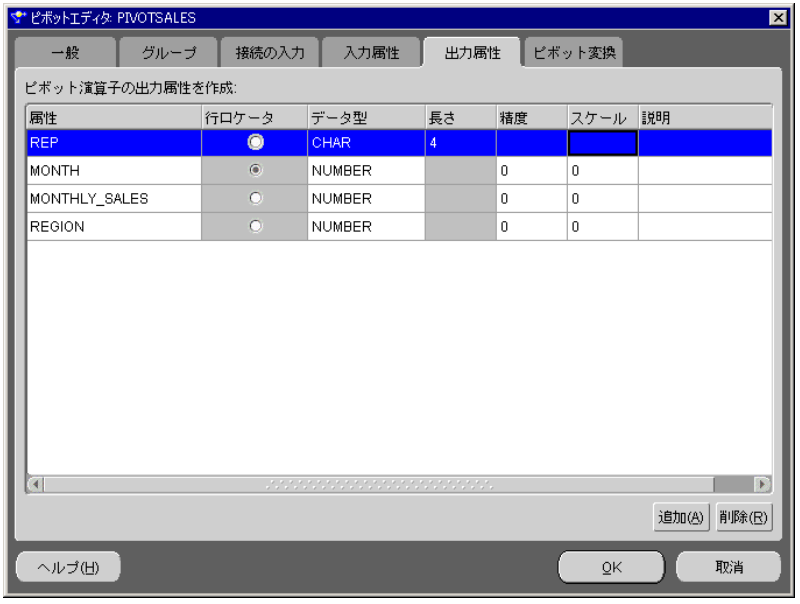
ピボットの「入力属性」ページでは、次のタスクを実行できます。

- **属性を追加する :** 「追加」ボタンを使用して入力属性を追加します。
- **属性のプロパティを変更する :** 属性名、データ型、長さ、精度、スケールを変更できます。
- **説明（オプション）を追加する :** 入力属性の説明を入力します。
- **属性キーを指定する :** オプションとして、「キー」チェック・ボックスを使用し、入力グループを一意に識別する属性を指定します。

出力属性

「出力属性」ページで、ピボット演算子の出力属性を作成します。「入力属性」タブまたはウィザード・ページで入力属性をキーとして指定した場合、Warehouse Builder では、これらの属性が編集または削除できない出力属性として表示されます。図 8-32 では、MONTH が行ロケータとして選択された出力属性を示します。

図 8-32 ピボットの「出力属性」ページ



ピボットの「出力属性」ページでは、次のタスクを実行できます。

- **属性のプロパティを変更する :** 前のタブまたはウィザード・ページでキーとして指定した属性を除いて、属性名、データ型、長さ、精度およびスケールを変更できます。
- **説明（オプション）を追加する :** 出力属性の説明を入力します。
- **行ロケータを指定する :** ピボット演算子に行ロケータを指定する必要はありませんが、指定しておくことをお薦めします。「出力属性」ページまたはタブで行ロケータを識別すると、出力データと入力データとの照合が容易になります。

  ピボット演算子では、行ロケータは、ソースの繰返しデータ・セットに対応する出力属性です。たとえば、ソース・データに、各月の個別の属性が含まれる場合、出力属性 'MONTH' を作成し、それを行ロケータとして指定します。

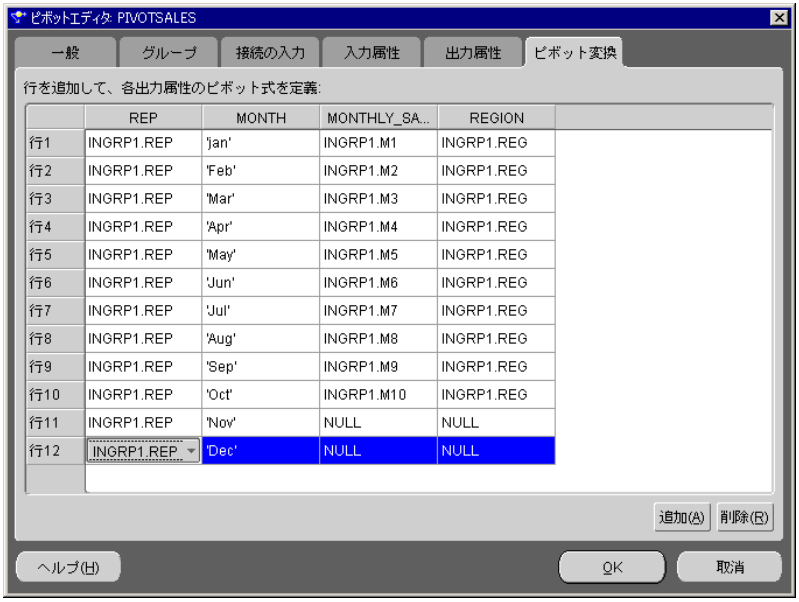
ピボット変換

「ピボット変換」ページで、各出力属性の式を記述します。

Warehouse Builder では、デフォルトで 2 行が表示されます。「追加」を使用して、ソースの 1 行から出力する行数を指定します。たとえば、ソースに 1 年の四半期ごとの属性が含まれる場合、ソースの各行の出力には 4 行を指定できます。ソース・データに、1 年の月ごとの属性が含まれる場合は、ソースの各行の出力には 12 行を指定できます。

図 8-33 の「ピボット変換」タブでは、各月の属性を使用してソースに定義されたピボット式を示します。

図 8-33「ピボット変換」ページ



ピボット式の記述は、次の出力タイプに基づきます。

- **行ロケータ :** 各行の名前を指定します。この名前は表にロードする値です。たとえば、行ロケータが 'MONTH' の場合、最初の行には 'Jan' を入力します。
- **ピボットされた出力データ :** リスト・ボックスで適切な式を選択します。たとえば、'Jan' に定義した行では、1 月の値のセットを返す式を選択します。
- **すでにキーとして指定された属性 :** Warehouse Builder によって、式が定義されます。

- **不要なデータ：**「ピボット変換」ページに出力として不要なデータが含まれる場合、'NULL' 式を使用します。Warehouse Builder では、'NULL' に定義された属性に対して、データのない繰返し行セットが出力されます。

ウィザードを使用して新しいピボット演算子を作成する場合は、ウィザードを閉じるときに「終了」をクリックします。マッピング・エディタに、定義された演算子が表示されます。

ピボット・エディタを使用して既存のピボット演算子を編集する場合は、演算子の編集が終わったときに、「OK」をクリックします。マッピング・エディタで演算子が更新され、変更内容が反映されます。

マッピング後プロセス演算子

マッピング実行後に実行するプロシージャを定義するには、マッピング後プロセス演算子を使用します。たとえば、マッピング後プロセス演算子を使用すると、マッピングが成功し、データがターゲットにロードされた後に、索引を再び有効にして作成できます。

マッピング後プロセス演算子は、マッピング実行後に、Warehouse Builder で定義されたファンクションまたはプロシージャをコールします。出力パラメータ・グループは戻り値の接続ポイントとなり（ファンクションとして実装されている場合）、ファンクションまたはプロシージャの出力パラメータを生成します。これら出力属性の接続には制約がありません。

マッピング後プロセス演算子では、選択した PL/SQL プロシージャまたはファンクションのパラメータ数と指示内容に対応するグループを使用します。グループと属性のこのリストを変更するには、調整を使用する必要があります。

1 つのマッピングで定義できるマッピング後プロセス演算子は 1 つのみです。マッピング後に複数のプロシージャを実行する場合は、複数のプロシージャを 1 つのプロシージャにラップする必要があります。

定数、データ・ジェネレータ、マッピング入力パラメータ、マッピング前プロセスからの出力をマッピング後プロセス演算子にマッピングできます。SQL*Loader マッピングでは、マッピング後プロセス演算子が無効になります。

マッピング後プロセス演算子をマッピング・エディタに追加した後、演算子プロパティ・ダイアログを使用して、プロセスを実行する実行条件を指定します。

マッピングでマッピング後プロセス演算子を使用する手順は次のとおりです。

1. マッピング後プロセス演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。

   「マッピング変換の追加」ダイアログが表示されます。

2. 「マッピング変換の追加」ダイアログで、変換を選択または作成します。「マッピング変換の追加」ダイアログの使用方法の詳細は、6-11 ページの「バインド可能な演算子の追加」を参照してください。

3. ソース演算子の出力属性を、マッピング後プロセス演算子の入力 / 出力グループに接続します。
4. 演算子の実行条件を設定します。

マッピング後プロセス演算子の実行条件を設定する手順は次のとおりです。

1. マッピング・キャンバスで、マッピング後プロセス演算子を右クリックし、「演算子のプロパティ」を選択します。
2. 「マッピング後プロセスの実行条件」をクリックして、次のいずれかの実行条件を選択します。

   ALWAYS: マッピングのエラーにかかわらずプロセスは実行されます。

   ON SUCCESS: マッピングがエラーなく完了した場合にのみプロセスは実行されます。

   ON ERROR: マッピングに設定された許容エラー数を超えてマッピングが完了した場合にのみプロセスは実行されます。

   ON WARNING: マッピングに設定された許容エラー数未満でマッピングが完了した場合にのみプロセスは実行されます。

   「ON ERROR」または「ON WARNING」を選択し、行ベース・モードでマッピングを実行する場合は、マッピングに設定された「**エラーの最大数**」**を確認する必要があります。**許容エラー数を表示するには、ナビゲーション・ツリーのマッピングを右クリックし、「構成」を選択して「ランタイム・パラメータ」を開きます。

マッピング前プロセス演算子

マッピング実行前に実行するプロシージャを定義するには、マッピング前プロセス演算子を使用します。たとえば、マッピング前プロセス演算子を使用して、ステージング領域に表をロードするマッピングを実行する前に、ステージング領域の表を切り捨てることができます。また、マッピング前プロセス演算子を使用して、ターゲットにデータをロードするマッピングを実行する前に、索引を無効にすることができます。その後、ターゲットにデータをロードするマッピングを実行した後で、マッピング後プロセス演算子を使用して、索引を再び有効にして作成することができます。

マッピング前プロセス演算子は、マッピング実行前に、Warehouse Builder でメタデータが定義されているファンクションまたはプロシージャをコールします。出力パラメータ・グループは戻り値の接続ポイントとなり（ファンクションとして実装されている場合）、ファンクションまたはプロシージャの出力パラメータを生成します。これら出力属性の接続には制約がありません。

マッピング前プロセス演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップすると、使用可能なライブラリ、カテゴリ、ファンクション、プロシージャを表示するダイアログが開きます。ツリーからファンクションまたはプロシージャを選択すると、定義済の入力パラメータと出力パラメータが表示されます。

マッピング前プロセス演算子では、選択した PL/SQL プロシージャまたはファンクションのパラメータ数と指示内容に対応するグループを使用します。

1 つのマッピングで使用できるマッピング前プロセス演算子は 1 つのみです。定数、マッピング入力パラメータ、マッピング前プロセスからの出力のみをマッピング後プロセス演算子にマッピングできます。

マッピング前プロセス演算子をマッピング・エディタに追加した後、演算子プロパティ・ダイアログを使用してマッピングを実行する条件を指定します。

マッピングでマッピング前プロセス演算子を使用する手順は次のとおりです。

1. マッピング前プロセス演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。

   「マッピング変換の追加」ダイアログが表示されます。

2. 「マッピング変換の追加」ダイアログで、変換を選択または作成します。「マッピング変換の追加」ダイアログの使用方法の詳細は、6-11 ページの「バインド可能な演算子の追加」を参照してください。

3. マッピング前プロセス演算子の出力属性を、ターゲット演算子の入力グループに接続します。

4. 演算子の実行条件を設定します。

マッピング前プロセス演算子を使用してマッピングの実行条件を設定する手順は次のとおりです。

1. マッピング・キャンバスで、マッピング前プロセス演算子を右クリックし、「演算子のプロパティ」を選択します。

2. 「マッピングの実行条件」をクリックして、次のいずれかの実行条件を選択します。

   ALWAYS: プロセス完了後、エラーにかかわらずマッピングが実行されます。

   ON SUCCESS: エラーなくプロセスが完了した場合にのみマッピングが実行されます。

   ON ERROR: プロセスの完了時にエラーがあった場合にのみマッピングが実行されます。

集合演算演算子

集合演算演算子では、マッピングに次の集合演算を使用できます。

- UNION（デフォルト）
- UNION ALL
- INTERSECT
- MINUS

デフォルトでは、集合演算演算子に 2 つの入力グループと 1 つの出力グループがあります。入力グループは演算子エディタを使用して追加できます。集合演算の入力グループに属性をマッピングすると、それらと同じ名前およびデータ型の属性が集合演算の出力グループに作成されます。出力グループの属性数は、最も多くの属性を含む入力グループの属性数と同じになります。

集合演算演算子を使用するには、次の条件を満たす必要があります。

- すべての集合の属性数が同じであること。
- 対応する属性のデータ型が一致すること。

対応する属性は、入力グループ内の属性の順序によって決まります。たとえば、入力グループ 1 の属性 1 は、入力グループ 2 の属性 1 に対応します。

集合演算は上から下へ順に適用する必要があります。集合演算の実行順序は、入力グループの順序によって決まります。この順序は MINUS 演算にのみ影響を与えます。たとえば、「A - B」と「B - A」は同じではありません。また、集合の構造は最初の入力グループ内の属性の順序によって異なります。たとえば、{empno, ename} は {ename, empno} と同じではありません。

マッピングで集合演算演算子を使用する手順は次のとおりです。

1. 集合演算演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. ソース属性を集合演算演算子のグループに接続します。
3. 演算子ヘッダーを右クリックし、「演算子のプロパティ ...」を選択します。

   集合演算のプロパティ・ウィンドウが表示されます。
4. 「集合演算」プロパティの右側のフィールドをクリックし、ドロップダウン・リストから演算の種類を選択します。
5. 集合演算のプロパティ・ウィンドウを閉じます。
6. 集合演算の出力グループをターゲットの入力グループに接続します。

ソーター演算子

行セットをソートするにはソーター演算子を使用します。ソーター演算子では、ソートする入力属性とソート方法（昇順または降順）を指定できます。Warehouse Builder では、マッピングで生成されたコードに ORDER BY 句を配置してデータがソートされます。

ソーター演算子では、1 つの入力 / 出力グループを使用します。ソーター演算子を使用すると、リレーショナル・データベース・ソースのデータをソートできます。ソーター演算子の後には任意の演算子を配置できます。

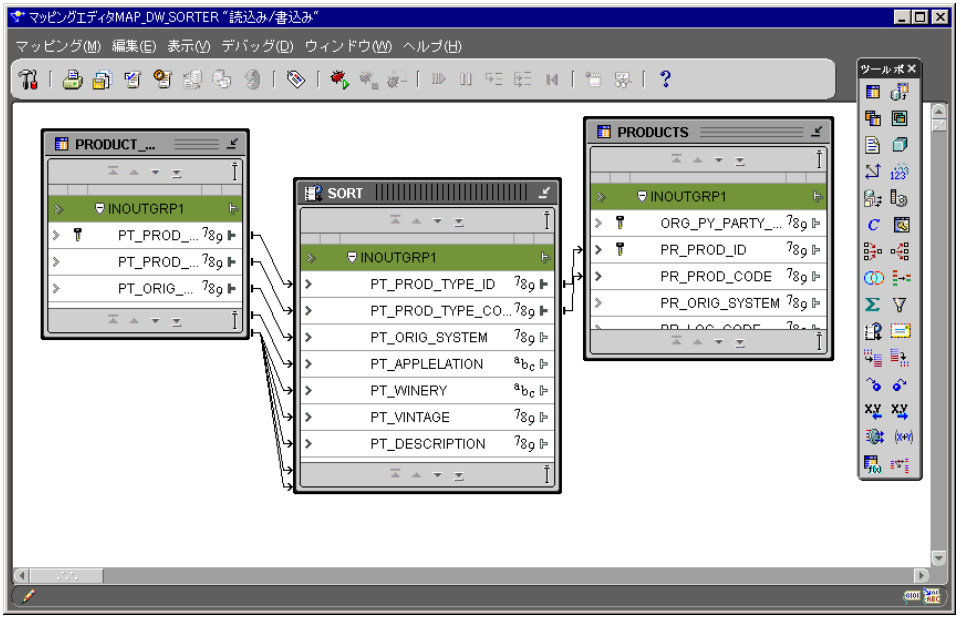
ソーター演算子では次のプロパティを指定します。

- **Order By 句 :** 入力 / 出力グループの属性の順序付きリスト。この順序付き属性リストと同じ順序でソートが実行されます。各属性について昇順または降順を設定できます。

マッピングでソーター演算子を使用する手順は次のとおりです。

1. ソーター演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. 図 8-34 に示すように、ソース演算子グループをソーター入力 / 出力グループに接続します。

図 8-34 ソーター演算子が表示されたマッピング・エディタ



3. ソーター演算子のヘッダーを右クリックし「演算子のプロパティ」を選択します。

   ソーターのプロパティ・ウィンドウが表示されます。

4. 「Order By 句」フィールドの「...」ボタンをクリックします。

   「Order By 句」ダイアログが表示されます。

5. ソートする属性を選択します。

   「使用可能な属性」リストから属性を選択し、「>」ボタンをクリックします。「使用可能な属性」リストのすべての属性を選択するには、「>>」ボタンをクリックします。

6. 属性に ORDER BY 句を適用します。

   「ORDER BY 属性」リストから属性を選択し、ドロップダウン・リストから「ASC」（昇順）または「DESC」（降順）を選択します。

7. 「OK」をクリックします。

スプリッタ演算子

1 つのソースのデータを複数のターゲットに分割するには、スプリッタ演算子を使用します。スプリッタ演算子では、ブール分割条件に基づいて、1 つの入力行セットを複数の出力行セットに分割します。各出力行セットのカーディナリティは、入力のカーディナリティと同じか、それより少なくなります。

1 つのソースのデータを複数のターゲットに分割する Warehouse Builder マッピングを構成して、SQL コードを最適化してパフォーマンスを向上させる Oracle9*i* のマルチテーブル・インサート機能を利用できます。詳細は、8-105 ページの「例 : 複数のターゲットを使用したマッピングの作成」を参照してください。

スプリッタ演算子では、REMAINING_ROWS という出力グループが作成されます。この出力グループには、他のどの出力グループにも属さないすべての入力行が含まれます。REMAINING_ROWS 出力グループを削除することはできますが、編集はできません。

スプリッタ演算子では次のプロパティを指定します。

- **分割条件 :** 分割条件のテキスト式テンプレート。コード生成時には、この式テンプレートの入力属性名がソース列で置換されます。この式は、WHERE 句で使用できる有効な SQL 式です。
- **データ型 :** 属性のデータ型。
- **精度 :** 属性の精度。数値型の属性でのみ使用します。
- **スケール :** 属性のスケール。数値型の属性でのみ使用します。
- **長さ :** 属性の長さ。文字列型の属性でのみ使用します。

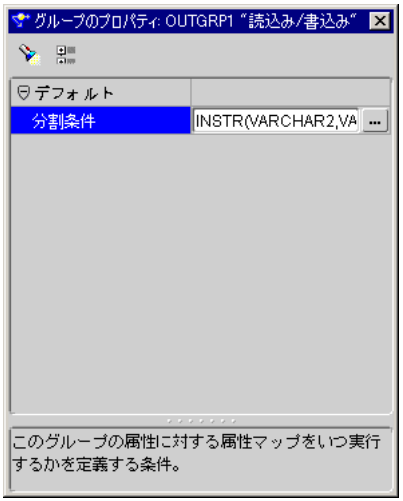
マッピングでスプリッタ演算子を使用する手順は次のとおりです。

1. スプリッタ演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. ソース演算子のグループをスプリッタ演算子の入力グループに接続します。

   対応する入力データと同じデータ型の出力属性が作成されます。
3. スプリッタ演算子のヘッダーを右クリックし「演算子のプロパティ」を選択します。

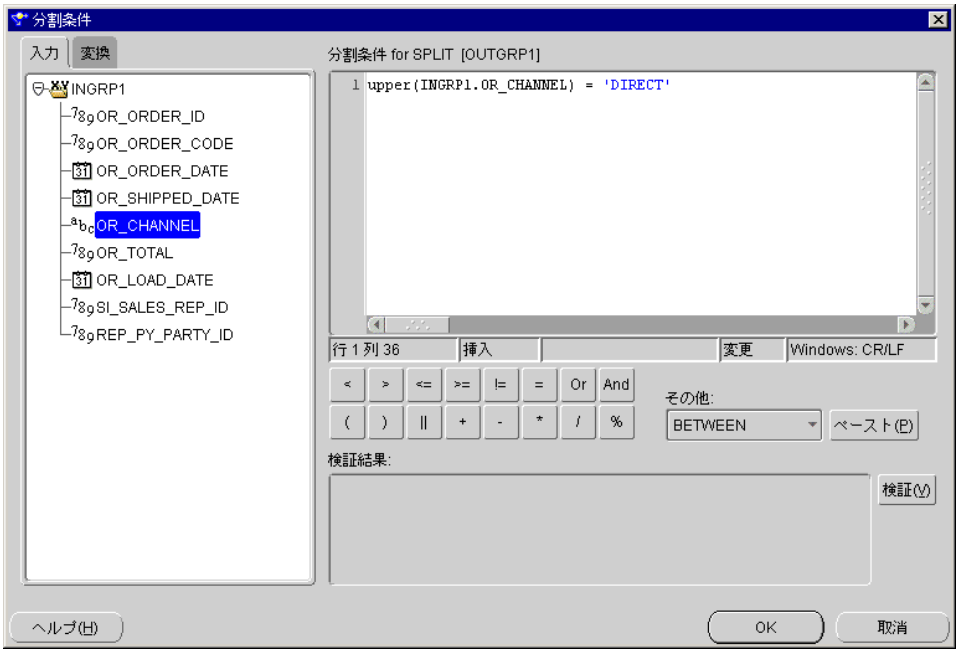
   図 8-35 に示すように、スプリッタのプロパティ・ウィンドウが表示されます。

図 8-35 分割条件を指定する「グループのプロパティ」ウィンドウ



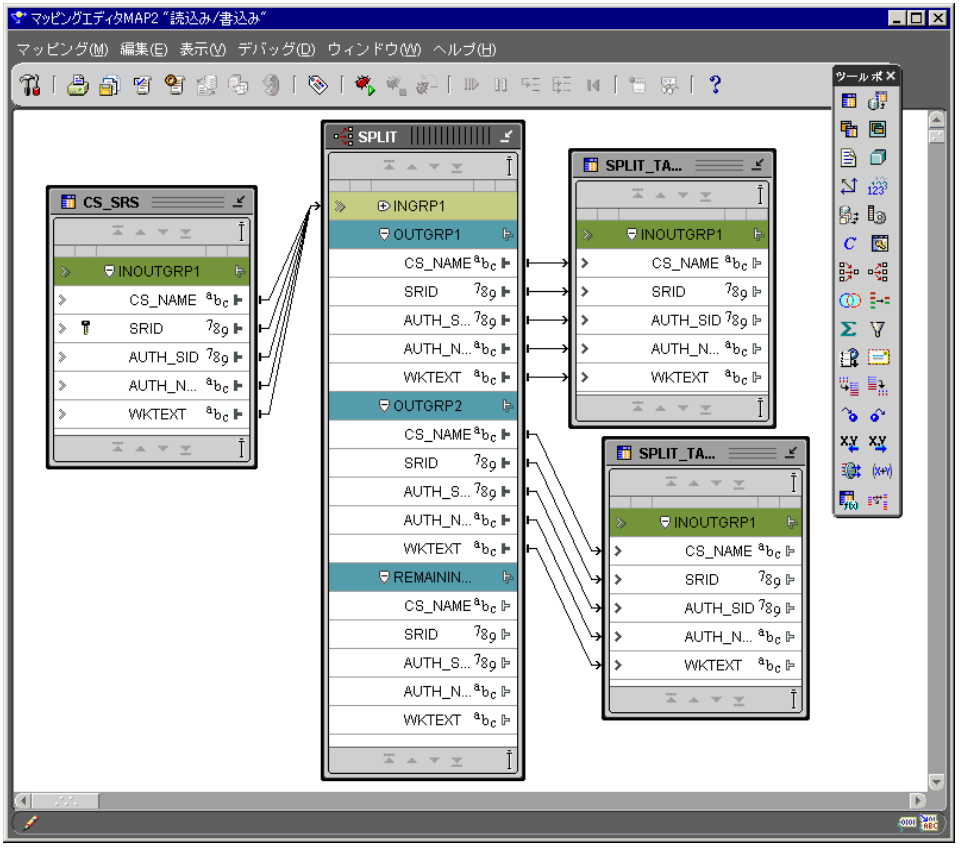
4. 「分割条件」フィールドに式を入力します。または、「...」をクリックし、図 8-36 に示すように Expression Builder で式を定義します。

図 8-36 分割条件が表示された Expression Builder



5. スプリッタのプロパティ・ウィンドウを閉じます。
6. REMAINING ROWS グループ以外の各出力グループについて式を定義します。
7. 出力グループをターゲットに接続します。

図 8-37 1 つのソースと複数のターゲットとのマッピング



例 : 複数のターゲットを使用したマッピングの作成

スプリッタ演算子を使用してマッピングを設計し、構成すると、Oracle9*i* Database のマルチテーブル・インサート機能を使用した SQL 文が生成されます。この SQL 文は、Oracle9*i* Database サーバーのパラレル問合せとパラレル DML サービスを利用します。

複数のターゲットを使用してマッピングを作成する手順は次のとおりです。

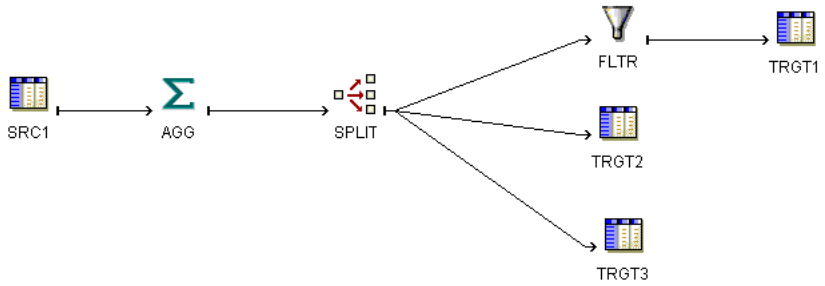
1. マッピングが含まれた Oracle ターゲット・モジュールを構成して、Oracle9*i* SQL を検証し、生成します。

   Warehouse Builder のナビゲーション・ツリーでターゲット・モジュールを右クリックし、「構成プロパティ」を選択します。「PL/SQL 生成モード」で、「Oracle9i」を選択します。

2. マッピング・エディタで、1 つのソース、スプリッタ演算子と複数のターゲットを使用したマッピングを設計します。

   ターゲットは、ビューやマテリアライズド・ビューではなく、表にする必要があります。各ターゲット表の列数は、999 未満である必要があります。スプリッタ演算子とターゲット間には、カーディナリティが変わる演算子を使用しないでください。たとえば、図 8-38 に示すように、スプリッタとターゲット間にフィルタは配置できますが、結合子や集計子は配置できません。

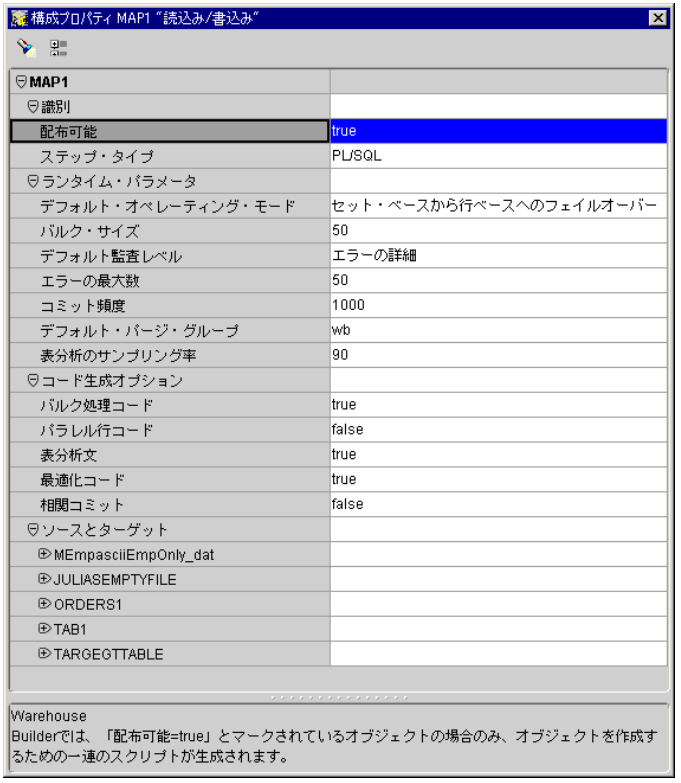
図 8-38 複数のターゲットを使用したマッピングの例



3. Warehouse Builder コンソールのナビゲーション・ツリーでマッピングを選択し、メニュー・バーの「オブジェクト」を選択してから「構成」を選択します。構成するマッピングを右クリックして、「構成」を選択することもできます。

   図 8-39 に示すように、マッピングの「構成プロパティ」ダイアログが表示されます。

図 8-39 表のマッピング用の「構成プロパティ」ウィンドウ



4. **ランタイム・パラメータ・リファレンスを開き、「デフォルト・オペレーティング・モード」を「セット・ベース」に設定します。**

5. **コード生成オプションのリファレンスを開き、「最適化コード」を「true」に設定します。**

   このマッピングを実行して、生成結果を表示すると、すべてのターゲットに対する SELECT と INSERT の合計数が返されます。

テーブル・ファンクション演算子

入力行セットを操作して、カーディナリティが異なる別の行セットを戻すには、テーブル・ファンクション演算子を使用します。従来のファンクションとは異なり、テーブル・ファンクションは、物理表のように問合せを行える出力行セットを返すことができます。

テーブル・ファンクションを使用すると、データ・ウェアハウスをロードするときに、パフォーマンスが大幅に向上します。

テーブル・ファンクションには、次のような特徴があります。

- テーブル・ファンクションは、名前別にパラメータを渡しません。
- 戻り型が TABLE of PLS Record の場合、選択する名前は PLS レコード・フィールドの名前と一致する必要があります。選択リストの PLS レコード・フィールドから 1 つのサブセットのみを選択できます。
- 戻り型が TABLE of T1%ROWTYPE の場合、選択する名前は表 T1 の列名と一致する必要があります。
- 戻り型が TABLE of Object Type の場合、選択する名前はオブジェクト・タイプの属性名と一致する必要があります。
- 戻り型が TABLE of Scalar（TABLE of NUMBER と同様）の場合、テーブル・ファンクションによって返されるスカラー値の取得には、Select COLUMN_VALUE のみを使用できます。

テーブル・ファンクション演算子の使用における前提条件

マッピングでテーブル・ファンクションのマッピング演算子を使用するには、Warehouse Builder の外部でターゲットのテーブル・ファンクションを作成する必要があります。バインドのないテーブル・ファンクション演算子でサポートされているデータベースのテーブル・ファンクションは、次の条件を満たす必要があります。

入力

- PLS レコード（PLS レコード・フィールド）を返す REF カーソルは、Warehouse Builder でサポートされるスカラー・データ型（0..n）である必要があります。
- 少なくとも 1 つの入力パラメータが必要です。

出力

戻り型 :

- TABLE of PLS Record（PLS レコード・フィールドは Warehouse Builder でサポートされるスカラー・データ型である必要があります）
- TABLE of Object Type（オブジェクト・タイプの属性は、Warehouse Builder でサポートされるスカラー・データ型である必要があります）
- TABLE of Scalar（Warehouse Builder でサポートされるスカラー・データ型）
- TABLE of ROWTYPE

マッピングのバウンドのないテーブル・ファンクションのマッピング演算子の場合 :

- 各 REF カーソル・タイプ・パラメータに 1 つのパラメータ・グループを追加する必要があります。
- 複数のスカラー・パラメータは、1 つのスカラー型パラメータ・グループの一部になることができます。
- パラメータ・グループとグループ内のパラメータは、任意の順序で入力できます。
- テーブル・ファンクションのマッピング演算子内のパラメータの位置は、ターゲット・ウェアハウスで作成するテーブル・ファンクションのパラメータの位置と同じである必要があります。

テーブル・ファンクション演算子のプロパティ

テーブル・ファンクションのマッピング演算子では次のプロパティを指定します。

一般

「一般」プロパティには、名前と説明があります。ターゲット・データベースにあるテーブル・ファンクションの名前を指定します。この説明はオプションです。

入力パラメータ・グループ

テーブル・ファンクション演算子には、次のタイプの入力パラメータを使用します。

- **入力パラメータ・タイプ :** 有効な入力パラメータ・タイプは、REF_CURSOR または SCALAR です。
- **REF_CURSOR:** PLS レコード {0...N} を返します。PLS レコード・フィールドは、Warehouse Builder でサポートされるスカラー・データ型である必要があります。
- **SCALAR:** Warehouse Builder でサポートされるスカラー・データ型です。
- **パラメータの位置 :** このパラメータ・グループに対応するテーブル・ファンクション・シグネチャのパラメータの位置を示します。

入力パラメータ

- **パラメータの位置 :** テーブル・ファンクション・シグネチャのパラメータの位置です。このプロパティは、スカラー・パラメータにのみ適用されます。

出力パラメータ・グループ

- **スカラー・タイプの表を戻します :** このプロパティは、テーブル・ファンクションの戻り型が TABLE of SCALAR であるかどうかを指定します。この情報は必須です。TABLE of SCALAR の選択リスト項目は、他の場合に妥当な名前があっても、Select COLUMN_VALUE である必要があるからです。

出力パラメータ

- **タイプ属性名 :** PLS レコード・フィールドの名前、オブジェクト・タイプの属性名、または ROWTYPE の列名です。このプロパティは、戻り型が TABLE of SCALAR である場合は適用されません。この名前を使用して、テーブル・ファンクションが起動されます。

マッピングでテーブル・ファンクション演算子を使用する手順は次のとおりです。

1. テーブル・ファンクションのマッピング演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。
2. 適切なソース属性を、テーブル・ファンクションのマッピング演算子の入力グループに接続します。

   これによって、入力属性が自動的に作成されます。
3. テーブル・ファンクション演算子を右クリックして、「編集」を選択します。
4. 「グループ」タブで「追加」を選択し、出力グループを追加します。

テーブル・ファンクションのマッピング演算子が含まれるマッピングを配布する前に、ターゲット・ウェアハウスでテーブル・ファンクションを手動で作成する必要があります。テーブル・ファンクションのマッピング演算子は、マッピングで生成されるコードにより、実際のテーブル・ファンクション・オブジェクトにバインドされます。

変換演算子

変換のマッピング演算子を使用すると、入力行セットのカーディナリティを維持しながら、PL/SQL ファンクションに基づいて行セット内の行の列値を変換できます。

変換のマッピング演算子は、リポジトリのいずれかのモジュールに含まれているファンクションまたはプロシージャにバインドされている必要があります。変換のマッピング演算子の入力と出力は、バインドされたリポジトリ・ファンクションまたはプロシージャの入力パラメータと出力パラメータに対応しています。変換のマッピング演算子がファンクションにバインドされている場合は、そのファンクションの結果に対応する演算子に結果の出力が追加されます。バインド先のファンクションまたはプロシージャがターゲット・システムに存在しない場合は、マッピングを配布する前に、そのファンクションまたはプロシージャを生成して、配布する必要があります。

Warehouse Builder では、ランタイム・スキーマに PL/SQL ライブラリ・ファンクションが事前定義されています。変換のマッピング演算子をマッピングに追加すると、これらのファンクションをバインド先のファンクションとして選択できます。さらに、グローバル共有ライブラリのファンクションやプロシージャも選択できます。

変換のマッピング演算子では次のプロパティを指定します。

- **ファンクション名 :** この演算子がバインドされているファンクションまたはプロシージャの名前。
- **プロシージャ :** 行ベース・モードを使用して、変換がロードされたかどうかを示します。セット・ベース・モードを使用できる変換と、行ベース・モードを使用できる変換があります。
- **データ型 :** 各属性に対応するバインド先ファンクションの入力、出力、または結果パラメータのデータ型を示します。変換のマッピングの出力が CHAR データ型の場合は、データをターゲットに移動する前に、Warehouse Builder によってその結果に RTRIM が適用されます。これにより、余分なスペースが出力結果に含まれなくなります。
- **デフォルト値 :** 属性のデフォルト値（デフォルト値がない場合は空白）。
- **オプション :** このブール値が true の場合は、その属性がオプションであることを示しています。オプションの属性をマッピングで接続する必要はありません。
- **ファンクション戻り値 :** このブール値が true の場合は、その出力属性がファンクションの結果の属性であることを示します。結果属性は名前付きの結果です。別の出力が名前付きの結果の場合、または結果の出力の名前を変更する場合に、このプロパティを使用します。

マッピングで変換のマッピング演算子を使用する手順は次のとおりです。

1. 変換のマッピング演算子をマッピング・エディタのキャンバスにドラッグ・アンド・ドロップします。

   「マッピング変換の追加」ダイアログが表示されます。

2. 「マッピング変換の追加」ダイアログで、新しい変換を作成するか、1 つ以上の変換を選択します。選択した各変換には、Warehouse Builder によって変換演算子が追加されます。これらのオプションの詳細は、6-11 ページの「バインド可能な演算子の追加」を参照してください。

3. ソース属性を、変換のマッピング演算子の入力に接続します。

4. （オプションの手順）いずれかの入力を右クリックして、「属性のプロパティ」を選択します。

   変換のマッピングのプロパティ・ウィンドウが表示されます。

5. 入力属性を選択します。「プロシージャ」プロパティが「True」に設定されている場合は、入力パラメータを接続しません。

6. 「属性のプロパティ」ウィンドウを閉じます。

7. 変換のマッピング演算子の出力属性をターゲット属性に接続します。

アンピボット演算子

アンピボット演算子では、複数の入力行が 1 つの出力行に変換されます。アンピボット演算子を使用すると、ソース・データ内の属性ごとにグループ化されているソース行セットからソースをいったん抽出し、そこから 1 つの行を作成できます。ピボット演算子と同様に、アンピボット演算子はマッピング内の任意の場所に配置できます。

例 : 販売データのアンピボット

表 8-24 では、SALES リレーショナル表のデータ・サンプルを示します。クロス集計フォーマットの 'MONTH' 列には、年の各月に対する 12 の文字値があります。販売数値はすべて、1 列の 'MONTHLY_SALES' に含まれています。

表 8-24 クロス集計フォーマットのデータ

REP	MONTH	MONTHLY_SALES	REGION
0675	Jan	10.5	4
0676	Jan	9.5	3
0679	Jan	8.7	3
0675	Feb	11.4	4
0676	Feb	10.5	3
0679	Feb	7.4	3
0675	Mar	9.5	4
0676	Mar	10.3	3
0679	Mar	7.5	3
0675	Apr	8.7	4
0676	Apr	7.6	3
0679	Apr	7.8	3

図 8-40 では、Warehouse Builder によって表がアンピボットされた後のリレーショナル表 'SALES' のデータを示します。'MONTH' 列に格納されていたデータ（Jan、Feb、Mar...）は、12 の個別の属性（M1、M2、M3...）に対応しています。'MONTHLY_SALES' に格納されていた販売数値は、各月の 12 の属性に配分されています。

図 8-40 クロス集計フォーマットからアンピボットされたデータ

ID	Reg	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
0675	4	10.5	11.4	9.5	8.7	7.4	7.5	7.8	9.7				
0676	3	9.5	10.5	10.3	7.6	8.0	7.8	8.7	8.9				
0679	3	8.7	7.4	7.5	7.8	9.7	10.3	7.6	8.0				
0683	2	9.5	10.5	10.3	9.5	8.7	7.4	7.8	8.7				
0684	1	11.4	9.5	8.7	7.4	7.5	10.3	9.5	8.7				
0687	1	9.5	8.7	7.4	7.8	8.7	7.4	7.5	7.8				
0690	1	8.7	7.4	7.8	8.7	11.4	9.5	8.7	7.4				

行ロケータ

アンピボット演算子を使用すると、複数の入力行が行ロケータに基づいて 1 つの行に変換されます。アンピボット演算子における行ロケータは、ソースから選択し、定義した出力属性セットと対応する属性です。アンピボット演算子には行ロケータが必要です。この例では、行ロケータは 'SALES' 表の 'MONTH' であり、アンピボットされた出力の属性 M1、M2、M3...M12 に対応します。

アンピボット演算子の使用方法

アンピボット演算子を使用する際には次のオプションがあります。

- **新しいアンピボット演算子を定義する :** ツールボックスのアンピボット演算子をマッピングにドラッグ・アンド・ドロップします。マッピング・エディタでウィザードが起動します。
- **既存のアンピボット演算子を編集する :** 演算子を右クリックして、「編集」を選択します。マッピング・エディタでアンピボット・エディタが起動します。

アンピボット演算子ウィザードまたはアンピボット・エディタのどちらを使用する場合も、次の各ページで指定します。

- 一般
- グループ
- 接続の入力
- 入力属性
- 行ロケータ
- 出力属性
- アンピボット変換

一般

「一般」ページで、アンピボット演算子の名前と説明（オプション）を指定します。ウィザードでは、演算子に "Unpivot" というデフォルト名が付きます。

グループ

「グループ」ページで、1 つの入力グループと 1 つの出力グループを指定します。

アンピボット演算子では、入力グループは、クロス集計フォーマットのソース・データを示します。出力グループは、複数の属性に配分されたターゲット・データを示します。

入力グループや出力グループの名前を変更したり、説明を追加したりできます。各アンピボット演算子には、1 つの入力グループと 1 つの出力グループが必要であるため、ウィザードでグループを追加または削除したり、グループの方向を変更したりできません。

接続の入力

「接続の入力」ページで、属性を選択し、アンピボット演算子にコピーおよびマッピングします。

アンピボット演算子の「接続の入力」ページで指定する手順は次のとおりです。

1. 左側のパネルで、グループ全体または個別の属性を選択します。

   名前を使用して特定の属性またはグループを検索するには、「検索」にテキストを入力し、「実行」をクリックします。次の一致文字列を検索するには、「実行」を再度クリックします。

   複数のグループまたは属性を選択するには、[Shift] キーを押しながら選択します。異なるグループの属性を選択する場合は、結合子演算子または集合演算子を使用してグループを結合する必要があります。

2. ページ中央にある「>」ボタンを使用して、選択した項目をウィザードの右側に移動します。

   入力接続リストからグループまたは属性を移動するには「<」ボタンを使用します。Warehouse Builder では、選択した項目が入力グループから削除され、ソース演算子とピボット演算子間のマッピング線も削除されます。

入力属性

「入力属性」ページで、「接続の入力」タブまたはウィザード・ページで選択した属性を変更します。

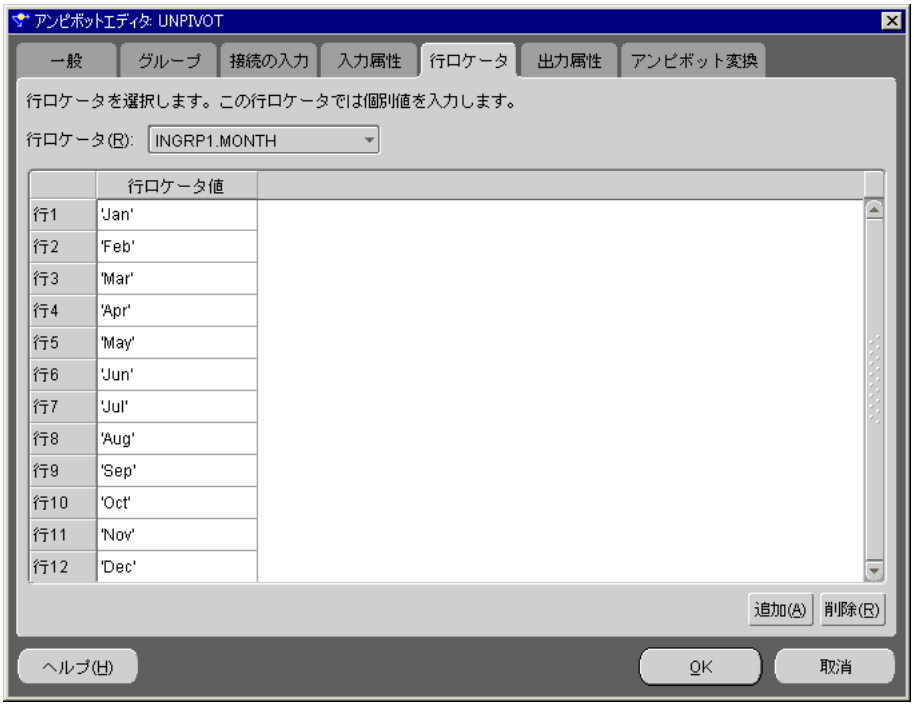
アンピボットの「入力属性」ページでは、次のタスクを実行できます。

- **属性を追加する :** 「追加」ボタンを使用して入力属性を追加します。
- **属性のプロパティを変更する :** 属性名、データ型、長さ、精度、スケールを変更できます。
- **説明（オプション）を追加する :** 入力属性の説明を入力します。
- **キー属性を指定する :** アンピボット演算子には 1 つ以上のキー属性を指定する必要があります。「キー」チェック・ボックスを使用して、入力グループを一意に識別する属性を指定します。キー属性の同じ値を持つ入力行から、アンピボットされた 1 つの出力行が生成されます。

行ロケータ

「行ロケータ」ページで、行ロケータを選択し、行ロケータに含まれる個別の値に値を割り当てます。図 8-41 では、行ロケータとして選択された属性 MONTH と、'Jan'、'Feb'、'Mar' などの値を示します。

図 8-41 アンピボットの「行ロケータ」ページ



アンピボットの「行ロケータ」ページで指定する手順は次のとおりです。

1. 「行ロケータ」リスト・ボックスから属性を選択します。

   アンピボット演算子における行ロケータは、出力属性セットに対応するソース・データの属性です。

2. 「追加」を使用して、行ロケータに存在する個別の値の数を指定します。

3. 各行ロケータの値には、ソース・データセットに表示されている値を入力します。

   文字列値の場合、一重引用符でテキストを囲みます。たとえば、行ロケータが 'MONTH' の場合、その属性には合計 12 の個別の値があります。「追加」をクリックして、各個別の値の行を追加します。行ロケータ値には、ソース・データセットに表示される値をそのまま入力します。たとえば、表 8-24 には、'Jan'、'Feb'、および 'Mar' の行ロケータ値があります。

出力属性

図 8-42 に示す「出力属性」ページで、アンピボット演算子の出力属性を作成します。

図 8-42 アンピボットの「出力属性」ページ

「入力属性」タブまたはウィザード・ページで入力属性をキーとして指定した場合、Warehouse Builder では、これらの属性が編集または削除できない出力属性として表示されます。

アンピボットの「出力属性」ページでは、次のタスクを実行できます。

- **属性を追加する :**「追加」を使用して、「行ロケータ」タブまたはウィザード・ページで指定した行に対応する出力属性の数を追加します。12 の行を指定した場合は、12 の出力属性と、キーとして指定しなかった他の入力属性の属性を指定します。
- **属性のプロパティを変更する :**「入力属性」タブまたはウィザード・ページでキーとして指定した属性を除いて、属性名、データ型、長さ、精度およびスケールを変更できます。
- **説明（オプション）を追加する :** 出力属性の説明を入力します。

アンピボット変換

図 8-43 に示す「アンピボット変換」ページで、各出力属性の式を記述します。

図 8-43「アンピボット変換」ページ

キーとして指定した属性の場合、Warehouse Builder によって一致する行と式が定義されます。Warehouse Builder には、最初の行にキー属性の一致する行が表示されます。その他の出力属性の場合、一致する行および式を指定します。

- **一致する行 :** リスト・ボックスで適切なオプションを選択します。たとえば、年の最初の月として 'M1' を定義した属性の場合、リスト・ボックスから 'Jan' を選択します。
- **式 :** リスト・ボックスで適切な式を選択します。データをアンピボットするために作成したすべての新しい属性には、対応するデータが含まれる同じ入力属性を選択します。たとえば、アンピボット属性 M1、M2、M3...M12 は、同じ式である INGRP1.MONTHLY_SALES を共有します。その他のすべての出力属性には、入力属性の一覧から対応する属性を選択します。

9

変換の使用方法

マッピングとプロセス・フローを設計する場合、特化された変換を使用して、データを変換します。この章では、変換の定義をインポートする方法、およびカスタム変換を作成する方法について説明します。

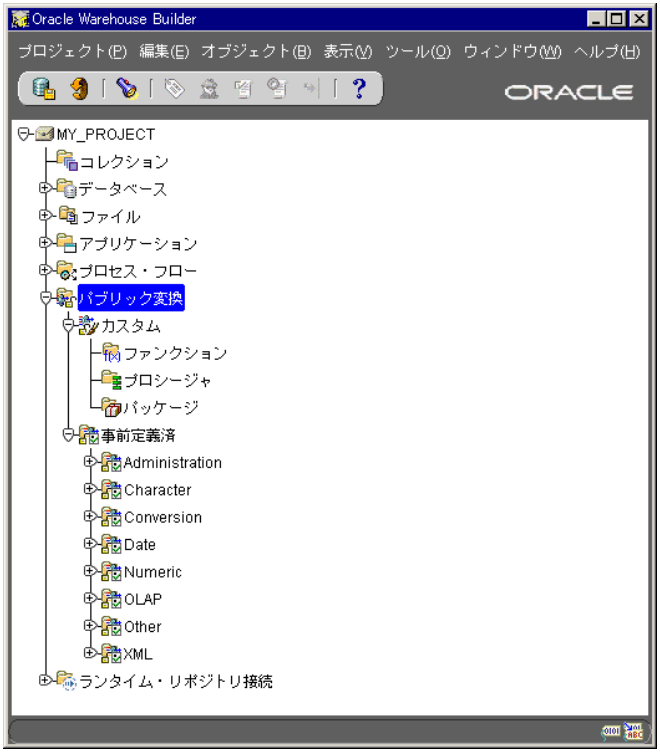
この章では、次のトピックについて説明します。

変換について

変換とは、データの変換を可能にする PL/SQL ファンクション、プロシージャおよびパッケージなどです。Warehouse Builder では、Oracle ライブラリに事前定義済の変換セットがあります。また、新規変換ウィザードを使用して、スタンドアロンのファンクション、プロシージャまたはパッケージを定義するカスタム変換を作成できます。ETL プロセスを定義するマッピングとプロセス・フローを設計する場合は、変換を使用します。

Warehouse Builder ナビゲーション・ツリーで、「パブリック変換」ノードを開き、図 9-1 に示すように、Warehouse Builder で使用可能な次のタイプの変換を表示します。

図 9-1 パブリック変換フォルダ



カスタム変換について

カスタム・ライブラリは、リポジトリにある異なるウェアハウス・モジュール間で共有される変換を格納します。次のカテゴリが含まれます。

- **ファンクション:** ファンクション・カテゴリは「パブリック変換」ノードで使用できます。このカテゴリには、スタンドアロン・ファンクションが含まれます。これらのファンクションは、ユーザーが定義するか、またはデータベースからインポートします。
- **プロシージャ:** プロシージャ・カテゴリは「パブリック変換」ノードで使用できます。このカテゴリには、変換として使用されるスタンドアロン・プロシージャがすべて含まれます。これらのプロシージャは、ユーザーが定義するか、またはデータベースからインポートします。プロシージャ変換は、任意の数（0 を含む）の入力パラメータを受け取って任意の数（0 を含む）の出力パラメータを生成します。
- **パッケージ:** PL/SQL パッケージは、Warehouse Builder で作成またはインポートされます。パッケージ本体は変更できます。パッケージ・ヘッダーはファンクションまたはプロシージャに対する署名で、変更できません。パッケージは、変換ライブラリのプロパティ・シートに表示できます。

これらの変換カテゴリは、「変換」ノードの各ウェアハウス・モジュール内でも使用できます。ウェアハウス・モジュール内にファンクションとプロシージャを作成する場合、同じリポジトリにあるプロジェクト間でこれらを共有することはできません。

事前定義済変換について

Warehouse Builder には、共通の変換を迅速かつ簡単に実行できる、変換の事前定義済カテゴリも用意されています。これらの組込みファンクションおよびプロシージャには、次のカテゴリに編成される標準の変換セットが含まれます。

- Administration
- Character
- Conversion
- Date
- Numeric
- OLAP
- Other
- XML

事前定義済変換の詳細は、『Oracle Warehouse Builder トランスフォーメーション・ガイド』を参照してください。

カスタム変換の定義

カスタム変換は、新規変換ウィザードを使用して作成できます。

カスタム変換を定義する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「パブリック変換」ノードを開いてから、「カスタム」ノードを開きます。Oracle ウェアハウス・モジュールのノードを開いてから、「変換」ノードを開くこともできます。
2. カテゴリを右クリックし、「ファンクションの作成」、「プロシージャの作成」または「パッケージの作成」を選択します。

   「新規変換ウィザード: ようこそ」ページが表示されます。
3. 「次へ」をクリックします。

   「新規変換ウィザード: 名前」ページが表示されます。
4. 新規変換の名前と説明を入力します。
5. ファンクションの戻り型を、ドロップダウン・リストから選択します。
6. 「次へ」をクリックします。

   図 9-2 に示すように、「新規変換ウィザード: パラメータ」ページが表示されます。
7. 変換の各パラメータを次のように定義します。
   a. 「追加」をクリックします。
   b. 「名前」列にパラメータの名前を入力します。
   c. タイプ、順番、入力パラメータ、出力パラメータ、または入力 / 出力パラメータのうちのどれか、および必須かどうかを指定します。

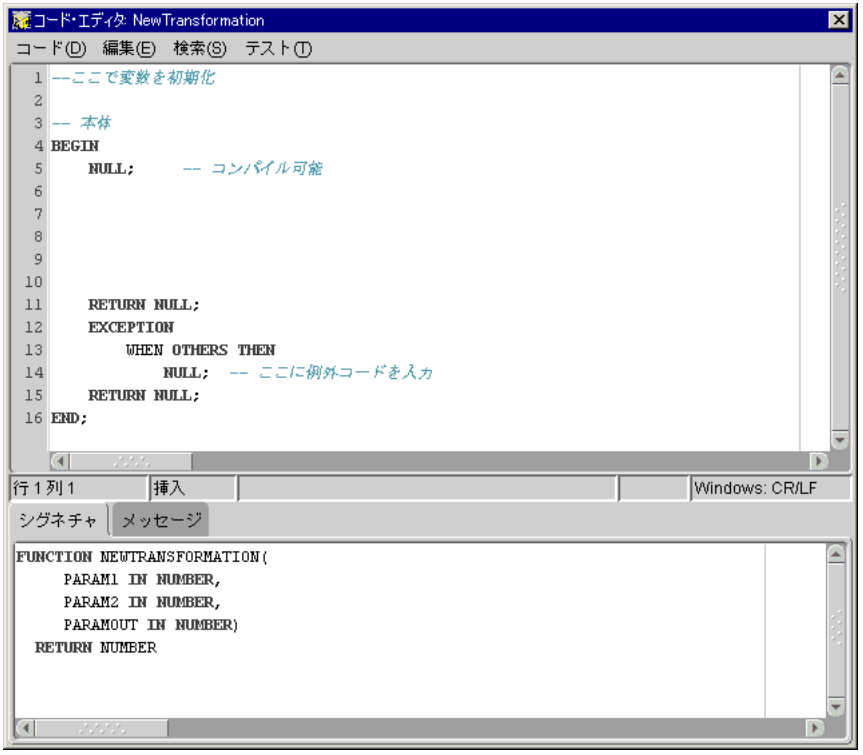
図 9-2「新規変換ウィザード : パラメータ」ページ

8. 「次へ」をクリックします。

   図 9-3 に示すように、「新規変換ウィザード : 実装」ページが表示されます。

9. 「コード・エディタ」をクリックしてコード・エディタを表示します。コード・エディタには、行番号、検索、配置および構文チェックがあります。

図 9-3 新規変換用のコード・エディタ



10. コード・エディタを閉じ、「次へ」をクリックします。

    「新規変換ウィザード: 終了」ページが表示されます。

11. 「終了」をクリックします。

    ファンクション、プロシージャまたはパッケージが作成され、ナビゲーション・ツリーの「パブリック変換」ノードおよび「カスタム」ノードにある対応するフォルダに表示されます。

PL/SQL のインポート

インポート・ウィザードを使用して、PL/SQL のファンクション、プロシージャおよびパッケージを Warehouse Builder プロジェクトにインポートできます。

次の手順は、他のソースから Warehouse Builder に PL/SQL パッケージをインポートする方法です。

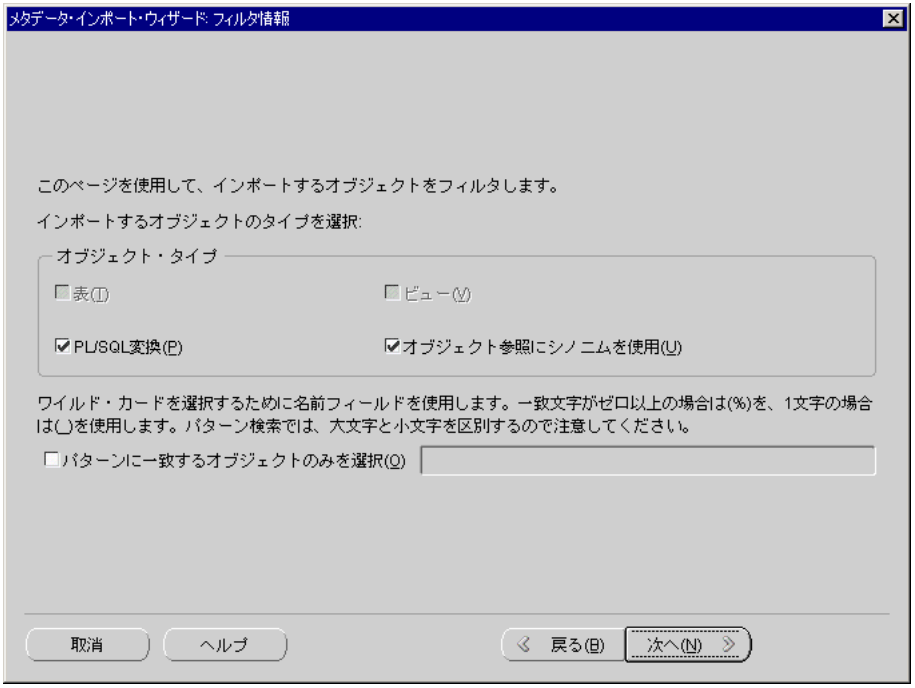
PL/SQL のファンクション、プロシージャまたはパッケージをインポートする手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「パブリック変換」ノードを開きます。
2. 「カスタム」ノードを右クリックし、「インポート」を選択します。または、ウェアハウス・モジュール名を右クリックして、「インポート」を選択します。

   「データベース・リンク情報」ダイアログが表示されます。
3. PL/SQL パッケージのインポート元であるシステムへの新しいデータベース・リンクを作成します。または、リストからデータベース・リンクを選択します。
4. 「OK」をクリックします。

   「メタデータ・インポート・ウィザード: ようこそ」ページが表示されます。
5. 「次へ」をクリックします。
6. 図 9-4 に示すように、「メタデータ・インポート・ウィザード: フィルタ情報」ページの「オブジェクト・タイプ」フィールドで、「PL/SQL 変換」を選択します。

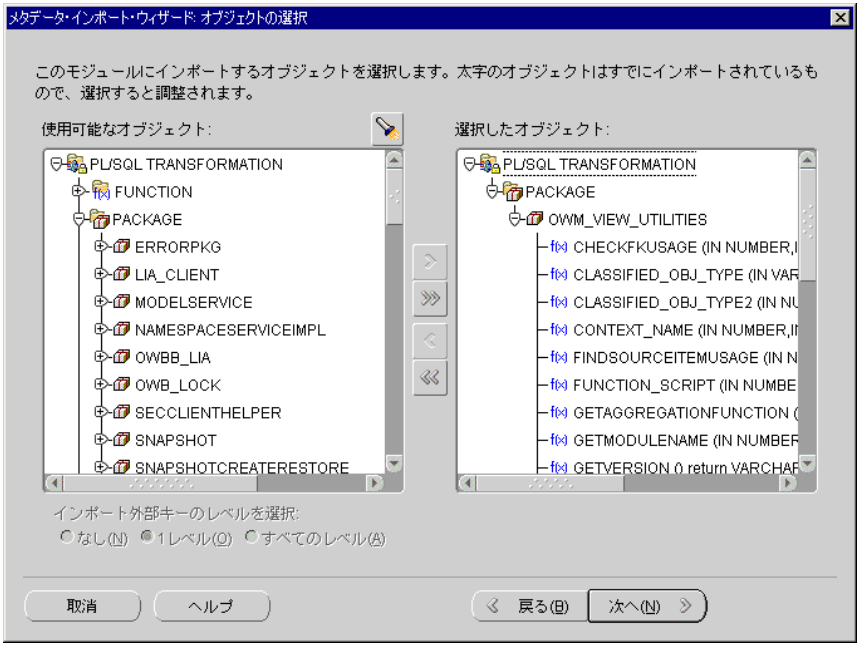
図 9-4 PL/SQL 変換の選択



7. 「次へ」をクリックします。

   図 9-5 に示すように、「メタデータ・インポート・ウィザード : オブジェクトの選択」ページが表示されます。

図 9-5「メタデータ・インポート・ウィザード : オブジェクトの選択」ページ



8. 「使用可能なオブジェクト」リストから、ファンクション、プロシージャまたはパッケージを選択します。「>」ボタンをクリックして任意のオブジェクトを移動するか、「>>」ボタンをクリックしてすべてのオブジェクトを移動し、オブジェクトを「選択したオブジェクト」リストに移動します。

   ファンクションが DETERMINISTIC の場合 :

   このヒントによって、ファンクション・コールが冗長にならないようにできます。ストアド・ファンクションが以前に同じ引数でコールされている場合、前の結果が使用できます。ファンクションの結果は、セッション変数またはスキーマ・オブジェクトの状態に依存しません。そうでない場合、結果はコールによって異なります。ファンクションベースの索引またはクエリー・リライトが有効なマテリアライズド・ビューからコールすることができるのは、DETERMINISTIC ファンクションのみです。

   パラレル実行のファンクションを使用可能にする場合 :

   このオプションは、ストアド・ファンクションがパラレル DML 評価の子セッションで、確実に使用できるかどうかを宣言します。メイン（ログオン）セッションの状態が、子セッションと共有されることはありません。子セッションには、それぞれセッションの開始時に初期化される各自の状態があります。ファンクションの結果は、セッション（静的）変数の状態に依存しません。そうでない場合、結果はセッションによって異なります。

9. 「次へ」をクリックします。

   図 9-6 に示すように、「メタデータ・インポート・ウィザード : サマリーおよびインポート」ページが表示されます。

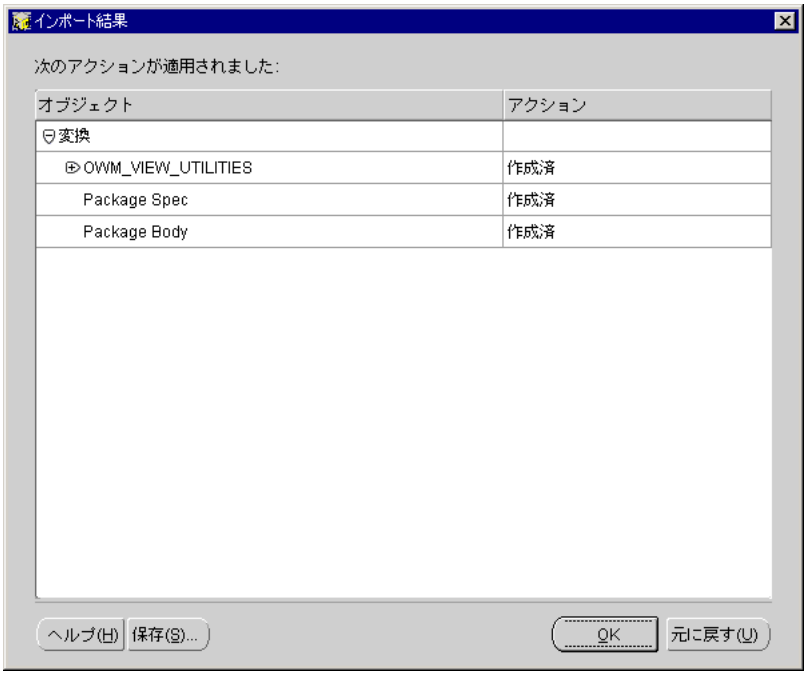
図 9-6「メタデータ・インポート・ウィザード : サマリーおよびインポート」ページ

10. インポート情報を確認します。選択した内容を変更するには、「戻る」をクリックします。

11. 「終了」をクリックしてインポートを実行します。

    図 9-7 に示すように、「インポート結果」ダイアログが表示されます。

図 9-7「インポート結果」ダイアログ



12. 「OK」をクリックしてインポートを続行します。インポート処理を取り消すには「元に戻す」をクリックします。

    インポートされた PL/SQL 情報が、ナビゲーション・ツリーの「カスタム」ノードに表示されます。

インポートされた PL/SQL を使用する際には、次の点に注意してください。

- インポートした PL/SQL のファンクションおよびプロシージャは、編集、保存および配布できます。
- ラップされた PL/SQL オブジェクトは読取りできません。
- インポートしたパッケージは、カテゴリ・プロパティ・シートで表示および変更できます。
- インポートしたパッケージの本体は編集できますが、仕様は編集できません。

変換プロパティの編集

変換プロパティ・シートで、ファンクション、プロシージャまたはパッケージを編集できます。プロパティの編集は必ず一貫して行ってください。たとえば、パラメータの名前を変更したら、実装コード内の名前も変更する必要があります。

ファンクション、プロシージャまたはパッケージを編集する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「パブリック変換」ノードを開いてから、「カスタム」ノードを開きます。
2. 編集するファンクション、プロシージャまたはパッケージの名前を右クリックし、「プロパティ」を選択します。

   変換プロパティ・シートが表示されます。パッケージの場合、パッケージの名前と説明のみが編集できます。ファンクションとプロシージャの場合は、プロパティ・シートに「名前」、「パラメータ」、「実装」の 3 つのタブが表示されます。

   名前 : ファンクションまたはプロシージャの名前と説明を編集できます。ファンクションの場合、データの戻り型も編集できます。

   パラメータ : ファンクションまたはプロシージャの新しいパラメータを編集、追加、削除できます。また、パラメータの属性を編集し、定義できます。

   実装 : パラメータの PL/SQL コードを確認します。「コード・エディタ」をクリックしてコードを編集します。

10

プロセス・フローの設計

ソースからターゲットへのデータ移動操作を定義するマッピングを作成した後に、プロセス・フローを作成し、定義できます。プロセス・フローは、マッピングと Warehouse Builder の外部アクティビティを相互に関連付けます。これら外部アクティビティには、電子メール、FTP コマンドおよびオペレーティング・システムの実行可能ファイルなどがあります。

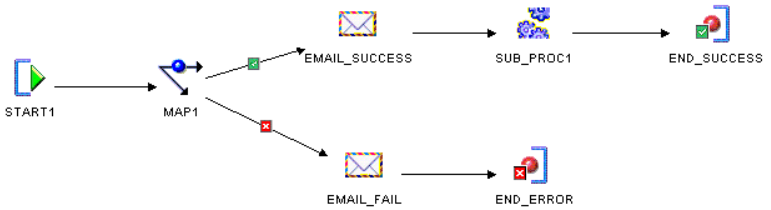
この章では、次のトピックについて説明します。

プロセス・フローについて

プロセス・フローは、マッピングと電子メール、FTP、オペレーティング・システム・コマンドなどの外部アクティビティの間に存在する依存関係を示します。

各プロセス・フローは、フローの各ストリームにある開始アクティビティで始まり、終了アクティビティで終わります。あるプロセス・フローが別のプロセス・フローを起動するように設計できます。図 10-1 は、マッピング MAP1 を起動するプロセス・フローの例を示しています。マッピングが正常に完了すると、電子メール通知 EMAIL_SUCCEED が送信され、別のプロセス・フロー SUBPROC1 が起動します。マッピングが失敗すると、電子メール EMAIL_FAIL が送信され、プロセス・フローが終了します。

図 10-1 プロセス・フローの例



Warehouse Builder でプロセス・フローを設計するときは、プロセス・フロー・エディタというインタフェースを使用します。Warehouse Builder のスクリプト言語である OMB Scripting Language を使用して、プロセス・フローの作成と定義を行うこともできます。OMB Plus スクリプト・インタフェースでプロセス・フローを作成および定義する方法の詳細は、『Oracle Warehouse Builder スクリプト・リファレンス』を参照してください。

Warehouse Builder プロセス・フローは、Workflow Management Coalition（WfMC）の XML Process Definition Language（XPDL）標準に準拠しています。プロセス・フローを作成する場合、XML ファイルは XPDL 形式で作成されます。作成された XML ファイルは、XPDL 標準に準拠しているワークフロー・エンジンで使用できます。

プロセス・フロー・モジュールについて

他のモジュールと同様、プロセス・フロー・モジュールを使用すると、共通のロケーションに配置されるオブジェクトをグループ化できます。手順は、10-3 ページの「プロセス・フロー・モジュールの作成」を参照してください。

Warehouse Builder のデプロイメント・マネージャで、プロセス・フローを Oracle Workflow または XPDL 準拠のプロセス・フロー・エンジンに配置できます。Oracle Workflow の詳細は、Oracle Workflow のマニュアルを参照してください。また、Oracle Enterprise Manager を使用して、プロセス・フローを実行しスケジュールすることができます。

プロセス・フロー・パッケージについて

プロセス・フロー・モジュールには、プロセス・フロー・パッケージが含まれます。プロセス・フロー・パッケージは、複数のプロセス・フローを 1 つの単位として配置する際のグループ化メカニズムです。実行時には、同じプロセス・フロー・パッケージ内にある他のプロセス・フローを起動する、プロセス・フローを起動できます。たとえば、図 10-2 は、プロセス・フロー SUBPROC1 を含む、プロセス・フロー PROC1 を示しています。SUBPROC1 を PROC1 に追加するには、これらのプロセス・フローが同じプロセス・フロー・パッケージ内のものである必要があります。

プロセス・フロー・パッケージの作成方法は、10-4 ページの「プロセス・フロー・パッケージの作成」を参照してください。

プロセス・フローを定義する手順

プロセス・フローを定義するには、次の手順を実行します。

1. プロセス・フロー・モジュールの作成（10-3 ページ）
2. プロセス・フロー・パッケージの作成（10-4 ページ）
3. プロセス・フロー定義の作成（10-5 ページ）
4. アクティビティの追加（10-10 ページ）
5. アクティビティの接続（10-15 ページ）
6. プロセス・フローの構成（11-40 ページ）
7. プロセス・フローの検証と生成。プロセス・フロー・コードの検証方法の詳細は、第 12 章「オブジェクトの検証」を参照してください。

プロセス・フロー・モジュールの作成

プロセス・フローを使用する前に、プロセス・フロー・モジュールを作成します。モジュールは、プロセス・フローのグループの検証、作成、配置を可能にするコンテナです。プロセス・フロー・モジュールには、プロセス・フローを含むプロセス・フロー・パッケージがあります。

プロセス・フロー・モジュールを作成する手順は次のとおりです。

1. ナビゲーション・ツリーで「プロセス・フロー」ノードを右クリックし、「プロセス・モジュールの作成」を選択します。

   「新規モジュール・ウィザード : ようこそ」ページが表示されます。

2. 「次へ」をクリックします。

   「新規モジュール・ウィザード : 名前」ページが表示されます。

3. 「新規モジュール・ウィザード: 名前」ページで、次の情報を入力します。

   モジュール名 : モジュール名は、プロジェクト内で一意にする必要があります。

   モジュール・ステータス : ステータスには、「開発」、「品質保証」または「製品」のいずれかを指定します。このステータスは、説明の目的でのみ使用されます。

   説明 : 説明テキストを入力します（オプション）。

4. 「次へ」をクリックします。

   「新規モジュール・ウィザード: ロケーション」ページが表示されます。

   ドロップダウン・ボックスからロケーションを選択するか、指定しないでおきます。ウィザードでロケーションを指定しない場合、モジュールにプロセス・フロー・パッケージを配置する前に、プロセス・フロー・モジュールのプロパティ・シートでロケーションを指定する必要があります。

5. 「次へ」をクリックします。

   「新規モジュール・ウィザード: 終了」ページが表示されます。新しいプロセス・フロー・モジュールの名前とロケーションを確認します。

6. 「終了」をクリックします。

   モジュールの定義が保存され、その名前がウェアハウス・モジュールのナビゲーション・ツリーに挿入されます。これで、プロセス・フロー・パッケージを作成できます。

プロセス・フロー・パッケージの作成

プロセス・フローのモジュールを作成すると、プロセス・フロー・パッケージを作成できるようになります。プロセス・フロー・パッケージは、相互に関連付けられるプロセス・フローを決定するグループ化メカニズムです。実行時には、同じプロセス・フロー・パッケージ内にある他のプロセス・フローを起動する、プロセス・フローを起動できます。

プロセス・フロー・パッケージを作成する手順は次のとおりです。

1. ナビゲーション・ツリーでプロセス・フロー・モジュールを右クリックし、「プロセス・フロー・パッケージの作成」を選択します。

   「プロセス・フロー・パッケージの作成」ダイアログが表示されます。

2. プロセス・フロー・パッケージの名前との説明（オプション）を入力します。

3. 「OK」をクリックします。

   これで、プロセス・フロー定義を作成できます。

プロセス・フロー定義の作成

プロセス・フローのモジュールとパッケージを作成すると、プロセス・フロー定義を作成できるようになります。

プロセス・フロー定義を作成する手順は次のとおりです。

1. ナビゲーション・ツリーでプロセス・フロー・パッケージを右クリックし、「プロセス・フローの作成」を選択します。

   「プロセス・フローの作成」ダイアログが表示されます。

2. プロセス・フローの名前と説明（オプション）を入力します。

3. 「OK」をクリックします。

   プロセス・フローが起動し、「グラフ・ビュー」タブに開始アクティビティと終了アクティビティが表示されます。

   これで、アクティビティと推移を持つプロセス・フローをモデル化できます。

プロセス・フロー・エディタについて

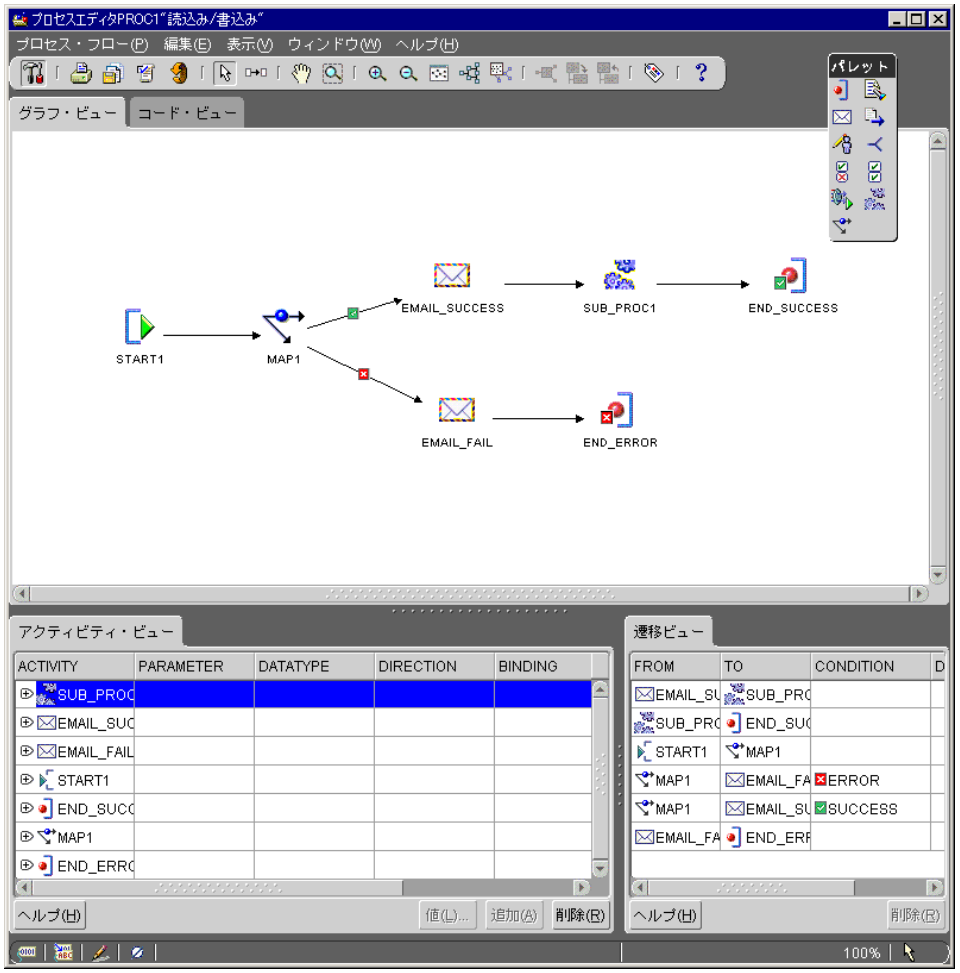
プロセス・フローのモジュールとパッケージの作成後、プロセス・フロー・エディタを使用して、プロセス・フローを作成、設計、編集します。プロセス・フロー・エディタには、ユーザーが追加する様々なアクティビティが含まれます。これらを推移に接続してフローを設計します。

アクティビティは、プロセス・フローの作業単位を表します。これらの作業単位には、Warehouse Builder の内部コンポーネントまたは外部コンポーネントを含めることができます。

推移は、アクティビティを起動する順序と条件を表します。

図 10-2 は、プロセス・フロー・エディタのサンプル・プロセス・フローを示しています。アクティビティは、キャンバスにアイコンとして表示されます。この例では、アクティビティには、開始、マッピング・アクティビティ、2 つの電子メール・アクティビティ、サブプロセスおよび 2 つの終了アクティビティが含まれます。アクティビティ間の矢印は、推移を表します。

図 10-2 プロセス・フロー・エディタ



プロセス・フロー・エディタには、次のコンポーネントがあります。

- **タイトル・バー:** エディタの最上部にあるタイトル・バーには、プロセス・フローの名前と、プロセス・フローでユーザーが持つアクセス権限が表示されます。図 10-2 では、名前は PROC1 で、ユーザーは読込み / 書込み権限を持っています。
- **メニュー・バー:** タイトル・バーの下に、プロセス・フロー・エディタのコマンドにアクセスできるメニュー・バーが表示されます。メニュー・バーにある各種のコマンドの詳細は、10-8 ページのメニュー・バーを参照してください。

- **ツールバー :** メニュー・バーの下に、よく使用するコマンドのアイコンを示すツールバーが表示されます。メニュー・バーにある各種のコマンドの詳細は、10-37 ページの表 10-14 を参照してください。
- **ツールボックス :** プロセス・フロー・エディタを最初に起動すると、ツールボックスは右上隅に表示されます。ツールボックスは、エディタの任意の場所に置けます。ツールバーの「ここをクリックするとツールパレットを切替え」アイコンをクリックして、ツールボックスの表示 / 非表示を選択できます。ツールボックスには、キャンバスにドラッグ・アンド・ドロップできる、アクティビテイ・アイコンがあります。ツールボックスの各アクティビティの詳細は、10-16 ページの表 10-2 を参照してください。
- **キャンバス :** キャンバスには、プロセス・フローの設計および変更を行うためのワークエリアがあります。プロセス・フロー・キャンバスには次の 2 つのタブがあります。「グラフ・ビュー」は、プロセス・フローをグラフ形式で表示します。「コード・ビュー」は、コードを表示します。新しいプロセスを最初に作成する場合、プロセス・フローは、開始アクティビティと終了アクティビティが表示された「グラフ・ビュー」タブを表示します。
- **「アクティビティ・ビュー」と「遷移ビュー」 :** キャンバスの下に、これらの編集ダイアログが表示されます。すべてのアクティビティと推移のプロパティを表示および編集するには、キャンバスの空白部分をクリックします。単一のアクティビティまたは推移のプロパティを表示および編集するには、ツールバーから「モードの選択」を選択し、推移またはアクティビティを選択します。
- **インジケータ・バー :** 下部パネルの「アクティビティ・ビュー」パネルと「遷移ビュー」パネルの下に、図 10-3 に示すように、モード・アイコン、インジケータおよび説明が表示されます。

図 10-3 プロセス・フロー・エディタのインジケータ・バー

左隅には、ネーミング・モード、改名モード、読込み / 書込み、検証モードが表示されます。

右隅には、倍率インジケータとナビゲーション・モードが表示されます。この図では、倍率は 140%、ナビゲーション・モードは「モードの選択」に設定されています。

キャンバスからアクティビティまたは推移を選択すると、説明（DESCRIPTION）として入力したテキストがインジケータ・バーに表示されます。上の図では、「MAP1 loads staging area tables.」と表示されます。

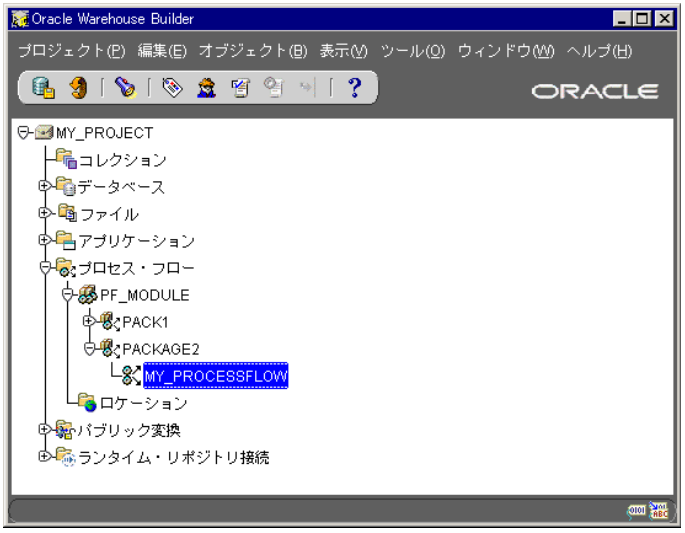
プロセス・フロー・エディタを表示する手順は次のとおりです。

1. ナビゲーション・ツリーの「プロセス・フロー」から、プロセス・フロー・モジュールを選択します。リストにプロセス・フロー・モジュールがない場合は、プロセス・フロー・モジュールを作成します。手順は、10-3 ページの「プロセス・フロー・モジュールの作成」を参照してください。

2. プロセス・フロー・モジュールからプロセス・フロー・パッケージを選択します。リストにプロセス・フロー・パッケージがない場合は、プロセス・フロー・パッケージを作成します。手順は、10-4 ページの「プロセス・フロー・パッケージの作成」を参照してください。

3. 図 10-4 に示すように、ナビゲーション・ツリーからプロセス・フローを選択します。プロセス・フロー・パッケージにプロセス・フローがない場合は、プロセス・フロー・パッケージを右クリックし、「プロセス・フローの作成」を選択します。

図 10-4 ナビゲーション・ツリーのプロセス・フロー



4. プロセス・フロー定義を作成した後、ナビゲーション・ツリーでプロセス・フローをダブルクリックして、それを開くことができます。

   または、プロセス・フローを選択し、「オブジェクト」メニューから「エディタ」を選択します。または、プロセス・フローを選択し、[Ctrl] を押しながら「O」という文字を入力します。または、プロセス・フローを右クリックし、ポップアップ・メニューから「エディタ」を選択します。

   プロセス・フロー・エディタが表示されます。デフォルトでは、プロセス・フロー・エディタが「グラフ・ビュー」をモードの選択で表示します。

プロセス・フロー・エディタでのナビゲーション

プロセス・フロー・エディタには、キャンバスでオブジェクトを移動、選択、表示するための各種ツールが含まれています。プロセス・フローの設計時によく使用するコマンドには、次のものがあります。

モードの選択 : キャンバスのオブジェクトを選択するには、モードの選択を使用します。アクティビティを選択すると、そのアクティビティが青い縁に囲まれてエディタに表示されます。アクティビティは、編集、移動または削除することができます。「アクティビティ・ビュー」でアクティビティを編集するか、右クリックしてさらにオプションを選択できます。推移を選択した場合、矢印が黒から青に変更されます。「遷移ビュー」で推移を編集するか、右クリックしてさらにオプションを選択できます。モードの選択をアクティブにするには、ツールバーのアイコンをクリックするか、メニューから「編集」と「モードの選択」を選択します。

推移を作成 : キャンバスのアクティビティを相互に接続するには、推移を作成モードを使用します。推移の作成は、描画ツールのみで行えます。推移を編集または削除するには、モードの選択に切り替えて、推移を選択します。推移を作成モードをアクティブにするには、ツールバーのアイコンをクリックするか、メニューから「編集」と「推移を作成」を選択します。

複雑なプロセス・フローをナビゲートおよび表示するには、次の便利なツールを使用します。

- パニング
- インタラクティブ・ズーム
- ズーム・イン
- ズーム・アウト
- ウィンドウに適合
- 自動レイアウト
- バーズ・アイ・ビュー
- ズーム
- 中央

プロセス・フロー・エディタで使用できるツールの詳細は、10-32 ページの「プロセス・フロー・エディタのメニューとツールバー」を参照してください。

アクティビティについて

プロセス・フローの基本設計要素には、アクティビティと推移が含まれます。アクティビティは、プロセス・フローの作業単位を表します。Warehouse Builder でプロセス・フローを設計するときは、プロセス・フロー・エディタのツールボックスからアクティビティを選択して、キャンバスにドラッグします。

各アクティビティの詳細は、10-16 ページの「プロセス・フローでのアクティビティの使用」を参照してください。

アクティビティの追加

プロセス・フローにアクティビティを追加する手順は次のとおりです。

1. ツールボックスからアクティビティを選択し、キャンバスにドラッグします。

   変換、マッピング、サブプロセスを追加する場合、「アクティビティの作成」ダイアログからオブジェクトを選択します。

   キャンバスにアクティビティが表示され、プロセス・フロー・エディタの左下隅にある「アクティビティ・ビュー」パネルにアクティビティの一覧が表示されます。

2. 「アクティビティ・ビュー」パネルで、アクティビティを選択します。

   アクティビティの名前を変更するか、デフォルト名をそのまま使用します。

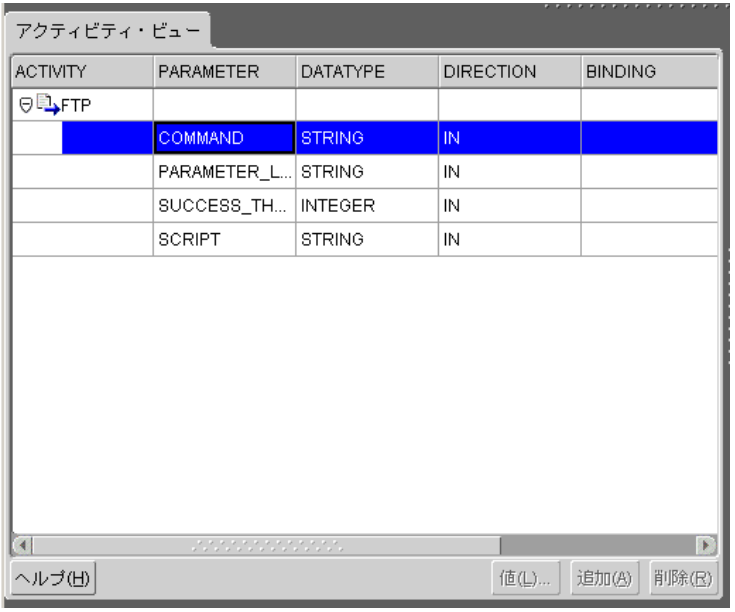
   アクティビティの説明（オプション）を追加できます。キャンバスでアクティビティを選択すると、プロセス・フロー・エディタの最下部にあるインジケータ・バーにその説明が表示されます。

   複雑なプロセス・フローを簡単にナビゲートおよび変更するには、各アクティビティの説明を追加します。

3. 「アクティビティ・ビュー」パネルで、アクティビティ名の左側にある「+」アイコンをクリックして、アクティビティを開きます。

   アクティビティのパラメータが表示されます。これらのプロパティは、アクティビティのタイプによって異なります。たとえば、図 10-5 は FTP アクティビティのプロパティを示しています。

図 10-5 FTP アクティビティのパラメータ



- **PARAMETER:** 図 10-5 にある COMMAND や PARAMETER などのアクティビティ・パラメータの名前。特定のパラメータの詳細は、10-16 ページの「プロセス・フローでのアクティビティの使用」で、アクティビティ名を検索してください。
- **DATATYPE:** 開始アクティビティ以外のすべてのアクティビティでは、パラメータのデータ型は読取り専用になります。すべてのデフォルト・パラメータに適切なデータ型が割り当てられます。
- **DIRECTION:** パラメータの方向は読取り専用になります。方向「IN」は、パラメータがアクティビティの入力パラメータであることを示します。
- **BINDING:** バインディングを使用して、プロセス・フローの外部からパラメータを渡します。パラメータを「BINDING」に割り当てた場合、「VALUE」に割り当てたテキストが上書きされます。バインディングの詳細は、10-12 ページの「パラメータのアクティビティへのバインディングと引渡し」を参照してください。
- **VALUE:** デフォルト値をパラメータに割り当てます。デフォルトを上書きするには、新しい値を入力します。
- **DESCRIPTION:** 各プロパティの説明を入力できます（オプション）。

パラメータのアクティビティへのバインディングと引渡し

外部の値をプロセス・フローに渡すことができます。実行時に様々なアクティビティに渡した値を動的に変更するには、異なるパラメータ値でプロセス・フローを実行します。

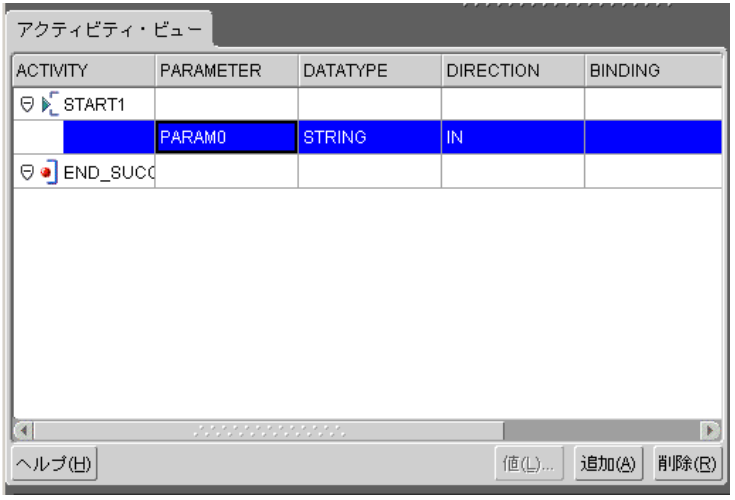
開始アクティビティに定義したパラメータに基づいて、アクティビティ・パラメータに値を入力できます。これは、バインディングと呼ばれています。アクティビティをバインドするには、アクティビティ・パラメータが、開始アクティビティの入力パラメータのデータ型と方向に一致する必要があります。プロセス・フローに日付を渡す場合は、日付が「mm-dd-yyyy」形式であることを確認します。

値をアクティビティに渡す手順は次のとおりです。

1. プロセス・フローに入力パラメータを追加するには、開始アクティビティを選択し、「アクティビティ・ビュー」パネルの最下部にある「追加」を選択します。

   図 10-6 に示すように、開始アクティビティの下にパラメータが追加されます。

図 10-6 パラメータの開始アクティビティへの追加



2. パラメータのプロパティを設定します。

   パラメータ名とデータ型を必要に応じて変更します。方向とバインディングは変更できません。方向は「IN」です。これは、パラメータとして入力パラメータのみを使用できることを示します。値には、パラメータ値を入力します。

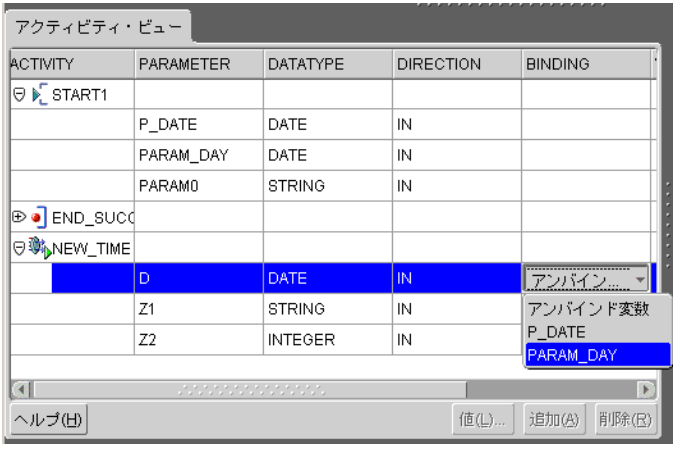
3. パラメータを受け取るアクティビティを選択します。

   この手順では、前の手順で使用した開始アクティビティ以外のアクティビティを選択します。

4. パラメータを受け取るアクティビティを開き、パラメータを選択して、「BINDING」を選択します。

   プロセス・フロー・エディタに、選択したアクティビティ・パラメータのデータ型と方向に一致する開始アクティビティ入力パラメータの一覧が表示されます。図 10-7 は、START1 で定義され、変換アクティビティ NEW_TIME のパラメータ Z2 の入力として選択された PARM_DAY を示しています。

図 10-7 アクティビティ・パラメータのバインディング

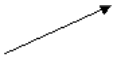
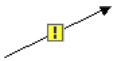


推移について

推移を使用して、アクティビティがプロセス・フローで発生する順序と条件を示します。推移を使用すると、直前のアクティビティの完了状態を基に、アクティビティを実行できます。

表 10-1 は、推移に指定できる条件タイプの一覧を表示し、キャンバスの関連アイコンを示しています。

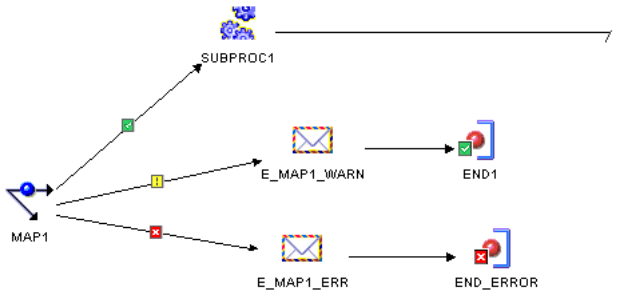
表 10-1 推移の条件タイプ

アイコン	推移	説明
	条件なし	推移をキャンバスに追加する場合、デフォルトでは推移に適用される条件がありません。プロセス・フロー・エディタは、直前のアクティビティが完了するとフローを続行します。プロセス・フローは、前のアクティビティの終了状態に関係なく続行します。
	成功	プロセス・フロー・エディタは、直前のアクティビティが正常に終了した場合にのみフローを続行します。
	警告	プロセス・フロー・エディタは、直前のアクティビティが警告で終了した場合にのみフローを続行します。
	エラー	プロセス・フロー・エディタは、直前のアクティビティがエラーで終了した場合にのみフローを続行します。

アクティビティが 1 つの発信アクティビティのみである場合、表 10-1 に示されている 4 つの条件のいずれかを指定できます。複数の送信の推移をアクティビティに追加する場合は、条件が矛盾していないことを確認してください。プロセス・フロー・ロジックによって、複数の送信の推移が true であると評価できる場合、矛盾が発生します。単一のアクティビティの場合、条件なしの送信の推移を指定するか、または SUCCESS、WARNING、ERROR の推移条件のいずれかを指定できます。推移条件は、SUCCESS、WARNING、ERROR、UNCONDITIONAL の順序で評価されます。

図 10-8 は、MAP1 の 3 つの可能な完了状態に基づいて、異なるアクティビティを起動させるプロセス・フローの一部を示しています。これらの条件は一度に 1 つのみ満たされるので、矛盾は発生しません。条件なし推移または別の条件付き推移を追加すると、2 つの推移条件が true になり、プロセス・フローは無効になります。

図 10-8 送信の推移の条件



アクティビティの接続

推移を使用して依存関係を作成する手順は次のとおりです。

1. プロセス・フロー・エディタで、「編集」および「推移を作成」を選択するか、メニュー・バーから「推移の作成」アイコンを選択します。

   小さな水平の矢印としてカーソルが表示されます。マウス・ボタンを使用してアクティビティを接続できます。

2. 接続するアクティビティにカーソルを置き、左のマウス・ボタンを押します。カーソルを次のアクティビティに置くまで、左のマウス・ボタンを押したままにします。

   2 つのアクティビティ間に矢印が表示され、「遷移ビュー」パネルにエントリが追加されます。

3. 「遷移ビュー」パネルで、推移を選択します。「遷移ビュー」パネルでは、各推移のプロパティが次のように表示されます。

   FROM: このプロパティは、読取り専用で、接続の最初のアクティビティを示します。

   TO: このプロパティは、読取り専用で、接続の 2 番目のアクティビティを示します。

   CONDITION: 成功、エラー、警告など、推移の条件を指定できます。条件を選択すると、関連アイコンがキャンバスにある推移線に挿入されます。

   DESCRIPTION: 推移の説明（オプション）を入力できます。キャンバスにある推移を選択すると、インジケータ・バーにこの説明が表示されます。

プロセス・フローでのアクティビティの使用

この項では、すべてのアクティビティの概要を示す表と、各アクティビティの使用方法を詳細に説明します。

表 10-2 は、各アクティビティを一覧表示し、プロセス・フロー・エディタの関連アイコンを示します。また、各アクティビティに許可されている受信および送信の推移の数も一覧表示します。

表 10-2 プロセス・フローのアクティビティ

アイコン	アクティビティ	概要	受信の推移	発信の推移
	AND	AND アクティビティを使用して、別のアクティビティを起動する前に、すべての受信アクティビティの完了を指定します。	複数が使用可	条件なしの推移 1 つ
	電子メール	電子メール・アクティビティを使用して、電子メールを送信します。たとえば、プロセス・フローでアクティビティのステータスについての通知を送信します。	1 つ以上が使用可	条件なしの推移 1 つ、または条件付きの推移 3 つまで
	終了	デフォルトでは、END_SUCCESS が各プロセス・フローに追加されます。成功、エラー、警告という最大 3 つの終了アクティビティをフローに含めることができます。	1 つ以上が使用可	使用不可
	外部プロセス	外部プロセス・アクティビティを使用して、Warehouse Builder で定義されていないアクティビティを表示し、それをプロセス・フローに組み込みます。	1 つ以上が使用可	条件なしの推移 1 つ、または条件付きの推移 3 つまで
	ファイルが存在	ファイルが存在アクティビティを使用して、ファイルが指定したドライブまたはディレクトリにあるかどうかを確認します。ファイルが存在アクティビティは、プロセス・フローを再開する前にファイルを受信します。	1 つ以上が使用可	条件なしの推移 1 つ、または条件付きの推移 3 つまで
	FORK	Fork アクティビティを使用して、アクティビティの完了後に複数のアクティビティを起動します。	1 つ以上が使用可	複数の条件なし推移

表 10-2 プロセス・フローのアクティビティ（続き）

アイコン	アクティビティ	概要	受信の推移	発信の推移
	FTP	FTP アクティビティを使用して、プロセス・フローの中でファイル転送プロトコル・コマンドを起動します。たとえば、FTP を使用して、マッピングが実行されるマシンにデータ・ファイルを移動します。	1 つ以上が使用可	条件なしの推移 1 つ、または条件付きの推移 3 つまで
	マッピング	マッピング・アクティビティを使用して、プロセス・フローに既存のマッピングを追加します。	1 つ以上が使用可	条件なしの推移 1 つ、または条件付きの推移 3 つまで
	OR	OR アクティビティを使用して、指定した複数のアクティビティの完了後に、アクティビティを起動します。	複数が使用可	条件なしの推移 1 つ
	開始	デフォルトでは、開始アクティビティが各プロセス・フローに追加されます。プロセス・フローでは、開始アクティビティでパラメータを渡すことができます。	使用不可	条件なしの推移 1 つ
	サブプロセス	サブプロセス・アクティビティを使用して、プロセス・フロー内に既存のプロセス・フローを埋め込むことができます。	1 つ以上が使用可	条件なしの推移 1 つ、または条件付きの推移 3 つまで
	トランスフォーム	Transform アクティビティを使用して、プロセス・フローに既存の変換を追加します。	1 つ以上が使用可	条件なしの推移 1 つ、または条件付きの推移 3 つまで

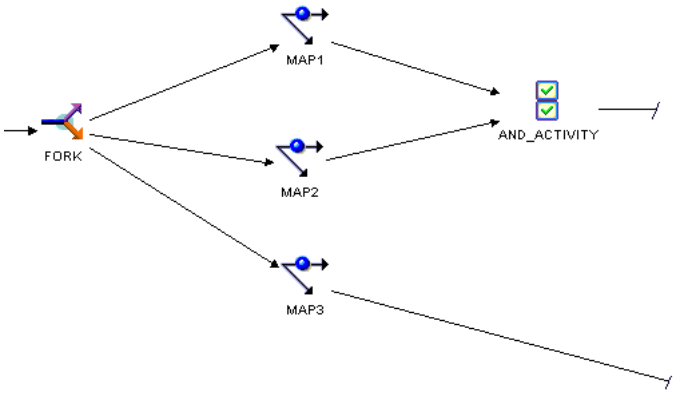
AND

AND アクティビティを使用して、プロセス・フローを再開する前に、複数のアクティビティの完了を指定します。AND アクティビティには、複数の受信の推移を含めることができますが、条件なしの送信の推移は 1 つしか含めることができません。

AND アクティビティでプロセス・フローを正確に設計するには、FORK アクティビティを AND アクティビティの前に置く必要があります。また、AND で処理する推移の数は、その前に置かれた FORK からの送信の推移の数より少ないか、それと同じである必要があります。FORK は、複数の条件なし推移を割り当てることができる唯一のアクティビティで、AND アクティビティに必要な複数のアクティビティを完了します。

図 10-9 は、プロセス・フローにおける AND アクティビティと FORK アクティビティを示しています。この例の AND_ACTIVITY は、MAP1 および MAP2 の完了結果を基に、その後のアクティビティを起動します。FORK アクティビティにある 3 つの送信の推移のうち、AND_ACTIVITY に割り当てられたのは 2 つであるため、プロセス・フローは有効です。また、プロセス・フローは、MAP3 に関連付けられている推移とアクティビティが削除された場合も有効です。

図 10-9 プロセス・フローの AND アクティビティ



電子メール

プロセス・フローでのアクティビティの完了後に、電子メール通知を送信するよう指定できます。たとえば、マッピングなどのアクティビティでエラーまたは警告が発生した場合、そのことを管理者に通知するのに役立ちます。

表 10-3 は、電子メール・アクティビティに設定するパラメータを一覧表示しています。

表 10-3 電子メール・アクティビティのパラメータ

パラメータ	説明
SMTP SERVER	送信メール・サーバーの名前。デフォルト値は、Local Host です。
ポート	送信メール・サーバーのポート番号。デフォルト値は、25 です。
FROM_ ADDRESS	プロセス・フロー通知の送信元の電子メール・アドレス。
REPLY_TO_ ADDRESS	受信者の返信先の電子メール・アドレスまたはメーリング・リスト。
TO_ADDRESS	プロセス・フロー通知を受信する電子メール・アドレスまたはメーリング・リスト。カンマまたはセミコロンを使用して、複数の電子メール・アドレスを区切ります。
CC_ADDRESS	プロセス・フロー通知のコピーを受信する電子メール・アドレスまたはメーリング・リスト。カンマまたはセミコロンを使用して、複数の電子メール・アドレスを区切ります。
BCC_ADDRESS	プロセス・フロー通知のブラインド・コピーを受信する電子メール・アドレスまたはメーリング・リスト。カンマまたはセミコロンを使用して、複数の電子メール・アドレスを区切ります。
IMPORTANCE	通知の重要度レベル。たとえば、High、Medium、Low などのテキストを入力できます。
SUBJECT	電子メールの件名行に表示されるテキスト。
MESSAGE_ BODY	電子メールの本文に表示されるテキスト。テキストの入力または貼付けを行うには、「アクティビティ・ビュー」パネルの最下部にある「値」を選択します。プロセス・フロー・エディタでは、入力するテキストの文字数が制限されません。

電子メール・アドレスの場合、表示名あり / なしで電子メール・アドレスを入力できます。たとえば、次のように入力できます。

```
jack.emp@oracle.com

Jack Emp<jack.emp@oracle.com>

Jack Emp[jack.emp@oracle.com]

Jack Emp[jack.emp@oracle.com],Jill Emp[jill.emp@oracle.com]

Jack Emp[jack.emp@oracle.com];Jill Emp[jill.emp@oracle.com]
```

終了

プロセス・フローのパスはすべて、終了アクティビティで終了する必要があります。

プロセス・フローを最初に作成するとき、デフォルトでは、終了タイプが「成功」の終了アクティビティが含められます。終了タイプを使用して、パスに含まれるロジックのタイプを示します。マッピングなどのアクティビティには 3 つの可能な結果があるので、プロセス・フロー・エディタには表 10-4 に示されているように、3 つの終了タイプがあります。これらの終了タイプを使用して、プロセス・フローのエラー処理ロジックを設計できます。

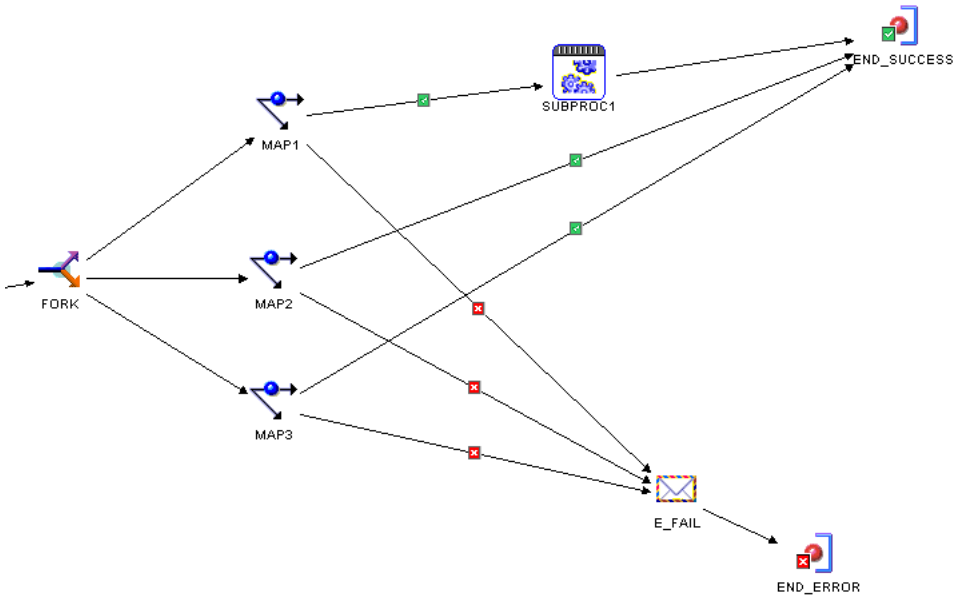
表 10-4 終了アクティビティのタイプ

アイコン	終了タイプ	説明
	成功	パス（複数も可）に、前のアクティビティの正常な完了を条件とするロジックがあることを示します。
	警告	パス（複数も可）に、前のアクティビティが警告を発して完了することを条件とするロジックがあることを示します。
	エラー	パス（複数も可）に、前のアクティビティがエラーを発して完了することを条件とするロジックがあることを示します。

これら 3 つの終了タイプの 1 つ、2 つまたはすべてを含んだプロセス・フローを設計できます。各終了タイプを使用できるのは 1 度のみです。終了タイプを重複して指定することはできません。各終了アクティビティには、単一または複数の受信の推移を含めることができます。

図 10-10 では、END_SUCCESS に 3 つの受信の推移があり、それぞれが前のアクティビティの正常な終了を条件とします。END_ERROR には、前のマッピング・アクティビティがエラーで終了した場合に実行される、電子メール・アクティビティからの受信の推移が 1 つあります。

図 10-10 プロセス・フローの終了アクティビティ



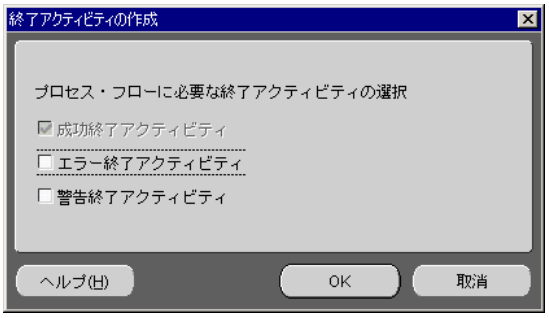
デフォルトでは、すべてのプロセス・フローに END_SUCCESS があります。終了アクティビティを別のタイプに変更することはできませんが、他のタイプのアクティビティを追加することはできます。

プロセス・フローに終了アクティビティを追加する手順は次のとおりです。

1. プロセス・フロー・エディタのツールボックスで、「終了」アイコンをキャンバスにドラッグ・アンド・ドロップします。

   図 10-11 に示すように、終了アクティビティを追加するダイアログが表示されます。

図 10-11 終了アクティビティの追加ダイアログ



2. 使用できる終了タイプのリストから選択します。

   プロセス・フローにある既存の終了タイプを選択することはできません。

3. 「OK」をクリックします。

   終了アクティビティ（複数も可）をキャンバスに追加します。

外部プロセス

外部プロセス・アクティビティを使用すると、Warehouse Builder で定義されていないアクティビティをプロセス・フローに組み込むことができます。

外部プロセス・アクティビティを起動するよう、1 つ以上の受信の推移を指定できます。送信の推移では、条件なしの推移を 1 つ指定するか、3 つの条件付き推移の中から 1 つ指定できます。

条件付きの送信の推移を指定する場合は、アクティビティの戻り値に基づいてステータスを設定するように指定できます。「ステータスとしてリターンを使用」の詳細は、11-40 ページの「プロセス・フローの構成」を参照してください。

コード生成時には、外部プロセスのジョブ制御コード（TCL スクリプト）が生成されます。このコードは、Warehouse Builder ワークフローの一部として配布できます。

表 10-5 は、FTP アクティビティに設定するパラメータを一覧表示しています。

表 10-5 外部プロセス・アクティビティのパラメータ

パラメータ	説明
COMMAND	定義した外部プロセスを実行するコマンド。`c:¥winnt¥system32¥cmd.exe` など、パスとファイル名を入力します。
PARAMETER_LIST	外部プロセスに渡されるパラメータのリスト。`?/c?c:¥¥temp¥¥run.bat` など、パスとファイル名を入力します。 プロセス・フロー・エディタは、最初に入力した文字がセパレータであると解釈します。たとえば、次の入力を `/c` および `dir` と解釈します。 `?/c?dir?` エスケープ文字として円記号を使用してください。たとえば、プロセス・フロー・エディタは次の入力を `-l`、`-s` 、`/` と解釈します。 /-l/-s/¥// 表 10-6 の置換変数も入力できます。
SUCCESS_THRESHOLD	完了ステータスを指定します。正常な完了を示す、オペレーティング・システムからの最大の戻り値を入力します。この値よりもオペレーティング・システムから戻された値が大きい場合、コマンドが失敗したことが示されます。デフォルト値は、`0` です。
SCRIPT	スクリプトまたはスクリプトのファイル名を入力できます。ファイル名を入力した場合、パラメータ・リストの ${Task.Input} 変数を使用して、ファイル名を渡します。 テキストの入力または貼付けを行うには、「アクティビティ・ビュー」パネルの最下部にある「値」を選択します。プロセス・フロー・エディタでは、入力するテキストの文字数が制限されません。 スクリプト内の改行は、[*Enter*] キーを押すことに相当します。そのため、スクリプトの最終行も実行されるよう改行を入れる必要があります。

表 10-6 は、FTP アクティビティで入力可能な置換変数を一覧表示しています。

表 10-6 外部プロセス・アクティビティの置換変数

変数	値
${Working.Host}	作業ロケーションのホスト値。
${Working.User}	作業ロケーションのユーザー値。
${Working.Password}	作業ロケーションのパスワード値。
${Working.Rootpath}	作業ロケーションのルート・パス値。
${Task.Input}	スクリプト・パラメータのファイル名を入力する際に、この変数を使用します。スクリプト・パラメータに入力したファイル名は、${Task.Input} によって渡されます。 パラメータ・リストでは、セパレータとして疑問符を使用し、次のように入力します。 `?"-s:${Task.Input}?`

ファイルが存在

ファイルが存在アクティビティを使用して、次のアクティビティを実行する前に、ファイルの存在を確認します。「アクティビティ・ビュー」パネルで、ファイル・パスの文字列値を入力します。

Workflow Engine のタイムアウト設定は、ファイルが存在アクティビティがファイルを待つ時間を決定します。タイムアウト設定が経過してもファイルがない場合は、ファイルが存在アクティビティがエラーで終了します。

FORK

FORK アクティビティを使用して、アクティビティの完了後に、複数のアクティビティを同時に起動します。

複数の受信の推移を FORK に割り当てることができます。FORK は、同時に実行される複数の条件なしの送信の推移を割り当てることができる唯一のアクティビティです。たとえば、図 10-12 では、MAP1 の完了後に、プロセス・フローで FTP、FDS、EMAIL というアクティビティが同時に実行されます。

図 10-12 同時実行を可能にする FORK アクティビティ

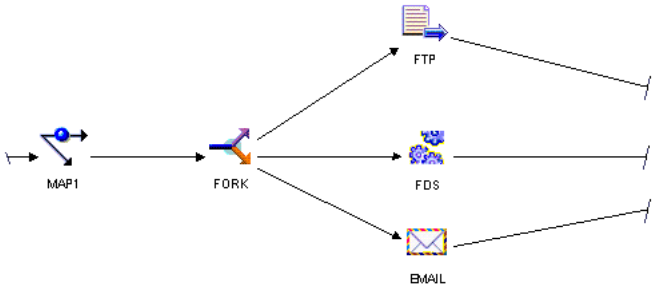
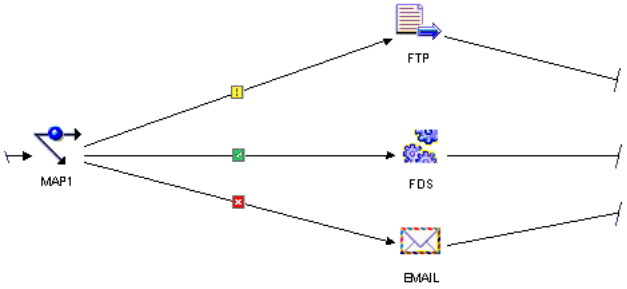


図 10-13 では、同じアクティビティが、FORK アクティビティなしで実行されています。この場合、MAP1 の完了状態を基に実行されるアクティビティは 1 つのみになります。

図 10-13 FORK アクティビティが存在しない際の条件付き実行



FORK で割り当てられる送信の推移と同時アクティビティの数は制限されません。同時実行を設計するときは、プロセス・フローを実行するワークフロー・エンジンまたはサーバーの制限を考慮した上で、FORK を設計してください。

FTP

FTP アクティビティを使用して、あるロケーションから別のロケーションにファイルを転送します。

FTP アクティビティを起動するよう、1 つ以上の受信の推移を指定できます。送信の推移では、条件なしの推移を 1 つ指定するか、3 つの条件付き推移の中から 1 つ指定できます。

条件付き送信の推移を指定する場合は、FTP アクティビティの戻り値に基づいてステータスを設定するように指定できます。たとえば、オペレーティング・システムは、FTP コマンドが正常に完了した場合、値 1 を返します。「構成プロパティ」ウィンドウで、「ステータスとしてリターンを使用」を true に設定した場合、プロセス・フローは正常に完了した推移に従って続行されます。「ステータスとしてリターンを使用」の詳細は、11-40 ページの「プロセス・フローの構成」を参照してください。

表 10-7 は、FTP アクティビティに設定するパラメータを一覧表示しています。

表 10-7 FTP アクティビティのパラメータ

パラメータ	説明
COMMAND	`c:¥WINNT¥System32¥ftp.exe` などのファイル転送プロトコル・コマンド。

PARAMETER_LIST	`/usr/bin/ftp` などの FTP コマンドに渡されるパラメータのリスト。プロセス・フロー・エディタは、最初に入力した文字がセパレータであると解釈します。たとえば、次の入力を `/c` および `dir` という 2 つのパラメータとして解釈します。
	`?/c?dir?`
	エスケープ文字として円記号を使用してください。たとえば、プロセス・フロー・エディタは次の入力を `-l`、`-s` 、`/` という 3 つのパラメータとして解釈します。
	`/-l/-s/¥//`
	表 10-8 の置換変数も入力できます。

SUCCESS_THRESHOLD	FTP コマンドの完了ステータスを指定します。正常な完了を示す、オペレーティング・システムからの最大の戻り値を入力します。この値よりもオペレーティング・システムから戻された値が大きい場合、コマンドが失敗したことが示されます。デフォルト値は、`0` です。
SCRIPT	スクリプトまたはスクリプトのファイル名を入力できます。ファイル名を入力した場合、パラメータ・リストの ${Task.Input} 変数を使用して、ファイル名を渡します。 テキストの入力または貼付けを行うには、「アクティビティ・ビュー」パネルの最下部にある「値」を選択します。プロセス・フロー・エディタでは、入力するテキストの文字数が制限されません。 スクリプト内の改行は、[*Enter*] キーを押すことに相当します。そのため、スクリプトの最終行も実行されるよう改行を入れる必要があります。

表 10-8 は、FTP アクティビティで入力可能な置換変数を一覧表示しています。作業ロケーションおよびリモート・ロケーションと関連付けられている変数は、プロセス・フロー構成時の設定値によって異なります。詳細は、11-40 ページの「プロセス・フローの構成」を参照してください。

表 10-8 FTP アクティビティの置換変数

変数	値
${Working.Host}	作業ロケーションのホスト値。
${Working.User}	作業ロケーションのユーザー値。
${Working.Password}	作業ロケーションのパスワード値。
${Working.Rootpath}	作業ロケーションのルート・パス値。
${Remote.Host}	リモート・ロケーションのホスト値。
${Remote.User}	リモート・ロケーションのユーザー値。
${Remote.Password}	リモート・ロケーションのパスワード値。
${Remote.RootPath}	リモート・ロケーションのルート・パス値。
${Task.Input}	スクリプト・パラメータのファイル名を入力する際に、この変数を使用します。スクリプト・パラメータに入力したファイル名は、${Task.Input} によって渡されます。 パラメータ・リストでは、セパレータとして疑問符を使用し、次のように入力します。 `?"-s:${Task.Input}?`

マッピング

マッピング・アクティビティを使用して、マッピング・エディタで定義し、構成した既存のマッピングを追加します。

複数の受信の推移をマッピング・アクティビティに割り当てることができます。送信の推移の場合、1 つの条件なし推移、または条件つき推移をそれぞれ 1 つ割り当てます。

プロセス・フローにマッピングを追加する場合、「アクティビティ・ビュー」パネルに構成プロパティを表示できます。プロセス・フロー・エディタのマッピング・アクティビティは、マッピング・エディタのマッピングからプロパティを継承します。プロセス・フロー・エディタで、プロパティのデータ型や方向を変更することはできません。

ただし、プロセス・フローにのみ影響し、マッピング・エディタのマッピング設定を変更しない、新しい値を割り当てることはできます。たとえば、プロセス・フロー・エディタでオペレーティング・モードをセット・ベースから行ベースに変更する場合、プロセス・フローは行ベース・モードで実行します。元のマッピングは、オペレーティング・モードとしてセット・ベース・モードを維持します。基礎となるマッピングのプロパティを変更する方法は、11-2 ページの「マッピング構成のリファレンス」を参照してください。

マッピングに入力パラメータのマッピング演算子が含まれる場合、データ型に従って値を指定します。入力パラメータのマッピング演算子をマッピングに追加した場合、プロセス・フロー・エディタでは PL/SQL 式の受信が予想されます。入力パラメータのマッピングが文字列の場合、文字列を二重引用符で囲みます。

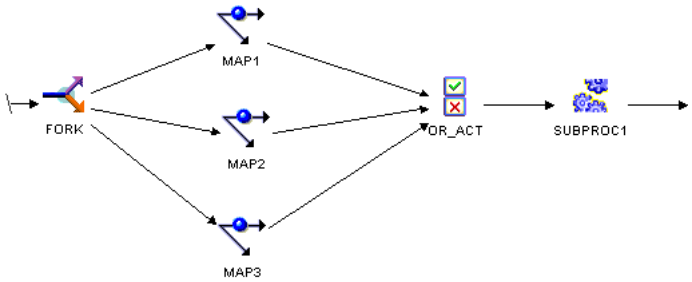
マッピング・エディタのマッピングに対して行った変更でプロセス・フローを更新する場合は、プロセス・フローからマッピング・アクティビティを削除した後、マッピング・アクティビティを再度追加します。

OR

OR アクティビティを使用して、その前に実行された複数のアクティビティの 1 つが完了した後、それに基づいてアクティビティを起動します。複数の受信の推移と 1 つの条件なしの送信の推移を OR アクティビティに割り当てることができます。

プロセス・フローの OR アクティビティは、その後のアクティビティをプロセス・フローの実行するたびに 1 回だけ起動されるようにします。たとえば、図 10-14 に示すように、SUBPROC1 はその前にある 3 つのマッピング・アクティビティのいずれかが完了すると、1 回起動されます。

図 10-14 プロセス・フローの OR アクティビティ



プロセス・フロー・エディタでは、OR アクティビティを省略して、3 つのマッピング・アクティビティの推移をそれぞれ図 10-14 の SUBPROC1 に割り当てることができます。ただし、このロジックは、プロセス・フローの同じ実行で、SUBPROC1 を 3 回起動します。OR アクティビティを使用すると、このような無駄を省くことができます。

開始

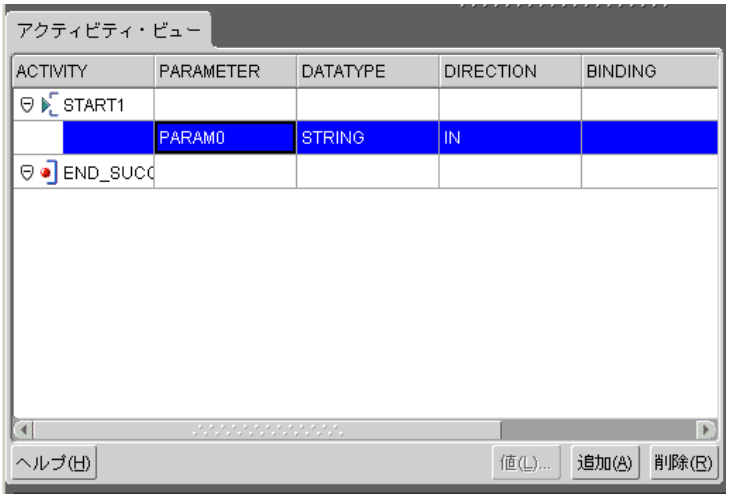
デフォルトでは、各プロセス・フローには開始アクティビティが 1 つあります。完了したプロセス・フローの入力パラメータになる、開始アクティビティの入力パラメータを設定できます。

パラメータを開始アクティビティに追加する手順は次のとおりです。

1. プロセス・フロー・エディタの「アクティビティ・ビュー」パネルで、開始アクティビティを選択します。
2. 「アクティビティ・ビュー」パネルの下部にあり、インジケータ・バーの上部にある「追加」を選択します。

   図 10-15 に示すように、開始アクティビティの下にパラメータが追加されます。

図 10-15 開始アクティビティに追加されたパラメータ



3. パラメータのプロパティを設定します。

   パラメータ名とデータ型を必要に応じて変更します。方向とバインディングは変更できません。方向は「IN」です。これは、パラメータとして入力パラメータのみを使用できることを示します。値には、パラメータ値を入力します。この値は実行時に上書きできます。

4. これで、プロセス・フローにある他のアクティビティへの入力として、パラメータを使用できます。手順は、10-12 ページの「パラメータのアクティビティへのバインディングと引渡し」を参照してください。

サブプロセス

サブプロセス・アクティビティを使用して、以前に作成したプロセス・フローを起動します。あるプロセス・フローから、同じプロセス・フロー・パッケージに含まれる他のプロセス・フローを起動できます。

プロセス・フローにサブプロセスを追加したら、これを使用して他のアクティビティと同様の方法で設計します。複数の受信の推移を割り当てることができます。送信の推移の場合は、条件なしの送信の推移を 1 つ、または条件付き送信の推移を 3 つまで割り当てることができます。

サブプロセス内の END アクティビティは、サブプロセスにのみ割り当てられ、プロセス・フローの終了ポイントとしては機能しません。

サブプロセスと他のアクティビティの重要な相違点として、サブプロセスの内容は表示できるが、その内容を親プロセス・フローでは編集できないことがあげられます。サブプロセスを編集するには、ナビゲーション・ツリーで基礎となるプロセス・フローを開きます。プロセス・フロー・エディタは、プロセス・フローの改名以外の変更内容をすべて、子プロセス・フローから親プロセス・フローへと伝播します。

注意： プロセス・フローの名前を変更する場合は注意してください。別のプロセス・フローによって参照されるプロセス・フローの名前を変更する場合、親プロセス・フローは無効になります。この場合、無効なサブプロセスを削除してから、子プロセス・フローの新しい名前と関連付けられた新しいサブプロセスを追加する必要があります。

プロセス・フローにサブプロセス・アクティビティを追加する手順は次のとおりです。

1. プロセス・フロー・エディタのツールボックスで、「サブプロセス」アイコンをキャンバスにドラッグ・アンド・ドロップします。

   サブプロセスとしてプロセス・フローを選択および追加するためのダイアログが表示されます。

2. プロセス・フロー・モジュールを開き、親プロセス・フローと同じプロセス・フロー・パッケージからプロセス・フローを選択します。

   プロセス・フローが親プロセス・フローのサブプロセス・アクティビティとして表示されます。

3. サブプロセスの内容を表示するには、サブプロセスを右クリックし、「ノードの拡張」を選択します。

   青い縁で囲まれたサブプロセスのグラフが表示されます。

トランスフォーム

トランスフォーム・アクティビティを使用して、Warehouse Builder 変換ライブラリからプロセス・フローに変換を追加する場合、「アクティビティ・ビュー」パネルに変換のパラメータが表示されます。

トランスフォーム・アクティビティを起動するよう、1 つ以上の受信の推移を指定できます。送信の推移では、条件なしの推移を 1 つ指定するか、3 つの条件付き推移の中から 1 つ指定できます。

条件付きの送信の推移を指定する場合は、アクティビティの戻り値に基づいてステータスを設定するように指定できます。「ステータスとしてリターンを使用」の詳細は、11-40 ページの「プロセス・フローの構成」を参照してください。

変換に対して行った変更でプロセス・フローを更新する場合は、プロセス・フローからトランスフォーム・アクティビティを削除した後、トランスフォーム・アクティビティを再度追加します。

プロセス・フロー・エディタのメニューとツールバー

この項には、プロセス・フロー・エディタにある次の設計コンポーネントについての情報が含まれます。

- メニュー・バー
- ツールバー

メニュー・バー

プロセス・フロー・エディタでプロセス・フローを設計するには、次のメニュー・オプションを使用します。

- プロセス・フロー
- 編集
- 表示
- ウィンドウ
- ヘルプ

プロセス・フロー

表 10-9 は、メニュー・バーの「プロセス・フロー」で使用可能なオプションを一覧表示しています。

表 10-9 メニュー・バー : プロセス・フロー

メニュー項目	説明
開く	現在のウェアハウス・モジュールで、別のプロセス・フロー・エディタを起動します。すでに開いているプロセス・フローを開こうとすると、そのプロセス・フローに切り替わります。
検証	現在のプロセス・フローとすべてのアクティビティを検証します。検証が完了すると、Warehouse Builder に「検証の詳細」ダイアログが表示されます。
生成	現在のプロセス・フロー用に生成されるサンプル・コードの表示に使用します。作成されたスクリプトは、「コード・ビュー」タブに表示されます。
プリンタの設定	スケール、余白、複数ページなどのオプションを持つ、印刷オプションの設定に使用します。
印刷プレビュー	プロセス・フローの印刷プレビューを表示します。拡大や印刷の設定プロパティを変更できます。
印刷	プロセス・フローのキャンバスの内容を印刷します。
別名で保存	指定したロケーションに、プロセス・フローの図を保存します。文書または Web ページ上の図を使用して、より魅力的なレポートやドキュメントを作成できます。マッピング図は、JPEG（.jpg）形式または Scalable Vector Graphics（.svg）形式のファイルに保存できます。
プロパティ	プロセス・フローのプロパティ・ウィンドウを起動します。
ウィンドウを閉じる	プロセス・フロー・エディタを終了します。

