ユーザーズ・ガイド

このガイドでは、Oracle Data Integrator のグラフィカル・ユーザー・インタフェースを構成するグラフィカル・コンポーネントの操作方法について説明します。また、Oracle Data Integrator での開発における一般タスクおよび作業例について説明します。このガイドには、Oracle Data Integrator の機能および操作に関する概念情報やバックグラウンド情報も含まれます。このガイドの対象読者は、Oracle Data Integrator を統合プロセスの開発ツールとして使用する開発者と管理者です。

ドキュメントの構成

このドキュメントの構成は、次のとおりです。

- **第 1 章「概要」**では、統合に関する概念情報と、Oracle Data Integrator の概要について説明します。
- **第 2 章から第 5 章**では、Oracle Data Integrator のグラフィック・モジュールで一般タスクを実行する方法について説明します。
- **第 6 章「Oracle Data Integrator の使用」**では、サポートされる特定のテクノロジと組み合せて Oracle Data Integrator を使用する場合に固有の手順について説明します。

概要

Oracle Data Integrator の概要

Oracle Data Integrator は、バッチ・モード、リアル・タイム・モード、同期モードおよび非同期モードでの、異種システムの間の高効率のデータの移動とトランスフォーメーションを合理化します。革新的でモジュール化された設計手法と、すべての主要なデータベース、データ・ウェアハウス・アプライアンス、分析アプリケーションおよび SOA スイートへの組み込まれた接続性により、ユーザーの生産性を劇的に向上させます。

Oracle Data Integrator に含まれるグラフィカル・モジュールとソフトウェア・エージェントにより、次のことが可能になります。

- アプリケーション・モデルのリバース・エンジニアリング
- データ整合性のチェック
- アプリケーション間のインタフェースの設計、テスト、操作および保守
- インタフェースにより処理されたデータ・フローのチェック、エラーの分離やリサイクリング
- 欠落しているデータ入力の識別

Oracle Data Integrator の対象ユーザー

Oracle Data Integrator モジュールの大部分は、アプリケーションのインタフェース開発プロジェクトに関わっているコンピュータの専門家（プロジェクト・マネージャ、プログラマ、データ管理者など）のために設計されています。たとえば次のようなユーザーです。

- すべてのソース・アプリケーションやターゲット・アプリケーションの、完全で実際的な知識があります。
- インタフェースの操作を担当します。

Oracle Data Integrator を選択する理由

企業では、ある職能専門になっている IT ソリューションがよく見られるようになっています。

このコンテキストでは、データは企業情報システム全体で同期される必要があります。

Oracle Data Integrator では、次のことが可能です。

- データの転送やトランスフォーメーションを含むプロジェクトの生産性を向上させます。このプロジェクトには、移行、パッケージのインストール、インターネット用のセキュアなデータベースの構築、データ・ウェアハウスやデータマートのインストールおよび異種アプリケーションの同期が含まれています。
- 古い IT アプリケーションと新しいアプリケーションの両方で、企業のデータ品質を向上させます。
- 非同期に実行されるインタフェースを持つことによって、アプリケーション間のダイアログを改善します。

インタフェース

インタフェースは、1 つ以上のソースからのターゲット・データストアのロードを定義する 1 組のルールで構成されています。統合プロジェクトは 1 群のインタフェースで構成されています。

インタフェースの作成

コンピュータ・テクノロジの世界は絶えず進化しており、企業は取り巻く環境における発展に短時間で適応する必要があります。このため、次のような目的を持つプロジェクトを絶えず実行しておくことが非常に重要です。

- ERP システムへのアプリケーションの移行
- ERP と ERP に入れられないアプリケーションの間の連携の維持
- 業務データ・ストア（ODS）のロードによる、最新の、統合された情報の提供
- データを整理された状態に維持し、すべてのユーザーの質問に迅速に回答できるようにしておくための、データウェアハウスのロード
- データマートのロードによる、速度とアクセス・ツールの点でユーザーにより適した表示の提供
- 情報を Web で公開するためのデータベースのロード
- 営業員、代理店、サプライヤ、顧客およびサード・パーティがインターネット経由で入力したデータで構成された、セキュリティの制約に厳密に準拠する本番データベースのロード

- 本番データベースからのテスト・データベースのロード

前述のプロジェクトはすべて異なった種類の統合プロジェクトです。それらが必要とする、様々なソリューションと更新メソッドの開発に、想定外のワークロードが発生する可能性があります。

インタフェースのメンテナンス

インタフェースはコードもドキュメントも膨大で、メンテナンスが困難です。これは開発と進化を阻害する要因になります。ただし、スケーラブルなメンテナンスは、プロジェクトのライフ・サイクルに欠かせません。

レプリケーションとトランスフォーメーション

データ・レプリケーション

レプリケーションは、リレーショナル・データベース管理システム（RDBMS）やソフトウェア・エンジニアリング・ワークベンチに内蔵されていることもある、比較的単純な技術的なプロセスです。

レプリケーションを使用すると、同じ数の表と同じデータ検証ルールを持つ 2 つの類似したデータ・モデルの間でデータ・フローを管理することができます。データ整合性制約がソース・モデルとターゲット・モデルの両方で同一の場合は、データ・エラーは処理されません。行やレコードの数がソースとターゲット両方で同一の場合、たとえばソース上に 18 か月、ターゲット上に 36 か月の履歴レベルを持つことができません。

レプリケーション・インタフェースの作成には、アプリケーションの詳細な知識は不要です。最もよく実装されたリソースは、ファイル・コピー、ミラー・ベースのロギング、スナップショット（Oracle）などです。

データ・トランスフォーメーション

データ・トランスフォーメーションの考え方を利用すると、ソースとターゲットの間のデータ・モデリングに実質的な相違を持たせることができます。この相違には、いくつかのレベルがあります。

- ソースとターゲットの間で根本的に異なるデータ・モデルのタイプ（スター、スノーフレーク、正規化済など）
- エラー処理とリサイクリング手順を含む整合性制約（必要に応じて RDBMS で宣言）
- 格納テクノロジ（ファイル、RDBMS など）
- ロギング・レベル（たとえば、ソース上では 18 か月、ターゲット上では 36 か月）

モデリングの相違は、データ・ウェアハウス、データマートおよび移行プロジェクトでは正常です。モデリングの相違は、データ検証ルールの相違、ひいては転送されたデータにエラーがある可能性を示します。

インタフェースによって実行されるトランスフォーメーションのタイプは次のとおりです。

- マッピング。ターゲット情報（表の列）を、場合によっては異なる環境に由来するソース情報に、計算ルール（数値計算、連鎖計算、連結、集約など）を使用してマップすることを可能にします。

- たとえば、アクティブな顧客のみを返す顧客ファイル上のフィルタのように、機能上有用なレコードのみを返すことが可能です。
- 増分または非同期のロード。変化のないデータを繰り返し転送しないようにすることによって、手順の高速化を可能にします。
- データ・モデルのデザインを変更する場合、人工の ID コード（シーケンス）を作成する必要があることがあります。このタイプの ID コードは、移行と、スター・モデリングのデータ・ウェアハウス・プロジェクトの両方で使用されます。

インタフェースのタイプ

実行の頻度、処理されるフローおよび起動モードから、いくつかのタイプのインタフェースが存在します。インタフェースは、ライフ・サイクルの中でタイプが変わることがあります。インタフェース・ツールがない場合、それを実行するにはインタフェースを完全に作成しなおす必要があります。次に説明する用語は排他的なものではなく、1 つのインタフェースが同時にいくつかの分類に属することがあります。

ワン・ショット・インタフェース

ワン・ショット・インタフェースは 1 回のみ実行されます。このインタフェースは、本番環境に投入する直前のアプリケーションの移行に使用されます。

リンク・インタフェース

リンク・インタフェースは 2 つの IT アプリケーションが常時リンクすることを可能にするもので、よく使用されます。

非同期インタフェース

これはタイトなフローで機能するリンク・インタフェースです。このタイプのインタフェースは、ソース・アプリケーションのすべてのイベントを、ほとんど即時に処理します。このタイプのインタフェースは、MOM（Message Oriented Middleware）に基づいているか、そうでない場合はソース・アプリケーション上にポーリング・アクションを必要とします。

バッチ・インタフェース

バッチ・インタフェースは、時刻に基づく頻度（通常は日次）によるゆるやかなフローで機能します。

キャンセル・インタフェースおよび上書きインタフェース

このタイプのバッチ・インタフェースは設定が最も簡単で、実行のたびにリロードするために目的の情報を削除します。ワン・ショット・インタフェースにお薦めします。リンク・インタフェースには、次の短所があります。

- ターゲット・データベース履歴がソース・データベースと同一である必要がありますが、データ・ウェアハウスでは、そういうことはあまりありません。
- リロードの前にデータを削除するため、ターゲット上で参照整合性（外部キー）の制約を宣言することができません。
- 大量のネットワーク・トラフィックが発生します。

増分によるインタフェース

ソース・システムで変更されている情報を処理するだけの、非同期リンク・インタフェースです。このインタフェースは、アクティビティ・ログ、最終更新日など、前回の処理以降に変更されたバッチを特定するあらゆる情報から機能します。

このタイプの非同期インタフェースは、処理する情報量を最小限に抑えるのに好適です。サーバー上の実行時間も改善され、ネットワーク・トラフィックも低減できます。

相違によるインタフェース

このタイプのインタフェースは、増分によるインタフェースに類似していて、ソース・データとターゲット・データの相違により変更を判定します。このタイプのメカニズムは、アップデートの数を減らしたり、データ・ウェアハウスへの情報の格納の日付を確定したりするのに有用です。

その他

インタフェースに関しては、情報システムの規格に特有のニーズが多数存在します。たとえば、あるサイトでは、ソース情報の消失日を維持するためにターゲット・データベースで論理削除を処理し、他のサイトでは、タイム・スタンプとログに関して高レベルの標準を設定する可能性があります。特定のレプリケーション・メカニズムを含むことができるインタフェース・システムを持つことは、そのために重要です。

格納テクノロジ

歴史的および戦略上の理由で、大部分の情報システムには、異なるデータ格納テクノロジが含まれています。この事実により、次のような問題が発生します。

- インタフェース開発チームが複数のスキルを持つ必要があります。
- データの内部表現の変換は、実行パフォーマンス上の理由でコストがかかります。
- 論理データ形式（日付、数など）の変換は、特に、日付についてよく発生する入力データのエラーの場合、生産性の点でコストがかかります。
- 長いトランザクション（障害リカバリの場合）では、いくつかのテクノロジが機能します。
- パフォーマンスは、ミドルウェアや標準のデータ・アクセス言語（SQL 92、ODBC など）の使用により低下します。
- 異なるシステムに由来するデータが一貫していない場合（たとえば、顧客ファイルとインボイス・ファイルの間など）、エラーの分離やリサイクリングが引き起こされます。

最も広く使用されている格納テクノロジは、ファイルおよびリレーショナル・データベースです。ただし、移行時には、インタフェース、継承したテクノロジ（データベース・ネットワーク、階層など）からのアップストリームがしばしば使用されます。ダウンストリームの新しい格納形式は、意思決定システム、主に多次元システム（OLAP）と複数の索引付けでよく見られるようになってきています。インタフェース構築ツールは、企業のすべてのデータ・フローを監督できる程度には、新技術に短時間で適応できる必要があります。

情報ストレージ・サポートに直接アクセスすることは常にお薦めできるわけではありません。このため、ソフトウェア・プログラムの多くは、格納モードを気にせずに一貫した情報を統合または抽出するための通信 API を提供しています。

次の表に、異なるストレージ・サポートへの **Oracle Data Integrator** のアクセス方法を示します。

サポート	アクセス・メソッド
固定ファイル/デリミタ付き（ASCII/EBCDIC）	ネイティブ
出力 XML ファイル	ネイティブ

入力 XML ファイル	ネイティブ
データ・サーバー	JDBC/ODBC
メッセージ・サーバー（MOM）	JMS
LDAP ディレクトリ	ネイティブ
プログラムの API	JCA
その他	交換ファイルを介してロードおよびアンロードするためのユーティリティ・プログラム

データ品質

Oracle Data Integrator でのデータ品質

転送されたデータをチェックすることには多くの利点があります。

- データ使用時の生産性が上昇します。データ・エラーは、アプリケーション開発だけでなく、その操作についても長期にわたって速度を低下させます。
- モデリングを検証します。検出されたエラーの原因はソース・データの不十分な品質とはかぎりません。モデルが不完全だとわかることもあります。アプリケーションを作成しなおす前にデータを移行させることで、新しいデータ・モデルを検証することができ、実際に近い設定のテストも可能になります。
- エンド・ユーザーに対するサービス品質が改善されます。

インタフェースの構築と操作の点でデータ品質を確保することは簡単ではありません。実際、それには誤っているデータの分離とリサイクリングの管理が必要です。これは、特にターゲットに整合性制約をチェックするアクティブなメカニズムが内蔵されている場合に複雑なプログラミングが増えることを意味します。操作時に、誤っているデータの修正手順を実装するようにします（ソース、ターゲットまたはリサイクル・フロー上に）。

データ・フローのチェック

データ・フロー・コントロールとは、インタフェースのソース・データをターゲット上の必須整合性レベルと比較してチェックすることです。データはターゲットに統合化される前にチェックされます。エラーが検出された場合、すべてのデータを（正しいデータも含め）統合しないか、正しいデータのみを統合するかのどちらかになります。いずれにしろ、誤っているデータを（必要な訂正を加えた後で）リサイクルすることは有用です。分離したエラーのリサイクルと訂正が意味を持つのは、増分ロードの場合だけです。この場合は、それぞれのインタフェース実行時にデータが自動的に返されないからです。

このタイプの手順をシミュレートすると有用です。これは、今でもこれが、別のデータ・モデルで宣言された整合性ルールと比較してデータ・セットをチェックする唯一の手段だからです。シミュレーションとは、ターゲット構造内のデータを更新しないでインタフェースを起動することです。

静的データ管理

モデル内のデータの整合性は、インタフェースでの使用に関係なく、エラー修正の所要時間を評価するため、および**アプリケーションの品質監査**を実行するためにチェックできます。

もちろん、データ品質管理は、データが含まれる格納テクノロジ内に存在しないルールにのみ適用することができます。このソース・テクノロジによってチェックされないルールは、セマンティック・リンク、妥当性限界、妥当性ドメイン、書式といった別のメタデータ・ディクショナリで宣言します。

ターゲット・データの品質

ターゲット・データの品質は、ターゲット・テクノロジがそのデータをチェックしない場合、内部メカニズム（トリガーまたは宣言部分整合性）によっていつでも検査できます。データ整合性はプログラムや SQL 問合せによって検証されます。これらの反復的な手動チェックを避けるには、ソースと同じ方法でこのタイプのチェックを自動化できることが重要です。

誤っているデータの処理

誤っているデータは、ソース上、ターゲット上（エラーが統合化されている場合）、またはリサイクルされたフロー上で修正できます。必要に応じて情報システムの任意の場所にあるデータを修正しつつコンサルティング用の中心的ツールを使用すると、マルチテクノロジのコンテキストで非常に有用である場合があります。

データ品質管理が可能である事例

Data Integrator に搭載されている品質管理機能の大部分は、リレーショナル・テクノロジにのみ適用可能です。

そのような機能は次のとおりです。

- ターゲット・データベース・ルールに応じてのフローのチェック
- 静的データのチェック（ソース上またはターゲット上）
- エラーのリサイクリングや分離
- 未統合の誤っているデータの参照と（手動修正用の）編集（フローまたはクリーンアップ）
- ソース表やターゲット表上の誤っているデータの参照と（手動修正用の）編集
- 結合（参照の一貫性）、妥当性限界、妥当性ドメイン、主キーと代替キーおよび複合的ルールのチェック
- エラーのリサイクリングと分離
- エラーの位置特定

Oracle Data Profiling と Data Quality for Data Integrator

Oracle Data Profiling および **Oracle Data Quality for Data Integrator**（あわせて **Oracle Data Quality Products** とも呼ばれる）は、Oracle Data Integrator のインライン・データ品質機能を拡張して、より高度なデータ・ガバナンス機能を提供します。

Oracle Data Profiling

Oracle Data Profiling は、データ調査と品質監視用のツールです。このツールを使用すると、ビジネス・ユーザーはメトリックを通じてデータの品質にアクセスしたり、そのデータに基づくルールを確認または推測したり、時間経過に伴うデータ品質の推移を監視したりすることができます。

Oracle Data Quality for Data Integrator

Oracle Data Quality for Data Integrator は、きわめて複雑なデータ品質ニーズにも対応する、包括的で優れたデータ品質プラットフォームです。強力なルールベース・エンジンと、堅牢でスケーラブルなアーキテクチャによって、企業のデータ統合戦略の中心を成すデータに対し、品質の確保、および名前やアドレスのクレンジング機構を提供します。

詳細情報

Oracle Data Profiling および Oracle Data Quality for Data Integrator の詳細は、Oracle Data Quality Products に付属する次のドキュメントを参照してください。

- 『Oracle Data Quality for Data Integrator スタート・ガイド』
- 『Oracle Data Quality チュートリアル』
- 『Oracle Data Quality Products ユーザーズ・ガイド』
- 「Oracle Data Quality の操作」も参照してください。

生産性

インタフェースには、数日から数か月まで変動するワークロードが必要です。過去にはファイル間のデータ移行が短時間で実装されることもありましたが、DBMS、MOM、LDAP、XML など多数のテクノロジが使用される現在では、インタフェースの作成に予想外の時間がかかることがあります。改善された点を示します。この負荷を生み出す要因を示します。また、これを改善する方法も示します。

RDBMS の進化

RDBMS では、主な進歩は誤っているデータの管理に関するものです。リレーショナル・データベースはアプリケーション・モデルに準拠しないデータの入力を拒絶します。実装されているメカニズムは、異なるタイプの整合性ルールの暗黙的チェックです。データ型（日付、通貨など）、主キー、代替キー、外部キー（表間のリンク）および複合ルールがあります。このコンテキストでは、インタフェースの構築で、モデルのたった 1 つの部分を考慮することは困難です。たとえば、エンティティ「Invoices」のロードの前に、エンティティ「Customers」「Payment」および他のパラメータ表を事前にロードしておく必要があります。最後に、このメカニズムがプロジェクトの組織に影響することに注意してください。事実、各インタフェースの構築とテストは、データ・モデルの機能的な依存関係の順序に従って行う必要があります。

さらに、インタフェースは、わずかなデータ・エラーが大きく不正確な結果につながるデータ・ウェアハウスのようなオープン・システムをより多くロードするようになっています。データが規則的に追加されるため、ターゲット・アプリケーションのテストによってこれらのエラーを検出することは困難です。また、ユーザーは自分の情報リクエストを、強力なアクセス・ツールを

使用して自分で構築します。このタイプのプロジェクトでは、すべての入力データが、すべてのデータ整合性ルールを含んでいるセマンティック・モデルに関して体系的にチェックされる必要があります。

この分野でデータ・アクセス・ツールが開放されたため、意思決定データベースは、毎日ロードすることが必要になっています。頻度レベルを高くすると、実行速度の速いインタフェースが必要になり、データ・エラーを阻止できません。このタイプのインタフェースをプログラミングするのは、データの品質管理がない、キャンセルして上書きするタイプの月次ローディングよりずっと複雑です。

テクノロジとニーズの進化

企業は、外部のリクエスト（カスタマ、サプライヤなど）により迅速に応えるために、外界に開かれ、アプリケーション間の対話を合理化する必要があります。そうしたニーズに応えるために市場が提供する新しいテクノロジ（MOM、XML、LDAP ディレクトリ）には、新しいアクセス方法と新しいスキルが必要です。

このコンテキストでは、インタフェース構築は技術的に複雑になり、多数の異なるスキルを必要とします。

正しいツールの重要性

インタフェースはいくつかのタイプの繰り返し使用するルール（マッピング、トランスフォーメーション、集約、フィルタなど）により駆動されます。そのプログラミングと動作モード、およびエラー管理についても同様です。これらのルールの設計、プログラミング、文書化および保守のための統合ツールがない場合、各ルール上のすべてのレベル（設計、文書化、プログラミング、操作、保守）で干渉が繰り返し発生します。インタフェースの管理ルールを格納し、解釈できるツールを使用すると、文書化とプログラミングのフェーズが 90%以上削減されます。さらに、大部分のインタフェース特性によって誘発される補足的なロードもなくなります。

ツールを使用すると、ルールの保守は、通常 1 人だけで処理できる、非常に孤立した運用になります。このため、ターゲット・データベースにより高いロギング・レベルを追加する場合は、キャンセルして上書き（削除してから各実行時にターゲットを再作成）するように作成されたインタフェースを完全に作成しなおす必要があります（たとえば、ターゲット・データベースに 5 年、ソースに 18 か月）。これらのルールを格納するツールを使用する場合、行う必要があるのは、トランスフォーメーション・ルール（マッピング、計算、フィルタ、集約など）を変更せずに、インタフェースのタイプを変更することだけです。

ナレッジ・モジュール

ナレッジ・モジュールの概要

ナレッジ・モジュール（KM）は、Oracle Data Integrator Open Connector テクノロジのコンポーネントです。KM には Oracle Data Integrator が特定のテクノロジに対する特定の組合せのタスクを実行するために必要とする知識が含まれています。

KM は、JDBC、JMS、JCA などの接続性レイヤーとの組合せで、テクノロジに対して定義されたタスク（たとえばそのテクノロジへの接続、テクノロジからのデータの抽出、データのトランスフォーメーション、そのチェック、その統合など）を実行する Open Connector を定義します。

Open Connector には、次のものの組合せが含まれます。

- 接続戦略（JDBC、インスタンス用のデータベース・ユーティリティ）
- 関係するテクノロジの正しい構文またはプロトコル（SQL、JMS など）
- すべての一時および作業用の表、ビュー、トリガーなどの作成と消去の制御
- データ処理とデータ・トランスフォーメーションの戦略
- データ移動オプション（ターゲット表の作成、挿入/削除、更新など）
- テクノロジ機能により異なるトランザクション管理（コミット/ロールバック）

様々なタイプのナレッジ・モジュール

Oracle Data Integrator の Open Connector は、6 つの異なるタイプのナレッジ・モジュールを使用します。

- **RKM（リバース・ナレッジ・モジュール）**は、特定のテクノロジのデータ・モデルの、カスタマイズされたリバース・エンジニアリングを行うために使用されます。
- **LKM（ロード・ナレッジ・モジュール）**は、ソース・データベースの表や他のシステム（ファイル、ミドルウェア、メインフレームなど）からデータを抽出するために使用されます。
- **JKM（ジャーナル化ナレッジ・モジュール）**は、変更追跡用にソース・データベースのデータ変更（挿入、更新および削除）の日誌を作成するために使用されます。
- **IKM（統合ナレッジ・モジュール）**は、ターゲット表にデータを統合（ロード）するために使用されます。
- **CKM（チェック・ナレッジ・モジュール）**は、ソースおよびターゲットの制約違反がないことをチェックするために使用されます。
- **SKM（サービス・ナレッジ・モジュール）**は、データ・サービスを作成するために必要なコードを生成するために使用されます。

動作の仕組み

設計時

Oracle Data Integrator でインタフェースを設計する場合、インタフェースには、データのロード、データのチェック、データ統合などいくつかのフェーズが含まれます。各フェーズで、次の項目を定義します。

- このフェーズの機能ルール（マッピング、制約など）
- このフェーズで使用されるナレッジ・モジュール。このフェーズのナレッジ・モジュールを構成するには、そのオプションを使用します。

実行時

Oracle Data Integrator は機能ルール、ナレッジ・モジュール、ナレッジ・モジュール・オプションおよびリポジトリに制約されているメタデータ（トポロジ、モデルなど）を使用して、定義したジョブを処理するタスクのリストを自動生成します。タスクには、そのジョブの接続、トランザクション管理および該当するコードが含まれます。これらのタスクは Open Connector を介してエージェントによって調整され、関係するソース、ターゲットおよびステージング領域サーバーによって実行されます。

ナレッジ・モジュールのカスタマイズ

Oracle Data Integrator に含まれ、大部分の標準的なデータ転送と統合ニーズをカバーする知識モジュール以外のナレッジ・モジュールは完全にオープンで、そのソース・コードは管理者が認証したすべてのユーザーに対して公開されます。これにより、クライアントとパートナは Oracle Data Integrator Open Connector を容易に拡張できるため、特定の戦略に対応したり、異なる手法を実装したり、他のテクノロジを実装したりできます。

ナレッジ・モジュールは容易にエクスポートしてリポジトリにインポートできるため、インストールした複数の Oracle Data Integrator の間で容易に配布できます。同時に、Oracle Data Integrator には KM のコードを暗号化するオプションがあるので、パートナとクライアントが知的財産（たとえば特定の手法、特定のテクノロジの高度な使用など）を保護することができます。

デザイナ

デザイナの概要

デザイナ・モジュールで操作できるものは次のとおりです。

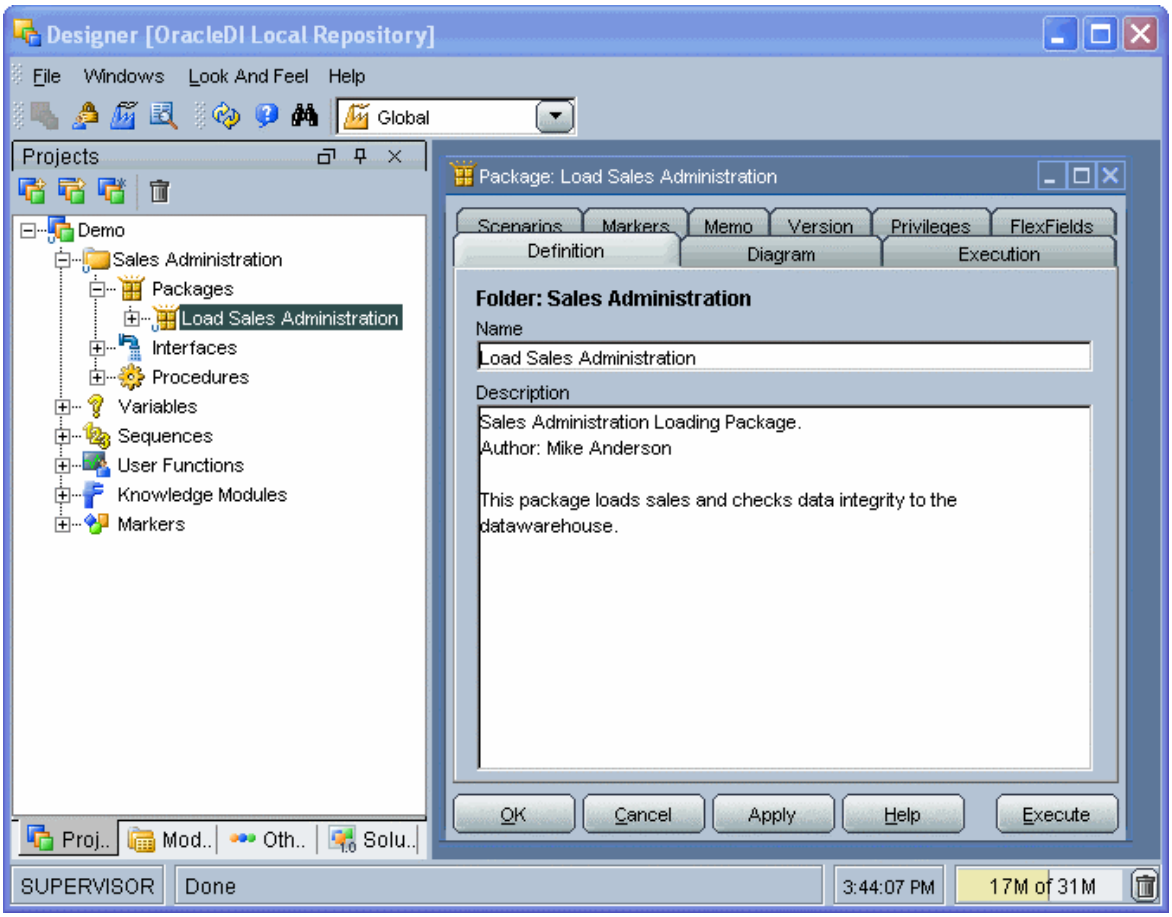
モデル: データとアプリケーションの構造の説明

プロジェクト: デザイナで作成される開発物

デザイナ・モジュールはこの情報を作業リポジトリに格納します。トポロジおよびセキュリティ情報はマスター・リポジトリで定義されているものを使用します。

デザイナのインタフェース

デザイナの GUI は次のように表示されます。



メニュー

メニューでは、プルダウン・メニューから次の機能にアクセスできます。

- インポート/エクスポート
- ウィザード
- オプションの表示
- モジュールまたはツリー表示のオープン
- ユーザーのパスワードおよびオプションの変更

ツールバー

ツールバーからは、次の操作を実行できます。

- 他のモジュールの起動
- ツリー表示のリフレッシュ
- オンライン・ヘルプの起動
- デフォルトの**コンテキスト**を選択します。

選択された**コンテキスト**が、アプリケーション・ウィンドウで選択されるすべてのコンテキストのデフォルトとして使用されます。また、データが参照されると（データストアでデータを右ク

リック）、データはツールバーで定義されたコンテキストで表示されます。たとえば、コンテキストが「開発」の場合、「開発」コンテキストでデータが参照されます。安全のため、すべてのコンテキストで認証されている場合でも、不適当な操作をしないように、コンテキストを変更する際には常に確認のパスワードを求められます。メニューバーには、許可されているコンテキストのみが表示されます。

ツリー表示

現行ユーザーが使用できるデザイナ・オブジェクトは、**プロジェクト**、**モデル**、**ソリューション**および**その他**（ユーザー・ファンクション、グローバル変数および順序）の各ツリー表示に整理して表示されます。

各ツリー表示は、メイン・ウィンドウの両側にドッキングできるフローティング・フレームに表示されます。フレームは重ねることもできます。複数のフレームを重ねた場合は、フレーム・ウィンドウの下部に表示されるタブから各フレームにアクセスできます。

ツリー表示フレームは、フレームのタイトルまたはタブを選択してドラッグすることにより、移動したり、ドッキングしたり、重ねたりできます。ツリー表示の位置を固定するには、**「ウィンドウ」**メニューから**「ビューのロック」**を選択します。

ツリー表示フレームがウィンドウに表示されないか、閉じている場合は、**「ウィンドウ」→「ビューの表示」**メニューを使用します。

各ツリー表示では、次の操作が可能です。

- ルート・オブジェクトの挿入またはインポート（フレーム・タイトルで該当するボタンをクリック）
- ノードの展開または折りたたみ（ノードをクリック）
- オブジェクトに関連付けられているメソッド（編集、削除など）のアクティブ化（ポップアップ・メニューを使用）
- オブジェクトの編集（オブジェクトをダブルクリック、または**ワークベンチ**にドラッグ・アンド・ドロップ）

ワークベンチ

編集または表示されているオブジェクトのウィンドウは、**ワークベンチ**に表示されます。

モデル

モデルの概要

モデルとは、物理スキーマに含まれている**データ構造**に対応するデータストアのセットです。モデルは**モデル・フォルダ**に整理することができます。

- 「モデルの作成およびリバース・エンジニアリング」を参照してください。

モデル・フォルダ

モデル・フォルダはモデルを集めてグループ化するオブジェクトです。たとえば、特定のテクノロジに基づくすべてのモデル、特定のサイトにあるすべてのモデルまたは特定のプロジェクトで使用するすべてのモデルをグループ化することができます。

サブモデル

サブモデルは、モデルのデータストアを階層構造に整理および分類するために使用されるオブジェクトです。構造のルートはモデル自身です。

リバース・エンジニアリング

モデルは、内部にデータストアがない状態で作成されます。モデルを**リバース・エンジニアリング**すると、データ構造を自動的に取得して Oracle Data Integrator でモデルのデータストアを定義することができます。モデルには次の 2 種類があります。

- **標準**リバース・エンジニアリングでは、標準 JDBC 機能を使用してメタデータをリクエストします。
- **カスタマイズ**されたリバース・エンジニアリングは、指定されたテクノロジ特有の方法で指定されたテクノロジ特有のリバース・ナレッジ・モジュール（RKM）を使用してメタデータを取得します。

データストア

データストアはデータを表構造として説明します。データストアは**列**で構成されています。

データストアは Oracle Data Integrator のリレーショナル・モデル内で定義されています。このため、次の要素をデータストアにアタッチすることが可能です。

キー

キーは、それぞれのデータストア行を一意に識別するデータストア列のセットです。それが索引でもある場合は、行アクセスも最適化されることがあります。一部のドライバはリバース・エンジニアリング時にキーの説明を取得します。リポジトリ内で直接キーを定義することも可能です。

参照

参照は 2 つのデータストアの間の機能リンクです。リレーショナル・モデルでは外部キーに対応します。たとえば INVOICE データストアが顧客番号を使用して CUSTOMER データストアを参照する場合です。

条件とフィルタ

条件または**フィルタ**は、SQL をサポートする RDBMS に基づいてデータストアにアタッチされている WHERE タイプの SQL 式です。データストア内のデータをフィルタまたはチェックします。

ジャーナル化

ジャーナル化は、データに対する変更を追跡することです。ジャーナル化は、Oracle Data Integrator では変更のないデータを転送しないように使用されます。この機能は、データ同期化とレプリケーションなど、多くの用途があります。

ジャーナル化は、ある特定のタイプのテクノロジに基づいて、モデル、サブモデルまたはデータストアに適用できます。

モデルの作成およびリバースエンジニアリング

モデルの作成

モデルは、物理スキーマに格納されたデータ構造に対応するデータストアのセットです。モデルは、常に**論理スキーマ** に基づきます。特定の**コンテキスト**内で、**論理スキーマ**は**物理スキーマ**に対応します。この物理スキーマのデータ・スキーマには、データストアとして表現される表、ファイル、JMS メッセージ、XML ファイルの要素などの物理構造が含まれます。

重要: 一般的に使用されるテクノロジには、独自のリバース・エンジニアリング方法とモデル作成方法があり、詳細は「Oracle Data Integrator の使用」に記載されています。作業を進める前にこの項を参照してください。

モデルを作成する手順:

1. **デザイナ**に接続します。
2. ツリーで**「モデル」**を選択します。
3. 右クリックし、**「モデルの挿入」**を選択します。
4. **「定義」**タブで、**「名前」**フィールドを入力します。
5. **「テクノロジ」**フィールドで、モデルのテクノロジを選択します。
6. **「論理スキーマ」**フィールドで、モデルの基礎となる論理スキーマを選択します。
7. **「リバース」**タブに移動し、モデルのリバース・エンジニアリングに使用する**「コンテキスト」**を選択します。**「適用」**をクリックします。

これでモデルは作成されましたが、データストアはまだ格納されていません。

モデルのリバース・エンジニアリング

モデルは、データストアのない状態で作成されます。モデルの**リバース・エンジニアリング**を行うと、モデルのデータストアを定義するためのデータ構造を自動的に取得できます。モデルには次の 2 種類があります。

- **標準**リバースエンジニアリングでは、標準の JDBC 機能を使用してメタデータを取得します。
- **カスタマイズ**・リバースエンジニアリングでは、リバース・ナレッジ・モジュール（RKM）を使用し、指定のテクノロジ用に定義されたメソッドを使用して、メタデータを取得します。

注意: カスタマイズ・リバース・エンジニアリングでは、より完全で全体的なカスタマイズが可能ですが、実行方法も複雑になります。RKM は、一部のテクノロジにのみ提供されています。

標準リバース・エンジニアリング

標準リバース・エンジニアリングを実行する手順:

1. **モデル**の**「リバース」**タブに移動します。
2. 次のフィールドを入力します。
   - **標準**
   - **コンテキスト:** リバース・エンジニアリングに使用するコンテキスト

- **リバースエンジニアリングするオブジェクトの型:** リバース・エンジニアリング・プロセスで処理する必要のあるオブジェクト・タイプのリスト

3. **「選択的リバース」**タブに移動します。
   - **「選択的リバース」**、**「新規データストア」**および**「リバースするオブジェクト」**の各ボックスを選択します。
4. リバース・エンジニアリング対象のデータストアのリストが表示されます。リバース・エンジニアリングするデータストアについては、選択したままとします。
5. **「リバース」**ボタンをクリックし、次に**「はい」**をクリックして変更を承認します。
6. Oracle Data Integrator により、選択したデータストアのリバース・エンジニアリング・プロセスが開始されます。プログレス・バーが表示されます。

リバース・エンジニアリングされたデータストアが、モデルの下に表示されます。

「リバース」および**「選択的リバース」**タブのオプションを使用すると、リバース・エンジニアリングを詳細に設定できます。詳細は、「モデル」を参照してください。

カスタマイズ・リバース・エンジニアリング

RKM を使用してカスタマイズ・リバース・エンジニアリングを実行する手順:

警告: RKM を使用する場合、その RKM をプロジェクトにインポートする必要があります。リバース操作で使用できるのは、インポートされた RKM のみです。

1. **モデル**の**「リバース」**タブに移動します。
2. 次のフィールドを入力します。
   - **カスタマイズ済**
   - **コンテキスト:** リバース・エンジニアリングに使用するコンテキスト
   - **オブジェクト・タイプ:** リバース・エンジニアリングするオブジェクトのタイプ（表、ビューなど）
   - **マスク:** %
   - **KM の選択:** `RKM <technology>.<name of the importation project>`
3. **「リバース」**ボタンをクリックし、次に**「はい」**をクリックして変更を承認します。
4. **「OK」**をクリックします。
5. **セッションを開始しました**ウィンドウが表示されます。**「OK」**をクリックします。

Oracle Data Integrator のログでモデルのリバース・エンジニアリング操作を追跡して確認します。正しく終了すると、リバース操作されたデータストアが、ツリー内のリバース・モジュールの下に表示されます。**「リバース」**および**「選択的リバース」**タブのオプションを使用すると、リバース・エンジニアリングを詳細に設定できます。詳細は、「モデル」を参照してください。また、RKM の説明も参照してください。

サブモデルの作成および編成

サブモデルは、階層構造内でモデルのデータストアを編成および分類できるオブジェクトです。構造のルートはモデルです。

分類操作は、次のように実行します。

- リバース・エンジニアリング中に、RKM によってサブモデルが作成され、それらのサブモデルにデータストアが挿入される場合があります。
- データストアをモデルにドラッグ・アンド・ドロップすることにより、手動で実行します。
- モデル名に基づく分類により、自動的に実行します。

サブモデルを作成する手順:

1. ツリー・ビューで**モデル**または**サブモデル**を選択します。
2. 右クリックし、**「サブモデルの挿入」**を選択します。
3. **「定義」**タブで、**「名前」**フィールドを入力します。
4. **「OK」**をクリックします。

データストアを含まない新規サブモデルが作成されます。

手動でデータストアをサブモデルに分類する手順:

1. ツリー・ビューで**データストア**を選択し、**サブモデル**内にドラッグ・アンド・ドロップします。

データストアがモデルから消去され、サブモデルに表示されます。

サブモデルへのデータストアの自動配布を設定する手順:

1. **サブモデル**の**「配布」**タブに移動します。
2. 次のフィールドを入力します。
   - **データストア配布ルール:** 処理の対象とし、自動割当てマスクと比較対照するデータストアを決定します。
   - **自動配布なし:** データストアは処理されません。
   - **すべての未分類のデータソースを自動配布...:** サブモデル・ツリーのルート・モデルに存在するデータストアが処理されます。
   - **すべてのデータストアの自動配布:** モデル（およびサブモデル）のすべてのデータストアが処理されます。
   - **自動割当てマスク:** このサブモデルに分類するデータストア名のパターン。
   - **リバース後のマスク適用順序:** リバース操作の最後に、すべてのルールがマスク適用順序で適用されます。したがって、すべてのデータストアに対して上位の順序にあるルールが優先されます。未分類のデータストアに対して上位の順序にあるルールは、他のルールのパターンにより無視されたデータストアにのみ適用されます。リバース操作の最後の時点で、新規データストアは未分類とみなされます。サブモデルですでに分類されているデータストアは、それぞれのサブモデルに付加されたままです。
3. **「OK」**をクリックします。

モデル・フォルダへのモデルの編成

モデルは、**モデル・フォルダ**に分類できます。モデル・フォルダには、他のモデル・フォルダを含めることができます。

モデル・フォルダを作成する手順:

1. **「モデル」**ビューを開きます。
2. 右クリックし、**「モデル・フォルダの挿入」**を選択します。

3. フォルダの**「名前」**と詳細な**「説明」**を入力します。
4. **「OK」**をクリックします。

空のモデル・フォルダが表示されます。

フォルダにモデルを移動する手順:

1. **「モデル」**ビューを開きます。
2. モデルを選択し、移動先となるモデル・フォルダのアイコンにドラッグ・アンド・ドロップします。

モデルが現在の場所から選択したモデル・フォルダに移動します。

注意: モデルは、一度に 1 つのフォルダにのみ格納できます。

注意: モデル・フォルダも、他のモデル・フォルダに移動できます。

注意: マーカーを使用してモデルを編成することも可能です。

データストアの作成

データストアを作成する手順:

1. ツリー・ビューで**モデル**または**サブモデル**を選択します。
2. 右クリックし、**「データストアの挿入」**を選択します。
3. **「定義」**タブで、次のフィールドを入力します。
   - **データストアの名前:** この名前は、ツリー内に表示され、プロジェクトからデータストアを参照する際に使用されます。
   - **リソース名:** 格納先のデータ・サーバーによって認識される形式でのオブジェクトの名前。これは、表、ファイル、JMS キューなどの名前です。
   - **別名:** これは、チェックおよびフィルタ式で使用される短縮名です。
4. データストアが**ファイル**・モデルに作成される場合は、次のようにします。
   1. **「ファイル」**タブに移動します。
   2. このタブのフィールドを入力します。詳細は、「データストア」を参照してください。
   3. データストアが**デリミタ付き**ファイルの場合は、**「列」**タブに移動し、**「リバース」**ボタンをクリックして列のリストをリバース・エンジニアリングします。
   4. データストアが**固定**ファイルの場合は、ウィザードを使用して列を定義できます。
   5. ファイル構造を記述する COBOL コピーブック・ファイルがある場合は、「COBOL コピーブックのリバース・エンジニアリング」を参照してください。
   6. **「OK」**をクリックします。

データストアが作成されます。ファイル固有のリバース・エンジニアリングを実行していなければ、データストアに列は含まれません。

データストアに列を追加する手順:

1. **データストア**の**「列」**タブで、  をクリックします。
2. 空の行が表示されます。列に関する情報を入力します。フィールドは、データストア・タイプに応じて変化します。これらのフィールドの詳細は、「データストア」を参照してください。

3. データストアに追加する列ごとに、手順 1 および 2 を繰り返します。
4. **「OK」**をクリックします。

注意: 固定ファイル・データストアの場合、列の作成を迅速化する目的で、列の開始位置を自動的に計算する**「自動調整」**オプションを使用できます。

データストアから列を削除する手順:

1. **データストア**の**「列」**タブで、削除する列を選択します。
2. をクリックします。列がリストから削除されます。

データストアのデータの編集および表示

データストアのデータを表示する手順:

1. **モデル**の**データストア**を選択します。
2. 右クリックして**「データの表示」**を選択します。

編集不可能なグリッド内にデータが表示されます。

データストアのデータを編集する手順:

1. **モデル**の**データストア**を選択します。
2. 右クリックして**「データ」**を選択します。

編集可能なグリッド内にデータが表示されます。**「SQL」**ボタンを使用すると、データストアのデータ表示問合せを変更し、データストアに任意の問合せを実行できます。

注意: 表示されるデータは、デザイナのツールバーに示された現在のコンテキストにおいて、モデルの論理スキーマに対応する物理スキーマに格納されたデータです。

注意: データストアのデータを編集できるのは、使用される接続とデータ・サーバーのユーザー権限で許可されており、かつデータストア構造によりデータストアの各行（主キーなど）を識別できる場合です。

共通フォーマット・デザイナ

概要

共通フォーマット・デザイナ

共通フォーマット・デザイナ（CFD）を使用すると、デザイナのユーザー・インタフェースでデータ・モデルを迅速に設計できます。データ・モデルは、完全に新規モデルとして設計するか、他のデータ・モデルの要素を使用して構築することができます。共通フォーマット・デザイナでは、モデルをデータ・サーバーに実装するためのデータ定義言語（DDL）スクリプトが自動的に生成されます。

共通フォーマット・デザイナでは、ユーザー・インタフェースを通じて既存のモデル設計を変更できます。また、Oracle Data Integrator で記述されているデータ・モデルとデータ・サーバーの実装の間に存在する差異を同期するための DDL スクリプトを自動的に生成できます。

たとえば、共通フォーマット・デザイナを使用して、様々な異なるソースを集約し、業務用のデータストア、データマートまたはマスター・データの基準となる形式を作成できます。

データ・モデルの設計

ダイアグラムとは

ダイアグラムは、サブモデル（またはデータ・モデル）に格納されたデータストアのサブセットのグラフィカル・ビューです。データ・モデルには、関連する複数のダイアグラムを含めることができます。

ダイアグラムは次のように作成します。

- モデルおよびサブモデルからデータストアを集約します。
- 空のデータストアを作成し、次の操作を実行します。
- これらのデータストアに他のデータストアの列を集約します。
- これらのデータストアに新規列を作成します。

他のモデルからデータストアと列を集約する理由

他のモデルまたはサブモデルからダイアグラムにデータストアと列を集約すると、Oracle Data Integrator により、モデルに追加されたデータストアまたは列の元のデータが追跡されます。元のデータストアおよび列に対する参照により、Oracle Data Integrator では、集約されたデータストアに対する統合インタフェースを自動的に生成できます（インタフェース IN）。

自動インタフェース生成は、他のモデルのデータストアおよび列から作成されていないデータストアと列をロードする場合には機能しません。ただし、統合インタフェースを手動で作成することや、自動マップされていない列に対して生成されたインタフェースを完成することは可能です。

グラフィカル・シノニム

ダイアグラムでは、データストアは**グラフィカル・シノニム**として複数表示されることがあります。シノニムは、データストアのグラフィカル表現です。グラフィカル・シノニムを使用すると、ダイアグラムが見やすくなります。

ダイアグラムからデータストアを削除する場合、デザイナにより、シノニムを削除するのか（データストアは残ります）、データストア自体を削除するのか（そのデータストアのすべてのシノニムが削除されます）を尋ねられます。

ダイアグラムの各参照は、データストアのグラフィカル・シノニムに関連付けられます。グラフィカル・シノニムは自由に作成することが可能で、参照のグラフィカル表現は、その参照に関連するデータストアの任意のグラフィカル・シノニムに移動できます。

ダイアグラムの使用方法

ダイアグラムでは、ダイアグラムから直接使用できるポップアップ・メニューを通じて、ダイアグラムに表示されるすべてのモデル要素（データストア、列、参照、フィルタなど）を編集できます。
ダイアグラムに加えた変更は、即座にモデルに適用されます。

新規ダイアグラムを作成する手順:

1. モデル・ビューで、データ・モデルを開いて**「ダイアグラム」**ノードを選択します。
2. 右クリックし、**「ダイアグラムの挿入」**を選択します。
3. **「ダイアグラム名」**と**「説明」**を入力します。

モデルの**「ダイアグラム」**ノードの下に、新規ダイアグラムが表示されます。

ダイアグラムに既存のデータストアを挿入する手順:

1. 「ダイアグラム」編集ウィンドウで、**「ダイアグラム」**タブを選択します。
2. モデルからデータストアを選択します。
3. そのデータストアをダイアグラムにドラッグします。
   現在のモデル（サブモデル）とは異なるモデル（サブモデル）からデータストアをドラッグすると、デザイナにより、そのデータストアのコピーを現在のモデルに作成するよう求められます。
   データストアがダイアグラムにすでに存在する場合、新規グラフィカル・シノニムを作成するか、データストアの複製を作成するよう求められます。

データストアの新規グラフィカル・シノニムがダイアグラムに表示されます。
別のモデルからデータストアを追加した場合、または複製を作成した場合は、新規データストアがモデルに表示されます。

注意: ダイアグラムにすでに存在するデータストアのグラフィカル・シノニムを作成するには、データストアのポップアップ・メニューの「グラフィカル・シノニムの作成」を選択します。

注意: ダイアグラムに挿入された表と表の間に元のモデルで参照（結合）が存在する場合、それらの参照もコピーされます。

ダイアグラムで新規データストアを作成する手順:

1. ダイアグラム・ウィンドウで、**「ダイアグラム」**タブを選択します。
2. ツールバーの**「データストアの追加」**  ボタンをクリックします。
3. ダイアグラム・ワークベンチをクリックします。
4. 新規データストア・ウィンドウが表示されます。データストアに対する他の操作（新規列の作成やキーの追加など）の詳細は、「データストアの作成」を参照してください。

他のデータストアから列を追加する手順:

1. 「ダイアグラム」編集ウィンドウで、**「ダイアグラム」**タブを選択します。
2. データストアの列を選択します。
3. その列をダイアグラムのデータストアにドラッグします。
   データストアに追加される新規列を含むデータストア・ウィンドウが表示されます。
4. **「OK」**をクリックします。新規列がデータストアに表示されます。

データストアのグラフィカル・シノニムを作成する手順:

1. ダイアグラムでデータストアを選択します。
2. 右クリックし、**「グラフィカル・シノニムの作成」**を選択します。

新規グラフィカル・シノニムがダイアグラムに表示されます。

注意: この操作では、新規データストアは追加されません。単にダイアグラム内にデータストアの新しい表現が作成されるだけです。

データストアに列、条件、フィルタまたはキーを追加する手順:

1. ダイアグラムでデータストアを選択します。
2. 右クリックし、「キーの追加」、「フィルタの追加」などを選択します。

ダイアグラムに既存の条件、参照またはフィルタを追加する手順:

1. 既存の条件、参照またはフィルタをダイアグラムにドラッグ・アンド・ドロップします。

条件、参照またはフィルタを関連付けるデータストアは、先にダイアグラムに存在している必要があります。この方法は、ダイアグラムにデータストアを追加した後にこれらのオブジェクトを作成した場合に便利です。

列のキーを編集する手順:

列がキー（主キーまたは代替キー）の一部である場合、ダイアグラムの列でキーを編集できます。

1. ダイアグラムで列を選択します。
2. 右クリックしてポップアップ・メニューからキーの名前を選択し、サブメニューで**「編集」**を選択します。

2 つのデータストア間の参照を作成する手順:

1. 「ダイアグラム」編集ウィンドウで、**「ダイアグラム」**タブを選択します。
2. ツールバーの**「参照の追加」**  ボタンをクリックします。
3. 参照の最初のデータストアをクリックし、マウス・ボタンを押しながら 2 番目のデータストアにカーソルをドラッグします。
4. マウス・ボタンを離します。新規参照ウィンドウが表示されます。
5. この参照のパラメータを設定し、**「OK」**をクリックします。

参照を別のグラフィカル・シノニムに移動する手順:

1. ダイアグラムで参照を選択します。
2. 右クリックし、**「表示オプション」**を選択します。
   表示オプション・ウィンドウが表示されます。
3. 参照の親および子として使用するシノニムを選択します。
4. **「OK」**をクリックします。
   選択したシノニムに参照表現が表示されます。

注意: この操作では、参照は変更されません。単にダイアグラム内の表現が変化するだけです。

DDL スクリプトの生成

データ・サーバーのデータ構造が変化した場合、通常は Oracle Data Integrator で増分リバース・エンジニアリングを実行し、データ・サーバーから新規メタデータを取得します。

Oracle Data Integrator でダイアグラムまたはデータ・モデルを設計または変更した場合は、モデル実装を格納しているデータ・サーバーにそのデータ・モデルまたは変更内容を実装する必要があります。この操作は、生成される DDL スクリプトで実行します。DDL スクリプトは、DDL コマンド（create table や alter table など）を含む Oracle Data Integrator プロシージャの形式で生成されます。データ・サーバーでこのプロシージャを実行し、変更を適用できます。

注意: DDL スクリプトのテンプレートは、アクション・グループとして定義されます。DDL スクリプトの生成を開始する前に、モデルのテクノロジに対応する適切なアクション・グループが存在することをトポロジ・マネージャで確認してください。

DDL スクリプトを生成する手順:

1. DDL スクリプトを生成するデータ・モデルを選択します。
2. 右クリックし、**「DDL の生成」**を選択します。
   Oracle Data Integrator によってデータ・スキーマからデータ構造が取得され、モデル定義と比較されます。ステータス・バーに進行状況が表示されます。検出された差異を含む **DDL の生成**ウィンドウが表示されます。
3. DDL スクリプトの生成に使用する**「アクション・グループ」**を選択します。
4. **「...」**ボタンをクリックし、プロシージャを作成するフォルダを選択します。
5. **「フィルタ」**チェック・ボックスを使用して、表示する変更のタイプをフィルタします。

注意: これらのフィルタは、表示にのみ適用され、いつでも変更できます。選択済の変更は、フィルタ・オプションが原因で表示されない可能性があります。

6. **「同期化」**列のチェック・ボックスを選択して、適用する変更を指定します。
   次のアイコンは、変更のタイプを示します。
   - **-**: データ・モデルには存在するが、データ・サーバーには存在しない要素
   - **+**: データ・サーバーには存在するが、データ・モデルには存在しない要素
   - **=**: データ・モデルとデータ・サーバーの両方に存在するが、プロパティ（列のサイズ変更など）または関連する要素（新規列を含む表など）が異なる要素
7. **「OK」**をクリックして DDL スクリプトを生成します。

Oracle Data Integrator により、DDL スクリプトが生成され、DDL コマンドを含むプロシージャが表示されます。

インタフェース IN/OUT の生成

共通フォーマット・デザイナを使用して集約された特定のモデルまたはデータストアについて、Oracle Data Integrator では次のオブジェクトを生成できます。

- **インタフェース IN:** これらの統合インタフェースは、他のデータストアまたは列から集約したモデルのデータストアをロードする際に使用します。元のデータストアからコンポジット・データストアへデータをマージする統合プロセスです。
- **インタフェース OUT:** これらの統合インタフェースは、モデルのデータストアからデータを抽出する際に使用します。これらのインタフェースは、モデルのデータストアをロード済の（**インタフェース IN** を含む）インタフェースを使用して生成されます。これらのインタフェースにより、統合プロセスが反転され、コンポジット・データストアのデータが元のデータストアに伝播されます。

たとえば、AIH は、他の複数のアプリケーションから取得した断片情報を集約します。AIH は、複数のデータ・モデルに基づいて作成され、ダイアグラムに集約されるコンポジット・データストアで構成されます。AIH は、インタフェース IN を使用してロードします。また、インタフェース OUT を使用して元のシステムにデータを送信できます。

インタフェース IN を生成する手順:

1. データ・モデルまたはデータストアを選択します。
2. 右クリックし、**「インタフェース IN の生成」**を選択します。
   Oracle Data Integrator により、現在のモデルまたはデータストアの作成に使用された元のデータストアと列が検索されます。
   インタフェース IN を生成できるデータストアのリストを含む**インタフェース IN の生成**ウィンドウが表示されます。
3. インタフェースの**「最適化コンテキスト」**を選択します。
4. **「...」**ボタンをクリックし、インタフェースを生成するフォルダを選択します。
5. **「候補データストア」**で、ロードするデータストアの**「インタフェースの生成」**チェック・ボックスを選択します。
6. **「インタフェース名」**列の内容を編集し、統合インタフェースの名前を変更します。
7. **「OK」**をクリックします。
   インタフェースの生成が開始されます。

生成されたインタフェースが指定したフォルダに配置されます。

警告: 自動的に生成されるインタフェースは、利用可能なメタデータを使用して作成されるため、常に期待どおりに動作するとはかぎりません。これらのインタフェースは、実行する前に慎重に確認して修正する必要があります。

重要: インタフェース IN の生成時にデータストア候補が見つからない場合、ロードしようとしているデータストアが他のデータストアまたは列から作成されていない可能性があります。自動インタフェース生成は、他のモデルのデータストアおよび列から作成されていないデータストアと列をロードする場合には機能しません。

インタフェース OUT を生成する手順:

1. データ・モデルまたはデータストアを選択します。
2. 右クリックし、**「インタフェース OUT の生成」**を選択します。
   Oracle Data Integrator により、それらのデータストアをロードする既存のインタフェースが検索されます。
   インタフェース OUT を生成できるデータストアのリストを含む**インタフェース OUT の生成**ウィンドウが表示されます。
3. インタフェースの**「最適化コンテキスト」**を選択します。
4. **「...」**ボタンをクリックし、インタフェースを生成するフォルダを選択します。
5. 「候補データストア」で、**「生成」**および**「インタフェースの生成」**チェック・ボックスを選択し、既存のインタフェースのターゲット・データストアからロードするデータストア候補の一部または全部を選択します。
6. **「インタフェース名」**列の内容を編集し、統合インタフェースの名前を変更します。
7. **「OK」**をクリックします。
   インタフェースの生成が開始されます。

生成されたインタフェースが指定したフォルダに配置されます。

警告: 自動的に生成されるインタフェースは、利用可能なメタデータを使用して作成されるため、常に期待どおりに動作するとはかぎりません。これらのインタフェースは、実行する前に慎重に確認して修正する必要があります。

重要: インタフェース OUT の生成時にデータストア候補が見つからない場合、インタフェース OUT を生成するよう選択したデータストアをロードするインタフェースが存在しない可能性があります。データストアに基づくインタフェース OUT は、そのデータストアをロードするインタフェースから生成されます。データストアをロードする有効なインタフェースが存在しない場合、そのデータストアに基づく伝播インタフェースは生成されません。

チェンジ・データ・キャプチャ

概要

チェンジ・データ・キャプチャ（CDC）機能により、Oracle Data Integrator は、他のアプリケーションに起因するソース・データの変更を追跡できます。これにより、統合インタフェースの実行時に、変更されていないデータをフロー内で処理せずに済みます。

ソース・データ・フローを変更されたデータのみに限定することは、データ同期やレプリケーションなどの多くの状況で有益です。この作業は、統合用のイベント指向アーキテクチャを設定する場合には必須となります。このようなアーキテクチャでは、ビジネス・プロセスの実行時に、アプリケーションによってデータに変更が加えられます（顧客の削除や新規購買注文など）。これらの変更は、Oracle Data Integrator により取得され、情報システム全体に伝播されるイベントに変換されます。

チェンジ・データ・キャプチャは、モデルのジャーナル化によって実行されます。モデルのジャーナル化は、そのモデルのデータストアのレコードに追加された変更（挿入、更新および削除）を取得するインフラストラクチャを設定することで構成されます。

Oracle Data Integrator では、次の 2 つのジャーナル化モードがサポートされます。

- **簡易ジャーナル化:** モデルの個々のデータストアの変更を追跡します。
- **整合セット・ジャーナル化:** モデルの各データストア間の参照整合性を考慮しながら、それらのデータストアのグループに対する変更を追跡します。このモードでジャーナル化されるデータストアのグループは、**整合セット**と呼ばれます。

ジャーナル化コンポーネント

ジャーナル化コンポーネントは次のとおりです。

- **ジャーナル:** 変更が記録される場所。ジャーナルには、変更されたレコードへの参照と、その変更のタイプ（挿入、更新、削除）のみが含まれます。
- **取得プロセス:** ジャーナル化では、データ表にトリガーを作成することで、またはデータ・サーバーのログ・ファイルからログ・データを取得するデータベース固有のプログラムを使用することで、ソース・データストアの変更を取得します。使用される取得プロセスの詳細は、ジャーナル化ナレッジ・モジュールに関するドキュメントを参照してください。
- **サブスクライバ:** チェンジ・データ・キャプチャ（CDC）では、パブリッシュ・サブスクライブ・モデルを使用します。サブスクライバは、データストアまたは整合セットで追跡される変更を使用するエンティティ（アプリケーション、統合プロセスなど）です。これらのエンティティは、モデルの CDC にサブスクライブして、変更の追跡を委任します。変更が取得されるのは、変更に対して少なくとも 1 つのサブスクライバが存在する場合のみです。取得された変更をすべてのサブスクライバが使用すると、それらの変更はジャーナルから破棄されます。

- **ジャーナル化ビュー:** 取得された変更と変更データへのアクセスを提供します。これらのビューは、ユーザーが取得された変更を表示する場合と、統合プロセスが変更データを検索する場合に使用されます。

これらのコンポーネントは、ジャーナル化インフラストラクチャに実装されます。

簡易ジャーナル化と整合セット・ジャーナル化の比較

簡易ジャーナル化では、1 つ以上のデータストアをジャーナル化できます。ジャーナル化された各データストアは、変更の取得時に個別に処理されます。

この方法には、次の例のような限界があります。たとえば、ORDER データストアと ORDER_LINE データストアの変更を処理する必要があるとします（このとき、参照整合性制約として、ORDER_LINE レコードは必ず関連する ORDER レコードを保持する必要があります）。ORDER_LINE への挿入が取得された場合、ORDER 内で関連する新規レコードも取得されているかどうかは保証されません。関連する ORDER レコードなしで ORDER_LINE レコードを処理すると、統合プロセスで参照整合性違反が発生する可能性があります。