編集

表 10-10 は、メニュー・バーの「編集」で使用可能なオプションを一覧表示しています。

表 10-10 メニュー・バー : 編集

メニュー項目	説明
モードの選択	モードの選択は、プロセス・フロー・エディタを起動する際のデフォルトのモードです。モードの選択を使用して、キャンバスでアクティビティを選択し、移動します。
推移を作成	推移を作成モードを使用して、キャンバスのアクティビティを接続するか、既存の推移を選択します。
パニング	パニングを使用して、キャンバスのオブジェクトを水平方向および垂直方向に移動します。「編集」メニューからパニング・オプションを選択した場合、手の形としてカーソルが表示されます。キャンバスのオブジェクトを移動するには、左のマウス・ボタンを押しながら、そのオブジェクトを水平方向および垂直方向に動かします。
インタラクティブ・ズーム	インタラクティブ・ズームを使用して、キャンバスにあるオブジェクトのサイズを縮小または拡大します。「編集」メニューからインタラクティブ・ズーム・オプションを選択した場合、中央に上向きおよび下向き矢印が入った拡大鏡としてカーソルが表示されます。キャンバスのオブジェクトを縮小するには、左のマウス・ボタンを押しながら、マウスを上方向に動かします。キャンバスのオブジェクトを拡大するには、左のマウス・ボタンを押しながら、マウスを下方向に動かします。オブジェクトが希望のサイズで表示されたら、左のマウス・ボタンを放します。
名前の変更	プロセス・フローのキャンバスで選択したオブジェクトの名前を編集可能にします。このメニュー・オプションは、アクティビティに対して有効です。
削除	選択したオブジェクトをプロセス・フローのキャンバスから削除します。このメニュー・オプションは、アクティビティと推移に対して有効です。

表示

表 10-11 は、メニュー・バーの「表示」で使用可能なオプションを一覧表示しています。

表 10-11 メニュー・バー : 表示

メニュー項目	説明
ズーム	選択した倍率にキャンバスを拡大します。「表示」メニューからズームを選択した場合、「200%」、「150%」、「100%」、「75%」、「50%」、「25%」の倍率レベルから選択できます。
ウィンドウに適合	キャンバスに表示されるよう、すべてのオブジェクトのサイズを変更します。
自動レイアウト	プロセス・フローのデフォルト・レイアウトに基づいて、キャンバスにあるすべてのオブジェクトのサイズを変更および配置変えします。自動レイアウトを使用して、オブジェクトを左から右へ順番に表示します。開始アクティビティは左側に表示され、終了アクティビティは右側に表示されます。
中央	オブジェクトをキャンバスの中央に移動します。
サブプロセスを拡張	サブプロセスのグラフをキャンバスに表示します。サブプロセスのグラフは、青い縁で囲まれます。
サブプロセスを表示	プロセス・フロー・エディタにサブプロセスを表示します。
親に戻る	「サブプロセスを表示」コマンドを使用した後、「親に戻る」を使用して、親プロセス・フローのプロセス・フロー・エディタをアクティブにします。
検証メッセージの表示	「検証結果」ダイアログを起動します。

ウィンドウ

表 10-12 は、メニュー・バーの「ウィンドウ」で使用可能なオプションを一覧表示しています。

表 10-12 メニュー・バー : ウィンドウ

メニュー項目	説明
バーズ・アイ・ビューを表示 / 非表示	キャンバスでのバーズ・アイ・ビューの表示または非表示に使用します。
アクティビティ・パネルを表示 / 非表示	キャンバスでの「アクティビティ・ビュー」パネルの表示または非表示に使用します。「アクティビティ・ビュー」パネルを表示するように選択した場合、プロセス・フロー・エディタの左下隅に表示されます。
推移パネルを表示 / 非表示	キャンバスでの「遷移ビュー」パネルの表示または非表示に使用します。「遷移ビュー」パネルを表示するように選択した場合、プロセス・フロー・エディタの左下隅に表示されます。
すべて配置	プロセス・フロー・エディタに、起動中の Warehouse Builder の画面をすべて配置します。

ヘルプ

表 10-13 は、メニュー・バーの「ヘルプ」で使用可能なオプションを一覧表示しています。

表 10-13 メニュー・バー : ヘルプ

メニュー項目	説明
目次	「目次」タブを選択した状態で、オンライン・ヘルプ・ビューアを起動します。コンテンツ・ツリーを開き、目次を表示します。
索引	「索引」タブを選択した状態で、オンライン・ヘルプ・ビューアを起動します。
検索	「検索」タブを選択した状態で、オンライン・ヘルプを起動します。
トピック	選択したトピックのオンライン・ヘルプを起動します。
Oracle Warehouse Builder バージョン情報	「Warehouse Builder バージョン情報」ダイアログを起動します。

ツールバー

表 10-14 は、プロセス・フロー・エディタのツールバーのボタンを表示される順番に一覧表示しています。

表 10-14 Process Flow Editor のツールバー

ボタン	説明
ここをクリックするとツール・パレットを切替え	パレット表示のオン / オフを切り替えます。
印刷	エディタの内容を印刷します。
別名で保存	指定したロケーションに、プロセス・フローの図を保存します。文書または Web ページ上の図を使用して、より魅力的なレポートやドキュメントを作成できます。マッピング図は、JPEG（.jpg）形式または Scalable Vector Graphics（.svg）形式のファイルに保存できます。
検証	現在のプロセス・フローを検証します。検証が完了すると、「検証結果」ダイアログが表示されます。
このビューをリポジトリと同期化	プロセス・フローを開いた後のリポジトリへの更新で、プロセス・フローを同期化します。他のユーザーが現在のプロセス・フローに影響する変更を行っている可能性があります。このような変更を確認し、必要な更新を行います。
モードの選択	モードの選択は、プロセス・フロー・エディタを起動する際のデフォルトのモードです。モードの選択を使用して、キャンバスのオブジェクトを選択します。アクティビティを選択すると、そのアクティビティが青い縁に囲まれてエディタに表示されます。アクティビティは、編集、移動または削除することができます。推移を選択した場合、矢印が黒から青に変更され、推移が編集できるようになります。
推移の作成	推移の作成モードを使用して、キャンバスのアクティビティを接続します。
パニング	パニングを使用して、キャンバスのオブジェクトを水平方向および垂直方向に移動します。「編集」メニューからパニング・オプションを選択した場合、手の形としてカーソルが表示されます。キャンバスのオブジェクトを移動するには、左のマウス・ボタンを押しながら、そのオブジェクトを水平方向および垂直方向に動かします。
インタラクティブ・ズーム	インタラクティブ・ズームを使用して、キャンバスにあるオブジェクトのサイズを縮小または拡大します。「編集」メニューからインタラクティブ・ズーム・オプションを選択した場合、中央に上向きおよび下向き矢印が入った拡大鏡としてカーソルが表示されます。キャンバスのオブジェクトを縮小するには、左のマウス・ボタンを押しながら、マウスを上方向に動かします。キャンバスのオブジェクトを拡大するには、左のマウス・ボタンを押しながら、マウスを下方向に動かします。オブジェクトが希望のサイズで表示されたら、左のマウス・ボタンを放します。

表 10-14 Process Flow Editor のツールバー（続き）

ボタン	説明
ズーム・イン	ズーム・イン・オプションは、中央にプラス記号（+）がある拡大鏡として表示されます。アイコンをクリックし、キャンバスのオブジェクトの倍率を上げます。
ズーム・アウト	ズーム・アウト・オプションは、中央にマイナス記号（-）がある拡大鏡として表示されます。アイコンをクリックし、キャンバスのオブジェクトの倍率を下げます。
ウィンドウに適合	キャンバスに表示されるよう、すべてのオブジェクトのサイズを変更します。
自動レイアウト	プロセス・フローのデフォルト・レイアウトに基づいて、キャンバスにあるすべてのオブジェクトのサイズを変更および配置変えします。自動レイアウトを使用して、オブジェクトを左から右へ順番に表示します。開始アクティビティは左側に表示され、終了アクティビティは右側に表示されます。
バーズ・アイ・ビュー	キャンバスにバーズ・アイ・ビューを表示します。
サブプロセスを拡張	サブプロセスのグラフをキャンバスに表示します。サブプロセスのグラフは、青い縁で囲まれます。
子プロセスの表示	サブプロセスのプロセス・フロー・エディタを起動します。
親に戻る	「サブプロセスを表示」コマンドを使用した後、「親に戻る」を使用して、親プロセス・フローのプロセス・フロー・エディタをアクティブにします。
プロパティ	「プロセス・フローのプロパティ」ダイアログを起動します。
ヘルプ	オンライン・ヘルプ・セットを起動します。

11

ETL オブジェクトの構成

設計フェーズでは、Warehouse Builder の設計オブジェクトを使用してターゲット・システムの論理モデルを定義しました。この章では、物理プロパティをマッピングとプロセス・フローに割り当てる際の、リファレンス情報と推奨事項について説明します。

この章では、次のトピックについて説明します。

マッピング構成のリファレンス

マッピングを正確に構成すると、ETL パフォーマンスを向上させることができます。この項は、パフォーマンスを向上させるためのデータのロード方法とコードの最適化方法を指定する、構成パラメータを設定する際にリファレンスとして使用します。

この項では、次のトピックについて説明します。

マッピングを構成する手順

マッピングの物理プロパティを構成する手順は次のとおりです。

1. ナビゲーション・ツリーからマッピングを選択します。メニュー・バーで「オブジェクト」→「構成」を選択します。

   または、構成を行うマッピングを右クリックして、ポップアップ・メニューから「構成」を選択します。

   図 11-1 に示すように、マッピングの「構成プロパティ」ダイアログが表示されます。

図 11-1 表のマッピング用の「構成プロパティ」ダイアログ

2. 「配布可能」を true に設定すると、配布可能にマークされたマッピング・エンティティのスクリプト・セットを生成できます。

   デフォルト設定は true です。マッピングの「配布可能」を false に設定すると、そのマッピングに対するスクリプトが生成されません。

3. 「ステップ・タイプ」を、選択したマッピングで生成するコードのタイプに設定します。

   選択できるオプションは、マッピングの設計と使用する演算子の種類によって異なります。Warehouse Builder では、マッピングの種類に応じて、PL/SQL、SQL*Loader、ABAP（SAP ソース・マッピング用）などの正しい値が設定されます。

4. **「ランタイム・パラメータ・リファレンス」**を開き、配布するマッピングを構成します。

   各ランタイム・パラメータの詳細は、11-4 ページの「ランタイム・パラメータ・リファレンス」を参照してください。

5. **「コード生成オプションのリファレンス」**を開き、マッピング用に生成されたコードを最適化するパフォーマンス・オプションを有効にします。

   各オプションの詳細は、11-7 ページの「コード生成オプションのリファレンス」を参照してください。

6. **「ソースとターゲットのリファレンス」**を開き、マッピングにおける演算子の物理プロパティを設定します。

   「構成プロパティ」ダイアログの「ソースとターゲット」の下に、マッピングのソース演算子およびターゲット演算子がすべて表示されます。各演算子には、編集可能なプロパティ・セットが格納されています。マッピングのソースとターゲットの構成の詳細は、11-9 ページの「ソースとターゲットのリファレンス」を参照してください。

7. マッピングにソースまたはターゲットとしてフラット・ファイルが含まれる場合は、追加のパラメータを構成します。フラット・ファイル・ソースとフラット・ファイル・ターゲットを含むマッピングの構成の詳細は、11-13 ページの「フラット・ファイル演算子の構成」を参照してください。

ランタイム・パラメータ・リファレンス

マッピングのランタイム・パラメータを構成する場合、マッピングのデフォルトの動作を設定します。デプロイメント・マネージャ、プロセス・フロー・エディタ、Oracle Enterprise Manager のいずれかでマッピングを実行する場合、これらのパラメータを上書きすることができます。

ランタイム・パラメータには、次のものがあります。

- デフォルト・オペレーティング・モード
- バルク・サイズ
- デフォルト監査レベル
- エラーの最大数
- コミット頻度
- デフォルト・パージ・グループ
- 表分析のサンプリング率

デフォルト・オペレーティング・モード

PL/SQL 実装でのマッピングの場合、デフォルト・オペレーティング・モードを選択します。選択したオペレーティング・モードは、マッピングのパフォーマンスに大きく影響します。オペレーティング・モードのパフォーマンスへの影響の詳細は、11-19 ページの「デフォルト・オペレーティング・モードの選択」を参照してください。次のオペレーティング・モードのうち 1 つを選択できます。

- **セット・ベース :** Warehouse Builder により、全データを挿入し、そのデータに対するすべての操作を実行する単一の SQL 文が生成されます。これによって、データ操作言語（DML）操作が高速化します。セット・ベース・モードでは、最適なパフォーマンスが得られますが、監査詳細は最低限になります。
- **行ベース :** Warehouse Builder により、データを行ごとに処理する文が生成されます。SQL カーソルには、SELECT 文が使用されます。それ以降のすべての文は、PL/SQL になります。データは行ごとに処理されるので、行ベースのオペレーティング・モードのパフォーマンスは最低ですが、監査詳細は最大になります。
- **行ベース（ターゲットのみ）:** Warehouse Builder により、SELECT 文のカーソルが生成され、そのカーソルにできるだけ多くの操作が含められます。各ターゲットでは、PL/SQL 挿入文が生成され、ターゲットに行が個別に挿入されます。
- **セット・ベースから行ベースへのフェイルオーバー :** Warehouse Builder により、セット・ベース・モードでマッピングが実行されます。エラーが発生した場合、実行は失敗し、行ベース・モードでもう一度マッピングが開始されます。
- **セット・ベースから行ベースへのフェイルオーバー（ターゲットのみ）:** マッピングが最初にセット・ベース・モードで実行されます。なんらかのエラーが発生した場合、その実行が行ベース（ターゲットのみ）モードにフェイルオーバーされます。

バルク・サイズ

「バルク・サイズ」を使用して、PL/SQL バルク処理の各バルクにある行数を指定します。バルク処理コードが true に設定されている場合にのみ、バルク・サイズ・パラメータを使用します。詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

デフォルト監査レベル

「デフォルト監査レベル」を使用して、パッケージの実行時に使用される監査レベルを示します。監査レベルによって、パッケージの実行時にランタイム・スキーマで取得される監査情報の量が指定されます。監査レベルの設定は次のとおりです。

- **なし :** ランタイムでは監査情報は記録されません。
- **統計 :** ランタイムでは統計的な監査情報が記録されます。
- **エラーの詳細 :** ランタイムではエラー情報と統計的な監査情報が記録されます。
- **完了 :** ランタイムではすべての監査情報が記録されます。監査レベルが「完了」に設定されたマッピングを実行すると、多量の診断データが生成され、割り当てられた表領域がすぐに満杯になってしまいます。

エラーの最大数

「エラーの最大数」を使用して、パッケージの実行時に許可されるエラーの最大数を示します。エラー数がエラーの最大数の値を超えると、パッケージの実行が終了します。

コミット頻度

コミット頻度は、バルク・モード以外のマッピングにのみ適用されます。バルク・モードのマッピングは、バルク・サイズに従ってコミットします。

デフォルト・オペレーティング・モードを行ベースに、バルク処理コードを false に設定する場合、パッケージの実行時に「コミット頻度」パラメータが使用されます。このパラメータで指定した行数が処理されると、データがデータベースにコミットされます。

バルク処理コードを true に設定する場合、コミット頻度をバルク・サイズと同じ値にします。2 つの値が異なる場合、バルク・サイズはコミット頻度を上書きし、各バルク・サイズが暗黙的にコミットされます。

デフォルト・パージ・グループ

デフォルト・パージ・グループは、パッケージの実行時に使用されます。ランタイム・スキーマの各監査レコードは指定されたパージ・グループに割り当てられます。

表分析のサンプリング率

表分析文（Analize 文）を true に設定する場合、ターゲット表で統計を収集するときに見積もりが取られます。データがターゲット表にロードされると、コストベースの最適化に使用される統計が各ターゲット表に収集されます。この分析に使用される各ターゲット表での行の割合を、このパラメータで設定できます。

コード生成オプションのリファレンス

コード生成オプションには、次のものがあります。

- バルク処理コード
- 行ベース・モード
- パラレル行コード
- 表分析文
- 最適化コード
- 相関コミット

バルク処理コード

この構成パラメータが true に設定されている場合、PL/SQL バルク処理コードが生成されます。PL/SQL バルク処理では、行ごとの実行のかわりに、複数行の収集、処理、書込みをバルクで行うことによって、行ベースの ETL パフォーマンスが向上します。各バルクのサイズは、構成パラメータであるバルク・サイズによって決定されます。パフォーマンスの観点で順序付けると、セット・ベース・モード、バルク処理、行ベース・モードの順になります。詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

行ベース・モード

行ベース・モードは、オペレーティング・モードのオプションです。行ベース・モードでは、ランタイム監査の量を最大限にすることができます。行ベース・モードで実行されるマッピングは、ソースからのデータをレコードごとに処理します。

パラレル行コード

このパラメータが true に設定されている場合、Oracle Database でサポートされているテーブル・ファンクションを使用して、コードが生成されます。この設定によって、行ベースおよび行ベース（ターゲットのみ）のオペレーティング・モードの Oracle Server で、テーブル・ファンクションによるパラレル実行が可能になり、パフォーマンスが向上します。

マッピングで PL/SQL コードが生成され、それを Oracle Database サーバーが実行する場合、このパラメータを true に設定できます。

マッピングで SQL* Loader コードが生成される場合、パラメータを false（デフォルト）に設定します。このプロパティは、PL/SQL コードを生成するマッピングにのみ使用できます。SQL* Loader コードを生成するマッピングの構成プロパティとしては使用できません。このプロパティは、セット・ベース・モードで使用できませんが、セット・ベースから行ベースへのフェイルオーバー・モードおよびセット・ベースから行ベースへのフェイルオーバー・モードには使用できます。このプロパティは、セット・ベース・モードが失敗し、コードが行ベース・モードまたは行ベース・ターゲット・モードで実行される場合も使用できます。このプロパティは、出力パラメータのマッピング、フラット・ファイル演算子または順序演算子をソースとして含むマッピングには使用できません。

次のシナリオでは、テーブル・ファンクションでパラレル処理を実行できません。

1. 入力カーソルがリモート・スキーマ内のオブジェクト（表、ビューなど）を参照する場合。たとえば、リモート・データベース・スキーマにある表にバインドされている表演算子を含むマッピングは、パラレルで実行できません。
2. 表ファンクションに渡された入力カーソルが、Oracle データベース以外にあるオブジェクトを参照する場合。オブジェクトが Oracle ゲートウェイを使用してアクセスされる際には、Oracle Database サーバーのパラレル・エンジンがパラレル処理を実行できません。たとえば、マッピングに DB2 データベース表にバインドされている表演算子が含まれる場合、マッピングはパラレルで実行されません。
3. パラレル DML 演算は、トリガーのある表では実行できません。たとえば、表、およびマップ前とマップ後のトリガー（表を操作する）を含むマッピングでは実行できません。
4. 更新および削除は、パーティション化された表でのみパラレル処理が可能です。更新および削除のパラレル処理は、パーティション内またはパーティション化されていない表では実行できません。たとえば、マッピングに削除または更新をロードする表演算子が含まれる場合、ターゲット表はパーティション化する必要があります。

表分析文

このパラメータを true に設定すると、結果のターゲット表が元のサイズの 2 倍または半分の場合、ターゲットのロード後、ターゲット表を分析するコード（Analyze 文）が生成されます。

最適化コード

このパラメータを true に設定し、スプリッタ演算子を含むマッピングのパフォーマンスを向上させ、複数のターゲット表に挿入します。このパラメータが true に設定され、マッピングが Oracle Database によって実行される場合、ソース・データの同じセットを基にしている複数の表にデータを挿入する、単一の SQL 文（マルチテーブル・インサート）が生成されます。

このパラメータが true に設定されていて、Oracle のターゲット・モジュール・データベースが Oracle Database の場合にのみ、マルチテーブル・インサートが実行されます。マルチテーブル・インサートは、スプリッタ演算子を含むセット・ベース・モードのマッピングでのみ実行されます。また、スプリッタ演算子とターゲットの間に、集計演算子や結合子演算子などのアクティブな演算子がないことも条件です。また、マルチテーブル・インサートは、表に対してのみ使用でき、ビュー、マテリアライズド・ビュー、ディメンション、キューブなどでは使用できません。各ターゲット表の列は、999 個未満である必要があります。複数ターゲットでのマッピングの作成方法の詳細は、8-73 ページの「例 : Name and Address 演算子を使用したレコードの操作」を参照してください。

行ベース・モードで実行されるマッピングまたは Oracle8*i* サーバーによって実行されるマッピングでは、このパラメータを false に設定します。また、監査が必要な場合は、このパラメータを false に設定する必要があります。true に設定した場合、すべてのターゲットに対して、SELECT と INSERT の合計数が返されます。

相関コミット

これは、複数ターゲットのマッピングに適用されます。true に設定した場合、選択したレコードに基づいてデータをコミットします。false に設定した場合、ターゲット処理（挿入、更新、削除）のターゲットに基づいてデータをコミットします。

マッピング動作は、選択したオペレーティング・モードによって異なります。相関コミットの詳細は、11-22 ページの「単一のソースから複数ターゲットへのデータのコミット」を参照してください。

ソースとターゲットのリファレンス

「構成プロパティ」ダイアログの「ソースとターゲット」の下に、マッピングのソース演算子およびターゲット演算子がすべて表示されます。構成で使用できるプロパティは、マッピングで使用されるソース演算子とターゲット演算子のタイプによって異なります。

表、ビュー、キューブなどのリレーショナルおよびディメンションのソースとターゲットの場合は、各演算子に次のようなプロパティが表示されます。

- バウンド名
- スキーマ
- データベース・リンク
- パーティション交換ロード
- ヒント
- 制約管理
- SQL*Loader パラメータ

バウンド名

コードが生成される場合、バウンド名によってソース演算子とターゲット演算子がそれぞれ特定されます。デフォルトでは、バウンド名は演算子の名前と同じです。

スキーマ

このパラメータは、下位互換性のために保持されています。

前のバージョンの Warehouse Builder では、「スキーマ」フィールドをクリックし、名前を入力することで、マッピングを特定のスキーマにリンクできました。

データベース・リンク

このパラメータは、下位互換性のために保持されています。

前のバージョンの Warehouse Builder では、ドロップダウン・リストから名前を選択することで、データベース・リンクを選択できました。ソース演算子はスキーマおよびデータベース・リンクに対して構成できますが、ターゲットはスキーマに対してのみ構成できます。ソースとターゲットは異なるスキーマにあってもかまいませんが、同じデータベース・インスタンス内にある必要があります。

パーティション交換ロード

このパラメータを true に設定し、パーティション交換ロード（PEL）をターゲット表で使用できるようにします。PEL のマッピングの設計方法の詳細は、11-30 ページの「パーティション交換ロードの使用方法」を参照してください。

ヒント

抽出またはロードのヒントを定義します。通常、アプリケーション開発者はアイデアをデータに反映させます。たとえば、開発者は、問合せの実行速度を上げる表のセット結合順序を把握しています。Warehouse Builder では、このようなアイデアを生成された SQL コードのパッケージに SQL オプティマイザ・ヒントとして組み込むことができます。

「ヒント」ダイアログからヒントを選択する場合、ヒントは「既存のヒント」フィールドに表示されます。「不可テキスト」列に適切な付加テキストを入力します。エディタによってマッピング定義にヒントがそのまま追加されます。このテキストの検証やチェックは行われません。

オプティマイザ・ヒントとその使用方法に関する詳細は、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

制約管理

次の制約管理パラメータを構成します。

- **制約の有効化 :** このプロパティを true に設定すると、データのロード前、ターゲット表にある外部キー制約がそのまま保持されます。このプロパティを false に設定すると、データのロード前に、ターゲット表にある外部キー制約が無効化され、ロード後に再び有効化されます。再有効化時に検出された制約違反は、ランタイム監査エラー表に記録されます。また、指定によっては、例外表に記録されます。

  制約を無効にすると、制約のチェックが行われなくなるので、ロード時間が短くなります。ただし、再有効化時にいずれかの行で例外が発生すると、それらの行に対する制約は未検証の状態のままになります。これらの行はランタイム監査エラー表に ROWID により記録されます。適切な修正を行うには、エラー行を手動で調べる必要があります。

  「制約の有効化」を false に設定する場合は、次の制限を受けます。

  – セット・ベースのオペレーティング・モードで false に設定すると、ロード前にターゲット上の外部キー制約が無効化され、ロード後に再び有効化されます。このプロパティは、ターゲット表を参照する他の表の外部キー制約には影響しません。SQL または PL/SQL パッケージではなく SQL*Loader を使用してロードする場合は、再有効化用の句が .ctl ファイルに追加されます。

  – セット・ベースから行ベースへのフェイルオーバーおよびセット・ベースから行ベースへのフェイルオーバー（ターゲットのみ）のオペレーティング・モードで false に設定すると、ロード前にターゲット上の外部キー制約が無効化され、セット・ベース・モードでロードに成功した場合に再び制約が有効化されます。この設定は、他の表を参照する外部キー制約には影響しません。ロードが行ベースにフェイルオーバーされた場合は、ロードは行ベース・モードで再実行され、すべての制約は有効のまま保持されます。

    注意 : 再有効化時に制約違反が検出されると、ロードはセット・ベースから行ベースにフェイルオーバーされません。

  – 行ベースまたは行ベース（ターゲットのみ）のオペレーティング・モードでは、このプロパティを false に設定しても、すべての外部キー制約は有効のまま保持されます。

  – TRUNCATE/INSERT DML タイプで false に設定すると、ロード前にターゲット表を参照する他の表の外部キー制約が無効化され、デフォルトのオペレーティング・モードに関係なく、ロード後に再び有効化されます。

- **例外表名 :** 再有効化時に外部キー制約に違反しているすべての行は、指定した例外表に記録されます。この表の自動切捨ては、ロードの前にも後にも実行されません。制約違反はランタイム監査エラー表にもロードされます。

  SQL および PL/SQL ロードで、例外表を指定していない場合、無効な行はデフォルトの表領域にある一時表にロードされてから、ランタイム監査エラー表にロードされます。表は、ロードが終わると削除されます。

  SQL*Loader のダイレクト・パス・ロードを使用する場合は、例外表を指定する必要があります。詳細は、SQL*Loader のドキュメントを参照してください。

SQL*Loader パラメータ

次の SQL* パラメータのプロパティを構成します。

- **パーティション名 :** そのロードがパーティション・レベルのロードであることを示します。パーティション・レベルのロードでは、表の 1 つ以上の特定のパーティションまたはサブパーティションをロードできます。全データベース、ユーザーおよびトランスポータブル表領域モードのロードは、パーティション・レベルのロードをサポートしていません。増分のロード（増分、累積および完全）は全データベース・モードでしか実行できないので、パーティション・レベルのロードは増分のロードに指定できません。すべてのモードにおいて、パーティション化されたデータは、パーティションまたはサブパーティションを選択してロードできるように、1 つの書式でロードされます。
- **ソート済索引句 :** データが事前ソートされた索引を識別します。この句はダイレクト・パス・ロードに対してのみ使用できます。ある索引によってソートされたデータは通常、他の索引に対して正しい順序ではないため、SORTED INDEXES 句では索引を 1 つのみ指定します。複数の索引に対して同じ順序で並んだデータの場合は、すべての索引を一度に指定できます。SORTED INDEXES 句にリストするすべての索引は、ダイレクト・パス・ロードの起動前に作成する必要があります。
- **単一行 :** これは、メモリー制限のあるシステム上で APPEND によるダイレクト・パス・ロードを実行する場合や、大規模な表に少数のレコードをロードする場合に使用することを目的としたものです。このオプションでは、各索引エントリを一度に 1 レコードずつ直接索引に挿入します。デフォルトでは、SQL*Loader は、SINGLEROW を使用して表にレコードを追加しません。索引エントリは一時領域に格納され、ロードの最後に元の索引とマージされます。この方法では優れたパフォーマンスを達成し、最適な索引を作成しますが、記憶領域が余分に必要となります。マージの実行中に元の索引、新規の索引および新規エントリ用の領域がすべて同時に記憶領域を占有するためです。SINGLEROW オプションでは、新規のエントリまたは新規の索引に対して記憶領域は不要です。作成された索引は新規にソートされた索引ほど最適ではないこともありますが、作成に使用される領域が小さくなります。追加の UNDO 情報が各索引の挿入に対して生成されるため、時間は長くかかります。使用できる記憶領域が限られている場合は、このオプションを選択してください。また、ロードされるレコード数が表のサイズに比べて小さい場合にも選択してください。1:20 以下の比率の場合、小さいと見なされます。

- **後続 NULL 列 :** 相対的に位置指定された列でレコード内に NULL 列として存在していない列をすべて処理するように、SQL*Loader を設定します。
- **スキップ対象レコード :** SQL*Loader で SKIP コマンドを起動します。SKIP では、ファイルの始まりからのロードされない論理レコード数を指定します。デフォルトではスキップされるレコードはありません。このパラメータにより、なんらかの理由で中断されたロードが続行されます。これは、従来のすべてのロードと単一表のダイレクト・ロード、および各表に同じ数のレコードがロードされる場合は複数表のダイレクト・ロードに対して使用されます。各表にロードされるレコードの数が異なる場合、複数表のダイレクト・ロードに対しては使用されません。
- **データベース・ファイル名 :** インポートするエクスポート・ファイル名を指定します。デフォルトの拡張子は .dmp です。複数のエクスポート・ファイルをエクスポートできるので、インポートするファイル名は複数指定する必要がある場合もあります。インポートされたファイルには読取りアクセスができるようにする必要があります。また、IMP_FULL_DATABASE ロールを持つ必要もあります。

フラット・ファイル演算子の構成

「構成プロパティ」ダイアログには、フラット・ファイルのマッピング演算子の追加設定が含まれます。これは、マッピングでの演算子の使用方法によって異なります。

- **ターゲットとしてのフラット・ファイル演算子 :** Warehouse Builder により、PL/SQL 配布コード・パッケージが生成されます。ターゲットとして使用されるフラット・ファイルのマッピング演算子と関連付けられたパラメータの構成の詳細は、11-14 ページの「ターゲットとしてのフラット・ファイル演算子」を参照してください。
- **ソースとしてのフラット・ファイル演算子 :** Warehouse Builder により、SQL*Loader スクリプトが生成されます。ソースとして使用されるフラット・ファイルのマッピング演算子と関連付けられたパラメータの構成の詳細は、11-16 ページの「ソースとしてのフラット・ファイル演算子」を参照してください。

ターゲットとしてのフラット・ファイル演算子

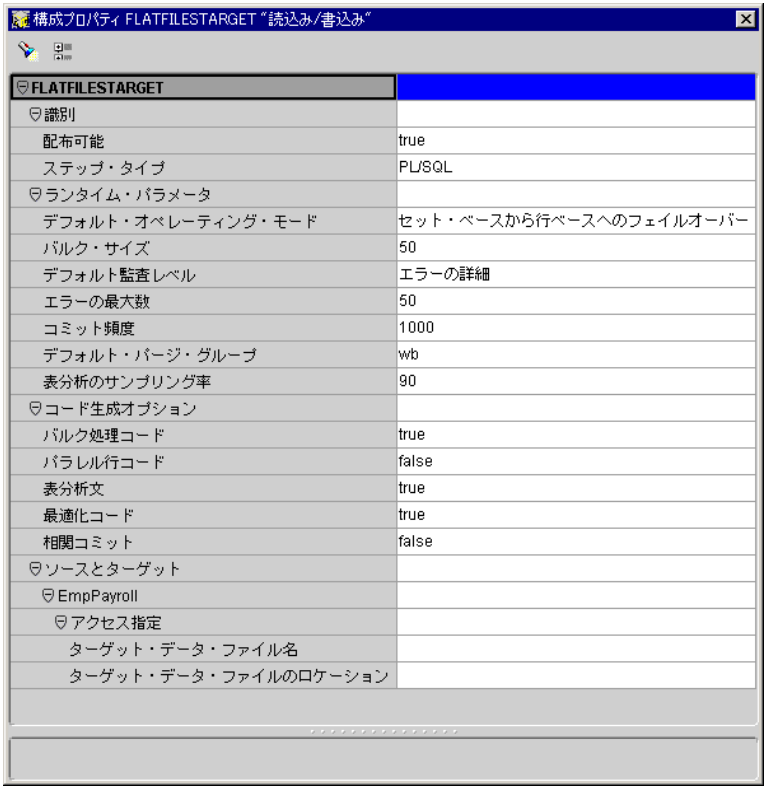
フラット・ファイル・ターゲットのマッピングに対して一意なプロパティを構成する手順は次のとおりです。

1. ナビゲーション・ツリーからマッピングを選択します。メニュー・バーで「オブジェクト」→「構成」を選択します。

   または、構成を行うマッピングを右クリックして、ポップアップ・メニューから「構成」を選択します。

   図 11-2 に示すように、「構成プロパティ」ダイアログが表示されます。

図 11-2「構成プロパティ」ダイアログ（フラット・ファイル・ターゲット）



2. 構成するパラメータを選択し、パラメータ名の右側にある空欄をクリックして値を編集します。

   各パラメータでリストからオプションを選択するか、値を入力することができます。または、「...」をクリックして別のプロパティ・ダイアログを表示します。

3. **「配布可能」**を true に設定すると、配布可能にマークされたマッピング・オブジェクトのスクリプト・セットを生成できます。デフォルト設定は true です。マッピングの「配布可能」を false に設定すると、そのマッピングに対するスクリプトが生成されません。

4. 「ステップ・タイプ」を、選択したマッピングで生成するコードのタイプに設定します。選択できるオプションは、マッピングの設計と使用する演算子の種類によって異なります。マッピングの種類に応じて、PL/SQL、ABAP（SAP ソース・マッピング用）、または SQL*Loader から選択できます。

5. マッピングを配布するロケーションを指定します。

6. **「ランタイム・パラメータ・リファレンス」**で、**「デフォルト・オペレーティング・モード」**を**「行ベース（ターゲットのみ）」**に設定します。このタイプのマッピングは、他のデフォルト・オペレーティング・モードではコードを生成しません。各ランタイム・パラメータの詳細は、11-4 ページの「ランタイム・パラメータ・リファレンス」を参照してください。

7. **「コード生成オプションのリファレンス」**を、11-7 ページの「コード生成オプションのリファレンス」で説明されているように設定します。

8. **「ソースとターゲットのリファレンス」**を、11-9 ページの「ソースとターゲットのリファレンス」で説明されているように設定します。

9. 「アクセス指定」では、「ターゲット・データ・ファイル名」でフラット・ファイル・ターゲットの名前を指定します。「ターゲット・データ・ファイルのロケーション」では、Warehouse Builder ランタイム・プラットフォームがインストールされているマシンにあるターゲット・ファイルを指定します。

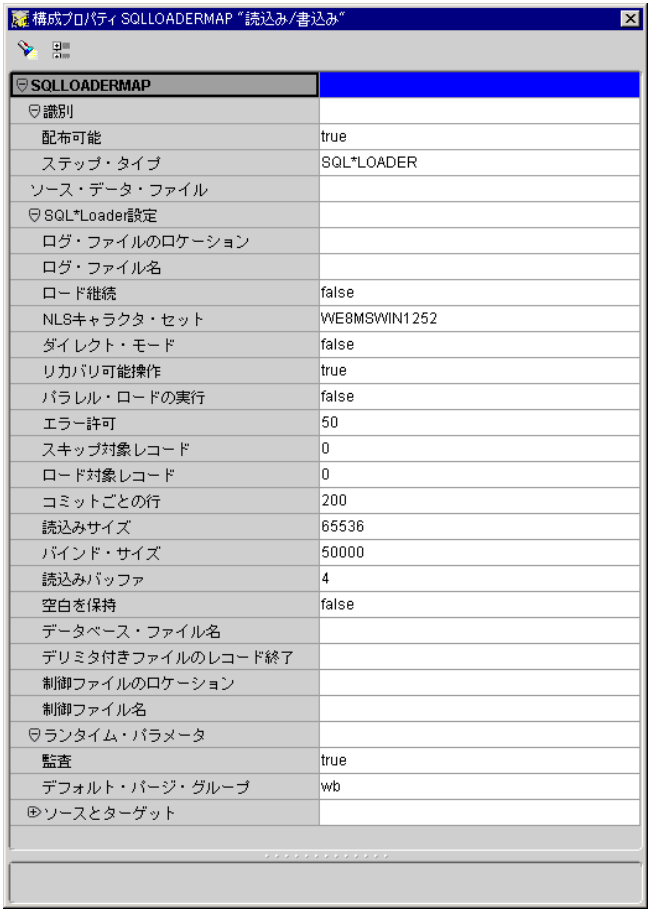
ソースとしてのフラット・ファイル演算子

ソースとしてフラット・ファイル演算子を持つマッピングを構成する手順は次のとおりです。

1. ナビゲーション・ツリーからマッピングを選択します。メニュー・バーで「オブジェクト」→「構成」を選択します。または、構成を行うマッピングを右クリックして、ポップアップ・メニューから「構成」を選択します。

   図 11-3 に示すように、「構成プロパティ」ダイアログが表示されます。

図 11-3「構成プロパティ」ダイアログ（フラット・ファイル・ソース）



2. 構成するパラメータを選択し、パラメータ名の右側にある空欄をクリックして値を編集します。

   各パラメータでリストからオプションを選択するか、値を入力することができます。または、「...」をクリックして別のプロパティ・ダイアログを表示します。

3. 「配布可能」を true に設定し、SQL*Loader スクリプトを生成します。

4. マッピングを実行するロケーションを「実行ロケーション」で指定します。

5. 「ソース・データ・ファイル」では、ロードされるデータ・ファイルの完全な物理ファイル名を入力します。

   ここでは、ファイルの完全パス名を、SQL*Loader スクリプトの配布先であるオペレーティング・システムの構文で指定する必要があります。入力した値は INFILE 句に配置されます。値を単一引用符で囲んで入力した場合、生成されるコードは LOAD になります。CONTINUE_LOAD は、複数表のダイレクト・ロードが中断され、再起動が必要な場合に使用します。これは演算子レベルの SKIP オプションとともに使用されます。

6. 「ログ・ファイルのロケーション」および「ログ・ファイル名」を指定します。

7. 「ロード継続」を設定します。

   SQL*Loader で、データ行または索引エントリの空き領域がなくなった場合、ロードは中断されます。「ロード継続」が true に設定されている場合、中断されたロードを続行しようとします。

8. 「NLS キャラクタ・セット」で、CHARACTERSET 句に配置するキャラクタ・セットを指定します。

9. 「ダイレクト・モード」で、DIRECT オプションの値を true または false として指定します。true は、ダイレクト・パス・ロードが実行されることを示します。false は、従来のロードが実行されることを示します。通常、ダイレクト・モードはより高速に実行されます。

10. 「リカバリ可能操作」で、true を選択すると、ロードがリカバリ可能になります。false は、ロードがリカバリ可能でなく、REDO ログにレコードが記録されないことを示します。このパラメータは、RECOVERABLE 句の出力を制御します。

11. フラット・ファイル・ソースのマッピング用に生成される SQL *Loader スクリプトの OPTIONS 句に影響する次のパラメータを構成します。

    パラレル・ロードの実行 : PARALLEL オプションの値を =TRUE または =FALSE で指定します。true は、複数の同時セッションでダイレクト・ロードを操作できることを示しています。

    エラー許可 : 1 以上の値を指定すると、ERRORS = n オプションが生成されます。SQL*Loader は、このエラー制限に達すると一貫した最初の点でそのロードを終了します。

スキップ対象レコード : 1 以上の値を指定すると、SKIP = n オプションが生成されます。この値は、ファイルの始まりからのロードされない論理レコード数を示します。値を指定しない場合、レコードはスキップされません。

ロード対象レコード : 1 以上の値を指定すると、LOAD = n オプションが生成されます。この値では、ロードされる最大レコード数を指定します。値を指定しない場合、すべてのレコードがロードされます。

コミットごとの行 : 1 以上の値を指定すると、ROWS = n オプションが生成されます。ダイレクト・パス・ロードでは、この値によって、データが保存される前にソースから読み込まれる行数が識別されます。従来のパス・ロードでは、この値によって、バインド配列内の行数を指定します。

読込みサイズ : 1 以上の値を指定すると、READSIZE = n オプションが生成されます。この値は、読込みバッファ・サイズの指定に使用されます。

バインド・サイズ : 1 以上の値を指定すると、BINDSIZE = n オプションが生成されます。この値は、バインド配列の最大サイズ（バイト数）を示しています。

読込みバッファ : 1 以上の値を指定すると、READBUFFERS n 句が生成されます。READBUFFERS では、ダイレクト・パス・ロード時に使用するバッファ数を指定します。必要な場合以外は READBUFFERS に値を指定しないでください。

空白を保持 : このパラメータを true に設定すると、PRESERVE BLANKS 句が生成されます。PRESERVE BLANKS は、オプションで囲みデリミタが存在しない場合に先頭の空白を保持します。また、フィールドが事前に決まっているサイズで指定されている場合、後続の空白もそのまま保持されます。

データベース・ファイル名 : このパラメータによって、ロードされる物理ファイルの特性を指定できます。これらのパラメータの初期値は、マッピングで使用されるフラット・ファイルのプロパティから設定されます。

このパラメータにブランク以外の値を設定すると、FILE= オプションが生成されます。生成されたコードでは、指定した値は単一引用符で囲まれています。

制御ファイルのロケーションおよび**制御ファイル名** : 制御ファイル名は、監査詳細に必要です。

各 SQL*Loader オプションと句の詳細は、『Oracle Database ユーティリティ』を参照してください。

12. 「ランタイム・パラメータ・リファレンス」を開き、配布するマッピングを構成します。

    監査 : 手順の実行時に監査が行われます。

    デフォルト・パージ・グループ : デフォルト・パージ・グループは、パッケージの実行時に使用されます。ランタイム・スキーマの各監査レコードは指定されたパージ・グループに割り当てられます。

13. 「**ソースとターゲットのリファレンス**」を開き、マッピングにおける演算子の物理プロパティを、11-9 ページの「ソースとターゲットのリファレンス」で説明されているように設定します。

PL/SQL マッピングの構成に関する計画

この項では、マッピングを構成するための様々な計画について説明します。この項は、指定されたシナリオにおける適切な構成設定の決定に使用します。

この項では、次のトピックについて説明します。

デフォルト・オペレーティング・モードの選択

PL/SQL 実装でのマッピングの場合、次のオペレーティング・モードのうち 1 つを選択します。

- セット・ベース
- 行ベース
- 行ベース（ターゲットのみ）
- セット・ベースから行ベースへのフェイルオーバー
- セット・ベースから行ベースへのフェイルオーバー（ターゲットのみ）

選択するデフォルト・オペレーティング・モードは、期待するパフォーマンス、必要な監査データの量、マッピングの設計方法によって異なります。マッピングには、行ベースへのフェイルオーバー・モードのオプションを除いても、最低 1 つで、最高 3 つの有効なオペレーティング・モードがあります。コードの作成時に、指定したデフォルトのオペレーティング・モードのコードだけでなく、選択していないモードのコードが生成されます。そのため、実行時には、デフォルト・オペレーティング・モードまたは他の有効なオペレーティング・モードのいずれかで実行するように選択できます。

マッピングの演算子のタイプによって、選択可能なオペレーティング・モードが制限されます。原則として、セット・ベース・モードで実行されるマッピングには、プロシージャとして使用される Match-Merge、Name and Address、変換以外のすべての演算子を含めることができます。行ベース・モードおよび行ベース（ターゲットのみ）モードの演算子をすべて含めることができますが、集計、結合子、キー検索などの SQL ベースの演算子の使用方法に重要な制限があります。SQL ベースの演算子を行ベース・モードのいずれかで使用するには、演算子と関連付けられた操作がカーソルに含められることを確認します。

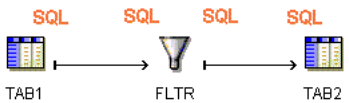
これらの一般的なルールについては、次の項で説明します。演算子の使用方法の詳細は、11-25 ページの「PL/SQL マッピングの無効な設計の回避策」を参照してください。

セット・ベース

セット・ベース・モードでは、すべてのデータを処理し、すべての操作を実行する単一の SQL 文が生成されます。データをセットで処理するとパフォーマンスが向上しますが、使用できる監査情報は制限されます。ランタイム監査でリポートされるのは、実行エラーのみになります。セット・ベース・モードでは、エラーを含む行の詳細を表示できません。

表 11-4 は、セット・ベースのオペレーティング・モードでの実行時に、マッピングのコード生成に使用される、単純なマッピングと関連ロジックを示しています。TAB1、FLTR、TAB2 は、SQL を使用してセットとして処理されます。

図 11-4 セット・ベース・モードでの単純なマッピングの実行



セット・ベース・モードのマッピングを正確に設計するには、Match-Merge 演算子や Name and Address 演算子など、行ごとの処理を必要とする演算子を含めないようにします。SQL で実行できないデータフローの演算子を含めた場合は、セット・ベース・コードが作成されず、セット・ベース・モードでのパッケージの実行時にエラーが発生します。

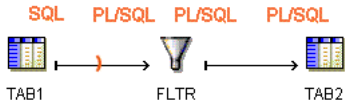
マッピングのターゲット演算子の場合、それらのロード・タイプを INSERT/UPDATE または UPDATE/INSERT のいずれかに設定します。セット・ベース・モードでは、UPDATE または DELETE のロードはサポートされていません。セット・ベースのマッピングで演算子を処理する方法の一覧は、11-28 ページの表 11-3 を参照してください。

行ベース

行ベース・モードでは、データを行ごとに処理する文が生成されます。SQL カーソルには、SELECT 文が使用されます。それ以降のすべての文は、PL/SQL になります。PL/SQL で実行されたすべての演算子では、ランタイム監査の全情報にアクセスできますが、カーソルで実行された演算では、限られた情報にしかアクセスできません。

表 11-5 は、行ベースのオペレーティング・モードでの実行時に、マッピングのコード生成に使用される、単純なマッピングと関連ロジックを示しています。TAB1 はカーソルに含まれ、SQL を使用してセットとして処理されます。FLTR および TAB2 は、PL/SQL を使用して行ごとに処理されます。

図 11-5 行ベース・モードでの単純なマッピングの実行



PL/SQL で実行できない SQL ベースの演算子がマッピングに含まれる場合、カーソルにそれらの演算子を含めてから、コードが生成されます。有効な行ベース・コードを生成するには、次の SQL ベース演算子のいずれかが使用される場合、それらの演算子をカーソルに含めるマッピングを設計します。

- 集計
- デュプリケータ解除
- 結合子
- キー参照
- 順序
- 集合
- ソーター

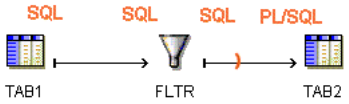
上の演算子をカーソルに含めるには、PL/SQL コードを作成する演算子を直前に配置しないようにします。つまり、プロシージャとして実装されている変換、ソースとして使用されるフラット・ファイル、Match-Merge 演算子、Name and Address 演算子の直後にこれら 7 つの SQL ベース演算子のいずれかが配置されている場合、行ベース・モードのマッピングは実行できません。有効な設計を行うには、PL/SQL を生成する演算子と SQL ベース演算子の間にステージング表を含めます。マッピングの設計方法の詳細は、11-25 ページの「PL/SQL マッピングの無効な設計の回避策」を参照してください。

行ベース（ターゲットのみ）

行ベース（ターゲットのみ）モードでは、SELECT 文のカーソルが生成され、そのカーソルにできるだけ多くの操作が含められます。各ターゲットには、行が個別に挿入されます。PL/SQL で実行されたすべての演算子では、ランタイム監査の全情報にアクセスできますが、カーソルで実行された演算では、限られた情報にしかアクセスできません。データの抽出と変換には高速のセット・ベース操作が、エラーの発生しやすいデータのロードには高度な監査機能が必要な場合、このモードを使用します。

表 11-6 は、行ベース（ターゲットのみ）のオペレーティング・モードでの実行時に、マッピングのコード生成に使用される、単純なマッピングと関連ロジックを示しています。TAB1 および FLTR はカーソルに含まれ、SQL を使用してセットとして処理されます。TAB2 は、行ごとに処理されます。

図 11-6 行ベース（ターゲットのみ）モードでの単純なマッピングの実行



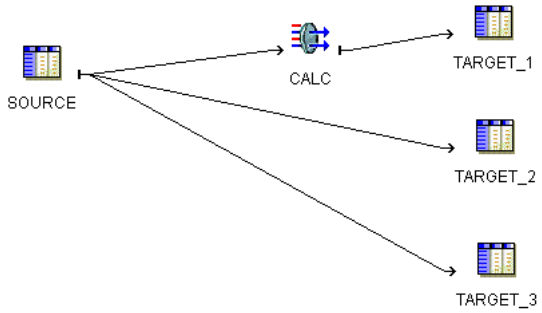
行ベース（ターゲットのみ）は、行ベースのオペレーティング・モードと同じ制限を SQL ベース演算子に加えます。また、複数ターゲットのマッピングの場合は、各ターゲットのカーソルでコードが生成されます。

単一のソースから複数ターゲットへのデータのコミット

共通ソースを基にした複数ターゲットを作成する場合は、ソースからの各行がすべてのターゲットで正確に反映されていることを確認します。

図 11-7 は、このケースを示すマッピングの例です。ターゲット表はすべて、ソース表に基づいています。TARGET_1 に行が追加されるたびに、他のターゲットにも 1 行を追加する必要があります。この関係がデータの再ロード時に維持されない場合、データは不正確になり、使用できなくなることもあります。

図 11-7 複数ターゲットが単一ソースに依存するマッピング



SOURCE からの行数が比較的少ない場合、3 つのターゲットの関係を維持するのは難しくありません。共通ソースに依存するターゲットの関係を手動で維持すると、ソースからの行数を増やしたり、より多くのターゲットと変換を持つ複雑なマッピングを設計する場合、手間がかかります。

Warehouse Builder を使用して、ソースにあるすべての行がすべてのターゲットで正確に反映されていることを確認するには、相関コミット計画でマッピングを構成します。

コミット計画

Warehouse Builder でマッピングのコミット計画を割り当てる場合、単一ソースに依存する複数ターゲットのコミットとロールバック動作を決定します。次の計画から選択できます。

- **独立コミット :** Warehouse Builder では、他のターゲットに関係なく、ターゲットごとに別々にコミットとロールバックを行います。同じ方法でロードされない複数ターゲットの結果が、大きくない場合や相関関係にない場合は、独立コミットを使用します。
- **相関コミット :** Warehouse Builder では、すべてのターゲットは集合的に扱われ、全ターゲットに対してコミットとロールバックが均一に実行されます。影響を受けるターゲットすべてに対して、ソースのあらゆる行が均一な影響を及ぼすようにすることが重要な場合に、相関コミットを使用します。

コミット計画とオペレーティング・モードの組合せによって、マッピング動作が決定されます。表 11-1 は、選択できる有効な組合せを示しています。

表 11-1 オペレーティング・モードの有効なコミット計画

オペレーティング・モード	相関コミット	独立コミット
セット・ベース	有効	有効
行ベース	有効	有効
行ベース（ターゲットのみ）	非推奨	有効

行ベース（ターゲットのみ）の相関コミットは使用しないでください。このオペレーティング・モードは、カーソルがターゲットのできるだけ近くに配置されるように定義されています。ほとんどの場合、SELECT 文の結果は 1 つのターゲットに対してのみ生成され、複数ターゲットへデータをコミットする意味がなくなります。行ベース（ターゲットのみ）と相関コミットを組み合せたマッピングを設計する場合、マッピングは実行されますが、相関コミットが実行されることはほとんどありません。

各オペレーティング・モードとコミット計画の組合せがマッピングに与える影響を理解するには、11-23 ページの図 11-7 にあるマッピングについて考えてみてください。ソース表からのデータが 1,000 個の新規行であると仮定します。マッピングが正常に実行された場合、各ターゲットに 1,000 行がロードされます。マッピングで Target_2 への 100 番目の新規行のロードに失敗した場合、コミット頻度、最大エラー数など他の構成設定からの影響を無視すると、次の結果が得られます。

- **セット・ベース / 相関コミット :** マッピングにエラーが 1 つでもあると、すべてのデータがロールバックされます。Target_2 への挿入中にエラーが検出された場合、その表のエラーがレポートされ、行はロードされません。Target_1 に挿入済のすべての行がロールバックされ、Target_3 には行がロードされません。どのターゲット表にも行が追加されません。エラーの詳細には、Target_2 へのロード中に検出したエラーのみがレポートされます。
- **行ベース / 相関コミット :** 各行が最初の行から個別に評価され、それらが 3 つのターゲットすべてにロードされます。行のロードは、Target_2 に行 100 をロードする際にエラーが発生するまで、続行されます。エラーがレポートされ、行はロードされません。Target_1 に挿入済の行 100 がロール・バックされ、Target_3 には行 100 がロードされません。その後、Target_1 への行 101 のロードが再開され、残りの行のロードが続行されます。その他のエラーが発生しないと、各ターゲットに 999 個の新規行が挿入されて、マッピングが完了します。ソース行がターゲットに正確に反映されます。
- **セット・ベース / 独立コミット :** Target_2 への挿入中にエラーが検出された場合、行がロードされず、その表のエラーがレポートされます。Target_3 への行の挿入は続行されますが、Target_1 の行はロールバックされません。その他のエラーが発生しないと、Target_2 に対する 1 つのエラー・メッセージがレポートされて、マッピングは完了します。Target_2 に行は挿入されず、Target_1 と Target_3 に 1,000 行ずつ挿入されます。ソース行は、ターゲットに正確に反映されません。

- **行ベース / 独立コミット :** 各行が最初の行から個別に評価され、ターゲットにロードされます。行のロードは、Target_2 に行 100 をロードする際にエラーが発生し、そのエラーがレポートされるまで、続行されます。Target_1 の行 100 はロールバックされず、Target_3 には行 100 が挿入されます。その後、残りの行のロードが続行されます。その他のエラーが発生しないと、Target_2 に 999 行が挿入されて、マッピングは完了します。他のターゲットには 1,000 行がそれぞれ挿入されます。ソース行は、ターゲットに正確に反映されません。

相関コミット計画の使用方法

相関コミットは、ロールバック・セグメントのサイズに影響します。セット・ベース・モードでデータをマージ（insert/update または updated/insert）する際には、ロールバック・セグメントに十分な空き領域があるかどうか注意する必要があります。

相関コミットは、PL/SQL バルク処理コードで透過的に操作されます。

相関コミットを使用してすべてのソース行が複数ターゲットに反映されるようにする場合、セット・ベースから行ベースへのフェイルオーバー・オペレーティング・モードを使用してください。この方法で、パフォーマンスとエラー処理のバランスを取ることができます。

相関コミット計画は、パーティション交換ロードで構成されているモードのマッピング、アドバンスト・キュー、Match-Merge、テーブル・ファンクション演算子を含むマッピングでは使用できません。

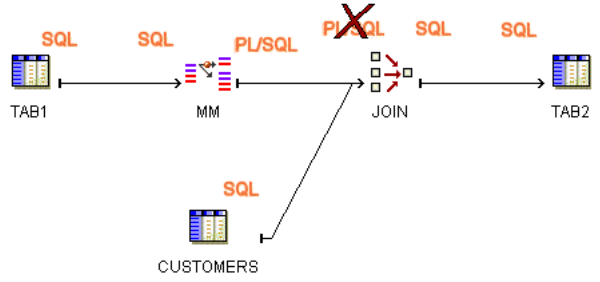
PL/SQL マッピングの無効な設計の回避策

次の基準に合致した PL/SQL マッピングのコードを生成します。

- 各演算子の出力コードは、次のダウンストリーム演算子の入力コード要件を満たします。
- マッピングが PL/SQL 出力のみを作成する演算子を含む場合、すべてのダウンストリーム・データフロー演算子は PL/SQL によって実装可能である必要があります。PL/SQL 出力をターゲットにロードした後、マッピングにある SQL 演算子を使用できます。

マッピングを設計する場合、マッピングの各演算子の入力および出力コードのタイプを記録して、その有効性を評価できます。たとえば、図 11-8 のマッピングは、Match-Merge 演算子 MM が PL/SQL 出力を作成しても、それに続く結合演算子が SQL 出力のみを受け取るので、無効であることがわかります。

図 11-8 結合演算子の入力要件を満たさないマッピング



マッピングで目的の結果を得るには、図 11-9 に示すように Match-Merge を実行する前にソース表を結合するか、図 11-10 に示すように結合を実行する前に Match-Merge からの結果をステージング表にロードします。

図 11-9 Match-Merge の実行前にソースを結合する有効なマッピング設計

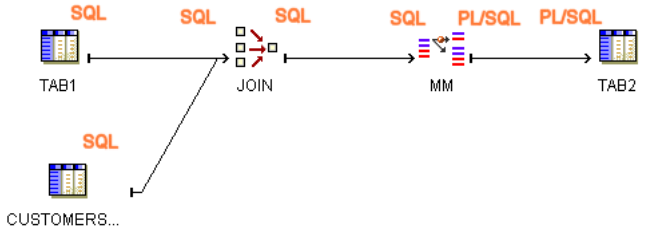


図 11-10 ステージング表を使用した有効なマッピング設計

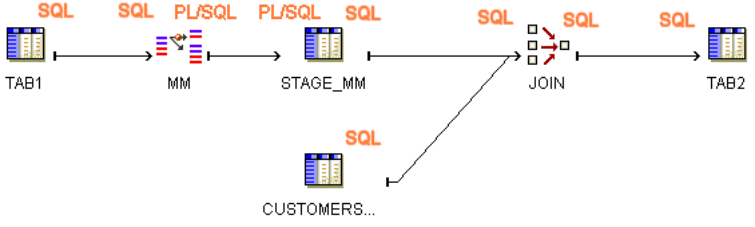


表 11-2 および表 11-3 は、各 Warehouse Builder 演算子の実装タイプを示しています。これらの表は、PL/SQL コードにカーソルの演算子と関連付けられた操作があるかどうかも示しています。指定したマッピング設計で、どのオペレーティング・モードが有効であるかを決定する際に、この情報を使用します。また、エラー処理中にどの監査詳細が使用できるかも決定します。

表 11-2 PL/SQL マッピングでのソースおよびターゲット演算子の実装

演算子	実装タイプ	セット・ベース・モードで有効か	行ベース・モードで有効か	行ベース（ターゲットのみ）で有効か
ソース演算子 : 表、キューブ、ビュー、外部表	SQL	はい。	はい。	はい。カーソルの一部。
ターゲット演算子 : 表、キューブ、ビュー	SQL PL/SQL	はい。ロードが =UPDATE または DELETE の場合を除く。	はい。	はい。カーソルの一部ではない。
ソースとしてのフラット・ファイル	PL/SQL の場合、Create External Table。	はい。	はい。	はい。カーソルの一部。
ターゲットとしてのフラット・ファイル	SQL	はい。ロードが =UPDATE または DELETE のターゲットを除く。	はい。	はい。カーソルの一部ではない。
ソースとしてのアドバンスト・キュー	SQL	はい。	はい。	はい。カーソルの一部。
ターゲットとしてのアドバンスト・キュー	SQL	はい。ロードが =UPDATE または DELETE のターゲットを除く。	はい。	はい。カーソルの一部ではない。
ソースとしての順序	SQL	はい。	はい。	はい。カーソルの一部。

表 11-3 PL/SQL マッピングでのデータ・フロー演算子の実装

演算子名	実装タイプ	セット・ベース・モードで有効か	行ベース・モードで有効か	行ベース（ターゲットのみ）モードで有効か
集計子	SQL	はい。	はい。カーソルの一部である場合のみ。	はい。カーソルの一部である場合のみ。
定数演算子	PL/SQL SQL	はい。	はい。	はい。
データ・ジェネレータ	SQL*Loader のみ	N/A	N/A	N/A
デュプリケータ解除	SQL	はい。	はい。カーソルの一部である場合のみ。	はい。カーソルの一部である場合のみ。
式	SQL PL/SQL	はい。	はい。	はい。
フィルタ	SQL PL/SQL	はい。	はい。	はい。
結合子	SQL	はい。	はい。カーソルの一部である場合のみ。	はい。カーソルの一部である場合のみ。
キー参照	SQL	はい。	はい。カーソルの一部である場合のみ。	はい。カーソルの一部である場合のみ。
入力パラメータのマッピング	SQL PL/SQL	はい。	はい。	はい。
出力パラメータのマッピング	SQL PL/SQL	はい。	はい。	はい。
Match-Merge	SQL 入力 PL/SQL 出力 (XREF グループからの PL/SQL 入力のみ)	いいえ。	はい。	はい。カーソルの一部ではない。
Name and Address	PL/SQL	いいえ。	はい。	はい。カーソルの一部ではない。
ピボット	SQL PL/SQL	はい。	はい。	はい。
マッピング後プロセス	不適切	はい。データフローに依存しない場合。	はい。	はい。

表 11-3 PL/SQL マッピングでのデータ・フロー演算子の実装（続き）

演算子名	実装タイプ	セット・ベース・モードで有効か	行ベース・モードで有効か	行ベース（ターゲットのみ）モードで有効か
マッピング前プロセス	不適切	はい。データフローに依存しない場合。	はい。	はい。
集合	SQL	はい。	はい。カーソルの一部である場合のみ。	はい。カーソルの一部である場合のみ。
ソーター	SQL	はい。	はい。カーソルの一部である場合のみ。	はい。カーソルの一部として。
スプリッタ	SQL	はい。	はい。	はい。
	PL/SQL			
テーブル・ファンクション	SQL または PL/SQL 入力	はい。	はい。	はい。
	SQL 出力のみ			
プロシージャとしての変換	PL/SQL	いいえ。	はい。	はい。カーソルの一部ではない。
DML を実行しないファンクションとしての変換	SQL	はい。	はい。	はい。カーソルに含まれる。
	PL/SQL			

パーティション交換ロードの使用方法

データ・パーティションを使用すると、ターゲット・システムでデータをロードまたは削除するときのパフォーマンスが向上します。このパフォーマンス機能は、パーティション交換ロード（PEL）と呼ばれています。

PEL は、比較的少量のデータを、多量の履歴データを含むターゲットにロードする際に使用してください。ターゲットにできるものとしては、データ・ウェアハウスの表、ディメンション、キューブなどがあります。

この項では、次のトピックについて説明します。

パーティション交換ロードについて

ターゲット・システムのパーティションを操作することによって、パーティション交換ロード（PEL）を使用して、即時にデータを追加または削除できます。表が空のパーティションで交換されると、新規データが追加されます。

PEL を使用すると、新規データをパーティションとしてターゲット表と交換し、ロードできます。たとえば、新規データを保持する表は、ターゲット表からのパーティション識別を引き受け、このパーティションはソース表の識別を引き受けます。この交換処理は、実際のデータの移動を伴わない DDL 操作です。図 11-11 は、この例を示しています。

図 11-11 パーティション交換ロードの概要

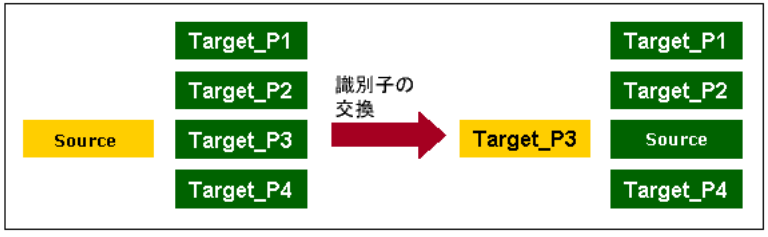


図 11-11 では、ソース表 *Source* からのデータは、4 つのパーティション（Target_P1、Target_P2、Target_P3、Target_P4）で構成されるターゲット表に挿入されます。新規データが Target_P3 にロードされる必要がある場合、パーティション交換演算子は、実際のデータを移動しないで、データ・オブジェクトの名前のみを交換します。交換後、Source という名前が、Target_P3 に変更され、Target_P3 という名前が、Source になります。ターゲット表には、Target_P1、Target_P2、Target_P3、Target_P4 の 4 つのパーティションが含まれたままです。Oracle Database で使用可能なパーティション交換演算子は、データの移動なしにロード処理を完了します。

PEL のマッピングの構成

パーティション交換ロードのマッピングを構成する手順は次のとおりです。

1. ナビゲーション・ツリーで、マッピングを右クリックし、「構成」を選択します。

   図 11-12 に示すように、「構成プロパティ」ウィンドウが表示されます。

図 11-12 PEL 構成プロパティ

2. デフォルトでは PEL はすべてのマッピングに対して無効です。「PEL 有効」を true に設定し、パーティション交換ロードを使用します。
3. 「データ・コレクション頻度」を使用して、マッピングを実行するたびに収集される新規データの量を指定します。このパラメータを設定して、年、四半期、月、日、時間、分ごとのデータの収集を指定します。これにより、パーティション数が決定されます。
4. パーティション交換を実行する前に、収集されたデータを保存する一時表を作成する場合は、「送る」を true に設定します。このパラメータを false に設定すると、一時表を作成せずに、パーティションとして直接ソース表がターゲット表に交換されます。詳細は、11-32 ページの「ダイレクト PEL と間接 PEL」を参照してください。
5. 「データの置換」を true に設定すると、新しく収集されたデータでターゲット・パーティションにある既存のデータが置き換えられます。false（デフォルト）に設定した場合、ターゲット・パーティションにある既存のデータは保持されます。新規データは、空ではないパーティションに挿入されます。このパラメータは、ローカル・パーティションに影響を与え、ターゲット表のパーティションの削除または交換に使用できます。表レベルで、「TRUNCATE/INSERT」プロパティを設定できます。

ダイレクト PEL と間接 PEL

パーティション交換によってターゲットをロードする場合、ターゲットを間接的または直接的にロードできます。

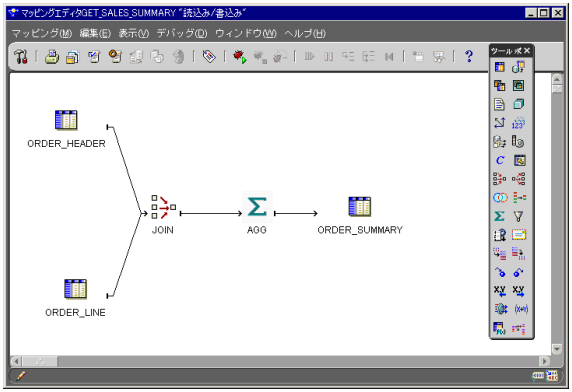
- **間接 PEL:** デフォルトでは、パーティション交換処理を開始する前に、ソース・データを保存する一時表が作成され、維持されます。たとえば、マッピングがリモート・ソースまたは複数ソースの結合を含む場合、間接 PEL を使用します。
- **ダイレクト PEL:** ターゲット構造に一致するマッピングのソースを設計します。たとえば、マッピングでダイレクト PEL を使用すると、以前実行したマッピングでロードしたファクト表をすぐに公開できます。

間接 PEL の使用方法

PEL を使用して設計したマッピングがリモート・ソースまたは複数ソースの結合を含む場合、パーティション交換を続行するには、ソース処理を実行し、データを移動する必要があります。そのため、ダイレクト PEL を false に設定したマッピングを構成します。Warehouse Builder では、ソース処理の結果を格納する一時表が透過的に作成され、維持されます。PEL の実行後、表は削除されます。

図 11-13 は、2 つのソースを結合し、集計を実行するマッピングを示しています。ORDER_SUMMARY 表にロードされたすべての新規データが常に同じパーティションにロードされる場合、このマッピングの間接 PEL を使用すると、ロードのパフォーマンスは向上します。この場合、集計子と ORDER_SUMMARY の間に、一時表が透過的に作成されます。

図 11-13 複数ソースでのマッピング



ターゲット表と同じ構造（列、索引、制約が同じ）を使用して、一時表が作成されます。Warehouse Builder では、最高のパフォーマンスを実現するために、一時表がパラレル直接パス・ロード INSERT を使用してロードされます。INSERT の後に、パラレルで一時表の索引と制約が作成されます。

例 : ダイレクト PEL を使用したファクト表の公開

ソース表がローカルで、データの質が良い場合は、ダイレクト PEL を使用します。ソースとターゲットが同じデータベースにあり、まったく同じ構造を持つマッピングを設計する必要があります。ソースとターゲットの索引、制約、列数、列タイプおよび列の長さは、すべて同じである必要があります。

たとえば、図 11-13 と同じマッピングがあり、データをターゲットへロードするタイミングをより細かく制御するとします。現状では、データ量が多いとロードに時間がかかり、ターゲット表がいつ更新されるかもわかりません。

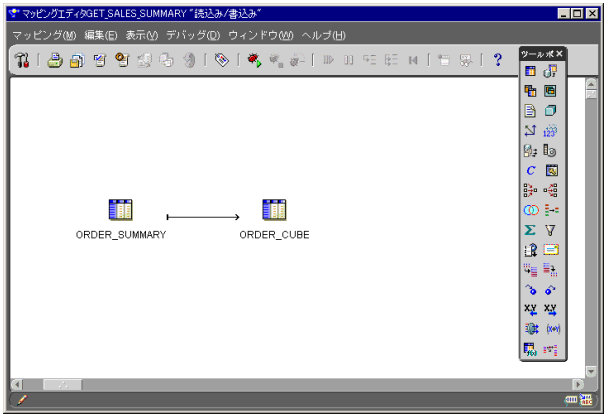
ダイレクト PEL を使用して、データを即時にターゲットにロードする手順は次のとおりです。

1. 必要な場合、ソース・データを結合するマッピングを 1 つ設計し、データの変換および検証を行ってから、それをステージング表にロードします。このマッピングは PEL を使用するように構成しないでください。

   ステージング表が、最終ターゲットの構造に正確に一致するように設計します。たとえば、図 11-13 のステージング表は ORDER_SUMMARY で、図 11-14 の最終ターゲット ORDER_CUBE と同じ構造である必要があります。

2. 図 11-14 に示すように、ステージング表から最終ターゲットにデータをロードする 2 番目のマッピングを作成します。このマッピングがダイレクト PEL を使用するように構成します。

図 11-14 Publish_Sales_Summary マッピング



3. Warehouse Builder のプロセス・フロー・エディタまたは Oracle Workflow を使用して、最初のマッピングの完了後に 2 番目のマッピングを起動します。

効果的な PEL の使用方法

次の条件が true の場合、スケーラビリティの高いロード・パフォーマンスを持つ PEL を効果的に使用できます。

- **表のパーティション化と表領域 :** ターゲット表は、1 つの DATE 列によってパーティション化される必要があります。すべてのパーティションは同じ表領域内に作成される必要があります。すべての表は同じ表領域内に作成されます。
- **既存の履歴データ :** ターゲット表には、多量の履歴データが格納される必要があります。PEL の使用例として、ターゲットが OLTP データベースや Web ログ・ファイルからデータを毎日収集するクリック・ストリーム・アプリケーションがあります。新規データは変換され、履歴データが格納されているターゲットにロードされます。
- **新規データ :** 新規データはすべて、ターゲット表の同じパーティション内にロードされる必要があります。たとえば、ターゲット表が日別に区切られている場合、日別のデータは 1 つのパーティションにロードされる必要があります。
- **ロード頻度 :** ロード頻度は、データ収集頻度以下である必要があります。
- **グローバル索引なし :** ターゲット表にはグローバル索引は含めないでください。

マッピングでのターゲットの構成

PEL のマッピングでターゲットを構成するには、次の手順を実行します。

- **手順 1: すべてのパーティションの作成**
- **手順 2: LOCAL オプションを使用したすべての索引の作成**
- **手順 3:「索引を使用」オプションを使用する主キーまたは一意キー**

手順 1: すべてのパーティションの作成

実行時にパーティションは自動的に作成されません。すべてのパーティションは、PEL を使用する前に作成する必要があります。5-7 ページの「パーティションの作成」を参照してください。

たとえば、新規データの収集頻度として月別を選択した場合、新規データの各月に対してパーティションを作成する必要があります。「構成プロパティ」ウィンドウを使用して、表、ディメンションまたはキューブのパーティションを作成します。図 11-15 は、表 ORDER_SUMMARY の構成プロパティ・インスペクタ・ウィンドウを示しています。この図は、この表に追加された 6 つのパーティションを示します。

PEL を使用するには、すべてのパーティションがネーミング規則に従って命名される必要があります。たとえば、2002 年 5 月のデータを保持するパーティションの場合、パーティション名は Y2002_Q2_M05 という形式にする必要があります。

PEL でパーティションを識別するには、次のいずれかの形式の名前を付ける必要があります。

Ydddd

Ydddd_**Q**d

Ydddd_**Q**d_**M**dd

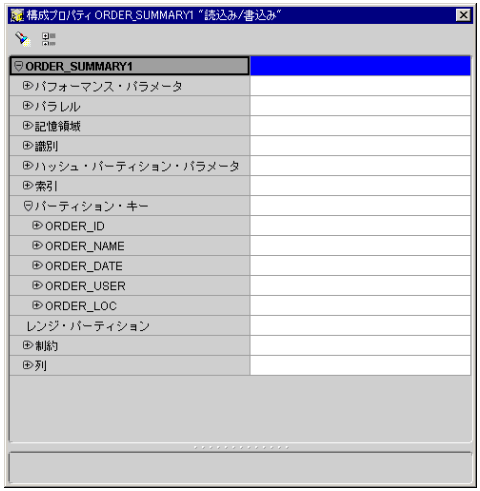
Ydddd_**Q**d_**M**dd_**D**dd

Ydddd_**Q**d_**M**dd_**D**dd_**H**dd

Ydddd_**Q**d_**M**dd_**D**dd_**H**dd_**M**dd

d には、10 進数の数字が入ります。すべての文字は、大文字にする必要があります。小文字は認識されません。

図 11-15 表 ORDER_SUMMARY の構成プロパティ

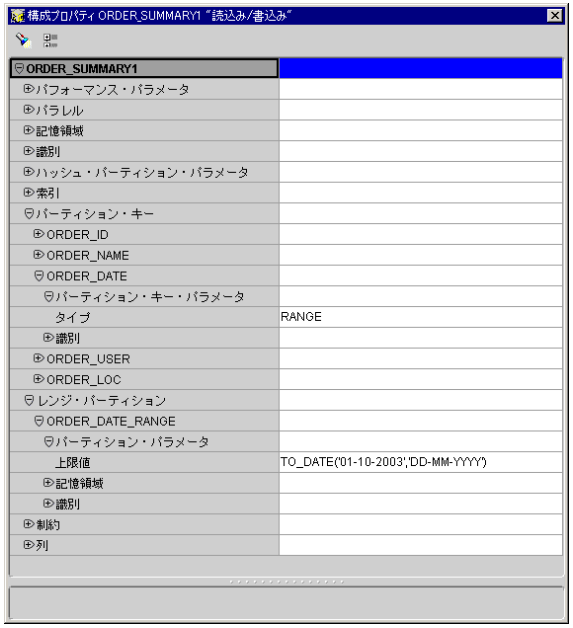


すべてのパーティションが有効な名前で追加された場合、各パーティションの「上限値」プロパティが自動的に計算されます。そうでない場合は、DDL 文が作成されるように、「上限値」を各パーティションに対して手動で構成する必要があります。次に、Warehouse Builder によって作成された DDL 文の例を示します。

```
. . .
PARTITION A_PARTITION_NAME
      VALUES LESS THAN (TO_DATE('01-06-2002','DD-MM-YYYY')),
. . .
```

図 11-16 は、「上限値」パラメータの自動生成された構成値を示しています。

図 11-16 自動生成された「上限値」設定



手順 2: LOCAL オプションを使用したすべての索引の作成

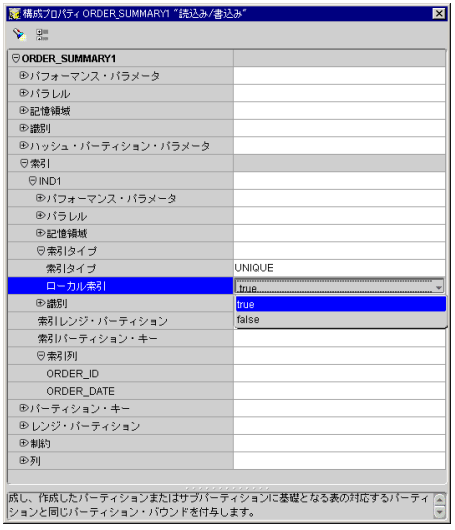
図 11-17 は、ORDER_SUMMARY 表に追加された索引（ORDER_SUMMARY_PK_IDX）を示しています。索引は、2 つの列、ORDER_DATE と ITEM_ID に対して作成し、次のように構成します。次のように構成します。

- 「索引タイプ」パラメータを UNIQUE に設定します。
- 「ローカル索引」パラメータを true に設定します。

これで、表 ORDER_SUMMARY にある一意のローカル索引の DDL 文が作成できるようになります。

ローカル索引を使用すると、PEL のパフォーマンスを最大限に活用できます。ローカル索引では、すべての索引が表と同じ方法でパーティション化されている必要があります。PEL を使用して一時表をターゲット表にスワップする場合、索引セグメントの識別もスワップされます。

図 11-17 ローカル索引としての索引の構成



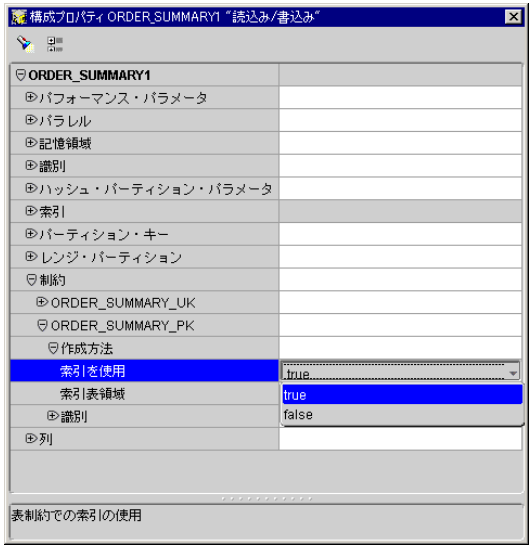
ローカル索引として索引が作成されている場合、Oracle サーバーでは、パーティション・キー列を索引の最初の列にする必要があります。上の例では、パーティション・キーは ORDER_DATE で、索引 ORDER_SUMMARY_PK_IDX の最初の列になります。

手順 3:「索引を使用」オプションを使用する主キーまたは一意キー

この手順では、主キー制約および一意キー制約がすべて、「索引を使用」オプションで作成されるように指定する必要があります。図 11-18 は、ORDER_SUMMARY 表の主キー制約 ORDER_SUMMARY_PK が、「索引を使用」オプションで指定されている例を示しています。

「索引を使用」オプションを使用すると、表に制約を追加する際に索引が自動的に作成されなくなります。サーバーは、制約と同じ列リストを持つ既存の索引を検索します。つまり、主キー制約または一意キー制約とともに、ユーザーが定義した一意のローカル索引も使用する必要があります。制約 ORDER_SUMMARY_PK で必要とされる索引は、11-37 ページの「手順 2: LOCAL オプションを使用したすべての索引の作成」で作成した ORDER_SUMMARY_PK_IDX です。

図 11-18「索引を使用」オプションを持つ制約の指定



Warehouse Builder での PEL 使用の制限

Warehouse Builder で PEL を使用する場合、次の制限があります。

- **日付パーティション・キーは 1 つ :** DATE データ型のパーティション・キー列は 1 つしか使用できません。数値パーティション・キーは、Warehouse Builder ではサポートされていません。
- **一般的な暦法のみ :** 現在の PEL メソッドは、世界中で使用されている一般的な暦法のみをサポートしています。ユーザー定義の会計年度別や四半期別のビジネス暦法は、現時点ではサポートされていません。
- **すべての日付パーティションが同じ表領域内にあること :** ターゲット（表、ディメンション、キューブ）のすべてのパーティションが、同じ表領域内に作成される必要があります。
- **すべての索引パーティションが同じ表領域内にあること :** ターゲット（表、ディメンション、キューブ）のすべての索引が、同じ表領域内に作成される必要があります。ただし、索引とデータでは、別の表領域を使用できます。

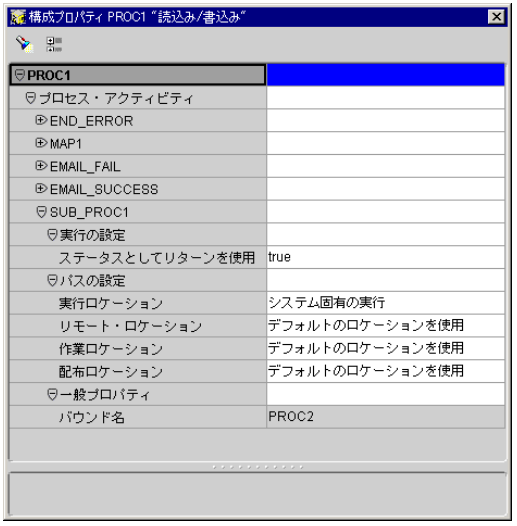
プロセス・フローの構成

プロセス・フローを構成する手順は次のとおりです。

1. プロセス・フロー・パッケージに移動し、プロセス・フローを右クリックして、「構成」を選択します。

   図 11-19 に示されているように、プロセス・フローの「構成プロパティ」シートが表示されます。

図 11-19 プロセス・フローの「構成プロパティ」シート



2. 「実行の設定」を開き、プロパティの「ステータスとしてリターンを使用」を設定します。

   この設定では、それぞれの出力で NUMBER を返すアクティビティの動作を指定します。これらアクティビティには、FTP、外部プロセス、変換などがあります。「ステータスとしてリターンを使用」を true に設定した場合、プロセス・フロー・エディタでは、送信の推移の条件がアクティビティの次の戻り値に基づいて割り当てられます。

   1 = 正常に完了した推移

   2 = 警告推移

   3 = エラー推移

3. 「パスの設定」を開き、プロセス・フローの各アクティビティで、次のプロパティを設定します。

   実行ロケーション: このアクティビティが実行される場所。Oracle Enterprise Manager を構成している場合、OEM エージェントがプロセス・フローを実行するように選択できます。

   リモート・ロケーション: FTP アクティビティのみが実行されるリモートの場所。

   作業ロケーション: FTP、FILE EXISTS および外部プロセスなどのアクティビティのみが実行される作業場所。

   配布ロケーション: 配布場所。この設定は、変換アクティビティにのみ適用されます。事前定義済の変換を参照するアクティビティの場合、「デフォルトのロケーションを使用」で設定を変更してから、有効な場所を指定する必要があります。

4. 「一般プロパティ」を開きます。プロセス・フローに表示されるアクティビティのオブジェクト名である、バウンド名が表示されます。バウンド名を持つアクティビティは、マッピング、変換、サブプロセスのみです。

第 III 部

配布および実行

この部には、次の章があります。

- 第 12 章「オブジェクトの検証」
- 第 13 章「ターゲット・システムの配布」
- 第 14 章「配布と実行の監査」

12

オブジェクトの検証

検証によって、データ・オブジェクトの定義と ETL オブジェクトの定義が確認され、配布時に起こりうるあらゆる問題またはエラーが検出されます。オブジェクトが無効な場合、そのオブジェクトの生成および配布は実行できません。設計プロセスの任意の時点で、オブジェクトを検証し、そのオブジェクトに対するスクリプトを生成できます。これらの機能は配布プロセスでも利用できます。ただし、オブジェクトを定義する際に、スタンドアロンの機能としてこれらの機能を実行すると、完全かつ有効な定義であることを確認できます。さらに、配布前にスクリプトを生成および表示して、問題がないことを確認できます。

この章では、次のトピックについて説明します。

検証について

検証とは、メタデータ定義や構成パラメータを検証するプロセスです。スクリプトを生成および配布する前に、これらの定義を有効にする必要があります。Warehouse Builder では一連の検証テストを実行して、データ・オブジェクトの定義が完全であり、そのスクリプトを生成および配布できることを確認します。これらのテストが完了すると、テスト結果が表示されます。Warehouse Builder でオブジェクト・エディタを起動し、無効なオブジェクトを修正してから、次に進みます。スタンドアロンの検証のほかに、オブジェクトを生成または配布する際にも検証は暗黙的に実行されます。

潜在的な問題を検出して、問題の発生時に対処するには、データ・オブジェクト定義の作成後と、配布するオブジェクトの構成後の 2 段階に分けて検証を実行します。この場合、オブジェクトの定義後よりもオブジェクトの構成後のほうが、広範囲な検証を実行できます。

ヒント： 発生時に問題を解決できるように、オブジェクトの作成時および構成時に検証を実行します。設計や構成を検証していても、同様のエラー・チェックが実行されます。

定義済のオブジェクトを検証する場合、設計したオブジェクトのメタデータ定義に対するエラー・チェックが行われます。たとえば、表を作成する場合、Warehouse Builder では列を定義する必要があります。このオブジェクトの検証時に、Warehouse Builder では、表のすべてのコンポーネントが定義されていることが確認されます。コンポーネントが不足している場合、「検証結果」ウィンドウに検証メッセージが表示されます。

構成後にオブジェクトを検証する場合、問題なくオブジェクトが生成および配布されるように、メタデータ定義に対するエラー・チェックが再度実行され、構成パラメータがチェックされます。その後、無効なオブジェクトを編集できます。

オブジェクトの検証

検証するオブジェクトをいつでも手動で選択できます。「オブジェクト」メニューから、またはオブジェクトを右クリックして、「検証」操作を選択します。

オブジェクトまたはオブジェクト・セットを検証する手順は次のとおりです。

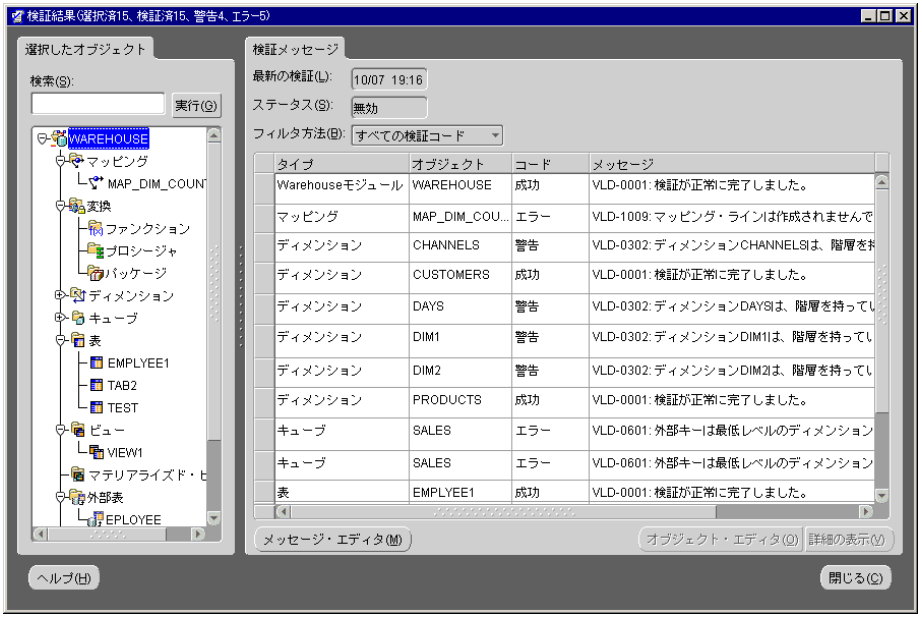
1. プロジェクトのナビゲーション・ツリーで、オブジェクトまたはオブジェクト・セットを選択します。

   複数のオブジェクトを選択するには [Ctrl] キーを使用します。モジュールやプロジェクトなどのオブジェクトを含むオブジェクトも選択できます。

2. 「オブジェクト」メニューから「検証」を選択するか、オブジェクトを右クリックしてポップアップ・メニューから「検証」を選択します。

   Warehouse Builder によって選択されたオブジェクト定義が検証され、図 12-1 が示すように、「検証結果」ダイアログに結果が表示されます。

図 12-1「検証結果」ダイアログ



検証結果の表示

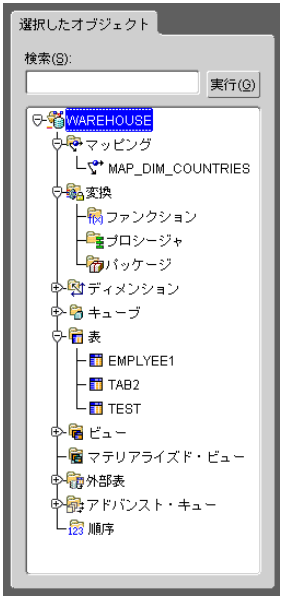
「検証結果」ウィンドウを使用すると、検証結果を表示したり、無効な定義を修正したりできます。「検証結果」ウィンドウでは、左側にナビゲーション・ツリー、右側に検証メッセージが表示されます。これらの領域を使用して、特定のオブジェクトを見つけたり、オブジェクトに対する検証メッセージを表示したりできます。ウィンドウのタイトル・バーには、選択されたオブジェクト数、検証されたオブジェクト数、警告数およびエラー数が表示されます。

注意： 「検証結果」ウィンドウは、定義の検証時に表示されます。これらの結果を後で表示するには、「表示」メニューから「検証メッセージ」を選択してください。

検証結果のナビゲーション・ツリー

検証されたオブジェクトまたはオブジェクト・セットは、図 12-2 に示すように、左側のナビゲーション・ツリーに表示されます。ナビゲーション・ツリーは、大規模なオブジェクト・セットの検証時に特定のオブジェクトを検索する際に役立ちます。

図 12-2 検証のナビゲーション・ツリー



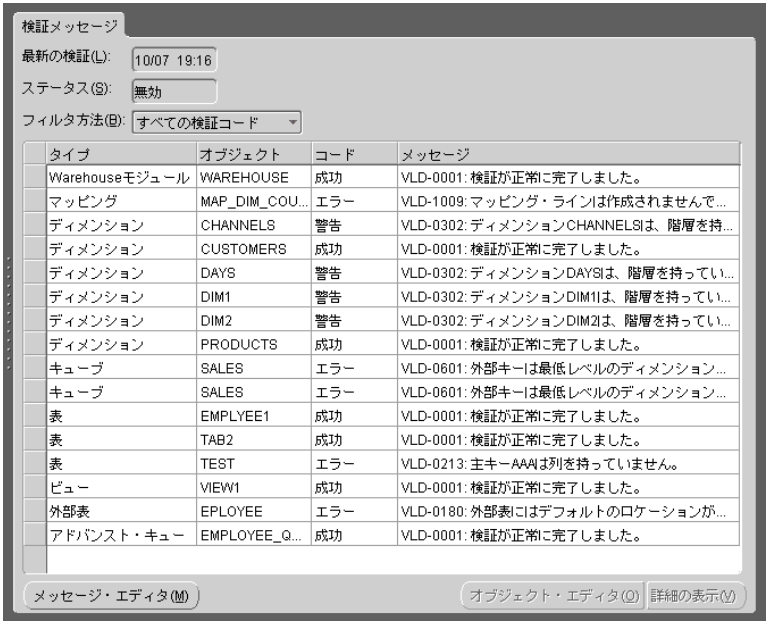
このツリーを使用すると、オブジェクトまたはオブジェクトのタイプを選択したり、右側に特定の検証メッセージを表示したりできます。特定のオブジェクトを検索するには、「検索」フィールドにオブジェクト名を入力して、「実行」をクリックします。検索対象のオブジェクトが現在の「検証結果」ウィンドウ内に含まれている場合、右側にその結果が表示されます。

検証メッセージ

検証メッセージは、「検証結果」ウィンドウの右側に表示されます。選択したオブジェクトに他のオブジェクトが含まれる場合、すべてのオブジェクトが結果に含まれます。たとえば、あるモジュールを検証のために選択した場合、そのモジュール定義のほかに、そのモジュールに含まれるすべてのオブジェクトの定義も検証されます。検証メッセージには、図 12-3 に示すように、オブジェクトのタイプ、オブジェクト名、検証コードおよび検証メッセージが表示されます。

注意： 1 つのオブジェクトに対して複数の検証メッセージが存在する場合があります。複数のエラーまたは警告が発生した場合、これらは別個に表示されます。

図 12-3 検証メッセージ



メッセージ・コードには、次の 3 種類があります。

- **成功 :** このオブジェクトが検証テストに合格したことを示します。このまま続行できます。
- **警告 :** 配布時に問題が発生する可能性があることを示します。これは、致命的なエラーではありません。警告は、潜在的な問題を知らせるために表示されます。
- **エラー :** オブジェクト定義が無効であることを示します。オブジェクト定義を生成できません。

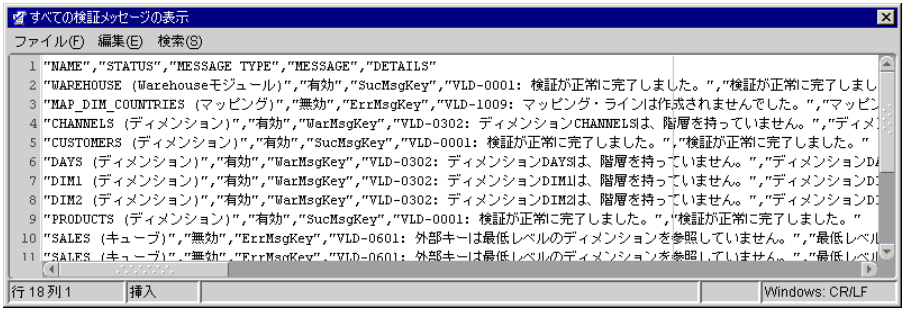
「検証結果」ウィンドウでは、次のボタンを使用できます。

- 「メッセージ・エディタ」ボタン
- 「オブジェクト・エディタ」ボタン
- 「詳細の表示」ボタン

「メッセージ・エディタ」ボタン

「メッセージ・エディタ」ボタンをクリックして、図 12-4 に示すように、テキスト・エディタですべての検証メッセージを表示します。ここに、それぞれの検証エラーまたは警告を解決した方法に関するメモを入力できます。次に、別の無効なオブジェクトの解決方法に関する記録として後で使用できるように、このファイルを保存します。このエディタを起動すると、現在の「検証結果」ウィンドウに含まれるすべてのオブジェクトが表示されます。

図 12-4 検証メッセージ・エディタ



「オブジェクト・エディタ」ボタン

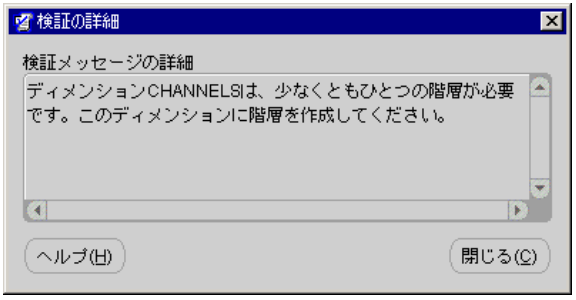
「オブジェクト・エディタ」ボタンをクリックして、選択したオブジェクトに対するオブジェクト・エディタを起動します。たとえば、「検証結果」ウィンドウで表が選択された状態で「オブジェクト・エディタ」をクリックすると、選択した表とともに表エディタが起動します。オブジェクトを編集して、オブジェクト・プロパティまたはオブジェクト定義に関する問題を修正できます。このボタンを有効にするには、検証メッセージ・グリッドの特定のオブジェクトを選択しておく必要があります。

注意：「検証結果」ウィンドウの「オブジェクト・エディタ」ボタンを使用してオブジェクトを編集する場合、オブジェクトは自動的に再検証されません。以前に無効だったオブジェクトをすべて再検証する必要があります。

「詳細の表示」ボタン

「詳細の表示」ボタンをクリックして、図 12-5 に示すように、メッセージに関するより総合的な情報を表示します。詳細なメッセージの一部には、特定の問題を解決するためのヒントやアドバイスが含まれているため、この情報は有用です。このボタンを有効にするには、検証メッセージ・グリッドの特定のオブジェクトを選択しておく必要があります。

図 12-5「検証の詳細」ダイアログ



無効なオブジェクトの編集

オブジェクトの検証時には、「検証結果」ウィンドウが表示されます。ここで無効なオブジェクトを表示したり、これらのオブジェクトに対するエディタに直接アクセスしたりできます。

無効な定義を編集する手順は次のとおりです。

1. 「検証結果」ウィンドウで、ツリーまたは検証メッセージ・グリッドから無効なオブジェクトを選択して、「オブジェクト・エディタ」をクリックします。

   選択したオブジェクトに対するエディタが起動します。

2. オブジェクトを編集して問題を修正します。
3. 問題を修正したらエディタを閉じ、再検証します。

生成済スクリプトの表示

配布前にオブジェクト定義を検証するだけでなく、スクリプトを生成および表示できます。このプロセスでは、Warehouse Builder によって、オブジェクト定義が検証され、オブジェクトを作成および移入するために必要なスクリプトが生成されます。生成結果および各スクリプトに対して生成されたコードを表示できます。これらのスクリプトをファイル・システムに保存することも可能です。

注意： この項で説明する方法で生成するスクリプトは、配布できない場合があります。これは、表示および検証のみを目的とする方法です。これらのスクリプトをファイル・システムに保存することも可能です。

スクリプトの生成

オブジェクトまたはオブジェクト・セットに対するスクリプトを生成する手順は次のとおりです。

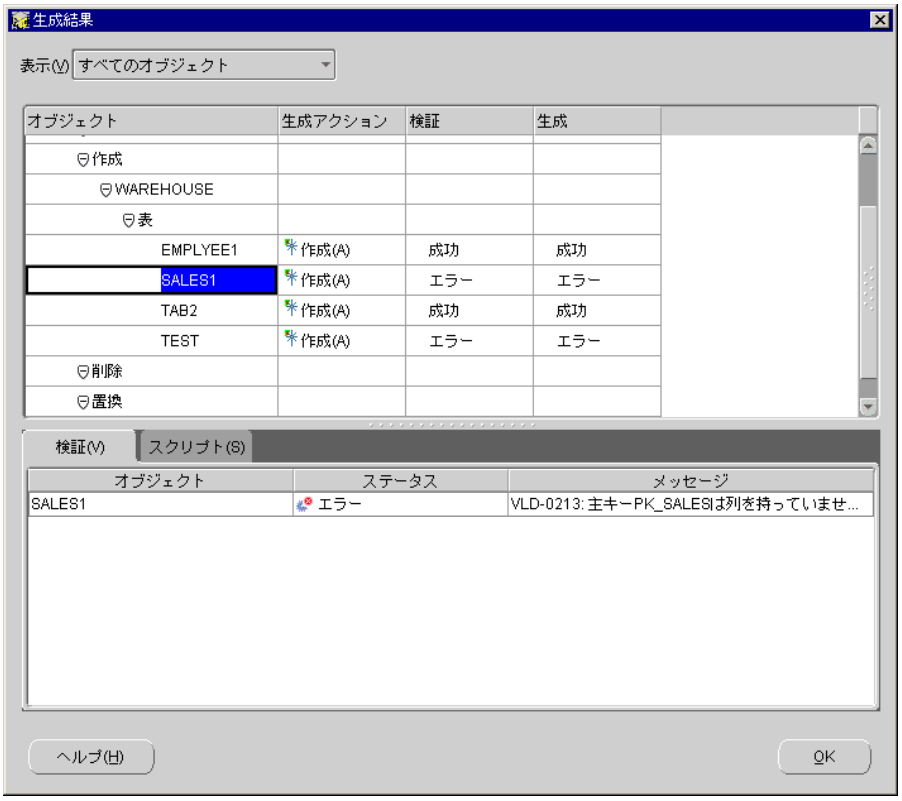
1. プロジェクトのナビゲーション・ツリーで、オブジェクトまたはオブジェクト・セットを選択します。

   [Ctrl] キーを使用して、複数のオブジェクトを選択します。モジュールを選択することもできますが、多数のオブジェクトが含まれる場合、スクリプト生成に時間がかかることがあります。

2. 「オブジェクト」メニューから「生成」を選択します。オブジェクトを右クリックし、「生成」を選択することもできます。

   Warehouse Builder によって、選択されたオブジェクトに対するスクリプトが生成されます。操作が完了すると、図 12-6 に示すように、「生成結果」ウィンドウが表示されます。

図 12-6「生成結果」ウィンドウ



生成結果の表示

「生成結果」ウィンドウは 2 つの主要な領域に分割されています。上部の領域はナビゲーション・ツリーに編成されているので、任意のオブジェクトを見つけることができます。表示されるオブジェクトを制限するドロップダウン・フィルタがあります。オブジェクト・タイプを開いたり閉じたりすることもできます。

注意：「生成結果」ウィンドウは、オブジェクトを作成するスクリプトの生成時に表示されます。これらの結果を後で表示するには、「表示」メニューから「生成済スクリプト」を選択してください。

オブジェクトを選択すると、下部の領域に特定の生成結果および検証結果が表示されます。選択したオブジェクトに対する検証メッセージを表示するには、図 12-7 に示すように、「検証」タブを選択します。このタブは表示のみに使用します。

図 12-7「検証」タブ

選択したオブジェクトを作成するために生成されるスクリプトの一覧を表示するには、図 12-8 に示すように、「スクリプト」タブを選択します。このタブには、オブジェクトの名前、スクリプトの名前および各スクリプトが生成するオブジェクトのタイプが表示されます。

図 12-8「スクリプト」タブ

スクリプトの表示

ターゲット・オブジェクトに対するスクリプトを生成した後、スクリプトを開いてコードを表示できます。Warehouse Builder によって次のタイプのスクリプトが生成されます。

- **DDL スクリプト :** データベース・オブジェクトを作成または削除します。
- **SQL*Loader 制御ファイル :** ファイル・ソースからデータを抽出または転送します。
- **ABAP スクリプト :** SAP システムからデータを抽出またはロードします。

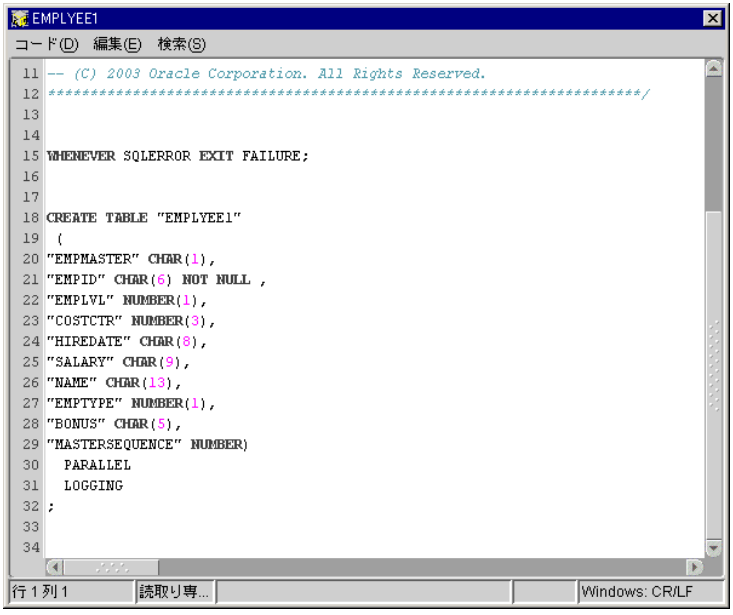
生成済スクリプトを表示する手順は次のとおりです。

1. 「生成結果」ウィンドウで、上部のセクションからオブジェクトを選択します。
2. ウィンドウの下部で「スクリプト」タブを選択します。

   「スクリプト」タブには、選択したオブジェクトに対して生成されたスクリプトの一覧が表示されます。
3. 特定のスクリプトを選択し、「コードの表示」ボタンをクリックします。

   図 12-9 に示すように、選択したスクリプトがコード・ビューアに表示されます。これは読取り専用です。

図 12-9 生成済スクリプト



スクリプトの保存

生成済スクリプトを表示する際、スクリプトをファイル・システムに保存することもできます。

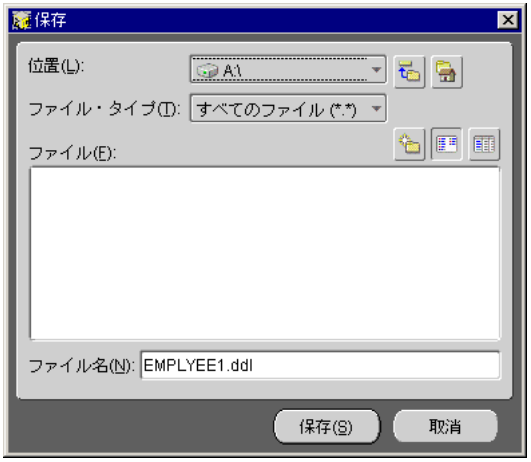
生成済スクリプトを保存する手順は次のとおりです。

1. 「生成結果」ウィンドウで、上部のセクションからオブジェクトを選択します。
2. ウィンドウの下部で「スクリプト」タブを選択します。

   「スクリプト」タブには、選択したオブジェクトに対して生成されたスクリプトの一覧が表示されます。
3. 特定のスクリプトを選択し、「別名で保存」ボタンをクリックします。

   図 12-10 に示すように、「保存」ダイアログで、スクリプト・ファイルを保存するロケーションを選択できます。

図 12-10「保存」オプション



13

ターゲット・システムの配布

ターゲット・システムの論理定義を設計および構成した後、ターゲット・システムの物理インスタンスを配布および作成できます。その後、配布済のマッピングとプロセス・フロー・スクリプトを実行して、データをロードまたは更新できます。すでに配布済のシステムで作業している場合、配布履歴を表示して、アップグレードの計画を立てることができます。これらのプロセスはすべて、Runtime Platform Service と呼ばれる Warehouse Builder のコンポーネントによって管理されます。

この章では、次のトピックについて説明します。

配布について

配布とは、論理設計または論理モデルからターゲット・システムを作成するプロセスです。これには、データ・オブジェクト（表、ビューおよびディメンションなど）を作成するDDL などのスクリプトの生成が含まれます。また、データをデータ・オブジェクトにロードする PL/SQL スクリプトや SQL*Loader の生成も含まれます。Runtime Repository には、各配布に関する詳細情報が格納されています。この情報によって、今後の配布に対するデフォルトの配布アクションが決定されます。たとえば、ターゲット・システム内にすでに存在するオブジェクト・セットを配布するときには、アップグレードがデフォルトのアクションになります。Runtime Audit Browser を使用して、配布データ・レポートにアクセスすることもできます。詳細は、第 14 章「配布と実行の監査」を参照してください。

ターゲット・システムの配布とアップグレード

Warehouse Builder でターゲット・システムを配布またはアップグレードする場合、デプロイメント・マネージャを使用することも、ナビゲーション・ツリーから直接オブジェクトを配布することもできます。デプロイメント・マネージャの総合的な配布コンソールでは、構成や検証など、配布のすべての側面を表示および管理できます。デプロイメント・マネージャでは、オブジェクトの配布履歴を表示して、オブジェクトの配布方法を決定することもできます。ナビゲーション・ツリーからオブジェクトを配布する場合には、これらのオプションは使用できません。

配布前に、次の事項を確認する必要があります。

- 配布可能なすべてのモジュールに対してロケーションが割り当てられていること。
- Runtime Repository がインストール済で、ナビゲーション・ツリーの Runtime Repository 接続によって Runtime Repository が参照されていること。
- 必要に応じてコネクタが定義されていること。

配布前にいつでも、Runtime Repository 接続、ロケーションおよびコネクタを定義できます。これらのオブジェクトは、データソースとターゲットの接続情報を定義するだけでなく、これらのロケーション間の接続も定義するため、配布に必要なものです。これらを定義したら、ターゲット・システムの配布作業に進みます。

Runtime Repository 接続の定義

Runtime Repository 接続とは、設計するターゲット・システムを構成するシステムとツールの集合を表す、Runtime Repository への接続です。Runtime Repository はまた、この集合への配布の管理を支援し、監査データを収集します。これらの接続は、Runtime Assistant を使用してインストールされた Runtime Repository への接続を表します。Runtime Repository の詳細は、『Oracle Warehouse Builder 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。

オブジェクトの配布時には、Runtime Repository 接続を選択する必要があります。論理ロケーションの物理的な詳細は、Runtime Repository 内に登録されています。Runtime Repository は、配布処理時に生成済スクリプトを適切な物理ロケーションに送ります。配布処理中に作成された監査データは、Runtime Repository に格納され、Runtime Audit Browser を使用して表示できます。配布を行う前に、Runtime Repository をインストールし、ナビゲーション・ツリーで Runtime Repository 接続を作成する必要があります。

Runtime Repository 接続の作成

新しい Runtime Repository 接続を作成する手順は次のとおりです。

1. プロジェクトを選択して、ナビゲーション・ツリーを開きます。
2. 「ランタイム・リポジトリ接続」ノードを選択します。
3. 「オブジェクト」メニューの「ランタイム・リポジトリ接続の作成」を選択するか、「ランタイム・リポジトリ接続」を右クリックして「ランタイム・リポジトリ接続の作成」を選択します。

   「新規ランタイム・リポジトリ接続ウィザード: ようこそ」ページが表示されます。
4. 「次へ」をクリックします。

   「新規ランタイム・リポジトリ接続ウィザード: 名前」ページが表示されます。このページで、Runtime Repository 接続について次の情報を定義します。

   - **名前:** Runtime Repository 接続の名前を入力します。名前は 30 文字以内にします。
   - **説明（オプション）:** Runtime Repository 接続の説明（オプション）を入力します。説明は 400 文字以内にします。

   新しい Runtime Repository 接続には、事前に割り当てられたデフォルト値はありません。

5. 「次へ」をクリックして続行します。

   「新規ランタイム・リポジトリ接続ウィザード: 詳細」ページが表示されます。このページで、Runtime Repository について次の情報を定義します。

   - **ホスト名 :** ホスト・マシンの名前を入力します。
   - **ポート番号 :** Oracle Listener のポート番号を指定します。デフォルトは 1521 です。
   - **サービス名 :** サービス名を入力します。
   - **ユーザーとして接続 :** この Runtime Repository へのアクセス権が付与されたユーザー名を入力します。
   - **ランタイム・リポジトリの所有者 :** 接続を作成する Runtime Repository の名前を入力します。これは、Runtime Assistant を使用して入力したスキーマ名を表します。

6. 「次へ」をクリックして続行します。

   「新規ランタイム・リポジトリ接続ウィザード: 終了」ページが表示されます。このページで、新しい Runtime Repository の定義を確認します。このページには、名前、ホスト名、ポート番号、サービス名、ユーザー名および Runtime Repository 名が表示されます。

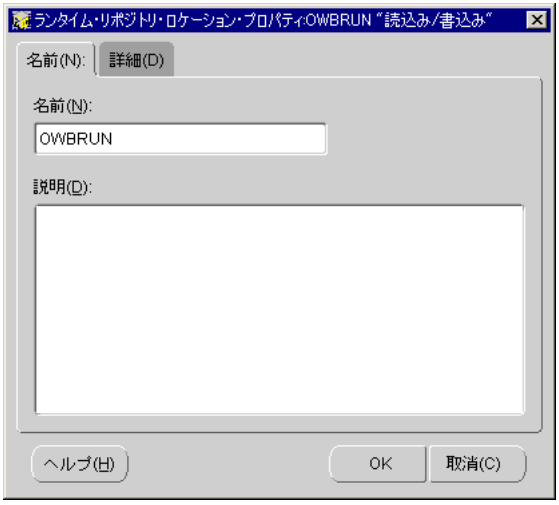
7. 「終了」をクリックして、定義した Runtime Repository 接続を作成します。

   ランタイム・リポジトリ接続が作成され、「Runtime Repository 接続」ノードの下に追加されます。

Runtime Repository 接続の編集

Runtime Repository 接続を編集するには、ナビゲーション・ツリーで Runtime Repository 接続を右クリックし、「プロパティ」を選択します。図 13-1 に示すように、「プロパティ」ウィンドウが表示されます。

図 13-1 Runtime Repository 接続のプロパティ



このページには、選択した Runtime Repository 接続のプロパティが表示されます。次の 2 つのタブを使用して、プロパティを表示および編集します。「OK」をクリックして変更内容を保存するか、「取消」をクリックしてウィンドウを閉じます。

「名前」タブ

- **名前 :** Runtime Repository 接続の名前が表示されます。名前は 30 文字以内にします。
- **説明 :** Runtime Repository 接続の説明（オプション）が表示されます。説明は 400 文字以内にします。

「詳細」タブ

- **ホスト名 :** ホスト・マシンの名前を入力します。
- **ポート番号 :** Oracle Listener のポート番号を指定します。
- **サービス名 :** サービス名を入力します。
- **ユーザーとして接続 :** この Runtime Repository へのアクセス権が付与されたユーザー名を入力します。
- **ランタイム・リポジトリの所有者 :** 接続を作成する Runtime Repository の名前を入力します。これは、Runtime Assistant を使用して入力したスキーマ名を表します。

オブジェクトの配布

オブジェクトを配布する準備が整ったら、デプロイメント・マネージャを使用することも、ナビゲーション・ツリーでオブジェクトを選択して配布することもできます。ナビゲーション・ツリーからオブジェクトを配布する方法は非常に簡単で、配布プロセスも短時間で終わります。Warehouse Builder でデフォルトの配布アクションを使用し、配布前にスクリプトを表示する必要がない場合は、この方法が便利です。デフォルトの配布アクションは、そのオブジェクトが最後に配布された後に、オブジェクトの設計に対してどのような変更が加えられたかによって決定されます。たとえば、変更されたオブジェクトを配布する場合、デフォルトの配布アクションはアップグレードになります。

デプロイメント・マネージャの総合的な配布コンソールでは、より柔軟な配布オプションを使用できます。詳細は、13-6 ページの「デプロイメント・マネージャの使用方法」を参照してください。

ナビゲーション・ツリーからオブジェクトを配布する手順は次のとおりです。

1. ナビゲーション・ツリーで配布可能なオブジェクトを選択します。
2. 「オブジェクト」メニューから「配布」を選択するか、オブジェクトを右クリックして「配布」を選択します。

   「ランタイム・リポジトリ接続の選択」ダイアログが表示されます。
3. Runtime Repository 接続を選択し、「OK」をクリックします。

   Warehouse Builder によって、選択したオブジェクトに対するスクリプトが生成され、「配布前のランタイム・リポジトリの生成結果」ウィンドウが表示されます。
4. 「配布」をクリックして続行するか、「取消」をクリックして取り消します。

   Warehouse Builder は、デフォルトの配布設定を使用して、オブジェクトを配布します。選択する Runtime Repository で、配布に関するデータが格納されます。

デプロイメント・マネージャの使用方法

デプロイメント・マネージャでは、Warehouse Builder を使用する際の最も柔軟な配布方法が提供されます。デプロイメント・マネージャを起動すると、現在のプロジェクト内にある設計オブジェクトにアクセスできます。デプロイメント・マネージャの起動後、ツリーからオブジェクトを選択したり、オブジェクトの配布アクションを設定したりできます。ツリー上では、これらのオブジェクトの隣に特別なアイコンが表示されます。オブジェクトを選択して配布アクションを設定した後、「配布」ボタンを使用して、オブジェクトを検証し、現在の設計に基づいてスクリプトを生成できます。これらの結果は、「配布前のランタイム・リポジトリの生成結果」画面に表示されます。配布するオブジェクトを確認するだけでなく、エラーも見つけることができます。現在の Runtime Repository 接続を使用して、ターゲット・ロケーションにスクリプトを配布し、配布処理を完了します。

デプロイメント・マネージャの起動

デプロイメント・マネージャを起動する手順は次のとおりです。

1. 「プロジェクト」メニューから「デプロイメント・マネージャ」を選択します。

   Runtime Repository 接続を選択するためのダイアログが表示されます。

2. ドロップダウン・リストから Runtime Repository 接続を選択し、「OK」をクリックします。

   選択した Runtime Repository の「接続情報」ダイアログが表示されます。

3. 選択した Runtime Repository のパスワードを入力します。

   特定の Runtime Repository に初めて接続する場合、パスワードを確認する必要があります。その後パスワードが保存され、Warehouse Builder セッション中、パスワードを再度入力することなく、リポジトリにアクセスできるようになります。

   図 13-2 に示すように、左側のウィンドウ内に現在のプロジェクトが表示された状態で、デプロイメント・マネージャが起動します。

図 13-2 デプロイメント・マネージャ

デプロイメント・マネージャについて

デプロイメント・マネージャは、次のコンポーネントで構成されています。

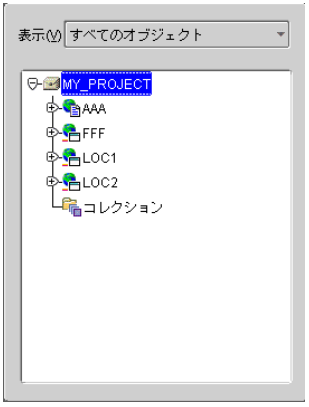
- デプロイメント・ツリー
- 「詳細」タブ
- 「履歴」タブ
- ツールバー

デプロイメント・ツリー

デプロイメント・マネージャを起動すると、デプロイメント・ツリーが左側に表示されます。図 13-3 に示すように、最初このツリーは閉じており、ロケーションとコレクションの一覧が表示されていますが、開いてその内容を表示することもできます。このツリーを使用して、ロケーションの登録、配布するオブジェクトの選択、および配布履歴の表示を行うことができます。関連付けられたロケーションを持たないモジュールは、デプロイメント・マネージャ内に表示されません。

注意： ロケーションに含まれる配布可能なオブジェクトを表示できるのは、Oracle ターゲット・ウェアハウスのロケーションのみです。すべてのソース・ロケーションおよびフラット・ファイル・ロケーションについては、登録目的のみでロケーションが表示されます。

図 13-3 デプロイメント・ツリー



ツリーでオブジェクトを選択すると、そのオブジェクトに含まれる配布可能なオブジェクトが右側に表示されます。[Ctrl] キーまたは [Shift] キーを使用して、デプロイメント・ツリーから複数のオブジェクトを選択できます。

表示セレクタ： 表示されるオブジェクトを制限するために、デプロイメント・ツリーの上部にあるドロップダウン・リストで表示を選択します。表 13-1 では、表示セレクタに表示される列を説明します。

表 13-1 表示

表示の名前	表示されるオブジェクト
すべてのオブジェクト	すでに選択されているフィルタを解除し、デプロイメント・ツリー内にすべてのオブジェクトを表示します。
変更済オブジェクト	最後に配布された後に、Warehouse BuilderDesign Repository 内で変更されたオブジェクトです。これらのオブジェクトの配布アクションは、デフォルトでアップグレードまたは置換に設定されています。
配布済オブジェクト	現在選択されている Runtime Repository を使用してすでに配布されているオブジェクトです。
投影された配布オブジェクト	配布目的で現在選択されているすべてのオブジェクトです。

「詳細」タブ

デプロイメント・マネージャの右側にある「詳細」タブには、デプロイメント・ツリー内で選択されたオブジェクトまたはオブジェクト・セットに関連した情報のサマリーが表示されます。オブジェクトのステータス変更に伴い、「詳細」タブにも変更が反映されます。また、タブの下部にあるボタンを使用して、「配布アクション」ステータスを変更できます。「デフォルト・アクション」ボタンをクリックすると「配布アクション」ステータスが変更され、「リセット」ボタンをクリックすると「配布アクション」列が「なし」にリセットされます。図 13-4 は、「詳細」タブが表示された「配布」ウィンドウを示します。

図 13-4 デプロイメント・マネージャの「詳細」タブ

表 13-2 では、「詳細」タブに表示される列を説明します。

表 13-2「詳細」タブの列

列	説明
オブジェクト	選択したオブジェクトの物理名を表示します。
設計ステータス	Design Repository に格納されたオブジェクトに関するステータス情報を表示します。 ステータスは次のとおりです。 新規 : Design Repository にオブジェクトが存在しますが、まだ配布されていません。 変更なし : 最後に配布された後、何も変更がありません。 削除済 : オブジェクトは Runtime Repository にありますが、Design Repository にはありません。 更新済 : オブジェクトは配布済で、最後に配布された後に Design Repository で更新されています。
配布アクション	オブジェクトの配布時に実行されるアクションです。アクションには、「作成」、「アップグレード」、「削除」、「置換」があります。
配布済	オブジェクトが最後に配布されたときのタイムスタンプです。
配布ステータス	ターゲットにあるオブジェクトの現在の配布ステータスです。
メッセージ	メッセージを表示します。

「デフォルト・アクション」ボタンをクリックすると、選択したオブジェクトのアクションが更新されます。デフォルトのアクションは、Design Repository と Runtime Repository に格納されているデータによって決定されます。オブジェクトの詳細は「詳細」タブに表示されます。次のルールによって、デフォルトのアクションが決定されます。

- 配布されていない設計オブジェクトは、アクションが「作成」に設定されます。
- すでに配布されているオブジェクトが最後に配布された後に修正されている場合、これらのオブジェクトのアクションは、ターゲット・システムの構成によって、「アップグレード」または「置換」に設定されます。
- オブジェクトを配布する必要がない場合、アクションは「なし」に設定されます。

注意 : 外部表については、これらのオブジェクトの処理方法が原因で、使用できるデフォルトのアクションはありません。外部表の場合、手動で「配布アクション」を設定する必要があります。

「履歴」タブ

表 13-5 に示すように、「履歴」タブには、選択したオブジェクトの配布履歴が表示されます。

図 13-5 デプロイメント・マネージャの「履歴」タブ

表 13-3 では、「履歴」タブに表示される列を説明します。

表 13-3「履歴」タブの列

列	説明
オブジェクト	選択したオブジェクトの物理名を表示します。
配布アクション	オブジェクトの配布時に実行されたアクションです。アクションには、「作成」、「アップグレード」、「削除」、「置換」があります。
配布済	オブジェクトが配布されたときのタイムスタンプです。
オブジェクト・ステータス	配布されたオブジェクトが有効であるか無効であるかを表示します。
配布ステータス	ターゲットにあるオブジェクトの現在の配布ステータスです。

ツールバー

表 13-6 に示すように、デプロイメント・マネージャの左上にあるツールバーには、複数のタスクへのショートカットが含まれています。

図 13-6 デプロイメント・マネージャのツールバー

表 13-4 では、これらのツールを説明します。

表 13-4 デプロイメント・マネージャのツールバー

アイコン	アクション	説明
	このビューをランタイム・プラットフォームと同期化	デプロイメント・マネージャに表示されるオブジェクトと、Runtime Repository に存在するオブジェクトを同期化します。この機能は、デプロイメント・マネージャでのみ使用できます。
	このビューをリポジトリと同期化	表示されるオブジェクトと Warehouse BuilderDesign Repository に存在するオブジェクトを同期化します。これは、マルチユーザーの場合に便利な機能です。また、メイン・コンソールの同期化ボタンと同じ機能を提供します。
	検索	デプロイメント・マネージャ内でオブジェクトを検索します。「オブジェクト検索」ダイアログが表示されます。検索するオブジェクトの名前または名前の一部を入力できます。このアイコンはいつでも選択できます。
	生成 / 配布	選択したオブジェクトを生成 / 配布します。
	実行	デプロイメント・ツリーで選択されたオブジェクトを実行します。これは、実行可能なオブジェクトが選択されている場合にのみ使用できます。
	ヘルプ	オンライン・ヘルプ・ナビゲータに『Oracle Warehouse Builder ユーザーズ・ガイド』を表示します。このアイコンはいつでも選択できます。

ロケーションの登録

オブジェクトを正常に配布するには、使用するすべてのロケーションを登録する必要があります。設計プロセス中にロケーションを作成する際、ロケーションの名前、タイプおよびバージョンのみを指定する論理定義を作成します。この論理情報は保存され、配布できるオブジェクトを決定するために使用されます。配布時にモジュールを使用するには、すべてのモジュールにロケーションが割り当てられている必要があります。これには、ターゲットだけでなく、ソースまたはファイルも含まれます。

ロケーションの登録時には、接続情報を指定します。配布の際には接続情報を使用して、様々なデータソースおよびターゲットに接続します。配布時に初めてロケーションを使用するときにのみ、この情報を指定する必要があります。接続情報を変更しないかぎり、同じ接続情報が今後の配布にも使用されます。

データベース接続

ホスト名、ポート番号とサービス名を使用するか、またはネット・サービス名とサービス名を使用して、データベース・ロケーションを定義できます。ネット・サービス名は、tnsnames.ora ファイルに定義されている名前です。ネット・サービス名は、該当する Oracle ホームに定義されています。たとえば、ネット・サービス名によって識別されるロケーションを配布するには、その名前が Runtime Platform Service の Oracle ホームに定義されている必要があります。

データベース・リンクは、<service-name>@<connector-name> の形式で生成されます。グローバル名が true に設定されている場合、ロケーションに対して指定するサービス名がリンク先のデータベースのグローバル名と一致する必要があります。

RAC システムの場合、サービス名は通常、クラスタ・サービス名になります。RAC を使用する場合は、ネット・サービス名の使用をお薦めします。ネット・サービス名を使用すると、クライアント側のロード・バランシングなど、tnsnames で定義される RAC 機能の一部を制御できるためです。

ロケーションの登録

ロケーションを登録する手順は次のとおりです。

1. デプロイメント・マネージャを起動して、デプロイメント・ツリーでロケーションを選択します。

   注意： 配布プロセス中でも、ロケーションを登録できます。配布時に、まだ登録されていない各ロケーションについて「ロケーション登録」ウィンドウが表示されます。

2. 「ファイル」を選択し、「登録」を選択します。ロケーションを右クリックして「登録」を選択することもできます。

   「ロケーション登録」ウィンドウが表示されます。ロケーションのタイプによって、接続情報の要件が異なります。事前に割り当てられたデフォルト値はありません。

3. ロケーションの登録詳細を入力し、「OK」をクリックしてデプロイメント・マネージャに戻ります。次の項では、各タイプのロケーションで必要な登録情報について、詳細に説明します。

Oracle ロケーションの場合：

- **スキーマ名：**データベースにログオンするためのユーザー名を入力します。最大 30 文字です。
- **パスワード：**指定したユーザー名に対するパスワードを入力します。最大 30 文字です。
- **サービス名：**データベース・サービス名を入力します。最大 30 文字です。

「ネット・サービス名」または「ホスト名」を選択して、次の接続詳細を指定します。

- **ネット・サービス名 :** データベースの tnsname を入力します。tnsname には tnsnames.ora ファイル内のサービス名が含まれます。最大 30 文字です。
- **ホスト名 :** マシンの名前を入力します。最大 30 文字です。
- **ポート番号 :** Oracle Listener のポート番号を指定します。最大 5 文字です。

Oracle 以外のロケーションの場合 :

- **スキーマ :** データベース・オブジェクトの所有者を入力します。最大 30 文字です。
- **ユーザーとして接続 :** データベースに接続するためのユーザーを入力します。最大 30 文字です。
- **パスワード :** 指定したユーザー名に対するパスワードを入力します。最大 30 文字です。
- **サービス名 :** Oracle 以外のデータベースのデータベース・サービス名を入力します。最大 30 文字です。

「ネット・サービス名」または「ホスト名」を選択して、次の接続詳細を指定します。

- **ネット・サービス名 :** データベースの tnsname を入力します。tnsname には tnsnames.ora ファイル内のサービス名が含まれます。最大 30 文字です。
- **ホスト名 :** Oracle 以外のデータベースを実行するマシンの名前を入力します。最大 30 文字です。
- **ポート番号 :** Oracle Listener のポート番号を指定します。最大 5 文字です。

ファイル・システムのロケーションの場合 :

- **ユーザー名 :** オペレーティング・システムに接続するためのユーザー名を入力します。このユーザー名は、FTP プロセスなどで使用します。最大 30 文字です。
- **パスワード :** 指定したユーザー名に対するパスワードを入力します。最大 30 文字です。
- **ホスト名 :** フラット・ファイルを含むマシンの名前を入力します。最大 30 文字です。
- **ルート・パス :** ファイルの格納場所であるディレクトリへのパスを指定します。指定するパスは、スラッシュで終了する必要があります。例 : c:/temp/

Oracle Enterprise Manager ロケーションの場合 :

- **ユーザー名 :** ジョブ登録権限を持つ Oracle Management Server ユーザーの名前を入力します。Oracle Enterprise Manager のスーパー・ユーザーは通常、適切な権限を所有しています。デフォルトのスーパー・ユーザーは、SYSMAN です。最大 30 文字です。
- **パスワード :** 指定したユーザー名に対するパスワードを入力します。最大 30 文字です。

- **OMS ドメイン :** Oracle Management Server を実行するノードの名前を入力します。最大 30 文字です。
- **ターゲットのエージェント名 :** コードを実行するノードの名前を入力します。最大 30 文字です。

Oracle Workflow ロケーションの場合 :

- **スキーマ :** Oracle Workflow サーバーのユーザー名を入力します。最大 30 文字です。
- **パスワード :** 指定したユーザー名に対するパスワードを入力します。最大 30 文字です。
- **サービス名 :** Workflow サーバーのデータベース・サービス名を入力します。最大 30 文字です。

「ネット・サービス名」または「ホスト名」を選択して、次の接続詳細を指定します。

- **ネット・サービス名 :** データベースの tnsname を入力します。tnsname には tnsnames.ora ファイル内のサービス名が含まれます。最大 30 文字です。
- **ホスト名 :** Workflow サーバーを実行するマシンの名前を入力します。最大 30 文字です。
- **ポート番号 :** Oracle Listener のポート番号を指定します。最大 5 文字です。

SAP ロケーションの場合 :

- **スキーマ :** SAP データ・オブジェクトへのアクセス権を持つユーザーの名前を入力します。最大 30 文字です。
- **パスワード :** 指定したユーザー名に対するパスワードを入力します。最大 30 文字です。
- **サービス名 :** SAP インスタンスのデータベース・サービス名を入力します。最大 30 文字です。