整合セット・ジャーナル化では、ORDER_LINE（または ORDER）の変更が取得されたときに、関連する ORDER（または ORDER_LINE）の変更も取得されていることが保証されます。整合セット・ジャーナル化では、取得された変更の整合性が保証されます。整合性が保証された使用可能な変更のセットは、**整合ウィンドウ**と呼ばれます。このウィンドウ内の各変更は、パッケージ内で統合インタフェースを設計および順序付けすることで、正しい順序で（ORDER、ORDER_LINE の順に）処理する必要があります。

整合セット・ジャーナル化は強力な方法ですが、設定も複雑になります。この方法は、データ変更の取得時に参照整合性制約を保証する必要がある場合に使用してください。パフォーマンス上の理由から、多数のサブスクライバを必要とする場合にも、整合セット・ジャーナル化をお薦めします。

注意: 整合セット・ジャーナル化と簡易ジャーナル化の両方を使用してモデル（またはモデル内のデータストア）をジャーナル化することはできません。

ジャーナル化の設定

次に、Oracle Data Integrator データ・モデルでのチェンジ・データ・キャプチャ（CDC）の基本的な設定プロセスを示します。各手順の詳細は、後述の説明を参照してください。

1. CDC パラメータを設定します。
2. CDC にデータストアを追加します。
3. 整合セット・ジャーナル化では、データストアの順序を設定します。
4. サブスクライバを追加します。
5. ジャーナルを開始します。

データ・モデルの CDC パラメータを設定する手順:

この手順では、モデルに使用するジャーナル化モードおよびジャーナル化ナレッジ・モジュールを選択または変更します。モデルがすでにジャーナル化されている場合、データ・モデルのジャーナル化パラメータを変更する前に、既存の構成によるジャーナル化を停止することをお薦めします。

1. ジャーナル化するデータ・モデルを編集し、**「ジャーナル化」**タブを選択します。
2. 設定するジャーナル化モード（**「一貫性セット」**または**「簡易」**）を選択します。

3. このモデルに使用する**「ジャーナル化 KM」**を選択します。リストに表示されるのは、データ・モデルのテクノロジとジャーナル化モードに適しており、事前に少なくとも 1 つのプロジェクトにインポートされているナレッジ・モジュールのみです。
4. この KM の**「オプション」**を設定します。オプションの詳細は、ナレッジ・モジュールの説明を参照してください。
5. **「OK」**をクリックして変更を保存します。

CDC にデータストアを追加する、または CDC からデータストアを削除する手順:

次に、ジャーナル化するデータストアにフラグを付けます。データストア・フラグの変更は、次回ジャーナルが（再）起動されたときに処理されます。モデルまたはサブモデルにフラグを付けると、モデルまたはサブモデルに含まれるすべてのデータストアにフラグが付けられます。

1. CDC を対象に追加または削除を行うデータストア、モデルまたはサブモデルを選択します。
2. 右クリックし、**「チェンジ・データ・キャプチャ」→「CDC に追加」**を選択してデータストア、モデルまたはサブモデルを CDC に追加します。または、**「チェンジ・データ・キャプチャ」→「CDC から削除」**を選択してそれらを CDC から削除します。
3. ツリー・ビューがリフレッシュされます。CDC に追加されたデータストアにマーカー・アイコンが表示されます。ジャーナル・アイコンは、小さい時計の形をしています。アイコンの色は、まだジャーナル・インフラストラクチャが設定されていないことを示す黄色です。

ジャーナル作成フェーズの後でもデータストアを CDC に追加できます。この場合、ジャーナルを再起動する必要があります。

注意: ジャーナルを実行しているデータストアを簡易モードの CDC から削除する場合、その個別のデータストアに対するジャーナルを停止する必要があります。整合セット・モードの CDC からデータストアを削除する場合、モデルに対するジャーナルを再起動する必要があります（他のデータストアのジャーナル化情報は維持されます）。

データストアの順序を設定する手順（整合セット・ジャーナル化のみ）:

データストアの順序を設定する必要があるのは、整合セット・ジャーナル化を使用する場合のみです。整合セットのデータストアは、変更データの使用時に参照整合性を維持する順序に調整する必要があります。たとえば、ORDER 表に ORDER_LINE データストアからインポートされた参照があり（つまり、ORDER_LINE には ORDER を参照する外部キー制約が含まれます）、ORDER と ORDER_LINE の両方が CDC に追加されている場合、ORDER データストアは ORDER_LINE より前の順序に設定する必要があります。PRODUCT データストアに ORDER と ORDER_LINE の両方からインポートされた参照がある場合（つまり、ORDER と ORDER_LINE の両方に、PRODUCT 表に対する外部キー制約が含まれます）、その順序はさらに繰り下がります。

1. ジャーナル化するデータ・モデルを編集し、**「ジャーナル化表」**タブを選択します。
2. データストアの順序が現在特に設定されていない場合、**「再編成」**ボタンをクリックします。この機能により、データ・モデルの外部キーに基づいて、ジャーナル化するデータストアの順序が推奨されます。この自動再構成は完全ではないため、推奨された順序を後でよく確認してください。
3. リストからデータストアを選択し、**「上」**および**「下」**ボタンを使用してリスト内を移動します。データストアの**「順序」**の値を直接編集することもできます。
4. データストアの順序が正しく設定されるまで手順 3 を繰り返し、**「OK」**をクリックして変更を保存します。

データストアの順序の変更は、次回ジャーナルが（再）起動されたときに処理されます。

注意: 既存のシナリオでこの CDC セットからの変更を使用している場合、CDC セットの新規順序を反映するようシナリオを再生成する必要があります。

注意: このタブの**「CDC から削除」**ボタンを使用すると、CDC からデータストアを削除できます。

サブスクライバを追加または削除する手順:

この手順により、取得された変更を使用するエンティティのリストを追加または削除します。

1. 整合セット・ジャーナル化を使用する場合、ジャーナル化対象のデータ・モデルを選択します。簡易ジャーナル化を使用する場合、データ・モデルまたは個々のデータストアを選択します。
2. 右クリックし、**「チェンジ・データ・キャプチャ」→「サブスクライバ」→「サブスクライブ」**を選択します。サブスクライバを選択するウィンドウが表示されます。
3. フィールドにサブスクライバ名を入力し、**「サブスクライバの追加」**ボタンをクリックします。
4. 追加するサブスクライバごとにこの操作を繰り返します。最後に**「OK」**をクリックします。

CDC にサブスクライバを追加するセッションが起動します。このセッションは、オペレータで追跡できます。

サブスクライバを削除する手順もほぼ同じです。かわりに**「チェンジ・データ・キャプチャ」→「サブスクライバ」→「サブスクライブ解除」**オプションを選択します。

ジャーナルの開始後もサブスクライバを追加できます。ジャーナルの起動後に追加されたサブスクライバは、それらがサブスクライバ・リストに追加された後に発生した変更のみを取得できます。

ジャーナルを開始または停止する手順:

CDC インフラストラクチャが存在しない場合、ジャーナルを開始すると CDC インフラストラクチャが作成されます。また、ジャーナル化されるデータストアの追加、削除および順序変更が検証されます。

注意: ジャーナルを停止すると、ジャーナル化インフラストラクチャ全体が削除され、取得されたすべての変更が失われます。ジャーナルを再起動しても、取得済の変更データは削除または変更されません。

1. ジャーナル化するデータ・モデルまたはデータストアを選択します。
2. 右クリックし、**「チェンジ・データ・キャプチャ」→「ジャーナルの開始」**を選択するか（ジャーナルを開始する場合）、**「チェンジ・データ・キャプチャ」→「ジャーナルを削除」**を選択します（ジャーナルを停止する場合）。

ジャーナルを起動または削除するセッションが開始されます。このセッションは、オペレータで追跡できます。

ジャーナル化インフラストラクチャは、物理レベルでジャーナル化 KM により実装されます。したがって、データ・モデルでジャーナル化が必要とされるコンテキストごとに、「サブスクライバの追加」および「ジャーナルの開始」操作を実行する必要があります。これらの操作は、Oracle Data Integrator パッケージを使用して自動化できます。異なるコンテキスト全体にジャーナル化インフラストラクチャをデプロイする場合、操作を自動化することをお薦めします。

一般的な状況では、開発者は開発環境のコンテキストで CDC を手動構成します。これで正常に稼働したら、パッケージを使用してテスト環境のコンテキストに CDC を自動的にデプロイします。最後に、同じパッケージを使用して本番環境のコンテキストに CDC をデプロイします。

ジャーナル化設定を自動化する手順:

1. デザイナで新規パッケージを作成します。
2. ジャーナル化するモデルまたはデータストアをドラッグ・アンド・ドロップします。新規ステップが表示されます。
3. ダイアグラムのステップ・アイコンをダブルクリックします。プロパティ・パネルが表示されます。
4. **「タイプ」**リストで、**「ジャーナル化モデル/データストア」**を選択します。
5. **「開始」**ボックスを選択してジャーナルを開始します。
6. **「サブスクライバの追加」**ボックスを選択し、**「サブスクライバ」**グループにサブスクライバのリストを入力します。
7. **「OK」**をクリックして保存します。
8. このパッケージのシナリオを生成します。

このシナリオがコンテキスト内で実行されると、モデル構成に従ってジャーナルが開始され、指定したサブスクライバがこのコンテキストを使用して作成されます。

サブスクライバとジャーナル管理は、異なるステップおよびパッケージに分割できます。サブスクライバの削除とジャーナルの停止も、同じ方法で自動化できます。詳細は、パッケージに関する項を参照してください。

ジャーナル化インフラストラクチャの詳細

ジャーナルが開始されると、ジャーナル化インフラストラクチャが次の場所でデプロイまたは更新されます（このインフラストラクチャがまだインストールされていない場合）。

- ジャーナル化ナレッジ・モジュールでトリガーを作成する場合、それらのトリガーは、ジャーナル化される表を含む Oracle Data Integrator の物理スキーマに対応する**データ・スキーマ**の表にインストールされます。ジャーナル化トリガーの名前の先頭には、物理スキーマの**「ジャーナル化要素接頭辞」**で定義された接頭辞が付きます。この接頭辞のデフォルト値は、`T$`です。データベース固有のプログラムは個別にインストールされます（詳細は、ナレッジ・モジュールに関するドキュメントを参照してください）。
- データ・サーバーの CDC 共通インフラストラクチャは、そのデータ・サーバーの**デフォルト**として指定された Oracle Data Integrator の物理スキーマに対応する**作業スキーマ**にインストールされます。この共通インフラストラクチャには、データ・サーバーでジャーナル化されるすべてのスキーマのサブスクライバや整合セットなどに関する情報が含まれます。この共通インフラストラクチャは、`SNP_CDC_`という接頭辞を名前に持つ表で構成されます。
- ジャーナル表とジャーナル化ビューは、ジャーナル化される表を含む Oracle Data Integrator の物理スキーマに対応する**作業スキーマ**にインストールされます。ジャーナル表とジャーナル化ビューの名前の先頭には、物理スキーマの**「ジャーナル化要素接頭辞」**で定義された接頭辞が付きます。デフォルト値は、ジャーナル表では `J$`で、ジャーナル化ビューでは `JV$`です。

注意: ジャーナル化インフラストラクチャの（トリガーを除く）すべてのコンポーネント（統合表、エラー表、ロード表などのすべての Data Integrator 一時オブジェクト）は、データ・サーバーの Oracle Data Integrator の物理スキーマに対応する**作業スキーマ**にインストールされます。これらの作業スキーマは、アプリケーション・データを格納するスキーマ（データ・スキーマ）とは別に維持する必要があります。

重要: ジャーナル化トリガーは、ジャーナル化されるデータと同じスキーマに必要に応じてインストールされる唯一のジャーナル化用コンポーネントです。サード・パーティのソフトウェア・パッケージに含まれる表にトリガーを作成する前に、その操作がソフトウェア規約やメンテナンス契約に違反しないことを確認してください。また、ソフトウェア・パッケージの一般的な動作を妨げることなく、トリガーをインストールして実行することが技術的に可能であることも確認してください。

ジャーナル化ステータス

モデルやインタフェースのデータストアには、デザイナの現在のコンテキストにおけるジャーナル化ステータスを示すアイコン・マーカーが割り当てられます。

- **OK:** ジャーナル化は、現在のコンテキストにおいてこのデータストアに対してアクティブです。また、インフラストラクチャはこのデータストアで稼働中です。
- **インフラストラクチャなし:** ジャーナル化はモデルでアクティブとしてマークされていますが、現在のコンテキストにおいて適切なジャーナル化インフラストラクチャが検出されませんでした。ジャーナルを開始する必要があります。インフラストラクチャに実装されているジャーナル化モードが、モデルに宣言されているモードと一致しない場合、この状態が発生する可能性があります。
- **断片:** ジャーナル化はモデルで非アクティブとしてマークされていますが、現在のコンテキストにおいてジャーナル化表などのジャーナル化インフラストラクチャの断片がこのデータストアで検出されました。ジャーナルを停止せずに CDC から表を削除した場合、この状態が発生する可能性があります。

変更データの使用方法

ジャーナル化を開始してサブスクライバで変更を追跡するよう設定したら、取得された変更を表示できます。

変更データを表示する手順:

1. ジャーナル化対象のデータストアを選択します。
2. 右クリックし、**「チェンジ・データ・キャプチャ」→「ジャーナル・データ」**を選択します。

変更データを含むウィンドウが表示されます。

注意: このウィンドウでは、ジャーナル化ビューを使用してデータを選択します。

変更データには、変更の詳細を示す次の 3 つの追加列が表示されます。

- `JRN_FLAG`: 変更のタイプを示すフラグ。挿入または更新されたレコードには値 `I` が使用され、削除されたレコードには値 `D` が使用されます。
- `JRN_SUBSCRIBER`: サブスクライバの名前。
- `JRN_DATE`: 変更のタイムスタンプ。

ジャーナル化データは、主に統合プロセス内で使用されます。変更データは、統合インタフェースのソースとして使用できます。その使用方法は、ジャーナル化モードに応じて異なります。

変更データの使用方法: 簡易ジャーナル化

簡易ジャーナル化による変更データを使用する場合、ジャーナル化されるデータストアをソースとして使用するインタフェースを設計します。

インタフェースの設計

ジャーナル化フィルタ

ジャーナル化されるデータストアをインタフェース・ダイアグラムに挿入すると、そのデータストアのプロパティ・パネルに**「ジャーナル化されたデータのみ」**チェック・ボックスが表示されます。

このボックスを選択すると、次の処理が行われます。

- **ジャーナル化列**（JRN_FLAG、JRN_DATE および JRN_SUBSCRIBER）がデータストアで使用可能になります。
- **ジャーナル化フィルタ**もこのデータストアで自動的に生成されます。このフィルタにより、取得されるソース・データの量が削減されます。このフィルタは、常にソースで実行されます。このフィルタはカスタマイズできます（一定の時間範囲内の変更を処理する、または特定タイプの変更のみを処理するなど）。
  特定のサブスクライバ用にすべての変更を取得するための一般的なフィルタは、
  `JRN_SUBSCRIBER = '<subscriber_name>'`です。

注意: 簡易ジャーナル化モードでは、（ジャーナル・フィルタの適用後に）インタフェースにより処理されたすべての変更は、自動的にインタフェースの最後に使用されたとみなされ、ジャーナルから削除されます。これらの変更は、後続のインタフェースで使用できません。

ナレッジ・モジュールのオプション

ジャーナル化データを処理する場合、統合ナレッジ・モジュールの SYNC_JRN_DELETE オプションは慎重に設定してください。このオプションでは、ジャーナルで削除対象（D）としてマークされ、ジャーナル化フィルタで除外されないレコードがターゲット・データストアから削除されます。このオプションを「いいえ」に設定すると、統合操作では挿入と更新のみが処理されます。

変更データの使用方法: 整合セット・ジャーナル化

整合ジャーナル化での変更データの使用方法は、インタフェース設計に関しては簡易ジャーナル化と同様です。ただし、セット内で変更の整合性を確保するために、インタフェースで変更データを処理する前と後に追加のステップが必要です。

変更データ使用前の操作

整合セット・ジャーナル化を使用する場合、変更データの使用前に次の操作を実行する必要があります。

- **ウィンドウの拡張: 整合ウィンドウ**は、参照整合性に違反することなく挿入、更新および削除を実行できる、整合セットのすべての表における使用可能な変更の範囲です。ウィンドウの拡張操作では、前回のウィンドウの拡張操作以降に発生した新規の変更を反映するようこのウィンドウを（再）計算します。この操作は、**ジャーナル化モデル・タイプ**のパッケージ・ステップを使用して実装します。この操作は、他のジャーナル化操作とは別にスケジュールできます。
- **サブスクライバのロック:** ウィンドウの拡張は整合セット全体に適用されますが、サブスクライバは変更を個別に使用します。この操作では、整合ウィンドウの変更を対象に、1 つ以上のサブスクライバに固有のスナップショットを実行します。このスナップショットには、サブ

スクライバにまだ使用されていない整合ウィンドウ内のすべての変更が含まれます。この操作は、**ジャーナル化モデル・タイプ**のパッケージ・ステップを使用して実装します。サブスクライバに取得された変更を使用する最初のインタフェースより前に、この操作を必ず実行する必要があります。

インタフェースの設計

整合セット・ジャーナル化の変更データも、パッケージ内に順序付けされたインタフェースを使用して処理します。

整合セット・ジャーナル化の使用時にインタフェースを設計する手順は、簡易ジャーナル化と同様です。ただし、次の点が異なります。

- （JRN_FLAG、JRN_DATE および JRN_SUBSCRIBER でフィルタされた）インタフェースで処理される変更は、インタフェースの最後に自動的にパージされません。これらの変更は、後続のインタフェースで再利用できます。これらの変更の使用をコミットし、不要なエントリをジャーナルから削除するには、それぞれサブスクライバのロック解除操作とジャーナルのパージ操作（後述の説明を参照）が必要です。
- 整合モードでは、ジャーナル化フィルタで JRN_DATE 列を使用しないでください。使用された変更をこのタイムスタンプでフィルタすると、変更の整合性が確保されないことがあります。

変更データ使用後の操作

変更データの使用後に、次の操作を実行する必要があります。

- **サブスクライバのロック解除:** この操作では、**サブスクライバのロック**操作でロックされた変更の使用をコミットします。この操作は、サブスクライバですべての変更が処理された後にのみ実行する必要があります。この操作は、**ジャーナル化モデル・タイプ**のパッケージ・ステップを使用して実装します。サブスクライバに取得された変更を使用する最後のインタフェースより後に、この操作を必ず実行する必要があります。変更を再度処理する必要がある場合は（エラーなどの場合）、この操作は実行しないでください。
- **ジャーナルのパージ:** サブスクライブしている変更をすべてのサブスクライバが使用した後でも、各エントリはジャーナル化表に残るため、それらを削除する必要があります。この作業は、ジャーナルのパージ操作で実行します。この操作は、**ジャーナル化モデル・タイプ**のパッケージ・ステップを使用して実装します。この操作は、他のジャーナル化操作とは別にスケジュールできます。

ウィンドウの拡張、サブスクライバのロック/ロック解除、またはジャーナルのパージの各ステップをパッケージで作成する手順:

1. 操作を実行するパッケージを開きます。
2. 操作を実行するモデルをドラッグ・アンド・ドロップします。
3. **「タイプ」**リストで、**「ジャーナル化モデル」**を選択します。
4. 実行する操作に対応するオプション・ボックスを選択します。
5. サブスクライバのロックまたはロック解除操作を実行する場合は、サブスクライバのリストを**「サブスクライバ」**グループに入力します。
6. **「OK」**をクリックします。

注意: ウィンドウの拡張またはジャーナルのパージは、データストアで実行できます。この操作は、同じ整合セットの表における変更を異なる頻度で処理するために提供されています。変

更の整合性が整合セット・レベルで維持されなくなる可能性があるため、このオプションは慎重に使用してください。

ジャーナル化ツール

Oracle Data Integrator には、取得された変更に関する情報のリフレッシュや、他のプロセスの起動を行うためにジャーナル化で使用できる次のようなツール・セットが付属しています。

- **SnpsWaitForData:** 表または表のセットの行が多くなるのを待機します。
- **SnpsWaitForLogData:** ジャーナル化表またはジャーナル化表のリストで一定数の変更が発生するのを待機します。このツールは、SnpsRefreshJournalCount をコールして、取得された新規の変更をカウントします。
- **SnpsWaitForTable:** 表が作成され、事前に指定された数の行が移入されるのを待機します。
- **SnpsRetrieveJournalData:** 指定されたジャーナル化サブスクライバの特定の表リストまたは CDC セットで、ジャーナル化イベントを取得します。データベース固有のプロセスを使用してジャーナル化表をロードする場合、このツールをコールする必要があります。このツールは、特定のナレッジ・モジュールとともに使用する必要があります。詳細は、ナレッジ・モジュールの説明を参照してください。
- **SnpsRefreshJournalCount:** 指定されたジャーナル化サブスクライバの特定の表リストまたは CDC セットで使用された行数をリフレッシュします。

これらの機能の詳細は、『Oracle Data Integrator Tools リファレンス・マニュアル』を参照してください。

ジャーナル化を使用するためのパッケージ・テンプレート

ジャーナル化データを使用するパッケージを設計する場合、多くのテンプレートを使用できます。次に、いくつかの一般的なテンプレートを示します。

テンプレート 1: 1 つの単純なパッケージ（整合セット）

- ステップ 1: ウィンドウの拡張 + サブスクライバのロック
- ステップ 2 ... n-1: ジャーナル化データを使用するインタフェース
- ステップ n: サブスクライバのロック解除 + ジャーナルのパージ

このパッケージは、すべての変更を処理するように毎分スケジュールされます。このテンプレートは、ジャーナル化表に定期的に変更が発生する場合に適しています。

テンプレート 2: 1 つの単純なパッケージ（簡易ジャーナル化）

- ステップ 1 ... n: ジャーナル化データを使用するインタフェース

このパッケージは、すべての変更を処理するように毎分スケジュールされます。このテンプレートは、ジャーナル化表に定期的に変更が発生する場合に適しています。

テンプレート 3: SnpsWaitForLogData の使用（整合セットまたは簡易ジャーナル化）

- ステップ 1: SnpsWaitForLogData。指定された間隔の経過後に新規ログ・データが検出されない場合、パッケージは終了します。
- ステップ 2: SnpsStartScen を使用して、テンプレート 1 または 2 と同等のシナリオを実行します。

このパッケージは、定期的にスケジュールされます。変更データが処理されるのは、新規の変更が検出された場合のみです。これにより、ジャーナル化表に不定期に変更が発生する場合に、不

要な処理を避けることができます（つまり、処理するデータのないインタフェースを実行せずに済みます）。

テンプレート 4: 個別のプロセス（整合セット）

このテンプレートでは、整合ウィンドウ、パージ、および（2 つの異なるサブスクライバでの）変更の使用を異なるパッケージに分割します。

パッケージ 1: ウィンドウの拡張

- ステップ 1: SnpsWaitForLogData。指定された間隔の経過後に新規ログ・データが検出されない場合、パッケージは終了します。
- ステップ 2: ウィンドウの拡張。

このパッケージは、毎分スケジュールされます。ウィンドウの拡張では、リソースを消費する可能性があります。新規データが発生した場合にのみこの操作を起動することをお薦めします。

パッケージ 2: ジャーナルのパージ（週末）

- ステップ 1: ジャーナルのパージ

このパッケージは、毎週金曜日に 1 回スケジュールされます。週全体のジャーナルが追跡されます。

パッケージ 3: サブスクライバ A での変更の処理

- ステップ 1: サブスクライバ A のロック
- ステップ 2 ... n-1: サブスクライバ A でジャーナル化データを使用するインタフェース
- ステップ n: サブスクライバ A のロック解除

このパッケージは、毎分スケジュールされます。このようなパッケージは、MOM でイベントを生成する場合などに使用します。

パッケージ 4: サブスクライバ B での変更の処理

- ステップ 1: サブスクライバ B のロック
- ステップ 2 ... n-1: サブスクライバ B でジャーナル化データを使用するインタフェース
- ステップ n: サブスクライバ B のロック解除

このパッケージは、毎日スケジュールされます。このようなパッケージは、夜間に変更データをデータ・ウェアハウスにロードする場合などに使用します。

データ・サービスの設定

データ・サービスは、データストアのデータや、それらのデータストアで取得された変更へのアクセスを提供する特殊な Web サービスです。これらの Web サービスは、Oracle Data Integrator によって自動的に生成され、Web サービス・コンテナ（通常は Java アプリケーション・サーバー）にデプロイされます。

次の例では、アプリケーション・サーバーとして Apache Tomcat 5.5 または Oracle Containers for Java（OC4J）を使用し、Web サービス・コンテナとして Apache Axis2 を使用します。これらの例は、他の Web サービス・コンテナを使用する場合には適切に調整する必要があります。

データ・サービスの設定には、次のステージが含まれます。

1. Web サービス・コンテナでデータソースを設定します。
2. トポロジを構成して Web サービス・コンテナを宣言します。
3. モデルを設定してデータ・サービスのデータストアおよび列を宣言します。

4. データ・サービスを生成、デプロイおよびテストします。

データソースの設定

手順を開始する前に、Web サービス・コンテナとアプリケーション・サーバーを先に構成しておく必要があります。詳細は、各コンポーネントの個々のドキュメントを参照してください。

Oracle Data Integrator によって生成されるデータ・サービスには、ソースとターゲットに関する接続情報は含まれません。かわりに、Web サービス・コンテナ内またはアプリケーション・サーバー上で定義されたデータソースが使用されます。これらのデータソースには、データへのアクセスに必要な接続プロパティが含まれます。各データソースは、Oracle Data Integrator のトポロジ内で定義されたデータ・サーバーに対応している必要があります。

また、指定したデータソースに必要なすべてのドライバをアプリケーション・サーバーにインストールする必要があります。

データソースを設定する手順:

1. 使用するデータソース用のドライバ・ファイル（.jar または.zip）を、アプリケーション・サーバーの適切なディレクトリにインストールします。Tomcat の場合、`/common/lib` を使用します。OC4J の場合、`ORACLE_HOME/j2ee/home/applib` を使用します。
2. アクセスするデータ・サーバーを参照する JDBC データソースを作成します。
   - Tomcat では、Web サービス・コンテナの `WEB-INF/context.xml` ファイルにデータのデータソースと接続プロパティを定義します。後で必要になるため、リソース名（次の例では緑で記載）を書き留めておいてください。トポロジに含まれる接続プロパティは、**青**で記載されています。
     次に例を示します。

```
<Context >
    <Resource
      name="jdbc/Oracle_SRV1/Oracle/dataServices"
      type="javax.sql.DataSource"
      driverClassName="oracle.jdbc.OracleDriver"
      url="jdbc:oracle:thin:@SRV1:1521:ORA10"
      username="ODI"
      password="ODI"
      maxIdle="2"
      maxWait="-1"
      maxActive="4"/>
</Context>
```

- OC4J では、次のようにデータソースを定義する必要があります。
  1. OC4J の管理インタフェースに接続します。
  2. **「管理」**タブで**「サービス」→「JDBC リソース」**を選択し、**「タスクに移動」**をクリックします。
  3. **「接続プール」**セクションの**「作成」**ボタンをクリックします。
  4. Axis2 アプリケーションを選択し、**「新規接続プール」→「続行」**を選択します。
  5. JDBC データソースの各フィールドを入力し、**「終了」**をクリックします。

6. 「データソース」セクションの「作成」ボタンをクリックします。
7. Axis2 アプリケーションを選択し、「マネージド・データソース」→「続行」を選択します。
8. JDBC データソースの各フィールドを入力し、「終了」をクリックします。OC4J でデータソースを構成する方法の詳細は、アプリケーション・サーバーのドキュメントを参照してください。

3. Web サービス・コンテナの `WEB-INF/web.xml` ファイルでデータソースへの参照を作成します。データソースの名前を `res-ref-name` タグに指定する必要があります。
次に、前の手順で定義したデータソースを参照する例を示します。

```
...
<resource-ref>
     <description>Data Integrator Data Services on
     Oracle_SRV1</description>
     <res-ref-name>jdbc/Oracle_SRV1/Oracle/dataServices</res-
     ref-name>
     <res-type>javax.sql.DataSource</res-type>
     <res-auth>Container</res-auth>
</resource-ref>
```

4. 新規データソースの設定を反映するため、Axis2 アプリケーションを再起動します。

トポロジの構成

Web サービス・コンテナの準備を完了したら、Oracle Data Integrator でそのコンテナにデータ・サービスをデプロイするために、そのコンテナをトポロジ内でデータ・サーバーとして宣言します。

注意: Web サービス・コンテナと、デプロイ・データを格納するサーバーを混同しないように注意してください。どちらも Oracle Data Integrator ではデータ・サーバーとして宣言されますが、Web サービス・コンテナはデータを格納せず、公開されたデータ・サービスをホストするだけです。

Oracle Data Integrator で宣言される Web サービス・コンテナでは、Web サービスのデプロイ・モードとして次のいずれかの方法を使用します。

- サーバーに直接ファイルをコピーします（サーバーに対するファイル・アクセス権を持つ場合）。
- FTP を通じてサーバーにアップロードします。
- Axis2 などの一部のサーバーでサポートされる Web サービス・アップロードの方法を使用してアップロードします。

Web サービス・コンテナを構成するための次の手順は、使用するデプロイ・モードに応じて異なります。

Web サービス・コンテナを構成する手順:

1. **トポロジ・マネージャ**の「物理アーキテクチャ」ビューで、「Axis2」テクノロジを選択します。右クリックし、「データサーバーの挿入」を選択します。別の Web サービス・コンテナを使用する場合は、かわりに適切なテクノロジを選択します。

2. 「定義」タブで、次のフィールドを入力します。

   - **名前:** Oracle Data Integrator に表示されるデータ・サーバーの名前。
   - **公開されたサービスのベース URL:** http://<Host>:<HTTP port>/axis2/services。
   - 選択したデプロイ方法に対応するオプションを指定します。
     - **ファイルのコピー:** Web サービスを配置するアプリケーション・サーバーのディレクトリを指定します。
     - **Web サービスのアップロード:** Axis2 アプリケーションのルート URL（通常は `http://<Host>:<HTTP port>/axis2/axis2-admin/`）と、Axis2 管理者のユーザー名およびパスワードを指定します。
     - **FTP アップロード:** デプロイ・ディレクトリの FTP URL と、そのディレクトリに対する書込み権限を保持するユーザーのユーザー名およびパスワードを指定します。
3. **「OK」**をクリックします。
   物理スキーマを作成するためのウィンドウが表示されます。
4. **「コンテキスト」**タブに移動し、データ・サービスをデプロイするコンテキストごとに論理スキーマを 1 つ定義します。
5. **「OK」**をクリックします。

注意: 物理スキーマは、Web サービス・コンテナごとに 1 つ定義すれば十分です。

トポロジの設定方法の詳細は、「トポロジの作成」を参照してください。

モデルの設定

モデルでデータ・サービスを構成するには、デプロイするデータストアが（リバース・エンジニアリングなどにより）正しく定義されており、アクセス可能である必要があります。詳細は、「モデルの作成およびリバース・エンジニアリング」を参照してください。

いずれかのプロジェクトに適切なサービス・ナレッジ・モジュール（SKM）をインポートしてあることを確認します。SKM には、データ・サービスの生成元となるコード・テンプレートが含まれます。KM のインポート方法の詳細は、「KM のインポート」を参照してください。

また、生成されるデータ・サービスに後述のプロパティが与える影響の詳細は、「生成サービスの概要」を参照してください。

モデルを設定してデータ・サービスを使用する手順:

1. モデルを開き、**「サービス」**タブに移動します。
2. **「アプリケーション・サーバー」**タブで、前に設定した Web サービス・コンテナを選択します。
3. 生成される WSDL で使用する**「ネームスペース」**を設定します。
4. Web サービスを格納するために生成される Java パッケージの名前に使用する**「パッケージ名」**を指定します。一般的には、`com.<company name>.<project name>`という形式です。
5. **「データソース名」**フィールドに、データソースの設定時にサーバーに対して定義したデータソース名をコピーして貼り付けます。この名前には、**`java:/comp/env/`**という接頭辞を付ける必要があります。

6. **「データ・サービス名」**を定義します。この名前は、モデル・レベルで稼働するデータ・サービスでのみ使用されます。データ・サービス名は、データストアごとに定義することもできます（後述の説明を参照）。
7. リストからサービス・ナレッジ・モジュール（SKM）を選択し、そのオプションを設定します。詳細は、SKM の説明を参照してください。ここに表示されるのは、プロジェクトにインポートされている SKM のみです。
8. **「配置済データストア」**タブに移動します。
9. Web サービスとして公開するすべてのデータストアを選択します。データストアごとに、**「データ・サービス名」**と**「公開済エンティティ」**の名前を指定します。
10. 「OK」をクリックして変更を保存します。

最初は必要ありませんが、生成されたデータ・サービスの構成をデータストア・レベルおよび列レベルで調整することも可能です。

データストア・レベルでデータ・サービスのオプションを構成する手順:

1. データストアを開き、**「サービス」**タブを選択します。
2. データストアをデプロイする場合、「データ・サービスとして配置」を選択します。
3. データストアの**「データ・サービス名」**と**「公開済エンティティ」**の名前を入力します。
4. **「OK」**をクリックして変更を保存します。

列レベルでデータ・サービスのオプションを構成する手順:

列ごとに許可する操作を指定できます。このオプションの重要な使用方法として、データ・サービスを通じた書込みから列をロックすることがあげられます。

1. 列を開き、**「サービス」**タブを選択します。
2. **「挿入」**、**「更新」**、**「選択」**のうち許可する操作を選択します。
3. **「OK」**をクリックして変更を保存します。

データ・サービスの生成およびデプロイ

モデルにデータ・サービスを生成すると、個々のデータストアにデータ・サービスを生成するのと同じ効果があります。

データストアまたはモデルにデータ・サービスを生成する手順:

1. データ・サービスを生成するモデルまたはデータストアを選択します。
2. 右クリックし、**「サービスの生成」**を選択します。
   生成の構成ウィンドウが表示されます。
3. 生成されるデータ・サービスを格納するパスを指定します。Oracle Data Integrator により、生成されたソース・コードとコンパイルされた Web サービスがこの場所に配置されます。このディレクトリは、生成後に削除できる一時的な場所です。
4. **「コンテキスト」**を指定します。コンテキストを選択することで、次の 3 つの効果があります。
   - 生成時の JDBC/Java データ型バインディングが決定します。
   - データの提供に使用される物理スキーマが決定します。
   - デプロイ先の物理 Web サービス・コンテナが決定します。

5. 1 つ以上の**「生成フェーズ」**を選択します。通常のデプロイでは、3 つのフェーズすべてを選択する必要があります。ただし、新規の SKM をテストする場合などには、生成フェーズのみを実行すると便利です。各フェーズの意味は、次の表を参照してください。

データストアまたは**モデル**・ウィンドウでは、**「サービス」**タブの**「生成および配置」**ボタンを使用することも可能です。

注意: モデルにデータ・サービスを生成すると、そのモデルに対応する 1 つのデータ・サービスと、公開用として有効化されるデータストアごとに追加のデータ・サービスが生成されます。

フェーズ	説明
生成	• ディレクトリ内のすべてのオブジェクトの削除 • SKM を使用した.java ファイルの生成
コンパイル	• Web サービス・フレームワークの抽出 • .java ファイルの.class ファイルへのコンパイル
デプロイ	• コンパイル済ファイルからの Java パッケージ（.aar）の生成 • 指定のデプロイ方法を使用したデプロイ・ターゲットへのパッケージのコピー

生成サービスの概要

モデル・レベルのサービス

データ・サービスは、モデルが整合セット CDC に対応している場合、モデル・レベルでのみ生成されます。次のサービスが生成されます。

- `extendWindow`（パラメータなし）: ウィンドウの拡張操作を実行します。
- `lock`（SubscriberName）: 名前付きサブスクライバの整合セットをロックします。複数のサブスクライバの整合セットをロックするには、複数の SnpsInvokeWebService ステップを使用するなどして、サービスを複数回コールします。
- `unlock`（SubscriberName）: 名前付きサブスクライバの整合セットをロック解除します。
- `purge`（パラメータなし）: 使用された変更をパージします。

これらの操作の詳細は、「チェンジ・データ・キャプチャ」を参照してください。

データストア・レベルのサービス

生成された各データ・サービスにより提供される操作の範囲は、その生成に使用された SKM に応じて異なります。現在、Oracle Data Integrator に付属する SKM では、いくつかの共通プロパティが共有されています。ほとんどの場合、公開されるエンティティの名前が各操作の名前の一部となります。次に、公開エンティティ Customer を使用した例を示します。

- **単一**のエンティティに対する操作。これらの操作では、主キーの値を指定することで単一のレコードを操作できます。新規行が存在する場合は、その行を記述する他のフィールドを指定する必要があります。
  例: `addcustomer`、`getcustomer`、`deletecustomer`、`updatecustomer`
- **フィルタ**で指定されたエンティティ・グループに対する操作。これらの操作では、1 つまたは複数のフィールドの値を指定してフィルタを定義し、各行に追加された変更に対応する他の

値をオプションで指定します。通常、戻される行の最大数も指定できます。
例: `getcustomerfilter`、`deletecustomerfilter`、`updatecustomerfilter`

- エンティティの**リスト**に対する操作。このリストは、前述の単一のエンティティの場合と同様に、個々のエンティティを複数指定することで作成します。
例: `addcustomerlist`、`deletecustomerlist`、`getcustomerlist`、`updatecustomerlist`

データ・サービスのテスト

生成されたデータ・サービスをテストする最も簡単な方法は、Oracle Data Integrator ツールの SnpsInvokeWebService のグラフィカル・インタフェースを使用することです。詳細は、「Web サービスの使用方法」も参照してください。

プロジェクト

プロジェクトの概要

プロジェクトは、Oracle Data Integrator を使用して開発されたオブジェクトのグループです。

- プロジェクトの管理

Oracle Data Integrator プロジェクト・コンポーネント

ツリー表示のプロジェクトの配下に、次のコンポーネントが格納されます。

フォルダ

プロジェクト内の一部のオブジェクトは、**フォルダ**と**サブフォルダ**に整理されます。

パッケージ

パッケージは、Oracle Data Integrator で最大の実行単位です。パッケージは、実行ダイアグラムに整理された一連の**手順**で構成されています。

- パッケージの詳細情報
- パッケージの作成

インタフェース

インタフェースは、1 つ以上のソース・データストアから、データストアまたは一時ターゲット構造をロードする方法を定義した一連のルールで構成されます。

- インタフェースの詳細情報
- インタフェースの作成

プロシージャ

固有プロシージャは、インタフェース・フレームワークにそぐわない操作をグループ化した、再利用可能なコンポーネントで、1 つ以上のソースからターゲットデータストアをロードします。

プロシージャの例:

- 待機してファイルを unzip する
- FTP を介してファイルを一括送信する
- 電子メールを受信する
- データベースをパージする

プロシージャは、トポロジで定義された論理スキーマで、コマンドを起動することができますが、OS コマンドや Oracle Data Integrator ツールを使用することもできます。

変数

変数の値は Oracle Data Integrator に格納されます。この値は、実行中に変化することもあります。

値には次の性質があります。

- 作成時に定義されたデフォルト値を持ちます。
- 変数を使用したシナリオの実行時にパラメータとして渡すことができます。
- 変数のリフレッシュ、設定および増分ステップで変更できます。
- パッケージで条件を作成して評価できます。
- インタフェース、プロシージャ、ステップなどで使用できます。

変数をプロジェクトの外部（グローバル・スコープ）で定義することにより、すべてのプロジェクトで使用されるようにすることもできます。

順序

順序は、使用時に自動的に増分される変数です。その値は、1 回使用されてから次に使用されるまで持続します。

順序は、インタフェース、プロシージャ、ステップなどで変数と同じように使用できます。

変数と同様に、プロジェクトの外部（グローバル・スコープ）で順序を定義すれば、すべてのプロジェクトで使用されるようにできます。

ユーザー関数

ユーザー関数を使用すると、テクノロジ依存の実装を定義する場合に、カスタマイズされた関数や関数の別名を定義できます。ユーザー関数は、インタフェースとプロシージャで使用できます。

ナレッジ・モジュール

Oracle Data Integrator は、指定されたテクノロジに関連するメソッドを定義するためにナレッジ・モジュールを使用します。これらのモジュールでは、専用の機能に特化したテクノロジ用のプロセスを生成できます。

注意: デフォルトのナレッジ・モジュールは Oracle Data Integrator に付属しており、使用前にプロジェクトにインポートする必要があります。

マーカー

開発の方法論または組織を反映するために、プロジェクトの要素にフラグを設定することがあります。

フラグはマーカーを使用して定義されます。このマーカーはグループに整理され、プロジェクト内で大部分のオブジェクトに適用できます。

シナリオ

パッケージ、インタフェース、プロシージャまたは変数コンポーネントが終了すると、それは**シナリオ**にコンパイルされます。シナリオは、スケジュール可能な本番環境用の実行単位です。

Oracle Data Integrator プロジェクトの管理

Oracle Data Integrator プロジェクトの一般的な管理手順は、次のとおりです。

1. モデルの作成およびリバース・エンジニアリング
2. プロジェクトの作成
3. マーカーの使用（オプション）
4. フォルダの作成および編成
5. ナレッジ・モジュール（KM）のインポート
6. 次の再利用可能オブジェクトの作成および変更
   - 変数
   - 順序
   - インタフェース
   - プロシージャ
   - ユーザー関数
7. インタフェースとプロシージャの単体テスト（手順 6 に戻る）
8. 手順 6 で作成した要素からのパッケージの構築
9. パッケージの統合テスト
10. シナリオの生成
11. シナリオのスケジュール
12. 本番環境でのシナリオ管理
13. メンテナンス、不具合の修正および追加変更（手順 6 に戻る）

新規プロジェクトの作成

プロジェクトを作成する手順:

1. **「プロジェクト」**ビューで**「プロジェクトの挿入」**ボタンをクリックします。

2. プロジェクトの**「名前」**を入力します。
3. 自動生成された**「コード」**をそのまま使用するか変更します。
4. **「OK」**をクリックします。

インタフェース、プロシージャおよびパッケージ用の空のフォルダを含む空のプロジェクトが表示されます。

新規フォルダの作成

インタフェース、プロシージャおよびパッケージは、**フォルダ**と**サブフォルダ**に編成します。

フォルダを作成する手順:

1. フォルダを作成する**プロジェクト**または**フォルダ**をクリックします。
2. 右クリックし、**「フォルダの挿入」**を選択します。
3. フォルダの**「名前」**を入力します。
4. **「OK」**をクリックします。

空のフォルダが表示されます。

パッケージの作成

パッケージの概要

パッケージは、Oracle Data Integrator で最大の実行単位です。パッケージは、実行ダイアグラム内に編成された一連の**ステップ**で構成されます。

- パッケージの作成

手順

手順にはいくつかのタイプがあります。それらは、次の手順群にグループ化することができます。

- **フロー（インタフェース）**: インタフェースを実行します。
- **プロシージャ**: プロシージャを実行します。
- **変数**: 変数の値の宣言、設定、リフレッシュ、または評価を行います。
- **Oracle Data Integrator Tools**: ツールボックスから使用可能なこれらのツールは、すべての Oracle Data Integrator の API コマンドへのアクセスや、オペレーティング・システム・コールを可能にします。
- **モデル、サブモデル**および**データストア:** これらのオブジェクトに対してジャーナル化、静的管理またはリバース・エンジニアリング操作を行います。

たとえば、「Populate Sales Datamart」パッケージを、次のジョブで構成することができます。

1. プロシージャ「System Backup」
2. インタフェース「Customer Group」
3. インタフェース「Customer」
4. インタフェース「Product」
5. 変数「Last Invoice ID」のリフレッシュ

6. インタフェース「Invoice Header」
7. インタフェース「Invoice Lines」

パッケージの作成

パッケージの作成は、次のプロセスに従って実行します。

1. 新規パッケージを作成します。
2. ステップを操作します（挿入、複製、削除など）。
3. ステップの順序を定義します。
4. パッケージを実行します。

1. 新規パッケージの作成

新規パッケージを作成する手順:

1. パッケージを作成する**フォルダ**の**「パッケージ」**ノードをクリックします。
2. 右クリックし、**「パッケージの挿入」**を選択します。
3. パッケージの**「名前」**を入力します。
4. **「OK」**をクリックします。

新規パッケージが作成されます。

2. ステップの操作

ステップを挿入する手順:

ステップの追加方法は、挿入するステップの性質に応じて異なります。詳細は、「ステップの追加」を参照してください。

ステップを削除する手順:

1. **パッケージ**の**「ダイアグラム」**タブで、をクリックします。
2. ステップのアイコンを右クリックし、**「ステップの削除」**を選択して**「OK」**をクリックします。

ダイアグラムからステップが削除されます。

ステップを複製する手順:

1. **パッケージ**の**「ダイアグラム」**タブで、をクリックします。
2. ステップの**アイコン**を右クリックし、**「ステップの複製」**を選択します。

ダイアグラムに新規ステップが表示されます。

ステップを実行する手順:

1. **パッケージ**の**「ダイアグラム」**タブで、をクリックします。
2. ステップのアイコンを右クリックし、**「ステップの実行」**を選択します。
3. 実行オプションを設定し、**「OK」**をクリックします。

ステップのリンク・オブジェクトを編集する手順:

ステップのリンク・オブジェクトは、ステップの作成元であるインタフェースやプロシージャなどです。

1. **パッケージ**の**「ダイアグラム」**タブで、をクリックします。
2. ステップのアイコンをダブルクリックします。リンク・オブジェクトの編集ウィンドウが表示されます。

ダイアグラムのステップを整列する手順:

ダイアグラムを見やすくするために、ダイアグラム内のステップを整列できます。

1. **パッケージ**の**「ダイアグラム」**タブで、をクリックします。
2. 次の方法で 1 つ以上の**ステップ**を選択します。
   - [Ctrl]キーを押しながら各ステップをクリックします。
   - 左のマウス・ボタンを押したままダイアグラムでカーソルをドラッグします。ステップを囲む選択ボックスが表示されます。
3. 選択したステップを整列するには、次のいずれかの方法を使用します。
   - ステップをドラッグしてその位置を調整します。
   - 右クリックし、コンテキスト・メニューの垂直または水平整列オプションを選択します。

ツールバーの**「再編成」**ボタンを使用して、各ステップを自動的に再調整することも可能です。

3. ステップの順序の定義

ステップを作成したら、データ処理チェーンに従って各ステップの順序を再設定する必要があります。このチェーンには、次のルールがあります。

- チェーンは、**「最初のステップ」**として定義された一意のステップから開始します。
- 各ステップには、**「成功」**または**「失敗」**という 2 つの終了状態があります。
- 失敗または成功したステップは、別のステップに続けるか、パッケージの最後とすることができます。
- 失敗した場合に備え、再試行回数を定義できます。

パッケージのエントリ・ポイントは 1 つのみですが、終了ステップは複数用意することが可能です。

失敗条件

次の表に、ステップが**「失敗」**状態になる条件を示します。これ以外の場合、ステップは**「成功」**状態で終了します。

ステップ・タイプ	失敗条件
フロー	• インタフェース・コマンドでエラーが発生した場合 • 許容される最大エラー数または最大エラー割合に到達した場合
プロシージャ	• プロシージャ・コマンドでエラーが発生した場合
変数のリフレッシュ	• リフレッシュ順序でエラーが発生した場合

変数の設定	• 変数の設定時にエラーが発生した場合（無効な値）
変数の評価	• ステップで定義された条件が無効の場合
変数の宣言	• なし
OS コマンドの実行	• OS コマンドのリターン・コードが 0（ゼロ）以外の場合
Oracle Data Integrator ツール	• Oracle Data Integrator コマンドのリターン・コードが 0（ゼロ）以外の場合
データストア、モデルまたはサブモデルのジャーナル化	• ジャーナル化コマンドでエラーが発生した場合
データストア、モデルまたはサブモデルのチェック	• チェック・プロセスでエラーが発生した場合
モデルのリバース	• リバース・プロセスでエラーが発生した場合

順序の定義

パッケージの最初のステップを定義する手順:

1. **「ダイアグラム」**タブで、をクリックします。
2. **最初のステップ**として定義するステップのアイコンをクリックします。
3. 右クリックし、**「最初のステップ」**ボックスを選択します。

ステップのアイコンに、**最初のステップ**であることを示す記号が表示されます。

失敗時の次のステップを定義する手順:

1. **「ダイアグラム」**タブで、をクリックします。
2. **失敗**する可能性のあるステップのアイコンをクリックします。
3. マウス・ボタンを押したまま、**失敗**時に進む必要のあるステップのアイコンまでカーソルを移動し、マウス・ボタンを離します。

2 つのステップの間に赤色の矢印が表示されます（変数の評価ステップの場合は緑色です）。

失敗時のパッケージの最終ステップを定義する手順:

1. **「ダイアグラム」**タブで、をクリックします。
2. **失敗**する可能性のあるステップのアイコンをクリックします。
3. プロパティ・ペインの**「拡張」**タブで、**「失敗後の処理」**セクションの**「終了」**を選択します。

エラーになる前の再試行回数を定義する手順:

1. **「ダイアグラム」**タブで、をクリックします。
2. **失敗**する可能性のあるステップのアイコンをクリックします。
3. プロパティ・ペインで、**「拡張」**タブに移動します。
4. **「試行回数」**および**「試行間隔」**を入力します。

成功時の次のステップを定義する手順:

1. **「ダイアグラム」**タブで、  をクリックします。
2. **成功**する可能性のあるステップのアイコンをクリックします。
3. マウス・ボタンを押したまま、**成功**時に進む必要のあるステップのアイコンまでカーソルを移動し、マウス・ボタンを離します。

2 つのステップの間に緑色の矢印が表示されます。

成功時のパッケージの最終ステップを定義する手順:

1. **「ダイアグラム」**タブで、  をクリックします。
2. **成功**する可能性のあるステップのアイコンをクリックします。
3. プロパティ・ペインの**「拡張」**タブで、**「成功後の処理」**セクションの**「終了」**を選択します。

4. パッケージの実行

パッケージを実行する手順:

1. **パッケージ**の**「実行」**タブで、**「実行」**をクリックします。
2. パッケージを保存していない場合、**「はい」**をクリックして変更を適用します。
3. 実行ウィンドウに移動し、**「OK」**をクリックします。

ステップの追加

パッケージにステップを追加する手順は、ステップのタイプに応じて異なります。

フロー（インタフェース）

パッケージにフロー・ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。
2. パッケージに追加する**インタフェース**をツリー・ビューから選択します。
3. インタフェースをダイアグラムにドラッグ・アンド・ドロップします。**フロー・ステップ**が表示されます。
4. ダイアグラムのステップ・アイコンをクリックします。プロパティ・パネルが表示されます。
5. **「一般」**タブで、**「名前」**フィールドを入力します。
6. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。
7. **「適用」**をクリックして保存します。

プロシージャ

パッケージにプロシージャ・ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。
2. パッケージに追加する**プロシージャ**をツリー・ビューから選択します。

3. プロシージャをダイアグラムにドラッグ・アンド・ドロップします。**プロシージャ・ステップ**が表示されます。
4. ダイアグラムのステップ・アイコンをクリックします。プロパティ・パネルが表示されます。
5. **「一般」**タブで、**「名前」**フィールドを入力します。
6. **「オプション」**タブで、プロシージャのオプションを設定します。
7. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。
8. **「適用」**をクリックして保存します。

変数

パッケージに変数の宣言ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。
2. パッケージに追加する**変数**をツリー・ビューから選択します。
3. 変数をダイアグラムにドラッグ・アンド・ドロップします。**変数ステップ**が表示されます。
4. ダイアグラムのステップ・アイコンをクリックします。プロパティ・パネルが表示されます。
5. **「一般」**タブで、**「名前」**フィールドを入力します。ステップの**「タイプ」**として**「変数の宣言」**を選択します。
6. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。
7. **「適用」**をクリックして保存します。

パッケージに変数のリフレッシュ・ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。
2. パッケージに追加する**変数**をツリー・ビューから選択します。
3. 変数をダイアグラムにドラッグ・アンド・ドロップします。**変数ステップ**が表示されます。
4. ダイアグラムのステップ・アイコンをクリックします。プロパティ・パネルが表示されます。
5. **「一般」**タブで、**「名前」**フィールドを入力します。ステップの**「タイプ」**として**「変数のリフレッシュ」**を選択します。
6. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。
7. **「適用」**をクリックして保存します。

パッケージに変数の設定ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。
2. パッケージに追加する**変数**をツリー・ビューから選択します。
3. 変数をダイアグラムにドラッグ・アンド・ドロップします。**変数ステップ**が表示されます。
4. ダイアグラムのステップ・アイコンをクリックします。プロパティ・パネルが表示されます。
5. **「一般」**タブで、**「名前」**フィールドを入力します。ステップの**「タイプ」**として**「変数の設定」**を選択します。
6. 実行する操作に応じて、**「割当て」**または**「増分」**を選択します。
7. 割り当てる**「値」**または増分量を入力します。

8. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。
9. **「適用」**をクリックして保存します。

パッケージに変数の評価ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。
2. パッケージに追加する**変数**をツリー・ビューから選択します。
3. 変数をダイアグラムにドラッグ・アンド・ドロップします。**変数ステップ**が表示されます。
4. ダイアグラムのステップ・アイコンをクリックします。プロパティ・パネルが表示されます。
5. **「一般」**タブで、**「名前」**フィールドを入力します。ステップの**「タイプ」**として**「変数の評価」**を選択します。
6. 評価用の**「オペレータ」**および比較する**「値」**を選択します。この値は、他の変数にすることもできます。
7. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。変数の評価ステップにおける成功とは、評価が肯定的結果となる場合です。
8. **「適用」**をクリックして保存します。

Oracle Data Integrator ツール

Oracle Data Integrator ツール・ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。
2. **「ツールボックス」**で、使用するツールを選択します。
3. ダイアグラムをクリックします。ツールに対応する**ステップ**が表示されます。
4. をクリックします。
5. ダイアグラムのステップ・アイコンをダブルクリックします。プロパティ・パネルが表示されます。
6. **「一般」**タブで、ステップの**「名前」**を入力します。
7. ツールのパラメータとして適切な値を入力または選択します。
8. **「コマンド」**タブに、生成された Oracle Data Integrator コマンドが表示されます。コードを編集する場合、をクリックすることで式エディタを使用できます。
9. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。
10. **「適用」**をクリックして保存します。

注意: Web サービスを起動する場合、カスタマイズされた GUI を使用できます。詳細は、「Web サービスの使用方法」を参照してください。

ヒント: SnpsStartScen（Oracle Data Integrator シナリオの起動）ステップを作成する場合、プロジェクトのツリー・ビューからダイアグラムにシナリオを直接ドラッグ・アンド・ドロップできます。また、**「追加変数」**を使用してシナリオ変数を入力できます。

パッケージに OS コマンド・ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。

2. **「ツールボックス」**で、（**「ユーティリティ」**グループの）**「OS コマンド」**を選択します。
3. ダイアグラムをクリックします。**OS コマンド・ステップ**が表示されます。
4. をクリックします。
5. ダイアグラムのステップ・アイコンをダブルクリックします。プロパティ・パネルが表示されます。
6. **「一般」**タブで、ステップの**「名前」**を入力します。
7. **「コマンドのテキスト」**を入力します。をクリックすることで式エディタを表示できます。
8. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。
9. **「適用」**をクリックして保存します。

警告: OS コマンドを使用すると、作成したパッケージがプラットフォーム依存となる可能性があります。

パッケージに Oracle Data Integrator コマンド・ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。
2. **「ツールボックス」**で、（**「ユーティリティ」**グループの）Oracle Data Integrator **コマンド**を選択します。
3. ダイアグラムをクリックします。Oracle Data Integrator **コマンド・ステップ**が表示されます。
4. をクリックします。
5. ダイアグラムのステップ・アイコンをダブルクリックします。プロパティ・パネルが表示されます。
6. **「一般」**タブで、ステップの**「名前」**を入力します。
7. **「コマンド」**タブに移動し、Oracle Data Integrator コマンドを入力します。コードを編集する場合、をクリックすることで式エディタを使用できます。**「一般」**タブに戻ると、Oracle Data Integrator コマンドが認識され、プロパティ・リストが表示されます。このタブを使用して、パラメータ値を入力できます。
8. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。
9. **「適用」**をクリックして保存します。

各 Oracle Data Integrator ツールの詳細は、『Oracle Data Integrator Tools リファレンス・マニュアル』を参照してください。

Oracle Data Integrator Open Tools ステップを挿入する手順:

Open Tools は、**「プラグイン」**グループの下にリストされ、他のツールと同じように使用できます。Open Tools をインストールして使用する方法の詳細は、『Oracle Data Integrator Tools リファレンス・マニュアル』の「Open Tools の使用」を参照してください。

モデルおよびデータストア

パッケージにチェック・ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。

2. パッケージに追加する**モデル**または**データストア**をツリー・ビューから選択します。
3. モデルまたはデータストアをダイアグラムにドラッグ・アンド・ドロップします。**モデル**または**データストア・ステップ**が表示されます。
4. ダイアグラムのステップ・アイコンをダブルクリックします。プロパティ・パネルが表示されます。
5. **「一般」**タブで、**「名前」**フィールドを入力します。ステップの**「タイプ」**として**「データストア/モデルのチェック」**を選択します。
6. チェックした表からエラーを削除する場合は、**「チェック済表からのエラーの削除」**を選択します。
7. **「オプション」**タブで、特定の KM オプションを設定します。
8. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。
9. **「適用」**をクリックして保存します。

注意: チェックを実行するには、モデルで CKM を定義する必要があります。

パッケージにジャーナル化ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。
2. パッケージに追加する**モデル**または**データストア**をツリー・ビューから選択します。
3. モデルまたはデータストアをダイアグラムにドラッグ・アンド・ドロップします。**モデル**または**データストア・ステップ**が表示されます。
4. ダイアグラムのステップ・アイコンをクリックします。プロパティ・パネルが表示されます。
5. **「一般」**タブで、**「名前」**フィールドを入力します。ステップの**「タイプ」**として**「データストア/モデルのジャーナル化」**を選択します。
6. 実行するジャーナル化操作に対応するパラメータを設定します。詳細は、「チェンジ・データ・キャプチャ」を参照してください。
7. **「オプション」**タブで、特定の KM オプションを設定します。
8. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。
9. **「適用」**をクリックして保存します。

注意: ジャーナル化を実行するには、モデルで JKM を定義する必要があります。

パッケージにリバース・エンジニアリング・ステップを挿入する手順:

1. **パッケージ**を開き、**「ダイアグラム」**タブに移動します。
2. パッケージに追加する**モデル**をツリー・ビューから選択します。
3. モデルをダイアグラムにドラッグ・アンド・ドロップします。**モデル・ステップ**が表示されます。
4. ダイアグラムのステップ・アイコンをクリックします。プロパティ・パネルが表示されます。
5. **「一般」**タブで、**「名前」**フィールドを入力します。ステップの**「タイプ」**として**「モデルのリバース」**を選択します。
6. **「オプション」**タブで、特定の KM オプションを設定します。

7. **「拡張」**タブで、成功または失敗後の処理に関するオプションを定義します（「ダイアグラム」を参照）。
8. **「適用」**をクリックして保存します。

注意: モデルでリバース・エンジニアリング・オプションを定義する必要があります。

シナリオの作成

シナリオの生成

コンポーネントを完成してテストしたら、実際の状況に対応するシナリオを生成してそのコンポーネントを実行できます。この実行は、**デザイナ**・モジュールで処理されます。

パッケージ、プロシージャ、インタフェースまたは変数のシナリオを生成できます。パッケージ以外のタイプは、単一ステップのシナリオです。

シナリオ変数は、シナリオの起動時にその動作をパラメータ化するために設定する必要のあるシナリオ用の変数です。

注意: 変数用に生成されたシナリオには、変数のリフレッシュ操作を実行する単一のステップが含まれます。

シナリオを生成する手順:

1. 「プロジェクト」ビューで、**パッケージ**、**インタフェース**、**プロシージャ**または**変数**をダブルクリックします。
   編集ウィンドウが表示されます。
2. **「シナリオ」**タブを選択します。
3. **「シナリオの生成」**ボタンをクリックします。
   新規シナリオ・ウィンドウが表示されます。
4. シナリオの**「名前」**を入力します。この名前は後でオペレーティング・システム・コマンドとして使用される可能性があるため、特殊文字や空白は使用しないでください。
5. シナリオの**「バージョン」**を入力します（文字に関する注意はシナリオ名の場合と同じです）。
6. **「OK」**をクリックします。
7. シナリオで変数を使用する場合、**シナリオ変数**ウィンドウでシナリオのパラメータとみなす変数を定義できます。すべての変数をパラメータとする場合は、**「すべてを使用」**を選択します。または、**「選択的使用」**を選択してパラメータ変数を指定します。

「プロジェクト」ビューにあるソース・オブジェクトの**「シナリオ」**ノードの下にシナリオが表示されます。

既存のシナリオは、同じ名前とバージョン番号で再生成できます。これにより、ソース・オブジェクトの内容から生成したシナリオで既存のシナリオを置換できます。

シナリオを再生成する手順:

1. ツリーで**シナリオ**を選択します。
2. 右クリックし、**「再生成」**を選択します。
3. **「OK」**をクリックします。

警告: シナリオの再生成は、元に戻すことができません。重要なシナリオの場合は、新しいバージョン番号でシナリオを生成することをお薦めします。

シナリオ・グループの生成

プロジェクトまたは**フォルダ**にグループ化された**パッケージ、インタフェース、プロシージャおよび変数**のセットを完成してテストしたら、シナリオを生成してそれらを実行するよう準備できます。この実行は、**デザイナ**・モジュールで処理されます。

シナリオ・グループを生成する手順:

1. オブジェクトのグループを含む**プロジェクト**または**フォルダ**を選択します。
2. 右クリックし、**「すべてのシナリオの生成」**を選択します。
3. **シナリオ生成**ウィンドウで、シナリオの**「生成モード」**を次から選択します。
   - **置換:** 最後に生成されたシナリオを新しく生成されたシナリオで置換します。
   - **作成:** バージョン番号を自動的に増分して、最後に生成されたシナリオと同じ名前で新規シナリオを生成します。まだシナリオが作成されていない場合、新規シナリオにはソース・コンポーネントの名前が割り当てられます。

注意: **「生成モード」**で**「作成」**を選択した場合、最後のシナリオの**バージョン**が整数であれば、バージョン番号は自動的に 1 ずつ増加します。それ以外の場合、バージョンには自動的に現在の日付が設定されます。

4. シナリオを生成するオブジェクトのタイプを選択します。
5. マーカー・グループのマーカーに従って、生成するコンポーネントをフィルタします。
6. **「OK」**をクリックします。
7. シナリオで変数を使用する場合、**シナリオ変数**ウィンドウでシナリオのパラメータとみなす変数を定義できます。すべての変数をパラメータとする場合は、**「すべてを使用」**を選択します。または、**「選択的使用」**を選択してパラメータ変数を指定します。

シナリオのエクスポート

エクスポート（およびインポート）の手順により、Oracle Data Integrator オブジェクトをあるリポジトリから別のリポジトリに移動できます。

単一のシナリオ群、またはシナリオ群のグループをエクスポートできます。

1 つのシナリオをエクスポートするには、「オブジェクトのエクスポート」を参照してください。

シナリオ・グループをエクスポートする手順:

1. シナリオのグループを含む**プロジェクト**または**フォルダ**を選択します。
2. 右クリックし、**「すべてのシナリオのエクスポート」**を選択します。
3. エクスポート・パラメータを入力します。
   「エクスポート・ディレクトリ」を指定しない場合、エクスポート・ファイルは**「デフォルトのエクスポート・ディレクトリ」**に作成されます。
4. シナリオをエクスポートする必要のあるオブジェクトのタイプを選択します。
5. **「OK」**をクリックします。

XML 形式のエクスポート・ファイルが指定した場所に作成されます。

シナリオの暗号化

シナリオを暗号化すると、重要なコードを保護できます。暗号化されたシナリオでは、ログに表示されるコマンドは実行されますが、ユーザーには解読できません。

Oracle Data Integrator では、**個人暗号化キー**に基づく DES 暗号化アルゴリズムが使用されます。このキーは、ファイルに保存して、暗号化または復号化操作を実行するために再利用できます。

警告: 暗号化されたシナリオを**暗号化キー**なしで復号化する方法はありません。したがって、このキーは安全な場所に保管しておくことを強くお薦めします。また、すべての開発作業でただ 1 つのキーを使用することをお薦めします。

シナリオを暗号化する手順:

1. 暗号化するシナリオを右クリックします。
2. **「暗号化」**を選択します。
3. **暗号化オプション**・ウィンドウで、次のいずれかの操作を実行します。
   - **「個人キーで暗号化します」**オプションを選択し、既存の暗号化キー・ファイルを選択します。
   - **「個人キーで暗号化します」**オプションを選択し、独自の**個人キー**に対応する文字列を入力するか貼り付けます。
   - **「新規暗号化キーを取得します」**オプションを使用して、Oracle Data Integrator でキーを生成します。
4. 暗号化が完了すると、**暗号化キー**・ウィンドウが表示されます。このウィンドウで、キーを保存できます。

注意: 文字数が少なすぎる個人キーを入力すると、**キー・サイズが無効**であるというエラーが発生します。この場合、より長い個人キーを入力してください。個人キーには、10 個以上の文字数が必要です。

シナリオを復号化する手順:

1. 復号化するシナリオを右クリックします。
2. **「復号化」**を選択します。
3. **シナリオ復号化**ウィンドウで、次のいずれかの操作を実行します。
   - 既存の暗号化キー・ファイルを選択します。
   - 独自の**個人キー**に対応する文字列を入力するか貼り付けます。

復号化が完了するとメッセージが表示されます。

インタフェースの作成

インタフェースの概要

インタフェースは、1 つ以上のソース・データストアからのデータストアまたは一時ターゲット構造のロードを定義する 1 組のルールで構成されています。

- インタフェースの作成

インタフェースのコンポーネント

ターゲット・データストア

ターゲット・データストアは、インタフェースによってロードされる要素です。永続データストア（モデル内で定義される）と、一時データストア（インタフェースによってステージング領域に作成される）があります。

ソース・データストア

ソース・データストアには、ターゲット・データストアをロードするために使用されるデータが含まれます。モデルからのデータストアと、インタフェースの一時データストア・ターゲットの、2 タイプのデータストアをインタフェースのソースとして使用することができます。

インタフェースのソース・データストアは、ロード処理時にフィルタでき、結合を使用してリレーション内に置かれる必要があります。**結合**と**フィルタ**はモデル定義から回復でき、またインタフェースで定義することもできます。

マッピング

マッピングはソースにあるトランスフォーメーション・ルールで、ターゲットをロードするためのデータの生成を可能にします。

フロー

フローは、マップされるデータのロードと統合に関する 1 組の戦略で、**ナレッジ・モジュール**上にあります。

コントロール戦略

フロー・コントロール戦略では、ターゲットに挿入する前にフローをチェックするために使用されるメソッドを定義できます。コントロール戦略は**チェック・ナレッジ・モジュール**（CKM）によって定義されます。

インタフェースは次のコンポーネントを使用します。

- モデル内で、ソースとターゲット、またはロード処理として定義された**データストア**。
- 適切なプロセスを生成するための**ナレッジ・モジュール**。
- 式の値またはカウンタを格納するための**変数**と**シーケンス**。
- トランスフォーメーション・ルールのコーディングを容易にする**ユーザー・ファンクション**。

インタフェースの作成

インタフェースを作成するには、次の操作を実行する必要があります。

1. 新規インタフェースを作成します。
2. ターゲット・データストアを定義します。
3. ソース・データストアを定義し、それらのソースでフィルタおよび結合を定義します。
4. ソース・データとターゲット・データ間のマッピングを定義します。
5. インタフェース・フローを定義します。
6. フロー制御を定義します。
7. テスト目的でインタフェースを実行します。

1. 新規インタフェースの作成

新規インタフェースを作成する手順:

1. デザイナ・ツリーで、インタフェースを作成する**プロジェクト**の**フォルダ**に含まれる**「インタフェース」**ノードを選択します。右クリックし、**「インタフェースの挿入」**を選択します。デザイナ・ウィンドウの右側のパネルに空のインタフェースが表示されます。
2. インタフェースの**「名前」**を入力します。
3. 必要に応じて**「ターゲットと異なるステージング領域」**ボックスを選択し、ステージング領域とする論理スキーマを選択します。

注意: ステージング領域のデフォルトは、ターゲットです。インタフェースに必要とされる変換機能がターゲットにない場合、異なる論理スキーマにステージング領域を配置する必要があります。これは、ファイルや JMS などの論理スキーマに当てはまります。また、一時ターゲット・データストアでインタフェースのステージング領域を定義する必要もあります。

4. **「ダイアグラム」**タブに移動して操作を続けます。

2. ターゲット・データストアの定義

ターゲット・データストアは、インタフェースによってロードされる要素です。（モデルで定義される）永続データストアまたは（インタフェースによりステージング領域に作成される）一時データストアを使用できます。

2.1 永続ターゲット・データストア

インタフェースに永続ターゲット・データストアを挿入する手順:

1. ツリーを開き、ターゲットとして挿入する**データストア**を表示します。
2. **データストア**を選択し、（「ダイアグラム」タブの右側に）インタフェースのターゲットとしてドラッグ・アンド・ドロップします。ダイアグラムにターゲット・データストアが表示されます。

インタフェースの永続ターゲット・データストアのデータを表示する手順:

1. **インタフェース**・ウィンドウの**「ダイアグラム」**タブで、ターゲット・データストアのタイトルを右クリックします。
2. **「データ」**を選択します。

ターゲット・データストアのデータを含むウィンドウが表示されます。一時ターゲット・データストアは、インタフェースによって作成されるため、そのデータは表示できません。

2.2 一時ターゲット・データストア

一時ターゲット・データストアを追加する手順:

1. **「ダイアグラム」**タブで、ターゲット・データストアのタイトル**「無題」**をダブルクリックします。
2. プロパティ・ペインで、このデータストアの**「名前」**を入力します。一時データストアをステージング領域の**「作業スキーマ」**または**「データ・スキーマ」**のどちらに作成するかを指定します。

注意: ステージング領域がターゲットと異なる場合にのみ、一時ターゲット・データストアを定義できます。

注意: 一時ターゲット・データストアは、フローを定義する際に KM オプションの CREATE_TARGET_TABLE をアクティブ化した場合にのみ作成されます。

列のない一時ターゲット・データストアが作成されます。列を追加して構造を定義する必要があります。

一時ターゲット・データストアに列を追加する手順:

1. ターゲット・データストアのタイトルを右クリックします。
2. **「列の追加」**を選択します。
3. 新しい空の列が表示されます。ターゲットでこの新規列をダブルクリックし、**「ターゲット列」**フィールドを入力します（「マッピング」を参照）。

一時ターゲット・データストアから列を削除する手順:

1. ターゲット・データストアから削除する**列**を右クリックします。
2. **「削除」**を選択します。

一時ターゲット・データストアにソース・データストアのすべての列を追加する手順:

1. ソース・**データストア**を追加します。
2. ソース・データストアを表す**エンティティ**のタイトルを選択します。
3. 右クリックし、**「ターゲットに追加します」**を選択します。

2.3 更新キーの定義

インタフェースで更新機能またはフロー制御機能を使用する場合、状況に応じてターゲット・データストアに更新キーを定義する必要があります。このキーにより、ターゲットへの挿入前に更新またはチェック対象となる各レコードが識別されます。このキーとして、モデルのターゲット・データストアで定義された一意キーか、またはインタフェースのキーとして指定された列のグループを使用できます。

一意キーから更新キーを定義する手順:

1. **「ダイアグラム」**タブで、ターゲット・データストアのタイトルをダブルクリックします。
2. プロパティ・パネルで、リストから**「更新キー」**を選択します。

注意: リストに表示されるのは、モデルに定義されているキーのみです。

列から更新キーを定義する手順:

1. **「ダイアグラム」**タブで、更新キーの一部となるターゲット・データストアの列の 1 つをダブルクリックします。
2. プロパティ・パネルで**「キー」**ボックスを選択します。マッピングの列の前に、キーの記号が表示されます。
3. 更新キーを構成する列ごとにこの操作を繰り返します。

3. ソース・データストアの定義

ソース・データストアには、ターゲット・データストアのロードに使用されるデータが含まれます。インタフェース・ソースとして、モデルのデータストアと、インタフェースの一時データストア・ターゲットという 2 つのタイプのデータストアを使用できます。

インタフェースのソース・データストアは、ロード・プロセス中にフィルタできます。また、ソース・データストアには、結合を通じてリレーションを設定する必要があります。結合とフィル

タは、モデル定義からリカバリできます。また、結合とフィルタはインタフェースで定義することもできます。

3.1 ソース・データストアの定義

インタフェースに永続タイプのソース・データストアを追加する手順:

1. **データストア**を選択し、（「ダイアグラム」タブの左側の）構成パネルにドラッグ・アンド・ドロップします。ダイアグラム・モデルにデータストアが表示されます。

警告: データストアにフィルタが存在するか、そのデータストアとダイアグラムの既存のデータストアの間に参照が存在する場合、それらはデータストアとともに表示されます。各参照およびフィルタは、インタフェースの結合およびフィルタとしてコピーされます。これらの結合およびフィルタは、モデルの既存の参照およびフィルタにはリンクしません。したがって、モデルの参照またはフィルタを変更しても、インタフェースの結合またはフィルタに影響を与えることはありません（逆も同様です）。

ジャーナル化されるデータストアの使用: データストアがジャーナル化される場合、インタフェース・フローではジャーナル化データのみを使用できます。ソース・データストアのプロパティの**「ジャーナル化されたデータのみ」**ボックスを選択します。ジャーナル化フィルタが自動的にダイアグラム内に作成されます。

インタフェースに一時タイプのソース・データストアを追加する手順:

1. **インタフェース**を選択し、（「ダイアグラム」タブの左側の）構成パネルにドラッグ・アンド・ドロップします。ダイアグラム・モデルにこのインタフェースのターゲット・データストアが表示されます。

インタフェースからソース・データストアを削除する手順:

1. ソース・データストアを表す**エンティティ**（表）のタイトルを右クリックします。
2. **「削除」**を選択します。確認ウィンドウが表示されます。**「はい」**を選択します。

関連するフィルタおよび結合とともにソース・データストアが削除されます。

インタフェースのソース・データストアのデータまたは行数を表示する手順:

1. ソース・データストアを表す**エンティティ**（表）のタイトルを右クリックします。
2. **「行数」**または**「データ」**を選択します。

ソース・データストアの行数またはデータを含むウィンドウが表示されます。

3.2 ソースでのフィルタの定義

ソース・データストアにフィルタを定義する手順:

1. 結合する最初の表を表すエンティティの列を選択し、選択したまま構成パネルにドラッグ・アンド・ドロップします。モデルに**フィルタ**が表示されます。
2. **「実装」**を変更して必要なフィルタを作成します。  をクリックすることで式エディタを表示できます。
3. 実行場所（**「ソース」**または**「ステージング領域」**）を選択します。
4.  をクリックして式を検証し、  をクリックして保存します。
5. **「アクティブ・フィルタ」**ボックスを選択して、インタフェースの実行でこのフィルタを使用するかどうかを定義します。

インタフェースのソース・データストアのフィルタを削除する手順:

1. 削除する**フィルタ**を選択します。

2. 右クリックし、**「削除」**を選択します。

フィルタ後のデータまたは行数を表示する手順:

1. **フィルタ**を右クリックします。
2. **「データ」**または**「行数」**を選択します。

フィルタ後のデータまたは行数を含むウィンドウが表示されます。

3.3 ソース間の結合の定義

インタフェースのソース・データストア間の結合を作成する手順:

1. 結合する最初の表を表すエンティティの列を選択し、選択したまま結合する 2 番目の表のエンティティの列にドラッグ・アンド・ドロップします。
2. 2 つの表をリンクする**リレーション**がモデルに表示されます。**「実装」**フィールドに、2 つの列の関係を示す等式も表示されます。
3. **「実装」**を変更して必要なリレーションを作成します。モデルに含まれるすべての表の列を結合テキストにドラッグ・アンド・ドロップして、複数表の結合を作成できます。をクリックすることで式エディタを表示できます。
4. をクリックして式を検証し、 をクリックして保存します。
5. 実行場所として**「ソース」**または**「ステージング領域」**を選択します。
6. 結合のタイプ（右側または左側、内部または外部、ISO）と順序番号を選択します。
7. **「アクティブな句」**ボックスを選択して、インタフェースの実行でこの結合を使用するかどうかを定義します。

インタフェースのソース・データストア間の結合を削除する手順:

1. 削除する結合を表す**リレーション**を右クリックします。
2. **「削除」**を選択します。

結合後のデータまたは行数を表示する手順:

1. 結合を表す**リレーション**を右クリックします。
2. **「データ」**または**「行数」**を選択します。

結合後のデータまたは行数を含むウィンドウが表示されます。

4. マッピングの定義

マッピングでは、データの生成でターゲットをロードするため、ソースで変換ルールを定義します。

マッピングは、ソースまたはターゲットが新規に挿入されるたびに、列名の一致を使用して自動的に処理されます。ユーザー定義のマッピングは、常に名前の一致によるマッピングに優先します。

名前の一致による自動マッピングを再生成する手順:

1. **ターゲット・データストア**を右クリックします。
2. **「自動マッピングの再実行」**を選択します。

ターゲット・データストアの列が、最も類似する名前を使用してソース・データストアの列に自動的にマップされます。

ターゲット列のマッピングを定義する手順:

1. ターゲット・データストアの列をダブルクリックします。
2. 構成パネルで**「実装」**を変更して必要な変換を実行します。モデルに含まれるすべての表の列は、テキストにドラッグ・アンド・ドロップできます。をクリックすることで式エディタを表示できます。
3. をクリックして式を検証し、をクリックして保存します。
4. 実行場所（**「ソース」**、**「ターゲット」**または**「ステージング領域」**）を選択します。可能な実行場所の制限の詳細は、リファレンス・マニュアルの「マッピング」を参照してください。
5. マッピングを**「挿入」**や**「更新」**操作で、または KM 固有の操作（**「UD1」**から**「UD5」**）で実行する場合、**「更新」**セクションの各ボックスを選択します。
6. **「アクティブ・マッピング」**ボックスを選択し、インタフェースの実行でこのこのマッピングを使用するかどうかを定義します。

5. インタフェース・フローの定義

「フロー」タブで、マップされたデータのロードおよび統合計画を定義します。Oracle Data Integrator では、インタフェースの**ダイアグラム**のマッピング・ルール、結合およびフィルタに応じてソースセットのグループが自動的に計算されます。また、データ・フローのデフォルトの KM が推奨されます。「フロー」タブには、データのロードと統合に使用されるソースセットおよび KM が表示されます。フローの計画を定義する場合、使用中の KM を変更する必要があります。

使用中の KM を変更する手順:

1. インタフェースの「フロー」タブで、タイトルをクリックして**ソースセット**を選択します。
2. プロパティ・パネルで、設定する計画に応じて**「LKM」**または**「IKM」**を選択します。KM の選択方法の詳細は、KM に関するドキュメントを参照してください。
3. KM の**「オプション」**を設定します。
4. すべてのソースセットでこの操作を繰り返し、**「適用」**をクリックします。

注意: KM のリストに表示されるのは、使用中のテクノロジに関連しており、プロジェクトにインポートされている KM のみです。

注意: 一部の IKM では、更新キーを使用する必要があります。

注意: ある KM から別の KM に変更する場合、同じタイプおよび同じ名前のオプションは維持されます。

6. フロー制御の設定

フロー制御の計画により、ターゲットへの挿入前にフローのチェックに使用する方法を定義できます。制御計画は、CKM で定義します。

インタフェースで使用する CKM を定義する手順:

1. インタフェースの**「制御」**タブで、**「CKM」**を選択します。KM の選択方法の詳細は、KM に関するドキュメントを参照してください。
2. KM の**「オプション」**を設定します。

3. チェックする**「制約」**を選択します。
4. 一定のエラー数（または割合）に到達したときにインタフェースを終了する場合、**「許容されているエラーの最大数」**を入力し、**「%」**ボックスを選択します。
5. **「適用」**をクリックします。

注意: CKM のリストに表示されるのは、使用中のテクノロジに関連しており、プロジェクトにインポートされている CKM のみです。

注意: 一部の CKM では、更新キーを使用する必要があります。

注意: 制約のリストに表示されるのは、モデルのターゲット・データストアで宣言されている制約のみです。

警告: フロー制御は、IKM で `FLOW_CONTROL` オプション（**「フロー」**タブ）がアクティブ化されている場合にのみ実行されます。

7. インタフェースの実行

インタフェースを作成したら、そのインタフェースをテストできます。

インタフェースを実行する手順:

1. インタフェースの**「実行」**タブで、**「実行」**をクリックします。
2. インタフェースを保存していない場合、保存するかどうかを尋ねられたら**「はい」**をクリックします。
3. **実行オプション**を定義し、**「OK」**をクリックします。

プロシージャの作成

プロシージャの作成

プロシージャは、インタフェース・フレームワークに適応しないアクションをグループ化できる、再利用可能なコンポーネントです。（1 つ以上のソースからターゲット・データストアをロードします。）

プロシージャは、論理スキーマで起動される一連の**コマンド**です。プロシージャには、関連するオプションのグループがあります。これらの**オプション**により、コマンドやコマンドのコードを実行するかどうかをパラメータ化します。

プロシージャの作成は、次の手順で構成されます。

1. 新規プロシージャを作成します。
2. プロシージャのオプションを定義します。
3. プロシージャのコマンドを作成および管理します。
4. プロシージャを実行します。

1. 新規プロシージャの作成

新規プロシージャを作成する手順:

1. デザイナのツリー・ビューで、**プロシージャ**を作成する**フォルダ**の**「プロシージャ」**ノードをクリックします。
2. 右クリックし、**「プロシージャの挿入」**を選択します。
3. プロシージャの**「名前」**を入力します。
4. プロシージャで同時に複数の接続を管理する場合、**「複数接続」**ボックスを選択します。

複数接続: ある接続（ソース接続）に対するコマンドにより取得したデータを別の接続（ターゲット接続）に対するコマンドで使用する場合、複数接続プロシージャを選択すると便利です。このデータは、実行エージェントを経由します。一度に1つの接続にアクセスする場合（別の接続にもアクセスできますが、一度にアクセスできるのは1つのみです）、**「複数接続」**ボックスは選択しないままとします。

5. **「ターゲット・テクノロジ」**を選択します。**「複数接続」**ボックスを選択した場合は、**「ソース・テクノロジ」**も選択します。汎用のプロシージャを作成する場合、これらのフィールドは空白のままとします。
6. **「適用」**をクリックします。

新規プロシージャが作成され、ツリー・ビューのプロシージャのリストに表示されます。

2. プロシージャのオプションの定義

プロシージャのオプションを作成する手順:

1. デザイナのツリー・ビューで、**プロシージャ**のノードをクリックします。
2. 右クリックし、**「オプションの挿入」**を選択します。オプション・ウィンドウが表示されます。
3. オプションの**「名前」**、**「タイプ」**および**「デフォルト値」**を入力します。
4. **「OK」**をクリックします。
5. プロシージャに必要な**オプション**ごとにこれらの操作を繰り返します。

注意: 「チェックボックス」オプション（「はい」または「いいえ」）を使用する場合にのみ、コマンドを実行するかどうかをパラメータ化できます。他のオプション（**「値」**および**「テキスト」**）の値は、getOption()置換メソッドの使用により、プロシージャの**コマンド**のコードでのみ取得できます。

ツリー・ビューの**「プロシージャ」**ノードの下に**オプション**が表示されます。

3. プロシージャのコマンドの作成および管理

プロシージャのコマンドラインを作成する手順:

1. **プロシージャ**の**「詳細」**タブに移動します。
2.  をクリックします。**コマンド・**ウィンドウが表示されます。
3. 次のフィールドを入力します。
   - **名前:** コマンドの名前。

- **エラーの無視:** コマンドがエラーを戻してもプロシージャを停止しない場合に選択します。
- **ログ・カウンタ:** このコマンドで処理される行数を記録するカウンタ（「挿入」、「更新」、「削除」または「エラー」）を示します。
- **ログ・レベル:** コマンドの重要度のレベル。このレベルに従って実行ログを調査します。

4. **「ターゲットに対するコマンド」**で、次のフィールドを入力します。
   - **テクノロジ:** このコマンドで使用するテクノロジ。設定しない場合、プロシージャ・ウィンドウで指定されているテクノロジが使用されます。
   - **スキーマ:** コマンド実行用の論理スキーマ。
   - **コンテキスト:** 実行用の強制コンテキスト。未定義のままにすると、実行コンテキストが使用されます。
   - **トランザクション:** コマンドを実行するトランザクション。
   - **コミット:** トランザクションのコマンドのコミット・モードを示します。
   - **トランザクション分離:** コマンドのトランザクション分離レベル。
   - **コマンド:** 実行するコマンドのテキスト。  をクリックすることで式エディタを表示できます。

注意: テクノロジに対応する適切な言語のコマンドを使用できます。OS コマンドを使用するには、**「オペレーティング・システム・テクノロジ」**を使用する必要があります。Oracle Data Integrator ツールを使用するには、**「Oracle Data Integrator API」**テクノロジを使用する必要があります。

注意: 置換メソッドを使用してコードを汎用化し、トポロジ要素への依存性を高めることをお薦めします。

警告: **「トランザクション」**、**「コミット」**および**「トランザクション分離」**オプションは、トランザクションをサポートするテクノロジでのみ機能します。

5. **「ソースに対するコマンド」**で手順 4 を繰り返します。
6. **「オプション」**タブに移動します。
7. オプションの値にかかわらず常にこのコマンドを実行する場合、**「常に実行」**ボックスを選択します。それ以外の場合、このコマンドを実行する各**オプション**を選択します。
8. **「OK」**をクリックします。

コマンドを複製する手順:

1. **プロシージャ**の**「詳細」**タブに移動します。
2. 複製するコマンドを選択します。
3. 右クリックし、**「複製」**を選択します。
   コマンド・ウィンドウが表示されます。これには、選択したコマンドのコピーが含まれます。
4. 変更を加え、**「OK」**をクリックします。

新規コマンドが表示されます。

コマンドラインを削除する手順:

1. **プロシージャ**の**「詳細」**タブに移動します。
2. 削除する**コマンドライン**をクリックします。

3. をクリックします。

リストからコマンドラインが削除されます。

コマンドラインを並べ替える手順:

コマンドラインは、**プロシージャ**・ウィンドウの**「詳細」**タブに表示される順序で実行されます。これらの順序は、必要に応じて変更します。

1. **プロシージャ**の**「詳細」**タブに移動します。
2. 移動する**コマンドライン**をクリックします。
3. および をクリックしてコマンドラインを適切な位置に移動します。

4. プロシージャの実行

プロシージャの準備が完了したら、そのプロシージャをテストできます。

プロシージャを実行する手順:

1. **プロシージャ**の**「実行」**タブで、**「実行」**をクリックします。
2. プロシージャを保存していない場合、保存するかどうかを尋ねられたら**「はい」**をクリックします。
3. **実行オプション**を設定し、**「OK」**をクリックします。

KM またはプロシージャの暗号化

ナレッジ・モジュール（KM）またはプロシージャを暗号化すると、重要なコードを保護できます。暗号化された KM またはプロシージャは、復号化しなければ読取りまたは変更できません。暗号化された KM またはプロシージャによりログに生成されるコマンドも、読み取ることはできません。

Oracle Data Integrator では、**個人暗号化キー**に基づく DES 暗号化アルゴリズムが使用されます。このキーは、ファイルに保存して、暗号化または復号化操作を実行するために再利用できます。

警告: 暗号化された KM またはプロシージャを**暗号化キー**なしで復号化する方法はありません。したがって、このキーは安全な場所に保管しておくことを強くお薦めします。また、すべての開発作業でただ 1 つのキーを使用することをお薦めします。

KM またはプロシージャを暗号化する手順:

1. 暗号化する KM またはプロシージャを右クリックします。
2. **「暗号化」**を選択します。
3. **暗号化オプション**・ウィンドウで、次のいずれかの操作を実行します。
   - **「個人キーで暗号化します」**オプションを選択し、既存の暗号化キー・ファイルを選択します。
   - **「個人キーで暗号化します」**オプションを選択し、独自の**個人キー**に対応する文字列を入力するか貼り付けます。
   - **「新規暗号化キーを取得します」**オプションを使用して、Oracle Data Integrator でキーを生成します。
4. 暗号化が完了すると、**暗号化キー**・ウィンドウが表示されます。このウィンドウで、キーを保存できます。

注意: 文字数が少なすぎる個人キーを入力すると、**キー・サイズが無効**であるというエラーが発生します。この場合、より長い個人キーを入力してください。個人キーには、10 個以上の文字数が必要です。

関連項目:

KM またはプロシージャの復号化

KM またはプロシージャの復号化

KM またはプロシージャを復号化する手順:

1. 復号化する KM またはプロシージャを右クリックします。
2. **「復号化」**を選択します。
3. **KM 復号化**ウィンドウで、次のいずれかの操作を実行します。
   - 既存の暗号化キー・ファイルを選択します。
   - 独自の**個人キー**に対応する文字列を入力するか貼り付けます。

復号化が完了するとメッセージが表示されます。

関連項目:

KM またはプロシージャの暗号化

変数と順序の作成

変数の作成および使用

変数は、単一値を格納するオブジェクトです。この値には、文字列、数値または日付を使用できます。値は、Oracle Data Integrator に格納して、実行時に更新できます。

変数の値は、論理スキーマで実行される問合せの結果に基づいて更新できます。たとえば、値としてデータベースの現在の日時を取得できます。

変数は、**グローバル**変数として作成するか、または**プロジェクト**内で作成することが可能です。グローバル変数はすべてのプロジェクトで使用できますが、プロジェクト変数はそれを定義したプロジェクト内でのみ使用できます。

変数の作成

変数を作成する手順:

1. プロジェクトの**「変数」**ノードをクリックするか、**「その他」**ビューの**「グローバル変数」**ノードをクリックします。
2. 右クリックし、**「変数の挿入」**を選択します。
3. **「名前」**、**「アクション」**、**「データ型」**および**「デフォルト値」**を指定します。詳細は、「変数」を参照してください。
4. 変数の値を問合せで設定する場合は、次のようにします。
   1. **「リフレッシュ中」**タブを選択します。
   2. コマンドを実行する論理スキーマを選択し、スキーマのテクノロジの言語でコマンド・テキストを編集します。  をクリックすることで式エディタを使用できます。

3.  をクリックして式の構文をチェックします。
4. **「リフレッシュ」**ボタンをクリックすると、即座に問合せを実行して変数をテストできます。

5. **「OK」**をクリックします。

ツリー・ビューに変数が表示されます。

変数を削除する手順:

1. 適切なツリー・ビューの変数をクリックします。
2. 右クリックし、**「削除」**を選択します。
3. **「OK」**をクリックします。

変数の使用

パッケージで変数を使用する手順:

パッケージ内の変数は、次のように様々な方法で使用できます。

- **宣言:** 変数をパッケージで（またはパッケージ内で使用されるトポロジの特定の要素で）使用する場合、パッケージに**変数の宣言**ステップを挿入することを強くお薦めします。このステップで、パッケージの変数を明示的に宣言します。
- **リフレッシュ: 変数のリフレッシュ・**ステップでは、変数の値を計算するコマンドまたは問合せを再実行できます。
- **割当て: 割当て**タイプの**変数の設定**ステップでは、変数の現在の値を設定します。
- **増分: 増分**タイプの**変数の設定**ステップでは、一定の分量ごとに数値を増減します。
- **条件評価: 変数の評価**ステップでは、比較の結果に基づいて変数およびブランチの現在の値をテストします。
- 変数は、インタフェース、プロシージャ、OS コマンドなどのステップの式でも使用できます。

変数のスコープ

変数は、そのスコープを明示的に指定することで使用できます（この方法で使用する必要があります）。スコープを指定する構文は、`#GLOBAL.<variable name>`（**グローバル**変数の場合）または`#<project code>.<variable name>`（**プロジェクト**変数の場合）です。

変数の値の使用方法

変数は、次のような Oracle Data Integrator のすべての式で使用できます。

- マッピング
- フィルタ
- 結合
- 制約
- ...

式のテキスト内で変数の値を置換するには、変数名の前に#文字を追加します。エージェントまたはグラフィカル・インタフェースにより、実行前にコマンドの変数の値が置換されます。

次に、YEAR というグローバル変数の使用例を示します。

```
Update CLIENT set LASTDATE = sysdate where DATE_YEAR = '#GLOBAL.YEAR' /*
DATE_YEAR is CHAR type */
Update CLIENT set LASTDATE = sysdate where DATE_YEAR = #GLOBAL.YEAR /*
DATE_YEAR is NUMERIC type */
```

注意: SQL 言語のバインド変数メカニズムも使用できますが、あまり効率的ではありません。リレーショナル・データベース・エンジンでは、問合せの実行計画の作成時に変数の値を確認できないためです。このメカニズムを使用するには、変数の前にコロン（:）を追加します。検索対象のデータ型が変数のデータ型と互換性があることを確認してください。
例: `update CLIENT set LASTDATE = sysdate where DATE_YEAR =:GLOBAL.YEAR`

注意: 式エディタでは、ほとんどの式に変数をドラッグ・アンド・ドロップできます。トポロジのリソース名やスキーマ名などのグラフィカル・モジュールのフィールドで、変数を置換変数として使用することも可能です。この場合、変数名（`#GLOBAL.MYTABLENAME` など）を Oracle Data Integrator のグラフィカル・モジュールのフィールドで直接使用する必要があります。

この方法を使用すると、ファイルや表の物理名などの実行時要素（**データストア**の**「リソース」**フィールド）またはそれらの場所（トポロジの**「物理スキーマのスキーマ（データ）」**）をパラメータ化できます。

シナリオへの変数の受渡し

動作をカスタマイズするためにシナリオに変数を渡すこともできます。これを行うには、シナリオを実行する OS コマンドラインで変数の名前とその値を渡します。詳細は、「OS コマンドからのシナリオの実行」を参照してください。

順序の作成および使用

順序は、使用時に自動的に増加する値です。

Oracle Data Integrator の順序は、主に一部の RDBMS でネイティブの順序が存在しないことを補う目的で提供されています。値は、リポジトリに格納するか、外部 RDBMS 表のセル内で管理することができます。

順序は、**グローバル**順序として作成するか、または**プロジェクト**内で作成することが可能です。グローバル順序はすべてのプロジェクトに共通ですが、プロジェクト順序はそれを定義したプロジェクト内でのみ使用できます。

Oracle Data Integrator では、次の 2 つのタイプの順序がサポートされます。

- **標準順序:** 現在の値がリポジトリに格納されます。
- **特定順序:** 現在の値が RDBMS 表のセルに格納されます。Oracle Data Integrator は、値を読み取り、（同時更新の場合）行をロックして最後の増分以降に処理された行を更新します。

注意: Oracle Data Integrator は、順序が複数ユーザー管理で使用される場合には順序をロックしますが、順序の再開位置については処理しません。つまり、SQL 文の ROLLBACK では、トランザクションが開始した時点の値に順序を戻すことができません。

注意: Oracle Data Integrator の順序は、一部の RDBMS で順序が存在しないことを補う目的で開発されています。RDBMS の順序が存在する場合は、その順序を使用してください。これにより、エージェントとデータベース間の対話が減少するため、処理速度が向上する可能性があります。

順序の作成

標準順序を作成する手順:

1. プロジェクトの**「順序」**ノードをクリックするか、**「グローバル順序」**ノードをクリックします。
2. 右クリックし、**「順序の挿入」**を選択します。
3. 順序の**「名前」**を選択し、**「標準順序」**を選択します。
4. 最初の**「位置」**と**「増分」**を入力します。
5. **「OK」**をクリックします。

ツリー・ビューに順序が表示されます。

特定順序を作成する手順:

1. プロジェクトの**「順序」**ノードをクリックするか、**「グローバル順序」**ノードをクリックします。
2. 右クリックし、**「順序の挿入」**を選択します。
3. 順序の**「名前」**を選択し、**「特定の順序」**を選択します。
4. **「増分」**の値を入力します。
5. 順序の値を格納する**「スキーマ」**、**「表」**および**「列」**の名前を入力します。
6. Oracle Data Integrator で**表**の特定の行を検索できるように、**「フィルタ」**を入力します。  をクリックすることで式エディタを使用できます。  をクリックして式を検証します。
7. **「OK」**をクリックします。

ツリー・ビューに順序が表示されます。

注意: Oracle Data Integrator で特定順序の値にアクセスする場合、スキーマで実行される問合せは、`SELECT` ***列*** `FROM` ***表*** `WHERE` ***フィルタ***の形式となります。

順序を削除する手順:

1. ツリー・ビューで**順序**をクリックします。
2. 右クリックし、**「削除」**を選択します。
3. **「OK」**をクリックします。

ツリー・ビューから順序が削除されます。

順序および ID 列の使用方法

順序のスコープ

変数とは異なり、順序のスコープはコード内で明示的に宣言する必要はありません。

順序の使用方法

順序は、次のような Oracle Data Integrator のすべての式で使用できます。

- マッピング
- フィルタ
- 結合
- 制約

- ...

順序は、次の構文によりすべての文で使用できます。

- #<SEQUENCE_NAME>_NEXTVAL: 順序値が増分され、式のテキスト内で直接置換されます。

SQL 文の場合にかぎり、順序は次の構文によりバインディング・モードで使用できます。

- :<SEQUENCE_NAME>_NEXTVAL: 順序値が増分され、問合せのパラメータとして RDBMS に渡されます。

注意: 順序はエージェントにより解決されるため、順序が増分されるのは、エージェントそれ自体が 1 つのレコードを処理した場合のみです。
例:
- SQL 文の `insert into fac select :NO_FAC_NEXTVAL, date_fac, mnt_fac` では、SQL 文により 10,000 行が処理されたとしても、順序値が増分されるのは 1 回のみです。この場合、エージェントは 1 つ 1 つのレコードを処理せずに、データベース・エンジンにコマンドを送信するだけのためです。
- すべての行を増分するには、データをエージェント経由で送信する必要があります。これを行うには、Oracle Data Integrator に固有の SELECT/INSERT 構文を次のように使用します。

```
SELECT date_fac, mnt_fac /* on the source connection */
INSERT into FAC (ORDER_NO, ORDER_DAT, ORDER_AMNT) values
(:NO_FAC_NEXTVAL, :date_fac, :mnt_fac) /* on the target connection */
```

順序を使用する際のヒント

表に挿入された各行で順序が確実に更新されるには、各行がエージェントに処理される必要があります。この動作を保証するには、次の手順を実行します。

1. ターゲットで実行される順序を格納するマッピングを作成します。
2. 挿入に対してのみアクティブになるようマッピングを設定します。更新では順序はサポートされません。
3. 増分更新 IKM を使用する場合、順序の移入される列が使用中の更新キーに含まれていないことを確認します。たとえば、データストアの主キーをロードするために順序を使用する場合、インタフェースの更新キーとして代替キーを使用する必要があります。
4. Oracle Data Integrator の順序をバインド構文（:<SEQUENCE_NAME>_NEXTVAL）で使用する場合、IKM によりすべてのデータをエージェント経由で送信するようデータ・フローを構成する必要があります。これは、生成された統合ステップをオペレータでチェックすることで検証できます。異なる接続で、単一の SELECT...INSERT 文ではなく個別の INSERT コマンドと SELECT コマンドが実行される必要があります。

注意: 次の制限に注意してください。
- 順序とともにマップされた列では、NOT NULL をチェックできません。
- 同様に、順序を参照している主キーまたは代替キーでは、静的制御およびフロー制御を実行できません。

ID 列

Oracle Data Integrator の順序は、ネイティブの順序をエミュレートする解決策として存在しています。一部のデータベースでは、増分する一意の値が自動的に移入される ID 列をネイティブに提供しています。

ID 列を移入する場合、次の手順を実行する必要があります。

1. ID 列をロードするマッピングは、空白および非アクティブにする必要があります。挿入または更新に対してアクティブにしないでください。

2. 増分更新 IKM を使用する場合、ID 列が使用中の更新キーに含まれていないことを確認します。ID 列が主キーの一部である場合、インタフェースの更新キーとして代替キーを定義する必要があります。

注意: 次の制限に注意してください。
- ID 列では NOT NULL をチェックできません。
- ID 列を含む主キーまたは代替キーでは、静的制御およびフロー制御を実行できません。

ナレッジ・モジュールの処理

KM またはプロシージャの復号化

KM またはプロシージャを復号化する手順:

1. 復号化する KM またはプロシージャを右クリックします。
2. **「復号化」**を選択します。
3. **KM 復号化**ウィンドウで、次のいずれかの操作を実行します。
   - 既存の暗号化キー・ファイルを選択します。
   - 独自の**個人キー**に対応する文字列を入力するか貼り付けます。

復号化が完了するとメッセージが表示されます。

関連項目:

KM またはプロシージャの暗号化

KM またはプロシージャの暗号化

ナレッジ・モジュール（KM）またはプロシージャを暗号化すると、重要なコードを保護できます。暗号化された KM またはプロシージャは、復号化しなければ読取りまたは変更できません。暗号化された KM またはプロシージャによりログに生成されるコマンドも、読み取ることはできません。

Oracle Data Integrator では、**個人暗号化キー**に基づく DES 暗号化アルゴリズムが使用されます。このキーは、ファイルに保存して、暗号化または復号化操作を実行するために再利用できます。

警告: 暗号化された KM またはプロシージャを**暗号化キー**なしで復号化する方法はありません。したがって、このキーは安全な場所に保管しておくことを強くお薦めします。また、すべての開発作業でただ 1 つのキーを使用することをお薦めします。

KM またはプロシージャを暗号化する手順:

1. 暗号化する KM またはプロシージャを右クリックします。
2. **「暗号化」**を選択します。
3. **暗号化オプション**・ウィンドウで、次のいずれかの操作を実行します。
   - **「個人キーで暗号化します」**オプションを選択し、既存の暗号化キー・ファイルを選択します。
   - **「個人キーで暗号化します」**オプションを選択し、独自の**個人キー**に対応する文字列を入力するか貼り付けます。

- **「新規暗号化キーを取得します」**オプションを使用して、Oracle Data Integrator でキーを生成します。

4. 暗号化が完了**すると、暗号化キー**・ウィンドウが表示されます。このウィンドウで、キーを保存できます。

注意: 文字数が少なすぎる個人キーを入力すると、**キー・サイズが無効**であるというエラーが発生します。この場合、より長い個人キーを入力してください。個人キーには、10 個以上の文字数が必要です。

関連項目:

KM またはプロシージャの復号化

KM のインポート

Oracle Data Integrator にナレッジ・モジュールをインポートする手順:

1. KM をインポートする**プロジェクト**を選択します。
2. プロジェクトを右クリックし、**「インポート」→「ナレッジ・モジュールのインポート」**を選択します。
3. **「インポート・ディレクトリ」**を指定し、そのディレクトリからインポートする 1 つ以上の**「ナレッジ・モジュール・ファイル」**を選択します。
4. **インポート・モード**として**「複製」**を選択し、**「OK」**をクリックします。

ナレッジ・モジュールが作業リポジトリにインポートされ、プロジェクト内で使用できるようになります。

関連項目:

オブジェクトのエクスポート

インポート・モード

置換モードでの KM のインポート

置換モードでの KM のインポート

通常、**ナレッジ・モジュール**は、複製モードで新規**プロジェクト**にインポートされます。

プロジェクトの KM を別の KM に置換し、すべてのインタフェースで自動的にその新規 KM を使用する場合、インポート置換モードを使用する必要があります。

置換モードでナレッジ・モジュールをインポートする手順:

1. 置換する**ナレッジ・モジュール**を選択します。
2. **ナレッジ・モジュール**を右クリックし、**「置換のインポート」**を選択します。
3. **「ナレッジ・モジュール・エクスポート・ファイル」**を指定します。
4. **「OK」**をクリックします。

現在のナレッジ・モジュールが新規ナレッジ・モジュールで置換されます。

警告: ナレッジ・モジュールを別のナレッジ・モジュールで置換する場合、新規モジュールのオプションは、古いモジュールのオプション名との類似性に基づいて設定されます。新規オプションは、デフォルト値に設定されます。インタフェースでこれらのオプションの値を確認することをお薦めします。

警告: 本質的に異なる KM で現在の KM を置換すると、問題が発生する可能性があります。新規 KM を使用してインタフェースの設計と実行を確認することをお薦めします。

関連項目:

オブジェクトのエクスポート

インポート・モード

KM のインポート

Web サービスの使用方法

Web サービスは、次のように起動できます。

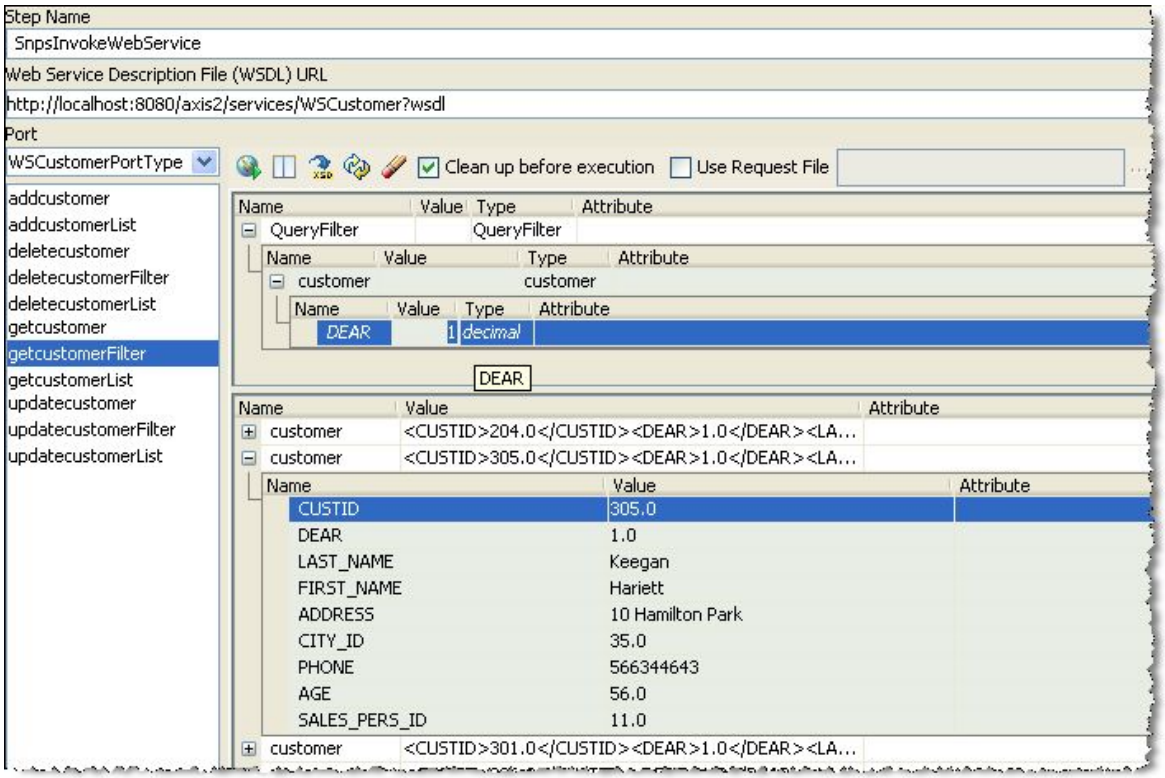
- データ・サービスをテストする目的で: この方法では、現在のデータ・サービスが正しく実行されているかどうかをチェックできます。詳細は、「データ・サービスの設定」を参照してください。
- SnpsInvokeWebService ツールを使用するパッケージで: このツールでは、サード・パーティの任意の Web サービスを起動できます。また、Oracle Data Integrator で処理できる XML ファイルにレスポンスを保存できます。

Web サービス・リクエストを作成するのに役立つ特別なグラフィカル・インタフェースがあります。

Web サービス用のグラフィカル・インタフェース

Web サービス用のグラフィカル・インタフェースは、次のように表示されます。

- 上部には、基本的なプロパティであるステップ名、WSDL の場所、およびポートが表示されます。
- 左下には、選択されたポートの操作リストが表示されます。2 番目のタブには、HTTP リクエストのオプションが表示されます。
- 右下の SOAP エディタには、Web サービス・リクエストおよびレスポンスが表示されます。



基本プロパティ

基本プロパティは次のとおりです。

- **ステップ名:** パッケージ・ステップの名前。
- **Web サービス記述ファイル(WSDL)の URL:** このファイルに、Web サービスと、リクエストおよびレスポンスの形式を記述します。
- **ポート:** WSDL で Web サービスにアクセスできる複数のポートを定義している場合、ここにポートのリストが表示されます。

「オプション」タブに、HTTP リクエストのオプションが表示されます。

- **タイムアウト:** Web サービス・リクエストは、応答を待機してこの時間が経過すると、サーバーからレスポンスが戻されないと判断し、エラーを生成します。
- **HTTP 認証:** このボックスを選択する場合、HTTP サーバーでの認証用にユーザーおよびパスワードを指定する必要があります。

ツールバー

ツールバーには、次の機能があります。

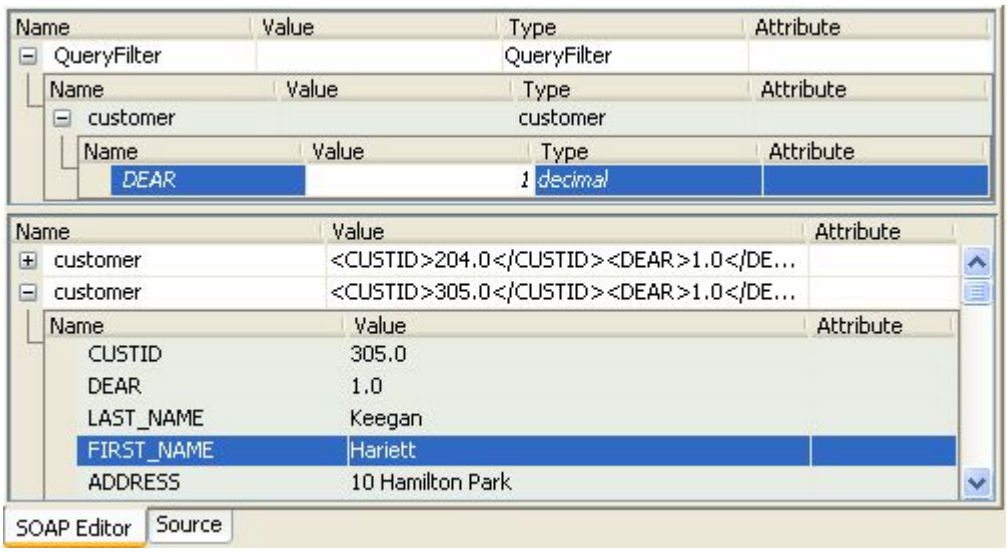
- **Web サービスの起動:** 現在の Web サービスを即座に起動し、SOAP エディタにレスポンスを表示します。
- **パネル位置の切替え:** SOAP エディタを縦または横に並べて表示します。
- **レスポンス XSD のエクスポート:** 現在のレスポンス XML スキーマ記述をファイルに保存します。

- **デフォルトのリクエストに戻す:** 現在のリクエストを破棄し、空のデフォルト・リクエスト構造に戻します。
- **空のオプション・コンポーネントの削除:** 問合せから空のオプション要素をすべて削除します。この操作は、有効な問合せを構成するために必要な場合があります。
- Clean up before execution: **「Web サービスの起動」**ボタンの使用時に、SOAP リクエストの空のオプション要素を自動的に削除します。このチェック・ボックスは、実行時のパッケージ・ステップには影響しません。
- Use Request File: SOAP エディタの内容ではなくファイルに格納された SOAP リクエストを使用します。
- Timeout (ms): リクエストの完了を待機する最大時間を指定します。

SOAP エディタ

SOAP エディタでは、Web サービスの SOAP リクエストをグラフィカルに作成し、レスポンスを表示できます。

SnpsInvokeWebService ステップを作成する場合、エディタで入力した SOAP リクエストはステップとともに保存されます。



エディタの上部には問合せの階層構造が表示され、下部にはレスポンスの構造が表示されます。この編成は、**「パネル位置の切替え」**ボタンを使用して変更できます。
SOAP リクエストおよびレスポンスの実際の XML ソースは、**「ソース」**タブに表示されます。

エディタでは、リクエストの各要素の**「値」**（およびオプションで**「属性」**）を入力できます。

> **警告:** 空の要素は、そのままの状態で Web サービスに渡されます。文字列の場合、これは空の文字列に対応します。数値型や日付型の場合、これはエラーの原因となる可能性があります。NULL の文字列、数値または日付を渡す場合、`nil="true"`属性を使用することをお薦めします。
> 空の要素を削除するには、**「空のオプション・コンポーネントの削除」**オプションを使用します。

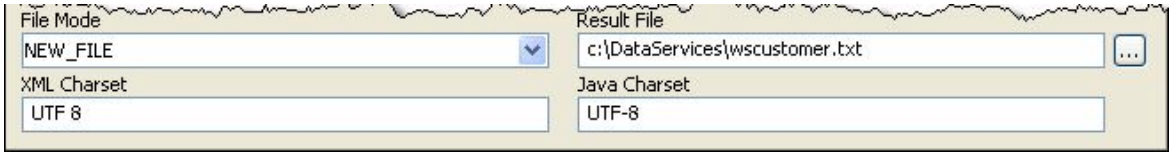
オプション要素は、イタリックで表示されます。繰返し可能要素は、名前の後に**「...(n*)」**というラベルが付けられます。

任意の要素を右クリックし、次のいずれかの操作を実行できます（実行可能な場合）。

- **コンテンツの複製:** 要素の構造と内容をコピーします。
- **構造の複製:** 構造をコピーしますが、すべてのフィールドを空白のまま残します。
- **削除:** 要素を削除します。
- **エクスポート・リクエスト:** SOAP リクエスト全体を XML ファイルにエクスポートします。

結果

インタフェースのこの部分は、パッケージで SnpsInvokeWebService ステップを使用する場合にのみ表示されます。これにより、レスポンスを XML ファイルに書き込む方法を制御します。



- **ファイル・モード**（`-RESPONSE_MODE`）**:** NEW_FILE、、FILE_APPEND、、NO_FILE のいずれか
- **結果ファイル**（`-RESPONSE_FILE`）**:** 書き込む結果ファイルの名前
- **XML キャラクタ・セット**（`-RESPONSE_XML_ENCODING`）**:** XML ファイルに書き込む文字コードの名前
- **Java キャラクタ・セット**（`-RESPONSE_FILE_CHARSET`）**:** ファイルの書込み時に使用する文字コードの名前

これらのパラメータの詳細は、「OdiInvokeWebService」を参照してください。

注意: 結果ファイルのパラメータが考慮されるのは、実行時のみです。**「Web サービスの起動」**ボタンの使用時には、結果ファイルは生成されません。

Web サービスの起動

Web サービス・リクエストを作成する手順:

1. パッケージで SnpsInvokeWebService ツール・ステップを作成するか、データストアを右クリックしてポップアップ・メニューの**「Web サービスのテスト」**を選択します。
2. WSDL の場所を入力します。次のいずれかを使用できます。
   - サーバーにデプロイされた WSDL の URL（`http://host:8080/axis2/services/WSCustomer?wsdl`）
   - ローカル・ファイルの場所（`c:/DataServices/WSCustomer.wsdl`）
3. 複数のポートが使用可能な場合、**「ポート」**を選択します。使用可能なポートがない場合、WSDL で問題が発生します。
4. 左側のリストから**「操作」**を選択します。
5. **SOAP エディタ**でリクエストを入力します。かわりに外部のリクエスト・ファイルを使用することもできます。

6. （オプション）**「空のオプション・コンポーネントの削除」**ボタンをクリックすると、入力されていないオプション要素を削除できます。一部の Web サービスでは、空の要素は無効とみなされます。
7. **「Web サービスの起動」**ボタンをクリックすると、即座に Web サービスを起動できます。レスポンスは、SOAP エディタに表示されます。
8. SnpsInvokeWebService ツール・ステップを作成する場合、結果ファイルのパラメータを定義し、**「適用」**をクリックしてステップを保存します。

Web サービス・レスポンスの処理

SnpsInvokeWebService を使用して Web サービスをコールする場合、レスポンスは XML ファイルに書き込まれます。

このファイルは、次のガイドラインに従って Oracle Data Integrator で処理できます。

- Web サービスを 1 回起動し、**「レスポンス XSD のエクスポート」**オプションを使用すると XML スキーマをエクスポートできます。
- この XML スキーマ・ファイルに基づいて SOAP レスポンスの XML モデルを作成し、XSD をリバース・エンジニアリングしてモデル構造を取得できます。詳細は、「XML ファイル・モデルの作成およびリバース・エンジニアリング」を参照してください。
- これで、標準の Oracle Data Integrator インタフェースを使用してレスポンスの情報を処理できます。詳細は、「XML ファイル用の適切な KM の選択」を参照してください。

注意: 各 XML ファイルは、Oracle Data Integrator のモデルとして定義されます。モデルを整理するためのモデル・フォルダを使用することをお薦めします。詳細は、「モデル・フォルダへのモデルの編成」を参照してください。

Oracle Data Quality の操作

Data Quality の概要

完全な Data Quality システムには、データのプロファイリング、整合性および品質が含まれます。

- **プロファイリング**により、データの調査および品質評価が可能になります。ビジネス・ユーザーは、データ品質の課題を明確に把握し、長期間にわたりデータ品質を監視および追跡できます。プロファイリングは、**Oracle Data Profiling** により処理されます。これを使用すると、ビジネス・ユーザーは、メトリックを介してデータの品質を評価し、このデータに基づいてルールを検出または推測し、長期にわたってデータ品質の変化を監視できます。
- **整合性**の制御は、使用する情報システムのアプリケーションに含まれるすべてのデータの一貫性を確保するために必要です。アプリケーション・データは、情報システムによって課せられる制約および宣言規則に対して有効でない場合があります。たとえば、顧客が指定されていない注文や商品が指定されていない注文明細行などが検出される可能性があります。**Oracle Data Integrator** には、これらの制約違反を検出し、リサイクルまたはレポート作成のために格納する作業環境が組み込まれています。Oracle Data Integrator では、静的チェックおよびフロー・チェックによって整合性のチェックが行われます。
- **品質**は、整合性にとどまらず、さらに複雑な品質プロセスにまで及びます。ルール・ベースのエンジンにより、統合プロセスの一環としてデータ品質基準が適用され、名前および住所を含むすべての種類のデータのクレンジング、標準化、強化、マッチングおよび重複除去が

行われます。**Oracle Data Quality for Data Integrator** によって、データ品質および名前と住所のクレンジングが企業のデータ統合戦略の中心となります。

品質プロセスの概要

1. **Oracle Data Integrator** で、クレンジングするデータが含まれる Quality 入力ファイルを作成します。
2. このファイルに基づいて、**Oracle Data Quality** にエンティティを作成します。
3. このエンティティをクレンジングする **Oracle Data Quality** プロジェクトを作成します。
4. このプロジェクトをランタイム用に **Oracle Data Quality** からエクスポートします。
5. **RKM Oracle Data Quality** を使用して、エンティティをリバース・エンジニアリングします。
6. インタフェースで Oracle Data Quality の入力ファイルおよび出力ファイルを使用します。
7. OdiDataQuality **ツール**を使用して、Oracle Data Integrator でこのプロジェクトを実行します。
8. パッケージ内のプロセスを順序付けします。

Quality 入力ファイルの作成

Oracle Data Quality では、クレンジングするデータが含まれるフラット・ファイルを Quality プロジェクトのソースとして使用します。この Quality 入力ファイルは、Data Integrator で作成し、インタフェースを使用して任意のソース・データストアからロードします。このファイルは、**「ファイル」**タブで次のパラメータが定義されているファイル・データストアです。

パラメータ	値
ファイル形式	区切り
ヘッダー(行数)	1
レコード・セパレータ	MS-DOS
フィールド・セパレータ	その他
[フィールド・セパレータ]その他	, （カンマ記号 - 16 進では 2C）
テキスト・デリミタ	"（二重引用符）
小数点セパレータ	空、指定せず

ファイル・データストアの作成の詳細は、「データストア」-「ファイル」を参照してください。フラット・ファイルのロードの詳細は、Oracle Data Integrator でのファイルの使用に関するトピックを参照してください。

エンティティの作成

Oracle Data Quality for Data Integrator へのデータ・ソースのインポートとは、デリミタ付きソース・ファイルに基づいてエンティティを作成することを意味します。

手順 1: ローダー接続の検証

管理者は、Oracle Data Quality for Data Integrator のインストール時に、少なくとも 1 つのローダー接続を設定する必要があります。ローダー接続を使用して、Oracle Data Quality 入力ファイルにアクセスします。入力ファイルがデリミタ付きファイルであるため、このローダー接続にはデリミタ付きローダー接続が必要です。手順 1 では、このデリミタ付きローダー接続の設定を検証する必要があります。また、必要なすべてのデータおよびスキーマ・ファイルが、ローダー接続により定義されたディレクトリにコピーされていることを確認してください。

メタベース・マネージャにアクセスできない場合は、メタベース管理者にローダー接続の確認を依頼してください。

メタベース・マネージャへのアクセス権を持つメタベース・ユーザーである場合は、次の手順に従います。

ローダー接続を検証する手順:

1. **メタベース・マネージャ**を開きます（**「スタート」→「プログラム」→「Oracle」→「Oracle Data Profiling and Quality」→「Metabase Manager」**）。
2. **管理モード**に設定されていることを確認します。
3. **Control Admin** ノードを展開します。
4. **「ローダー接続」**をダブルクリックします。
5. 右側の**ローダー接続のリスト・ビュー**に、各ローダー接続とその名前、タイプ、データ・ファイルおよびパラメータが表示されます。情報を参照して、管理者によって作成された**ローダー接続**が**デリミタ付きローダー接続**であること、データおよびスキーマのディレクトリが正しい場所を指していることを確認します。