「ネット・サービス名」または「ホスト名」を選択して、次の接続詳細を指定します。

- **ネット・サービス名 :** データベースの tnsname を入力します。tnsname には tnsnames.ora ファイル内のサービス名が含まれます。最大 30 文字です。
- **ホスト名 :** SAP を実行するマシンの名前を入力します。最大 30 文字です。
- **ポート番号 :** Oracle Listener のポート番号を指定します。最大 5 文字です。

配布するオブジェクトの選択

配布するオブジェクトを選択する手順は次のとおりです。

1. デプロイメント・マネージャを起動します。

   デプロイメント・マネージャでは、左側のペインにプロジェクト・ツリーが閉じた状態で表示されます。

2. プロジェクトを開いて、内容を表示します。次のフィルタのいずれかを選択して、オブジェクト・リストの一部を表示します。

   - **変更済 :** 最後に配布された後に変更されたすべてのオブジェクトを表示します。
   - **配布済 :** Runtime Repository に存在するすべてのオブジェクトを表示します。
   - **投影された配布オブジェクト :** 配布済のオブジェクト、および配布のために現在選択されているオブジェクトをすべて表示します。

3. 配布するオブジェクトまたはオブジェクト・セットを選択します。オブジェクトに関する詳細が、ウィンドウの右側ペインにある「詳細」タブに表示されます。

4. 初めての配布でオブジェクトを選択する場合は、「配布アクション」列をクリックして、ドロップダウン・リストから「作成」を選択します。

   これが初めての配布でない場合は、他のアクションを選択できます。アクションには、「作成」、「アップグレード」、「削除」、「置換」があります。

   また、「デフォルト・アクション」をクリックすることもできます。これによって、各オブジェクトの「配布アクション」がデフォルト値に変更されます。オブジェクトが以前配布されていない場合は、デフォルトのアクションが「作成」になります。

5. 配布するすべてのオブジェクトに対してこのプロセスを続行します。デプロイメント・ツリー上では、配布のために選択されたオブジェクトの隣に特別なアイコンが表示されます。

注意 : ドロップダウン・リストに表示されていたとしても、すべてのオブジェクトに対してすべての配布アクションが使用できるわけではありません。たとえば、マッピングに対する配布アクションとして「アップグレード」を選択することはできません。古いマッピングを置き換える場合は、「置換」アクションを使用する必要があります。

配布履歴の表示

デプロイメント・マネージャでは、プロジェクト内のオブジェクトの配布履歴を表示できます。これは、アップグレードについて判断する際に役立ちます。

配布履歴を表示する手順は次のとおりです。

1. デプロイメント・マネージャを起動します。
2. 「履歴」タブを選択します。
3. ここでツリー上の項目を選択して、配布履歴を表示できます。

配布の完了

配布するオブジェクトを選択した後、いくつかの手順を実行して配布を完了します。

配布を完了する手順は次のとおりです。

1. 「ファイル」メニューから「配布」をクリックするか、ツールバーの「配布」アイコンをクリックします。

   オブジェクトを初めてロケーションに配布する場合、そのロケーションの物理的な詳細を指定する必要があります。

2. ロケーションの物理的な接続詳細を指定して、「OK」を押します。

   「配布前のランタイム・リポジトリの生成結果」ページが表示されます。

   ダイアログの下半分にあるスクリプトを選択し、「コードの表示」をクリックすると、生成されたスクリプトが表示されます。

3. 「配布」をクリックして、配布を確認します。

   配布が完了し、配布結果が表示されます。

4. 配布結果を確認し、「OK」をクリックします。

   これで、配布が終了しました。配布したオブジェクトの配布アクションと配布ステータスは、デプロイメント・マネージャ内で更新されます。

配布スクリプトの保存

場合によっては、特定の配布でのすべての配布スクリプトを保存すると役立つこともあります。これには、デプロイメント・マネージャを使用します。

配布スクリプトを保存する手順は次のとおりです。

オブジェクトを配布する際、配布が完了する前に「配布前のランタイム・リポジトリの生成結果」ページが表示されます。このページで、個々のファイルまたは配布全体の仕様を保存できます。

配布済オブジェクトの実行

配布後、2 つのタイプのオブジェクト（マッピングとプロセス・フロー）を実行できます。Warehouse Builder を使用して設計および配布されたマッピングとプロセス・フローを実行するための、最も直接的な方法は、デプロイメント・マネージャを使用することです。ターゲット・システムにプロセス・フローまたはマッピングを配布すると、これらは実行可能になります。デプロイメント・マネージャのプロジェクト・ツリーでマッピングまたはプロセス・フローを選択し、実行します。

マッピングとプロセス・フローの実行

実行するオブジェクトを選択する手順は次のとおりです。

1. デプロイメント・マネージャを起動して、実行するオブジェクトを含むプロジェクトを開きます。

   デプロイメント・マネージャでは、左側のペインにプロジェクト・ツリーが開いた状態で表示されます。すべてのオブジェクトは、割り当てられたロケーションの下に表示されます。ロケーション・ノードを開いてオブジェクトを表示します。

2. デプロイメント・ツリー・フィルタを使用して、「配布済オブジェクト」を選択します。

   これによって、デプロイメント・ツリーに表示されるオブジェクトが、配布済のオブジェクトに制限されます。

3. デプロイメント・ツリーでマッピングまたはプロセス・フローを選択します。

   選択されたオブジェクトは、デプロイメント・マネージャの右側にある「詳細」タブに表示されます。

   注意： 一度に実行できるのは、1 つのマッピングまたは 1 つのプロセス・フローのみです。

4. 「ファイル」メニューから「実行」をクリックするか、ツールバーの「実行」をクリックします。

   「実行」ダイアログが表示されます。

5. 実行名と実行パラメータを指定します。

6. 「OK」をクリックします。

   オブジェクトが実行され、その結果が表示されます。

7. 結果を確認し、「OK」をクリックします。

   これで実行が完了します。引き続き他のマッピングまたはプロセス・フローを実行できます。

プロセス・フローの実行についての詳細

Warehouse Builder では、プロセス・フローを実行するための主要なオプションが 2 つあります。前述のようにデプロイメント・マネージャを使用して Warehouse Builder 内からプロセス・フローを実行することも、Oracle Workflow からプロセス・フローを実行することもできます。さらに、Warehouse Builder を使用して Oracle Enterprise Manager と統合し、これらのプロセス・フローの実行をスケジュールすることもできます。

Oracle Workflow の詳細は、Oracle Workflow のマニュアルを参照してください。

プロセス・フローを SQL*Plus からも実行できます。

SQL*Plus によるマッピングおよびプロセス・フローの実行

デプロイメント・マネージャを使用して配布済オブジェクトを実行することに加えて、SQL*Plus を使用することもできます。このために、Warehouse Builder で用意されているスクリプト（スクリプト名は sqlplus_exec_template.sql）を使用することができます。このスクリプトは *<OWB home>¥owb¥rtp¥sql* にあります。

Warehouse Builder が生成したスクリプトを SQL*Plus で実行する手順は次のとおりです。

1. SQL*Plus セッションを開始してから、次の構文を使用して sqlplus_exec_template.sql スクリプトを SQL プロンプトで実行します。

   ```
   @sqlplus_exec_template.sql rt_owner location_name {PLSQL |SQL_
   LOADER | PROCESS} task_name system_params custom_params
   ```

2. 次の定義を使用して、スクリプトの構文において各変数で使用する値を決定します。

   - `rt_owner`: ランタイム・リポジトリの所有者です。
   - `location_name`: PL/SQL マッピングまたはプロセス・フローの場合、マッピングの配布に使用された場所の名前を指定します。SQL*Loader マッピングの場合、この値は PlatformSchema に設定する必要があります。この変数では大文字と小文字が区別されます。
   - {`PLSQL | SQL_LOADER | PROCESS`}: 次のいずれかを選択して、オブジェクト・タイプを定義します。
   - `task_name`: マッピングまたはプロセス・フローの名前です。
   - `system_params`: カンマで区切り二重引用符で囲みます。カンマと ¥ は、¥ でエスケープ処理できます。
   - `custom_params`: マッピング入力パラメータがある場合、ここで入力パラメータを定義する必要があります。`system_params` と同じ形式を使用します。

3. このスクリプトを使用して複数のマッピングを実行するには、この同じスクリプトを複数回実行する必要があります。

   たとえば次は、PL/SQL マッピング（マッピング名は *MY_MAPPING*）を *MY_WAREHOUSE* の場所から、ランタイム所有者（*MY_RUNTIME*）で、システム・パラメータやカスタム・パラメータを指定しないで実行するために使用します。

   ```
   @sqlplus_exec_template.sql MY_RUNTIME MY_WAREHOUSE PLSQL MY_
   MAPPING "," ","
   ```

   次は、SQL*Loader マッピング（マッピング名は *MY_LOAD*）をランタイム所有者（*MY_RUNTIME*）で、システム・パラメータやカスタム・パラメータを指定しないで実行するために使用します。配布中に使用した場所として location_name を定義しないことに注意してください。`PlatformSchema` 変数では大文字と小文字を区別する必要があります。

   ```
   @sqlplus_exec_template.sql MY_RUNTIME PlatformSchema SQL_LOADER
   MY_LOAD "," ","
   ```

マッピングとプロセス・フローのスケジューリング

Warehouse Builder からマッピングとプロセス・フローを実行するだけでなく、Warehouse Builder を使用して Oracle Enterprise Manager（OEM）内にジョブを作成し、これらが特定の時刻に実行されるようにスケジュールすることもできます。

スケジューリングを開始する前に、次の事項を確認してください。

- Oracle Enterprise Manager（OEM）リポジトリが作成されていること。
- ノードとデータベースの両方に、優先接続情報が設定されていること。
- バッチ・ジョブを実行するオペレーティング・システム権限を持つ OEM ユーザーとして、ログオンしていること。
- Runtime Platform Service がインストールされているノードで、OEM Intelligent Agent が稼動していること。
- Oracle Management Server（OMS）が稼動していること。
- スケジューリング対象のマッピングとプロセス・フローが配布されていること。

OEM でマッピングまたはプロセス・フローをスケジュールする手順は次のとおりです。

1. Warehouse Builder を起動して、Runtime Repository 接続の詳細を確認します。Runtime Repository ユーザーとして接続し、Runtime Repository の所有者としてログオンする必要があります。
2. デプロイメント・マネージャを起動して、スケジューリング対象のオブジェクトのロケーション、名前および配布ステータスを確認します。

   配布ステータスは「成功しました」になっている必要があります。

3. Oracle Enterprise Manager Console を起動し、スーパーユーザーとして Oracle Management Server にログインします。デフォルトのスーパーユーザーは、Sysman です。
4. 「ジョブ」メニューから「ジョブの作成」を選択するか、[Ctrl] を押しながら [J] を押します。

   「ジョブの作成」ウィンドウが開き、5 つのタブが表示されます。
5. 「一般」タブでジョブの名前を指定し、ターゲット・タイプを 1 つ選択し、使用可能なターゲットの中から特定のターゲットを 1 つ選択します。

   「追加」ボタンを使用して、選択したターゲットを「選択済みターゲット」列に移動します。
6. 「タスク」タブを選択し、「使用可能タスク」リストから「SQL*Plus スクリプトの実行」を選択します。「追加」ボタンを使用して、選択したタスクを移動します。
7. 「パラメータ」タブを選択し、パラメータを指定します。

   Warehouse Builder に付属のスクリプト `oem_exec_template.sql` をインポートするか、スクリプトの内容をコピーおよび貼り付けることができます。
8. Oracle Warehouse Builder に付属のスクリプト oem_exec_template.sql をインポートします。

   このスクリプトは `<OWB home>¥owb¥rtp¥sql` にあります。
9. このフォルダに移動し、スクリプトを開きます。スクリプトは、「スクリプト・テキスト」フィールドに表示されます。スクリプトの必須パラメータは次のとおりです。
   - ランタイム・リポジトリの所有者
   - ターゲット・ロケーション
   - ターゲット・タイプ : PL/SQL、SQL_LOADER または PROCESS のいずれか
   - タスク名（マッピングまたはプロセス・フローの名前）
   - システム・パラメータ（カンマで区切られ、二重引用符で囲まれています。カンマと ¥ は ¥ でエスケープできます）
   - カスタム・パラメータ（システム・パラメータを参照）

   スクリプトは、Runtime Access User のもとで実行する必要があります。これらは、優先接続情報として設定できます。適切な接続情報を持っていることを確認してください。
10. ジョブを直接発行するか、今後スケジュールするものとしてジョブ・ライブラリに追加します。あるいは、これらを両方実行できます。

14

配布と実行の監査

Warehouse Builder を使用してスクリプトを配布および実行する際、各配布および実行に関するデータは、Runtime Repository に格納されます。クライアントまたは Oracle Portal で、Runtime Audit Browser を使用してレポートを表示すると、このデータにアクセスできます。

この章では、次のトピックについて説明します。

配布と実行を監査する理由

ターゲットがどのようにロードされているか、マッピングとプロセス・フローの設定およびパラメータをさらに最適化するには何をすればいいのかを考慮する場合に、配布と実行の監査情報から貴重な詳細が得られることがあります。また、配布情報に関するフィードバックが即座に得られるため、オブジェクトの配布履歴を確認できます。

これらのレポートによって、ETL ランタイム情報の概要と詳細の両方にアクセスできます。この情報には、次の内容が含まれます。

- マッピングおよびプロセス・フローごとの時間
- プロセス・フローごとのアクティビティ詳細
- エラーの詳細
- 個別のターゲット環境を管理するための配布情報

Runtime Audit Browser で使用できるこれらのレポートは、Warehouse Builder Public View を使用して作成されており、Runtime Repository 内に格納されているすべてのデータにアクセスできます。このデータを表示するために、Runtime Audit Browser の事前に作成されたレポート機能を使用できます。また、Public View を使用して独自のカスタム・レポートを作成することもできます。Public View の詳細は、付録 D「Warehouse Builder Public View」を参照してください。

Runtime Audit Browser について

Warehouse Builder Runtime Audit Browser は、過去の配布および実行に関する詳細情報を表示できるブラウザ・ベースのレポート・ツールです。レポートは、Runtime Repository に格納されたデータから生成されます。

Warehouse Builder では、次の 2 つのバージョンのレポート・ツールが提供されます。

- クライアント・ブラウザ・バージョン
- Oracle Portal バージョン

両方のツールで提供されるレポートと機能は同じです。唯一の相違点は、クライアント・ブラウザ・バージョンは Warehouse Builder Design Client とともにインストールされ、クライアントと同じメニューから起動できるという点です。Oracle Portal バージョンを使用する場合、クライアントをインストールする必要はありませんが、Oracle Portal を実行するアプリケーション・サーバー上に Runtime Audit Browser がインストールされている必要があります。

Runtime Audit Browser のセットアップおよびバージョン選択の詳細は、『Oracle Warehouse Builder 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。

クライアント・ブラウザ・バージョンの起動

クライアント・ブラウザ・バージョンを起動する手順は次のとおりです。

1. 「スタート」→「プログラム」→「Oracle」→「Warehouse Builder」→「Start OWB OC4J Instance」を選択します。

   OC4J インスタンスが初期化されます。

2. 「スタート」→「プログラム」→「Oracle」→「Warehouse Builder」→「OWB Runtime Audit Browser」を選択します。

   Warehouse Builder Browser が起動し、図 14-1 に示すように、Runtime Repository の接続情報ページが表示されます。エラー・メッセージが表示される場合は、Warehouse Builder Browser ウィンドウが正しく構成されていない可能性があります。Warehouse Builder Runtime Audit Browser の構成手順は、『Oracle Warehouse Builder 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。

図 14-1 クライアント・ブラウザ・ランタイム接続

3. ドロップダウン・リストからロールを選択します。

   詳細は、14-5 ページの「ロールの選択」を参照してください。

4. Runtime Repository の接続情報を指定し、「配布レポートの表示」または「実行レポートの表示」をクリックします。

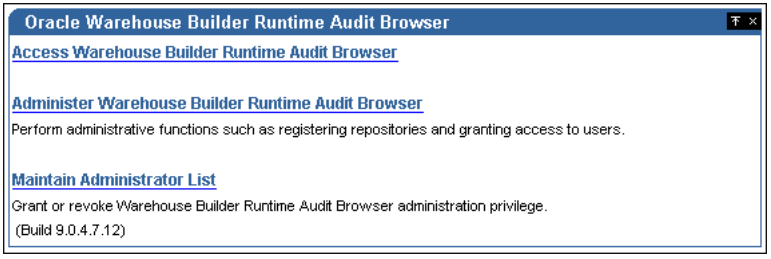
   接続情報が確認され、Runtime Repository レポートが表示されます。リポジトリに接続した後の Runtime Audit Browser の使用方法の詳細は、14-6 ページの「Runtime Repository レポートの表示」を参照してください。

Oracle Portal バージョンの起動

Oracle9*i*AS Portal バージョンを起動する手順は次のとおりです。

1. ブラウザを起動し、Oracle9*i*AS Portal に接続し、Runtime Audit Browser ポートレットへのアクセス権を持つユーザーとしてログインします。
2. 図 14-2 に示すように、Runtime Audit Browser ポートレットが含まれるページへ移動します。

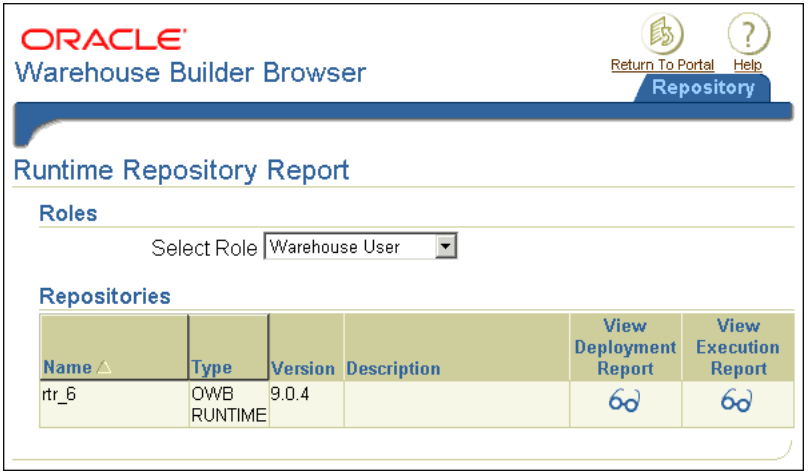
図 14-2 Runtime Audit Browser ポートレット



3. 「Warehouse Builder Runtime Audit Browser にアクセスします」リンクを選択します。

   図 14-3 に示すように、「Runtime Repository レポート」ページに、アクセス可能なリポジトリの一覧が表示されます。

図 14-3「ランタイム・リポジトリ・レポート」ページ



4. ドロップダウン・リストからロールを選択します。

   詳細は、14-5 ページの「ロールの選択」を参照してください。

5. アクセスする Runtime Repository を見つけ、「配布レポートの表示」列または「実行レポートの表示」列にあるアイコンをクリックします。

   選択したレポートは、次の画面に表示されます。リポジトリに接続した後の Runtime Audit Browser の使用方法の詳細は、14-6 ページの「Runtime Repository レポートの表示」を参照してください。

ロールの選択

Runtime Audit Browser にログオンするときには、ロールを選択する必要があります。選択するロールによって、Runtime Audit Browser で使用できる情報および機能が異なります。

次の 3 つのロールからいずれかを選択できます。

- QA ユーザー
- ウェアハウス・ユーザー
- ウェアハウス・エンジニア

すべてのロールは、Runtime Audit Browser レポートを表示するためのアクセス権を持ちますが、ロールによって、特定のレポート内のデータと使用できる機能が異なります。表 14-1 では、各ロールおよび主要な相違点を説明します。

表 14-1 ロールの説明

ロール	説明
QA ユーザー・ロール	このロールを選択すると、「削除」ボタンが使用可能になり、実行エラー診断レポートからエラー診断データのソース値を表示できます。
ウェアハウス・ユーザー・ロールまたはウェアハウス・エンジニア・ロール	これらのロールのいずれかを選択すると、「削除」ボタンが無効になり、実行エラー診断レポートから使用できない情報としてソース値が表示されます。

Runtime Repository レポートの表示

Runtime Audit Browser を使用して、Runtime Repository に格納されている配布および実行監査情報を表示します。クライアント・バージョンまたは Oracle Portal バージョンのいずれかから Runtime Repository に接続した後、使用可能なリポジトリから配布レポートまたは実行レポートを選択します。

配布レポートの表示

配布レポートには、特定の配布に関する監査データが含まれています。最初にポートレットから Runtime Repository にアクセスすると、Runtime Repository 全体に関する広範なレポートが表示されます。その後、ドリルダウンして、特定の情報を含むレポートを扱うことができます。

次のタブ付きセクションを使用して、詳細を表示します。

- 配布スケジュール
- オブジェクト・サマリー
- ロケーション

配布スケジュール

「配布スケジュール」タブ付きセクションでは、配布の時刻に基づいて配布データを検索できます。「フィルタ・オプション」セクションを使用して、特定のデータ範囲内で発生した配布をフィルタリングします。その後、特定の配布にドリルダウンして、オブジェクト情報を表示できます。「フォーカス」列にあるアイコンを使用して、1 つの行の配布詳細のみを表示することもできます。

オブジェクト・サマリー

「オブジェクト・サマリー」タブ付きセクションでは、オブジェクトのタイプに基づいて配布データを表示できます。Runtime Repository 全体について最初にこのタブを選択すると、すべてのオブジェクトの一覧が表示されます。「フィルタ・オプション」セクションを使用して、オブジェクト・タイプと配布ステータスに基づいてフィルタリングします。

次の情報が表示されます。

- 名前
- タイプ
- ロケーション
- 最新の配布
- オブジェクト・ステータス

特定のオブジェクトまたはロケーションにドリルダウンして、さらに詳細を表示できます。

ロケーション

「ロケーション」タブ付きセクションでは、ロケーションに基づいて配布データを表示できます。Runtime Repository 全体について最初にこのタブを選択すると、すべての登録済ロケーションの一覧が表示されます。

次の情報が表示されます。

- 名前
- タイプ
- タイプ・バージョン
- 最新の配布

特定のロケーションにドリルダウンして、さらに詳細を表示できます。

実行レポートの表示

実行レポートには、特定の実行に関する監査データが含まれています。最初にポートレットから Runtime Repository にアクセスすると、Runtime Repository 全体に関する広範なレポートが表示されます。その後、ドリルダウンして、特定の情報を含むレポートを扱うことができます。

次のタブ付きセクションを使用して、詳細を表示します。

- 実行スケジュール
- 実行サマリー
- 実行
- トレース

実行スケジュール

「実行スケジュール」タブ付きセクションでは、実行の時刻に基づいて実行データを検索できます。「フィルタ・オプション」セクションを使用して、特定のデータ範囲内で発生した実行をフィルタリングします。特定のマッピングまたはプロセス・フローをフィルタリングすることもできます。

その後、特定の実行にドリルダウンして、情報を表示できます。「フォーカス」列にあるアイコンを使用して、1 つの行の詳細のみを表示することもできます。

実行サマリー

「実行サマリー」タブ付きセクションでは、マッピングまたはプロセス・フローに基づいて実行データを表示できます。Runtime Repository 全体について最初にこのタブを選択すると、すべてのオブジェクトの一覧が表示されます。「フィルタ・オプション」セクションを使用して、オブジェクトと実行ステータスに基づいてフィルタリングします。

次の情報が表示されます。

- 名前
- タイプ
- 最新の実行
- 実行ステータス
- 「実行レポート」アイコン

実行

「実行レポート」アイコンをクリックすると、実行レポートの「実行」タブ付きセクションが表示されます。マッピングまたはプロセス・フローの各実行について、1 つの実行レポートが提供されます。

このレポートの「実行」タブ付きセクションを使用して、次の情報を表示します。

- 実行の詳細
- 実行パラメータ
- ステップの詳細
- エラー・メッセージ
- 監査の詳細

トレース

実行レポートの「トレース」タブ付きセクションでは、マッピングまたはプロセス・フローの特定の実行に関する詳細なトレース情報が表示されます。

次の情報が表示されます。

- 行キー
- 重大度
- ソース / ターゲット
- 表名
- アクション
- 診断レポートの表示

監査データの削除

Runtime Repository からデータを削除することもあります。各レポート・ページ上の「削除」ボタンを使用して、Runtime Repository からデータを削除します。この便利な機能によって、監査表で保持されるデータの量が制限され、関連データの検索が容易になり、パフォーマンスが維持されます。「削除」ボタンを使用するには、QA ユーザーとしてログオンしている必要があります。ウェアハウス・ユーザーまたはウェアハウス・エンジニアとしてログオンしている場合、「削除」ボタンは無効になります。

注意： 「監査レベル」構成パラメータが「完了」に設定されたマッピングを実行している場合、多量の診断トレース・データが生成されます。これによって、表領域が一杯になります。この場合は、「エラー行およびトレース行の削除」ボタンを使用します。

Runtime Audit Browser の管理

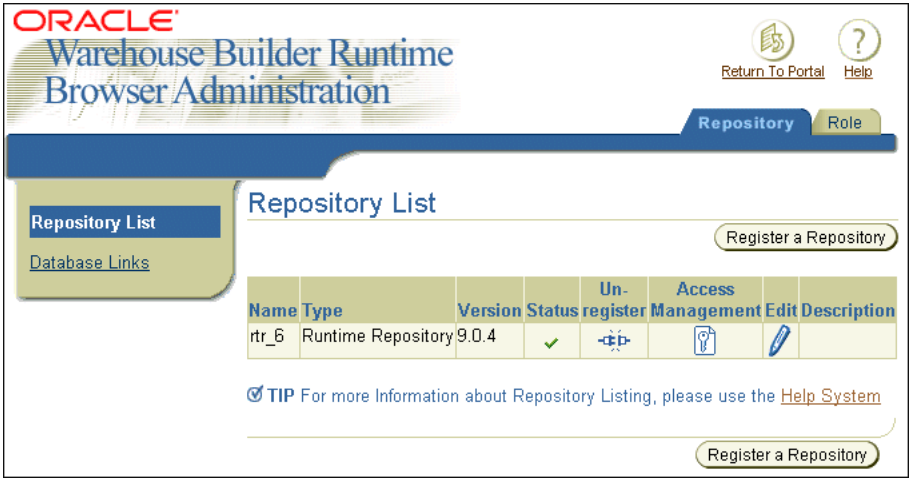
Runtime Audit Browser の管理機能は、Oracle Portal バージョンでのみ使用できます。Runtime Audit Browser ポートレットの「Warehouse Builder Runtime Audit Browser の管理」リンクおよび「管理者リストの保持」リンクから、次の機能にアクセスします。

- 使用可能なリポジトリの管理 : 使用可能なリポジトリの接続情報を表示したり、新しいリポジトリを登録したりできます。
- ロールの管理 : ユーザー・ロールとアクセス権を定義できます。
- 管理者一覧の管理 : 管理者レベルの権限をどのユーザーに付与するかを定義できます。

使用可能なリポジトリの管理

図 14-4 に示すように、「リポジトリ・リスト」ページを使用して、すべての登録済 Runtime Repository の一覧を表示します。監査レポートにアクセスするには、すべてのリポジトリを登録している必要があります。

図 14-4「リポジトリ・リスト」ページ



リポジトリを登録解除するには、次を実行します。

- 登録解除するリポジトリの名前を見つけ、「登録解除」列にあるアイコンをクリックします。

  そのリポジトリは即座に登録解除され、ページがリフレッシュされて更新された一覧が表示されます。

リポジトリへのアクセス権を管理するには、次を実行します。

- リポジトリを見つけ、「アクセス管理」列にあるアイコンをクリックします。

  次のページに「アクセス管理」が表示されます。これを使用すると、ユーザーまたはグループに対して、リポジトリへのアクセス権を付与または取り消しできます。ヘルプおよび手順もこのページで提供されます。

リポジトリ情報を編集するには、次を実行します。

- リポジトリを見つけ、「編集」列にあるアイコンをクリックします。

  「ページの編集」ページが表示されます。ヘルプおよび手順もこのページで提供されます。

リポジトリを登録するには、次を実行します。

- 「リポジトリの登録」をクリックします。

  「リポジトリの登録」ページが表示されます。リポジトリの接続詳細を指定できるように準備を整えておきます。ヘルプおよび手順もこのページで提供されます。

注意： リポジトリを登録する前に、データベース・リンクを定義しておく必要があります。

データベース・リンクを追加または表示するには、次を実行します。

- 左側の列にある「データベース・リンク」リンクをクリックします。

  「データベース・リンク」ページが表示されます。ヘルプおよび手順もこのページで提供されます。

リポジトリの登録

「リポジトリの登録」ページを使用して、Runtime Repository を登録します。登録済 Runtime Repository でのみ監査データを表示できます。

リポジトリを登録する手順は次のとおりです。

1. Runtime Repository の名前を入力します。
2. データベース・リンクを指定するか、懐中電灯またはトーチのアイコンをクリックして、すでに作成されたデータベース・リンクの一覧からデータベース・リンクを選択します。
3. 説明（オプション）を入力します。
4. 「適用」をクリックして、リポジトリを登録します。

   「リポジトリ・リスト」ページに、登録済リポジトリの更新された一覧が表示されます。

リポジトリ定義の編集

「ページの編集」ページを使用して、リポジトリの登録情報を編集します。編集できるのは、一度に 1 つのリポジトリのみです。

次のフィールドを編集できます。

- 名前
- データベース・リンク
- 説明

「適用」をクリックして、編集内容を保存します。「リポジトリ・リスト」ページが表示されます。

データベース・リンクの管理

「データベース・リンク」ページを使用して、データベース・リンクの表示、作成、編集および削除を行います。

データベース・リンクを作成するには、次を実行します。

- データベース・リンク一覧の下にある「データベース・リンクの作成」をクリックします。

  「データベース・リンクの作成」ページが表示されます。詳細な接続情報を指定できるように準備を整えておきます。ヘルプおよび手順もこのページで提供されます。

データベース・リンクを削除するには、次を実行します。

- 削除するデータベース・リンクの名前を見つけ、「削除」列にあるアイコンをクリックします。

  そのデータベース・リンクは即座に削除され、ページがリフレッシュされて更新された一覧が表示されます。

データベース・リンクを編集するには、次を実行します。

- データベース・リンクを見つけ、「編集」列にあるアイコンをクリックします。

  「データベース・リンクの編集」ページが表示されます。ヘルプおよび手順もこのページで提供されます。

データベース・リンクの作成

「データベース・リンクの作成」ページを使用して、作成する新しいデータベース・リンクの接続情報を指定します。

データベース・リンクを作成するには、次を指定します。

- データベース・リンク名
- Warehouse Builder Design Repository のユーザー名
- Warehouse Builder Design Repository のユーザー・パスワード
- ホスト・アドレス
- ホスト・サービス名
- ホスト・プロトコル
- ホスト・ポート番号

「適用」をクリックして、データベース・リンク情報を発行します。情報が有効である場合、「データベース・リンク」ページに、更新されたデータベース・リンクの一覧が表示されます。

データベース・リンクの編集

「データベース・リンクの編集」ページを使用して、特定のデータベース・リンクに関する詳細を表示したり、変更したりします。

次のフィールドを編集できます。

- Warehouse Builder Design Repository のユーザー名
- Warehouse Builder Design Repository のユーザー・パスワード
- リモート・データベース情報

「適用」をクリックして、編集内容を保存します。「データベース・リンク」ページが表示されます。

「データベース・リンク・エラー・ステータス」ページの表示

「データベース・リンク・エラー・ステータス」ページを使用して、エラーに関する詳細を表示します。このページは表示のみに使用します。

次の情報が表示されます。

- **名前 :** データベース・リンクの名前
- **作成 :** データベース・リンクが作成されたときのタイムスタンプ
- **ステータス :** エラー
- **説明 :** データベース・リンクが有効でない理由

ロールの管理

「ロールの管理」ページを使用して、ロール・タイプごとにアクセス権を定義します。ロール・タイプによって、レポートでの監査情報の表示方法が決まります。ロールの詳細は、14-5 ページの「ロールの選択」を参照してください。

次の 3 つのロール・タイプがあります。

- QA ユーザー
- ウェアハウス・ユーザー
- ウェアハウス・エンジニア

ロール・タイプへのアクセス権を管理するには、次を実行します。

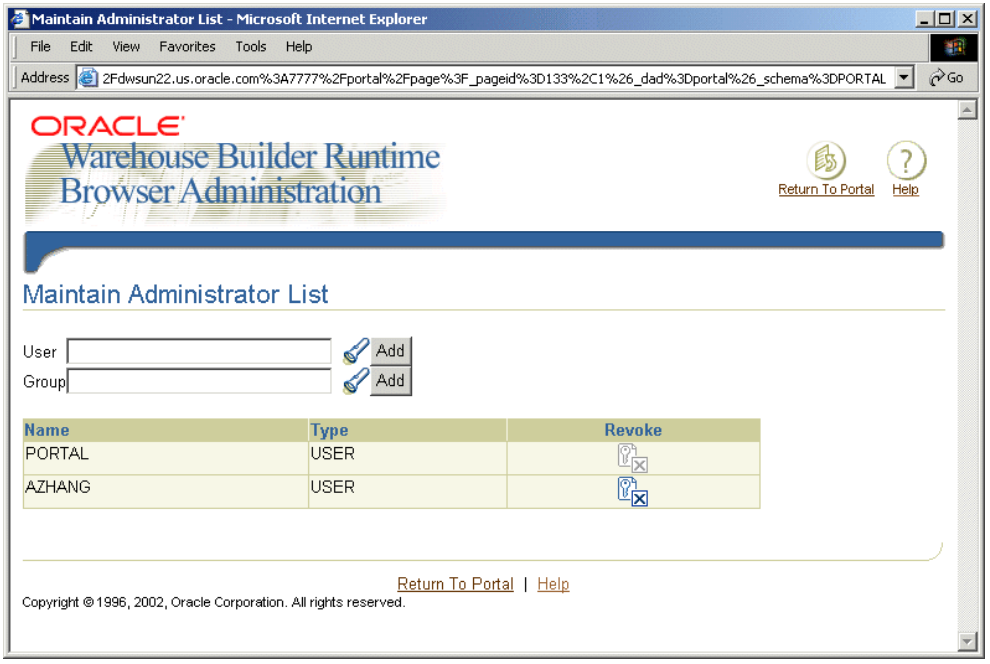
- ロール・タイプを選択し、「アクセス管理」列にあるアイコンを選択します。

  「アクセス管理」ページが表示されます。ここで、どのユーザーおよびグループに対してアクセス権を付与または取り消すかを定義できます。

管理者一覧の管理

図 14-5 に示すように、「管理者リストの保持」ページを使用して、ユーザーおよびグループに対して管理者権限を追加または取り消します。

図 14-5「管理者リストの保持」



注意： 管理者一覧に変更を加えるには、管理者権限を持っている必要があります。

ユーザーまたはグループを追加するには、次を実行します。

- 空白のフィールドにユーザーまたはグループ名を入力し、「追加」をクリックします。懐中電灯またはトーチのアイコンを選択して、ユーザーまたはグループを検索することもできます。

  入力したユーザー名またはグループ名が有効な場合、管理者一覧に追加されます。

ユーザーまたはグループの権限を取り消すには、次を実行します。

- ユーザー名またはグループ名を見つけて、「再起動」列にあるアイコンをクリックします。

  そのユーザーまたはグループが管理者一覧から削除されます。

使用可能なランタイム監査レポート

次の各表は、Runtime Audit Browser で使用可能な監査レポートを示します。

- 表 14-2「ウェアハウス / リポジトリ・レポート」(14-17 ページ)
- 表 14-3「プロセス・レポート」(14-17 ページ)
- 表 14-4「プロセス実行レポート」(14-18 ページ)
- 表 14-5「マッピング・レポート」(14-18 ページ)
- 表 14-6「マッピング実行レポート」(14-18 ページ)
- 表 14-7「実行エラー・レポート」(14-18 ページ)
- 表 14-8「データ・オブジェクト・レポート」(14-19 ページ)
- 表 14-9「ロケーション・レポート」(14-19 ページ)
- 表 14-10「配布エラー・レポート」(14-19 ページ)
- 表 14-11「管理レポート」(14-19 ページ)

ウェアハウス / リポジトリ・レポート

表 14-2 ウェアハウス / リポジトリ・レポート

レポート	説明
配布スケジュール	選択したリポジトリで発生したすべての配布アクションのサマリーが表示されます。これは最初に、現在の日付に基づくカレンダー・ビューとして表示されます。このレポートには、配布済オブジェクトへのハイパーリンクが含まれます。
実行スケジュール	すべての実行プロセス（日付 / 時刻の順）と、それらのプロセスに含まれているタスクのサマリーが表示されます。これは最初に、現在の日付に基づくカレンダー・ビューとして表示されます。このレポートには、プロセス、マッピング、プロセス実行およびマッピング実行へのハイパーリンクが含まれます。
実行サマリー	すべてのプロセスのサマリー詳細（アルファベット順または日付 / 時刻の順）が表示されます。プロセス実行およびマッピング実行の詳細が含まれます。このレポートには、プロセス、マッピング、プロセス実行およびマッピング実行へのハイパーリンクが含まれます。
オブジェクト・サマリー	選択したリポジトリに配布されたすべてのオブジェクトのサマリーが表示されます。各オブジェクトについて最後の配布の詳細が表示されます。このレポートには、配布済のプロセス、マッピングおよびデータ・オブジェクトへのハイパーリンクが含まれます。
ロケーション	オブジェクトの配布先であるすべてのロケーションの一覧が表示されます。このレポートには、様々なロケーションへのハイパーリンクが含まれます。

プロセス・レポート

表 14-3 プロセス・レポート

レポート	説明
プロセスの実行	プロセス実行のサマリー詳細が表示されます。このレポートには、プロセス実行レポートへのハイパーリンクが含まれます。
プロセスの配布	配布履歴の詳細が表示されます。このレポートには、プロセスに含まれるサブプロセスとマッピングへのハイパーリンクと、発生した配布エラーへのハイパーリンクが含まれます。

プロセス実行レポート

表 14-4 プロセス実行レポート

レポート	説明
プロセスの実行	実行詳細、リターン・ステータス、ランタイム・パラメータ、プロセスに含まれるマッピング実行のサマリーが表示されます。このレポートには、プロセス・レポート、マッピング実行レポート、実行エラー・レポートへのハイパーリンクが含まれます。

マッピング・レポート

表 14-5 マッピング・レポート

レポート	説明
マップの実行	マッピング実行のサマリー詳細が表示されます。このレポートには、マッピング実行レポートへのハイパーリンクが含まれます。
マップの配布	マッピングの配布履歴の詳細が表示されます。このレポートには、発生した配布エラーへのハイパーリンクが含まれます。

マッピング実行レポート

表 14-6 マッピング実行レポート

レポート	説明
マップの実行	実行詳細、リターン・ステータスおよびランタイム・パラメータが表示されます。このレポートには、実行エラーへのハイパーリンクが含まれます。
マップの実行トレース	使用されたソース・データ値とターゲット・データ値の診断トレースが表示されます。これは診断に役立ちます。これは、「監査レベル」パラメータが適切な値に設定されたマッピングを実行した場合にのみ、使用可能になります。

実行エラー・レポート

表 14-7 実行エラー・レポート

レポート	説明
エラー診断レポート	マッピングまたはプロセス・フローの実行中に発生した実行エラー・メッセージの詳細が表示されます。

データ・オブジェクト・レポート

表 14-8 データ・オブジェクト・レポート

レポート	説明
データ・オブジェクトの配布	特定のオブジェクトの配布履歴の詳細が表示されます。このレポートには、発生した配布エラーへのハイパーリンクが含まれます。

ロケーション・レポート

表 14-9 ロケーション・レポート

レポート	説明
配布スケジュール	選択したロケーションで発生したすべての配布アクションのサマリーが表示されます。これは最初に、現在の日付に基づくカレンダー・ビューとして表示されます。このレポートには、配布済オブジェクトへのハイパーリンクが含まれます。
オブジェクト・サマリー	選択したロケーションに配布されたすべてのオブジェクトのサマリーが表示されます。各オブジェクトについて最後の配布の詳細が表示されます。このレポートには、配布済のプロセス、マッピングおよびデータ・オブジェクトへのハイパーリンクが含まれます。

配布エラー・レポート

表 14-10 配布エラー・レポート

レポート	説明
エラーの詳細	特定の配布エラーについて、配布中に発生したすべての関連エラー、警告および情報メッセージの詳細が表示されます。

管理レポート

表 14-11 管理レポート

レポート	説明
サービス・ノード・レポート	Runtime Repository のクラスタ・ノードの現在のステータスが表示されます。このレポートでは、ノードが使用中であるかどうかが示されます。

第 IV 部

メタデータの管理

この部には、次の章があります。

- 第 15 章「Metadata Loader（MDL）を使用したインポートおよびエクスポート」
- 第 16 章「メタデータ変更管理」
- 第 17 章「メタデータの参照およびレポート」
- 第 18 章「Warehouse Builder Browser の管理」

15

Metadata Loader（MDL）を使用したインポートおよびエクスポート

Metadata Loader（MDL）を使用すると、既存のリポジトリ・メタデータのバックアップを転送、更新、リストアできるだけでなく、新規リポジトリへのデータの移入もできます。また、メタデータのスナップショットを作成し、バックアップ、比較、リストアの目的で使用することもできます。

この項では、次のトピックについて説明します。

Metadata Loader を使用したインポートおよびエクスポートの概要

Warehouse Builder では、バックアップ、履歴管理、バージョン管理の目的でメタデータをコピーおよび移動できる機能がいくつか提供されています。

Metadata Loader（MDL）ユーティリティを使用すると、ナビゲーション・ツリー上のどのタイプのオブジェクトが対象でも、メタデータのインポートとエクスポートができます。MDL には、Warehouse Builder Design Client と OMB Plus スクリプト・インタフェースのどちらからもアクセスできます。Warehouse Builder をアップグレードするときは、インポートおよびエクスポートの機能を使用して、メタデータのバックアップやメタデータの移行を行います。

エクスポートされたファイルを、サードパーティの管理ツール（Oracle Repository、ClearCase、SourceSafe など）に移動することもできます。プロジェクトのプロパティにバージョン番号を入力すると、この設定でエクスポートおよびインポートのバージョンをより簡単に追跡できます。

また、OMB Plus スクリプト・インタフェースを使用してメタデータのスナップショットを作成することにより、メタデータの変更管理を行うこともできます。スナップショットを利用すると、Warehouse Builder スクリプトを使用してメタデータ・オブジェクトの定義を取得できます。スナップショットは、メタデータのバックアップおよびバージョン管理に使用してください。メタデータの変更管理の詳細は、第 16 章「メタデータ変更管理」を参照してください。

Metadata Loader を使用したメタデータのインポートおよびエクスポート

MDL を使用すると、リポジトリが常駐するプラットフォームのオペレーティング・システムが異なる場合であっても、複数のリポジトリ間でメタデータ・オブジェクトをコピーまたは移動できます。

MDL は、メタデータ・エクスポートとメタデータ・インポートの 2 つのユーティリティで構成されています。エクスポート・ユーティリティを使用すると、リポジトリからメタデータ・オブジェクトを抽出して、情報をテキスト・ファイルに書き込むことができます。インポート・ユーティリティでは、エクスポートされたテキスト・ファイルのメタデータ情報が読み取られ、リポジトリへメタデータ・オブジェクトが挿入されます。MDL では独自の形式が使用され、MDL インポート・ユーティリティでは、MDL 形式（MDL エクスポートで作成されたファイル）のみが読み取られます。

MDL は、Warehouse Builder コンソールから、またはコマンドライン・インタフェースを使用して操作できます。コマンドライン・インタフェースで MDL を使用する方法については、15-28 ページの「Metadata Loader のコマンドライン・ユーティリティの使用方法」を参照してください。コンソール・メニューを使用する場合は、グラフィカル・インタフェースに従ってエクスポートまたはインポートの処理を実行できます。

Metadata Loader を使用して、次のいずれかのタスクを実行します。

- **メタデータのバックアップ :** MDL は、障害時リカバリ計画における重要な要素です。既存のリポジトリのメタデータを含むファイルをバックアップとしてエクスポートし、必要に応じてエクスポートしたファイルを使用してリポジトリをリストアできます。
- **新規リポジトリのシード :** 既存のリポジトリからデータをエクスポートして、新規リポジトリの基礎として使用できます。
- **リポジトリの移行 :** メタデータをファイルにエクスポートして、再インポートできます。この作業は通常、新しいバージョンの Warehouse Builder にアップグレードするときに行われます。前のバージョンの Warehouse Builder からのアップグレードについては、『Oracle Warehouse Builder 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。
- **メタデータのコピー :** ユーザーの開発環境が複数ある場合、同じメタデータが複数作成される可能性があります。MDL を使用すると、単一セットのメタデータを複数のリポジトリに簡単にロードできます。

この項では、次のトピックについて説明します。

- MDL に必要なアクセス権限（15-3 ページ）
- Metadata Loader の結果について（15-4 ページ）
- Metadata Loader のログ・ファイルについて（15-5 ページ）

MDL に必要なアクセス権限

Warehouse BuilderDesign Repository では、同じリポジトリ・スキーマに複数のクライアントが同時にアクセスできます。リポジトリ・オブジェクトを変更する場合、Warehouse Builder では、ロックを使用して、1 つのクライアントのみにアクセスが許可されます。オブジェクトがロックされている間は、そのオブジェクトを他のクライアントから表示しようとしても、ユーザーがコミットした最後のトランザクションを反映したオブジェクトしか表示されません。

ヒント : 最新のメタデータをエクスポートするには、他のクライアントがリポジトリにアクセスしていない状態にする必要があります。

表示されたプロンプトで「OK」をクリックすると、MDL では、メタデータのインポートが成功した後に（情報や警告のメッセージのみが表示される場合も含めて、インポートでエラー・メッセージが表示されない場合）、リポジトリに対する変更がコミットされます。また、MDL では、インポートが失敗した後にロールバックが実行されます。つまり、MDL ファイル内のオブジェクトに一致するリポジトリで、プライマリ・オブジェクトナビゲーション・ツリーの第 1 レベルにあるオブジェクト）ごとに、Warehouse Builder によってロックが適用されるということです。これらのオブジェクトには、プロジェクト、モジュール、表など（その他のオブジェクトも該当する）があります。個々の列はロックされません。したがって、メタデータをインポートする間は、これらのオブジェクトをロックする必要があります。インポート対象のオブジェクトを他のユーザーがロックしている場合、MDL は失敗します。

ヒント： メタデータのインポートが成功するには、他のクライアントがリポジトリにアクセスしていない状態にする必要があります。

リポジトリ内に、MDL のインポートによって影響を受けるオブジェクトが多すぎる場合、MDL は自動的にシングル・ユーザー・モードに切り替わります。つまり、MDL インポートが完了するまでは、他のユーザーはリポジトリにログオンできません。シングル・ユーザー・モードによって、多数のロックを使用する際の MDL のパフォーマンス低下を防ぐことができます。シングル・ユーザー・モードでは、リポジトリのエンキュー・リソースが消費されることが少なくなります。MDL をシングル・ユーザー・モードに切り替えようとしているときに、このリポジトリに他のユーザーがログインする場合、シングル・ユーザー・モードに切り替えることができず、失敗します。

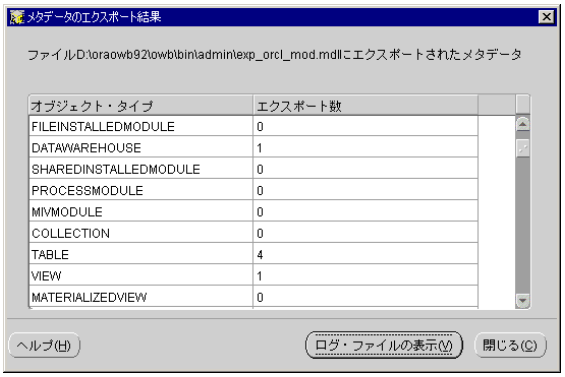
メタデータをインポートするには、MDL_IMPORT セキュリティ権限も必要です。セキュリティの詳細は、19-8 ページの「PL/SQL でのセキュリティ管理」を参照してください。

Metadata Loader の結果について

エクスポートまたはインポートのユーティリティを使用するたびに、MDL によって、アクションの結果がレポートされ、診断情報や統計情報がログ・ファイルに書き込まれます。

MDL では、インポートまたはエクスポートの後に、ダイアログを使用して結果がレポートされます。詳細情報が必要な場合は、図 15-1 に示す「メタデータのエクスポート結果」ダイアログ・ボックスで「ログ・ファイルの表示」をクリックすると、詳細なログを表示できます。

図 15-1 メタデータのエクスポート結果



結果ダイアログには、ファイルまたはリポジトリで見つかったメタデータ・オブジェクトと、エクスポートまたはインポートされた各オブジェクトの数が表示されます。結果ダイアログを使用すると、すべてのオブジェクトがエクスポートまたはインポートされたことを確認できます。MDL では、エクスポートまたはインポートされたオブジェクトが識別され、適切なオブジェクト・リストと比較されます。「エクスポート数」列または「インポート済み」列がゼロになっている場合、リポジトリ内でそのタイプのオブジェクトが見つからなかったことを示します。ただし、リポジトリまたはインポートされた MDL ファイルに存在するオブジェクトに関してゼロが表示された場合は、そのオブジェクトをインポートまたはエクスポートするときに、MDL で障害が発生しています。

Metadata Loader のログ・ファイルについて

リポジトリのメタデータをエクスポートまたはインポートするたびに、ログ・ファイルに診断情報や統計情報が書き込まれます。デフォルトでは、ログ・ファイルは、「作業環境」ダイアログの「メッセージ・ログ」タブで指定されているディレクトリおよびパスにあります。ログ・ファイルのロケーションは、MDL の起動時に別途指定できます。

例 15-1 は、典型的なインポート・ログ・ファイルの内容を示しています。

例 15-1 インポート結果を示すログ・ファイル

```
Import started at 04/25/2001 4:59:46 PM
********************************************************************************
*  Import for OWB Release: 3.0.0.0.0   Version: 3.0.0.3.0
*  User: user30_3i   Connect String: epaglina-pc:1521:ora8i
*  Data File: d:¥owb3000¥sco_dim_time_phy_m_tgt.mdl
*  Log File: d:¥owb3000¥imp_dim_time_phy_m_tgt.log
*  Trace: B
*  Trace File: d:¥owb3000¥imp_dim_time_phy_m_tgt.trc
*  Physical Names: Y   Mode: CREATE
*  Ignore Universal Identifier: Y   Commit At End: Y
```

```
********************************************************************************
Informational at line 15: MDL-1207 PROJECT with physical name <PRJ_Dimension> not imported
because it already exists.
Informational at line 21: MDL-1207 DATAWAREHOUSE with physical name <WH> not imported
because it already exists.
Informational: MDL-1134 COMMIT issued at end of import data file.

Counts for OWB Import Utility
-----------------------------

Total Projects Processed by Import = 1

-------------------------------

Project = PRJ_Dimension

Entity in Project                    Added     Replaced    Skipped
-------------------------------      -----     --------    -------
DATAWAREHOUSE:                         0           0          1
DIMENSION:                             1           0          0
LEVEL:                                 3           0          0
HIERARCHY:                             2           0          0
LEVELRELATIONSHIP:                     4           0          0
COLUMN:                               18           0          0
UNIQUEKEY:                             8           0          0
PRIMARYKEY:                            2           0          0
CONFIGPARAM:                          60           0         60
CHILDCONFIG:                          14           0          0

Import ended at 04/25/2001 4:59:52 PM
```

ログ・ファイルを使用すると、エクスポートとインポートのアクティビティの詳細な監視およびトラブルシューティングを行えます。ログ・ファイルには、次のタイプのステータス・メッセージが書き込まれます。

- **情報 :** メタデータ・オブジェクトが見つからない、オブジェクトがインポートされたかどうか、インポートまたはエクスポートされなかった理由など、インポートまたはエクスポートに関する情報を提供します。
- **警告 :** オブジェクトのインポートまたはエクスポートに関する警告ですが、ロードの失敗や中断に関する情報は含まれません。警告は、予期しないロード結果が得られる可能性があることを知らせるものです。
- **エラー :** MDL エクスポートまたはインポートが中断され、正常に実行できなかったことを示します。エラー・メッセージでは、失敗の理由が簡単に説明されます。

このログには、追加、置換およびスキップされたオブジェクトの合計数も表示されます。あるオブジェクトでいずれかの列がゼロになっている場合、リポジトリ内またはインポートされた MDL ファイル内でそのタイプのオブジェクトが見つからなかったことを示します。

詳細エラー・ログ

MDL インポートの実行中にエラーが発生した場合は、エラー・メッセージが表示されます。

「詳細」をクリックすると、エラーが発生したリポジトリ・オブジェクトとオブジェクト行を示す詳細なエラー・ログが表示されます。詳細なメッセージでは、メタデータ・オブジェクトの定義が不適切、オブジェクトが重複しているなどの問題が通知されるため、既存のメタデータを含むリポジトリにメタデータをインポートする場合に便利です。

図 15-2 は、詳細エラー・メッセージの例です。

図 15-2 詳細エラー・メッセージ

この例では、リポジトリ・オブジェクトが CUST ディメンション、インポートされたオブジェクトが TOTAL レベルです。このダイアログは、TOTAL というレベルがすでに存在するため、TOTAL レベルを CUST ディメンションにインポートできないことを説明しています。

メタデータのエクスポート

Metadata Loader では、すべてのリポジトリ・オブジェクトをエクスポートできます。また、表の列やその制約、データ・ロード構成パラメータ、名前付き属性セットなど、メタデータ・オブジェクトに属する情報もエクスポートできます。MDL を使用すると、プロジェクト全体またはプロジェクト内のオブジェクトのサブセットをエクスポートできます。

リポジトリ・メタデータをエクスポートすると、Metadata Loader によって、抽出されたメタデータがデリミタ付きテキスト・ファイルに書き込まれます。エクスポートされた MDL ファイルにデフォルトのパスとファイル名を割り当てることにより、このファイルはリポジトリの外に格納されます。

この項では、次のトピックについて説明します。

メタデータをエクスポートする前に

メタデータをエクスポートする前に、次の条件を満たしていることを確認します。

- **必要なアクセス権限。**最新のメタデータをエクスポートするには、リポジトリに読込み / 書込みモードでアクセスしているクライアントが他にないことを確認します。詳細は、15-3 ページの「MDL に必要なアクセス権限」を参照してください。
- **十分なディスク記憶域。**メタデータのエクスポート先であるマシンに十分なディスク記憶域がない場合、エクスポートは失敗します。エクスポート先のマシンには、メタデータ・ファイル全体を格納できる記憶域が必要です。エクスポート・ユーティリティでは、メタデータ・ファイルを部分的に保存することはできません。

メタデータ・エクスポート・ユーティリティについて

メタデータを Warehouse Builder Design Repository からエクスポートするには、次のいずれかを使用します。

- **Metadata Loader のコマンドライン・ユーティリティ。**コマンドライン・ユーティリティを使用すると、クライアント・インタフェースでは実行できない追加作業を行えます。コマンドラインでエクスポートする方法については、15-28 ページの「Metadata Loader のコマンドライン・ユーティリティの使用方法」を参照してください。
- **Warehouse Builder Design Client インタフェース。**クライアント・インタフェースの使用方法については、15-8 ページの「メタデータのエクスポート」を参照してください。

コマンドラインまたはクライアント・インタフェースを使用して、プロジェクト全体、コレクション、モジュールまたはオブジェクトのサブセットをエクスポートできます。オブジェクトのサブセットをエクスポートする場合は、選択した各オブジェクトの定義、およびそのサブセットが属する親オブジェクトの定義がエクスポートされます。これによって、MDL では、メタデータのインポート中にオブジェクトのツリー・リレーションシップを維持できます。

たとえば、1 つのディメンションをエクスポートした場合、エクスポート・ファイルには次のオブジェクトの定義が含まれます。

- ディメンション（およびその階層とレベル）
- ディメンションが属するディメンション・ノード
- ディメンション・ノードが属するモジュール
- モジュールが属するプロジェクト

オブジェクトのサブセットをエクスポートする場合は、参照先のすべてのオブジェクトをエクスポートしてください。インポートする場合も同様です。メタデータ・インポート・ユーティリティでは、オブジェクトの参照が不完全な場合でもリポジトリ・オブジェクトをインポートできます。

たとえば、キューブをエクスポートする場合、外部キー参照はエクスポートされますが、参照先であるディメンションはエクスポートされません。ディメンション表のメタデータが変更されると、新規のリポジトリにインポートされた外部キー参照は不適切になります。

Warehouse Builder Design Client を使用したメタデータのエクスポート

メタデータ・エクスポート・ユーティリティを使用すると、Warehouse Builder Design Repository から MDL ファイルへオブジェクトをエクスポートできます。

Warehouse Builder Design Client インタフェースを使用してメタデータをリポジトリからエクスポートする手順は次のとおりです。

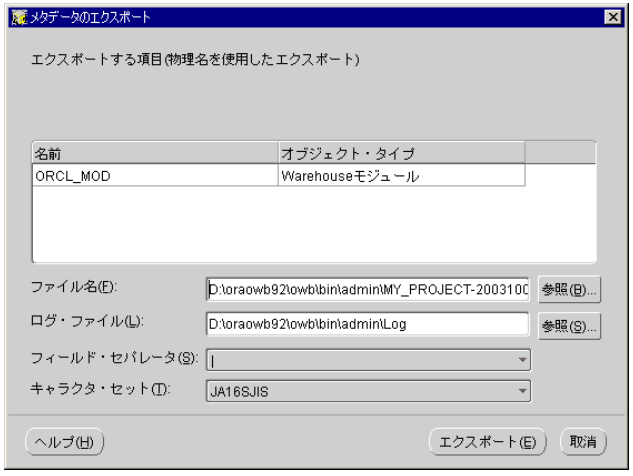
1. Warehouse Builder コンソールから、エクスポートするオブジェクトを選択します。

   表やオブジェクトのグループなど、個々のオブジェクトをエクスポートできます。プロジェクト、ノードまたはモジュールをエクスポートするときは、その中に含まれるオブジェクトもエクスポートします。コレクションをエクスポートするときは、その参照先であるオブジェクトもエクスポートします。

2. 「プロジェクト」メニューから「メタデータのエクスポート」を選択し、「ファイル」を選択します。

   「メタデータのエクスポート」ダイアログに、エクスポートするオブジェクトの名前とタイプが表示されます。「メタデータのエクスポート」ダイアログでは、図 15-3 に示すように、エクスポート・ファイルのデフォルト設定も表示されます。

図 15-3「メタデータのエクスポート」ダイアログ



3. 次のデフォルト設定をそのまま使用するか、または変更します。

   ファイル名 : 作成するエクスポート・ファイルの名前を入力するか、「参照」をクリックしてディレクトリまたはファイルを選択します。割り当てるファイル名の末尾は必ず **.mdl** とします。

   ログ・ファイル : Warehouse Builder では、エクスポートに関する情報がログ・ファイルに記録されます。新しいパスとファイル名を入力して、ファイルのロケーションを変更できます。フィールドにパスとファイル名を入力するか、「参照」をクリックしてディレクトリまたはファイル名を選択します。

   フィールド・セパレータ : エクスポート・ファイル内の表のフィールドは、デフォルトではパイプ（|）で区切られます。ファイルの中に、パイプ（|）記号がデータの一部として含まれている場合は、デフォルトのフィールド・セパレータを、リストから選択してカレット（^）に変更できます。

   キャラクタ・セット : エクスポート・ファイルで使用するキャラクタ・セットを選択します。デフォルトのキャラクタ・セットは、Warehouse Builder Design Client システムで定義されています。リストを使用して、出力キャラクタ・セットを変更します。

4. 「エクスポート」をクリックします。

   エクスポート・ユーティリティを実行する前にリポジトリのメタデータに変更を加えた場合、「メタデータのエクスポート確認」ダイアログが表示されます。変更を保存するには、「コミット」をクリックします。前に保存した状態に戻すには、「ロールバック」をクリックします。変更をコミットするには、メタデータのエクスポート先のリポジトリに対する読込みおよび書込みのアクセス権が必要です。

   「メタデータのエクスポート進行状況」ダイアログに進行状況が表示されます。エクスポートが完了すると、「メタデータのエクスポート結果」ダイアログが表示されます。

   エクスポート・プロセスの詳細情報を表示するには、「ログ・ファイルの表示」をクリックします。

メタデータ・エクスポート・ファイルの形式

Metadata Loader では、例 15-2 に示すように、キーワードや位置を使用して .mdl エクスポート・ファイルの形式が指定されます。

例 15-2 エクスポート・ファイルのサンプル・レコード

```
#Project data <PhysicalName> <LogicalName> <UniversalID> <Version Label>
PROJECT|WarehouseName|Warehouse Name|A86184D5336911D58E9000B0D02A59E4|null
#Dimension <PhysicalName> <LogicalName> <UniversalID> <Prefix> <UsageType> <Imported>
<Generated>
DIMENSION|Channels|Channels Dimension Data
Mart|7E727655029911D58DC900C04F48E9ED|ch|null|N|N
```

この例では、エクスポート・ファイル内の各レコードはキーワードで始まり、その後に、1 つ以上の可変長フィールドが指定されています。表のフィールドは、デフォルトではパイプ（|）で区切られます。

プロジェクトのアーカイブ

プロジェクトをアーカイブすると、Warehouse Builder Design Repository 内に保存されているメタデータが外部のロケーションにコピーされます。これによって、ある特定の時点でのデータを安全に保管できます。Warehouse Builder では、このプロセスに役立つアーカイブ・ウィザードを使用できます。これらのユーティリティを実行すると、まず最初にデータがファイル・システムに書き込まれます。次に、書き込まれたデータをファイル・システムからサードパーティの管理ツール（Oracle Repository、ClearCase、SourceSafe など）にファイルを移動できます。

注意： アーカイブおよびリストアのユーティリティは、Oracle Warehouse Builder の次回のリリースではサポートされません。

プロジェクトをアーカイブまたはリストアするには、最初に、「作業環境」ページでアーカイブ / リストア設定を行う必要があります。これらの設定を行わずにアーカイブまたはリストアしようとすると、エラー・メッセージが表示されます。

プロジェクトのバージョン・ラベル

プロジェクトをアーカイブする前に、「プロジェクト・プロパティ」ダイアログでそのプロジェクトのバージョン・ラベルを更新できます。Warehouse Builder では、アーカイブ / リストアで使用するバージョン・ラベルを 2 つの場所でセットアップできます。

- その 1 つは、新規プロジェクト・ウィザードです。新規プロジェクト・ウィザードには、バージョンのプロパティを定義するための手順が含まれています。ここで設定するバージョン・ラベルは、プロジェクトのアーカイブ時に使用するバージョン・ラベルです。

- プロジェクトを作成した後でバージョン・ラベルを編集するには、そのプロジェクトの「プロパティ」ダイアログを開きます。次に、「バージョン・プロパティ」タブをクリックして、プロジェクトのバージョン・ラベルを変更します。

アーカイブとエクスポートの違い

アーカイブとリストアは、インポートとエクスポートとは異なります。表 15-1 は、アーカイブとエクスポートの相違点を示します。

表 15-1 アーカイブとエクスポートの違い

機能	アーカイブ	エクスポート
キャラクタ・セット	UTF8	ユーザーが設定
フィールド・セパレータ	パイプ記号（|）	ユーザーが設定
読取り専用の検出	検出し、再試行を要求	検出し、処理を中断
ダンプ・フォーマット	MDL	MDL
ログ・ファイル名	生成	生成（ユーザーが設定）
ファイルのロケーション	作業環境では次の構造で構成される `$ARCHIVE_HOME/ project_ name/Label/ Archive_ Name`	ユーザーが定義

プロジェクトのアーカイブ

プロジェクトをアーカイブする前に、「プロジェクト・プロパティ」ダイアログでそのプロジェクトのバージョン・ラベルを更新できます（15-12 ページの「プロジェクトのバージョン・ラベル」を参照）。

プロジェクトをアーカイブする手順は次のとおりです。

1. 「プロジェクト」メニューから「アーカイブ」を選択します。

   プロジェクトを右クリックして、ポップアップ・メニューから「アーカイブ」を選択する方法もあります。

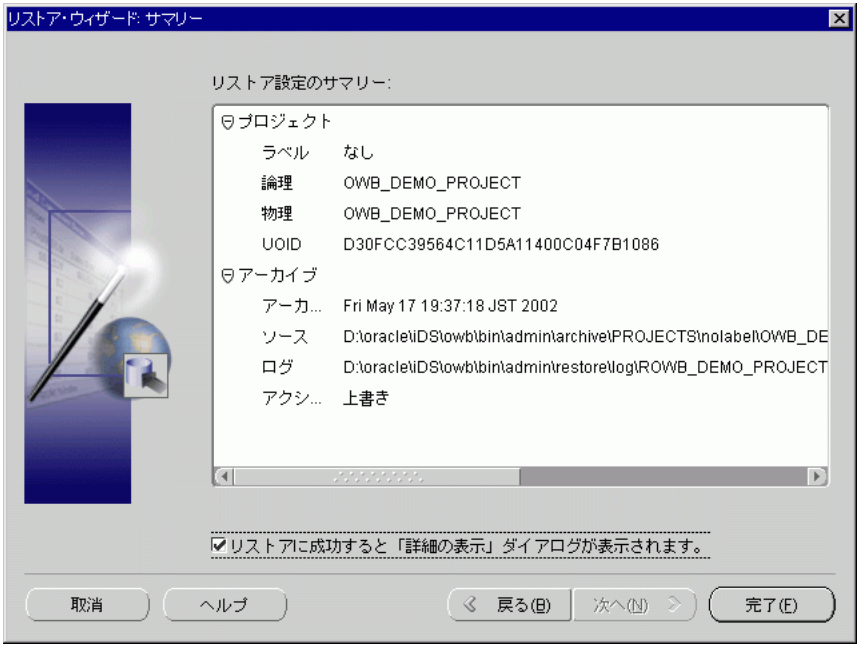
   図 15-4 のような、「アーカイブ・ウィザード: ようこそ」ページが表示されます。

2. 「次へ」をクリックします。

   アーカイブ・プロセスを実行する前に、「アーカイブ・ウィザード: サマリー」ページにアーカイブ設定が一覧表示されます。アーカイブ・プロセスの完了後にアーカイブの詳細情報を表示する場合は、「アーカイブに成功すると、「詳細の表示」ダイアログが表示されます。」にチェックマークを付けます。

ウィザードでは設定を変更できません。「アーカイブ・ウィザード : サマリー」ページに間違いがある場合は、「取消」をクリックし、アーカイブ / リストアの作業環境設定を変更してからアーカイブを続行してください。

図 15-4「アーカイブ・ウィザード : サマリー」ページ

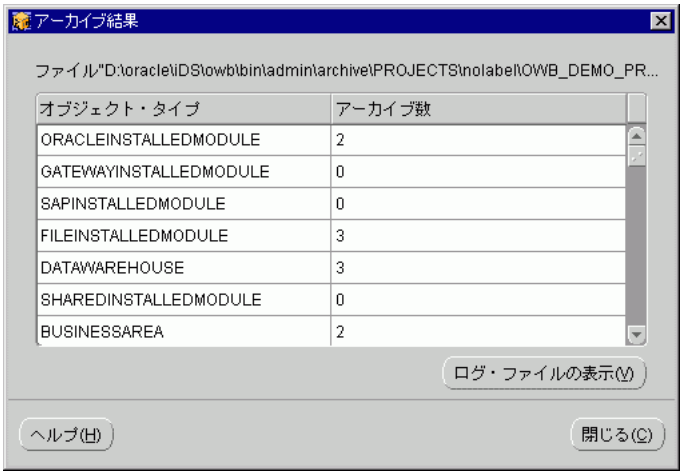


3. 「終了」をクリックします。

   アーカイブ・プロセスが開始され、進行状況ウィンドウが表示されます。進行状況バーが 100% に達すると、アーカイブ・プロセスの完了です。

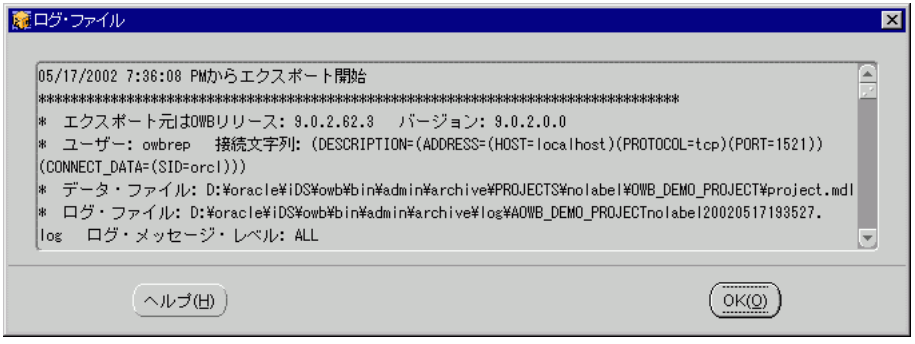
   「アーカイブに成功すると、「詳細の表示」ダイアログが表示されます。」にチェックマークを付けた場合は、「アーカイブ結果」ダイアログが表示されます。このダイアログでは、表 15-5 に示すように、アーカイブされたオブジェクト・タイプの名前とその各タイプの数が表示されます。

図 15-5 アーカイブ結果



アーカイブ・プロセスの詳細情報を表示するには、「ログ・ファイルの表示」をクリックします。表 15-6 のように、ログ・ファイルの内容がすべて表示されます。

図 15-6 アーカイブ・ログ・ファイル



メタデータのインポート

インポート・ユーティリティでは、エクスポートされたテキスト・ファイルのメタデータ情報が読み取られ、リポジトリへメタデータ・オブジェクトが挿入されます。メタデータ・インポート・ユーティリティでは、メタデータ・エクスポート・ユーティリティで作成されたファイルのみが読み取られます。

また、表の列やその制約、データ・ロード構成パラメータ、名前付き属性セットなど、エクスポートされたメタデータ・オブジェクトに属する情報をインポートできます。MDL を使用すると、プロジェクトまたはコレクションにオブジェクトをインポートできます。

DB2 や Informix などゲートウェイ・モジュールのメタデータを含む .mdl ファイルを、以前のバージョンの Warehouse Builder からインポートする場合、プロジェクト内の対応するソース・モジュール・フォルダにメタデータがインポートされないことがあります。インポートされたファイルは、ナビゲーション・ツリーの「その他」ノードに格納されます。ゲートウェイ・モジュールのメタデータは、手動で適切なソース・モジュール・フォルダにコピーする必要があります。

次の項では、メタデータ・インポート・ユーティリティの使用方法について説明します。

メタデータをインポートする前に

メタデータをインポートする前に、次の条件を満たしていることを確認します。

- **必要なセキュリティ権限 :** インポートを開始するには、MDL_IMPORT 権限が必要です。セキュリティの詳細は、19-8 ページの「PL/SQL でのセキュリティ管理」を参照してください。
- **必要なアクセス権限 :** 読込み / 書込みのアクセス権を持つユーザーのみが、Metadata Loader のインポート・ユーティリティを使用できます。インポート・ユーティリティではリポジトリが変更されるため、インポート前にメタデータ・オブジェクトをロックする必要があります。詳細は、15-3 ページの「MDL に必要なアクセス権限」を参照してください。
- **現在のリポジトリのバックアップ :** 大きなファイルのインポートや複雑なインポートを行う前に、既存のリポジトリを（エクスポートまたはメタデータのスナップショットの形式で）バックアップすることを検討してください。メタデータのエクスポートの詳細は、15-8 ページの「メタデータのエクスポート」を参照してください。メタデータのスナップショットの詳細は、第 16 章「メタデータ変更管理」を参照してください。

- **複数言語のサポート（MLS）のベース言語の互換性 :** ベース言語は、リポジトリで使用されるデフォルトの言語であり、インストール中に OWB Repository Assistant を使用して設定されます。この設定は、リポジトリのインストール後に変更できません。ベース言語が異なるメタデータ・オブジェクトをリポジトリにロードすると、エラーになります。リポジトリでベース言語を設定する際の詳細は、『Oracle Warehouse Builder 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。

メタデータ・インポート・ユーティリティについて

メタデータを Warehouse Builder Design Repository にインポートするには、次のいずれかを使用します。

- **Metadata Loader のコマンドライン・ユーティリティ :** コマンドライン・ユーティリティを使用すると、クライアント・インタフェースでは実行できない追加作業を行えます。たとえば、データをロードするために、構成パラメータのデフォルト値を無効にできます。コマンドラインでインポートする方法については、15-28 ページの「Metadata Loader のコマンドライン・ユーティリティの使用方法」を参照してください。
- **Warehouse Builder Design Client インタフェース :** クライアント・インタフェースの使用方法については、15-16 ページの「メタデータのインポート」を参照してください。

インポートの検証規則

すでにエクスポートされているメタデータから定義セットをインポートする場合、Warehouse Builder プロジェクト内の既存の定義を更新できます。ただし、特定のメタデータ定義は、更新されていることを確認する必要があります。次のようなエラーが発生する可能性があります。

- **マッピング定義。** メタデータ・インポート・ユーティリティで、インポートされたマッピング演算子が物理オブジェクトにバインドされません。ソース MDL ファイルに表示されるマッピング定義によってターゲット・リポジトリ内のマッピング定義が置き換えられる場合、新しいリポジトリ・マッピング定義はバインドされません。新しいマッピング演算子を、対応する物理オブジェクトにあわせて調整する必要があります。マッピング演算子がバインドされていないという警告メッセージがログ・ファイルに生成されます。
- **外部キー定義。** ソース MDL ファイルに、ターゲット・リポジトリにない一意キーまたは主キーに対する外部キー参照が含まれている可能性があります。MDL ファイルに表示されている外部キーの参照先である一意キーまたは主キーがターゲット・リポジトリに存在しない場合、ログ・ファイルに警告メッセージが生成されます。このメッセージは、外部キーの参照先のキーがリポジトリに含まれていないという内容になります。

Warehouse Builder Design Client を使用したメタデータのインポート

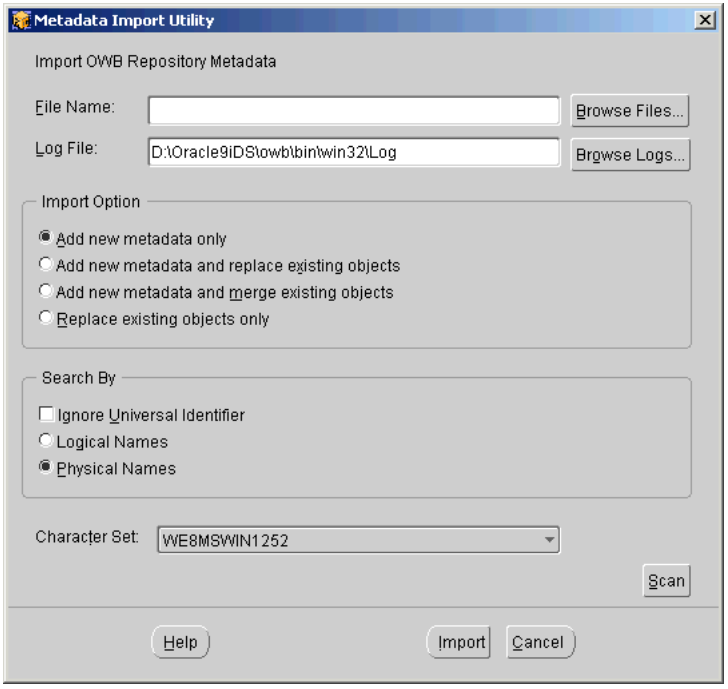
メタデータ・インポート・ユーティリティを使用すると、MDL ファイルから Warehouse Builder Design Repository にオブジェクトをインポートできます。

Warehouse Builder Design Client を使用してエクスポート・ファイルからオブジェクトをインポートする手順は次のとおりです。

1. メタデータのインポート先のプロジェクトを選択します。
2. Warehouse Builder コンソールから「プロジェクト」を選択し、「メタデータのインポート」を選択します。

   図 15-7 のような、「メタデータ・インポート・ユーティリティ」ダイアログが表示されます。

図 15-7 メタデータ・インポート・ユーティリティ



3. インポート・ファイルとそのログの名前とロケーションを次のように指定します。

   ファイル名 : MDL ファイルの名前を入力するか、「参照」をクリックして、インポートする MDL ファイルを選択します。

   ログ・ファイル : Warehouse Builder では、インポートに関する情報がログ・ファイルに記録されます。新しいパスとファイル名を入力して、ファイルのロケーションを変更できます。フィールドにパスとファイル名を入力するか、「参照」をクリックしてディレクトリまたはファイル名を選択します。

4. 「インポート・オプション」を 1 つ選択します。インポート・オプションの詳細は、15-21 ページの「インポート・モード」を参照してください。次のインポート・オプションから選択できます。

   新規メタデータのみの追加 : 新規オブジェクトをリポジトリに追加します。

   新規メタデータの追加および既存オブジェクトの置換 : 新規オブジェクトをリポジトリに追加し、既存オブジェクトを置換します。

   新規メタデータの追加および既存オブジェクトのマージ : 新規オブジェクトを追加し、リポジトリ内の既存オブジェクトに列をマージします。

   既存オブジェクトのみの置換 : リポジトリ内の既存オブジェクトを置換します。

5. 「一致基準」で、インポート・ファイル内のメタデータとリポジトリ内のメタデータの比較に使用する一致基準を指定します。詳細は、15-22 ページの「メタデータの一致基準」を参照してください。

   ユニバーサル ID を無視 : インポートするオブジェクトの検索でユニバーサル ID を使用しません。

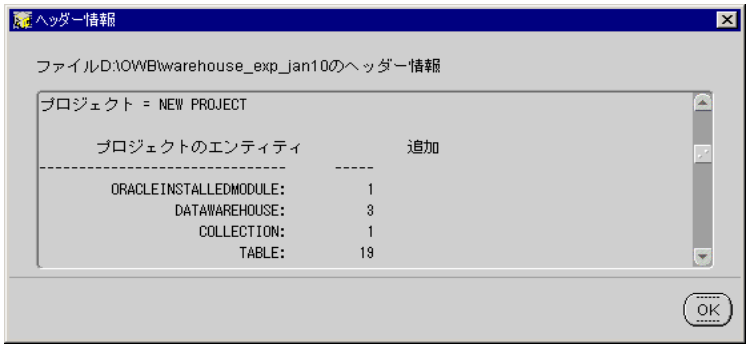
   名前 : インポートするオブジェクトの物理名を使用してリポジトリを検索し、そのオブジェクトがすでに存在するかどうかを確認します。

   キャラクタ・セット : インポート・ファイルの作成で使用するキャラクタ・セットのタイプを選択します。デフォルトのキャラクタ・セットは、Warehouse Builder Design Client マシンで定義されています。ドロップダウン・リストを使用して、出力キャラクタ・セットを変更します。

   OWB Repository Assistant を使用すると、新規の言語およびキャラクタ・セットを追加できます。詳細は、『Oracle Warehouse Builder 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。

6. 「スキャン」をクリックすると、図 15-8 に示すように、エクスポート済メタデータのヘッダー情報が表示されます。「ヘッダー情報」ダイアログには、選択したメタデータ・ファイルに含まれているオブジェクト・タイプの合計数が表示されます。

図 15-8 ヘッダー情報

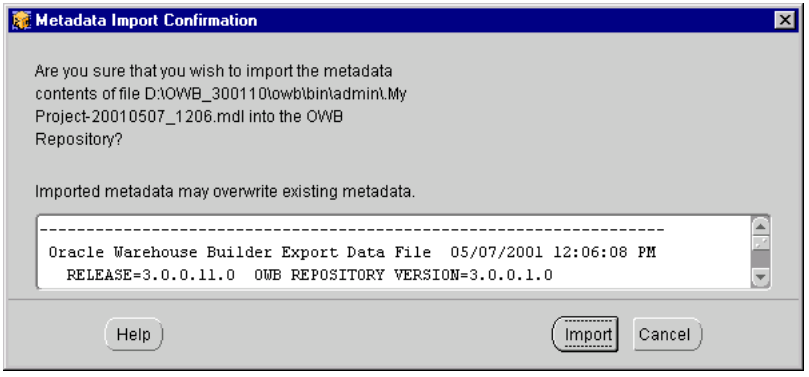


7. 「インポート」をクリックします。

   インポートを開始する前にデータを変更した場合は、「メタデータのインポート確認」ダイアログが表示されます。変更を保存するときは、「コミット」をクリックします。変更を無視して、前に保存した状態に戻すときは「ロールバック」をクリックします。

   エクスポート済メタデータ情報を確認していない場合は、図 15-9 に示すように、「メタデータのインポート確認」ダイアログが表示されます。

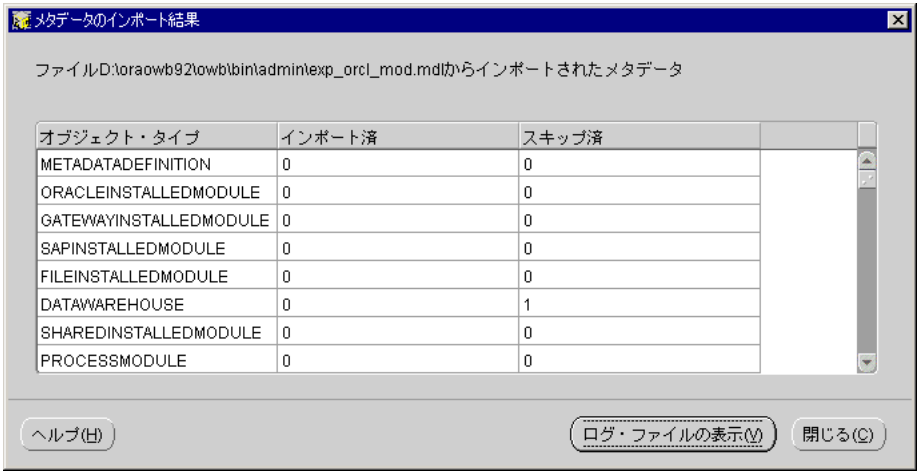
図 15-9 メタデータのインポート確認



8. 「インポート」をクリックして続行します。

   図 15-10 に示す「メタデータのインポート進行状況」パネルが表示されます。

図 15-10 メタデータのインポート結果



このダイアログには、インポートされたオブジェクトの数とスキップされたオブジェクトの数がオブジェクト・タイプ別に表示されます。インポート・プロセスの詳細情報を表示するには、「ログ・ファイルの表示」をクリックします。

インポート・モード

メタデータ・インポート・ユーティリティのグラフィカル・ユーザー・インタフェースは、次のいずれかのモードで動作します。

- **新規メタデータのみの追加（作成モード）:** 新規オブジェクトをリポジトリに追加します。MDL ファイルのオブジェクトがリポジトリ内にすでに存在する場合、変更はありません。
- **新規メタデータの追加および既存オブジェクトの置換（更新モード）:** 新規オブジェクトをリポジトリに追加し、既存オブジェクトを MDL ファイル内のオブジェクトで上書きします。
- **新規メタデータの追加および既存オブジェクトのマージ（マージ・モード）:** 新規オブジェクトを追加し、リポジトリ内の既存オブジェクトが MDL ファイル内のオブジェクトと異なる場合のみ、既存オブジェクトを上書きします。
- **既存オブジェクトのみの置換（置換モード）:** リポジトリ内の既存オブジェクトのみを置換します。このモードを使用しないと、メタデータ・オブジェクトをインポートするときに、既存のメタデータがすべて上書きされます。

更新モードまたは置換モードでインポートした場合、既存のオブジェクトの子が完全に置換され、最終的なオブジェクトは MDL ファイル内のオブジェクトとまったく同じになります。追加または置換されない既存のリポジトリ・オブジェクトの子はすべて削除されます。この処理は、子オブジェクトがマッピングに追加されているか、または表やビューの外部キーであるか、主キーであるか、一意キーであるかにかかわらず実行されます。

たとえば、MDL エクスポート・ファイルで、CUST 表に含まれている 3 つの列の物理名が Last_Name、First_Name、Middle_Init であるとします。一方、リポジトリにも同じ表がすでに作成されており、物理名がそれぞれ Last_Name、First_Name、Status、license_ID である 4 つの列が含まれているとします。置換操作では、列 Last_Name と First_Name が置き換えられ、列 Middle_Init が追加されます。また、列 Status と license_ID は削除されます。最終的に、Warehouse Builder Design Repository の CUST 表には、エクスポート・ファイルの CUST 表と同じメタデータが保存されます。

ヒント： 置換モードを使用すると、データ制約、メタデータ物理プロパティ設定、データ・ロード・プロパティ、マッピング定義などが失われる可能性があります。置換モードを使用する場合は、置換モードでインポートする前の状態にリポジトリをバックアップからリストアできることを確認してください。

メタデータの一致基準

メタデータ・インポート・ユーティリティでは、まずリポジトリに存在するメタデータ・オブジェクトが検索され、インポート元のファイルに含まれているオブジェクトと比較されます。比較方法は、ロード・モードと検索方法によって決まります。次の方法を使用できます。

- **物理名 :** 物理名は、エクスポート・ファイルにエクスポートされます。インポート時には、物理名に基づいて、そのオブジェクトの作成、置換またはマージが決まります。この方法は、ターゲット・ディレクトリ内のオブジェクト名が変更され、それらのオブジェクトで新規の UOID を作成する場合に使用します。
- **ユニバーサル・オブジェクト ID:** メタデータ・エクスポート・ユーティリティでは、エクスポート済の行オブジェクトについて、ユニバーサル・オブジェクト ID（UOID）という、システム生成の一意の ID がそれぞれ割り当てられます。行オブジェクトの UOID の目的は、オブジェクト表で行オブジェクトを一意に識別することです。インポート時に、MDL インポート・ユーティリティでは、これらの UOID に基づいて、行オブジェクトを作成するか、置換するか、またはマージするかが判断されます。この方法は、ターゲット・リポジトリ内のオブジェクト名が変更されていても、様々なリポジトリ間で UOID を維持する場合に使用します。

デフォルトでは、インポート・ユーティリティでは UOID による検索が行われます。ただし、ターゲット・リポジトリにすでに存在する MDL ファイル内のマッピング用 UOID は無視されます。

注意： MDL インポートをマージ・モードで実行する場合、既存のマッピングにマージするには、検索基準として UOID を使用する必要があります。また、MDL ファイルのマッピングにユニバーサル ID が存在しない場合は、名前が一致するマッピングにそのマッピングをマージできません。詳細は、15-21 ページの「インポート・モード」を参照してください。

検索方法はそれぞれ、いくつかの異なる組合せでインポート・モードと結合できます。組合せによって、インポート・プロセスで得られる結果は異なります。選択するモードによって、インポート前にリポジトリでメタデータ・オブジェクトを検索する方法が決まります。

たとえば、リポジトリ・オブジェクトの論理名がエクスポート・ファイルにエクスポートされている場合は、これらの論理名に基づいてリポジトリ内が検索されます。同じ論理名を持つオブジェクトが見つからない場合、実行される処理は選択したインポート・モードによって異なります。

表 15-2 では、MDL ファイル名が一致しないリポジトリ・オブジェクトについて、使用可能なインポート・モードで得られる結果を示します。

表 15-2 名前が一致しない場合の処理（インポート・モード別）

インポート・モード	結果
作成モード	新規オブジェクトが作成されます。
置換モード	置換するオブジェクトが見つからないため、そのオブジェクトをスキップすることを知らせる警告メッセージがログ・ファイルに書き込まれます。
更新モード	新規オブジェクトが作成されます。
マージ・モード	新規オブジェクトが作成されます。

表 15-3 では、MDL ファイル名が一致するリポジトリ・オブジェクトについて、使用可能なインポート・モードで得られる結果を説明します。

表 15-3 名前が一致する場合の処理（インポート・モード別）

インポート・モード	結果
作成モード	同じオブジェクトがすでに存在するため、そのオブジェクトがスキップされたことを通知するメッセージがログ・ファイルに書き込まれます。
置換モード	オブジェクトが置換されます。
更新モード	オブジェクトが置換されます。
マージ・モード	オブジェクトがマージされます。

MDL は、インポートされているメタデータを読み取って処理した後、ステータス情報と診断情報をログ・ファイルに書き込みます。インポートが完了すると、「メタデータのインポート結果」ダイアログが表示されます。

プロジェクトのリストア

プロジェクトをリストアすると、外部から取り込んだデータを使用して、Warehouse Builder Design Repository 内にメタデータが再作成されます。Warehouse Builder では、このプロセスに役立つリストア・ウィザードを使用できます。

注意： アーカイブおよびリストアのユーティリティは、Oracle Warehouse Builder の次回のリリースではサポートされません。

プロジェクトをアーカイブまたはリストアするには、最初に、「作業環境」ページでアーカイブ / リストア設定を行う必要があります。これらの設定を行わずにアーカイブまたはリストアしようとすると、エラー・メッセージが表示されます。

リストアとインポートの違い

アーカイブとリストアは、インポートとエクスポートとは異なります。表 15-4 は、リストアとインポートの相違点を示します。

表 15-4 リストアとインポートの違い

機能	リストア	インポート
キャラクタ・セット	UTF8	ユーザーが設定
プロジェクト全体の置換	可能	プロジェクトを削除せず、MDL モードによっては置換を実行
ダンプ・フォーマット	MDL	MDL
ユニバーサル ID の保持	常に保持	ユーザーが設定
名前の保持	常に保持	ユーザーが設定
ログ・ファイル名	生成	生成（ユーザーが設定）
モード	置換	作成 / 更新 / 置換 / マージ

プロジェクトのリストア

プロジェクトをリストアするには、次の手順に従います。

プロジェクトをリストアする手順は次のとおりです。

1. 「プロジェクト」メニューから「リストア」を選択します。

   「リストア・ウィザード: ようこそ」ページが表示されます。

2. 「次へ」をクリックします。

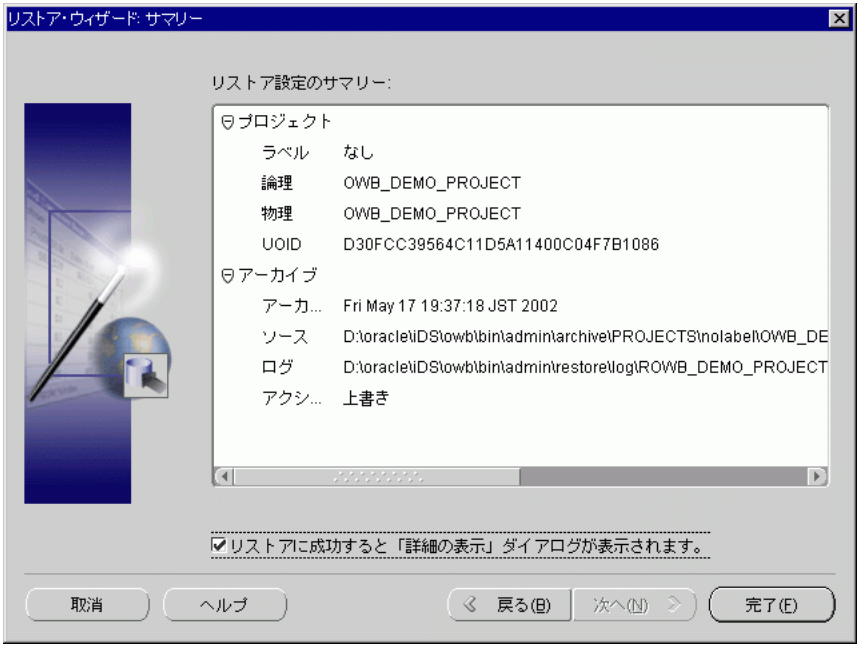
   図 15-11 のような、「アーカイブの選択」ページが表示されます。リストアするアーカイブ・ファイルを入力します。または、「参照」をクリックして、アーカイブ・ファイルを選択します。

図 15-11「アーカイブの選択」ページ

3. 「次へ」をクリックします。

   図 15-12 に示すように、「リストア・ウィザード : サマリー」ページに、リストア・プロセスを実行する前のリストア設定が一覧表示されます。リストア・プロセスの完了後にリストアの詳細情報を表示する場合は、「リストアに成功すると、「詳細の表示」ダイアログが表示されます。」にチェックマークを付けます。

図 15-12「リストア・ウィザード : サマリー」ページ

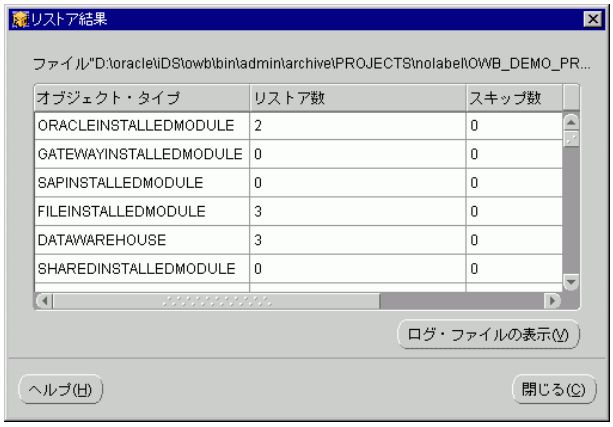


4. 「終了」をクリックします。

   リストア・プロセスが開始され、進行状況ウィンドウが表示されます。進行状況バーが100% に達すると、リストア・プロセスが完了します。

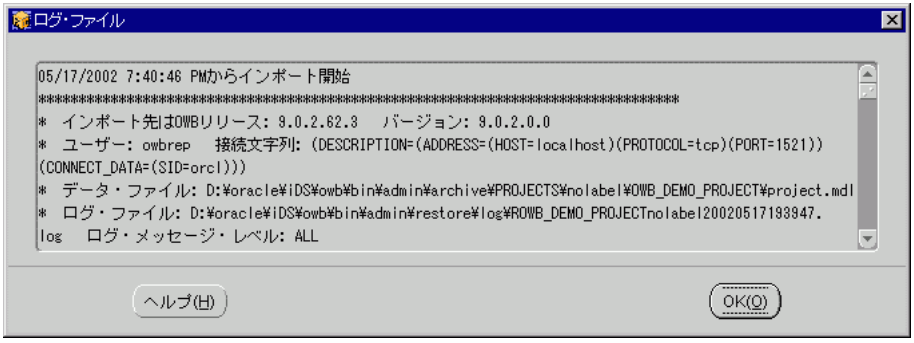
   「リストアに成功すると、「詳細の表示」ダイアログが表示されます。」にチェックマークを付けた場合は、図 15-13 に示すように、「リストア結果」ダイアログが表示されます。このダイアログには、オブジェクト・タイプの名前、リストアされたオブジェクト・タイプの数、およびスキップされたオブジェクト・タイプの数が表示されます。

図 15-13 リストア結果



アーカイブ・プロセスの詳細情報を表示するには、「ログ・ファイルの表示」をクリックします。図 15-14 のように、ログ・ファイルの内容がすべて表示されます。

図 15-14 リストア・ログ・ファイル



Metadata Loader のコマンドライン・ユーティリティの使用方法

MDL は、ユーザー・インタフェースではなく、コマンドラインからも操作できます。

この項では、次のトピックについて説明します。

Metadata Loader のコマンドライン・ユーティリティのアップグレード目的での使用方法に関する関連情報は、『Oracle Warehouse Builder 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。

コマンドラインでの MDL パラメータ・ファイルの作成

コマンドライン・インタフェースを使用すると、メタデータの移動方法を、GUI を使用した場合より詳細なレベルでカスタマイズできます。たとえば、メタデータをエクスポートするときは、コマンドラインを使用すると、構成値のエクスポートを無効にする、エクスポート・ファイル内のセパレータ文字を変更する、選択したエクスポート処理のパラメータ・ファイルを使用するなどの柔軟な機能を利用できます。

メタデータをインポートする場合にコマンドラインを使用すると、各オブジェクトに固有のインポート・アクションを作成できるという柔軟性があります。これらの操作によって、プロジェクト構造が同じである複数のリポジトリ内のメタデータを自動的に整理統合し、同期化できます。

エクスポート・ユーティリティとインポート・ユーティリティを実行するためのスクリプトは、*OWBHOME*¥owb¥bin¥win32 ディレクトリに配置されています。いずれのユーティリティもパラメータ・セットに基づいて実行されます。MDL パラメータは、次の方法で指定できます。

- コマンドライン・プロンプトに対する応答として MDL パラメータを入力します。
- MDL パラメータ・ファイルを作成します。
- コマンドライン・プロンプトに対する応答として MDL パラメータを入力し、MDL パラメータ・ファイルを作成します。

コマンドライン・ユーティリティを使用したメタデータのエクスポート

MDL ではデフォルトで、データをロードするための構成パラメータの値もエクスポートされますが、この設定はコマンドラインで上書きできます。プロジェクト内でファイルをエクスポートする方法を選択できます。たとえば、3 つのソース・モジュールと 2 つのターゲット・モジュールをエクスポートする場合は、別々にエクスポートするか、まとめてエクスポートするかを選択できます。

コマンドラインでプロジェクトをエクスポートする手順は次のとおりです。

1. MDL パラメータ・ファイルを作成します。
2. メタデータ・エクスポート・ユーティリティを実行します。

   次のコマンドによってメタデータ・エクスポート・ユーティリティを起動し、前述のパラメータ・ファイルを指定します。

   ```
   w:¥owb¥bin¥win32>exp parfile=e:¥MDL¥EXP_Directives
   Processing ... Export successful.
   ```

   オブジェクトはファイルにエクスポートされ、メタデータ・インポート・ユーティリティを使用してリポジトリにインポートできます。

エクスポート・ユーティリティのキーワード

MDL パラメータ・ファイルは、エクスポート・ユーティリティの実行に必要なパラメータ・セットが記述されたテキスト・ファイルです。エクスポート・パラメータのフォーマットは次のとおりです。

Keyword=Value

値のかわりに、任意の文字列を表すワイルドカード文字（*）を使用したり、名前付きオブジェクトのリストを指定したりできます。

Keyword=*

Keyword=(value-1, value-2, ..., -k)

たとえば、エクスポートする表セットを次のように指定します。

```
TABLES=(Customers, Products, Days)
```

例 15-3 は、モジュールをインポートするための典型的なパラメータ・ファイルを示しています。

例 15-3 パラメータ・ファイルの形式

```
USERID=GCCWH/GCCWH@dwdoc11-pc:1521:ora816
PROJECT=GCCWarehouse
FILE=e:¥MDL¥GCCWarehouse-exp-JUL01
FIELDSEPARATOR=|
LOG=e:¥MDL¥GCCWarehouse-exp-JUL01-LOG
CONFIGPARAM=N
```

表 15-5 は、エクスポート・パラメータで使用するキーワードの一覧です。スクリプトにコメントを付けるには、コメント記号（#）を使用します。レコードの最初の列にこの記号を挿入し、その後にテキストを入力します。

表 15-5 エクスポート・ユーティリティ・パラメータのキーワード

ユーティリティのプロンプト	キーワード	説明
Username/password @host:port:sid	USERID	ユーザー名、パスワード、接続を表す文字列。
	USERNAME	Warehouse Builder Design Repository にアクセスするためのユーザー名。
	PASSWORD	USERNAME に割り当てられているパスワード。
	HOST	Warehouse Builder Design Repository のマシン名。
	PORT	Warehouse Builder Design Repository データベース・リスナーのポート。
	SID	Warehouse Builder Design Repository データベースの SID。
プロジェクト名	PROJECT	プロジェクト名。ワイルドカードを使用してプロジェクトを指定できます。ただし、ワイルドカードを使用する場合は、その後に他のオブジェクト・タイプ・キーワードを指定できません。共有変換をエクスポートするには、`PROJECT=Global Shared` を使用します。
エクスポート・ファイル	FILE	エクスポートしたデータ用のファイル名。
フィールド・セパレータ	FIELDSEPARATOR	フィールド・セパレータ：|、^ または ~。

表 15-5 エクスポート・ユーティリティ・パラメータのキーワード（続き）

ユーティリティのプロンプト	キーワード	説明
ログ・ファイル	LOG	エクスポートのステータス情報と統計情報用のファイル名。
パラメータ・ファイル	PARFILE	キーワードが保存されているパラメータ・ファイル。
	CONFIGPARAM	構成値のエクスポート（Y/N）。デフォルトは Y です。
	TRACE	デバッグ・メッセージ。次のオプションを指定できます。 S - メッセージを画面に書き込みます。 Y - メッセージをファイルに書き込みます。 B - メッセージを画面およびファイルに書き込みます。
	TRACEFILE	トレース・ファイルの名前。
	PHYSICALNAMES	エクスポートするオブジェクトの検索で物理名を使用します（Y/N）。デフォルトは N です。
	CHARACTERSET	エクスポート・データ・ファイルで使用するキャラクタ・セット。
	MODULES	MODULE の値としてワイルドカードまたは複数値フォーマットを使用した場合、その後に他のオブジェクト・タイプ・キーワードを指定できません。単純フォーマットを使用する場合は、このキーワードを複数回使用し、その直後に該当するオブジェクト・タイプのキーワードを任意のフォーマット（単純、ワイルドカード、複数値）で指定できます。
	TABLES	
	VIEWS	
	FILES	
	SEQUENCES	
	MATERIALIZED VIEWS	
	DIMENSIONS	
	FACTS	

表 15-5 エクスポート・ユーティリティ・パラメータのキーワード（続き）

ユーティリティのプロンプト	キーワード	説明
	TRANSFORM CATEGORIES	ワイルドカードまたは複数値フォーマットを使用する場合は、その後に FUNCTIONS キーワードを指定できません。単純フォーマットを使用する場合は、このキーワードを複数回使用できます。また、その直後に、任意のフォーマット（単純、ワールドカード、複数値）で FUNCTIONS キーワードを指定できます。
	FUNCTIONS	
	MAPPINGS	
	COLLECTIONS	
	LOCATIONS	
	CONNECTORS	
	RUNTIMEREPOSITORY CONNECTIONS	
	STANDALONEFUNCTIONS	
	STANDALONEPROCEDURES	
	ADVANCEDQUEUES	
	EXTERNALTABLES	
	PROCESSES	
	SNAPSHOTS	
	QUERYOBJECTS	
	REPORTS	
	REPORTGROUPS	
	IOBUSINESSAREAS	
	HELP	詳細リストを表示するには、HELP=Y を使用します。
	#	パラメータ・ファイルで使用するコメント行。

コマンドライン・ユーティリティを使用したメタデータのインポート

選択したモジュールをインポートする手順は次のとおりです。

1. MDL パラメータ・ファイルを作成します。
2. メタデータ・インポート・ユーティリティを実行します。

   次のコマンドによってインポート・ユーティリティを起動し、前述の MDL パラメータ・ファイルを指定します。

```
w:¥owb¥bin¥win32>imp parfile=e:¥MDL¥IMP_Directives.txt
Processing ...
Import successful.
```

インポート・ユーティリティのキーワード

MDL エクスポートと同様、ファイルからオブジェクトをインポートするよう MDL インポートに命令するには、プロンプトに応答するか、またはパラメータ・セットが指定されたファイルを作成します。例 15-4 は、モジュールをインポートするための典型的なパラメータ・ファイルを示しています。

例 15-4 パラメータ・ファイルの形式

```
USERID=GCCWH/GCCWH@dwdoc11-pc:1521:ora816
FILE=e:¥MDL¥gccstar-exp
LOG=e:¥MDL¥gccstar-imp-LOG
MODE=CREATE
CONFIGPARAM=N
```

表 15-6 は、インポート・パラメータで使用するキーワードの一覧です。

表 15-6 インポート・ユーティリティ・パラメータのキーワード

ユーティリティのプロンプト	キーワード	説明
Username/passw @host:port:sid	USERID	ユーザー名、パスワード、接続を表す文字列。
	USERNAME	Warehouse Builder Design Repository にアクセスするためのユーザー名。
	PASSWORD	USERNAME に割り当てられているユーザー・パスワード。
	HOST	Warehouse Builder Design Repository のマシン名。
	PORT	Warehouse Builder Design Repository のポート。

表 15-6 インポート・ユーティリティ・パラメータのキーワード（続き）

ユーティリティのプロンプト	キーワード	説明
	SID	Warehouse Builder Design Repository の SID。
インポート・ファイル	FILE	データのインポート先ファイルの名前。
インポート・モード	MODE	CREATE、REPLACE、UPDATE または INCREMENTALUPDATE。
ログ・ファイル	LOG	エクスポートのステータス情報と統計情報用のファイル名。
パラメータ・ファイル	PARFILE	キーワードが保存されているパラメータ・ファイル。
	CONFIGPARAM	構成値のインポート（Y/N）。デフォルトは Y です。
	TRACE	デバッグ・メッセージ。次のオプションを指定できます。 S - メッセージを画面に書き込みます。 Y - メッセージをファイルに書き込みます。 B - メッセージを画面およびファイルに書き込みます。
	TRACEFILE	トレース・ファイルの名前。
	PHYSICALNAMES	インポートするオブジェクトの検索に物理名を使用します（Y/N）。デフォルトは Y です。
	CHARACTERSET	エクスポート・データ・ファイルで使用するキャラクタ・セット。
	HELP	詳細リストを表示するには、HELP=Y を使用します。
	#	パラメータ・ファイルで使用するコメント行。
	IGNOREUniversalID	ユニバーサル ID を無視し、検索基準として使用しません（Y/N）。デフォルトは N です。
	PRESERVEDESCRIPTION	すでに存在しているオブジェクトの説明が MDL データ・ファイルに含まれていない場合、そのオブジェクトの説明を保持します（Y/N）。デフォルトは N です。
	SINGLEUSER	インポートの実行時にシングル・ユーザー・ロックを要求します（Y/N）。デフォルトは N です。

MODE パラメータを指定しない場合は、デフォルトの CREATE が適用されます。

インポート・ユーティリティでは、MDL パラメータ・ファイルの実行に加えて、パラメータ・ファイルにアクション・プランを指定できます。アクション・プランでは、インポートしたファイル内の各オブジェクトに対して行う特定の処理を定義できます。最初に、オブジェクトをインポートするか、スキップするかまたは削除するかを指定する必要があります。オブジェクトのインポートを選択した場合は、インポート・モードを CREATE、UPDATE、REPLACE または INCREMENTAL UPDATE のいずれかに設定できます。

例 15-5 は、アクション・プランを定義した MDL パラメータ・ファイルの例です。

例 15-5 MDL アクション・プラン

```
USERID=user_sample/user_sample@test-pc:1521:ora8i
#
FILE=e:¥test¥data¥sample_file.mdl
LOG=e:¥test¥log ¥imp_sample_file.log
#
MODE=ACTIONPLAN
PHYSICALNAMES=Y
IGNOREUOID=Y
#
# User-Specified Action Plan
#
ACTION=NONE
PROJECT=MY PROJECT
MODULES=(DATAWAREHOUSE)
#
ACTION=CREATE
TABLES=(TABLE_3)
FACTS=(FACT1, FACT2, FACT3)
SEQUENCES=(SEQ_A, SEQ_B, SEQ_C)
#
ACTION=REPLACE
TABLES=(TABLE_1, TABLE_2)
DIMENSIONS=(DIM1, DIM2, DIM3)
#
ACTION=DELETE
 TABLES=(TABLE_A, TABLE_B)
#
# Switching to a different module
ACTION=REPLACE
MODULES=(FLAT_FILE)
FILES=(FILE_1, FILE_2)
#
ACTION=CREATE
FILES=(FILE_3)
#
ACTION=DELETE
FILES=(FILE_X)
```

分割を使用した Warehouse Builder マッピングのエクスポート / インポート

分割ユーティリティを使用すると、MDL インポート・ユーティリティで多数のマッピングをインポートする際のメモリー制約に対応できます。このユーティリティは、すべてのマッピングを一度ではなく分割して移行するエクスポート・スクリプトとインポート・スクリプトを生成します。スクリプトを生成すると、CREATE モードを使用した MDL パラメータ・ファイルが作成されます。これらのファイルは編集できます。

データが大きいために MDL インポートを実行できない場合は、分割ユーティリティを使用してマッピング・データを分割し、エクスポートとインポートを再実行します。その他すべてのオブジェクト・タイプは、標準の MDL ユーティリティを使用してエクスポート / インポートする必要があります。小さい単位に分割できるのはマッピングのみです。マッピング以外のすべてのエンティティをエクスポートするには、次のように指定されたパラメータ・ファイルを使用します。

- VIEWS=*
- TABLES=*
- SEQUENCES=*
- MATERIALIZEDVIEWS=*
- CUBES=*
- FILES=*
- DIMENSIONS=*
- VIRTUALTABLES=*
- TEMPORARYTABLES=*
- TRANSFORMCATEGORIES=*

分割ユーティリティは、Warehouse Builder プロジェクトのモジュール内のマッピングを分割します。分割サイズは、このアプリケーションに備わっているファイル内のパラメータで指定します。

expsplit バッチ・スクリプトでは次の引数を指定できます。

- パラメータ・ファイル。c:¥temp¥owb_apps.txt という形式で指定します。このパラメータ・ファイルには、マッピング数、データ・ファイル名、拡張子を指定する特殊なキーワードが含まれます（この後の説明を参照）。
- パラメータ・ターゲット・ファイルの接頭辞。c:¥temp¥owb_apps "1 から始まるピース番号 " という形式で指定します。生成されたパラメータ・ファイルには接尾辞 .txt が追加されます。また、エクスポート・バッチ・ファイルには接尾辞 .bat が追加され、インポート・バッチ・ファイルには接尾辞 _imp.bat が追加されます。

次の例は、分割ユーティリティの起動方法を示します。

```
expsplit exampleparams.txt c:¥temp¥ora_apps
```

パラメータ・ファイル exampleparams.txt を使用します。このファイルには次のパラメータが含まれます。

- userid=apps/apps@130.35.12.73:1521:orcl0
- PHYSICALNAMES=Y
- LOG=c:¥temp¥owb_data_apps
- LOGEXT=log
- FILE=c:¥temp¥owb_data_apps
- FILEEXT=dat
- FIELDSEPARATOR=^
- PROJECT=EDWPRJ
- MODULES=EDW_COMMON_MODULE
- TYPE=MAPPINGS
- COUNT=70

このファイルは、Warehouse Builder Metadata Loader のエクスポート・パラメータ・ファイルと似ています。相違点は表 15-7 のとおりです。

表 15-7 分割ユーティリティで使用するエクスポート・パラメータのキーワード

パラメータ・ファイルのキーワード	説明
FILE	データのエクスポート先ファイルの名前は、データ・ファイルの接頭辞、チャンク番号、ファイル拡張子（FILEEXT）で構成されます。
FILEEXT	データ・ファイルの拡張子。
PHYSICALNAMES	名前の照合で使用します。
LOG	ログ・ファイル名は、そのログ・ファイルの接頭辞、チャンク番号、ファイル拡張子（FILEEXT）で構成されます。
LOGEXT	ログ・ファイルの拡張子。
PROJECT	プロジェクト名を 1 つだけ指定します。
MODULES	モジュール名を 1 つだけ指定します。
TYPE	必ず MAPPINGS を指定します。
COUNT	各エクスポート・チャンクに書き込むマッピングの数。

Warehouse Builder プロジェクトのマッピングを分割する場合、生成されるパラメータ・ファイルの名前は次のようになります。

```
owb_apps1.txt
owb_apps2.txt
```

リポジトリからデータをエクスポートするためのバッチ・ファイル c:¥temp¥owb_apps.bat（指定したパラメータ・ターゲット・ファイルの接頭辞）が生成されます。さらに、同じリポジトリにデータを CREATE モードでインポートするためのインポート・バッチ・ファイルが作成されます。他のターゲット・データベースを使用する場合は、これらのファイルを編集できます。

分割ユーティリティを使用してデータを移行する手順は次のとおりです。

1. コマンドラインまたは Warehouse Builder を使用して、マッピング以外のすべてのオブジェクトの MDL エクスポートを実行します。

   マッピング以外のすべてのオブジェクトをコマンドラインからエクスポートするには、次のキーワードが指定されたパラメータ・ファイルを使用します。

   ```
   VIEWS=*

   TABLES=*

   SEQUENCES=*

   MATERIALIZEDVIEWS=*

   FACTS=*

   FILES=*

   DIMENSIONS=*

   VIRTUALTABLES=*

   TEMPORARYTABLES=*

   TRANSFORMCATEGORIES=*
   ```

   Warehouse Builder を使用してエクスポートを実行する場合は、マッピング以外のオブジェクトを選択して、エクスポートします。

2. 新しいエクスポート・ファイルをターゲット・リポジトリにインポートします。

3. 分割ユーティリティを使用して、マッピングを分割してエクスポートします。

   ```
   expsplit exampleparams.txt c:¥temp¥ora_apps
   ```

   このユーティリティを実行すると、ソース・リポジトリに接続してマッピングが分割され、exampleparams.txt に従って複数のパラメータ・ファイルが作成されます。これらのパラメータ・ファイルはエクスポート時に使用します。さらに、エクスポート・バッチ・ファイルとインポート・バッチ・ファイルも作成されます。

表 15-8 は、作成されるファイルの一覧です。

表 15-8 分割ユーティリティによって作成されるファイル

説明	ファイル名
エクスポートを実行するためのバッチ・ファイル	c:¥temp¥ora_apps.bat
インポートを実行するためのバッチ・ファイル	c:¥temp¥ora_apps_imp.bat
エクスポート・バッチ・ファイルとインポート・バッチ・ファイルが使用する複数のパラメータ・ファイル	c:¥temp¥ora_apps1.txt c:¥temp¥ora_apps2.txt c:¥temp¥ora_apps3.txt

4. エクスポート・バッチ・ファイルを実行して、パラメータ・ファイルで指定されているロケーション（手順 3 で指定した変数 FILE）にマッピングをエクスポートします。
5. 生成されたパラメータ・ファイル `c:¥temp¥ora_apps1.txt`、`c:¥temp¥ora_apps2.txt` を変更します。接続情報にターゲット・リポジトリを指定します。
6. インポート・バッチ・ファイル `c:¥temp¥ora_apps_imp.bat` を実行して、データをインポートします。

16

メタデータ変更管理

Oracle Warehouse Builder では、スナップショットを使用したメタデータ変更管理およびバージョン管理ができます。メタデータ・スナップショットは、バックアップ、バージョン管理、比較およびリストアの目的で使用できます。

この項では、次のトピックについて説明します。

この項では、Warehouse Builder グラフィカル・ユーザー・インタフェースを使用したメタデータ変更管理機能について説明します。OMB Plus でスナップショットを作成および管理する際に使用するコマンドおよび引数の詳細は、『Oracle Warehouse Builder スクリプト・リファレンス』を参照してください。

メタデータ・スナップショットについて

ファイル・システムに .mdl ファイルが別個に格納される MDL エクスポートとは異なり、スナップショットはオブジェクトとしてデータベースに格納されます。すべてのファースト・クラス・オブジェクト（ナビゲーション・ツリーからアクセスできるオブジェクト）について、スナップショットを作成できます。Warehouse Builder オブジェクトの分類方法については、E-5 ページの「オブジェクト所有権ツリー」を参照してください。

スナップショットは、リポジトリ内のオブジェクトを説明するものであり、そのオブジェクトに関連付けられています。該当するオブジェクトとそのオブジェクトの関係に関する情報がすべて含まれています。コレクションには実際のオブジェクトのショートカットが含まれているため、コレクションのスナップショットは、ショートカットが指す実際のオブジェクトすべてのスナップショットになります。

子オブジェクトを含む親オブジェクトのスナップショットを作成した後で子オブジェクトが変更された場合、スナップショットを比較すると、親オブジェクトが変更されたことを確認できます。たとえば、プロジェクトのカスケード・スナップショットを作成すると、そのプロジェクト内のモジュールはすべて子になります。変更されたモジュールがある場合、スナップショットとリポジトリを比較すると、プロジェクトが変更されたことを確認できます。

1 つのオブジェクトに対して、定義はリポジトリ内で 1 つしか指定できませんが、複数のスナップショットを作成して様々な時点の状態を示すことができます。

注意： コレクションのスナップショットを作成すると、コレクション内にあるショートカットのスナップショットではなく、実際のオブジェクトのスナップショットが作成されます。このスナップショットからオブジェクトをリストアすると、コレクションと実際のオブジェクトが両方ともリストアされます。そのため、コレクションをリストアする際に実際のオブジェクトではなくコレクションのみを変更する場合、まずリストアするオブジェクトを選択解除する必要があります。

スナップショットおよびオブジェクトの削除

Warehouse Builder ナビゲーション・ツリーからオブジェクトを削除すると、「削除確認」ダイアログが表示されます。このオブジェクトをごみ箱に入れるように指示するチェック・ボックスを選択して「OK」をクリックすると、削除するオブジェクトのスナップショットを作成するという旨の「スナップショット・アクション」ダイアログが表示されます。

スナップショットのタイプ

Warehouse Builder で作成可能なスナップショットには、完全スナップショットとシグネチャ・スナップショット、およびカスケード・スナップショットとカスケードなしスナップショットがあります。

完全スナップショットとシグネチャ・スナップショット

このカテゴリには、次のスナップショットを指定します。

- **完全スナップショット:** 取得時に選択されていたコンポーネントの完全なメタデータ・セットが含まれます。完全スナップショットでは、作成にかかる時間も必要な記憶域も、シグネチャ・スナップショットより多くなりますが、以前の履歴ポイントまでリポジトリをリストアする際に使用できる完全なメタデータを作成できます。また、完全スナップショットは、他のリポジトリ・オブジェクトと同様にエクスポートできます。
- **シグネチャ・スナップショット:** オブジェクトのシグネチャを取得します。シグネチャ名には、選択対象のメタデータ・コンポーネントについて、別のスナップショットと比較したときに変更を検出し、デルタ値を計算するための十分な情報が含まれています。シグネチャ・スナップショットでは、スナップショットを短時間で作成できますが、比較および情報提供の用途にのみ使用できます。2 つのスナップショットの比較レポートでは、新規のオブジェクトや見つからなかったオブジェクト、変更されたオブジェクトを識別できます。シグネチャ・スナップショットを、エクスポートおよびインポートすることはできません。

空き領域がなくなってきた場合、リストアで不要になった完全スナップショットをシグネチャ・スナップショットに変換すると、使用される領域が減ります。この変換によって、スナップショット履歴が保持され、リポジトリ内の領域もかなり節約できます。現在、Warehouse Builder では、スクリプトを使用した変換のみが可能です。

注意: 完全スナップショットをシグネチャ・スナップショットに変換すると、そのスナップショットをリストアの用途では使用できなくなります。

カスケード・スナップショットとカスケードなしスナップショット

カスケードでは、スナップショットに含めるオブジェクトを指定できます。

- **カスケードなしスナップショット:** 子オブジェクトを含むコンテナ・オブジェクト（プロジェクトやモジュールなど）であっても、オブジェクト自体の定義のみが含まれます。たとえば、表を含む Warehouse Module A のカスケードなしスナップショットを作成すると、スナップショットには Module A の定義のみが含まれます。
- **カスケード・スナップショット:** オブジェクトの展開された内容が含まれます。2 つの表を含む Warehouse Module A のカスケード・スナップショットを作成すると、スナップショットには Module A の定義と、2 つの表の定義が含まれます。

スナップショットの組合せ

これら 2 つのカテゴリは、相互に排他的ではありません。表 16-1 は、親オブジェクトのスナップショットの組合せごとに得られる結果を示しています。

表 16-1 スナップショットのタイプの組合せ

スナップショットのタイプ	2 つの表（Table1、Table2）を含む Module A のスナップショットの結果
シグネチャ、カスケード・スナップショット	このタイプのスナップショットには、Module A、Table1 および Table2 に関する情報が含まれます。 この情報は、これら 3 つのオブジェクトのリストアには使用できません。比較目的でのみ使用できます。
完全、カスケード・スナップショット	このタイプのスナップショットには、Module A、Table1 および Table2 の定義が含まれます。 このスナップショットは、これら 3 つのオブジェクトのリストアに使用できます。また、比較目的のみでも使用できます。
シグネチャ、カスケードなしスナップショット	このタイプのスナップショットには、Module A のみの情報が含まれます。 この情報は、Module A のリストアには使用できません。比較目的でのみ使用できます。 Table1 または Table2 に変更があっても、比較したときに Module A の変更は確認できません。
完全、カスケードなしスナップショット	このタイプのスナップショットには、Module A のみの定義が含まれます。 このスナップショットは、Module A のリストアまたは比較に使用できますが、子オブジェクトには使用できません。 Table1 または Table2 に変更があり、それ以外に Module A に変更がない場合は、比較したときに Module A の変更は確認できません。

スナップショットの作成

メタデータ変更管理では、指定した時点での、メタデータ・リポジトリまたはリポジトリに含まれる特定のオブジェクトの内容を取得するスナップショットを作成できます。スナップショットを使用すると、メタデータの変更を検出し、レポートできます。

注意： コレクションのスナップショットを作成すると、コレクション内にあるショートカットのスナップショットではなく、実際のオブジェクトのスナップショットが作成されます。このスナップショットからオブジェクトをリストアすると、コレクションと実際のオブジェクトが両方ともリストアされます。そのため、コレクションをリストアする際に実際のオブジェクトではなくコレクションのみを変更する場合、まずリストアするオブジェクトを選択解除する必要があります。

Warehouse Builder でスナップショットを作成する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーから、スナップショットに含めるコンポーネントを複数選択します。たとえば、Oracle ウェアハウス・モジュールから表、マッピング、ディメンションを選択できます。
2. 複数選択したコンポーネントを右クリックし、ポップアップ・メニューから「スナップショットの作成」を選択します。

   「スナップショットの作成 - ようこそ」ページが表示されます。
3. 「次へ」をクリックします。

   「スナップショットの作成 - ステップ 3 の 1: 名前」ページが表示されます。
4. 次の情報を入力します。

   名前 : 新規のスナップショットの名前を 30 文字以内で入力します。

   タイプ : スナップショットのタイプ（完全またはシグネチャ）を指定します。スナップショットのタイプの詳細は、16-3 ページの「スナップショットのタイプ」を参照してください。

   説明 : 新規のスナップショットの説明を 2000 文字以内で入力します。
5. 「次へ」をクリックします。

   「スナップショットの作成 - ステップ 3 の 2: コンポーネント」ページが表示されます。

   このページには、次の 3 つの列が含まれています。

   オブジェクト : スナップショットに含める Warehouse Builder コンポーネントを示します。

   タイプ : Warehouse Builder コンポーネントのタイプを指定します。たとえば、ウェアハウス・モジュール、表、ディメンション、マッピング、ビューなどです。

カスケード: このオプションで、スナップショットにおけるオブジェクトの格納方法を選択できます。オブジェクトのカスケードを選択する場合、スナップショットにそのオブジェクトの子コンポーネントすべてを含めることになります。たとえば、Oracle ターゲット・モジュールをカスケードするときは、そのモジュールに含まれる表、ディメンション、キューブおよびその他のオブジェクトをすべて選択します。このオプションは、モジュール、プロジェクト、コレクションなど、フォルダレベルのオブジェクトにのみ適用されます。

6. スナップショットに含める各コンポーネントで「カスケード」オプションを選択して、「次へ」をクリックします。

   「スナップショットの作成 - ステップ 3 の 3: 依存性」ページが表示されます。

   このページでは、スナップショット内のオブジェクトの依存するコンポーネントを選択できます。たとえば、外部キーを含む表は、参照キーを含む表に依存しています。一意キーを含む表をスナップショットに含める場合は、依存性レベルの数を 1 に指定します。ここで指定できる最大値は 100 です。

7. 必要に応じて、「プレビュー」をクリックすると、指定したレベル数の依存コンポーネントのリストが表示されます。

   「依存性のプレビュー」ダイアログでは、スナップショットに含まれる依存するコンポーネントをプレビューできます。依存するコンポーネントのリストは、「深さ」ページで指定したレベル数に基づいて表示されます。たとえば、表を含むマッピングをスナップショットに含めて、レベル数を 1 に指定した場合、このダイアログには、そのマッピングに含まれている表が表示されます。この表に外部キーが含まれている場合、レベル数を 2 に指定すると、このダイアログには一意キーを含む関連する表が表示されます。

8. 「次へ」をクリックします。

   「スナップショットの作成 - 終了」ページが表示されます。

9. このウィザードで選択した内容を確認して、新規のスナップショットを作成します。変更する場合は、「戻る」をクリックします。または、「終了」をクリックして、スナップショットを作成します。

   作成したスナップショットが「メタデータ変更管理」ウィンドウに表示されます。

スナップショットの更新

スナップショットの作成後、コンポーネントを追加してスナップショットを更新できます。この項では、既存のスナップショットを更新する方法について説明します。

既存のスナップショットを更新する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーから、スナップショットに追加するコンポーネントを選択します。たとえば、Oracle ウェアハウス・モジュールから表、マッピング、ディメンションを選択できます。
2. 選択したコンポーネントを右クリックし、ポップアップ・メニューから「スナップショットへの追加」を選択します。

   「スナップショットへの追加 - ようこそ」ページが表示されます。
3. 「次へ」をクリックします。

   「スナップショットへの追加」ページが表示されます。
4. ドロップダウン・フィールドから更新する既存のスナップショットを選択して、「次へ」をクリックします。

   「スナップショットへの追加」ページが表示されます。このページには、次の 3 つの列が含まれています。

   オブジェクト : スナップショットに追加する Warehouse Builder コンポーネントを示します。

   タイプ : Warehouse Builder オブジェクトのタイプを指定します。たとえば、ウェアハウス・モジュール、表、ディメンション、マッピング、ビューなどです。

   カスケード : このオプションで、スナップショットにおけるオブジェクトの格納方法を選択できます。オブジェクトのカスケードを選択する場合、スナップショットにそのオブジェクトの子コンポーネントすべてを含めることになります。たとえば、モジュールをカスケードするときは、そのモジュールに含まれる表、ディメンション、キューブおよびその他のオブジェクトをすべて選択します。このオプションは、モジュールやプロジェクトなど、フォルダレベルのオブジェクトのみに適用されます。
5. 追加するコンポーネントで「カスケード」オプションを選択して、「次へ」をクリックします。

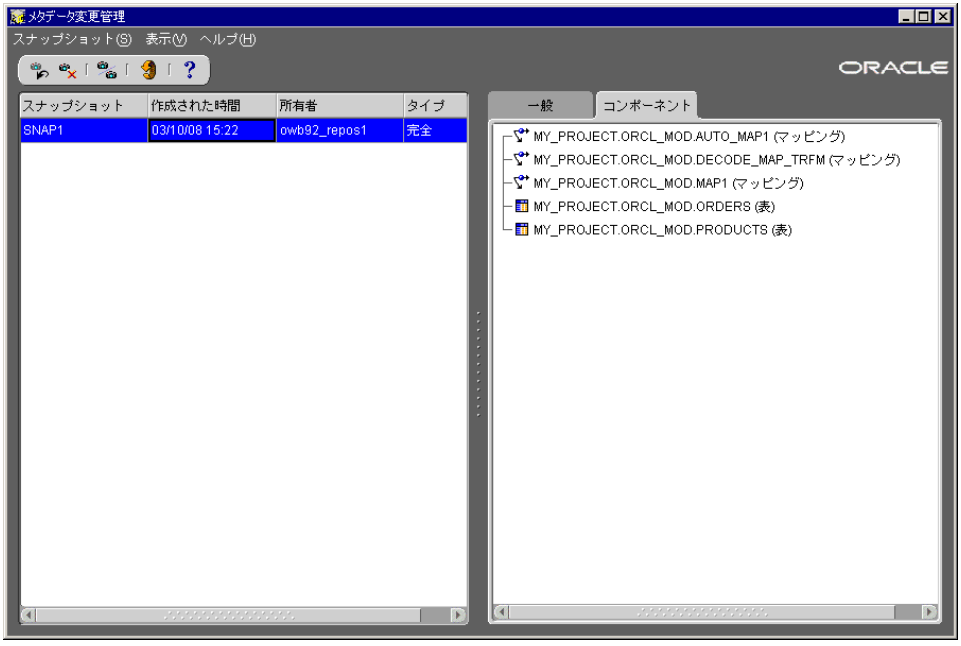
   「スナップショットへの追加 - 終了」ページが表示されます。
6. スナップショットに指定した選択内容を確認します。変更する場合は、「戻る」をクリックします。または、「終了」をクリックして、スナップショットを更新します。

   更新されたスナップショットは、「メタデータ変更管理」ウィンドウで確認できます。

「メタデータ変更管理」ウィンドウ

Warehouse Builder では、「メタデータ変更管理」ウィンドウからスナップショットを管理できます。このウィンドウを開くには、Warehouse Builder コンソールから、「プロジェクト」→「変更管理」を選択します。図 16-1 は、「メタデータ変更管理」ウィンドウを示しています。

図 16-1「メタデータ変更管理」ウィンドウ



このウィンドウは 2 つの列で構成されています。左側の列には、リポジトリで使用できるスナップショットと、それらの作成日時、所有者、タイプが表示されます。任意の列ヘッダーをクリックすると、その列を基準としてスナップショットのリストをソートできます。

右側の列は、「一般」、「コンポーネント」の 2 つのタブで構成されています。「一般」タブには、選択したスナップショットの名前、タイプおよび説明が表示されます。「コンポーネント」タブには、選択したスナップショットに含まれるすべてのオブジェクトが、ナビゲーション・ツリー形式で表示されます。このツリーのルートは、該当するスナップショットに含まれているコンポーネントによって異なります。たとえば、スナップショット内の最上位レベルのオブジェクトがウェアハウス・モジュールである場合は、これがルートになります。スナップショットにプロジェクト全体が含まれている場合は、プロジェクトがルートになります。スナップショットに 3 つの表のみが含まれている場合は、これらの表がツリーのルートになります。

「メタデータ変更管理」ウィンドウから、次のアクティビティを実行できます。

- 「スナップショットの比較」（16-9 ページ）
- 「スナップショットのリストア」（16-12 ページ）
- 「スナップショットのリストア」（16-12 ページ）
- 「スナップショットの削除」（16-14 ページ）

スナップショットの比較

Warehouse Builder では、ユニバーサル・オブジェクト ID（UOID）をすべてのコンポーネントの比較基準として使用し、2 つのスナップショットの比較またはスナップショットと現在のリポジトリ・オブジェクトとの比較ができます。

2 つのスナップショットを比較する手順は次のとおりです。

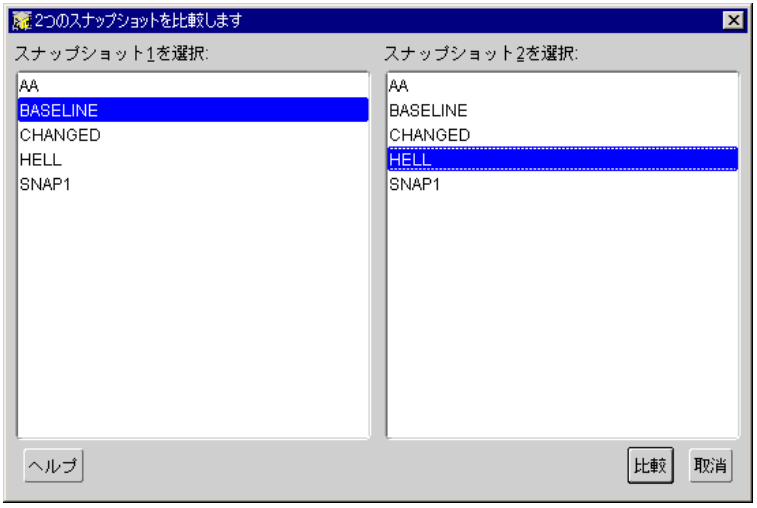
1. 次のいずれかの方法を使用します。

   「メタデータ変更管理」ウィンドウから、「スナップショット」→「比較」を選択します。

   または、「メタデータ変更管理」ウィンドウに表示されるスナップショットの一覧で比較対象の 2 つのスナップショットを選択してから右クリックし、「比較」を選択します。

   図 16-2 のような、「2 つのスナップショットを比較します」ダイアログが表示されます。比較対象の 2 つのスナップショットが選択済の場合、このダイアログのそれぞれの列に、その 2 つが選択された状態で表示されます。

図 16-2「2 つのスナップショットを比較します」ダイアログ



2. 「スナップショット 1 を選択」列と「スナップショット 2 を選択」列から比較対象のスナップショットを選択して、「比較」をクリックします。

   Warehouse Builder の「スナップショットの比較」ダイアログに、比較結果が表示されます。詳細は、16-11 ページの「「スナップショットの比較」ダイアログ」を参照してください。

現在のリポジトリ・オブジェクトとスナップショット・コンポーネントを比較する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーから、比較対象のオブジェクトを選択します。
2. 選択したオブジェクトを右クリックし、「スナップショットとの比較」を選択します。

   「スナップショット選択」ダイアログが表示されます。このダイアログには、該当するオブジェクトのバージョンを含むすべてのスナップショットが表示されます。
3. 現在のリポジトリ・オブジェクトと比較するスナップショットを選択して、「OK」をクリックします。

   「スナップショットの比較」ダイアログに比較結果が表示されます。詳細は、16-11 ページの「「スナップショットの比較」ダイアログ」を参照してください。リポジトリ・オブジェクトが変更されていない場合、Warehouse Builder に、変更がないことを示すメッセージが表示されます。

「スナップショットの比較」ダイアログ

2 つのスナップショットを比較する場合や、スナップショット・コンポーネントをリポジトリ・オブジェクトと比較する場合は、結果が「スナップショットの比較」ダイアログに表示されます。

> **注意：** 2 つのスナップショットが同一である場合、Warehouse Builder では、2 つのスナップショットが同一であるというメッセージが表示されます。この場合、「スナップショットの比較」ダイアログは表示されません。

一番上の「表示」フィルタ・オプションを使用すると、変更されたオブジェクトのみを表示できます。また、「すべてのオブジェクト」を選択すると、変更されているかどうかにかかわらず、スナップショット内のすべてのオブジェクトを表示できます。

「スナップショットの比較」ダイアログは 2 つの列で構成されています。左側の列には、比較結果がナビゲーション・ツリー形式で表示されます。ツリーのアイコンは、表 16-2 にあるように、スナップショット・コンポーネントのステータスを示しています。

表 16-2「スナップショットの比較」のアイコン

アイコン	ステータス
	スナップショット 1 とスナップショット 2 の両方にコンポーネントが含まれています。変更はありません。
	コンポーネントはスナップショット 1 にのみあります。
	コンポーネントはスナップショット 2 にのみあります。
	オブジェクトは両方のスナップショット内にありますが、変更されています。

右側の列は、「一般」、「プロパティ」、「リンク」の 3 つのタブで構成されています。これらのタブには次の情報が表示されます。

一般 : 比較結果の概要が表示されます。左側の列のナビゲーション・ツリーからオブジェクト名を選択すると、その物理名、ビジネス名およびタイプが表示されます。この列には、オブジェクトのプロパティ、関連する子コンポーネントまたはリンクに変更があるかどうかも表示されます。

プロパティ : オブジェクトのプロパティが変更されているかどうか、またはオブジェクトが両方のスナップショットに含まれるか、または一方にのみ含まれるかが表示されます。

リンク : オブジェクトのリンクや関連付けに変更があるかどうかが表示されます。「差分」フィールドの下のアイコンは、表 16-2 にあるように、オブジェクトのステータスを示しています。

比較結果を XML ファイルとして保存するには、「別名で保存」をクリックします。または、「閉じる」をクリックして、ダイアログを閉じます。

考慮事項

スナップショットを比較する際には、Warehouse Builder では、各オブジェクトの一意のオブジェクト識別子（UOID）を使用して照合が行われます。オブジェクトを削除してから、同じメタデータを使用して再作成する場合、削除済オブジェクトと同じように見えても、オブジェクトの UOID は異なります。オブジェクトは再作成されているため、このオブジェクトを前のバージョンのオブジェクトと比較すると、すべてのメタデータが変更されているという結果になります。そのため、変更されたメタデータだけでなく、変更されていないメタデータも、比較結果に表示されます。

Warehouse Builder では、UOID を使用してノードが照合されるため、MDL インポートまたはインテリジェンス・オブジェクトの導出中にオブジェクトが削除および再作成されると、結果は異なります。これは、MDL インポート中に IGNORE UOID = TRUE を実行する場合でも同じです。

スナップショットのリストア

スナップショットをリストアすると、リポジトリ内にあるオブジェクトの現在の定義が、そのオブジェクトのスナップショット・イメージで置換されます。プロジェクト、コレクションまたはフォルダのカスケード・スナップショットを完全スナップショットからリストアする場合、リストアする子オブジェクトを選択できます。特定の子オブジェクトを選択することによって、スナップショットを完全にリストアするか、またはその一部のみをリストアするかを指定できます。

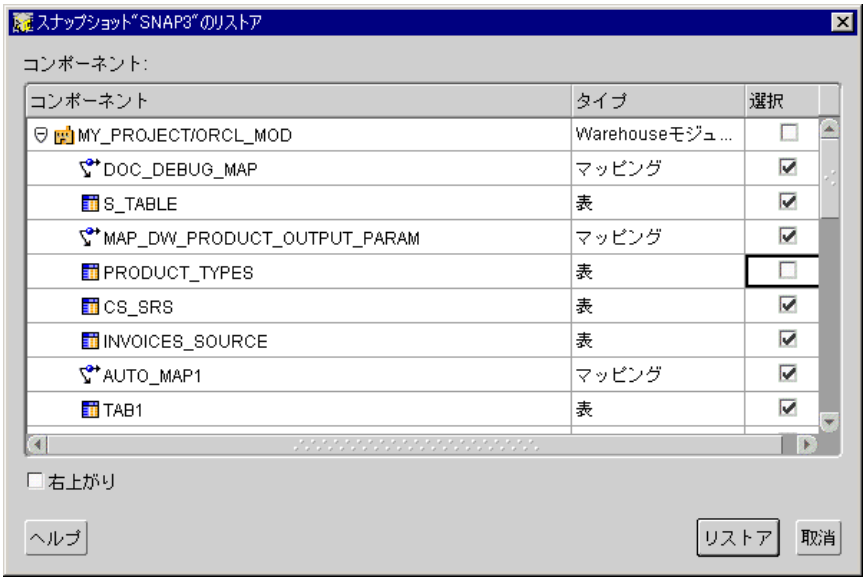
スナップショットをリストアする手順は次のとおりです。

1. 「メタデータ変更管理」ウィンドウを開いて、リストアするスナップショットを選択します。
2. そのスナップショットを右クリックして、ポップアップ・メニューから「リストア」を選択します。または、「スナップショット」メニューから「リストア」を選択します。

   作業をコミットしていない場合は、スナップショットをリストアする前にコミットすることを求める画面が表示されます。

   図 16-3 のような、「スナップショットのリストア」ダイアログが表示されます。

図 16-3「スナップショットのリストア」ダイアログ



このダイアログでは、リストアするコンポーネントをスナップショットから選択できます。デフォルトでは、「リストア」をクリックすると、すべてのコンポーネントがリストアされます。特定の子オブジェクトを選択して、スナップショットの一部のみをリストアすることもできます。

フォルダをリストアすると、その子オブジェクトもすべてリストアされます。たとえば、ウェアハウス・モジュールをリストアすると、そのモジュールに含まれるすべてのオブジェクトも自動的にリストアされます。このウェアハウス・モジュール内の選択したオブジェクトのみをリストアするには、まずウェアハウス・モジュールまたはその他のフォルダのチェック・ボックスの選択を解除してから、そのフォルダ内のオブジェクトを個別に選択してリストアする必要があります。

「リストア」ダイアログは次の列およびオプションで構成されています。

コンポーネント : スナップショットに含まれている Warehouse Builder コンポーネントが表示されます。

タイプ : Warehouse Builder オブジェクトのタイプを指定します。たとえば、ウェアハウス・モジュール、表、ディメンション、マッピング、ビューなどです。

選択 : この列で、リストアするコンポーネントのチェック・ボックスを選択します。

右上がり : コンポーネントの親がリポジトリ内に存在しない場合に、このオプションでリストアを制御します。たとえば、元はプロジェクト MY_PROJECT の下のモジュール MY_WH にあった表 T1 をリストアするとします。「右上がり」チェック・ボックスを選択すると、T1 を含むダミーの MY_PROJECT および MY_WH が作成されます。「右上がり」を選択しないと、この表の親フォルダが検索できないため、リストアは失敗します。

「リストア」をクリックして、選択したスナップショット・オブジェクトをリポジトリにリストアします。リストアの終了後、現在開いているエディタがすべて閉じられるという警告が表示されます。「はい」をクリックして続行し、リストアを完了します。

スナップショットの削除

スナップショットまたはスナップショット内のコンポーネントを、「メタデータ変更管理」ウィンドウから削除できます。

スナップショットを削除する手順は次のとおりです。

1. 「メタデータ変更管理」ウィンドウを開いて、削除するスナップショットを選択します。
2. そのスナップショットを右クリックして、ポップアップ・メニューから「削除」を選択します。または、「スナップショット」メニューから「削除」を選択します。

   「削除確認」ダイアログが表示されます。
3. 「はい」をクリックして、選択したスナップショットをリポジトリから削除します。

スナップショット内のコンポーネントを削除する手順は次のとおりです。

1. 「メタデータ変更管理」ウィンドウを開いて、削除するコンポーネントが含まれているスナップショットを選択します。
2. 「メタデータ変更管理」ウィンドウの右側の列から、「コンポーネント」タブを選択します。

   右側の列に、スナップショットのコンポーネントが表示されます。
3. 削除するコンポーネントを右クリックし、ポップアップ・メニューから「削除」を選択します。または、「スナップショット」メニューから「削除」を選択します。

   「削除確認」ダイアログが表示されます。
4. 「はい」をクリックして、選択したコンポーネントをスナップショットから削除します。

メタデータ・スナップショットの使用方法

この項では、OMB Plus でのスナップショットの使用方法の概要を説明します。OMB Plus でスナップショットを作成および管理する際に使用するコマンドおよび引数の詳細は、『Oracle Warehouse Builder スクリプト・リファレンス』を参照してください。

履歴管理

履歴管理には、スナップショットの作成、削除、リストアおよび変更が含まれます。

スナップショットの作成

スナップショットを作成するには、次の構文を使用します。

```
OMBCREATE SNAPSHOT 'SNAPSHOTNAME' SET PROPERTIES
(DESCRIPTION,SNAPSHOTTYPE) VALUES (Your description value, 'FULL')

ADD FIRST_CLASS_OBJECT_TYPE FULLY_QUALIFIED_FIRST_CLASS_OBJECT_NAME
…
```

例:

```
OMBCREATE SNAPSHOT S1 SET PROPERTIES (DESCRIPTION, SNAPSHOTTYPE)
VALUES ('Sample Snapshot', Y ,Y)

ADD TABLE 'MY_PROJECT/WH/EMP'

ADD TABLE 'MY_PROJECT/WH/DEPT'

ADD DIMENSION 'MY_PROJECT/WH/DIM1'

ADD MAPPING 'AnotherProject/WH1/MAP1'
```

スナップショットの削除

スナップショットを削除するには、次の構文を使用します。

```
OMBDROP SNAPSHOT 'SNAPSHOTNAME'
```

例:

```
OMBDROP SNAPSHOT 'SS1'
```

スナップショットの変更

既存のスナップショットを直接変更し、必要に応じて履歴を変更できます。また、ALTER コマンドを使用して、ルート以外のスナップショットを削除することもできます（ルート・スナップショットの詳細は、16-3 ページの「カスケード・スナップショットとカスケードなしスナップショット」を参照）。

スナップショットを変更するには、次の構文を使用します。

```
OMBALTER SNAPSHOT 'SNAPSHOTNAME'

( SET PROPERTIES (SNAPSHOTTYPE) VALUES ('FULL'/'SIGNATURE')

)?

( ADD FIRST_CLASS_OBJECT_TYPE RELATIVE_FIRST_CLASS_OBJECT_NAME

 | DELETE FIRST_CLASS_OBJECT_TYPE RELATIVE_FIRST_CLASS_OBJECT_NAME

 | MODIFY FIRST_CLASS_OBJECT_TYPE RELATIVE_FIRST_CLASS_OBJECT_NAME

)+
```

例：

```
OMBALTER SNAPSHOT S1

ADD TABLE 'MY_PROJECT/MOD1/T1'

ADD TABLE 'MY_PROJECT/MOD1/T2'

DELETE DIMENSION 'MY_PROJECT/WH/DIM1'
```

このコマンドでは、2 つの表が追加され、既存のディメンションがスナップショットの定義から削除されます。

スナップショットのリストア

スナップショットをリストアすると、リポジトリ内にあるオブジェクトの現在の定義が、そのオブジェクトのスナップショット・イメージで置換されます。プロジェクト、コレクションまたはフォルダのカスケード・スナップショットを完全スナップショットからリストアする場合、リストアする子オブジェクトを選択できます。特定の子オブジェクトを選択することによって、スナップショットを完全にリストアするか、またはその一部のみをリストアするかを指定できます。

スナップショット全体のリストア スナップショット全体をリストアするには、次の構文を使用します。

```
OMBRESTORE SNAPSHOT 'SNAPSHOTNAME'
```

例:

```
OMBRESTORE SNAPSHOT 'SS1'
```

このコマンドでは、スナップショット内のすべてのオブジェクトがリストアされます。一部のインスタンスのリストアが失敗すると、操作全体が失敗し、エラー・メッセージが表示されます。リストアが失敗するということは、スナップショット内にあるすべてのオブジェクトの親が現在のリポジトリに存在しないということです。

スナップショットから選択したオブジェクトのリストア スナップショットから選択したオブジェクトをリストアするには、次の構文を使用します。

```
OMBRESTORE SNAPSHOT 'SNAPSHOTNAME' FOR FIRST_CLASS_OBJECT_TYPE
'RELATIVE_FIRST_CLASS_OBJECT_NAME'
```

例:

```
OMBRESTORE SNAPSHOT 'SS1' FOR TABLE 'MY_PROJECT/WH/EMP' MAPPING
'AnotherProject/WH1/MAP1'
```

このコマンドでは、表 EMP およびマッピング MAP1 の現在のリポジトリのメタデータ定義が、スナップショット SS1 で定義されている表およびマッピングで置換されます。

孤立したスナップショット スナップショットがあるオブジェクトを削除しても、そのオブジェクトのスナップショットは削除されません。したがって、オブジェクトを削除すると、スナップショットが孤立する場合、つまり所属するオブジェクトがない状態でリポジトリに残る場合があります。孤立したスナップショットをリストアしようとすると、Warehouse Builder でエラー・メッセージが表示されます。

孤立したスナップショットの親をリストアするには、次の構文を使用します。

```
OMBRESTORE SNAPSHOT 'SNAPSHOT_NAME' CASCADE UP
```

スナップショットでは Warehouse Builder のトランザクション・メカニズムが使用されるため、ロールバックが原因でスナップショットが孤立することはありません。新規のオブジェクトとそのスナップショットをコミットしないで作成した後、変更をロールバックすると、オブジェクトとスナップショットの両方が削除されます。

スナップショットのエクスポートおよびインポート

OMB Plus スクリプト・インタフェースおよび Warehouse Builder ユーザー・インタフェースを使用する場合、エクスポートおよびインポートできるのは完全スナップショットのみです。完全スナップショットのエクスポートおよびインポートは、スナップショットを別のリポジトリに移動するため、または今後のアップグレードに使用するために行います。完全スナップショットをエクスポートすると、スナップショットが Metadata Loader の `.mdl` ファイルに変換されます。完全スナップショットをインポートすると、`.mdl` ファイルがスナップショットに変換され、レジストリ内でスナップショットの記憶域として割り当てられている表領域にスナップショットが格納されます。

変更管理

変更管理には、スナップショットの一覧表示、情報の取得および比較が含まれます。

スナップショットの一覧表示

指定したリポジトリ内の使用可能なすべてのスナップショットを一覧表示したり、リポジトリ内の特定のオブジェクトに関連付けられているスナップショットを一覧表示することができます。

リポジトリの全スナップショットの一覧表示 任意のリポジトリについて、すべてのスナップショットを一覧表示するには、次の構文を使用します。

```
OMBLIST SNAPSHOTS regexp
```

例:

```
OMBLIST SNAPSHOTS
```

このコマンドでは、該当するリポジトリについて、すべてのスナップショットが一覧表示されます。

オブジェクトに関連付けられているスナップショットの一覧表示 特定のオブジェクトについて、既存のスナップショットを一覧表示するには、次の構文を使用します。

```
OMBLIST SNAPSHOTS FOR FIRST_CLASS_OBJECT_TYPE 'RELATIVE_FIRST_CLASS_
OBJECT_NAME'
```

例:

```
OMBLIST SNAPSHOTS FOR TABLE 'EMP'
```

このコマンドでは、現在の表 EMP に関連付けられているすべてのスナップショットの名前が表示されます。

スナップショットに関する情報の取得

スナップショットに関する情報を取得するには、次の構文を使用します。

```
OMBRETRIEVE SNAPSHOT 'SNAPSHOTNAME' GET
( PROPERTIES (TIMESTAMP|DESCRIPTION|SNAPSHOTTYPE)
| CONTENTS (ALL | MAPPINGS | TABLES |…)
 ( START WITH 'FIRST_CLASS_OBJECT_NAME')?(CASCADE | NO CASCADE)? ) )
```

例:

- `OMBRETRIEVE SNAPSHOT S1 GET PROPERTIES (DESCRIPTION)`

  このコマンドでは、スナップショット S1 の説明が表示されます。

```
Sample Snapshot
```

- `OMBRETRIEVE SNAPSHOT S1 GET CONTENTS ALL`

  次のように表示されます。

```
TABLE 'MY_PROJECT/WH/EMP'
TABLE 'MY_PROJECT/WH/DEPT'
DIMENSION 'MY_PROJECT/WH/DIM1'
MAPPING 'AnotherProject/WH1/MAP1'
```

- `OMBRETRIEVE SNAPSHOT S1 GET CONTENTS TABLES`

  次のように表示されます。

```
TABLE 'MY_PROJECT/WH/EMP'
TABLE 'MY_PROJECT/WH/DEPT'
```

スナップショットの比較

OMB Plus ユーティリティを使用すると、2 つのスナップショットを相互に比較したり、またはスナップショットと現在のリポジトリ・オブジェクトを比較することができます。Warehouse Builder では、ユニバーサル・オブジェクト ID（UOID）が、すべての比較の基準として使用されます。

2 つのスナップショットを比較する場合や、スナップショットをオブジェクトと比較する場合は、一方をソース、もう一方をターゲットとして指定します。たとえば、特定のプロパティをオブジェクトにリストアする目的で、スナップショットと現在のモジュール・オブジェクトを比較する場合は、スナップショットがソース、オブジェクトがターゲットとなります。

次の比較レポート・フィルタから選択できます。

- **Objects that are found in the source only:** この方法では、ソースに存在し、ターゲットに存在しないオブジェクトのみが識別されます。
- **Objects that are found in the target only:** この方法では、ターゲット・オブジェクトに存在し、ソース・オブジェクトに存在しないオブジェクトのみが識別されます。
- **Changed objects found in both source and target:** この方法では、ソースとターゲットの両方に存在するものの、それぞれの定義が異なるオブジェクトが識別されます。
- **Unchanged objects found in both source and target:** この方法では、ソースとターゲットで同一であるオブジェクトが識別されます。
- **すべてのオブジェクト :** この方法では、ターゲットに存在するすべてのオブジェクトと、ソースにのみ存在するオブジェクトが識別されます。

2 つのスナップショットを相互に比較したり、スナップショットと現在のリポジトリ・オブジェクトを比較するときに、比較レポートを生成できます。比較レポートは XML 形式で生成され、新規オブジェクト、なくなったオブジェクトおよび変更されたオブジェクトを識別できます。スナップショットの比較レポートは、`mcmview.bat` コマンドライン・ユーティリティまたは OMB Plus から生成できます。

OMB Plus でのスナップショットの比較 OMB Plus からスナップショットの比較レポートを生成するには、生成された XML 用に定義されている、パッケージ済の XML Schema を使用します。詳細は、『Oracle Warehouse Builder スクリプト・リファレンス』を参照してください。

OMB Plus で 2 つのスナップショットを比較するには、次の構文を使用します。

```
OMBCOMPARE SNAPSHOT 'SRC_SNAPSHOTNAME' WITH

( SNAPSHOT 'TGT_SNAPSHOTNAME' | CURRENT )

FOR FIRST_CLASS_OBJECT_TYPE 'RELATIVE_FIRST_CLASS_OBJECT_NAME'

SHOW (ALL | CREATED | DELETED | MODIFIED | UNMODIFIED) OUTPUT TO
'XMLFILENAME'
```

例 :

- `OMBCOMPARE SNAPSHOT 'SS1' WITH SNAPSHOT 'SS2' FOR TABLE 'EMP' SHOW ONLY DELETED OUTPUT TO 'c:¥diffresults¥emp_diff.xml'`
- `OMBCOMPARE SNAPSHOT 'SS1' WITH CURRENT FOR TABLE 'MY_PROJECT/WH/EMP' SHOW ONLY DELETED OUTPUT TO 'c:¥diffresults¥emp_diff.xml'`

コマンドライン・ユーティリティ MCMVIEW.BAT でのスナップショットの比較 コマンドライン・ユーティリティ mcmview.bat も、スナップショットの比較レポートの生成用にパッケージされています。このコマンドライン・ユーティリティでは、レポートを HTML 形式で読み取り、表示するために、XML が準備されます。このユーティリティは owb/bin/win32 ディレクトリにあり、OWB 比較 XML ファイル名をパラメータとして渡すことによって実行します。

たとえば、コマンド mcmview c:¥temp¥tables_021111.xml を実行すると、tables_021111.xml.html という名前のファイルが win32 ディレクトリに作成され、接尾辞 .html が XML ファイル名に追加されます。

スナップショットの使用方法の提案

MDL メタデータ・インポート・ユーティリティや影響分析など、Warehouse Builder の他の機能と組み合せて使用すると、スナップショットはメタデータ管理に役立ちます。また、スナップショットは、Warehouse Builder のユーザー固有のニーズを満たすように設計されています。

スナップショットを Warehouse Builder の他の機能と組み合せた使用方法

スナップショットを MDL メタデータ・インポートと組み合せて使用すると、インポートによって現在のメタデータ・リポジトリに生じる影響を特定できます。リポジトリのスナップショットの比較に基づいて、メタデータ・インポートを実行するアクションを、より詳細に選択します。メタデータ・インポートの前に影響分析を使用すると、インポート中に予定されている変更によって影響を受ける依存オブジェクトを特定できます。また、配布中にスナップショットを使用して、再配布が必要なオブジェクトを特定することもできます。

データ・ウェアハウスの設計者およびマネージャ用のスナップショット・ビュー

メタデータ・スナップショットの粒度は、Warehouse Builder のユーザー固有のニーズにあわせて変わります。ウェアハウス設計者は、スナップショットを使用して、マッピング、エンティティ、プロセス・フローなど個々のコンポーネントの内容を様々な時点で取得できます。一方、データ・ウェアハウス・マネージャは、スナップショットを使用して、ウェアハウス・モジュールまたはプロジェクト全体の履歴を検出または保持できます。スナップショットのビューには、OMB Plus を使用してアクセスできます。詳細は、『Oracle Warehouse Builder スクリプト・リファレンス』を参照してください。

「The Object to Snapshot view for designers」を使用すると、データ・ウェアハウス設計者は、作業のスナップショットを作成できます。

- たとえば、特定のマッピングまたはエンティティの操作中に、開発者はそのエンティティの完全スナップショットを作成できます。後で、その開発者は完全スナップショットを使用して、マッピングを以前の状態にリストアできます。

- または、特定のコンポーネントのシグネチャ・スナップショットを作成し、後にそのスナップショットを使用して、コンポーネントに発生した変更を識別することもできます。

「The Snapshot to Object view for managers」を使用すると、データ・ウェアハウス・マネージャは、メタデータの履歴を調査できます。

- マネージャは、このビューを使用して、リポジトリ内に作成されたスナップショットの監視、削除および比較を行うことができます。
- このビューを使用すると、管理者は、ウェアハウスの配布、過去の時点へのウェアハウスのリストアなど、大規模なアクションを実行できます。
- データ・ウェアハウス・マネージャは、このビューを使用して、古く不要なスナップショットを一括削除することもできます。
- マネージャは、特定のオブジェクトに関連付けられているすべてのスナップショットを 1 つのウィンドウに表示し、リポジトリの 1 つにまとまったビューを使用してスナップショットを管理できます。
- 最後に、このビューを使用すると、データ・ウェアハウス・マネージャは、新しいバージョンの Warehouse Builder ソフトウェアに合うように、スナップショット記憶域をアップグレードできます。

17

メタデータの参照およびレポート

Warehouse Builder Design Browser は、リポジトリに格納されているメタデータを参照し、そのメタデータに関するレポートを表示するために使用します。メタデータ・レポートには、Warehouse Builder Design Client からアクセスできます。このクライアントは、Internet Application Server 管理者が作成および維持しているブラウザを使用して開きます。

この章では、次のトピックについて説明します。

Warehouse Builder Design Browser について

Warehouse Builder Design Browser を使用すると、リポジトリのすべてのメタデータ・オブジェクトや、それらのオブジェクト間の関係に関するレポートを、表示および生成できます。

Warehouse Builder Design Browser を使用する場合、2 つのオプションがあります。Warehouse Builder Design Client とともにインストールされるクライアント・バージョン、または Portal バージョン、つまり Oracle Database Application Server に統合されている Warehouse Builder Design Browser のバージョンを使用できます。使用可能なバージョンは、Warehouse Builder 管理者がインストールの際に行った手順によって異なります。Warehouse Builder Design Browser の 2 つのバージョンの詳細は、『Oracle Warehouse Builder 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。

Warehouse Builder Design Browser は複数のポートレットで構成されており、これらを追加してポータル・ページをカスタマイズできます。各ポートレットを複数追加することもできます。たとえば、2 つのレポートのポートレットを追加し、異なるリポジトリで各レポートを実行できます。

メタデータに関するレポートの表示および生成で使用するポートレットには、次のものがあります。

- ナビゲーション・ポートレット（17-3 ページ）
- お気に入りポートレット（17-4 ページ）
- レポート・ポートレット（17-5 ページ）

ナビゲーション・ポートレット

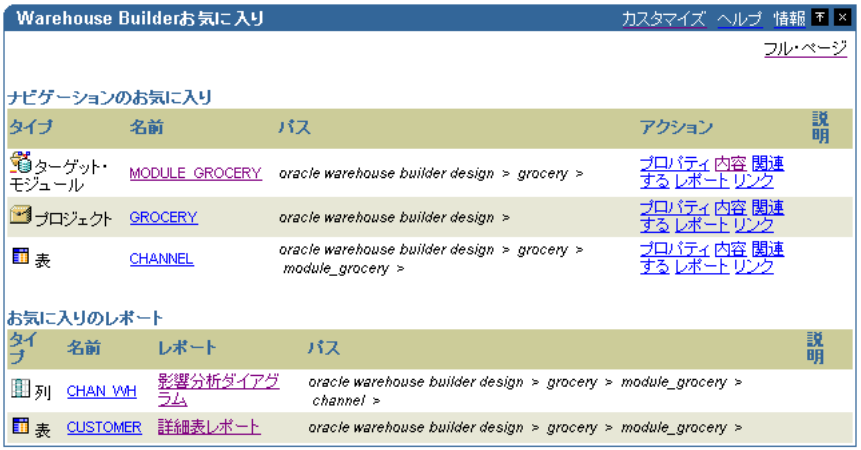
ナビゲーション・ポートレットでは、図 17-1 に示すように、現在のユーザーがアクセスできるリポジトリを参照できます。ナビゲーション・ポートレットの機能は、ポートレットの領域内に表示されることを除けば、Navigator のメイン・ページの機能と同じです。また、「フル・ページ」リンクをクリックすると、Navigator をフル・ページ・モードで表示できます。

図 17-1 ナビゲーション・ポートレット

お気に入りポートレット

お気に入りポートレットでは、図 17-2 に示すように、「Oracle Portal」ホームページからお気に入りの「ナビゲータ」ページやレポートにアクセスできます。このポートレットには、デフォルトのお気に入りは設定されていません。Warehouse Builder Navigator の「お気に入りに追加」リンクを使用して、お気に入りに追加する「ナビゲータ」ページやレポートを選択します。また、「フル・ページ」リンクをクリックすると、「お気に入り」をフル・ページ・モードで表示できます。

図 17-2 お気に入りポートレット



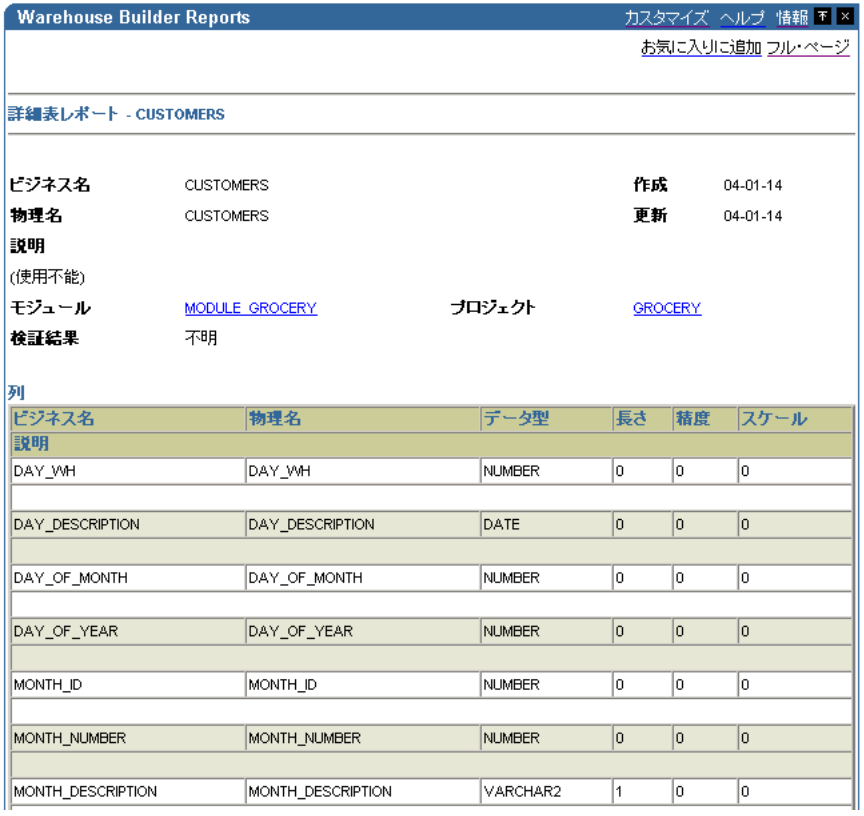
お気に入りの「ナビゲータ」ページを表示するには、「名前」列の下にある項目名を選択します。また、アクション・リンクを使用して、プロパティ、内容、関連項目、レポート、または項目に関連するリンクを表示することもできます。

お気に入りのレポートを表示するには、「レポート」列の下にあるレポート名を選択します。「名前」列の下の項目名をクリックすると、その項目の「ナビゲータ」ページが表示されます。

レポート・ポートレット

レポート・ポートレットでは、図 17-3 に示すように、「Oracle Portal」ホームページに Warehouse Builder レポートが表示されます。このポートレットには、デフォルトのレポートは設定されていません。Report を表示するには、レポートをお気に入りに追加し、このポートレットの「カスタマイズ」リンクを使用してリンクに追加します。

図 17-3 レポート・ポートレット



注意：「Oracle Portal」ホームページにレポート・ポートレットがある場合、ホームページを再ロードするたびにリフレッシュされます。このため、ホームページの表示が遅くなることがあります。

Warehouse Builder Design Browser の使用方法

Warehouse Builder Browser は、メタデータ・レポートを参照し、特定の情報を検索するために使用します。

Warehouse Builder Design Browser の開始

Design Browser のクライアントまたは Portal バージョンは、Warehouse Builder Design Client 内から開始できます。どちらのバージョンも、Warehouse Builder Design Client の外部から開始できます。

Warehouse Builder Design Client から Design Browser を開始する手順は次のとおりです。

1. メニューから「表示」→「レポート」を選択します。

   インターネット・ブラウザが起動され、Warehouse BuilderBrowser Navigator が表示されます。

   インターネット・ブラウザにエラー・メッセージが表示される場合は、Design Browser の構成が間違っている可能性があります。Design Browser の構成方法の詳細は、『Oracle Warehouse Builder 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。

2. 「ナビゲータ」ページを使用してメタデータ・レポートを定義します。

   「ナビゲータ」ページがルート・レベルで表示され、ユーザーのログイン・リポジトリのみが表示されます。「管理」ページはありません。すべての権限が与えられています。「依存性索引のリフレッシュ」リンクの下に、「系統」および「影響分析」のオプションが表示されます。「MLS」は使用できます。カスタム・レポートはありません。「お気に入り」は使用できます。

Design Browser のクライアント・バージョンを Warehouse Builder Design Client の外部から開始する手順は次のとおりです。

1. OC4J を開始します。

   Windows の場合 : デスクトップの「スタート」メニューから、「プログラム」→「Oracle - Warehouse Builder がインストールされている Oracle ホーム・ディレクトリ名」→「Warehouse Builder」→「Start OWB OC4J Instance」を選択します。MS-DOS ウィンドウが表示されます。

   UNIX の場合 : *ORACLE_HOME*/owb/bin/unix を検索し、startOwbbInst.sh を実行します。

   「Oracle Application Server Container for J2EE 10*g*（9.0.4.0.0）initialized」というメッセージが表示されます。

2. メッセージ・ウィンドウを最小化します。

3. Design Browser Client を開始します。

   Windows の場合 : デスクトップの「スタート」メニューから、「プログラム」→「Oracle - Warehouse Builder がインストールされている Oracle Home ディレクトリ名」→「Warehouse Builder」→「OWB Design Browser」を選択します。

   UNIX の場合 : *ORACLE_HOME*/owb/bin/unix を検索し、openDB.sh を実行します。

4. 次の情報を指定して、Design Browser Client にログインします。

   HTTP Server ホストのポート : デフォルトは 7778 です。以前のバージョンの Oracle HTTP Server のデフォルトは 80 です。

   DAD: 参照する Warehouse Builder Design Repository の名前。

   リポジトリのユーザー名とパスワード : Design Repository にインストール中に割り当てられるユーザー名とパスワード。Warehouse Builder のユーザー名とパスワードとは異なる場合があります。

Design Browser の Portal バージョンの開始

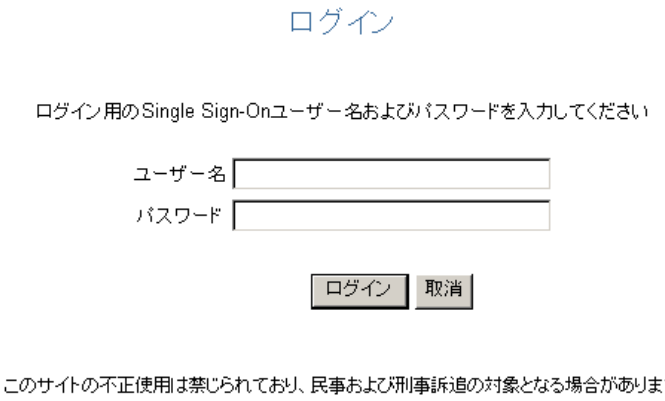
Oracle Database Application Server に統合された Warehouse Builder Design Browser を開始するには、Oracle Portal のインターネット・アドレスと、Oracle Portal のユーザー・アカウントが必要です。必要な情報の詳細は、Internet Application Server（iAS）管理者に問い合せてください。

Warehouse Builder Browser を開始する手順は次のとおりです。

1. インターネット・ブラウザを開き、URL を Oracle Portal のアドレスに設定します。
2. 「ログイン」をクリックします。

   Warehouse Builder Browser では、図 17-4 に示すように、標準の Oracle Portal ログイン画面が使用されます。Warehouse Builder Browser 固有の画面ではありません。

図 17-4「Single Sign-On」ページ



3. Warehouse Builder Browser ポートレットへのアクセス権を持つユーザー名とパスワードを使用して、Oracle Portal にログオンします。ユーザー・アカウントの割当ては、Internet Application Server（iAS）管理者が行います。ユーザーには、それぞれ固有のユーザー名とパスワードが割り当てられます。

   ログイン後、デフォルトの「Oracle Portal」ホームページが表示されます。Warehouse Builder Browser のポートレットを表示するには、「Oracle Portal」ホームページなどのページに追加してから、そのページを表示する必要があります。ポートレットを追加するには、「ページの編集」オプションを使用します。ポートレットの追加方法については、18-6 ページの「ポートレットの追加」を参照してください。

   図 17-5 は、Warehouse Builder のポートレットを表示する Portal ページの例を示しています。

図 17-5 Warehouse Builder ポートレットを表示する Portal ページの例

Warehouse Builder Browser のナビゲーション

Warehouse Builder Browser を開始すると、図 17-6 に示すように、「ナビゲータ」ページが最初に表示されます。このページは、次の部分で構成されています。

- ヘッダー・セクション
- パス・セクション
- 「プロパティ」タブ、「内容」タブ、「関連する」タブ、「レポート」タブ、「リンク」タブで構成されるタブ・セクション。

図 17-6 Warehouse Builder Navigator

ヘッダー・セクション

Warehouse Builder Browser のすべてのページで、図 17-7 に示すようにヘッダーがページの最上部に表示されます。

図 17-7 Navigator ヘッダー

ヘッダーには、ページのタイトルと役に立つリンクが含まれています。タイトルの右側には、「ホーム」リンクがあります。タイトルの下には、「お気に入りに追加」、「お気に入りを参照」、「カスタマイズ」、「ログアウト」のリンクがあります。

パス・セクション

「ナビゲータ」ページの中央には、現在選択されているパスに関する情報が表示されます。

図 17-8 項目定義

パス・セクションには、次の項目が含まれます。

- **項目名 :** Warehouse BuilderBrowser の左側には、項目名が表示されます。これは、ナビゲート先の項目名です。図 17-8 の場合、これは GROCERY という名前のプロジェクトです。
- **Question マーク :** Warehouse BuilderBrowser の右端には、「ヘルプ」アイコンが表示されます。ヘルプ・リンクから、「Navigation Page」のオンライン・ヘルプが表示されます。
- **パス :** Warehouse Builder Browser の項目名の下には、パスが表示されます。「パス」は、項目のコンテキストを示します。階層内の項目や、各項目のタイプが表示されます。パスの項目をクリックすると、その項目を現在の項目にできます。図 17-8 の場合、パスは `Path: MyRepository: Repository > GROCERY:Project` です。
- **ロール :** Warehouse Builder Browser の右側にある「ヘルプ」アイコンの下には、現在のユーザーのロールが表示されます。図 17-8 の場合、ロールは `Warehouse User` です。

ボディ・セクション

Warehouse Builder Browser のすべてのページで、詳細情報はページの下部に表示されます。任意のタブを選択すると、タブに固有の詳細が表示されます。

「プロパティ」タブ

「プロパティ」タブには、図 17-9 に示すように、現在の項目のプロパティの名前 / 値ペアが表示されます。プロパティ名は、現在の項目のタイプによって異なります。

図 17-9「プロパティ」タブ

表 17-1 では、「プロパティ」タブに表示される列を説明します。

表 17-1 プロパティの列の値

列名	値
プロパティ名	Warehouse Builder Public View で定義したプロパティの名前。Public View の詳細は、18-28 ページの「カスタム・レポートの作成」を参照してください。
プロパティ値	現在の項目の Warehouse Builder Design Repository からのプロパティの値。

「内容」タブ

「内容」タブには、現在の項目に含まれる項目が一覧表示されます。たとえば、現在の項目が表である場合、「内容」タブには、列、キーおよび外部キーが表示されます。このタブは、Warehouse Builder Design Repository をドリルダウンするときに使用します。図 17-10 では、現在の項目は Warehouse Builder Design Repository と、リポジトリに含まれる項目のリストです。表 17-2 に、「内容」タブの列とその値の概要を示します。

図 17-10「内容」タブ

表 17-2 内容の列の値

列名	値
物理名	項目の名前。
タイプ	項目タイプを視覚的に表すアイコンと、項目タイプ。
アクション	現在の項目へのリンクとして名前を付けられたタブに移動する一連のリンク。 **プロパティ：**「プロパティ」を選択すると、その行の項目が現在の項目となり、その項目は「プロパティ」タブに表示されます。 **内容：**「内容」を選択すると、その行の項目が現在の項目となり、その項目は「内容」タブに表示されます。 **関連する：**「関連する」を選択すると、その行の項目が現在の項目となり、その項目は「関連する」タブに表示されます。 **レポート：**「レポート」を選択すると、その行の項目が現在の項目となり、その項目は「レポート」タブに表示されます。 **リンク：**「リンク」を選択すると、その行の項目が現在の項目となり、その項目は「リンク」タブに表示されます。

「関連する」タブ

「関連する」タブでは、図 17-11 に示すように、項目間の関係を表示できます。たとえば、現在の項目が表である場合、外部キーによって関連付けられた表を識別できます。このタブを使用して、リポジトリ中の項目をドリルダウンできます。項目の 1 つを現在の項目にすると、パスの表示は新しい項目のパスに切り替わります。

図 17-11「関連する」タブ

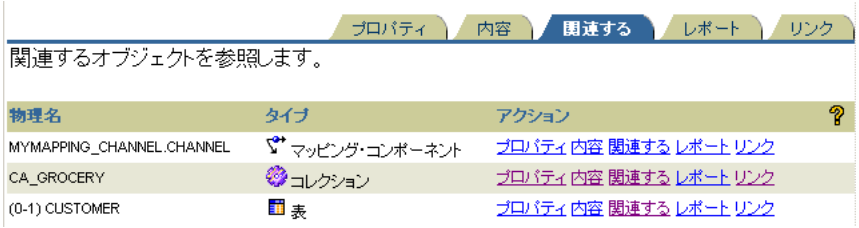


表 17-3 に、「関連する」タブの列と、その値の簡単な説明を示します。

表 17-3「関連する」の列の値

列名	値
名前	項目の名前。
タイプ	項目タイプを視覚的に表すアイコンと、項目タイプ名。
アクション	現在の項目へのリンクとして名前を付けられたタブに移動する一連のリンク。 **プロパティ：**「プロパティ」を選択すると、その行の項目が現在の項目となり、その項目は「プロパティ」タブに表示されます。 **内容：**「内容」を選択すると、その行の項目が現在の項目となり、その項目は「内容」タブに表示されます。 **関連する：**「関連する」を選択すると、その行の項目が現在の項目となり、その項目は「関連する」タブに表示されます。 **レポート：**「レポート」を選択すると、その行の項目が現在の項目となり、その項目は「レポート」タブに表示されます。 **リンク：**「リンク」を選択すると、その行の項目が現在の項目となり、その項目は「リンク」タブに表示されます。

「レポート」タブ

「レポート」タブには、図 17-12 に示すように、現在の項目に使用できるレポートが表示されます。「レポート」タブから表示できるレポートは、ユーザー・ロールによって異なります。Warehouse ユーザーは、Warehouse エンジニアとは異なるレポート・セットを閲覧できます。表 17-4 に、「レポート」タブの列とその値の概要を示します。

図 17-12「レポート」タブ

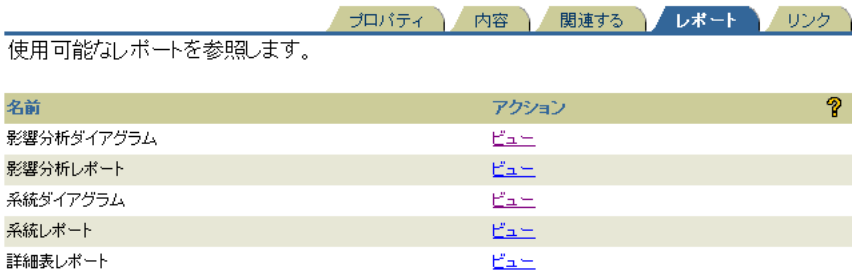


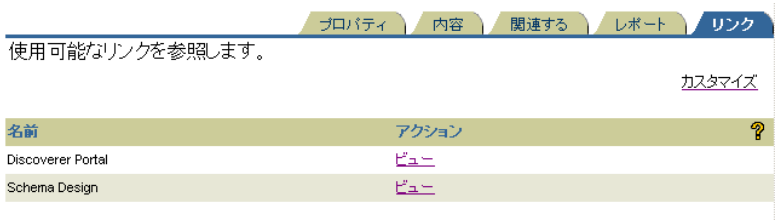
表 17-4 レポートの列の値

列名	値
名前	レポートの名前。
アクション	**ビュー:** レポートを表示します。

「リンク」タブ

「リンク」タブでは、図 17-13 に示すように、他の情報ソースをオブジェクトまたはオブジェクトのタイプに関連付けることができます。URL で記述できれば、たとえば、Oracle Portal コンテンツ領域や他の Web ページに格納されたドキュメントなど、あらゆる情報にリンクできます。

図 17-13「リンク」タブ



「リンク」タブの「カスタマイズ」リンクを使用すると、リンクの編集、削除または追加ができます。

> **注意：** 編集または削除できるのは、作成済のリンクのみです。他のリポジトリ・ユーザーが作成したリンクを編集または削除することはできません。

リンクを編集する手順は次のとおりです。

1. リンクが関連付けられたオブジェクトまたはオブジェクト・タイプにナビゲートし、「リンク」タブを選択します。
2. 「リンク」タブから「カスタマイズ」を選択します。

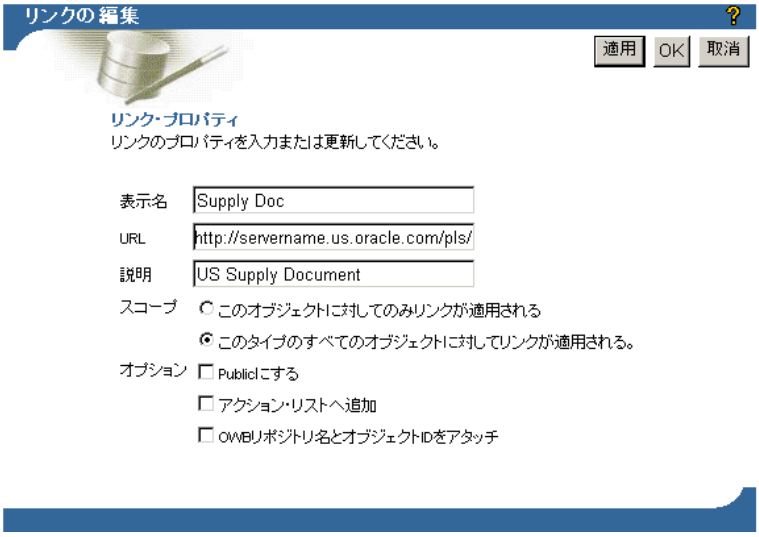
   図 17-14 に示すように、「Warehouse Builder リンクの参照」ページに「既存のリンク」が表示されます。

図 17-14「Warehouse Builder リンクの参照」ページ

3. 編集するリンクで「編集」を選択します。

   図 17-15 に示すように、「リンクの編集」ページに現在のリンク設定が表示されます。

図 17-15「リンクの編集」ページ



4. リンク・プロパティを編集し、「OK」をクリックします。

リンクを削除する手順は次のとおりです。

1. リンクが関連付けられたオブジェクトまたはオブジェクト・タイプにナビゲートし、「リンク」タブを選択します。
2. 「リンク」タブから「カスタマイズ」を選択します。

   「Warehouse Builder リンクの参照」ページに「既存のリンク」が表示されます。
3. 削除するリンクに「リンクから削除」を選択します。

   リンクが削除されると、「Warehouse Builder リンクの参照」ページを表示しても、既存のリンク・リストには削除したリンクが含まれていません。
4. 「OK」をクリックして、「リンク」タブへ戻ります。

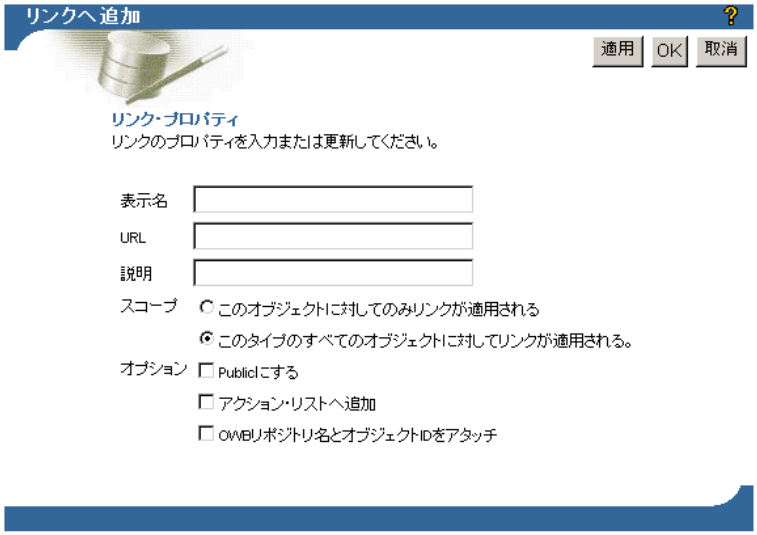
リンクを追加する手順は次のとおりです。

1. 新しいリンクを関連付けるオブジェクトまたはオブジェクト・タイプにナビゲートし、「リンク」タブを選択します。
2. 「リンク」タブから「カスタマイズ」を選択します。

   「Warehouse Builder リンクの参照」ページに「既存のリンク」が表示されます。
3. 「リンクへ追加」を選択します。

   図 17-16 に示すように、「リンクへ追加」ページが表示されます。

図 17-16「リンクへ追加」ページ



4. 表 17-5 を使用してリンク・プロパティを指定し、「OK」をクリックします。

表 17-5 リンク・プロパティ

フィールド	説明
表示名	リンクの名前を入力します。
URL	外部 Web サイトの場合、完全な URL を入力します。 Oracle Portal コンテンツ領域にあるファイルの場合、Oracle Portal Navigator を使用して、ファイルがリストされているコンテンツ領域へ移動します。ファイルを右クリックし、URL ロケーションをコピーします。「リンクへ追加」ページへ戻り、「URL」フィールドにコピーした URL を貼り付けます。
説明	リンクのオプションの説明を入力します。
スコープ	次のオプションから 1 つ選択します。 ■ このオブジェクトに対してのみリンクが適用される : たとえば、現在の項目が Customer Table である場合、このオプションを選択すると、リンクが Customer Table のみで使用可能になります。 ■ このタイプのすべてのオブジェクトに対してリンクが適用される : たとえば、現在の項目が Customer Table である場合、このオプションを選択すると、リンクがすべての表で使用可能になります。
オプション	次のオプションから選択します。 ■ Public にする : このオプションは、他のユーザーにこのリンクへのアクセスを許可する場合に選択します。 ■ アクション・リストへ追加 : このオプションは、現在の項目に「内容」タブのリンクを追加する場合に選択します。 ■ Warehouse Builder リポジトリ名とオブジェクト ID をアタッチ : このオプションは、Warehouse Builder Design Repository の名前とオブジェクト ID をリンクに付与する場合に選択します。

お気に入りの参照

「お気に入りを参照」を選択すると、「Warehouse Builder お気に入り」ページが表示されます。すべてのナビゲーション・ページやレポートには、右上の隅に「お気に入りに追加」リンクがあります。このリンクを選択すると、現在のレポートまたは「ナビゲーション」ページへのリンクが、「お気に入り」リストに追加されます。また、「お気に入り」ページの機能を使用して、リストから項目を削除することも、リストのフォーマットを変更することもできます。

「お気に入り」ページには、図 17-17 に示すように、選択したレポートおよびナビゲーションが表示されます。

図 17-17「Warehouse Builder お気に入り」ページ

一般ページ情報

Warehouse Builder Browser の各ページの上部には、現在表示されているページのタイプを定義するヘッダーが表示されます。ヘッダーには、デフォルトのホームページ、ヘルプ・ページ、カスタマイズ・ページおよびログアウトのリンクがあります。

「お気に入り」ページをカスタマイズする場合、「カスタマイズ」リンクを選択します。ページのカスタマイズの詳細は、17-23 ページの「「お気に入り」ページのカスタマイズ」を参照してください。

「お気に入り」のフィルタ

特定のリポジトリ、ロール、または項目タイプを選択すると、2 つの「お気に入り」表のコンテンツを絞り込むことができます。これらの各項目には「すべて」オプションを選択できます。図 17-18 は、「お気に入り」ページのフィルタを示しています。

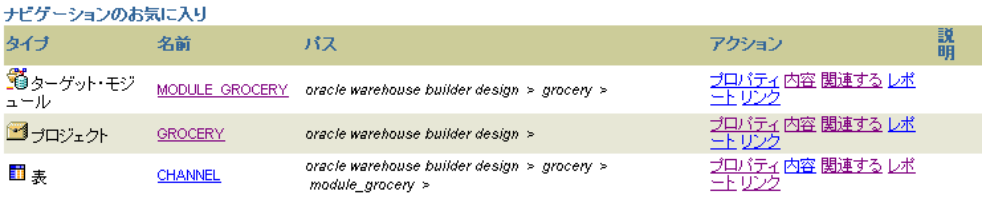
図 17-18「お気に入り」のフィルタ

ナビゲーションのお気に入り

「お気に入り」ページには、「ナビゲーションのお気に入り」が次の形式で表示されます。図 17-19 は、「ナビゲーションのお気に入り」ページを示しています。

- **タイプ :** 項目タイプと、そのタイプを表すアイコンを示します。
- **名前 :** 項目の名前です。名前を選択すると、その項目の「内容」ページが表示されます。
- **パス :** 「パス」には、「ナビゲータ」ページの「パス」フィールドに示される項目の完全な修飾名が表示されます。「お気に入りのカスタマイズ」ページを使用すると、この列を削除できます。
- **アクション :** アクションは、その項目に対する「ナビゲータ」ページのタブへのリンクを提示します。
- **説明 :** 「お気に入りのカスタマイズ」ページでお気に入りに追加した、オプションの記述テキストが表示されます。

図 17-19 ナビゲーションのお気に入り

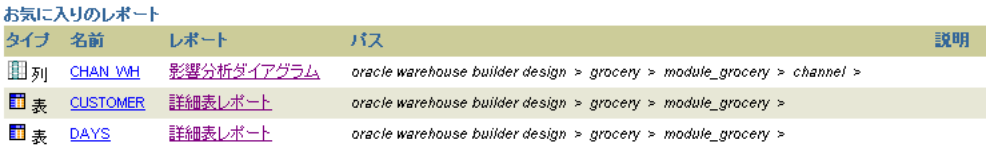

ナビゲーションのお気に入り

タイプ	名前	パス	アクション	説明
ターゲット・モジュール	MODULE_GROCERY	oracle warehouse builder design > grocery >	プロパティ 内容 関連する レポート リンク	
プロジェクト	GROCERY	oracle warehouse builder design >	プロパティ 内容 関連する レポート リンク	
表	CHANNEL	oracle warehouse builder design > grocery > module_grocery >	プロパティ 内容 関連する レポート リンク	

お気に入りのレポート

「お気に入り」ページには、「お気に入りのレポート」が「ナビゲーションのお気に入り」と同じ形式で表示されます。図 17-20 は「お気に入りのレポート」ページを示しています。

図 17-20 お気に入りのレポート

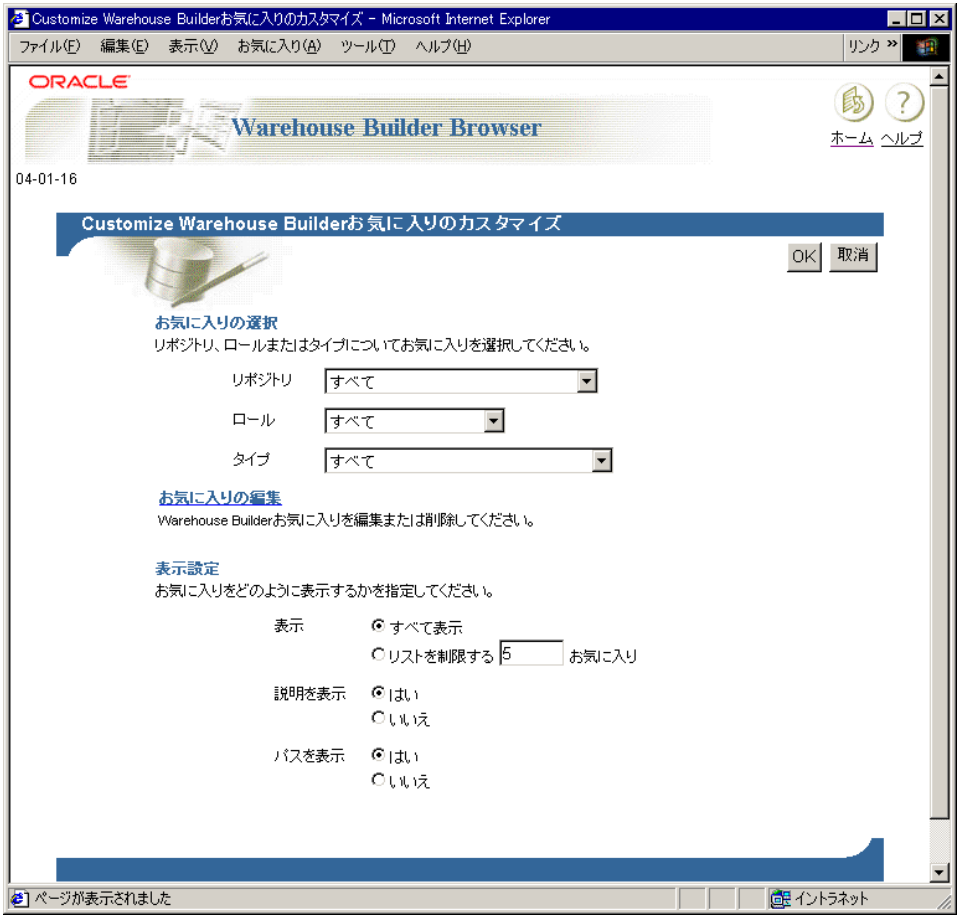

お気に入りのレポート

タイプ	名前	レポート	パス	説明
列	CHAN_WH	影響分析ダイアグラム	oracle warehouse builder design > grocery > module_grocery > channel >	
表	CUSTOMER	詳細表レポート	oracle warehouse builder design > grocery > module_grocery >	
表	DAYS	詳細表レポート	oracle warehouse builder design > grocery > module_grocery >	

「お気に入り」ページのカスタマイズ

お気に入りのページをカスタマイズするには、「Warehouse Builder お気に入り」ページで「カスタマイズ」を選択します。図 17-21 と図 17-22 に示すように、「お気に入りのカスタマイズ」ページが表示されます。「お気に入り」ページのお気に入りのレイアウトを変更できます。

図 17-21「お気に入りのカスタマイズ」ページ



このページには、次のオプションが含まれています。

- **OK:** 変更を適用し、前のページに戻ります。
- **取消 :** 変更を無効にし、前のページに戻ります。
- **表示 :** 表に表示される最大行数を設定します。表示範囲を超えた項目がある場合、「次へ」と「前へ」をクリックして、他の項目を表示できます。
- **説明を表示 :** 説明列を表示するには「はい」を、説明列を非表示にするには「いいえ」を選択します。
- **パスを表示 :** 「はい」を選択すると、パス列が表示されます。「いいえ」を選択すると、列が表示されません。

図 17-22「Warehouse Builder お気に入り」ページ

表 17-6 では、「Warehouse Builder お気に入り」ページに表示される列を説明します。

表 17-6「Warehouse Builder お気に入り」ページのアクション

アクション	説明
編集	「編集」アクションを使用して、お気に入りの説明を入力します。この説明は、「お気に入り」リストの列として表示されます。 「カスタマイズ」ページへ戻るには、「OK」または「取消」を選択します。
お気に入りから削除	このアクションを使用して、「お気に入り」表からエントリを削除します。ただし、Warehouse Builder Design Repository から項目が削除されることはありません。

Warehouse Builder レポートの表示

Warehouse Builder Design Repository の Browser レポートは、次のツールを使用して表示できます。

- Warehouse Builder Design Client
- Warehouse Builder Browser

Warehouse Builder Design Client でのレポートの表示

Warehouse Builder Design Client を使用して、メタデータ・レポートを表示できます。ただし、まず Oracle Portal がインストールされていることを確認する必要があります。メタデータ・レポートを表示するには、まず 18-27 ページの「Warehouse Builder Design Client の構成」で説明されている手順を完了してください。

Warehouse Builder Design Client からレポートにアクセスする手順は次のとおりです。

1. Warehouse Builder でプロジェクトを開き、ナビゲーション・ツリーから任意のノードを開いて任意のオブジェクトを選択します。または、マッピング・エディタ、プロセス・フロー・エディタなど、任意のエディタを開きます。
2. 「表示」メニューから「レポート」を選択します。

   「作業環境」ウィンドウで定義した Oracle Portal サイトのログオン画面が起動されます。詳細は、2-12 ページの「Warehouse Builder の起動」を参照してください。
3. Internet Application Server（*i*AS）シングル・サインオン・ユーザー名およびパスワードを入力します。

   17-28 ページの図 17-25 に示すように、選択したオブジェクトに使用できるレポートのリストが Warehouse Builder Browser に表示されます。

4. レポートの「ビュー」リンクをクリックします。

   レポートが個別のウィンドウに表示されます。図 17-23 に表サマリー・レポートの例を示します。

図 17-23 Warehouse Builder レポート

レポートを印刷するには、ブラウザのメニューまたはツールバーから「印刷」を選択します。

カスタム・レポートの表示

Warehouse Builder Browser からカスタム・レポートを表示する手順は次のとおりです。

1. Warehouse Builder Browser にログオンし、レポートへのアクセス権限を持つロールを選択し、「参照」をクリックします。
2. レポート・タイプのインスタンスへ移動し、「レポート」タブを選択します。
3. カスタム・レポートがレポート・リストに表示されます。「ビュー」アクション・リンクを選択します。

   レポートが表示されます。

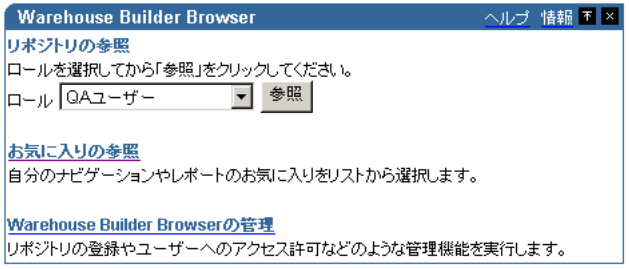
Warehouse Builder Browser でのレポートの表示

Warehouse Builder Design Client をインストールしなくても、Warehouse Builder Browser からレポートを表示できます。まず、Oracle Portal を起動します。ここで、Warehouse Builder Browser ポートレットを選択できます。

Warehouse Builder Browser でメタデータ・レポートを表示する手順は次のとおりです。

1. 図 17-24 に示すように、Oracle9*i* Warehouse Builder Browser Launcher ポートレットの「ロール」ドロップダウン・リストより「Warehouse Builder エンジニア」、「QA ユーザー」、「Warehouse Builder ユーザー」から目的のユーザーを選択します。

図 17-24 リポジトリの選択



2. 「参照」をクリックします。

   「Warehouse Builder Navigation」ページが表示されます。リポジトリを選択し、リポジトリ内を移動することも、レポートを作成する項目を検索することもできます。

3. 「レポート」タブを選択します。

   17-28 ページの図 17-25 に示すように、その項目に選択できるレポートが Warehouse Builder Browser に表示されます。

4. 「ビュー」リンクを選択して、レポートを表示します。

Warehouse Builder のメタデータ・レポート

Warehouse Builder では、図 17-25 に示すように、Warehouse Builder Public View を使用する一連のレポートが含まれています。

図 17-25 Warehouse Builder のメタデータ・レポート

名前	アクション
アドバンスト・キュー・サマリー・レポート	ビュー
キューブ・サマリー・レポート	ビュー
ディメンション・サマリー・レポート	ビュー
トランスフォーム・マッピング・サマリー・レポート	ビュー
トランスフォーム・ライブラリ・サマリー・レポート	ビュー
ビュー・サマリー・レポート	ビュー
ファンクション・サマリー・レポート	ビュー
プロシージャ・サマリー・レポート	ビュー
マテリアライズド・ビュー・サマリー・レポート	ビュー
外部表サマリー・レポート	ビュー
表サマリー・レポート	ビュー
詳細ターゲット・モジュール・レポート	ビュー
順序サマリー・レポート	ビュー

これらのレポートを使用して、メタデータを確認できます。次のレポートを使用できます。

- サマリー・レポート（17-29 ページ）
- 詳細レポート（17-30 ページ）
- インプリメンテーション・レポート（17-33 ページ）
- 系統および影響分析のレポートとダイアグラム（17-33 ページ）
- 系統および影響分析ダイアグラム（17-34 ページ）

関連情報は、次を参照してください。

- カスタム・レポートの作成（18-28 ページ）

サマリー・レポート

サマリー・レポートは、ソース・モジュールおよびターゲット・システムに対して実行できます。各サマリー・レポートには、モジュール内に含まれた項目リストが表示されます。表示される項目の種類は、選択したサマリー・レポートによって決まります。たとえば、表サマリー・レポートには、モジュール中のすべての表が表示されます。また、マテリアライズド・ビュー・サマリー・レポートでは、モジュール中のすべてのマテリアライズド・ビューが表示されます。さらに、モジュールを識別するヘッダー情報も表示されます。項目名を選択すると、その項目の詳細レポートが表示されます。

サマリー・レポートは、次のオブジェクトに対して実行できます。

- アドバンスト・キュー
- コレクション
- キューブ
- ディメンション
- 外部表
- ファイル
- ファンクション
- マテリアライズド・ビュー
- プロシージャ
- 順序
- 表ライブラリ
- テーブル・ファンクション
- 表
- 変換マップ
- ビュー

図 17-26 に、ディメンション・サマリー・レポートの一部を示します。

図 17-26 ディメンション・サマリー・レポート

ディメンション・サマリー・レポート - MODULE_GROCERY

ビジネス名	MODULE_GROCERY	作成	04-01-14
物理名	MODULE_GROCERY	更新	04-01-19

説明

This is an Oracle target module containing the schema and the mappings that are used to load it.

プロジェクト	GROCERY	ステータス	Development
検証結果	はい	アプリケーション・タイプ	Oracle Warehouse Application
システム・タイプ	Oracle Database 8i/9i	データ・ソースへアクセスするインテグレータ	Oracle OWB Integrator for Oracle DB and Apps 3.0
データベース・リンク	(使用不能)		

ディメンション

ビジネス名	物理名
説明	
CHANNELS	CHANNELS
Sales channels	
CUSTOMERS1	CUSTOMERS1
Dimension of customer information.	

詳細レポート

詳細レポートは、ある項目に関する総合的な情報を提示します。たとえば、詳細表レポートには、表の列、キー、外部キーおよび物理構成パラメータが表示されます。表 17-7 に、詳細レポートとその内容を示します。

表 17-7 詳細レポート

レポート	内容
詳細アドバンスト・キュー・レポート	
詳細コレクション・レポート	コレクション内の項目
詳細コネクタ・レポート	
キューブ・インプリメンテーション・レポート	
詳細キューブ・レポート	メジャー、ディメンションおよび物理構成パラメータ
ディメンション・インプリメンテーション・レポート	

表 17-7 詳細レポート（続き）

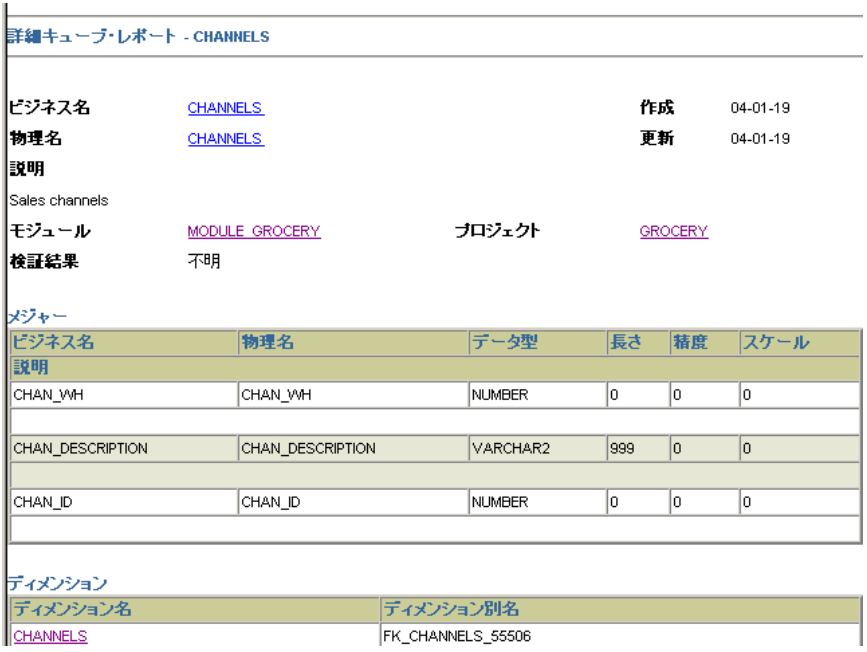
レポート	内容
詳細ディメンション・レポート	階層、レベル、レベル属性および物理構成パラメータ
詳細外部表レポート	
詳細ファイル・モジュール・レポート	
詳細ファイル・レポート	レコードおよび物理構成パラメータ
詳細ファンクション・ライブラリ・レポート	ファンクションおよび物理構成パラメータ
詳細ファンクション・レポート	パラメータおよびファンクション実装
詳細インストレーション・レポート	プロジェクト
詳細ロケーション・レポート	
詳細マテリアライズド・ビュー・レポート	列、キー、外部キーおよび物理構成パラメータ
詳細モジュール・レポート	キューブ、ディメンション、ファンクション・ライブラリ、マテリアライズド・ビュー、順序、表、ビュー、変換マップ、ファイルおよび物理構成パラメータ
詳細オブジェクト・レポート	
詳細プロセス・フロー・アクティビティ・レポート	
詳細プロセス・フロー・モジュール・レポート	
詳細プロセス・フロー・パッケージ・レポート	
詳細プロセス・フロー・パッケージ・レポート	
詳細プロセス・フロー・レポート	
詳細プロセス・フロー推移レポート	
詳細プロジェクト・レポート	モジュールおよびビジネス領域
詳細レコード・レポート	フィールド
詳細リポジトリ・レポート	
詳細ランタイム・リポジトリ接続レポート	
詳細順序レポート	列および物理構成パラメータ

表 17-7 詳細レポート（続き）

レポート	内容
詳細テーブル・ファンクション・レポート	
詳細表レポート	列、キー、外部キーおよび物理構成パラメータ
詳細トランスフォーム・マップ・レポート	使用オブジェクト、マップ・コンポーネント、項目マップおよび物理構成パラメータ
詳細トランスフォーム・マップ・レポート	
詳細ビュー・レポート	列、キー、外部キーおよび物理構成パラメータ

図 17-27 に、詳細キューブ・レポートの一部を示します。

図 17-27 詳細キューブ・レポート



詳細キューブ・レポート - CHANNELS

ビジネス名 CHANNELS　作成 04-01-19
物理名 CHANNELS　更新 04-01-19
説明
Sales channels
モジュール MODULE_GROCERY　プロジェクト GROCERY
検証結果 不明

メジャー

ビジネス名	物理名	データ型	長さ	精度	スケール
説明					
CHAN_WH	CHAN_WH	NUMBER	0	0	0
CHAN_DESCRIPTION	CHAN_DESCRIPTION	VARCHAR2	999	0	0
CHAN_ID	CHAN_ID	NUMBER	0	0	0

ディメンション

ディメンション名	ディメンション別名
CHANNELS	FK_CHANNELS_55506

インプリメンテーション・レポート

インプリメンテーション・レポートは、ディメンションとキューブで実行できます。インプリメンテーション・レポートには、論理オブジェクトを実装する際の物理オブジェクトの使用に関する情報が表示されます。表 17-8 に、使用可能なインプリメンテーション・レポートを示します。

表 17-8 インプリメンテーション・レポート

レポート	内容
キューブ・インプリメンテーション・レポート	キューブを実装する表、メジャーを実装する列、Dimension Use にキューブを実装する外部キー。
ディメンション・インプリメンテーション・レポート	レベルとレベルを実装する表、レベル属性とレベル属性を実装する列

系統および影響分析のレポートとダイアグラム

系統および影響分析のレポートとダイアグラムは、キューブ、ディメンション、マテリアライズド・ビュー、表、ビューおよびレコードに使用できます。

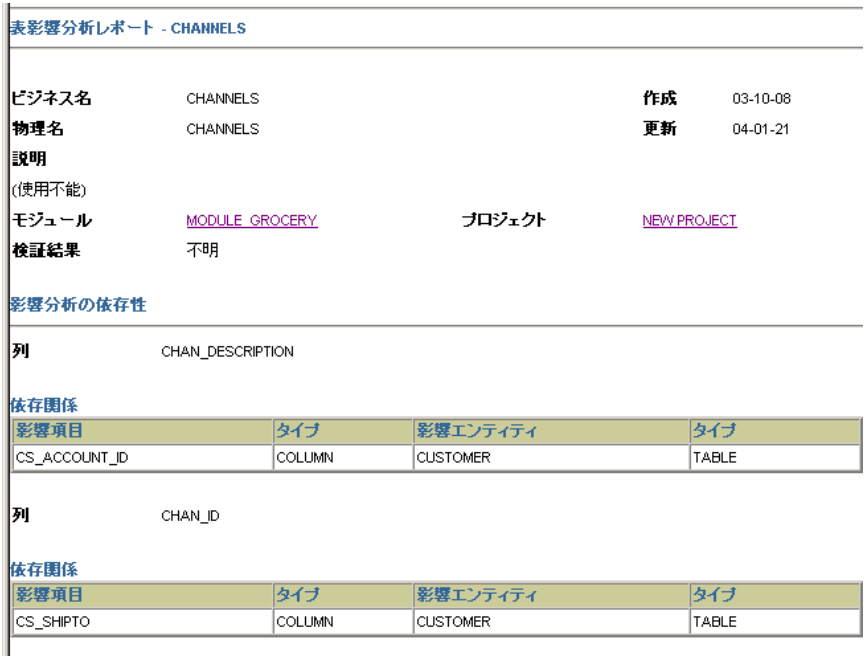
影響分析レポート 影響分析レポートには、レポートのサブジェクトに属するすべての項目が表示されます。マッピング名と、マッピングされる項目名も表示されます。このレポートにより、選択した項目に関連するすべての項目に対してワンステップの影響分析が得られます。

たとえば、マップのソースとして使用する表のすべての列を一覧表示する場合に、このレポートを使用します。

系統レポート 系統レポートは、影響分析レポートと同様です。このレポートには、マッピングのターゲットとして使用する項目が表示されます。

図 17-28 に、影響分析レポートの一部を示します。

図 17-28 影響分析レポート



表影響分析レポート - CHANNELS

ビジネス名	CHANNELS	作成	03-10-08
物理名	CHANNELS	更新	04-01-21
説明			
(使用不能)			
モジュール	MODULE_GROCERY	プロジェクト	NEW PROJECT
検証結果	不明		

影響分析の依存性

列 CHAN_DESCRIPTION

依存関係

影響項目	タイプ	影響エンティティ	タイプ
CS_ACCOUNT_ID	COLUMN	CUSTOMER	TABLE

列 CHAN_ID

依存関係

影響項目	タイプ	影響エンティティ	タイプ
CS_SHIPTO	COLUMN	CUSTOMER	TABLE

系統および影響分析ダイアグラム

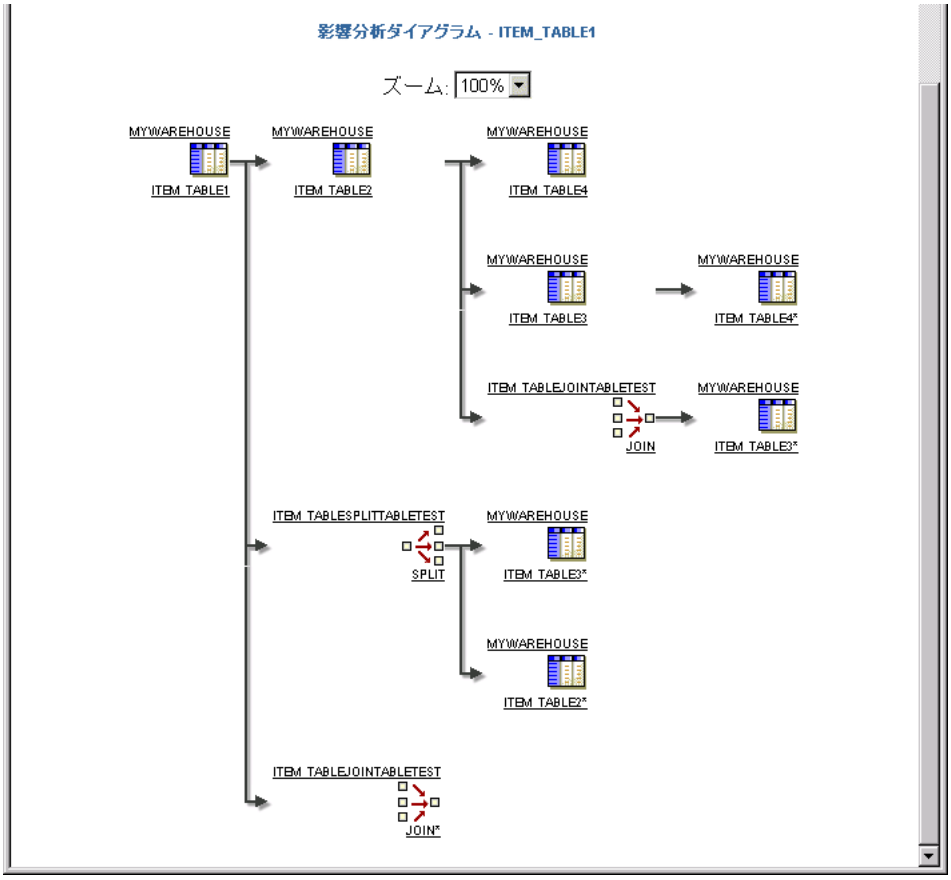
系統ダイアグラムには、図のサブジェクトの構成に使用するすべてのオブジェクトと変換がグラフィック表示されます。系統は、オブジェクト・レベルまたは項目レベルのいずれかで実行できます。オブジェクト・レベルでは、ダイアグラムに、表、ビュー、マテリアライズド・ビュー、ディメンション、キューブ、レコードおよび演算子を含めることができます。項目レベルでは、ダイアグラムに、列、メジャー、フィールド、演算子パラメータおよびレベル属性を含めることができます。

系統ダイアグラムは、画面の右側にサブジェクトとともに表示されます。

影響分析ダイアグラムは、サブジェクトへの変更によって影響を受ける可能性があるすべてのオブジェクトや変換を表示することを除き、系統ダイアグラムと同じです。サブジェクトは、画面の左側に表示されます。

図 17-29 に、影響分析ダイアグラムの例を示します。

図 17-29 影響分析ダイアグラム



系統および影響分析ダイアグラムは、依存性索引に基づいて作成します。ダイアグラムに最新データを表示させるためには、この索引をリフレッシュする必要があります。Warehouse Builder Public View に基づくカスタム・レポートを作成することもできます。詳細は、18-28 ページの「カスタム・レポートの作成」を参照してください。

注意： Warehouse Builder メタデータ・レポートにアクセスするには、構成済の Warehouse Builder Browser ポートレットを含む Oracle Portal サイトへのアクセス権限が必要です。

18

Warehouse Builder Browser の管理

Warehouse Builder ポートレットは、Warehouse Builder Browser と Oracle Portal との統合により、追加およびカスタマイズできます。Warehouse Builder Browser では、Warehouse Builder Public View を使用して、すべてのリポジトリのオブジェクトとオブジェクト間のリレーションシップの事前作成レポートを作成できます。また、Public View を使用して独自のカスタム・レポートも作成できます。

この章では、次のトピックについて説明します。

Warehouse Builder Browser の概要

Warehouse Builder Browser は、リポジトリ・メタデータに関するレポートの拡張、アクセスおよび実行に使用できる、Web ベースのアプリケーションです。メタデータの表示や、これらのレポートへのアクセスを行うには、Oracle AS Portal に対するアクセス権限が必要です。

Warehouse Builder を最初にインストールするとき、またはアップグレードするときは、スタート・メニューの Warehouse Builder Browser Assistant を使用して、Warehouse Builder Browser をインストールできます。Browser Assistant を使用すると、メタデータ・レポートのアクセスおよび作成に使用する Oracle AS Portal をセットアップできます。Browser Assistant は、インストール・プロセス中に実行することも、後で実行することもできます。

Warehouse Builder Browser をインストールすると、「Portal」ホームページに Warehouse Builder ポートレットを追加できます。Warehouse Builder Design Client または Oracle Portal からブラウザを使用して、メタデータ・レポートを実行できます。

Oracle AS Portal について

Oracle AS Portal では、HTML ベースのインターフェースを使用してデータベース・オブジェクトを作成および表示できます。HTML ベースのインターフェースを作成するためのツールを利用できます。Portal を使用すると、関連するアプリケーションやデータを 1 つの Web サイトに集中させ、パーソナライズして表示できます。

Oracle AS Portal サイトの基礎となる構築ブロックを、ポートレットといいます。ポートレットとは、情報源の要約や情報源へのアクセスに繰り返し使用できる情報コンポーネントです。ポートレットには、ポータル・サイトでスタンドアロンにする、他のポートレットまたはポータル・サイトにリンクさせる、相互にネストさせるなどの使用方法があります。

Warehouse Builder Browser を Oracle AS Portal と統合すると、メタデータ・レポート・ポートレットを使用できます。

Warehouse Builder Browser 管理用ポートレット

Warehouse Builder Browser は複数のポートレットで構成されており、これを追加してポータル・ページをカスタマイズできます。各ポートレットは複数追加できます。たとえば、2 つのレポートのポートレットを追加し、異なるリポジトリで各レポートを実行できます。

Warehouse Builder Browser の管理に使用するポートレットには、次のものがあります。

- ポートレット・ラウンチャ
- 管理ポートレット
- レポート・ポートレット

ポートレットにアクセスするには、まず Warehouse Builder Design Repository にアクセスできるように Warehouse Builder Browser を構成する必要があります。

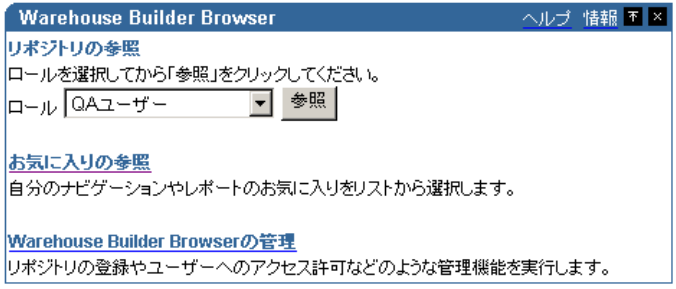
また、ビジネス・インテリジェンスのニーズに応じて、カスタム・ポートレットを追加することもできます。詳細は、18-6 ページの「ポートレットの追加」を参照してください。

ポートレット・ラウンチャ

ポートレット・ラウンチャでは、図 18-1 に示すように、使用可能なすべての機能にアクセスできます。他のポートレットには、使用可能なすべての機能のサブセットが含まれます。

- **リポジトリの参照 :** ロールを選択し、「参照」をクリックして、現在のユーザーに有効なリポジトリを参照します。Warehouse Builder Browser のメイン・ページがフル・ページ・モードで表示されます。リポジトリのリストから選択し、Navigator を使用してそのリポジトリの詳細を表示できます。
- **お気に入りを参照 :** このリンクを選択すると、「Warehouse Builder お気に入り」をフル・ページ・モードで表示できます。
- **Warehouse Builder Browser の管理 :** このリンクを選択すると、「Warehouse Builder 管理」をフル・ページ・モードで表示できます。これらのページを使用して、Browser を構成できます。

図 18-1 ポートレット・ラウンチャ



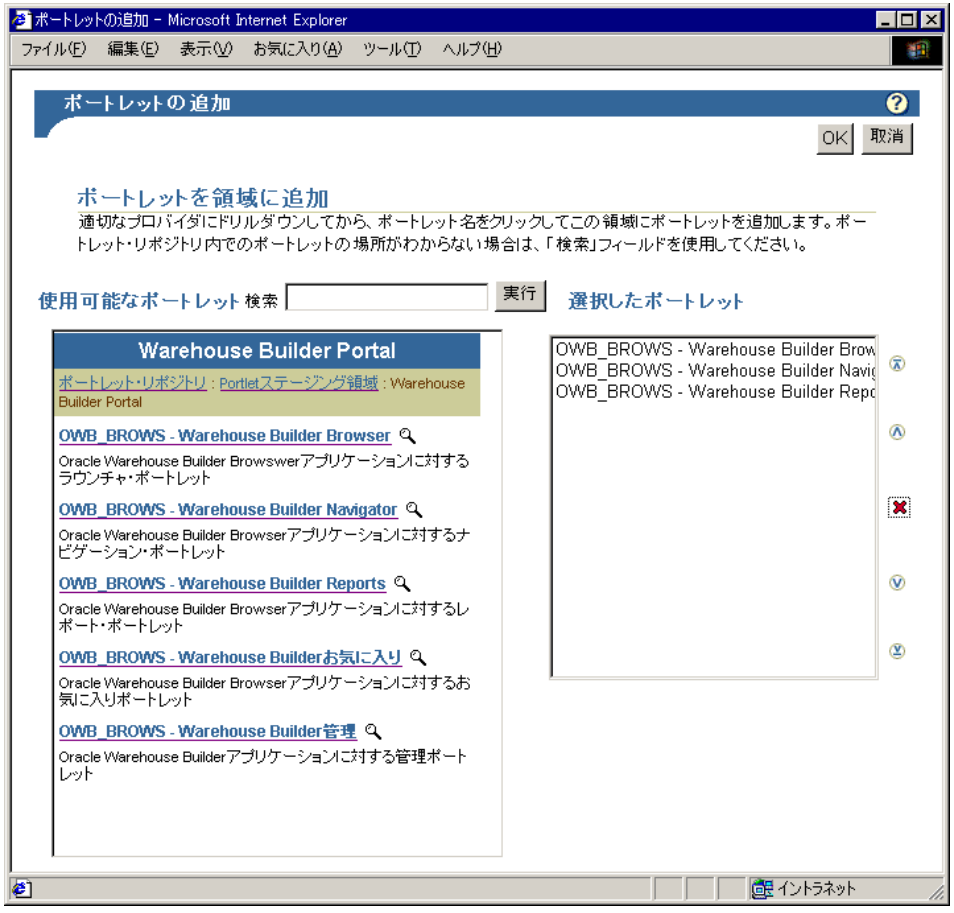
管理ポートレット

管理ポートレットでは、図 18-2 に示すように、「Oracle Portal」ホームページから次の管理機能にアクセスできます。

- **Warehouse Builder リポジトリの登録 :** Warehouse Builder Design Repository を登録し、データベース・リンクを管理します。
- **カスタム・レポートの登録 :** カスタム・レポートを登録します。
- **失効ユーザーのパージ :** 古い Warehouse Builder Browser の設定を削除します。
- **リソース管理 :** Warehouse Builder Browser リソースへのアクセス権限を管理します。
- **作業環境管理 :** ファイルや既存のスキーマの作業環境の設定を保存およびロードします。

- **依存性索引の管理：** 依存性索引をリフレッシュする頻度を指定して、「影響分析」のパフォーマンスを改善します。

図 18-2 管理ポートレット



レポート・ポートレット

レポート・ポートレットでは、図 18-3 に示すように、「Oracle Portal」ホームページに Warehouse Builder レポートが表示されます。レポート・ポートレットには、お気に入りのレポートの 1 つが表示されます。このポートレットを最初にページに追加するとき、このポートレットにデフォルトのレポートは含まれません。まず、カスタマイズ・オプションを使用して、お気に入りリストからレポートを選択し、追加する必要があります。

「Oracle Portal」ホームページにレポート・ポートレットがある場合、ホームページを再ロードするたびにリフレッシュされます。このため、ホームページの表示が遅くなることがあります。

図 18-3 レポート・ポートレット

ポートレットの追加

Warehouse Builder Browser ポートレットは、Oracle Portal を実行するマシンにインストールした後で Oracle Portal に追加できます。

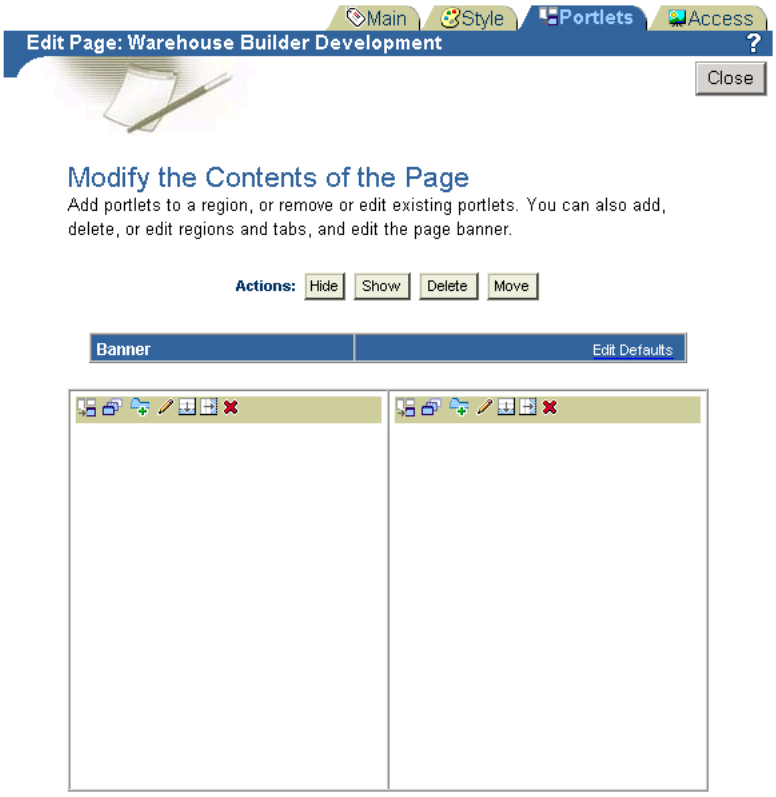
「Oracle Portal」ページのカスタマイズの詳細は、Oracle Portal のドキュメントを参照してください。

ページにポートレットを追加する手順は次のとおりです。

1. 「Oracle Portal」ページで、ページの右上隅から「ページの編集」リンクを選択します。

   「ページの編集」により、図 18-4 に示すように、「Oracle Portal」ページのコンテンツが表示されます。

図 18-4 Oracle Portal の「ページの編集」



2. 「ポートレットの追加」アイコンを選択します。

   「ポートレットの追加」ページでは、図 18-5 に示すように、左側に選択可能なポートレットのリスト、ページの右側に選択したポートレットのリストが表示されます。

図 18-5「ポートレットの追加」ページ

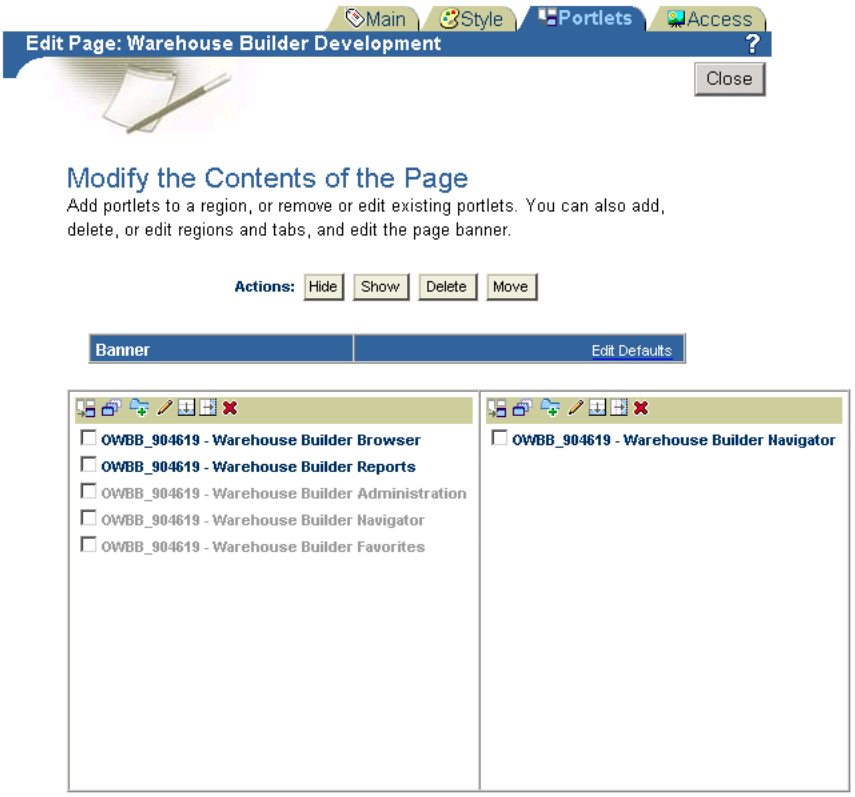
3. 「Oracle Portal」ホームページに追加するポートレットを選択します。

   追加したポートレットは、右側の列に表示されます。矢印ボタンを使用してポートレットを編成したり、「X」ボタンを使用して削除したりできます。

4. 作業が終了したら、「OK」をクリックします。

   「ページの編集」に、図 18-6 のような、追加した Warehouse Builder ポートレットが表示されます。

図 18-6「ページの編集」



Warehouse Builder Browser の管理

「Warehouse Builder 管理」ページには、完全な管理者権限を所有している Oracle Portal ユーザーのみがアクセスできます。

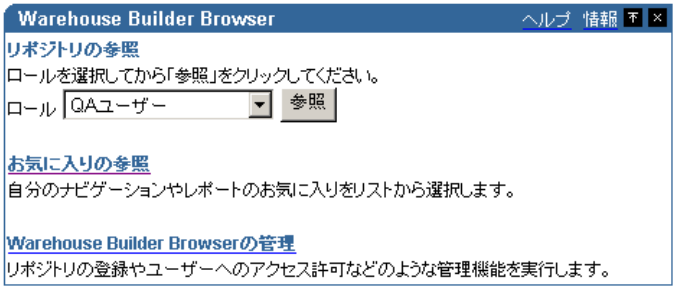
「Oracle Portal」ホームページで、「Warehouse Builder Browser の管理」リンクを選択し、「Warehouse Builder 管理」ページにアクセスします。

図 18-7 は、Warehouse Builder Browser の管理ポートレットを示しています。

図 18-7 Warehouse Builder Browser の管理ポートレット

図 18-8 は、Warehouse Builder Browser のポートレット・ラウンチャを示しています。

図 18-8 Warehouse Builder Browser のポートレット・ラウンチャ



「管理」ページには、次の管理アクションのリンクが含まれています。

- Warehouse Builder リポジトリの登録
- カスタム・レポートの登録
- リソース管理
- 作業環境の管理
- 依存性索引の管理

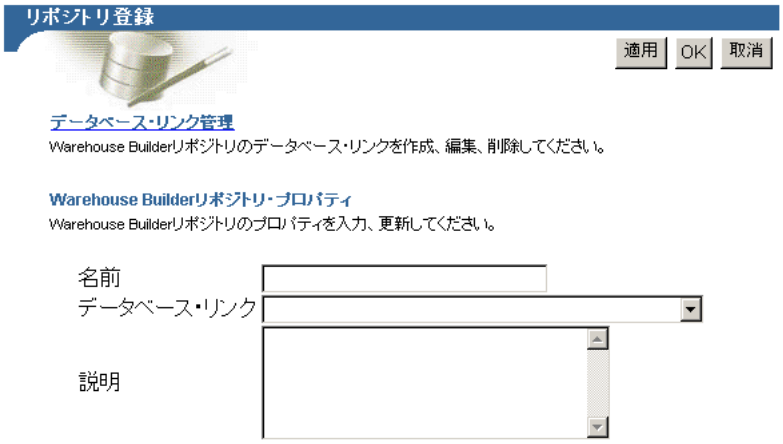
Warehouse Builder リポジトリの登録

リポジトリ・メタデータ・レポートにアクセスするには、Warehouse Builder Design Repository を Warehouse Builder Browser に登録する必要があります。

Warehouse Builder Design Repository を登録する手順は次のとおりです。

1. 「Warehouse Builder 管理」ホームページで「Register an OWB Repository」をクリックします。図 18-9 のような、「リポジトリ登録」ページが表示されます。

図 18-9「リポジトリ登録」ページ



2. Warehouse Builder Design Repository のプロパティを指定します。表 18-1 にプロパティを示します。

表 18-1 Warehouse Builder リポジトリのプロパティ

フィールド	説明
名前	ブラウザ・システムでリポジトリを識別するユーザー定義名。この名前は、ナビゲーション・ページに表示されます。
データベース・リンク	リポジトリへのアクセスに使用するデータベース・リンク名。このリンクは、「データベース・リンク管理」ページを使用して作成済である必要があります。このフィールドは、リポジトリがブラウザ・システムと同じデータベースにある場合でも指定する必要があります。
説明	ユーザー定義の記述テキスト。これは、リポジトリのナビゲーション・ページに表示されます。

3. 「適用」をクリックして、リポジトリを登録します。
4. 「OK」をクリックします。

   リポジトリが、「Warehouse Builder 管理」ホームページに表示されます。

リポジトリの管理

「リソース管理」ページには、登録済リポジトリがすべて表示されます。表 18-2 は、リポジトリの次に表示されるアクションを示します。

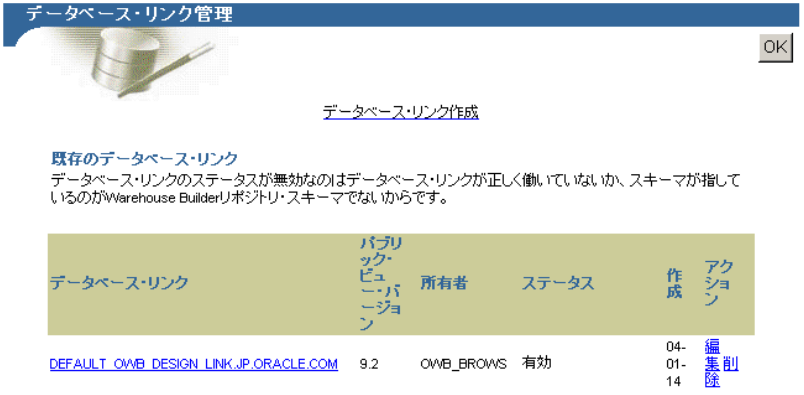
表 18-2 リポジトリ管理

アクション	説明
アクセス	ユーザーへのリポジトリ・アクセス権限を付与または取り消します。
編集	リポジトリのプロパティを編集します。
登録解除	リポジトリを登録解除します。登録を解除すると、ブラウザ・システムを使用してリポジトリを参照できなくなります。再び参照するには、リポジトリを再登録する必要があります。

データベース・リンクの作成

図 18-10 に示す「データベース・リンク管理」ページを使用して、ブラウザから Warehouse Builder Design Repository へ接続します。

図 18-10「データベース・リンク管理」ページ

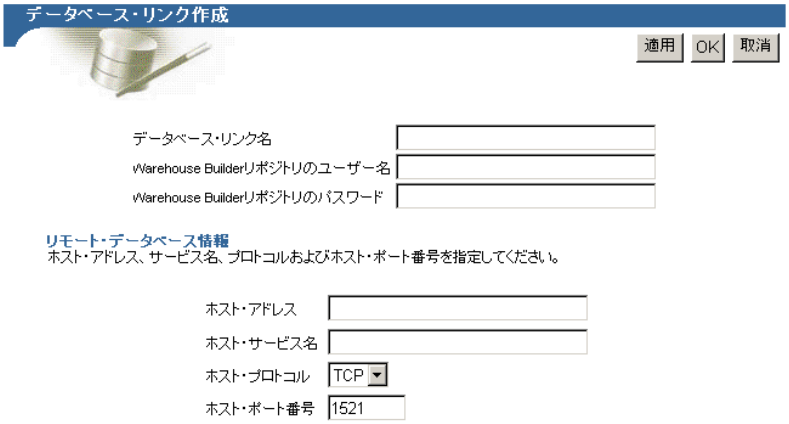


データベース・リンクを作成する手順は次のとおりです。

1. 「Warehouse Builder 管理」ホームページで「Register an OWB Repository」をクリックします。「リポジトリ登録」ページが表示されます。
2. 「データベース・リンク管理」のリンクをクリックします。
3. 「データベース・リンク管理」ページから「データベース・リンクの作成」を選択します。

   図 18-11 は、「データベース・リンクの作成」ページを示しています。

図 18-11「データベース・リンクの作成」ページ



4. データベース・リンク名を指定します。
5. Warehouse Builder Design Repository のユーザー名とパスワードを指定します。
6. リモート・データベース情報を指定します。

   ホスト・アドレス、サービス名、プロトコルおよびホスト・ポート番号を指定します。
7. 「適用」をクリックして、リンクを接続します。
8. 「OK」をクリックします。

   新しいリンクが、「データベース・リンク管理」ページに表示されます。

データベース・リンクの表示

データベース・リンクを表示する手順は次のとおりです。

1. 「データベース・リンク管理」ページで、データベース・リンク名を選択します。

   「データベース・リンクの表示」ページに、選択したデータベース・リンクの詳細レポートが表示されます。

2. 「OK」をクリックします。

   ブラウザは「データベース・リンク管理」ページに戻ります。

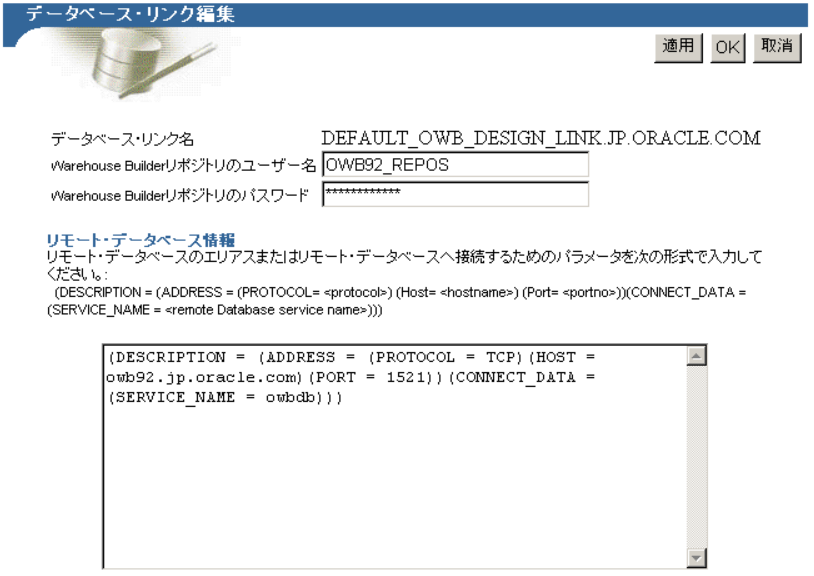
データベース・リンクの編集

データベース・リンクを編集する手順は次のとおりです。

1. 「データベース・リンク管理」ページで、変更するデータベース・リンクに「編集」を選択します。「編集」リンクは「アクション」列の下にあります。

   図 18-12 のような、「データベース・リンク編集」ページが表示されます。

図 18-12「データベース・リンク編集」ページ



2. データベース・リンクを編集し、「適用」をクリックします。
3. 「OK」をクリックします。

データベース・リンクの削除

データベース・リンクを削除すると、永久的に削除されます。再び使用するためには、新しいリンクを作成する必要があります。

データベース・リンクを削除する手順は次のとおりです。

1. データベース・リンクを使用して Warehouse Builder Design Repository を登録している場合、Warehouse Builder Design Repository を登録解除します。
2. 「データベース・リンク管理」ページで、削除するデータベース・リンクに「削除」を選択します。「削除」リンクは「アクション」列の下にあります。

   データベース・リンクが削除され、ブラウザは「データベース・リンク管理」ページに戻ります。

リポジトリの登録解除

リポジトリを登録解除する手順は次のとおりです。

1. 「Warehouse Builder Browser」ページで、「Warehouse Builder Browser の管理」リンクを選択します。
2. 「リソース管理」を選択します。

   図 18-13 のような、「Warehouse Builder 管理」ページが表示されます。ページの一番下にある表には、登録済のリポジトリが表示されます。

図 18-13 登録されたリポジトリとロール

リソース	タイプ	アクション
ポートレット・ラウンチャ	ポートレット	アクセス
Oracle Warehouse Builder Design	リポジトリ	アクセス 編集 登録解除
QAユーザー	ロール	アクセス
Warehouse Engineer	ロール	アクセス
Warehouseユーザー	ロール	アクセス

3. 登録を解除するリポジトリを選択し、「登録解除」リンクをクリックします。

   リポジトリが登録解除され、登録されたリポジトリのリストから消去されます。これにより、Browser で参照できなくなります。

カスタム・レポートの登録

カスタム・レポートは、Oracle Portal 機能などのツールを使用して作成したアプリケーション・コンポーネントです。レポートを登録すると、レポートの起動に必要な情報がブラウザ・システムに渡されます。詳細は、18-28 ページの「カスタム・レポートの作成」を参照してください。

カスタム・レポートを登録する手順は次のとおりです。

1. ポートレット・ラウンチャで、「Warehouse Builder Browser の管理」リンクをクリックし、「カスタム・レポートの登録」リンクを選択します。

   図 18-14 のような、「カスタム・レポートの登録」ページが表示されます。

図 18-14「カスタム・レポートの登録」ページ

2. レポートの表示名を入力します。
3. ドロップダウン・リストからタイプとリポジトリを選択します。表 18-3 にカスタム・レポートのプロパティを示します。

表 18-3 カスタム・レポートのプロパティ

フィールド	説明
表示名	レポートの名前。この名前は、「レポート・リスト」ページに表示されます。
タイプ名	このレポートで出力されるデータ・タイプ名。
パッケージ	このレポートを実装する PL/SQL パッケージのフルネーム。
リポジトリ	このレポートのターゲット・オブジェクトを含むリポジトリ名。

4. `<schema>.<package>` の書式で、レポートの修飾パッケージ名を入力します。パッケージ名は、レポートの「開発」ページに表示されます。
5. 「適用」または「OK」をクリックして、登録を完了します。

   レポートが、管理ページのリソース・リストに表示されます。

ロールへのカスタム・レポートの追加

カスタム・レポートを Warehouse Builder Browser ロールに追加する手順は次のとおりです。

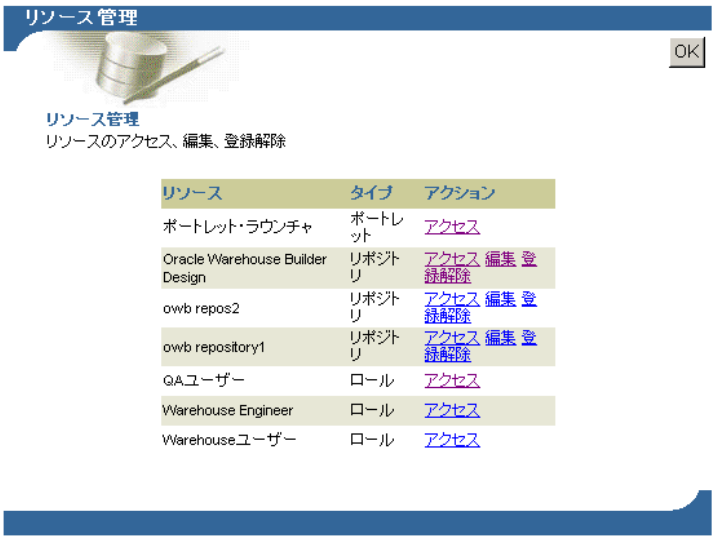
1. カスタム・レポートのリソース・リスト・エントリから「ロール」アクション・リンクをクリックします。
2. レポートにアクセスする各ロールで、「追加」アクション・リンクをクリックします。

リソース管理

リソース管理を使用すると、Warehouse Builder Browser の多くの概観にアクセスできるだけでなく、リポジトリやレポートの編集、登録および登録解除ができます。

「リソース管理」ページでは、図 18-15 に示すように、Warehouse Builder Browser に登録されているすべてのリソースを表示する表が含まれています。次の項では、これらのリソースを変更する方法について説明します。

図 18-15 リソース管理



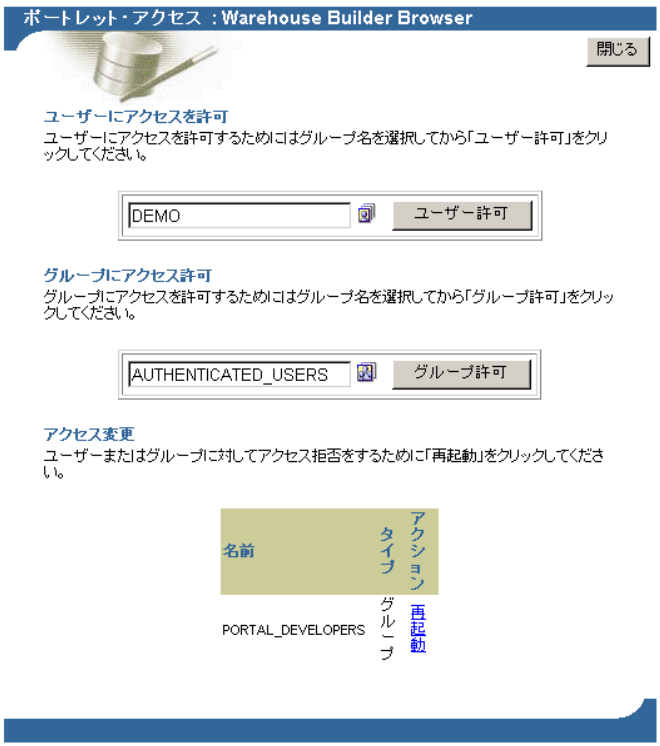
ユーザー・アカウントの追加

Warehouse Builder Browser のユーザー・アカウントを追加する手順は次のとおりです。

1. 「リソース管理」ページで、図 18-15 の最初の項目である「ポートレット・ラウンチャ」というリソースの横にある「アクセス」アクションを選択します。

   図 18-16 のような「ポートレット・アクセス」ページが Warehouse Builder Browser に表示されます。

図 18-16「ポートレット・アクセス」



「ポートレット・アクセス」ページから次のタスクを実行できます。

- Single Sign-On ユーザーにアクセス権を付与するには、ドロップダウン・リストからユーザー名を選択し、「ユーザー許可」をクリックします。
- Oracle Portal グループにアクセス権を付与するには、ドロップダウン・リストからグループ名を選択し、「グループ許可」をクリックします。
- アクセス権を取り消すには、該当する表の行で「再起動」をクリックします。

Warehouse Builder Browser を使用するには、Single Sign-On ユーザーまたはそのユーザーに関連付けられているグループで、次のいずれかの Warehouse Builder の定義済ロールに対する権限が必要です。

- **Warehouse Engineer:** Warehouse Builder を使用してウェアハウスを作成するユーザー。
- **QA ユーザー :** 配布する前に、ウェアハウスの品質をテストするユーザー。
- **Warehouse ユーザー :** 基礎となるメタデータを理解するため、配布済のウェアハウスを使用するユーザー。

管理者は、これらのロールをユーザーやグループに割り当てられます。すべての定義済のレポートや「ナビゲーション」ページは、すべてのロールで使用できます。カスタム・レポートを追加するときに、それらを異なるロールに割り当てることができます。

QA ユーザー・ロールの場合、検証が失敗したオブジェクトでは、図 18-17 に示すように、「ナビゲーション」ページの「内容」タブにエラー・アイコンが表示されます。このエラー・アイコンは、他のロールのレポートには表示されません。

図 18-17 検証エラー

カスタム・レポートの管理

「リソース管理」ページには、登録済のカスタム・レポートがすべて表示されます。表 18-4 は、リポジトリの次に表示されるアクションを示します。

表 18-4 カスタム・レポートの管理

アクション	説明
ロール	レポートを 1 つ以上のロールに割り当てます。レポートをロールに割り当てると、そのロールに該当するレポート・リストに表示されます。
編集	カスタム・レポートのプロパティを編集します。
登録解除	レポートを登録解除します。登録を解除すると、ブラウザ・システムを使用してレポートを参照できなくなります。

カスタム・レポートをロールに割り当てると、そのレポートは、ロールに適切なレポート・リスト・ページに追加されます。レポートの名前とサブジェクト・タイプが、ページ上の左隅に表示されます。使用可能なロールのリストは、ページの一番下にある表に表示されます。

作業環境の管理

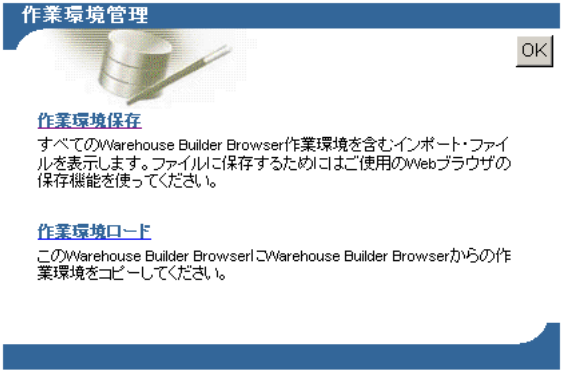
「作業環境管理」ページを使用して、Warehouse Builder Browser の作業環境を保存およびロードします。これにより、作業環境を維持したまま、新しいバージョンの Warehouse Builder Browser にアップグレードできます。スキーマ全体で、スキーマ作業環境をコピーできます。

保存できる作業環境には次のものがあります。

- お気に入り
- 登録済のカスタム・レポート
- 登録済の Warehouse Builder Design Repository
- ロール、リポジトリ、カスタム・レポートおよびポートレット・ラウンチャに対応するアクセス権限
- 外部リンク

図 18-18 は、「作業環境管理」ページを示しています。

図 18-18「作業環境管理」ページ



作業環境の保存

作業環境をファイルに保存する手順は次のとおりです。

1. 「作業環境管理」ページで「作業環境保存」を選択します。

   作業環境は、テキスト形式で個別のウィンドウに表示されます。

2. ブラウザのメニュー・バーから「ファイル」を選択し、「別名で保存」を選択してファイルを保存します。

   このファイルは、SQL* Plus などのツールを使用して、Warehouse Builder Browser にロードできます。

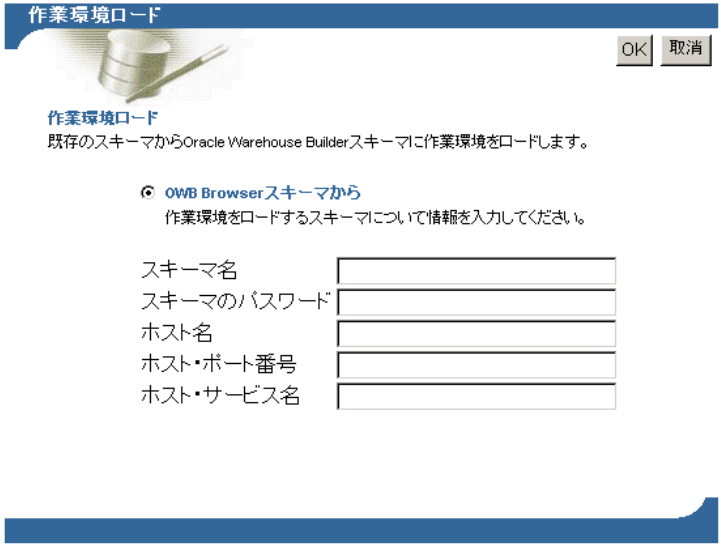
作業環境のロード

既存のスキーマから作業環境をロードする手順は次のとおりです。

1. 「作業環境管理」ページで「作業環境ロード」を選択します。

   図 18-19 は、「作業環境ロード」ページを示しています。

図 18-19「作業環境ロード」ページ



2. 既存の Warehouse Builder Browser スキーマから、「スキーマ名」、「スキーマのパスワード」、「ホスト名」、「ホスト・ポート番号」、「ホスト・サービス名」の情報を指定します。

3. 「OK」をクリックして、現在の Warehouse Builder スキーマに作業環境をロードします。

   ステータス・ページに、ロードした作業環境と、発生したエラーが表示されます。作業環境をロードするには、すべてのエラーを解決する必要があります。データベース・リンクが見つからないことによるエラーが発生した場合、「データベース・リンク作成」ページへのリンクが表示されます。

   データベース・リンクは自動的に作成されません。ロードしている作業環境にリポジトリへの参照が含まれる場合、ロードを正常に実行するには、これらのリポジトリへのデータベース・リンクを作成する必要があります。

依存性索引の管理

「依存性索引の管理」ページを使用して、各リポジトリの依存性索引のリフレッシュ頻度オプションを指定します。依存性索引を使用すると、系列および影響分析ダイアグラムを実行する際のパフォーマンスを向上できます。依存性索引は、Warehouse Builder Navigator の「リポジトリ」ページから常にリフレッシュできます。

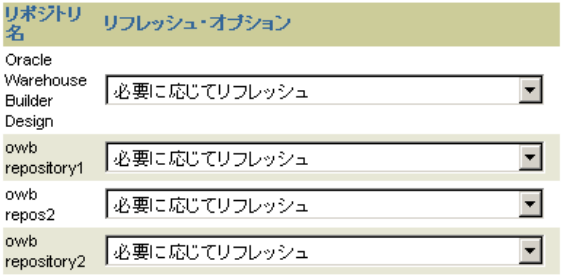
リフレッシュ・オプションの設定

依存性リフレッシュ・オプションを指定する手順は次のとおりです。

1. Warehouse Builder Browser を表示し、「管理」ページまたはポートレットで「依存性索引の管理」を選択します。

   「依存性索引の管理」ページでは、図 18-20 に示すように、使用可能なリポジトリとリフレッシュ・オプションが表示されます。

図 18-20「リフレッシュ・オプション」の設定



2. ドロップダウン・リストからオプションの 1 つを選択し、「OK」をクリックします。表 18-5 に各オプションを示します。

表 18-5 依存性索引オプション

オプション	説明
必要に応じてリフレッシュ	索引をリフレッシュするには、「依存性索引のリフレッシュ」リンクをアクティブにする必要があります。このリンクは、アクセス可能なすべてのリポジトリを表示する「ナビゲータ」ページにあります。依存性索引は、このアクション・リンクがアクティブである場合のみリフレッシュされます。 これは、変更頻度の低いリポジトリを使用する場合に最適なオプションです。
セッションの最初のダイヤグラム要求でリフレッシュ	セッション中にリポジトリの系統または影響分析ダイアグラムを初めて実行したときに、依存性索引がリフレッシュされます。 このオプションは、現在の情報が必要でも、セッション中に発生したリポジトリの更新を考慮する必要がない場合に最適です。
ダイヤグラム要求毎にリフレッシュ	系列または影響分析ダイアグラムが要求されるたびに、依存性索引がリフレッシュされます。 このオプションは、図やレポートにリポジトリの最新情報を反映する場合に最適です。

要求時に依存性索引をリフレッシュ

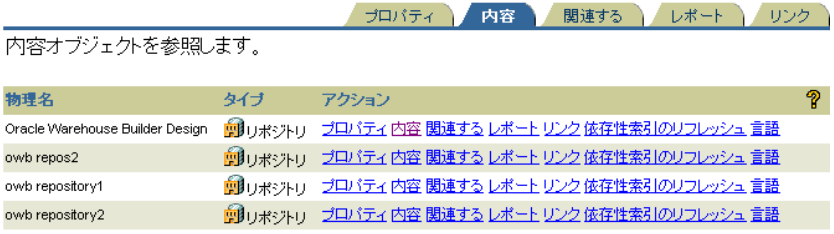
依存性索引は、常にリフレッシュできます。まったくリフレッシュされていない系統または影響分析ダイアグラムを実行すると、図を表示する前に、自動リフレッシュが実行されます。

依存性索引をリフレッシュする手順は次のとおりです。

1. ポートレット・ラウンチャから Warehouse Builder Browser を起動します。

   「内容」タブでは、図 18-21 に示すように、使用可能なリポジトリが表示されます。

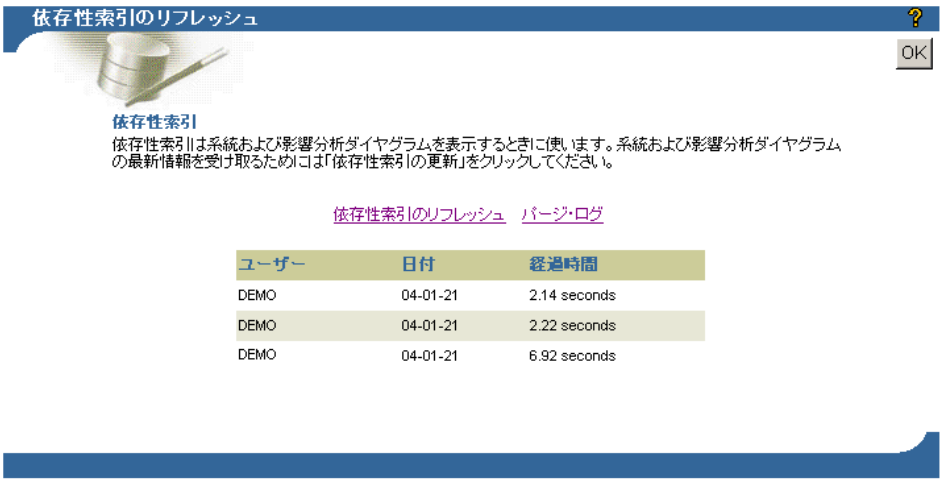
図 18-21 依存性索引のリフレッシュ



2. 「依存性索引のリフレッシュ」を選択します。

   「依存性索引のリフレッシュ」ページが表示されます。ページの一番下には、図 18-22 に示すように、前回のリフレッシュのログが表示されます。経過時間が表示されるため、この操作にかかる時間を確認できます。

図 18-22 依存性索引リフレッシュ・ログ



3. 「依存性索引のリフレッシュ」を選択して、リポジトリの最新データに基づいて依存性索引をリフレッシュします。

   リフレッシュが完了すると、ログに、リフレッシュのユーザー名、日付および経過時間が表示されます。ログを削除するには、「パージ・ログ」を選択します。最新のリフレッシュ以外のリフレッシュのログが削除されます。

その他の管理タスク

ポートレット・リポジトリをリフレッシュする手順は次のとおりです。

1. 「Oracle Portal」ホームページで「管理」タブをクリックします。
2. ポートレット・リポジトリというポートレットが見つかるまで、下にスクロールします。
3. 「ポートレット・リポジトリの更新」をクリックします。
4. 「リフレッシュ」をクリックします。

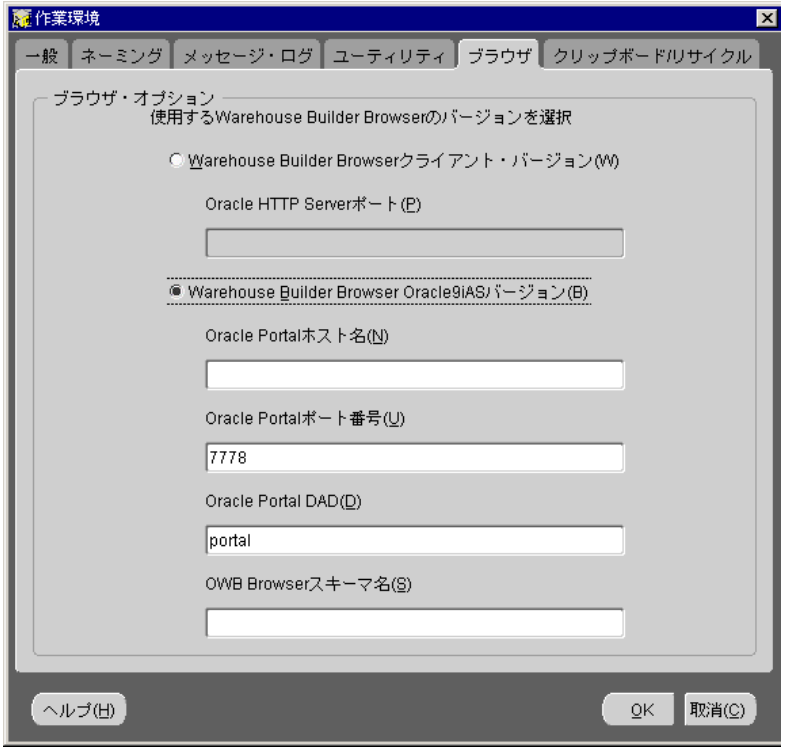
Warehouse Builder Design Client の構成

Warehouse Builder コンソールには、Warehouse Builder 環境の構成に使用するタブを含む「作業環境」ダイアログがあります。「ブラウザ」タブでは、Oracle Portal 用のネットワークおよび IP 接続情報を設定します。これにより、Warehouse Builder Browser を使用してメタデータ・レポートを表示できます。

「作業環境」ダイアログにアクセスする手順は次のとおりです。

1. 「プロジェクト」メニューから「作業環境」をクリックします。
2. 図 18-23 に示すように、「作業環境」ダイアログから「ブラウザ」タブをクリックします。

図 18-23「ブラウザ」タブ



3. 次の情報を指定します。

   Oracle Portal ホスト名

   Oracle Portal ポート番号

   Oracle Portal DAD

   OWB Browser スキーマ名

4. 「OK」をクリックします。

カスタム・レポートの作成

Warehouse Builder Public View および Standard Query Language（SQL）を使用して、メタデータのカスタム・レポートを作成できます。これらのビューで、メタデータのリポジトリ表にアクセスし、データ定義、変換、および配布領域のレポートを作成できます。パブリック・ビューは、Warehouse Builder Browser または他のレポート・ツールを使用して表示できます。

使用可能な Public View と、それらに含まれるオブジェクトの詳細は、付録 D「Warehouse Builder Public View」を参照してください。