注意: 完全なメタベース権限を持つメタベース・ユーザーの場合は、新規ローダー接続を作成できます。

手順 2: エンティティの作成およびデータのインポート

エンティティ作成ウィザードを使用して、エンティティを作成します。このウィザードを使用すると、各手順を実行し、ロードするデータを選択できます。インタフェースを使用して、接続およびスキーマの設定も指定できます。また、エンティティでのデータの表示をカスタマイズするオプションも指定できます。

デリミタ付きソース・ファイルを Oracle Data Quality for Data Integrator にインポートする手順:

1. **Oracle Data Quality for Data Integrator** にインポートするフラット・ファイルを、ローダー接続の定義時に指定したデータ・ディレクトリ内にコピーします。
2. Windows の**「スタート」**メニューをクリックし、**「プログラム」→「Oracle」→「Oracle Data Profiling and Quality」→「Oracle Data Profiling and Quality」**を選択します。
3. メタベース・ユーザーとしてユーザー・インタフェースにログインします。
   Oracle Data Profiling and Quality ユーザー・インタフェースが開きます。
4. **メイン・メニュー**で、**「分析」→「エンティティの作成...」**を選択します。
5. 右上の**ペイン**に、**エンティティ作成ウィザード**が開きます。

6. **エンティティ作成ウィザードの「接続」**ページで、手順 1 で確認した管理者から与えられたローダー接続を選択します。
7. フィルタおよび**接続**の設定をデフォルトのままにし、**「次へ」**をクリックします。
8. 手順 4 で選択したローダー接続を使用して、**Oracle Data Quality** がデータ・ソースに接続します。接続に失敗する場合は、メタベース管理者に連絡してください。
9. **「エンティティの選択」**ダイアログで、インポートするデータ・ソース・ファイル名をリストから選択し、**「次へ」**をクリックします。
10. 選択したデータ・ファイルのスキーマ設定を選択します。この設定は、「Quality 入力ファイルの作成」で説明したファイルのパラメータに対応します。
    - **デリミタ: ,**（カンマ）
    - **引用符: "**（二重引用符）
    - **属性情報:** 1 行目の名前
    - **「レコードの末尾に CR/LF を付ける」**を選択
    - **文字エンコーディング:** ascii

デリミタ付きファイルのエンティティの構成の詳細は、Oracle Data Quality for Data Integrator のヘルプを参照してください。

注意: Oracle Data Integrator を使用してファイルを生成する場合、これらのファイル形式のパラメータは、データストア定義の「ファイル」タブで指定したファイル形式と合致する必要があります。

11. スキーマの**設定**を選択したら、**「プレビュー」**をクリックします。**プレビュー・**モードでは、選択したスキーマ設定に基づいて、エンティティでデータがどのように表示されるかを示します。データは下部のリスト・ビュー内に表示されます。プレビュー・モードを使用して、新規エンティティでのデータの表示をカスタマイズします。
12. **「クローズ」**をクリックして続行します。
13. **「次へ」**を**クリック**します。**「パラメータのロード」**ダイアログが開きます。次のようにパラメータを指定します。
    - **「すべての行」**オプションを選択します。
    - **ジョブ名**はデフォルトのままにします。
14. **「次へ」**を**クリック**して続行します。
15. **「設定の確認」**で、設定のリストを確認し、**「終了」**をクリックしてエンティティの作成ジョブをスケジュールします。**ジョブのスケジュール・**ウィンドウが開きます。
16. **「今すぐ実行」**をクリックします。

手順 3: エンティティの確認

データのインポート処理中に、**Oracle Data Quality for Data Integrator** により、データ・ファイルは**エンティティ**、**属性**および**行**の 3 つの基本コンポーネント（メタベース・オブジェクト）に変換されます。

次に示す確認作業を実行して、必要なデータがメタベースに正常にインポートされ、メタベース・エクスプローラに適切に表示されることを確認します。

1. インポートしたデータ・ファイルごとに、1 つのエンティティが対応していることを確認します。
2. アンダースコア（_）またはマイナス記号（-）の文字を例外として、列名に特殊文字が含まれていないことを確認します。マイナス記号およびアンダースコアは、データのロード処理時に空白に変換されます。
3. インポートしたフィールドごとに、1 つの属性が対応していることを確認します。
4. インポートしたデータ行ごとに、1 つのエンティティ行が存在することを確認します。

プロファイリング・プロジェクトの作成

これで **Oracle Data Profiling** を使用して**データ・プロファイリング・プロジェクト**を実行する準備ができました。プロファイリングにより、エンタープライズ・データの品質を確認および分析します。最も詳細なレベルでデータを分析して、データの異常、フィルタおよびデータ・ルールの破損、データ・リレーションシップの不整合および経営目標を妨げる可能性があるその他のデータに関する懸念事項を識別します。

データ・プロファイリングの詳細は、Oracle Data Quality for Data Integrator ヘルプの Oracle Data Profiling の使用に関する章を参照してください。

Quality プロジェクトの作成

ここでは、**Oracle Data Quality プロジェクト**を作成して、データを検証および変換し、不一致や冗長性などのデータの問題を解決します。

Oracle Data Quality for Data Integrator は、各国固有のデータを含め、多数のビジネス・コンテキストおよびアプリケーション間でフィールド、値およびレコードの修復および修正を行う強力なツールです。**Oracle Data Quality for Data Integrator** では、データを標準化、クレンジングおよび向上する処理、カスタマイズのためのチューニングおよびリアルタイムでの結果の表示が可能です。

Quality プロジェクトは、入力ファイルをクレンジングし、出力ファイルにクレンジング後のデータをロードします。Oracle Data Quality プロジェクトの最後に、Data Quality プロジェクトに応じて、入力ファイルが複数の出力ファイルに分割されることがあります。

重要: Data Quality プロジェクトには多数の一時エンティティが含まれ、統合プロセスに有効でないエンティティもあります。フィルター・ベースのエンティティ名を使用することで、リバース・エンジニアリングの対象を Oracle Integrator で使用するエンティティのみに制限できます。このフィルタを効果的に使用するには、**Oracle Data Integrator** で使用するエンティティを一貫性のある方法により Quality プロジェクト内で名前変更することをお薦めします。たとえば、エンティティを `ODI_IN_XXX`、出力（および一致しない）ファイルを `ODI_OUT_XXX`（XXX はエンティティ名）に名前変更します。

Data Quality プロジェクトの詳細は、Oracle Data Quality for Data Integrator ヘルプの Oracle Data Quality の操作に関する章を参照してください。

Data Quality プロジェクトのエクスポート

Oracle Data Integrator では、Oracle Data Quality からエクスポートしたプロジェクトを実行できます。Data Quality プロジェクトの作成が完了したら、Oracle Data Integrator からエクスポートする必要があります。エクスポートするプロジェクトには、プロジェクトで選択した各プロセ

ス・モジュールのデータ・ファイル、データ・ディクショナリ言語（DDL）ファイル、設定ファイル、出力および統計ファイル、ユーザー定義の表およびスクリプトが含まれます。エクスポートしたプロジェクトは、ユーザー・インタフェースなしで UNIX または Windows プラットフォームで実行可能であり、必要となるのは Oracle Data Quality サーバーのみです。

バッチ・スクリプトを作成する手順:

1. **エクスプローラ**または**プロジェクト・ワークフロー**で、**Oracle Data Quality プロジェクト**を右クリックし、**「エクスポート」→「ODQ バッチ・プロジェクト」→「データなし」**を選択します。
2. **「フォルダの参照」**で、プロジェクトのエクスポート先のフォルダを選択します。
3. **「OK」**をクリックします。
   ファイルをコピー中であることを示すメッセージ・ウィンドウが表示されます。

   このエクスポート処理により、指定した場所にメタベース（<metabase_name>）の名前が付いたフォルダが作成されます。このフォルダには、projectN サブフォルダ（N は Oracle Data Quality でのプロジェクト識別子）が含まれます。このプロジェクト・フォルダには、様々なフォルダとともに、次のフォルダが含まれます。

   - **data:** このフォルダには、入出力データおよび一時データ・ファイルが含まれます。これらのファイルの拡張子は.DAT です。このエクスポートではデータなしを指定したため、このフォルダは空です。
   - **ddl:** このフォルダには、エンティティのメタデータ・ファイル（.DDX および.XML）が含まれます。これらのファイルには、データ・ファイルのフィールドが記述されます。ファイルには接頭辞 eNN_が付けられます（NN はエンティティ番号）。各エンティティは、2 つのメタデータ・ファイルに記述されます。eNN_<name of the entity>**.ddx** は、重複している可能性がある列を持つエンティティの記述です（固定ファイル用）。enNN_<name of the entity**_csv.ddx** は、重複していない列を持つエンティティの記述です（固定ファイルおよびデリミタ付きファイル用）。これらのファイルをリバース・エンジニアリング・プロセスで使用することをお薦めします。
   - **scripts:** このフォルダには、構成ファイル config.txt およびバッチ・スクリプト runprojectN が含まれます。このスクリプトにより、品質プロセスが実行されます。このスクリプトは、Oracle Data Integrator によってトリガーされます。
   - **settings:** このフォルダには、設定ファイルが含まれます（.ddt、.sto、.stt、.stx）。
4. メッセージ・ウィンドウが終了したら、指定したフォルダを調べ、すべてのフォルダおよびファイルが正しく作成されていることを確認します。
5. エクスポートしたプロジェクトをランタイム・マシン上のフォルダに移動します。このマシンには、Oracle Data Quality サーバーがインストールされている必要があります。これは Quality プロジェクトを実行するマシンです。
6. テキスト・**エディタ**で、projectN フォルダの/scripts サブフォルダ内にあるバッチ・スクリプト（runprojectN）および構成ファイル（config.txt）を開きます。
7. 次の変更を行ってプロジェクトのランタイム・ディレクトリを構成します。
   - config.txt で、DATABASE パラメータ用に、projectN フォルダを含むディレクトリの場所（絶対パス）を指定します。
   - runprojectN で、TS_PROJECT パラメータ用に、ProjectN ディレクトリの場所（絶対パス）を指定します。

たとえば、config.txt ファイルおよび runproject2.*ファイルの場所が C:/oracle/oracledq/metabase_data/metabase/oracledq/project2/scripts/である場合は、次のように指定します。

- config.txt:
  DATABASE= "C:/oracle/oracledq/metabase_data/metabase/oracledq"
- runprojectN.*:
  set
  TS_PROJECT=C:\oracle\oracledq\metabase_data\metabase\oracledq\project2

8. 変更を保存し、config.txt ファイルを閉じます。
9. **runprojectN** で、ファイルの最終行のコメントを削除します（最終行の先頭にある::文字を削除）。
10. 変更を保存し、runprojectN ファイルを閉じます。
11. Oracle Data Integrator では、CSV 形式のファイル（通常、カンマ区切りで 1 行目がヘッダー行）を使用して Data Quality プロジェクトに入力データを提供し、出力データも同じ形式であることを予定しています。
    /settings ディレクトリで、エディタを使用して、プロジェクトの最初のプロセスに対応する設定ファイルを開きます。最初のプロセスが変換である場合、通常、このファイルの名前は eN_transfmr_p1.stx です（N は Quality 入力ファイルに対応するエンティティの内部 ID）。
12. 設定**ファイル**の入力パラメータを次のように変更します。
    - DATA_FILE_NAME に、実行時の Quality 入力ファイルの名前および場所（絶対パス）を指定します。
    - FILE_DELIMITER に、Quality 入力ファイルで使用するデリミタを指定します。
    - START_RECORD に、データが開始する行番号を指定します。たとえば、1 行目がヘッダー行である場合、この値は 2 です。

    たとえば、Quality 入力ファイル customer_master.csv（カンマ区切り、1 行目はヘッダー行）が C:/oracle/oracledq/metabase_data/metabase/oracledq/Data/にある場合、次のセクションを編集する必要があります。

```
<CATEGORY><INPUT><PARAMETER><INPUT_SETTINGS>
          <ARGUMENTS>
          <ENTRY>
              <ENTRY_ID>1</ENTRY_ID>
              <FILE_QUALIFIER>Customer_Master(1)</FILE_QUALIFIER>
              <DATA_FILE_NAME>$(INPUT)/e1_customer_master.dat</DATA
_FILE_NAME>
              <DDL_FILE_NAME>$(DDL)/e1_customer_master.ddx</DDL_FIL
E_NAME>
              <FILE_DELIMITER/>
              <USE_QUOTES_AS_QUALIFIER/>
              <START_RECORD/>
```

    編集後のセクションは、次のとおりです。

```
<CATEGORY><INPUT><PARAMETER><INPUT_SETTINGS>
```

```
                <ENTRY>
                  <ENTRY_ID>1</ENTRY_ID>
                  <FILE_QUALIFIER>Customer_Master(1)</FILE_QUALIFIER>
                  <DATA_FILE_NAME>C:\oracle\oracledq\metabase_data\meta
    base\oracledq\Data\customer_master.csv</DATA_FILE_NAME>
                  <DDL_FILE_NAME>$(DDL)/e1_customer_master.ddx</DDL_FIL
    E_NAME>
                  <FILE_DELIMITER>,</FILE_DELIMITER>
                  <USE_QUOTES_AS_QUALIFIER/>
                  <START_RECORD>2</START_RECORD>
```

13. 変更を保存して設定ファイルを閉じます。
14. /settings ディレクトリで、出力（クレンジング済）データを生成するプロセスの設定に対応するファイルを開きます。通常、データ・リコンストラクタ・プロセスで終了するクレンジング・プロジェクトの場合、名前に eNN_datarec_pXX.stx が付きます。設定ファイル内で次の値を変更して、生成される出力ファイルのフルパスを指定します。

```
<CATEGORY><OUTPUT><PARAMETER>
        <OUTPUT_SETTINGS>
          <ARGUMENTS>
            <FILE_QUALIFIER>OUTPUT</FILE_QUALIFIER>
            <DATA_FILE_NAME>C:\oracle\oracledq\metabase_data\metabase\or
acledq\Data\customer_master_cleansed.csv</DATA_FILE_NAME>
            <DDL_FILE_NAME>$(DDL)/e36_us_datarec_p11.ddx</DDL_FILE_NAME>
```

15. 変更を保存して設定ファイルを閉じます。
16. 有用な出力ファイルを生成するデータ品質プロセスがある場合（たとえば、1 国につき 1 つのデータ・リコンストラクタ）、プロセスごとに前述の 2 つの手順を繰り返します。

エンティティのリバース・エンジニアリング

品質プロセスに入力データを提供し、その出力データを Data Integrator の統合プロセスで使用するには、これらのエンティティのリバース・エンジニアリングが必要です。この操作は、Oracle Data Quality RKM に基づいてカスタマイズされたリバース・エンジニアリング・メソッドを使用して実行します。RKM は、Data Quality プロジェクトの/ddl フォルダにある.ddx ファイルからメタデータを読み取ります。

Data Quality プロジェクトのエンティティのリバース・エンジニアリングを実行する手順:

1. **RKM Oracle Data Quality** を Oracle Data Integrator プロジェクトにインポートします。
2. トポロジ・マネージャで、ファイル・テクノロジの物理スキーマを挿入します。ディレクトリ（**スキーマ**）およびディレクトリ（**作業スキーマ**）の両方に、データ・フォルダの絶対パスを指定します。たとえば、C:\oracle\oracledq\metabase_data\metabase\oracledq\projectN\data のようにします。

   このディレクトリには、変換を実行するために使用されるエージェントからアクセスできる必要があります。**Oracle Data Integrator** は、スキーマでインタフェースのソースおよびターゲット・データ構造を検索します。RKM は出力データ・ファイルにアクセスし、それをリバース・エンジニアリングします。

3. ファイル・モデルを作成し、/ddl フォルダをリバース・エンジニアリングします。
   1. **デザイナ**に接続します。
   2. ツリーで**「モデル」**を選択します。
   3. 右クリックし、**「モデルの挿入」**を選択します。
   4. **「定義」**タブで、**「名前」**フィールドを入力します。
   5. **「テクノロジ」**フィールドで、**「ファイル」**を選択します。
   6. **「論理スキーマ」**フィールドで、モデルの基礎となる**論理スキーマ**を選択します。
   7. **「リバース」**タブに移動し、次を選択します。
      1. **カスタマイズしたリバース**
      2. リバースエンジニアリング・コンテキスト
      3. **リバースエンジニアリングするオブジェクトの型:** 表
      4. **KM:** RKM Oracle Data Quality を選択
   8. RKM のオプションを設定します。

パラメータ	デフォルト値	説明
DDX_FILE_NAME	`*.ddx`	処理する DDX ファイルのマスク。Quality プロジェクトで使用するエンティティに対してネーミング規則を使用している場合、これらのエンティティのみが返されるようにマスクを入力します。たとえば、入力および出力エンティティに対して `ODI_IN_XX` および `ODI_OUT_XX` ネーミング規則を使用した場合、`ODI*_csv.ddx` マスクを指定します。
FILE_FORMAT	`D`	生成されるファイル・データストアに使用する形式。 • **D（区切り）:** 1 行にある複数のフィールドが、レコード・セパレータで分割されています。 • **F（固定）:** 1 行にある複数のフィールドが、区切られず、長さが固定されています。
RECORD_SEPARATOR	`MS-DOS`	ファイル内で行と行（またはレコードとレコード）を区切っている 1 つ以上の文字。 • **MS-DOS:** DOS の改行記号。 • **UNIX:** UNIX の改行記号。 他のすべての値は、空のレコード・セパレータに対応します。

FIELD_SEPARATOR	,（カンマ）	レコード内のフィールドを区切るために使用する文字。
USE_FRIENDLY_NAMES	いいえ	リバース・エンジニアリング・プロセスで、DDX ファイルに指定されたフィールド名に基づいてデータストアの列に対してユーザーにわかりやすい名前を生成する場合は、はいに設定します。
USE_LOG	はい	リバース・エンジニアリング・プロセスのアクティビティをログ・ファイルに記録する場合は「はい」に設定します。
LOG_FILE_NAME	`/temp/reverse.log`	ログ・ファイルの名前。

4. **「適用」**をクリックします。これでモデルは作成されましたが、データストアはまだ格納されていません。
5. **「リバース」**をクリックします。これで「モデル」表示に表示されるデータストアがモデルに格納されます。

インタフェースでの Oracle Data Quality の入力ファイルおよび出力ファイルの使用

Oracle Data Integrator で、Data Quality の入力および出力ファイルをソースまたはターゲットとして使用するインタフェースを作成できます。

たとえば、次のことができます。

- 様々なソースからデータストアを使用して入力ファイルをロードするインタフェースの作成。
- クレンジング後に出力データを元のソースに再統合するインタフェースの作成。

Data Integrator からの Quality プロジェクトの実行

OdiDataQuality ツールは、バッチ・ファイルを実行して Oracle Data Quality プロジェクトを実行します。このツールは、`runprojectN` スクリプト・ファイルのパスをパラメータとして受け取ります。同期モード（ツールが品質プロセスの完了を待機）または非同期モードで実行できます。

OdiDataQuality ツールおよびパラメータの詳細は、OdiDataQuality のトピックを参照してください。

パッケージ内のプロセスの順序付け

次のプロセスを順序付けして、Oracle Data Integrator でパッケージを作成します。

1. 1 つ以上のインタフェースによる、クレンジングするデータを含む Quality 入力ファイルの作成。
2. **OdiDataQuality** ツール・ステップによる、Oracle Data Quality プロセスの起動。

3. 1 つ以上のインタフェースによる、Oracle Data Quality 出力ファイルからターゲット・データソースへのデータのロード。

ユーザー関数の使用方法

ユーザー関数を使用すると、インタフェースまたはプロシージャで使用できるカスタマイズされた関数を定義できます。
これらの関数は、1 つ以上のテクノロジに実装されます。

例:

Oracle の nvl(**VARIABLE,DEFAULT_VALUE**) 関数は、値 **VARIABLE** または **DEFAULT_VALUE**（**VARIABLE** が NULL の場合）を戻します。これと同等の関数を持たないテクノロジも存在するため、次の式で置換する必要があります。

```
case when VARIABLE is null
          then DEFAULT_VALUE
     else VARIABLE
end
```

ユーザー関数では、NullValue(**VARIABLE,DEFAULT_VALUE**) という関数を宣言し、前述の構文に対応する 2 つの実装を定義できます。実行時に、命令が実行されるテクノロジに応じて、NullValue 関数は一方または他方の構文で置換されます。

関数は、**グローバル**関数として作成するか、または**プロジェクト**内で作成することが可能です。グローバル関数はすべてのプロジェクトに共通ですが、プロジェクト関数はそれが定義されているプロジェクトにのみ関連付けられます。

ユーザー関数の作成

ユーザー関数を作成する手順:

1. 単一**プロジェクト**または複数**プロジェクト**の**「ユーザー関数」**ノードをクリックします。
2. 右クリックし、**「ユーザー関数の挿入」**を選択します。
3. 次のフィールドを入力します。
   - **名前:** ユーザー関数の名前。たとえば、NullValue と指定します。
   - **グループ:** ユーザー関数のグループ。存在しないグループ名を入力すると、関数の保存時にそのグループ名で新規グループが作成されます。
   - **構文:** 式エディタに表示されるユーザー関数の構文。関数の引数をこの構文で指定する必要があります。たとえば、NullValue($(variable), $(default)) と指定します。
4. **「OK」**をクリックします。

ツリー・ビューに関数が表示されます。実装が含まれないため、まだ使用できません。

実装を作成する手順:

1. **ユーザー関数**の**「実装」**タブで、**「追加」**をクリックします。
2. **「実装構文」**に実装のコードを入力します。たとえば、nvl($(variable), $(default)) と指定します。
3. 実装の**「関連付けられているテクノロジ」**の各ボックスを選択します。

4. 新規**テクノロジ**でこの構文を使用する場合、**「新しいテクノロジを自動的に含む」**ボックスを選択します。
5. **「OK」**をクリックします。

実装を変更する手順:

1. **ユーザー関数**の**「実装」**タブで、**実装**を選択して**「編集」**をクリックします。
2. この実装の**「実装構文」**と**「関連付けられているテクノロジ」**を変更します。
3. 新規テクノロジでこの構文を使用する場合、**「新しいテクノロジを自動的に含む」**ボックスを選択します。
4. **「OK」**をクリックします。

実装を削除する手順:

1. **ユーザー関数**の**「実装」**タブで、**実装**を選択して**「削除」**をクリックします。

ユーザー関数の使用方法

ユーザー関数は、次のような Oracle Data Integrator のすべての式で使用できます。

- マッピング
- フィルタ
- 結合
- 制約
- ...

ユーザー関数は、その**構文**を指定することで直接使用できます。たとえば、`NullValue(CITY_NAME, 'No City')`と指定します。

マーカーおよびメモの使用方法

プロジェクトとモデルのほぼすべての要素には、プロジェクトの概要説明や開発支援を目的として、説明的なマーカーとメモを関連付けることができます。

マーカー

フラグは、マーカーを使用して定義します。これらのマーカーは、グループに編成し、プロジェクトまたはモデル内のほとんどのオブジェクトに適用できます。

一般的なマーカー・グループは、次のとおりです。

- 開発サイクル（開発、テスト、本番）
- 優先度（低、中、高、緊急）
- 進捗状況（10%、20%など）

グローバル・マーカーとプロジェクト・マーカー

マーカーは、プロジェクトまたは**「その他」**ビュー（グローバル・マーカー）で定義します。プロジェクト・マーカーは、そのプロジェクトのオブジェクトでのみ使用できます。グローバル・マーカーは、リポジトリのすべてのモデルで使用できます。

オブジェクトへのフラグ付け

オブジェクトにアイコン・マーカーでフラグを付ける手順:

1. **「プロジェクト」**または**「モデル」**ツリー・ビューでオブジェクトを選択します。
2. 右クリックして**「マーカーの追加」**を選択し、マーカー・グループと設定するマーカーを指定します。

マーカー・アイコンがツリー・ビューに表示されます。マークされたオブジェクトは、マーカーのノードの下にも表示されます。そのため、特定のマーカーが付けられたすべてのオブジェクトを参照できます。

ツリー・ビューで自動増分マーカー・グループに属するアイコン・マーカーをクリックすると、マーカーがマーカー・グループ内の次のマーカーに切り替わり、アイコンもそれに従って変化します。

注意: **「マーカーおよびメモ・フラグの表示」**オプションが選択されていない場合、マーカーは表示されません。詳細は、後述の説明を参照してください。

オブジェクトに文字列、数値および日付マーカーでフラグを付ける手順:

1. **「プロジェクト」**または**「モデル」**ツリー・ビューでオブジェクトをダブルクリックします。
2. オブジェクト編集ウィンドウで、**「マーカー」**タブを選択します。
3. **「マーカーの挿入」**ボタンをクリックします。
4. 新規行で、**「グループ」**および**「マーカー」**を選択します。必要に応じて**「値」**も設定できます。

マーカーに関連付けられたアイコンがある場合、そのアイコンがツリー・ビューに表示されます。

マーカーを使用したフィルタ

マーカーは、情報目的で使用できます（プロジェクトの進捗状況およびリソースのグローバル・ビューを提供するなど）。また、マーカーは、パッケージをフィルタしてシナリオ生成を自動化する場合にも使用できます。詳細は、「シナリオ・グループの生成」を参照してください。

特定のマーカーを使用しているすべてのオブジェクトのリストは、マーカーのノードの下に表示されます。

マーカーのカスタマイズ

新規プロジェクトは、デフォルト・マーカー付きで作成されます。特定のプロジェクトのマーカーやグローバル・マーカーは、カスタマイズできます。

マーカー・グループを定義する手順:

1. デザイナのツリー・ビューで、**「プロジェクト」**の**「マーカー」**ノードをクリックするか、**「その他」**ビューの**「グローバル・マーカー」**ノードをクリックします。
2. 右クリックし、**「マーカー・グループの挿入」**を選択します。
3. **「グループ名」**を入力し、その**「表示プロパティ」**と**「属性」**を定義します。
4. **「マーカーの挿入」**ボタンをクリックし、グループに新規マーカーを作成します。
5. マーカーの**「アイコン」**を選択します。マーカーに日付または数値が格納される場合、アイコンは<「なし」>に設定します。

6. マーカーの**「名前」**、**「タイプ」**、およびその他のオプションを選択します。
7. 手順 4 から 6 を繰り返し、グループに別のマーカーを追加します。
8. **「OK」**をクリックします。

メモ

メモは、事実上すべてのオブジェクトに添付できる文字数に制限のないテキストで、**「メモ」**タブに表示されます。オブジェクトにメモが添付されると、オブジェクトの横に  アイコンが表示されます。

オブジェクトのメモを編集する手順:

1. オブジェクトを右クリックします。
2. **「メモの編集」**を選択します。
3. **「メモ」**タブが選択された状態でオブジェクトが表示されます。

マーカーおよびメモの非表示化

画面を見やすくするため、ツリー・ビューに含まれるすべてのマーカーとメモのフラグを一時的に非表示にできます。

すべてのマーカーおよびメモのフラグを非表示にする手順:

1. **「表示」**メニューの**「マーカーおよびメモ・フラグの表示」**オプションを選択解除します。この設定は、コンピュータごとに保存されます。

相互参照の使用方法

Oracle Data Integrator のオブジェクト（データストア、モデル、インタフェースなど）は、単純な使用上の関連（統合インタフェースによるナレッジ・モジュールの使用）からコード解釈関係などの複雑な関連（インタフェースのマッピングまたはフィルタでの変数の使用）に及ぶ様々な関係性により、相互にリンクされています。これらの関係は、相互参照として実装されます。相互参照は、作業リポジトリ内の関連オブジェクト間の整合性をチェックおよび維持するために使用されます。たとえば、相互参照により、作業リポジトリのいずれかの場所で現在参照されているオブジェクトは、削除されないよう保護されます。

すべての関係が相互参照として表示されるわけではありません。

- マスター・リポジトリのオブジェクトとの関係（テクノロジとデータ・モデルの関連など）は、相互参照として実装されますが、オブジェクト・コード（コンテキスト・コード、テクノロジ・コード、データ型コードなど）に基づく弱い関係として実装されます。これらのコードが変更されると、参照内の整合性が失われる可能性があります。
- 作業リポジトリの強い関係（プロジェクトに属するフォルダ）は、グラフィカル・ユーザー・インタフェースおよびリポジトリ内で（ホスト・データベースの外部キーとして）強制的に適用されます。これらの関係は、通常、破損することはありません。

相互参照の表示

オブジェクトを変更する場合、その変更による他の開発作業への影響を分析する必要があります。たとえば、列の長さを変更する場合、状況によっては、その列をソースまたはターゲットとして

使用している統合インタフェースも変更する必要があります。相互参照により、特定のオブジェクトの参照元または参照先オブジェクトを即座に識別できるため、効果的な影響分析を実行できます。

相互参照を表示する手順:

相互参照は、デザイナのツリー・ビューの次の場所で確認できます。

- **「プロジェクト」**および**「その他」**ツリー・ビューでは、オブジェクト・ノードの下に、現在のオブジェクトを参照しているオブジェクトのリスト付きで**「使用中」**ノードが表示されます。たとえば、変数の場合、その変数を参照しているステップを含むパッケージと、マッピングやフィルタなどでその変数を使用しているインタフェースが表示されます。
- **「モデル」**ツリー・ビューでは、オブジェクト・ノードの下に**「使用中」**ノードが表示され、現在のデータストア、モデルまたはサブモデルをインタフェースのソースまたはターゲットとして（あるいはパッケージ・ステップで）参照しているオブジェクトがリストされます。**「移入に使用」**および**「移入者」**ノードには、現在のデータストアを移入元または移入先とするデータストアが表示されます。

これらの相互参照ノードは、開くことができます。参照元または参照先のオブジェクトは、相互参照ノードから表示または編集できます。

欠落参照の解決

バージョン・リストア操作を実行すると、作業リポジトリのオブジェクトが、存在しないオブジェクトを参照するという状況が発生する可能性があります。この状況は、一般的に、統合インタフェースで使用されていたすべての関連モデルをリストアせずに、古いバージョンのプロジェクトをリストアした場合に発生します。

このような状況で、存在しないオブジェクトを参照するオブジェクト（インタフェースなど）を開くと、Oracle Data Integrator で**欠落参照**のエラー・メッセージが表示されます。相互参照が欠落しているオブジェクトは、ツリー・ビューで欠落参照マーカー によりマークされます。その親オブジェクトは、警告アイコン によりマークされます。

オブジェクトの欠落参照の詳細を表示する手順:

1. 欠落参照マーカーのあるオブジェクトを編集します。編集ウィンドウで、**「参照がありません」**タブを選択します。
2. 相互参照が欠落している参照オブジェクトのリストは、このタブに表示されます。

欠落参照を解決する手順:

欠落相互参照は、次の 2 つの方法で解決できます。

- 欠落参照オブジェクトをインポートまたはリストアします。詳細は、「バージョン管理の使用方法」および「インポート/エクスポート」を参照してください。
- 欠落オブジェクトに対する参照を削除するため、参照元オブジェクトを変更します（たとえば、存在しない変数を参照する変数のリフレッシュ・ステップをパッケージから削除し、それを別の変数で置換します）。

重要: テキスト（インタフェースのマッピング、結合、フィルタや、プロシージャ・コマンドなど）に欠落参照が含まれる場合、このテキストに適用される最初の変更が、追加チェックなしでこのテキストの欠落相互参照に対する修正とみなされます。

バージョン管理の使用方法

Oracle Data Integrator では、変更を管理および保護するための包括的なシステムを備えたプロジェクト・マネージャを提供しています。バージョン管理システムでは、プロジェクトやモデルなどの開発済オブジェクトに**フラグ**を自動的に設定し、新規または変更済などのステータスを表示できます。また、これらのオブジェクトを基準となるチェックポイントとしてバックアップし、後で各チェックポイントからリストアを実行できます。これらのチェックポイントは、個々のオブジェクトには**バージョン**の形式で作成され、オブジェクトの一貫したグループには**ソリューション**の形式で作成されます。

重要: Hypersonic SQL、IBM DB2 UDB、IBM DB2/400、Informix、Microsoft SQL Server、Oracle、Sybase AS Anywhere、Sybase AS Enterprise の各データベース・エンジンにインストールされたマスター・リポジトリでは、バージョン管理がサポートされます。

オブジェクト・フラグの操作

デザイナでオブジェクトが作成または変更されると、ツリー・ビューのオブジェクト・アイコンにそのステータスを示すフラグが表示されます。

- : オブジェクト・ステータスは**挿入**です。
- : オブジェクト・ステータスは**更新**です。

オブジェクトが挿入、更新または削除されると、その親オブジェクトにも再帰的に更新のフラグが付けられます。たとえば、パッケージにステップが挿入されると、ステップに**挿入**のフラグが付けられ、このステップを含むパッケージ、フォルダおよびプロジェクトにも**更新**のフラグが付けられます。

オブジェクト・バージョンがチェックインされると（詳細はバージョンの説明を参照）、そのオブジェクトのフラグはリセットされます。

バージョンの操作

バージョンは、オブジェクトのバックアップ・コピーです。バージョンは、任意の時点でチェックインし、後からリストアできます。各バージョンの保存先は、マスター・リポジトリです。バージョンは、オブジェクト・ウィンドウの**「バージョン」**タブに表示されます。

プロジェクトの次のオブジェクトは、バージョンとしてチェックインできます。

- プロジェクト、フォルダ
- パッケージ、シナリオ
- インタフェース、プロシージャ、ナレッジ・モジュール
- 順序、ユーザー関数、変数
- モデル

バージョンをチェックインする手順:

1. バージョンをチェックインするオブジェクトを選択します。
2. 右クリックし、**「バージョン」→「作成」**を選択します。
3. ウィンドウが表示されます。このウィンドウで、**「前のバージョン」**（>>）ボタンを使用してチェックイン済のバージョンのリストを開きます。

4. 「**バージョン**」フィールドにバージョン番号が自動的に生成されます。必要に応じてこのバージョン番号を変更します。
5. このバージョンの詳細を「**説明**」フィールドに入力します。
6. 「**OK**」をクリックします。

バージョンをチェックインすると、オブジェクトのフラグがリセットされます。

オブジェクトの以前のバージョンを表示する手順:

- オブジェクトの編集時に、チェックイン済のバージョンのリストが「**バージョン**」タブに表示されます。このとき、チェックイン日付とチェックイン操作を実行したユーザーの名前も表示されます。

バージョンをリストアする手順:

1. バージョンをリストアするオブジェクトを選択します。
2. 右クリックし、「**バージョン**」→「**リストア**」を選択します。
3. 既存のバージョンのリストを含むウィンドウが表示されます。
4. リストアするバージョンを選択し、「**OK**」をクリックします。
5. 「**OK**」をクリックしてリストア操作を確認します。

警告: バージョンのリストア操作は、元に戻すことができません。現在のオブジェクトは永久に削除され、選択したバージョンで置換されます。

バージョンを参照する手順:

Oracle Data Integrator には、リポジトリに格納されたバージョンを表示できる**バージョン・ブラウザ**というツールが付属しています。

1. デザイナのメニューで、「**ファイル**」→「**バージョン・ブラウザ**」を選択します。
2. オブジェクト・タイプとオブジェクト名のドロップダウン・ボックスを使用して、バージョンのリストを表示するオブジェクトをフィルタします。

バージョン・ブラウザでは、既存のバージョンの**リストア**、XML ファイルへの**エクスポート**、または**削除**が可能です。

注意: バージョン・ブラウザには、ブラウザを起動した時点で存在していたバージョンが表示されます。起動時以降に作成されたすべての新規バージョンを表示するには、「リフレッシュ」ボタンをクリックします。

バージョン・ブラウザでバージョンを削除する手順:

1. バージョン・ブラウザを起動します。
2. 削除するバージョンを選択します。
3. 右クリックし、「**削除**」を選択します。

バージョンが削除されます。

バージョン・ブラウザでバージョンをリストアする手順:

1. バージョン・ブラウザを起動します。
2. リストアするバージョンを選択します。
3. 右クリックし、「**リストア**」を選択します。
4. 「**OK**」をクリックしてリストア操作を確認します。

リポジトリにバージョンがリストアされます。

バージョン・ブラウザでバージョンをエクスポートする手順:

この操作では、リストアせずにファイルにバージョンをエクスポートします。このエクスポートは、別のリポジトリにインポートできます。

1. バージョン・ブラウザを起動します。
2. エクスポートするバージョンを選択します。
3. 右クリックし、**「エクスポート」**を選択します。
4. **「エクスポート・ディレクトリ」**を選択し、**「エクスポート名」**を指定します。確認なしで既存のエクスポート・ファイルを削除する場合、**「既存のファイルを警告なしで置換します」**チェック・ボックスを選択します。
5. **「OK」**をクリックします。

指定した場所にバージョンがエクスポートされます。

ソリューションの操作

ソリューションは、相互に依存するオブジェクトのバージョンの包括的な整合セットです。他のオブジェクトと同様に、ソリューションは任意の時点でバージョンとしてチェックインし、後でリストアできます。ソリューションは、マスター・リポジトリに保存されます。ソリューションには、ソリューションの**要素**と呼ばれるバージョンのグループが集約されます。

ソリューションは、相互参照を使用して自動的に集約されます。相互参照のスキャンにより、特定のオブジェクトに必要なすべての依存オブジェクトがソリューションに自動的に含められます。たとえば、ソリューションにプロジェクトを追加すると、そのプロジェクトのインタフェースで使用されているすべてのモデルのバージョンが、自動的にチェックインされてソリューションに追加されます。ソリューションを対象に要素を手動で追加または削除することも可能です。

ソリューションは、デザイナの**「ソリューション」**ツリー・ビューに表示されます。

ソリューションには次のオブジェクトを追加できます。

- プロジェクト
- モデル
- シナリオ
- グローバル変数、ユーザー関数および順序

ソリューションを作成する手順:

1. **「ソリューション」**ツリー・ビューを開きます。
2. 右クリックし、**「ソリューションの挿入」**を選択します。**新規ソリューション・**ウィンドウが表示されます。
3. ソリューションの**「名前」**と**「説明」**を入力します。
4. **「適用」**をクリックします。

この操作により、要素を追加できる空のシェルとしてソリューションが表示されます。

ソリューションで要素を操作する手順:

- 要素を追加するには、ツリー・ビューのオブジェクトをソリューション・ウィンドウの**「要素」**リストにドラッグします。
  Oracle Data Integrator により相互参照がスキャンされ、この要素が正しく動作するために必

要な**必須要素**がすべて追加されます。追加されたオブジェクトが、最後のチェックイン・バージョン以降に挿入または更新されている場合、それらのオブジェクトの新規バージョンを作成するよう求められます。

- ソリューションから要素を削除するには、**「要素」**リストで削除する要素を選択し、**「削除」**ボタンをクリックします。要素がリストから削除されます。オブジェクトの既存のチェックイン・バージョンは、影響を受けません。
- ソリューションに保存されているバージョンにオブジェクトをロールバックするには、リストアする要素を選択し、**「リストア」**ボタンをクリックします。選択した要素がソリューションのバージョンからすべてリストアされます。
- **「同期化」**オプションをクリックすると、自動的に必須要素が追加され、不要な要素が削除されてソリューションが最新の状態に更新されます。

ソリューションを同期する手順:

ソリューションを同期すると、ソリューションに含まれていない必須要素の追加、変更された要素の新規バージョンの作成、および不要な要素の削除が自動的に行われます。同期プロセスにより、ソリューションの内容は、リポジトリに保存された要素（プロジェクトやモデルなど）に基づいて最新の状態に更新されます。

1. 同期するソリューションを開きます。
2. **「同期化」**ボタンをクリックします。
3. Oracle Data Integrator により、相互参照がスキャンされます。相互参照によりソリューションが最新であることがわかると、メッセージが表示されます。それ以外の場合、ソリューションで追加または削除対象となる要素のリストが表示されます。
   これらの要素は、**「プリンシパル要素」**（手動で追加される要素）、**「必須要素」**（主要要素により直接的または間接的に参照される要素）および**「未使用の要素」**（主要要素により参照されなくなった要素）に分類されます。
4. **「同意」**ボックスを選択し、必須要素をバージョン化して含めるか、不要な要素を削除します。
5. **「OK」**をクリックしてソリューションを同期します。新規バージョンを作成する必要のある要素に、バージョン作成ウィンドウが表示される場合があります。

ソリューションの内容を最新の状態に維持するため、この操作は定期的に実行する必要があります。また、ソリューション・バージョンをチェックインする前にもこの操作を実行する必要があります。

ソリューションをチェックインおよびリストアする手順:

- ソリューション・バージョンをチェックインおよびリストアする手順は、単一の要素に使用される方法と同様です。詳細は、バージョンの説明を参照してください。

重要: ソリューションをリストアする場合、ソリューション内の要素は自動的にリストアされません。各要素は、**ソリューション**・ウィンドウから手動でリストアする必要があります。

オペレータでのバージョン

オペレータでソリューションを使用して、本番環境にシナリオをインポートできます。

ソリューションからシナリオをリストアする手順:

1. **「ソリューション」**タブに移動します。
2. ソリューションをダブルクリックして開きます。

3. シナリオを選択します。プロジェクトやインタフェースなどの他の要素は、リストアできません。
4. **「リストア」**ボタンをクリックします。

これで、**「シナリオ」**タブでシナリオにアクセスできます。

シナリオのリストアには、**バージョン・ブラウザ**も使用できます。

同時変更の処理

複数のユーザーが同じ Oracle Data Integrator プロジェクトまたはモデルを同時に操作できます。各ユーザーはすべて同じリポジトリに接続されるため、ユーザーが加える変更は同時に発生するとみなされます。

Oracle Data Integrator では、これらの同時変更を処理するために、同時編集チェックおよびオブジェクト・ロックという 2 つの方法を提供しています。この 2 つの方法は、同時に、または個別に使用できます。

同時編集チェック

ユーザー・パラメータの**「同時編集のチェック」**を true に設定すると、ユーザーは、自分が保存しようとしているオブジェクトに対して他のユーザーが作業を実行した場合に、その作業を削除できなくなります。詳細は、「ユーザー・パラメータ」を参照してください。

このパラメータが true に設定されている場合、オブジェクトに対する変更をユーザーが保存しようとすると、そのユーザーがファイルを開いて以降、他のユーザーが同じオブジェクトに変更を追加していないかどうかが Oracle Data Integrator によってチェックされます。他のユーザーが変更を追加している場合、オブジェクトは保存できないため、変更を保存しようとしたユーザーはその変更を取り消す必要があります。

オブジェクト・ロック

自動オブジェクト・ロック

このメカニズムは、自動的にアクティブ化されます。オブジェクトをユーザー・インタフェースで開くと、そのオブジェクトをロックするかどうかを尋ねるポップアップ・ウィンドウが表示されます。オブジェクトのロック中は、ロックを所有するユーザーのみがオブジェクトを変更できます（編集や削除など）。他のユーザーは、実行などのその他の操作を行うことができますが、その際警告が表示されます。

ユーザーにロックされたオブジェクトは、黄色のロック・アイコン付きで表示されます。別のオブジェクトにロックされたオブジェクトは、赤色のロック・アイコン付きで表示されます。

編集ウィンドウを閉じると、オブジェクトのロックを解除するかどうかを尋ねるポップアップ・ウィンドウが表示されます。

これらのウィンドウは、**「開く時にオブジェクトをロック」**および**「閉じる時にオブジェクトをロック解除」**ユーザー・パラメータで構成します。詳細は、「ユーザー・パラメータ」を参照してください。

ユーザー・インタフェースを閉じる場合のロックの解放

Oracle Data Integrator を閉じる場合、ロックしているオブジェクトのロックを解除するか、または開いているオブジェクトを保存するかどうかを尋ねるウィンドウが表示されます。

Oracle Data Integrator に接続していない場合でも、オブジェクトのロックを維持できます。その間、他のユーザーはこれらのオブジェクトを編集できません。

ロックの手動管理

オブジェクトのロックは手動で管理することもできます。

オブジェクトを手動でロックする手順:

1. ツリー・ビューでオブジェクトを選択します。
2. 右クリックし、**「ロック」→「ロック」**を選択します。

ロック・アイコンがツリー・ビューのオブジェクトの横に表示されます。

オブジェクトを手動でロック解除する手順:

1. ツリー・ビューでオブジェクトを選択します。
2. 右クリックし、**「ロック」→「ロック解除」**を選択します。

ロック・アイコンがツリー・ビューのオブジェクトの横に表示されます。

すべてのロックを管理する手順:

1. **「ファイル」→「ロックされたオブジェクト」**を選択します。

ロックを解除できるすべてのロック済オブジェクトが新規ウィンドウに表示されます。このリストを使用して、オブジェクトのロックとロック解除を行うことができます。

重要: **「スーパーバイザ」**権限を持つユーザーは、他のすべてのユーザーのロックを削除できます。

実行

デザイナまたはオペレータからのシナリオの実行

この実行モードでは、**デザイナ**または**オペレータ**からシナリオを起動します。

デザイナまたはオペレータからシナリオを起動する手順:

1. **「プロジェクト」**ビュー（デザイナの場合）または**「シナリオ」**ビュー（オペレータの場合）で必要なシナリオを選択します。
2. 右クリックし、**「実行」**を選択します。
3. **「コンテキスト」**と必要な**「エージェント」**を選択します。ローカル・マシンでシナリオを実行するには、**「ローカル（エージェントなし）」**オプションを選択します。
4. **「OK」**をクリックします。
5. シナリオでパラメータとして変数を使用する場合、割り当てる値を選択します。変数の「最後の値」を選択すると、現在の値が使用されます。使用可能な値がない場合は、デフォルト値が使用されます。

6. Oracle Data Integrator により新規セッションが作成および開始されると、**「セッションを開始しました」**というメッセージが表示されます。
7. セッションの実行状況は、**オペレータ**で監視できます。

OS コマンドからのシナリオの実行

シナリオは、オペレーティング・システムのコマンドを使用して起動できます。

OS コマンドからシナリオを起動する手順:

1. **シェル**（UNIX）、**コマンド・プロンプト**（Windows）または **QSH** セッション（AS/400）を起動します。
2. このシェルで、startscen.bat（Windows）または startscen.sh（UNIX および AS/400）コマンドを実行します。

構文: `startscen <Name> <Version> <Context code> [<Log_Level>] [-SESSION_NAME=<session name>] [-KEYWORDS=<keywords>] [-NAME=<agent_name>] [-v=<trace level>] [<variable>=<value>]*`

警告: Windows プラットフォームでは、等号（=）または空白を含むコマンド引数は、二重引用符で囲む必要があります。Windows のコマンド・コールは、UNIX のコマンド・コールとは異なる場合があります。次に例を示します。

```
startscen.bat SCEN 001 GLOBAL "-v=5" "PROJ1.STATE=MICHIGAN"
```
（Windows）

```
./startscen.sh SCEN 001 GLOBAL -v=5 PROJ1.STATE=MICHIGAN
```
（UNIX）

次の表に、必須およびオプションの様々なパラメータをリストします。パラメータの前にはハイフン（-）を付け、可能な値の前には等号（=）を付けます。この場合、コマンドを入力するオペレーティング・システムに固有の文字保護の構文に準拠する必要があります。

パラメータ	説明
`<Name>`	シナリオの名前（必須）。
`<Version>`	シナリオのバージョン（必須）。バージョン指定を`-1` とすると、シナリオの最後のバージョンが実行されます。
`<Context>`	シナリオの実行コンテキスト（必須）。
`-v=<trace level>`	表示を許可するトレース（冗長モード）。次の 5 つのトレース・レベルがあります。 1. 各セッションの開始と終了を表示 2. レベル 1 に加え、各ステップの開始と終了を表示 3. レベル 2 に加え、実行された各タスクを表示 4. レベル 3 に加え、実行された SQL 問合せを表示 5. 完全トレース（通常、サポート連絡時に要求されるレベル） 一部のトレースは大量となる可能性があるため、次のコマンドを使用してその内容をテキスト・ファイルにリダイレ

	クトすることをお薦めします。 Windows の場合: "-v=5" > trace.txt UNIX の場合: -v=5 > trace.txt
<Log_Level>	シナリオ実行のロギング・レベルを定義します。この値以下のロギング・レベルが設定されたすべてのタスクは、実行の最後にログに保存されます。インタフェースが異常終了した場合は、この値にかかわらずすべてのタスクが保存されます。 このパラメータの形式は、LEVEL<n>です（<n>は 0 から 5 までの適切なロギング・レベル）。デフォルトのロギング・レベルは 5 です。 例: startscen.bat SCENAR 1 GLOBAL LEVEL5
-SESSION_NAME=<session_name>	実行ログに表示されるセッションの名前。
-KEYWORDS=<keywords>	このセッションに関連するキーワードのリスト。これらのキーワードにより、セッションの識別が容易になります。このリストは、各キーワードのカンマ区切りリストです。
-NAME=<agent_name>	このセッションの実行ログに表示されるエージェント名。 **注意:** このエージェント名は、トポロジの論理エージェントまたは物理エージェントに対応しません。このパラメータでは、リモート・エージェントでセッションを実行できません。この機能は、SnpsStartScen API により提供されます。
<variable>=<value>	シナリオの実行で<value>を<variable>に割り当てることができます。<variable>は、プロジェクト変数またはグローバル変数です。プロジェクト変数の名前は、<Project Code>.<Variable Name>とする必要があります。グローバル変数の名前は、GLOBAL.<variable Name>とする必要があります。 このパラメータは、複数の変数を割り当てるために繰り返すことができます。 **注意:** startscen コマンドラインでは、変数名の前にハッシュ記号（#）を使用しないでください。
作業リポジトリ接続パラメータ	**これらのパラメータは、作業リポジトリへの接続を指定するために使用し、odiparams ファイルに指定します。**
-SECU_DRIVER=<driver name>	マスター・リポジトリにアクセスするための JDBC ドライバ。 例:oracle.jdbc.driver.OracleDriver

-SECU_URL=<url>	マスター・リポジトリにアクセスするための JDBC URL。 例:jdbc:oracle:thin:@168.67.0.100:1522:ORCL
-SECU_USER=<user>	マスター・リポジトリに接続するためのユーザー（データベース・ユーザー）。
-SECU_PASS=<password>	マスター・リポジトリに接続するユーザーのパスワード。このパスワードは、コマンド agent ENCODE <password>を使用して暗号化する必要があります。
-ODI_USER=<user>	シナリオを実行する権限を持つ Oracle Data Integrator ユーザー。たとえば、このユーザーがコンテキストに対する権限を持っていない場合、実行に失敗します。デフォルト値は SUPERVISOR です。
-ODI_PASS=<password>	Oracle Data Integrator ユーザーのパスワード。このパスワードは、コマンド agent ENCODE <password>を使用して暗号化する必要があります。
- WORK_REPOSITORY=<work repository name>	実行対象のシナリオを含む作業リポジトリの名前。

Web サービスを使用したシナリオの実行またはセッションの再開

Web サービスを使用したシナリオの実行

Oracle Data Integrator の公開 Web サービスを使用すると、Web サービスからシナリオを実行できます。

実行の前に、使用している Web サービス・サーバーに Oracle Data Integrator の公開 Web サービスをインストールする必要があります。

Web サービスを使用してシナリオを実行するには、OdiInvoke Web サービスを起動します。使用するポートは、invokeScenario と呼ばれます。
この Web サービスにより、エージェントで適切な作業リポジトリに接続し、特定のシナリオを起動します。パラメータは、OS コマンドからシナリオを実行する場合のパラメータと同様です。

次に、この Web サービスに対する単純な SOAP リクエストを示します。パラメータの詳細なリストは、WSDL ファイルを参照してください。

```
<invokeScenarioRequest>
     <invokeScenarioRequest>
          <RepositoryConnection>
               <!-- Connection information to the repository -->
               <!-- This example is an Oracle Repository -->
               <JdbcDriver>oracle.jdbc.driver.OracleDriver</JdbcDriver>
               <JdbcUrl>jdbc:oracle:thin:@srv011:1521:ORA10G</JdbcUrl>
               <JdbcUser>repo</JdbcUser>
               <JdbcPassword>snp65934</JdbcPassword>
               <OdiUser>SUPERVISOR</OdiUser>
```

```
            <OdiPassword>PASS</OdiPassword>
            <WorkRepository>WORKREP</WorkRepository>
        </RepositoryConnection>
        <Command>
            <!-- Scenario, version and execution context -->
            <ScenName>LOAD_DW</ScenName>
            <ScenVersion>001</ScenVersion>
            <Context>GLOBAL</Context>
            <SyncMode>1</SyncMode>
        </Command>
        <Agent>
            <!-- Agent that will execute the scenario -->
            <Host>srv001</Host>
            <Port>20910</Port>
        </Agent>
    </invokeScenarioRequest>
</invokeScenarioRequest>
```

警告: Web サービスでは、パスワードをプレーン・テキストの形式で取得します。そのため、保護されていないネットワーク上で Web サービスを起動する場合は、セキュアなプロトコル（HTTPS）を使用することを強くお薦めします。

注意: Web サービス・コンテナは、エージェントにアクセス可能である必要があります。また、エージェントはリポジトリにアクセス可能である必要があります。

注意: エージェントがすでに作業リポジトリに接続している場合、RepositoryConnection 構造全体は必要ありません。たとえば、スケジューラ・エージェントの場合、エージェントにすべてのリポジトリ接続情報が含まれています。この場合、ODIUser および ODIPassword サブ要素のみが必要です。

シナリオ実行により、次のような SOAP レスポンスが戻されます。

```
<odi:invokeScenarioResponse xmlns:odi="xmlns.oracle.com/odi/OdiInvoke">
    <odi:CommandResultType>
        <odi:Ok>true</odi:Ok>
        <odi:SessionNumber>1148001</odi:SessionNumber>
    </odi:CommandResultType>
</odi:invokeScenarioResponse>
```

レスポンスは、次のようにシナリオ実行に使用している SyncMode に応じて異なります。

- **同期**モード（SyncMode=1）では、セッションが完了するとレスポンスが戻され、レスポンスには実行結果が反映されます。
- **非同期**モード（SyncMode=2）では、セッションが開始するとレスポンスが戻され、レスポンスにはセッションが正しく開始されたという事実のみが反映されます。

Web サービスを使用したセッションの再開

Oracle Data Integrator の公開 Web サービスを使用すると、Web サービスからセッションを再開できます。

実行の前に、使用している Web サービス・サーバーに Oracle Data Integrator の公開 Web サービスをインストールする必要があります。

Web サービスを使用してセッションを再開するには、OdiInvoke Web サービスを起動します。使用するポートは、invokeSession と呼ばれます。
この Web サービスにより、エージェントで適切な作業リポジトリに接続し、特定のセッションを再開します。

次に、この Web サービスに対する単純な SOAP リクエストを示します。パラメータの詳細なリストは、WSDL ファイルを参照してください。

```
<invokeSessionRequest>
     <invokeSessionRequest>
          <RepositoryConnection>
               <!-- Connection information to the repository -->
               <!-- This example is an Oracle Repository -->
               <JdbcDriver>oracle.jdbc.driver.OracleDriver</JdbcDriver>
               <JdbcUrl>jdbc:oracle:thin:@srv011:1521:ORA10G</JdbcUrl>
               <JdbcUser>repo</JdbcUser>
               <JdbcPassword>snp65934</JdbcPassword>
               <OdiUser>SUPERVISOR</OdiUser>
               <OdiPassword>PASS</OdiPassword>
               <WorkRepository>WORKREP</WorkRepository>
          </RepositoryConnection>
          <Command>
               <!-- Session Number -->
               <SessionNumber>3001</SessionNumber>
               <SyncMode>1</SyncMode>
          </Command>
          <Agent>
               <!-- Agent that will execute the session -->
               <Host>srv001</Host>
               <Port>20910</Port>
          </Agent>
     </invokeSessionRequest>
</invokeSessionRequest>
```

警告: Web サービスでは、パスワードをプレーン・テキストの形式で取得します。そのため、保護されていないネットワーク上で Web サービスを起動する場合は、セキュアなプロトコル（HTTPS）を使用することを強くお薦めします。

注意: Web サービス・コンテナは、エージェントにアクセス可能である必要があります。また、エージェントはリポジトリにアクセス可能である必要があります。

注意: エージェントがすでに作業リポジトリに接続している場合、RepositoryConnection 構造全体は必要ありません。たとえば、スケジューラ・エージェントの場合、エージェントにすべてのリポジトリ接続情報が含まれています。この場合、ODIUser および ODIPassword サブ要素のみが必要です。

セッションの実行から、次のような SOAP レスポンスが戻されます。

```
<odi:invokeSessionResponse xmlns:odi="xmlns.oracle.com/odi/OdiInvoke">
     <odi:CommandResultType>
          <odi:Ok>true</odi:Ok>
          <odi:SessionNumber>3001</odi:SessionNumber>
     </odi:CommandResultType>
</odi:invokeSessionResponse>
```

レスポンスは、次のようにセッションの実行に使用している SyncMode に応じて異なります。

- **同期**モード（SyncMode=1）では、セッションが完了するとレスポンスが戻され、レスポンスには実行結果が反映されます。
- **非同期**モード（SyncMode=2）では、セッションが開始するとレスポンスが戻され、レスポンスにはセッションが正しく開始されたという事実のみが反映されます。

HTTP URL からのシナリオの実行

メタデータ・ナビゲータ・モジュールでは、Web ページまたは HTTP URL からシナリオを起動できます。

注意: この方法でシナリオを実行するには、最初にメタデータ・ナビゲータをインストールしておく必要があります。

動作の仕組み

メタデータ・ナビゲータには、**HTTP POST** リクエストに応答する **StartScen サーブレット**が付属します。このサーブレットの機能は、OdiStartScen ツールと同じです。

- StartScen サーブレットをコールするには、メタデータ・ナビゲータ・サーバーの /snpsrepexp/startscen.do リソースに対する HTTP POST リクエストを使用します。リクエストとともに適切な HTTP パラメータを渡す必要があります。
- サーブレットからは、HTTP レスポンスが XML、HTML またはプレーン・テキスト形式で戻されます。

StartScen サーブレットのパラメータ

パラメータ	説明
agent_name	エージェントが稼働するマシンのネットワーク名または IP アドレス。
agent_port	エージェントがリスニングするポート。

master_driver	マスター・リポジトリにアクセスするための JDBC ドライバ。 例:oracle.jdbc.driver.OracleDriver
master_url	マスター・リポジトリにアクセスするための JDBC URL。 例:jdbc:oracle:thin:@168.67.0.100:1522:ORCL
master_user	マスター・リポジトリに接続するためのデータベース・ユーザー。
master_psw	データベース・ユーザーのパスワード。このパスワードは、コマンド agent ENCODE <password>を使用して暗号化する必要があります。
work_repository	実行対象のシナリオを格納する作業リポジトリの名前。
snps_user	シナリオを実行する権限を持つ Oracle Data Integrator ユーザーの名前。このユーザーがコンテキストに対する権限を持っていない場合、シナリオは実行できません。
snps_psw	Oracle Data Integrator ユーザー名のパスワード。このパスワードは、コマンド agent ENCODE <password>を使用して暗号化する必要があります。
scen_name	シナリオの名前。
scen_version	シナリオのバージョン番号。バージョン指定を-1 とすると、シナリオの最新のバージョンが実行されます。
context_code	シナリオを起動する実行コンテキスト。
log_level	シナリオ実行のロギング・レベルを定義します。この値以下のロギング・レベルが設定されたすべてのタスクは、実行の最後にログに保存されます。ただし、シナリオが異常終了した場合は、この設定にかかわらずすべてのタスクが保存されます。 このパラメータの範囲は、整数の 0 から 5 までです。
http_reply	HTTP レスポンスのタイプ。このレスポンスは、セッションが開始されたかどうかを示します。可能なレスポンス・タイプは、XML、HTML およびテキストです。デフォルトは XML です。 このパラメータの有効な値は、XML | HTML | TXT です。 XML、HTML またはテキスト形式でのレスポンスに加え、StartScen サーブレットからのすべての HTTP レスポンスは、HTTP ヘッダーの Cookie として戻されます。

シナリオのパラメータ値

変数パラメータを含むシナリオでは、HTTP パラメータとしてパラメータ値を指定する必要があります。

たとえば、VAR1 という変数がある場合、HTTP リクエスト・パラメータの VAR1=<VAR1_VALUE>を定義する必要があります。

サーブレット・レスポンス

サーブレット・レスポンスは、セッションが正しく開始されたかどうかを示します。

> **警告:** セッションが正しく開始されたという通知は、セッションそれ自体が成功したかどうかとは無関係です。

レスポンスには、次の 3 つの要素が含まれます。

要素	説明
snps_exe_ok	セッションが開始されたかどうかを示すブール値（true | false）
snps_session_no	リポジトリのセッションに対応する一意の ID 番号
snps_error_msg	シナリオが正しく起動されなかった場合（snps_exe_ok = false）のエラー・メッセージ

サンプル・レスポンス

HTTP ヘッダーCookie

ヘッダーには、snps_exe_ok、snps_session_no および snps_error_msg という 3 つの Cookie が含まれます。

XML

```
<snps_scen_result>
     <snps_exe_ok>true|false</snps_exec_ok>
     <snps_session_no>session number</snps_session_no>
     <snps_error_msg>error message</snps_error_msg>
</snps_scen_result>
```

テキスト

```
snps_exe_ok=true|false
snps_session_no=session number
snps_error_msg=error message
```