Oracle9*i*AS Portal でのカスタム・レポートの作成

Oracle9*i*AS Portal を使用してカスタム・レポートを作成する手順は次のとおりです。

1. Oracle Portal にログオンし、「ビルダー」ページから「構築」タブを選択します。
2. 「データベース・ポートレット・プロバイダ」ポートレット内の「データベース・ポートレット・プロバイダの作成」リンクをクリックします。
3. 「Portal DB プロバイダ名」、「Portal DB プロバイダの表示名」、「Portal DB プロバイダのスキーマ」を入力します。

   「Portal DB プロバイダのスキーマ」には OWB Browser のスキーマを選択します。
4. 「ナビゲータ」ページから「プロバイダ」タブを選択し、作成した Portal DB プロバイダをクリックします。
5. 「レポート」リンクをクリックします。
6. 次に表示されるページから「SQL 問い合わせからのレポート」リンクを選択します。
7. レポート名と表示名を入力し、「次へ」をクリックします。
8. SQL 問合せを入力してレポートを定義し、「終了」をクリックします。

SQL 問合せを入力した後に、「終了」をクリックせずに「次へ」をクリックすると、レポートの外観をカスタマイズするページへ進みます。後でレポートをカスタマイズする場合は、「編集」アクションを選択します。

SQL 問合せは、Warehouse Builder Design Repository へのデータベース・リンクを参照する必要があります。Warehouse Builder Browser のインストール時に作成した default_owb_link を使用できます。レポートの SQL 問合せでは、PUBLIC データベース・リンク、またはレポートが存在するアプリケーション・スキーマ内のリンクのみをコールできます。

レポートは、Warehouse Builder Browser スキーマ以外のスキーマにあってもかまいませんが、Warehouse Builder Browser スキーマで実行する必要があります。ポータル・レポートの実行権限を付与するには、「Oracle Portal」ホームページ→「構築」→「プロバイダ」タブ→「Portal DB プロバイダ」→「レポート」→「アクセス」タブを選択します。

次の問合せにより、内部の情報システムを一覧表示する簡単なプロジェクト・レポートが表示されます。

```
select * from all_iv_information_systems@default_owb_link where project_id = :id
```

「Warehouse Builder Browser Navigation」ページから実行すると、マーカー `:id` が自動的に適切な値で置換されます。

次の環境で、レポートを実行できることを確認します。

- **SQL*Plus:** レポートを所有するユーザーとしてログオンします。所有者は、レポートの「開発」タブに表示されます。SQL* Plus では、マーカー `:id` が有効な project_id で置換されます。
- **Oracle Portal:** 「開発」タブで、「カスタマイズ」リンクを選択し、Id とラベルされた編集ボックスに有効な project_id を入力して、「レポートの実行」をクリックします。

第 V 部

Warehouse Builder の統合と拡張

この部には、次の章があります。

- 第 19 章「Warehouse Builder 機能の拡張」
- 第 20 章「データの品質 : Name and Address のクレンジング」
- 第 21 章「Warehouse Builder での SAP R/3 データの使用」
- 第 22 章「その他の BI 製品と Warehouse Builder のメタデータの統合」

19

Warehouse Builder 機能の拡張

この章では、現在の Warehouse Builder 機能を拡張する方法について説明します。この章では、次のトピックについて説明します。

Oracle Metabase（OMB）Plus について

この章では、Oracle Metabase（OMB）Plus を使用して実行できる、Warehouse Builder 機能の拡張タスクについて説明します。

OMB Plus は、Oracle Warehouse Builder 用の、柔軟性の高い高水準のコマンドライン・メタデータ・アクセス・ツールです。OMB Plus では、変数サポート、条件とループによる制御構造、エラー処理、標準ライブラリ・プロシージャなどを含む構文構成体を記述できます。また、Warehouse Builder メタデータ・リポジトリおよび Runtime Repository にアクセスできます。OMB Plus を使用することで、リポジトリをナビゲートし、リポジトリ内のメタデータを管理および操作できます。

この章の後半では、OMB Scripting Language 言語を使用して、Warehouse Builder で専門的なタスクを実行する方法について説明します。個々の OMB Plus コマンドの構文情報は、『Oracle Warehouse Builder スクリプト・リファレンス』を参照してください。

ユーザー定義プロパティ

Warehouse Builder では、ユーザー定義プロパティ（UDP）を使用して、その Design Repository を拡張することができます。各リポジトリ・オブジェクトには事前定義済のプロパティ・セットがあります。UDP を作成することで、カスタム・プロパティをオブジェクトに追加できます。

UDP の作成と管理には、Oracle MetaBase（OMB）Scripting Language を使用します。UDP は、OMB または Warehouse Builder Database Client で表示できます。クライアントの場合、UDP はプロパティ・シートと Warehouse Builder Browser に表示されます。

UDP の動作方法はシステム固有のプロパティと同じで、オブジェクトのロック、マルチユーザー・アクセス、トランザクションおよびセキュリティに関する Warehouse Builder ルールに従います。オブジェクトのメタデータ・スナップショットを取得するとき、Warehouse Builder では関連する UDP が取得されます。また、Metadata Loader（MDL）を使用して、UDP をインポートおよびエクスポートできます。

ユーザー定義プロパティの管理

Warehouse Builder 管理者は、エンド・ユーザーが Warehouse Builder Design Repository にアクセスする前に、すべてのユーザー定義プロパティをリポジトリに定義する必要があります。この作業の際には、既存オブジェクトの UDP に対する値の設定は行いません。

すべてのユーザー定義プロパティは中央の Design Repository に登録し、クライアントにローカル登録しないでください。UDP を作成または編集するには、Warehouse Builder リポジトリにシングル・ユーザー・アクセスする必要があります。

ユーザー定義プロパティの作成および操作には、次の OMB Plus コマンドを使用できます。

- OMBDEFINE
- OMBDESCRIBE

UDP の作成およびコミット時には、OMB により次の検証が行われます。

- ネームスペース・チェック。同一のクラス階層に、同じ名前を持つプロパティが複数定義されていないことが確認されます。Warehouse Builder で将来的に導入される名前と競合しないように、ユーザー定義プロパティには接頭辞 'UDP_' が挿入されます。
- プロパティ値のチェック。定義されたデフォルト値が、指定されているデータ型と矛盾しないことが確認されます。
- ユーザー・アクセス・チェック。リポジトリ全体に対して、シングル・ユーザー・アクセス権があることが確認されます。

OMBDEFINE

OMBREDEFINE CLASS_DEFINITION は、UDP の操作に使用できます。ディメンション・オブジェクトに UDP を作成するには、次の文を発行します。この文では、UDP 定義がクラス定義 'DIMENSION' に追加されます。

```
OMBREDEFINE CLASS_DEFINITION 'DIMENSION_TABLE'
     ADD PROPERTY_DEFINITION 'UDP_Dim' SET PROPERTIES (TYPE, DEFAULT_VALUE)
     VALUES ('INTEGER', '100')
```

次のコマンドは、プロパティを 'COLUMN' 型に追加します。このプロパティは、表、ビュー、マテリアライズド・ビュー、外部表および順序プロパティ・シートに表示されます。

```
OMBREDEFINE CLASS_DEFINITION 'COLUMN'
     ADD PROPERTY_DEFINITION 'UDP_Col' SET PROPERTIES (TYPE, DEFAULT_VALUE)
     VALUES ('STRING', 'foo')
```

次のコマンドは、指定したプロパティの名前またはデフォルト値を変更します。

```
OMBREDEFINE CLASS_DEFINITION 'TABLE' MODIFY PROPERTY_DEFINITION 'UDP_Tbl'
     SET PROPERTIES (DEFAULT_VALUE, BUSINESS_NAME)
     VALUES ('99', 'UDP_Tbl')
```

次のコマンドは、'Table' クラスから UDP_Tbl プロパティを削除します。このアクションは元に戻すことができないため、非常に危険であり、あまりお奨めできません。このプロパティ定義に対してカスタマイズされた、リポジトリ内のすべてのプロパティ値が、回復不能になります。

```
OMBREDEFINE CLASS_DEFINITION 'TABLE' DELETE PROPERTY_DEFINITION 'UDP_Tbl'
```

OMBDESCRIBE

OMBDESCRIBE はメタデータ要素の属性を表示するために、クラス定義に対して使用できます。OMBDESCRIBE を使用して、指定したオブジェクト・タイプのユーザー定義プロパティを一覧表示できます。たとえば、次のコマンドはディメンションのユーザー定義プロパティを一覧表示します。

```
OMBDESCRIBE CLASS_DEFINITION 'DIMENSION_TABLE' GET PROPERTY_DEFINITIONS
```

また、OMBDESCRIBE を使用することで、プロパティ定義のプロパティを確認することもできます。たとえば、ディメンション・クラス定義にある UDP_Dim というユーザー定義プロパティのデータ型、デフォルト値およびビジネス名を取得するには、次のコマンドを発行します。

```
OMBDESCRIBE CLASS_DEFINITION 'DIMENSION_TABLE' PROPERTY_DEFINITION  'UDP_Dim'
     GET PROPERTIES (TYPE, DEFAULT_VALUE, BUSINESS_NAME)
```

CHAR、NUMBER、DATE などのユーザー定義プロパティのデータ型を指定できます。

ユーザー定義プロパティの表示

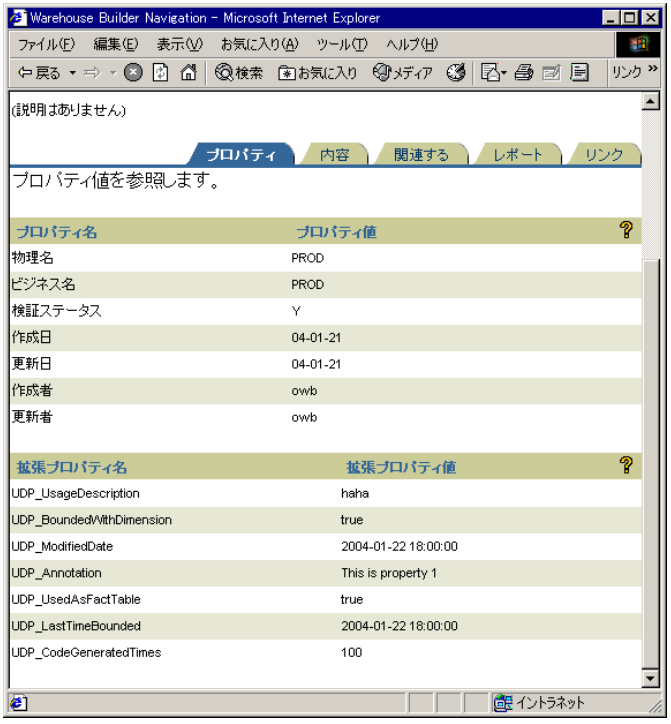
ユーザー・インタフェースでは、次のコンポーネントに UDP を表示できます。

- Warehouse Builder Design Client
- Warehouse Builder Design Browser

Warehouse Builder Design Client

スクリプトを使用して UDP を作成した後は、関連するプロパティ・シートの「ユーザー定義プロパティ」タブに UDP が表示されます。UDP が作成されていない場合は、「ユーザー定義プロパティ」タブは表示されません。たとえば、最初は「ディメンション・プロパティ」シートに「ユーザー定義プロパティ」タブはありません。しかし、UDP をディメンションに追加した後は、図 19-1 に示すように「ディメンション・プロパティ」シートに UDP が表示されます。

図 19-1 ユーザー定義プロパティのサンプル・プロパティ・シート



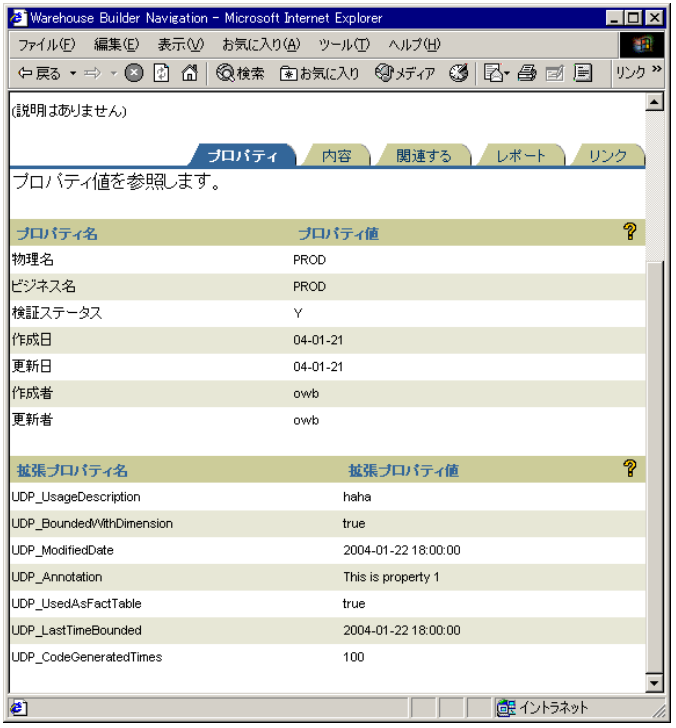
このタブでは、左側のパネルに、オブジェクトのナビゲーション・ツリーがあります。右側のパネルには、関連するすべてのオブジェクトと対応する拡張プロパティが表示されます。また、スクリプトで UDP を作成したときに指定した値とカテゴリが表示されます。「ユーザー定義プロパティ」タブでは、値は変更できますがカテゴリの変更はできません。カテゴリの編集には、OMB Plus を使用する必要があります。

Warehouse Builder Design Browser

UDP は Warehouse Builder Browser にも表示されます。Warehouse Builder Browser は、Warehouse Builder 用のメタデータ管理とレポート・ポータルです。Browser には、オブジェクトのプロパティ、オブジェクト間の関係および系統と影響分析レポートが表示されます。

特定のオブジェクトに UDP を定義した場合は、図 19-2 に示すように、その UDP 名と値が「拡張プロパティ名」および「拡張プロパティ値」に表示されます。

図 19-2 ユーザー定義プロパティのサンプル・プロパティ・シート



他のリポジトリへの UDP の転送

特定のリポジトリから別のリポジトリに変更を伝播する第 1 の方法は、MDL の使用です。MDL を使用して、ユーザー定義プロパティのメタデータ定義とその内容をインポートおよびエクスポートすることができます。

UDP のエクスポート

UDP のエクスポートは、コマンドラインからのみ可能です。MDL 制御ファイルでは、オプションとして DEFINITIONFILE=*filename* を指定して、メタデータ定義をエクスポートします。次に例を示します。

```
## Sample Export file
USERID=UserName/Password@HostName:PortID:OracleServiceName
#
DEFINITIONFILE=Drive:¥DirectoryName¥filename.mdd

FILE=Drive:¥DirectoryName¥filename.mdl
LOG=Drive:¥DirectoryName¥filename.log
```

UDP のインポート

UDP のインポートは、コマンドラインからのみ可能です。インポート時には、すべてのオブジェクトのユーザー定義プロパティが更新されます。MDL 制御ファイルでは、オプションとして DEFINITIONFILE=filename を指定して、メタデータ定義をインポートします。次に例を示します。

```
## Sample Import file
USERID=UserName/Password@HostName:PortID:OracleServiceName
#
DEFINITIONFILE=Drive:¥DirectoryName¥filename.mdd

FILE=Drive:¥DirectoryName¥filename.mdl
LOG=Drive:¥DirectoryName¥filename.log
```

UDP をインポートする際は、次の検索条件のいずれかを使用します。

- **ユニバーサル ID:** メタデータ定義には、ユニバーサル・オブジェクト ID（UOID）が含まれます。UOID はリポジトリ間でオブジェクトを一意に識別します。UOID を使用して MDL ファイルをインポートする場合、MDL では UOID を基準にメタデータ定義が検索されます。ソース MDL ファイルのメタデータ定義名がリポジトリのメタデータ定義と異なる場合は、名前が変更されます。

- **物理名 :** 物理名を使用してメタデータ定義が検索されます。

インポート・モードに関係なく、インポート済のメタデータ定義がリポジトリにない場合は追加され、リポジトリにある場合は更新されます。リポジトリにあるメタデータ定義が、MDL により削除されることはありません。

メタデータ定義の更新時に、名前が異なる場合（検索条件は UOID）にのみオブジェクト名の変更とデフォルト値の更新が行われます。データ型は変更されません。

PL/SQL でのセキュリティ管理

この項では、次のトピックについて説明します。

リポジトリ・ユーザーのメンテナンス

リポジトリ所有者により登録された識別可能な複数の Warehouse Builder ユーザーが、中央にある同一のリポジトリ・スキーマにアクセスできます。

Warehouse Builder には、次のメンテナンス・タスクを実行するためのユーティリティ・プロシージャが含まれています。

- ロール保護パスワードの更新

  リポジトリ所有者は、ロールの保護パスワードを明示的に使用するわけではありませんが、パスワードは頻繁に変更することをお薦めします。

  リポジトリ所有者は、SQL Plus を使用してリポジトリ・スキーマに接続し、次の文を発行する必要があります。

  ```
  Call WBSecurityHelper.updateRolePwd('newpwd');
  ```

  'newpwd' は、リポジトリ所有者がロール保護用に選択した新しいパスワードです。変更されたパスワード暗号は、OWB_ROLE_INFO という表に表示されます。

- リポジトリ・ユーザーの登録

  Warehouse Builder ユーザーは、リポジトリと同じデータベース・インスタンスにデータベース・ユーザーとして配置されている必要があります。データベース・ユーザーのデフォルト・ロールは ALL にしないでください。'After user xxx DEFAULT Role x' を使用して、デフォルト・ロール・プロパティを設定します。

リポジトリ・ユーザーを登録するには、SQL Plus で次の文を実行します。

```
call WBSecurityHelper.registerOWBUser ('username')
```

- リポジトリ・ユーザーの登録解除

  リポジトリ・ユーザーを登録解除するには、SQL Plus で次の文を実行します。

  ```
  Call WBSecurityHelper.unregisterOWBUser('username');
  ```

- すべてのリポジトリ・ユーザーの表示

  リポジトリ所有者は、リポジトリがあるデータベースに接続し、次の文を発行する必要があります。

  ```
  Set serveroutput on;
  Call WBSecurityHelper.listOWBUser( );
  ```

  `listOWBUsers()` では、ユーザー・インタフェースへのデータのダンプ出力に `DBMS_OUTPUT.put_line` が使用されるため、`Set serveroutput on` は必須です。それ以外の場合、`DBMS_OUTPUT.put_line` は出力を中間のデータ構造にのみダンプ出力します。

  これらのユーザー・メンテナンス・タスクは、リポジトリ所有者がリポジトリ内で実行する必要があります。

PL/SQL セキュリティ・パッケージのプラグイン・インタフェース仕様

この項では、Warehouse Builder に付属する PL/SQL セキュリティ・パッケージのプラグイン・インタフェース仕様について説明します。各自の Warehouse Builder Design Repository にプラグイン・インタフェースを実装して、Warehouse Builder に用意されているダミーの PL/SQL パッケージ本体を置換する必要があります。インタフェース仕様とダミー実装はリポジトリ・スキーマからも使用できます。

次の項で説明するファンクションとプロシージャは、ファンクション `isSecurityServiceCustomized` が戻り値として 1 を返すように変更されている場合、すべての Warehouse Builder 操作に対して起動されます。パッケージをカスタマイズしない場合は、`isSecurityServiceCustomized` のデフォルトの戻り値は 0 です。

Warehouse Builder のインストール時は、次の項で説明するファンクションとプロシージャは空です。これらのファンクションとプロシージャを実装するには、それらにセキュリティ・ロジックを実装し、操作、ユーザー、オブジェクト・タイプなどのオブジェクトを接続するための独自のセキュリティ・データ・モデルを構築する必要があります。

```
CREATE OR REPLACE PACKAGE WBSecurityServiceImpl AS

FUNCTION isSecurityServiceCustomized RETURN NUMBER;

/*
```

ファンクションの使用方法 : isSecurityServiceCustomized() では、セキュリティ・サービスの実装が区別されます。PL/SQL セキュリティ実装をカスタマイズするか、Warehouse Builder に用意されている実装を使用します。

戻り値 : この仕様の PL/SQL パッケージを実装する場合は 1 を、カスタマイズする場合は 0 を返します。

```
*/
PROCEDURE securityCheckForCreation(outcome OUT NUMBER,
userId IN VARCHAR2,
objectUOIDOperationInvokedOn IN VARCHAR2,
status IN VARCHAR2,
parentModuleUOID IN VARCHAR2,
parentProjUOID   IN VARCHAR2,
repos_Schema IN VARCHAR2,
objectType IN NUMBER);

/*
```

Procedure: securityCheckForCreation: 操作に対するセキュリティ・チェックの作成に使用します。ユーザーがオブジェクトを作成しようとすると、このプロシージャがコールされ、作成操作が可能かどうかが実装で確認されます。

引数の説明 :

Outcome:1: 作成操作は可能です。

Outcome:0: 作成操作は不可能です。

userId: ログイン・ユーザーのデータベース・ユーザー名。

objectUOIDOperationInvokedOn: 新しいオブジェクトを作成する親フォルダの UOID。プロジェクトやスナップショットなどの親フォルダを持たないオブジェクトの場合、この引数は NULL です。

status: モジュールの属性で、この仕様では WB_DEV_STATUS、WB_QA_STATUS および WB_PROD_STATUS が定義されています。この属性は、モジュールのステータスを説明しています。プロジェクト、モジュール、スナップショットなどの、階層内でどのモジュールの子にもなっていないオブジェクトの場合、この引数は NULL です。

parentModuleUOID: モジュールの UOID。プロジェクトやスナップショットなどの、階層内でどのモジュールの子にもなっていないオブジェクトの場合、この引数は NULL です。

parentProjUOID: プロジェクトの UOID。プロジェクトやスナップショットなどの、階層内でどのモジュールの子にもなっていないオブジェクトの場合、この引数は NULL です。

repos_schema: 操作の対象となる中央のリポジトリ・スキーマ名。

objectType: 作成するオブジェクトのタイプ。この仕様で定義されているオブジェクト・タイプの定数のいずれかになります。

```
*/
PROCEDURE securityCheck(outcome OUT NUMBER,
           userId IN VARCHAR2,
    operation IN NUMBER,
    objectUOIDOperationInvokedOn IN VARCHAR2,
    objectTypeOperationInvokedOn IN NUMBER,
    status IN VARCHAR2,
    parentModuleUOID IN VARCHAR2,
    parentProjUOID   IN VARCHAR2,
    repos_Schema IN VARCHAR2);
/*
```

PROCEDURE: securityCheck は、次の操作に対して使用します。

```
WB_EDIT
WB_DELETE
WB_VALIDATE
WB_GENERATION
WB_VERSION
```

これらの操作のいずれかを起動すると、このプロシージャがコールされ、その操作が可能かどうかが確認されます。

引数の説明 :

Outcome:1: 作成操作は可能です。

Outcome:0: 作成操作は不可能です。

userId: ログイン・ユーザーのデータベース・ユーザー名。

operation: 前述の定数のいずれかになります。

objectUOIDOperationInvokedOn: ターゲット・オブジェクトの UOID。

objectTypeOperationInvokedOn: 操作が起動されるオブジェクトのタイプ（この仕様で定義されているオブジェクト・タイプの定数のいずれかになります)。

status: モジュールの属性。

WB_DEV_STATUS

WB_QA_STATUS,

WB_PROD_STATUS

この属性は、モジュールのステータスの説明に使用します。プロジェクト、モジュールまたはスナップショットで操作を実行する場合、この引数は NULL です。

parentModuleUOID: モジュールの UOID。モジュールに対して操作を起動する場合、objectUOIDOperationInvokedOn と parentModuleUOID は同じです。階層内でどのモジュールの子にもなっていないプロジェクトまたはスナップショットに対して操作を起動する場合、この引数は NULL です。

parentProjUOID: プロジェクトの UOID。プロジェクトに対して操作を起動する場合、objectUOIDOperationInvokedOn と parentProjUOID は同じです。階層内でどのプロジェクトの子にもなっていないスナップショットに対して操作を起動する場合、この引数は NULL です。

repos_schema: 操作の対象となる中央のリポジトリ・スキーマ名。

```
*/
PROCEDURE securityCheckForService(outcome OUT NUMBER,
          userId IN VARCHAR2,
    serviceOp IN NUMBER,
    moduleUOID IN VARCHAR2,
    projUOID   IN VARCHAR2,
    repos_Schema IN VARCHAR2);

/*
```

PROCEDURE securityCheckForService は、次のサービス操作で使用します。

WB_DEPLOY

WB_MDL_IMPORT

WB_MDL_EXPORT

WB_BRIDGE_IMPORT

WB_BRIDGE_EXPORT

WB_SOURCE_IMPORT

WB_RUNTIME_EXECUTE

WB_SNAPSHOT_RESTORE

引数の説明 :

Outcome:1: 作成操作は可能です。

Outcome:0: 作成操作は不可能です。

userId: ログイン・ユーザーのデータベース・ユーザー名。

serviceOp: 前述の定数のいずれかになります。

moduleUOID: serviceOp で指定された操作の起動対象となるモジュールの UOID。serviceOp が WB_DEPLOY、WB_BRIDGE_EXPORT、WB_RUNTIME_EXECUTE の場合は、結果的に NULL になります。WB_MDL_IMPORT、WB_MDL_EXPORT、WB_SOURCE_IMPORT、WB_SNAPSHOT_RESTORE の場合は有効です。

projUOID: serviceOp で指定された操作の起動対象となるプロジェクトの UOID。serviceOp が WB_DEPLOY、WB_BRIDGE_EXPORT、WB_RUNTIME_EXECUTE の場合は、結果的に NULL になります。WB_MDL_IMPORT、WB_MDL_EXPORT、WB_SOURCE_IMPORT、WB_SNAPSHOT_RESTORE の場合は有効です。

repos_schema: 操作の対象となる中央のリポジトリ・スキーマ名。

*/

--BEGIN CONSTANT DEFINITION

```
CUSTOM_SHARED_LIBRARY CONSTANT VARCHAR2(100):='9E012195D16211D48D7100B0D02A59E8';
/*
```

CUSTOM_SHARED_LIBRARY: 事前定義済の Warehouse Builder カスタム共有ライブラリ・フォルダに対する UOID 定数。ユーザー・インタフェースでは、このフォルダは「カスタム」という名前で「パブリック変換」ノードの下に配置されます。

このフォルダには、ユーザーが作成したグローバル共有ライブラリの変換がすべて含まれます。

カスタム・フォルダに変換を作成する権限を持つユーザーを制御するには、プロシージャ securityCheckForCreation の引数 objectUOIDOperationInvokedOn がこの定数と同じであるかどうかを確認します。共有ファンクションまたはプロシージャで WB_EDIT や WB_DELETE などの操作を起動できるユーザーを制御するには、プロシージャ securityCheck の引数 parentModuleUOID がこの定数と同じであるかどうかを確認します。

このモジュールは事前定義されているため、ステータスを変更したり、このモジュールのステータスをアクセス制御に適用することはできません。

*/

--Definition of constants for all basic operations

```
WB_EDIT   CONSTANT INTEGER := 0;
WB_DELETE   CONSTANT INTEGER := 1;
WB_REFERENCE   CONSTANT INTEGER := 2;
WB_CREATE        CONSTANT INTEGER := 3;
WB_VALIDATE   CONSTANT INTEGER := 4;
WB_GENERATION   CONSTANT INTEGER := 5;
WB_VERSION  CONSTANT INTEGER := 6;
/*
```

この操作定数グループを SecurityCheck プロシージャで使用します。引数 operation にこの定数の 1 つを使用します。また、この定数を使用して各操作を起動できるユーザーを制御することもできます。

*/

--Definition of constants for all service type operations

```
WB_DEPLOY   CONSTANT INTEGER := 100;
WB_MDL_IMPORT   CONSTANT INTEGER := 101;
WB_MDL_EXPORT   CONSTANT INTEGER := 102;
WB_BRIDGE_IMPORT CONSTANT INTEGER := 103;
WB_BRIDGE_EXPORT CONSTANT INTEGER := 104;
WB_SOURCE_IMPORT CONSTANT INTEGER := 105;
WB_RUNTIME_EXECUTE  CONSTANT INTEGER :=106;
WB_SNAPSHOT_RESTORE   CONSTANT INTEGER := 107;
/*
```

このサービス操作定数グループを SecurityCheckForService プロシージャで使用します。引数 serviceOp にこの定数の 1 つを使用します。また、この定数を使用して各サービス操作を起動できるユーザーを制御することもできます。

*/

--Definition of the module status

```
WB_DEV_STATUS  CONSTANT VARCHAR2(100) := 'DEV_STATUS';
WB_QA_STATUS  CONSTANT VARCHAR2(100) := 'QA_STATUS';
WB_PROD_STATUS  CONSTANT VARCHAR2(100) := 'PROD_STATUS';
/*
```

このモジュール・ステータス定数グループを securityCheckForCreation プロシージャおよび SecurityCheck プロシージャで使用します。モジュールの子を作成する場合（securityCheckForCreation）やモジュールまたはモジュールの子に対して操作を起動する場合（SecurityCheck）は、これらの定数を使用します。それ以外の場合、引数 status は NULL です。また、この定数を使用してモジュールのステータスに基づくアクセス制御を実装することもできます。

*/

--Definition of object type

```
WB_PROJECT   CONSTANT INTEGER := 1;
WB_ORACLE_MODULE   CONSTANT INTEGER := 2;
WB_GATEWAY_MODULE CONSTANT INTEGER := 3;
WB_SAP_MODULE  CONSTANT INTEGER := 4;
WB_FLAT_FILE_MODULE CONSTANT INTEGER := 5;
WB_SHARED_MODULE  CONSTANT INTEGER := 6;
WB_REPOS_MODULE   CONSTANT INTEGER := 7;
WB_COLLECTION  CONSTANT INTEGER := 8;
WB_WAREHOUSE   CONSTANT INTEGER := 9;
```

```
WB_TABLE   CONSTANT INTEGER := 10;
WB_VIEW    CONSTANT INTEGER := 11;
WB_MATERIALIZED_VIEW  CONSTANT INTEGER := 12;
WB_SEQUENCE   CONSTANT INTEGER := 13;
WB_DIMENSION_TABLE  CONSTANT INTEGER := 14;
WB_CUBE_TABLE   CONSTANT INTEGER := 15;
WB_FLAT_FILE   CONSTANT INTEGER := 16;
WB_PACKAGE   CONSTANT INTEGER := 17;
WB_TRANSFORMATION   CONSTANT INTEGER := 18;
WB_MAPPING   CONSTANT INTEGER := 19;
WB_MIV_MODULE   CONSTANT INTEGER := 20;
WB_CONNECTOR   CONSTANT INTEGER := 21;
WB_LOCATION   CONSTANT INTEGER := 22;
WB_RUNTIME_REPOSITORY   CONSTANT INTEGER := 23;
WB_BUSINESS_AREA CONSTANT INTEGER := 24;
WB_INTELLIGENCE_MODULE CONSTANT INTEGER := 25;
WB_PROCESS_FLOW  CONSTANT INTEGER := 26;
WB_PROCESS_FLOW_MODULE  CONSTANT INTEGER := 27;
WB_PROCESS_FLOW_PACKAGE CONSTANT INTEGER:=  28;
WB_QUERY_OBJECT  CONSTANT INTEGER:=  29;
WB_ADVANCED_QUEUE  CONSTANT INTEGER:=  30;
WB_EXTERNAL_TABLE  CONSTANT INTEGER:=  31;
WB_REPORT   CONSTANT INTEGER:=  32;
WB_REPORT_GROUP   CONSTANT INTEGER:=  33;
WB_REPORT_MODULE  CONSTANT INTEGER:=  34;
WB_OBJECT_TYPE   CONSTANT INTEGER:=  35;
WB_SNAPSHOT  CONSTANT INTEGER:=  36;
/*
```

オブジェクト・タイプ定数グループは securityCheckForCreation プロシージャおよび securityCheck プロシージャで使用します。securityCheckForCreation の引数 objectType および securityCheck の引数 objectTypeOperationInvokedOn は、これらの定数のいずれかになります。この定数を使用して、どのユーザーがどのオブジェクト・タイプを作成できるかまたはどのユーザーが WB_EDIT や WB_DELETE などの操作をどのオブジェクト・タイプに対して起動できるかを制御できます。

```
*/

/*
```

引数の数が多いため、使用されるプロシージャはユーザーが操作を起動するオブジェクトがプロジェクトの子であるか、またはモジュールの子であるかによって異なります。その情報を次の項で示します。

```
*/

--Children of Project

/*
```

WB_SNAPSHOT および WB_PROJECT はプロジェクトの子ではありません。タイプ定数のリストにある他のオブジェクトはプロジェクトの子です。

*/

--Children of Module

/* The following are not children of any module: WB_PROJECT、WB_ORACLE_MODULE、WB_GATEWAY_MODULE、WB_SAP_MODULE、WB_FLAT_FILE_MODULE、WB_SHARED_MODULE、WB_REPOS_MODULE、WB_COLLECTION、WB_WAREHOUSE、WB_MIV_MODULE、WB_LOCATION、WB_CONNECTOR、WB_RUNTIME_REPOSITORY、WB_BUSINESS_AREA、WB_INTELLIGENCE_MODULE、WB_PROCESS_FLOW_MODULE、WB_REPORT_MODULE、WB_SNAPSHOT

Other object types can be the children of module.

*/

```
END WBSecurityServiceImpl;
```

パッケージ仕様の定数定義

最初に、基本的な操作の定数定義について説明します。Warehouse Builder では、次の操作に対して、各操作がオブジェクト・インスタンス・レベルで可能かどうかが確認されます。

- **WB_EDIT**: 編集（更新）操作。
- **WB_DELETE**: 削除操作。
- **WB_REFERENCE**: 他のオブジェクト内にあるオブジェクトを参照する操作。たとえば、表をマッピングにドラッグするときの操作。
- **WB_CREATE**: 作成操作。
- **WB_VALIDATE**: オブジェクト定義の精度のチェックに使用する検証操作。
- **WB_GENERATION**: Warehouse Builder での定義用 SQL スクリプトの生成操作。
- **WB_VERSION**: オブジェクトに対するバージョニング操作。スナップショット・バージョンへのオブジェクトの追加、スナップショット・バージョンからのオブジェクトの削除、バージョニングされた既存オブジェクトの新規コピーでの置換などを実行できます。

次に、サービス操作の定数定義について説明します。Warehouse Builder では、次の操作に対して、各操作がシステム全体レベルで可能かどうかが確認されます。サービス操作を起動する権限があるユーザーは、すべてのオブジェクトに対してその操作を起動する必要があります。WB_MDL_IMPORT、WB_MDL_EXPORT、WB_BRIDGE_IMPORT、WB_SOURCE_IMPORT、WB_SNAPSHOT_RESTORE などの操作では、プロジェクトまたはモジュール・レベルでのセキュリティが提供されます。

- **WB_DEPLOY**: 最初に SQL スクリプトを生成し（スクリプトがない場合）、次に生成したスクリプトをランタイム・データベース・スキーマに配布します。
- **WB_MDL_IMPORT**: メタデータをフラット・ファイル・フォーマットから Warehouse Builder にインポートします。
- **WB_MDL_EXPORT**: Warehouse Builder からメタデータをフラット・ファイルにエクスポートします。
- **WB_BRIDGE_IMPORT**: 別のインポート・ソリューションとして、MDL とは異なるフォーマットを使用する Discoverer、OLAP、CWM およびサードパーティへのブリッジが用意されています。
- **WB_BRIDGE_EXPORT**: 別のエクスポート・ソリューションとして、MDL とは異なるフォーマットを使用する Discoverer、OLAP、CWM およびサードパーティへのブリッジが用意されています。
- **WB_SOURCE_IMPORT**: 表やビューなどのデータベース・オブジェクトのメタデータ情報を、指定したデータベース・リンクから Warehouse Builder にインポートします。
- **WB_RUNTIME_EXECUTE**: 配布された SQL スクリプトをランタイム・データベース・スキーマから実行します。
- **WB_SNAPSHOT_RESTORE**: 特定のスナップショットから設計領域を、個別にまたは完全にリストアします。

次は、モジュール・ステータスの定数定義です。モジュールのステータスは、モジュールのプロパティ・ページを使用して Warehouse Builder Design Client から変更できます。

- **WB_DEV_STATUS**: 開発ステータスです。
- **WB_QA_STATUS**: 品質保証ステータスです。
- **WB_PROD_STATUS**: 製品ステータスです。

次は、オブジェクト・タイプの定数定義です。

- WB_PROJECT: プロジェクト
- WB_ORACLE_MODULE: Oracle モジュール
- WB_GATEWAY_MODULE: Gateway モジュール
- WB_SAP_MODULE: SAP モジュール
- WB_FLAT_FILE_MODULE: フラット・ファイル・モジュール
- WB_SHARED_MODULE: 共有モジュール
- WB_REPOS_MODULE: リポジトリ・モジュール
- WB_COLLECTION: コレクション
- WB_WAREHOUSE: ウェアハウス・モジュール
- WB_TABLE: 表
- WB_VIEW: ビュー
- WB_MATERIALIZED_VIEW: マテリアライズド・ビュー
- WB_SEQUENCE: 順序
- WB_DIMENSION_TABLE: ディメンション表
- WB_CUBE_TABLE: キューブ表
- WB_FLAT_FILE: フラット・ファイル
- WB_PACKAGE: パッケージ
- WB_TRANSFORMATION: 変換
- WB_MAPPING: マッピング
- WB_MIV_MODULE: MIV モジュール
- WB_CONNECTOR: コネクタ
- WB_LOCATION: ロケーション
- WB_RUNTIME_REPOSITORY: Runtime Repository
- WB_BUSINESS_AREA: ビジネス領域
- WB_INTELLIGENCE_MODULE: インテリジェンス・モジュール
- WB_PROCESS_FLOW: プロセス・フロー
- WB_PROCESS_FLOW_MODULE: プロセス・フロー・モジュール
- WB_PROCESS_FLOW_PACKAGE: プロセス・フロー・パッケージ

- WB_QUERY_OBJECT: 問合せオブジェクト
- WB_ADVANCED_QUEUE: アドバンスト・キュー
- WB_EXTERNAL_TABLE: 外部表
- WB_REPORT: レポート
- WB_REPORT_GROUP: レポート・グループ
- WB_REPORT_MODULE: レポート・モジュール
- WB_OBJECT_TYPE: オブジェクト・タイプ
- WB_SNAPSHOT: スナップショット

PL/SQL インタフェースの実装

PL/SQL インタフェースの実装時に、Warehouse Builder Design Client から渡される引数に応じて操作を受け入れるか拒否するかを決めます。また、Warehouse Builder Design Repository には、`ALL_IV_FIRSTCLASS_OBJECTS` というパブリック・ビューが用意されています。このビューには、オブジェクトに関する次の情報が表示されます。

- オブジェクトの UOID とクラス・タイプ
- オブジェクト名とビジネス名
- 作成者と更新者
- 作成タイムスタンプと更新タイムスタンプ

オブジェクトの UOID を指定すると、パブリック・ビューでこれらの情報を参照できます。また、パブリック・ビューでこれらの情報を使用して、操作を可能にするかどうかを決めることができます。

コール元から渡される引数によって、次のような異なるレベルのセキュリティ粒度が適用されます。

- **リポジトリ・レベル :** リポジトリ全体を固定するかどうかについて、引数 `repos_Schema` がチェックされます。
- **プロジェクト・レベル :** プロジェクト全体を固定するかどうかについて、引数 `parentProjUOID` がチェックされます。
- **モジュール・レベル :** モジュール全体を固定するかどうかについて、引数 `parentModuleUOID` がチェックされます。
- **オブジェクト・クラス・タイプ・レベル :** タイプ全体へのアクセスを受け入れるか拒否するかについて、引数 `objectTypeOperationInvokedOn` がチェックされます。
- **オブジェクト・インスタンス・レベル :** オブジェクトの UOID がチェックされます。

- **開発プロセス・レベル :** モジュール全体に対するアクセス制御を決定するために、引数 `status`（開発、品質保証または製品）がチェックされます。ステータスは、Warehouse Builder ユーザー・インタフェースのモジュールのプロパティ・ページで変更可能なすべてのモジュールにある属性です。

考慮事項

セキュリティ・フレームワークを実装するには、独自のセキュリティ・データ・モデルを配布する必要があります。たとえば、オブジェクト・レベルのセキュリティが必要な場合は、どのユーザーがどの操作をオブジェクトに対して実行できるかを定義した表または一連の表を作成する必要があります。また、このデータ・モデルを維持するために独自のプロシージャまたはユーザー・インタフェースを実装する必要もあります。

ここでは、PL/SQL インタフェースの実装に関する考慮事項を示します。

- モジュールのステータスに基づいてアクセス制御を決定する場合、モジュールへの編集アクセスは管理グループに属するユーザーのみに制限することをお薦めします。
- 表などのオブジェクト・インスタンスに対して、検証または生成権限を持ち編集権限を持たないユーザーが実行できるのはオブジェクトの検証または生成のみで、生成したオブジェクトを永続的に維持する権限はありません。
- ユーザーが Warehouse Builder Design Client から操作（操作 A）を起動したときに、関連する操作 B も起動されるケースがたまにあります。この場合、そのユーザーに操作 A を起動する権限が付与されていても、操作 B を起動する権限がなければ、操作 A は失敗することになります。

  この問題を解決するには、ユーザーに操作 A と操作 B の両方の権限を付与するか、どちらの権限も付与しないようにします。

  たとえば、マッピング・エディタで、バウンド・オブジェクトをマッピングに追加ときまたはオブジェクトに対してインバウンド調整を実行する際に、バウンド・オブジェクトの位置へのコネクタがない場合は、Warehouse Builder によりコネクタが自動的に作成されます。このユーザーがマッピング・エディタへのバウンド・オブジェクトの追加またはインバウンド調整を完了できるようにするには、マッピングでの `EDIT` 権限とコネクタの `CREATE` 権限をともに付与する必要があります。
- PL/SQL プロシージャの実装では、コミットまたはロールバック・コマンドをいっさい発行しないでください。

  PL/SQL のセキュリティ関連プロシージャのコールに使用される接続はメタデータ全体の維持にも使用されているため、この接続は Warehouse Builder 専用のトランザクション・マネージャにより制御され、コミットまたはロールバックのタイミングもそこで決められます。

コミットまたはロールバックを起動すると、リポジトリが破損します。PL/SQL プロシージャでデバッグ情報を記録する場合は、自律型トランザクションを使用して別の子プロシージャで実行できます（プロシージャの先頭でプラグマ autonomous_transaction を使用します)。このプロシージャでは、コミットまたはロールバック・コマンドを発行できます。

- ユーザーにオブジェクトの CREATE 権限を付与する場合、そのユーザーにはオブジェクトの EDIT 権限も必要です。オブジェクトが作成された後で、EDIT 権限を取り消して、ポリシーを変更できます。
- PL/SQL セキュリティ・プロシージャの実装時は、Warehouse Builder で作成されたリポジトリ・データベース・オブジェクトに対して DML 操作を実行しないでください。リポジトリが破損するか、Warehouse Builder Design Client が機能不全または機能停止に陥るおそれがあります。

  別のユーザーが Warehouse Builder Design Repository にログインしている間に、セキュリティ用の実装パッケージをインストールすると、「ORA-04061: パッケージ本体 "OWB_REPO_513.WBSECURITYSERVICEIMPL" の既存状態は無効になりました 」という SQL エラーを受け取ることがあります。この場合は、Warehouse Builder を終了し、インストールを再実行する必要があります。
- 次のようなプロシージャがあるとします。

```
PROCEDURE securityCheckForService(outcome OUT NUMBER,
           userId IN VARCHAR2,
    serviceOp IN NUMBER,
    moduleUOID IN VARCHAR2,
    projUOID   IN VARCHAR2,
repos_Schema IN VARCHAR2)
```

  この moduleUOID と projUOID は、Warehouse Builder のモジュールとプロジェクトの UOID を意味します。. セキュリティが適用されるのは、Warehouse Builder Design Repository にあるオブジェクトに対してのみです。

  MDL ファイルをインポートする場合またはリポジトリにないプロジェクトまたはモジュールを含むスナップショットをリストアする場合は、これらのプロジェクトまたはモジュールに対するセキュリティ・チェックはいっさい実行されません。リポジトリ所有者はプロジェクトやモジュールなどのフォルダ・オブジェクトに対して MDL インポートまたはスナップショット・リストアを実行することをお薦めします。

サンプルのセキュリティ・ポリシーの実装

この項では、Warehouse Builder で使用可能な 3 種類の高度なセキュリティ・ポリシーについて説明します。

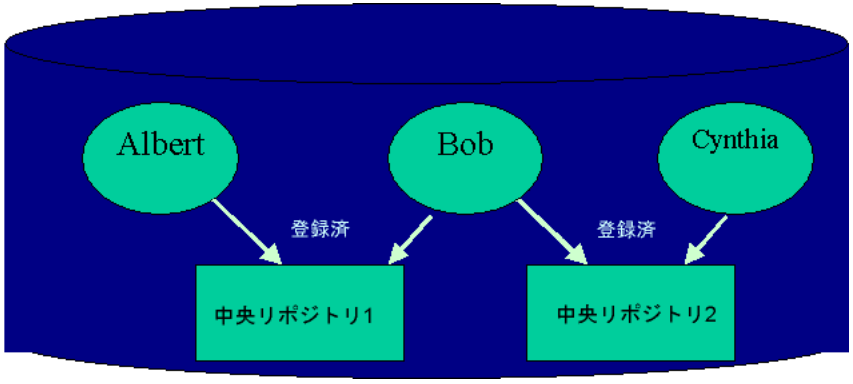
- **プロアクティブ・セキュリティ：** Warehouse Builder では、Warehouse Builder Design Repository に入っているセキュリティ PL/SQL 実装パッケージをカスタマイズしてプラグインすることにより、ユーザー組織の規定するセキュリティ・ルールに従って、ユーザーへのアクセス制御を定めることができます。
- **リアクティブ・セキュリティ：** メタデータの履歴を使用して監査情報を追跡し、こうした監査情報を基にセキュリティ・ポリシーを決定します。
- **データ管理：** Warehouse Builder では、技術管理者でなく、個人または数名のグループが、メタデータの一部を「所有」できます。そのため、メタデータのセキュリティ管理において、メタデータの所有が重要な役割を果たすようになりました。企業のソース、ターゲット、ETL 操作をモデル化するときおよびビジネス・ユーザー用の企業構造をポータルまたは非定型問合せツールから取得するときにメタデータ・リポジトリを使用する場合、データ管理者の役割は重要です。Warehouse Builder では、事前定義済のメタデータ・セキュリティ認可方針にデータ管理者の職責を組み込むことができます。

Warehouse Builder のセキュリティ・アーキテクチャ

この項では、Warehouse Builder のセキュリティ・モデルおよびアーキテクチャについて説明します。また、ツールを使用して既存のインフラストラクチャ・セキュリティ・システムを強化する各種方法について説明します。

Warehouse Builder のサーバー側セキュリティ・アーキテクチャを図 19-3 に示します。

図 19-3 Warehouse Builder のサーバー側セキュリティ・アーキテクチャ



リポジトリ・データベースには 5 つのスキーマがあります。2 つは Warehouse Builder Design Repository で（中央リポジトリまたは Warehouse Builder Design Repository とも呼ぶ）、3 つはリポジトリ・ユーザー、Albert、Bob および Cynthia です。Albert, Bob and Cynthia. リポジトリ・ユーザー・スキーマは、必ず中央リポジトリと同じデータベース・インスタンスに配置します。また、これらの Design Repository リには、リポジトリ・ユーザーの登録および登録解除を管理する管理者を割り当てる必要があります。Warehouse Builder Design Repository の管理者のみが中央リポジトリ・スキーマにアクセスできます。Albert、Bob および Cynthia はデータベースにスキーマを持ちますが、これらのユーザーが Warehouse Builder Design Repository にアクセスするには、対象となる中央リポジトリの管理者によって登録される必要があります。これらのユーザーは Warehouse Builder メタデータ表の所有者でないため、SQL*Plus などのアクセス・ツールを使用してメタデータを変更することはできません。

詳細は、19-8 ページの「リポジトリ・ユーザーのメンテナンス」を参照してください。

カスタマイズ可能なセキュリティ認可フレームワークの使用

Warehouse Builder では、PL/SQL セキュリティ・パッケージ仕様が発行され、中央リポジトリにロードされます。新しい Warehouse Builder Design Repository の作成時にロードされるパッケージ実装では、デフォルトですべての登録ユーザーにすべての権限が付与されます。一方、Warehouse Builder 管理者は独自のセキュリティ・ポリシーを設計し、設計した方針をセキュリティ・パッケージ仕様のフレームワークに従って実装できます。このカスタマイズしたセキュリティ実装は、Warehouse Builder 中央スキーマにプラグインできます。中央リポジトリでのそれ以降のすべての Warehouse Builder アクションは、この新しいセキュリティ・ポリシーを介して渡されます。

オブジェクトを変更しようとすると、使用ツールのあらゆる操作でセキュリティ・チェックが実行されます。たとえば、オブジェクトの作成、更新、削除操作などがチェックされます。セキュリティ・チェックでは、中央リポジトリにある PL/SQL セキュリティ実装パッケージがコールされます。セキュリティ実装パッケージの仕様の詳細は、19-19 ページの「PL/SQL インタフェースの実装」を参照してください。

セキュリティ実装パッケージでは、中央リポジトリにある利用可能な情報を使用できます。たとえば、権限が付与されているかどうかを Warehouse Builder Design Client に示すためのリレーショナル参照表を、Excel スプレッドシートを参照して作成できます。

Warehouse Builder セキュリティ実装パッケージのデフォルトでは、すべての操作が許可されます。表 19-1 にサンプルのセキュリティ権限を示します。

表 19-1 ユーザー Albert の権限

オブジェクト	作成	編集	削除	生成	配布
表	Y	Y	Y	N	N
ビュー	Y	Y	Y	N	N
マテリアライズド・ビュー	Y	Y	Y	N	N
ディメンション	N	N	N	N	N
キューブ	N	N	N	N	N
マッピング	N	Y	N	N	N
プロセス・フロー	N	Y	N	N	N

プロジェクトの固定

プロジェクト MY_PROJECT を固定し、その内容にいっさいアクセスできなくする場合、次の制限が適用されます。

- 固定プロジェクト内では、オブジェクトを作成、編集または削除できません。
- 固定プロジェクト内では、オブジェクトを変更するサービスを起動できません。たとえば、固定プロジェクト内で、MDL インポート、ソース・インポートまたはスナップショット・リストアは実行できません。
- 固定プロジェクト内でのランタイム・プロシージャの配布、エクスポートおよび実行はできます。
- 固定プロジェクト内での検証および生成はできます。
- 固定プロジェクトからスナップショットへのオブジェクトの追加または固定プロジェクトからのオブジェクトの削除はできません。

Warehouse Builder に固定プロジェクトのセキュリティ・ポリシーを実装するには、次のファイルを使用します。これらのファイルはインストール CD の samples/security_feature/frozenproject にあります。

frozenProject.pkb: セキュリティ・ポリシーの実装が保持されます。

frozenProject.sql: 表 19-2 に示すような構造を持つ表が含まれます。管理者はプロジェクトをこの表に挿入し、isFrozen フラグを 1 に設定することでプロジェクトを固定できます。

表 19-2 のようにプロジェクトを固定および固定解除するには、リポジトリ所有者は SQL Plus から次の SQL 文を発行する必要があります。

```
insert into frozen_projects (projectName,isFrozen) values('SAMPLEPROJECT1', '1');
insert into frozen_projects (projectName,isFrozen) values('SAMPLEPROJECT2', '0');
insert into frozen_projects (projectName,isFrozen) values('SAMPLEPROJECT3', '1');
commit;
```

表 19-2 frozenProject.sql のファイル構造の例

プロジェクト名	isFrozen
SampleProject 1	1
SampleProject 2	0
SampleProject 3	1

開発サイクル・ベースのセキュリティ

データ・ウェアハウス・プロジェクトが開発、品質保証および製品フェーズからなるサイクルで進行する場合、Warehouse Builder では、フェーズごとにメタデータを定義および変更するなどセキュリティ・ポリシーを柔軟に決定できます。たとえば、各ユーザーをエンジニアリング、品質保証およびサポート・エンジニアリングの 1 つまたは複数のグループに分類したとします。サイクルの各フェーズに応じて、特定のグループに特定のアクションのみを許可します。表 19-3 に、各グループに許可されたアクションを示します。

表 19-3 **の凡例 :** A = 管理者（リポジトリ所有者）、E = エンジニア、Q = 品質保証、S = サポート・エンジニア

表 19-3 各グループに許可されたアクション

アクション	開発	QA	製品
作成	A、E	A、Q	A、S
編集	A、E	A、Q	A、S
削除	A、E	A、Q	A、S
参照	A、E	A、Q	A、S
検証	A、E	A、Q	A、S
生成	A、E	A、Q	A、S
バージョニング	A、E	A、Q	A、S
MDL のインポート	A、E	A、Q	A、S
ブリッジのインポート	A、E	A、Q	A、S

表 19-3 各グループに許可されたアクション（続き）

アクション	開発	QA	製品
ソースのインポート	A、E	A、Q	A、S
スナップショットのリストア	A、E	A、Q	A、S
ランタイム実行	A、E、Q、S	A、E、Q、S	A、E、Q、S
配置	A、E、Q、S	A、E、Q、S	A、E、Q、S
MDL のエクスポート	A、E、Q、S	A、E、Q、S	A、E、Q、S
ブリッジのエクスポート	A、E、Q、S	A、E、Q、S	A、E、Q、S

たとえば、「製品」とマークされたモジュールでのオブジェクトの作成、編集または削除は、管理者とサポート・エンジニアにのみ許可されています。

このセキュリティ・ポリシーには、次の例外が含まれます。

- 管理者（リポジトリ所有者）のみが、プロジェクトやスナップショットなどのモジュールより高レベルのオブジェクトに対して操作を起動できます。
- 品質保証とサポート・エンジニアリングはモジュールを作成または削除できません。
- すべてのユーザーが、ロケーション、コレクション、コネクタ、ビジネス領域などの非モジュール依存オブジェクトに対して操作を実行できます。
- グローバルな共有ライブラリとその変換オブジェクトには、すべてのユーザーがアクセスできます。
- グループのメンバーのみがモジュールのステータスを他のステータスに変更できます。たとえば、品質保証メンバーのみがモジュールの開発ステータスを「品質保証」から他のステータスに変更できます。また、エンジニアのみがモジュールのステータスを「開発」から他のステータスに変更できます。開発グループでは、一貫性のあるバグ修正ポリシーの実装にこの方法を使用できます。

この特化したセキュリティ・ポリシーを実装するには、次のファイルを使用します。これらのファイルはインストール CD の samples/security_feature/developcycle にあります。

developCyclePolicyLoader.sql: 表 19-3 に指定された情報が含まれます。

developCyclePolicy.sql: ユーザーとグループの対応付けを追跡する表が含まれます。たとえば、各表には表 19-4 に示すような情報が配置されます。

表 19-4 のようにユーザーをグループに割り当てるには、リポジトリ所有者は SQL Plus で次の SQL 文を発行する必要があります。

```
insert into user_group_assignment(userName,groupID) values('ALBERT1', 1);
insert into user_group_assignment(userName,groupID) values('ANDREW', 3);
insert into user_group_assignment(userName,groupID) values('BOB', 2);
insert into user_group_assignment(userName,groupID) values('CYNTHIA', 1);
```

表 19-4 ユーザーとグループの対応付けのサンプル表

ユーザー名	OWB グループ
Albert	エンジニア
Albert	サポート・エンジニア
Bob	品質保証
Cynthia	エンジニア
...	...

developCyclePolicy.pkb: 開発サイクル・ベースのセキュリティ・ポリシーを実装する主要なビジネス・ロジックが含まれます。

実装では、オブジェクトが操作される際に、必ずステータス・フラグが使用されてチェックが行われます。Warehouse Builder では、開発ステータスをモジュールの粒度で定義できるため、開発ステータスはモジュールの子孫であるオブジェクトに対してのみ関連します。

ユーザーが MDL ファイルのインポートまたはリポジトリ内にないプロジェクトやモジュールを含むスナップショットのリストアを試みた場合、Warehouse Builder では、リポジトリ内のオブジェクトに対してのみセキュリティが適用されます。サンプル実装の `developCyclePolicy.pkb` では、ユーザーは MDL インポートまたはスナップショット・リストアによって、プロジェクトまたはモジュール、およびその子オブジェクトを作成できます。リポジトリ所有者は、プロジェクトやモジュールなどのフォルダ・オブジェクトに対して MDL インポートまたはスナップショット・リストアを実行することをお薦めします。

リアクティブ・セキュリティと監査ベースのセキュリティ

リアクティブ・セキュリティの対象は、潜在的なセキュリティ侵害の識別と追跡です。セキュリティに監査を使用することで、所有権が追跡され、品質が確保されます。Warehouse Builder では、オブジェクト・レベルで監査が実行されます。つまり、監査証跡は最小の粒度レベルで定義できます。たとえば、誰がいつ、特定の列、制約またはレベル属性を作成または更新したかを監査できます。このセキュリティ・アーキテクチャを使用してユーザーが設定されている場合、この情報は Warehouse Builder Browser を使用して表示できます。

また、この監査情報を使用して、セキュリティ・ポリシーを実装することもできます。特定のオブジェクトの所有権は、オブジェクトを作成したユーザーを基準に推測できます。これにより、オブジェクトの所有者が属するグループ以外のユーザーはそのオブジェクトに対していっさいのアクションを実行できないというようなセキュリティ・ポリシーを構成できます。たとえば、Marketing グループの Albert が新しい Promotions キューブを作成したとします。Albert は、すべてのメジャーとディメンションを Promotions キューブに追加しました。Inventory グループに属する Bob には、この Promotions キューブを変更、削除、生成または配布する権限はありません。しかし、同じ Marketing グループに属する Cynthia には Albert と同じ権限があり、Promotions キューブに対してあらゆる変更（削除も含めて）を実行できます。

次のセキュリティ実装では、オブジェクトの作成者を追跡するための、オブジェクト（表）を中央リポジトリに配置する必要があります。オブジェクトの作成者は監査列「作成者」に取得され、この列は Warehouse Builder のすべてのオブジェクトに対して作成されます。

オブジェクトは Warehouse Builder Public View システム内に配置できます。Warehouse Builder オブジェクトに使用する汎用コードを記述する場合は、ALL_IV_FIRSTCLASS_OBJECTS ビューへの結合をお薦めします。

パブリック・ビュー内にオブジェクトを配置するには、引数 object_UOID を ALL_IV_FIRSTCLASS_OBJECTS ビューにある UOID に結合する必要があります。UOID は Unique Object Identifier の略称で、全リポジトリにわたってオブジェクトを一意に識別することができます。

基本的な操作には次のポリシーを使用します。

- オブジェクト X を作成できるのは、オブジェクト X を作成しようとしているユーザーが、そのコンテナ・オブジェクト（親オブジェクト）の「作成者」に表示されているユーザーと同じ部門に属する場合のみです。
- オブジェクト X の変更、削除、検証、生成、バージョニングができるのは、その「作成者」がオブジェクト X の作成者と同じ部門に属するユーザーである場合のみです。

- 一般ユーザーは、配布、MDL インポート、MDL エクスポート、ブリッジ・インポート、ブリッジ・エクスポート、ソース・インポート、ランタイム実行、スナップショット・リストアなどのサービス操作を起動できません。この制限はセキュリティ・パッケージ実装を変更することで緩和できますが、インポート・サービスとスナップショット・リストア・サービスについては必ずこの制限を適用します。これらの操作は管理者のみが起動できます。

このセキュリティ・ポリシーでは、各部門に管理者を定義する必要があります。これらのグループ管理者には、MDL インポート、ソース・インポート、スナップショット・リストアなどの重要な操作が許可されます。また、これらの操作では、他の管理者のグループに属すオブジェクトが変更または削除されることがあります。そのため、これは重要なロールです。この制限が課されるのは、MDL インポート、ソース・インポート、スナップショット・リストアが起動されるときに、ファースト・クラス・オブジェクト・レベルではセキュリティ・チェックが実行されないためです。管理者は、表 19-5 に示す構造に従って定義します。

表 19-5 管理者ロール

ユーザー	グループ（外部キー）	IsAdmin
Albert	Marketing	0
Bob	Sales	0
Cynthia	Marketing	1
Derrick	Finance	0
Eeyore	Sales	1

中央の管理者（リポジトリ所有者）には、すべての操作が許可されます。中央の管理者が作成するオブジェクトはどのグループにも属さず、すべてのグループで共有できます。プロジェクトやモジュールなどのコンテナ・オブジェクトを部門間で共有する必要がある場合は、この方法をお薦めします。

次の項では、管理者ロールのルールについて説明します。

- 部門管理者は、一連のメタデータ・オブジェクト（自己の管理部門に属するオブジェクトとは限りません）を変更するサービス操作の実行に責任を持ちます。MDL インポートを使用してオブジェクトを作成した後は、管理者の部門に属するすべてのユーザーがオブジェクトを編集または削除できます。
- 部門管理者は、他部門に属すオブジェクトの変更や削除など、一般ユーザーには制限されているすべての操作を実行することもできます。
- 中央の管理者は、プロジェクトやウェアハウス・モジュールなどのコンテナ・オブジェクトの作成に責任を持ちます。この共有コンテナ内には、すべての部門のユーザーがオブジェクトを作成できます。

- 中央の管理者は、他部門に属すオブジェクトの変更や削除など、一般ユーザーには制限されているすべての操作を実行することもできます。

このポリシーを実装するには、次の 2 つのファイルを使用します。これらのファイルはインストール CD の samples/security_feature/creatorIsOwner にあります。

creatorIsOwner.sql: 表 19-5 に例示された表が含まれます。

creatorIsOwner.pkb: このポリシーを実装するセキュリティ実装が含まれます。

データ管理

オブジェクトの作成者は、特定のサブジェクト領域にあるメタデータの品質に責任を持つデータ管理者の一団に属します。サブジェクト領域には、マーケティング、在庫、販売、予算、製品、財務などがあります。Warehouse Builder では、これらのサブジェクト領域は、Warehouse Builder コレクション・オブジェクトを使用して取得されます。コレクション・オブジェクトは、Warehouse Builder ナビゲーション・ツリーに表示されるモジュール、表、ファクト、マッピング、プロセス・フロー、ディメンション、ファイルなどの実装のオブジェクトへのショートカットで構成されます。これらのコレクション・オブジェクトによって、Warehouse Builder オブジェクトをサブジェクト領域別に分類できます。コレクションは所有権を意味しているわけではありません。メタデータの中には、複数のコレクションに属すものもあります。たとえば、顧客ディメンションのメタデータがマーケティング、販売および受注部門のデータ管理者によって共有される場合があります。

特定のタイプのメタデータに対する所有者が複数の場合は、前項で説明した「作成者が所有者」のポリシーは実装できません。同じメタデータ・オブジェクトに対して、複数のデータ管理者がセキュリティ権限を持てます。また、オブジェクトのメタデータの作成者が、そのメタデータに割り当てられたデータ・スチュワードである必要はありません。たとえば、Warehouse Builder 管理者がスキーマから必要な表をインポートする場合があります。

Warehouse Builder コレクション・オブジェクトは、データ管理セキュリティ・ポリシーにおける中心的な要素です。データ管理へのアクセスは制限し、ユーザーが新しいオブジェクトをそのリストに追加できないようにする必要があります。前述の例では、Warehouse Builder 管理者のみがコレクション・リストへの追加を許可されています。開発者はオブジェクトを作成するたびに管理者に連絡し、作成したオブジェクトをコレクション・リストに追加してもらう必要があります。オブジェクトの作成者にオブジェクトの変更を許可するには、条件をセキュリティ・ポリシーに追加する必要があります。したがって、部門ごとに管理者を置くのではなく、このポリシーでは管理者を 1 人に限定します。

このポリシーを理解するには、コレクション内に定義されるショートカットの動作を理解する必要があります。コレクションに定義されるショートカットは 2 種類あります。1 つは明示的なショートカットで、ユーザーが明示的に作成します。もう 1 つは暗黙的なショートカットで、親なしで子ショートカットが作成されたときに、Warehouse Builder により作成されます。たとえば、コレクションに表を追加すると、その表のすべての親オブジェクトが自動的にコレクションに追加されます。この追加された表は明示的なショートカットで、Oracle モジュールなどの親フォルダは暗黙的なショートカットです。暗黙的なショートカットは、すべての子が削除されたときに削除されます。たとえば、ある表へのショートカットがすべて削除され、その親モジュールへのショートカットが暗黙的であった場合は、モジュールへのショートカットも削除されます。

ユーザーと対応する部門コレクションは、表 19-6 に示す構造に従って設定する必要があります。

表 19-6 ユーザーと対応する部門コレクション

コレクション ID（一意キー制約）	コレクション名	プロジェクト名	コレクション ID（外部キー制約）	ユーザー名
1	販売コレクション	My Project	1	Arthur
2	財務コレクション	My Project	2	Billy
3	マーケティング・コレクション	My Project	3	Caroline
4	AR コレクション	My Project	4	David
5	AP コレクション	My Project	3	Edward
6	在庫コレクション	My Project	4	Frederick
7	製造コレクション	My Project	2	George
8	HR コレクション	My Project	3	George

表 19-6 では、ユーザー George は複数の部門に属すため、両コレクションにあるオブジェクトにアクセスできます。コレクションは、Warehouse Builder に定義されているコレクション名とコレクション ID に一致する必要があります。前述の表を使用することで、セキュリティ管理目的に使用できるコレクションの番号を限定できます。Warehouse Builder では、セキュリティ・インフラストラクチャとは関係ない他のコレクションも定義できます。

このセキュリティ・ポリシーについて説明します。

- ユーザーがオブジェクトを作成すると、そのオブジェクトがコレクションに登録されるまでの間、そのユーザーと中央の管理者のみがオブジェクトを変更、削除、検証または生成できます。
- 中央の管理者のみがコレクションを変更する権限を持ちます。

- 中央の管理者のみが、配布、MDL インポート、MDL エクスポート、ブリッジ・インポート、ブリッジ・エクスポート、ソース・インポート、ランタイム実行、スナップショット・リストアなどのサービス・タスクを実行できます。
- 自分以外のユーザーが作成したオブジェクトが自分のアクセスできるコレクションに明示的に登録されている場合、ユーザーはそのオブジェクトの編集、削除、検証または生成のみを実行できます。暗黙的なショートカットで登録されたモジュールが削除されないようにするには、明示的なショートカットをコレクションで使用する必要があります。
- ユーザーは、コレクションに明示的に登録されていないコンテナにオブジェクトを作成できます。コレクションが明示的に登録された後は、そのコンテナ・オブジェクトを所有または参照するコレクションに対して権限がある場合、ユーザーはこのコンテナでオブジェクトの作成のみ実行できます。たとえば、あるユーザーがモジュール X への暗黙的なショートカットを持っている場合（モジュール X には複数の表があるため）、そのユーザーにはモジュール X にオブジェクトを作成する権限はありません。ただし、そのユーザーがモジュール X に明示的なショートカットを持つ場合は、モジュールに子オブジェクトを作成できます。

このセキュリティ・ポリシーは次の 2 つのファイルに実装されています。これらのファイルはインストール CD の samples/security_feature/stewardship にあります。

stewardship.sql: 表 19-6 に示す構造と同じ構造を持つ表を作成する DDL が含まれます。

stewardship.plb: セキュリティ・パッケージの実装です。

これらはセキュリティ・インタフェースのサンプル実装です。これらを変更または組み合せることで、各組織の要件に合ったより高度なセキュリティ・ポリシーを実装できます。このサンプル実装をテンプレートとして使用し、各自のセキュリティ・ポリシーを実装してください。

20

データの品質 : Name and Address のクレンジング

この章には、第 8 章「マッピング演算子の使用方法」で説明した Name and Address 演算子の使用方法についての追加情報が含まれます。この章では、次のトピックについて説明します。

入力ロール

入力ロールは、どのような種類の名前およびアドレス情報がデータ行にあるかを示します。Name and Address 演算子の各入力属性には、ソース属性に含まれるデータに最も近い入力ロールを指定する必要があります。入力ロールは、自由区分形式のデータ用の分割されていない行タイプのロール（行 1 など）であっても、特定の入力属性ごとに分割されたロール（人名、名、第 1 アドレス、市区町村など）であってもかまいません。分割されたロールでは、Name and Address 演算子に対して、ソース属性の内容に関する情報がより多く与えられます。

表 20-1 に、Name and Address 演算子で使用する入力ロールとその説明を示します。

表 20-1 Name and Address 演算子の入力ロール

入力ロール	説明
名	名。ニックネームや短縮名も含まれます。
ミドル・ネーム	ミドル・ネームまたはイニシャル。ミドル・ネームが 1 つのみの場合に使用します。ミドル・ネームが複数ある場合は最初のものを使用します（George Herbert Walker Bush であれば Herbert）。
ミドル・ネーム 2	2 番目のミドル・ネーム（George Herbert Walker Bush であれば Walker）。
ミドル・ネーム 3	3 番目のミドル・ネーム（Ethel May Roberta Louise Mertz であれば Louise）。
姓	姓または名字。
最初の部分名	個人名の最初の部分で、次のものが含まれます。 ■ プリネーム ■ 名 ■ ミドル・ネーム（1 つまたは複数） これらの構成要素が 1 つのソース列に含まれる場合に使用します。
最後の部分名	個人名の最後の部分で、次のものを含みます。 ■ 姓 ■ ポストネーム これらの構成要素がすべて 1 つのソース列に含まれる場合に使用します。
プリネーム	名前の前に付けられた名前を修飾する情報。Ms.、Mr.、Dr. など。
ポストネーム	名前を修飾する世代などの情報。Jr. や Ph.D など。

表 20-1 Name and Address 演算子の入力ロール（続き）

入力ロール	説明
人名	個人のフルネームで、次のものが含まれます。 ■ 最初の部分名（プリネーム、名およびミドル・ネーム） ■ 最後の部分名（姓およびポストネーム） これらの構成要素がすべて 1 つのソース列に含まれる場合に使用します。
人名 2	入力に複数の人物が含まれる場合、2 番目の人物に使用します。
人名 3	入力に複数の人物が含まれる場合、3 番目の人物に使用します。
会社名	会社または組織の名前。
第 1 アドレス	私書箱、巡回路または番地で、次のものが含まれます。 ■ 町村名 ■ 番地 ■ 地図区分（SW、N など） ■ 道路の種類（大通り、街路、主要道路など） 区画名と区画番号は含まれません。
第 2 アドレス	番地の後半部分で、次のものが含まれます。 ■ 区画名 ■ 区画番号 たとえば、第 2 アドレスが Suite 2100 の場合、区画名は STE（Suite の標準化された表記) で区画番号は 2100 です。
アドレス	アドレス全体で、次のものが含まれます。 ■ 第 1 アドレス ■ 第 2 アドレス これらの構成要素が 1 つの列を共有する場合に使用します。
アドレス 2	汎用的なアドレス行。
地区	地区または郡区（南アメリカやラテン・アメリカのアドレスで一般的)。
市区町村	市区町村の名前。
州（都道府県）	州（都道府県）の名前。
郵便番号	郵便番号。
国名	正式な国名。
国コード	ISO 3166-1993 (E) で規定されている 2 文字または 3 文字の国コード。たとえば、アメリカ合衆国は US または USA、カナダは CA または CAN です。

表 20-1 Name and Address 演算子の入力ロール（続き）

入力ロール	説明
最終行	アドレスの最終行で、次のものが含まれます。 ■ 市区町村 ■ 州（都道府県） ■ 郵便番号 これらの構成要素がすべて 1 つのソース列に含まれる場合に使用します。
行 1、行 2、行 3、行 4、行 5、行 6、行 7、行 8、行 9、行 10	会社名、人名およびアドレスを自由な形式で入力するなど、目的別に分割されていないロールに使用します。これらは汎用的な用途で使用でき、あらゆる種類の名前およびアドレス・データを格納できます。このデータ内容については、パーサーで情報解析されません。この入力ロールではなく、目的別に分割された入力ロールを使用することをお薦めします。

出力コンポーネント

各出力コンポーネントは、タイトル、標準化された名、町村番号、町村名、町村タイプなど、目的別に分割された名前またはアドレスのエンティティを表します。出力コンポーネントは対応する出力属性に割り当てられ、各属性が構成する名前およびアドレスの各部分を識別します。表 20-2 に、Name and Address 演算子の出力コンポーネントとその説明を示します。

表 20-2 Name and Address 演算子の出力コンポーネント

親ノード	出力コンポーネント	説明
名前	プリネーム[1]	名前の前に表示される肩書や敬称。Ms. や Dr. など。
名前	標準化された名前[1]	名の一般的な呼び名。たとえば、Theodore は Ted、James は Jim。
名前	標準化されたミドル・ネーム[1]	ミドル・ネームの一般的な呼び名。たとえば、Theodore は Ted、James は Jim。ミドル・ネームが 1 つのみの場合に使用されます。ミドル・ネームが複数ある場合は最初のものが使用されます（George Herbert Walker Bush であれば Herbert）。
名前	標準化されたミドル・ネーム 2[1]	2 番目のミドル・ネームの一般的な呼び名。たとえば、Theodore は Ted、James は Jim。
名前	標準化されたミドル・ネーム 3[1]	3 番目のミドル・ネームの一般的な呼び名。たとえば、Theodore は Ted、James は Jim。

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
名前	ポストネーム[1]	名前の最後に付け加えられた世代を表す情報。Sr.、Jr.、III など。
名前	その他のポストネーム[1]	名前の最後に付け加えられた資格、学位、所属を表す情報。Ph.D、M.D、R.N など。
名前	名前指示部[1]	人名の指定に使用します。ATTN（to the attention of）、C/O（care of）など。
名前	関連[1]	他の人物に関連付けられた情報。Trustee for など。
名前	人名[1]	名、ミドル・ネーム、姓。
名前 : 人名	名[1]	入力名の「名」部分。
名前 : 人名	ミドル・ネーム[1]	ミドル・ネームまたはイニシャル。ミドル・ネームが 1 つの場合に使用します。ミドル・ネームが複数ある場合は最初のものを使用します（George Herbert Walker Bush であれば Herbert）。
名前 : 人名	ミドル・ネーム 2[1]	2 番目のミドル・ネーム（George Herbert Walker Bush であれば Walker）。
名前 : 人名	ミドル・ネーム 3[1]	3 番目のミドル・ネーム（Ethel May Roberta Louise Mertz であれば Louise）。
名前 : 人名	姓[1]	姓または名字。
名前 : 導入済	性別[1]	次の性別が使用されます。 ■ M= 男性 ■ F= 女性 ■ N= 中性（男性または女性のいずれか） ■ 空白 = 性別不明
名前 : 導入済	人数	レコードで参照される人物の数。たとえば、人名に John と Jane Doe があるレコードの「人数」は 2 になります。
名前 : ビジネス	会社名[1]	会社または組織の名前。部門名も含まれます。
名前 : ビジネス	会社数[1]	レコードで参照される会社の数。
アドレス	アドレス[2]	アドレス全体で、次のものが含まれます。 ■ 第 1 アドレス ■ 第 2 アドレス

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
アドレス	第 1 アドレス[2]	私書箱、巡回路または番地で、次のものが含まれます。 ■ 町村名 ■ 番地 ■ 地図区分（SW、N など） ■ 道路の種類（大通り、街路、主要道路など） 区画名と区画番号は含まれません。
アドレス : 第 1 アドレス	町村番号[2]	番地や建物番号などのアドレスを識別する番号。アドレスで最初に表記されるものを指すこともあります。たとえば、200 Oracle Parkway の場合、「町村番号」は 200 です。
アドレス : 第 1 アドレス	前方向[2]	町村名の前に付けられた方角を表す語。たとえば、100 N University Drive の場合、「前方向」は N です。
アドレス : 第 1 アドレス	町村名[2]	町村の名前。
アドレス : 第 1 アドレス	町村タイプ[2]	道路の種類の識別子。ST、AVE、RD、DR、HWY など。
アドレス : 第 1 アドレス	後方向[2]	町村名の後に付けられた方角を表す語。たとえば、100 15th Ave. S. の場合、「後方向」は S です。
アドレス	第 2 アドレス[2]	番地の後半部分で、次のものが含まれます。 ■ 区画名 ■ 区画番号 たとえば、第 2 アドレスが Suite 2100 の場合、区画名は STE（Suite の標準化された表記) で区画番号は 2100 です。
アドレス : 第 2 アドレス	区画名[2]	第 2 アドレスの種類。APT や STE など。たとえば、第 2 アドレスが Suite 2100 の場合、区画名は STE（Suite の標準化された表記）です。
アドレス : 第 2 アドレス	区画番号[2]	第 2 アドレスを識別する番号。アパート番号や部屋番号など。たとえば、第 2 アドレスが Suite 2100 の場合、区画番号は 2100 です。

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
アドレス	最終行	アドレスの最終行で、次のものが含まれます。 ■ 市区町村 ■ 州または都道府県（州名のかわりに国名が指定されている場合、その国名は含まれません） ■ 完全なアドレスが指定されている場合は書式設定された郵便番号
アドレス : 最終行	地区	地区または郡区（南アメリカやラテン・アメリカのアドレスで一般的）。
アドレス : 最終行	市区町村	市区町村の名前。米国の都市名は、米国郵政公社で使用されている名前に変換されます。
アドレス : 最終行	州（都道府県）	州（都道府県）の名前。
アドレス : 最終行	郵便番号	郵便番号全体。スペースおよび英数字以外の文字は削除されます。
アドレス : 最終行	書式化された郵便番号	書式設定された郵便番号。スペースおよび英数字以外の文字（ダッシュなど）も含まれます。
アドレス : 最終行	配布ポイント	アメリカ合衆国およびオーストラリアに適用されます。 ■ アメリカ合衆国の場合、これは 2 桁の郵便配布ポイントで、9 桁の完全な郵便番号とチェック・ディジットに結合され、配布ポイント・バーコードを形成します。 ■ オーストラリアの場合、これは 9 桁の配布ポイントです。
アドレス : 最終行	国コード	国際標準化機構により規定されている ISO 3166-1993 (E) の 2 文字の国コード。たとえば、アメリカ合衆国は US、カナダは CA です。
アドレス : 最終行	国コード 3	国際標準化機構により規定されている ISO 3166-1993 (E) の 3 文字の国コード。たとえば、アメリカ合衆国は USA、フランスは FRA、ウクライナは UKR です。
アドレス : 最終行	国名	正式な国名。
アドレス : その他のアドレス行	ボックス名[2]	郵便局の私書箱アドレスに使用される名前。PO Box 95 であれば、私書箱名は PO BOX です。
アドレス : その他のアドレス行	ボックス番号[2]	郵便局の私書箱アドレスに使用される番号。PO Box 95 であれば、私書箱名は 95 です。

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
アドレス : その他のアドレス行	ルート名[2]	地方無料郵便配達巡回路アドレスに使用される巡回路名。アドレスが Route 5 Box 10 の場合、巡回路名は RTE（Route の標準化された表記）です。
アドレス : その他のアドレス行	ルート番号[2]	地方無料郵便配達巡回路アドレスに使用される巡回路番号。アドレスが Route 5 Box 10 の場合、巡回路番号は 5 です。
アドレス : その他のアドレス行	ビル名	建物の名前。Cannon Bridge House など。建物名はイギリスでよく使用されます。
アドレス : その他のアドレス行	複合	ビル、キャンパスなどの複合建築物。たとえば、USS John F. Kennedy、Shadow Green Apartments、Cedarvale Gardens、Concordia College など。アドレスに複数の複合建築物がある場合は、「出力コンポーネント」ダイアログの「インスタンス」フィールドを使用して、どの複合建築物を返すか指定できます。
アドレス : その他のアドレス行	他のアドレス	その他の様々なアドレス情報。電話番号や電子メール・アドレスなど。 この種の情報を表す複数のフィールドがあるレコードでは、使用するインスタンスを「Name and Address operator Output Components」画面で指定することで、必要な情報をいくつでも抽出できます。たとえば、レコードに電子メール・アドレスと電話番号の両方がある場合は、「他のアドレス 1」および「他のアドレス 2」を使用することで、両方の情報を抽出できます。
エラー・ステータス : Name and Address	適切なグループ	使用される値は T または F のいずれかです。名前グループ、アドレス・グループ、または名前およびアドレス・グループの処理に成功したかどうかを示します。 ■ 名前グループの場合、T は名前の解析に成功したことを意味します。 ■ アドレス・グループの場合、T は、郵便照合データベースでそのアドレスが見つかったか（郵便照合データベースがインストールされている場合）、アドレスが正常に解析された（郵便データベースがインストールされていない場合）ことを意味します。 ■ 名前およびアドレス・グループの場合、T は名前とアドレスの両方が正常に処理されたことを意味します。 ■ F= グループの解析に失敗しました。 このフラグを「解析済」フラグなどの他のフラグとともに使用し、さらにスプリッタ演算子を使用すると、解析に失敗したレコードを別グループに分離し、そこで個別にアドレス指定できます。

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
エラー・ステータス : Name and Address	解析済	名前またはアドレスが解析されたかどうかを示します。 ■ T= 名前またはアドレスの解析に成功しました。解析警告が発生した場合でも、名前またはアドレスは正常に解析されたと見なされます。 ■ F= 名前またはアドレスを解析できません。 解析警告フラグ（「名前警告」や「市町村警告」など）のステータスをチェックしてください。
エラー・ステータス : Name and Address	解析ステータス	郵便照合ソフトウェアの解析ステータス・コード。
エラー・ステータス : Name and Address	解析ステータスの説明	郵便照合ソフトウェアの解析ステータスのテキストによる説明。
エラー・ステータス : 名前のみ	適切な名前	名前が正常に解析されたかどうかを示します。 ■ T= 名前の解析に成功しました。解析警告が発生した場合でも、名前は正常に解析されたと見なされます。 ■ F= 名前を解析できません。
エラー・ステータス : 名前のみ	名前警告	異常なデータまたはエラーの可能性があるデータが名前で検出されたかどうかを示します。 ■ T= 名前の解析に問題があるか、異常なデータが検出されました。解析ステータスコンポーネントをチェックして、警告の原因を調べてください。 ■ F= 名前の解析に問題はありません。
Error Checking: アドレスのみ	適切なアドレス	アドレスが正常に処理されたかどうかを示します。 ■ T= 正常に処理されました。アドレスが郵便照合データベース内で検索されました。該当する国の郵便照合データベースがインストールされていない場合は、アドレスが正常に解析されたことを示します。 ■ F= 正常に処理されませんでした。アドレスで示されている国の郵便照合データベースがインストールされている場合は、そのアドレスがデータベースに登録されていないことを示します。該当する国の郵便照合データベースがインストールされていない場合は、そのアドレスを解析できなかったことを示します。 郵便照合が可能な国と不可能な国のレコードが両方ある場合は、このフラグが目安になります。

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
Error Checking: アドレスのみ	検出済	アドレスに示されている国の郵便照合データベースに、そのアドレスが登録されているかどうかを示します。 ■ T= 郵便照合データベースにアドレスは登録されています。 ■ F= 郵便照合データベースにアドレスは登録されていません。これは、そのアドレスが正しくないことを意味しています。また、該当する国に対して郵便照合を実行できないこともあります。 次の項のフラグがすべて true の場合にかぎり、このフラグは true になります。郵便照合を使用できる場合は、このフラグでレコードの質を判断できます。
Error Checking: アドレスのみ : 検出済	市町村検出済	郵便照合で市区町村が検出されたかどうかを示します。 ■ T= 市区町村は検出されました。 ■ F= 市区町村は検出されませんでした。
Error Checking: アドレスのみ : 検出済	町村番号が検出済	郵便照合で町村名が検出されたかどうかを示します。 ■ T= 町村名は検出されました。 ■ F= 町村名は検出されませんでした。
Error Checking: アドレスのみ : 検出済	町村番号が検出済	指定されている町村の有効な番地範囲内で、その町村番号が検出されたかどうかを示します。 ■ T= 町村番号は検出されました。 ■ F= 指定されている町村の有効な番地範囲内で、その町村番号は検出されませんでした。
Error Checking: アドレスのみ : 検出済	町村コンポーネントが検出済	「前方向」や「後方向」などの町村名のコンポーネントが検出されたかどうかを示します。 ■ T= 町村名のコンポーネントが検出されました。 ■ F= 町村名のコンポーネントは検出されませんでした。

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
Error Checking: アドレスのみ : 検出済	曖昧でない一致が検出済	郵便データベースで完全に一致するアドレスが検出されたかどうかを示します。 ■ T= 入力レコードと完全に一致するエントリが、郵便データベースで唯一のエントリとして検出されました。 ■ F= アドレスが曖昧です。一致するアドレス・エントリが郵便データベースで複数検出され、どれが正しいか決定できません。たとえば、入力アドレスが 100 4th Avenue のときに、郵便データベースに 100 4th Ave N と 100 4th Ave S という 2 つのエントリがあるとします。この場合、入力アドレスに方角情報がないため、解析は失敗します。
Error Checking: アドレスのみ	市町村警告	異常なデータまたはエラーの可能性があるデータが市区町村で検出されたかどうかを示します。 ■ T= 市区町村の解析に問題があります。 ■ F= 市区町村の解析に問題はありません。
Error Checking: アドレスのみ	町村警告	異常なデータまたはエラーの可能性があるデータが町村アドレスで検出されたかどうかを示します。 ■ T= 町村アドレスの解析に問題があります。 ■ F= 町村アドレスの解析に問題はありません。
Error Checking: アドレスのみ	検証可能なアドレス	その国のアドレスに対して郵便照合を実行できるかどうかを示します。一致処理の結果は示されません。 ■ T= その国のアドレスに対して郵便照合を実行できます。 ■ F= その国のアドレスに対して郵便照合を実行できません。アドレスで示されている国の郵便照合データベースはインストールされていますが、このアドレスに対して郵便照合を実行できないことを示します。アドレスに該当する国の郵便照合データベースがインストールされていません。

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
Error Checking: アドレスのみ	アドレスが修正済	一致処理中に何らかの方法でアドレスが修正されたかどうかを示します。標準化は修正処理とは見なされません。 ■ T= アドレスのコンポーネントの一部が、標準化以外の方法で変更されました。この場合は、アドレスのどのコンポーネントが変更されたかを示す修正フラグの 1 つが true になっています。 ■ F= アドレスのコンポーネントは標準化を除き、いっさい変更されていません。
Error Checking: アドレスのみ : アドレスが修正済	郵便番号が修正済	一致処理中に郵便番号が修正されたかどうかを示します。この修正には、郵便番号の拡張追加処理も含まれます。 ■ T= 郵便番号は修正されました。 ■ F= 郵便番号は修正されていません。
Error Checking: アドレスのみ : アドレスが修正済	市区町村が修正済	一致処理中に市区町村の名前が修正されたかどうかを示します。郵便サービスで使用される市区町村名は、郵便番号入力を使用して決められます。 ■ T= 一致処理中に市区町村名が修正されました。 ■ F= 市区町村名は修正されていません。
Error Checking: アドレスのみ : アドレスが修正済	町村が修正済	一致処理中に町村名が修正されたかどうかを示します。正しい町村名が郵便サービスで使用される代替名に変更される場合もあります。 ■ T= 一致処理中に町村名が修正されました。 ■ F= 町村名は修正されていません。
Error Checking: アドレスのみ : アドレスが修正済	町村コンポーネントが修正済	一致処理中に「前方向」や「後方向」などの町村名のコンポーネントが修正されたかどうかを示します。 ■ T= 町村名のコンポーネントが修正されました。 ■ F= 町村名のコンポーネントは修正されていません。

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
Error Checking: アドレスのみ	アドレス・タイプ	アドレスのタイプ。次に一般的な例を示します。実際の値は、郵便一致ソフトウェアのベンダーによって異なります。 ■ F= 会社 ■ G= 一般的な配達先 ■ H= 高層アパートまたはオフィス・ビル ■ HD= 高層のデフォルト。この場合は、ビル全体に 1 つの Zip+4 郵便番号が適用されます。アドレスに階や部屋番号がある場合、Name and Address 演算子では、より詳細な郵便番号が検出されます。この場合、レコードは H タイプとして扱われ、階や部屋ごとに異なる Zip+4 番号が使用されます。 ■ B= 郵便受け ■ R= 地方コード ■ S= 町村 ■ M= 軍事関係 ■ P= 私書箱
Error Checking: アドレスのみ	解析する国	レコードの最終的な解析に使用された国パーサー。
国固有 : アメリカ	ZIP 5	5 桁からなるアメリカ合衆国の郵便番号。アメリカ合衆国のアドレスにのみ適用されます。
国固有 : アメリカ	ZIP 4	5 桁からなるアメリカ合衆国の郵便番号に追加される 4 桁の接尾辞で、より詳細なロケーションの指定に使用されます。アメリカ合衆国のアドレスにのみ適用されます。
国固有 : アメリカ	都市名	プエルトリコで使用される都市の区画名。アメリカ合衆国（プエルトリコ）のアドレスにのみ適用されます。

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
国固有 : アメリカ	LACS フラグ	アドレスに LACS 変換が必要かどうかを示します。 Locatable Address Conversion System（LACS）は、911 緊急システムが実装されているときに、新しいアドレスを提供するシステムです。911 アドレス変換は、多くの場合、地方スタイルのアドレスから都市スタイルのアドレスへの変更ですが、既存の都市スタイルのアドレスで名前や番号が変更される場合もあります。 ■ T= アドレスに 9-1-1 変換（元の地方無料郵便配達巡回路アドレスから新しい町村アドレスに）が必要で、LACS ベンダーに発行する必要があります。 ■ F= アドレスに変換は必要ありません。 アメリカ合衆国のアドレスにのみ適用されます。
国固有 : アメリカ	CART	4 文字からなる米国郵政公社の配達ルート。アメリカ合衆国のアドレスにのみ適用されます。
国固有 : アメリカ	DPBC チェック数値	デリバリ・ポイント・バーコードを形成するチェックディジット。アメリカ合衆国のアドレスにのみ適用されます。
国固有 : アメリカ : 地理	メトロポリタン統計地域	メトロポリタン統計地域（Metropolitan Statistical Area: MSA）番号。たとえば、0000 はそのアドレスが MSA 内にないことを意味します。この場合、通常は地方を意味します。アメリカ合衆国のアドレスにのみ適用されます。
国固有 : アメリカ : 地理	マイナー調査地域	マイナー調査地域アメリカ合衆国のアドレスにのみ適用されます。
国固有 : アメリカ : 地理	緯度	赤道から北方向への隔たりを表す角度。赤道の北側は正の値、南側は負の値で表されます（北アメリカは常に正の値）。
国固有 : アメリカ : 地理	経度	グリニッジ子午線から東方向への隔たりを表す角度。グリニッジ子午線の東側は正の値、西側は負の値で表されます（北アメリカは常に負の値）。
国固有 : アメリカ : 地理	FIPS 区	米国連邦情報処理標準（Federal Information Processing Standard: FIPS）により規定されている 3 桁の国コード。アメリカ合衆国のアドレスにのみ適用されます。
国固有 : アメリカ : 地理	FIPS 区	米国連邦情報処理標準（FIPS）により各郡に割り当てられている完全な（州に郡を加えた）コード。FIPS 郡コードは州ごとに一意のため、完全な FIPS コードでは、2 桁の州コードに続けて 3 桁の郡コードが使用されます。アメリカ合衆国のアドレスにのみ適用されます。

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
国固有 : アメリカ : 地理	調査 ID	アメリカ合衆国の調査で使用される地区およびブロック・グループ番号。最初の 6 桁が地区番号で、最終桁がその地区内のブロック・グループ番号。これらのコードは、人口統計データベース内のコードとの照合に使用されます。アメリカ合衆国のアドレスにのみ適用されます。
国固有 : イギリス	地方コード	イギリスのアドレスにのみ適用されます。たとえば、次のアドレスには「地方コード」に 23591 が割り当てられます。 Chobham Rd Knaphill Woking GU21 2TZ
国固有 : イギリス	地方名	イギリスのアドレスにのみ適用されます。たとえば、次のアドレスには「地方名」に KNAPHILL が割り当てられます。 Chobham Rd Knaphill Woking GU21 2TZ
国固有 : イギリス	区名	イギリスのアドレスにのみ適用されます。
国固有 : カナダ	インストレーション・タイプ	カナダにおける郵便施設のタイプ。 ■ STN= 支局 ■ RPO= 郵便出張所 たとえば、アドレスが PO Box 7010, Scarborough ON M1S 3C6 の場合、「インストレーション・タイプ」は STN です。
国固有 : カナダ	Installation Name	カナダにおける郵便施設の名前。たとえば、アドレスが PO Box 7010, Scarborough ON M1S 3C6 の場合、「Installation Name」は AGINCOURT です。
国固有 : 香港	デリバリー・オフィス・コード	香港のアドレスにのみ適用されます。たとえば、次のアドレスには「デリバリー・オフィス・コード」に WCH が割り当てられます。 Oracle 39/F The Lee Gardens 33 Hysan Ave Causeway Bay

表 20-2 Name and Address 演算子の出力コンポーネント（続き）

親ノード	出力コンポーネント	説明
国固有 : 香港	デリバリー・ビート・コード	香港のアドレスにのみ適用されます。たとえば、次のアドレスには「デリバリー・ビート・コード」に S06 が割り当てられます。 Oracle 39/F The Lee Gardens 33 Hysan Ave Causeway Bay
国固有 : 香港	アドレス 2	第 2 のアドレス行で、アドレスに番地と建物 / 階の両方があるときに使用されます。香港にのみ適用されます。

[1] このコンポーネントが複数回発生するレコードでは、使用するインスタンスを「Name and Address operator Output Attributes」コンポーネント・ダイアログで指定できます。たとえば、「会社名」に 2 つのインスタンスがあるレコードでは、「会社 1」と「会社 2」の 2 つの出力属性を追加し、最初のインスタンスを一方に 2 番目のインスタンスを他方に割り当てることで、両方を抽出できます。

[2] 2 通りのアドレスがあるレコードでは、どちらを標準アドレス（「アドレス」コンポーネントに割り当てられるアドレス）とし、どちらを補助アドレスとするかを指定できます。これは、「Name and Address operator Output Attributes」コンポーネント・ダイアログで指定できます。

Name and Address 演算子でサポートされている国

表 20-3 に、Name and Address クレンジング・ソフトウェアの一部のベンダーにおいてサポートされている国を示します。このリストはベンダーによって異なります。

表 20-3 Name and Address でサポートされている国

名前	Name and Address 演算子で使用可能な郵便照合ソフトウェア
アルゼンチン	×
オーストラリア	可能
ベルギー	×
ブラジル	可能
カナダ	可能
チリ	×
コロンビア	×
デンマーク	×
フランス	可能
ドイツ	可能

表 20-3 Name and Address でサポートされている国（続き）

名前	Name and Address 演算子で使用可能な郵便照合ソフトウェア
香港	可能
インド	×
アイルランド	×
イタリア	×
メキシコ	可能
マレーシア	×
オランダ	可能
ニュージーランド	×
ペルー	×
フィリピン	×
ポルトガル	可能
シンガポール	可能
南アフリカ	×
スペイン	可能
スウェーデン	×
スイス	×
アラブ首長国連邦	×
イギリス	グレート・ブリテンのみ
アメリカ合衆国	可能
ベネズエラ	×

国による郵便認定

Oracle Warehouse Builder Name and Address で使用される郵便照合ソフトウェアは、様々な国で認定されています。この認定は、使用する郵便照合ソフトウェアのベンダーにより異なります。認定の中には次のものが含まれます。

- アメリカ合衆国 - 米国郵政公社に対する CASS 認定
- カナダ - カナダ郵便局に対する SERP 認定
- オーストラリア - オーストラリア郵便局に対する AMAS 認定

米国郵政公社の CASS 認定

Coding Accuracy Support System（CASS) は、米国郵政公社（USPS）と郵便業界の協力により開発されたプロセスです。このプロセスは、郵便利用業者にとって、アドレス照合ソフトウェアの品質を測定する共通プラットフォームとなり、あらゆる郵便に適用される 5 桁の郵便番号、ZIP+4 番号、デリバリ・ポイント・コードおよび配達ルート・コードの精度を検証します。オートメーション料金の郵便物の生成に使用されるアドレス一覧はすべて、CASS 認定ソフトウェアにより照合される必要があります。Oracle Pure Name and Address は CASS 認定です。CASS レポートは USPS 指定のテキスト・ファイルで、Oracle9*i* Pure Name and Address により生成されます。USPS 要件を満たすには、送信者は CASS レポートをオリジナルのフォームで USPS に送信する必要があります。

カナダ郵便局の SERP 認定

カナダ郵便局は、ソフトウェア・パッケージの評価用に Software Evaluation and Recognition Program（SERP）というテスト・プログラムを開発しています。このプログラムでは、カナダ郵便局の要件に対して、メーリング・リストの妥当性を検証する機能または検証および修正する機能、あるいはその両方が評価されます。SERP 要件を満たす郵便照合プログラムは、カナダ郵便局の Web サイトに一覧表示されています。

インセンティブ・レターメール、広告付きアドメール、刊行物メールなどを利用する顧客は、アドレス照合プログラムの要件を満たしている必要があります。顧客は、使用しているデータベースとカナダ郵便局のアドレス・データを比較することで、Statement of Accuracy を取得することができます。

オーストラリア郵便局の AMAS 認定

Address Matching Approval System（AMAS®）は、アドレッシングの質を向上させるためにオーストラリア郵便局により開発されたソフトウェア承認プログラムです。このプログラムは、アドレス照合ソフトウェアの次の機能をテストおよび測定する基準を提供します。

- 郵便アドレス・ファイル（Postal Address File: PAF）に対するアドレスの照合および修正。
- 各アドレス・レコードに対する一意の配布ポイント ID（DPID）® の追加。これは、メールのバーコード化への手順の 1 つです。

AMAS により、ベンダーは次の機能を持つアドレス照合ソフトウェアを開発できます。

- バーコード作成用アドレスの準備
- 高品質なアドレッシング
- 郵便物の事前区分け申請による割引条件の獲得

事前区分けサービス料金は、利用者が、最新バージョンの PAF で有効なデリバリ・ポイント ID（DPID）を持つ AMAS 承認ソフトウェアを使用していることが条件です。

郵便局で入手可能な事前区分け申請書では、郵便物が適切に準備されていることを宣言する必要があります。

各マッピングに生成できる郵便レポートは 1 つのみです。作成される郵便レポートは、Name and Address 演算子定義に指定されている主国に対するものです。マッピングされるすべてのアドレスが、Name and Address ウィザードの「定義」ページで選択した主国にあるのが理想的です。詳細は、8-77 ページの「定義」を参照してください。

Name and Address Server の構成

Name and Address 演算子は、ランタイム・スキーマにインストールされている UTL_NAME_ADDR パッケージをコールする PL/SQL コードを生成します。ターゲット・スキーマには、UTL_NAME_ADDR パッケージを参照するプライベート・シノニムの NAME_ADDR が定義されます。UTL_NAME_ADDR パッケージは Java パッケージをコールします。Java パッケージは処理要求を外部の Name and Address Server に送信し、Trillium などのサードパーティ製の Name and Address 処理ライブラリとやり取りします。

サーバー・オプションは、サーバー・プロパティ・ファイルの NameAddr.properties を使用して構成できます。このファイルは、Oracle Warehouse Builder のサーバー側インストールの ORACLE_HOME にある owb/bin/admin に配置されています。次のコードは、いくつかの重要なプロパティとそのデフォルト設定です。

```
TraceLevel=0
SocketTimeout=120
ClientThreads=16
Port=4040
```

TraceLevel プロパティは多くの場合、サーバー通信を診断して郵便照合プログラム・パーサーからの出力を表示する際に変更されます。他のプロパティはほとんど変更されません。

- **TraceLevel:** owb/bin/admin フォルダにあるファイル NASvrTrace.log へのログ出力を有効にします。このファイルにはすべての送受信データが記録され、作成したマッピングが Name and Address Server と通信し、Name and Address Server がサービス・プロバイダからの出力を受信していることを確認できます。トレース・ログにはすべてのサーバー入力および出力が記録され、マッピングの実行により解析要求が作成されているかどうかを調べるのに役立ちます。ログを有効にするには TraceLevel=1 を設定します。ただし、トレースを実行するとパフォーマンスが低下し、大きなログ・ファイルが作成されます。ログの生成を無効にするには TraceLevel=0 を設定します。
- **ClientThreads:** このパラメータは将来の開発用に予約されています。デフォルト値の 16 のままにしてください。
- **Port:** サーバーがリスニングに使用するポートを指定します。インストーラにより初期設定で割り当てられます。デフォルトのポートが他のプロセスと競合する場合は、この値を変更できます。ポートを変更する場合は、*runtime_schema*.nas_connection 表にあるポート属性も変更し、接続を構築する utl_name_addr パッケージを有効にする必要があります。

Name and Address Server の起動と停止

プロパティ・ファイルの編集または表のメンテナンスを行ったときは、その変更を有効にするために、Name and Address Server をいったん停止し再起動する必要があります。自動シャットダウン・プロパティが FALSE で、プロパティ・ファイルの編集または表のメンテナンス時にサーバーが実行中の場合は、手動でサーバーを停止および再起動する必要があります。

Name and Address Server を手動で停止するには、次を実行します。

- Windows の場合は、*ORACLE_HOME*/owb/bin/win32/NAStop.bat を実行します。
- UNIX の場合は、*ORACLE_HOME*/owb/bin/unix/NAStop.sh を実行します。

Warehouse Builder でマッピングを起動することで、Name and Address Server を自動的に再起動できます。また、手動での再起動も可能です。

Name and Address Server を手動で再起動するには、次を実行します。

- Windows の場合は、*ORACLE_HOME*/owb/bin/win32/NAStart.bat を実行します。
- UNIX の場合は、*ORACLE_HOME*/owb/bin/unix/NAStart.sh を実行します。

マッピングでの Name and Address 演算子の使用方法のヒント

マッピングで Name and Address 演算子を定義および使用するときには、次の使用方法のヒントを参照することをお薦めします。

Name and Address データでのエラー修正

Oracle Warehouse Builder Name and Address は、Name and Address のクレンジングを専門とするサードパーティ・ソフトウェア・ベンダーが提供する Name and Address ソフトウェアとデータの使用を前提に構築されています。この項では、こうした製品に関する追加の注意事項と一般的な解析上の問題点について説明します。

Name and Address の解析と修正はデータの質を大きく向上させますが、注意深く適用しないと、データの質を低下させるおそれがあります。使用済データが持つ問題領域には限りがないため、Name and Address の解析、郵便照合または間違ったデータのクレンジングではエラーがつきものです。

他の種類の解析と同様、Name and Address 解析も、キーワードとキーワードに含まれるパターンの識別に依存します。キーワードの数が少ないコンピュータ言語とは異なり、自由形式の Name and Address データでは、キーワード数が莫大でしかもキーワード・セットが 100% 完全であることがないため、どの地域のものであっても解析が非常に困難です。たとえば、ある Name and Address クレンジング・ソフトウェアに付属するアメリカ合衆国用のパターン表には、80,000 を超える定義済キーワードと 5,000 を超えるパターンが含まれています。キーワード・セットは何百万ものレコードを分析することで構築されますが、その新しいデータの・セットに、未定義のキーワードが含まれることもあります。

自由形式の Name and Address レコード行の大半が持つ共通パターンは、数字、文字および英数文字列で構成されるため、多くの場合、解析は英数字のパターンにのみ基づいて実行されます。より困難なケースでは、英数字のパターンが曖昧で、アドレス行または名前行のどちらでも、複数の解釈が成り立ちます。また、特定のパターンが見つからない場合もあります。こうした場合、コンピュータ言語では、コンパイル・エラーがレポートされ実行可能なイメージが生成されませんが、Name and Address の解析エラーでは、解析ステータス・コードが設定され、Name and Address 要素は解析不能になる場合があります。

ステータス・コードの使用

解析エンジンがどこかの時点で失敗する可能性が高いため、Name and Address 処理を実行する際は必ずエラー処理も考慮してください。データ・マッピングの制御には、ステータス・コードを使用します。品質の基準はアプリケーションによって異なるため、特定レコードに適用する品質を制御するフラグが数多く用意されています。郵便照合がサポートされる国の場合は、アドレスが郵便データベース内の有効なエントリであることを確認する「適切なグループ」フラグが最適な品質尺度です。CASS または SERP 認証の郵便では、このフラグはレコードの受入れまたは拒否を決める最適な基準になります。

郵便レポートを実行しない場合は、アドレスが郵便データベースになくても使用できます。たとえば、交差点のアドレスやビル名を使用したアドレスは郵便データベースで見つからないことがありますが、配達には問題ありません。解析ステータスが有効な場合は、アドレス要素の標準化によりメリットが得られます。「適切なグループ」フラグが郵便照合の失敗を示す場合は、他の解析警告フラグまたはエラー・フラグを使用して、解析ステータスを判断することができます。「解析済」フラグは、解析プロセスの成功または失敗を示します。「解析済」が解析の成功を示す場合でも、異常なデータを示す解析警告フラグをチェックする必要があります。解析警告がある場合は、それらのレコードを手動でチェックすることをお薦めします。「解析済」が解析の失敗を示す場合は、元のデータを保存して、データの損失を防ぐ必要があります。

分割された入力ロールの使用

入力データの粒度に合せるために、幅広い種類の入力ロールが用意されています。行タイプの入力には行ベースの入力ロールを使用でき、「名」などの細分化された属性には分割された入力ロールを使用できます。分割された入力ロールはデータの内容に関してパーサーにより多くの情報を提供し、その結果、解析がより正確になるため、可能なかぎり、分割された入力ロールを使用することをお薦めします。一部の分割された入力ロールは、より大きく分割されたロールと重複します。たとえば、市区町村、州（都道府県）および郵便番号の組合せにより、「最終行」ロールを再定義できます。こうした割当ての再定義は行わないでください。ロールが重複している場合は、より細かく分割されたロールの方が具体的なため優先されます。

分割された Name and Address 属性の中で元の値を保持する必要があるものは、ターゲットに直接マッピングします（それらを Name and Address 演算子内にマッピングします）。この処理は、名、ミドル・ネームおよび姓でよく行われます。名を標準化された名前に変更（たとえば、Peggy を Margaret に）すると、不適切な場合があるためです。このダイレクト・マッピングにより、個人名が会社名として解析された際の名前データの損失を防げます。照合および（分化されたデータの）マージ用に標準化された名またはミドル・ネームを取得するには、同じ属性を Name and Address 演算子にマッピングします。ただし、肩書、性別、標準化された名などの記録された出力のみが演算子からの出力として使用されます。名、ミドル・ネームまたは姓は入力と同じであるため、それらを出力に使用するメリットはありません。この推奨事項は、名、ミドル・ネームおよび姓がすでに知られている場合の分割されたデータにのみ適用されます。

スプリッタ演算子での有効なレコードと無効なレコードの分離

スプリッタ演算子を使用して、有効なレコードと無効なレコードを別々のターゲットにマッピングする場合を想定してください。無効なレコードの割合が非常に低い場合またはそのデータ数が少ない場合は、無効なレコードを手動で修正することを検討します。ターゲット・スキーマへの今後のマッピングに備えて、各レコードに修正済レコードのフラグを手動で設定できます。「IS_GOOD_GRP」フラグが F でレコードが手動修正用の特別なターゲットにマッピングされている場合は、郵便照合の成功レベルを示す追加コンポーネントをマッピングできます。コンポーネントは、市区町村、町村名、番地範囲または町村名のコンポーネント（町村名の前後に付けられた方角を表す語や道路の種類など）の各レベルで、照合の成功を示すために使用できます。

21

Warehouse Builder での SAP R/3 データの使用

Warehouse Builder SAP インテグレータを使用すると、SAP アプリケーション・データ・ソースから Warehouse Builder Design Repository にメタデータ・オブジェクト定義をインポートできます。この章では、マッピングでの SAP オブジェクトの使用方法、マッピング用の PL/SQL および ABAP コードの生成方法およびターゲットへのそれらの配布方法について説明します。また、この章では、SAP データを抽出しターゲットにロードする方法についても説明します。

この章では、次のトピックについて説明します。

Warehouse Builder SAP インテグレータについて

Warehouse Builder SAP インテグレータを使用すると、SAP アプリケーション・ソース・システムに接続して、Warehouse Builder Design Repository 内のプロジェクトに SAP ソース定義をインポートできます。

次に、ABAP または PL/SQL コードを生成して、SAP R/3 3x および R/3 4x システムからターゲット・システムにデータを抽出、変換およびロードできます。

SAP ビジネス領域について

SAP アプリケーション・システムでは、データベースとメタデータ・オブジェクトが様々なビジネス領域に論理的にグループ化されています。SAP では、ビジネス領域は、製品や市場分野の分類に使用する企業内の編成単位になります。たとえば、財務会計（FI）ビジネス領域には、財務会計トランザクションに関連するデータが分類されています。このトランザクションには、総勘定元帳会計、買掛金、売掛金、決算と報告などが含まれます。

SAP 定義を Warehouse Builder にインポートするときは、「Business Component Hierarchy」ダイアログにあるグラフィカルなナビゲーション・ツリーを使用して、SAP ソース・アプリケーションのビジネス領域構造を検索できます。このナビゲーション・ツリーにより、SAP アプリケーション・サーバーから SAP メタデータ・オブジェクトを選択できます。

SAP 表のタイプ

SAP インテグレータを使用すると、SAP ビジネス領域のメタデータまたは関連する任意の ABAP ディクショナリ・オブジェクトをインポートできます。

SAP インテグレータでは、次の SAP 表タイプについて、定義をインポートし配布コードを生成できます。

- **透過 :** 透過表は最初に ABAP ディクショナリに定義され、データベース内に作成されます。また、透過表は R/3 システムとは無関係に使用できます。透過表には、PL/SQL コードおよび ABAP コードのどちらでも生成できます。
- **クラスタ :** クラスタ表は、ABAP ディクショナリ表の一種です。この表には、すべてのグループのデータベース表に関する情報が含まれます。クラスタ表は SAP データベースには作成されません。クラスタ表はデータベース表ではなくデータ・ディクショナリ表であるため、ABAP コードのみを生成できます。

  プール : 複数の表からのデータが、データベース内の表プールにまとめて格納されます。プール表は ABAP ディクショナリ内にあり、データベースには認識されません。プール表には ABAP コードのみを生成できます。

Windows で必要なファイル

Warehouse Builder SAP インテグレータでは、リモート・ファンクション・コールに使用する librfc32.dll というダイナミック・リンク・ライブラリ・ファイルがクライアント・マシンに必要です。このファイルは、SAP アプリケーション・インストール CD にあります。このファイルをクライアント・システム上の以下の Warehouse Builder ディレクトリにコピーします。

```
X:¥%ORACLE_HOME%¥bin¥admin
```

ここで、"X:" は Warehouse Builder Design Client が存在するドライブで、"¥%ORACLE_HOME%" は Warehouse Builder の Oracle ホームのパスです。

SAP ソース・モジュールを作成し SAP 表をインポートしたにもかかわらず、表内の列が表示されない場合は、librfc32.dll ファイルに互換性がないと考えられます。.dll ファイルのバージョンまたはビルド番号を、NT エクスプローラ・ウィンドウで確認してください。

次のバージョンが現在、Warehouse Builder でサポートされます。

ファイル・バージョン : 4640,5,123,2956

ビルド : Wednesday, August 09 23:46:33 2000

ファイル・サイズ : 1,945,138 バイト

製品バージョン : 46D,123

このバージョンの .dll ファイルがインストール CD に用意されています。

UNIX で必要なファイル

Warehouse Builder SAP インテグレータでは、リモート・ファンクション・コールに使用する librfccm.so というダイナミック・リンク・ライブラリ・ファイルがクライアント・マシンに必要です。このファイルは、SAP アプリケーション・インストール CD にあります。このファイルをクライアント・システム上の以下の Warehouse Builder ディレクトリにコピーします。

```
X:¥$ORACLE_HOME¥bin¥admin
```

ここで、"X:" は Warehouse Builder Design Client が存在するドライブで、"¥$ORACLE_HOME" は Warehouse Builder の Oracle ホームのパスです。

また、X:¥$ORACLE_HOME¥bin¥admin を UNIX の環境変数パス LD_LIBRARY_PATH に追加します。

SAP メタデータ・オブジェクトの定義

新規モジュール・ウィザードを使用して、SAP ソース・モジュールを定義できます。Warehouse Builder では、ソース SAP アプリケーションからインポートしたメタデータの格納に、この定義が使用されます。

この項では、次のトピックについて説明します。

- SAP モジュール定義の作成（21-4 ページ）
- SAP メタデータ定義のインポート（21-11 ページ）
- SAP ソース・モジュールの更新（21-18 ページ）

関連情報は、次を参照してください。

- データベース・ソースの接続の構成（4-5 ページ）
- 第 4 章「データ定義のインポート」
- SAP Application 3.0 のドキュメント

SAP モジュール定義の作成

新規モジュール・ウィザードでは、使用するソースに、SAP R/3 バージョン 3.x または SAP R/3 バージョン 4.x のシステム・タイプを選択できます。アプリケーションのバージョンを選択した後は、Warehouse Builder リポジトリと SAP アプリケーション・サーバー間の接続情報を設定する必要があります。モジュール・プロパティ・シートを使用して、この情報を編集できます。

注意： SAP モジュール定義を作成するには、最初に SAP アプリケーション・サーバーへの接続情報をシステム管理者から入手する必要があります。

接続情報の設定時に、次の接続タイプを選択できます。

- リモート・ファンクション・コール（RFC）

  これはデフォルトの接続タイプです。リモート・ファンクション・コールは、コール元とは異なるシステムで実行されているファンクション・モジュールを検索します。リモート・ファンクションは同一システム内からでも（リモート・コールとして）コールできますが、通常はコール元とコール先は異なるシステムにあります。このメソッドを使用する場合は、SAP アプリケーション・サーバーの具体的な IP アドレス情報が必要になります。

- SAP リモート・ファンクション・コール（SAPRFC.INI）

  SAP では、独自の初期化ファイルを使用して、IP アドレス情報を自動検出できます。SAPRFC.INI を使用すると、2 つの SAP システム間（R/3 または R/4）または SAP システムと非 SAP システム間でリモート・コールを実行できます。このメソッドは、SAP 固有の接続情報がわかっており IP 接続情報を自動化する場合に便利です。

注意： SAPRFC.INI 接続タイプを使用するには、ファイル SAPRFC.INI が次のディレクトリにインストールされている必要があります。

`X:¥ORACLE_HOME¥wbapp`

ここで、`X:¥ORACLE_HOME` は Warehouse Builder の Oracle ホームのパスです。このファイルは、SAP アプリケーション・クライアントのインストール CD にあります。詳細は、システム管理者に問い合せてください。

新規モジュール・ウィザードでは、SAP アプリケーション・サーバーにあるメタデータに基づいてモジュールが自動作成されます。

SAP ソース・モジュールを作成する手順は次のとおりです。

1. Warehouse Builder コンソールのナビゲーション・ツリーから、「アプリケーション」ノードを開きます。
2. 「SAP」ノードを右クリックして、「Create SAP R3 Source Module」を選択します。

   Warehouse Builder「新規モジュール・ウィザード : ようこそ」ページが表示されます。
3. 「次へ」をクリックします。

   図 21-1 に示す「新規モジュール・ウィザード : 名前」ページが表示されます。

図 21-1「新規モジュール・ウィザード : 名前」ページ

4. 「新規モジュール・ウィザード : 名前」ページで、次の情報を入力します。

   モジュールの名前 : 一意なモジュール名を、1 ～ 30 文字の英数字で入力します。空白は使用できません。

   モジュールのステータス : モジュールのステータスを、ドロップダウン・リストから選択します。選択できるステータスは「開発」、「品質保証」または「製品」です。

   これらのステータスの 1 つを選択することで、ウェアハウス設計のバージョンを記録できます。

   説明 : 作成するモジュールの説明を入力します（オプション）。

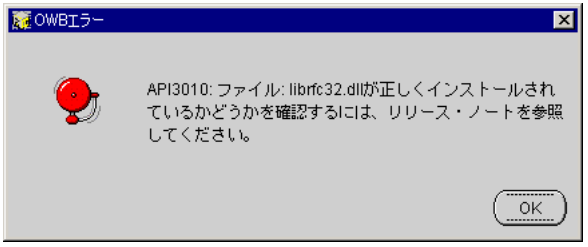
5. 「次へ」をクリックします。

   「新規モジュール・ウィザード : データ・ソース情報」ページが表示されます。

6. ドロップダウン・リストから、SAP アプリケーションのバージョンを正しく選択します（「SAP R/3 3.x」または「SAP R/3 4.x」）。

   `librfc32.dll` ファイルが見つからない場合は、図 21-2 に示すようなエラー・メッセージが表示されます。

図 21-2 librfc32.dll ファイルが見つからないことを示すエラー・メッセージ



手順を進める前に、`librfc32.dll` ファイルをロードする必要があります。詳細は、21-3 ページの「Windows で必要なファイル」と 21-3 ページの「UNIX で必要なファイル」を参照してください。

7. 「次へ」をクリックします。

   図 21-3 に示すような「RFC 接続」が選択された「新規モジュール・ウィザード : 接続情報」ページが表示されます。

図 21-3「新規モジュール・ウィザード : 接続情報」ページ

次の接続タイプのいずれかを選択します。

リモート・ファンクション・コール（RFC）はデフォルトの接続タイプです。詳細は、21-5 ページのリモート・ファンクション・コール（RFC）を参照してください。

SAP リモート・ファンクション・コール（SAPRFC.INI）。詳細は、21-5 ページの SAP リモート・ファンクション・コール（SAPRFC.INI）を参照してください。

8. 該当するフィールドに接続情報を入力します。このページに表示されるフィールドは、選択した接続タイプによって異なります。

   手順を進める前に、SAP アプリケーション・サーバーへの接続情報をシステム管理者から入手する必要があります。

   「RFC 接続」タイプを選択した場合は、図 21-4 に示すように次の接続情報が必要です。

   アプリケーション・サーバー : SAP アプリケーション・サーバーの別名または IP アドレスを入力します。

   システム番号 : SAP ユーザー・インタフェース・ログイン用の SAP システム番号を入力します。この番号は SAP アプリケーション構成に必須で、SAP システム管理者により与えられます。

   クライアント : SAP クライアント番号を入力します。この番号は SAP アプリケーション構成に必須で、SAP システム管理者により与えられます。

   ユーザー名 : SAP ユーザー・インタフェース用のユーザー名を入力します。この名前は SAP アプリケーション構成に必要で、SAP システム管理者により与えられます。

   言語 : 英語の場合は「EN」、ドイツ語の場合は「DE」を選択します。「DE」を選択した場合、説明のテキストがドイツ語で表示され、その他のテキストはすべて英語で表示されます。

図 21-4「新規モジュール・ウィザード : 接続情報」ページ（デフォルトの RFC の場合）

「SAPRFC.INI ファイル」接続タイプを選択した場合は、次の接続情報が必要です。

RFC 接続先 : SAP 接続情報の別名を入力します。

クライアント : SAP クライアント番号を入力します。

ユーザー名 : SAP ユーザー・インタフェース用の SAP ユーザー名を入力します。

言語 : 英語の場合は「EN」、ドイツ語の場合は「DE」を選択します。「DE」を選択した場合、説明のテキストがドイツ語で表示され、その他のテキストはすべて英語で表示されます。

9. 「次へ」をクリックします。

   図 21-5 に示すような、SAP アプリケーションの「ログイン」ダイアログが表示されます。

図 21-5「SAP ログイン」ダイアログ

10. SAP ユーザー・インタフェース用のユーザー名とパスワードを入力し、「ログイン」をクリックします。

「新規モジュール・ウィザード: データ・ソース情報」ページで入力した SAP アプリケーションのバージョン番号がソース・システムのアプリケーション・バージョンに一致しない場合は、エラー・メッセージが表示されます。「戻る」をクリックして「データ・ソース情報」ページに戻り、アプリケーションのバージョンを修正します。

エラー・メッセージが表示されない場合は、SAP アプリケーションにログインし、「終了」ページが表示されます。

メタデータ・インポート・ウィザードに直接進むには、ページの下部にあるチェック・ボックスを選択します。メタデータのインポートは後で行うこともできます。その場合は、このチェック・ボックスの選択を解除します。

11. 「終了」をクリックします。

新規モジュール・ウィザードによって新しい SAP ソース・モジュールが作成され、その名前がプロジェクトのナビゲーション・ツリーの「アプリケーション」ノード内に挿入されます。

SAP メタデータ定義のインポート

SAP ソース・モジュールを作成した後に、メタデータ・インポート・ウィザードを使用して SAP 表からメタデータ定義をインポートできます。メタデータ・インポート・ウィザードでは、インポートする SAP オブジェクトのフィルタリング、対象オブジェクトの確認およびオブジェクトの再インポートを実行できます。透過表、クラスタ表またはプール表のメタデータをインポートできます。

この項では、次のトピックについて説明します。

メタデータ・インポート・ウィザードを開く

「新規モジュール・ウィザード: 終了」ページからメタデータ・インポート・ウィザードに直接進む手順を実行した場合は、「メタデータ・インポート・ウィザード: ようこそ」ページが表示されます。メタデータのインポートを後で実行することを選択した場合は、作成した SAP モジュールからメタデータ・インポート・ウィザードを開く必要があります。

メタデータ・インポート・ウィザードを開く手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「アプリケーション」ノード、「SAP」ノードの順に開きます。
2. メタデータのインポート先となる SAP ソース・モジュールを右クリックして、ポップアップ・メニューから「インポート」を選択します。

   「メタデータ・インポート・ウィザード: ようこそ」ページが表示されます。
3. 「次へ」をクリックします。

SAP メタデータのフィルタリング

メタデータ・インポート・ウィザードにはフィルタ・ページがあり、その中で目的のメタデータを選択できます。このページには、2 つのフィルタ方法が用意されています。

- ビジネス・コンポーネント

  この方法を選択すると「Business Component Hierarchy」ツリーが表示されます。SAP ビジネス領域を参照し、インポートするメタデータを検索します。選択したビジネス領域内にある表と SAP アプリケーションの表名の一覧を参照できます。
- テキスト文字列照合

  「メタデータ・インポート・ウィザード: フィルタ情報」ページにあるフィールドにテキスト文字列情報を入力して表を検索します。SAP アプリケーション・データベースの内容をよく知っている場合は、この方法でより直接的な検索を実行できます。

「ビジネス・コンポーネント」により SAP メタデータをフィルタリングする手順は次のとおりです。

1. 「ビジネス・コンポーネント」を選択します。次に、「参照」をクリックして、「SAP R/3 ビジネス・コンポーネント階層」ダイアログを表示します。

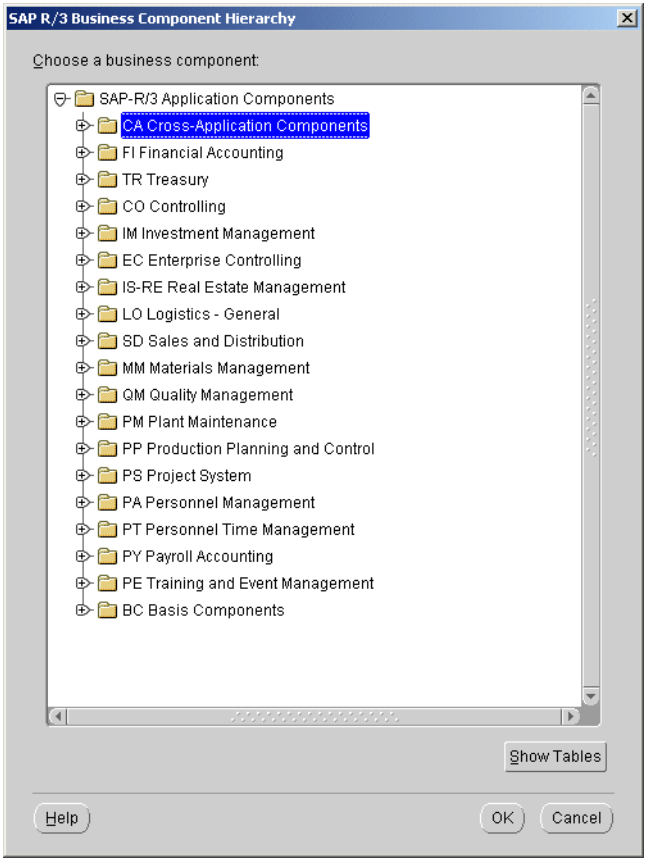
   SAP アプリケーション・サーバーがあるネットワーク・ロケーション、使用されている LAN のタイプ、SAP アプリケーション・データベースのサイズによっては、このダイアログが表示されるまでに 2 分から 10 分くらいかかることがあります。

   「ビジネス・コンポーネント・ツリーをロード中」ダイアログが表示されます。

2. 「OK」をクリックします。

   図 21-6 に示すような「Business Component Hierarchy」ダイアログが表示されます。

図 21-6「Business Component Hierarchy」ダイアログ

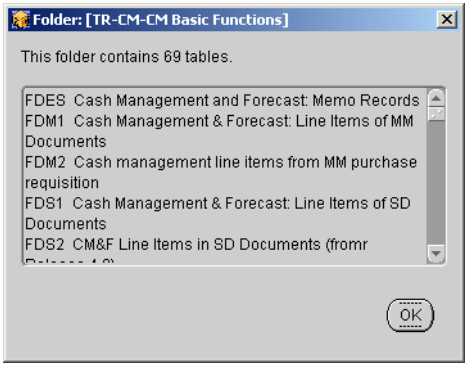


このダイアログを使用して、インポートするメタデータ・オブジェクトを含む SAP ビジネス領域を選択します。

3. 目的のフォルダを選択して「表の表示」をクリックし、そのビジネス・コンポーネントで使用可能な表を表示します。

   図 21-7 に示すように、選択したビジネス・コンポーネントに含まれる表の一覧が「フォルダ」ダイアログに表示されます。

図 21-7「フォルダ」ダイアログ



このダイアログを参照して、選択している表の数が正しいことを確認してください。

ビジネス・コンポーネントの中には、1000 を超える表を持つものもあります。ネットワーク接続の速度やソース・システムおよびターゲット・システムの処理能力によっては、このような大量のメタデータのインポートには、数時間またはそれ以上かかる場合があります。

4. 「OK」をクリックします。

   「メタデータ・インポート・ウィザード : フィルタ情報」ページが表示され、選択した SAP ビジネス領域がその「ビジネス・コンポーネント」フィールドに表示されます。

5. 「テキスト文字列、オブジェクトは次のもの」を選択します。「名前の一致」入力フィールドまたは「説明の一致」入力フィールドを選択して文字列を入力すると、一致する表が SAP データ・ソースから取得されます。

   - 「名前の一致」フィールドでは大文字と小文字は区別されませんが、「説明の一致」フィールドでは区別されます。
   - 選択したテキスト文字列入力フィールドには、必ず文字列を入力してください。空のままにしておくことはできません。

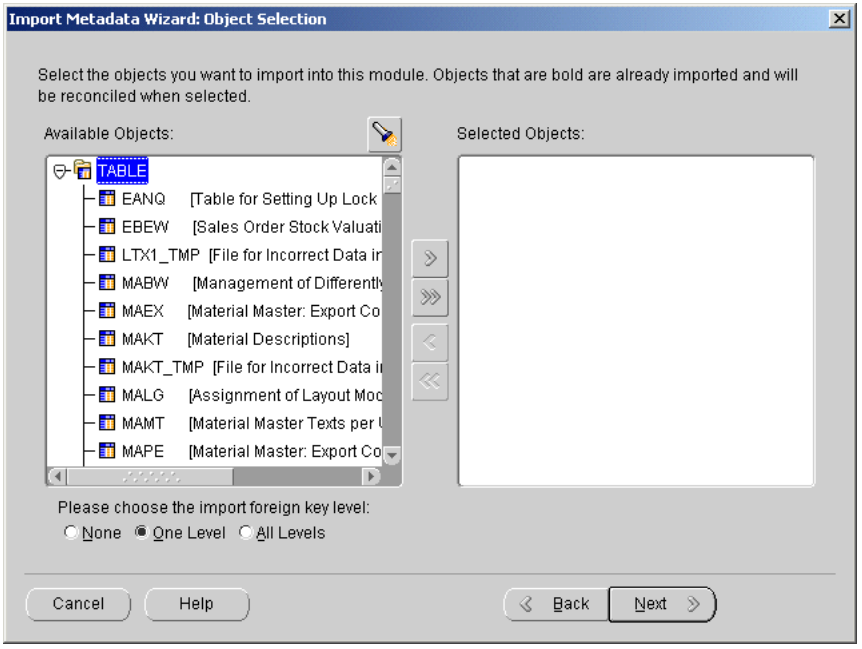
- ワイルド・カード文字を使用して、オブジェクトを選択するフィルタを作成できます。% は 0 個以上の一致する文字、_ は 1 個の一致する文字を表します。

  たとえば、A0 という名前のビジネス領域で、名前に CURRENCY という語を含む表を検索する場合は、`A0%CURRENCY%` と入力します。CURRENCY という名前の後に 1 文字が続く表のみに検索を絞り込む場合は、`A0%CURRENCY_` と入力します。

6. インポートする表の数を、「表示オブジェクトの最大数」フィールドで指定します。
7. 「次へ」をクリックします。

   図 21-8 に示すような「メタデータ・インポート・ウィザード : オブジェクトの選択」ページが表示され、各表の説明が表示されます。

図 21-8「メタデータ・インポート・ウィザード : オブジェクトの選択」ページ



メタデータ・インポート・ウィザードでは、インポートする各表と外部キー関係がある表もインポートするかどうかを選択できます。

8. 「使用可能なオブジェクト」リストから表を選択し、「>」ボタンを使用して「選択したオブジェクト」リストに移動します。

   SAP インテグレータでは表の定義のみがインポートされます。選択した表が、「メタデータ・インポート・ウィザード: オブジェクトの選択」ページの「選択したオブジェクト」リストに表示されます。

9. インポートに適用する外部キー・レベルを選択します。

   なし: 「選択したオブジェクト」リストに表示されているオブジェクトのみをインポートします。

   1 レベル: 「選択したオブジェクト」リストに表示されているオブジェクトと外部キー関係によって直接リンクされているすべての表をインポートします。

   すべてのレベル: 「選択したオブジェクト」リストに表示されているオブジェクトと外部キー関係によってリンクされているすべての表をインポートします。

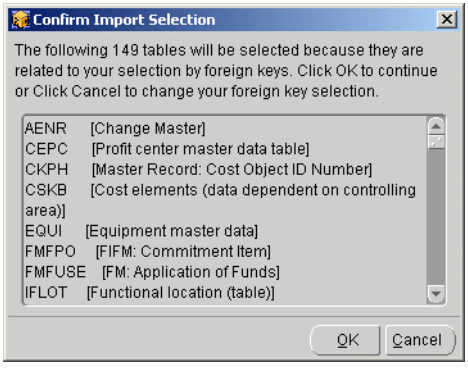
   選択した外部キー・レベルは、インポート対象として選択したすべての表に適用されます。

   「すべてのレベル」を選択すると、外部キー制約によって相互に関連付けられている表がすべてインポートされるため、メタデータのインポートにかかる時間が長くなります。このオプションは、本当に必要な場合にのみ選択してください。

10. 「次へ」をクリックします。

    「メタデータ・インポート・ウィザード: オブジェクトの選択」ページにある外部キー・レベルのラジオ・ボタンが「1 レベル」または「すべてのレベル」に設定されている場合は、図 21-9 に示すような「選択のインポートの確認」ダイアログが表示されます。

図 21-9「選択のインポートの確認」ダイアログ



このダイアログを参照して、選択している表の数が正しいことを確認してください。

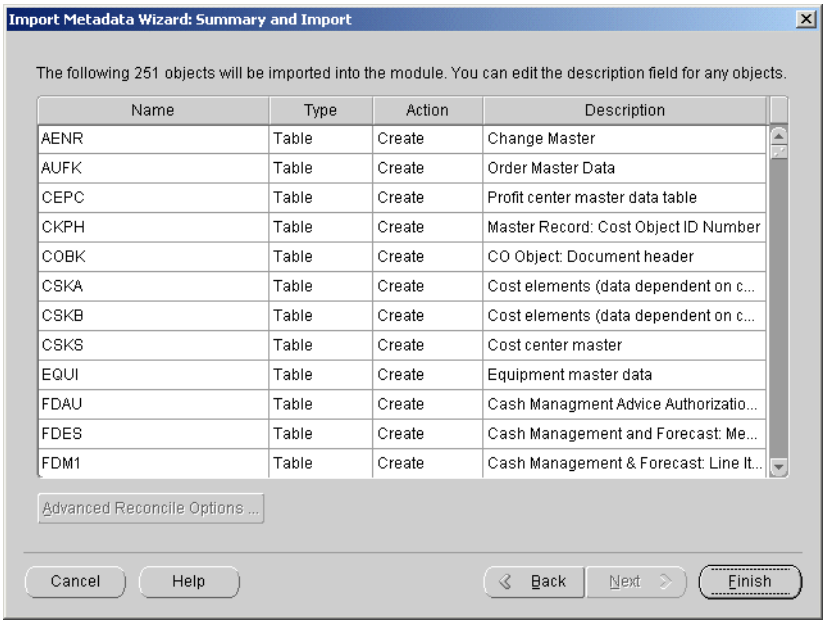
11. 「OK」をクリックします。

    選択したオブジェクトが、「メタデータ・インポート・ウィザード: オブジェクトの選択」ページの右側のリストに表示されます。

12. 「次へ」をクリックします。

    選択した表の定義が、ウィザードによって SAP アプリケーション・サーバーからインポートされ、SAP ソース・モジュールに保存されます。次に、図 21-10 に示すような「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページが表示されます。

図 21-10「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページ



「メタデータ・インポート・ウィザード: サマリーおよびインポート」ページの情報を確認します。

各表の説明フィールドを選択して新しい説明を入力することによって、表の説明を編集できます。

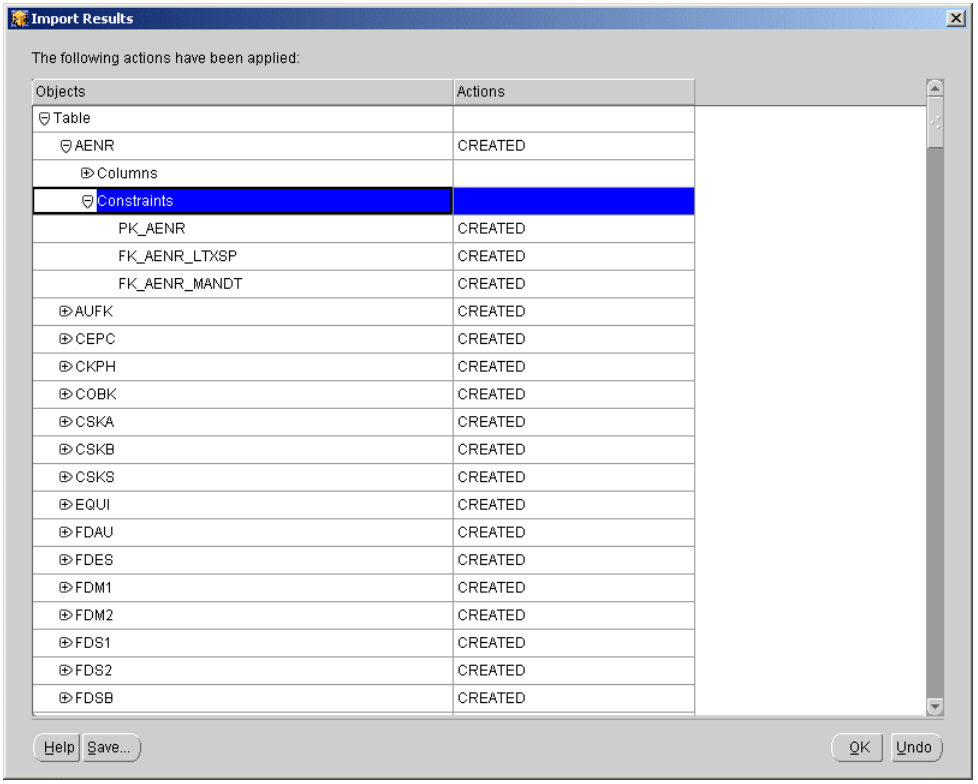
13. 「終了」をクリックします。

    SAP インテグレータによって、表定義が SAP アプリケーション・サーバーから読み取られ、メタデータ・オブジェクトが Warehouse Builder Design Repository に作成されます。

    Warehouse Builder Design Repository への SAP メタデータのインポートにかかる時間は、表のサイズと数および SAP アプリケーション・サーバーとリポジトリ間の接続状況により決まります。500 を超えるオブジェクトのインポートには、数時間またはそれ以上かかることがあります。特に、異なる LAN 上にあるサーバーに接続する場合は長時間かかります。

    インポートが完了すると、図 21-11 に示すような「インポート結果」ダイアログが表示されます。

図 21-11「インポート結果」ダイアログ



14. 「OK」をクリックします。

SAP オブジェクトの再インポート

SAP オブジェクトを再インポートする場合は、メタデータ・インポート・ウィザードのインポート手順を実行します。インポートを開始する前に、インポートするオブジェクトと同じ名前を持つ表がインポート先のソースにないかどうかが確認されます。再インポートされた表の名前は、「メタデータ・インポート・ウィザード : サマリーおよびインポート」ページに太字で表示されます。また「拡張調整オプション」ボタンがアクティブになり、再インポートに使用する各オプションを制御できます。

SAP ソース・モジュールの更新

SAP アプリケーション・バージョンを更新したとき、SAP サーバーを移行したときおよびネットワーク接続構成を変更したときは、既存の SAP ソース・モジュール定義を更新する必要があります。また、メタデータの再インポート時も、この情報を確認する必要があります。

SAP モジュールは、「モジュール・プロパティ」ダイアログでそのプロパティを編集することで更新できます。

SAP オブジェクト定義を更新する手順は次のとおりです。

1. Warehouse Builder ナビゲーション・ツリーで、「アプリケーション」ノード、「SAP」ノードの順に開きます。
2. 更新する SAP ソース・オブジェクトを右クリックして、ポップアップ・リストから「プロパティ」を選択します。

   「モジュール・プロパティ」ダイアログが表示されます。
3. 該当するタブを選択して、SAP オブジェクトのプロパティを編集します。

   名前 : モジュールの一意の識別子を変更するには、このタブを使用します。「名前」タブでは、次の内容を編集できます。

SAP オブジェクトの名前を編集します。名前によってオブジェクトに一意の値を割り当てます。名前には、30 文字以内の英数字を使用できます。30 文字の制限は物理名モードにのみ適用されます。

必要に応じて、SAP オブジェクトのステータスを変更します。「開発」、「品質保証」または「製品」を選択します。

SAP オブジェクトの説明を編集します。このフィールドには 4000 文字まで入力できます。たとえば、このフィールドを使用して、モジュールの目的を注記したり、プロジェクトのエンド・ユーザーが必要とする情報とモジュールとの関係を記載できます。

ソース : このタブは読取り専用です。このページを使用して、次の情報を参照します。

アプリケーション・タイプ : データ・ソースに該当するアプリケーション・タイプを示します。たとえば、SAP R/3 4.x などです。

システム・タイプ : ソース・データのホストに使用されているデータベース・システムのタイプ（SAP アプリケーション）を示します。

Integrator used to access the data source: ソース・データへの接続に使用された Warehouse Builder インテグレータを示します。

接続 : SAP アプリケーション・サーバーへのネットワーク接続情報が変更された場合のみ、このタブの情報を編集します。

SAP オブジェクトの ETL プロセスの定義

SAP ソース・モジュールを定義しメタデータをインポートした後は、ETL マッピングを定義して、データを SAP ソースからターゲットに抽出およびロードできます。Warehouse Builder SAP インテグレータは SAP オブジェクト用に特化されたマッピング・ツールとして機能し、これを使用してマッピングを構成し、メタデータの配布に使用する ABAP または PL/SQL コードを生成できます。

この項では、次のトピックについて説明します。

- SAP オブジェクトを含むマッピングの定義（21-19 ページ）
- SAP オブジェクトに対するコード生成の構成（21-24 ページ）
- SAP 定義の生成（21-29 ページ）

SAP オブジェクトを含むマッピングの定義

マッピング・エディタを使用して、SAP ソースのマッピングを定義できます。SAP マッピングは他のタイプのマッピングと類似していますが、次の重要な相違点があります。

- Warehouse Builder の SAP インテグレータでは、コピーおよびマップ・ウィザードを使用してターゲット演算子を定義できます。このウィザードは SAP 透過表にのみ使用できます。
- SAP オブジェクトには、表、キューブ、ディメンション、フィルタおよび結合子マッピング演算子のみ使用できます。

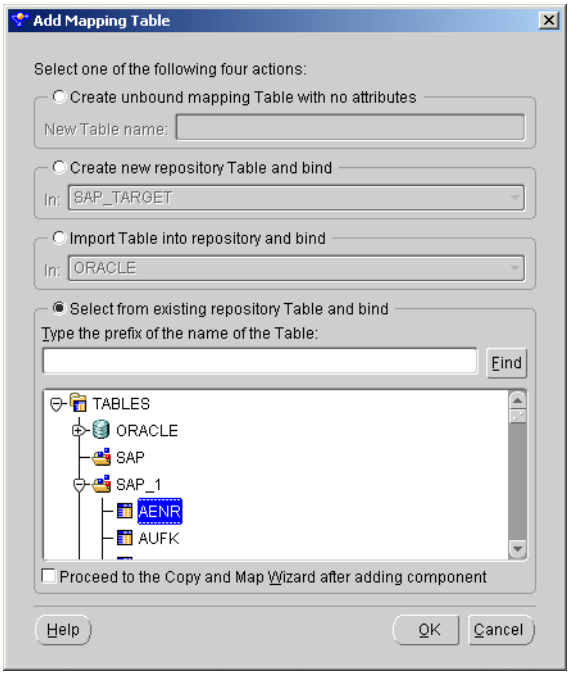
SAP オブジェクトのマッピングへの追加

SAP オブジェクトをマッピングに追加する手順は次のとおりです。

1. ツールボックスからマッピング・エディタ・キャンバスに、「表のマッピング」アイコンをドラッグ・アンド・ドロップします。

   図 21-12 に示すような「マッピング表の追加」ダイアログが表示されます。

図 21-12「マッピング表の追加」ダイアログ



2. 「既存リポジトリ表からの選択およびバインド」を選択します。

   ダイアログの下部のフィールドに、SAP 表のリストが表示されます。SAP 表の定義は、SAP ソース・モジュールにインポート済です。

3. ソース表の名前を選択して、「OK」をクリックします。

   マッピング・キャンバスに、SAP 表を示す表のマッピングが配置されます。

   マッピング・ターゲットの定義には、コピーおよびマップ・ウィザードを使用できます。または、使い慣れた他のタイプのマッピング演算子を使用して定義することもできます。

コピーおよびマップ・ウィザードでのターゲットの定義

SAP 演算子をソースとして選択すると、コピーおよびマップ・ウィザードを使用するオプションが使用可能になり、SAP ソースのマッピング先になるターゲット・オブジェクトを定義できます。このウィザードを使用して、SAP ソースの属性と同じ属性を持つターゲット・オブジェクトを作成できます。コピーおよびマップ・ウィザードでは、次の 3 つの機能を実行できます。

- 選択したソース・オブジェクトのコピー。
- ソース・オブジェクト定義に基づくターゲット・オブジェクトの作成。
- 新規ターゲット・オブジェクトへのソース・オブジェクトのマップ。

コピーおよびマップ・ウィザードは、ソースとして 1 つの表を選択している場合のみ使用できます。複数の表を選択する場合は、マッピング・ターゲットを手動で定義する必要があります。

コピーおよびマップ・ウィザードを使用する手順は次のとおりです。

1. 「マッピング表の追加」ダイアログで、ソースとして使用する SAP 表を選択します。
2. 「コンポーネントを追加した後コピーおよびマップ・ウィザードに進む」チェック・ボックスを選択します。
3. 「OK」をクリックします。

   「コピーおよびマップ・ウィザード : ターゲット名」ページが表示されます。
4. 「次へ」をクリックします。

   「コピーおよびマップ・ウィザード : ターゲット名」ページが表示されます。
5. 次の情報を入力します。

   物理名 : 作成するターゲット・オブジェクトの一意の物理名です。名前は 30 文字以内で指定します。スペースは使用できません。

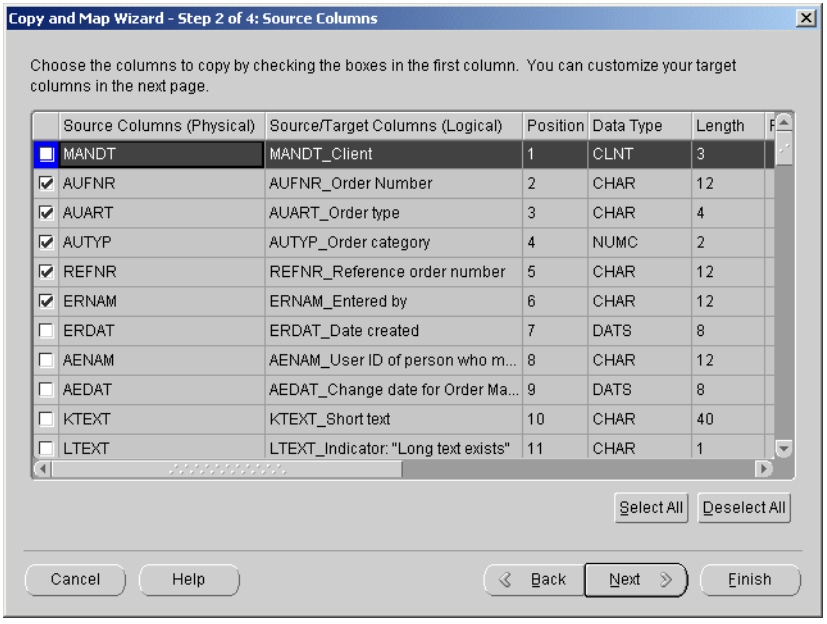
   オブジェクト・タイプ : SAP ソース表からマップする表のみを作成できます。このフィールドの選択肢は事前に選択されており、読取り専用です。

   アプリケーション : ターゲット・オブジェクトを含むターゲット・アプリケーションです。このフィールドの選択肢は事前に選択されており、読取り専用です。

   説明 : 作成するターゲット・オブジェクトの説明に使用するオプションのフィールドです。このフィールドには、最大 2MB のファイル情報を入力できます。
6. 「次へ」をクリックします。

   図 21-13 に示すような「Source Columns」ページが表示されます。

図 21-13「Source Columns」ページ



デフォルトでは、ソース表内のすべての列が SAP アプリケーションでの定義どおりに、選択済の状態でこのページに表示されます。選択を解除するには、次のいずれかの操作を行います。

- 列の横のチェック・ボックスを選択します。
- 表内の列をすべて選択解除するには、「すべて選択解除」をクリックします。

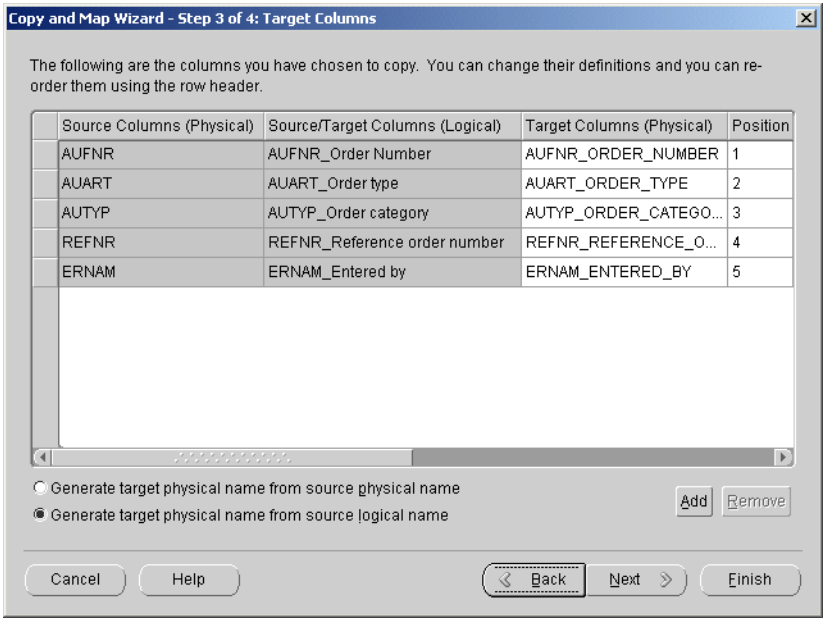
少なくとも、1 つの列を選択してコピーする必要があります。列を選択するには、次のいずれかの操作を行います。

- 列の横のチェック・ボックスを選択します。
- 表内の列をすべてコピーするには、「すべて選択」をクリックします。

7. 「次へ」をクリックします。

   図 21-14 に示すような「Target Columns」ページが表示されます。

図 21-14「Target Columns」ページ



このページを使用すると、選択した列のカスタマイズまたはターゲット表への新規列の追加を行うことができます。表内の最初の 2 列には、列の物理名とビジネス名が表示されます。列の物理名は変更できます。デフォルトの物理名がソース物理名からコピーされます。次のフィールドを編集できます。

- **Target Columns (Physical):** ターゲット列の物理名を変更するには、値をクリックしてテキストを編集します。新規列を追加するには、「追加」をクリックして、名前のフィールドに新しい名前を入力します。新規列を追加する場合は、名前を付ける必要があります。
- **位置 :** 行ヘッダーを上下にドラッグして列の位置を変更します。
- **データ型 :** 各列のデータ型を変更します。データ型は、Oracle データ型に自動的に変換されます。
- **NOT NULL:** （オプション）列に NULL 値を許可しない場合はチェックマークを付けます。このオプションは、制約チェックが必要なロードで一意キーおよび主キーに使用します。

■ ターゲット・オブジェクト内に新規列を追加するには、「追加」をクリックします。「削除」オプションは、作成した列に対してのみ使用できます。ソース・オブジェクトで定義された列をコピーしない場合は、「戻る」をクリックして「Source Columns」ページに戻り、その列を選択解除します。

- 「ソース物理名からターゲット物理名を生成」をクリックし、物理ソース列名から物理ターゲット列名をコピーします。
- 「ソースビジネス名からターゲット物理名を生成」をクリックし、論理ソース列名から物理ターゲット列名をコピーします。ビジネス名には、Oracle 物理名の制限である 30 文字を超える名前を指定できます。

8. 「OK」をクリックして列を自動的に改名するか、「取消し」をクリックして列を手動で改名します。

   ターゲット列を改名すると、その新規名は、ターゲット物理名の生成方法に関係なく保持されます。生成済のターゲット列名が、改名されたターゲット列名により置換されます。たとえば、ある表のソース物理名が MANDT、そのビジネス名が MANDT_Client であるとします。デフォルトのターゲット物理名を MANDT から MANDT_Client_Name に変更すると、この名前は保持されます。

9. 「次へ」をクリックします。

   「コピーおよびマップ・ウィザード : サマリーおよびコピー」ページが表示されます。このページの情報を確認します。

10. 「終了」をクリックします。

    マッピング・エディタによって、新規作成されたターゲット表にマップされたソース表が表示されます。バウンド・ターゲット表は、Warehouse Builder ナビゲーション・ツリーの「TABLES」ノードの下に表示されます。

SAP オブジェクトに対するコード生成の構成

SAP ソースを含むマッピングの構成は、他の種類のソースを含むマッピングの構成と類似しています。

- 「演算子のプロパティ」ウィンドウを使用してロード・プロパティを設定します。
- 「構成プロパティ」ウィンドウを使用してコード生成プロパティを定義します。
- ABAP コードを生成する場合は、「構成プロパティ」ウィンドウでディレクトリと初期化ファイルを設定します。

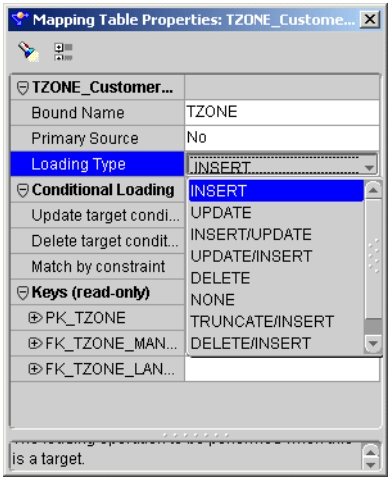
ロード・タイプの設定

SAP 演算子のロード・タイプを設定する手順は次のとおりです。

1. マッピング・エディタから、SAP ソース演算子を右クリックし、ポップアップ・メニューから「演算子のプロパティ」を選択します。

   図 21-15 に示すような「演算子のプロパティ」ウィンドウが表示されます。

図 21-15 SAP ターゲットの「Mapping Table Operator Properties」ウィンドウ



2. 「ロード・タイプ」ドロップダウン・リストから、目的のロード・タイプを選択します。マッピングのステップ・タイプとして ABAP コードを指定する場合は、表 21-1 に示す SQL*Loader コードが生成されます。

表 21-1 ABAP コードでのロード・タイプ

ロード・タイプ	ABAP コードに生成される SQL* Loader コード
INSERT	APPEND
CHECK/INSERT	INSERT
TRUNCATE/INSERT	TRUNCATE
DELETE/INSERT	REPLACE
その他のタイプすべて	APPEND

3. 「演算子のプロパティ」ウィンドウを閉じて、設定を保存します。

ステップ・タイプ・パラメータの設定

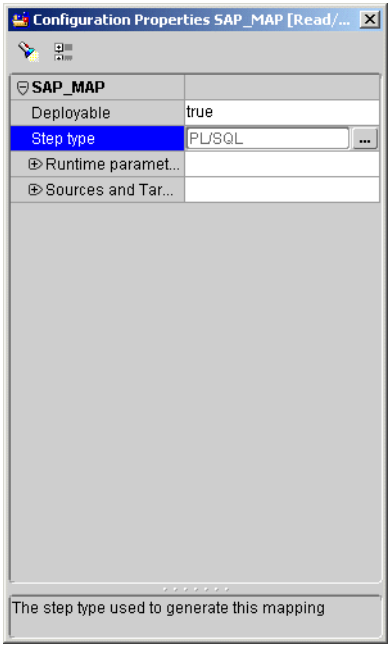
このパラメータによって、SAP マッピング用に生成するコードのタイプを選択できます。ソースにクラスタ表またはプール表が含まれる場合は、生成コードに ABAP を選択する必要があります。

ステップ・タイプを選択する手順は次のとおりです。

1. 対象のマッピングを右クリックして、ポップアップ・メニューから「構成」を選択します。

   図 21-16 に示すような「構成プロパティ」ウィンドウが表示されます。

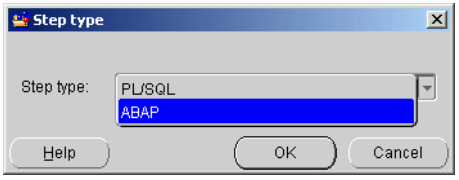
図 21-16「構成プロパティ」ウィンドウの「ステップ・タイプ」



2. 「ステップ・タイプ」フィールドをクリックして、「...」ボタンをクリックします。

   図 21-17 に示すような「ステップ・タイプ」ダイアログが表示されます。

図 21-17「ステップ・タイプ」ダイアログ



ドロップダウン・リストから、生成するコードのタイプを選択します。ABAP スクリプトまたは PL/SQL スクリプト（透過表用にのみ使用可能）を選択できます。

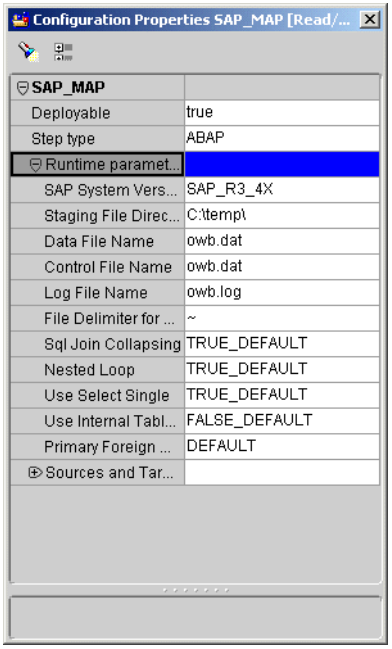
3. 「OK」をクリックします。

「構成プロパティ」ウィンドウが表示されます。

ランタイム・パラメータの設定

ステップ・タイプを「ABAP」に設定した場合は、「構成プロパティ」ウィンドウの「ランタイム・パラメータ」ノードを開いて、図 21-18 に示すような ABAP コード生成に固有の設定を表示できます。これらの設定にはコード生成を最適化するプロパティが事前定義されていますので、いっさい変更しないでください。設定を変更した場合、コード生成プロセスの速度が遅くなる可能性があります。

図 21-18「構成プロパティ」ウィンドウの「ランタイム・パラメータ」



SAP マッピングで使用できるランタイム・パラメータの一覧を次に示します。

- **SAP システム・バージョン :** ABAP コードを配布する SAP システムのバージョン番号を指定します。
- **ステージング・ファイル・ディレクトリ :** ABAP コードによって生成されるデータが存在するディレクトリのロケーションを指定します。
- **データ・ファイル名 :** コード生成時に作成されるデータ・ファイルの名前を指定します。
- **制御ファイル名 :** コード生成時に作成される制御ファイルの名前を指定します。
- **ログ・ファイル名 :** コード生成時に作成されるログ・ファイル名を指定します。このファイルはデバッグに役立ちます。
- **ステージング・ファイルのファイル・デリミタ :** SQL データ・ファイルの列セパレータを指定します。
- **SQL 結合失敗 :** 可能であれば、ABAP コードを生成するための次のヒントを指定します。

  ```
  Select < > into < > from (T1 as T1 inner join T2 as T2) on <condition >
  ```

  デフォルト設定は True です。
- **ネステッド・ループ :** 可能であれば、結合用のネステッド・ループ・コードを生成するためのヒントを指定します。
- **Select Single の使用 :** 可能であれば、Select Single... が生成されるかどうかを示します。
- **参照専用表に内部表を使用 :** 可能であれば、内部表を使用してコードを生成するかどうかを指定します。

生成ターゲットのディレクトリの設定

SAP データをリポジトリにロードする前に、「Physical Configuration」プロパティ・ウィンドウで、各ディレクトリ・パラメータが適切に設定されていることを確認する必要があります。

生成ターゲットのディレクトリにおける ABAP コードの生成に関連するパラメータの一覧を次に示します。

- **ABAP ディレクトリ :** ABAP スクリプトの格納ロケーションを設定します。デフォルトは `abap¥` です。
- **ABAP 拡張子 :** ABAP スクリプトのファイル名拡張子を設定します。デフォルトは `.abap` です。

- **ABAP 実行パラメータ・ファイル:** ABAP ジョブにおけるパラメータ・スクリプトの実行パラメータ・ファイルに付ける接尾辞を指定します。デフォルトは `_run.ini` です。
- **ABAP スプール・ディレクトリ:** スクリプト生成処理で、ABAP スクリプトがバッファに使用されるロケーションを設定します。デフォルトは `abap¥log¥` です。

SAP 定義の生成

SAP 透過表を含むマッピングでの PL/SQL コードの生成方法は、Warehouse Builder における他の PL/SQL マッピングのコード生成方法と同じです。ただし、プール表およびクラスタ表には ABAP コードを生成する必要があります。

SAP ソース・オブジェクトの作成および移入に必要なスクリプトが、Warehouse Builder によって検証および生成されます。

コードの生成時に、作成する名物理オブジェクトに対して 1 つのスクリプトが生成されます。たとえば、作成中の各索引に対して 1 つのスクリプトが存在します。これは、後でウェアハウス全体ではなく、1 つのオブジェクトを再配布する必要がある場合に有効です。

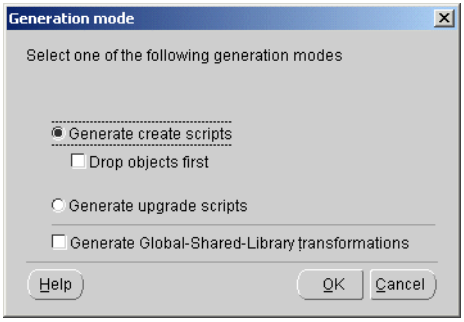
関連情報は、第 13 章「ターゲット・システムの配布」を参照してください。

SAP マッピング用のスクリプトを生成する手順は次のとおりです。

1. ウェアハウス・モジュール・エディタのメニューから、「モジュール」→「生成」を選択します。

   図 21-19 に示すような「生成モード」ダイアログが表示されます。

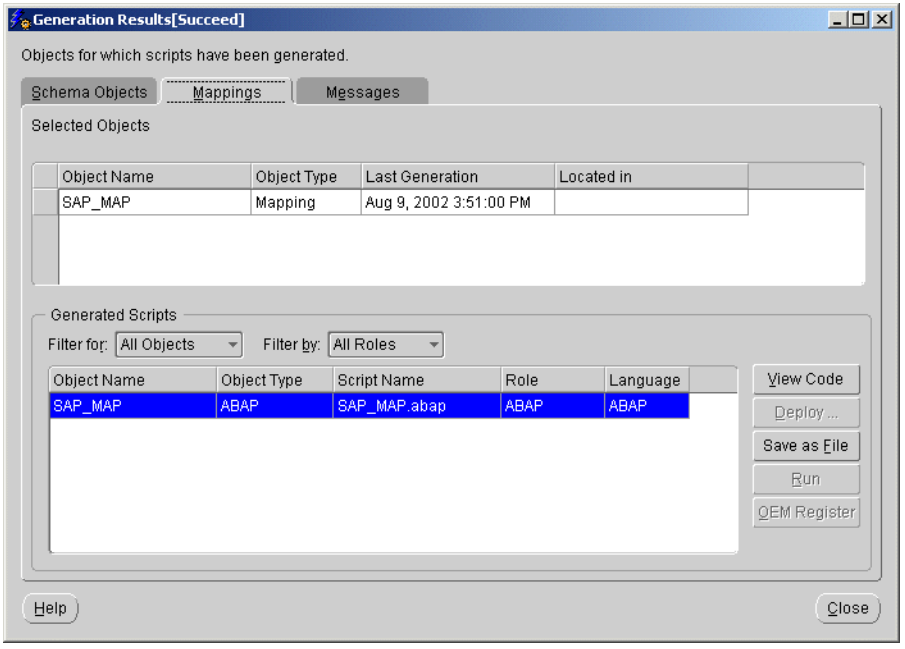
図 21-19「生成モード」ダイアログ



2. 適切な生成モードを選択して、「OK」をクリックします。

   図 21-20 に示すような「生成結果」ダイアログが表示されます。

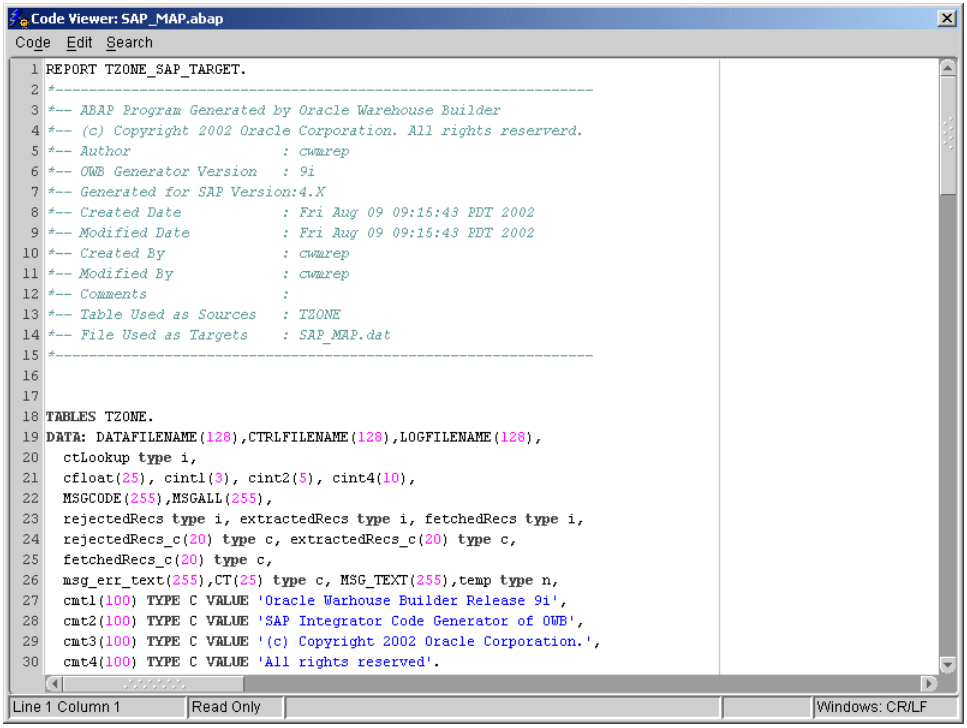
図 21-20「生成結果」ダイアログ



3. 「コードの表示」をクリックします。

   図 21-21 に示すように、生成済コードがコード・ビューアに表示されます。

図 21-21 コード・ビューアに表示された生成済コード

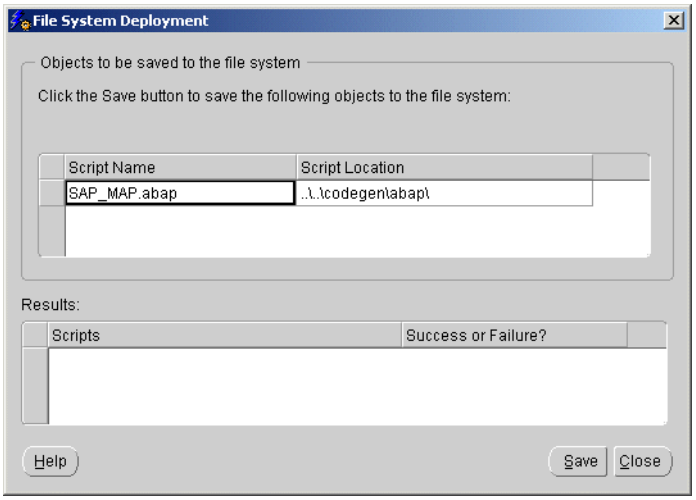


このコード・エディタを通して、ファイルを編集、印刷または保存できます。ウィンドウを閉じて「生成結果」ダイアログに戻ります。

4. 「生成結果」ダイアログで、「ファイルとして保存」をクリックして、生成された ABAP コードをハード・ドライブに保存します。

   図 21-22 に示すような「ファイル・システムの配布」ダイアログが表示されます。

図 21-22「ファイル・システムの配布」ダイアログ



5. 「保存」をクリックして生成されたスクリプトをファイル・システムに保存します。ABAP コードは任意のファイル拡張子を使用して保存できます。たとえば、MAP1.abap のように接尾辞に続けて .abap を使用することも、他の任意のネーミング規則を使用することもできます。

SAP データのリポジトリへのロード

SAP マッピング用の ABAP コードを生成したら、Warehouse Builder によって、データをロードする ABAP プログラムが作成されます。このプログラムは、SAP ユーザー・インタフェースから実行する必要があります。このプログラムにより、生成済コードをアップロードし、SAP システム上で実行します。次に、データを Warehouse Builder ステージング領域にロードしてから、SQL*Loader を使用してデータをウェアハウス表にアップロードします。

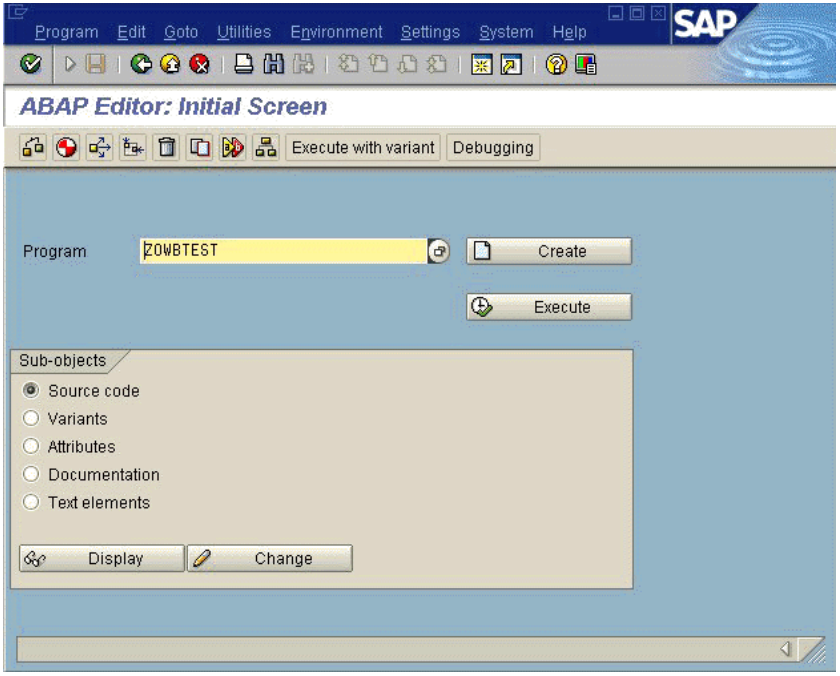
SAP システムで SAP ユーザー・インタフェースを使用して、ABAP コードをアップロードおよび実行する手順は次のとおりです。

1. SAP ユーザー・インタフェースを開き、OP コードの `SE38` を指定します。
2. ABAP コードを実行するプログラムを作成します（例 : ZOWBTEST1）。プログラム作成の詳細な手順は、SAP のドキュメントを参照してください。テスト目的ですでにプログラムが作成されている場合は、そのプログラムを使用して ABAP コードを実行できます。

   デフォルトでは、「Source Code」が選択されています。

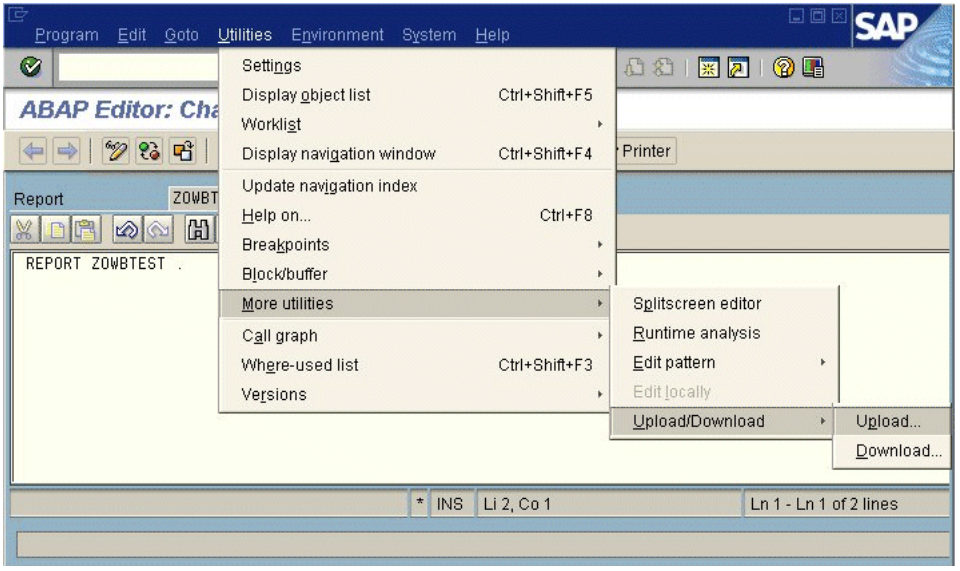
   図 21-23 に、SAP ABAP エディタを示します。

図 21-23 SAP ABAP エディタ



3. 「Change」をクリックします。

図 21-24 SAP ABAP エディタの「Upload」コマンド



4. 図 21-24 に示すように、ABAP エディタ画面で「Utilities」→「Upload/Download」→「Upload」を選択します。

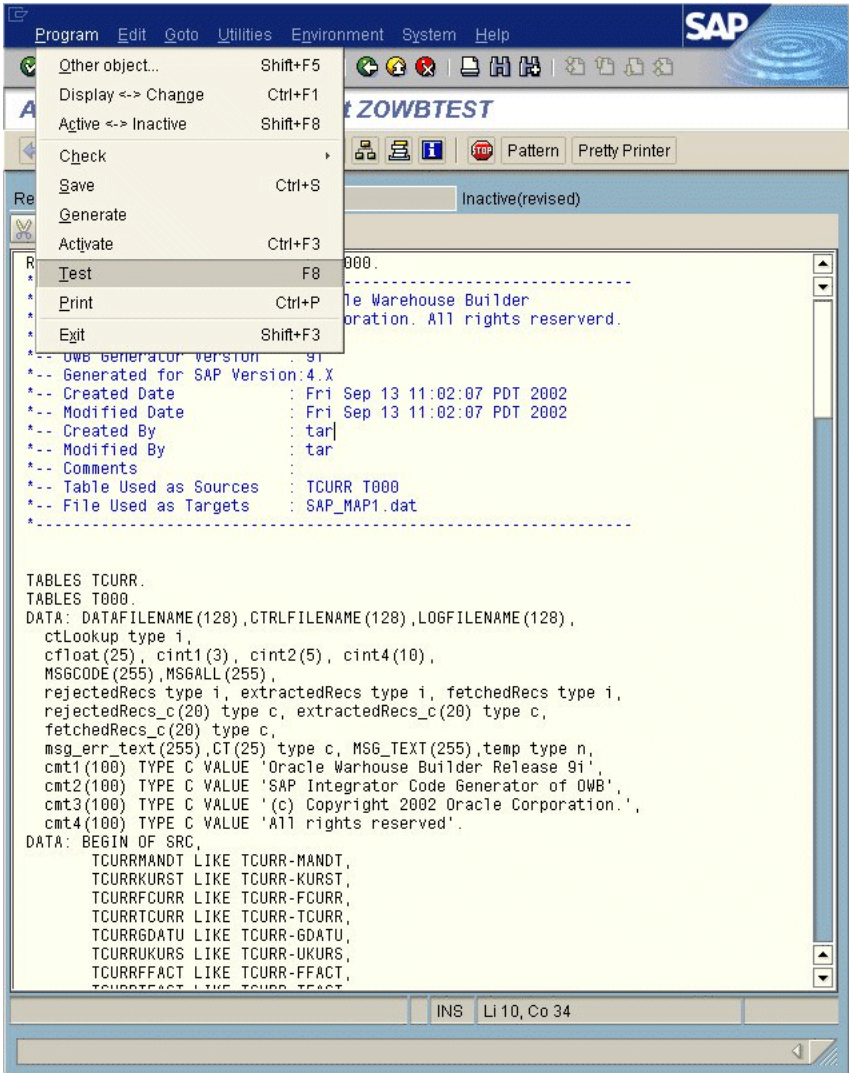
   図 21-25 に示すような「Import from a Local File」ダイアログが表示されます。

図 21-25 SAP の「Import from a Local File」ダイアログ

5. Warehouse Builder によって生成された ABAP コードのあるロケーションを指定します。

6. Click **「Transfer」をクリックします。**

図 21-26 SAP での ABAP コードの実行



7. [F8] を押して ABAP コードを実行します。または、「Program」→「Check」を選択してから、「Program」→「Execute」を選択してコードを実行することもできます。

   Warehouse Builder で生成された ABAP コードが、SAP アプリケーション・サーバーで実行されます。

8. FTP を使用して SAP アプリケーションからデータをフェッチし、Warehouse Builder のステージング領域に送信します。
9. SQL*Loader を使用してデータをウェアハウスの表にアップロードします。次にコマンドラインの例を示します。

```
SQLLDR USERID=scott/tiger CONTROL=abap_datactlfile.dat LOG=yourlogfile.log
```

透過表の PL/SQL スクリプトの配布

SAP 透過表の PL/SQL の配布方法は、Oracle データベース・ソースの PL/SQL スクリプトの配布方法と同じです。Oracle データ・ウェアハウスで PL/SQL スクリプトが実行され、リモート・クエリーを実行して SAP アプリケーションから表データを抽出します。

関連情報は、第 13 章「ターゲット・システムの配布」を参照してください。

22

その他の BI 製品と Warehouse Builder のメタデータの統合

Warehouse Builder には、そのメタデータを他のビジネス・インテリジェンス製品と共有するためのユーティリティが用意されています。この章では、最初に、Warehouse Builder 転送ウィザードを使用して他のシステムにエクスポートできる一連の Warehouse Builder オブジェクトの定義を格納するコレクションの作成方法について説明します。次に、Warehouse Builder メタデータを使用したビジネス・インテリジェンス（BI）統合とオンライン分析処理（OLAP）を可能にする手順について説明します。この章では、次のトピックについて説明します。

Warehouse Builder の統合機能の概要

この章の次の項では、ビジネス・インテリジェンス（BI）統合について説明します。

- **Warehouse Builder 転送ウィザード :** このウィザードを使用すると、様々な BI ツールに格納されたメタデータを同期化、統合、および使用できます。CWM 準拠アプリケーション、Oracle Discoverer、Oracle Express および Oracle OLAP Server とメタデータを交換できます。転送ウィザードの使用手順については、この章で説明します。22-6 ページの「転送ウィザードを使用したビジネス・インテリジェンス統合」を参照してください。
- **Warehouse Builder とのオンライン分析統合 :** Warehouse Builder は、他のデータ・ソースから Oracle Database に、多次元 OLAP オブジェクトを設計、配布およびロードできる唯一のツールです。データをロードした後は、ツールとアプリケーションを使用して、複雑な分析クエリを実行し、ビジネス上の質問の回答を得ることができます。Warehouse Builder では、同じメタデータから、リレーショナル用と分析用両方のデータ・ストアを、作成および管理できるようになりました。22-16 ページの「Warehouse Builder でのオンライン分析処理（OLAP）」を参照してください。

コレクションの定義

コレクションは、Warehouse Builder の領域で、他のツールやシステムにエクスポートするメタデータはここに格納されます。コレクションの配布には、Warehouse Builder 転送ウィザードを使用できます。コレクションを使用して次のタスクを実行します。

- 大規模な論理ウェアハウスの編成
- コレクション内の一連のオブジェクトの検証および生成
- Warehouse Builder 転送ウィザードによる他のツールへのメタデータのエクスポート

Warehouse Builder 転送ウィザードを使用して、Warehouse Builder から Oracle Discoverer、OLAP Server および CWM にコレクションをエクスポートできます。他ツールへのコレクションのエクスポートの詳細は、22-14 ページの「Warehouse Builder からのメタデータのエクスポート」を参照してください。

コレクションを作成する際に、新規オブジェクトや既存オブジェクトのコピーは作成しません。プロジェクト内にすでに存在しているオブジェクトを指すショートカットを作成します。ショートカットを使用することで、ベース・オブジェクトへのアクセスとその変更を迅速に実行できます。

1 つのプロジェクトに複数のコレクションを定義でき、オブジェクトは複数のコレクションから参照できます。たとえば、プロジェクトにアクセスする各ユーザーは、頻繁に使用するオブジェクトに対して自分専用のコレクションを作成できます。また、各ユーザーは、同一オブジェクト（マッピング、表、プロセス・フローなど）を別々のコレクションに追加できます。

各ユーザーは、ショートカットまたはベース・オブジェクトを削除することもできます。削除されたオブジェクトへのショートカットは削除されるか、またはコレクション内でグレー表示されます。グレー表示されたショートカットをコレクションから削除するには、削除するショートカットを右クリックして、「削除」を選択します。

ユーザーがコレクション内のオブジェクトを開くと、オブジェクトはそのユーザー用にロックされます。これにより、他のユーザーが別のコレクションからこのオブジェクトを編集することができなくなります。

コレクションの作成

コレクションを定義するには、新規コレクション・ウィザードを使用します。

新規コレクションを定義する手順は次のとおりです。

1. ナビゲーション・ツリーで、プロジェクトのノードを選択して開きます。
2. 「コレクション」ノードを右クリックして「コレクションの作成」を選択します。

   「新規コレクション・ウィザード: ようこそ」ページが表示されます。
3. 「次へ」をクリックします。

   「新規コレクション・ウィザード - ステップ 1/2: 名前」ページが表示されます。
4. コレクションの名前と説明（オプション）を入力します。
5. 「次へ」をクリックします。

   「新規コレクション・ウィザード - ステップ 2/2: コンテンツ」ページが表示されます。
6. 左側のパネルで、プロジェクトのノードを選択して開きます。

   コレクションに追加するオブジェクトの一覧が表示されます。
7. 左側のパネルで、「使用可能なオブジェクト」からプロジェクトを選択します。

   複数のオブジェクトを選択するには [Ctrl] キーを使用します。オブジェクトは、オブジェクト・レベルまたはモジュール・レベルで選択できます。たとえば、「ファイル」ノードでは、対象のフラット・ファイル・モジュールで、特定のファイルまたはすべてのファイルを追加できます。

   モジュールまたは別のコレクションを追加する場合は、モジュールまたはコレクションへの参照が作成されます。また、そのモジュールまたはコレクションに含まれるオブジェクトへの参照も作成されます。
8. 「>」ボタンをクリックします。

   右側のパネルで、「選択したオブジェクト」にオブジェクトの一覧が表示されます。この一覧からオブジェクトを削除するには § そのオブジェクトを選択して「<」ボタンをクリックします。

9. 「次へ」をクリックします。

   「新規コレクション・ウィザード: 終了」ページが表示されます。

10. 「新規コレクション・ウィザード: 終了」ページを使用して、コレクション用に選択したオブジェクトを確認します。

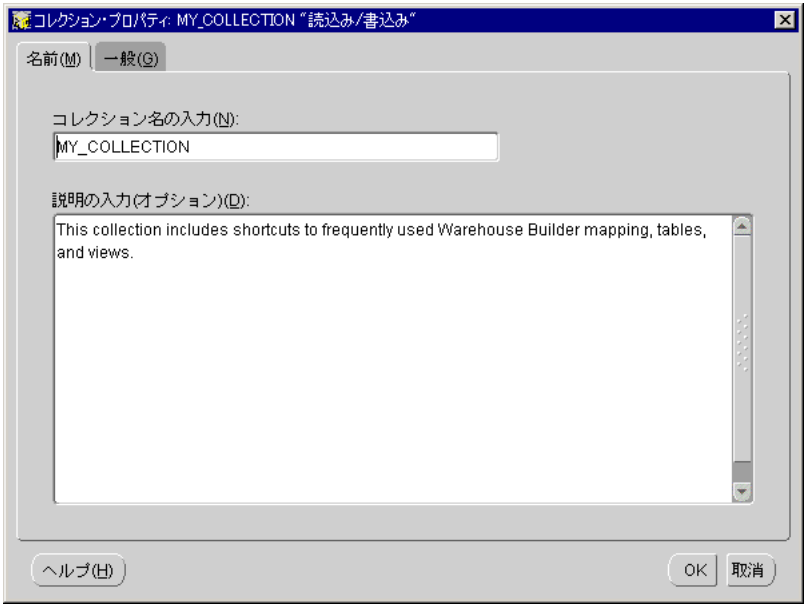
    選択した内容を変更するには、「戻る」をクリックします。

    「終了」を選択すると、コレクションが作成され、ナビゲーション・ツリーに追加されます。コレクション下にあるすべてのオブジェクトは、そのアイコンにショートカット記号が挿入されます。

コレクションの編集

コレクションを編集するには、ナビゲーション・ツリーで目的のコレクションを右クリックし、「プロパティ」を選択します。図 22-1 に示すようなプロパティ・ウィンドウが表示されます。

図 22-1 コレクションのプロパティ・ウィンドウ



プロパティ・ウィンドウには、情報の編集に使用する次のタブがあります。

- 名前
- 内容

「名前」タブでは、次の情報を指定します。

- **名前 :** コレクション名を変更できます。プロジェクト内で一意となるコレクション名を入力します。物理ネーミング・モードでは、1 ～ 200 の文字数で名前を入力します。空白は使用できません。論理モードでは、最大文字数は 200 で、空白も使用できます。
- **説明 :** テキストによる説明（オプション）は、4000 文字以内で編集できます。

「内容」タブでは、次の情報を指定します。

「新規コレクション・ウィザード - ステップ 2/2: コンテンツ」ページを使用して、コレクションで参照するオブジェクトを選択します。

「新規コレクション・ウィザード - ステップ 2/2: コンテンツ」ページの作業を完了する手順は次のとおりです。

1. 左側のパネルで、プロジェクトのノードを選択して開きます。

   コレクションに追加するオブジェクトの一覧が表示されます。

2. 左側のパネルで、「使用可能なオブジェクト」からプロジェクトを選択します。

   複数のオブジェクトを選択するには [Ctrl] キーを使用します。オブジェクトは、オブジェクト・レベルまたはモジュール・レベルで選択できます。たとえば、「ファイル」ノードでは、対象のフラット・ファイル・モジュールで、特定のフラット・ファイルまたはすべてのフラット・ファイルを追加するように選択できます。

3. 「>」ボタンをクリックします。

   右側のパネルで、「選択したオブジェクト」にオブジェクトの一覧が表示されます。この一覧からオブジェクトを削除するには、そのオブジェクトを選択して「<」ボタンをクリックします。

転送ウィザードを使用したビジネス・インテリジェンス統合

Warehouse Builder 転送ウィザードは、フォーマットの異なる複数のソースに保存されているメタデータを同期化、統合し、使用できるようにします。Warehouse Builder 転送ウィザードを使用すると、各種ツールからメタデータをインポートしたり、各種ツールへメタデータをエクスポートしたりできます。

転送ウィザードでは、次の 2 つの主要タスクを実行できます。

1. 様々なターゲットへのブリッジを使用して、Warehouse Builder Design Repository から選択したメタデータをエクスポートします。次のターゲットへのエクスポートができます。

   - CWM（Common Warehouse Metamodel）準拠アプリケーション・バージョン 1.0
   - Oracle Discoverer 4*i* および Oracle9*i*AS Discoverer
   - Oracle Express
   - Oracle OLAP Server

   Oracle Discoverer および Oracle Express のブリッジは、Windows プラットフォームで Oracle Warehouse Builder Design Client が実行されている場合にのみ使用できます。

2. ブリッジを使用して、ソース・ツールから Warehouse Builder Design Repository へ選択したメタデータをインポートします。次のソースからのインポートができます。

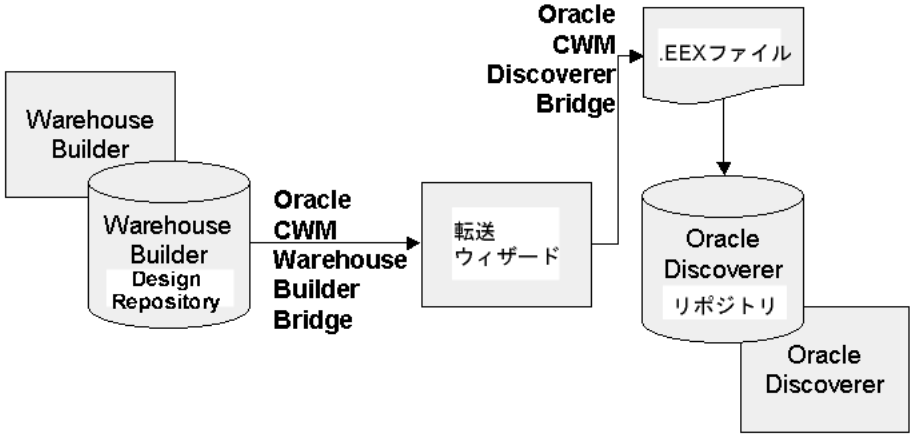
   - Oracle OLAP Server
   - OMG CWM（Common Warehouse Metamodel）準拠アプリケーション・バージョン 1.0
   - Computer Associates Erwin（3.5.1）
   - Powersoft PowerDesigner（バージョン 6）

   ERwin ブリッジは、Windows プラットフォームで Oracle Warehouse Builder Design Client が実行されている場合にのみ使用できます。

CMW アプリケーションでは、XML Metadata Interchange（XMI）標準に準拠した中間的な XML ファイルが作成されます。転送ウィザードを使用する場合、このプロセスは透過的に実行されます。ソース・パラメータとターゲット・パラメータを指定すると、転送ウィザードによってエクスポート、変換、ダウンロードが実行されます。

図 22-2 は、Warehouse Builder から Discoverer にメタデータをエクスポートする場合の転送プロセスを示します。

図 22-2 Warehouse Builder 転送ウィザードによるモデル



Meta Integration Model Bridge（MIMB）との統合

Warehouse Builder は Meta Integration Model Bridge（MIMB）と統合できます。MIMB は、独自のメタデータ・ファイルやリポジトリにあるメタデータを、標準の CWM フォーマットに変換します。このフォーマットであれば、転送ウィザードを使用して Warehouse Builder にインポートできます。MIMB と統合した後は、CA ERwin、Sybase PowerDesigner などの各種ソースから、これらのブリッジを介してメタデータをインポートすることもできます。

MIMB 統合によって、次のソースから Warehouse Builder にメタデータをインポートできます。

- OMG CWM 1.0: JDBC 1.0/2.0 を使用したデータベース・スキーマ
- OMG CWM 1.0: Meta Integration Repository 3.0
- OMG CWM 1.0: Meta Integration Works 3.0
- OMG CWM 1.0: XML DTD 1.0 (W3C)
- OMG CWM 1.0: HL7 3.0 用の XML DTD
- Acta Works 5.x
- 適応リポジトリ
- ArgoUML

- BusinessObjects Data Integrator
- BusinessObjects Designer 5.1.3 から 5.1.5
- CA COOL
- CA ERwin
- CA ParadigmPlus
- Hyperion Application Builder
- IBM
- Informatica PowerMart 5.1
- Merant App Master Designer 4.0
- Miscrodoft Visio
- Oracle Designer
- Popkin System Architect
- ProActivity 3.x & 4.0
- Rational Rose
- Select SE 7.0
- Silverrun
- Sybase Power Designer
- Unisys Rose
- Visible IE Advantage 6.1
- XML/XMI の各種バージョン

Warehouse Builder を MIMB と統合するには、次の手順に従います。

Meta Integration Model Bridge のダウンロード

MIMB がシステムにダウンロードされたら、Warehouse Builder によりインストールが自動認識され、「Warehouse Builder 転送ウィザード: メタデータ・ソースとターゲットの識別」ページに、新しいインポート・オプションが表示されます。

MIMB をダウンロードする手順は次のとおりです。

1. 次の Web サイトから Model Bridge (personal) 製品をダウンロードします。
   http://www.metaintegration.net/Products/Downloads/
2. MIMB をインストールするには、システムのセットアップを実行します。
3. インストール中に、「Setup Type」ページでインストール・タイプに「Typical with Java Extensions」を選択します。

   インストール中のマシンに JDK が見つからないというメッセージが表示された場合は、JNI ライブラリ・ディレクトリのパス名を指定する必要があります。パスの環境変数には、メタ統合ディレクトリを含める必要があります。メタ統合ディレクトリが含まれない場合は、パスの環境変数に *c:¥program files¥metaintegration¥win32* を追加する必要があります。
4. Warehouse Builder 転送ウィザード（インポート）を起動して、Warehouse Builder Design Client で MIMB を有効にします。Warehouse Builder 転送ウィザードを使用したメタデータのインポートの詳細は、22-9 ページの「Warehouse Builder へのメタデータのインポート」を参照してください。

Warehouse Builder へのメタデータのインポート

この項では、Warehouse Builder 転送ウィザードを使用して、メタデータを Warehouse Builder にインポートする方法について説明します。

インポート用の転送ウィザードを起動する手順は次のとおりです。

1. 「プロジェクト」メニューから「メタデータのインポート」を選択し、「ブリッジ」を選択します。

   「Oracle Warehouse Builder 転送ウィザード: ようこそ」ウィンドウが表示され、この転送ウィザードで実行する手順が示されます。

   転送ウィザードのバージョン情報を表示するには、「Oracle OWB 転送ツールについて」をクリックします。各ブリッジのバージョン情報を参照するには、「Oracle OWB 転送ツールについて」ダイアログの「Bridge のバージョン」ボタンをクリックします。
2. 「次へ」をクリックします。

   図 22-3 に示すような「Oracle Warehouse Builder 転送ウィザード: メタデータ・ソースとターゲットの識別」ウィンドウが表示されます。

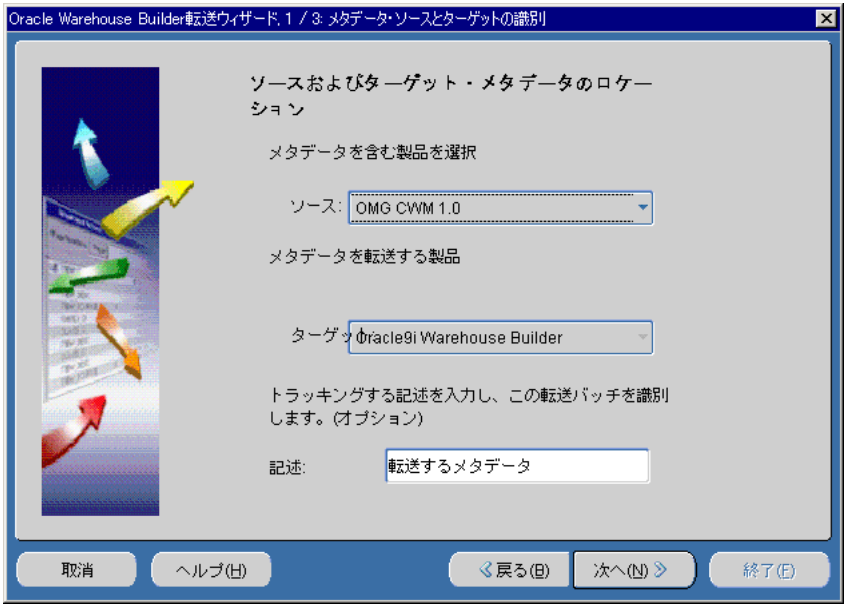
3. 「ソース」フィールドで、メタデータ・ソースを指定します（Oracle9*i* OLAP、OMG CWM 1.0、CA ERwin、PowerDesigner）。Meta Integration Model Bridge（MIMB）と統合する場合は、さらに多種類のソースからメタデータをインポートできます。詳細は、22-7 ページの「Meta Integration Model Bridge（MIMB）との統合」を参照してください。

   MIMB と統合する場合は、「ソース」フィールドの横に「情報」ボタンが表示されます。ソース・タイプを選択して「情報」をクリックすると、そのソース・タイプに使用するブリッジの使用方法について、詳細が表示されます。

4. 必要に応じて、転送するメタデータの説明を「説明」フィールドに入力します。

   この説明は、転送プロセスの実行中に進行状況バーに表示されます。

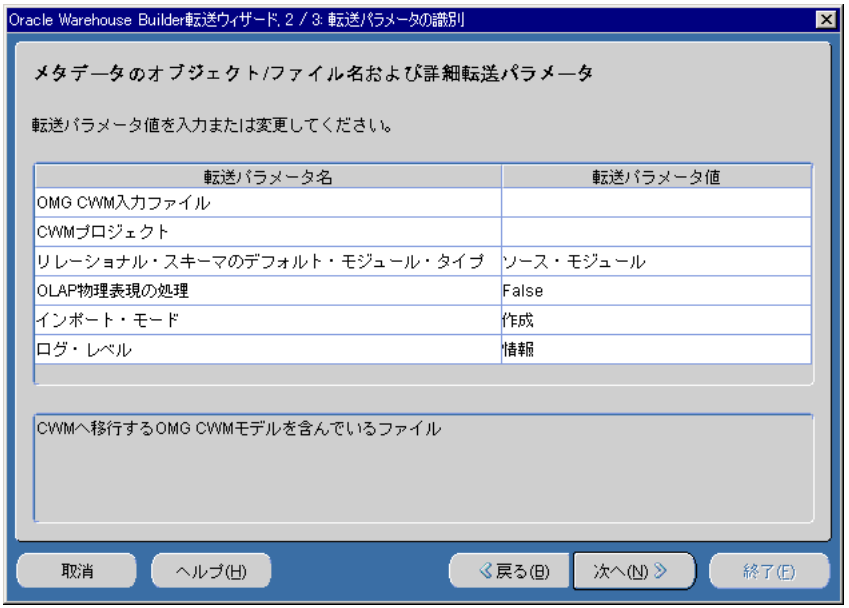
図 22-3「Oracle Warehouse Builder 転送ウィザード : メタデータ・ソースとターゲットの識別」ウィンドウ



5. 「次へ」をクリックします。

   図 22-4 に示すような「Oracle Warehouse Builder 転送ウィザード : 転送パラメータの識別」ウィンドウが表示されます。

図 22-4「Oracle Warehouse Builder 転送ウィザード : 転送パラメータの識別」ウィンドウ

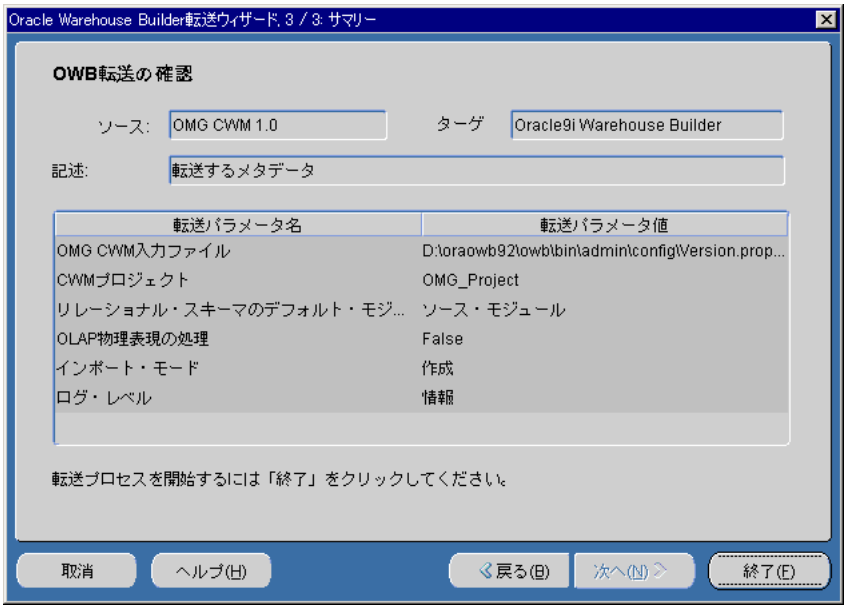


このウィンドウには、選択したメタデータ・ソースで使用するパラメータのリストが表示されます。詳細は、22-40 ページの「転送パラメータ」を参照してください。

Meta Integration Model Bridge（MIMB）を実行している場合は、このウィザード・ページの下のボックスにパラメータの説明が表示されます。詳細は、22-7 ページの「Meta Integration Model Bridge（MIMB）との統合」を参照してください。

6. 「次へ」をクリックします。図 22-4 に示すような「Oracle Warehouse Builder 転送ウィザード : サマリー」ウィンドウが表示されます。

図 22-5「Oracle Warehouse Builder 転送ウィザード : サマリー」ウィンドウ



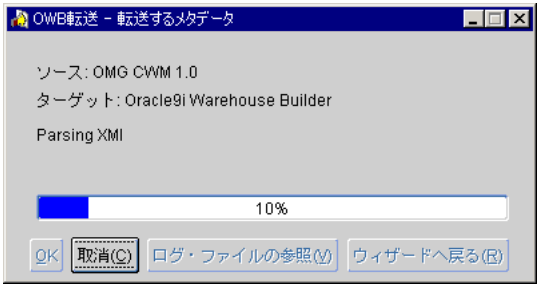
7. 入力内容を確認します。

   入力内容が間違っている場合は、「戻る」をクリックして前の画面に戻り、該当する設定を変更します。

8. 「Oracle Warehouse Builder 転送ウィザード : 確認」ウィンドウで「終了」をクリックします。

   図 22-6 に示すように、転送ウィンドウにステータス・バーが表示されます。

図 22-6 データ転送の進行状況



   転送ウィンドウのタイトルには、入力した説明が表示されます。説明を入力しなかった場合は、タイトルが表示されません。

   必要な転送時間は、転送するメタデータのサイズによって異なります。数分で完了する場合もあれば、1 時間以上かかることもあります。

9. 次のいずれかの操作を行います。

   - 転送が正常に終了した場合（100% と表示された場合）は、「OK」をクリックします。
   - 転送エラー・メッセージが表示された場合は、「ログ・ファイルの表示」をクリックしてログを確認します。（転送に成功した場合もログを表示できます）。

     後で参照できるようにログを保存するには、「別名で保存」をクリックして「保存」ダイアログを表示します。ログを保存するフォルダを選択し、「保存」をクリックします。
   - 情報ログでエラーの原因を確認した場合は、更新の必要な情報を書き留めておきます。「OK」をクリックしてログを閉じます。転送ウィンドウで「ウィザードへ戻る」をクリックし、「転送パラメータ」ウィンドウで不正な情報を更新します。データを再転送します。
   - 情報ログでエラーの原因を確認できない場合は、トレース・ログを作成します。「OK」をクリックして、現在のログを閉じます。転送ウィンドウで「ウィザードへ戻る」をクリックし、「転送パラメータ」ウィンドウで「ログ・レベル」を「トレース」に変更します。データを再転送します。

   転送に成功した場合は、出力ファイルが作成され、指定したフォルダに保存されます。

Warehouse Builder からのメタデータのエクスポート

Warehouse Builder 転送ウィザードでは、次のタイプのターゲットにメタデータをエクスポートできます。

- OMG CWM 標準に準拠したファイル
- Oracle Discoverer 4*i* および Oracle9*i*AS Discoverer
- Oracle Express
- Oracle OLAP

これらのターゲットにメタデータをエクスポートするには、まずは Warehouse Builder にコレクションを定義する必要があります。22-2 ページの「コレクションの定義」を参照してください。

Warehouse Builder 転送ウィザードを使用してメタデータをエクスポートする手順は次のとおりです。

1. 「プロジェクト」メニューから「メタデータのエクスポート」を選択し、「ブリッジ」を選択します。

   「Oracle Warehouse Builder 転送ウィザード : ようこそ」ウィンドウが表示され、この転送ウィザードで実行する手順が示されます。

2. 「次へ」をクリックします。

   図 22-7 に示すような「Oracle Warehouse Builder 転送ウィザード : メタデータ・ソースとターゲットの識別」ウィンドウが表示されます。

3. 「ターゲット」フィールドで、データのエクスポート先となるターゲット（OMG、Oracle Express、Discoverer、OLAP Server）を選択します。

4. 必要に応じて、転送するメタデータの説明を「説明」フィールドに入力します。

   この説明は、転送プロセスの実行中に進行状況バーに表示されます。

図 22-7「Oracle Warehouse Builder 転送ウィザード : メタデータ・ソースとターゲットの識別」ウィンドウ

5. 「次へ」をクリックします。

   「Oracle Warehouse Builder 転送ウィザード : 転送パラメータの識別」ウィンドウが表示されます。

   「転送パラメータ」ウィンドウの表には、入力または選択の必要なパラメータのリストが表示されます。このウィンドウに表示されるパラメータは、前の手順で選択したターゲットによって異なります。詳細は、22-40 ページの「転送パラメータ」を参照してください。

6. 「次へ」をクリックします。

   「OWB 転送の確認」ウィンドウが表示されます。

7. 入力内容を確認します。

   入力内容が間違っている場合は、「戻る」をクリックして前の画面に戻り、該当する設定を変更します。

8. 「終了」をクリックします。

   転送ウィンドウにステータス・バーが表示されます。

   転送ウィンドウのタイトルには、「メタデータ・ソースとターゲットの識別」ウィンドウで入力した転送の説明が表示されます。説明を入力しなかった場合は、タイトルが表示されません。

   必要な転送時間は、転送するメタデータのサイズによって異なります。数分で完了する場合もあれば、1 時間以上かかることもあります。

9. 次のいずれかの操作を行います。

   - 転送が正常に終了した場合は、「OK」をクリックします。
   - 転送エラー・メッセージが表示された場合は、「ログ・ファイルの表示」をクリックしてログを確認します。転送に成功した場合もログを表示できます。

     後で参照できるようにログを保存するには、「別名で保存」をクリックして「保存」ダイアログを表示します。ログを保存するフォルダを選択し、「保存」をクリックします。
   - 情報ログでエラーの原因を確認した場合は、更新の必要なデータを書き留めておきます。「OK」をクリックしてログを閉じます。転送ウィンドウで「ウィザードへ戻る」をクリックし、「転送パラメータ」ウィンドウで不正なデータを更新します。データを再転送します。
   - 情報ログでエラーの原因を確認できない場合は、トレース・ログを作成します。「OK」をクリックして、現在のログを閉じます。転送ウィンドウで「ウィザードへ戻る」をクリックし、「転送パラメータ」ウィンドウで「ログ・レベル」を「トレース」に変更します。データを再転送します。

転送に成功した場合は出力ファイルが作成され、指定したフォルダに保存されます。

Warehouse Builder でのオンライン分析処理（OLAP）

通常、ビジネス組織では、分析、予測およびプラニングの要件が複雑です。分析用のビジネス・インテリジェンス（BI）・アプリケーションは、データベースにあるデータを使用して重要なビジネス上の問題に回答することで、ソリューションを提供します。これらのアプリケーションで実行される分析クエリには、時系列分析、行間計算、集計済の履歴データと現行データへのアクセス、予測計算などが含まれ、SQL クエリでは実行できないものもあります。

Oracle Database の OLAP オプションを使用すると、ビジネス・インテリジェンス・アプリケーションをサポートする同一データベース・システムで、すべての分析タスクが実行可能になります。詳細は、『Oracle OLAP アプリケーション開発者ガイド』を参照してください。

Warehouse Builder は、他のデータ・ソースから Oracle Database に、多次元 OLAP オブジェクトを設計、配布およびロードできる唯一のツールです。データをロードした後は、ツールとアプリケーションを使用して、複雑な分析クエリを実行し、ビジネス上の質問の回答を得ることができます。

Warehouse Builder では、同じメタデータから、リレーショナル用と分析用両方のデータ・ストアを、作成および管理できるようになりました。

アナリティック・ワークスペースについて

アナリティック・ワークスペース（AW）は、Oracle データベース内のコンテナで、多次元データ・オブジェクトと、OLAP DML で記述されるプロシージャが格納されます。AW は分析処理の中枢です。1 つのデータベースに複数のアナリティック・ワークスペースを作成できます。それらはリレーショナル・スキーマによって所有され、複数のユーザーで共有できます。Warehouse Builder を使用して、別の AW、フラット・ファイルまたはリレーショナル表にあるデータを、特定の AW に移行できます。

Warehouse Builder での OLAP の有効化

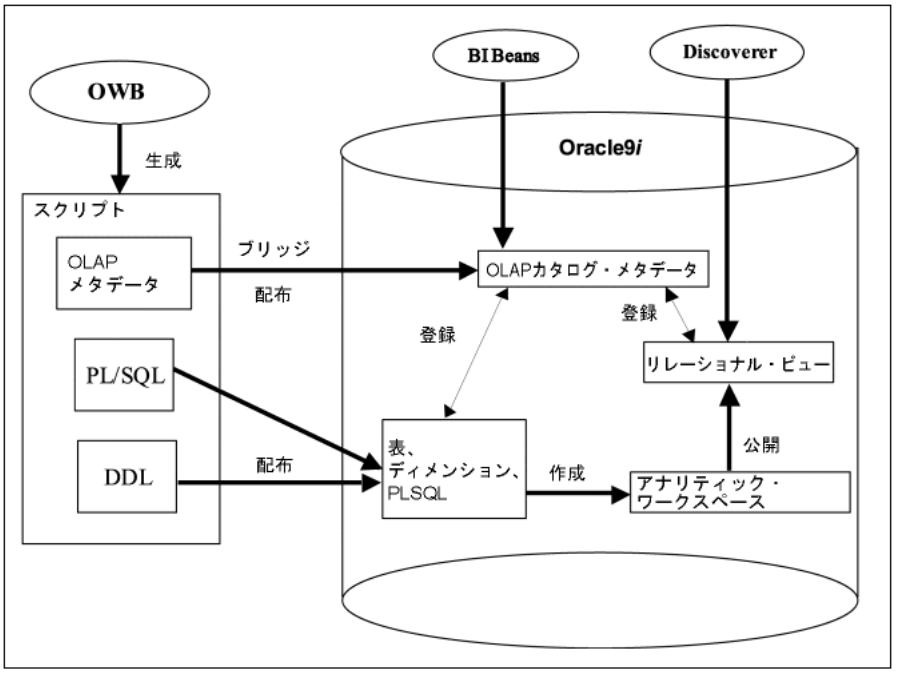
Warehouse Builder では、Oracle Database の OLAP 処理に使用するデータ・ストアを準備できます。Warehouse Builder を使用して、データ・ウェアハウスに対して複雑な分析を可能にするオンライン分析処理（OLAP）オブジェクトを設計、配布およびロードできます。

Warehouse Builder を使用せずに分析クエリを実行するには、MOLAP データベースで次のタスクを手動で実行する必要があります。

1. PL/SQL コマンドまたは OLAP DDL コマンドを使用して、アナリティック・ワークスペースにオブジェクトを定義します。
2. アナリティック・ワークスペースのロードを定義します。
3. 依存関係用のスクリプトを作成します。
4. リレーショナル・ビューに OLAP メタデータをマッピングします。
5. マテリアライズド・ビューを作成します。

Warehouse Builder では、これらのタスクをすべて実行して、リレーショナル・スター・スキーマから、図 22-8 に示すような OLAP 環境を作成できます。最初に、Warehouse Builder 設計環境を使用して、OLAP メタデータを作成します。データベースに AW を作成し分析処理を可能にするために必要な、ディメンションやキューブなどの多次元オブジェクトを定義できます。

図 22-8 Warehouse Builder の使用による ROLAP 環境の作成



Warehouse Builder Transfer Bridge を使用して、データベースにこのメタデータを配布し、AW、AW 上のリレーショナル・ビューおよびリレーショナル・ビューに基づく OLAP メタデータを作成します。次に、Warehouse Builder でマッピングまたはプロセス・フローを定義し、各種データ・ソースから AW にデータをロードします。AW に OLAP データをロードした後は、Oracle Discoverer や BI Beans などの BI ツールを使用して、データに対して複雑な分析クエリを実行できます。

この項では、リレーショナル・オンライン分析処理（ROLAP）環境を設計および作成する次のタスクを、Warehouse Builder で実行する方法について説明します。

- リレーショナル・ウェアハウスの作成
- OLAP メタデータの作成
- OLAP メタデータの配布
- アナリティック・ワークスペースへのデータのロード

OLAP メタデータの作成

Warehouse Builder のウィザードとエディタを使用して、必要なディメンション、階層、属性およびキューブを定義できます。ディメンションおよびキューブ作成の詳細は、3-58 ページの「ディメンション定義の作成」および 3-67 ページの「キューブ定義の作成」を参照してください。OLAP 準拠のメタデータを作成するには、この後の項で説明する各ガイドラインに従ってください。

Warehouse Builder での OLAP ディメンションの定義

ディメンションを作成するときは、ディメンション・レベル属性に対する実装列を定義する際に、表 22-1 に示す拡張語を使用してください。表 22-1 に、Warehouse Builder 転送ウィザードを使用して、Oracle OLAP カタログにディメンションを配布するときに、OLAP 準拠レベルの記述子として生成されるレベル属性名の一覧を示します。

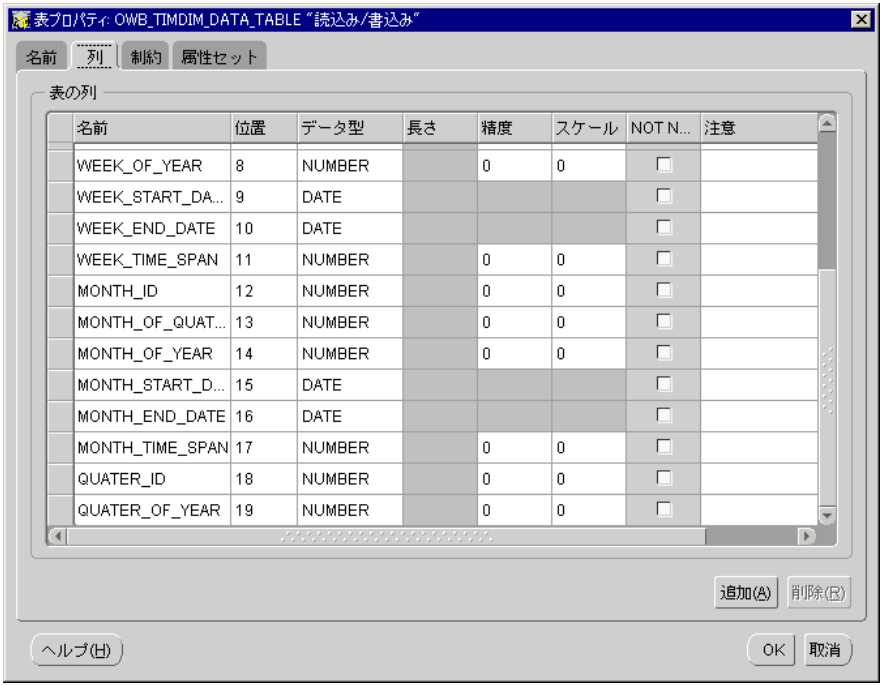
レベルの作成時に ID という名前の Warehouse Builder レベル属性が生成されますが、これはディメンション属性には使用しないでください。データベースのディメンション・レベル属性は ID 列には作成されないため、このレベル属性はこの命名方法に従いません。

表 22-1 ディメンション属性の接尾辞

Warehouse Builder での物理レベル属性名の接尾辞	作成されるディメンション属性
_NAME または NAME	Short_Description または Long_Description
_END_DATE または END_DATE	End_Date
_TIME_SPAN または TIME_SPAN	Time_Span
_PRIOR_PERIOD または PRIOR_PERIOD	Prior_Period
_YEAR_AGO_PERIOD または YEAR_AGO_PERIOD	Year_Ago_Period

Warehouse Builder では、図 22-9 に示すように、指定された拡張語が使用されている実装列を含むディメンション・レベル属性ごとに、ディメンション属性が作成されます。たとえば、レベルごとに終了日が指定されている場合は、各レベルにディメンション属性も作成されます。会計年度の終了日とカレンダーの終了日がレベルごとに指定されている場合は、レベル属性名を使用して 2 つのディメンション属性が作成されます。Warehouse Builder でのディメンション作成の詳細は、3-58 ページの「ディメンション定義の作成」を参照してください。

図 22-9 列へのディメンション属性の実装

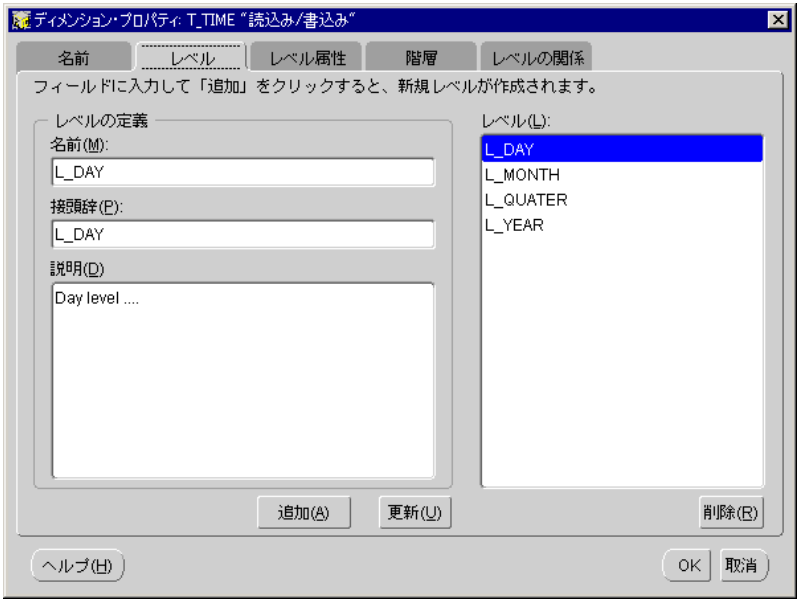


時間ディメンション Warehouse Builder には、時間ディメンションの例が、Metadata Loader（.MDL）ファイルの形式で用意されています。*OWB_Home_Directory*¥owb¥misc¥time ディレクトリに移動します。このディレクトリにある readme.txt ファイルに、事前パッケージされた .MDL ファイルおよび .SQL ファイルの使用方法が説明されています。

Warehouse Builder で時間ディメンションを作成するには、次のガイドラインに従ってください。

- ディメンション名には、T_TIME のように、大文字の拡張語 _TIME を使用します。これにより、Transfer Bridge で、そのディメンションに種類が時間ディメンションであることを示す記述子が作成されます。図 22-10 に、このルールを使用して作成した時間ディメンションの例を示します。

図 22-10 時間ディメンションの作成例



- 時間ディメンションのレベルには、表 22-2 に示す拡張語を使用します。

  週レベルの接尾辞（_WEEK）が表にない点に注意してください。週は、カレンダー上の月、四半期、年に対して正しくロールアップできません。複数の期間にわたる場合があるためです。時間ディメンションで、カレンダー期間のかわりに会計年度期間を使用すると、週は他の期間に正しくロールアップされ、週レベルでの作業が可能になります。

表 22-2 時間ディメンション属性の接尾辞

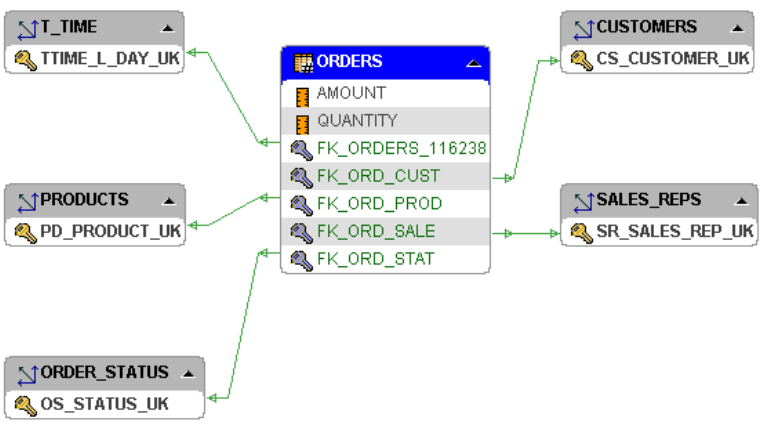
Warehouse Builder での物理レベル名	Oracle Database で作成される時間ディメンション・レベル
_DAY	日レベル型
_MONTH	月レベル型
_QUARTER	四半期レベル型
_YEAR	年レベル型

Warehouse Builder でのキューブの定義

キューブは、OLAP 処理の主要なオブジェクトで、整理された直観的な形式でデータを表現するものです。Warehouse Builder では、新規キューブ・ウィザードを使用してキューブを定義できます。キューブ作成の詳細は、3-67 ページの「キューブ定義の作成」を参照してください。

図 22-11 に、Warehouse Builder で作成したキューブのサンプルを示します。

図 22-11 Warehouse Builder で作成したキューブ



Warehouse Builder でのコレクションの定義

コレクションは、Warehouse Builder の領域で、OLAP データベースなどに配布するメタデータはここに格納されます。Warehouse Builder 転送ウィザードを使用して、コレクションを配布し、OLAP カタログの移入と AW の作成に使用する OLAP メタデータを作成できます。

最初に、Warehouse Builder で、OLAP 対応のディメンションとキューブをすべて含むコレクションを定義する必要があります。

Warehouse Builder 転送ウィザードを使用した OLAP メタデータの配布

メタデータの定義が終了したら、作成した定義を Oracle OLAP データベースに配布できます。OLAP 環境を作成するには、Warehouse Builder で次の手順を実行します。

DDL スクリプトの生成と配布

OLAP データベースにリレーショナル・ストレージ・オブジェクト（ディメンション、表およびキューブ）を作成する DDL スクリプトを生成および配布します。マッピングまたはプロセス・フローを後で実行して、これらの構造にデータをロードできます。「アナリティック・ワークスペースへのロード」を参照してください。生成と配布の詳細は、第 13 章「ターゲット・システムの配布」を参照してください。

Warehouse Builder 転送ウィザードを使用したメタデータの配布

Warehouse Builder 転送ウィザードを実行して、Oracle OLAP Server にメタデータを配布します。転送ウィザードでは、OLAP カタログを移入し、その中にアナリティック・ワークスペース・オブジェクトを作成し、さらにアナリティック・ワークスペース上にリレーショナル・ビューを作成します。また、必要に応じて、マテリアライズド・ビューも作成します。

OLAP メタデータを配布する手順は次のとおりです。

1. 「プロジェクト」メニューから「メタデータのエクスポート」→「ブリッジ」を選択して、Warehouse Builder 転送ウィザードを起動します。

   「Oracle Warehouse Builder 転送ウィザード : ようこそ」ページが表示されます。

2. 「次へ」をクリックします。

3. 「Oracle Warehouse Builder 転送ウィザード : メタデータ・ソースとターゲットの識別」ページで、次の情報を入力します。

   ソース : Oracle Warehouse Builder エクスポート

   ターゲット : Oracle OLAP

   説明 : オプション

4. 「次へ」をクリックします。

5. 「Oracle Warehouse Builder 転送ウィザード : 転送パラメータの識別」ページで、次の情報を入力します。

   エクスポートされた OWB のコレクション : エクスポートする多次元メタデータを含むコレクションを指定します。

   AW に配布 : アナリティック・ワークスペースにメタデータを配布するかどうかを指定します。

   AW 名 : メタデータの配布先となるアナリティック・ワークスペースの名前を指定します。

   AW オブジェクト接頭辞 : AW でのディメンションとキューブの接頭辞。

   キューブ・データをロード : AW にキューブのデータをロードします。このオプションを指定しない場合、プログラムをロードするキューブのみロードされます。

   ディメンションのサロゲート・キーを生成 : ディメンション値が全レベル間で一意でない場合は、ディメンションのサロゲート・キーを生成します。このオプションは、サロゲート・キーの生成に使用できます。

   ビュー定義の生成 : アナリティック・ワークスペースにビュー定義を生成するかどうかを指定します。このオプションを選択すると、SQL ビューに基づいて、アナリティック・ワークスペースにキューブの OLAP メタデータを生成することもできます。

   マテリアライズド・ビューの生成 : リレーショナル・キューブに、マテリアライズド・ビューを生成するかどうかを指示します。

   生成済ビュー・ディレクトリ : 生成されたビュー・スクリプトを格納するディレクトリの名前を指定します。データベース・オプションに「PL/SQL を展開」を選択すると、このディレクトリはクライアント側に置かれます。これ以外のオプションを選択すると、これはサーバー側のディレクトリ・オブジェクトとなります。

   データベース中に PL/SQL を展開 : OLAP データベースに PL/SQL を配布するかどうかを指定します。このオプションを選択すると、OLAP メタデータが配布されます。また、マテリアライズド・ビューおよびアナリティック・ワークスペースのビューの配布も選択できます。

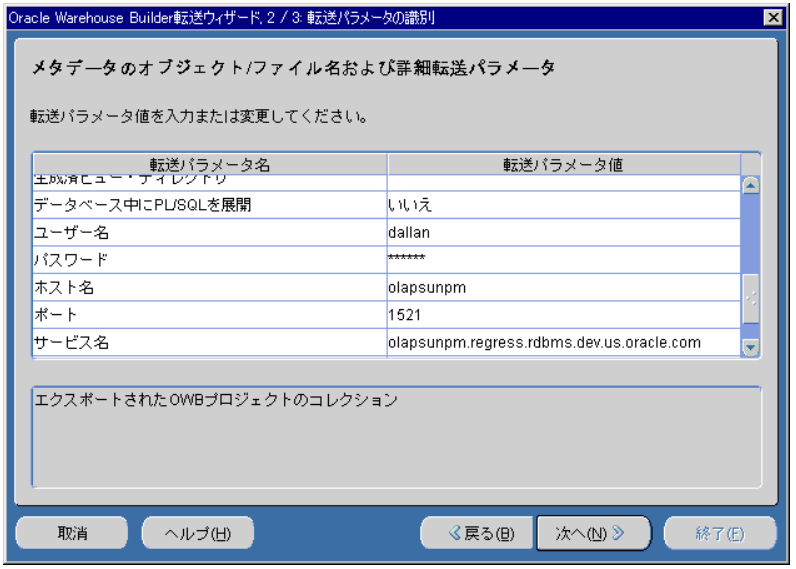
   OLAP インスタンス情報 : OLAP ターゲット・インスタンスへのアクセスに使用するホスト名、ポート、サービス名、ユーザー名およびパスワードを指定します。

   PL/SQL 出力ファイル : PL/SQL 出力ファイルのロケーションを指定します。

   ログ・レベル : 転送の完了後にログ・ファイルから取得する情報の詳細レベルを指定します。

   図 22-12 に、「Oracle Warehouse Builder 転送ウィザード : ステップ 2/3」を示します。

図 22-12 Warehouse Builder 転送ウィザード : ステップ 2/3



6. 「次へ」をクリックします。

   「Oracle Warehouse Builder 転送ウィザード : サマリー」ページが表示されます。

7. 前の手順で指定した情報を確認します。「終了」をクリックして配布を開始します。

   Transfer Bridge により、次のことを実行するスクリプトが生成されます。

   - コレクションにあるキューブの OLAP メタデータを作成する。
   - オプションとして、ROLAP メタデータに基づいて AW にメタデータを作成する。
   - オプションとして、AW に基づいて SQL ビューと OLAP メタデータを作成するスクリプトを生成する。これらのビューの詳細は、『Oracle OLAP アプリケーション開発者ガイド』を参照してください。
   - BI Beans および OLAP API のパフォーマンスを向上させるために、オプションとして、リレーショナル・データに基づいてマテリアライズド・ビューを作成するスクリプトを生成する。これらのビューの詳細は、『Oracle OLAP アプリケーション開発者ガイド』を参照してください。

OLAP へのエクスポートのデバッグ

ブリッジに「データベース中に PL/SQL を展開」オプションを使用すると、ログ・ファイルに診断情報が記録されます。生成された PL/SQL を手動で実行する場合は、次の方法がデバッグに役立ちます。

OLAP エクスポートをデバッグするには、生成された PL/SQL コードをファイルに保存して、SQL*Plus で実行します。さらに詳細なエラー情報を取得するには、SQL*Plus でサーバー出力をオンに切り替えます。たとえば、SQL*Plus でスクリプトを実行する前に、次のコードを実行する必要があります。

```
SET SERVEROUTPUT ON SIZE 99999
```

エラーが発生したときは、詳細エラー・メッセージが表示され、エラーのタイプが説明されます。OLAP Bridge でよくあるエラーは、ディメンションなどの依存オブジェクトを、サーバーのリレーショナル要素に配布する際の失敗です。たとえば、リレーショナル・ディメンションを配布する前にブリッジを実行すると、表 22-3 に示すようなエラーが表示されます。

表 22-3 ディメンションの配布に失敗したときのエラー・メッセージ

エラーの詳細	説明
ERROR: dimension_not_found	OLAP で発生したエラーのタイプ
Object Type: DIMENSION	該当するオブジェクト・タイプ
Object Owner: RTS60	スキーマ名
Object Name: DD	オブジェクト名
Secondary Name	セカンド・クラスのオブジェクト名（列名）
Tertiary Name	サード・クラスのオブジェクト名（レベル属性名）

アナリティック・ワークスペースへのロード

OLAP データベース・システムにメタデータを配布し終わったら、次のどちらかの方法を使用して、アナリティック・ワークスペースにリレーショナル・ソース・データをロードできます。

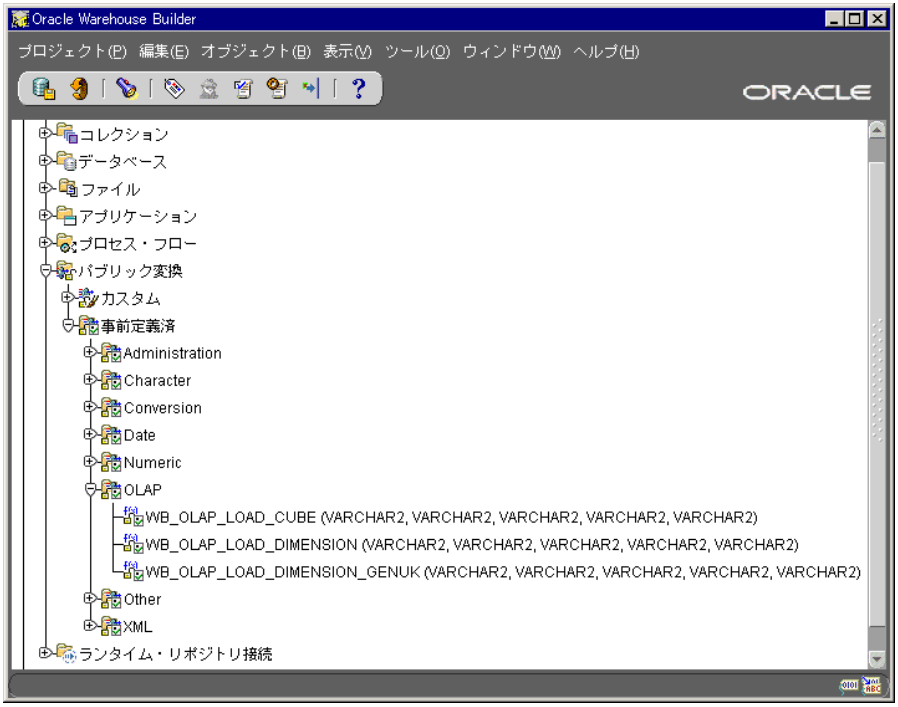
- Warehouse Builder に付属する PL/SQL ルーチンを使用するマッピング後プロセスでマッピングを設計する。
- Warehouse Builder に付属する PL/SQL ルーチンを含むプロセス・フローを設計する。

Warehouse Builder には PL/SQL パッケージが用意されており、これを使用すると、リレーショナル・データを作成し、アナリティック・ワークスペースにロード（またはリフレッシュ）できます。このパッケージは、Oracle Database に付属する DBMS-OLAP プロシージャのラッパーを形成します。詳細は、『Oracle OLAP アプリケーション開発者ガイド』にある DBMS_AWM ルーチンの説明を参照してください。このパッケージには、増分のロードおよびカスタマイズした集計を実行するルーチンがあります。

Warehouse Builder の中で PL/SQL パッケージを見つける手順は次のとおりです。

1. Warehouse Builder のツリーで、「パブリック変換」ノードを開きます。
2. 図 22-13 に示すように、「事前定義済」ノード、「OLAP」ノードの順に開きます。

図 22-13 Warehouse Builder の PL/SQL パッケージ



ラッパー・パッケージの WB_OLAP_LOAD_CUBE、WB_OLAP_LOAD_DIMENSION および WB_OLAP_LOAD_DIMENSION_GENUK が表示されます。

アナリティック・ワークスペースへロードするマッピングの作成

複数のディメンション表を結合および変換し、それらをディメンションまたはキューブにロードするマッピングを作成します。アナリティック・ワークスペースにデータをロードするには、ロード値を含む定数とともにマッピング後プロセスを定義する必要があります。

アナリティック・ワークスペースを作成しデータをロードするマッピング後プロセスをマッピングに追加する手順は次のとおりです。

1. ツールボックスからマッピング・エディタのキャンバスに、マッピング後プロセス演算子をドラッグ・アンド・ドロップします。

   「マッピング変換の追加」ダイアログが表示されます。

2. 「既存リポジトリ変換からの選択およびバインド」を選択します。
3. Warehouse Builder のナビゲーション・ツリーから OLAP パッケージを検索します。パッケージ検索の詳細は、前の項で説明した手順を参照してください。
4. 「OK」をクリックします。
5. マッピング後プロセスの変換がキャンバスに表示されます。

   WB_OLAP_LOAD_CUBE プロシージャを選択した場合は、次の属性が表示されます。

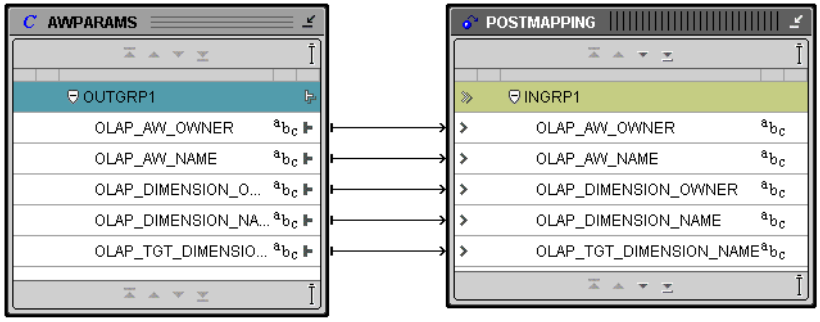
   - **OLAP_AW_OWNER:** 作成または保持されるアナリティック・ワークスペースの所有者。
   - **OLAP_AW_NAME:** 作成または保持されるアナリティック・ワークスペースの名前。
   - **OLAP_CUBE_OWNER:** キューブの所有者。
   - **OLAP_CUBE_NAME:** キューブの名前。
   - **OLAP_TGT_CUBE_NAME:** ターゲットでのキューブの名前。

   WB_OLAP_LOAD_DIMENSION プロシージャまたは WB_OLAP_LOAD_DIMENSION_GENUK プロシージャを選択した場合は、次の属性が表示されます。

   - **OLAP_AW_OWNER:** 作成または保持されるアナリティック・ワークスペースの所有者。
   - **OLAP_AW_NAME:** 作成または保持されるアナリティック・ワークスペースの名前。
   - **OLAP_DIMENSION_OWNER:** ディメンションの所有者。
   - **OLAP_DIMENSION_NAME:** ディメンションの名前。
   - **OLAP_TGT_DIMENSION_NAME:** ターゲット・システムでのディメンションの名前。

6. ツールボックスからマッピング・エディタのキャンバスに、CONSTANT 演算子をドラッグ・アンド・ドロップします。
7. CONSTANT 演算子を右クリックし、ポップアップ・メニューから「編集」を選択します。
8. 定数エディタで「出力」タブを選択します。
9. マッピング後プロセスで定義した属性と同じ属性を定数で定義します。
10. 図 22-14 に示すように、CONSTANT 演算子からマッピング後プロセスの属性に、属性をマッピングします。

図 22-14 AW にロードするマッピング後プロセスの定義

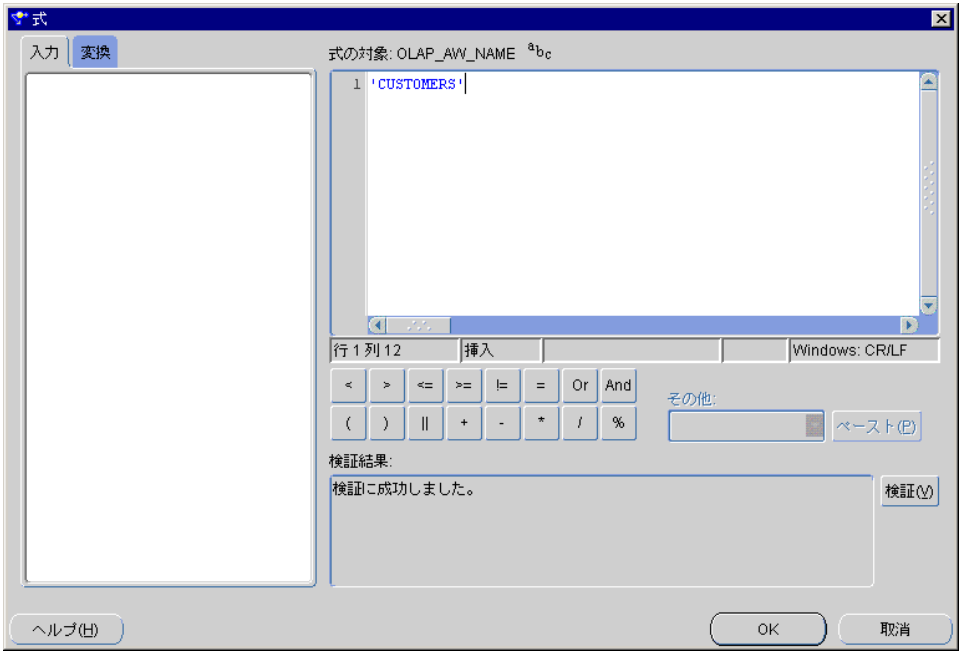


11. CONSTANT 演算子で各属性を右クリックし、「属性のプロパティ」を選択します。

    「属性のプロパティ」ダイアログが表示されます。
12. 「式」フィールドをクリックして、「...」ボタンをクリックします。

    Expression Builder が表示されます。
13. Expression Builder で、属性の値を入力します。たとえば、図 22-15 に示すように、属性 OLAP_DIMENSION_NAME には、配布するディメンションの名前である「CUSTOMERS」を入力します。

図 22-15 Expression Builder



14. CONSTANT 演算子のすべての属性に式を割り当てるまで、この作業を繰り返します。

これで、マッピングを検証して、PL/SQL コードを生成する準備が整いました。PL/SQL を生成した後は、マッピングを実行して OLAP インスタンスにキューブおよびディメンションをロードできます。Warehouse Builder では、選択によっては、完全に解決されたキューブがアナリティック・ワークスペースにロードされます。

アナリティック・ワークスペースにロードするプロセス・フローの作成

AW に OLAP ディメンションおよびキューブをロードするプロセス・フローを作成することもできます。Warehouse Builder に付属する PL/SQL ルーチンを含むプロセス・フローを設計します。

Warehouse Builder の中で PL/SQL パッケージを見つける手順は次のとおりです。

1. プロセス・フローを作成します。詳細は、第 10 章「プロセス・フローの設計」を参照してください。
2. キャンバスに「トランスフォーム」アクティビティをドラッグ・アンド・ドロップします。

   変換を選択するダイアログが表示されます。
3. 「パブリック」ノードを展開します。
4. 「事前定義済」ノード、「OLAP」ノードの順に開きます。

   ラッパー・プロシージャの WB_OLAP_LOAD_CUBE、WB_OLAP_LOAD_DIMENSION および WB_OLAP_LOAD_GENUK が表示されます。
5. これらのパッケージのいずれかを選択します。
6. 「アクティビティ・ビュー」パネルで、ディメンションおよびキューブのパラメータを定義します。

WB_OLAP_LOAD_CUBE パッケージを選択した場合は、次のパラメータが表示されます。それぞれのパラメータについて、「VALUE」列に値を入力します。オプションとして、それぞれのパラメータについて、「DESCRIPTION」列に説明を入力します。

- **OLAP_AW_OWNER:** 作成または保持されるアナリティック・ワークスペースの所有者。
- **OLAP_AW_NAME:** 作成または保持されるアナリティック・ワークスペースの名前。
- **OLAP_CUBE_OWNER:** ROLAP キューブの所有者。
- **OLAP_CUBE_NAME:** ROLAP キューブの名前。
- **OLAP_TGT_CUBE_NAME:** ターゲット・アナリティック・ワークスペースでのキューブの名前。

WB_OLAP_LOAD_DIMENSION プロシージャを選択した場合は、次のパラメータが表示されます。それぞれのパラメータについて、「VALUE」列に値を入力します。オプションとして、それぞれのパラメータについて、「DESCRIPTION」列に説明を入力します。

- **OLAP_AW_OWNER:** 作成または保持されるアナリティック・ワークスペースの所有者。
- **OLAP_AW_NAME:** 作成または保持されるアナリティック・ワークスペースの名前。
- **OLAP_DIMENSION_OWNER:** ROLAP ディメンションの所有者。
- **OLAP_DIMENSION_NAME:** ROLAP ディメンションの名前。
- **OLAP_TGT_DIMENSION_NAME:** ターゲット・アナリティック・ワークスペースでのディメンションの名前。

これで、プロセス・フローを検証して、XPDL コードを生成する準備が整いました。XPDL コードを生成した後は、プロセス・フローを実行して OLAP インスタンスにキューブおよびディメンションをロードできます。Warehouse Builder では、アナリティック・ワークスペースにキューブがロードされ、OLAP ルーチンにより、キューブを解決する動的な集計マップが作成されます。

事前計算済集計または増分リフレッシュを、特定レベルで実行する必要がある場合は、『Oracle OLAP アプリケーション開発者ガイド』にある DBMS_AWM パッケージの操作の詳細を参照してください。

マッピングまたはプロセス・フローでのクローニングのデバッグ

クローニング変換のデバッグに必要な情報にアクセスするには、PL/SQL ラッパーを作成して、必要なロギングをカスタマイズします。

PL/SQL ラッパーを作成するには、Warehouse Builder 付属の WB_OLAP_LOAD_DIMENSION および WB_OLAP_LOAD_CUBE と同じシグネチャを持つパブリック変換を作成します。パブリック変換の実装では、次のような OLAP ルーチンを使用してロギングを有効にできます。

```
CWM2_OLAP_MANAGER.BEGIN_LOG(<directory_alias>, <filename>); CWM2_OLAP_MANAGER.END_
LOG;
```

この変換を使用して、ログ・ファイルを開始および終了できます。このコール中に、ディメンションまたはキューブをロードする適切な Warehouse Builder ルーチンを起動できます。たとえば、ディメンションのロードをトレースするには、ディレクトリのオブジェクトを使用した次のコード・ブロックを含むプロシージャを作成できます。

```
'MY_OLAP_TRACE';
CWM2_OLAP_MANAGER.BEGIN_LOG('MY_OLAP_TRACE', olap_dimension_name || '.log'); OWB_
OLAP_LOAD_DIMENSION(olap_aw_owner, olap_aw_name, olap_dimension_owner, olap_
dimension_name, olap_tgt_dimension_name);
CWM2_OLAP_MANAGER.END_LOG;
```

このコードにより、データ・ロード時の問題のデバッグに役立つレベルのトレース情報を取得できます。

Warehouse Builder OLAP Bridge での注意事項

Warehouse Builder OLAP では、最善の方法として、Warehouse Builder での設計定義、Warehouse Builder OLAP Bridge、およびそれが生成する OLAP メタデータが統合されます。OLAP ディメンションでは、階層の最下位レベルがその説明に使用されるレベル属性を持つことが、最低限要求されます。この要件は、最善の方法を使用することで満たされます。

Warehouse Builder OLAP Bridge の実行モード

Warehouse Builder OLAP Bridge には、標準モードとデバッグ・モードという 2 種類の実行モードがあります。これらのモードは次の項で説明します。

標準モード : ブリッジから PL/SQL を配布する

ブリッジから PL/SQL スクリプトを直接配布する場合は、次の内容が実行されます。

- OLAP カタログに ROLAP カタログ・メタデータをロードする。
- AW に ROLAP キューブをクローニングする（MOLAP ソリューション - オプション）。
- CWM2 にメタデータを作成する OLAP 生成スクリプトを実行し、OLAP API とともに AW に格納されている MOLAP キューブを有効にする（オプション）。
- マテリアライズド・ビューを生成する OLAP 生成済スクリプトを実行し、ROLAP キューブのマテリアライズド・ビューを作成する（オプション）。

ROLAP の配布 AW クローン、OLAP API 生成および MV 生成に対して「いいえ」を選択すると、ROLAP OLAP カタログ・メタデータのみが生成されます。クローニングの前に ROLAP モデルを取得するには、これらのオプションをオフにします。

AW のクローニング AW クローンに「はい」を選択すると、ROLAP キューブの MOLAP クローンが AW に作成されます。

OLAP API の有効化 クライアント上で、ディレクトリ・パラメータに有効なディレクトリを指定します。OLAP API を有効にするスクリプトが、クライアントのこのディレクトリに生成されます。ブリッジによって、このスクリプトが実行されます。スクリプトは、コレクション内のディメンションごとおよびキューブごとに生成されます。このスクリプトでは、生成されるオブジェクトの物理名に接尾辞として .sql が挿入され、クライアント・ファイル・システム上のディレクトリに、このオブジェクトが常駐します。コレクション内の各ディメンションおよびキューブには削除用のスクリプトも生成され、その物理名には接尾辞として _drop.sql が挿入されます。このスクリプトもクライアント・ファイル・システム上のディレクトリに常駐します。これは、SQL オブジェクト・タイプとビューの削除に使用できます。

マテリアライズド・ビューの生成 クライアントの有効なディレクトリを、ディレクトリ・パラメータに指定します。MV スクリプトが、クライアントのこのディレクトリに生成されます。ブリッジによって、このスクリプトが実行されます。スクリプトは、コレクション内のディメンションごとおよびキューブごとに生成されます。このスクリプトでは、生成されるオブジェクトの物理名に接尾辞として _mv.sql が挿入され、クライアント・ファイル・システム上のディレクトリに配置されます。

デバッグ・モード : 手動で PL/SQL を配布する

生成された PL/SQL スクリプトを SQL*Plus から実行すると、次の内容が実行されます。

- OLAP カタログに ROLAP カタログ・メタデータをロードする。
- AW に ROLAP キューブをクローニングする（MOLAP ソリューション - オプション）。
- OLAP API とともに AW に保存されている MOLAP キューブを有効にするスクリプトを生成する（オプション）。
- ROLAP キューブのマテリアライズド・ビューを定義するスクリプトを生成する（オプション）。

ROLAP の配布 AW クローン、OLAP API 生成および MV 生成に対して「いいえ」を選択すると、ROLAP OLAP カタログ・メタデータのみが生成されます。この手順は、AW クローニングの前に完了する必要があります。クローニングの前に正しい ROLAP モデルを取得するには、これらのオプションをオフにします。

AW のクローニング AW クローンに「はい」を選択すると、ROLAP キューブの MOLAP クローンが AW に作成されます。

OLAP API の有効化 ディレクトリ・パラメータに有効なディレクトリ・オブジェクトを指定します。OLAP API を有効にするスクリプトが、サーバーのこのディレクトリ・オブジェクトに生成されます。このスクリプトを実行するのはユーザーです。コレクション内の各ディメンションおよびキューブにスクリプトが生成され、その物理名には接尾辞として .sql が挿入されます。生成されたスクリプトは、サーバー・ファイル・システム上のディレクトリ・オブジェクトのロケーションに格納されます。コレクション内のディメンションごとおよびキューブごとに、削除用のスクリプトも生成されます。この削除スクリプトでは、生成されるオブジェクトの物理名に接尾辞として _drop.sql が挿入され、サーバー・ファイル・システム上のディレクトリ・オブジェクトのロケーションに格納されます。これは、SQL オブジェクト・タイプとビューの削除に使用できます。

マテリアライズド・ビューの生成 ユーザーは、ディレクトリ・パラメータに有効なディレクトリ・オブジェクトを指定する必要があります。MV スクリプトが、サーバーのこのディレクトリ・オブジェクトに生成されます。このスクリプトを実行するのはユーザーです。コレクション内の各ディメンションおよびキューブにスクリプトが生成され、その物理名には接尾辞として _mv.sql が挿入され、サーバー・ファイル・システム上のディレクトリ・オブジェクトのロケーションに配置されます。

ブリッジのカスタマイズ

AW のクローニングに応じて、ブリッジをカスタマイズできます。カスタマイズは、Warehouse Builder で各キューブの説明内に特殊なタグを使用することで、キューブ単位で実行できます。こられのタグとその内容は、OLAP カタログの最終的なキューブ説明から削除されます。事前ロード・ブロックは、AW でのディメンションのセグ幅の定義または複合ディメンションの定義に使用できます。この構造を使用すると、AW からのパフォーマンスが向上します。事後ロード・ブロックは、AW での計算済メジャーの定義に使用できます。カスタマイズに使用できるタグのセットは次の 2 つです。

```
<PRELOAD>
anonymous_plsql_block
</PRELOAD>
```

および

```
<POSTLOAD>
anonymous_plsql_block
</POSTLOAD>
```

ブリッジにより、キューブの名前とともに OLAP ロード仕様が生成されます。事前ロード・ブロックを使用すると、OLAP API により、このロード仕様を増やし、複合仕様を定義できます。この複合仕様はさらに、元のロード仕様に追加できます。

次は、新しい複合仕様を定義する事前ロード・ブロックの例です。この例では、セグ幅が 555555 のディメンション CUSTOMERS と PRODUCTS を含む CMP1 という名前の複合ディメンションが、AW に作成されます。

```
<PRELOAD>
declare
  cnam varchar2(32);
  cub varchar2(30);
  cst varchar2(30);
BEGIN
   cnam := 'ORDERS_CST';
   cub := 'ORDERS';
   cst := 'CMP1';
   begin
     dbms_awm.delete_awcomp_spec(cnam, USER, cub);
   EXCEPTION when others then null;
   end;

   dbms_awm.create_awcomp_spec(cnam, USER, cub);
   dbms_awm.add_awcomp_spec_member(cnam, USER, cub, 'T_TIME', 'DIMENSION',
USER, 'T_TIME');
   dbms_awm.set_awcomp_spec_member_seg(cnam, USER, cub, 'T_TIME', 7777777);
   dbms_awm.add_awcomp_spec_member(cnam, USER, cub, cst, 'COMPOSITE');
   dbms_awm.add_awcomp_spec_comp_member(cnam, USER, cub, cst, cst||'_
CUSTOMERS',
'DIMENSION', USER, 'CUSTOMERS');
   dbms_awm.add_awcomp_spec_comp_member(cnam, USER, cub, cst, cst||'_PRODUCTS',
'DIMENSION',
USER, 'PRODUCTS');
   dbms_awm.set_awcomp_spec_member_seg(cnam, USER, cub, cst, 555555);
   dbms_awm.add_awcubeload_spec_comp(cub, USER, cub, cnam);
   EXCEPTION
     WHEN OTHERS THEN raise;
END;
</PRELOAD>
```

このコードにより、複合仕様「ORDERS_CST」が、ブリッジにより作成されたロード仕様「ORDERS」に追加されます。このコードは、OLAP リフレッシュ・キューブ・ルーチンの前に実行されます。OLAP API とその機能の詳細は、『Oracle OLAP アプリケーション開発者ガイド』を参照してください。

事後ロード・ブロックは、AW にキューブが作成された後に実行されます。事後ロード・ブロックは、ブリッジの実行時に作成されたオブジェクトに基づいて、AW での新しい計算済メジャーの定義に使用できます。AW は、ブリッジの実行が終了した時点で更新およびコミットされます。

クローニング動作（増分のロード、事前計算済集計）

Warehouse Builder で使用される OLAP クローニング・ルーチンと変換として使用されるルーチンは、AW での ROLAP キューブのクローニングとそのデータのリフレッシュに使用されるベース・ルーチンです。これらのルーチンで定義されているリフレッシュは、ディメンションまたはファクト表の完全ロードです。次の項では、増分のロードおよび事前計算済集計を実行する方法について説明します。

増分のロード

Warehouse Builder OLAP 統合におけるクローニングおよびロード動作のデフォルトは、全データ・セットのロードです。AW で増分のロードを実行するには、OLAP ルーチンを使用して、ディメンションまたはキューブから実際にロードする行を絞り込むフィルタ（SQL WHERE 句）を定義する必要があります。変換 ORDERS_LOAD_FILTER では、OLAP フィルタ・ルーチンを使用して、ROLAP ファクト表から選択した行セット（次の例では month_id > 33）のみをロードできます。次に、OLAP API の使用例を示します。

```
procedure ORDERS_LOAD_FILTER
BEGIN
 dbms_awm.create_awcubeload_spec ('ORDERS_FIL', USER, 'ORDERS', 'LOAD_DATA');

 -- Define the where clause for the load specification
 dbms_awm.Add_AWCubeLoad_Spec_Filter('ORDERS_FIL',USER,'ORDERS',USER,'ORDERS','
month_id>33');
 dbms_awm.refresh_awcube (USER, 'AWS', 'AWORDERS', 'ORDERS_FIL');

    EXCEPTION
        WHEN OTHERS THEN
            NULL;  -- enter any application specific exception code here
END;
```

事前計算済集計

デフォルトでは、クローニング時に Warehouse Builder OLAP Bridge によって、AW にランタイム集計マップ（AGGMAP）が作成され、キューブの完全な解決に使用されます。カスタマイズした事前計算済集計を実行してキューブを最適化する場合は、OLAP API を使用して、こうした定義を構築できます。カスタム集計は、プロセス・フローのアクティビティとして実行できます。

次の例では、変換 ORDERS_BUILD で OLAP ルーチンを使用して、AW 内のキューブの特定レベルに対して事前計算しています。次は、OLAP API の使用方法の例です。

```
procedure ORDERS_BUILD
  aggspec VARCHAR2(32);
  AW_NAME VARCHAR2(32);
  CUBE_NAME VARCHAR2(32);
BEGIN
  aggspec := 'AGGORDERS';
  AW_NAME := 'TESTAW';
  CUBE_NAME := 'AWORDERS';

  dbms_awm.create_awcubeagg_spec(aggspec, USER, AW_NAME, CUBE_NAME);

  -- Only aggregate city and customer for the CUSTOMERS dimension
  dbms_awm.add_awcubeagg_spec_level(aggspec, USER, AW_NAME, CUBE_NAME,
'AWCUSTOMERS', 'CITY');
  dbms_awm.add_awcubeagg_spec_level(aggspec, USER, AW_NAME, CUBE_NAME,
'AWCUSTOMERS',
'CUSTOMER');

  -- Only aggregate day and month for the time dimension
  dbms_awm.add_awcubeagg_spec_level(aggspec, USER, AW_NAME, CUBE_NAME, 'AWT_TIME',
'DAY');
  dbms_awm.add_awcubeagg_spec_level(aggspec, USER, AW_NAME, CUBE_NAME, 'AWT_TIME',
'MONTH');

  dbms_awm.add_awcubeagg_spec_measure(aggspec, USER, AW_NAME, CUBE_NAME, 'AMOUNT');

  -- Now build the cube. This may take some time on large cubes.
  dbms_awm.aggregate_awcube(USER, AW_NAME, CUBE_NAME, aggspec);

   EXCEPTION
        WHEN OTHERS THEN
            NULL;  -- enter any application specific exception code here
END;
```

Warehouse Builder ブリッジ : 転送パラメータおよび転送に関する考慮事項

この項では、次のトピックについて説明します。

転送パラメータ

Warehouse Builder 転送ウィザードでは、Object Management Group の Common Warehouse Metamodel、Computer Associates の ERwin、Powersoft の PowerDesigner、Oracle9*i* OLAP Server などのデータ・ウェアハウス・ツールやファイル・システムから、Warehouse Builder にメタデータをインポートできます。また、Warehouse Builder から Oracle Discoverer、Oracle Express、Oracle9*i* OLAP Server などの Oracle ツールに、メタデータをエクスポートすることもできます。

Warehouse Builder 転送ウィザードを使用する際には、選択したメタデータのソースやターゲットに基づいて転送パラメータの情報を入力する必要があります。表 22-4 に、それぞれのデータ・ソース・タイプまたはデータ・ターゲット・タイプで指定する転送パラメータを列挙示します。

表 22-4 OLAP Server インポートに指定する転送パラメータ

転送パラメータ名	説明
ユーザー名	メタデータのインポート元から OLAP Server インスタンスへのアクセスに使用するユーザー名。
パスワード	メタデータのインポート元から OLAP Server インスタンスへのアクセスに使用するパスワード。
ホスト名	メタデータのインポート元からアクセスする OLAP Server インスタンスのホスト・システム。
ポート	メタデータのインポート元から OLAP Server インスタンスへのアクセスに使用するポート番号。
サービス名	メタデータのインポート元から OLAP Server インスタンスへのアクセスに使用するサービス名。
メジャー・フォルダ	Warehouse Builder にインポートするメジャーが格納されているフォルダの名前を指定します。メジャーのインポート時に、指定したメジャーに対応するキューブを含むコレクションが作成されます。
プロジェクト	インポート中に Warehouse Builder に作成される新規プロジェクトの名前を入力します。このパラメータは、インポート・モードに基づきます。メタデータを置換または更新する場合は、メタデータのインポート先とする既存のプロジェクトを指定する必要があります。
リレーショナル・スキーマのデフォルト・モジュール・タイプ	メタデータ、ソース、ウェアハウスのインポート先とするモジュールのタイプを選択します。転送ウィザードにより、適切なタイプのモジュールが作成され、そこにメタデータがインポートされます。

表 22-4 OLAP Server インポートに指定する転送パラメータ（続き）

転送パラメータ名	説明
OLAP 物理表現の処理	スター・スキーマからメタデータをインポートする場合、その物理表現を保持するときは「はい」、論理定義のみをインポートするときは「いいえ」を選択します。スノーフレーク・スキーマをインポートする場合、その物理実装は失われ、Warehouse Builder にインポートされるのは論理定義のみです。

表 22-5 OMG CWM ファイルのインポート用転送パラメータ

転送パラメータ名	説明
OMG CWM 入力ファイル	インポートする OMG CWM モデルを含むファイルを指定します。ファイルのロケーションを参照するには、「...」をクリックします。
CWM プロジェクト	インポートする OMG CWM モデルを含むファイルを指定します。ファイルのロケーションを参照するには、「...」をクリックします。
リレーショナル・スキーマのデフォルト・モジュール・タイプ	メタデータ、ソース、ウェアハウスのインポート先とするモジュールのタイプを選択します。転送ウィザードにより、適切なモジュールが作成され、そこにメタデータがインポートされます。
OLAP 物理表現の処理	スター・スキーマからメタデータをインポートする場合、その物理表現を保持するときは「はい」、論理定義のみをインポートするときは「いいえ」を選択します。スノーフレーク・スキーマをインポートする場合、その物理実装は失われ、Warehouse Builder にインポートされるのは論理定義のみです。
インポート・モード	作成、置換、更新、増分更新の中から、インポート・モードを選択します。
ログ・レベル	転送に割り当てられるログ・レベル : エラー、情報およびトレース。 **エラー :** 転送で発生したエラーの一覧。 **情報 :** メタデータ・オブジェクトの転送の詳細（エラーの詳細も含む）。 **トレース :** デバッグ用のトレース。これには、エラーおよび情報ログの詳細も含まれます。