サンプル

HTML フォームでのシナリオの起動

この例では、フォームを使用して StartScen サーブレットをコールします。サーブレットは、メタデータ・ナビゲータのインストール・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。HTML ページのソースは、Oracle Data Integrator のインストール環境の/demo/http/ディレクトリにあります。

```
1  <html>
2     <title>HTTP StartScen Example</title>
3     <body>
4        <h1>Enter appropriate values for your Oracle Data Integrator
         scenario</h1>
```

```
5       <form method="post" name="form1" action="./startscen.do">
6          <table>
7             <tr>
8                <td><b>HTTP Parameter Name</b></td>
9                <td><b>HTTP Parameter Value</b></td>
10            </tr>
11            <tr>
12               <td><b>agent_name</b></td>
13               <td><input type="text" name="agent_name"
                  value=""/></td>
14            </tr>
15            <tr>
16               <td><b>agent_port</b></td>
17               <td><input type="text" name="agent_port"
                  value=""/></td>
18            </tr>
19            <tr>
20               <td><b>master_driver</b></td>
21               <td><input type="text" name="master_driver"
                  value=""/></td>
22            </tr>
23            <tr>
24               <td><b>master_url</b></td>
25               <td><input type="text" name="master_url"
                  value=""/></td>
26            </tr>
27            <tr>
28               <td><b>master_user</b></td>
29               <td><input type="text" name="master_user"
                  value=""/></td>
30            </tr>
31            <tr>
32               <td><b>master_psw</b></td>
33               <td><input type="text" name="master_psw"
                  value=""/></td>
34            </tr>
35            <tr>
36               <td><b>work_repository</b></td>
37               <td><input type="text" name="work_repository"
                  value=""/></td>
38            </tr>
39            <tr>
40               <td><b>snps_user</b></td>
```

```
41                <td><input type="text" name="snps_user"
                  value=""/></td>
42             </tr>
43             <tr>
44                <td><b>snps_psw</b></td>
45                <td><input type="text" name="snps_psw" value=""/></td>
46             </tr>
47             <tr>
48                <td><b>scen_name</b></td>
49                <td><input type="text" name="scen_name"
                  value=""/></td>
50             </tr>
51             <tr>
52                <td><b>scen_version</b></td>
53                <td><input type="text" name="scen_version"
                  value=""/></td>
54             </tr>
55             <tr>
56                <td><b>context_code</b></td>
57                <td><input type="text" name="context_code"
                  value=""/></td>
58             </tr>
59             <tr>
60                <td><b>log_level<b></td>
61                <td><input type="text" name="log_level"
                  value=""/></td>
62             </tr>
63             <tr>
64                <td><b>http_reply</b> (XML|HTML|TXT)</td>
65                <td><input type="text" name="http_reply"
                  value=""/></td>
66             </tr>
67             <tr>
68                <td>Add all other necessary variables for your
scenario here</td>
69                <td/>
70                <td><b>NAME_OF_YOUR_VARIABLE</b></td>
71                <td><input type="text"
                  name="NAME_OF_YOUR_VARIABLE"/></td>
72             </tr>
73          </table>
74          <p><input type="submit"/></p>
75       </form>
```

```
76     </body>
77 </html>
```

HTTP URL からのシナリオの起動

次の例では、単純な URL からシナリオを起動する方法を示します。この例では、メタデータ・ナビゲータがインストールされているリモート・サーバーのサーブレットをコールします。HTML ページのソースは、Oracle Data Integrator のインストール環境の/demo/http/ディレクトリにあります。緑で記載されているパラメータは、実際の構成では無効なため、変更する必要があります。

```
1  <html>
2     <head>
3        <title>Script Example of HTTP StartScen</title>
4        <script type="text/javascript">
5   <!--
6        function submitForm(form)
7        {
8        alert("The scenario is being launched");
9   w = window.open(form.action, "resultwindow",
    ",height=360,width=360,scrollbars=yes, resizable=yes, toolbar=no,
    status=yes, menubar=no");
10  w.focus();
11       form.submit();
12       }
13  -->
14       </script>
15    </head>
16    <body>
17       <form method="post" name="form1"
         action="http://mars:8080/snpsrepexp/startscen.do"
         target="resultwindow">
18
19       <p> Click <a
         href="JavaScript:submitForm(document.form1)">HERE</a> to start
         the scenario "</p>
20       <input type="hidden" name="agent_name" value="MARS"/>
21       <input type="hidden" name="agent_port" value="20910"/>
22       <input type="hidden" name="master_driver"
         value="oracle.jdbc.driver.OracleDriver"/>
23       <input type="hidden" name="master_url"
         value="jdbc:oracle:thin:@mars:1521:mars"/>
24       <input type="hidden" name="master_user" value="snpm32"/>
25       <input type="hidden" name="master_psw"
         value="NENDKGNEJMKCHBHDHEHJDBGBGFDGGH"/>
```

```
26        <input type="hidden" name="work_repository"
          value="WorkRep3D_DEMO_SGS"/>
27        <input type="hidden" name="snps_user" value="SUPERVISOR"/>
28        <input type="hidden" name="snps_psw"
          value="LELKIELGLJMDLKMGHEHJDBGBGFDGGH"/>
29        <input type="hidden" name="scen_name" value="LOAD_SALES"/>
30        <input type="hidden" name="scen_version" value="5"/>
31        <input type="hidden" name="context_code" value="GLOBAL"/>
32        <input type="hidden" name="log_level" value="5"/>
33        <input type="hidden" name="http_reply" value="HTML"/>
34        <!--- List of variables -->
35        <input type="hidden" name="PROJECT.VAR1" value="VAR1_VALUE"/>
36        <input type="hidden" name="PROJECT.VAR2" value="VAR2_VALUE"/>
37        </form>
38     </body>
39 </html>
```

異なるリポジトリでのシナリオの操作

状況によっては、生成元とは異なる作業リポジトリでシナリオを操作する必要があります。

例

次に、このタイプのプロセスが発生する組織の例を 2 つ示します。

- 同じソフトウェア・アプリケーションが稼働する多くの代理店を持つ企業の場合。IT 本部では、データを中央データ・センターに一元化するパッケージとシナリオを開発します。これらのシナリオは、各代理店でまったく同じ方法で実行するよう設計されます。
- ソフトウェア・アプリケーションを開発、調整および運用する目的で 3 つの異なる IT 環境を維持している企業の場合。この企業のプロセスでは、完全に独立した環境が要求されるため、リポジトリを共有できません。

前提条件

この組織の前提条件は、作業リポジトリを各環境（サイト、代理店または環境）にインストールしていることです。この作業リポジトリに関連するマスター・リポジトリのトポロジには、論理アーキテクチャの観点から互換性が存在する必要があります（同じ論理スキーマ名）。物理アーキテクチャで記述される接続特性は、異なっていてもかまいません。

注意:
プロシージャまたはインタフェースで明示的にコンテキスト・コードを指定する場合、ターゲット・トポロジに同じコンテキスト・コードが含まれる必要があります。
トポロジ（つまり、物理アーキテクチャと論理アーキテクチャ）は、開発環境のマスター・リポジトリからエクスポートし、ターゲット・リポジトリにインポートできます。この操作は、トポロジ・モジュールを使用して実行します。この場合、物理トポロジ（サーバー・アドレス）は、シナリオを操作する前にパーソナライズする必要があります。トポロジのインポートでは、ターゲット・リポジトリにすでに存在するデータ・サーバーは変更されず、単に新規データ・サーバーが参照されます。

異なる作業リポジトリでシナリオを操作する手順:

1. 元のリポジトリからシナリオをエクスポートします（右クリックして「エクスポート」）。
2. シナリオ・エクスポート・ファイルをターゲット環境に転送します。
3. ターゲット環境でデザイナ・モジュールを起動します（ターゲット・リポジトリに接続）。
4. エクスポート・ファイルからシナリオをインポートします。

外部スケジューラでのシナリオのスケジュール

シナリオの起動を外部スケジューラに統合するには、OS コマンドを使用して起動をスケジューラに統合します。

シナリオが正しく終了した場合、リターン・コードは 0（ゼロ）です。正しく終了しなかった場合、0 以外の値になります。リターン・コードには、この目的で提供されているオペレーティング・システム環境変数としてアクセスできます。

次に例として、mercure という Windows サーバーで Windows AT スケジューラを使用し、毎月 1 日の午前 0:00 に GLOBAL コンテキストでシナリオ DW のバージョン 3 をスケジュールする方法を示します。

```
AT \\mercure 00 :00 /every 1 "c:\program files\odi\bin\startscen" DW 3
GLOBAL
```

セッションの再開

エラーが発生したセッションまたはオペレータにより停止されたセッションは、再開できます。再開操作は、オペレータから実行するか、コマンドラインを通じて実行します。

注意: 再開できるのは、ステータスが**エラー**または**待機中**のセッションのみです。セッションは、最後の未完了タスク（通常はエラーが発生したタスク）から再開されます。

オペレータからセッションを再開する手順:

1. **オペレータ**を起動します。
2. ツリー・ビューから再開するセッションを選択します。
3. 右クリックし、**「再起動」**を選択します。
4. 表示されるウィンドウで**「OK」**をクリックします。

Oracle Data Integrator によりセッションが再開されると、**「セッションを開始しました」**というメッセージが表示されます。

OS コマンドからセッションを再開する手順:

1. シェルまたは Windows のコマンド・プロンプトを起動します。
2. Oracle Data Integrator のインストール・ディレクトリの/bin ディレクトリに移動します。
3. 次のコマンドを入力します。
   `restartsession.bat <session number> ["-V=<verbose level>"]`（Windows）

   または

   `./restartsession.sh <session number> [-V=<verbose level>]`（UNIX）

Oracle Data Integrator により、セッションが再開されます。セッションの実行状況は、オペレータで監視できます。

Web サービスを使用してセッションを再開する手順:

Oracle Data Integrator の公開 Web サービスを使用すると、Web サービスからセッションを再開できます。

詳細は、「Web サービスを使用したセッションの再開」を参照してください。

関連項目:

- OS コマンドからのシナリオの実行
- デザイナまたはオペレータからのシナリオの実行

その他のタスク

作業リポジトリへの接続

作業リポジトリに接続してデザイナ・モジュールを起動する手順:

1. **「スタート」**メニューで**「プログラム」→「Oracle Data Integrator」→「Designer」**を選択するか、**デザイナ**のスクリプト（`bin/designer.bat` または `bin/designer.sh`）を起動します。
2. **「新規」**ボタン（**「ログイン名」**フィールドの右側にある最初のボタン）をクリックします。
3. 次のフィールドを入力します。

   Oracle Data Integrator 接続:

   - **ログイン名:** 一般的な別名（Repository など）
   - **ユーザー:** SUPERVISOR（大文字を使用）
   - **パスワード:** SUNOPSIS（大文字を使用）

   DBMS 接続(マスター・リポジトリ):

   - **ユーザー:**（作業リポジトリではなく）マスター・リポジトリに対して作成した表の所有者のユーザーID またはログイン。
   - **パスワード:** このユーザーのパスワード。
   - ドライバ・リスト: 作成した**マスター・リポジトリ**をホストする DBMS に接続する際に必要なドライバを選択します。
   - **URL: マスター・リポジトリ**をホストするデータ・サーバーの完全なパス。詳細は、「JDBC URL のサンプル」を参照してください。

   作業リポジトリ:

   - **作業リポジトリ名:** 前の手順で作業リポジトリに指定した名前（WorkRep1 など）。このフィールドの右側にあるボタンをクリックすると、現在のマスター・リポジトリで使用可能な作業リポジトリのリストを表示できます。
4. **「テスト」**をクリックして接続動作を確認します。
5. **「OK」**をクリックします。**デザイナ**・モジュールが起動します。

重要: SUPERVISOR アカウントのデフォルト・パスワードは、SUNOPSIS です。セキュリティ上問題となるため、このパスワードはできるだけ早く変更してください。

ユーザー・パスワードの変更

管理者は、**セキュリティ・マネージャ**のユーザー・プロパティを通じてユーザーのパスワードを変更できます。ユーザーも、グラフィカル・モジュールで自分のパスワードを変更できます。

ユーザー・パスワードを変更する手順:

1. 変更するパスワードを保持するユーザーとしてログインします。
2. **「ファイル」**メニューで、**「パスワードの変更」**を選択します。
3. 現在のパスワードを入力し、次に新規パスワードを 2 回入力します。
4. **「OK」**をクリックして保存します。

ユーザー・パスワードが変更されます。

警告: パスワードが接続プロパティに格納されている場合、次回グラフィカル・モジュールを通じてログインする際に、接続情報を更新する必要があります。また、このユーザーのログインおよびパスワード情報を含むパラメータ・ファイルもすべて更新する必要があります。

注意: パスワードは、管理者の定義によるパスワード・ポリシーに常に準拠する必要があります。準拠しない場合、変更を保存しようとするとエラーが発生します。

ユーザー・パラメータの設定

Oracle Data Integrator では、デフォルト・ディレクトリやウィンドウ位置などのユーザー・パラメータが保存されます。

ユーザー・パラメータは、`/bin` の `userpref.xml` ファイルに保存されます。

ユーザー・パラメータを設定する手順:

1. **「ファイル」**メニューで、**「ユーザー・パラメータ」**を選択します。
2. **ユーザー・パラメータを編集中**ウィンドウで、必要に応じてパラメータの値を変更します。
3. **「OK」**をクリックして保存し、ウィンドウを閉じます。

使用可能なユーザー・パラメータのリストは、リファレンス・マニュアルを参照してください。

デザイナ: レポートの生成

デザイナでレポートを生成する手順:

1. レポートを生成するオブジェクトを選択します。
2. 右クリックして**「印刷」**を選択し、生成するレポートのタイプを選択します。
3. レポートを書き込むファイルの名前を指定します。パスを指定しない場合、ファイルは PDF ファイルのデフォルト・ディレクトリに書き込まれます。この設定は、ユーザー・プリファレンスです。
4. **「OK」**をクリックします。

手順 3 で指定したファイルに、**Adobe™ PDF** 形式でレポートが書き込まれます。

「ファイルを生成後に開きますか。」オプションを選択した場合、**Acrobat® Reader™**が起動して生成されたレポートが表示されます。

> **注意:** 生成されたレポートを表示するには、ユーザー・パラメータで **Acrobat® Reader™**の場所を指定しておく必要があります。

オブジェクト・タイプごとに次のタイプのレポートを生成できます。

オブジェクト	レポート
プロジェクト	ナレッジ・モジュール
フォルダ	フォルダ、パッケージ、インタフェース、プロシージャ
モデル	モデル
サブモデル	サブモデル

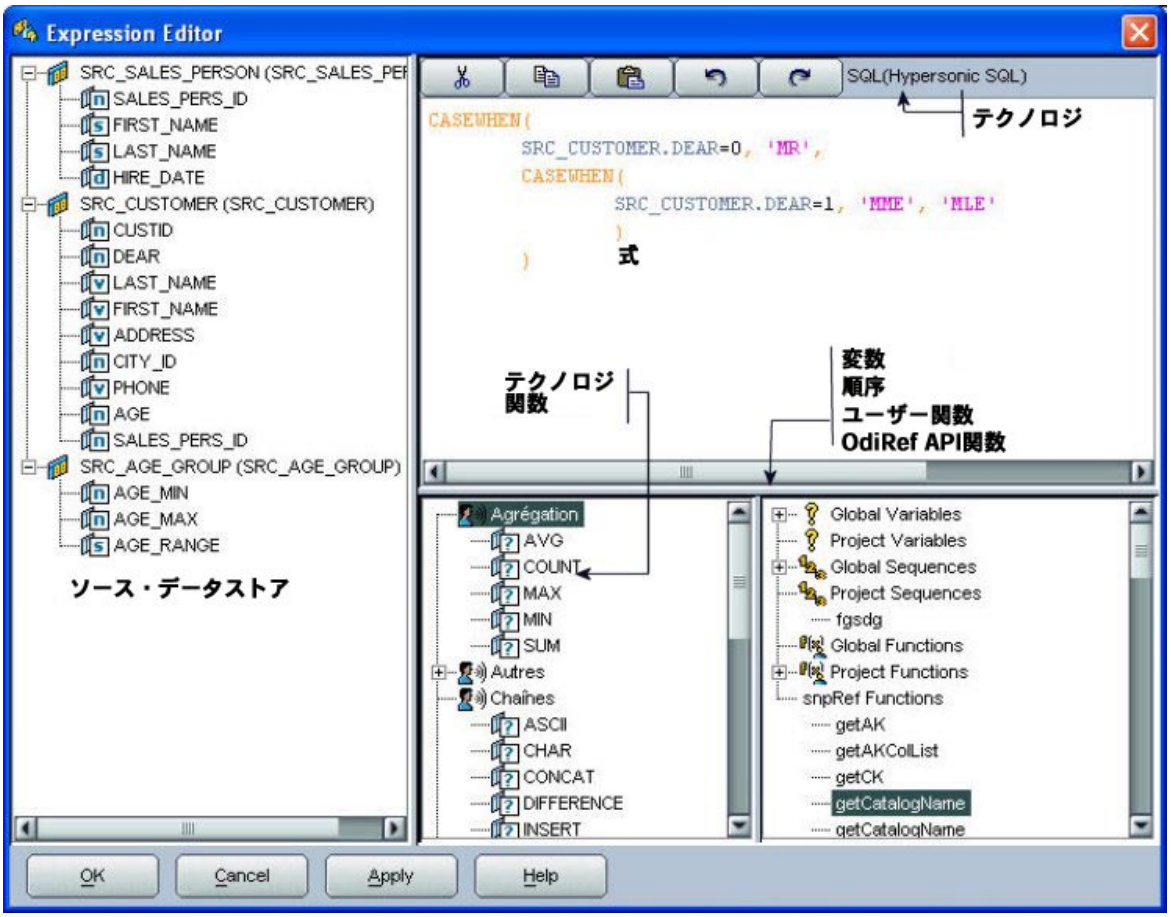
式エディタの使用方法

式エディタは、  ボタンをクリックすることで起動できます。式エディタは、マッピング、フィルタ、結合、またはその他の問合せの式を記述する場合に便利です。式エディタを使用すると、一般的な構文エラーを回避できます。式エディタには、関数や他のオブジェクトの役に立つ参照も含まれます。

次に、式エディタの図を示します。式エディタは、次の部分で構成されます。

- **ソース・データストア:** インタフェースの式を編集する場合、ここにソース・データストアおよび列の名前が表示されます。
- **式:** 式の現在のテキスト。直接ここにコードを入力するか、他のパネルから要素をドラッグ・アンド・ドロップします。
- **テクノロジ関数:** 特定の**テクノロジ**に適した言語要素および関数のリスト。
- **変数、順序、ユーザー関数および odiRef API:** 次のものが含まれます。
  - プロジェクトおよびグローバル変数
  - プロジェクトおよびグローバル順序
  - プロジェクトおよびグローバル・ユーザー定義関数
  - OdiRef 置換メソッド

標準の編集機能（切取り、コピー、貼付け、UNDO）は、ツールバー・ボタンを通じて使用できます。



オブジェクトの検索

Oracle Data Integrator の検索機能を使用すると、デザイナ、オペレータ、トポロジ・マネージャなどの各グラフィカル・モジュールでオブジェクトを迅速に検索できます。

検索は、それぞれ特定の範囲内で実行されます。**検索範囲**は、Oracle Data Integrator の機能ドメインに対応する事前定義されたオブジェクト・タイプのセットです。
たとえば、メタデータの検索範囲では、メタデータ関連のすべてのオブジェクト（データストア、モデル、参照など）が検索され、プロシージャの検索範囲ではプロシージャとプロシージャのコマンドおよびオプションが検索されます。

検索基準は次のとおりです。

- Oracle Data Integrator オブジェクトのすべての範囲とすべてのタイプ
- 特定の範囲内のすべてのオブジェクト・タイプ
- 特定のオブジェクト・タイプ

検索を実行する手順:

1. モジュール・ツールバーの**「検索ビュー」**ボタンをクリックし、検索パネルを開きます。
2. **「検索範囲」**を選択するか、すべての Oracle Data Integrator オブジェクトのリストに対応する「<すべての有効範囲>」を選択します。

3. **「オブジェクト・タイプ」**を選択するか（この選択はオプションで検索範囲によりフィルタされます）、すべてのオブジェクト・タイプを検索する「<すべてのオブジェクト型>」を選択します。
4. 次の検索基準を指定します。
    - **名前:** このフィールドは常に使用可能です。検索するオブジェクトの名前を入力します。
    - **お気に入り基準:** 特定のオブジェクト・タイプの最も効果的な基準に対応するフィールド。**「オブジェクト・タイプ」**を選択しない場合、このパネルは使用できません。
    - **日付およびユーザー:** オブジェクトのユーザー、基準の日付、または最新の更新に基づいて検索をフィルタします。
5. **「拡張」**タブを選択すると、特定のオブジェクト・タイプに関する追加の検索基準を指定できます。**「オブジェクト・タイプ」**を選択しない場合、このタブには**「名前」**フィールドのみが表示されます。
6. すべてのテキストを大/小文字を区別して一致させる場合、**「大/小文字一致」**ボックスを選択します。
7. **「検索」**ボタンをクリックします。
検索が開始されます。プログレス・バーが表示されます。
8. 検索が完了すると、**「階層表示」**または**「線形表示」**に結果が表示されます。ビューを切り替えるには、一番下のタブを使用します。
オブジェクトを右クリックすることで、そのオブジェクトを編集、削除または表示できます。
9. 現在の検索基準をファイル内に保存するには、**「検索条件を保存」**ボタンをクリックします。保存した検索基準を後で使用するには、**「検索基準のロード」**ボタンを使用します。

検索のヒント

- 大量のオブジェクトが検索されることを避けるため、検索範囲と適切な検索基準を指定してください。特定のタイプおよび基準に一致するオブジェクトが多すぎると、検索が遅くなる可能性があります。
- **「名前」**などのテキスト・フィールドでは、基準として入力された値が、オブジェクトのテキスト・フィールド内のどこかに含まれていれば一致します。たとえば、**「名前」**フィールドに Data という語を入力して検索すると、Project Data Warehouse や Data Mart Structure などのオブジェクトが戻されます。数値や日付などの他のフィールド・タイプでは、正確に一致するオブジェクトのみが戻されます。

インポート/エクスポート

作業リポジトリのインポート

作業リポジトリをインポートまたはエクスポートすると、あるリポジトリから別のリポジトリにすべてのモデルおよびプロジェクトを移動できます。

作業リポジトリをインポートする手順:

1. デザイナで、**「ファイル」→「インポート」→「作業リポジトリ」**を選択します。
2. **インポート・モード**と、インポート元の**フォルダ**または **Zip ファイル**を選択します。

3. **「OK」**をクリックします。

指定したファイルが作業リポジトリにインポートされます。

関連項目:

作業リポジトリのエクスポート

インポート・モード

マスター・リポジトリのエクスポート

マスター・リポジトリのインポート

トポロジまたはセキュリティ設定のエクスポート

トポロジまたはセキュリティ設定のインポート

作業リポジトリのエクスポート

作業リポジトリをインポートまたはエクスポートすると、あるリポジトリから別のリポジトリにすべてのモデルおよびプロジェクトを移動できます。

作業リポジトリをエクスポートする手順:

1. デザイナで、**「ファイル」→「エクスポート」→「作業リポジトリ」**を選択します。
2. 次のパラメータを指定します。
   - **ディレクトリへエクスポート:** エクスポート・ファイルを作成するディレクトリ。
   - **圧縮されたエクスポートを作成:** すべてのオブジェクトを、複数の個別ファイルではなく単一の圧縮ファイルにエクスポートします。
   - **ZIP ファイル名:** 圧縮ファイルの名前（**「圧縮されたエクスポートを作成」**オプションを使用している場合）。
   - **キャラクタ・セット:** エクスポート・ファイルで使用するエンコーディング。このエンコーディングは、XML ファイル・ヘッダーで`<?xml version="1.0" encoding="ISO-8859-1"?>`のように指定されます。
   - **Java キャラクタ・セット:** ファイル生成に使用する Java キャラクタ・セット。
3. **「OK」**をクリックします。

指定したエクスポート・ディレクトリにエクスポート・ファイルが作成されます。

関連項目:

作業リポジトリのインポート

マスター・リポジトリのエクスポート

マスター・リポジトリのインポート

トポロジまたはセキュリティ設定のインポート

トポロジまたはセキュリティ設定のエクスポート

オブジェクトのインポート

インポートおよびエクスポートにより、オブジェクト（インタフェース、ナレッジ・モジュール、モデルなど）をあるリポジトリから別のリポジトリに移動できます。

Oracle Data Integrator でオブジェクトをインポートする手順:

1. インポートする適切なオブジェクトをツリーから選択します。適切なオブジェクトが存在しない場合は、作成します。
2. オブジェクトを右クリックして**「インポート」**を選択し、インポートするオブジェクトのタイプを選択します。
3. **「エクスポート・ディレクトリ」**を指定し、**「インポート名」**で 1 つ以上のファイルを選択します。**インポート・モード**を選択し、**「OK」**をクリックします。

XML 形式のファイルが作業リポジトリにインポートされ、Oracle Data Integrator に表示されます。

関連項目:

オブジェクトのエクスポート

インポート・モード

オブジェクトのエクスポート

エクスポートおよびインポートにより、Oracle Data Integrator オブジェクトをあるリポジトリから別のリポジトリに移動できます。

Oracle Data Integrator からオブジェクトをエクスポートする手順:

1. Oracle Data Integrator の適切なツリー・ビューで、エクスポートするオブジェクトを選択します。
2. オブジェクトを右クリックし、**「エクスポート」**を選択します。このメニュー項目が表示されない場合は、エクスポート機能に対応しないタイプのオブジェクトです。
3. エクスポート・パラメータを設定し、**「OK」**をクリックします。
   少なくとも**「エクスポート名」**は指定する必要があります。**「エクスポート・ディレクトリ」**を指定しない場合、エクスポート・ファイルは**「デフォルトのエクスポート・ディレクトリ」**に作成されます。

オブジェクトが XML ファイルとして指定した場所にエクスポートされます。

関連項目:

オブジェクトのインポート

オブジェクト・エクスポート・パラメータ

複数オブジェクトのエクスポート

複数オブジェクトのエクスポート

「複数オブジェクトをエクスポート」メニュー項目を使用して、1 つ以上のオブジェクトを一度にエクスポートできます。これにより、複数のオブジェクトを単一の Zip ファイルまたはディレクトリにエクスポートすることや、オブジェクトの既存のリストを再利用しながらエクスポートすることができます。

このような作業を行うためのより効果的なメカニズムは、ソリューションを使用することです。「バージョン管理の使用方法」を参照してください。

複数オブジェクトを同時にエクスポートする手順:

1. デザイナ、トポロジ・マネージャ、セキュリティ・マネージャまたはオペレータのメニューから**「複数オブジェクトをエクスポート」**を選択します。

2. エクスポートのオプションを指定します。
   - **ディレクトリへエクスポート:** エクスポート先のディレクトリ。これには、既存のディレクトリを指定する必要があります。
   - **ZIP ファイルとして保存:** Zip ファイルを生成するかどうかを指定します。オブジェクトは、.xml ファイルとしてディレクトリに直接エクスポートするか、.xml ファイルを含む Zip ファイルとしてエクスポートします。
     - **ZIP ファイル名:** Zip ファイルを生成する場合、そのファイルに名前を付ける必要があります。
   - **子オブジェクトをエクスポート:** 子オブジェクト（パッケージのステップ、プロジェクトのフォルダなど）をエクスポートするかどうかを選択します。通常、このボックスは選択する必要があります。
   - **既存のファイルを警告なしで置換します:** 選択しない場合、エクスポート・プロセスでファイルが上書きされる前に警告メッセージが表示されます。
   - **XML バージョン:** エクスポート・ファイルのヘッダーに書き込まれる XML バージョン。
   - **キャラクタ・セット:** XML ヘッダーに書き込まれるキャラクタ・セットの名前。
   - **Java キャラクタ・セット:** エンコーディング・プロセスで使用される Java キャラクタ・セットの名前。非ヨーロッパ系のキャラクタ・セットの場合は、この設定を変更する必要があります。
3. エクスポートするオブジェクトのリストを次のように指定します。
   - Oracle Data Integrator のグラフィカル・モジュールからオブジェクトをドラッグ・アンド・ドロップします。たとえば、トポロジ・マネージャのオブジェクトをデザイナのオブジェクトと組み合せることができます。
   - **「オブジェクトのリストをロード」**をクリックして前に保存しておいたオブジェクトのリストをロードします。この方法は、オブジェクトの同じリストを定期的にエクスポートする場合に便利です。
   - オブジェクトのリストを保存するには、**「このリストを保存」**を選択してファイル名を指定します。ファイルがすでに存在する場合は、警告なしに上書きされます。

複数オブジェクトを同時にインポートするには、ソリューションを使用する必要があります。

関連項目:

オブジェクトのエクスポート

オペレータ

オペレータの概要

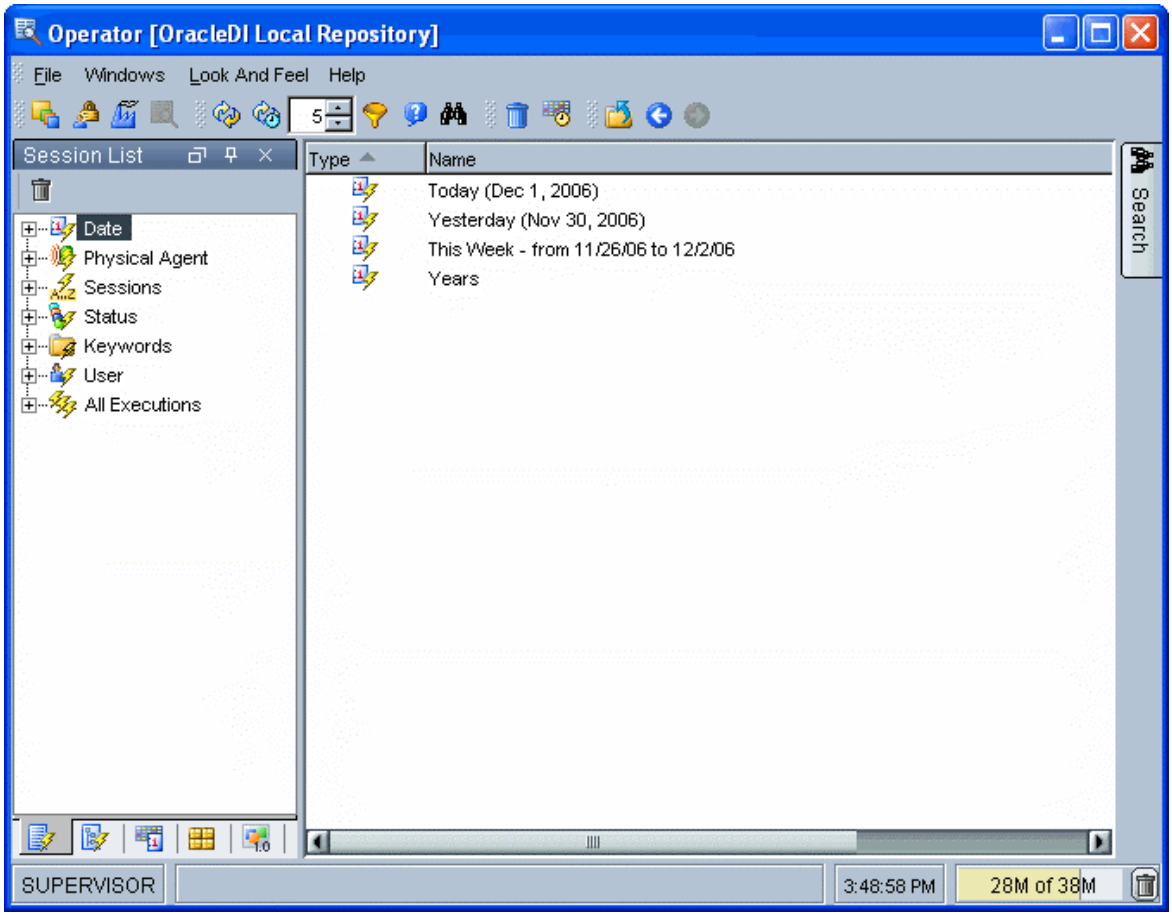
オペレータ・モジュールを使用すると、**セッション**でのインタフェース実行と、本番での**シナリオ**を管理することができます。

オペレータ・モジュールは、マスター・リポジトリで定義されているトポロジを使用して、この情報を作業リポジトリに格納します。

- オペレータの操作

オペレータのインタフェース

オペレータの GUI は次のように表示されます。



メニュー

メニューでは、プルダウン・メニューから次の機能にアクセスできます。

- インポート
- ログ消去
- スケジューリング
- オプションの表示
- モジュールまたはツリー表示のオープン
- ユーザーのパスワードおよびオプションの変更

ツールバー

ツールバーからは、次の操作を実行できます。

- 他のモジュールの起動
- ログの参照

- ログの消去
- スケジューリング情報の表示
- 手動または自動でのログのリフレッシュ
- オンライン・ヘルプの起動

ツリー表示

現行ユーザーが使用できるオペレータ・オブジェクトは、次のようにツリー表示に整理して表示されます。

- **セッション・リスト** すべてのセッションを、日付、物理エージェント、状態、キーワードなどを基準に整理して表示します。
- **階層セッション**には、子セッションとともに階層構造に整理された実行セッションが表示されます。
- **スケジューリング**には、物理エージェントとスケジュールのリストが表示されます。
- **シナリオ**には、使用できるシナリオのリストが表示されます。

各ツリー表示は、メイン・ウィンドウの両側にドッキングできるフローティング・フレームに表示されます。フレームは重ねることもできます。複数のフレームを重ねた場合は、フレーム・ウィンドウの下部に表示されるタブから各フレームにアクセスできます。

ツリー表示フレームは、フレームのタイトルまたはタブを選択してドラッグすることにより、移動したり、ドッキングしたり、重ねたりできます。ツリー表示の位置を固定するには、**「ウィンドウ」**メニューから**「ウィンドウ・レイアウトのロック」**を選択します。

ツリー表示フレームがメイン・ウィンドウに表示されないか、閉じている場合は、**「ウィンドウ」→「ビューの表示」**メニューを使用します。

各ツリー表示では、次の操作が可能です。

- ノードの展開または折りたたみ（ノードをクリック）
- オブジェクトに関連付けられているメソッド（編集、削除など）のアクティブ化（ポップアップ・メニューを使用）
- オブジェクトの編集（オブジェクトをダブルクリック、または**ワークベンチ**にドラッグ・アンド・ドロップ）

ワークベンチ

Workbench には、ツリー表示で現在選択されているサブオブジェクトのリストが表示されます。たとえば、ツリー表示で手順が選択されていると、この手順に関するタスクのリストがワークベンチに表示されます。各リストにある列は、列タイトルのポップアップ・メニューを使用してカスタマイズできます。

編集または表示されているオブジェクトのウィンドウは、**ワークベンチ**に表示されます。

オペレータの操作

セッション

セッション、ステップ、タスク

セッションは、実行エージェントによって起動される（シナリオ、インタフェース、パッケージ、プロシージャなどの）実行状態です。セッションは**ステップ**で構成され、ステップは**タスク**で構成されます。

ステップは、タスクとセッションの間に位置する実行単位です。ステップは、パッケージまたはシナリオ内の 1 つのステップに対応します。たとえば、インタフェースまたは単一の変数を実行する場合、セッションには 1 つのセッション・ステップのみが含まれます。

タスクは、最小の実行単位です。タスクは、KM 内のプロシージャ・コマンド、プロシージャ、変数の割当てなどに対応します。

ステータス

セッション、ステップまたはタスクには、常にステータスが割り当てられます。

次の 6 つのステータス値があります。

- **完了**: セッション、ステップまたはタスクの実行は成功しました。
- **エラー**: セッション、ステップまたはタスクは、エラーのために終了しました。
- **実行中**: セッション、ステップまたはタスクは実行中です。
- **待機中**: セッション、ステップまたはタスクは実行を待機中です。
- **警告**（タスクのみ）: タスクはエラーのために終了しましたが、このタスクではエラーが許可されるため、セッションは停止されませんでした。
- **キュー**（セッションのみ）: セッションは、実行でエージェントが使用可能になるのを待機中です。

セッションが終了すると、そのステータスは最後に実行されたステップのステータス（**「完了」**または**「エラー」**）になります。**ステップ**が終了すると、そのステータスは最後に実行された**タスク**のステータスになります（タスクにより**「警告」**が戻された場合を除きます。その場合、ステップのステータスは**「完了」**になります）。

エラーの処理

エラーを分析する手順:

1. オペレータのツリー・ビューで、エラーの発生したセッション、ステップおよびタスクを識別します。
2. タスクのエラー・マーカーをダブルクリックします。タスク・ウィンドウが表示されます。

- **「実行」**タブに、セッションを停止したエラーを示す**リターン・コード**と**メッセージ**が表示されます。
- **「定義」**タブに、失敗した**順序**が表示されます。

ログの管理

Oracle Data Integrator には、ログ・データを管理するための複数のソリューションがあります。

- オペレータでセッションをフィルタすると、オペレータに特定の実行セッションのみが表示されます。
- ログをパージして、過去のセッションの情報を削除できます。
- アーカイブ目的のために、ログをインポートおよびログをエクスポートできます。

シナリオ

オペレータからシナリオを実行する手順:

1. オペレータの**「シナリオ」**ビューで、必要なシナリオを選択します。
2. 右クリックし、**「実行」**を選択します。
3. **「コンテキスト」**と必要な**「エージェント」**を選択します（クライアント・ステーションによる実行の場合、「ローカル(エージェントなし)」オプションを選択します）。
4. **「OK」**をクリックします。

Oracle Data Integrator により実行セッションが作成および開始され、「セッションを開始しました」というメッセージが表示されます。セッションの実行状況は、ログで監視できます。

シナリオを実行する前に、シナリオをデザイナで生成するか、ファイルからインポートする必要があります。

シナリオを削除する手順:

1. シナリオを右クリックし、**「削除」**を選択します。
2. **「OK」**をクリックして削除を確認します。
3. リンクしているセッションを削除するかどうかを指定します。リンクしているセッションは、シナリオの実行により作成されたセッションです。
   - 今後この質問を表示しない場合は、**「この選択を記憶する」**チェック・ボックスを選択します。後でこの設定を変更するには、**「シナリオにリンクされたセッションの削除」**ユーザー・パラメータを変更します。
4. **「OK」**をクリックします。

オペレータでのセッションのフィルタ

ユーザー、ステータス、セッション期間などのパラメータに基づいてログ・セッションをフィルタすると、**オペレータ**に特定の実行セッションのみを表示できます。現在のフィルタに一致しないセッションは、ビューで非表示になりますが、ログからは削除されません。

特定のセッションをフィルタする手順:

1. **オペレータ**を起動します。
2. ツールバーの  **「フィルタ」**ボタンをクリックします。
3. 次のパラメータの適切な値を選択します。デフォルト設定では、すべてのセッションが選択されます。
   - **セッション番号:** すべてのセッションを表示する場合、空白を使用します。

- **セッション名:** ワイルドカードとして%を使用します。つまり、DWH%と指定すると、名前が DWH で始まるすべてのセッションが一致します。
- **コンテキスト:** セッションの実行コンテキスト。
- **エージェント:** セッションの実行に使用されるエージェント。
- **ユーザー:** セッションを起動したユーザー。
- **ステータス:**「実行中」、「待機中」など。
- **日付:** 実行の日付。日付の基準として**「FROM」**または**「TO」**（あるいはその両方）を指定します。
- **継続時間が次より長い:** 指定した秒数を超える期間。

4. **「適用」**をクリックして現在のフィルタをプレビューします。
5. **「OK」**をクリックします。

指定した値に一致しないセッションは、セッション・リストでマスクされます。ツールバーの**「フィルタ」**ボタンが押し込まれて表示されます。

フィルタを解除する手順:

1. ツールバーの**「フィルタ」**ボタンが押し込まれて表示されている場合、そのボタンをクリックします。

現在のフィルタが解除され、すべてのセッションがリストに表示されます。

スケジュール情報の表示

スケジュール情報により、エージェントのスケジュール済タスクを一覧表示できます。

重要: スケジュール情報は、エージェントのスケジュールから取得されます。正確なスケジュール情報を表示するには、エージェントを起動してスケジュールをリフレッシュする必要があります。

オペレータからスケジュール情報を表示する手順:

1. **オペレータ**を起動します。
2. ツールバーの**「スケジューリング情報」**ボタンをクリックします。

すべてのエージェント用の**スケジューリング情報**ウィンドウが表示されます。

トポロジ・マネージャからスケジュール情報を表示する手順:

1. **トポロジ・マネージャ**を起動します。
2. **「物理アーキテクチャ」**ビューで、計画を表示する**物理エージェント**を選択します。
3. 右クリックし、**「スケジューリング情報」**を選択します。

そのエージェント用の**スケジューリング情報**ウィンドウが表示されます。

ログのパージ

ログをパージすると、過去のセッションの情報を削除できます。この手順により、作業リポジトリの過去のセッションが消去されます。パージは定期的に実行することをお薦めします。パージ作業は、OdiPurgeLog ツールを使用して自動化できます。

ログをパージする手順:

1. **オペレータ**を起動します。
2. ツールバーの**「ログのパージ」**ボタンをクリックするか、メニュー・バーの**「ファイル」→「ログのパージ」**を選択します。
3. 削除するセッションの日付および時間範囲を設定します。オプションで、**「コンテキスト」**、**「エージェント」**および**「ユーザー」**フィルタを指定します。また、ワイルドカードとして%を使用し、**「セッション名」**マスクを指定することもできます。
   指定した時間範囲内に存在し、3 つのフィルタおよびセッション・マスクに一致するセッションのみが削除されます。
   「ページ・シナリオ・レポート」ボックスを選択すると、（各シナリオの**「実行」**ノードの下に表示される）シナリオ・レポートもパージされます。
4. **「OK」**をクリックします。

Oracle Data Integrator により、ログからセッションが削除されます。

注意: オペレータで 1 つ以上のセッションを選択し、**[Del]**キーを押してセッションを削除することもできます。この方法を使用する場合は十分注意してください。多数のセッションを削除する場合、パフォーマンスの問題が生じる可能性があります。その場合は、ツールバーの**「ログのパージ」**ボタンを使用してください。

デザイナまたはオペレータからのシナリオの実行

この実行モードでは、**デザイナ**または**オペレータ**からシナリオを起動します。

デザイナまたはオペレータからシナリオを起動する手順:

1. **「プロジェクト」**ビュー（デザイナの場合）または**「シナリオ」**ビュー（オペレータの場合）で必要なシナリオを選択します。
2. 右クリックし、**「実行」**を選択します。
3. **「コンテキスト」**と必要な**「エージェント」**を選択します。ローカル・マシンでシナリオを実行するには、**「ローカル（エージェントなし）」**オプションを選択します。
4. **「OK」**をクリックします。
5. シナリオでパラメータとして変数を使用する場合、割り当てる値を選択します。変数の「最後の値」を選択すると、現在の値が使用されます。使用可能な値がない場合は、デフォルト値が使用されます。
6. Oracle Data Integrator により新規セッションが作成および開始されると、**「セッションを開始しました」**というメッセージが表示されます。
7. セッションの実行状況は、**オペレータ**で監視できます。

セッションの再開

エラーが発生したセッションまたはオペレータにより停止されたセッションは、再開できます。再開操作は、オペレータから実行するか、コマンドラインを通じて実行します。

注意: 再開できるのは、ステータスが**エラー**または**待機中**のセッションのみです。セッションは、最後の未完了タスク（通常はエラーが発生したタスク）から再開されます。

オペレータからセッションを再開する手順:

1. **オペレータ**を起動します。
2. ツリー・ビューから再開するセッションを選択します。
3. 右クリックし、**「再起動」**を選択します。
4. 表示されるウィンドウで**「OK」**をクリックします。

Oracle Data Integrator によりセッションが再開されると、**「セッションを開始しました」**というメッセージが表示されます。

OS コマンドからセッションを再開する手順:

1. シェルまたは Windows のコマンド・プロンプトを起動します。
2. Oracle Data Integrator のインストール・ディレクトリの/bin ディレクトリに移動します。
3. 次のコマンドを入力します。
   `restartsession.bat <session number> ["-V=<verbose level>"]`（Windows）

   または

   `./restartsession.sh <session number> [-V=<verbose level>]`（UNIX）

Oracle Data Integrator により、セッションが再開されます。セッションの実行状況は、オペレータで監視できます。

Web サービスを使用してセッションを再開する手順:

Oracle Data Integrator の公開 Web サービスを使用すると、Web サービスからセッションを再開できます。

詳細は、「Web サービスを使用したセッションの再開」を参照してください。

関連項目:

- OS コマンドからのシナリオの実行
- デザイナまたはオペレータからのシナリオの実行

本番環境へのシナリオのインポート

デザイナで生成したシナリオは、オペレータ・モジュールを通じて開発または実行リポジトリにインポートし、本番環境で実行できます。

同様に、シナリオのグループを一度に移動するため、複数のシナリオを含むソリューションをインポートすることも可能です。詳細は、「バージョン管理の使用方法」を参照してください。

本番環境にシナリオをインポートする手順:

1. **オペレータ**を起動します。
2. メニュー・バーで**「ファイル」→「シナリオのインポート」**を選択するか、**「シナリオ」**ビューで右クリックして**「シナリオのインポート」**を選択します。
3. **「インポート・ディレクトリ」**を指定し、インポートする 1 つ以上のシナリオを選択します。
4. インポート・モードを選択します。

5. シナリオとともにエクスポートされたスケジュールもインポートする場合は、**「スケジュールのインポート」**オプションを選択します。
6. **「OK」**をクリックします。

インポートしたシナリオが**「シナリオ」**ノードに表示されます。エクスポート時にシナリオ・フォルダ内にあったシナリオは、同じシナリオ・フォルダに再インポートされます。

ソリューションをインポートする手順:

1. **「ソリューション」**ビューに移動します。
2. ツリー・ビューを右クリックし、**「ソリューションのインポート」**を選択します。
3. **「インポート・ディレクトリ」**を指定し、インポートする 1 つ以上のソリューションを選択します。
4. インポート・モードを選択します。
5. **「OK」**をクリックします。
6. ソリューションをダブルクリックして開きます。
7. **「プリンシパル要素」**または**「必須要素」**ビューで、リストアする 1 つ以上のシナリオを選択します。
8. **「リストア」**ボタンをクリックします。

注意: オペレータ内からリストアできるのは、シナリオのみです。

シナリオ・フォルダ

オペレータでは、複数のシナリオをシナリオ・フォルダにグループ化して簡単に整理できます。シナリオ・フォルダには、別のシナリオ・フォルダを含めることができます。

シナリオ・フォルダを作成する手順:

1. **「シナリオ」**ビューに移動します。
2. ツリー・ビューの任意の場所を右クリックして**「シナリオ・フォルダの挿入...」**を選択するか、  **「シナリオ・フォルダの追加」**ボタンをクリックします。
3. シナリオをドラッグ・アンド・ドロップしてシナリオ・フォルダに移動します。

シナリオ・フォルダを削除する手順:

1. シナリオ・フォルダを右クリックし、**「削除」**を選択します。
2. すべてのシナリオに加え、そのフォルダに含まれる他のシナリオ・フォルダもすべて削除されます。

セッション・フォルダの使用方法

セッション・フォルダでは、特定のキーワードとともに起動されたセッションを自動的にグループ化できます。セッション・フォルダは、**「セッション・リスト」**ビューの**「キーワード」**ノードに作成します。各セッション・フォルダには、関連する 1 つ以上のキーワードを割り当てます。その後、すべてのキーワードに一致するセッションが起動されると、自動的にこのセッション・フォルダの下に分類されます。

新規セッション・フォルダを作成する手順:

1. オペレータで、**「セッション・リスト」**ビューに移動します。
2. **「キーワード」**ノードを右クリックし、**「セッション・フォルダの挿入」**を選択します。
3. **「フォルダ名」**を指定します。
4. 1 つ以上のキーワードを指定します。各キーワードを入力したら、**「追加」**ボタンをクリックしてリストに追加します。

注意: 特定のセッション・フォルダのすべてのキーワードを含むセッションのみが、そのセッション・フォルダの下に表示されます。キーワードの一致基準では、大文字と小文字が区別されます。

次に、セッション・フォルダのキーワードが一致する場合の例を示します。

セッション・フォルダ・キーワード	セッション・キーワード	一致するか
DWH、Test、Batch	Batch	一致しません（すべてのキーワードが一致する必要があります）。
Batch	DWH、Batch	一致します（セッションの別のキーワードは無視されます）。
DWH、Test	Test、dwh	一致しません（一致基準では大文字と小文字が区別されます）。

キーワードを含むシナリオを起動する手順:

1. SnpsStartScen ツールを`-KEYWORDS` パラメータ付きで使用します。キーワードはカンマで区切る必要があります。

注意: セッション・フォルダのキーワード一致は動的に処理されます。セッション・フォルダのキーワードが変更されるか、新規フォルダが作成されると、既存のセッションが即座に再分類されます。

オブジェクトの検索

Oracle Data Integrator の検索機能を使用すると、デザイナ、オペレータ、トポロジ・マネージャなどの各グラフィカル・モジュールでオブジェクトを迅速に検索できます。

検索は、それぞれ特定の範囲内で実行されます。**検索範囲**は、Oracle Data Integrator の機能ドメインに対応する事前定義されたオブジェクト・タイプのセットです。
たとえば、メタデータの検索範囲では、メタデータ関連のすべてのオブジェクト（データストア、モデル、参照など）が検索され、プロシージャの検索範囲ではプロシージャとプロシージャのコマンドおよびオプションが検索されます。

検索基準は次のとおりです。

- Oracle Data Integrator オブジェクトのすべての範囲とすべてのタイプ
- 特定の範囲内のすべてのオブジェクト・タイプ
- 特定のオブジェクト・タイプ

検索を実行する手順:

1. モジュール・ツールバーの  **「検索ビュー」**ボタンをクリックし、検索パネルを開きます。
2. **「検索範囲」**を選択するか、すべての Oracle Data Integrator オブジェクトのリストに対応する「<すべての有効範囲>」を選択します。
3. **「オブジェクト・タイプ」**を選択するか（この選択はオプションで検索範囲によりフィルタされます）、すべてのオブジェクト・タイプを検索する「<すべてのオブジェクト型>」を選択します。
4. 次の検索基準を指定します。
   - **名前:** このフィールドは常に使用可能です。検索するオブジェクトの名前を入力します。
   - **お気に入り基準:** 特定のオブジェクト・タイプの最も効果的な基準に対応するフィールド。**「オブジェクト・タイプ」**を選択しない場合、このパネルは使用できません。
   - **日付およびユーザー:** オブジェクトのユーザー、基準の日付、または最新の更新に基づいて検索をフィルタします。
5. **「拡張」**タブを選択すると、特定のオブジェクト・タイプに関する追加の検索基準を指定できます。**「オブジェクト・タイプ」**を選択しない場合、このタブには**「名前」**フィールドのみが表示されます。
6. すべてのテキストを大/小文字を区別して一致させる場合、**「大/小文字一致」**ボックスを選択します。
7. **「検索」**ボタンをクリックします。
   検索が開始されます。プログレス・バーが表示されます。
8. 検索が完了すると、**「階層表示」**または**「線形表示」**に結果が表示されます。ビューを切り替えるには、一番下のタブを使用します。
   オブジェクトを右クリックすることで、そのオブジェクトを編集、削除または表示できます。
9. 現在の検索基準をファイル内に保存するには、  **「検索条件を保存」**ボタンをクリックします。保存した検索基準を後で使用するには、  **「検索基準のロード」**ボタンを使用します。

検索のヒント

- 大量のオブジェクトが検索されることを避けるため、検索範囲と適切な検索基準を指定してください。特定のタイプおよび基準に一致するオブジェクトが多すぎると、検索が遅くなる可能性があります。
- **「名前」**などのテキスト・フィールドでは、基準として入力された値が、オブジェクトのテキスト・フィールド内のどこかに含まれていれば一致します。たとえば、**「名前」**フィールドに Data という語を入力して検索すると、Project Data Warehouse や Data Mart Structure などのオブジェクトが戻されます。数値や日付などの他のフィールド・タイプでは、正確に一致するオブジェクトのみが戻されます。

ログのインポートおよびエクスポート

ログ・データのインポート

ログ・データのインポートでは、アーカイブのためにエクスポートされたログ・ファイルを作業リポジトリにインポートできます。

ログをインポートする手順:

1. **オペレータ**を起動します。
2. メニュー・バーで、「**ファイル**」→「**インポート**」→「**ログのインポート**」を選択します。
3. **ジャーナルのインポート**・ウィンドウで、**フォルダ**または **ZIP ファイル**をジャーナルのインポート元として選択します。
4. 使用する**フォルダ**または **ZIP ファイル**を指定します。
5. 「**OK**」をクリックします。

指定した**フォルダ**または **ZIP ファイル**が作業リポジトリにインポートされます。

警告: 同じ ID を持つリポジトリへのジャーナルのインポートは許可されません。

ログ・データのエクスポート

ログ・データのエクスポートでは、アーカイブのためにログ・ファイルをエクスポートできます。

ログをエクスポートする手順:

1. **オペレータ**を起動します。
2. メニュー・バーで、「**ファイル**」→「**ログのエクスポート**」を選択します。
3. **ログのインポート**・ウィンドウで、後述のエクスポート・パラメータを設定し、「**OK**」をクリックします。
4. 指定した場所にログ・データがエクスポートされます。

ログのエクスポート・パラメータ

オペレータでのログのエクスポートには、次のオプションがあります。

プロパティ	説明
ディレクトリへエクスポート	エクスポート・ファイルが作成されるディレクトリ
ZIP ファイルにエクスポート	このオプションを選択すると、すべてのログ・エクスポート・ファイルを含む単一の圧縮ファイルが作成されます。選択しない場合、ログ・エクスポート・ファイルのセットが作成されます。
ZIP ファイル名	圧縮されたエクスポート・ファイルの名前を指定します。
フィルタ	この一連のオプションにより、指定したパラメータに応じてエクスポートするログ・ファイルをフィルタできます。
FROM および TO	実行の日付: 日付の基準として「**FROM**」または「**TO**」（あるいはその両方）を指定します。
エージェント	セッションの実行に使用される**エージェント**。

コンテキスト	**コンテキスト:** セッションの実行コンテキスト。
セッションの状態	可能な状態は、完了、エラー、キュー、実行中、待機中および警告です。
ユーザー	セッションを起動した**ユーザー**。
セッション名	ワイルドカードとして%を使用します。つまり、DWH%と指定すると、名前がDWH で始まるすべてのセッションが一致します。
拡張オプション	この一連のオプションにより、出力ファイル形式をパラメータ化できます。
キャラクタ・セット	エクスポート・ファイルで指定されたエンコーディング。XML ファイル・ヘッダーの encoding パラメータの値です。 `<?xml version="1.0" encoding="ISO-8859-1"?>`
Java キャラクタ・セット	ファイル生成に使用される Java キャラクタ・セット。

注意: OdiExportLog ツールを使用してもログ・データをエクスポートできます。

トポロジ・マネージャ

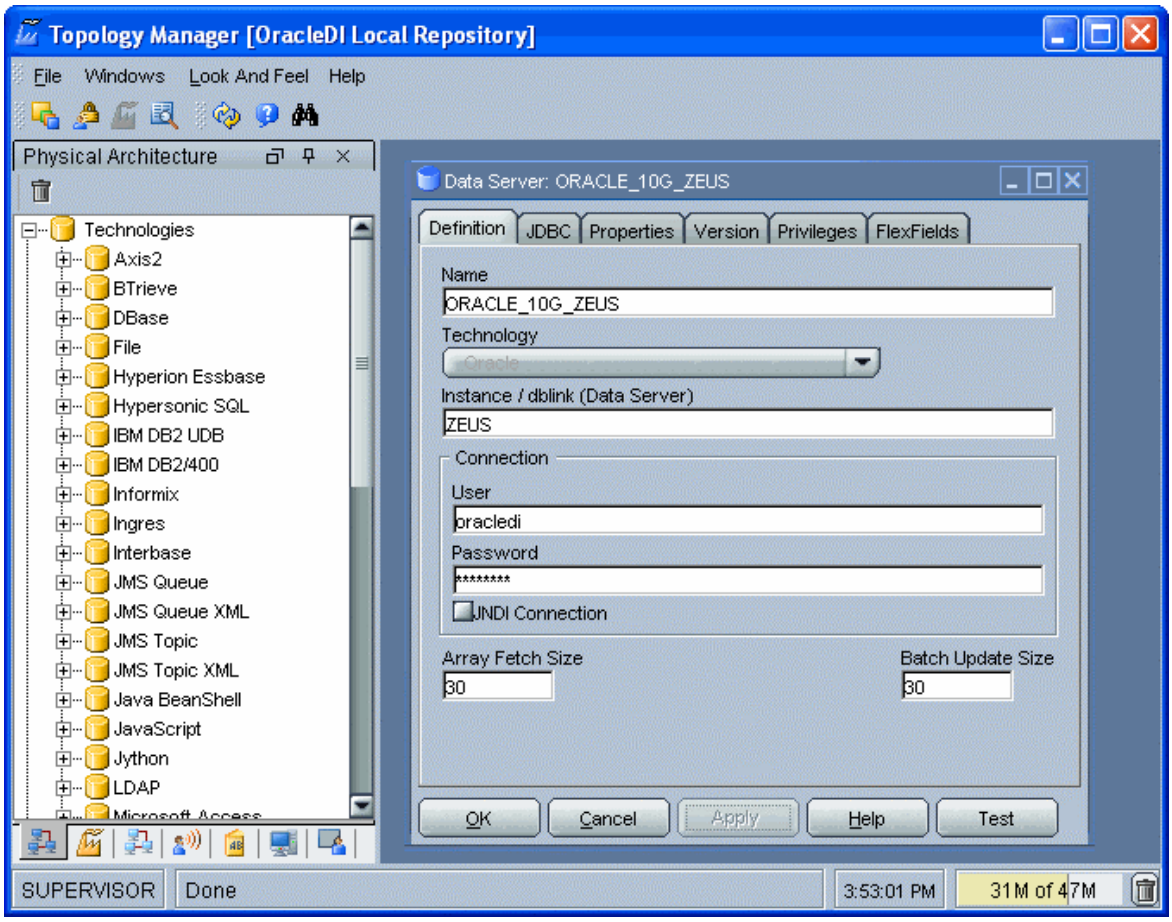
トポロジ・マネージャの概要

トポロジ・マネージャ・モジュールを使用すると、情報システムのトポロジ、**テクノロジ**とその**データ型**、その**テクノロジ**と含まれる**スキーマ**にリンクされた**データ・サーバー**、**コンテキスト**、**言語**および**エージェント**を管理できます。さらに、トポロジを使用すると**リポジトリ**も管理できます。

トポロジ・モジュールは、この情報をマスター・リポジトリに格納します。この情報は、他のすべてのモジュールで使用できます。

トポロジ・マネージャのインタフェース

トポロジ・マネージャの GUI は次のように表示されます。



メニュー

メニューでは、プルダウン・メニューから次の機能にアクセスできます。

- インポート/エクスポート
- ウィザード
- オプションの表示
- モジュールまたはツリー表示のオープン
- ユーザーのパスワードおよびオプションの変更

ツールバー

ツールバーからは、次の操作を実行できます。

- 他のモジュールの起動
- ツリー表示のリフレッシュ
- オンライン・ヘルプの起動

ツリー表示

現行ユーザーが使用できるトポロジ・マネージャ・オブジェクトは、次のようにツリー表示に整理して表示されます。

- **物理アーキテクチャ**。**テクノロジ**とそれに関連付けられた**データ・サーバー**と**物理スキーマ**、および**物理エージェント**が含まれます。
- **論理アーキテクチャ**。**テクノロジ**とそれに関連付けられた**論理スキーマ**、および**論理エージェント**が含まれます。
- 論理アーキテクチャと物理アーキテクチャをリンクする**コンテキスト**。
- **言語**。使用できる様々なタイプの言語が記述されます。
- **リポジトリ**。実際のマスター・リポジトリと、アタッチされた作業リポジトリが含まれます。

各ツリー表示は、メイン・ウィンドウの両側にドッキングできるフローティング・フレームに表示されます。フレームは重ねることもできます。複数のフレームを重ねた場合は、フレーム・ウィンドウの下部に表示されるタブから各フレームにアクセスできます。

ツリー表示フレームは、フレームのタイトルまたはタブを選択してドラッグすることにより、移動したり、ドッキングしたり、重ねたりできます。ツリー表示の位置を固定するには、**「ウィンドウ」**メニューから**「ウィンドウ・レイアウトのロック」**を選択します。

ツリー表示フレームがメイン・ウィンドウに表示されないか、閉じている場合は、**「ウィンドウ」→「ビューの表示」**メニューを使用します。

各ツリー表示では、次の操作が可能です。

- ルート・オブジェクトの挿入またはインポート（フレーム・タイトルで該当するボタンをクリック）
- ノードの展開または折りたたみ（ノードをクリック）
- オブジェクトに関連付けられているメソッド（編集、削除など）のアクティブ化（ポップアップ・メニューを使用）
- オブジェクトの編集（オブジェクトをダブルクリック、または**ワークベンチ**にドラッグ・アンド・ドロップ）

ワークベンチ

編集または表示されているオブジェクトのウィンドウは、**ワークベンチ**に表示されます。

トポロジの概要

トポロジ・マネージャ・モジュールを使用すると、Oracle Data Integrator を使用して作業するアーキテクチャやコンポーネントの正しい物理的および論理的な見取り図を思いのままに作成することができます。

- トポロジの作成

物理アーキテクチャ

物理アーキテクチャは、情報システムの様々な要素と、Oracle Data Integrator で考慮される特性を定義します。

テクノロジは書式付きデータを処理します。このため、各テクノロジは 1 つ以上のデータ型に関連付けられ、Oracle Data Integrator はそれを使用してデータ処理スクリプトを生成します。

注意: 各タイプのデータベース（Oracle、DB2 など）、ファイル形式（XML、ファイル）またはアプリケーション・ソフトウェアは、Oracle Data Integrator ではテクノロジによって表現されます。

データを格納して返す物理コンポーネントはデータ・サーバーと定義されます。ビジネス・ロジックに応じて異なった情報を格納できるデータ・サーバーは、いくつかの物理スキーマに分割できます。データ・サーバーは、常に単一のテクノロジにリンクされます。

注意: Data Integrator で使用されるすべてのデータベース・サーバー、JMS メッセージ・ファイル、フラット・ファイルのグループなどは、データ・サーバーとして宣言する必要があります。
Oracle Data Integrator で使用されるすべてのスキーマ、データベース、JMS トピックなどは、物理スキーマとして宣言する必要があります。

最後に、物理アーキテクチャには物理エージェントの定義が含まれます。これは、Oracle Data Integrator ジョブをリモート・マシン上で実行できるようにする Java ソフトウェア・コンポーネントです。

コンテキスト

コンテキストは、情報システムの物理アーキテクチャ（真のアーキテクチャ）のコンポーネントを、Oracle Data Integrator 論理アーキテクチャ（ユーザーが作業するアーキテクチャ）のコンポーネントに統合します。

論理アーキテクチャ

論理アーキテクチャは、ユーザーが、構造上同一であるが別個の場所にあるデータストアが含まれている物理スキーマを、テクノロジごとに構造化されたいくつかの論理スキーマにグループ化することを可能にします。

同じ考え方で、論理アーキテクチャは論理エージェントを定義します。これは、異なったコンテキストで同じ機能を果すすべての物理エージェントに一意の名前を付けられるようにします。

例: 次の 2 つの**物理スキーマ**に対して、**論理スキーマ** Accounting を対応付けることができます。
- 開発**コンテキスト**で使用される Accounting Oracle sample
- 本番**コンテキスト**で使用される Accounting corporate
これら 2 つの**物理スキーマ**は、構造的には同一（会計データを含む）ですが、物理的には異なります。これらの物理スキーマは、2 つの Oracle スキーマ上に存在し、通常は 2 つの異なる Oracle サーバー（データ・サーバー）に配置されます。

言語

このタイプのコンポーネントは、テクノロジにリンクされ、Oracle Data Integrator によって使用される各言語に固有の特性を定義します。

リポジトリ

トポロジのこの部分には、マスター・リポジトリと作業リポジトリの 2 タイプのリポジトリに関連する情報が含まれます。

ホスト

ホストと用例を使用すると、グラフィック・モジュールへのユーザー・アクセスを管理できます。

トポロジの定義

トポロジの作成

トポロジを作成する場合のガイドラインとなる手順は、次のとおりです。

1. 現在の構成に適したコンテキストを作成します。
2. Oracle Data Integrator で使用されるサーバーに対応するデータ・サーバーを作成します。
3. データ・サーバーごとに物理スキーマを作成します。
4. 論理スキーマを作成し、コンテキスト内の物理スキーマに関連付けます。
5. マシン上で（リスナーとして、またはスケジューラ・モードで）稼働するエージェントごとに物理エージェントを作成します。
6. 論理エージェントを作成し、コンテキスト内の物理エージェントに関連付けます。

コンテキストの作成

コンテキストを作成する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで**「トポロジ」→「コンテキスト」**を選択します。
3. 右クリックし、**「コンテキストの挿入」**を選択します。
4. 次のフィールドを入力します。
    - **名前:** Oracle Data Integrator のグラフィカル・インタフェースに表示されるコンテキスト名
    - **コード:** 異なるリポジトリ間でコンテキストを参照および識別するためのコンテキストのコード
    - **パスワード:** ユーザーがこのコンテキストを操作する際に要求されるパスワード
5. **「OK」**をクリックします。

データ・サーバーの作成

データ・サーバーは、コマンドを発行する目的で Oracle Data Integrator に接続されたデータベース・サーバー・インスタンス、JMS サーバー・インスタンス、スクリプト・エンジンなどに対応します。このデータ・サーバーに、**物理スキーマ**と呼ばれる下位区分が作成されます。

重要: 一般的に使用されるテクノロジには、独自のデータ・サーバー作成方法があり、詳細は「Oracle Data Integrator の使用」に記載されています。作業を進める前にこの項を参照してください。

前提条件

一部のテクノロジでは、次の要素のインストールと構成が必要です。

- JDBC ドライバのインストール
- クライアント・コネクタのインストール
- データソースの構成
- ...

詳細は、データ・サーバーを通じて接続するテクノロジのドキュメントを参照してください。接続情報は、テクノロジに応じて変化する可能性があります。付属のサーバー・ドキュメントを参照し、サーバー管理者に連絡して接続方法を定義してください。

データ・サーバーの作成

データ・サーバーを作成する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで**「トポロジ」→「物理アーキテクチャ」→「テクノロジ」**を選択し、データ・サーバーに関連付けるテクノロジを選択します。
3. 右クリックし、**「データ・サーバーの挿入」**を選択します。
4. **「定義」**タブで、次のフィールドを入力します。
   - **名前:** Oracle Data Integrator に表示されるデータ・サーバーの名前。
   - **...（データ・サーバー）:** データ・サーバーの物理名。データ・サーバーがネイティブな形式で相互接続される場合、この名前を入力します。このパラメータは、一部のテクノロジでのみ必須です。
   - **ユーザー/パスワード:** データ・サーバーに接続するためのユーザー名とパスワード。このパラメータは、一部のテクノロジでのみ必須です。
5. データ・サーバーの接続パラメータを定義します。
   - データ・サーバーで **JNDI** アクセスがサポートされる場合:
     1. **「JNDI 接続」**ボックスを選択します。
     2. **「JNDI」**タブに移動し、次のフィールドを入力します。
        - **JNDI 認証**
        - **JNDI ユーザー/パスワード:** JNDI ディレクトリに接続するためのユーザー名とパスワード
        - **JNDI プロトコル:** 接続で使用するプロトコル
        - **JNDI ドライバ:** JNDI 接続を使用するためのドライバ
        - **JNDI URL:** JNDI 接続を使用するための URL
        - **JNDI リソース:** 接続パラメータを含むディレクトリ要素
   - データ・サーバーで **JDBC** アクセスがサポートされる場合:

- 「**JNDI 接続**」ボックスを選択解除します。
- 「**JDBC**」タブに移動し、次のフィールドを入力します。
  - **JDBC ドライバ:** データ・サーバーに接続するための JDBC ドライバの名前
  - **JDBC URL:** データ・サーバーに接続するための URL

6. 「**テスト**」をクリックします。
7. **テスト接続**ウィンドウの「**テスト**」をクリックします。
8. **接続に成功**したことを示すウィンドウが表示されます。「**OK**」をクリックします。
9. 「**OK**」をクリックしてデータ・サーバーの作成を承認します。

このデータ・サーバーの最初の**物理スキーマ**に対応する作成ウィンドウが表示されます。

データ・サーバー接続のテスト

データ・サーバーとの接続をテストするには、次の手順を実行します。

1. テストする**データ・サーバー**のウィンドウを開きます。
2. 「**テスト**」ボタンをクリックします。
3. テストを実行する「**エージェント**」を選択します。「内部」は、ローカル・ステーションが接続を試行することを示します。
4. 「**詳細**」をクリックすると、JDBC ドライバの特性および機能を取得できます。
5. 「テスト」をクリックしてテストを開始します。

テストに成功すると、「接続に成功しました」というウィンドウが表示され、失敗するとエラー・ウィンドウが表示されます。このエラー・ウィンドウで「**詳細**」ボタンを使用すると、接続が失敗した原因の詳細情報を取得できます。

物理スキーマの作成

Oracle Data Integrator の**物理スキーマ**は、次のスキーマのペアに対応します。

- （データ）**スキーマ:** Oracle Data Integrator は、このスキーマでインタフェースのソースおよびターゲット・データ構造を検索します。
- **作業スキーマ:** Oracle Data Integrator は、このスキーマで、データ・スキーマに格納されたソースとターゲットに関連する一時作業データ構造を作成および操作します。

重要: 一般的に使用されるテクノロジには、独自の物理スキーマ作成方法があり、詳細は「Oracle Data Integrator の使用」に記載されています。作業を進める前にこの項を参照してください。

注意: すべてのテクノロジで複数のスキーマがサポートされるわけではありません。一部のテクノロジでは、1 つのデータ・サーバーに 1 つのスキーマのみが含まれるため、作業スキーマとデータ・スキーマを指定しません。

注意: 一部のテクノロジでは、一時構造の作成がサポートされません。これらのテクノロジでは、作業スキーマは使用されません。

注意: 物理スキーマに関連するデータ・サーバーに指定されたユーザーは、そのスキーマに対する適切な権限を付与されている必要があります。

物理スキーマの作成

物理スキーマを作成する手順:

注意: データ・サーバーの作成直後の場合、手順 1 は無視してください。**物理スキーマ**・ウィンドウがすでに表示されているはずです。

1. **データ・サーバー**を選択して右クリックし、**「物理スキーマの挿入」**を選択します。**物理スキーマ**・ウィンドウが表示されます。
2. テクノロジで複数のスキーマがサポートされる場合:
   1. **「...（スキーマ）」**で、この Data Integrator の物理スキーマのデータ・スキーマを選択または入力します。テクノロジでスキーマのリスト表示がサポートされる場合、スキーマのリストが表示されます。
   2. **「...（作業スキーマ）」**で、この Data Integrator の物理スキーマの作業スキーマを選択または入力します。テクノロジでスキーマのリスト表示がサポートされる場合、スキーマのリストが表示されます。
3. このスキーマをこのデータ・サーバーのデフォルト・スキーマに設定する場合、**「デフォルト」**ボックスを選択します（最初の物理スキーマは、常にデフォルト・スキーマになります）。詳細は、「物理スキーマ」を参照してください。
4. **「コンテキスト」**タブに移動します。
5. この新規**物理スキーマ**に対応する**「コンテキスト」**と既存の**「論理スキーマ」**を選択し、手順 9 に進みます。
   このテクノロジの論理スキーマが存在しない場合は、手順 7 に進みます。
6.  ボタンをクリックします。
7. 左側の列で既存の**「コンテキスト」**を選択し、右側の列に**「論理スキーマ」**の名前を入力します。この論理スキーマは、自動的に作成され、このコンテキスト内の物理スキーマに関連付けられます。

警告: 特定の**コンテキスト**では、**論理スキーマ**は 1 つの**物理スキーマ**にのみ関連付けることができます。

8. **「OK」**をクリックします。

論理スキーマの作成

論理スキーマを作成する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで**「トポロジ」→「論理アーキテクチャ」→「テクノロジ」**を選択し、スキーマに関連付けるテクノロジを選択します。
3. 右クリックし、**「論理スキーマの挿入」**を選択します。
4. スキーマの**「名前」**を入力します。
5. 左側の列の**「コンテキスト」**ごとに、右側の列で既存の**「物理スキーマ」**を選択します。物理スキーマは、自動的にこのコンテキスト内の論理スキーマに関連付けられます。必要なすべてのコンテキストでこの操作を繰り返します。
6. **「OK」**をクリックします。

物理エージェントの作成

このエージェントは、TCP/IP ポートのリスナーとして設定できる Java サービスです。

このサービスでは、次のことが可能です。

- Oracle Data Integrator モジュールから必要時にセッション（モデル・リバース、パッケージ、シナリオ、インタフェースなど）を実行できます。これを行うには、リスナー・エージェントを起動する必要があります。
- 必要時の実行に加え、スケジュール済のシナリオを実行できます。物理エージェントには、オプションの**スケジューラ**が含まれており、事前定義されたスケジュールに従って自動的にシナリオを起動できます。これを行うには、スケジューラ・エージェントを起動する必要があります。

物理エージェントでは、他の物理エージェントに実行を委任できます。このプロセスにより、エージェント間でのロード・バランシングが可能になります。

物理エージェントを作成する手順:

1. **トポロジ・マネージャ**に接続します。
2. 右**クリック**し、**「エージェントの挿入」**を選択します。
3. 次の**フィ**ールドを入力します。
    - **名前:** Java グラフィカル・インタフェースで使用されるエージェントの名前。
    - **ホスト:** エージェントが起動されるマシンのネットワーク名または IP アドレス。
    - **ポート:** エージェントで使用するリスニング・ポート。デフォルトのポートは 20910 です。
    - **セッションの最大数:** エージェントでサポートするセッションの最大数。
4. ロード・バランシングを設定する場合、**「ロード・バランシング」**タブに移動し、現在のエージェントの実行を委任する物理エージェントのセットを選択します。
5. エー**ジェント**を起動したら、**「テスト」**をクリックします。接続に成功したことを示すウィンドウが表示されます。
6. **「OK」**をクリックします。

論理エージェントの作成

論理エージェントを作成する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで**「トポロジ」→「論理アーキテクチャ」→「エージェント」**を選択します。
3. 右**クリック**し、**「論理エージェントの挿入」**を選択します。
4. エー**ジェントの「名前」を入力します。**
5. **左側の列の「コンテキスト」ごとに、右側の列で既存の「物理エージェント」を**選択します。物理エージェントは、自動的にこのコンテキスト内の論理エージェントに関連付けられます。必要なすべてのコンテキストでこの操作を繰り返します。
6. **「OK」**をクリックします。

データ型の自動リバース・エンジニアリング

データ型のリバース・エンジニアリングにより、データ型の情報を直接データ・サーバー・レベルで復元してマスター・リポジトリに入力する必要がなくなります。ただし、一部のデータ・サーバー・アクセス・ドライバでは、この機能を使用できません。

注意: この手順は、トポロジの新規テクノロジでのみ実行してください。

テクノロジのデータ型を自動的にリバース・エンジニアリングする手順:

1. テクノロジのデータ・サーバー接続を作成およびテストします。
2. **テクノロジ**・ウィンドウを開きます。
3. **「リバース」**ボタンをクリックします。リバース・エンジニアリングされた**データ型**がツリーのテクノロジの下に表示されます。
4. **データ型**ウィンドウの**「変換先」**タブで、各データ型を他のテクノロジに変換するための情報を入力します。

アクションの使用方法

概要

アクションは、データ定義言語（DDL）コマンドのテンプレートです。アクションは、共通フォーマット・デザイナが、データ・サーバーにデータ・モデルを実装するスクリプトや、Oracle Data Integrator で記述されたデータ・モデルとデータ・サーバーの実装の間に存在する差異を同期するスクリプトを生成する場合に使用されます。

アクションは、DDL 操作（表の作成、参照の削除など）に対応します。各アクションには、DDL 操作を実行するのに必要なコマンドに対応する複数の**アクション行**が含まれます（たとえば、表を削除する場合、最初にそのすべての制約を削除する必要があります。）。

アクションは、**アクション・グループ**に分類されます。アクション・グループは、Oracle や SQL-92 などの DDL コマンドの任意の構文または用途（あるいはその両方）に対応します。

汎用アクションとテクノロジ・アクションの比較

トポロジ・マネージャのアクションは、テクノロジに関連付けるか、**「一般アクション」**ビューで作成します。

汎用アクションは、デフォルト構文に対応します。汎用アクションは、SQL-92 構文をサポートするデータベースと互換性があります。特定のアクション・グループを持たないテクノロジの場合や、使用するアクション・グループから特定のアクションが欠落している場合は、汎用アクションを使用します。

アクションの作成

アクションを作成するには、最初にアクション・グループを作成する必要があります。

テクノロジのアクション・グループを作成する手順:

1. テクノロジを開き、**「アクション・グループ」**ノードをクリックします。右クリックし、**「アクション・グループの挿入」**を選択します。

2. アクション・グループの**「名前」**、**「コード」**および**「説明」**を入力します。
3. このテクノロジで作成する新規モデルに対してこのアクション・グループを選択する場合、**「デフォルト・グループ」**ボックスを選択します。特定のテクノロジに対してデフォルト・アクション・グループを選択しない場合、新規モジュールでは汎用アクションが使用されます。
4. **「OK」**をクリックしてアクション・グループを保存します。

アクション・グループでアクションを作成する手順:

1. アクション・グループを選択して右クリックし、**「アクションの挿入」**を選択します。
2. **「名前」**および**「説明」**フィールドを入力し、アクションの**「タイプ」**を選択します。
3. **「詳細」**タブに移動します。
4. **「追加」**アイコンを選択してアクション行を追加します。
   「削除」アイコンを使用して無効なアクション行を削除することや、**「上」**および**「下」**ボタンを使用してアクション行を再編成することができます。
5. 表示された**アクション行**ウィンドウで、次のフィールドを入力します。
   - **名前:** アクション行の名前。生成されるプロシージャ・コマンドにはこの名前が付けられます。
   - **エラーの無視:** このオプションは、特定のコマンドがエラーを戻してもプロシージャを停止しない場合に選択します。（たとえば、存在しない表の削除によりプロセス全体がブロックされることがなくなります。）
   - **コミット:** コマンドの最後にコミットを発行するかどうかを指定します。
   - **分離レベル:** コマンドのトランザクション分離レベル。
   - **アクション・テキスト:** 実行するコマンドのテキスト。をクリックすることで式エディタを表示できます。
6. **「OK」**をクリックしてアクション行を保存します。
7. 作成するアクション行ごとに手順 4 から 6 を繰り返します。
8. **「OK」**をクリックしてアクションを保存します。

注意: アクション行は、適切な置換メソッドを使用して記述する必要があります。詳細は、『Oracle Data Integrator メソッド・リファレンス』の「アクションでの置換メソッドの使用」を参照してください。

注意: 新規のアクションまたはグループを作成する場合、既存のアクションまたはグループをコピーし、それを要件に合せて変更することをお薦めします。この操作を実行するには、[Ctrl]キーを押しながらオブジェクトをドラッグ・アンド・ドロップします。

アクション・グループの更新

ナレッジ・モジュールと同様に、アクション・グループでもインポート置換機能を使用して更新を管理できます。

置換モードでアクション・グループをインポートする手順:

1. 置換する**アクション・グループ**を選択します。
2. **アクション・グループ**を右クリックし、**「置換のインポート」**を選択します。

3. エクスポート・ファイルを指定します。
4. **「OK」**をクリックします。

アクション・グループが新規アクション・グループで置換されます。更新されたグループで DDL スクリプトを再生成する必要があります。

アクションの DDL スクリプトへの変換

DDL スクリプトは、DDL スクリプト生成の構成時に選択された様々なオプションに基づいてアクションから生成されます。DDL スクリプトは、アクション行に対応する各プロシージャ行を含む Oracle Data Integrator プロシージャの形式で生成されます。アクション行は、選択された変更操作を不要なコマンドの繰返しを避けて実行するアルゴリズムを使用して、プロシージャ内で順序付けされます。

DDL スクリプトでのアクションの順序付け

アクションは、通常、生成されたスクリプト内で次の順序で出現します。

1. 開始アクション
2. 削除アクション
3. 変更アクション: 変更、有効化、無効化、名前変更
4. 追加（および作成）アクション
5. 不明なアクション
6. 終了アクション

リポジトリの管理

マスター・リポジトリへの接続

マスター・リポジトリに接続する手順:

1. **「スタート」**メニューで**「プログラム」→「Oracle Data Integrator」→「Topology Manager」**を選択するか、**トポロジ・マネージャ**のスクリプト（`bin/topology.bat` または `bin/topology.sh`）を起動します。
2. **「新規」**ボタン（**「ログイン名」**フィールドの右側にある最初のボタン）をクリックします。
3. 次のフィールドを入力します。

   Oracle Data Integrator 接続:

   - **ログイン名:** 一般的な別名（Repository など）
   - **ユーザー:** SUPERVISOR（大文字を使用）
   - **パスワード:** SUNOPSIS（大文字を使用）

   DBMS 接続(マスター・リポジトリ):

   - **ユーザー:** マスター・リポジトリに対して作成した表の所有者のユーザーID またはログイン。
   - **パスワード:** このユーザーのパスワード。

- **ドライバ・リスト:** 作成したマスター・リポジトリをサポートする DBMS に接続する際に必要なドライバを選択します。
- **URL:** リポジトリをホストするデータ・サーバーの完全なパス。詳細は、「JDBC URL のサンプル」を参照してください。

4. 「**テスト**」をクリックして接続動作を確認します。
5. 「**OK**」をクリックして設定を適用し、「**OK**」をクリックします。**トポロジ・マネージャ**が起動します。

重要: SUPERVISOR アカウントのデフォルト・パスワードは、SUNOPSIS です。セキュリティ上問題となるため、このパスワードはできるだけ早く変更してください。

マスター・リポジトリの作成

マスター・リポジトリを作成する作業は、表を作成する作業と、各種のテクノロジの定義を自動的にインポートする作業で構成されます。

マスター・リポジトリを作成する手順:

1. 「**スタート**」メニューで「**プログラム**」→「**Oracle Data Integrator**」→「**Repository Management**」→「**Master Repository Creation**」を選択するか、`bin/repcreate.bat` または `bin/repcreate.sh` を起動します。
2. 次のフィールドを入力します。
   - **ドライバ:** リポジトリをホストするテクノロジにアクセスするためのドライバ。詳細は、「JDBC URL のサンプル」を参照してください。
   - **URL:** リポジトリをホストするデータ・サーバーの完全なパス。詳細は、「JDBC URL のサンプル」を参照してください。
   - **ユーザー:** 表の所有者のユーザーID またはログイン（**snpm** という名前で作成済）。
   - **パスワード:** このユーザーのパスワード。
   - **ID:** デフォルトの 0 とは異なる新規リポジトリの特定の ID。これにより、リポジトリ間のインポートおよびエクスポートが影響を受けます。
   - **テクノロジ:** リポジトリの基礎となるテクノロジをリストから選択します。
   - **言語:** マスター・リポジトリの言語を選択します。
3. 「**OK**」をクリックして設定を適用します。

ディクショナリの作成が開始されます。コンソールに進行状況が表示されます。マスター・リポジトリをテストするには、「マスター・リポジトリへの接続」を参照してください。

作業リポジトリの作成

必要に応じて、複数のマスター・リポジトリに対して複数の作業リポジトリを指定できます。ただし、バージョン管理上の目的のため、1 つの作業リポジトリは 1 つのマスター・リポジトリにのみリンクできます。

作業リポジトリの詳細は、リファレンス・マニュアルを参照してください。

作業リポジトリを作成する手順:

1. **トポロジ**・モジュールを通じてマスター・リポジトリに接続します。詳細は、「マスター・リポジトリへの接続」を参照してください。
2. アイコン・リストで**「トポロジ」→「リポジトリ」→「作業リポジトリ」**を選択して右クリックし、**「作業リポジトリの挿入」**を選択します。作業リポジトリの接続パラメータを指定するためのウィンドウが表示されます。
3. **接続**ウィンドウで、次のパラメータを指定します。
   - **名前:** 作業リポジトリの接続名を入力します。
   - **テクノロジ:** 作業リポジトリをホストするサーバーのテクノロジを選択します。
   - **ユーザー:** 作業リポジトリに対して作成およびホストする表の所有者のユーザーID またはログイン。
   - **パスワード:** このユーザーのパスワード。
   - **「JDBC」タブ→「JDBC ドライバ」:** 作業リポジトリをホストする DBMS に接続するためのドライバ。詳細は、「JDBC URL のサンプル」を参照してください。
   - **「JDBC」タブ→「URL JDBC」:** 作業リポジトリをホストするデータ・サーバーの完全なパス。詳細は、「JDBC URL のサンプル」を参照してください。
4. **「テスト」**をクリックします。

   注意: 接続テストが適切に完了しなかった場合、**「OK」**でこのウィンドウを閉じないでください。

5. **「OK」**をクリックして、作業リポジトリをホストするサーバーに接続するためのパラメータを適用します。リポジトリに一意の名前とユーザーID コード番号を指定するためのウィンドウが表示されます。
6. **作業リポジトリ**・ウィンドウで、次のパラメータを指定します。
   - **ID:** 1 から 998 までの一意の番号をリポジトリに指定します。
   - **名前:** 作業リポジトリに一意の名前を指定します（WorkRep1 など）。
   - **タイプ:** リストの「デザイナ」を選択します。
7. **「OK」**をクリックして設定を適用します。作業リポジトリの作成が開始され、コンソールに様々な作業ステップが表示されます。
8. 作業リポジトリの作成が完了すると、作業リポジトリ・ウィンドウは閉じます。これで、**デザイナ**・モジュールおよび**オペレータ**・モジュールを通じてこのリポジトリにアクセスできます。詳細は、「作業リポジトリへの接続」を参照してください。

マスター・リポジトリのエクスポート

マスター・リポジトリのインポートおよびエクスポートの手順により、（トポロジおよびセキュリティの両ドメインに関連する）リポジトリ全体をあるリポジトリから別のリポジトリに移動できます。

マスター・リポジトリをエクスポートする手順:

1. トポロジ・マネージャで、**「ファイル」→「エクスポート」→「マスター・リポジトリ」**をクリックします。
2. 次のエクスポート・パラメータを入力します。

- **ディレクトリへエクスポート:** エクスポート・ファイルが作成されるディレクトリ。
- **ZIP ファイルにエクスポート:** このオプションを選択すると、すべてのエクスポート・ファイルを含む単一の圧縮ファイルが作成されます。選択しない場合、エクスポート・ファイルのセットが作成されます。
- **ZIP ファイル名:** 圧縮ファイルの名前（**「圧縮されたエクスポートを作成」**オプションを選択している場合）。
- **キャラクタ・セット:** エクスポート・ファイルに指定するエンコーディング。XML ファイル・ヘッダーの `encoding` パラメータの値です。
- **Java キャラクタ・セット:** ファイル生成に使用する Java キャラクタ・セット。
- **バージョンのエクスポート:** リポジトリに格納されたオブジェクトのすべての格納済バージョンをエクスポートします。エクスポートされるリポジトリのサイズを削減し、無関係なプロジェクト作業の転送を避けるには、このオプションを選択解除します。
- **ソリューションのエクスポート:** リポジトリに格納されたすべての格納済ソリューションをエクスポートします。同じ理由により、このオプションを選択解除できます。

3. **「OK」**をクリックします。

指定したエクスポート・ディレクトリにエクスポート・ファイルが作成されます。

関連項目:

マスター・リポジトリのインポート

トポロジまたはセキュリティ設定のエクスポート

トポロジまたはセキュリティ設定のインポート

マスター・リポジトリのインポート

マスター・リポジトリのインポートおよびエクスポートの手順により、（トポロジおよびセキュリティの両ドメインに関連する）リポジトリ全体をあるリポジトリから別のリポジトリに移動できます。

インポート操作は、エクスポートされたオブジェクトを既存のリポジトリにインポートするために、または新規マスター・リポジトリの作成時にトポロジで実行できます。

既存のマスター・リポジトリにマスター・リポジトリをインポートする手順:

1. トポロジ・マネージャで、**「ファイル」→「インポート」→「マスター・リポジトリ」**をクリックします。
2. **インポート・モード**と、インポート用の**「フォルダ」**または**「ZIP ファイル」**を選択し、**「OK」**をクリックします。

指定したファイルが現在のマスター・リポジトリにインポートされます。

エクスポートを使用して新規マスター・リポジトリを作成する手順:

1. **「スタート」**メニューで**「プログラム」→「Oracle Data Integrator」→「Repository Management」→「Master Repository Import」**を選択するか、`bin/mimport.bat` または `bin/mimport.sh` を起動します。
2. 次のフィールドを入力します。
   - **ドライバ:** リポジトリをホストするテクノロジにアクセスするためのドライバ。詳細は、「JDBC URL のサンプル」を参照してください。

- **URL:** リポジトリをホストするデータ・サーバーの完全なパス。詳細は、「JDBC URL のサンプル」を参照してください。
- **ユーザー:** 表の所有者のユーザーID またはログイン（**snpm** という名前で作成済）。
- **パスワード:** このユーザーのパスワード。

3. **「テスト」**ボタンをクリックして接続をテストします。接続に成功する必要があります。
4. 次のフィールドを入力します。

- **識別子:** 新規マスター・リポジトリの数値識別子。

警告: すべてのマスター・リポジトリに異なる識別子を割り当てる必要があります。選択する識別子が既存のリポジトリの識別子と異なることを確認してください。

- **テクノロジ:** リポジトリの基礎となるテクノロジをリストから選択します。
- **言語:** マスター・リポジトリの言語を選択します。

5. 圧縮エクスポート・ファイルを使用する場合、**「ZIP ファイルを使用」**ボックスを選択し、マスター・リポジトリのエクスポートを含むファイルを選択します。圧縮されていないエクスポートを使用する場合、エクスポートを含むディレクトリを選択します。
6. **「OK」**をクリックします。

新規リポジトリが作成され、エクスポート済のコンポーネントがこのマスター・リポジトリにインポートされます。

関連項目:

マスター・リポジトリのエクスポート

トポロジまたはセキュリティ設定のエクスポート

トポロジまたはセキュリティ設定のインポート

インポート・モード

作業リポジトリのパスワードの変更

作業リポジトリのパスワードを変更する手順:

1. **トポロジ・マネージャ**の**「リポジトリ」**ビューで、作業リポジトリをダブルクリックします。リポジトリ編集ウィンドウが起動します。
2. **「パスワードの変更」**ボタンをクリックします。
3. 旧パスワードと新パスワードを入力します。
4. **「OK」**ボタンをクリックします。

ホストおよび使用環境の管理

Oracle Data Integrator では、複数のモジュール（デザイナ、トポロジなど）にアクセスできます。ホストと使用環境により、これらのモジュールへのアクセスを管理します。

定義

Oracle Data Integrator において:

- Oracle Data Integrator モジュールが稼働するマシンは、**ホスト**と呼ばれます。
- このマシンで使用される特定のモジュールは、**使用環境**と呼ばれます。

1 つのホストに複数の使用環境が含まれることもあります。たとえば、管理者のマシンにはトポロジおよびセキュリティという使用環境が含まれるのが普通です。

使用環境

使用環境により、Oracle Data Integrator に付属するモジュールの 1 つをホストで使用できるかどうかが決定されます。

使用環境には、次の 3 つの使用環境タイプがあります。

- **常に許可:** ホストでは常に特定のモジュールを使用できます。
- **拒否:** ホストでは特定のモジュールを使用できません。
- **自動**（デフォルト動作）: ホストでは、リポジトリにモジュールで接続する際に、自動的にその特定のモジュールにアクセスします。

同じホストの同じモジュールに 2 つの使用環境を割り当てることはできません。

デフォルト動作

新規マシンでモジュールをリポジトリに接続すると、そのマシンを識別するホストがホスト・リストに自動的に追加され、起動されたモジュールの使用環境（自動）がホストの使用環境リストに追加されます。

ホストおよび使用環境の管理

管理者は、手動でホストを追加および削除できます。また、使用環境を追加、編集および削除することで、特定のマシンにおける特定のモジュールの使用を許可または禁止できます。

たとえば、管理者は、開発者のマシンでそのホストの使用環境を「拒否」に設定することで、トポロジ・モジュールの使用を禁止できます。

重要: ホストおよび使用環境の削除操作は制限されています。削除するホストまたは使用環境ごとに、テクニカル・サポートから提供される特定の**削除キー**が必要です。

ホストおよび使用環境の作成と編集

手動でホストを作成する手順:

重要: 作成後にホストのパラメータを編集することはできません。間違ったホスト作成を避けるため、（特定のホストからモジュールを使用してリポジトリに接続する）自動ホスト作成機能を使用することを強くお薦めします。自動の使用環境でこれらのホストを作成した後に、各使用環境を編集してください。

1. **トポロジ・マネージャ**に接続します。
2. ツリーで**「トポロジ」→「ホスト」**を選択します。
3. 右クリックし、**「ホストの挿入」**を選択します。
4. **「定義」**タブで、次のフィールドを入力します。

- **マシン名:** マシンを識別する一意の名前。通常は、マシンのネットワーク名です。
- **IP アドレス:** マシンおよびリポジトリ・サーバーに共通のネットワーク上でのマシンの IP アドレス。

5. 「**OK**」をクリックします。

ホストの使用環境を編集する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで「**トポロジ**」→「**ホスト**」を選択します。編集するホストをクリックします。
3. 右クリックして「**編集**」を選択します。
4. 「**使用方法**」タブに移動します。
5. ボタンをクリックして使用環境を追加します。
6. 使用環境ごとに「**モジュール**」と「**使用方法**」を選択します。
7. 「**OK**」をクリックします。

ホストおよび使用環境の削除

ホストを削除する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで「**トポロジ**」→「**ホスト**」を選択します。編集するホストをクリックします。
3. 右クリックし、「**削除**」を選択します。
4. **削除キー**を求めるウィンドウが表示されます。
5. 「**ファイル・ロック解除の作成**」ボタンをクリックします。
6. 場所を選択してロック解除ファイルの名前を指定し、「**OK**」をクリックします。
7. ロック解除ファイル（.duk ファイル）をテクニカル・サポートに提出します。**削除キー**を入手する必要があります。
8. 入手したキーを「**削除キー**」フィールドに入力します。
9. 「**OK**」をクリックします。

選択したホストがホスト・リストから削除されます。

使用環境を削除する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで「**トポロジ**」→「**ホスト**」を選択します。編集するホストをクリックします。
3. 右クリックして「**編集**」を選択します。
4. 「**使用方法**」タブに移動します。
5. 使用環境を選択し、ボタンをクリックします。
6. **削除キー**を求めるウィンドウが表示されます。
7. 「**ファイル・ロック解除の作成**」ボタンをクリックします。
8. 場所を選択してロック解除ファイルの名前を指定し、「**OK**」をクリックします。
9. ロック解除ファイル（.duk ファイル）をテクニカル・サポートに提出します。**削除キー**を入手する必要があります。

10. 入手したキーを**「削除キー」**フィールドに入力します。
11. **「OK」**をクリックします。

選択した使用環境がホストの使用環境から削除されます。

ホストのエクスポート

ホストは、マスター・リポジトリをエクスポートする場合、またはトポロジをエクスポートする場合にエクスポートされます。

エージェントの管理

リスナー・エージェントの起動

このエージェントは、TCP/IP ポートのリスナーとして設定できる Java TCP/IP サービスです。

このエージェントは、次のように動作します。

- 起動すると、ポート上のリスナーとして稼働します。
- 実行問合せがエージェントに到着すると、エージェントはその問合せに関連するタスクを実行し、ポート上のリスナーとして結果を戻します。

注意: リスナー・エージェントにはスケジュールされた実行が存在しないため、スケジュール済のシナリオは処理されません。

リスナー・エージェントを起動する手順:

1. UNIX シェル、DOS または QSH（AS/400）セッションを開始します。
2. 必要なパラメータ付きで適切なコマンド・ファイル（Windows の場合は agent.bat、UNIX または AS/400-QSH の場合は agent.sh）を実行します。

エージェントがリスナーとして起動します。

例:

- Windows の場合: `agent "-port=20300" "-v=5"`と入力すると、コンソールにレベル 5 のトレースを出力する設定でリスナー・エージェントがポート 20300 上に起動します。
- UNIX の場合: `./agent.sh -name=AGENTNT` と入力すると、AGENTNT という名前のリスナー・エージェントが起動します。

エージェント・パラメータ

次の表に、エージェントを構成するための様々なパラメータをリストします。パラメータの前にはハイフン（-）を付け、可能な値の前には等号（=）を付けます。この場合、コマンドを入力するオペレーティング・システムに固有の文字列区切りの構文に準拠する必要があります。

パラメータ	説明
`-port=<port>`	エージェントがリスニングするポート。このパラメータが指定されない場合、エージェントはデフォルト・ポートの 20910 でリスナーとして稼働しま

	す。
-help	エージェントを起動せずに、使用可能なオプションとパラメータを表示します。
-name=<agent name>	使用する物理エージェントの名前。 Oracle Data Integrator では、セッションを実行するエージェントを識別するために次の場合にこのパラメータを必要とします。 • 複数のエージェントが同じマシン（の異なるポート）に宣言されている場合。 • マシンの IP 構成が不十分で、Oracle Data Integrator でエージェントを識別できない場合。この問題は、AS/400 で多く発生します。 • エージェントの IP アドレス（127.0.0.1 つまりループバック）でエージェントを識別できない場合。
-v=<trace level>	エージェントでトレースを生成できます（冗長モード）。次の 5 つのトレース・レベルがあります。 1. 各セッションの開始と終了を表示 2. レベル 1 に加え、各ステップの開始と終了を表示 3. レベル 2 に加え、実行された各タスクを表示 4. 実行された SQL 問合せを表示 5. 完全トレース（通常はサポート用） 一部のトレースは大量となる可能性があるため、次のコマンドを使用してその内容をテキスト・ファイルにリダイレクトすることをお薦めします。 • Windows の場合: `agent.bat "-v=5" > trace.txt` • UNIX の場合: `agent.sh -v=5 > trace.txt`

関連項目:

- スケジューラ・エージェントの起動
- Web エージェントの起動
- デザイナまたはオペレータからのシナリオの実行
- OS コマンドからのシナリオの実行
- エージェントの停止

スケジューラ・エージェントの起動

このエージェントは、事前定義されたスケジュールを通じて、または必要時にシナリオを実行するために使用できる Java TCP/IP サービスです。

このエージェントは、次のように動作します。

- 起動すると、作業リポジトリに接続してスケジュール済の実行（エージェント用に計画されたすべてのシナリオ・スケジュール）を取得し、ポート上のリスナーとして稼働します。
- 実行問合せがエージェントに到着すると、エージェントはその問合せに関連するタスクを実行します。

- メモリー内のスケジュール済の実行に従ってシナリオを起動する必要がある場合、エージェントはそのシナリオを実行し、ポート上のリスナーとして結果を戻します。
- エージェントのスケジュール済の実行は、**物理エージェント**・ウィンドウの**「スケジュールの更新」**ボタンをクリックすることでいつでも更新できます。

注意: スケジューラ・エージェントには、リスナー・エージェントのすべての機能が含まれます。

スケジューラ・エージェントを起動する手順:

1. UNIX シェル、DOS または QSH（AS/400）セッションを開始します。
2. 必要なパラメータ付きで適切なコマンド・ファイル（Windows NT の場合は agentscheduler.bat、UNIX または AS/400-QSH の場合は agentscheduler.sh）を実行します。

エージェントが起動し、作業リポジトリのスケジュール済の実行が取得されます。

例:

- Windows の場合: `agentscheduler "-port=20300" "-v=5"`
- UNIX の場合: `agentscheduler -name=AGENTNT`

エージェント・パラメータ

注意: スケジューラ・エージェントを起動するには、スケジュールを含む作業リポジトリに接続するためのパラメータを指定する必要があります。これらのパラメータは、通常、odiparams ファイルに指定します。

次の表に、必要に応じてエージェントを構成するための様々なパラメータをリストします。パラメータの前にはハイフン（-）を付け、可能な値の前には等号（=）を付けます。この場合、コマンドを入力するオペレーティング・システムに固有の文字列区切りの構文に準拠する必要があります。

パラメータ	説明
`-port=<port>`	エージェントがリスニングするポート。このパラメータが指定されない場合、エージェントはデフォルト・ポートの 20910 でリスナーとして稼働します。
`-help`	エージェントを起動せずに、使用可能なオプションとパラメータを表示します。
`-name=<agent name>`	使用する物理エージェントの名前。 Oracle Data Integrator では、セッションを実行するエージェントを識別するために次の場合にこのパラメータを必要とします。 • 複数のエージェントが同じマシン（の異なるポート）に宣言されている場合。 • マシンの IP 構成が不十分で、Oracle Data Integrator でエージェントを識別できない場合。この問題は、AS/400 で多く発生します。 • エージェントの IP アドレス（127.0.0.1 つまりループバッ

	ク）でエージェントを識別できない場合。
-v=<trace level>	エージェントでトレースを生成できます（冗長モード）。次の 5 つのトレース・レベルがあります。 1. 各セッションの開始と終了を表示 2. レベル 1 に加え、各ステップの開始と終了を表示 3. レベル 2 に加え、実行された各タスクを表示 4. 実行された SQL 問合せを表示 5. 完全トレース（通常はサポート用） 一部のトレースは大量となる可能性があるため、次のコマンドを使用してその内容をテキスト・ファイルにリダイレクトすることをお薦めします。 • Windows の場合: `agent.bat "-v=5" > trace.txt` • UNIX の場合:`agent.sh -v=5 > trace.txt`
作業リポジトリ接続パラメータ	**これらのパラメータは、作業リポジトリへの接続を指定するために使用し、odiparams ファイルに指定します。**
-SECU_DRIVER=<driver name>	マスター・リポジトリにアクセスするための JDBC ドライバ。 例:`oracle.jdbc.driver.OracleDriver`
-SECU_URL=<url>	マスター・リポジトリにアクセスするための JDBC URL。 例:`jdbc:oracle:thin:@168.67.0.100:1522:ORCL`
-SECU_USER=<user>	マスター・リポジトリに接続するためのユーザー（データベース・ユーザー）。
-SECU_PASS=<password>	マスター・リポジトリに接続するユーザーのパスワード。このパスワードは、コマンド `agent ENCODE <password>`を使用して暗号化する必要があります。
-WORK_REPOSITORY=<work repository name>	実行対象のシナリオを含む作業リポジトリの名前。

関連項目:

- リスナー・エージェントの起動
- Web エージェントの起動
- デザイナまたはオペレータからのシナリオの実行
- OS コマンドからのシナリオの実行
- スケジュール情報の表示
- エージェントの停止

スケジュール情報の表示

スケジュール情報により、エージェントのスケジュール済タスクを一覧表示できます。

重要: スケジュール情報は、エージェントのスケジュールから取得されます。正確なスケジュール情報を表示するには、エージェントを起動してスケジュールをリフレッシュする必要があります。

オペレータからスケジュール情報を表示する手順:

1. **オペレータ**を起動します。
2. ツールバーの**「スケジューリング情報」**ボタンをクリックします。

すべてのエージェント用の**スケジューリング情報**ウィンドウが表示されます。

トポロジ・マネージャからスケジュール情報を表示する手順:

1. **トポロジ・マネージャ**を起動します。
2. **「物理アーキテクチャ」**ビューで、計画を表示する**物理エージェント**を選択します。
3. 右クリックし、**「スケジューリング情報」**を選択します。

そのエージェント用の**スケジューリング情報**ウィンドウが表示されます。

Web エージェントの起動

Web エージェントは、Web インタフェースを通じてシナリオを実行するために使用できる Java TCP/IP サービスです。

このエージェントは、次のように動作します。

- 起動すると、このエージェントは HTTP サーバーを開始します。
- この HTTP サーバーは、`/bin/web` ディレクトリに保存された Web ページを公開します（このディレクトリにはシナリオ起動アプレットを格納できます）。
- ユーザーがシナリオ起動アプレットを通じてシナリオを起動すると、エージェントはシナリオを実行してその結果をアプレットに渡します。

注意: Web エージェントには、スケジューラ・エージェントとリスナー・エージェントのすべての機能が含まれます。

Web エージェントを起動する手順:

1. UNIX シェル、Windows コマンドラインまたは QSH（AS/400）セッションを開始します。
2. 必要なパラメータ付きで適切なコマンド・ファイル（Windows の場合は agentweb.bat、UNIX または AS/400-QSH の場合は agentweb.sh）を実行します。

エージェントが起動します。

例:

- Windows の場合: `agentweb.bat "-port=20300" "-WEB_PORT=8080"`
- UNIX の場合: `agentweb.sh`

エージェント・パラメータ

注意: Web エージェントを起動するには、シナリオが格納されている作業リポジトリに接続するためのパラメータを指定する必要があります。これらのパラメータは、通常、odiparams ファイルに保存します。

次の表に、必要に応じてエージェントを構成するための様々なパラメータをリストします。パラメータの前にはハイフン（-）を付け、可能な値の前には等号（=）を付けます。この場合、使用するコマンドラインに固有の文字列区切りの構文に準拠する必要があります。

パラメータ	説明
`-port=<port>`	エージェントがリスニングするポート。このパラメータが指定されない場合、エージェントはデフォルト・ポートの 20910 でリスニングします。
`-help`	エージェントを起動せずに、使用可能なオプションとパラメータを表示します。
`-name=<agent name>`	使用する物理エージェントの名前。 Oracle Data Integrator では、特定のセッションを実行したエージェントを識別するために次の場合にこのパラメータを必要とします。 • 複数のエージェントが同じマシン（の異なるポート）に宣言されている場合。 • マシンの IP 構成が原因で Oracle Data Integrator でエージェントを識別できない場合。この問題は、AS/400 で多く発生します。 • エージェントの IP アドレス（127.0.0.1 つまりループバック）でエージェントを識別できない場合。
`-v=<trace level>`	エージェントの冗長性のレベルを指定します。次の 5 つのトレース・レベルがあります。 1. 各セッションの開始と終了を表示 2. 各ステップの開始と終了も表示 3. 実行された各タスクも表示 4. 実行された各 SQL 問合せも表示 5. すべての情報を表示（通常はサポート連絡用） 一部のレベルでは大量の出力が生成される可能性があるため、次のコマンドを使用してその内容をテキスト・ファイルにリダイレクトすることをお薦めします。 • Windows の場合: `agentweb.bat "-v=5" > trace.txt` • UNIX の場合:`agentweb.sh -v=5 > trace.txt`
`-WEB_PORT=<http port>`	エージェントが Web リクエストをリスニングする HTTP ポー

	ト。
作業リポジトリ接続パラメータ	**これらのパラメータは、作業リポジトリへの接続方法を指定するために使用し、odiparams ファイルに指定します。**
-SECU_DRIVER=<driver name>	マスター・リポジトリにアクセスするための JDBC ドライバ。 例:oracle.jdbc.driver.OracleDriver
-SECU_URL=<url>	マスター・リポジトリにアクセスするための JDBC URL。 例:jdbc:oracle:thin:@168.67.0.100:1522:ORCL
-SECU_USER=<user>	マスター・リポジトリに接続するためのデータベース・ユーザー。
-SECU_PASS=<password>	マスター・リポジトリに接続するユーザーのパスワード。このパスワードは、コマンド agent ENCODE <password>を使用して暗号化する必要があります。
-WORK_REPOSITORY=<work repository name>	実行対象のシナリオを含む作業リポジトリの名前。

関連項目:

- リスナー・エージェントの起動
- スケジューラ・エージェントの起動
- デザイナまたはオペレータからのシナリオの実行
- OS コマンドからのシナリオの実行
- エージェントの停止

エージェントの停止

TCP/IP ポートでリスニングしているリスナー、スケジューラまたは Web エージェントは、agentstop コマンドを使用して停止できます。

エージェントを停止する手順:

1. UNIX シェル、CMD または QSH（AS/400）セッションを開始します。
2. 必要なパラメータ付きで適切なコマンド・ファイル（Windows の場合は agentstop.bat、UNIX または AS/400-QSH の場合は agentstop.sh）を実行します。

リスニング中のエージェントが停止します。

重要: セキュリティ上の理由から、エージェントを停止できるのは、エージェントのプロセスが開始されたのと同じマシン上でコマンドラインが実行された場合のみです。
リモート・エージェントは停止できません。

例:

- Windows の場合: agentstop "-PORT=20300"と入力すると、ポート 20300 のリスナー・エージェントが停止します。

- UNIX の場合: `./agentstop.sh` と入力すると、デフォルト・ポートのリスナー・エージェントが停止します。

コマンド・パラメータ

次の表に、このコマンドの様々なパラメータを示します。パラメータの前にはハイフン（-）を付け、可能な値の前には等号（=）を付けます。この場合、関連するオペレーティング・システムで使用される文字列区切りの構文に準拠する必要があります。

パラメータ	説明
-PORT=<port>	停止するエージェントがリスニングしているポート。このパラメータが指定されない場合、デフォルト・ポートの 20910 が使用されます。

関連項目:

- リスナー・エージェントの起動
- スケジューラ・エージェントの起動
- Web エージェントの起動

ロード・バランシング

Oracle Data Integrator では、物理エージェント間のロード・バランシングを実装しています。

概要

各物理エージェントは、次の特性とともに定義されます。

- 同時に実行できる**セッションの最大数**
- セッションの実行を委任できる**リンクされた**物理エージェントの数（オプション）

エージェントの負荷は、そのエージェントにおける任意の時点での割合（実行中のセッション数/最大セッション数）により決定されます。

最大セッション数の決定

最大セッション数は、エージェントが稼働するマシンの性能に応じて設定する必要のある値です。また、この値は、ユーザーが Data Integrator エージェントに付与する処理性能の程度に応じて設定することもできます。

セッションの委任

リンク・エージェント付きのエージェントでセッションが開始されると、Oracle Data Integrator により、負荷のより少ないリンク・エージェントが特定され、そのリンク・エージェントにセッションが委任されます。

ユーザー・パラメータの「新しいロード・バランシングの使用」を使用している場合、1 つのセッションが終了するたびに残りのセッションが再分散されます。つまり、処理するセッションがなくなったエージェントには、通常、別のエージェントからセッションが再度割り当てられます。

注意: エージェントは、自分自身にリンクできます。自分自身にリンクしないエージェントは、リンク・エージェントにセッションを委任できるだけであり、セッションを実行することはありません。

注意: 委任は、カスケード状のリンク・エージェントで機能します。さらに、エージェント・リンクのループを作成することも可能です。このオプションはお薦めしません。

使用不可能なエージェント

特定のエージェントで、実行中のセッション数がその**最大セッション数**に等しくなると、エージェントは、実行中のセッション数がそのエージェントの最大セッション数を下回るまで着信セッションをキュー・ステータスに設定します。

ロード・バランシングの設定

ロード・バランシングを設定する手順:

1. 物理エージェントのセットを定義し、それらをルート・エージェントにリンクします（「物理エージェントの作成」を参照）。
2. ルート・エージェントとリンク・エージェントを起動します。
3. ルート・エージェントで実行を開始します。Oracle Data Integrator により、リンク・エージェント間で実行負荷が分散されます。

注意: セッションの実行エージェントは、オペレータのセッション・ウィンドウに表示されます。

注意: エージェント間で作業をロード・バランシングする場合、エージェントに名前を付ける（つまり、`-NAME` パラメータ付きでエージェントを開始する）必要があります。詳細は、「リスナー・エージェントの起動」を参照してください。

関連項目:

- 物理エージェント
- 物理エージェントの作成
- セッション

その他のタスク

パスワードの暗号化

エージェントやシナリオの起動時に参照されるスクリプトのパスワードは、セキュリティ上の理由により暗号化する必要があります。

パスワードを暗号化する手順:

1. シェルまたは DOS コマンド・プロンプトを起動します。
2. Oracle Data Integrator のインストール・ディレクトリの`/bin` ディレクトリに移動します。
3. 次のコマンドを入力します。
   `agent ENCODE <password>`
   ここで、`<password>`は暗号化するパスワードです。

Oracle Data Integrator により、暗号化されたパスワードの文字列が戻されます。

暗号化されたパスワードを復号化する方法はありません。

トポロジ: レポートの生成

トポロジ・レポートを生成する手順:

1. **トポロジ・マネージャ**で**「ファイル」→「印刷」**を選択し、生成するレポートを指定します。**「論理アーキテクチャ」**、**「物理アーキテクチャ」**および**「コンテキスト」**のレポートを生成できます。
2. 出力ファイルの場所を入力するか、出力ファイルを選択します。
3. **「OK」**をクリックします。

Adobe™ PDF 形式のレポートが、手順 2 で指定したファイルに生成されます。ディレクトリを指定しない場合、このファイルは PDF 生成用のデフォルト・ディレクトリに配置されます。

「ファイルを生成後に開きますか。」オプションを選択した場合、**Acrobat® Reader™**が起動して生成されたレポートが表示されます。

注意: 生成されたレポートを表示するには、ユーザー・パラメータで **Acrobat® Reader™**プログラムの場所を指定しておく必要があります。

ユーザー・パラメータの設定

Oracle Data Integrator では、デフォルト・ディレクトリやウィンドウ位置などのユーザー・パラメータが保存されます。

ユーザー・パラメータは、`/bin` の `userpref.xml` ファイルに保存されます。

ユーザー・パラメータを設定する手順:

1. **「ファイル」**メニューで、**「ユーザー・パラメータ」**を選択します。
2. **ユーザー・パラメータを編集中**ウィンドウで、必要に応じてパラメータの値を変更します。
3. **「OK」**をクリックして保存し、ウィンドウを閉じます。

使用可能なユーザー・パラメータのリストは、リファレンス・マニュアルを参照してください。

オブジェクトの検索

Oracle Data Integrator の検索機能を使用すると、デザイナ、オペレータ、トポロジ・マネージャなどの各グラフィカル・モジュールでオブジェクトを迅速に検索できます。

検索は、それぞれ特定の範囲内で実行されます。**検索範囲**は、Oracle Data Integrator の機能ドメインに対応する事前定義されたオブジェクト・タイプのセットです。
たとえば、メタデータの検索範囲では、メタデータ関連のすべてのオブジェクト（データストア、モデル、参照など）が検索され、プロシージャの検索範囲ではプロシージャとプロシージャのコマンドおよびオプションが検索されます。

検索基準は次のとおりです。

- Oracle Data Integrator オブジェクトのすべての範囲とすべてのタイプ
- 特定の範囲内のすべてのオブジェクト・タイプ

- 特定のオブジェクト・タイプ

検索を実行する手順:

1. モジュール・ツールバーの**「検索ビュー」**ボタンをクリックし、検索パネルを開きます。
2. **「検索範囲」**を選択するか、すべての Oracle Data Integrator オブジェクトのリストに対応する「<すべての有効範囲>」を選択します。
3. **「オブジェクト・タイプ」**を選択するか（この選択はオプションで検索範囲によりフィルタされます）、すべてのオブジェクト・タイプを検索する「<すべてのオブジェクト型>」を選択します。
4. 次の検索基準を指定します。
    - **名前:** このフィールドは常に使用可能です。検索するオブジェクトの名前を入力します。
    - **お気に入り基準:** 特定のオブジェクト・タイプの最も効果的な基準に対応するフィールド。**「オブジェクト・タイプ」**を選択しない場合、このパネルは使用できません。
    - **日付およびユーザー:** オブジェクトのユーザー、基準の日付、または最新の更新に基づいて検索をフィルタします。
5. **「拡張」**タブを選択すると、特定のオブジェクト・タイプに関する追加の検索基準を指定できます。**「オブジェクト・タイプ」**を選択しない場合、このタブには**「名前」**フィールドのみが表示されます。
6. すべてのテキストを大/小文字を区別して一致させる場合、**「大/小文字一致」**ボックスを選択します。
7. **「検索」**ボタンをクリックします。
   検索が開始されます。プログレス・バーが表示されます。
8. 検索が完了すると、**「階層表示」**または**「線形表示」**に結果が表示されます。ビューを切り替えるには、一番下のタブを使用します。
   オブジェクトを右クリックすることで、そのオブジェクトを編集、削除または表示できます。
9. 現在の検索基準をファイル内に保存するには、**「検索条件を保存」**ボタンをクリックします。保存した検索基準を後で使用するには、**「検索基準のロード」**ボタンを使用します。

検索のヒント

- 大量のオブジェクトが検索されることを避けるため、検索範囲と適切な検索基準を指定してください。特定のタイプおよび基準に一致するオブジェクトが多すぎると、検索が遅くなる可能性があります。
- **「名前」**などのテキスト・フィールドでは、基準として入力された値が、オブジェクトのテキスト・フィールド内のどこかに含まれていれば一致します。たとえば、**「名前」**フィールドに Data という語を入力して検索すると、Project Data Warehouse や Data Mart Structure などのオブジェクトが戻されます。数値や日付などの他のフィールド・タイプでは、正確に一致するオブジェクトのみが戻されます。

インポート/エクスポート

トポロジまたはセキュリティ設定のエクスポート

トポロジまたはセキュリティをエクスポートしてからインポートすると、あるマスター・リポジトリから別のマスター・リポジトリにドメインを移動できます。

次のドメインをエクスポートできます。

- **トポロジ:** トポロジ全体（論理アーキテクチャと物理アーキテクチャ）。
- **論理トポロジ:** テクノロジ（接続、データ型または言語情報）、論理エージェントおよび論理スキーマ。
- **セキュリティ:** ユーザー、プロファイル、権限およびホスト。このエクスポートは、**セキュリティ・マネージャ**から実行できます。
- **実行環境:** テクノロジ、データ型、コンテキスト、物理スキーマおよびエージェント。

トポロジまたはセキュリティをエクスポートする手順:

1. トポロジ・マネージャ（またはセキュリティ・マネージャ）の**「ファイル」→「エクスポート」**メニューで、次のいずれかのメニュー・オプションを選択します。
   - **トポロジ**
   - **論理トポロジ**
   - **セキュリティ設定**
   - **実行環境**
2. 次のエクスポート・パラメータを入力します。
   - **ディレクトリへエクスポート:** エクスポート・ファイルを作成するディレクトリ。
   - **ZIP ファイルにエクスポート:** このオプションを選択すると、すべてのエクスポート・ファイルを含む単一の圧縮ファイルが作成されます。選択しない場合、エクスポート・ファイルのセットが作成されます。
   - **ZIP ファイル名:** 圧縮ファイルの名前（**「圧縮されたエクスポートを作成」**オプションを選択している場合）。
   - **キャラクタ・セット:** エクスポート・ファイルに指定するエンコーディング。XML ファイル・ヘッダーの `encoding` パラメータの値です。
   - **Java キャラクタ・セット:** ファイル生成に使用する Java キャラクタ・セット。
3. **「OK」**をクリックします。

指定したエクスポート・ディレクトリにエクスポート・ファイルが作成されます。

関連項目:

マスター・リポジトリのエクスポート

マスター・リポジトリのインポート

トポロジまたはセキュリティ設定のインポート

オブジェクトのエクスポート

エクスポートおよびインポートにより、Oracle Data Integrator オブジェクトをあるリポジトリから別のリポジトリに移動できます。

Oracle Data Integrator からオブジェクトをエクスポートする手順:

1. Oracle Data Integrator の適切なツリー・ビューで、エクスポートするオブジェクトを選択します。
2. オブジェクトを右クリックし、**「エクスポート」**を選択します。このメニュー項目が表示されない場合は、エクスポート機能に対応しないタイプのオブジェクトです。
3. エクスポート・パラメータを設定し、**「OK」**をクリックします。
   少なくとも**「エクスポート名」**は指定する必要があります。**「エクスポート・ディレクトリ」**を指定しない場合、エクスポート・ファイルは**「デフォルトのエクスポート・ディレクトリ」**に作成されます。

オブジェクトが XML ファイルとして指定した場所にエクスポートされます。

関連項目:

オブジェクトのインポート

オブジェクト・エクスポート・パラメータ

複数オブジェクトのエクスポート

トポロジまたはセキュリティ設定のインポート

トポロジまたはセキュリティをエクスポートしてからインポートすると、あるマスター・リポジトリから別のマスター・リポジトリにドメインを移動できます。

トポロジ・エクスポートをインポートする手順:

1. トポロジ・マネージャで**「ファイル」→「インポート」**をクリックし、次のいずれかのオプションを選択します。
   - **トポロジ**
   - **論理トポロジ**
   - **実行環境**
2. **インポート・モード**と、インポート用の**「リポジトリ」**または**「ZIP ファイル」**を選択します。
3. **「OK」**をクリックします。

指定したファイルがマスター・リポジトリにインポートされます。

セキュリティ・エクスポートをインポートする手順:

1. トポロジ・マネージャで**「ファイル」→「インポート」→「セキュリティ設定」**をクリックし、次のいずれかのオプションを選択します。
2. **インポート・モード**と、インポート用の**「リポジトリ」**または**「ZIP ファイル」**を選択します。
3. **「OK」**をクリックします。

指定したファイルがマスター・リポジトリにインポートされます。

関連項目:

マスター・リポジトリのエクスポート

マスター・リポジトリのインポート

トポロジまたはセキュリティ設定のエクスポート

インポート・モード

オブジェクトのインポート

インポートおよびエクスポートにより、オブジェクト（インタフェース、ナレッジ・モジュール、モデルなど）をあるリポジトリから別のリポジトリに移動できます。

Oracle Data Integrator でオブジェクトをインポートする手順:

1. インポートする適切なオブジェクトをツリーから選択します。適切なオブジェクトが存在しない場合は、作成します。
2. オブジェクトを右クリックして**「インポート」**を選択し、インポートするオブジェクトのタイプを選択します。
3. **「エクスポート・ディレクトリ」**を指定し、**「インポート名」**で 1 つ以上のファイルを選択します。**インポート・モード**を選択し、**「OK」**をクリックします。