表 22-6 CA ERwin のインポート用転送パラメータ

転送パラメータ名	説明
入力モデル・ファイル	ERwin からエクスポートされたファイルを選択します。 ファイルのロケーションを参照するには、「...」をクリックします。
OWB プロジェクト	新しくインポートされたプロジェクトの名前を入力します。このプロジェクトは、Warehouse Builder 内の「プロジェクト」ノードの下にあります。このパラメータは、インポート・モードに基づきます。メタデータを置換または更新する場合は、メタデータのインポート先とする既存のプロジェクトを指定する必要があります。
OWB モジュール名	メタデータ、ソース、ウェアハウスのインポート先とするモジュールのタイプを選択します。転送ウィザードにより、適切なモジュールが作成され、そこにメタデータがインポートされます。
リレーショナル・スキーマのデフォルト・モジュール・タイプ	インポート時に利用する名前マッチング・モード。デフォルトの設定では物理名モードになっています。
OLAP 物理表現の処理	スター・スキーマからメタデータをインポートする場合、その物理表現を保持するときは「はい」、論理定義のみをインポートするときは「いいえ」を選択します。スノーフレーク・スキーマをインポートする場合、その物理実装は失われ、Warehouse Builder にインポートされるのは論理定義のみです。
インポート・モード	作成、置換、更新、増分更新の中から、インポート・モードを選択します。
ログ・レベル	転送に割り当てられるログ・レベル : エラー、情報およびトレース。 **エラー :** 転送で発生したエラーの一覧。 **情報 :** メタデータ・オブジェクトの転送の詳細（エラーの詳細も含む）。 **トレース :** デバッグ用のトレース。これには、エラーおよび情報ログの詳細も含まれます。

表 22-7 Powersoft PowerDesigner のインポート用転送パラメータ

転送パラメータ名	説明
入力モデル・ファイル	PowerDesigner からエクスポートされたファイルを選択します。ファイルのロケーションを参照するには、「...」をクリックします。

表 22-7 Powersoft PowerDesigner のインポート用転送パラメータ（続き）

転送パラメータ名	説明
OWB プロジェクト	新しくインポートされたプロジェクトの名前を入力します。このプロジェクトは、Warehouse Builder 内の「プロジェクト」ノードの下にあります。このパラメータは、インポート・モードに基づきます。メタデータを置換または更新する場合は、メタデータのインポート先とする既存のプロジェクトを指定する必要があります。
OWB モジュール名	インポート中に作成する Warehouse Builder モジュールの名前を入力します。
PDM 名のマッピング	Physical Data Model（CDM）に対して論理名（NAME）を使用するか、物理名（CODE）を使用するかを選択できます。デフォルトの設定では CODE になっています。
サブモデルのマッピング	Warehouse Builder にインポートするサブモデルの名前。デフォルトの選択（*）では、モデル内のすべてのサブモデルがインポートされます。
リレーショナル・スキーマのデフォルト・モジュール・タイプ	インポート時に利用する名前マッチング・モード。デフォルトの設定では物理名モードになっています。
OLAP 物理表現の処理	スター・スキーマからメタデータをインポートする場合、その物理表現を保持するときは「はい」、論理定義のみをインポートするときは「いいえ」を選択します。スノーフレーク・スキーマをインポートする場合、その物理実装は失われ、Warehouse Builder にインポートされるのは論理定義のみです。
インポート・モード	作成、置換、更新、増分更新の中から、インポート・モードを選択します。
ログ・レベル	転送に割り当てられるログ・レベル : エラー、情報およびトレース。 **エラー :** 転送で発生したエラーの一覧。 **情報 :** メタデータ・オブジェクトの転送の詳細（エラーの詳細も含む）。 **トレース :** デバッグ用のトレース。これには、エラーおよび情報ログの詳細も含まれます。

表 22-8 OMG CWM へのメタデータのエクスポート用転送パラメータ

転送パラメータ名	説明
エクスポートされた OWB のコレクション	Warehouse Builder からエクスポートするコレクションを選択します。デフォルトは「すべてのコレクション」です。ドロップダウン・リストからコレクションを個別に選択することもできます。
OMG CWM 出力ファイル	エクスポートするコレクション用の出力ファイルを入力します。ファイルを参照および選択するには、フィールドと「...」ボタンをクリックします。
ログ・レベル	ログ・レベル（エラー、情報、トレース）を転送に割り当てます。 **エラー :** 転送で発生したエラーの一覧。 **情報 :** メタデータ・オブジェクトの転送の詳細（エラーの詳細も含む）。 **トレース :** デバッグ用のトレース。これには、エラーおよび情報ログの詳細も含まれます。
OWB 変換済言語	エクスポートする変換済言語を選択します。デフォルトは、リポジトリの MLS ベース言語になります。特定の言語を選択してエクスポートするか、「すべて」を選択してサポートされているすべての言語をエクスポートできます。

表 22-9 Discoverer へのメタデータのエクスポート用転送パラメータ

転送パラメータ名	パラメータの説明
エクスポートされた OWB のコレクション	Warehouse Builder からエクスポートするコレクションを選択します。デフォルトは「すべてのコレクション」です。ドロップダウン・リストからコレクションを個別に選択することもできます。
Discoverer EUL 所有者	Discoverer EUL 所有者の名前。
Discoverer スキーマの所有者	Discoverer スキーマの所有者の名前。
ディメンションの再使用	ディメンションの再使用を True または False で指定するオプション（同一のディメンションを指定するキューブにおいて複数の外部キー制約として定義されます）。 外部キー列の変更した論理名を、Discoverer 内の個別のフォルダに表示する場合は「true」を選択します。

表 22-9 Discoverer へのメタデータのエクスポート用転送パラメータ（続き）

転送パラメータ名	パラメータの説明
Discoverer 出力ファイル	転送ウィザードで生成されて Discoverer にインポートされる Discoverer .EEX ファイル（メタデータを含む）のパスとファイル名。 パスを入力するか、Discoverer 出力ファイル・パラメータにある転送パラメータ値の列の「参照」をクリックしてパスを選択します。.EEX ファイルの名前を任意で作成して、パスの最後に入力します。
ログ・レベル	ログには、転送の成否を示すデータが表示されます。転送ウィザードでは、デバッグを支援する 2 つのログ・レベル（エラーおよびトレース）がさらに提供されます。 **エラー :** 転送で発生したエラーの一覧。 **情報 :** メタデータ・オブジェクトの転送の詳細（エラーの詳細も含む）。 **トレース :** デバッグ用のトレース。これには、エラーおよび情報ログの詳細も含まれます。
OWB 変換済言語	エクスポートする変換済言語を選択します。デフォルトは、リポジトリの MLS ベース言語になります。特定の言語を選択してエクスポートするか、「すべて」を選択してサポートされているすべての言語をエクスポートできます。言語ごとに EEX ファイルが作成されます。

表 22-10 Express へのメタデータのエクスポート用転送パラメータ

転送パラメータ名	パラメータの説明
エクスポートされた OWB のコレクション	Warehouse Builder からエクスポートするコレクションを選択します。デフォルトは「すべてのコレクション」です。ドロップダウン・リストからコレクションを個別に選択することもできます。
Express ユーザー	Express リポジトリの接続用ユーザー ID。
Express ユーザー・パスワード	Express ユーザー用のパスワード。
Express 接続文字列	Express の接続用 ODBC 文字列で、次の構文を使用します。 host_machine_name:port_number:database_sid
Express 表の所有者	Express 表の所有者の名前。
RAA Version Number	Relational Access Administrator のバージョン。

表 22-10 Express へのメタデータのエクスポート用転送パラメータ（続き）

転送パラメータ名	パラメータの説明
ログ・レベル	ログには、転送の成否を示すデータが表示されます。転送ウィザードでは、デバッグを支援する 2 つのログ・レベル（エラーおよびトレース）がさらに提供されます。 **エラー :** 転送で発生したエラーの一覧。 **情報 :** メタデータ・オブジェクトの転送の詳細（エラーの詳細も含む）。 **トレース :** デバッグ用のトレース。これには、エラーおよび情報ログの詳細も含まれます
OWB 変換済言語	エクスポートする変換済言語を選択します。デフォルトは、リポジトリの MLS ベース言語になります。特定の言語を選択してエクスポートするか、「すべて」を選択してサポートされているすべての言語をエクスポートできます。

表 22-11 Oracle9*i* OLAP Server へのメタデータのエクスポート用転送パラメータ

転送パラメータ名	パラメータの説明
エクスポートされた OWB のコレクション	Warehouse Builder からエクスポートするコレクションを選択します。デフォルトは「すべてのコレクション」です。ドロップダウン・リストからコレクションを個別に選択することもできます。
ユーザー名	Oracle9*i* OLAP Server のユーザー名。
パスワード	Oracle9*i* OLAP Server のパスワード。
ホスト名	Oracle9*i* OLAP Server のホスト名。
ポート	Oracle9*i* OLAP Server のポート番号。
SID	Oracle9*i* OLAP サーバーのデータベース SID。
PL/SQL 出力ファイル	生成された PL/SQL ファイルのファイル名。PL/SQL ファイルでは実行時に 1 つのパラメータを渡す必要があります。このパラメータは、OLAP オブジェクトが作成されるスキーマです。したがって、ファイル 'sales_history.sql' をスキーマ 'SH' で実行する場合は、'sqlplus OLAPDBA/<passwd>@sales_history.sql SH' と入力します。
データベース中に PL/SQL を展開	「はい」または「いいえ」。データベースで生成済 PL/SQL を実行するか、スケジュールされた実行用に PL/SQL を保存します。

表 22-11 Oracle9*i* OLAP Server へのメタデータのエクスポート用転送パラメータ（続き）

転送パラメータ名	パラメータの説明
ログ・レベル	ログには、転送の成否を示すデータが表示されます。転送ウィザードでは、デバッグを支援する 2 つのログ・レベル（エラーおよびトレース）がさらに提供されます。 **エラー :** 転送で発生したエラーの一覧。 **情報 :** メタデータ・オブジェクトの転送の詳細（エラーの詳細も含む）。 **トレース :** デバッグ用のトレース。これには、エラーおよび情報ログの詳細も含まれます。
OWB 変換済言語	エクスポートする変換済言語を選択します。デフォルトは、リポジトリの MLS ベース言語になります。特定の言語を選択してエクスポートするか、「すべて」を選択してサポートされているすべての言語をエクスポートできます。
AW に配布	アナリティック・ワークスペースにメタデータを配布するかどうかを指定します。
AW 名	作成または保持されるアナリティック・ワークスペースの名前。
AW オブジェクト接頭辞	メタデータの配布先となるアナリティック・ワークスペースの名前を指定します。
キューブ・データをロード	AW にキューブのデータをロードします。このオプションを指定しない場合、プログラムをロードするキューブのみロードされます。
ディメンションのサロゲート・キーを生成	ディメンション値が全レベル間で一意でない場合は、ディメンションのサロゲート・キーを生成します。このオプションは、サロゲート・キーの生成に使用できます。
ビュー定義の生成	アナリティック・ワークスペースにビュー定義を生成するかどうかを指定します。このオプションを選択すると、SQL ビューに基づいて、アナリティック・ワークスペースにキューブの OLAP メタデータを生成することもできます。
マテリアライズド・ビューの生成	リレーショナル・キューブに、マテリアライズド・ビューを生成するかどうかを指示します。
生成済ビュー・ディレクトリ	生成されたビュー・スクリプトを格納するディレクトリの名前を指定します。データベース・オプションに「PL/SQL を展開」を選択すると、このディレクトリはクライアント側に置かれます。これ以外のオプションを選択すると、これはサーバー側のディレクトリ・オブジェクトとなります。

転送時の注意点

メタデータを Warehouse Builder Design Repository に転送する前に、あるいは Warehouse Builder Design Repository から転送する前に、メタデータを正常に転送できるように、ソース・ツール内やターゲット・ツール内で作業を行う必要があります。

この項では、メタデータをソース・ツールからリポジトリへ転送する手順、またはリポジトリからターゲット・ツールへ転送する手順を説明します。また、転送時にソース・ツールから抽出されるオブジェクトと、それらのオブジェクトに対応する Warehouse オブジェクトも示しています。

Warehouse Builder 転送ウィザードでは、Warehouse Builder Design Repository のメタデータをコレクションとしてエクスポートします。メタデータをエクスポートする前に、Warehouse Builder Design Repository 内にコレクションを作成する必要があります。

この項では、次のトピックについて説明します。

- Object Management Group の CWM 標準システムからのメタデータのインポート
- Computer Associates の ERwin 3.5.1 からのメタデータのインポート
- Powersoft の PowerDesigner 6.0 からのメタデータのインポート
- Oracle9i OLAP Server からのメタデータのインポート
- Oracle Discoverer 4i および Oracle9iAS Discoverer へのメタデータのエクスポート
- Object Management Group の CWM 標準システムへのメタデータのエクスポート
- Oracle Express へのメタデータのエクスポート
- Oracle9i OLAP Server へのメタデータのエクスポート

Object Management Group の CWM 標準システムからのメタデータのインポート

表 22-12 は、Object Management Group（OMG）の Common Warehouse Metamodel（CWM）ファイル・システムから Warehouse Builder Design Repository へのオブジェクト変換を示しています。

表 22-12 Object Management Group の CWM 標準変換

Warehouse Builder のオブジェクト	OMG CWM にエクスポートされたオブジェクト
プロジェクト	パッケージ
モジュール	スキーマ
ディメンション	ディメンション
レベル	レベル
レベル属性	属性
階層	階層
キューブ	キューブ
キューブ属性	メジャー
表	表
列	列
外部キー	外部キー
一意キー	一意制約 / 主キー
ビュー	ビュー
列	列

Computer Associates の ERwin 3.5.1 からのメタデータのインポート

ERwin のファイルを Warehouse Builder Design Repository にインポートする際は、転送するメタデータを含むファイルを .ERX ファイルとして保存します。

表 22-13 は、ERwin から Warehouse Builder Design Repository へのオブジェクト変換を示しています。

表 22-13 ERwin から Warehouse Builder にインポートする際のオブジェクト変換

ERwin のオブジェクト	Warehouse Builder にインポートされたオブジェクト
表	表
列	列
外部キー	外部キー
主キー	一意キー
ビュー	ビュー
列	列

Powersoft の PowerDesigner 6.0 からのメタデータのインポート

PowerDesigner のファイルを Warehouse Builder Design Repository にインポートする際は、転送するメタデータを含むファイルを .PDM ファイルとして保存します。

表 22-14 は、PowerDesigner から Warehouse Builder Design Repository へのオブジェクト変換を示しています。

表 22-14 PowerDesigner から Warehouse Builder にインポートする際のオブジェクト変換

PowerDesigner のオブジェクト	Warehouse Builder にインポートされたオブジェクト
表	表
列	列
索引	外部キー
索引	一意キー
ビュー	ビュー
列	列

Oracle9*i* OLAP Server からのメタデータのインポート

表 22-15 は、Oracle9*i* OLAP Server から Warehouse Builder Design Repository へのオブジェクト変換を示しています。

表 22-15 Oracle9*i* OLAP Server から Warehouse Builder にインポートする際のオブジェクト変換

Oracle9*i* の OLAP オブジェクト	Warehouse Builder にインポートされたオブジェクト
スキーマ	モジュール
キューブ	キューブ
メジャー	メジャー
ディメンション	ディメンション
レベル	レベル
階層	階層
属性	レベル属性
メジャー・フォルダ	コレクション

Oracle9*i* OLAP Server で使用可能な構造の中には、Warehouse Builder でサポートされていないものがあります。たとえば、スノーフレーク・モデリングやビューに基づくキューブはサポートされていません。これらの構造を使用する OLAP 定義では、オブジェクトの論理的な定義をインポートできても、物理表現はインポートできません。

OLAP からのインポートでの注意事項

この項では、OLAP Server からキューブおよびディメンションをインポートする際のガイドラインを示します。

Warehouse Builder で接頭辞を使用して生成されたレベル ID 列は、その接頭辞とともにインポートされます。レベル ID 列はリレーショナル・ディメンション・レベル属性としてモデル化されないため、接頭辞なしの名前はデータベース内にありません。そのため、ディメンションをインポートするとき、定義済の Warehouse Builder レベル属性を持つ列を識別するレベルでは、その名前にも接頭辞が使用されます。

これらの接頭辞は手動で削除できます。たとえば、Warehouse Builder でディメンション CUSTOMERS に、レベル接頭辞が CT のレベル COUNTRY を定義すると、このレベルには列を識別するレベル属性が COUNTRY_ID という名前で作成されます。生成されたコードでは、COUNTRY は CT_COUNTRY_ID にマッピングされ、接頭辞が列名に組み込まれます。この列をインポートすると、Warehouse Builder で定義したレベル属性は CT_COUNTRY_ID という名前になります。

ブリッジ・パラメータ「OLAP 物理表現の処理」を True に設定すると、スター・スキーマのリレーショナル・ディメンションの物理構造がインポートされます。キューブがスノーフレークの場合は、設定とは関係なく、キューブの論理定義がインポートされ物理実装は廃棄されます。キューブが複数のスキーマでディメンションを参照している場合は、このディメンションは無視されます。

Oracle Discoverer 4*i* および Oracle9*i*AS Discoverer へのメタデータのエクスポート

Warehouse Builder 転送ウィザードを使用して Warehouse Builder Design Repository のファイルを Discoverer にエクスポートする場合は、次の用語マトリックスを参照して、Discoverer に Warehouse Builder Design Repository のオブジェクトをマッピングする方法を確認できます。表 22-16 は、Warehouse Builder Design Repository から Discoverer へのオブジェクト変換を示しています。

表 22-16 Warehouse Builder から Discoverer へのオブジェクト変換

Warehouse Builder のオブジェクト	Discoverer にエクスポートされたオブジェクト
ディメンション	フォルダ
レベル	アイテム階層ノード
レベル属性	項目
階層	階層
キューブ	フォルダ
キューブ属性	項目
表	フォルダ
列	項目
一意キー	キー
ビュー	フォルダ（ビュー）
列	項目
アイテム・クラスとしてフラグ付けされた属性	アイテム・クラス
コレクション	ビジネス領域

メタデータを Discoverer に転送する前に、メタデータが正常に転送され、かつ Discoverer で適切に表示されるように、Warehouse Builder 内で追加作業を行う必要があります。次の作業を実行します。

- ディメンションを再使用するために Warehouse Builder を構成します。
- Warehouse Builder のウェアハウス・モジュールでコレクションを作成し、転送されたメタデータが Discoverer 内のビジネス領域として表示されるようにします。
- Warehouse Builder 表のいくつかの列を非表示にし、Discover Administrator ではグレー表示に、Discoverer Desktop では使用不可能になるようにします。
- Discoverer でアイテム・クラスとして使用する属性を指定します。
- Discoverer 内のアイテム階層ノードにどの属性を配置するかを指定するために名前を付けます。

次の項では、これらの作業のための手順を説明します。

ディメンションを再使用するための Warehouse Builder の構成

Warehouse Builder 内では、1 つのディメンションを指定する複数の外部キー制約を 1 つのキューブに含められます。たとえば、カレンダー・ディメンションでは、発注日や出荷日などを含められます。この機能は、ディメンションの再使用と呼ばれています。Discoverer 内でディメンション・ロールを正常に表示するには、これらの制約でそれぞれ別個のフォルダを作成する必要があります。ディメンション・ロールは、Discoverer での問合せの作成を容易にするためには不可欠です。カレンダー・ディメンションの例では、発注日のフォルダと出荷日のフォルダがそれぞれ表示されます。

注意： Warehouse Builder 転送ウィザードで「ディメンションの再使用」パラメータ値を TRUE に設定する場合のみ、次の手順に従ってください。

Discoverer でディメンション・ロールを正常に表示するには、Warehouse Builder でダミー表を作成する必要があります。このダミー表はディメンション・ロールになり、表の列はキューブ・ディメンションで使用される列になります。

ダミー表のネーミング規則としては、次の形式に従うことをお薦めします。

< ディメンション参照 >##< ロール参照 >

ここでディメンション参照やロール参照は、この表がどのディメンションのどのロールを表すのかを示しています。表の論理名はロール名になります。ダミー表をより詳細に識別するには、「配布可能」オプションを false に設定します。

ダミー表と実ディメンションは、表に含まれるどの列を使用しても対応付けられます。ダミー列とキューブの対応付けは、列の物理名をキューブの外部キー制約の物理名と同じものにすることによって行います。

次の例では、Discoverer にエクスポートするためにディメンション・ロールを設定する方法を示します。

Warehouse Builder でのディメンションとキューブの定義

1. 通常の方法で DAY、PRODUCT および OPERATOR などのディメンションを作成します。
2. 通常の方法で SALES、FLIGHTS などのキューブを作成します。
3. 次の制約を作成します。
   - SALES to DAY（ORDER_DATE 用）、制約の物理名は SALES_DAY_FK
   - SALES to DAY（SHIP_DATE 用）、制約の物理名は SALES_DAY2_FK
   - SALES to PRODUCT（PURCHASE 用）、制約の物理名は SALES_PROD_FK
   - FLIGHTS to DAY（ORDER_DATE 用）、制約の物理名は FLIGHTS_DAY_FK
   - FLIGHTS to PRODUCT（TICKET 用）、制約の物理名は FLIGHTS_PROD_FK
   - FLIGHTS to OPERATOR、制約の物理名は FLIGHTS_OPERATOR_FK

ダミー表の定義

表エディタを使用して、ダミー表を次のように作成します。

1. DAY（ORDER_DATE）ロールでは、表を次のように定義します。
   - 物理名 DAY##ORD（論理名 ORDER_DATE）
   - 列の物理名 SALES_DAY_FK
   - 列の物理名 FLIGHTS_DAY_FK
2. DAY（SHIP_DATE）ロールでは、表を次のように定義します。
   - 物理名 DAY##SHIP（論理名 SHIP_DATE）
   - 列の物理名 SALES_DAY2_FK
3. PRODUCT（PURCHASE）ロールでは、表を次のように定義します。
   - 物理名 PROD##PUR（論理名 PURCHASE）
   - 列の物理名 SALES_PROD_FK
4. PRODUCT（TICKET）ロールでは、表を次のように定義します。
   - 物理名 PROD##TIC（論理名 TICKET）
   - 列の物理名 FLIGHTS_PROD_FK

5. ダミー表を右クリックしてポップアップ・メニューから「構成」を選択し、「配布可能」値を false に設定します。

階層内で特定のレベル属性を使用するには、レベル属性の物理名を次のどちらかにする必要があります。

- < レベル名 >_NAME
- NAME

たとえば、製品ディメンションに、次の属性（物理名）を持つ 3 つのレベルで構成される 1 つの階層 p_hierarchy があるとします。

PRODUCT_TYPE:

- T_ID（一意キー（レベル））
- T_NAME
- T_DESCRIPTION

PRODUCT_CATEGORY

- C_ID（一意キー（レベル））
- C_NAME
- C_DESCRIPTION

PRODUCT

- P_ID（一意キー（レベル）; 物理主キー）
- P_NAME
- P_DESCRIPTION

ブリッジのデフォルトの動作では、T_ID、C_ID および P_ID に基づいて階層が構築されます。たとえば、ID のかわりに名前を使用するには、物理属性名を持つ次のレベルを定義します。

PRODUCT_TYPE

- T_ID（一意キー（レベル））
- PRODUCT_TYPE_NAME（または NAME）
- T_DESCRIPTION

PRODUCT_CATEGORY

- C_ID（一意キー（レベル））
- PRODUCT_CATEGORY_NAME（または NAME）
- C_DESCRIPTION

PRODUCT

- P_ID（一意キー（レベル）; 物理主キー）
- PRODUCT_NAME（または NAME）
- P_DESCRIPTION

この場合は、ブリッジにより、階層ノード属性としてそれぞれの NAME 属性が使用されます。

転送前のデータを非表示にする方法

Warehouse Builder の表の特定の列（廃止された列または未使用の列など）を非表示にして、それらが Discoverer Administrator ではグレー表示に、Discoverer Desktop では使用不可能になるよう設定できます。これらの列を非表示にするには、Warehouse Builder でカスタムのエンティティ属性セットを作成します。

Warehouse Builder で表の列を非表示にする手順は次のとおりです。

1. Discoverer に転送する前に Warehouse Builder のメイン・ナビゲータ・ツリーで、非表示にする表の列が存在するウェアハウスのターゲット・モジュールをダブルクリックします。

   ウェアハウス・モジュール・エディタが表示されます。

2. ウェアハウス・モジュール・エディタのナビゲーション・ツリーを開きます。
3. 列を非表示にするキューブまたはディメンション表を右クリックし、ポップアップ・メニューから「プロパティ」を選択します。

   その表の「プロパティ」ウィンドウが表示されます。

4. 「プロパティ」ウィンドウの「属性セット」タブを選択します。
5. 「追加」をクリックしてから属性の型を BRIDGE_TYPE に設定して、エクスポート用の新規の属性セットを作成します。
6. この属性セットの中に含める属性を選択し、「拡張」をクリックします。

   「拡張された属性セット・プロパティです」ダイアログが表示されます。

7. 「Hidden」フィールドの下にある対応するボックスをチェックし、非表示にする列を選択します。
8. このダイアログを使用すると、アイテム・クラスを定義でき、デフォルトの集計操作や Discoverer 内に存在する表の列のデフォルトの位置も設定できます。

転送されたデータの Discoverer へのインポート

選択したメタデータを転送ウィザードで転送した後は、.EEX ファイルを Discoverer にインポートします。

Warehouse Builder からインポートされたプロジェクトを Discoverer 4*i* および Oracle9*i*AS Discoverer でアクセスする方法の詳細は、『Oracle Discoverer Administrator 管理ガイド』を参照してください。

Discoverer におけるディメンションの再使用のネーミング規則

転送ウィザードを使用して Warehouse Builder のメタデータを Discoverer に転送する際に「ディメンションの再使用」パラメータを TRUE に選択すると、Discoverer でディメンション・ロールを正常に表示することを指定するフラグが設定されます。詳細は、22-53 ページの「ディメンションを再使用するための Warehouse Builder の構成」を参照してください。

ディメンション・ロールは、Discoverer での問合せの作成を容易にするためには不可欠です。Discoverer では、2 つの表の間に複数の結合が存在すると、それらの表を表す 2 つのフォルダが問合せで最初に参照される時点で、どの結合を使用するか認識している必要があります。それ以降は、問合せの他の部分でもその結合を常に使用します。問合せを実行する際に別の結合を使用する必要がある場合は、同じ表に基づいた別のフォルダを作成するしかありません。このような複数の結合を使用するシナリオは、ディメンション・モデルによくあることです。

Discoverer では、ディメンション・ロールをサポートするため、ディメンションの各ロールで別個のフォルダが用意されます。たとえば、DAY ディメンションと SALES キューブが、ORDER_DATE と SHIP_DATE で結合されている場合は、Discoverer にディメンション・ロールのフォルダが 2 つ作成されます。同じロールが別のキューブで必要な場合には、最初に作成されたロールが再使用されます。たとえば、FLIGHTS キューブも ORDER_DATE ロールを使用している場合は、SALES キューブによって参照されているフォルダが参照されます。

ディメンションの再使用（Warehouse Builder の同じディメンションを複数の外部キー制約が指定している状態）は、Discoverer のナビゲーション・ツリーにおいて次の 2 つの方法で表示されます。

- Warehouse Builder のディメンション・フォルダの名前は、次の形式で表示されます。

```
DIMENSION LOGICAL NAME : RoleName
```

- ロールが定義されていない場合は、次のデフォルト・ロールが生成されます。

```
DIMENSION LOGICAL NAME : Default
```

前述の例（「ディメンションを再使用するための Warehouse Builder の構成」の例）のフォルダは、Discoverer で次のように表示されます。

DAY（非表示）

DAY : ORDER_DATE

DAY : SHIP_DATE

FLIGHTS

OPERATOR（非表示）

OPERATOR : デフォルト（前述のデフォルト・ロールを参照）

PRODUCT（非表示）

PRODUCT : PURCHASE

PRODUCT : TICKET

SALES

Object Management Group の CWM 標準システムへのメタデータのエクスポート

Warehouse Builder Design Repository からのメタデータを OMG CWM 標準システムにエクスポートする際は、Warehouse Builder 転送ウィザードによって OMG CWM の規格に準拠するファイルへ変換されます。表 22-17 は、Warehouse Builder Design Repository から OMG CWM 規格へのオブジェクト変換を示しています。

表 22-17 Warehouse Builder から OMG CWM へのオブジェクト変換

Warehouse Builder のオブジェクト	OMG CWM にエクスポートされたオブジェクト
プロジェクト	パッケージ
モジュール	スキーマ
ディメンション	ディメンション
レベル	レベル
レベル属性	属性
階層	階層
キューブ	キューブ
キューブ属性	メジャー
表	表
列	列
外部キー	外部キー

表 22-17 Warehouse Builder から OMG CWM へのオブジェクト変換（続き）

Warehouse Builder のオブジェクト	OMG CWM にエクスポートされたオブジェクト
一意キー	一意制約 / 主キー
ビュー	ビュー
列	列

Oracle Express へのメタデータのエクスポート

Warehouse Builder から Oracle Express へメタデータをエクスポートした後に、Express 内のメタデータにアクセスするには、次の手順に従います。

1. RAA をオープンします。
2. RAA リポジトリのユーザー名とパスワードを入力します。
3. 転送したプロジェクトをオープンします。ブリッジを実行した後に、プロジェクトがドロップダウン・リストに表示されます。
4. プロジェクトがオープンしたら「Express データベースのメンテナンス」を選択して、RAA リポジトリのユーザー名、パスワードおよびサービス名を入力します。
5. 「メンテナンス・プロシージャの作成」を選択して、プロシージャを指定し、「general maintenance」を選択します。

   RDBMS ログインが自動的に表示されます。
6. 「実行時に修飾済セレクションを生成」を選択します。
7. ディメンション処理のデフォルトを選択します。
8. Express データベースとログ・ファイルに名前を付けます。これらは OES サーバー・マシン上の d:¥orant¥database¥express_info に保存されます。
9. 「OES」を選択し、「OES Batch Manager」を選択して、このジョブを監視します（「ジョブ（複数）」→「モニター」）。

   初期の Express データベースが作成されました。

OSA の構成

データを参照する前に、次の構成の手順に従います。

1. OSA Application Manager をオープンします。
2. 「Database Setup」を選択し、「作成」を選択します。

   これで、.dsc ファイルを作成できます。転送したプロジェクトに合った名前を付けてください。「編集」を選択してリモート thin クライアントを選択します。構成ファイルをデータベースとログ・ファイルを保存したディレクトリから見つけて、転送したプロジェクトに関連する .rdc ファイルを選択します。
3. 「Communication Setup」→「Define」→「作成」を選択します。
4. 設定に名前を付けてから、thin クライアントとリモート・システムを選択します。これはサーバー・マシンです。RPC の場合には、TCP/IP を選択します。
5. これで、OSA をオープンできます。「新規」を選択し、「レポート」または「Graphs」を選択して、データを表示できます。この際、Warehouse Builder Design Repository に基づいて正しいディメンション、階層およびメジャーを最初に選択する必要があります。

Oracle9*i* OLAP Server へのメタデータのエクスポート

Oracle9*i* OLAP オブジェクトを配布するには、最初に Warehouse Builder のウェアハウス・オブジェクトをデータベース・サーバーに配布する必要があります。OLAP のメタデータをさらにエクスポートするには、メタデータのエクスポートのブリッジを使用します。

表 22-18 は、Warehouse Builder Design Repository オブジェクトから Oracle9*i* の OLAP オブジェクトへのオブジェクト変換を示しています。

表 22-18 Warehouse Builder から Oracle9*i* OLAP Server にエクスポートする際のオブジェクト変換

Warehouse Builder のオブジェクト	エクスポートされた OLAP オブジェクト
キューブ	キューブ
メジャー	メジャー
ディメンション	ディメンション
レベル	レベル
階層	階層
レベル属性	属性
コレクション	メジャー・フォルダ

Oracle9*i* OLAP Server にメタデータをエクスポートした後に、PL/SQL ファイルが生成されます。SQL*Plus を使用して、サーバーにこのファイルを配布できます。たとえば、スキーマ SH においてファイル sales_history.sql を実行するには、次のように入力します。

```
sqlplus OLAPDBA/<passwd> @sales_history.sql SH
```

これにより、OLAP キューブ、追加のディメンション・メタデータ、および SH スキーマのメジャー・フォルダが定義されます。このスクリプトが実行される前にウェアハウスのデータベース・オブジェクトを配布しなければ、オブジェクトの依存条件が満たされません。

第 VI 部

付録

この部には、次の付録があります。

- 付録 A「キーボードのショートカット」
- 付録 B「予約語」
- 付録 C「XML ツールキットの使用」
- 付録 D「Warehouse Builder Public View」
- 付録 E「パブリック・オブジェクト」

A

キーボードのショートカット

この付録には、Warehouse Builder のコマンドにアクセスするキーボード・オプションが記載されています。Windows 95 および NT 用に設計された大部分のアプリケーションがサポートする規則については、Microsoft Windows の『キーボード ガイド』を参照してください。この付録では、次のトピックについて説明します。

汎用 Windows キー

表 A-1 は、汎用 Windows キーのキーボード・シーケンスとその説明の一覧です。

表 A-1 汎用 Windows キー

キーボード・シーケンス	説明
[F1]	ヘルプ機能を起動します。
[Esc]	現在のウィンドウまたはメニューを閉じて、フォーカスを親（現在のウィンドウまたはメニューをアクティブにする前の設定）に移動します。現在は、「取消」コマンド・ボタンを使用できます。
削除	選択した項目を削除します。
[Tab]	フォーカスを次のコントロールに移動します。
[Shift]+[Tab]	フォーカスを前のコントロールに移動します。
[Space]	フォーカスがボタンにあるとき、現在の押しボタンをアクティブにします。チェックボックスの選択 / 選択解除を切り替えます。
[Enter]	フォーカスがボタンにあるとき、現在の押しボタンをアクティブにします。

ツリー・ビューの制御キー

表 A-2 は、ツリー・ビューの制御キーのキーボード・シーケンスとその説明の一覧です。

表 A-2 ツリー・ビューの制御キー

キーボード・シーケンス	説明
[→]	階層関係を伴うオブジェクトにフォーカスがあるとき、ウィンドウ・キャンバスでツリーを開きます。
[←]	階層関係を伴うオブジェクトにフォーカスがあるとき、ウィンドウ・キャンバスでツリーを閉じます。
[↑]、[↓]	前または次のオブジェクトを選択します。
[Home]	ウィンドウ・キャンバスの最初のコントロールにフォーカスを移動します。
[End]	ウィンドウ・キャンバスの最後のコントロールにフォーカスを移動します。

Warehouse Builder のアクセラレータ・キー・セット

アクセラレータ・キーは、よく使用するアクションのキーボード・ショートカットです。アクセラレータ・キーでは、特定のキーストロークの組合せを使用することにより、メニューを介さなくても対応するメニュー項目と同じ機能を実行できます。表 A-3 は、アクセラレータ・キーのキーボード・シーケンスとその説明の一覧です。

表 A-3 Warehouse Builder のアクセラレータ・キー・セット

キーボード・シーケンス	説明
[Ctrl]+[N]	新規ウィンドウを起動して、オブジェクトを作成します。
[Ctrl]+[X]	テキストを切り取ります。
[Ctrl]+[O]	オブジェクト・エディタを起動します。
[Ctrl]+[C]	テキストをコピーします。
[Ctrl]+[R]	オブジェクトのプロパティを開きます。
[Ctrl]+[V]	テキストを貼り付けます。
[Ctrl]+[F]	「オブジェクト検索」ダイアログを開きます。
[Shift]+[→]	テキストを選択します。
[Ctrl]+[S]	「コミット確認」ダイアログを開きます。
[Ctrl]+[P]	「印刷」ダイアログを開きます。
[Ctrl]+[Tab]	グリッドから使用可能な次のコントロールに移動し、使用可能なショートカット・ページ・タブを選択します。

メニュー・コマンドおよびアクセス・キー

メニュー・タイトルとメニュー項目には、下線付きのアクセス・キーが表示されます。アクティブなウィンドウ内にあるコントロールまたはメニューをアクティブにするには、[Alt] と同時にアクセス・キーを押します。メニュー項目に下線付きの文字が表示されていない場合は、上下の矢印キーを使用してメニュー項目にフォーカスを移動し、[Enter] を押します。アクセス・キーでは、[Alt] キーを使用しなくてもコントロールまたはメニュー項目を選択できる場合があります。開いているメニューから項目を選択する際は、[Alt] を使用しないでアクセス・キーを押します。表 A-4 は、メニュー・コマンドおよびアクセス・キーのキーボード・シーケンスとその説明の一覧です。

表 A-4 メニュー・コマンドおよびアクセス・キー

キーボード・シーケンス	説明
[Alt]	アクティブなウィンドウのメニュー・バーをアクティブにします。左端のメニュー名が選択されます。
[Alt]+ 任意の印字文字	メイン・メニュー・バーの下線付き文字（アクセス・キー）のメニューをアクティブにします。
任意の印字文字	開いているメニューで下線付き文字（アクセス・キー）のメニューをアクティブにします。
[←]、[→]	メニュー・バーのメニュー間で矢印の方向にフォーカスを移動します。元のメニューが開いていた場合は移動先のメニューが開き、先頭の項目にフォーカスが移ります。
[↑]、[↓]	選択したメニューを開きます。開いているメニューでは、前のコマンドおよび次のコマンドを選択します。
[Enter]	フォーカスがメニュー・タイトルにあるとき、その選択したメニューを開きます。フォーカスがメニュー項目にあるとき、そのメニュー項目をアクティブにします。
[Alt]+[F4]	アクティブなウィンドウを閉じて、現在のウィンドウまたはメニューがアクティブになる前の設定にフォーカスを戻します。
[Shift]+[F10]	アクティブなオブジェクトのショートカット・メニューを開きます（フォーカスがオブジェクト、項目にあるとき）。

マッピング・エディタのキーボード操作

[Tab] キーと矢印キーを使用して、マッピング・エディタ・キャンバス内を移動できます。

[Tab] キーを押すと、キャンバス上の次のオブジェクトに移動してそれを選択できます。オブジェクトが 1 つも選択されていないときは、マッピング・エディタ・キャンバスの左上隅に一番近いノードが選択されます。移動する順序は、オブジェクトがキャンバスに表示されている位置によって決まります。Z の形でジグザグに移動します。

選択中の属性グループ内では、上下の矢印キーを使用して属性から属性へ移動できます。

入力属性が選択されているときは、左の矢印キーで受信マッピング線に移動できます。属性 1 つにつき、受信マッピング線は 1 つしかありません。右の矢印キーは使用できません。

オブジェクトを選択して [Delete] キーを押すと、選択したオブジェクトはキャンバスから削除されます。削除できるオブジェクトは、次のとおりです。

- 演算子。削除される前に本当に削除してもよいかを確認するプロンプトが表示されます。
- 属性に対する送信または受信のマッピング線。属性グループまたはヘッダーで開始または終了するマッピング線は削除できません。

自動マッピングのニーモニック・キー割当て

「自動マッピング」ダイアログの要素に容易にアクセスできるように、次のニーモニック・キーが割り当てられています。表 A-5 は、ニーモニック・キーのキーボード・シーケンスとその説明の一覧です。

表 A-5 自動マッピングのニーモニック・キー割当て

キーボード・シーケンス	説明
[Alt]+[o]	「OK」
[Alt]+[c]	「取消」
[Alt]+[y]	「ソース属性をターゲット・グループにコピーおよび一致」
[Alt]+[p]	「ソースおよびターゲット属性の位置による一致」
[Alt]+[n]	「名前による一致」
[Alt]+[i]	「大 / 小文字の違いを無視」
[Alt]+[s]	「特殊文字を無視」
[Alt]+[e]	「ソースの接頭辞を無視」
[Alt]+[u]	「ソースの接尾辞を無視」
[Alt]+[t]	「ターゲットの接頭辞を無視」
[Alt]+[a]	「ターゲットの接尾辞を無視」
[Alt]+[g]	「実行」
[Alt]+[d]	「表示マッピング」
[Alt]+[h]	「ヘルプ」

B

予約語

この付録には、Warehouse Builder でオブジェクト名として使用できない予約語が記載されています。

予約語

次の表は、予約語の一覧です。これらの予約語は、Warehouse Builder のプロジェクトで物理オブジェクト名として使用しないでください。

- ABORT
- ACCEPT
- ACCESS
- ADD
- ALL
- ALTER
- AND
- ANY
- ARRAY
- ARRAYLEN
- AS
- ASC
- ASSERT
- ASSIGN
- AT
- AUDIT
- AUTHORIZATION
- AVG
- BASE_TABLE
- BEGIN
- BETWEEN
- BINARY_INTEGER
- BODY
- BOOLEAN
- BY
- CASE
- CHAR
- CHAR_BASE
- CHECK
- CLOSE
- CLUSTER
- CLUSTERS
- COLAUTH
- COLUMN
- COMMENT
- COMMIT
- COMPRESS
- CONNECT
- CONSTANT
- CRASH
- CREATE
- CURRENT
- CURRVAL
- CURSOR
- DATA_BASE
- DATABASE
- DATE
- DBA
- DEBUGOFF
- DEBUGON
- DECIMAL
- DECLARE
- DEFAULT
- DEFINITION
- DELAY
- DELETE
- DELTA
- DESC
- DIGITS
- DISPOSE
- DISTINCT
- DO
- DROP
- DUAL
- ELSE
- ELSIF
- END
- ENTRY
- EXCEPTION
- EXCEPTION_INIT
- EXCLUSIVE
- EXISTS
- EXIT
- FALSE
- FETCH
- FILE
- FLOAT
- FOR
- FORM
- FROM
- FUNCTION
- GENERIC
- GOTO
- GRANT
- GROUP
- HAVING
- IDENTIFIED
- IF
- IMMEDIATE
- IN
- INCREMENT
- INDEX
- INDEXES
- INDICATOR
- INITIAL
- INSERT
- INTEGER
- INTERFACE
- INTERSECT
- INTO

- IS
- LEVEL
- LIKE
- LIMITED
- LOCK
- LONG
- LOOP
- MAX
- MAXEXTENTS
- MIN
- MINUS
- MLSLABEL
- MOD
- MODE
- MODIFY
- NATURAL
- NATURALN
- NEW
- NEXTVAL
- NOAUDIT
- NOCOMPRESS
- NOT
- NOWAIT
- NULL
- NUMBER
- NUMBER_BASE
- OF
- OFFLINE
- ON
- ONLINE
- OPEN
- OPTION
- OR
- ORDER
- OTHERS
- OUT
- PACKAGE
- PARTITION
- PCTFREE
- PLS_INTEGER
- POSITIVE
- POSITIVEN
- PRAGMA
- PRIOR
- PRIVATE
- PRIVILEGES
- PROCEDURE
- PUBLIC
- RAISE
- RANGE
- RAW
- REAL
- RECORD
- REF
- RELEASE
- REMR
- RENAME
- RESOURCE
- RETURN
- REVERSE
- REVOKE
- ROLLBACK
- ROW
- ROWID
- ROWLABEL
- ROWNUM
- ROWS
- ROWTYPE
- RUN
- SAVEPOINT
- SELECT
- SEPARATE
- SESSION
- SET
- SIZE
- SMALLINT
- SPACE
- SQL
- SQLCODE
- SQLERRM
- START
- STATEMENT
- STDDEF[1]
- STDDEV
- SUBTYPE
- SUCCESSFUL
- SUM
- SYNONYM
- SYSDATE
- TABAUTH
- TABLE
- TABLEDEF[1]
- TABLEPAT[1]
- TABLES
- TASK
- TERMINATE
- TIME
- THEN
- TO
- TRIGGER
- TRUE
- TYPE
- UID
- UNION
- UNIQUE
- UPDATE
- USE
- USER
- USERDEF[1]
- VALIDATE
- VALUES
- VARCHAR

- VARCHAR2
- VARIANCE
- VIEW
- WHEN
- WHENEVER
- WHERE
- WHILE
- WITH
- WORK
- WRITE
- XOR

[1] STDDEF、TABLEDEF、TABLEPAT、USERDEF が予約語となるのは、Oracle Warehouse Builder Name and Address コンポーネントを購入して、解析表のメンテナンスを行う場合のみです。

C

XML ツールキットの使用

この付録では、XML ツールキットを使用して XML 文書からデータを取得し、そのデータをターゲット・スキーマに保存する方法について説明します。このツールキットでは、フォーマットの異なる様々なソースに保存されている XML 文書からデータを抽出し、そのデータを Oracle Database の表、CLOB データベース列、アドバンスト・キュー、およびその他の Oracle データベース・オブジェクトに保存できます。

この付録では、次のトピックについて説明します。

概要

XML 文書は、XML（eXtensible Markup Language）仕様に準拠しています。XML を使用すれば、Web 上での文書配信に最適な独自のマークアップ文書を設計し、E-Commerce をはじめとする様々なアプリケーションを実装できます。

XML ツールキットは、PL/SQL プロシージャ、ファンクション、パッケージの集まりであり、XML 文書からデータを取得して、Oracle データベース・オブジェクトにロードすることができます。たとえば、ファイルに保存されている XML 文書からデータを取得し、データ・ウェアハウス内の複数の表にそのデータをロードできます。

注意： 現時点では、XML ツールキットの使用は比較的小規模なファイルのみに制限されています。詳細は、Warehouse Builder のリリース・ノートを参照してください。

ソースからのデータの取得

次のオブジェクトに保存されている XML 文書からデータを取得できます。

- ファイル
- 複数のファイル
- CLOB データベース列
- 行ベースのアドバンスト・キュー
- オブジェクトまたは CLOB ベースのアドバンスト・キュー
- URL

XML ツールキットに対してデータのソースを指定するには、このツールキットで定義されている要素名を使用します。たとえば、ファイルを指定するには要素名 file を使用します。この後の項および例では、各要素名とその属性（使用する場合）、および有効な構文について説明します。

ターゲットへのデータの保存

XML ツールキットでは、XML 文書から取得したデータを様々な Oracle データベース・オブジェクトに保存できます。XML ツールキットでは次のオブジェクトにデータを保存できます。

- ターゲットとなる Oracle Database の表（または複数の表）
- 更新可能なビュー
- オブジェクト表
- 更新可能なオブジェクト・ビュー
- オブジェクトまたは CLOB ベースのアドバンスト・キュー

データのターゲットを指定するには、XML ツールキットで定義されている要素名を使用します。たとえば、ターゲットの表を指定するには、要素名 target を使用します。

要素名 target には、ランタイム操作を制御するいくつかの属性があります。たとえば、XML 文書から抽出したデータがターゲット表と一致しない場合は、データのフォーマットを変更する属性として XSL スタイル・シートを指定できます。この後の項および例では、各要素名とその属性、および有効な構文について説明します。

ランタイム制御の使用

抽出プロセスとロード・プロセスの実行中に、そのロードをコミットするタイミングと各バッチのサイズを制御できます。これらを指定するには、要素名 `runtimeConfig` を使用します。

XML ツールキットのコール

XML ツールキットは、PL/SQL プロシージャとファンクションの集まりです。XML ツールキットを使用するには、最初に、このツールキットのいずれかのプロシージャまたはファンクションをコールする Warehouse Builder 変換を作成する必要があります。その後、マッピング前プロセスまたはマッピング後プロセスを使用して、作成した変換をコールできます。

通常は、XML 文書からデータを抽出し、そのデータをステージング表へロードする変換を作成します。この変換を一般化するには、取得元の XML 文書をランタイム・パラメータとして指定します。変換を作成した後、この変換をコールして、ステージング表をロードするマッピング前プロセスをマッピングに追加します。このマッピングによって、ステージング表内のデータが変換され、ターゲット表にロードされます。マッピング後プロセスは、マッピングが終了する前にステージング表を切り捨てます。

2 つのエントリ・ポイント

変換は、次のいずれかの API エントリ・ポイントを使用して XML ツールキットを実行できます。

- PL/SQL プロシージャ **`wb_xml_load(control_file)`**
- PL/SQL ファンクション **`wb_xml_load_f(control_file)`**

いずれのコールも XML 文書からデータを抽出し、そのデータをデータベース・ターゲットにロードします。ただし、このファンクションは、実際に読み込まれた文書の数を返します。制御ファイル（XML 文書）は、XML 文書のソース、ターゲット、およびランタイム制御を指定します。

変換を定義した後、通常はマッピング前プロセスまたはマップ後プロセスとしてその変換をコールします。

制御ファイルは XML 文書です。要素名 OWBXMLRuntime は、この文書の最上位レベル（ルート）の要素を表します。その他の要素名は、文書のソースとターゲットを表しています。

この変換を作成して名前を付けておけば、マッピング内でこの変換を名前で参照できます。通常、変換は、マッピング前プロセスまたはマッピング後プロセスとして使用します。

注意：

- 文書ソースおよび XSL スタイル・シートのファイル名は、Oracle データベース・インスタンスをサポートしているノードに対応しています。
- 前述のサンプル制御ファイルでは、テキストと連結文字の間に空白が挿入されています。これらの空白は読みやすさを考慮して挿入されたものなので、必ずしも入れる必要はありません。
- 制御ファイルは XML 文書です。C-20 ページの「制御ファイルの Document Type Definition」を参照してください。

一般的な制御ファイル

ここで紹介する 9 つのサンプル制御ファイルは、次のソースからデータを抽出し、そのデータをデータベース・ターゲットにロードするための制御ファイルです。

1. 1 つのファイル
2. 1 つのファイル（要素名の変更が必要）
3. 1 つのファイルから複数のターゲット
4. 複数のファイル
5. 分割された大きなファイル
6. CLOB 列
7. URL
8. 行ベースのアドバンスト・キュー
9. オブジェクト・タイプ・ベースのアドバンスト・キュー

最初の 5 つの例では、ファイルに保存されている XML 文書からデータを抽出します。その他の例では、データベース・オブジェクトまたは URL で指定された文書からデータを抽出します。これらの例は、XML ツールキットで処理できるすべてのソースを扱っています。

ファイルに保存されている XML 文書

XML ツールキットでは、1 つのファイル、または同じディレクトリ内の複数のファイルに保存されている文書からデータを抽出できます。また、これらのソースからデータを抽出し、そのデータを複数の表に保存することもできます。

XML 文書内のデータがターゲット・オブジェクトと一致しない可能性もあります。その場合は、ターゲットにあわせてデータをフォーマット変更するための XSL スタイル・シートを使用します。2 番目の例では、XSL スタイル・シートの参照方法を紹介します。ここでは、ターゲットの XSLFile 属性の値を指定します。XML 文書から抽出したデータと、データ・ウェアハウス内のターゲット表の列名とが一致するケースはほとんどないので、通常は、スタイル・シートを使用してソース・データのフォーマットを変更する必要があります。

最後の例では、サイズの大きい XML 文書をいくつかに分割し、各部を別々にロードすることによって処理効率を向上させます。この方法を使用するには、ソース・ファイルの splitElement 属性の値を指定します。

XML 文書の要素名がターゲット表の列名と正確に一致しており、その XML 文書がファイルに保存されている場合は、XML 文書からデータを抽出し、ターゲット表にロードするための制御ファイルを簡単に作成できます。

XML 文書

次の項の XML 文書（ORDERS）は、<OWBhome>¥owb¥xmltoolkit ファイルに保存されています。このファイル名は、Oracle データベース・インスタンスに対応しています。

```
<ORDERS>
 <?xml version="1.0"?>
 <ROW>
     <ID>100</ID>
     <ORDER_DATE>2000.12.20</ORDER_DATE>
     <SHIPTO_NAME>Jeff Q. Vintner</SHIPTO_NAME>
     <SHIPTO_STREET>500 Marine World Parkway</SHIPTO_STREET>
     <SHIPTO_CITY>Redwood City</SHIPTO_CITY>
     <SHIPTO_STATE>CA</SHIPTO_STATE>
     <SHIPTO_ZIP>94065</SHIPTO_ZIP>
   </ROW>
</ORDERS>
```

ターゲット表

次の項の DDL で記述されている次のターゲット表（PURCHASE_ORDERS）の列名は、前述の XML 文書の要素名と一致しています。

```
create table Purchase_Orders (
id varchar2(10) not null,
order_date date not null,
shipto_name varchar2(60) not null,
shipto_street varchar2(80) not null,
shipto_city varchar2(30) not null,
shipto_state varchar2(2) not null,
shipto_zip varchar2(9))
```

制御ファイル

次の項の制御ファイルは、ORDERS 文書からデータを抽出し、そのデータを Purchase_Orders 表にロードするよう XML ツールキットに指示します。

```
'<OWBXMLRuntime>'||
  '<XMLSource>'||
    '<file>¥ora817¥examples¥ex1.xml</file>'||
  '</XMLSource>'||
  '<targets>'||
    '<target dateFormat="yyyy.MM.dd">Purchase_Orders</target>'||
  '</targets>'||
'</OWBXMLRuntime>'
```

dateFormat 属性について

XML SQL Utility（XSU）が表内の order_date 列（DATE 型）にテキスト・データを正しく保存できるよう、ターゲット要素に対して dateFormat 属性を定義する必要があります。詳細は、『Oracle XML API リファレンス』を参照してください。

XSL プロセッサでは、様々なモードで保存されているデータを格納できます。それぞれサンプル制御ファイルが用意されています。変換が wb_xml_load_f ファンクションの wb_xml_load プロシージャを使用して XML ツールキットをコールする場合は、完全な制御ファイルを使用する必要があります。

```
wb_xml_load (control_info VARCHAR2)
wb_xml_load_f (control_info VARCHAR2) RETURN NUMBER
```

制御ファイルは、ソース、ターゲット、およびランタイム制御を指定した XML 文書です。この項の例では、様々な状況に対応した制御ファイルの定義方法を示します。

Warehouse Builder 変換

この制御ファイルを使用して変換を作成し、その変換をマッピング前プロセスからコールすると、次のような処理が実行されます。

1. XML ツールキットが文書 ORDERS をバッファに読み込み、そのデータを解析します。
2. XML ツールキットが Oracle XML SQL Utility（XSU）をコールし、解析したデータを Purchase_Orders 表にロードします。
3. 変換がマッピングに制御を返します。

これは、文書と表が正確に一致している単純なケースです。次の例では、より一般的なケース、つまり文書と表が正確に一致していない場合の処理方法を示します。

文書と表の不一致 XML 文書の要素名とターゲット列の列名が一致していない場合は、XSL スタイル・シートを使用して、データをターゲット表へロードする前にフォーマットを変更する必要があります。この例では、XSLFile 属性を使用してスタイル・シートを指定し、データのフォーマットを変更します。この例の制御ファイルでは、データを抽出した後、そのデータをフォーマット変更してから PURCHASE_ORDERS 表へロードするよう指示します。

XML 文書 次の項の XML 文書（purchaseORDER）は、¥ora817¥examples¥ex2.xml ファイルに保存されています。このファイル名は、Oracle データベース・インスタンスに対応しています。

```
<purchaseOrder>
<id>103123-4</id>
<orderDate>2000-10-20</orderDate>
<shipTo country="US">
     <name>Alice Smith</name>
     <street>123 Maple Street</street>
     <city>Mill Valley</city>
     <state>CA</state>
     <zip>90952</zip>
</shipTo>
<comment>Hurry, my lawn is going wild!</comment>
<items>
     <item>
          <partNum>872-AA</partNum>
          <productName>Lawnmower</productName>
          <quantity>1</quantity>
          <USPrice>148.95</USPrice>
          <comment>Confirm this is electric</comment>
     </item>

     <item>
          <partNum>845-ED</partNum>
          <productName>Baby Monitor</productName>
          <quantity>1</quantity>
          <USPrice>39.98</USPrice>
          <shipDate>1999-05-21</shipDate>
     </item>
</items>
</purchaseOrder>
```

ターゲット表

ターゲット表（PURCHASE_ORDERS）の列名は、C-6 ページを参照してください。

制御ファイル

次の項の制御ファイルは、purchaseORDER 文書からデータを抽出し、そのデータをフォーマット変更してからターゲット表 PURCHASE_ORDERS にロードするよう XML ツールキットに指示します。

```
'<OWBXMLRuntime>'||
'     <XMLSource>'||
'            <file>¥ora817¥examples¥ex2.xml</file>'||
'    </XMLSource>'||
'     <targets>'||
'            <target XSLFile="¥ora817¥examples¥ex2_1.xsl" dateFormat="yyyy-MM-dd">'||
'                                    Purchase_Orders||'
'        </target>'||
'    </targets>'||
'</OWBXMLRuntime>'
```

この制御ファイルでは、要素 target に XSLFile 属性が追加されています。

スタイル・シート

スタイル・シート自体は、XML 文書から取得して解析したデータ項目と、ターゲット表の列名とを関連付ける単純な XML 文書です。スタイル・シートの抜粋を次に示します。解析されたデータ項目とターゲット列をどのように関連付けるかを示しています。

```
<SHIPTO_NAME>
         <xsl:value-of select="shipTo/name"/>
</SHIPTO_NAME>
<SHIPTO_STREET>
       <xsl:value-of select="shipTo/street"/>
</SHIPTO_STREET>
<SHIPTO_CITY>
       <xsl:value-of select="shipTo/city"/>
</SHIPTO_CITY>
<SHIPTO_STATE>
       <xsl:value-of select="shipTo/state"/>
</SHIPTO_STATE>
```

この制御文を変換内で使用すれば、XML 文書からデータを抽出し、そのデータをターゲット表にロードすることができます。

Warehouse Builder 変換

この制御ファイルを使用して変換を作成し、その変換をマッピング前プロセスからコールすると、次のような処理が実行されます。

1. XML ツールキットが文書 ORDERS をバッファに読み込み、そのデータを解析します。
2. XML ツールキットはスタイル・シートに従って、解析されたデータと表の列を関連付けます。
3. Oracle XML SQL Utility（XSU）が解析済データを PURCHASE_ORDERS にロードします。

変換がマッピングに制御を返します。

複数のターゲット

この制御文は、purchaseORDER 文書からデータを抽出し、そのデータをフォーマット変更した後で 2 つのターゲット表 PURCHASE_ORDERS と ITEMS に保存します。これら 2 つの表では異なる列名が使用されているので、各ターゲットについてスタイル・シートを指定する必要があります。

XML 文書 C-8 ページの例で使用した XML スクリプトがソース文書（purchaseORDER）になります。

ターゲット表 Purchase_Orders の列名は、C-6 ページの「ターゲット表」を参照してください。制御ファイルのスクリプト理解に、項目の列名は必要ありません。

制御ファイル 次の項の制御ファイルは、purchaseORDER 文書からデータを抽出し、そのデータをフォーマット変更してから 2 つのターゲット表にロードするよう XML ツールキットに指示します。

```
'<OWBXMLRuntime>'||
'     <XMLSource>'||
'            <file>¥ora817¥examples¥ex2.xml</file>'||
'    </XMLSource>'||
'     <targets>'||
'            <target XSLFile="¥ora817¥examples¥ex2_1.xsl" dateFormat="yyyy-MM-dd">'||
'                                    Purchase_Orders||'
'           </target>'||
'            <target XSLFile="¥ora817¥examples¥ex2_2.xsl" dateFormat="yyyy-MM-dd">'||
'                                    Items||'
'        </target>'||
'    </targets>'||
'</OWBXMLRuntime>'
```

この制御ファイルは前の例と似ていますが、ここでは、もう 1 つのターゲット（ITEMS）を指定しています。また、XML 文書から取得して解析したデータ項目と Items 表の列を関連付けるためのスタイル・シート（ex2_2.xsl）も指定しています。

非常に大きい XML 文書

非常に大きい XML 文書からデータを抽出するときは、メモリーを大量に消費するため、ロード・パフォーマンスが低下する可能性があります。このような場合、XML 文書を分割すれば、必要メモリー容量が少なくなり、処理効率を向上させることができます。

XML 文書を分割するには、XMLSource 要素で splitElement 属性の値を指定します。この値は、ソース XML 文書内の要素の名前です。できるだけテキスト・ブロックを示す名前を使用してください。

たとえば、XML 文書内で要素名 Category を使用して、複数の書籍を分類しているとします。文書内には多数のカテゴリがあり、さらに各カテゴリには多数の書籍が含まれています。したがって、次の項の例のように splitElement 属性を指定すると、ソース文書がカテゴリと同じ数に分割されます。

制御ファイル 次の項の制御ファイルは、カタログに基づいて BookAreUs 文書を分割し、分割した BookAreUs 文書から抽出したデータをターゲット表 books にロードするよう XML 文書に指示します。

```
'<OWBXMLRuntime>'||
'     <XMLSource splitElement="Category">'||
'               <file>¥ora817¥examples¥ex4.xml</file>'||
'    </XMLSource>'||
'     <targets>'||
'             <target XSLFile="ora817¥examples¥ex4.xsl">books</target>'||
'   </targets>'||
'</OWBXMLRuntime>'
```

この操作によって効率がどの程度アップするかは、選択した分割要素によって異なります。分割要素に基づく分割数が少ない場合は、パフォーマンスが向上するほどメモリー資源を節約できません。

その他のオブジェクトとして保存されている XML 文書

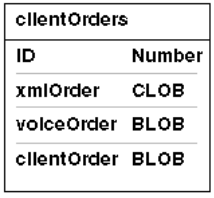
XML ツールキットでは、データベース・オブジェクトとして保存されている文書、または URL で指定されたロケーションに保存されている文書からデータを抽出できます。この項の例では、これらのケースで使用する制御ファイルを紹介します。

CLOB として保存されている文書

このサンプル制御ファイルは、表（clientOrders）に CLOB 列として保存されている XML 文書からデータを抽出し、そのデータを PURCHASE_ORDERS 表にロードします。

clientOrders 表には、マルチメディア情報を含む各注文の詳細情報が保存されています。注文自体は xmlOrder 列に保存されます。voiceOrder 列には注文の音声記録、clientOrder 列には注文した顧客の写真が格納されます。

図 C-1 CLOB として保存されているサンプル文書



XML 文書 clientOrders 表の先頭行に保存されている XML 文書は、最初の例で使用した注文とまったく同じです。注文を定義している XML 文書全体は、C-6 ページを参照してください。

制御ファイル 次の項の制御ファイルは、clientOrders 表の先頭行に保存されている文書からデータを抽出し、そのデータを Purchase_Orders 表にロードするよう XML ツールキットに指示します。

```
'<OWBXMLRuntime>'||
  '<XMLSource>'||
'<CLOB whereClause="where id=''1''">'||
     '<table>clientOrders</table>'||
     '<CLOBColumn>xmlOrder</CLOBColumn>'||
'</CLOB>'||
'</XMLSource>'||
'<targets>'||
'<target dateFormat="yyyy.MM.dd">Purchase_Orders</target>'||
'</targets>'||
'</OWBXMLRuntime>'
```

制御ファイルについてのコメント 次は、この制御ファイルについての補足説明です。属性に割り当てる値について説明します。

1. この表の最初の行は、CLOB 要素の属性として指定されます。

   ```
   CLOB whereClause="where id=''1''"
   ```

   clientOrders 表の ID 列は各行を識別します。

2. dateFormat 属性は、ORDER_DATE 要素の解析結果である文字列値のフォーマットを指定します。解析したデータを Purchase_Orders の DATE 型列にロードするには、この情報が必要になります。

3. XML 文書の要素名は Purchase_Orders 表の列名とまったく同じなので、スタイル・シートは必要ありません。両者が異なる場合は、スタイル・シートが必要になります。

4. XML 文書から取得して解析したデータを複数の表に保存する場合は、ターゲット要素を追加します。もちろん、ターゲット表の列名が XML 要素と異なる場合はスタイル・シートが必要になります。

オブジェクト・ベースのアドバンスト・キューとして保存されている文書

このサンプル制御ファイルは、Oracle のアドバンスト・キュー（AQ）に追加された XML 文書からデータを抽出し、そのデータを PURCHASE_ORDERS 表にロードします。

E-Commerce アプリケーションではアドバンスト・キューを使用して、注文の受け付け、処理、ルーティング、完了を自動化できます。その際、キューをアクティブのままにして、キューに注文が追加されるまで待機するよう設定できます。また、キューを非アクティブにするまでの最大待機時間も指定できます。

次の制御ファイルは、newOrders（Object-Based AQ）に定期的に追加される XML 文書からデータを抽出し、そのデータを PURCHASE_ORDERS 表にロードします。この制御ファイルでは、キューに新しいエントリが追加されるまで待機するよう指示し、タイムアウト値（秒数または無制限）を設定しています。CLOBColumn 属性は、ロードする XML が含まれているオブジェクト型の列を指定します。

アドバンスト・キューの詳細は、『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイドおよびリファレンス』を参照してください。

XML 文書 newOrders キューに定期的に追加される XML 文書は、最初の例で使用した注文とまったく同じです。注文を定義している XML 文書全体は、C-6 ページを参照してください。

制御ファイル 次の項の制御ファイルは、newOrdersAQ に定期的に追加される文書からデータを抽出し、そのデータを PURCHASE_ORDERS 表に保存するよう XML ツールキットに指示します。

```
'<OWBXMLRuntime>'||
'<XMLSource>'||
'<AQ wait="WAIT" waitTime="WAIT_FOREVER">'||
'<AQName>newOrders</AQName>'||
'<ObjectType>xmlMessages</ObjectType>'||
'<CLOBColumn>xmlEntry</CLOBColumn>'||
'</AQ>'||
'</XMLSource>'||
'<targets>'||
'<target dateFormat="yyyy.MM.dd">Purchase_Orders</target>'||
'</targets>'||
'</OWBXMLRuntime>');
```

waitTime 属性は、キューにエントリが追加されるまでデキュー・コールを実行しないように設定されています。

行ベースのアドバンスト・キューとして保存されている文書

このサンプル制御ファイルは、Oracle Database のアドバンスト・キュー（AQ）に追加された XML 文書からデータを抽出し、そのデータを PURCHASE_ORDERS 表にロードします。

E-Commerce アプリケーションではアドバンスト・キューを使用して、注文の受け付け、処理、ルーティング、完了を自動化できます。その際、キューをアクティブのままにして、キューに注文が追加されるまで待機するよう設定できます。また、キューを非アクティブにするまでの最大待機時間も指定できます。

次の制御ファイルは、newOrders（RAW AQ）に定期的に追加される XML 文書からデータを抽出し、そのデータを PURCHASE_ORDERS 表にロードします。この制御ファイルでは、キューに新しいエントリが追加されるまで待機するよう指示し、タイムアウト値（秒数または無制限）を設定しています。次の例は、アドバンスト・キュー・ベースのオブジェクトに保存されているキューについて、制御ファイルをリライトする方法を示しています。

アドバンスト・キューの詳細は、『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイドおよびリファレンス』を参照してください。

XML 文書 newOrdersAQ キューに定期的に追加される XML 文書は、最初の例で使用した注文とまったく同じです。注文を定義している XML 文書全体は、C-6 ページを参照してください。

制御ファイル 次の項の制御ファイルは、newOrdersAQ アドバンスト・キューに定期的に追加される文書からデータを抽出し、そのデータを Purchase_Orders 表に保存するよう XML ツールキットに指示します。

```
'<OWBXMLRuntime>'||
'<XMLSource>'||
'<AQ wait="WAIT" waitTime="20">'||
'<AQName>raw_queue</AQName>'||
'</AQ>'||
'</XMLSource>'||
'<targets>'||
          '<target dateFormat="yyyy.MM.dd">Purchase_Orders</target>'||
'</targets>'||
'</OWBXMLRuntime>');
```

この waitTime 属性では、空のキューに対してデキュー・コールが実行されたとき、そのキューに文書が追加されるまで最大 20 秒間待機するよう指定しています。

URL に保存されている文書

このサンプル制御ファイルは、URL で指定された文書からデータを抽出し、そのデータを Books 表にロードします。

```
'<OWBXMLRuntime>'||
'<XMLSource>'||
      '<URL parameterTable="ptable">'||
             ' <![CDATA[http://oracle.com//xmlserv/library]]>'||
      '</URL>'||
'</XMLSource>'||
'<targets>'||
     '<target XSLFile="¥ora817¥examples¥lib.xsl"  ignoreCase="TRUE">Books'||
     '</target>'||
'</targets>'||
'</OWBXMLRuntime>'
```

ターゲット要素には 2 つの属性が含まれています。1 つは、ソースの要素名とターゲットの列名を一致させるためのスタイル・シートを指定し、もう 1 つは大 / 小文字を無視するよう指定しています。XML を動的に生成する場合はパラメータ表を使用します。この場合、パラメータ表の列名はパラメータ名として使用され、データはパラメータ値として使用されます。動的パラメータの作成で使用できるのは、VARCHAR2 列、VARCHAR 列、CHAR 列、NUMBER 列のみです。

制御ファイルの要素名と属性

XML ツールキットの制御ファイルは XML 文書です。次の 2 つの項では、制御ファイルのすべての要素について説明します。最初の項では、制御ファイルの作成時に使用する各要素について説明します。その次の項では、制御ファイルの Document Type Definition（DTD）について説明します。

制御ファイルの要素についての次の説明では、各要素を 3 種類（XML 文書のソースを表す要素、ロード操作のターゲットを表す要素、XML ツールキットのランタイム環境を制御する要素）に分類しています。

ソース要素

表 C-1 は、XML 文書のソース定義で使用するすべての要素とその属性を示しています。

表 C-1 ソース要素

名前	説明
OWBXMLRuntime	制御ファイル文書のルート要素。 属性 : なし
XMLSource	その他の要素を使用してすべてのソースを定義します。 属性 : splitElement ソース XML 文書を複数の文書に分割します。大きな XML 文書をロードする際、必要メモリーを削減する場合に使用します。この属性を使用した場合、DOMParserConfig のすべての情報が無視されます。
file	1 つのファイル内の 1 つの XML 文書をソースとして指定します。 属性 : なし
directory	XML 文書のソースとして 1 つのディレクトリを指定します。mask 属性で指定したマスクと一致するすべての XML 文書が抽出されます。 属性 : mask ディレクトリ内のファイルをフィルタリングします。Windows * マスクのみ使用できます。たとえば、"*.xml" のように指定します。

表 C-1 ソース要素（続き）

名前	説明
AQ	アドバンスト・キューに保存されている XML 文書のソースを指定します。 属性 : consumerName dequeueMode navigation visibility wait waitTime messageID correlation これらの属性は、アドバンスト・キューがサポートしている様々なデキュー・オプションです。詳細は、『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイドおよびリファレンス』を参照してください。
AQFullName	アドバンスト・キューの正式名を指定します。SCHEMA.AQ_NAME または AQ_NAME として指定できます。 属性 : なし
CLOBColumn	CLOB 列を含むアドバンスト・キュー・ベースのオブジェクトを指定します（オブジェクト /CLOB ベースの AQ）。この要素を指定しない場合は、AQ が RAW based AQ であるとみなされます。 属性 : なし
CLOB	データベース表の CLOB 列に保存されている XML 文書のソースを指定します。 属性 : whereClause CLOB 列から XML 文書を取得するための問合せで使用する where 句を指定します。
CLOB.table	CLOB 列を含む表またはビューを指定します。 属性 : なし
CLOBColumn	CLOB 列の名前を指定します。 属性 : なし

表 C-1 ソース要素（続き）

名前	説明
URL	XML 文書が保存されているアドレスを URL として指定します。 属性 : proxy プロキシ・サーバーのポート番号。 parameterTable パラメータに基づき、URL から文書を複数回取得するようツールキットに指示します。パラメータ名は、パラメータ表の列名です。VARCHAR 型、VARCHAR2 型、CHAR 型または NUMBER 型の列のみが読込み対象となります。 lowerParameterNames TRUE の場合はパラメータ名を小文字で入力し、FALSE の場合は大文字で入力します。

ターゲット要素

表 C-2 は、ターゲットへのデータのロードで使用するすべての要素とその属性を示しています。複数の XML 文書から取得し、解析したデータに対して、複数のターゲットを定義できます。

表 C-2 ターゲット要素

名前	説明
ターゲット	すべてのデータベース・ターゲットを指定します。target 要素を使用して個々のターゲットを指定します。 属性 : なし
target	データベース・ロードのターゲットを指定します。 属性 : truncateFirst TRUE の場合、ロードの前にターゲット・オブジェクトが切り捨てられます。 XSLfile 解析済のデータと個々のターゲット列とを関連付けるときに使用する XSL スタイル・シートのアドレス。 XSLRefURL XSL スタイル・シートで指定されている外部参照の解決で使用する URL。 LoadType ロード操作（INSERT、UPDATE または DELETE）を指定します。UPDATE または DELETE を指定した場合は、truncateFirst と truncateBeforeEachLoad が FALSE に設定されます。 次の属性は、『Oracle XML API リファレンス』で定義されているすべての XU 属性です。 ignoreCase commitBatch rowTag（この属性を設定する場合は、VARCHAR を 'NULL' にします） dateFormat batchSize keyColumnList updateColumnList

ランタイム制御

表 C-3 は、ランタイム環境を指定する要素を示しています。

表 C-3 ランタイム制御

名前	説明
runtimeConfig	XML ツールキットのランタイム構成を指定します。 **属性 :** commitAfterLoad TRUE の場合、ロードの終了時にコミットが発行されます。

制御ファイルの Document Type Definition

次の項の Document Type Definition は、XML ツールキットの制御ファイルで使用できるすべての要素とその属性、順序、使用条件を示しています。

```
<!ELEMENT OWBXMLRuntime (XMLSource, targets, runtimeConfig?)>
<!ELEMENT XMLSource ((file | directory | URL | CLOB | AQ ),DOMParserConfig?)>
<!ATTLIST XMLSource splitElement CDATA #IMPLIED>
<!ELEMENT file (#PCDATA)>
<!ELEMENT directory (#PCDATA)>
<!ATTLIST directory mask CDATA "*.xml">
<!ELEMENT CLOB (table, CLOBColumn)>
<!ELEMENT table (#PCDATA)>
<!ATTLIST CLOB whereClause CDATA #IMPLIED>
<!ELEMENT URL (#PCDATA)>
<!ATTLIST URL proxy CDATA #IMPLIED
     proxyPort CDATA "80"
     parameterTable CDATA #IMPLIED
     lowerParameterNames (TRUE | FALSE) "TRUE">
<!ELEMENT AQ (AQName, (ObjectType, CLOBColumn)?)>
<!ELEMENT AQName (#PCDATA)>
<!ELEMENT ObjectType (#PCDATA)>
<!ELEMENT CLOBColumn (#PCDATA)>
<!ATTLIST AQ
     consumerName CDATA #IMPLIED
     dequeueMode (REMOVE | BROWSE | LOCKED) "REMOVE"
     navigation (NEXT_MESSAGE | NEXT_TRANSACTION | FIST_MESSAGE
          "NEXT_MESSAGE"
     visibility (ON_COMMIT | IMMEDIATE) "IMMEDIATE"
     wait (WAIT_FOREVER | WAIT_NONE | WAIT) "WAIT_FOREVER"
     waitTime CDATA #IMPLIED
     messageID CDATA #IMPLIED
     correlation CDATA #IMPLIED>
<!ELEMENT targets (target+)>
```

```
<!ELEMENT target (#PCDATA)>
<!ATTLIST target
      truncateFirst (TRUE | FALSE) "TRUE"
      truncateBeforeEachLoad (TRUE | FALSE) "FALSE"
     XSLFile CDATA #IMPLIED
     XSLRefURL CDATA #IMPLIED
     loadType (INSERT| UPDATE | DELETE) "INSERT"
     ignoreCase (TRUE | FALSE) "TRUE"
     commitBatch CDATA "0"
     rowTag CDATA "ROW"
     dateFormat CDATA "MM/dd/yyyy HH:mm:ss"
     batchSize CDATA "17"
     keyColumnList NMTOKENS #IMPLIED
     updateColumnList NMTOKENS #IMPLIED>
<!ELEMENT runtimeConfig EMPTY>
<!ATTLIST
     runtimeConfig
     commitAfterLoad (TRUE | FALSE) "TRUE"
     batchSize CDATA "10">
<!ELEMENT DOMParserConfig EMPTY>
<!ATTLIST DOMParserConfig
     showWarnings (TRUE | FALSE) "FALSE"
     retainCDATASection (TRUE | FALSE) "FALSE"
     debugMode (TRUE | FALSE) "FALSE"
     preserveWhitespace (TRUE | FALSE) "FALSE"
     validationMode (TRUE | FALSE) "FALSE"
     baseURL CDATA #IMPLIED>
```

参考資料

XML ツールキットでは、いくつかの Oracle XML SDK ソフトウェア・コンポーネントが使用されています。詳細は、次のマニュアルを参照してください。

- 『Oracle Streams アドバンスト・キューイング・ユーザーズ・ガイドおよびリファレンス』
- 『Oracle アプリケーション開発者ガイド - XML』
- 『Oracle XML API リファレンス』

XML 仕様については、次の Web サイトを参照してください。

- http://www.w3.org/XML/
- http://www.XML.com

D

Warehouse Builder Public View

Warehouse Builder では、Design Repository と Runtime Repository の両方について、一連のビューが事前作成されています。これらのビューを Warehouse Builder Public View と呼びます。このビューを使用すると、Design Repository と Runtime Repository に格納されているデータにアクセスできます。この付録では、各 Public View とその説明の一覧を示します。

Warehouse Builder Design Repository の Public View

Warehouse Builder Runtime Repository の Public View

Warehouse Builder Design Repository の Public View

Design Repository には、すべての設計メタデータが含まれます。次の Public View を使用すると、システムの設計に関するデータにアクセスできます。Warehouse Builder Browser では、これらのビューを使用して、メタデータのレポートを作成します。

汎用モデル・ビュー

データ・モデル・ビュー

汎用モデル・ビュー

表 D-1 ALL_IV_ALL_OBJECTS

列名	データ型	説明
OBJECT_ID	NUMBER(9)	オブジェクトの ID。
OBJECT_UOID	VARCHAR2(255)	オブジェクトの UOID。
OBJECT_TYPE	VARCHAR2(4000)	オブジェクトのタイプ。
OBJECT_NAME	VARCHAR2(4000)	オブジェクトの物理名。
BUSINESS_NAME	VARCHAR2(4000)	オブジェクトのビジネス名。
CONTEXT_NAME	VARCHAR2(4000)	オブジェクトの名前。接頭辞は、そのモジュールの名前と（存在する場合は）プロジェクトの名前。
DESCRIPTION	VARCHAR2(4000)	オブジェクトの説明。
PARENT_OBJECT_ID	NUMBER(9)	オブジェクトのコンテナ・オブジェクト ID。ディメンションに対するモジュールや列に対する表などが、コンテナ・オブジェクトになる。
PARENT_OBJECT_TYPE	VARCHAR2(4000)	親オブジェクトのタイプ。
PARENT_OBJECT_NAME	VARCHAR2(4000)	親オブジェクトの名前。
IS_VALID	VARCHAR2(13)	このオブジェクトは有効か？これが意味を持つのは、検証できるオブジェクトのみ。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-2 ALL_IV_OBJECTS

列名	データ型	説明
OBJECT_ID	NUMBER(9)	オブジェクトの ID（このビューと 2.1 ビューとの違いは、2.1 ビューにはこのビューにあるオブジェクトがすべて含まれ、かつ、MCM サービス用にアーカイブされたスナップショット・オブジェクトがすべて含まれること）。
OBJECT_TYPE	VARCHAR2(4000)	オブジェクトのタイプ。
OBJECT_NAME	VARCHAR2(4000)	オブジェクトの物理名。
BUSINESS_NAME	VARCHAR2(4000)	オブジェクトのビジネス名。

表 D-2 ALL_IV_OBJECTS（続き）

列名	データ型	説明
DESCRIPTION	VARCHAR2(4000)	オブジェクトの説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-3 ALL_IV_OBJECT_PROPERTIES

列名	データ型	説明
OBJECT_ID	NUMBER(9)	オブジェクトの ID。
OBJECT_TYPE	VARCHAR2(4000)	オブジェクトのタイプ。
OBJECT_NAME	VARCHAR2(255)	オブジェクトの物理名。
PROPERTY_ID	NUMBER(9)	オブジェクトのプロパティの ID。
PROPERTY_NAME	VARCHAR2(255)	プロパティ名の ID。
PROPERTY_VALUE	VARCHAR2(4000)	プロパティの値。

表 D-4 ALL_IV_MLS_OBJECTS

列名	データ型	説明
OBJECT_ID	NUMBER(9)	オブジェクトの ID（2.2 ビューと同じセットをカバー）。
LANGUAGE_ID	VARCHAR2(255)	言語の ID（OWB により内部で定義済）。言語名を取得する場合は、2.5 ビューと結合させること。
BUSINESS_NAME	VARCHAR2(4000)	オブジェクトのビジネス名。
DESCRIPTION	VARCHAR2(4000)	オブジェクトの説明。

表 D-5 ALL_IV_SUPPORTED_LANGUAGES

列名	データ型	説明
LANGUAGE_ID	VARCHAR2(255)	言語の ID。
LANGUAGE_NAME	VARCHAR2(64)	言語の名前。
ISBASELANGUAGE	VARCHAR2(1)	（EN、FR などの）ベース言語か？

表 D-6 ALL_IV_MODULES

列名	データ型	説明
PROJECT_ID	NUMBER(9)	このモジュールが属するプロジェクトの ID。
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
INFORMATION_SYSTEM_ID	NUMBER(9)	モジュールの ID。
INFORMATION_SYSTEM_NAME	VARCHAR2(255)	モジュールの物理名。
SCHEMA_ID	NUMBER(9)	モジュールの ID（繰返し列、下位互換性の確保のためのみ）。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
BUSINESS_NAME	VARCHAR2(4000)	モジュールのビジネス名。
DESCRIPTION	VARCHAR2(4000)	モジュールの説明。
STATUS	VARCHAR2(40)	モジュールのステータス（dev、QA、prod）。
IS_VALID	VARCHAR2(13)	このモジュールは有効か？
LOCATION_ID	NUMBER(9)	このモジュールに関連付けられているロケーションの ID。
LOCATION_NAME	VARCHAR2(255)	このモジュールに関連付けられているロケーションの名前。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-7 ALL_IV_PROJECTS

列名	データ型	説明
PROJECT_ID	NUMBER(9)	プロジェクトの ID。
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
BUSINESS_NAME	VARCHAR2(4000)	プロジェクトのビジネス名。
DESCRIPTION	VARCHAR2(4000)	プロジェクトの説明。
VERSION_LABEL	VARCHAR2(255)	プロジェクトのバージョン。
IS_VALID	VARCHAR2(13)	このプロジェクトは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。

表 D-7 ALL_IV_PROJECTS（続き）

列名	データ型	説明
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-8 ALL_IV_INFORMATION_SYSTEMS

列名	データ型	説明
PROJECT_ID	NUMBER(9)	このモジュールが属するプロジェクトの ID。
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
INFORMATION_SYSTEM_ID	NUMBER(9)	モジュールの ID。
INFORMATION_SYSTEM_NAME	VARCHAR2(255)	モジュールの物理名。
INFORMATION_SYSTEM_TYPE	VARCHAR2(4000)	モジュールのタイプ。
BUSINESS_NAME	VARCHAR2(4000)	モジュールのビジネス名。
DESCRIPTION	VARCHAR2(4000)	モジュールの説明。
PRODUCT_TYPE	VARCHAR2(255)	モジュールのアプリケーション・タイプ（Oracle アプリケーションやファイル・ベース・アプリケーションなど）。
SYSTEM_TYPE	VARCHAR2(255)	このアプリケーションを保持するシステムのタイプ（PRODUCT_TYPE で表される）。
VERSION_LABEL	NUMBER(9)	モジュールのバージョン。
VENDOR	VARCHAR2(40)	ベンダー名。
DATABASE_LINK	VARCHAR2(40)	このモジュールのデータ記憶域を物理的に指すデータベース・リンクの名前。これが意味を持つのは、データベース・アプリケーションのみ。
INTEGRATOR_NAME	VARCHAR2(255)	このモジュールの外部データ・ソースへのアクセスに使用する OWB インテグレータ・コンポーネント名。
IS_VALID	VARCHAR2(13)	このモジュールは有効か ?
LOCATION_ID	NUMBER(9)	このモジュールに関連付けられているロケーションの ID。
LOCATION_NAME	VARCHAR2(255)	関連付けられているロケーションの物理名。
STATUS	VARCHAR2(17)	ステータス（dev、QA、prod）。

表 D-8 ALL_IV_INFORMATION_SYSTEMS（続き）

列名	データ型	説明
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-9 ALL_IV_INSTALLATIONS

列名	データ型	説明
INSTALLATION_ID	NUMBER(9)	OWB リポジトリの ID。
INSTALLATION_NAME	VARCHAR2(255)	OWB リポジトリの物理名。
BUSINESS_NAME	VARCHAR2(4000)	OWB リポジトリのビジネス名。
DESCRIPTION	VARCHAR2(4000)	OWB リポジトリの説明。
INSTALLED_VERSION	VARCHAR2(40)	OWB リポジトリのバージョン。
RELEASE	VARCHAR2(40)	OWB クライアントのバージョン。
REPOSITORY_MODEL_VERSION	NUMBER(9)	OWB モデルのバージョン。
PUBLIC_VIEW_VERSION	CHAR(5)	OWB Public View のバージョン。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-10 ALL_IV_FILE_MODULES

列名	データ型	説明
PROJECT_ID	NUMBER(9)	このモジュールが属するプロジェクトの ID。
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
INFORMATION_SYSTEM_ID	NUMBER(9)	このファイル・モジュールの ID。
INFORMATION_SYSTEM_NAME	VARCHAR2(255)	このファイル・モジュールの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このファイル・モジュールのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このファイル・モジュールの説明。

表 D-10 ALL_IV_FILE_MODULES（続き）

列名	データ型	説明
PRODUCT_TYPE	VARCHAR2(255)	モジュールのアプリケーション・タイプ（Oracle アプリケーションやファイル・ベース・アプリケーションなど）。
SYSTEM_TYPE	VARCHAR2(255)	このアプリケーションを保持するシステムのタイプ（PRODUCT_TYPE で表される）。
VERSION_LABEL	NUMBER(9)	モジュールのバージョン。
VENDOR	VARCHAR2(40)	ベンダー名。
DATABASE_LINK	VARCHAR2(40)	このモジュールのデータ記憶域を物理的に指すデータベース・リンクの名前。これが意味を持つのは、データベース・アプリケーションのみ。
DIRECTORY	VARCHAR2(4000)	このファイル・モジュールが接続するディレクトリの名前。
INTEGRATOR_NAME	VARCHAR2(255)	このモジュールの外部ファイル・システムへのアクセスに使用する OWB インテグレータ・コンポーネント名。
IS_VALID	VARCHAR2(13)	このモジュールは有効か？
LOCATION_ID	NUMBER(9)	このモジュールに関連付けられているロケーションの ID。
LOCATION_NAME	VARCHAR2(255)	関連付けられているロケーションの物理名。
STATUS	VARCHAR2(17)	ステータス（dev、QA、prod）。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-11 ALL_IV_GATEWAY_MODULES

列名	データ型	説明
PROJECT_ID	NUMBER(9)	このモジュールが属するプロジェクトの ID。
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
INFORMATION_SYSTEM_ID	NUMBER(9)	このモジュールの ID。
INFORMATION_SYSTEM_NAME	VARCHAR2(255)	このモジュールの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このモジュールのビジネス名。

表 D-11 ALL_IV_GATEWAY_MODULES（続き）

列名	データ型	説明
DESCRIPTION	VARCHAR2(4000)	このモジュールの説明。
PRODUCT_TYPE	VARCHAR2(255)	モジュールのアプリケーション・タイプ（Oracle アプリケーションやファイル・ベース・アプリケーションなど）。
SYSTEM_TYPE	VARCHAR2(255)	このアプリケーションを保持するシステムのタイプ（PRODUCT_TYPE で表される）。
VERSION_LABEL	NUMBER(9)	モジュールのバージョン。
VENDOR	VARCHAR2(40)	ベンダー名。
DATABASE_LINK	VARCHAR2(40)	このモジュールのデータ記憶域を物理的に指すデータベース・リンクの名前。これが意味を持つのは、データベース・アプリケーションのみ。
INTEGRATOR_NAME	VARCHAR2(255)	このモジュールの外部データ・システムへのアクセスに使用する OWB インテグレータ・コンポーネント名。
STRONG_TYPE_NAME	VARCHAR2(255)	採用しているゲートウェイ・コンポーネント（Informix や Sybase など）の区別に使用。
IS_VALID	VARCHAR2(13)	このモジュールは有効か？
LOCATION_ID	NUMBER(9)	このモジュールに関連付けられているロケーションの ID。
LOCATION_NAME	VARCHAR2(255)	関連付けられているロケーションの物理名。
STATUS	VARCHAR2(17)	ステータス（dev、QA、prod）。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-12 ALL_IV_PACKAGED_APPS_MODULES

列名	データ型	説明
PROJECT_ID	NUMBER(9)	このモジュールが属するプロジェクトの ID。
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
INFORMATION_SYSTEM_ID	NUMBER(9)	このモジュールの ID（基本的にビューは、Oracle アプリケーションや SAP などに関連する）。

表 D-12 ALL_IV_PACKAGED_APPS_MODULES（続き）

列名	データ型	説明
INFORMATION_SYSTEM_NAME	VARCHAR2(255)	このモジュールの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このモジュールのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このモジュールの説明。
PRODUCT_TYPE	VARCHAR2(255)	モジュールのアプリケーション・タイプ（Oracle アプリケーションやファイル・ベース・アプリケーションなど）。
SYSTEM_TYPE	VARCHAR2(255)	このアプリケーションを保持するシステムのタイプ（PRODUCT_TYPE で表される）。
VERSION_LABEL	NUMBER(9)	モジュールのバージョン。
VENDOR	VARCHAR2(40)	ベンダー名。
DATABASE_LINK	VARCHAR2(40)	このモジュールのデータ記憶域を物理的に指すデータベース・リンクの名前。これが意味を持つのは、データベース・アプリケーションのみ。
INTEGRATOR_NAME	VARCHAR2(255)	このモジュールの外部データ・システムへのアクセスに使用する OWB インテグレータ・コンポーネント名。
IS_VALID	VARCHAR2(13)	このモジュールは有効か？
LOCATION_ID	NUMBER(9)	このモジュールに関連付けられているロケーションの ID。
LOCATION_NAME	VARCHAR2(255)	関連付けられているロケーションの物理名。
STATUS	VARCHAR2(17)	ステータス（dev、QA、prod）。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-13 ALL_IV_PREDEFINED_MODULES

列名	データ型	説明
PROJECT_ID	NUMBER(9)	このモジュールが属するプロジェクトの ID。
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
INFORMATION_SYSTEM_ID	NUMBER(9)	このモジュールの ID（基本的にビューは、Oracle の事前定義済変換やパブリック変換などに関連する）。

表 D-13 ALL_IV_PREDEFINED_MODULES（続き）

列名	データ型	説明
INFORMATION_SYSTEM_NAME	VARCHAR2(255)	このモジュールの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このモジュールのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このモジュールの説明。
PRODUCT_TYPE	VARCHAR2(255)	モジュールのアプリケーション・タイプ（Oracle アプリケーションやファイル・ベース・アプリケーションなど）。
SYSTEM_TYPE	VARCHAR2(255)	このアプリケーションを保持するシステムのタイプ（PRODUCT_TYPE で表される）。
VERSION_LABEL	NUMBER(9)	モジュールのバージョン。
VENDOR	VARCHAR2(40)	ベンダー名。
DATABASE_LINK	VARCHAR2(40)	このモジュールのデータ記憶域を物理的に指すデータベース・リンクの名前。これが意味を持つのは、データベース・アプリケーションのみ。
INTEGRATOR_NAME	VARCHAR2(255)	このモジュールの外部データ・システムへのアクセスに使用する OWB インテグレータ・コンポーネント名。
IS_VALID	VARCHAR2(13)	このモジュールは有効か？
STATUS	VARCHAR2(17)	ステータス（dev、QA、prod）。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-14 ALL_IV_PROCESS_MODULES

列名	データ型	説明
PROJECT_ID	NUMBER(9)	このモジュールが属するプロジェクトの ID。
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
INFORMATION_SYSTEM_ID	NUMBER(9)	このモジュールの ID（基本的にビューは、Oracle プロセス・フロー・モジュールなどに関連する）。
INFORMATION_SYSTEM_NAME	VARCHAR2(255)	このモジュールの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このモジュールのビジネス名。

表 D-14 ALL_IV_PROCESS_MODULES（続き）

列名	データ型	説明
DESCRIPTION	VARCHAR2(4000)	このモジュールの説明。
PRODUCT_TYPE	VARCHAR2(255)	モジュールのアプリケーション・タイプ（Oracle アプリケーションやファイル・ベース・アプリケーションなど）。
SYSTEM_TYPE	VARCHAR2(255)	このアプリケーションを保持するシステムのタイプ（PRODUCT_TYPE で表される）。
VERSION_LABEL	NUMBER(9)	モジュールのバージョン。
VENDOR	VARCHAR2(40)	ベンダー名。
DATABASE_LINK	VARCHAR2(40)	このモジュールのデータ記憶域を物理的に指すデータベース・リンクの名前。これが意味を持つのは、データベース・アプリケーションのみ。
INTEGRATOR_NAME	VARCHAR2(255)	このモジュールの外部データ・システムへのアクセスに使用する OWB インテグレータ・コンポーネント名。
IS_VALID	VARCHAR2(13)	このモジュールは有効か？
LOCATION_ID	NUMBER(9)	このモジュールに関連付けられているロケーションの ID。
LOCATION_NAME	VARCHAR2(255)	関連付けられているロケーションの物理名。
STATUS	VARCHAR2(17)	ステータス（dev、QA、prod）。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-15 ALL_IV_WAREHOUSE_MODULES

列名	データ型	説明
PROJECT_ID	NUMBER(9)	このモジュールが属するプロジェクトの ID。
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
INFORMATION_SYSTEM_ID	NUMBER(9)	このモジュールの ID（基本的にビューは、Oracle ウェアハウス・モジュールなどに関連する）。
INFORMATION_SYSTEM_NAME	VARCHAR2(255)	このモジュールの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このモジュールのビジネス名。

表 D-15 ALL_IV_WAREHOUSE_MODULES（続き）

列名	データ型	説明
DESCRIPTION	VARCHAR2(4000)	このモジュールの説明。
PRODUCT_TYPE	VARCHAR2(255)	モジュールのアプリケーション・タイプ（Oracle アプリケーションやファイル・ベース・アプリケーションなど）。
SYSTEM_TYPE	VARCHAR2(255)	このアプリケーションを保持するシステムのタイプ（PRODUCT_TYPE で表される）。
VERSION_LABEL	NUMBER(9)	モジュールのバージョン。
VENDOR	VARCHAR2(40)	ベンダー名。
DATABASE_LINK	VARCHAR2(40)	このモジュールのデータ記憶域を物理的に指すデータベース・リンクの名前。
INTEGRATOR_NAME	VARCHAR2(255)	このモジュールの外部データ・システムへのアクセスに使用する OWB インテグレータ・コンポーネント名。
IS_VALID	VARCHAR2(13)	このモジュールは有効か？
LOCATION_ID	NUMBER(9)	このモジュールに関連付けられているロケーションの ID。
LOCATION_NAME	VARCHAR2(255)	関連付けられているロケーションの物理名。
STATUS	VARCHAR2(17)	ステータス（dev、QA、prod）。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

データ・モデル・ビュー

表 D-16 ALL_IV_ADVANCED_QUEUES

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このキューが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
QUEUE_ID	NUMBER(9)	このキューの ID。
QUEUE_NAME	VARCHAR2(255)	このキューの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このキューのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このキューの説明。
LOAD_TYPE_ID	NUMBER(9)	ロード・タイプの ID。
LOAD_TYPE_NAME	VARCHAR2(255)	ロード・タイプの名前。
QUEUE_TABLE_NAME	VARCHAR2(40)	キュー表の名前。
IS_VALID	VARCHAR2(13)	このキューは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-17 ALL_IV_ATTR_GROUPS

列名	データ型	説明
DATA_ENTITY_ID	NUMBER(9)	この属性グループが属するデータ・エンティティの ID。
DATA_ENTITY_TYPE	VARCHAR2(4000)	データ・エンティティのタイプ。
DATA_ENTITY_NAME	VARCHAR2(255)	データ・エンティティの物理名。
ATTRIBUTE_GROUP_NAME	VARCHAR2(255)	この属性グループの物理名。
ATTRIBUTE_GROUP_ID	NUMBER(9)	この属性グループの ID。
BUSINESS_NAME	VARCHAR2(4000)	この属性グループのビジネス名。
DESCRIPTION	VARCHAR2(4000)	この属性グループの説明。
ATTRIBUTE_GROUP_TYPE	VARCHAR2(40)	属性グループのタイプ。
UPDATED_ON	DATE	更新のタイムスタンプ。

表 D-17 ALL_IV_ATTR_GROUPS（続き）

列名	データ型	説明
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-18 ALL_IV_ATTR_GROUP_ITEM_USES

列名	データ型	説明
ATTRIBUTE_GROUP_ID	NUMBER(9)	このデータ・アイテムが属する属性グループの ID。
ATTRIBUTE_GROUP_NAME	VARCHAR2(255)	属性グループの名前。
DATA_ITEM_ID	NUMBER(9)	このデータ・アイテムの ID。
DATA_ITEM_TYPE	VARCHAR2(4000)	このデータ・アイテムのタイプ。
DATA_ITEM_NAME	VARCHAR2(255)	このデータ・アイテムの物理名。
POSITION	NUMBER(9)	属性グループにおけるこのデータ・アイテムの位置。

表 D-19 ALL_IV_CHECK_CONSTRAINTS

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このチェック制約が属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
RELATION_ID	NUMBER(9)	このチェック制約が属するリレーション・エンティティの ID。
RELATION_NAME	VARCHAR2(255)	リレーション・エンティティの物理名。
CONSTRAINT_ID	NUMBER(9)	このチェック制約の ID。
CONSTRAINT_NAME	VARCHAR2(255)	このチェック制約の物理名。
BUSINESS_NAME	VARCHAR2(4000)	このチェック制約のビジネス名。
DESCRIPTION	VARCHAR2(4000)	このチェック制約の説明。
CONSTRAINT_TEXT	VARCHAR2(255)	このチェック制約のテキスト表現。
UPDATED_ON	DATE	更新のタイムスタンプ。

表 D-19 ALL_IV_CHECK_CONSTRAINTS（続き）

列名	データ型	説明
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-20 ALL_IV_COLUMNS

列名	データ型	説明
ENTITY_ID	NUMBER(9)	この列が属するデータ・エンティティの ID。
ENTITY_TYPE	VARCHAR2(4000)	データ・エンティティのタイプ。
ENTITY_NAME	VARCHAR2(255)	データ・エンティティの物理名。
COLUMN_ID	NUMBER(9)	この列の ID。
COLUMN_NAME	VARCHAR2(255)	この列の物理名。
BUSINESS_NAME	VARCHAR2(4000)	この列のビジネス名。
DESCRIPTION	VARCHAR2(4000)	この列の説明。
POSITION	NUMBER(9)	データ・エンティティにおけるこの列の位置。
DATA_TYPE	VARCHAR2(255)	この列のデータ型。
LENGTH	NUMBER(9)	この列のデータ長。
PRECISION	NUMBER(9)	この列のデータ精度。
SCALE	NUMBER(9)	この列のデータ・スケール。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-21 ALL_IV_CONSTRAINTS

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	この制約が属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
RELATION_ID	NUMBER(9)	このチェック制約が属するリレーショナル・エンティティの ID。

表 D-21 ALL_IV_CONSTRAINTS（続き）

列名	データ型	説明
RELATION_NAME	VARCHAR2(255)	リレーショナル・エンティティの物理名。
CONSTRAINT_ID	NUMBER(9)	この制約の ID。
CONSTRAINT_NAME	VARCHAR2(255)	この制約の物理名。
CONSTRAINT_TYPE	VARCHAR2(21)	この制約のタイプ（チェック・キー、主キー、外部キー）。
BUSINESS_NAME	VARCHAR2(4000)	この制約のビジネス名。
DESCRIPTION	VARCHAR2(4000)	この制約の説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-22 ALL_IV_CUBES

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このキューブが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
CUBE_ID	NUMBER(9)	このキューブの ID。
CUBE_NAME	VARCHAR2(255)	このキューブの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このキューブのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このキューブの説明。
IS_VALID	VARCHAR2(13)	このキューブは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-23 ALL_IV_CUBE_DIMENSIONS

列名	データ型	説明
CUBE_ID	NUMBER(9)	このディメンションが関連付けられたキューブの ID。
CUBE_NAME	VARCHAR2(255)	キューブの物理名。
DIMENSION_ID	NUMBER(9)	このディメンションの ID。
DIMENSION_NAME	VARCHAR2(255)	このディメンションの物理名。

表 D-24 ALL_IV_CUBE_MEASURES

列名	データ型	説明
CUBE_ID	NUMBER(9)	このメジャーが属するキューブの ID。
CUBE_NAME	VARCHAR2(255)	キューブの物理名。
MEASURE_ID	NUMBER(9)	このメジャーの ID。
MEASURE_NAME	VARCHAR2(255)	このメジャーの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このメジャーのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このメジャーの説明。
POSITION		キューブ内でのこのメジャーの位置。
DATA_TYPE	VARCHAR2(255)	このメジャーのデータ型。
LENGTH	NUMBER(9)	このメジャーのデータ長。
PRECISION	NUMBER(9)	このメジャーのデータ精度。
SCALE	NUMBER(9)	このメジャーのデータ・スケール。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-25 ALL_IV_CUBE_MEASURE_DIM_USES

列名	データ型	説明
CUBE_ID	NUMBER(9)	キューブの ID（このビューが冗長であることに注意。2.23 ビューと 2.24 ビューを結合して作成するもので、後で削除される）。
CUBE_NAME	VARCHAR2(255)	キューブの物理名。
MEASURE_ID	NUMBER(9)	このキューブに属するメジャーの ID。
MEASURE_NAME	VARCHAR2(255)	メジャーの物理名。
DIMENSION_ID	NUMBER(9)	このキューブに関連付けられたディメンションの ID。
DIMENSION_NAME	VARCHAR2(255)	ディメンションの物理名。
DIMENSION_ALIAS	VARCHAR2(255)	ディメンションの別名。

表 D-26 ALL_IV_DIMENSIONS

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このディメンションが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
DIMENSION_ID	NUMBER(9)	このディメンションの ID。
DIMENSION_NAME	VARCHAR2(255)	このディメンションの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このディメンションのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このディメンションの説明。
PLURAL_NAME	VARCHAR2(40)	このディメンションの複数名。
DIMENSION_PREFIX	VARCHAR2(40)	このディメンションの接頭辞。
IS_VALID	VARCHAR2(13)	このディメンションは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-27 ALL_IV_DIM_HIERARCHIES

列名	データ型	説明
DIMENSION_ID	NUMBER(9)	この階層が属するディメンションの ID。
DIMENSION_NAME	VARCHAR2(255)	このディメンションの物理名。
HIERARCHY_ID	NUMBER(9)	この階層の ID。
HIERARCHY_NAME	VARCHAR2(255)	この階層の物理名。
HIERARCHY_PREFIX	VARCHAR2(40)	この階層の接頭辞。
BUSINESS_NAME	VARCHAR2(4000)	この階層のビジネス名。
DESCRIPTION	VARCHAR2(4000)	この階層の説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-28 ALL_IV_DIM_HIERARCHY_LEVELS

列名	データ型	説明
LEVEL_USE_ID	NUMBER(9)	このレベルとこの親レベルが参加しているレベルの関係の ID。
HIERARCHY_ID	NUMBER(9)	このレベルとこの親レベルが属する階層の ID。
HIERARCHY_NAME	VARCHAR2(255)	階層の物理名。
LEVEL_ID	NUMBER(9)	このレベルの ID。
LEVEL_NAME	VARCHAR2(255)	このレベルの物理名。
LEVEL_DESCRIPTION	VARCHAR2(4000)	このレベルの説明。
PARENT_LEVEL_ID	NUMBER(9)	この親レベルの ID。
PARENT_LEVEL_NAME	VARCHAR2(255)	この親レベルの物理名。
POSITION	NUMBER(9)	このレベルの位置。

表 D-29 ALL_IV_DIM_LEVELS

列名	データ型	説明
DIMENSION_ID	NUMBER(9)	このレベルが属するディメンションの ID。
DIMENSION_NAME	VARCHAR2(255)	ディメンションの物理名。
LEVEL_ID	NUMBER(9)	このレベルの ID。
LEVEL_NAME	VARCHAR2(255)	このレベルの物理名。
LEVEL_PREFIX	VARCHAR2(40)	このレベルの接頭辞。
BUSINESS_NAME	VARCHAR2(4000)	このレベルのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このレベルの説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-30 ALL_IV_DIM_LEVEL_ATTRIBUTES

列名	データ型	説明
LEVEL_ID	NUMBER(9)	この属性が属するレベルの ID。
LEVEL_NAME	VARCHAR2(255)	レベルの物理名。
ATTRIBUTE_ID	NUMBER(9)	この属性の ID。
ATTRIBUTE_NAME	VARCHAR2(255)	この属性の物理名。
BUSINESS_NAME	VARCHAR2(4000)	この属性のビジネス名。
DESCRIPTION	VARCHAR2(4000)	この属性の説明。
POSITION		レベル内でのこの属性の位置。
DATA_TYPE	VARCHAR2(255)	この属性のデータ型。
LENGTH	NUMBER(9)	この属性のデータ長。
PRECISION	NUMBER(9)	この属性のデータ精度。
SCALE	NUMBER(9)	この属性のデータ・スケール。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-31 ALL_IV_EXTERNAL_COLUMNS

列名	データ型	説明
ENTITY_ID	NUMBER(9)	この列が属する外部表の ID。
ENTITY_NAME	VARCHAR2(255)	外部表の名前。
COLUMN_ID	NUMBER(9)	この列の ID。
COLUMN_NAME	VARCHAR2(255)	この列の物理名。
BUSINESS_NAME	VARCHAR2(4000)	この列のビジネス名。
DESCRIPTION	VARCHAR2(4000)	この列の説明。
POSITION	NUMBER(9)	外部表内でのこの列の位置。
DATA_TYPE	VARCHAR2(255)	この列のデータ型。
LENGTH	NUMBER(9)	この列のデータ長。
PRECISION	NUMBER(9)	この列のデータ精度。
SCALE	NUMBER(9)	この列のデータ・スケール。
SOURCE_FIELD_ID	NUMBER(9)	この列のマップ先フィールドの ID。
SOURCE_FIELD_NAME	VARCHAR2(255)	ソース・フィールドの物理名。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-32 ALL_IV_EXTERNAL_TABLES

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	この外部表が属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
LOCATION_ID	NUMBER(9)	モジュールの配布先ロケーションの ID。
LOCATION_NAME	VARCHAR2(255)	ロケーションの物理名。
TABLE_ID	NUMBER(9)	外部表の ID。
TABLE_NAME	VARCHAR2(255)	外部表の物理名。
BUSINESS_NAME	VARCHAR2(4000)	外部表のビジネス名。
DESCRIPTION	VARCHAR2(4000)	外部表の説明。
SOURCE_RECORD_ID	NUMBER(9)	この外部表のマップ先レコードの ID。

表 D-32 ALL_IV_EXTERNAL_TABLES（続き）

列名	データ型	説明
SOURCE_RECORD_NAME	VARCHAR2(255)	ソース・レコードの物理名。
SOURCE_FILE_NAME	VARCHAR2(255)	このソース・レコードが属するファイルの物理名。
ACCESS_PARAMETERS	VARCHAR2(4000)	ソース・レコードへのアクセスに使用するパラメータの式。
IS_VALID	VARCHAR2(13)	この外部表は有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-33 ALL_IV_FOREIGN_KEYS

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	この外部キーが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
ENTITY_ID	NUMBER(9)	この外部キーが属するデータ・エンティティの ID。
ENTITY_NAME	VARCHAR2(255)	データ・エンティティの物理名。
ENTITY_TYPE	VARCHAR2(4000)	データ型の種類（表、ビューなど）。
FOREIGN_KEY_ID	NUMBER(9)	この外部キーの ID。
FOREIGN_KEY_NAME	VARCHAR2(255)	この外部キーの物理名。
BUSINESS_NAME	VARCHAR2(4000)	この外部キーのビジネス名。
DESCRIPTION	VARCHAR2(4000)	この外部キーの説明。
KEY_ID	NUMBER(9)	この外部キーに関連付けられているキーの ID。
KEY_NAME	VARCHAR2(255)	キーの物理名。
IS_DISABLED	CHAR(1)	この外部キーは無効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-34 ALL_IV_KEYS

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このキーが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	このモジュールの物理名。
ENTITY_ID	NUMBER(9)	このキーが属するデータ・エンティティの ID。
ENTITY_NAME	VARCHAR2(255)	データ・エンティティの物理名。
ENTITY_TYPE	VARCHAR2(4000)	データ・エンティティの種類（表、ビューなど）。
KEY_ID	NUMBER(9)	このキーの ID。
KEY_NAME	VARCHAR2(255)	このキーの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このキーのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このキーの説明。
IS_PRIMARY	VARCHAR2(9)	このキーは主キーか？
IS_DISABLED	CHAR(1)	このキーは無効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-35 ALL_IV_KEY_COLUMN_USES

列名	データ型	説明
KEY_ID	NUMBER(9)	この列が関連付けられているキーの ID。
KEY_NAME	VARCHAR2(255)	キーの物理名。
KEY_TYPE	VARCHAR2(11)	キーのタイプ（主キー、一意キー、外部キー）。
COLUMN_ID	NUMBER(9)	この列の ID。
COLUMN_NAME	VARCHAR2(255)	この列の物理名。
POSITION	NUMBER(9)	キーを持つこの列の位置。

表 D-36 ALL_IV_MATERIALIZED_VIEWS

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このマテリアライズド・ビューが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
VIEW_ID	NUMBER(9)	このマテリアライズド・ビューの ID。
VIEW_NAME	VARCHAR2(255)	このマテリアライズド・ビューの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このマテリアライズド・ビューのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このマテリアライズド・ビューの説明。
QUERY_TEXT	VARCHAR2(4000)	このマテリアライズド・ビューの問合せ文のテキスト表現。
IS_VALID	VARCHAR2(13)	このマテリアライズド・ビューは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-37 ALL_IV_OBJECT_TYPES

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このオブジェクト・タイプが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
FOLDER_ID	NUMBER(9)	このオブジェクト・タイプが属するフォルダの ID。
FOLDER_NAME	VARCHAR2(255)	フォルダの物理名。
OBJECT_TYPE_ID	NUMBER(9)	このオブジェクト・タイプの ID。
OBJECT_TYPE_NAME	VARCHAR2(255)	このオブジェクト・タイプの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このオブジェクトのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このオブジェクト・タイプの説明。
TYPE	VARCHAR2(40)	このオブジェクト・タイプの種類。
IS_VALID	VARCHAR2(13)	このオブジェクト・タイプは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。

表 D-37 ALL_IV_OBJECT_TYPES（続き）

列名	データ型	説明
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-38 ALL_IV_RECORD_FIELDS

列名	データ型	説明
FIRSTCLASS_OBJECT_ID	NUMBER(9)	このレコード・フィールドが属するファースト・クラス・オブジェクトの ID（通常、この ID は次のリレーショナル ID と同じ）。
FRISTCLASS_OBJECT_NAME	VARCHAR2(255)	ファースト・クラス・オブジェクトの物理名。
RELATION_ID	NUMBER(9)	このレコード・フィールドが属するリレーショナル・エンティティの ID。
RELATION_NAME	VARCHAR2(255)	リレーショナル・エンティティの物理名。
RECORDFIELD_ID	NUMBER(9)	このレコード・フィールドの ID。
RECORDFIELD_NAME	VARCHAR2(255)	このレコード・フィールドの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このレコード・フィールドのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このレコード・フィールドの説明。
POSITION	NUMBER(9)	このレコード・フィールドの位置。
DATA_TYPE	VARCHAR2(255)	このレコード・フィールドのデータ型。
LENGTH	NUMBER(9)	このレコード・フィールドのデータ長。
PRECISION	NUMBER(9)	このレコード・フィールドのデータ精度。
SCALE	NUMBER(9)	このレコード・フィールドのデータ・スケール。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-39 ALL_IV_RELATIONS

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このリレーショナル・エンティティが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
RELATION_ID	NUMBER(9)	このリレーショナル・エンティティの ID。
RELATION_NAME	VARCHAR2(255)	このリレーショナル・エンティティの物理名。
RELATION_TYPE	VARCHAR2(16)	このリレーショナル・エンティティの種類（表、ビュー、順序、マテリアライズド・ビューなど）。
BUSINESS_NAME	VARCHAR2(4000)	このリレーショナル・エンティティのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このリレーショナル・エンティティの説明。
IS_VALID	VARCHAR2(13)	このリレーショナル・エンティティは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-40 ALL_IV_SEQUENCES

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	この順序が属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
SEQUENCE_ID	NUMBER(9)	この順序の ID。
SEQUENCE_NAME	VARCHAR2(255)	この順序の物理名。
BUSINESS_NAME	VARCHAR2(4000)	この順序のビジネス名。
DESCRIPTION	VARCHAR2(4000)	この順序の説明。
IS_VALID	VARCHAR2(13)	この順序は有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-41 ALL_IV_VIEWS

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このビューが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
VIEW_ID	NUMBER(9)	このビューの ID。
VIEW_NAME	VARCHAR2(255)	このビューの物理名。
QUERY_TEXT	VARCHAR2(4000)	このビューの問合せのテキスト表現。
BUSINESS_NAME	VARCHAR2(4000)	このビューのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このビューの説明。
IS_VALID	VARCHAR2(13)	このビューは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-42 ALL_IV_TABLES

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	この表が属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
TABLE_ID	NUMBER(9)	この表の ID。
TABLE_NAME	VARCHAR2(255)	この表の物理名。
BUSINESS_NAME	VARCHAR2(4000)	この表のビジネス名。
DESCRIPTION	VARCHAR2(4000)	この表の説明。
IS_VALID	VARCHAR2(13)	この表は有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

実装モデル・ビュー

表 D-43 ALL_IV_CUBE_IMPLS

列名	データ型	説明
IMPLEMENTATION_ID	NUMBER(9)	このキューブの ID（この列は後で更新される）。
ITEM_ID	NUMBER(9)	このキューブに属するアイテムの ID。
ITEM_TYPE	VARCHAR2(18)	2 つのタイプ : キューブ・メジャーと、ディメンションをポイントする外部キー（キューブ・ディメンション用）。
ITEM_NAME	VARCHAR2(255)	アイテムの物理名。
CUBE_ID	NUMBER(9)	このキューブの ID。
CUBE_NAME	VARCHAR2(255)	このキューブの物理名。
DIMENSION_ID	NUMBER(9)	関連付けられているディメンションの ID。
DIMENSION_NAME	VARCHAR2(255)	関連付けられている名前の物理名。
DIMENSION_ALIAS	VARCHAR2(255)	関連付けられているディメンションの別名（キューブ内の外部キーの名前）。
COLUMN_ID	NUMBER(9)	アイテムの実装列の ID。
COLUMN_NAME	VARCHAR2(255)	アイテムの実装列の物理名。
POSITION	NUMBER	実装列の位置。
TABLE_ID	NUMBER(9)	このキューブの実装表の ID。
TABLE_NAME	VARCHAR2(255)	実装表の物理名。
FOREIGN_KEY_ID	NUMBER(9)	ディメンションを指す外部キーの ID。
FOREIGN_KEY_NAME	VARCHAR2(255)	外部キーの物理名。
DIM_IMPLEMENTATION_ID	NUMBER(9)	現在値。NULL に設定されているが、後で更新される。

表 D-44 ALL_IV_DIM_IMPLS

列名	データ型	説明
IMPLEMENTATION_ID	NUMBER(9)	このディメンションに属するアイテムの ID（この列は後で更新される）。
ITEM_ID	NUMBER(9)	このディメンションに属するアイテムの ID。
ITEM_TYPE	VARCHAR2(18)	アイテムのタイプ（固定値 : レベル属性）。
ITEM_NAME	VARCHAR2(255)	アイテムの物理名。

表 D-44 ALL_IV_DIM_IMPLS（続き）

列名	データ型	説明
DIMENSION_ID	NUMBER(9)	ディメンションの ID。
DIMENSION_NAME	VARCHAR2(255)	ディメンションの物理名。
COLUMN_ID	NUMBER(9)	アイテムの実装列の ID。
COLUMN_NAME	VARCHAR2(255)	実装列の物理名。
POSITION	NUMBER	列の位置。
TABLE_ID	NUMBER(9)	このディメンションの実装表の ID。
TABLE_NAME	VARCHAR2(255)	このディメンションの実装表の物理名。

表 D-45 ALL_IV_DIM_LEVEL_IMPLS

列名	データ型	説明
IMPLEMENTATION_ID	NUMBER(9)	このレベルに属するアイテムの ID（この列は後で更新される）。
ITEM_ID	NUMBER(9)	このレベルに属するアイテムの ID。
ITEM_TYPE	VARCHAR2(18)	アイテムのタイプ（固定値 : レベル属性）。
ITEM_NAME	VARCHAR2(255)	アイテムの物理名。
DIMENSION_ID	NUMBER(9)	ディメンションの ID。
DIMENSION_NAME	VARCHAR2(255)	ディメンションの物理名。
LEVEL_ID	NUMBER(9)	このレベルの ID。
LEVEL_NAME	VARCHAR2(255)	このレベルの物理名。
COLUMN_ID	NUMBER(9)	実装列の ID。
COLUMN_NAME	VARCHAR2(255)	実装列の物理名。
POSITION	NUMBER	列の位置。
TABLE_ID	NUMBER(9)	このレベルの実装表の ID。
TABLE_NAME	VARCHAR2(255)	このレベルの実装表の物理名。

表 D-46 ALL_IV_FUNCTION_IMPLS

列名	データ型	説明
FUNCTION_ID	NUMBER(9)	ファンクションの ID。
FUNCTION_NAME	VARCHAR2(255)	ファンクションの物理名。
FUNCTION_ IMPLEMENTATION_ID	NUMBER(9)	このファンクションのファンクション・インプリメンテーションの ID。
FUNCTION_ IMPLEMENTATION_NAME	VARCHAR2(255)	ファンクション・インプリメンテーションの物理名。
LANGUAGE	VARCHAR2(255)	実装に使用されている言語の名前。
SCRIPT	VARCHAR2(4000)	このファンクションの実装スクリプト。
BUSINESS_NAME	VARCHAR2(4000)	この実装のビジネス名。
DESCRIPTION	VARCHAR2(4000)	この実装の説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

フラット・ファイル・ビュー

表 D-47 ALL_IV_FIELDS

列名	データ型	説明
RECORD_ID	NUMBER(9)	このフィールドが属するレコードの ID。
RECORD_NAME	VARCHAR2(255)	レコードの物理名。
FIELD_ID	NUMBER(9)	このフィールドの ID。
FIELD_NAME	VARCHAR2(255)	このフィールドの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このフィールドのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このフィールドの説明。
POSITION	NUMBER(9)	このフィールドの位置。
DATA_TYPE	VARCHAR2(255)	このフィールドのデータ型。
LENGTH	NUMBER(9)	このフィールドのデータ長。
PRECISION	NUMBER(9)	このフィールドのデータ精度。
SCALE	NUMBER(9)	このフィールドのデータ・スケール。
PICTURE	VARCHAR2(40)	フィールドのピクチャ。
SIGN_TYPE	NUMBER(9)	フィールドのサイン・タイプ。
USAGE	VARCHAR2(40)	フィールドの使用方法。
MASK	VARCHAR2(255)	フィールドのマスク。
NULLIF	VARCHAR2(40)	フィールドの NULLIF 値。
DEFAULTIF	VARCHAR2(40)	フィールドの DEFAULTIF 値。
SQL_DATA_TYPE	VARCHAR2(40)	フィールドの SQL データ型。
SQL_LENGTH	NUMBER(9)	フィールドの SQL データ長。
SQL_PRECISION	NUMBER(9)	フィールドの SQL 精度。
SQL_SCALE	NUMBER(9)	フィールドの SQL データ・スケール。
START_POSITION	VARCHAR2(40)	フィールドの開始位置。
OCCURS	NUMBER(9)	構造内の属性配列の発生（次の列で識別される）。
STRUCTURE_ID	NUMBER(9)	フィールドを含む構造の ID。
STRUCTURE_NAME	VARCHAR2(255)	構造の物理名。
UPDATED_ON	DATE	更新のタイムスタンプ。

表 D-47 ALL_IV_FIELDS（続き）

列名	データ型	説明
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-48 ALL_IV_FILES

列名	データ型	説明
INFORMATION_SYSTEM_ID	NUMBER(9)	このファイルが属するモジュールの ID。
INFORMATION_SYSTEM_NAME	VARCHAR2(255)	モジュールの物理名。
FILE_ID	NUMBER(9)	このファイルの ID。
FILE_NAME	VARCHAR2(255)	このファイルの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このファイルのビジネス名。
DESCRIPTION	VARCHAR2(4000)	ファイルの説明。
FILE_FORMAT	VARCHAR2(10)	このファイルのフォーマット。
IS_VALID	VARCHAR2(13)	このファイルは有効か？
RECORD_CLASSIFIER_POSITION	NUMBER(9)	このファイルのレコード Classifier の場所。
RECORD_CLASSIFIER_LENGTH	NUMBER(9)	このファイルのレコード Classifier の長さ。
RECORD_SIZE	VARCHAR2(40)	このファイルのレコード・サイズ。
N_PHYSICAL_RECORDS_IN_LOGICAL	NUMBER(9)	1 論理レコード当たりの物理レコード数。
CONTINUATION_AT_END	CHAR(1)	終了時に継続するかどうか。
CONTINUATION_DELIMITER	VARCHAR2(40)	継続の区切り文字記号。
RECORD_DELIMITER	VARCHAR2(40)	レコード・デリミタ記号。
FIELD_DELIMITER	VARCHAR2(40)	フィールド・デリミタ記号。
TEXT_START_DELIMITER	VARCHAR2(1)	テキスト開始区切り文字記号。
TEXT_END_DELIMITER	VARCHAR2(1)	テキスト終了区切り文字記号。
SOURCE_FROM	VARCHAR2(4000)	このファイルのディレクトリ・パス。
UPDATED_ON	DATE	更新のタイムスタンプ。

表 D-48 ALL_IV_FILES（続き）

列名	データ型	説明
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-49 ALL_IV_RECORDS

列名	データ型	説明
FILE_ID	NUMBER(9)	このレコードが属するファイルの ID。
FILE_NAME	VARCHAR2(255)	ファイルの物理名。
RECORD_ID	NUMBER(9)	このレコードの ID。
RECORD_NAME	VARCHAR2(255)	このレコードの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このレコードのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このレコードの説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

コレクション・ビュー

表 D-50 ALL_IV_COLLECTIONS

列名	データ型	説明
PROJECT_ID	NUMBER(9)	このコレクションが属するプロジェクトの ID（このビューは古い分類ビューを置換）。
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
COLLECTION_ID	NUMBER(9)	このコレクションの ID。
COLLECTION_NAME	VARCHAR2(255)	このコレクションの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このコレクションのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このコレクションの説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-51 ALL_IV_COLLECTION_REFERENCES

列名	データ型	説明
COLLECTION_ID	NUMBER(9)	この参照が属するコレクションの ID（このビューは古い classification_item ビューを置換）。
COLLECTION_NAME	VARCHAR2(255)	コレクションの物理名。
COLLECTION_REFERENCE_ID	NUMBER(9)	このコレクション参照の ID。
COLLECTION_REFERENCE_TYPE	VARCHAR2(4000)	このコレクション参照のタイプ。
COLLECTION_REFERENCE_NAME	VARCHAR2(255)	コレクション参照の物理名。
BUSINESS_NAME	VARCHAR2(4000)	このコレクション参照のビジネス名。
DESCRIPTION	VARCHAR2(4000)	このコレクション参照の説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

ファンクション・モデル・ビュー

表 D-52 ALL_IV_FUNCTIONS

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このファンクションが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
FUNCTION_LIBRARY_ID	NUMBER(9)	このファンクションが属するファンクション・ライブラリの ID。
FUNCTION_LIBRARY_NAME	VARCHAR2(255)	ファンクション・ライブラリの物理名。
FUNCTION_ID	NUMBER(9)	このファンクションの ID。
FUNCTION_NAME	VARCHAR2(255)	このファンクションの物理名。
FUNCTION_TYPE	VARCHAR2(13)	このファンクションのタイプ（ファンクション、プロシージャ、テーブル・ファンクション）。
SIGNATURE	VARCHAR2(4000)	このファンクションのシグネチャ。
IS_VALID	VARCHAR2(13)	このファンクションは有効か？
BUSINESS_NAME	VARCHAR2(4000)	このファンクションのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このファンクションの説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-53 ALL_IV_FUNCTION_LIBRARIES

列名	データ型	説明
INFORMATION_SYSTEM_ID	NUMBER(9)	このファンクション・ライブラリが属するモジュールの ID。
INFORMATION_SYSTEM_NAME	VARCHAR2(255)	モジュールの物理名。
FUNCTION_LIBRARY_ID	NUMBER(9)	このファンクション・ライブラリの ID。
FUNCTION_LIBRARY_NAME	VARCHAR2(255)	このファンクション・ライブラリの物理名。
FUNCTION_LIBRARY_TYPE	VARCHAR2(40)	このファンクション・ライブラリのタイプ。
IS_VALID	VARCHAR2(13)	このファンクション・ライブラリは有効か？
BUSINESS_NAME	VARCHAR2(4000)	このファンクション・ライブラリのビジネス名。

表 D-53 ALL_IV_FUNCTION_LIBRARIES（続き）

列名	データ型	説明
DESCRIPTION	VARCHAR2(4000)	このファンクション・ライブラリの説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-54 ALL_IV_FUNCTION_PARAMETERS

列名	データ型	説明
FUNCTION_ID	NUMBER(9)	このパラメータが属するファンクションの ID。
FUNCTION_NAME	VARCHAR2(255)	ファンクションの物理名。
PARAMETER_ID	NUMBER(9)	このパラメータの ID。
PARAMETER_NAME	VARCHAR2(255)	このパラメータの物理名。
PARAMETER_TYPE	VARCHAR2(40)	このパラメータのタイプ。
BUSINESS_NAME	VARCHAR2(4000)	このパラメータのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このパラメータの説明。
POSITION	NUMBER(9)	ファンクション内でのこのパラメータの位置。
DATA_TYPE	VARCHAR2(255)	このパラメータのデータ型。
LENGTH	NUMBER(9)	このパラメータのデータ長。
PRECISION	NUMBER(9)	このパラメータのデータ精度。
SCALE	NUMBER(9)	このパラメータのデータ・スケール。
DEFAULT_VALUE	VARCHAR2(4000)	このパラメータのデフォルト値。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-55 ALL_IV_TABLE_FUNCTIONS

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このテーブル・ファンクションが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
FUNCTION_LIBRARY_ID	NUMBER(9)	このテーブル・ファンクションが属するファンクション・ライブラリの ID。
FUNCTION_LIBRARY_NAME	VARCHAR2(255)	ファンクション・ライブラリの物理名。
FUNCTION_ID	NUMBER(9)	このテーブル・ファンクションの ID。
FUNCTION_NAME	VARCHAR2(255)	このテーブル・ファンクションの物理名。
FUNCTION_TYPE	VARCHAR2(13)	このテーブル・ファンクションのタイプ（固定値 : テーブル・ファンクション）。
SIGNATURE	VARCHAR2(4000)	このテーブル・ファンクションのシグネチャ。
IS_VALID	VARCHAR2(13)	このテーブル・ファンクションは有効か ?
BUSINESS_NAME	VARCHAR2(4000)	このテーブル・ファンクションのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このテーブル・ファンクションの説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

構成モデル・ビュー

表 D-56 ALL_IV_OBJECT_CONFIGURATIONS

列名	データ型	説明
CONFIGURED_OBJECT_ID	NUMBER(9)	構成されているオブジェクトの ID。
CONFIGURED_OBJECT_NAME	VARCHAR2(255)	オブジェクトの物理名。
CONFIGURED_OBJECT_TYPE	VARCHAR2(4000)	オブジェクトのタイプ。
CONFIGURATION_PARAMETER_KEY	VARCHAR2(128)	構成パラメータのキー。
CONFIGURATION_PARAMETER_NAME	VARCHAR2(64)	構成パラメータの名前。
PARAMETER_NLSKEY	VARCHAR2(64)	パラメータの National Language Support（NLS）キー
CONFIGURATION_PARAMETER_TYPE	CHAR(23)	構成パラメータのタイプ。
ARGUMENT	VARCHAR2(128)	構成パラメータの値。
GROUP_NAME	VARCHAR2(322)	構成グループの名前。
GROUP_NLSKEY	VARCHAR2(64)	構成グループの National Language Support（NLS）キー
LANGUAGE	VARCHAR2(64)	この構成に使用されている言語の名前。

配布モデル・ビュー

表 D-57 ALL_IV_CONNECTORS

列名	データ型	説明
LOCATION_ID	NUMBER(9)	このコネクタを所有するロケーションの ID。
LOCATION_NAME	VARCHAR2(255)	ロケーションの物理名。
CONNECTOR_ID	NUMBER(9)	このコネクタの ID。
CONNECTOR_NAME	VARCHAR2(255)	このコネクタの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このコネクタのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このコネクタの説明。
REFERENCED_LOCATION_ID	NUMBER(9)	このコネクタが参照するロケーションの ID。
REFERENCED_LOCATION_NAME	VARCHAR2(255)	このコネクタが参照するロケーションの物理名。
IS_VALID	VARCHAR2(13)	このコネクタは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-58 ALL_IV_LOCATIONS

列名	データ型	説明
PROJECT_ID	NUMBER(9)	このロケーションが属するプロジェクトの ID。
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
LOCATION_ID	NUMBER(9)	このロケーションの ID。
LOCATION_NAME	VARCHAR2(255)	このロケーションの物理名。
LOCATION_TARGET_TYPE	VARCHAR2(40)	このロケーションのターゲット・タイプ。
LOCATION_TARGET_VERSION	VARCHAR2(40)	このロケーションのターゲット・バージョン。
APPLICATION_TYPE	VARCHAR2(255)	接続先ロケーションのアプリケーション・タイプ。
SYSTEM_TYPE	VARCHAR2(255)	この接続先ロケーションのシステム・タイプ。
IS_VALID	VARCHAR2(13)	このロケーションは有効か？
BUSINESS_NAME	VARCHAR2(4000)	このロケーションのビジネス名。

表 D-58 ALL_IV_LOCATIONS（続き）

列名	データ型	説明
DESCRIPTION	VARCHAR2(4000)	このロケーションの説明。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-59 ALL_IV_RUNTIME_REPOSITORIES

列名	データ型	説明
PROJECT_ID	NUMBER(9)	この Runtime Repository が属するプロジェクトの ID（ランタイム・ロケーション、または単にロケーションとも呼ぶ）
PROJECT_NAME	VARCHAR2(255)	プロジェクトの物理名。
LOCATION_ID	NUMBER(9)	このランタイム・ロケーションの ID。
LOCATION_NAME	VARCHAR2(255)	このランタイム・ロケーションの物理名。
LOCATION_TYPE	VARCHAR2(255)	このランタイム・ロケーションのタイプ。
APPLICATION_TYPE	VARCHAR2(255)	このロケーションが接続されているアプリケーションのタイプ。
SYSTEM_TYPE	VARCHAR2(255)	このロケーションが接続されているシステムのタイプ。
BUSINESS_NAME	VARCHAR2(4000)	このランタイム・ロケーションのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このランタイム・ロケーションの説明。
HOST	VARCHAR2(40)	このロケーションの接続のホスト名。
SERVICE_NAME	VARCHAR2(40)	このロケーションの接続のサービス名。
PORT	NUMBER(9)	このロケーションの接続のポート。
USERNAME	VARCHAR2(40)	このロケーションの接続のユーザー名。
SCHEMA	VARCHAR2(40)	このロケーションの接続のスキーマ名。
IS_VALID	VARCHAR2(13)	このランタイム・ロケーションは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

マッピング・モデル・ビュー

表 D-60 ALL_IV_XFORM_MAPS

列名	データ型	説明
INFORMATION_SYSTEM_ID	NUMBER(9)	このマップが属するモジュールの ID。
INFORMATION_SYSTEM_NAME	VARCHAR2(255)	モジュールの物理名。
MAP_ID	NUMBER(9)	このマップの ID。
MAP_NAME	VARCHAR2(255)	このマップの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このマップのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このマップの説明。
COMPOSITE_MAP_COMPONENT_ID	NUMBER(9)	このマップの ID（この列は冗長。後で削除される）。
COMPOSITE_MAP_COMPONENT_NAME	VARCHAR2(255)	このマップの物理名（この列は冗長。後で削除される）。
IS_VALID	VARCHAR2(13)	このマップは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-61 ALL_IV_XFORM_MAP_COMPONENT

列名	データ型	説明
MAP_ID	NUMBER(9)	このマップ・コンポーネントが属するマップの ID。
MAP_NAME	VARCHAR2(255)	マップの物理名。
MAP_COMPONENT_ID	NUMBER(9)	このマップ・コンポーネントの ID（マップ演算子とも呼ぶ）。
MAP_COMPONENT_NAME	VARCHAR2(255)	このマップ・コンポーネントの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このマップ・コンポーネントのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このマップ・コンポーネントの説明。
OPERATOR_TYPE	VARCHAR2(4000)	このマップ・コンポーネントのタイプ（フィルタ、結合子、表など）。

表 D-61 ALL_IV_XFORM_MAP_COMPONENT（続き）

列名	データ型	説明
COMPOSITE_MAP_COMPONENT_ID	NUMBER(9)	マップの ID（この列は冗長。後で削除される）。
COMPOSITE_MAP_COMPONENT_NAME	VARCHAR2(255)	マップの物理名（この列は冗長。後で削除される）。
DATA_ENTITY_ID	NUMBER(9)	このマップ・コンポーネントの調整対象であるデータ・エンティティの ID。
DATA_ENTITY_NAME	VARCHAR2(255)	データ・エンティティの物理名。
DATA_ENTITY_TYPE	VARCHAR2(4000)	データ・エンティティのタイプ。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-62 ALL_IV_XFORM_MAP_DETAILS

列名	データ型	説明
MAP_COMPONENT_ID	NUMBER(9)	マップ・コンポーネントの ID。
PARAMETER_ID	NUMBER(9)	このマップ・コンポーネントが所有するパラメータの ID。
POSITION	NUMBER(9)	パラメータの位置。
BUSINESS_NAME	VARCHAR2(4000)	パラメータのビジネス名。
PARAMETER_NAME	VARCHAR2(255)	パラメータの物理名。
TRANSFORMATION_EXPRESSION	VARCHAR2(4000)	このパラメータの変換のテキスト表現。
DESCRIPTION	VARCHAR2(4000)	このパラメータの説明。
SOURCE_EXPRESSION	VARCHAR2(4000)	このパラメータのデータ・ソース表現。

表 D-63 ALL_IV_XFORM_MAP_PARAMETERS

列名	データ型	説明
MAP_COMPONENT_ID	NUMBER(9)	このパラメータが属するマップ・コンポーネントの ID。
MAP_COMPONENT_NAME	VARCHAR2(255)	マップ・コンポーネントの物理名。
PARAMETER_ID	NUMBER(9)	このパラメータの ID。
PARAMETER_NAME	VARCHAR2(255)	このパラメータの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このパラメータのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このパラメータの説明。
MAP_ID	NUMBER(9)	マップ・コンポーネントを含むマップの ID。
MAP_NAME	VARCHAR2(255)	マップの物理名。
PARAMETER_GROUP_NAME	VARCHAR2(255)	パラメータ・グループ名の物理名。
PARAMETER_GROUP_ID	NUMBER(9)	パラメータ・グループの ID。
PARAMETER_TYPE	VARCHAR2(5)	パラメータのタイプ（IN、OUT、INOUT）。
POSITION	NUMBER(9)	グループ内でのパラメータの位置。
DATA_TYPE	VARCHAR2(40)	パラメータのデータ型。
TRANSFORMATION_EXPRESSION	VARCHAR2(4000)	このパラメータの変換のテキスト表現。
DATA_ITEM_ID	NUMBER(9)	このパラメータの調整対象であるデータ・アイテムの ID。
DATA_ITEM_TYPE	VARCHAR2(40)	データ・アイテムのタイプ。
DATA_ITEM_NAME	VARCHAR2(255)	データ・アイテムの物理名。
SOURCE_PARAMETER_ID	NUMBER(9)	ソース・パラメータの ID（このパラメータの接続元）。
SOURCE_PARAMETER_NAME	VARCHAR2(255)	ソース・パラメータの物理名。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-64 ALL_IV_XFORM_MAP_MAP_PROPERTIES

列名	データ型	説明
MAP_COMPONENT_ID	NUMBER(9)	このプロパティが属するマップ・コンポーネントの ID。
MAP_COMPONENT_NAME	VARCHAR2(255)	マップ・コンポーネントの物理名。
PROPERTY_ID	NUMBER(9)	このプロパティの ID。
PROPERTY_NAME	VARCHAR2(255)	このプロパティの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このプロパティのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このプロパティの説明。
PROPERTY_GROUP_NAME	VARCHAR2(255)	このプロパティ・グループの物理名。
PROPERTY_VALUE	VARCHAR2(4000)	このプロパティの値。

プロセス・フロー・モデル・ビュー

表 D-65 ALL_IV_PACKAGES

列名	データ型	説明
SCHEMA_ID	NUMBER(9)	このプロセス・パッケージが属するモジュールの ID。
SCHEMA_NAME	VARCHAR2(255)	モジュールの物理名。
PACKAGE_ID	NUMBER(9)	このプロセス・パッケージの ID。
PACKAGE_NAME	VARCHAR2(255)	このプロセス・パッケージの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このプロセス・パッケージのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このプロセス・パッケージの説明。
IS_VALID	VARCHAR2(13)	このプロセス・パッケージは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-66 ALL_IV_PROCESSES

列名	データ型	説明
PACKAGE_ID	NUMBER(9)	このプロセスが属するプロセス・パッケージの ID。
PACKAGE_NAME	VARCHAR2(255)	プロセス・パッケージの物理名。
PARENT_PROCESS_ID	NUMBER(9)	このプロセスの親プロセスの ID。
PARENT_PROCESS_NAME	VARCHAR2(255)	このプロセスの親プロセスの物理名。
PROCESS_ID	NUMBER(9)	このプロセスの ID。
PROCESS_NAME	VARCHAR2(255)	このプロセスの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このプロセスのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このプロセスの説明。
BOUND_OBJECT_ID	NUMBER(9)	バウンド・オブジェクトの ID。
BOUND_OBJECT_NAME	VARCHAR2(255)	バウンド・オブジェクトの名前。
IS_VALID	VARCHAR2(13)	このプロセスは有効か？
UPDATED_ON	DATE	更新のタイムスタンプ。

表 D-66 ALL_IV_PROCESSES（続き）

列名	データ型	説明
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-67 ALL_IV_PROCESS_ACTIVITIES

列名	データ型	説明
PROCESS_ID	NUMBER(9)	このアクティビティが属するプロセスの ID。
PROCESS_NAME	VARCHAR2(255)	プロセスの物理名。
ACTIVITY_ID	NUMBER(9)	このプロセス・アクティビティの ID。
ACTIVITY_NAME	VARCHAR2(255)	このアクティビティの物理名。
ACTIVITY_TYPE	VARCHAR2(4000)	このアクティビティのタイプ。
BOUND_OBJECT_ID	NUMBER(9)	バウンド・オブジェクトの ID。
BOUND_OBJECT_NAME	VARCHAR2(255)	バウンド・オブジェクトの名前。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-68 ALL_IV_PROCESS_PARAMETERS

列名	データ型	説明
PARAMETER_OWNER_ID	NUMBER(9)	このパラメータの所有オブジェクトの ID。
PARAMETER_OWNER_NAME	VARCHAR2(255)	このパラメータの所有オブジェクトの物理名。
PARAMETER_OWNER_TYPE	CHAR(14)	所有オブジェクトのタイプ。
PARAMETER_ID	NUMBER(9)	このパラメータの ID。
PARAMETER_NAME	VARCHAR2(255)	このパラメータの物理名。
BUSINESS_NAME	VARCHAR2(4000)	このパラメータのビジネス名。
DESCRIPTION	VARCHAR2(4000)	このパラメータの説明。
POSITION	NUMBER(9)	このパラメータの位置。
DATA_TYPE	VARCHAR2(40)	このパラメータのデータ型。

表 D-68 ALL_IV_PROCESS_PARAMETERS（続き）

列名	データ型	説明
DEFAULT_VALUE	VARCHAR2(4000)	このパラメータのデフォルト値。
DIRECTION	VARCHAR2(3)	このパラメータの方向（IN、OUT）。
IS_FINAL	CHAR(1)	最終プロセスか。
BOUNDDATA_ID	NUMBER(9)	このパラメータのバウンド・データの ID。
BOUNDDATA_NAME	VARCHAR2(255)	バウンド・データの物理名。
BOUNDDATA_TYPE	VARCHAR2(40)	バウンド・データのタイプ。
BOUNDDATA_VALUE	VARCHAR2(4000)	バウンド・データの値。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-69 ALL_IV_PROCESS_TRANSITIONS

列名	データ型	説明
PROCESS_ID	NUMBER(9)	この推移が属するプロセスの ID。
PROCESS_NAME	VARCHAR2(255)	プロセスの物理名。
TRANSITION_ID	NUMBER(9)	この推移の ID。
TRANSITION_NAME	VARCHAR2(255)	この推移の物理名。
BUSINESS_NAME	VARCHAR2(4000)	この推移のビジネス名。
DESCRIPTION	VARCHAR2(4000)	この推移の説明。
CONDITION	VARCHAR2(255)	この推移の条件。
TRANSITION_ORDER	NUMBER(9)	この推移の順序。
SOURCE_ACTIVITY_ID	NUMBER(9)	この推移のソース・アクティビティの ID。
SOURCE_ACTIVITY_NAME	VARCHAR2(255)	ソース・アクティビティの物理名。
TARGET_ACTIVITY_ID	NUMBER(9)	この推移のターゲット・アクティビティの ID。
TARGET_ACTIVITY_NAME	VARCHAR2(255)	ターゲット・アクティビティの物理名。
UPDATED_ON	DATE	更新のタイムスタンプ。

表 D-69 ALL_IV_PROCESS_TRANSITIONS（続き）

列名	データ型	説明
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

表 D-70 ALL_IV_PROCESS_VARIABLES

列名	データ型	説明
PROCESS_ID	NUMBER(9)	この変数が属するプロセスの ID。
PROCESS_NAME	VARCHAR2(255)	プロセスの物理名。
VARIABLE_ID	NUMBER(9)	このプロセス変数の ID。
VARIABLE_NAME	VARCHAR2(255)	このプロセス変数の物理名。
BUSINESS_NAME	VARCHAR2(4000)	このプロセス変数のビジネス名。
DESCRIPTION	VARCHAR2(4000)	このプロセス変数の説明。
POSITION	NUMBER(9)	この変数の位置。
DATA_TYPE	VARCHAR2(40)	この変数のデータ型。
DEFAULT_VALUE	VARCHAR2(4000)	この変数のデフォルト値。
IS_FINAL	CHAR(1)	最終プロセスか。
UPDATED_ON	DATE	更新のタイムスタンプ。
CREATED_ON	DATE	作成のタイムスタンプ。
UPDATED_BY	VARCHAR2(255)	更新者。
CREATED_BY	VARCHAR2(255)	作成者。

Warehouse Builder Runtime Repository の Public View

Runtime Repository には、配布監査データおよび実行監査データがすべて含まれます。Public View を使用すると、このデータにアクセスできます。Runtime Audit Browser では、これらのビューを使用して、監査レポートを作成します。

配布監査ビュー

実行監査ビュー

配布監査ビュー

表 D-71 ALL_RT_AUDIT_LOCATIONS

列名	データ型	説明
location_audit_id	NUMBER (22)	audit_location に対する内部の主キー。
runtime_version	VARCHAR2(64)	ランタイム・バージョンの番号。
client_version	VARCHAR2(64)	Design Client のバージョン番号。
client_repository	VARCHAR2 (30)	クライアント・リポジトリの名前。
client_repository_version	VARCHAR2(64)	クライアント・リポジトリのバージョン番号。
repository_user	VARCHAR2 (30)	Design Repository のユーザー名。
generation_time	DATE	配布が生成された日付。
deployment_audit_id	NUMBER (22)	配布の内部監査 ID。
deployment_sequence_number	NUMBER (10)	配布内のこのロケーションの順序番号。
deployment_audit_name	VARCHAR2(64)	ロケーションの監査名。
deployment_audit_status	VARCHAR2	INACTIVE、READY または COMPLETE。
location_audit_status	VARCHAR2	INACTIVE、READY、BUSY_PREPARE、BUSY_UNPREPARE、BUSY_DEPLOY、BUSY_UNDO、BUSY_FINALIZE または COMPLETE。
location_uoid	VARCHAR2(32)	ロケーションのクライアント UOID。
location_name	VARCHAR2(64)	ロケーションの名前。
location_type	VARCHAR2(64)	ロケーションのタイプ（ODB、OWF、OEM）。
location_type_version	VARCHAR2(64)	ターゲットのバージョン。
number_script_run_errors	NUMBER (10)	検出されたエラーの数。
number_script_run_warnings	NUMBER (10)	検出された警告の数。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。
updated_on	DATE	監査データが更新された時。
updated_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-72 ALL_RT_AUDIT_LOCATION_MESSAGES

列名	データ型	説明
message_audit_id	NUMBER (22)	audit_location_message に対する内部キー。message_line_number とともに使用する場合は主キー。
location_audit_id	NUMBER (22)	audit_location に対する内部キー。
message_severity	VARCHAR2	INFORMATIONAL、WARNING、ERROR または RECOVERY。
message_line_number	NUMBER (10)	1 行メッセージの場合は 1。 複数行メッセージの場合は >0。 （message_audit_id とともに使用する場合は主キーを形成）
message_text	VARCHAR2(4000)	plain_text または nls_key。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-73 ALL_RT_AUDIT_LOCATION_FILES

列名	データ型	説明
file_audit_id	NUMBER (22)	audit_location_file に対する内部の主キー。
location_audit_id	NUMBER (22)	audit_location に対する内部キー。
file_type	VARCHAR2(64)	SQLLoaderLogFile、ShellOutputStream、ShellErrorStream、FTPOutputStream または FTPErrorStream。
file_text	CLOB	ファイルの内容。
format	VARCHAR2	TEXT または HTML。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-74 ALL_RT_AUDIT_OBJECTS

列名	データ型	説明
object_audit_id	NUMBER (22)	audit_object に対する内部の主キー。
parent_object_audit_id	NUMBER (22)	親 audit_script_run に対する内部キー。
location_audit_id	NUMBER (22)	audit_location に対する内部キー。

表 D-74 ALL_RT_AUDIT_OBJECTS（続き）

列名	データ型	説明
location_sequence_number	NUMBER (10)	ロケーション内のこのオブジェクトの順序番号。
object_uoid	VARCHAR2(32)	配布済オブジェクトの UOID。
object_name	VARCHAR2(64)	配布済オブジェクトの名前。
object_type	VARCHAR2(64)	配布済オブジェクトのタイプ（PLSQLMap、Table、Dimension、SQLLoaderControlFile など）。
client_version_tag	VARCHAR2 (80)	このオブジェクトのクライアント・バージョン識別子。
number_script_run_errors	NUMBER (10)	検出されたエラーの数。
number_script_run_warnings	NUMBER (10)	検出された警告の数。
status_when_deployed	VARCHAR2	VALID、INVALID、REMOVED または UNCERTAIN
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。
update_on	DATE	監査データが更新された時。
updated_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-75 ALL_RT_AUDIT_SCRIPT_RUNS

列名	データ型	説明
script_run_audit_id	NUMBER (22)	audit_script_run に対する内部の主キー。
location_audit_id	NUMBER (22)	audit_location に対する内部キー。
object_audit_id	NUMBER (22)	audit_object に対する内部キー。
script_run_audit_status	VARCHAR2	BUSY、COMPLETE、UNCERTAIN、FAILED または INACTIVE。
operation	VARCHAR2	DEPLOY または UNDO。
script_action	VARCHAR2	CREATE、DROP、UPGRADE または REPORT。
script	CLOB	アクションの実行に使用するスクリプト。
script_format	VARCHAR2	TEXT または HTML。
script_generation_time	DATE	スクリプトが作成された時。
number_script_run_errors	NUMBER	検出されたエラーの数。
number_script_run_warnings	NUMBER	検出された警告の数。

表 D-75 ALL_RT_AUDIT_SCRIPT_RUNS（続き）

列名	データ型	説明
elapse_time	NUMBER (10)	経過した秒数。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。
updated_on	DATE	監査データが更新された時。
updated_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-76 ALL_RT_AUDIT_SCRIPT_MESSAGES

列名	データ型	説明
message_audit_id	NUMBER (22)	audit_script_file に対する内部の主キー。
script_run_audit_id	NUMBER (22)	audit_script_run に対する内部キー。
message_severity	VARCHAR2	INFORMATIONAL、WARNING、ERROR または RECOVERY。
message_line_number	NUMBER (10)	1 行メッセージの場合は 1。 複数行メッセージの場合は >0。 （message_audit_id とともに使用する場合は主キーを形成）
message_text	VARCHAR2(4000)	plain_text または nls_key。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-77 ALL_RT_AUDIT_SCRIPT_FILES

列名	データ型	説明
file_audit_id	NUMBER (22)	audit_script_file に対する内部の主キー。
script_run_audit_id	NUMBER (22)	audit_script_run に対する内部キー。
file_type	VARCHAR2(64)	SQLLoaderLogFile、ShellOutputStream、ShellErrorStream、FTPOutputStream または FTPErrorStream。
file_text	CLOB	ファイルの内容。
format	VARCHAR2	TEXT または HTML。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。

実行監査ビュー

表 D-78 ALL_RT_AUDIT_EXECUTIONS

列名	データ型	説明
execution_audit_id	NUMBER (22)	audit_execution に対する内部の主キー。
parent_execution_audit_id	NUMBER (22)	親 audit_execution に対する内部キー。
top_level_execution_audit_id	NUMBER (22)	トップレベルの audit_execution に対する内部キー。
execution_name	VARCHAR2(64)	実行の名前。
task_name	VARCHAR2(64)	実行されたタスクの名前。
task_type	VARCHAR2(64)	実行されたタスクのタイプ（PL/SQL、ProcessFlow）。
task_input	CLOB	タスクの入力ストリーム。
exec_location_uoid	VARCHAR2(32)	実行が行われるロケーションの UOID。
exec_location_name	VARCHAR2(64)	実行が行われるロケーションの名前。
exec_location_type	VARCHAR2(64)	実行が行われるロケーションのタイプ（Runtime Platform、OEM）。
exec_location_type_version	VARCHAR2(64)	実行が行われるロケーションのバージョン。
object_uoid	VARCHAR2(32)	実行されるマッピングのクライアント UOID。
object_name	VARCHAR2(64)	実行されるマッピングの名前。
object_type	VARCHAR2(64)	実行されるマッピングのタイプ。
object_location_uoid	VARCHAR2(32)	マッピングが配布されるロケーション UOID。
object_location_name	VARCHAR2(64)	マッピングが配布されるロケーション名。
object_location_type	VARCHAR2(64)	マッピングが配布されるロケーション・タイプ。
object_location_type_version	VARCHAR2(64)	マッピングが配布されるロケーションのバージョン。
return_result	VARCHAR2(64)	FAILURE、OK、OK_WITH_WARNINGS または OK_WITH_ERRORS。
return_code	NUMBER (10)	<0 : 失敗 >= 0 : 成功
execution_audit_status	VARCHAR2	INACTIVE、BUSY、READY または COMPLETE。
elapse_time	NUMBER (10)	経過した秒数。
number_task_errors	NUMBER (10)	検出されたエラーの数。

表 D-78 ALL_RT_AUDIT_EXECUTIONS（続き）

列名	データ型	説明
number_task_warnings	NUMBER (10)	検出された警告の数。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。
updated_on	DATE	監査データが更新された時。
updated_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-79 ALL_RT_AUDIT_EXECUTION_PARAMS

列名	データ型	説明
parameter_audit_id	NUMBER (22)	audit_execution_param に対する内部の主キー。
execution_audit_id	NUMBER (22)	audit_execution に対する内部キー。
custom_parameter_uoid	VARCHAR2(32)	カスタム・パラメータの UOID。
parameter_name	VARCHAR2(64)	パラメータの名前。
parameter_type	VARCHAR2	BOOLEAN、CHAR、DATE、FLOAT、NUMBER、VARCHAR、VARCHAR2、OPERATING_MODE または AUDIT_LEVEL。
parameter_kind	VARCHAR2	SYSTEM または CUSTOM。
parameter_mode	VARCHAR2	IN、OUT または INOUT。
value_kind	VARCHAR2 (12)	INPUT VALUE または OUTPUT VALUE。
value	VARCHAR2(4000)	パラメータ値の文字表現。

表 D-80 ALL_RT_AUDIT_EXEC_MESSAGES

列名	データ型	説明
message_audit_id	NUMBER (22)	audit_exec_message に対する内部キー。message_line_number とともに使用する場合は主キー。
execution_audit_id	NUMBER (22)	audit_execution に対する内部キー。
message_severity	VARCHAR2	INFORMATIONAL、WARNING、ERROR または RECOVERY。
message_line_number	NUMBER (10)	1 行メッセージの場合は 1。 複数行メッセージの場合は >0。 （message_audit_id とともに使用する場合は主キーを形成）

表 D-80 ALL_RT_AUDIT_EXEC_MESSAGES（続き）

列名	データ型	説明
message_text	VARCHAR2(4000)	Plain_text または nls_key。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-81 ALL_RT_AUDIT_EXEC_FILES

列名	データ型	説明
file_audit_id	NUMBER (22)	audit_exec_file に対する内部の主キー。
execution_audit_id	NUMBER (22)	audit_execution に対する内部キー。
file_type	VARCHAR2(64)	ファイルのタイプ。
file_text	CLOB	ファイルの内容。
format	VARCHAR2	TEXT または HTML。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-82 ALL_RT_AUDIT_MAP_RUNS

列名	データ型	説明
map_run_id	NUMBER (22)	audit_map_run に対する内部の主キー。
execution_audit_id	NUMBER (22)	audit_execution に対する内部キー。
map_uoid	VARCHAR2(255)	マッピングの UOID。
map_name	VARCHAR2 (80)	マッピングの名前。
map_type	VARCHAR2 (30)	PLSQLMap または SQLLoaderControlFile。
start_time	DATE	マッピングを開始した時。
end_time	DATE	マッピングを終了した時。
elapse_time	NUMBER (10)	経過した秒数。
run_status	VARCHAR2 (8)	RUNNING、FAILURE または COMPLETE。
physical_name	VARCHAR2 (80)	SQL*Loader 実行用 .dat ファイルの完全階層名。
load_date	VARCHAR2 (30)	SQL*Loader 実行のロード日付。
load_time	VARCHAR2 (30)	SQL*Loader 実行のロード時間。
number_errors	NUMBER (10)	検出されたエラーの数。

表 D-82 ALL_RT_AUDIT_MAP_RUNS（続き）

列名	データ型	説明
number_records_selected	NUMBER (10)	ソース表から選択されたレコードの数。
number_records_inserted	NUMBER (10)	ターゲット表に挿入されたレコードの数。
number_records_updated	NUMBER (10)	ターゲット表で更新されたレコードの数。
number_records_deleted	NUMBER (10)	ターゲット表で削除されたレコードの数。
number_records_discarded	NUMBER (10)	SQL*Loader 実行で廃棄されたレコードの数。
number_records_merged	NUMBER (10)	ターゲット表でマージされたレコードの数。
number_records_corrected	NUMBER (10)	ターゲット表で修正されたレコードの数。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。
updated_on	DATE	監査データが更新された時。
updated_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-83 ALL_RT_AUDIT_MAP_RUN_SOURCES

列名	データ型	説明
map_run_id	NUMBER (22)	audit_map_run に対する内部キー。
source_name	VARCHAR2(2000)	ソース表を表すマッピング演算子の名前。
source_dblink	VARCHAR2(2000)	ソース表を表すマッピング演算子のデータベース・リンクの名前。

表 D-84 ALL_RT_AUDIT_MAP_RUN_TARGETS

列名	データ型	説明
map_run_id	NUMBER (22)	audit_map_run に対する内部キー。
target_name	VARCHAR2(2000)	ターゲットを表すマッピング演算子の名前。

表 D-85 ALL_RT_AUDIT_STEP_RUNS

列名	データ型	説明
step_id	NUMBER (22)	audit_step_run に対する内部の主キー。
map_run_id	NUMBER (22)	audit_map_run に対する内部キー。
map_step	NUMBER (22)	手順番号 0 または 1。 PL/SQL マッピングの場合 § 通常は § セット・ベースの実行用には 0、行ベースまたは行ベース・ターゲットの実行用には 1。
step_name	VARCHAR2 (80)	セット・ベースの実行用マッピングの名前か、セット・ベースまたはセット・ベース・ターゲットの実行用マッピング・オブジェクトの名前。
step_type	VARCHAR2(18)	セット・ベース、行ベースまたは行ベース・ターゲット。
start_time	DATE	マッピング・ステップを開始した時。
end_time	DATE	マッピング・ステップを終了した時。
elapse_time	NUMBER (10)	要した秒数。
run_status	VARCHAR2 (8)	RUNNING または COMPLETE。
number_errors	NUMBER (10)	検出されたエラーの数。
number_records_selected	NUMBER (10)	ソース表から選択されたレコードの数。
number_records_inserted	NUMBER (10)	ターゲット表に挿入されたレコードの数。
number_records_updated	NUMBER (10)	ターゲット表で更新されたレコードの数。
number_records_deleted	NUMBER (10)	ターゲット表で削除されたレコードの数。
number_records_discarded	NUMBER (10)	SQL*Loader 実行で廃棄されたレコードの数。
number_records_merged	NUMBER (10)	ターゲット表でマージされたレコードの数。
number_records_corrected	NUMBER (10)	ターゲット表で修正されたレコードの数。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。
updated_on	DATE	監査データが更新された時。
updated_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-86 ALL_RT_AUDIT_STEP_RUN_SOURCES

列名	データ型	説明
step_id	NUMBER (22)	audit_step_run に対する内部キー。
map_run_id	NUMBER (22)	audit_map_run に対する内部キー。
map_step	NUMBER (22)	手順番号 0 または 1。 PL/SQL マッピングの場合 § 通常は § セット・ベースの実行用には 0、行ベースまたは行ベース・ターゲットの実行用には 1。
source_name	VARCHAR2(2000)	ソース表を表すマッピング演算子の名前。
source_dblink	VARCHAR2(2000)	ソース表を表すマッピング演算子のデータベース・リンクの名前。

表 D-87 ALL_RT_AUDIT_STEP_RUN_TARGETS

列名	データ型	説明
step_id	NUMBER (22)	audit_step_run に対する内部キー。
map_run_id	NUMBER (22)	audit_map_run に対する内部キー。
map_step	NUMBER (22)	手順番号 0 または 1。 PL/SQL マッピングの場合 § 通常は § セット・ベースの実行用には 0、行ベースまたは行ベース・ターゲットの実行用には 1。
target_name	VARCHAR2(2000)	ターゲットを表すマッピング演算子の名前。

表 D-88 ALL_RT_AUDIT_MAP_RUN_ERRORS

列名	データ型	説明
run_error_id	NUMBER (22)	map_run_error に対する内部の主キー。
step_id	NUMBER (22)	audit_step_run に対する内部キー。
map_run_id	NUMBER (22)	audit_map_run に対する内部キー。
map_step	NUMBER (22)	手順番号 0 または 1。 PL/SQL マッピングの場合 § 通常は § セット・ベースの実行用には 0、行ベースまたは行ベース・ターゲットの実行用には 1。
cursor_rowkey	NUMBER (22)	カーソルから返される行を識別する値。セット・ベースの実行にエラーが発生した場合は 0。
run_error_number	NUMBER (10)	メッセージ番号。

表 D-88 ALL_RT_AUDIT_MAP_RUN_ERRORS（続き）

列名	データ型	説明
run_error_message	VARCHAR2(2000)	メッセージ・テキスト。
target_name	VARCHAR2 (80)	ターゲットを表すマッピング演算子の名前。
target_column	VARCHAR2 (80)	列名。未知または該当なしの場合は f* 秩 B
statement	VARCHAR2(2000)	INSERT や BATCH INSERT などの値か、PL/SQL 文。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。
updated_on	DATE	監査データが更新された時。
updated_by	VARCHAR2 (30)	データベース・ユーザー名。

表 D-89 ALL_RT_AUDIT_MAP_RUN_TRACE

列名	データ型	説明
trace_id	NUMBER (22)	map_run_trace に対する内部の主キー。
map_run_id	NUMBER (22)	audit_map_run に対する内部キー。
map_step	NUMBER (22)	手順番号 0 または 1。 PL/SQL マッピングの場合 § 通常は § セット・ベースの実行用には 0、行ベースまたは行ベース・ターゲットの実行用には 1。
cursor_rowkey	NUMBER (22)	カーソルから返されるエラー行を識別する値。セット・ベースの実行の場合は 0。
type	VARCHAR2 (30)	NEW（トレースの場合）または ERROR（エラーの場合）。
role	VARCHAR2 (30)	S（ソースの場合）または T（ターゲットの場合）。
action	VARCHAR2 (30)	SELECT などの値か、PL/SQL 文。
table_name	VARCHAR2 (80)	ソース / ターゲット表を表すマッピング演算子の名前。
created_on	DATE	監査データが作成された時。
created_by	VARCHAR2 (30)	データベース・ユーザー名。
updated_on	DATE	監査データが更新された時。
updated_by	VARCHAR2 (30)	データベース・ユーザー名。

E

パブリック・オブジェクト

Warehouse Builder アーキテクチャは、ファースト・クラス・オブジェクトやセカンド・クラス・オブジェクトなど、いくつかのクラスのオブジェクトで構成されています。ユーザー定義プロパティやメタデータ・スナップショットは、Oracle Warehouse Builder メタデータを記述するモデルの各種クラス定義オブジェクト上に作成できます。

この付録では、次のトピックについて説明します。

ファースト・クラス・オブジェクトについて

ファースト・クラス・オブジェクト（FCO）は、Warehouse Builder インタフェースから操作できるメタデータ・リポジトリにあるコンポーネントを表します。ファースト・クラス・オブジェクトは、常にではありませんが多くの場合、他のオブジェクトを所有しています。たとえば、ファースト・クラス・オブジェクト TABLE は、次のセカンド・クラス・オブジェクトを所有する場合があります。TABLE_COLUMN、UNIQUE_KEY、FOREIGN_KEY および CHECK_CONSTRAINT。

グラフィカル・ユーザー・インタフェースで Warehouse Builder にアクセスする場合、ファースト・クラス・オブジェクトは大抵、ナビゲーション・ツリーに表示されます。同様に、OMB Plus で Warehouse Builder にアクセスする場合、FCO は OMBCREATE、OMBALTER、OMBRETRIEVE および OMBDELETE コマンドのオブジェクトであると見なすことができます。

セカンド・クラス・オブジェクトについて

セカンド・クラス・オブジェクト（SCO）は、依存オブジェクト・コンポーネントを表します。SCO は必ず別のオブジェクトによって所有されます。また、別のオブジェクトを所有することもできます。たとえば、ファースト・クラス・オブジェクト MAPPING はセカンド・クラス・オブジェクト MAPPING_OPERATOR を所有し、その下に ATTRIBUTES が所有されます。

グラフィカル・ユーザー・インタフェースで Warehouse Builder にアクセスする場合、セカンド・クラス・オブジェクトは大抵、ファースト・クラス・オブジェクトでのみ操作できるオブジェクトのことを指します。同様に、OMB Plus で Warehouse Builder にアクセスする場合、ファースト・クラス・オブジェクトに対するコマンドでのみセカンド・クラス・オブジェクトの定義を操作できます。

サード・クラス・オブジェクトとフォース・クラス・オブジェクトについて

サード・クラス・オブジェクトとフォース・クラス・オブジェクトは、他のオブジェクトに所有されたオブジェクトの相対的な位置付けを表します。所有権が複数レイヤーになるオブジェクトのみに当てはまります。たとえば、DIMENSTION_TABLE（ファースト・クラス・オブジェクト）が INDEX_COLUMN を所有する使用例において、INDEX_COLUMN はセカンド・クラス・オブジェクトになります。しかし、ファースト・クラス・オブジェクト CUBE_TABLE がセカンド・クラス・オブジェクト INDEX を所有し、その下に INDEX_COLUMN が所有されている使用例では、INDEX_COLUMN はサード・クラス・オブジェクトになります。

これらの位置付けの詳細は、E-5 ページの「オブジェクト所有権ツリー」を参照してください。

Warehouse Builder クラス定義オブジェクト

OMBDESCRIBE CLASS_DEFINITION コマンドと OMBREDEFINE CLASS_DEFINITION コマンドの引数として、Warehouse Builder クラス定義オブジェクトを使用できます。

- ACTIVITY
- ACTIVITYPARAMETER
- ADVANCED_QUEUE
- ADVANCED_QUEUE OPERATOR
- AGGREGATOR OPERATOR
- BUSINESS_AREA
- BUSINESS_AREA_ SHORTCUT
- CHECK_ CONSTRAINT
- COLLECTION
- COLUMN
- CONNECTOR
- CUBE OPERATOR
- CUBE_TABLE
- DEDUPLICATOR OPERATOR
- DIMENSION OPERATOR
- DIMENSION_TABLE
- FUNCTION_BASE OPERATOR
- GATEWAY_MODULE
- GROUP
- HIERARCHY
- INDEX
- INDEX_COLUMN
- INPUT_PARAMETER OPERATOR
- JOINER OPERATOR
- KEY_LOOKUP OPERATOR
- LEVEL
- LEVEL_ATTRIBUTE
- LOCATION
- MAPPING
- MATERIALIZED_ VIEW
- MATERIALIZED_ VIEW OPERATOR
- MEASURE
- POSTMAPPING_PROCESS OPERATOR
- PREMAPPING_PROCESS OPERATOR
- PROCESS_DATA
- PROCESS_FLOW
- PROCESS_FLOW_ MODULE
- PROCESS_FLOW_ PACKAGE
- PROJECT
- RECORD
- REF_CURSOR_TYPE
- RUNTIME_REPOSITORY_ CONNECTION
- SAP_MODULE
- SEQUENCE
- SEQUENCE OPERATOR
- SET_OPERATION OPERATOR
- SORTER OPERATOR
- SPLITTER OPERATOR

■ EXPRESSION OPERATOR	■ NAME_AND_ ADDRESS OPERATOR	■ SUBPROCESS
■ EXTERNAL_TABLE	■ OBJECT_TYPE	■ TABLE
■ EXTERNAL_TABLE OPERATOR	■ ORACLE_MODULE	■ TABLE OPERATOR
■ EXTERNAL_TABLE_ COLUMN	■ OUTPUT_ PARAMETER OPERATOR	■ TABLE_FUNCTION OPERATOR
■ FIELD	■ PACKAGE	■ TRANSFORMATION OPERATOR
■ FILTER OPERATOR	■ PARAMETER	■ TRANSITION
■ FLAT_FILE	■ PARAMETER_BASE OPERATOR	■ UNIQUE_KEY
■ FLAT_FILE OPERATOR	■ PIVOT OPERATOR	■ UNPIVOT OPERATOR
■ FLAT_FILE_MODULE	■ PLSQL_RECORD_ TYPE	■ VARIABLES OPERATOR
■ FOREIGN_KEY	■ PLSQL_TABLE_TYPE	■ VIEW
■ FUNCTION	■ PROCEDURE	■ VIEW OPERATOR

データ型の中には次のものが含まれます。

- STRING
- INTEGER
- BOOLEAN
- DATE

オブジェクト所有権ツリー

このツリーを使用すると、ファースト・クラス・オブジェクトやセカンド・クラス・オブジェクトを識別したり、他のオブジェクトを所有するオブジェクトを識別できます。この情報を活用すると、ユーザー定義プロパティやメタデータ変更管理（メタデータ・スナップショット）が理解でき、Oracle Warehouse Builder アーキテクチャがわかるようになります。

次のファースト・クラス・オブジェクトは、他のオブジェクトを所有しません。

- PROJECT
- ORACLE_MODULE
- FLATFILE_MODULE
- SAP_MODULE
- GATEWAY_MODULE
- COLLECTION
- PROCESS_FLOW_MODULE
- PROCESS_FLOW_PACKAGE
- CONNECTOR
- LOCATION
- RUNTIME_REPOSITORY_CONNECTION

次のファースト・クラス・オブジェクトは、他のオブジェクトを所有します。

- **ADVANCED_QUEUE**

  次のセカンド・クラス・オブジェクトが含まれます。

  - OBJECT_TYPE

- **CUBE_TABLE**

  次のセカンド・クラス・オブジェクトが含まれます。

  - COLUMN
  - FOREIGN_KEY
  - UNIQUE_KEY
  - INDEX: サード・クラス・オブジェクトの INDEX_COLUMN が含まれます。
  - PARTITION
  - PARTITION_KEY

- **DIMENSION TABLE**

次のセカンド・クラス・オブジェクトが含まれます。

- COLUMN
- FOREIGN_KEY
- HIERARCHY
- INDEX
- INDEX_COLUMN
- LEVEL: サード・クラス・オブジェクトの FOREIGN_KEY、LEVEL_ATTRIBUTE および UNIQUE_KEY が含まれます。
- UNIQUE_KEY
- PARTITION
- PARTITION_KEY

■ **EXTERNAL TABLE**

次のセカンド・クラス・オブジェクトが含まれます。

- CHECK_CONSTRAINT
- EXTERNAL_TABLE_COLUMN
- FOREIGN_KEY
- UNIQUE_KEY

■ **FLAT_FILE_MODULE**

次のセカンド・クラス・オブジェクトが含まれます。

- FLAT_FILE
- RECORD: サード・クラス・オブジェクトの FIELD が含まれます。
- FIELD

■ **FUNCTION**

次のセカンド・クラス・オブジェクトが含まれます。

- PARAMETER

■ **FUNCTION CATEGORY**

次のセカンド・クラス・オブジェクトが含まれます。

- PLSQL_RECORD_TYPE
- PLSQL_TABLE_TYPE
- REF_CURSOR_TYPE

- **MAPPING**

  次のセカンド・クラス・オブジェクトが含まれます。

  - OPERATOR
  - ADVANCED_QUEUE OPERATOR
  - AGGREGATOR OPERATOR
  - CUBE OPERATOR
  - DATA_ENTITY OPERATOR
  - DATA_ENTITY_KEYS OPERATOR
  - DIMENSION OPERATOR
  - EXPRESSION OPERATOR
  - EXTERNAL_TABLE OPERATOR
  - FILTER OPERATOR
  - FLAT_FILE OPERATOR
  - FUNCTION_BASE OPERATOR
  - INPUT_PARAMETER OPERATOR
  - JOINER OPERATOR
  - KEY_LOOKUP OPERATOR
  - MATERIALIZED_VIEW OPERATOR
  - NAME_AND_ADDRESS OPERATOR
  - PARAMETER_BASE OPERATOR
  - PIVOT OPERATOR
  - POSTMAPPING_PROCESS OPERATOR
  - PREMAPPING_PROCESS OPERATOR
  - SEQUENCE OPERATOR
  - SET_OPERATION OPERATOR
  - SORTER OPERATOR
  - SPLITTER OPERATOR
  - TABLE OPERATOR
  - TABLE_FUNCTION OPERATOR

  - TRANSFORMATION OPERATOR
  - UNPIVOT OPERATOR
  - VARIABLES OPERATOR
  - VIEW OPERATOR: フォース・クラス・オブジェクト・パラメータを含むサード・クラス・オブジェクト・グループが含まれます。
- **MATERIALIZED_VIEW**

  次のセカンド・クラス・オブジェクトが含まれます。
  - CHECK_CONSTRAINT
  - COLUMN
  - FOREIGN_KEY
  - UNIQUE_KEY
  - INDEX: サード・クラス・オブジェクトの INDEX_COLUMN が含まれます。
  - PARTITION
  - PARTITION_KEY
- **PROCEDURE**

  次のセカンド・クラス・オブジェクトが含まれます。
  - PARAMETER
- **PROCESS_FLOW**

  次のセカンド・クラス・オブジェクトが含まれます。
  - SUBPROCESS
  - PROCESS_DATA
  - ACTIVITY: サード・クラス・オブジェクトの ACTIVITYPARAMETER が含まれます。
- **SEQUENCE**

  次のセカンド・クラス・オブジェクトが含まれます。
  - COLUMN
- **TABLE**

  次のセカンド・クラス・オブジェクトが含まれます。
  - CHECK_CONSTRAINT
  - COLUMN

  - FOREIGN_KEY
  - UNIQUE_KEY
  - INDEX: サード・クラス・オブジェクトの INDEX_COLUMN が含まれます。
  - PARTITION
  - PARTITION_KEY
- **VIEW**

  次のセカンド・クラス・オブジェクトが含まれます。
  - CHECK_CONSTRAINT
  - COLUMN
  - FOREIGN_KEY
  - UNIQUE_KEY
  - INDEX: サード・クラス・オブジェクトの INDEX_COLUMN が含まれます。

用語集

ABAP

SAP R/3 システム（ビジネス・アプリケーション・サブシステム）用アプリケーションを開発するためのプログラミング言語。

API

「**Application Program Interface（API）**」を参照。

Application Program Interface（API）

オペレーティング・システムまたはその他のプログラム環境（データベース、Web サーバー、JVM など）と通信するための公開プログラム・インタフェースのセット。言語形式およびメッセージ形式のインタフェースで構成される。これらのメッセージは通常、アプリケーションの開発に使用できる関数やメソッドをコールする。『Oracle Warehouse Builder Java API Reference』は **Oracle MetaBase Plus（OMB Plus）**からアクセスできる。

AQ

「**アドバンスト・キュー（advanced queue : AQ）**」を参照。

AW

「**アナリティック・ワークスペース（analytic workspace）**」を参照。

Common Warehouse Metamodel（CWM）

Oracle データ・ウェアハウス、意思決定支援および **Oracle Warehouse Builder** などの **OLAP** ツールで使用される**リポジトリ**規格。CWM リポジトリ・**スキーマ**は、他の製品で共有できるオープンな規格のリポジトリである。

Connectivity Agent

Generic Connectivity Agent は、Oracle Database **サーバー**に含まれる。この**エージェント**は、**クライアント**・システムにインストールされている ODBC ドライバまたは OLE DB ドライバのルールに従ってデータ転送が行われる場合、低コストのデータ統合を実現するために使用される。

CWM

「**Common Warehouse Metamodel（CWM）**」を参照。

DDL

「**データ定義言語（DDL）**」を参照。

Design Repository

Oracle データベースにインストールされた Design Repository には、**Oracle Warehouse Builder** で使用されるすべてのオブジェクトの**メタデータ**定義が格納される。ここには、作成する**ターゲット**・システムの設計情報がすべて格納される。クライアント・ユーザー・インタフェースや **Oracle MetaBase Plus（OMB Plus）**を使用すると、ここに格納したメタデータにアクセスできる。このリポジトリは、OWB Repository Assistant を使用して作成される（「**Runtime Repository**」と対比）。

DML

「**データ操作言語（DML）**」を参照。

ETL

「**抽出、変換、ロード（extraction, transformation, and loading: ETL）**」を参照。

eXtensible Markup Language（XML）

データを記述するためのオープンな規格の 1 つ。W3C がインターネット用に、SGML 構文のサブセットを使用して開発した。現在の規格はバージョン 1.0 で、1998 年 2 月に W3C 勧告として公開されている。

FCO

「**ファースト・クラス・オブジェクト（First Class Object : FCO）**」を参照。

HTML

「**Hypertext Markup Language（HTML）**」を参照。

Hypertext Markup Language（HTML）

Web ブラウザに送られ、World Wide Web のベースとして動作するファイルを作成するためのマークアップ言語。HTML の次バージョンは、**XML** を応用したもので、xHTML と呼ばれている。

IP アドレス（IP address）

ネットワーク上のノードの識別に使用される。ネットワーク上の各コンピュータには、一意の IP アドレスが割り当てられる。この IP アドレスは、ネットワーク ID と一意のホスト ID で構成される。通常、このアドレスはドット付き 10 進数表記法で表される。つまり、144.45.9.22 のように、8 ビットごとの 10 進数がピリオドで区切られる。

Metadata Loader（MDL）

Oracle Warehouse Builder のユーティリティ。Design Repository をバックアップする。**Design Repository** から .MDL テキスト・**ファイル**に**メタデータのエクスポート**を行うことができ、また、.MDL ファイルから**リポジトリ**に**メタデータのインポート**を行うことができる。必要に応じて、バージョン情報付きでメタデータをエクスポートできる。インポート時には、既存の**メタデータ**・オブジェクトを上書きしたり、他のオプションを使用して既存のメタデータ・オブジェクトと結合することができる。

OLAP

「**オンライン分析処理（Online Analytical Processing : OLAP）**」を参照。「**オンライン・トランザクション処理（Online Transactional Processing : OLTP）**」と対比。

OLTP

「**オンライン・トランザクション処理（Online Transactional Processing : OLTP）**」を参照。「**オンライン分析処理（Online Analytical Processing : OLAP）**」と対比。

Open Database Connectivity（ODBC）

Oracle ODBC ドライバは、1 つのアプリケーションから多数の異なるデータ・ソースへのアクセスを可能にする標準インタフェースを提供する。アプリケーションのソース・コードを、データ・**ソース**ごとに再コンパイルする必要はない。**データベース**・ドライバは、アプリケーションを特定のデータ・ソースにリンクする。データベース・ドライバは、特定のデータ・ソースにアクセスするために必要に応じてアプリケーションから起動できる DLL（Dynamic Link Library）である。したがって、アプリケーションは、データベース・ドライバが存在するデータ・ソースに自由にアクセスできる。

Oracle Enterprise Manager（OEM）

グラフィカルなコンソール、エージェント、標準サービスおよびツールを統合した別の Oracle 製品。Oracle 製品を管理する統合された包括的なシステム管理プラットフォームを実現する。

Oracle MetaBase Plus（OMB Plus）

Oracle Warehouse Builder のスクリプト機能を使用するためのツール。

Oracle MetaBase（OMB）

Oracle Warehouse Builder リポジトリ。バージョン管理、拡張性、マルチユーザー・ロック、コピーおよび貼付けなどのサービスを含む。

Oracle Server

オラクル社が販売する**リレーショナル・データベース管理システム**。Oracle Server のコンポーネントには、カーネルと、DBA および**データベース**・ユーザーが使用する各種ユーティリティが含まれる。

Oracle Transparent Gateway（transparent gateway）

SQL ベースのアプリケーションが、リレーショナルおよび非リレーショナルのデータ・ソースを、Oracle **データベース**と同様の方法でアクセスすることを可能にするソフトウェア製品。

Oracle Universal Installer（OUI）

Oracle **データベース**・ソフトウェアとその関連コンポーネントのインストールを簡単にするグラフィカル・**ユーザー・インタフェース**。

Oracle Warehouse Builder

Oracle Warehouse Builder はビジネス・インテリジェンス・ツールで、エンタープライズ・データ・ウェアハウス、データ・マート、ビジネス・インテリジェンス・アプリケーションの設計や配布を行う統合型ソリューション。これを使用すると、分散しているデータのソースとターゲット間でデータ統合を行う際の複雑な問題を解決できる。加えて、Warehouse Builder には、開発するシステムのライフサイクルを保持するのに必要な機能がすべて用意されている（「**データ・ウェアハウス（data warehouse）**」、「**データ・マート（data mart）**」、「**ソース（source）**」、「**ターゲット（target）**」も参照）。

データの移動と変換、データ・ウェアハウスの開発と実装、**メタデータ**管理、Oracle **データベース**とメタデータの作成および管理を目的とする包括的なツールセット。グラフィカル・**ユーザー・インタフェース**のほかに、Warehouse Builder では、**Oracle MetaBase Plus（OMB Plus）**の形式で **API** を提供する。これにより、OMB Scripting Language を使用して Warehouse Builder の全機能にアクセスすることができる。

Oracle Warehouse Builder Name and Address

この製品は、Oracle Warehouse Builder とは別個にライセンスされており、ユーザー・データをクレンジングする。Oracle Warehouse Builder Name and Address の**演算子**で使用する Name and Address のデータを提供する、サードパーティ製データ・ファイルで構成される。これらのファイルは年 4 回更新され、更新プログラムは MetaLink から入手できる。

ORACLE_HOME

Oracle 製品が実行される環境を指す。この環境には、インストールされた製品ファイルのロケーション、製品のバイナリ・ファイルを指定する `PATH` 変数、レジストリ・エントリ、ネット・サービス名、プログラム・グループなどが含まれる。

Oracle Universal Installer のデフォルト設定を使用して OFA 準拠の**データベース**をインストールすると、`X:¥ORACLE_BASE` の下に Oracle ホーム（このガイドでは `¥ORACLE_HOME`）が配置される。Oracle ホームには、Oracle ソフトウェアの実行可能ファイルとネットワーク・ファイルのサブディレクトリが含まれる（「**サービス名（service name）**」も参照）。

PL/SQL

SQL を拡張する Oracle の第 3 世代の手続き型言語。PL/SQL により、SQL 文とプロシージャ構造を併用できる。PL/SQL は、SQL の扱いやすさと柔軟性に、構造化プログラミング言語の手続き機能（IF...THEN、WHILE、LOOP など）を組み合せたものである。**データベース**に PL/SQL が格納されていなくても、アプリケーションからデータベースに PL/SQL のブロックを送信することができる。個々の SQL 文を送信しないため、ネットワーク・トラフィックの負荷を軽減できる。PL/SQL を使用すると、プロシージャ、ファンクション、パッケージなどの PL/SQL プログラム・ユニットを定義して実行できる。PL/SQL は実行時に解釈および解析されるので、コンパイルは不要である（「**パッケージ（package）**」、「**Structured Query Language（SQL）**」、「**SQL*Plus**」も参照）。

RDBMS

「**リレーショナル・データベース管理システム（relational database management system : RDBMS）**」を参照。

Runtime Platform Service（RTPS）

サービスの**実行**および**配布**を実現する、**Oracle Warehouse Builder** ソフトウェアのサーバー側のコンポーネント。こうしたサービスを実行できるようにするには Runtime Platform Service をアクティブにしておく必要がある。Runtime Platform Service では、マッピングとプロセス・フローの実行が、Warehouse Builder 内から管理され、実行と配布の監査データがすべて、**Runtime Repository** に格納される。リモート実行の場合は、**Oracle Enterprise Manager（OEM）** の Management Server に接続される。Runtime Platform Service は、データベース・ジョブを通じてコールされる。このジョブは**データベース**を起動したときに自動的に起動され、データベースをシャットダウンしたときに自動的にシャットダウンされる。

Runtime Repository

配布と**実行**のデータすべてを格納する **Oracle Warehouse Builder** のリポジトリ。この**リポジトリ**は、Runtime Assistant を使用して作成される（「**Design Repository**」と対比）。

SCO

「**セカンド・クラス・オブジェクト（Second Class Object : SCO）**」を参照。

SQL

「**Structured Query Language（SQL）**」を参照。

SQL*Loader

オペレーティング・システム・ファイルから Oracle の**データベース**表へデータをロードするための Oracle ツール。多量のデータを最も効率的にロードできる（「**Structured Query Language（SQL）**」も参照）。

SQL*Plus

Oracle **データベース**に対する **Structured Query Language（SQL）**文を実行するための Oracle ツール。Oracle SQL には、ANSI/ISO 規格の SQL 言語への数多くの機能拡張が含まれる（「**PL/SQL**」も参照）。

Structured Query Language（SQL）

リレーショナル・データベースへのアクセス用に開発された、業界標準のプログラミング言語。ユーザーが必要な処理を SQL で記述すると、SQL 言語コンパイラによって、**データベース**内を検索して必要なタスクを実行する手続きが自動的に生成される。一般に、SQL 言語には次の特徴がある。

- **行**単位の処理ではなく、データのセットに対する操作をサポートする。
- 物理ロケーションに関係なく、データにアクセスできる。
- 手続き型言語ではない。つまり、取得するデータのみを記述し、そのデータの取得方法は記述しない。

Oracle Database では、SQL と **PL/SQL** がサポートされている（「**SQL*Plus**」も参照）。

SYSDBA

特殊な**データベース**管理**ロール**の 1 つ。ADMIN OPTION と SYSOPER のすべてのシステム権限が含まれる。また、SYSDBA により、CREATE DATABASE アクションの実行と時間ベースのリカバリも可能になる（「**SYSOPER**」も参照）。

SYSOPER

特殊な**データベース**管理**ロール**の 1 つ。これによりデータベース管理者は、STARTUP、SHUTDOWN、ALTER DATABASE OPEN/MOUNT、ALTER DATABASE BACKUP、ARCHIVE LOG および RECOVER を実行できる。また、RESTRICTED SESSION 権限も含まれる（「**SYSDBA**」も参照）。

SYSTEM 表領域（SYSTEM tablespace）

SYSTEM **表領域**は、他の表領域と異なり、Oracle を稼働させるために、表領域に含まれるすべてのデータ・ファイルがオンラインである必要がある。SYSTEM のデータ・ファイルのいずれかでメディア障害が発生した場合、**データベース**をマウントしてリカバリする必要がある。

SYSTEM ユーザー名（SYSTEM username）

各データベースで自動的に作成される 2 つの標準の**データベース**管理者ユーザー名の 1 つ（もう 1 つは **SYS ユーザー名**）。Oracle ユーザーの SYSTEM は、MANAGER というパスワードを持つ。SYSTEM ユーザー名は、データベース管理者がデータベースのメンテナンスの際に優先的に使用する**ユーザー名**である。

SYS ユーザー名（SYS username）

各**データベース**で自動的に作成される、2 つの標準 DBA **ユーザー名**の 1 つ（もう 1 つは **SYSTEM ユーザー名**）。Oracle ユーザーの SYS は、MANAGER というパスワードを持つ。

tnsnames.ora

ネット・サービス名にマッピングされた接続記述子を格納する**ファイル**。このファイルは、すべてのクライアントまたは個々のクライアントでの使用を可能にするため、中央またはローカル管理できる（「**サービス名（service name）**」、「**クライアント（client）**」も参照）。

UDP

「**ユーザー定義プロパティ（user-defined properties : UDP）**」を参照。

XML

「**eXtensible Markup Language（XML）**」を参照。

アクティビティ（activity）

プロセス・フローにおいて、ワークフロー・エンジンで実行される作業単位を表す。これらの作業単位には、**Oracle Warehouse Builder** の内部コンポーネントおよび外部コンポーネントを含めることができる。外部アクティビティの例は、プロセス・フローを続行する前にローカル・マシンやリモート・マシン上に**ファイル**があるかどうかをチェックするファイルが存在アクティビティである。内部アクティビティの例は、同時アクティビティを起動するためのロジックを指定する FORK アクティビティである。

アドバンスト・キュー（advanced queue : AQ）

メッセージ・キューを含むデータベース統合システム。AQ は、エンタープライズ・データ統合の中心的な役割を担っている。AQ を使用すると、アプリケーションの統合に必要なメッセージ管理や通信が可能になる。**Oracle Warehouse Builder** は、ウェアハウス設計のデータ・ソースとターゲットとして AQ をサポートしている。Warehouse Builder アーキテクチャでは、AQ は**ファースト・クラス・オブジェクト**になる（「**ソース（source）**」、「**ターゲット（target）**」も参照）。

アナリティック・ワークスペース（analytic workspace）

OLAP での使用可能な形式で、データを編成および格納するオブジェクトを含む単一**ファイル**。アナリティック・ワークスペースの構造と内容を決定するには、オブジェクト（ディメンション、変数、プログラムなど）を定義する必要がある。アナリティック・ワークスペースのディクショナリにそれらが定義された時点で、OLAP でのデータの入力、変更、使用が可能になる。

アンピボット（unpivot）

マッピングでは、アンピボット**演算子**は、複数の入力行を 1 つの出力**行**に変換する。これにより、ソース・データ内の属性ごとにグループ化されているソース行セットから**ソース**をいったん抽出し、そこから 1 つの行を作成できるようになる（「**ピボット（pivot）**」も参照）。

一時表領域（temporary (TEMP) tablespace）

SQL 文の処理中に作成された一時表または索引の**表領域**。

インスタンス（instance）

Oracle インスタンスは、システム・グローバル領域（SGA）と Oracle バックグラウンド・プロセスの組合せである。**データベース**が起動されるたびに、システム・グローバル領域が割り当てられ、Oracle バックグラウンド・プロセスが起動される。インスタンスがシャットダウンされると、SGA の割当ては解除される。

インテグレータ（integrator）

Oracle Warehouse Builder と協働して、**ソース**・データの定義、設計および**抽出**を容易にするソフトウェア。インテグレータの例には、Oracle Applications Integrator および SAP Integrator がある。

ウェアハウス管理者（Warehouse administrator）

ウェアハウス・**データベース**およびウェアハウス管理アプリケーションを管理する、情報のスペシャリスト。ウェアハウス管理者は、たとえば、ウェアハウス・データベースの周期的な更新を管理および監視する（「**データ・ウェアハウス（data warehouse）**」も参照）。

ウォッチ（watch）

デバッグ・セッション中に使用されるデータ・フロー内のポイント。特定の**演算子**を通過する、またはすでに通過したデータの監視に使用される。

エージェント（agent）

リモート・コンピュータ上で実行され、中央**サーバー**と通信するソフトウェア・ルーチン。中央サーバーは、ローカルで動作するソフトウェアの実行をエージェントに委任できる。Warehouse Builder では、エージェントを使用して、様々なマシン上でソフトウェアをコールできる。

エディタ（editor）

Oracle Warehouse Builder のウィンドウ。オブジェクトやオブジェクト間の関係を定義または編集するために使用される。

エンキュー（enqueue）

エンキューとは、**データベース**・リソースへのアクセスをシリアライズする共有メモリー構造のこと。セッションまたは**トランザクション**と関連付けられる（「**ロック（lock）**」も参照）。

演算子（operator）

マッピングの基本となる設計要素（「**データ・フロー演算子（data flow operator）**」、「**ソース演算子（source operator）**」、「**ターゲット演算子（target operator）**」も参照）。

エンディアン（endian）

1 つのワードにおけるバイトの数値的な配置を表す「バイトの並べ方」のこと。ワードはコンピュータの記憶域の基本単位であり、ワードの長さは 8 ビット、16 ビット、32 ビットまたは 64 ビットにすることができる。ビッグ・エンディアンは、最上位のバイト（桁）が構造の左端に配置される通常の並べ方で、人間が計算を行う際に使用する数値形式である。一部の CPU では、リトル・エンディアンの並べ方でワードを処理する。リトル・エンディアンはビッグ・エンディアンの逆で、最下位のバイト（桁）が左端に配置される。CPU が最下位の桁から数値計算を行うため、リトル・エンディアンの数値は、すでに処理する際の順序で並べられていることになる。

オペレーティング・システム（operating system）

コンピュータのリソースを管理するシステム・ソフトウェア。メモリーの割当てなどの基本的なタスクを実行したり、コンピュータ・コンポーネント間の通信を可能にする。**Oracle Warehouse Builder** は、次のオペレーティング・システムで使用できる。

- Windows: NT、2000、XP
- UNIX: Solaris、Linux、HP-UX

親（parent）

1. **階層**内の所定の値より上の**レベル**にある値。たとえば、時間**ディメンション**における値「Q1-99」は、値「Jan-99」の親である（「**子（child）**」も参照）。
2. **Oracle Warehouse Builder** の**メタデータ**・オブジェクトの場合 : 下位レベルのオブジェクトを所有する、**ナビゲーション・ツリー**のオブジェクト。たとえば、**プロジェクト**は、所有するモジュールの親オブジェクトである。さらに**データベース**・モジュールは、所有する**表**オブジェクトの親オブジェクトである。

オンライン・トランザクション処理（Online Transactional Processing : OLTP）

OLTP システムは、**データベース**に対する更新を特徴としているアプリケーション・システムである。OLTP システムの例として、E-Business システムや ERP アプリケーション（たとえば、Oracle アプリケーションや SAP R/3）などがある。さらに特殊な例としては、電話料金請求システム、クレジット・カード取引システム、航空券予約システムなどがある。OLTP システムは、トランザクション・システムとも呼ばれている。「**オンライン分析処理（Online Analytical Processing : OLAP）**と対比（「**トランザクション（transaction）**」も参照）。

オンライン分析処理（Online Analytical Processing : OLAP）

OLAP 機能は、動的で多次元な履歴データ分析を特徴としており、次のような操作をサポートする。

- ディメンション全体および階層内での計算
- トレンドの分析
- 階層内でのドリルアップおよびドリルダウン
- ディメンションのオリエンテーションを変更するための回転

OLAP ツールは、多次元**データベース**に対して実行できる。また、**リレーショナル・データベース**と直接対話させることもできる「**オンライン・トランザクション処理（Online Transactional Processing : OLTP）**」と対比（「**ドリル（drill）**」、「**階層（hierarchy）**」も参照）。

階層（hierarchy）

データの編成方法として、レベルに順序を付けて使用する**論理構造**。階層はデータ**集計操作**の定義に使用できる。たとえば、時間**ディメンション**では、「月」 **レベル**から「四半期」レベル、さらに「年」レベルでデータを集計する際に階層が使用される。また、階層は、階層内のレベルが集計された合計を示しているかどうかに関係なく、ナビゲーション・**ドリル**・パスの定義にも使用できる。

外部キー（foreign key）

列または列セットの各値が、関連付けられた**表**の一意キーまたは**主キー**内の値と一致することを要求する**整合性制約**。また、Oracle が参照する依存データが変更された場合、そのデータをどのように処理するかを指定する参照整合性アクションも、外部キーの整合性制約によって定義される。

外部結合（outer join）

複数の表のいずれかで、1 つ以上の列に対して外部結合**演算子**（+）を使用する**結合**条件。結合条件を満たす行がすべて返される。また、外部結合演算子を使用しない**表**からすべての行が返される。その表には、外部結合演算子と一致する行が存在しない。たとえば、出荷と受領を組み合せると、受領済の出荷だけでなく、受領済かどうかに関係なくすべての出荷の**レコード**が生成される。受領済品目のデータはその出荷にアタッチされ、受領済でない出荷には空または NULL のフィールドがアタッチされる（と対比「**内部結合（inner join）**」）。

外部表（external table）

フラット・ファイルなど、外部データの**レコード**型の 1 つと関連付けられている読取り専用の**表**。外部表は、非リレーショナル・**ソース**のデータをリレーショナル表形式で表現する。Warehouse Builder アーキテクチャでは、外部表は**ファースト・クラス・オブジェクト**になる。

加算的（additive）

加算することで集計できる**ファクト / メジャー**。加算ファクトは最も一般的なタイプのファクトである。ファクト / メジャーの例には、販売価格、原価および収益がある（「**非加算的（nonadditive）**」、「**準加算的（semi-additive）**」と対比）。

カスケード・スナップショット（cascade snapshot）

親オブジェクト（**プロジェクト**、フォルダ、**マッピング**など）の、**Oracle Warehouse Builder** での**メタデータ・スナップショット**。**親**オブジェクトの情報のみでなく、そのすべての**子**オブジェクトの情報も含む。たとえば、2 つの表（Table1 と Table2）を含む、**データベース・モジュール** Module A のカスケード・スナップショットを取得し、そのいずれかの表をリポジトリ内で変更すると、Module A の子オブジェクトの 1 つが変更されたため、その変更が比較レポートで示される（「**カスケードなしスナップショット（no cascade snapshot）**」と対比）。

カスケードなしスナップショット（no cascade snapshot）

選択したオブジェクトの情報のみを取得し、その**子**オブジェクトの情報を取得しない**メタデータ・スナップショット**。たとえば、2 つの表（Table1 と Table2）を含む、**データベース・モジュール**の Module A のカスケードなしスナップショットを取得し、そのいずれかの表をリポジトリ内で変更しても、Module A の変更が比較レポートで示されない（「**カスケード・スナップショット（cascade snapshot）**」と対比）。

完全スナップショット（full snapshot）

比較だけでなくリストアにも使用できる十分なオブジェクト情報を含む**メタデータ・スナップショット**。完全スナップショットは**シグネチャ・スナップショット**より多くの領域を必要とするが、**メタデータのリストア**は完全スナップショットからのみ実行可能。完全スナップショットからシグネチャ・スナップショットへの変換は可能であるが、その逆は不可。

キー（key）

ある種の整合性制約の定義に含まれる**列**または列のセット。キーは、**リレーショナル・データベース**の様々な表や列の間の関係を表す（「**整合性制約（integrity constraint）**」、「**外部キー（foreign key）**」、「**主キー（primary key）**」も参照）。

キューブ（cube）

多次元**スキーマ**におけるデータの基本構造。キューブには、ディメンション、階層、レベル、メジャーなどが含まれる。Warehouse Builder アーキテクチャでは、キューブは**ファースト・クラス・オブジェクト**になる（「**ディメンション（dimension）**」、「**階層（hierarchy）**」、「**レベル（level）**」、「**メジャー（measure）**」も参照）。

行（row）

マッピングにおけるデータ処理の基本単位。行は、**表**内の 1 つのエンティティまたは**レコード**に関連する属性または値のセットである。これは、1 つのレコードに対応する**列**情報の集合である。行は構造化されており、属性によって定義されている。各**属性**には、名前や**データ型**、長さ、スケールおよび精度などが指定されている。

行セット（row set）

行セットは、**マッピング**の**演算子**で設定および取得されるゼロ個以上の構造化されたデータ行によって構成される。マッピングでは、行セットの**ソース**からの抽出、変換、および**ターゲット**へのロードを、演算子を使用して定義する。行セット内の行数は、行セットのカーディナリティと呼ばれている。

行ベース（row based）

オペレーティング・モードのオプション。行ベース・モードでは、ランタイム監査の量を最大限にすることができる。**行**ベース・モードで実行される**マッピング**では、**ソース**・レコードにあるデータが**レコード**単位で処理される（「**セット・ベース（set based）**」と対比）。

クライアント（client）

クライアント / **サーバー**のアーキテクチャにおいては、キーボード、ディスプレイ、ポインティング・デバイス（マウスなど）を介してユーザーと対話するフロントエンド・**データベース**・アプリケーション。クライアント部分でデータにアクセスする必要はない。サーバー部分で管理しているデータの要求、処理、表示のみが行われる。

クリップボード（clipboard）

切り取られたオブジェクトやコピーされたオブジェクトが格納される、**Oracle Warehouse Builder クライアント**の領域。一度に格納できるオブジェクトは 1 つのみ。

グループ（group）

マッピングでは、各**演算子**の属性はいくつかのグループに分けられる。グループは、属性の方向を示す。したがって、入力グループには、演算子に入力される属性が含まれる。出力グループには、演算子の出力属性が含まれる（「**属性（attribute）**」も参照）。

クレンジング（cleansing）

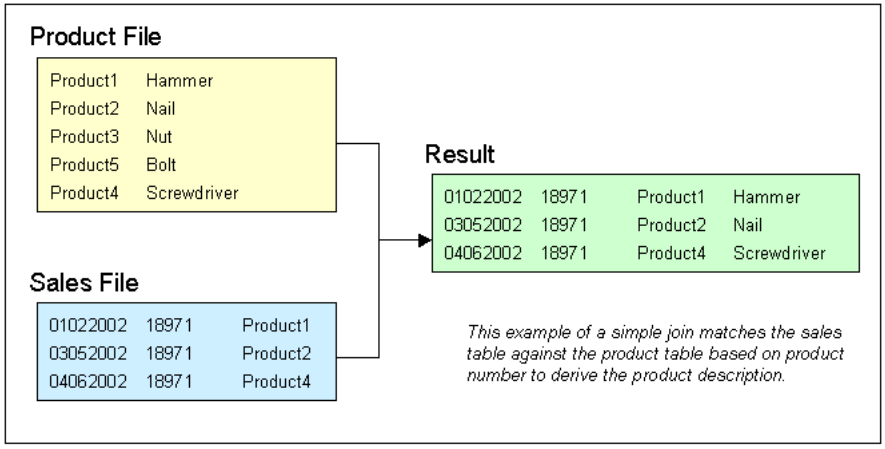
ソース・データの非一貫性を解決し、異常を修正する処理。通常は、**ETL** 処理の一部（「**抽出、変換、ロード（extraction, transformation, and loading: ETL）**」も参照）。

クロス集計（crosstab）

行と列のマトリックスに項目を編成する表示レイアウト。項目は、行と列のページ・エッジに表示される。概要情報を表示したり、地域と月ごとの売上げなど、ディメンション全体でのデータ変化を表示したりする場合に、クロス集計を使用できる。マトリックスと呼ばれることもある。

結合（join）

リレーショナル・データベース管理において、複数の**表**からデータを選択する**問合せ**。結合では、特定の条件に基づいて、1 つの表（**ファイル**）と別の表（ファイル）を照合し、その 2 つの表のデータで新しい表を作成する。たとえば、顧客表と注文表を結合して、特定の商品を購入したすべての顧客の表を作成できる。結合は、複数の表が FROM 句に含まれることを特徴としている。Oracle は、WHERE 句で指定された条件を使用してこれらの表の行を組み合せて、その結果の行を返す。この条件は、結合条件と呼ばれ、通常は、結合されたすべての表の列が比較される（「**内部結合（inner join）**」、「**外部結合（outer join）**」も参照）。



権限（privilege）

ある種の **Structured Query Language（SQL）**文を実行したり、他のユーザーのオブジェクトにアクセスするための権利。

子（child）

1. ディメンション階層の場合：**階層**内の所定の値より下の**レベル**にある値。たとえば、時間**ディメンション**における値「Jan-99」は、値「Q1-99」の子である。子の値が複数の階層に属している場合、その値は 2 つ以上の**親**を持つ子となる。
2. **Oracle Warehouse Builder** の**メタデータ**・オブジェクトの場合：上位レベルのオブジェクトに属している**ナビゲーション・ツリー**のオブジェクト。たとえば、**表**は**データベース・モジュール**の子オブジェクト、データベース・モジュールは**プロジェクト**の子オブジェクトである。

コード生成（code generation）

Design Repository に格納されている**メタデータ**を取得し、設計済**モデル**を**ターゲット**・システムに作成するコードを生成する、**Oracle Warehouse Builder** の処理。

固定長フィールド（fixed-length field）

フィールドの始まりと終わりが**デリミタ**文字ではなく、位置で指定される。固定長フィールドのサイズは、フィールド・サイズの指定どおりになる。たとえば、25 バイトの名前フィールドでは、各**レコード**で 25 バイトが占有される（「**固定長フラット・ファイル（fixed-length flat file）**」、「**フラット・ファイル（flat file）**」も参照）。

固定長フラット・ファイル（fixed-length flat file）

レコードに固定長フィールドが含まれる**フラット・ファイル**。レコードに**デリミタ**で区切られた可変長フィールドが含まれる「**デリミタ付きフラット・ファイル（delimited flat file）**」と対比（「**レコード（record）**」、「**固定長フィールド（fixed-length field）**」も参照）。

コネクタ（connector）

ロケーション間の**接続**情報の定義に使用される、**Oracle Warehouse Builder** の**ナビゲーション・ツリー**にあるオブジェクト。コネクタは、Oracle **データベース**・**モジュール**のロケーションでのみ定義できる。Warehouse Builder アーキテクチャでは、コネクタは**ファースト・クラス・オブジェクト**になる（「**ロケーション（location）**」も参照）。

コミット（commit）

データベースのデータに永続的な変更（挿入、更新、削除）を行うこと。変更がコミットされるまでは、古いデータと新しいデータの両方が存在するため、変更を格納したり、データを前の状態に戻すことができる（「**ロールバック（rollback）**」も参照）。

コレクション（collection）

Oracle Warehouse Builder の**プロジェクト**内にあるオブジェクトのグループ化に使用できるグループ化メカニズム。コレクションには、同じプロジェクトに属するオブジェクトへのショートカットが含まれる。Oracle Discoverer などのツールに**メタデータのエクスポート**を実行する場合は、コレクション内に必要な**メタデータ**をグループ化する必要がある。Warehouse Builder アーキテクチャでは、コレクションは**ファースト・クラス・オブジェクト**になる。

コンソール（console）

Oracle Warehouse Builder アプリケーションのメイン・ウィンドウ。コンソールには、メニュー・バー、ランチャーおよび**ナビゲーション・ツリー**が含まれる。

サード・クラス・オブジェクト（Third Class Object）

Oracle Warehouse Builder アーキテクチャでは、サード・クラス・オブジェクトとフォース・クラス・オブジェクトは、他のオブジェクトに所有されたオブジェクトの相対的な位置付けを表す。所有権が複数層になるオブジェクトのみに当てはまる。たとえば、DIMENSTION_TABLE（ファースト・クラス・オブジェクト）が INDEX_COLUMN を所有する使用例において、INDEX_COLUMN はセカンド・クラス・オブジェクトになる。しかし、ファースト・クラス・オブジェクト CUBE_TABLE がセカンド・クラス・オブジェクト INDEX を所有し、その下に INDEX_COLUMN が所有されている使用例では、INDEX_COLUMN はサード・クラス・オブジェクトになる。

（「**ファースト・クラス・オブジェクト（First Class Object : FCO）**」、「**セカンド・クラス・オブジェクト（Second Class Object : SCO）**」も参照）

サーバー（server）

クライアント / サーバー・アーキテクチャにおいて、Oracle ソフトウェアを実行し、共有データへの同時アクセスに必要な機能を処理するコンピュータ。サーバーは、クライアント・アプリケーションで生成された **SQL** 文や **PL/SQL** 文を受信し、処理する。

サービス名（service name）

TNS 接続記述子に含まれるネットワーク・アドレスにマッピングされる、短くて、扱いやすい名前。ユーザーは、該当するサービス名を知っているだけで TNS 接続を実行できる（「**tnsnames.ora**」も参照）。

索引（index）

索引は、表およびクラスタと関連付けられた、省略可能な構造である。**表**の 1 つ以上の**列**に索引を作成すると、その表に対する **SQL** 文の実行を高速化できる（「**ビットマップ索引（bitmap index）**」も参照）。

サブジェクト領域（subject area）

組織の一部またはナレッジの領域を表示および識別する分類システム。通常、**データ・マート**は、販売、マーケティング、地域などのサブジェクト領域をサポートするために開発される。

サロゲート・キー（surrogate key）

RDBMS で生成される一意の**主キー**。このキーは、**データベース**内のどのデータからも導出されず、主キーとしてのみ使用される。**Oracle Warehouse Builder** では、順序の**マッピング演算子**を使用してこのタイプのキーを生成できる（「**順序（sequence）**」も参照）。

参照（lookup）

事前定義済の値の**表**（配列、マトリックスなど）やデータ・**ファイル**の中で実行されるデータ検索。

式（expression）

既存の値から新しい値を算出する、「SALARY + COMMISSION」のような計算式。式は、**列**名、関数、演算子、定数などで構成できる。計算式は、コマンドや **SQL** 文などに含まれる。

シグネチャ（signature）

差分解析の実行に必要なオブジェクト情報のみを含む、Warehouse Builder **メタデータ**・オブジェクトのコンパクトな定義（「**シグネチャ・スナップショット（signature snapshot）**」も参照）。

シグネチャ・スナップショット（signature snapshot）

シグネチャ情報のみを取得する**メタデータ・スナップショット**。シグネチャ・スナップショットを使用して、スナップショットと現在のリポジトリ・オブジェクト、またはスナップショット同士の比較レポートを生成できる（「**完全スナップショット（full snapshot）**」と対比）。

システム識別子（system identifier : SID）

Oracle **インスタンス**の一意の名前。Oracle データベースを切り替えるには、システム識別子を指定する必要がある。システム識別子は、**tnsnames.ora** ファイルにある接続記述子の CONNECT DATA 部分と、alistener.ora **ファイル**のネットワーク・リスナーの定義で指定されている。

実行（execution）

Oracle Warehouse Builder では、配布されたマッピングおよびプロセス・フローを実行するプロセス（「**配布（deployment）**」も参照）。

集計操作（aggregation）

データ値を単一の値に統合する処理。たとえば、1 日単位で集めた販売データを週レベルに集計したり、週のデータを月レベルに集計するなどの処理がこれに該当する。その後、データは**集計データ**として参照できる。集計操作（aggregation）と集計操作（summarization）は同義語であり、集計データ（aggregate data）と集計データ（summary data）は同義語である。

集計データ（aggregate data）

集計されたデータ。たとえば、特定製品の売上数量を 1 日、1 か月、四半期、1 年ごとに集計できる（「**集計操作（aggregation）**」も参照）。

従来型ビュー（conventional view）

「**ビュー（view）**」を参照。

主キー（primary key）

表の PRIMARY KEY 制約の定義に含まれる**列**または列のセット。主キーの値は、**表**の**行**を一意に識別する。各表では、主キーを 1 つしか定義できない（「**外部キー（foreign key）**」も参照）。

出力コンポーネント（output component）

Name and Address 演算子では、ある属性がどんな名前およびアドレスのコンポーネントを構成するかが、出力コンポーネントによって示される。各出力コンポーネントは、タイトル、標準化された名、町村番号、町村名、町村タイプなど、目的別に分割された名前またはアドレスのエンティティを表す。Name and Address 演算子の出力属性には、それぞれ名前またはアドレスにおける役割を示す出力コンポーネントが必要である（「**入力ロール（input role）**」も参照）。

準加算的（semi-additive）

すべてのディメンションではなく、一部のディメンションを加算することで集計できる**ファクト / メジャー**。たとえば、社員数および在庫数など（「**加算的（additive）**」、「**非加算的（nonadditive）**」と対比）。

順序（sequence）

順次に生成された一連の番号である**データベース・スキーマ**・オブジェクト。Oracle では、**SYSTEM 表領域**内の 1 つのデータ・ディクショナリ**表**に行として格納される。順序定義には、次の一般情報が示される。順序名、順序が昇順または降順かどうか、順序番号の間隔、およびその他の情報。Warehouse Builder アーキテクチャでは、順序は**ファースト・クラス・オブジェクト**になる。

推移（transition）

プロセス・フローでは、推移は、プロセス・フロー内でアクティビティが発生する順序と条件を示す。推移を使用すると、直前のアクティビティの完了状態を基に、**アクティビティ**を実行できる。

スキーマ（schema）

関連する**データベース**・オブジェクトの集合。表、ビュー、順序、ストアド・プロシージャ、シノニム、索引、クラスタ、データベース・リンクなどの**論理構造**が含まれる。スキーマは、データベース・ユーザーによって所有され、そのユーザーと同じ名前を持つ。リレーショナル・スキーマは、データベース・ユーザー ID でグループ化される（「**スノーフレーク・スキーマ（snowflake schema）**」、「**スター・スキーマ（star schema）**」も参照）。

スター・スキーマ（star schema）

スキーマの設計が多次元データ・**モデル**を示すリレーショナル・**スキーマ**。スター・スキーマは、最低 1 つの**ファクト表**と最低 1 つの**ディメンション表**で構成される。これらの表は、外部キーで関連付けられている（「**スノーフレーク・スキーマ（snowflake schema）**」も参照）。

スナップショット（snapshot）

「**メタデータ・スナップショット（metadata snapshot）**」を参照。

スノーフレーク・スキーマ（snowflake schema）

ディメンション表が一部または完全に正規化された**スター・スキーマ**のタイプ（「**正規化（normalize）**」も参照）。

正規化（normalize）

リレーショナル・データベース内のデータを複数の**表**に分割し、データの冗長性を排除する処理（**非正規化（denormalize）**と対比）。

整合性制約（integrity constraint）

表の**列**のルールを定義する、宣言的な方法。整合性制約は、**データベース**と関連付けられたビジネス・ルールを実施し、無効な情報が表に入力されるのを防ぐ。

生成（generation）

「**コード生成（code generation）**」を参照。

制約（constraint）

制約は、**データベース**管理者が指定したガイドラインにデータを準拠させるメカニズムを提供する。制約の最も一般的なタイプとしては、一意制約（所定の**列**のすべての値を一意にする）、NOT NULL 制約、外部キー制約（異なる表にある 2 つのキーに**主キー** / **外部キー**関係を持たせる）などがある。

セカンド・クラス・オブジェクト（Second Class Object : SCO）

Oracle Warehouse Builder アーキテクチャでは、セカンド・クラス・オブジェクト（SCO）は、依存オブジェクト・コンポーネントを表す。SCO は必ず別のオブジェクトによって所有される。また、別のオブジェクトを所有することができる。たとえば、ファースト・クラス・オブジェクト `MAPPING` はセカンド・クラス・オブジェクト `MAPPING_OPERATOR` を所有し、その下に `ATTRIBUTES` が所有されている。SCO には、`COLUMN`、`FOREIGN_KEY` およびすべてのマッピング演算子などもある。

グラフィカル・ユーザー・インタフェースで Warehouse Builder にアクセスする場合、セカンド・クラス・オブジェクトは大抵、ファースト・クラス・オブジェクトでのみ操作できるオブジェクトのことを指す。同様に、OMB Plus で Warehouse Builder にアクセスする場合、ファースト・クラス・オブジェクトに対するコマンドでのみセカンド・クラス・オブジェクトの定義を操作できる。

（「**ファースト・クラス・オブジェクト（First Class Object : FCO）**」、「**サード・クラス・オブジェクト（Third Class Object）**」も参照）

接続（connection）

ユーザー・プロセスと Oracle **インスタンス**との間の通信経路。

セット・ベース（set based）

オペレーティング・モードのオプション。Warehouse Builder において、1 つの **SQL** コマンドですべてのデータを挿入できるようにする。データがクリーンで、監査の拡張を必要としない場合には、セット・ベース・モードを使用することにより、**DML** 処理を高速化することができる（「**行ベース（row based）**」と対比）。

ソース（source）

マッピング内のデータが導出される**データベース**、アプリケーション、**ファイル**またはその他の記憶域機能。

ソース演算子（source operator）

マッピングの**データ・ソース**として動作する**演算子**。マッピングは、ソース演算子からのデータ**抽出**、1 つ以上の**データ・フロー演算子**による**変換**および**ターゲット演算子**へのロードを意味する。ソース演算子の例には、表演算子、フラット・ファイル演算子、外部表演算子などがある。

属性（attribute）

オブジェクトに格納できるデータの定義済項目。オブジェクト（エンティティ）内の**列**で表されるフィールド。Oracle Warehouse Builder では、この用語は次の 2 つのコンテキストで使用される。

1. 1 つの**レベル**の要素を特徴づける**ディメンション**の列。
2. **マッピング**の**演算子**では、属性はデータ・レベル・エントリを表す。

属性セット（attribute set）

Warehouse Builder の表やビューの場合、属性セットには、指定した順序で並べられた選択済の列が含まれる。Warehouse Builder では、表および定義済制約に対して事前定義済の**属性**セットが生成される。マッピングの設計時には、独自の属性セットを作成することができる。属性セットを Oracle Discoverer にエクスポートする場合は、**ブリッジ**型の属性セットを作成できる。

祖先（ancestor）

階層内の所定の値より上の**レベル**にある値。たとえば、時間**ディメンション**における値「1999」は、値「Q1-99」および値「Jan-99」の祖先である。

ソフトウェア・ライブラリ（Software Library）

Oracle Warehouse Builder に現在インストールされているインテグレータが含まれる。管理環境でアクセス可能。

ターゲット（target）

抽出、変換、ロード処理の任意の部分に関する中間的または最終的な結果を保持する。ETL処理全体の**ターゲット**は、多くの場合**データ・ウェアハウス**である。

ターゲット演算子（target operator）

マッピングのデータ・**ターゲット**として動作する**演算子**。マッピングは、**ソース演算子**からのデータ**抽出**、1 つ以上の**データ・フロー演算子**による**変換**およびターゲット演算子へのロードを意味する。ターゲット演算子の例には、表演算子やフラット・ファイル演算子などがある。

妥当性チェック（validation）

メタデータ定義や構成パラメータを検証するプロセス。

短縮名（short name）

短縮名は、**Oracle Warehouse Builder** で作成されたすべての関連オブジェクトの語根名として使用される一意の識別子。

抽出（extraction）

ETL の初期フェーズの一部。**ソース**からデータを取り出す処理（「**抽出、変換、ロード（extraction, transformation, and loading: ETL）**」も参照）。

抽出、変換、ロード（extraction, transformation, and loading: ETL）

ETL は、**ソース**・データへのアクセスや操作、**データ・ウェアハウス**内や他のタイプの**ターゲット**へのソース・データのロードに関する方法を意味する。これらの処理の実行順序は多様である。ETL のかわりに ETT（**抽出**、**変換**、転送）や ETM（抽出、変換、移動）が使用されることもある。

調整（reconcile）

マッピングでは、定義したすべての**演算子**に、**Oracle Warehouse Builder** Design Repository 内の定義を対応付けることができる。演算子を調整する場合は、演算子が対応するリポジトリ・オブジェクトと一致していることを確認する必要がある。**マッピング**へのインバウンド調整、またはマッピングからのアウトバウンド調整を実行できる。インバウンド調整では、指定したリポジトリ・オブジェクトの定義を使用して、演算子が更新または調整される。アウトバウンド調整では、マッピングの演算子に対する変更内容を反映するよう、選択した**リポジトリ**・オブジェクトが更新される。

データ・ウェアハウス（data warehouse）

トランザクション処理用ではなく、**問合せ**と分析用に設計された**リレーショナル・データベース**。データ・ウェアハウスには、通常、**トランザクション**・データから導出された履歴データが含まれるが、別のソースからのデータを含めることもできる。データ・ウェアハウスは、トランザクションの作業負荷から分析に関する作業負荷を切り離し、ビジネスにおける各種ソースからのデータ統合を可能にする。

データ・ウェアハウス環境は、リレーショナル・データベースに加え、**ETL** ソリューション、**OLAP** エンジン、**クライアント**分析ツール、およびデータ収集やビジネス・ユーザーへのデータ配信処理を管理する他のアプリケーションで構成されることが多い（「**抽出、変換、ロード（extraction, transformation, and loading: ETL）**」、「**オンライン分析処理（Online Analytical Processing : OLAP）**」も参照）。

データ型（datatype）

1. データの標準形式。Oracle のデータ型は、CHAR、NCHAR、VARCHAR2、NVARCHAR2、DATE、NUMBER、LONG、CLOB、NCLOB、RAW および LONG RAW である。ただし、Oracle **データベース・サーバー**では、他の標準データ型も識別および変換される。
2. ある項目にプロパティとして関連付けられる、名前付きの固定属性のセット。データ型を使用することにより、データの特性を定義できる（「**属性（attribute）**」も参照）。

データ・ソース（data source）

データベース、アプリケーション、データ定義**ソース**、またはデータを提供する**ファイル**。

データ操作言語（DML）

表内のデータを変更する INSERT、UPDATE、DELETE などの文が含まれる。

データ定義言語（DDL）

データ構造を定義または変更する CREATE/ALTER TABLE/INDEX などの文が含まれる。

データ・フロー演算子（data flow operator）

マッピングにおいてデータ・フロー演算子は、ソースからターゲットに流れるデータを変更または変換する。結合子演算子はデータ・フロー演算子の一例である。結合子演算子は、カーディナリティの異なる複数のソースから取り込んだ複数の行セットを結合し、1 つの出力**行セット**を生成する（「**マッピング（mapping）**」、「**演算子（operator）**」、「**ソース演算子（source operator）**」、「**ターゲット演算子（target operator）**」も参照）。

データベース（database）

1 つの単位として処理される、データの集合。データベースの目的は、関連する情報を格納したり検索することである。

Oracle データベースは、1 つの単位として処理される一連の**オペレーティング・システム**・ファイルで、そこに Oracle データベース・**サーバー**がデータ・ディクショナリ表やユーザー表を格納する。データベースには、データベース・ファイル、REDO ログ・ファイル、制御ファイルの 3 種類のファイルが必要である（「**表（table）**」、「**リレーショナル・データベース（relational database）**」、「**リレーショナル・データベース管理システム（relational database management system : RDBMS）**」も参照）。

データベース・オブジェクト（database object）

データベースで作成され、格納されるもの。データベース・オブジェクトの例には、表、ビュー、シノニム、索引、順序、クラスタ、列などがある（「**表（table）**」、「**ビュー（view）**」、「**索引（index）**」、「**順序（sequence）**」、「**列（column）**」も参照）。

データベース・サーバー（database server）

ORACLE Server カーネルを実行し、**データベース**を格納するコンピュータ。

データベース・リンク（database link）

ローカル・**データベース**に格納されるオブジェクト。**リモート・データベース**とその通信パスを識別し、リモート・データベースの**ユーザー名**およびパスワードもオプションで識別する。定義したデータベース・リンクは、リモート・データベースの表に対する問合せの実行に使用できる。DBlink とも呼ばれる。**SQL*Plus** では、`DESCRIBE` コマンドか `COPY` コマンドでデータベース・リンクを参照できる。

データ・マート（data mart）

販売、マーケティング、金融など、特定のビジネス分野向けに設計された、簡単な**データ・ウェアハウス**。データ・マートは、多くの場合、組織の一部門によって構築および管理される。データ・マートの対象は、1 つのサブジェクトに絞られるため、一般的にデータは少数のソースから抽出される。依存型のデータ・マートの場合、データは、エンタープライズ規模のデータ・ウェアハウスから導出できる。独立型のデータ・マートの場合、データは、組織内の業務システムや外部データから直接収集できる。

ディメンション（dimension）

データを分類するための構造。通常は 1 つ以上の階層で構成される。個別ディメンションとメジャーを組み合せると、エンド・ユーザーはビジネス上の疑問に応答できる。最も一般的なディメンションは、「顧客」、「製品」および「時間」である。Oracle9*i* の場合、ディメンションは、**列**セットの各組の間の階層（**親** / **子**）関係を定義する**データベース・オブジェクト**である。Warehouse Builder アーキテクチャでは、ディメンションは**ファースト・クラス・オブジェクト**になる。

ディメンション・オブジェクト（dimensional object）

ディメンション・オブジェクトはリレーショナル・オブジェクトと類似しているが、データを識別および分類するための追加**メタデータ**を含む。ディメンション・オブジェクトを定義するときには、より構造化した形式でデータを格納しやすくなるように、論理的な関係を記述する。ディメンション・オブジェクトには、ディメンションやキューブなどがある（「**リレーショナル・オブジェクト（relational object）**」と対比）。

ディメンション値（dimension value）

ディメンションを構成するリスト内の要素の 1 つ。たとえば、コンピュータ会社には製品ディメンションに、「LAPPC」と「DESKPC」というディメンション値がある。地理ディメンション値には「Boston」や「Paris」がある。時間ディメンション値には「MAY96」や「JAN97」がある。

デバッグ（debug）

コンピュータのプログラム・コードにある機能不全の要素やエラーを見つけ、修正すること。**Oracle Warehouse Builder** では、この用語は、データ・フロー内のエラーや問題を検出するための処理を指す。

デプロイメント・マネージャ（Deployment Manager）

Oracle Warehouse Builder の総合的な**配布**コンソール。構成や**妥当性チェック**など、配布のすべての側面を表示および管理できる。また、オブジェクトの配布履歴を表示して、オブジェクトの配布方法を決定することもできる。

デリミタ（delimiter）

1 つの項目や一連のデータを他の項目やデータと区切るために使用する文字または文字の組合せ。たとえば、カンマ区切りレコードでは、各フィールドのデータがカンマで区切られる。固定長レコードでは、デリミタ文字を使用せず、各フィールドの始まりと終わりの位置を指定する（「**フラット・ファイル（flat file）**」、「**レコード（record）**」も参照）。

デリミタ付きフラット・ファイル（delimited flat file）

フィールドをデリミタで区切った**フラット・ファイル**。「**固定長フラット・ファイル（fixed-length flat file）**」と対比（「**レコード（record）**」、「**デリミタ（delimiter）**」も参照）。

転送ウィザード（Transfer Wizard）

多種多様なビジネス・インテリジェンス・ツールにおいて、**Design Repository** に格納されている**メタデータ**を同期化、統合、使用するための **Oracle Warehouse Builder** ユーティリティ。メタデータは、ブリッジを使用してインポートおよびエクスポートできる。また、メタデータは、OMG ファイル、Oracle Discoverer、Oracle Express、ERwin、Predesignate および Oracle OLAP Server などの間で交換することもできる。

天然キー（natural key）

データに対してネイティブな**キー**。各**行**の識別に使用される。たとえば、状態コードである CA と DC は天然キーとして使用される。

問合せ（query）

Structured Query Language（SQL）の SELECT 文。任意の組合せ、**式**、順序でデータを取得する、問合せは読取り専用の操作である。つまり、データは取得されるが、変更されない。問合せは多くの場合、**DML** 文であると見なされる。

問合せの結果（query results）

問合せで取得したデータ。

同期（synchronization）

マルチユーザー環境において他のユーザーがコミットした変更でオブジェクト定義を更新するための方法。**ナビゲーション・ツリー**に表示されるオブジェクトは、**リポジトリ**内のオブジェクト定義を使用して同期化される（「**コミット（commit）**」も参照）。

導出ファクト / 導出メジャー（derived fact (or measure)）

数学的な操作またはデータ**変換**を使用して、既存のデータから生成される**ファクト / メジャー**。導出ファクト（またはメジャー）の例には、平均、合計、割合および差異などがある。

トランザクション（transaction）

1. データベースのコンテキスト：1 人のユーザーが実行する、1 つ以上の **SQL** 文で構成される論理作業単位。1 つのトランザクションに含まれるすべての文は、まとめてコミットまたはロールバックされる。Oracle と互換性がある ANSI/ISO SQL 標準に準拠したトランザクションは、ユーザーの最初の実行 SQL 文によって開始される。トランザクションは、ユーザーが明示的にコミットまたはロールバックした時点で終了される（「**オンライン・トランザクション処理（Online Transactional Processing : OLTP）**」、「**コミット（commit）**」、「**ロールバック（rollback）**」も参照）。
2. データ・ウェアハウス概念のコンテキスト：ビジネス・アクティビティの最小単位。例：銀行預金、在庫での単一品目の移動、小売販売など。**OLTP** システムはトランザクションの処理に使用されるが、**OLAP** システムは上位レベルでのトランザクション・データの解析に使用される（「**オンライン・トランザクション処理（Online Transactional Processing : OLTP）**」、「**オンライン分析処理（Online Analytical Processing : OLAP）**」も参照）。

ドリル（drill）

ある項目から関連する一連の項目にナビゲートすること。ドリル操作では、通常、**階層**内レベルの上下へのナビゲートを行う。データを選択する場合は、階層内でドリルダウンまたはドリルアップすることによって、階層をそれぞれ拡張または縮小できる（「**レベル（level）**」、「**ドリルダウン（drill down）**」、「**ドリルアップ（drill up）**」も参照）。

ドリルアップ（drill up）

階層内で**親**の値に関連している一連の子の値を縮小すること（「**ドリル（drill）**」、「**ドリルダウン（drill down）**」、「**レベル（level）**」も参照）。

ドリルダウン（drill down）

階層内で**親**の値に関連している**子**の値を拡張表示すること（「**ドリル（drill）**」、「**ドリルアップ（drill up）**」、「**レベル（level）**」も参照）。

内部結合（inner join）

これは**結合**のデフォルト・タイプである。一致条件があると、結果を示す**行**が生成される。たとえば、出荷と受領を組み合せると、受領済の出荷のみが生成される（「**外部結合（outer join）**」と対比）。

ナビゲーション・ツリー（navigation tree）

Oracle Warehouse Builder オブジェクトをプロジェクトごとに整理する、メイン・コンソールの作業領域。すべてのオブジェクトが展開可能なフォルダに整理される点で、**ファイル**・システムに類似している（「**プロジェクト（project）**」、「**ファースト・クラス・オブジェクト（First Class Object : FCO）**」も参照）。

二重アドレス（dual address）

Name and Address 演算子の二重アドレスには、同じアドレス・レコードの私書箱と町村の両方が含まれる。二重アドレスを持つレコードの場合は、1 つのアドレス行が正規アドレスになり、もう 1 つのアドレス行が二重アドレスになる。二重アドレスの例は、次のとおり。

PO Box 2589
4439 Mormon Coulee Rd
La Crosse WI 54601-8231

入力ロール（input role）

Name and Address 演算子では、入力ロールは、ソースから抽出したデータ行に含まれる名前またはアドレスの情報の種類を表す。入力ロールは、自由形式データについては目的別に分割されていないもの（行指向）にすることができ、特定の入力属性については目的別に分割された（ファースト・ネーム、第 1 アドレス、市区町村などを指定）ものにすることができる。入力ロールには可能なかぎり、目的別に分割されていないロール（「行 1」など）ではなく、目的別に分割されたロール（「人名」など）を使用する必要がある。分割されたロールでは、Name and Address 演算子に対して、ソース属性の内容に関する情報がより多く与えられる（「**出力コンポーネント（output component）**」も参照）。

パーティション（partition）

1 つのディレクトリ・**サーバー**に格納されている、重複していない一意のディレクトリ・ネーミング・コンテキスト。

パーティション交換ロード（partition exchange loading : PEL）

Oracle Warehouse Builder の実行時におけるパフォーマンス機能。データ・パーティションの使用により、**ターゲット**・システム内でデータをロードまたは削除する際のパフォーマンスが向上する（「**パーティション（partition）**」も参照）。

配布（deployment）

以前に設計した論理**モデル**から物理**ターゲット**・システムを作成する処理。

バインド（bind）

マッピング・エディタの**演算子**と **Design Repository** のオブジェクトを接続するアクション。

バインドなしオブジェクト（unbound object）

Design Repository で定義されたオブジェクトに接続されていない**マッピング**・オブジェクト**演算子**（「**バウンド・オブジェクト（bound object）**」と対比）。

バウンド・オブジェクト（bound object）

Design Repository で定義されたオブジェクトに接続されている、**Oracle Warehouse Builder** の**マッピング**・オブジェクト**演算子**（「**バインドなしオブジェクト（unbound object）**」と対比）。

バウンド名（bound name）

マッピングのオブジェクト**演算子**に接続されているオブジェクトの**物理名**。

パッケージ（package）

関連するプロシージャ、ファンクション、その他のパッケージ構成を**データベース**内の 1 つの単位としてまとめてカプセル化し、格納する方法。パッケージを使用すると、データベース管理者やアプリケーション開発者にとって組織上のメリットがあると同時に、機能が強化され、データベースのパフォーマンスが向上する（「**PL/SQL**」も参照）。

非加算的（nonadditive）

加算することで集計できない**ファクト / メジャー**。たとえば、平均など（「**加算的（additive）**」、「**準加算的（semi-additive）**」と対比）。

ビジネス名（business name）

ビジネス・アプリケーションで使用される **Oracle Warehouse Builder** オブジェクトの名前。Warehouse Builder の以前のリリースでは、**論理名**と呼んでいた（「**物理名（physical name）**」と対比）。

非正規化（denormalize）

表をフラットな状態のままにするため、**表**内の冗長性を許可する処理（「**正規化（normalize）**」と対比）。

ビットマップ索引（bitmap index）

索引の目的は、所定の**キー**値を含む、**表**内の行に対するポインタを提供することである。通常の索引では、各キーと、そのキー値を含む行に対応する ROWID のリストを格納する。ビットマップ索引では、ROWID と各キー値のリストのかわりにビットマップを使用する。ビットマップ内の各ビットは使用可能な ROWID に対応し、ビットが設定されると、対応する ROWID を持つ**行**にキー値があることを意味する。マッピング関数が、ビット位置を実際の ROWID に変換する。ビットマップ索引は、WHERE 句に複数の条件がある問合せに最も効果がある。

ピボット（pivot）

マッピングでピボット**演算子**を使用すると、複数の属性が含まれる単一**行**を複数の行に変換できる。この演算子は、マッピングで複数の行ではなく、複数の属性間に含まれるデータを変換する際に使用する（「**アンピボット（unpivot）**」も参照）。

ビュー（view）

ビューは、1 つの表または複数の表のデータを、カスタマイズされた形式で表示する「ストアド**問合せ**」であると見なすことができる。ビューは、データを実際に格納するわけではなく、基になる実表からデータを抽出する。制約はいくつかあるが、表と同様、ビューに対しても、問合せ、更新、挿入、削除を行える。ビューに対して実行した操作はすべて、ビューの実表に影響を与える Warehouse Builder アーキテクチャでは、ビューは**ファースト・クラス・オブジェクト**になる（「**マテリアライズド・ビュー（materialized view）**」も参照）。

表（table）

リレーショナル・データベース管理システムの記憶域の基本単位。表は、エンティティとリレーションシップを表し、1 つ以上の情報単位（行）で構成される。各行には、同じ種類の値（列）が格納される。各列には、列名、**データ型**（CHAR、NCHAR、VARCHAR2、NVARCHAR2、DATE、NUMBER など）、幅が指定される（幅は、DATE のようにデータ型で事前定義されている場合がある）。表を作成した後、有効なデータ行を挿入できる。その後で、表に対して、問合せ、削除、更新を行える。定義済のビジネス・ルールを表のデータに適用するため、表の整合性制約とトリガーも定義できる。Warehouse Builder アーキテクチャでは、表は**ファースト・クラス・オブジェクト**になる（「**行（row）**」、「**列（column）**」も参照）。

表示セット（display set）

マッピングにおいて表示セットは、**演算子**属性のサブセットの視覚的な表現を指す。表示セットを使用すると、演算子内に表示する属性の数を制限して、複雑なマッピングを簡略化できる。

表領域（tablespace）

関連する**論理構造**をグループにまとめた、**データベース**の記憶域単位。データベースは、表領域と呼ばれる、1 つ以上の論理記憶域単位に分割される。表領域は、セグメントと呼ばれる論理記憶域単位に分割される。セグメントはさらにエクステントに分割される。

ファースト・クラス・オブジェクト（First Class Object : FCO）

Oracle Warehouse Builder アーキテクチャでは、ファースト・クラス・オブジェクト（FCO）は、Warehouse Builder インタフェースから操作できるメタデータ・リポジトリにあるコンポーネントを表す。ファースト・クラス・オブジェクトは多くの場合、他のオブジェクトを所有している（必ずしもそうではない）。たとえば、ファースト・クラス・オブジェクト TABLE は、次のセカンド・クラス・オブジェクトを所有する。TABLE_COLUMN、UNIQUE_KEY、FOREIGN_KEY および CHECK_CONSTRAINT。FCO には、EXTERNAL_TABLE、CONNECTOR、MAPPING などもある。

グラフィカル・ユーザー・インタフェースで Warehouse Builder にアクセスする場合、ファースト・クラス・オブジェクトは大抵、**ナビゲーション・ツリー**に表示される。同様に、**Oracle MetaBase Plus（OMB Plus）** で Warehouse Builder にアクセスする場合、FCO は OMBCREATE、OMBALTER、OMBRETRIEVE および OMBDELETE コマンドのオブジェクトであると見なすことができる。

（「**セカンド・クラス・オブジェクト（Second Class Object : SCO）**」、「**サード・クラス・オブジェクト（Third Class Object）**」も参照）

ファイル（file）

1 つの単位として処理されるデータの集合。リスト、文書、索引、メモ、手順など。主に、磁気テープや磁気ディスクに格納されているデータを指す。Warehouse Builder アーキテクチャでは、ファイルは**ファースト・クラス・オブジェクト**になる。

ファイルから表へのマッピング（file-to-table mapping）

フラット・ファイルから**データ・ウェアハウス**内の表へのデータのマップ（「**フラット・ファイル（flat file）**」、「**表（table）**」も参照）。

ファクト / メジャー（fact/measure）

調査や分析の対象となるデータで、通常は数値データや**加算的**データ。ファクトやメジャーに関する値は、事前に判明しているわけではなく、考察されてから格納される。ファクト / メジャーの例には、販売価格、原価および収益がある。ファクトとメジャーは同義語である。ファクトは主にリレーショナル環境で使用され、メジャーは主に多次元環境で使用される（「**ディメンション（dimension）**」、「**属性（attribute）**」も参照）。

Oracle Warehouse Builder では、ファクトまたはファクト表は、キューブと呼ばれている（「**導出ファクト / 導出メジャー（derived fact (or measure)）**」、「**キューブ（cube）**」も参照）。

ファクト表（fact table）

ファクト / メジャーを格納する、**スター・スキーマ**の**表**。ファクト表には、ファクトを含む列や**ディメンション**表の**外部キー**である列など、2 種類の**列**がある。通常、ファクト表の**主キー**は、すべての外部キーで構成されるコンポジット・キーである。

ファクト表には、詳細レベルのファクトまたは集計されたファクト（集計されたファクトを含むファクト表は、集計表と呼ばれることがある）のいずれかが含まれている。通常、ファクト表には同じ**集計操作**レベルのファクトが含まれている。

Oracle Warehouse Builder では、ファクトまたはファクト表は、キューブと呼ばれている（「**キューブ（cube）**」を参照）。

フィルタ（filter）

データを選択すること。すべてのデータが、フィルタで使用されるパターン（マスク）と比較され、一致するデータだけが「通過」する。これがフィルタと呼ばれる由縁である。たとえば、電子メールのクライアントとサーバーは、重要なメッセージを抽出してユーザーに通知したり、スパムと見られるテキスト・パターンを検索して削除することができる。**SQL** の WHERE 句には、問い合せするデータについてのフィルタ条件が含まれる。

フォース・クラス・オブジェクト（Fourth Class Object）

Oracle Warehouse Builder アーキテクチャでは、サード・クラス・オブジェクトとフォース・クラス・オブジェクトは、他のオブジェクトに所有されたオブジェクトの相対的な位置付けを表す。所有権が複数層になるオブジェクトのみに当てはまる。たとえば、DIMENSTION_TABLE（ファースト・クラス・オブジェクト）が INDEX_COLUMN を所有する使用例において、INDEX_COLUMN はセカンド・クラス・オブジェクトになる。しかし、ファースト・クラス・オブジェクト CUBE_TABLE がセカンド・クラス・オブジェクト INDEX を所有し、その下に INDEX_COLUMN が所有されている使用例では、INDEX_COLUMN はサード・クラス・オブジェクトになる。

（「**ファースト・クラス・オブジェクト（First Class Object : FCO）**」、「**セカンド・クラス・オブジェクト（Second Class Object : SCO）**」も参照）

物理インスタンス（physical instance）

配置すると**スキーマ**になる関連**データベース**・オブジェクトの集合。データベース・オブジェクトには、表、ビューおよびその他のオブジェクトが含まれる。

物理構造（physical structures）

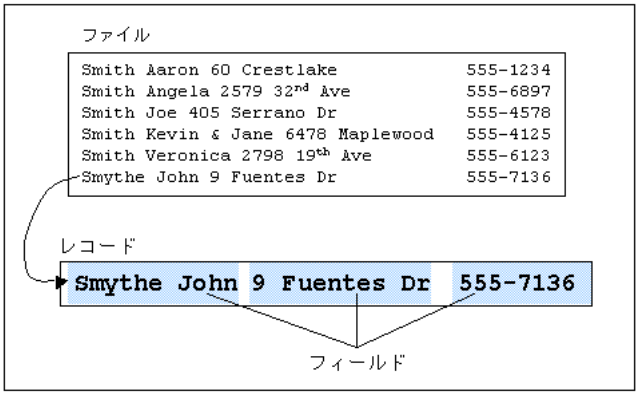
Oracle データベースの物理**データベース**構造には、データ・ファイル、REDO ログ・ファイル、制御ファイルなどが含まれる（「**論理構造（logical structures）**」も参照）。

物理名（physical name）

リポジトリ・オブジェクトの一意の名前。『Oracle Database **SQL** リファレンス』で定義されているように、**スキーマ**・オブジェクトの基本構文規則に準拠する。**DDL** スクリプトの生成に使用される。ユーザーは、新しいオブジェクトの物理名を作成したり、既存オブジェクトの物理名を変更することができる。

フラット・ファイル（flat file）

各**データベース**が 1 つの**表**に格納される、比較的シンプルなデータベース・システム。フラット・ファイルは別の**ファイル**との関連性がなく、別のファイルへのリンクも含んでいない。主にスタンドアロン・リストに使用される。フラット・ファイル同士の関連付けは、そのようにアプリケーションがプログラムされている場合のみ可能である。一方、**リレーショナル・データベース**・システムでは複数の表を使用して情報を格納でき、表ごとに異なる**レコード**形式を使用できる。ファイル同士を関連付ける必要がある場合（たとえば、顧客と注文、ベンダーと購入先など）は、フラット・ファイル・マネージャではなく、リレーショナル・データベース・マネージャを使用する（「**デリミタ付きフラット・ファイル（delimited flat file）**」、「**固定長フラット・ファイル（fixed-length flat file）**」、「**論理レコード（logical record）**」も参照）。



ブリッジ（bridge）

Oracle Warehouse Builder のコンポーネントの 1 つ。これを使用すると、**CWM** 準拠アプリケーション、Oracle Discoverer、Oracle Express、Oracle9*i* **OLAP** サーバーなど、様々なツールとの間で**メタデータ**のインポートやエクスポートを行うことができる。

ブレーク・ポイント（breakpoint）

不連続、変更または中断が発生するポイント。**Oracle Warehouse Builder** では、この用語は、**演算子**が期待どおりに動作しているかどうかを確認するために割り込むことのできる、データ・フロー内のポイントを表す。

プロジェクト（project）

Oracle Warehouse Builder のプロジェクトは、Warehouse Builder **Design Repository** における最大の記憶域オブジェクトである。ユーザーはプロジェクトを使用して、作業を整理できる。プロジェクトは、ソースやターゲットの特定セットの**抽出、変換、ロード**に必要な**メタデータ**定義を格納して整理する。これらの定義には、データ・ソース、ターゲット・ウェアハウス・オブジェクト、ソースからターゲットへのマッピング、変換操作、構成パラメータなどがある。これらの定義は、プロジェクト内のフォルダに整理される。通常、プロジェクトには関連するメタデータがある。複数のプロジェクトを作成すると、設計を組織的に作成できる。

プロセス・フロー（process flow）

プロセス・フローは、マッピングと電子メール、FTP、**オペレーティング・システム**・コマンドなどの外部アクティビティの間に存在する依存関係を示す。アクションの順序がフローとして視覚的に表される。プロセス・フローの基本設計要素には、**アクティビティ**と**推移**が含まれる。Warehouse Builder アーキテクチャでは、プロセス・フローは**ファースト・クラス・オブジェクト**になる。

ベース言語（base language）

Design Repository に割り当てられた言語。すべての**メタデータ**は、この言語で格納される。**リポジトリ**で新しいオブジェクトを作成するには、表示言語をこの言語に設定する必要がある。現時点でサポートされているベース言語はアメリカ英語のみ。

ペイロード（payload）

指定の間隔で実行される論理 ICE 演算の集合をカプセル化するプロトコル構造。ペイロードは、この仕様に含まれるプロトコル定義に従ってフォーマットされた **XML** 文書の単一インスタンスである。

変換（transformation）

データを操作する処理。コピー操作以外の操作は変換である。変換の例としては、複数の**ソース**・オブジェクトからのデータのクレンジング、集計、統合などがある。Warehouse Builder アーキテクチャでは、変換は**ファースト・クラス・オブジェクト**になる。

変換ライブラリ（Transformation Library）

ソース・オブジェクトと**ターゲット**・オブジェクトとの間でデータが送受信される際に、再使用可能なデータ**変換**式を格納する。

ポートレット（portlet）

他の Web サイトやアプリケーションにある情報コンポーネント。これを使用して、カスタマイズされた Web ページを構築できる。たとえば、Warehouse Builder Browser を構成するポートレットを追加し、Warehouse Builder 内を移動して**メタデータ**についてレポートする Web ページへとカスタマイズすることが可能。

マスター / ディテール（master-detail）

マスター・レコードに対応するディテール・レコードがある、複数レコード・タイプの**フラット・ファイル**。たとえば、「部門」情報を含むマスター・レコードと、その部門の各従業員の「従業員」情報を含む、マスター・レコードのディテール・レコードという、2 つの**レコード**・タイプがある簡単なマスター / ディテール・フラット・ファイルが考えられる。

マッピング（mapping）

ソース・オブジェクトと**ターゲット**・オブジェクトとの間の関係およびデータ・フローに関する定義。Warehouse Builder アーキテクチャでは、マッピングは**ファースト・クラス・オブジェクト**になる。

マテリアライズド・ビュー（materialized view）

キューブ（および場合によっては**ディメンション**表）の**集計データ**または結合データ、あるいはその両方のデータを含む計算済の表。これは、サマリー表や集計表とも呼ばれる。マテリアライズド・ビューでは、**問合せ**の結果を別の**スキーマ**・オブジェクトに格納することによって、**表**データへの間接的なアクセスを可能にする。**索引**と同様、マテリアライズド・ビューでは、多量の記憶領域が使用され、マスター表のデータが変更されたときにはリフレッシュする必要がある。また、**SQL** 実行のパフォーマンスを向上させ、SQL アプリケーションおよびユーザーに対して透過的である。索引との相違点は、マテリアライズド・ビューが、SELECT 文、INSERT 文、UPDATE 文、DELETE 文などを使用して直接アクセスできることである。Warehouse Builder アーキテクチャでは、マテリアライズド・ビューは**ファースト・クラス・オブジェクト**になる（「**従来型ビュー（conventional view）**」と対比）。

メジャー（measure）

調査や分析の対象となるデータで、通常は数値データや**加算的**データ。ファクトやメジャーに関する値は、事前に判明しているわけではなく、考察されてから格納される。ファクト / メジャーの例には、販売価格、原価および収益がある。ファクトとメジャーは同義語である。ファクトは主にリレーショナル環境で使用され、メジャーは主に多次元環境で使用される。

メタデータ（metadata）

データおよびその他の構造（オブジェクト、ビジネス・ルール、プロセスなど）を記述するデータ。たとえば、**データ・ウェアハウス**の**スキーマ**設計は、通常、メタデータとして**リポジトリ**に格納され、データ・ウェアハウスの作成と移入に使用するスクリプトの生成に使用される。メタデータはリポジトリに含まれる。

例 : データの場合は、データ・ウェアハウスの作成と移入に使用される、**ソース**から**ターゲット**への**変換**の定義。情報の場合は、リレーショナル・モデリング・ツールの内部に格納される表、列、関連付けの定義。ビジネス・ルールの場合は、1,000 アイテムを販売するたびに行われる 10 パーセントの値引き。

メタデータ・スナップショット（metadata snapshot）

メタデータ履歴管理のために、**Design Repository**（またはリポジトリ内の選択したオブジェクト）の状態を取得する方法。メタデータ・スナップショットと現在の**リポジトリ**の比較レポートや、2 つのメタデータ・スナップショットの比較レポートを生成できる。

完全スナップショットを使用すると、比較レポートを生成するだけでなく、オブジェクトをスナップショットから現在のリポジトリにリストアすることもできるが、**シグネチャ・スナップショット**は比較目的でしか使用できない。**カスケード・スナップショット**は、選択したオブジェクトとそのすべての**子**オブジェクト（存在する場合）の情報を取得するが、**カスケードなしスナップショット**は、選択したオブジェクトの情報のみを取得する。Warehouse Builder アーキテクチャでは、スナップショットは**ファースト・クラス・オブジェクト**になる。

メタデータのアーカイブ（archive metadata）

Metadata Loader（MDL）で、メタデータをバージョン情報とともにエクスポートして、**Oracle Warehouse Builder** の**メタデータ**をバックアップすること（「**メタデータのエクスポート（export metadata）**」、「**メタデータのリストア（restore metadata）**」も参照）。

メタデータのインポート（import metadata）

以前にエクスポートした**メタデータ**を、**Metadata Loader（MDL）**ユーティリティを使用して、**Oracle Warehouse Builder Design Repository** に読み込ませること。インポートしたメタデータは、既存オブジェクトとマージしたり、既存オブジェクトを上書きすることができる。MDL には独自の**ファイル**形式があり、MDL ユーティリティを使用してエクスポートおよびインポートできるのは、.MDL ファイルだけである（「**メタデータのリストア（restore metadata）**」、「**メタデータのエクスポート（export metadata）**」も参照）。

メタデータのエクスポート（export metadata）

Export ユーティリティを使用して、**メタデータ**のコピー（物理ファイルではない）を**リポジトリ**から**抽出**すること。その後、Import ユーティリティを使用して、データをリポジトリにインポートできる（「**Metadata Loader（MDL）**」、「**メタデータのアーカイブ（archive metadata）**」も参照）。

メタデータのリストア（restore metadata）

Metadata Loader（MDL）ユーティリティを使用して、アーカイブされた**メタデータ**を現在の **Design Repository** にインポートすること。メタデータをリストアすると、**ナビゲーション・ツリー**に**プロジェクト**が追加作成され、既存のオブジェクトは上書きされない（「**メタデータのアーカイブ（archive metadata）**」、「**メタデータのインポート（import metadata）**」も参照）。

モジュール（module）

プロジェクト内の格納オブジェクト。**ソース**・オブジェクトと**ターゲット**・オブジェクトの編成に役立つ。モジュールは、**Oracle Warehouse Builder** のメイン・**プロジェクト**・**ナビゲーション・ツリー**上に表示され、データベース、ファイル、アプリケーションおよびプロセス・フロー用の**メタデータ**・コンテナとして使用される。ソース・モジュールには、取得元となる既存のソース・システムに入っているメタデータが含まれる。ターゲット・モジュールには、設計対象のメタデータが含まれる。

モデル（model）

作成する内容を示すオブジェクト。典型的なスタイル、計画または設計。**データ・ウェアハウス**の構造を定義するメタデータ（「**メタデータ（metadata）**」も参照）。

ユーザー・インタフェース（user interface : UI）

ユーザーがソフトウェア・アプリケーションと対話する方法を定義する、メニュー、画面、キーボード・コマンド、マウス・クリック、コマンド言語の組合せ。**Oracle Warehouse Builder** では、グラフィカル・ユーザー・インタフェース、**Oracle MetaBase Plus（OMB Plus）**のスクリプト・インタフェース、いくつかのコマンドライン機能拡張および **SQL** 機能拡張が備わっている。

ユーザー定義プロパティ（user-defined properties : UDP）

ユーザー独自のニーズに応じてメタデータを拡張できるように、**Oracle Warehouse Builder** では、オブジェクトにプロパティを追加作成できる。ユーザー定義プロパティは、**ナビゲーション・ツリー**上のどのオブジェクト・タイプにも作成できる。通常、ユーザー定義プロパティには、オブジェクトに追加するビジネス、設計またはバージョニングの情報などが格納される。ユーザー定義プロパティには、接頭辞 `UDP_` が必要である。

ユーザー名（username）

ユーザーが Oracle **サーバー**および他のユーザーにより識別されるための名前。ユーザー名には必ずパスワードが関連付けられており、Oracle **データベース**に接続するにはこの両方を入力する必要がある。

緩やかに変化する次元（slowly changing dimension : SCD）

データ・ウェアハウスにおいて現在のデータと履歴データの両方を時系列で管理するために作成された**ディメンション**。ディメンション・レコードの履歴を管理する際には、**ETL** の最も重要なタスクの 1 つと見なされ、実装される。

要素（element）

オブジェクトまたはプロセス。たとえば、**ディメンション**はオブジェクト、**マッピング**はプロセスであり、両方とも要素である。

ランタイム環境（runtime environment）

配布後の環境のこと。オブジェクトが配布されたロケーションを指す。**Runtime Repository** には、配布およびランタイム環境に関するデータがすべて格納される。

ランチャー（launcher）

Oracle Warehouse Builder コンソール・ウィンドウのパネル。アクティブな環境を制御する各種のボタンを備えている。このような環境には、プロジェクト環境、管理環境および変換ライブラリ環境がある。

リポジトリ（repository）

メタデータの格納場所。リポジトリ環境には、ビジネスのメタデータの完全セットが格納される。

リモート・データベース（remote database）

デフォルトのデータベース以外の**データベース**。リモート・コンピュータ（特に CONNECT、COPY、**SQL*Plus** のコマンドで参照するリモート・コンピュータ）上に常駐させられる。

リレーショナル・オブジェクト（relational object）

リレーショナル・オブジェクトは、**リレーショナル・データベース**と同様に、表および表から導出されたオブジェクトを使用して、すべてのデータを格納およびリンクする。**Oracle Warehouse Builder** で定義するリレーショナル・オブジェクトは、データの格納に使用される、**データベース**内の物理コンテナである。ウェアハウス作成後の問合せの実行は、リレーショナル・オブジェクトから行われる。リレーショナル・オブジェクトには、表、ビュー、マテリアライズド・ビュー、順序などがある（「**ディメンション・オブジェクト（dimensional object）**」と対比）。

リレーショナル・データベース（relational database）

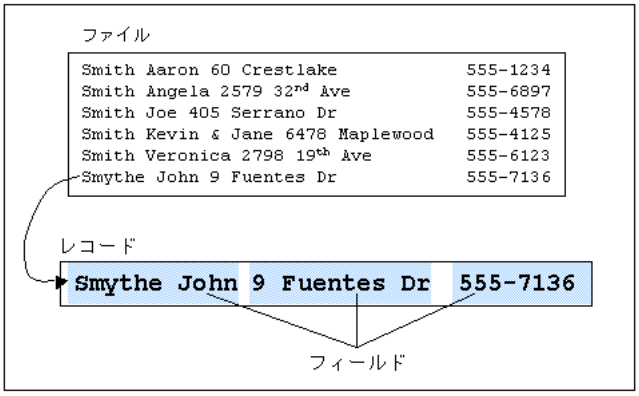
構造化されたデータの集合。同一の列セットを持つ、1 つ以上の行で構成される複数の表を関連付けてデータが格納される。表同士の関係が定義されることにより、複数の表のデータがリンクされる。このリンクは、両方の表に共通する 1 つ以上のフィールドに基づくものである。リレーショナル・データベースでは、データ同士の関連方法や、**データベース**からデータを抽出する方法に関する前提条件がほとんど不要であるため、強力な機能を提供できる。そのため、同じデータベースを様々な視点で表示することが可能である。リレーショナル・システムの重要な機能は、1 つのデータベースを複数の表にわたって展開できることである。この点で、**フラット・ファイル**・データベースとは異なる。フラット・ファイル・データベースでは、各データベースが 1 つの表で自己完結している。

リレーショナル・データベース管理システム（relational database management system : RDBMS）

共有データを格納および取得するためのコンピュータ・プログラム。リレーショナル・システムでは、1 つ以上の行で構成される表にデータが格納される。各行には、同じ列セットが含まれる。Oracle は、リレーショナル・データベース管理システムの 1 つである。他のタイプの**データベース**・システムは、階層型またはネットワーク型データベース・システムと呼ばれる。

レコード（record）

フラット・ファイル・データベース管理システムでは、レコードは、情報の完全セットを表す。レコードはいくつかのフィールドで構成される。各フィールドには、1 つの情報項目が格納されている。レコードの集合は**ファイル**を構成する。たとえば、従業員ファイルには、名前フィールド、住所フィールド、電話番号フィールドを含むレコードが格納される。



列（column）

データの特定のドメインを表す、**データベース**表の垂直領域。列には、列名と固有の**データ型**がある。たとえば、従業員情報の**表**では、全従業員の雇用開始日が 1 つの列を構成する（「**行（row）**」も参照）。

レベル（level）

階層内の位置。たとえば、時間**ディメンション**には、「月」、「四半期」および「年」レベルのデータを示す階層がある。

レベル値の表（level value table）

ディメンションおよび**階層**定義の一部として作成したレベルの値またはデータを格納する**データベース表**（「**ディメンション（dimension）**」、「**レベル（level）**」も参照）。

ローカル・エンキュー（local enqueues）

クラスタ・**データベース**において、共有データ構造への同時アクセスを調整するための**同期**メカニズム。**インスタンス**が 1 つの場合、**ローカル・ロック**と呼ばれる（「**エンキュー（enqueue）**」も参照）。

ローカル・ロック（local locks）

インスタンスが 1 つの場合、共有データ構造への同時アクセスを調整するための**同期**メカニズム。クラスタ・**データベース**では、**ローカル・エンキュー**と呼ばれる。

ロール（role）

ユーザーまたは他のロールに付与される、関連する複数の権限の名前付きグループ。

ロールバック（rollback）

現在の**トランザクション**内で保留中のデータ変更を、**SQL** の ROLLBACK コマンドで取り消すこと（「**コミット（commit）**」も参照）。

ロケーション（location）

ターゲット・システムの配布先の物理ロケーションを表す、**Oracle Warehouse Builder** の**ナビゲーション・ツリー**上のオブジェクト。Warehouse Builder アーキテクチャでは、ロケーションは**ファースト・クラス・オブジェクト**になる（「**コネクタ（connector）**」、「**配布（deployment）**」も参照）。

ロケール（locale）

言語的および文化的設定に関する、特定地域の情報の集合。一般にロケールは、National Language Support データ・ファイルかグローバリゼーション・サポート・データ・ファイルに定義されている言語、地域、キャラクタ・セット、言語設定およびカレンダー情報で構成される。

ロック（lock）

1 人のユーザーにオブジェクトを変更する権限が与えられ、他のユーザーには読取り専用アクセス権のみが与えられるマルチユーザー環境で使用される。プロジェクト共有時のオブジェクトの整合性が維持される。ロックは、1 人のユーザーがコミットまたはロールバックして、開いているエディタやプロパティ・シートを閉じるまで、そのユーザーによって維持される（「**プロジェクト（project）**」、「**コミット（commit）**」、「**ロールバック（rollback）**」も参照）。

論理構造（logical structures）

Oracle **データベース**の論理構造には、表領域、**スキーマ**・オブジェクト、データ・ブロック、エクステント、セグメントなどが含まれる。物理構造と論理構造は別個のものなので、論理記憶域構造へのアクセスに影響を与えることなく、データの物理記憶域を管理することができる（「**物理構造（physical structures）**」と対比）。

論理名（logical name）

「**ビジネス名（business name）**」を参照。

論理レコード（logical record）

フラット・ファイル内では、複数の物理レコードを 1 つの論理**レコード**に対応付けることにより、レコードを論理的に編成できる。

索引

H

I

M

N

O

P

R

S

T

U

V

W

X

Z

あ

か

さ

た

な

は

ま

や

ら

わ