XML 形式のファイルが作業リポジトリにインポートされ、Oracle Data Integrator に表示されます。

関連項目:

オブジェクトのエクスポート

インポート・モード

技術環境のエクスポート

この機能により、任意のディレクトリに技術環境の詳細を含むカンマ区切り（.csv）ファイルを生成できます。
この情報は、サポート用として役立ちます。

このファイルの形式はカスタマイズできます。

技術環境ファイルを生成する手順:

1. トポロジ・マネージャで、**「ファイル」→「エクスポート」→「技術環境のエクスポート」**を選択します。
   技術環境のエクスポート・ウィンドウが表示されます。
2. 次のプロパティを指定します。
   - **エクスポート・ディレクトリ**
   - **ファイル名**
   - **キャラクタ・セット:** エクスポート・ファイルのキャラクタ・セット。
   - フィールド・コード: 生成される各レコードの最初のフィールドには、行に存在する情報の種類を識別するためのコードが含まれます。これらのコードは、必要に応じてカスタマイズできます。後述のコード・リストを参照してください。
   - **レコード・セパレータ**
   - **フィールド・セパレータ**
   - **フィールド・デリミタ**
3. **「OK」**をクリックします。

レコード・コード

プロパティ	説明
Oracle Data Integrator 情報レコード・コード	Oracle Data Integrator の現在のリリースおよび現在のユーザーが記録される行を識別するためのコード。このコードはレコードの最初のフィールドに使用される。
マスター・レコード・コード	マスター・リポジトリに関する情報を含む行のコード。
作業レコード・コード	作業リポジトリに関する情報を含む行のコード。
エージェント・レコード・コード	稼働中の様々なエージェントに関する情報を含む行のコード。
テクノロジ・レコード・コード	データ・サーバーやそのバージョンなどに関する情報を含む行のコード。

セキュリティ・マネージャ

セキュリティ・マネージャの概要

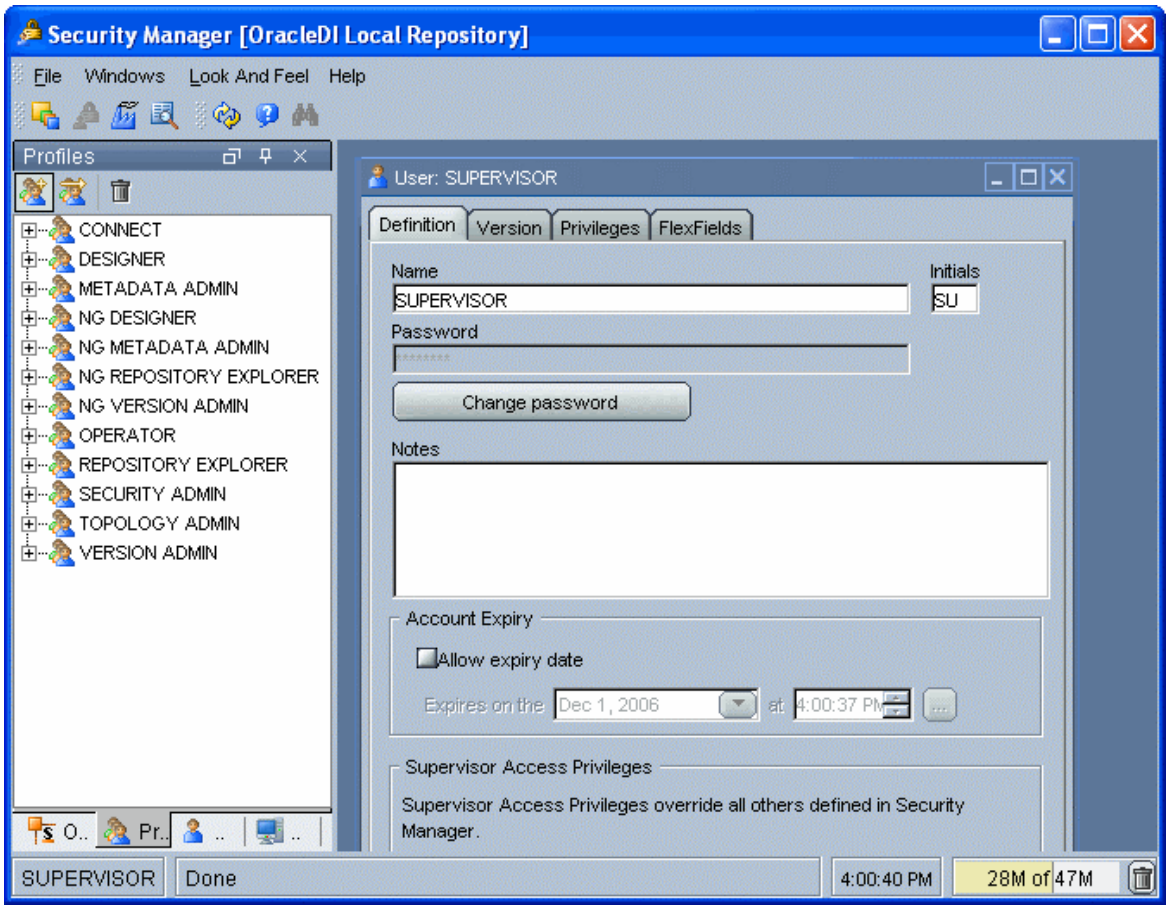
セキュリティ・マネージャ・モジュールを使用すると、Oracle Data Integrator のセキュリティを管理することができます。セキュリティ・マネージャ・モジュールを使用すると、**ユーザー**および**プロファイル**が作成されます。汎用**オブジェクト**（データ・サーバー、データ型など）で**メソッド**（編集、削除など）にユーザー権限を割り当てるため、およびオブジェクト・**インスタンス**（Server 1、Server 2 など）での権限を微調整するために使用します。

セキュリティ・マネージャ・モジュールは、この情報をマスター・リポジトリに格納します。この情報は、他のすべてのモジュールで使用できます。

- セキュリティ・ポリシーの定義

セキュリティ・マネージャのインタフェース

セキュリティ・マネージャの GUI は次のように表示されます。



メニュー

メニューでは、プルダウン・メニューから次の機能にアクセスできます。

- インポート/エクスポート
- ウィザード
- オプションの表示
- モジュールまたはツリー表示のオープン
- ユーザーのパスワードおよびオプションの変更

ツールバー

ツールバーからは、次の操作を実行できます。

- 他のモジュールの起動
- ツリー表示のリフレッシュ
- オンライン・ヘルプの起動

ツリー表示

現在のユーザーが使用できるセキュリティ・マネージャ・オブジェクトは、次のように整理してツリー表示に表示されます。

- **オブジェクト**。Oracle Data Integrator の要素タイプ（データストア、モデルなど）が記述されます。
- ユーザーの**プロファイル**と、その**権限**。
- **ユーザー**およびその**権限**。

各ツリー表示は、メイン・ウィンドウの両側にドッキングできるフローティング・フレームに表示されます。フレームは重ねることもできます。複数のフレームを重ねた場合は、フレーム・ウィンドウの下部に表示されるタブから各フレームにアクセスできます。

ツリー表示フレームは、フレームのタイトルまたはタブを選択してドラッグすることにより、移動したり、ドッキングしたり、重ねたりできます。ツリー表示の位置を固定するには、**「ウィンドウ」**メニューから**「ウィンドウ・レイアウトのロック」**を選択します。

ツリー表示フレームがメイン・ウィンドウに表示されないか、閉じている場合は、**「ウィンドウ」→「ビューの表示」**メニューを使用します。

各ツリー表示では、次の操作が可能です。

- ルート・オブジェクトの挿入またはインポート（フレーム・タイトルで該当するボタンをクリック）
- ノードの展開または折りたたみ（ノードをクリック）
- オブジェクトに関連付けられているメソッド（編集、削除など）のアクティブ化（ポップアップ・メニューを使用）
- オブジェクトの編集（オブジェクトをダブルクリック、または**ワークベンチ**にドラッグ・アンド・ドロップ）

ワークベンチ

編集または表示されているオブジェクトのウィンドウは、**ワークベンチ**に表示されます。

セキュリティ・ポリシーの定義

セキュリティの概要

オブジェクト、インスタンスおよびメソッド

オブジェクトは、Oracle Data Integrator を通じて処理できる要素（エージェント、モデル、データストアなど）を表現したものです。オブジェクトは、Oracle Data Integrator オブジェクト・コンポーネント（Java クラス）の可視部分に相当します。オブジェクトと（1 つ以上の）オブジェクト・インスタンスの概念は、まとめて考える必要があります（Oracle Data Integrator では、これらの概念はオブジェクト指向プログラミングの概念とほぼ同じです）。

インスタンス（オブジェクト・インスタンス）は、オブジェクト・タイプ（オブジェクト）に関連付けられます。たとえば、プロジェクト MY_PROJ_1 は、プロジェクト・オブジェクト・タイプのインスタンスです。同様に、YOUR_PROJ_2 は、プロジェクト・オブジェクト・タイプの別のインスタンスです。

メソッドは、**オブジェクト**で実行できるアクションのタイプです。各オブジェクトには、そのオブジェクトに固有の一連のメソッドがあります。Data Integrator のメソッドの概念は、オブジェクト指向プログラミングの概念とほぼ同じです。

警告: オブジェクトとメソッドは、Oracle Data Integrator で事前定義されており、変更できません。

注意: 特定のオブジェクトは、**二重名**（エージェント/コンテキスト、プロファイル/メソッドなど）を持ちます。これらのオブジェクトは、オブジェクト間のリンクを表します。これらのリンクもオブジェクトです。たとえば、エージェント/コンテキストは、コンテキストを通じて作成される物理/論理エージェントの関係に対応します。このオブジェクトに対する権限により、トポロジでこの関係を変更できます。

プロファイル

プロファイルは、Oracle Data Integrator を操作するための一般的な権限モデルを表します。1 つ以上のプロファイルをユーザーに割り当てることができます。

プロファイル別の認可は、特定の**プロファイル**に対応する**オブジェクト**の**メソッド**に設定されます。この設定により、このプロファイルを持つユーザーに（オプションで、または自動的に）メソッドを通じてオブジェクトに対する権限を付与できます。

- プロファイル別の認可が、プロファイル・ツリーのオブジェクトの下にあるメソッドに存在する場合、このプロファイルを持つユーザーは、（オプションで、または自動的に）メソッドを通じてオブジェクト・インスタンスを操作する権限を付与されます。
- プロファイル別の認可が存在しない場合、このプロファイルを持つユーザーは、どのような状況においてもオブジェクト・インスタンスのメソッドを起動できません。

汎用プロファイルと汎用ではないプロファイルの比較

汎用プロファイルでは、すべてのオブジェクトのメソッドに対して**「一般権限」**オプションが選択されています。つまり、このようなプロファイルを持つユーザーは、プロファイルが認可されるオブジェクトのすべてのインスタンスに含まれるすべてのメソッドに対して、**デフォルトで**認可されます。

汎用ではないプロファイルでは、すべてのオブジェクトのメソッドに対して**「一般権限」**オプションが選択されていないため、**デフォルトでは**インスタンスのすべてのメソッドに対して認可されません。管理者は、各インスタンスのメソッドに対する権限をユーザーに付与する必要があります。

管理者がユーザーにどのインスタンスに対する権限もデフォルトでは付与せず、かわりにインスタンスごとに権限を付与する場合、ユーザーには汎用ではないプロファイルを割り当てる必要があります。

管理者がユーザーにオブジェクト・タイプのすべてのインスタンスに対する権限をデフォルトで付与する場合、ユーザーには汎用プロファイルを割り当てる必要があります。

ユーザー

セキュリティ・マネージャ・モジュールのユーザーは、Oracle Data Integrator ユーザーを表し、リポジトリへの接続に使用されるログイン名に対応します。

ユーザーは、次の権限を継承します。

- ユーザーがすでに保持しているプロファイル権限
- オブジェクトに対する権限
- インスタンスに対する権限

ユーザー別の認可は、特定の**ユーザー**に対応する**オブジェクト**の**メソッド**に設定されます。この設定により、ユーザーに（オプションで、または自動的に）メソッドを通じてオブジェクトに対する権限を付与できます。

- メソッドに対応するユーザー別の認可が、ユーザー・ツリーのオブジェクトの下に存在する場合、そのユーザーは、（オプションで、または自動的に）メソッドを通じてオブジェクト・インスタンスを操作する権限を付与されます。
- ユーザー別の認可が存在しない場合、そのユーザーは、どのような状況においてもオブジェクト・インスタンスのメソッドを起動できません。

オブジェクト・インスタンス別の認可は、特定の**ユーザー**に対応する**オブジェクト・**インスタンスに設定されます。この設定により、ユーザーに、オブジェクト・インスタンスの特定のメソッドに対する権限を付与できます。

インスタンスに対応するオブジェクト・インスタンス別の認可がユーザー・ツリーに存在する場合、そのユーザーに、特定のインスタンスのオブジェクト・メソッドに対する特定の権限を付与できます（これらの権限はウィンドウで指定します）。インスタンスがユーザーの**インスタンス**のツリーに存在しない場合、実際のインスタンスを生成したオブジェクトに対するユーザー別またはプロファイル別の認可が適用されます。

オブジェクト・インスタンス別の認可は、作業リポジトリごとに定義できます。オブジェクト・インスタンスは、作業リポジトリ間で（シノニム・モードによるエクスポートまたはインポートを使用して）レプリケートするか、コンテキストなどの異なる作業リポジトリで使用することができます。複数のリポジトリでこのようなインスタンスを使用できる場合、権限の付与と拒否は、すべてのリポジトリで実行する、どのリポジトリでも実行しない、または選択したリポジトリでのみ実行することが可能です。

たとえば、開発リポジトリからテスト・リポジトリにプロジェクトをレプリケートすることはよくあります。開発者に開発リポジトリの自分のプロジェクトに対する編集権限を付与する一方で、テスト・リポジトリにおける編集権限は付与しないよう設定できます。テスト・リポジトリでは、開発者にプロジェクトに対する参照権限のみを付与します。

セキュリティ・ポリシー

Oracle Data Integrator のセキュリティについては、主に次の 2 つのアプローチがあります。

- 高度にセキュアなアプローチ: ユーザーは、汎用ではないプロファイルのみを使用し、オブジェクトに対するどのような権限も保持しません。セキュリティ管理者は、インスタンス（特定のインタフェースなど）に対する認可をユーザーに付与する必要があります。このポリシーは、設定が複雑です。
- 一般的なアプローチ: ユーザーは、保持するプロファイルの権限を継承します。このポリシーは、ほとんどの環境に適しています。

これら 2 つのアプローチを組み合せることが可能です。

高度にセキュアなポリシーを定義する手順:

1. 作業方法に適したプロファイルを作成し、オブジェクトに対する汎用ではない権限を各プロファイルに付与します。Oracle Data Integrator に付属する汎用ではないプロファイルを使用できます。

2. ユーザーを作成します。
3. ユーザーに汎用ではないプロファイルを付与します。
4. ユーザーにオブジェクト・インスタンスに対する権限を付与します。この操作は、新規インスタンスを作成するたびに繰り返す必要があります。

一般的なポリシーを定義する手順:

1. 作業方法に適したプロファイルを作成し、オブジェクトに対する汎用権限を各プロファイルに付与します。Oracle Data Integrator に付属する汎用プロファイルを使用できます。
2. ユーザーを作成します。
3. ユーザーに汎用プロファイルを付与します。

ユーザーがセキュアなパスワードを使用できるよう、パスワード・ポリシーを定義することもできます。

プロファイルの作成

新規プロファイルの作成

新規プロファイルを作成する手順:

1. ツリーで**「プロファイル」**を選択します。
2. 右クリックし、**「プロファイルの挿入」**を選択します。
3. **「名前」**を入力します。
4. **「OK」**をクリックします。

ツリーにプロファイルが表示されます。

プロファイルの削除

プロファイルを削除する手順:

1. 削除する**プロファイル**を選択します。
2. 右クリックし、**「削除」**を選択します。
3. **「OK」**をクリックします。

ツリーからプロファイルが削除されます。

ユーザーの作成

新規ユーザーの作成

新規ユーザーを作成する手順:

1. ツリーで**「ユーザー」**を選択します。
2. 右クリックし、**「ユーザーの挿入」**を選択します。
3. **「名前」**、**「イニシャル」**、および（**「パスワードの入力」**ボタンをクリックして）ユーザーの**「パスワード」**を入力します。

4. 「**OK**」をクリックします。

ツリーにユーザーが表示されます。

ユーザーへのプロファイルの割当て

ユーザーにプロファイルを割り当てる手順:

1. ツリーを開き、プロファイルを割り当てる**ユーザー**を表示します。
2. 割り当てる**プロファイル**を選択し、ツリーの**ユーザー**にドラッグ・アンド・ドロップします。
3. 「**OK**」をクリックします。

プロファイルがユーザーに割り当てられます。

ユーザーからのプロファイルの削除

ユーザーからプロファイルを削除する手順:

1. ツリーを開き、**ユーザー**のブランチで削除する**プロファイル**を表示します。
2. ツリーから削除する（ユーザーの下の）**プロファイル**を選択します。
3. 右クリックし、**「削除」**を選択します。
4. 「**OK**」をクリックします。

プロファイルがユーザーから削除されます。

ユーザーの削除

ユーザーを削除する手順:

1. 削除する**ユーザー**を選択します。
2. 右クリックし、**「削除」**を選択します。
3. 「**OK**」をクリックします。

ツリーからユーザーが削除されます。

権限の管理

プロファイルまたはユーザー別の認可の割当て

プロファイルまたはユーザー別の認可を割り当てる手順:

1. ツリーを開き、認可を割り当てる**ユーザー**または**プロファイル**を表示します。
2. **オブジェクト**の下で割り当てる**メソッド**を選択し、**ユーザー**または**プロファイル**にドラッグ・アンド・ドロップします。
3. 「**OK**」をクリックします。

認可がユーザーまたはプロファイルに割り当てられます。

プロファイルまたはユーザーにオブジェクトのすべてのメソッドに対する認可を割り当てる手順:

1. ツリーを開き、認可を割り当てる**ユーザー**または**プロファイル**を表示します。

2. すべてのメソッドを割り当てる**オブジェクト**を選択し、**ユーザー**または**プロファイル**にドラッグ・アンド・ドロップします。
3. **「OK」**をクリックします。

すべてのメソッドに対する認可がユーザーまたはプロファイルに割り当てられます。

プロファイルまたはユーザー別の認可の削除

プロファイルまたはユーザー別の認可を削除する手順:

1. **ユーザー**または**プロファイル**のブランチの下で、削除する**メソッド**を選択します。
2. 右クリックし、**「削除」**を選択します。
3. **「OK」**をクリックします。

認可がユーザーまたはプロファイルから削除されます。

プロファイルまたはユーザーからオブジェクトのすべてのメソッドに対する認可を削除する手順:

1. **ユーザー**または**プロファイル**のブランチの下で、削除する権限割当てを含む**オブジェクト**を選択します。
2. 右**クリック**し、**「削除」**を選択します。
3. **「OK」**をクリックします。

すべての認可がユーザーまたはプロファイルから削除されます。

オブジェクト・インスタンス別の認可の割当て

ユーザーにオブジェクト・インスタンス別の認可を割り当てる手順:

1. **セキュリティ・マネージャ**のツリーを開き、認可を追加する**ユーザー**を表示します。
2. **デザイナ**を起動し、ツリーを開いて認可を割り当てるオブジェクト・インスタンスを表示します。
3. **デザイナ**の**ツリー**で**オブジェクト・インスタンス**を選択し、**セキュリティ・マネージャ**の**ユーザー**にドラッグ・アンド・ドロップします。
4. 表示される**オブジェクト・**インスタンス別の認可ウィンドウで権限を調整します。
   リストで選択したメソッドに対して、次の単純な権限ポリシーを実装できます。
   - すべてのリポジトリでこれらすべてのメソッドを許可するには、☑**「すべてのリポジトリ内の選択したメソッドを許可」**ボタンをクリックします。
   - すべてのリポジトリでこれらすべてのメソッドを拒否するには、☒**「すべてのリポジトリ内の選択したメソッドを拒否」**ボタンをクリックします。
   - 特定の作業リポジトリでこれらすべてのメソッドを許可するには、?**「選択したリポジトリ内の選択したメソッドを許可」**ボタンをクリックし、リストでリポジトリを選択します。
5. **「OK」**を**クリック**します。

オブジェクト・インスタンス別の認可がユーザーまたはプロファイルに割り当てられます。

注意: ユーザーが汎用権限を保持するメソッドは、オブジェクト・インスタンス別の認可ウィンドウにリストされません。

オブジェクト・インスタンス別の認可の削除

ユーザーからオブジェクト・インスタンス別の認可を削除する手順:

1. 変更する権限を含むインスタンスのオブジェクト・インスタンス別の認可ウィンドウを表示します。
2. **「すべてを拒否」**ボタンをクリックします。
3. **「OK」**をクリックします。

ユーザーから認可が削除されます。この操作後、インスタンスに対するすべての権限がユーザーから失われると、**セキュリティ・マネージャ**のツリーからインスタンスが削除されます。

セキュリティ・クリーンアップ・ツールを使用すると、使用されていないオブジェクト・インスタンス別のオブジェクト認可をすべて削除できます。

未使用の認可のクリーンアップ

オブジェクト・インスタンス別の認可は、マスター・リポジトリに格納されます。ただし、すべての作業リポジトリからこれらのオブジェクトが削除されても、認可は削除されません。さらに、一部の未使用の認可が、エクスポート・ファイルや格納済のソリューションに現在保存されているオブジェクトを参照している場合などには、それらの認可を保持することもできます。

マスター・リポジトリから未使用の認可を削除するには、定期的にセキュリティ・クリーンアップ・ツールを使用する必要があります。未使用の認可は、マスター・リポジトリまたはすべての作業リポジトリに存在しないオブジェクトを参照している場合、削除されます。

注意: 各リポジトリでオブジェクトの存在をチェックするには、マスター・リポジトリに関連するすべての作業リポジトリにアクセス可能である必要があります。

注意: Oracle Data Integrator オブジェクト自体は削除されず、セキュリティ権限情報のみが削除されます。

未使用の認可を削除する手順:

1. メニューで**「ファイル」→「セキュリティ設定のクリーンアップ」**を選択します。
2. **「クリーニング可能」**タブで、削除する不要なセキュリティ設定を選択します。
   - クリーンアップできないセキュリティ設定は、「クリーニング不可」タブに表示されます。
   - **「すべて選択」**ボタンを使用すると、クリーンアップする設定をすべて選択できます。
3. **「選択したセキュリティ設定の削除」**をクリックします。

Security Manager の使用

パスワード・ポリシーの定義

パスワード・ポリシーは、ユーザー・パスワードに適用されるルールのセットで構成されます。このルールのセットは、ユーザーによるパスワードの定義時にチェックされます。

パスワード・ポリシーを定義する手順:

1. **セキュリティ・マネージャ**で、**「ファイル」→「パスワード・ポリシー」**を選択します。**パスワード・ポリシー**・ウィンドウが表示されます。このウィンドウに、ルールのリストが表示されます。
2. **「ルールの追加」**  ボタンをクリックします。
新規ルール定義ウィンドウが表示されます。ルールは、パスワードに対してチェックされる条件のセットです。
3. この新規ルールの**「名前」**と**「説明」**を設定します。
4. パスワードの値または長さに関する条件を追加します。たとえば、このウィンドウでパスワードの最小の長さを定義できます。
5. パスワードがルールに一致するとみなすために、少なくとも 1 つの条件を検証するか、すべての条件を検証するかを選択します。
6. **「OK」**をクリックします。
7. 必要な数のルールを追加し、その中で現在のポリシーでアクティブにするルールを選択します。すべてのルールに一致するパスワードのみが、現在のポリシーに対して有効とみなされます。
8. パスワードの有効期限も定義できます。この日数を超過したパスワードは、自動的に期限切れとなります。ユーザーは、パスワードを変更する必要があります。
9. **「OK」**をクリックしてパスワード・ポリシーを更新します。

ユーザー・パラメータの設定

Oracle Data Integrator では、デフォルト・ディレクトリやウィンドウ位置などのユーザー・パラメータが保存されます。

ユーザー・パラメータは、`/bin` の `userpref.xml` ファイルに保存されます。

ユーザー・パラメータを設定する手順:

1. **「ファイル」**メニューで、**「ユーザー・パラメータ」**を選択します。
2. **ユーザー・パラメータを編集中**ウィンドウで、必要に応じてパラメータの値を変更します。
3. **「OK」**をクリックして保存し、ウィンドウを閉じます。

使用可能なユーザー・パラメータのリストは、リファレンス・マニュアルを参照してください。

オブジェクトのエクスポート

エクスポートおよびインポートにより、Oracle Data Integrator オブジェクトをあるリポジトリから別のリポジトリに移動できます。

Oracle Data Integrator からオブジェクトをエクスポートする手順:

1. Oracle Data Integrator の適切なツリー・ビューで、エクスポートするオブジェクトを選択します。
2. オブジェクトを右クリックし、**「エクスポート」**を選択します。このメニュー項目が表示されない場合は、エクスポート機能に対応しないタイプのオブジェクトです。
3. エクスポート・パラメータを設定し、**「OK」**をクリックします。
少なくとも**「エクスポート名」**は指定する必要があります。**「エクスポート・ディレクト**

リ」を指定しない場合、エクスポート・ファイルは**「デフォルトのエクスポート・ディレクトリ」**に作成されます。

オブジェクトが XML ファイルとして指定した場所にエクスポートされます。

関連項目:

オブジェクトのインポート

オブジェクト・エクスポート・パラメータ

複数オブジェクトのエクスポート

オブジェクトのインポート

インポートおよびエクスポートにより、オブジェクト（インタフェース、ナレッジ・モジュール、モデルなど）をあるリポジトリから別のリポジトリに移動できます。

Oracle Data Integrator でオブジェクトをインポートする手順:

1. インポートする適切なオブジェクトをツリーから選択します。適切なオブジェクトが存在しない場合は、作成します。
2. オブジェクトを右クリックして**「インポート」**を選択し、インポートするオブジェクトのタイプを選択します。
3. **「エクスポート・ディレクトリ」**を指定し、**「インポート名」**で 1 つ以上のファイルを選択します。**インポート・モード**を選択し、**「OK」**をクリックします。

XML 形式のファイルが作業リポジトリにインポートされ、Oracle Data Integrator に表示されます。

関連項目:

オブジェクトのエクスポート

インポート・モード

トポロジまたはセキュリティ設定のエクスポート

トポロジまたはセキュリティをエクスポートしてからインポートすると、あるマスター・リポジトリから別のマスター・リポジトリにドメインを移動できます。

次のドメインをエクスポートできます。

- **トポロジ:** トポロジ全体（論理アーキテクチャと物理アーキテクチャ）。
- **論理トポロジ:** テクノロジ（接続、データ型または言語情報）、論理エージェントおよび論理スキーマ。
- **セキュリティ:** ユーザー、プロファイル、権限およびホスト。このエクスポートは、**セキュリティ・マネージャ**から実行できます。
- **実行環境:** テクノロジ、データ型、コンテキスト、物理スキーマおよびエージェント。

トポロジまたはセキュリティをエクスポートする手順:

1. トポロジ・マネージャ（またはセキュリティ・マネージャ）の**「ファイル」→「エクスポート」**メニューで、次のいずれかのメニュー・オプションを選択します。:
   - **トポロジ**
   - **論理トポロジ**

- **セキュリティ設定**
- **実行環境**

2. 次のエクスポート・パラメータを入力します。
   - **ディレクトリへエクスポート:** エクスポート・ファイルを作成するディレクトリ。
   - **ZIP ファイルにエクスポート:** このオプションを選択すると、すべてのエクスポート・ファイルを含む単一の圧縮ファイルが作成されます。選択しない場合、エクスポート・ファイルのセットが作成されます。
   - **ZIP ファイル名:** 圧縮ファイルの名前（**「圧縮されたエクスポートを作成」**オプションを選択している場合）。
   - **キャラクタ・セット:** エクスポート・ファイルに指定するエンコーディング。XML ファイル・ヘッダーの `encoding` パラメータの値です。
   - **Java キャラクタ・セット:** ファイル生成に使用する Java キャラクタ・セット。
3. **「OK」**をクリックします。

指定したエクスポート・ディレクトリにエクスポート・ファイルが作成されます。

関連項目:

マスター・リポジトリのエクスポート

マスター・リポジトリのインポート

トポロジまたはセキュリティ設定のインポート

トポロジまたはセキュリティ設定のインポート

トポロジまたはセキュリティをエクスポートしてからインポートすると、あるマスター・リポジトリから別のマスター・リポジトリにドメインを移動できます。

トポロジ・エクスポートをインポートする手順:

1. トポロジ・マネージャで**「ファイル」→「インポート」**をクリックし、次のいずれかのオプションを選択します。
   - **トポロジ**
   - **論理トポロジ**
   - **実行環境**
2. **インポート・モード**と、インポート用の**「リポジトリ」**または**「ZIP ファイル」**を選択します。
3. **「OK」**をクリックします。

指定したファイルがマスター・リポジトリにインポートされます。

セキュリティ・エクスポートをインポートする手順:

1. トポロジ・マネージャで**「ファイル」→「インポート」→「セキュリティ設定」**をクリックし、次のいずれかのオプションを選択します。
2. **インポート・モード**と、インポート用の**「リポジトリ」**または**「ZIP ファイル」**を選択します。
3. **「OK」**をクリックします。

指定したファイルがマスター・リポジトリにインポートされます。

関連項目:

マスター・リポジトリのエクスポート

マスター・リポジトリのインポート

トポロジまたはセキュリティ設定のエクスポート

インポート・モード

Oracle Data Integrator の使用

はじめに

この項では、よく使用されるテクノロジに関する Oracle Data Integrator の手順について説明します。

テクノロジ	ドキュメント
Oracle	• Oracle データ・サーバーの作成 • Oracle の物理スキーマの作成 • Oracle モデルの作成およびリバース・エンジニアリング • Oracle 用の適切な KM の選択 • Oracle での一般的エラー
AS/400 および iSeries の DB2	• DB2/400 データ・サーバーの作成 • DB2/400 の物理スキーマの作成 • DB2/400 モデルの作成およびリバース・エンジニアリング • DB2/400 用の適切な KM の選択 • iSeries での CDC の使用方法 • DB2/400 での一般的エラー • iSeries および AS/400 への Java エージェントのインストール
Microsoft Excel	• Microsoft Excel データ・サーバーの作成 • Microsoft Excel の物理スキーマの作成 • Microsoft Excel モデルの作成およびリバース・エンジニアリング • Microsoft Excel 用の適切な KM の選択 • Microsoft Excel での一般的エラー
フラット・ファイル	• ファイル・データ・サーバーの作成 • ファイルの物理スキーマの作成 • ファイル・モデルの作成およびリバース・エンジニアリング • ファイル用の適切な KM の選択 • COBOL コピーブックのリバース・エンジニアリング

	• ウィザードを使用した固定ファイルのリバース・エンジニアリング
XML	• XML データ・サーバーの作成 • XML の物理スキーマの作成 • XML ファイル・モデルの作成およびリバース・エンジニアリング • XML ファイル用の適切な KM の選択 • XML での一般的エラー

Oracle

Oracle データ・サーバーの作成

Oracle **データ・サーバー**は、特定の Oracle ユーザー・アカウントに接続された Oracle データベース・インスタンスに対応します。このユーザーは、このインスタンスの複数のスキーマ（データ・サーバーの下に作成された Oracle Data Integrator の**物理スキーマ**に対応）にアクセスします。

前提条件

JDBC ドライバ

Oracle Data Integrator のインストール先の各マシンに、タイプ 4 の Oracle JDBC ドライバ（**Oracle シン・ドライバ**）をインストールする必要があります。このドライバでは、TCP/IP ネットワーク・レイヤーを直接使用するため、他のコンポーネントをインストールする必要はありません。このドライバは、Data Integrator のインストール・ディレクトリの`/drivers` サブディレクトリにコピーする必要のある `classes12.zip` ファイルに含まれます。Oracle Data Integrator では、このドライバのデフォルト・バージョンがインストールされます。Oracle で問題が発生した場合は、このデフォルト・ドライバを、現在使用しているリリースの Oracle に付属するドライバで置き換えることをお薦めします。

Oracle JDBC OCI ドライバを通じて、または ODBC を使用して Oracle サーバーに接続することも可能です。これらの方法では、あまり高いパフォーマンスを得ることはできません。タイプ 4 のドライバを使用することをお薦めします。

接続情報

Oracle DBA から次の情報を入手する必要があります。

- Oracle データベースをホストするマシンのネットワーク名または IP アドレス
- Oracle リスナーのリスニング・ポート
- Oracle インスタンスの名前（SID）
- 接続インスタンスの TNS 別名
- Oracle ユーザーのログインおよびパスワード

データ・サーバーの作成

Oracle データ・サーバーを作成する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで**「トポロジ」→「物理アーキテクチャ」→「テクノロジ」→「Oracle」**を選択します。
3. 右クリックし、**「データ・サーバーの挿入」**を選択します。
4. **「定義」**タブで、次のフィールドを入力します。
   - **名前:** Oracle Data Integrator に表示されるデータ・サーバーの名前。
   - **インスタンス/dblink:** この Oracle インスタンスで使用される TNS 別名。この名前は、リモート・サーバーからこのサーバーのオブジェクトを識別するのに使用されます。このフィールドは、OCI ドライバや DBLink を使用する場合には必須です。
   - **ユーザー/パスワード:** データ・スキーマに対する選択権限と、このデータ・サーバーの下に作成される Oracle の**物理スキーマ**で示される作業スキーマに対する選択、挿入およびオブジェクト作成権限を持つパスワード付きの Oracle ユーザー。
5. **「JDBC」**タブで、次のフィールドを入力します。
   - **JDBC ドライバ:** `oracle.jdbc.driver.OracleDriver`
   - **JDBC URL:** `jdbc:oracle:thin:@<network name or ip address of the Oracle machine>:<port of the Oracle listener (1521)>:<name of the Oracle instance>`
6. **「テスト」**をクリックします。
7. **テスト接続**ウィンドウの**「テスト」**をクリックします。
8. **接続に成功**したことを示すウィンドウが表示されます。**「OK」**をクリックします。接続に失敗した場合は、「Oracle での一般的エラー」を参照してください。
9. **「OK」**をクリックしてデータ・サーバーの作成を承認します。

このデータ・サーバーの最初の**物理スキーマ**に対応する作成ウィンドウが表示されます。「Oracle の物理スキーマの作成」を参照してください。

Oracle の物理スキーマの作成

物理スキーマは、Oracle における次のスキーマのペアに対応します。

- （データ）**スキーマ:** Oracle Data Integrator は、このスキーマでインタフェースのソースおよびターゲット表を検索します。
- **作業スキーマ:** Oracle Data Integrator は、このスキーマで、データ・スキーマに格納されたソースとターゲットに関連する一時作業表を作成および操作します。

注意: 物理スキーマに関連するデータ・サーバーに指定された Oracle ユーザーは、前述の操作を実行するための適切な権限を付与されている必要があります。

物理スキーマの作成

Oracle の物理スキーマを作成する手順:

注意: データ・サーバーの作成直後の場合、手順 1 は無視してください。**物理スキーマ**・ウィンドウがすでに表示されているはずです。

1. Oracle **データ・サーバー**を選択して右クリックし、**「物理スキーマの挿入」**を選択します。**物理スキーマ**・ウィンドウが表示されます。
2. **「スキーマ（スキーマ）」**で、この Oracle Data Integrator の物理スキーマのデータ・スキーマを選択します。このフィールドには、Oracle インスタンスのスキーマ・リストが表示されます。
3. **「スキーマ（作業スキーマ）」**で、この Oracle Data Integrator の物理スキーマの作業スキーマを選択します。このフィールドには、Oracle インスタンスのスキーマ・リストが表示されます。
4. このスキーマをこのデータ・サーバーのデフォルト・スキーマに設定する場合、**「デフォルト」**ボックスを選択します（最初の物理スキーマは、常にデフォルト・スキーマになります）。詳細は、「物理スキーマ」を参照してください。
5. **「コンテキスト」**タブに移動します。
6. この新規**物理スキーマ**に対応する**「コンテキスト」**と既存の**「論理スキーマ」**を選択し、手順 9 に進みます。
   Oracle の論理スキーマが存在しない場合は、手順 7 に進みます。
7.  ボタンをクリックします。
8. 左側の列で既存の**「コンテキスト」**を選択し、右側の列に**「論理スキーマ」**の名前を入力します。この Oracle の論理スキーマは、自動的に作成され、このコンテキスト内の物理スキーマに関連付けられます。

警告: 特定の**コンテキスト**では、**論理スキーマ**は 1 つの**物理スキーマ**にのみ関連付けることができます。

9. **「OK」**をクリックします。

Oracle モデルの作成およびリバース・エンジニアリング

Oracle モデルの作成

Oracle **モデル**は、Oracle スキーマに格納されたビューおよび表に対応するデータストアのセットです。モデルは、常に**論理スキーマ** に基づきます。特定の**コンテキスト**内で、**論理スキーマ**は**物理スキーマ**に対応します。この物理スキーマのデータ・スキーマには、Oracle モデルの表およびビューが格納されます。

Oracle モデルを作成する手順:

1. **デザイナ**に接続します。
2. ツリーで**「モデル」**を選択します。
3. 右クリックし、**「モデルの挿入」**を選択します。
4. **「定義」**タブで、**「名前」**フィールドを入力します。

5. **「テクノロジ」**フィールドで、**「Oracle」**を選択します。
6. **「論理スキーマ」**フィールドで、モデルの基礎となる論理スキーマを選択します。
7. **「リバース」**タブに移動し、モデルのリバース・エンジニアリングに使用する**「コンテキスト」**を選択します。**「適用」**をクリックします。

これでモデルは作成されましたが、データストアはまだ格納されていません。

Oracle モデルのリバース・エンジニアリング

モデルは、データストアなしで作成されます。**リバース・エンジニアリング**操作では、モデルの表およびビューの構造を取得して、モデルのデータストア定義を作成します。リバース・エンジニアリングのタイプには、ドライバの機能のみを使用する**標準**リバース・エンジニアリングと、RKM（リバース・ナレッジ・モジュール）を使用して Oracle ディクショナリから直接オブジェクトの構造を取得する**カスタマイズ**・リバース・エンジニアリングという 2 つのタイプがあります。

注意: カスタマイズ・リバース・エンジニアリングでは、より完全で全体的なカスタマイズが可能ですが、実行方法も複雑になります。Oracle では標準リバース・エンジニアリングを使用することをお薦めします。

標準リバース・エンジニアリング

Oracle で標準リバース・エンジニアリングを実行する手順:

1. Oracle **モデル**の**「リバース」**タブに移動します。
2. 次のフィールドを入力します。
   - **標準**
   - **コンテキスト:** リバース・エンジニアリングに使用するコンテキスト
   - **リバースエンジニアリングするオブジェクトの型:** リバース・エンジニアリング・プロセスで処理する必要のあるオブジェクト・タイプのリスト
3. **「選択的リバース」**タブに移動します。
   - **「選択的リバース」**、**「新規データストア」**および**「リバースするオブジェクト」**の各ボックスを選択します。
4. リバース・エンジニアリング対象のデータストアのリストが表示されます。リバース・エンジニアリングするデータストアについては、選択したままとします。
5. **「リバース」**ボタンをクリックし、次に**「はい」**をクリックして変更を承認します。
6. Oracle Data Integrator により、選択したデータストアのリバース・エンジニアリング・プロセスが開始されます。プログレス・バーが表示されます。

リバース・エンジニアリングされたデータストアが、モデルの下に表示されます。

「リバース」および**「選択的リバース」**タブのオプションを使用すると、リバース・エンジニアリングを詳細に設定できます。詳細は、「モデル」を参照してください。

カスタマイズ・リバース・エンジニアリング

Oracle で RKM によりカスタマイズ・リバース・エンジニアリングを実行する手順（Oracle リリース 7.3 以上）:

警告: RKM を使用する場合、その RKM をプロジェクトにインポートする必要があります。リバース操作で使用できるのは、インポートされた RKM のみです。

1. Oracle **モデル**の**「リバース」**タブに移動します。
2. 次のフィールドを入力します。
   - **カスタマイズ済**
   - **コンテキスト:** リバース・エンジニアリングに使用するコンテキスト
   - **オブジェクト・タイプ:** リバース・エンジニアリングするオブジェクトのタイプ（表、ビューなど）
   - **マスク:** %
   - **KM の選択:** `RKM Oracle.<name of the importation project>`
3. **「リバース」**ボタンをクリックし、次に**「はい」**をクリックして変更を承認します。
4. **「OK」**をクリックします。
5. **セッションを開始しました**ウィンドウが表示されます。**「OK」**をクリックします。

オペレータで Oracle モデルのリバース・エンジニアリング操作を追跡して確認します。正しく終了すると、リバース操作されたデータストアが、ツリー内のリバース・モジュールの下に表示されます。**「リバース」**および**「選択的リバース」**タブのオプションを使用すると、リバース・エンジニアリングを詳細に設定できます。詳細は、「モデル」を参照してください。この RKM の詳細は、RKM Oracle の説明を参照してください。

Oracle 用の適切な KM の選択

インタフェースまたはチェックに使用する KM の選択により、そのインタフェースまたはチェックの機能およびパフォーマンスが決定されます。次に、Oracle データ・サーバーに関連する様々な環境で KM を選択する際に役立つ推奨事項を示します。

KM の一般的な情報は、「ナレッジ・モジュール」を参照してください。

注意: プロジェクトのインタフェースで使用できるのは、そのプロジェクトにインポートされたナレッジ・モジュールのみです。KM のインポート方法の詳細は、「KM のインポート」を参照してください。

Oracle からの、または Oracle へのデータのロード

Oracle は、インタフェースのソース、ターゲットまたはステージング領域として使用できます。Oracle と別のタイプのデータ・サーバー間でデータをロードするために「インタフェース・フロー」タブで LKM を選択することは、インタフェースのパフォーマンスにとって重要です。

Oracle からロードする際の LKM の選択

次の場合に推奨される LKM です。

- Oracle ソースからステージング領域にロードする場合

複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

ターゲットまたはステージング領域のテクノロジ	推奨 KM	説明
Oracle	LKM Oracle to Oracle (dblink)	ビューをソース・サーバーに、シノニムをターゲットに作成
Oracle	LKM ISO SQL to Oracle	DBLink を使用できない場合
MS SQL Server	LKM ISO SQL to MSSQL (bulk)	SQL Server のバルク・ローダーを使用
Sybase	LKM ISO SQL to Sybase (bcp)	Sybase のバルク・ローダーを使用
すべて	**LKM ISO SQL to SQL**	**汎用 KM**

Oracle にロードする際の LKM の選択

次の場合に推奨される LKM です。

- ソースから Oracle ステージング領域にロードする場合

複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

ソースまたはステージング領域のテクノロジ	推奨 KM	説明
Oracle	LKM Oracle to Oracle (dblink)	ビューをソース・サーバーに、シノニムをターゲットに作成。
JMS	LKM JMS to SQL	JMS はステージング領域として使用不可。
ファイル	LKM File to Oracle (sql*loader)	Oracle のバルク・ローダーを使用。ファイルはステージング領域として使用不可。
ファイル	LKM ISO File to SQL	Oracle のバルク・ローダーより低速。ファイルはステージング領域として使用不可。
すべて	**LKM ISO SQL to Oracle**	**汎用 LKM より高速（統計を使用）。**
すべて	**LKM ISO SQL to SQL**	**汎用 KM。**

Oracle でのデータのチェック

ステージング領域が Oracle 上に存在する場合、静的制御用のみでなく、フロー制御用の Oracle の制御計画を Oracle モデルで選択できます。

Oracle 用の CKM の選択

Oracle でチェックを実行する際に推奨される KM です。複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

推奨 KM	説明
CKM Oracle	レコードの識別に Oracle の行 ID を使用
CKM ISO SQL	**レコードの識別に主キーを使用**

Oracle でのデータの統合

Oracle でのデータ統合計画は、非常に多く存在し、複数のモードが関連します。インタフェースの「フロー」タブでの IKM の選択により、統合におけるパフォーマンスと使用可能な機能が決定されます。

Oracle 用の IKM の選択

選択する統合モードに応じて Oracle で統合を実行する際に推奨される KM です。

特定のターゲットとともにリストされている IKM は、複数テクノロジ対応の KM です。これらは Oracle がステージング領域である場合に使用できます。フロー制御はサポートされません。

複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

モード	ターゲット	推奨 KM	説明
更新		IKM Oracle Incremental Update	Oracle 用に最適化済
更新		**IKM ISO SQL Incremental Update**	**汎用 KM**
固有		IKM Oracle Slowly Changing Dimension	タイプ 2 の緩やかに変化するディメンションをサポート
追加	任意の RDBMS	IKM ISO SQL to SQL Append	フロー制御なし
追加	JMS	IKM ISO SQL to JMS Append	フロー制御なし
追加	ファイル	IKM ISO SQL to File Append	フロー制御なし
追加		**IKM ISO SQL Control Append**	**汎用 KM**

Oracle での一般的エラー

Oracle を原因とするエラーの検出

Oracle Data Integrator でのエラーは、通常、次のように表示されます。

```
java.sql.SQLException: ORA-01017: ユーザー名/パスワードが無効です。ログオンは拒否されました。
at ...
at ...
...
```

java.sql.SQLException というコードは、単に JDBC ドライバを通じて問合せがデータベースに発行され、エラーが戻されたことを示します。このエラーは、データベースまたはドライバのエラーであることが多く、その方向で解釈する必要があります。

太字で示されたテキストの部分のみを最初に検討する必要があります。このテキストを Oracle のドキュメントで検索してください。ここにあるような（赤字で示した）Oracle に固有のエラー・コードが含まれる場合は、エラーをすぐに識別できます。

このようなエラーが実行ログで検出された場合、データベースに送信された SQL コードを分析してエラーの原因を特定する必要があります。このコードは、エラーの発生した**タスク**の**「説明」**タブに表示されます。

次に、Oracle サーバーでよくあるエラーを、その主な原因とともに示します。

一般的なエラー

接続エラー

UnknownDriverException

JDBC ドライバが不適切です。ドライバの名前をチェックしてください。

I/O Exception: Connection refused(DESCRIPTION=(TMP=)(VSNNUM=135290880)(ERR=12505)(ERROR_STACK=(ERROR=(CODE=12505)(EMFI=4))))

JDBC URL のインスタンス名が無効です。

I/O Exception: The Network Adapter could not establish the connection

JDBC URL における Oracle リスナー・ポートの IP アドレスとマシン名が不適切です。

ORA-01017: ユーザー名/パスワードが無効です。ログオンは拒否されました。

データ・サーバー定義に指定されたユーザーまたはパスワード（あるいはその両方）が無効です。このエラーは、特定の Oracle Data Integrator コマンド（SqlUnload など）で発生することもあります。

Protocol violation

このエラーは、Oracle JDBC ドライバと接続先データベース間に互換性がないことを示します。接続時、または Oracle データベースで実行した最初の操作時にこのエラーが発生する場合は、使用中のデータベース・インストールに付属するリリースの Oracle JDBC ドライバをインストールしてください。

ORA-00600: 内部エラー・コード

Oracle データベースの内部エラー。ドライバに互換性がないことが原因の可能性があります。

ORA-12154: TNS: サービス名を解決できませんでした。

TNS 別名の解決。この問題は、OCI ドライバの使用時に、または DBLink を通じた KM の使用時に発生する可能性があります。マシンの TNS 別名の構成をチェックしてください。

ORA-02019: 指定されたリモート・データベースは存在しません。

存在しない DBLink を通じて KM を使用しています。KM のオプションと前提条件をチェックしてください。

インタフェースのエラー

ORA-00900: SQL 文が無効です。
ORA-00923: FROM キーワードが指定の位置にありません。

インタフェースで生成されたコード、またはプロシージャで入力されたコードが Oracle で無効です。この問題は、通常、マッピングまたは結合のフィルタにおける入力エラーに関連しています。よくある原因は、欠落した引用符や閉じられていないカッコです。

他によくある原因は、SQL 以外の構文に対するコール（`EXECUTE SCHEMA.PACKAGE.PROC(PARAM1, PARAM2)`という構文を使用した Oracle ストアド・プロシージャへのコールなど）です。

ストアド・プロシージャに対する有効な SQL コールは、次のとおりです。

```
BEGIN
SCHEMA.PACKAGE.PROC(PARAM1, PARAM2);
END;
```

構文 `EXECUTE SCHEMA.PACKAGE.PROC(PARAM1, PARAM2)`は、SQL*Plus に固有であり、Oracle JDBC シン・ドライバでは動作しません。

ORA-00904: 列名が無効です。

マッピング、結合またはフィルタの指定エラーです。列名ではない文字列が列名として解釈されたか、列名のスペルが間違っています。

このエラーは、最近変更された構造を含むデータストアに関連するエラー表にアクセスする場合にも発生する可能性があります。エラー表に変更内容を反映させるか、エラー表を削除して次回実行時に Oracle Data Integrator でエラー表を再作成する必要があります。

ORA-00903: 表名が無効です。

使用される表（ソースまたはターゲット）が、Oracle スキーマに存在しません。コンテキストに対応するマッピング論理スキーマおよび物理スキーマをチェックし、そのコンテキストでアクセスされるスキーマに表が物理的に存在していることを確認してください。

ORA-00972: 識別子が長すぎます。

Oracle のオブジェクト識別子には制限があります（通常は 30 文字）。この制限を超えると、このエラーが発生します。インタフェースの実行時に作成された表がこの制限を超えた場合、このエラーが発生します（詳細は実行ログを参照してください）。

Oracle テクノロジのトポロジで、オブジェクト名（表および列）の最大長が現在の Oracle 構成に対応していることを確認してください。

ORA-01790: 式には対応する式と同じデータ型を持つ必要があります。

（マッピングや結合などで）暗黙的に変換できない異なる 2 つの値を結合しようとしています。これらの値には明示的な変換関数を使用してください。

DB2 for iSeries

DB2/400 データ・サーバーの作成

DB2/400 **データ・サーバー**は、特定のユーザー・アカウントに接続され、AS/400 システムにインストールされた 1 つの DB2 データベースに対応します。このユーザーは、このシステムの**ライブラリ**（データ・サーバーの下に作成された Oracle Data Integrator の**物理スキーマ**に対応）にアクセスします。

前提条件

JDBC ドライバ

タイプ 4 の DB2/400 用 JDBC ドライバを使用するのが適切です。**IBM JT/400** ドライバまたは **HiT JDBC/400** ドライバをお薦めします。これらのドライバでは、TCP/IP ネットワーク・レイヤーを直接使用するため、他のコンポーネントをクライアント・マシンにインストールする必要はありません。

ODBC を通じて、マシンにインストールされた IBM クライアント・アクセス・コンポーネントと接続することも可能です。この方法では、あまり高いパフォーマンスを得ることはできず、リバース・エンジニアリングなどのいくつかの機能もサポートされません。そのため、この方法はお薦めしません。

IBM JT/400 およびネイティブ・ドライバ

このドライバは、Oracle Data Integrator のインストール・ディレクトリの `/drivers` サブディレクトリにコピーする必要のある `jt400.zip` ファイルに含まれます。Oracle Data Integrator では、このドライバのデフォルト・バージョン（Oracle Data Integrator でテストおよび検証済）がインストールされます。このドライバは無償です。

AS/400 マシンにインストールされた Java アプリケーションと DB2/400 を接続する場合、IBM 社では、JT/400 ドライバ（jt400.jar）のかわりに JT/400 ネイティブ・ドライバ（jt400native.jar）を使用することを推奨しています。ネイティブ・ドライバでは、AS/400 に対するアクセスが最適化されますが、このドライバは AS/400 でのみ動作します。

1 つの接続で両方のドライバをシームレスにサポートする目的で、Oracle Data Integrator では、**Oracle Data Integrator Driver Wrapper for AS/400** を提供しています。このラッパーは、可能であればネイティブ・ドライバを使用して接続を行い、それ以外の場合は JT/400 ドライバを使用します。AS/400 システムにインストールされたエージェントが稼働している場合は、このラッパーを使用することをお薦めします。

ドライバ・ラッパーをインストールして構成する手順:

1. `snpsdb2.jar` ファイル（ラッパー・パッケージ）がすべてのマシンのドライバとしてインストールされていることを確認します。詳細は、「JDBC および JMS ドライバのインストール」を参照してください。
2. トポロジ・マネージャで、次の情報を使用してドライバおよび URL を AS/400 サーバー用に変更します。
   - ドライバ: `com.sunopsis.jdbc.driver.wrapper.SnpsDriverWrapper`
   - URL:
     `jdbc:snps400:<machine_name>[;param1=valeur1[;param2=valeur2...]]`

3. AS/400 システムの Oracle Data Integrator インストール環境で、エージェント起動コマンドに次の Java プロパティを設定し、エージェントを再起動します。
    - HOST_NAME: 現在のマシンの AS/400 ホスト名のカンマ区切りリスト
    - HOST_IP: 現在のマシンの IP アドレス

JAVA CL コマンドで起動したエージェントのプロパティを設定する場合、 次の構文例を使用できます。

```
JAVACLASS('oracle.odi.Agent') +
     PARM('-SECU_DRIVER=com.ibm.as400..access.AS400JDBCDriver'... '-
     PORT=20910') +
     CLASSPATH('/ODI/LIB/odi.zip:/ODI/DRIVERS/snpsdb2.jar:/ODI/DRIVERS/j
     t400native.jar'...) +
     ... +
     PROP((HOST_NAME 'HAL,HALM,HALW,HALE') (HOST_IP '192.168.0.13'))
```

snpsdb2.jar（ドライバ・ラッパー）と jt400native.jar（ネイティブ・ドライバ）のファイルがクラスパスに含まれ、最後の行でプロパティを指定しています。

UNIX シェル・スクリプトを使用してエージェントを起動する場合、次のように odiparams.sh を編集します。

```
ODI_ADDITIONAL_JAVA_OPTIONS="-DHOST_NAME=HAL,HALM,HALW,HALE -
HOST_IP=192.168.0.13"
```

すべてのドライバ・パラメータのリストは、「JDBC ドライバ・サンプル」を参照してください。

HiT JDBC/400

このドライバは、JT/400 ドライバと同様です。このドライバのライセンスは、Oracle Data Integrator とは別に供与されます。詳細は、www.hitsw.com を参照してください。

接続情報

システム管理者から次の情報を入手する必要があります。

- AS/400 システムのネットワーク名または IP アドレス
- AS/400 ユーザーのログインおよびパスワード

データ・サーバーの作成

DB2/400 データ・サーバーを作成する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで**「トポロジ」→「物理アーキテクチャ」→「テクノロジ」→「IBM DB2/400」**を選択します。
3. 右クリックし、**「データ・サーバーの挿入」**を選択します。
4. **「定義」**タブで、次のフィールドを入力します。
    - **名前:** Oracle Data Integrator に表示されるデータ・サーバーの名前
    - **ユーザー/パスワード:** 少なくともデータ・ライブラリに対する選択権限と、このデータ・サーバーの下に作成される**物理スキーマ**で示される作業ライブラリに対する選択、挿入およびオブジェクト作成権限を持つパスワード付きの AS/400 ユーザー

5. 「**JDBC**」タブで、使用するドライバに応じて次のフィールドを入力します。

   IBM JT/400 ドライバを使用する場合:

   - **JDBC ドライバ:** `com.ibm.as400.access.AS400JDBCDriver`
   - **JDBC URL:** `jdbc:as400://<network name or IP address of the AS/400 machine>`

   HiT JDBC/400 ドライバを使用する場合:

   - **JDBC ドライバ:** `hit.as400.As400Driver`
   - **JDBC URL:** `jdbc:as400://<network name or IP address of the AS/400 machine>`

6. 「**テスト**」をクリックします。
7. **テスト接続**ウィンドウの「**テスト**」をクリックします。
8. **接続に成功**したことを示すウィンドウが表示されます。「**OK**」をクリックします。接続に失敗した場合は、「DB2/400 での一般的エラー」を参照してください。
9. 「**OK**」をクリックしてデータ・サーバーの作成を承認します。

このデータ・サーバーの最初の**物理スキーマ**に対応する作成ウィンドウが表示されます。「DB2/400 の物理スキーマの作成」を参照してください。

DB2/400 の物理スキーマの作成

物理スキーマは、DB2/400 における次のライブラリ（またはコレクション）のペアに対応します。

- （データ）**スキーマ:** Oracle Data Integrator は、このスキーマでインタフェースのソースおよびターゲット表（ファイル）を検索します。
- **作業スキーマ:** Oracle Data Integrator は、このスキーマで、データ・ライブラリに格納されたソースとターゲットに関連する一時作業表を作成および操作します。

注意: 物理スキーマに関連するデータ・サーバーに指定されたユーザー・プロファイルは、前述の操作を実行するための適切な権限を付与されている必要があります。

注意: ほとんどのナレッジ・モジュールでは、ステージング領域で、およびターゲット・データストアへの書込み時に、コミット制御を使用します。AS/400 のステージング領域として**コレクション**を使用し、ターゲット表をジャーナル化することを強くお薦めします。

物理スキーマの作成

DB2/400 の物理スキーマを作成する手順:

注意: データ・サーバーの作成直後の場合、手順 1 は無視してください。**物理スキーマ**・ウィンドウがすでに表示されているはずです。

1. DB2/400 **データ・サーバー**を選択して右クリックし、「**物理スキーマの挿入**」を選択します。**物理スキーマ**・ウィンドウが表示されます。
2. 「**スキーマ（スキーマ）**」で、この Oracle Data Integrator の物理スキーマのデータ・ライブラリを選択します。このフィールドには、DB2/400 データベースのライブラリ・リストが表示されます。

3. **「スキーマ（作業スキーマ）」**で、この Oracle Data Integrator の物理スキーマの作業ライブラリを選択します。このフィールドには、DB2/400 データベースのライブラリ・リストが表示されます。
4. このスキーマをこのデータ・サーバーのデフォルト・スキーマに設定する場合、**「デフォルト」**ボックスを選択します（最初の物理スキーマは、常にデフォルト・スキーマになります）。詳細は、「物理スキーマ」を参照してください。
5. **「コンテキスト」**タブに移動します。
6. この新規**物理スキーマ**に対応する**「コンテキスト」**と既存の**「論理スキーマ」**を選択し、手順 9 に進みます。
   DB2/400 の論理スキーマが存在しない場合は、手順 7 に進みます。
7.  ボタンをクリックします。
8. 左側の列で既存の**「コンテキスト」**を選択し、右側の行に**「論理スキーマ」**の名前を入力します。この DB2/400 の論理スキーマは、自動的に作成され、このコンテキスト内の物理スキーマに関連付けられます。

警告: 特定の**コンテキスト**では、**論理スキーマ**は 1 つの**物理スキーマ**にのみ関連付けることができます。

9. 「OK」をクリックします。

DB2/400 モデルの作成およびリバース・エンジニアリング

DB2/400 モデルの作成

DB2/400 **モデル**は、DB2/400 ライブラリに格納されたビュー、表およびファイルに対応するデータストアのセットです。モデルは、常に**論理スキーマ** に基づきます。特定の**コンテキスト**内で、**論理スキーマ**は単一の**物理スキーマ**に対応します。この物理スキーマのデータ・スキーマには、DB2/400 モデルの表、ファイルおよびビューが格納されます。

DB2/400 モデルを作成する手順:

1. **デザイナ**に接続します。
2. ツリーで**「モデル」**を選択します。
3. 右クリックし、**「モデルの挿入」**を選択します。
4. **「定義」**タブで、**「名前」**フィールドを入力します。
5. **「テクノロジ」**フィールドで、**「IBM DB2/400」**を選択します。
6. **「論理スキーマ」**フィールドで、モデルの基礎となる論理スキーマを選択します。
7. **「リバース」**タブに移動し、モデルのリバース・エンジニアリングに使用する**「コンテキスト」**を選択します。**「適用」**をクリックします。

これでモデルは作成されましたが、データストアはまだ格納されていません。

DB2/400 モデルのリバース・エンジニアリング

モデルは、データストアなしで作成されます。**リバース・エンジニアリング**操作では、モデルの表、ファイルおよびビューの構造を取得して、モデルのデータストア定義を作成します。リバース・エンジニアリングのタイプには、ドライバの機能のみを使用する**標準**リバース・エンジニア

リングと、RKM（リバース・ナレッジ・モジュール）を使用して DB2/400 ディクショナリから直接オブジェクトの構造を取得する**カスタマイズ・**リバース・エンジニアリングという 2 つのタイプがあります。

標準リバース・エンジニアリング

DB2/400 で標準リバース・エンジニアリングを実行する手順:

1. DB2/400 **モデル**の**「リバース」**タブに移動します。
2. 次のフィールドを入力します。
   - **標準**
   - **コンテキスト:** リバース・エンジニアリングに使用するコンテキスト
   - **リバースエンジニアリングするオブジェクトの型:** リバース・エンジニアリング・プロセスで処理する必要のあるオブジェクト・タイプのリスト
3. **「選択的リバース」**タブに移動します。
   - **「選択的リバース」**、**「新規データストア」**および**「リバースするオブジェクト」**の各ボックスを選択します。
4. リバース・エンジニアリング対象のデータストアのリストが表示されます。リバース・エンジニアリングするデータストアについては、選択したままとします。
5. **「リバース」**ボタンをクリックし、次に**「はい」**をクリックして変更を承認します。
6. Oracle Data Integrator により、選択したデータストアのリバース・エンジニアリング・プロセスが開始されます。プログレス・バーが表示されます。

リバース・エンジニアリングされたデータストアが、モデルの下に表示されます。

「リバース」および**「選択的リバース」**タブのオプションを使用すると、リバース・エンジニアリングを詳細に設定できます。詳細は、「モデル」を参照してください。

DB2/400 用の適切な KM の選択

インタフェースまたはチェックに使用する KM の選択により、そのインタフェースまたはチェックの機能およびパフォーマンスが決定されます。次に、DB2/400 サーバーに関連する様々な環境で KM を選択する際に役立つ推奨事項を示します。

KM の一般的な情報は、「ナレッジ・モジュール」を参照してください。

注意: プロジェクトのインタフェースで使用できるのは、そのプロジェクトにインポートされたナレッジ・モジュールのみです。KM のインポート方法の詳細は、「KM のインポート」を参照してください。

DB2/400 からの、または DB2/400 へのデータのロード

DB2/400 は、インタフェースのソース、ターゲットまたはステージング領域として使用できます。DB2/400 と別のタイプのデータ・サーバー間でデータをロードするためにインタフェースの「フロー」タブで LKM を選択することは、インタフェースのパフォーマンスにとって重要です。

DB2/400 からロードする際の LKM の選択

次の場合に推奨される LKM です。

- DB2/400 ソースからステージング領域にロードする場合

複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

ターゲットまたはステージング領域のテクノロジ	推奨 KM	説明
Oracle	LKM ISO SQL to Oracle	汎用 LKM より高速（統計を使用）
MS SQL Server	LKM ISO SQL to MSSQL (bulk)	SQL Server のバルク・ローダーを使用
Sybase	LKM ISO SQL to Sybase (bcp)	Sybase のバルク・ローダーを使用
すべて	**LKM ISO SQL to SQL**	**汎用 KM**

DB2/400 にロードする際の LKM の選択

次の場合に推奨される LKM です。

- ソースから DB2/400 ステージング領域にロードする場合

複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

ソースまたはステージング領域のテクノロジ	推奨 KM	説明
JMS	LKM ISO JMS to SQL	JMS はステージング領域として使用不可
ファイル	LKM ISO File to SQL	ファイルはステージング領域として使用不可
すべて	**LKM ISO SQL to SQL**	**汎用 KM**

DB2/400 でのデータのチェック

ステージング領域が DB2/400 上に存在する場合、静的制御用のみでなく、フロー・コントロール用の DB2/400 の制御計画を DB2/400 モデルで選択できます。

DB2/400 用の CKM の選択

DB2/400 でチェックを実行する際に推奨される KM です。複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

推奨 KM	説明
CKM ISO SQL	**レコードの識別に主キーを使用**

DB2/400 でのデータの統合

DB2/400 でのデータ統合計画は、非常に多く存在し、複数のモードが関連します。インタフェースの「フロー」タブでの IKM の選択により、統合におけるパフォーマンスと使用可能な機能が決定されます。

DB2/400 用の IKM の選択

選択する統合モードに応じて DB2/400 で統合を実行する際に推奨される KM です。

特定のターゲットとともにリストされている IKM は、複数テクノロジ対応の KM です。これらは DB2/400 がステージング領域である場合に使用できます。フロー制御はサポートされません。

複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

モード	ターゲット	推奨 KM	説明
更新		IKM DB2/400 Incremental Update	DB2/400 用に最適化済
更新		**IKM ISO SQL Incremental Update**	**汎用 KM**
追加	任意の DBMS	IKM ISO SQL to SQL Append	フロー制御なし
追加	JMS	IKM ISO SQL to JMS Append	フロー制御なし
追加	ファイル	IKM ISO SQL to File Append	フロー制御なし
追加		**IKM ISO SQL Control Append**	**汎用 KM**

iSeries での CDC の使用方法

Oracle Data Integrator では、iSeries におけるチェンジ・データ・キャプチャ（CDC）は次の 2 つの方法で処理されます。

- ジャーナル化表で**トリガーを使用する方法**。この方法は、「JKM DB2/400 簡易」または「JKM DB2/400 一貫性」で設定します。この CDC は、他のシステムでの CDC と同じです。詳細は、「チェンジ・データ・キャプチャ」を参照してください。
- **ネイティブ iSeries トランザクション・ジャーナルを読み取る方法**。この方法は、「JKM DB2/400 ジャーナル簡易」で設定し、「LKM DB2/400 Journal to SQL」で使用します。この方法では、通常、パフォーマンスも期待できますが、整合セット CDC はサポートされず、後述のプラットフォーム固有の構成が必要になります。

動作の仕組み

iSeries のトランザクション・ジャーナルには、一定期間のデータ変更の履歴全体が含まれます。トランザクション・ジャーナルは、ジャーナル化される表で iSeries システムにより処理されます。ジャーナル化表は、コレクションに基づく表か、またはジャーナル・レシーバおよびジャーナルが作成されてジャーナル化が開始された表です。

トランザクション・ジャーナルの読取りは、Oracle Data Integrator に付属するジャーナル取得用の CDCRTVJRN RPG プログラムにより実行されます。このプログラムは、必要時に Oracle Data Integrator CDC インフラストラクチャの表（J$表）をその内容とともにトランザクション・ジャーナルからロードします。

このプログラムは、iSeries システムでスケジュールするか、CDCRTVJRN という同じ名前のストアド・プロシージャを通じて KM によりコールすることができます。
このストアド・プロシージャは、「JKM DB2/400 ジャーナル簡易」によって自動的に作成され、データ抽出が必要な場合に「LKM DB2/400 Journal to SQL」によって起動されます。

CDCRTVJRN プログラムの詳細

このプログラムは、特定の表のネイティブ iSeries ジャーナルに接続し、Oracle Data Integrator ジャーナル（J$）内に変更データ情報を取得します。

このプログラムは、次のように動作します。

1. ジャーナル化表の属性の取得:
   a. 表の属性の取得: PK 列、J$表の名前、最終ジャーナル読取り日時
   b. 短縮名やレコード・サイズなどの拡張属性（QSYS.QADBXREF システム表を使用）
   c. iSeries ジャーナルの場所（QADBRTVFD() API を使用）
2. PK 列情報の取得:
   a. 短縮名やデータ型などの PK 列の属性（QSYS.QADBIFLD システム表を使用）
   b. 実際の物理長などの拡張属性（QUSLFLD() API を使用）
   c. 主キー列のデータの前処理（RPG から SQL へのデータ型変換）
3. J$表へのネイティブ・ジャーナル情報の抽出:
   a. ネイティブ・ジャーナルの読取り（QJoRetrieveJournalEntries() API を使用）
   b. RAW データのネイティブ SQL データへの変換および J$表への取得
   c. 変更回数の更新

このプログラムでは次のパラメータを使用します。

パラメータ	RPG タイプ	SQL タイプ	説明
SbsTName	A138	Char(138)	<Lib>.<Table>という形式のサブスクライバ表の完全名。 例: ODILIB.SNP_SUBSCRIBERS
JrnTName	A138	Char(138)	ジャーナルからの抽出を実行する表の完全名。 例: FINANCE.MY_COMPANY_ORDERS

JrnSubscriber	A50	Char(50)	現在のサブスクライバの名前。通常、事前にサブスクライバのリストに追加されています。
LogMessages	A1	Char(1)	スプール・ファイルでのロギングをアクティブ化するフラグ。可能な値は、Y（ロギングの有効化）と N（ロギングの無効化）です。

iSeries への CDC コンポーネントのインストール

ネイティブのジャーナル読取りを可能にするために、iSeries システムにインストールする主なコンポーネントは次の 2 つです。

- **CDCRTVJRN プログラム**。このプログラムは、iSeries システムにインストールされているアーカイブに含まれます。インストール・プロセスは、後述の説明を参照してください。
- **CDC インフラストラクチャ**。これには、標準の CDC オブジェクト（J$表やビューなど）と、JKM により作成され、LKM によりジャーナルの読取りに使用される **CDCRTVJRN ストアド・プロシージャ**が含まれます。このストアド・プロシージャは、CDCRTVJRN プログラムを実行します。

重要: このプログラムは、この iSeries データ・サーバーのデフォルト作業ライブラリとしてトポロジに定義されたライブラリに導入する必要があります。後述の例では、このライブラリは ODILIB と呼ばれます。

CDCRTVJRN プログラムのインストール

CDCRTVJRN プログラムをインストールする手順:

1. プログラム保管ファイルを/tools/cdc_iseries/ディレクトリからシステムの一時ディレクトリ（C:¥temp）にコピーします。この例では、このコピーを SAVESNPCDC と名付けます。
2. 端末を使用して iSeries に接続します。
3. まだ存在しない場合、デフォルト作業ライブラリを作成します。
   例: CRTLIB LIB(ODILIB)
4. このライブラリに、保管ファイルと同じ名前を持つ空の保管ファイルを作成します（必須）。
   例: CRTSAVF FILE(ODILIB/SAVESNPCDC)
5. iSeries システムのライブラリにローカルの保管ファイルをアップロードし、前の手順で作成した保管ファイルを上書きします。アップロードは必ずバイナリ・モードで実行してください。次に、アップロードを実行する際の一連の FTP コマンドを示します。

   ```
   FTP 192.168.0.13
   LCD C:¥TEMP
   BI
   CD ODILIB
   PUT SAVESNPCDC
   BYE
   ```

6. 次のように RSTOBJ コマンドを使用して、保管ファイルからターゲット・ライブラリに CDCSNPRELE ライブラリのオブジェクトをリストアします。

   ```
   RSTOBJ OBJ(*ALL) SAVLIB(CDCSNPRELE) DEV(*SAVF) OBJTYPE(*ALL)
   SAVF(ODILIB/SAVESNPCDC) RSTLIB(ODILIB)
   ```

7. オブジェクトが正しくリストアされたことを確認します。ターゲット・ライブラリには、CDCRTVJRN というプログラム・オブジェクトが含まれている必要があります。次のコマンドを使用するとこのオブジェクトを表示できます。
   `WRKOBJ OBJ(ODILIB/CDCRTVJRN)`

CDCRTVJRN ストアド・プロシージャ

このプロシージャは、CDCRTVJRN プログラムをコールするために使用します。このプロシージャは、ジャーナル化の起動時に「JKM DB2/400 ジャーナル簡易」ナレッジ・モジュールにより自動的に作成されます。ジャーナル化の起動の詳細は、「チェンジ・データ・キャプチャ」を参照してください。

参考用として、このストアド・プロシージャの構文を次に示します。

```
create procedure ODILIB.CDCRTVJRN(
     SbsTName  char(138), /* Qualified Subscriber Table Name */
     JrnTName  char(138), /* Qualified Table Name            */
     Subscriber char(50) , /* Subscriber Name                 */
     LogMsg    char(1)   /* Create a Log (Y - Yes, N - No) */
)
language rpgle
external name 'ODILIB/CDCRTVJRN'
```

重要: このストアド・プロシージャとプログラムは、この iSeries データ・サーバーのデフォルト作業ライブラリとしてトポロジに定義されたライブラリにインストールされます。

ネイティブ・ジャーナルでの CDC の使用方法

プログラムをインストールして CDC を設定したら、ネイティブ・ジャーナルを使用するために「LKM DB2/400 Journal to SQL」を使用して iSeries システムからジャーナル化データを抽出します。取得プロセスが起動されるのは、LKM で `RETRIEVE_JOURNAL_ENTRIES` オプションが Y に設定されている場合です。

ジャーナル読取り時の問題

CDCRTVJRN プログラムの制限

CDCRTVJRN プログラムには、次の制限が存在します。

- サポートされる iSeries のバージョンは、v5r2 です。その他のバージョンについては、サポートに連絡してください。
- ソース表はジャーナル化する必要があります。また、iSeries ジャーナルは、iSeries データ・サーバーに指定されたユーザーにより読取り可能である必要があります。
- ソース表には、Oracle Data Integrator で定義された 1 つの PK が含まれる必要があります。
- Oracle Data Integrator で宣言された PK は、データファイルの物理レコードに含まれる最初の 4096 オクテット内に存在する必要があります。
- PK の列数は、16 以下である必要があります。
- PK の列数に追加される PK 列名の合計文字数は、255 以下である必要があります。

- ラージ・オブジェクト・データ型は、PK ではサポートされません。PK でサポートされる SQL タイプは、SMALLINT、INTEGER、BIGINT、DECIMAL（パック形式）、NUMERIC（ゾーン形式）、FLOAT、REAL、DOUBLE、CHAR、VARCHAR、CHAR VARYING、DATE、TIME、TIMESTAMP および ROWID のみです。
- CDCRTVJRN の複数のインスタンスを同じシステムで同時に起動することはできません。
- iSeries ジャーナルで順序番号を再初期化する際に、ジャーナル・エントリ使用日付（SNP_SUBSCRIBERS.JRN_CURFROMDATE）が順序初期化日付より前であると、プログラムに重大な問題が発生する可能性があります（プログラムのハングなど）。この問題を回避するには、より新しい日付を手動で SNP_SUBSCRIBERS.JRN_CURFROMDATE に設定する必要があります。

CDCRTVJRN プログラムのトラブルシューティング

ジャーナル読取りプロセスは、次の方法でトレース・モードに移行できます。

- 問合せツールから、LogMsg パラメータを Y に設定して CDCRTVJRN ストアド・プロシージャをコールします。
- CREATE_SPOOL_FILE LKM オプションを強制的に 1 に設定し、インタフェースを再起動します。

読取りプロセスのログは、WRKSPLF コマンドを使用して参照できるスプール・ファイルに格納されます。

DSPJRN コマンドを使用すると、iSeries ジャーナルの RAW データも参照できます。

DB2/400 での一般的エラー

エラー・メッセージの解釈

Oracle Data Integrator でのエラーは、通常、次のように表示されます。

ODBC ドライバのエラー・メッセージ

```
java.sql.SQLException: [IBM][Client Access ODBC Driver][32 bits][DB2/400
SQL]Communication link failure.Comm RC=0xb
at sun.jdbc.odbc.JdbcOdbc.createSQLException(Unknown Source)
at sun.jdbc.odbc.JdbcOdbc.standardError(Unknown Source)
...
```

IBM JT/400 ドライバのエラー・メッセージ

```
java.sql.SQLException: The application server rejected the
connection.(Signon was canceled.)
at com.ibm.as400.access.JDError.throwSQLException(JDError.java:336)
at
com.ibm.as400.access.AS400JDBCConnection.setProperties(AS400JDBCConnecti
on.java:1984)
...
```

HiT JDBC/400 ドライバのエラー・メッセージ

```
java.sql.SQLException: Cannot open a socket on host: ha, port: 8471
(Exception: java.net.UnknownHostException: ha).
at hit.as400sql.d.<init>([DashoPro-V1.3-013000])
at hit.as400.As400Driver.newConnection([DashoPro-V1.3-013000])
...
```

java.sql.SQLException というコードは、単に JDBC ドライバ（または JDBC/ODBC ブリッジ）を通じて問合せがデータベースに発行され、エラーが戻されたことを示します。このエラーは、データベースまたはドライバのエラーであることが多く、その方向で解釈する必要があります。

太字で示されたテキストの部分のみを最初に検討する必要があります。このテキストをドライバまたはデータベースのドキュメントで検索してください。DB2/400 に固有のエラー・コードが含まれる場合は、エラーをすぐに識別できます。

このようなエラーが実行ログで検出された場合、データベースに送信された SQL コードを分析してエラーの原因を特定する必要があります。このコードは、エラーの発生した**タスク**の**「説明」**タブに表示されます。

次に、DB2/400 サーバーでよくあるエラーを、その主な原因とともに示します。

一般的なエラー

接続エラー

UnknownDriverException

JDBC ドライバが不適切です。ドライバの名前をチェックしてください。

The application requester cannot establish the connection.(<name or IP address>)
Cannot open a socket on host: <name or IP address>, port: 8471 (Exception: java.net.UnknownHostException:<name or IP address>)

Oracle Data Integrator でデータベースに接続できません。マシン名または IP アドレスが無効であるか、DB2/400 サービスまたは AS/400 の TCP/IP インタフェースが起動していません。
同じマシン名または IP アドレスを使用して AS/400 マシンに ping を試行し、システム管理者に依頼して適切なサービスが起動していることを確認してください。

Datasource not found or driver name not specified

JDBC URL に指定された ODBC データソースが不適切です。

The application server rejected the connection.(Signon was canceled.)
Database login failed, please verify userid and password.
Communication Link Failure.Comm RC=8001 - CWBSY0001 - ...

使用しているユーザー・プロファイルが無効です。このエラーは、無効なユーザー名または不適切なパスワードを入力したときに発生します。

Communication Link Failure.

ODBC 接続でエラーが発生しました。詳細は、クライアント・アクセスに関するドキュメントを参照してください。

インタフェースのエラー

SQL7008 &1 in &2 not valid for operation.The reason code is 3.

iSeries 400 システムでは、ジャーナル化によるコミット制御を実装しています。コミット制御を利用するアプリケーションでは、使用するファイルをジャーナル化する必要があります。ほとんどのナレッジ・モジュールでは、ステージング領域で、およびターゲット・データストアへの書込み時に、コミット制御を使用します。AS/400 のステージング領域として**コレクション**を使用し、ターゲット表をジャーナル化することを強くお薦めします。ナレッジ・モジュールを変更することで、ナレッジ・モジュールでのコミット制御の使用を停止できます。

SQL5001 - Column qualifier or table &2 undefined.
SQL5016 - Object name &1 not valid for naming convention

JDBC 接続または ODBC データソースの構成で不適切なネーミング規則を使用しています。ODBC 管理者に依頼して正しいネーミング規則（*SQL または*SYS）を使用するようデータソースを変更するか、JDBC URL の適切なオプションを使用して正しいネーミング規則を強制します（`jdbc:as400://195.10.10.13;naming=system` など）。**物理スキーマ**の**ローカル・オブジェクト・マスク**でシステム・ネーミング規則を使用する場合、`%SCHEMA.%OBJECT` のかわりに`%SCHEMA/%OBJECT` を入力する必要があります。

現在のアプリケーションが*SYS を使用するよう特に構成されていないかぎり、常に*SQL を使用する必要があります。Oracle Data Integrator では、デフォルトで*SQL ネーミング規則が使用されます。

SQL0204 &1 in &2 type *&3 not found.

アクセスしようとしている表は存在しません。これは、コンテキスト選択または操作順序におけるエラーに関連する可能性があります（この表が、別のインタフェースによって作成される必要のある一時表の場合など）。

Hexadecimal characters appear in the target tables.Accentuated characters are incorrectly transferred.

iSeries コンピュータでは、言語 ID または CCSID をファイル、表およびフィールド（列）に関連付けます。CCSID 65535 は、ファイルまたはフィールドが言語非依存であることを示す汎用コードです（16 進データなど）。したがって、ドライバによる変換は実行されません。ファイルの CCSID を更新しない場合、`ccsid=<ccsid code>`および `convert _ccsid_65535=yes|no` というフラグに基づいて、JDBC URL で変換が強制される可能性があります。詳細は、ドライバのドキュメントを参照してください。

SQL0901 SQL system error

このエラーは、DB2/400 システムの内部エラーです。

SQL0206 Column &1 not in specified tables.

マッピング、結合またはフィルタの指定エラーです。列名ではない文字列が列名として解釈されたか、列名のスペルが間違っています。

このエラーは、最近変更された構造を含むデータストアに関連するエラー表にアクセスする場合にも発生する可能性があります。エラー表に変更内容を反映させるか、エラー表を削除して次回実行時に Oracle Data Integrator でエラー表を再作成する必要があります。

iSeries および AS/400 への Java エージェントのインストール

一部のデータベースは、iSeries および AS400 のマシンにインストールされています。Oracle Data Integrator の実行エージェントをこれらのマシンにインストールし、次のことを実行できます。

- AS/400 でのロード・プロセスの実行
- OS400 のシステム・コマンドの実行
- ネットワーク・フローの削減（AS/400 にソースとターゲットの両方が存在する場合）

AS/400 マシンに Oracle Data Integrator の実行エージェントを実装するためのインストール手順は、次のとおりです。

システムの準備

技術上の前提条件

- 次の PTF が適用された AS/400、AS/400 V5R1 または V4R4M0:
  - SF61800
  - SF55849
  - SF54922
- インストールするプログラム:
  - IBM Toolbox for Java
  - Java 仮想マシン 1.3.1
- AS/400 マシンで構成および起動された TCP/IP サービス

コンポーネント

使用する Java および JDBC コンポーネントは、次のとおりです。

- Oracle Data Integrator 実行エージェント
- Java 仮想マシン 1.3.1
- データベース・アクセス用の JDBC ドライバ

ドライバに関する注意: AS/400 から DB2/400 に接続する場合、AS/400 用のドライバ・ラッパーの使用を検討する必要があります。詳細は、「DB2/400 データ・サーバーの作成」を参照してください。

実行エージェントのインストール

ファイルのインストール

1. AS/400 に Oracle Data Integrator ファイルを格納するためのディレクトリ・ツリーを作成します。

```
MKDIR DIR('/odi')
MKDIR DIR('/odi/bin')
MKDIR DIR('/odi/lib')
MKDIR DIR('/odi/lib/scripting')
MKDIR DIR('/odi/drivers')
```

2. Oracle Data Integrator の CD の/oracledi ディレクトリから、手順 1 で作成した AS/400 のディレクトリに（FTP などを使用して）ファイルを転送します。
コピーするファイルの場所は、次のとおりです。

```
/oracledi/bin/
/oracledi/lib/
/oracledi/lib/scripting（Oracle Data Integrator のスクリプト機能を使用する場合）
/oracledi/drivers（使用するドライバのみコピーが必要）
```

Java プログラムの作成

AS/400 でのパフォーマンス上の理由から、Java パッケージ（.class、.jar または.zip ファイル）は **Java プログラム**に変換する必要があります。

.class、.jar または.zip ファイルから Java プログラムを作成する手順:

1. 次のように CL コマンドを実行します。

```
CRTJVAPGM CLSF('<.class, .zip or .jar file
location>')OPTIMIZE(40)
```

たとえば、odi.zip ファイル（Java エージェント）から Java プログラムを作成するには、次のコマンドを実行します。

```
CRTJVAPGM CLSF('/odi/lib/odi.zip') OPTIMIZE(40)
```

注意: Java プログラムの作成は、Java パッケージの内容によっては長い時間がかかります。実際に使用するクラスのプログラムのみを作成することをお薦めします。

実行エージェントの起動

実行エージェントは、次の 2 つの方法で起動できます。

- **シェル・インタプリタ（QSH または STRQSH）の使用:** これは、UNIX 互換シェルを実装する OS/400 のオプションです。多くの UNIX コマンド（ls、chmod、chown など）に加え、java コマンドも使用できます。Oracle Data Integrator の/bin に含まれる UNIX スクリプト（.sh 拡張子）で標準の構文を使用することで、Oracle Data Integrator を起動できます。エージェントを実行する前に、odiparams.sh ファイルを構成する必要があります。AS/400 では、グラフィカル・モジュール（トポロジやデザイナなど）を使用できます。
- **OS/400 コマンド（CL）の使用:** RUNJVA または JAVA CL コマンドで Java アプリケーションを実行します。エージェントやシナリオを起動する場合、CL プログラムを使用すると便利です。これらの操作用のプログラム・テンプレートを次に示します。

JVM バージョンに関する注意: 複数の Java マシンが AS/400 にインストールされている場合、状況によっては Java コマンドで使用される Java のバージョンを強制的に指定する必要があります。
- **QSH** の場合: バージョンを強制するには、Java コマンドのフラグ-Djava.version=<java version>（-Djava.version=1.3.1 など）を使用します。
- **OS/400 コマンド**の場合: JAVA コマンド・パラメータとして-PROP((<property> <value>) (<property> <value>))のようにプロパティを渡します。たとえば、-PROP((java.version 1.3.1)のように指定します。

エージェントの名前付けに関する注意: 通常、AS/400 では、エージェントをリスナーとして実行する際にそのエージェントの名前を明示的に指定します。そのため、エージェントの実行時には、エージェントのフラグ-name=<agent name>を使用する必要があります。

エージェントの実行

```
             PGM        PARM(&NAME &PORT &VERB)
/* Command AGENT */
/* Parameters: */
/*   &NAME: physical name of the agent */
```

```
/*   &PORT: port number */
/*   &VERB: verbose mode -V=[1..5] */
/* Example of call: */
/*   CALL PGM(<myLib/myPGM>) PARM('-NAME=myAgt' '-PORT=20910' '-V=5') */
             DCL        VAR(&NAME) TYPE(*CHAR) LEN(128)
             DCL        VAR(&PORT) TYPE(*CHAR) LEN(30)
             DCL        VAR(&VERB) TYPE(*CHAR) LEN(30)
/* All classes below should be compiled */
/* with the CRTJVAPGM command */
/* with optimize 40. */
             DCL        VAR(&PROJ) TYPE(*CHAR) LEN(512) +
                          VALUE('/odi/lib/odi.zip:+
                          /odi/lib/sunjce_provider.jar:+
                          /odi/lib/commons-net.jar:+
                          /odi/lib/local_policy.jar:+
                          /odi/lib/jakarta-ant-optional.jar:+
                          /odi/lib/US_export_policy.jar:+
                          /odi/lib/jce1_2_2.jar')
/* Replace the drivers below with your own drivers. */
             DCL        VAR(&JDBC) TYPE(*CHAR) LEN(512) +
                          VALUE('/odi/drivers/jt400Native.jar:+
                          /odi/drivers/snpsdb2.jar:+
                          /odi/drivers/ojdbc14.jar')
/* Build the Java CLASSPATH */
             DCL        VAR(&PATH) TYPE(*CHAR) LEN(1024)
             CHGVAR     &PATH (&PROJ *tcat ':' *tcat &JDBC)
/* Start the Agent */
             SBMJOB     CMD(JAVA CLASS(oracle.odi.Agent)
CLASSPATH(&PATH) +
                  PARM('&NAME &PORT &VERB') +
                          OPTIMIZE(40) +
                          OUTPUT(*PRINT))
             ENDPGM
```

シナリオの実行

```
             PGM        PARM(&SCEN &VERS &CTX &VERB)
/* Command STARTSCEN */
/* Parameters: */
/*   &SCEN: scenario name */
/*   &VERS: scenario version */
/*   &CTX: context */
```

```
/*   &VERB: verbose mode -V=[1..5] */
/* Example of call: */
/*   CALL PGM(<myLib/myPGM>) PARM('myScen' 'myVers' 'GLOBAL' '-V=5') */
             DCL        VAR(&SCEN) TYPE(*CHAR) LEN(30)
             DCL        VAR(&VERS) TYPE(*CHAR) LEN(30)
             DCL        VAR(&CTX) TYPE(*CHAR) LEN(30)
             DCL        VAR(&VERB) TYPE(*CHAR) LEN(30)
/* All classes below should be compiled */
/* with the CRTJVAPGM command */
/* with optimize 40. */
             DCL        VAR(&PROJ) TYPE(*CHAR) LEN(512) +
                          VALUE('/odi/lib/odi.zip:+
                          /odi/lib/sunjce_provider.jar:+
                          /odi/lib/commons-net.jar:+
                          /odi/lib/local_policy.jar:+
                          /odi/lib/jakarta-ant-optional.jar:+
                          /odi/lib/US_export_policy.jar:+
                          /odi/lib/jce1_2_2.jar')
/* Replace the drivers below with your own drivers. */
             DCL        VAR(&JDBC) TYPE(*CHAR) LEN(512) +
                          VALUE('/odi/drivers/jt400Native.jar:+
                          /odi/drivers/snpsdb2.jar:+
                          /odi/drivers/ojdbc14.jar')
/* Adapt all parameters below to your environment before use. */
             DCL        VAR(&DRV) TYPE(*CHAR) LEN(128) +
                          VALUE('-
SECU_DRIVER=com.ibm.as400.access.AS400JDBCDriver')
             DCL        VAR(&URL) TYPE(*CHAR) LEN(128) +
                          VALUE('-
SECU_URL=jdbc:as400://195.10.10.13;libraries=ODI')
             DCL        VAR(&USER) TYPE(*CHAR) LEN(30) +
                          VALUE('-SECU_USER=QSECOFR')
             DCL        VAR(&PASS) TYPE(*CHAR) LEN(128) +
                          VALUE('-SECU_PASS=XYZ')
             DCL        VAR(&WREP) TYPE(*CHAR) LEN(30) +
                          VALUE('-WORK_REPOSITORY=WORKREP1')
             DCL        VAR(&SUSER) TYPE(*CHAR) LEN(30) +
                          VALUE('-ODI_USER=SUPERVISOR')
             DCL        VAR(&SPASS) TYPE(*CHAR) LEN(128) +
                          VALUE('-ODI_PASS=XYZ')
             DCL        VAR(&PATH) TYPE(*CHAR) LEN(1024)
```

```
/* Build the Java CLASSPATH */
             DCL         VAR(&PATH) TYPE(*CHAR) LEN(1024)
             CHGVAR      &PATH (&PROJ *tcat ':' *tcat &JDBC)
/* Execute the Scenario */
             SBMJOB      CMD(JAVA CLASS(oracle.odi.Agent)
CLASSPATH(&PATH) +
                           PARM(&DRV &URL &USER &PASS &WREP &SUSER &SPASS
+
                           SCEN &SCEN &VERS &CTX &VERB))
             ENDPGM
```

注意: 指定するパスワードは、コマンド agent ENCODE <password>を使用して暗号化する必要があります。

Excel

Microsoft Excel データ・サーバーの作成

Microsoft Excel **データ・サーバー**は、ローカル・ネットワークを通じてアクセス可能な単一の Microsoft Excel スプレッドシートに対応します。

前提条件

ODBC の構成

Microsoft Excel スプレッドシートには、ODBC 接続を通じてのみアクセスできます。Microsoft ODBC データ ソース アドミニストレータを使用して適切な ODBC データソースを定義する必要があります。

新規 ODBC データソースの宣言

新規 ODBC データソースを宣言するには、**構成パネル**から **Microsoft ODBC データ ソース アドミニストレータ**を実行し、次の手順に従います。

1. **「追加」**ボタンをクリックして新規データソースを追加します。
2. Microsoft Excel 用の適切なドライバとして「Microsoft Excel Driver (*.xls)」を選択し、**「完了」**をクリックします。
3. このデータソースに名前（トポロジで必要とされる別名）を付け、**「ブックの選択」**ボタンを使用して適切なスプレッドシートを選択します。
4. Excel スプレッドシートに書込みを行う場合は、**「オプション」**で**「読み取り専用」**チェック・ボックスを選択解除します。
5. **「OK」**をクリックします。

Excel スプレッドシートの構成

単一の Excel スプレッドシートには複数のページを含めることができ、各ページには複数の表を格納できます。スプレッドシートの構成方法にかかわらず、Oracle Data Integrator により各表の

場所を特定できる必要があります。Oracle Data Integrator は、名前で表を識別します。表に名前を付けるには、次の手順に従います。

1. Microsoft Excel でスプレッドシートを開きます。
2. 表のすべてのセルを選択します。
3. メニューで**「挿入」→「名前」→「定義」**を選択します。
4. 表の名前を入力し、**「OK」**をクリックします。この名前は、Oracle Data Integrator で Excel スプレッドシートをリバース・エンジニアリングするときに、データストア名として表示されます。

注意: 表の最初の行は、Oracle Data Integrator がリバース・エンジニアリング・プロセス時に列名をフェッチするヘッダー行に対応します。

注意: 空の表を定義する場合、名前を定義するプロセスで最初の行のみを選択します。Oracle Data Integrator により、データの挿入時にこの行の下に自動的に新規行が追加されます。

データ・サーバーの作成

Microsoft Excel データ・サーバーを作成する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで**「トポロジ」→「物理アーキテクチャ」→「テクノロジ」→「Microsoft Excel」**を選択します。
3. 右クリックし、**「データ・サーバーの挿入」**を選択します。
4. **「定義」**タブで、次のフィールドを入力します。
   - **名前:** Oracle Data Integrator に表示されるデータ・サーバーの名前。
   - **ユーザー/パスワード:** このフィールドは使用しません。
   - **バッチ更新:** 1 を指定します。
   - **配列フェッチ:** 1 を指定します。
5. **「JDBC」**タブで、次のフィールドを入力します。
   - **JDBC ドライバ:** `sun.jdbc.odbc.JdbcOdbcDriver`
   - **JDBC URL:** `jdbc:odbc:<AliasName>`

     ここで、**<AliasName>**は ODBC データソースの名前です。

警告: ODBC を通じて Microsoft Excel スプレッドシートにアクセスする場合、そのスプレッドシートが Microsoft Excel セッションで現在開かれていないことを最初に確認する必要があります。開かれていると、予期しない結果が発生する可能性があります。

6. **「テスト」**をクリックします。
7. **テスト接続**ウィンドウの**「テスト」**をクリックします。
8. **接続に成功**したことを示すウィンドウが表示されます。**「OK」**をクリックします。接続に失敗した場合は、「Microsoft Excel での一般的エラー」を参照してください。
9. **「OK」**をクリックしてデータ・サーバーの作成を承認します。

このデータ・サーバーの最初の**物理スキーマ**に対応する作成ウィンドウが表示されます。
「Microsoft Excel の物理スキーマの作成」を参照してください。

Microsoft Excel の物理スキーマの作成

物理スキーマは、Excel において特別な意味はありません。Oracle Data Integrator では、Microsoft Excel **データ・サーバー**ごとにただ 1 つの物理スキーマが必要です。

Microsoft Excel の物理スキーマを作成する手順:

注意: データ・サーバーの作成直後の場合、手順 1 は無視してください。**物理スキーマ**・ウィンドウがすでに表示されているはずです。

1. Microsoft Excel **データ・サーバー**を選択して右クリックし、**「物理スキーマの挿入」**を選択します。**物理スキーマ**・ウィンドウが表示されます。
2. このスキーマをこのデータ・サーバーのデフォルトに設定する場合、**「デフォルト」**ボックスを選択します（最初の物理スキーマは、常にデフォルトになります）。詳細は、「物理スキーマ」を参照してください。Microsoft Excel の場合、他の物理スキーマは必要ありません。
3. **「コンテキスト」**タブに移動します。
4. この新規**物理スキーマ**に対応する**「コンテキスト」**と既存の**「論理スキーマ」**を選択し、手順 7 に進みます。
   Microsoft Excel の論理スキーマが存在しない場合は、手順 5 に進みます。
5. ボタンをクリックします。
6. 左側の列で既存の**「コンテキスト」**を選択し、右側の列に**「論理スキーマ」**の名前を入力します。この Microsoft Excel の論理スキーマは、自動的に作成され、このコンテキスト内の物理スキーマに関連付けられます。

警告: 特定の**コンテキスト**では、**論理スキーマ**は 1 つの**物理スキーマ**にのみ関連付けることができます。

7. **「OK」**をクリックします。

Microsoft Excel モデルの作成およびリバース・エンジニアリング

Microsoft Excel モデルの作成

Microsoft Excel **モデル**は、Microsoft Excel スプレッドシートに格納された表に対応するデータストアのセットです。モデルは、常に**論理スキーマ**に基づきます。特定の**コンテキスト**内で、**論理スキーマ**は単一の**物理スキーマ**に対応します。この物理スキーマに対応するデータ・スキーマには、Microsoft Excel モデルの表が格納されます。

Microsoft Excel モデルを作成する手順:

1. **デザイナ**に接続します。
2. ツリーで**「モデル」**を選択します。
3. 右クリックし、**「モデルの挿入」**を選択します。
4. **「定義」**タブで、**「名前」**フィールドを入力します。
5. **「テクノロジ」**フィールドで、**「Microsoft Excel」**を選択します。

6. **「論理スキーマ」**フィールドで、モデルの基礎となる論理スキーマを選択します。
7. **「リバース」**タブに移動し、モデルのリバース・エンジニアリング時に使用する**「コンテキスト」**を選択します。**「適用」**をクリックします。

これでモデルは作成されましたが、データストアはまだ格納されていません。

Microsoft Excel モデルのリバース・エンジニアリング

モデルは、データストアなしで作成されます。**リバース・エンジニアリング**操作では、モデルの表の構造をリカバリして、適切なデータストア定義を作成します。リバース・エンジニアリングのタイプには、ドライバの機能のみを使用する**標準**リバース・エンジニアリングと、RKM（リバース・ナレッジ・モジュール）を使用してオブジェクトの構造を取得する**カスタマイズ**・リバース・エンジニアリングという 2 つのタイプがあります。

注意: デフォルトでは、Microsoft Excel 用の RKM はありません。標準リバース・エンジニアリングを使用することをお薦めします。

標準リバース・エンジニアリング

Microsoft Excel で標準リバース・エンジニアリングを実行する手順:

1. Microsoft Excel **モデル**の**「リバース」**タブに移動します。
2. 次のフィールドを入力します。
   - **標準**
   - **コンテキスト:** リバース・エンジニアリングに使用するコンテキスト
   - **リバースエンジニアリングするオブジェクトの型:** リバース・エンジニアリング・プロセスで処理する必要のあるオブジェクト・タイプのリスト
3. **「選択的リバース」**タブに移動します。
   - **「選択的リバース」**、**「新規データストア」**および**「リバースするオブジェクト」**の各ボックスを選択します。
4. リバース・エンジニアリング対象のデータストアのリストが表示されます。リバース・エンジニアリングしないデータストアについては、選択を解除します。
5. **「リバース」**をクリックし、次に**「はい」**をクリックして変更を承認します。
6. Oracle Data Integrator により、選択したデータストアのリバース・エンジニアリングが開始され、プログレス・バーが表示されます。

リバース・エンジニアリングされたデータストアが、モデルの下に表示されます。

「リバース」および**「選択的リバース」**タブのオプションを使用すると、リバース・エンジニアリングを詳細に設定できます。詳細は、「モデル」を参照してください。

Microsoft Excel 用の適切な KM の選択

インタフェースにおける KM の選択により、そのインタフェースまたはチェックの機能およびパフォーマンスが決定されます。次に、Microsoft Excel サーバーを含む特定の環境に適した KM を選択する際に役立つ推奨事項を示します。

KM の一般的な情報は、「ナレッジ・モジュール」を参照してください。

注意: プロジェクトのインタフェースで使用できるのは、そのプロジェクトにインポートされたナレッジ・モジュールのみです。KM のインポート方法の詳細は、「KM のインポート」を参照してください。

注意: 単純な Excel スプレッドシートの場合、CSV（カンマ区切り）ファイル形式を使用したファイル・アクセスが可能です。この方法では、ODBC アクセスを使用せずに大量のファイルを高速に処理できます。

Microsoft Excel からの、または Microsoft Excel へのデータのロード

Microsoft Excel は、インタフェースのソースまたはターゲットとして使用できます（ステージング領域には使用できません）。Microsoft Excel と別のタイプのデータ・サーバー間でデータをロードするために使用する LKM をインタフェースの（「フロー」タブで）選択することは、インタフェースの実行にとって重要です。

Microsoft Excel からデータをロードする際の LKM の選択

次の場合に推奨される LKM です。

- Microsoft Excel ソースからステージング領域にロードする場合

複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

ターゲットまたはステージング領域のテクノロジ	推奨 KM	説明
Sybase	LKM ISO SQL to Sybase (bcp)	汎用 LKM より高速（バルク・ロードを使用）
Microsoft SQL Server	LKM ISO SQL to MSSQL (bulk)	汎用 LKM より高速（バルク・ロードを使用）
Oracle	LKM ISO SQL to Oracle	汎用 LKM より高速（統計を使用）
すべて	**LKM ISO SQL to SQL**	**汎用 KM**

Microsoft Excel へのデータの統合

Microsoft Excel 用の IKM の選択

次の場合に推奨される IKM です。

- ステージング領域から Microsoft Excel ターゲットに統合する場合

Microsoft Excel はステージング領域として使用できません。**「ターゲットと異なるステージング領域」**オプションを選択する必要があります。

モード	ターゲット	推奨 KM	説明

追加	IKM ISO SQL to SQL Append	フロー制御なし。静的制御は Excel で使用不可。

Microsoft Excel での一般的エラー

エラー・メッセージの解釈

Oracle Data Integrator でのエラーは、通常、次のように表示されます。

```
java.sql.SQLException: java.sql.SQLException: [Microsoft][ODBC Driver
Manager] Data source name not found and no default driver specified
RC=0xb
at ...
...
```

java.sql.SQLException というコードは、単に JDBC-ODBC ブリッジを通じて問合せが発行され、エラーが戻されたことを示します。このエラーは、データベースまたはドライバのエラーであることが多く、その方向で解釈する必要があります。

太字で示されたテキストの部分のみを最初に検討する必要があります。このテキストを ODBC ドライバまたは Excel のドキュメントで検索してください。ここにあるような（赤字で示した）固有のエラー・コードが含まれる場合は、エラーをすぐに識別できます。

このようなエラーが実行ログで検出された場合、SQL コードを分析してエラーの原因を特定する必要があります。このコードは、エラーの発生した**タスク**の**「説明」**タブに表示されます。

次に、Excel でよくあるエラーを、その主な原因とともに示します。

一般的なエラー

接続エラー

UnknownDriverException

JDBC ドライバが不適切です。ドライバの名前をチェックしてください。

[Microsoft][ODBC Driver Manager] Data source name not found and no default driver specified RC=0xb
Datasource not found or driver name not specified

JDBC URL に指定された ODBC データソースが不適切です。

インタフェースのエラー

The Microsoft Jet Database engine could not find the object <object name>

アクセスしようとしている表は存在しないか、Excel スプレッドシートに定義されていません。

Too few parameters.Expected 1.

Excel スプレッドシートの存在しない列にアクセスしようとしています。

Operation must use an updateable query.

このエラーは、Excel DSN の定義時に「読み取り専用」オプションを選択解除していない場合に発生する可能性があります。このオプションを選択解除し、インタフェースを再実行してください。

ファイル

ファイル・データ・サーバーの作成

ファイル・**データ・サーバー**は、ファイル・フォルダのセットのコンテナです（各ファイル・フォルダは物理スキーマに対応します）。

トポロジにデフォルトで含まれる FILE_GENERIC データ・サーバーは、ほとんどの要件に適しています。通常、ファイル・データ・サーバーを作成する必要はありません。ファイルの物理スキーマを作成するだけで済みます。

前提条件

JDBC ドライバ

パフォーマンスの点からすると、フラット・ファイルの処理時には、データベース・ユーティリティを使用する方が常に有利です。Oracle Data Integrator には、これらのユーティリティを使用したナレッジ・モジュールが含まれます。ただし、利便性を考慮して、Oracle Data Integrator にはタイプ 4 のフラット・ファイル用 JDBC ドライバが付属しています。このドライバでは、ASCII と EBCDIC（レガシー）両方のファイル形式がサポートされます。

ODBC を通じてフラット・ファイルに接続することも可能です。この方法では、パフォーマンスを期待できず、リバース・エンジニアリングなどのいくつかの機能もサポートされません。そのため、この方法はお薦めしません。

フラット・ファイル用 JDBC ドライバ

このドライバは、製品に組み込まれており、Oracle Data Integrator に無償で付属します。このドライバは、Oracle Data Integrator のインストール時にインストールされ、追加構成を必要としません。

データベース・ユーティリティ

ほとんどのデータベースには、フラット・ファイルと対話するための独自のユーティリティがあります。どのユーティリティの場合も、エージェントまたは Oracle Data Integrator のインストール・ディレクトリからデータベース・クライアント・ソフトウェアにアクセス可能である必要があります。次に、いくつかの例を示します。

- Oracle: SQL*Loader
- Sybase: bcp
- Microsoft SQL Server: bcp
- Teradata: fastload および multiload

  これらのユーティリティは、すべて Oracle Data Integrator から直接使用できます。各ユーティリティの適切なインストール方法は、使用しているデータベースのドキュメントを参照してください。

データ・サーバーの作成

ファイル・データ・サーバーを作成する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで**「トポロジ」→「物理アーキテクチャ」→「テクノロジ」→「ファイル」**を選択します。
3. 右クリックし、**「データ・サーバーの挿入」**を選択します。
4. **「定義」**タブで、次のフィールドを入力します。
   - **名前:** Oracle Data Integrator に表示されるデータ・サーバーの名前。
   - **ユーザー/パスワード:** 使用しません。
5. **「JDBC」**タブで、使用するドライバに応じて次のフィールドを入力します。
   - **JDBC ドライバ:** `com.sunopsis.jdbc.driver.file.FileDriver`
   - **JDBC URL:** `jdbc:snps:dbfile`
6. **「テスト」**をクリックします。
7. **テスト接続**ウィンドウの**「テスト」**をクリックします。
8. **接続に成功**したことを示すウィンドウが表示されます。**「OK」**をクリックします。
9. **「OK」**をクリックしてデータ・サーバーの作成を承認します。

このデータ・サーバーの最初の**物理スキーマ**に対応する作成ウィンドウが表示されます。
「ファイルの物理スキーマの作成」を参照してください。

ファイルの物理スキーマの作成

物理スキーマは、次のディレクトリのペアに対応します。

- （データ）**スキーマ:** Oracle Data Integrator は、このスキーマでインタフェースのソースおよびターゲット・ファイルを検索します。
- **作業スキーマ:** Oracle Data Integrator は、このスキーマで、データ・ライブラリに格納されたソースとターゲットに関連する一時ファイルを最終的に作成して操作します。

注意: データ・スキーマと作業スキーマは、それぞれディレクトリを表します。このディレクトリには、変換を実行するために使用されるエージェントからアクセスできる必要があります。このディレクトリの場所は、絶対パス（m:/public/data/files）で指定するか、エージェントの起動ディレクトリを基準とする相対パス（../demo/files）で指定します。パスには、（実行場所に依存しない）UNC を使用することを強くお薦めします。
エージェントなしで変換が実行される場合、ディレクトリの場所は、Oracle Data Integrator のインストール先ディレクトリが基準となります。

注意: 特に UNIX では、これらのディレクトリに対する読取り/書込み権限をエージェントに付与する必要があります。これは、読取り専用ファイルにアクセスする場合でも同様です（無効なファイル・レコードに対してエラー・ファイルが作成されるため）。

注意: ファイル・パスは、Windows と UNIX の場合で異なります。この情報の設定時には、エージェントにより使用されるプラットフォームを考慮してください。

物理スキーマの作成

ファイルの物理スキーマを作成する手順:

注意: データ・サーバーの作成直後の場合、手順 1 は無視してください。**物理スキーマ**・ウィンドウがすでに表示されているはずです。

1. **トポロジ**で、物理スキーマを作成するファイル・**データ・サーバー**を選択して右クリックし、**「物理スキーマの挿入」**を選択します。**物理スキーマ**・ウィンドウが表示されます。
2. ソースまたはターゲット・ファイルを含むディレクトリへのパスを**「ディレクトリ（スキーマ）」**に入力します。
3. この物理スキーマの作業ディレクトリを**「ディレクトリ（作業スキーマ）」**に入力します。
4. このスキーマをこのデータ・サーバーのデフォルト・スキーマに設定する場合、**「デフォルト」**ボックスを選択します（最初の物理スキーマは、常にデフォルト・スキーマになります）。詳細は、「物理スキーマ」を参照してください。
5. **「コンテキスト」**タブに移動します。
6. この新規**物理スキーマ**に対応する**「コンテキスト」**と既存の**「論理スキーマ」**を選択し、手順 9 に進みます。
   ファイルの論理スキーマが存在しない場合は、手順 7 に進みます。
7. ボタンをクリックします。
8. 左側の列で既存の**「コンテキスト」**を選択し、右側の列に**「論理スキーマ」**の名前を入力します。このファイルの論理スキーマは、自動的に作成され、このコンテキスト内の物理スキーマに関連付けられます。

警告: 特定の**コンテキスト**では、**論理スキーマ**は 1 つの**物理スキーマ**にのみ関連付けることができます。

9. **「OK」**をクリックします。

ファイル・モデルの作成およびリバース・エンジニアリング

ファイル・モデルの作成

ファイル・**モデル**は、ディレクトリに格納されたファイルに対応する**データストア**のセットです。モデルは、常に**論理スキーマ**に基づきます。特定の**コンテキスト**内で、**論理スキーマ**は単一の**物理スキーマ**に対応します。この物理スキーマのデータ・スキーマは、モデルに記述されたすべてのファイルを格納するディレクトリ（最終的にはサブディレクトリ）です。

ファイル・モデルを作成する手順:

1. **デザイナ**に接続します。
2. ツリーで**「モデル」**を選択します。
3. 右クリックし、**「モデルの挿入」**を選択します。
4. **「定義」**タブで、**「名前」**フィールドを入力します。
5. **「テクノロジ」**フィールドで、**「ファイル」**を選択します。
6. **「論理スキーマ」**フィールドで、モデルの基礎となる論理スキーマを選択します。

7. **「リバース」**タブに移動し、モデルのリバース・エンジニアリングに使用する**「コンテキスト」**を選択します。**「適用」**をクリックします。

これでモデルは作成されましたが、データストアはまだ格納されていません。

ファイル・モデルのリバース・エンジニアリング

モデルは、データストアなしで作成されます。**リバース・エンジニアリング**操作では、モデルのファイルの構造を収集して、モデルのデータストア定義を作成します。

リバース・エンジニアリングには、次の 3 つのタイプがあります。

- **標準**リバース・エンジニアリング: デリミタ付きファイルでのみ使用可能です。この操作はファイルごとに実行されます。
- **カスタマイズ**・リバース・エンジニアリング: RKM（リバース・ナレッジ・モジュール）を使用して、Microsoft Excel スプレッドシートからモデルのすべてのファイル構造を取得します。
- **COBOL コピーブック**・リバース・エンジニアリング: ファイルを記述するコピーブックが存在する場合、固定ファイルで使用可能です。詳細は、「COBOL コピーブックのリバース・エンジニアリング」を参照してください。
- **列設定ウィザード**: 固定ファイルで使用可能です。詳細は、「ウィザードを使用した固定ファイルのリバース・エンジニアリング」を参照してください。

注意: ファイルのカスタマイズ・リバース・エンジニアリング用に、「`RKM File from Excel`」という特定の RKM が提供されています。この RKM を使用するには、少なくとも 1 つのプロジェクトにこの RKM をインポートする必要があります。

標準リバース・エンジニアリング

ファイルの標準リバース・エンジニアリングを実行する手順（デリミタ付きファイルのみ）:

1. ファイル・**モデル**を右クリックし、**「データストアの挿入」**を選択します。
2. **「定義」**タブで、次のフィールドを入力します。
    - **名前:** データストアの名前。
    - **リソース名:** サブディレクトリ（必要な場合）とファイル名。**「...」**ボタンを使用してファイル名を取得できます。
3. **「ファイル」**タブに移動して、ファイルのタイプを指定します。
    - 形式は**「区切り」**とします。
    - **「ヘッダー」**の行数を指定します。（ヘッダーが存在する場合、Oracle Data Integrator では、ヘッダーの最初の行を使用してファイルの列名を付けます。）
    - **「レコード・セパレータ」**を選択します。
    - **「フィールド・セパレータ」**として使用する文字を選択または入力します。
    - ファイルで使用している場合、**「テキスト・デリミタ」**を入力します。
    - ファイルに小数点が含まれる場合、**「小数点セパレータ」**を入力します。
4. **「適用」**をクリックしてファイル定義を保存します。
5. **「列」**タブに移動して、ファイル構造をリバース・エンジニアリングします。

- **「リバース」**ボタンをクリックします。
- リバース・エンジニアリングされた列の形式と長さをチェックします。Oracle Data Integrator では、タイプと長さが予測されますが、デフォルト値が使用されることもあります（文字列の場合、通常は 50）。
- **適用**または**「OK」**をクリックして列の設定を保存します。

カスタマイズ・リバース・エンジニアリング

このリバース・エンジニアリングの手順では、Microsoft Excel スプレッドシートにファイル・グループの記述が含まれます。例として、Oracle Data Integrator の`/demo/excel` サブディレクトリに含まれる `file_repository.xls` ファイルを編集します。次の手順では、実際のフラット・ファイル構造の記述に基づいてこのファイルを変更済であることを前提とします。

カスタマイズ・リバース・エンジニアリングを実行するには、次の手順に従います。

1. ファイル記述を含む Excel スプレッドシートに対応する **ODBC Microsoft Excel データソースを追加します**。
2. このスプレッドシートに対して**データ・サーバー、物理スキーマおよび論理スキーマを定義します**。
3. RKM の「RKM `File from Excel`」を使用して**カスタマイズ・リバース・エンジニアリングを実行します**。

ODBC Microsoft Excel データソース・ドライバ（*.xls）を追加する手順:

1. **Microsoft ODBC アドミニストレータ**を起動します。
2. **「システム データ ソース」**を追加します。
3. ドライバとして「`Microsoft Excel Driver (*.xls)`」を選択します。
4. データソースに **`SUNOPSIS_XL_FILE_REPO`** という名前を付け、`/demo/excel/file_repository.xls` ファイルを選択します。

Microsoft Excel スプレッドシートに対してデータ・サーバー、物理スキーマおよび論理スキーマを定義する手順:

1. **トポロジ・マネージャ**・モジュールを起動します。
2. 次のパラメータを使用して Microsoft Excel **データ・サーバー**を追加します。

   名前: `EXCEL_FILE_REPOSITORY`

   JDBC ドライバ: `sun.jdbc.odbc.JdbcOdbcDriver`

   JDBC URL: `jdbc:odbc:`**`SUNOPSIS_XL_FILE_REPO`**
3. **「適用」**をクリックして変更を適用します。
4. デフォルトの物理スキーマに**論理スキーマ**を追加します。
5. 物理スキーマの**「コンテキスト」**タブで、  をクリックします。
6. 新規行でリバース・エンジニアリングのコンテキストを選択し、2 番目の列に `EXCEL_FILE_REPOSITORY` と入力します。この名前は必須です。
7. **「適用」**をクリックして変更を適用します。

カスタマイズ・リバース・エンジニアリングを実行する手順:

1. **デザイナ**・モジュールを起動します。

2. 「RKM File from Excel」ナレッジ・モジュールを少なくとも 1 つのプロジェクトにインポートします。
3. ファイル・**モデル**をクリックし、右クリックして**「編集」**を選択します。
4. **「リバース」**タブで、次のパラメータを設定します。
   - **「カスタマイズ済」**を選択
   - **コンテキスト:** リバース・コンテキスト
   - **KM:** RKM File from Excel
5. **「リバース」**をクリックします。
6. **実行ログ**でリバース・エンジニアリング・プロセスを追跡できます。

重要: Microsoft Excel の論理スキーマを定義する必要があります。この論理スキーマは、EXCEL_FILE_REPOSITORY という名前で、file_repository.xls ファイル（または同様の構造を持つ別のファイル）を参照している必要があります。

重要: Microsoft Excel ファイルの file_repository.xls は、リバース・エンジニアリングを実行する前に閉じる必要があります。

ファイル用の適切な KM の選択

インタフェースまたはチェックに使用する KM の選択により、そのインタフェースまたはチェックの機能およびパフォーマンスが決定されます。次に、ファイル・サーバーに関連する様々な環境で KM を選択する際に役立つ推奨事項を示します。

KM の一般的な情報は、「ナレッジ・モジュール」を参照してください。

注意: プロジェクトのインタフェースで使用できるのは、そのプロジェクトにインポートされたナレッジ・モジュールのみです。KM のインポート方法の詳細は、「KM のインポート」を参照してください。

ファイルからの、またはファイルへのデータのロード

ファイルは、インタフェースのソースまたはターゲットとして使用できますが、ステージング領域としては**使用できません**。インタフェースの「フロー」タブで LKM を選択することは、インタフェースのパフォーマンスを決定するうえで重要です。

ファイルからロードする際の LKM の選択

次の場合に推奨される LKM です。

- ファイル・ソースからステージング領域にロードする場合

複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

ターゲットまたはステージング領域のテクノロジ	推奨 KM	説明
Oracle	LKM File to Oracle	汎用 LKM より高速（SQL*Loader を使用）

	(sql*loader)	
すべて	**LKM ISO File to SQL**	**汎用 KM**

ファイルでのデータの統合

ファイル用の IKM の選択

次の場合に推奨される IKM です。

- ステージング領域からファイル・ターゲットに統合する場合

選択する統合モードに応じてファイルに統合を実行する際に推奨される KM です。複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

モード	ターゲット	推奨 KM	説明
追加	ファイル	IKM ISO SQL to File Append	フロー制御なし

注意: すでに存在するターゲット・ファイルを削除せずにターゲット・ファイルを作成する場合（TRUNCATE オプション）、この KM に次の手順を挿入する必要があります。
- **テクノロジ:** ファイル
- **コマンド:** `CREATE TABLE`
- **「エラーの無視」**を選択するか、インタフェースの「フロー」タブでファイル作成のトリガーに指定できるオプションを作成

COBOL コピーブックのリバース・エンジニアリング

COBOL コピーブックのリバース・エンジニアリングにより、Data Integrator で COBOL コピーブック・ファイルに格納されている記述からレガシー・ファイル構造を取得できます。

COBOL コピーブックをリバース・エンジニアリングする手順:

1. **固定**形式のファイル・データストアを作成するか開きます。
2. **「列」**タブに移動します。
3. **「COBOL コピーブックのリバース」**ボタンをクリックします。
4. 次のフィールドを入力します。
   - **ファイル:** コピーブック・ファイルの場所
   - **キャラクタ・セット:** コピーブック・ファイルのキャラクタ・セット
   - **説明書式（EBCDIC | ASCII）:** ファイル記述（コピーブック）形式
   - **データ形式（EBCDIC | ASCII）:** ファイル・データ形式
5. **「OK」**をクリックします。

コピーブックに記述された列がリバース・エンジニアリングされ、列リストに表示されます。

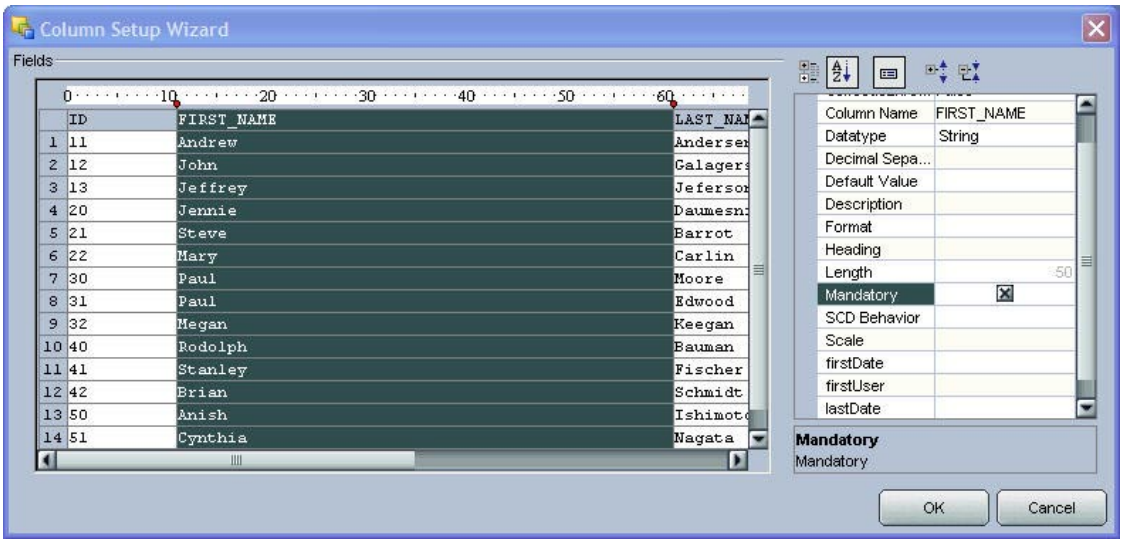
注意: Oracle Data Integrator でデータ型が宣言されていない列は、データ型なしで表示されます。

ウィザードを使用した固定ファイルのリバース・エンジニアリング

Oracle Data Integrator には、固定ファイルの列をグラフィカルに定義するためのウィザードが用意されています。

ウィザードを使用して固定ファイルをリバース・エンジニアリングする手順:

1. **固定**形式のファイル・データストアを作成するか開きます。
2. **「列」**タブに移動します。
3. **「リバース」**ボタンをクリックします。次のようなウィンドウが表示されます。このウィンドウには、ファイルの最初のレコードが表示されます。



4. ファイル内容の上にあるルーラーをクリックし、列を区切るマーカーを作成します。ルーラーを右クリックするとマーカーを削除できます。
5. 列は、事前に生成された名前（C1、C2 など）で作成されます。列名を編集するには、ルーラーの下の列ヘッダー行をクリックします。
6. 右側のプロパティ・パネルで、選択した列のすべてのパラメータを編集できます。少なくとも各列の**「列名」**、**「データ型」**および**「長さ」**は設定する必要があります。
7. 列の定義が完了したら、**「OK」**をクリックします。

XML

XML データ・サーバーの作成

XML **データ・サーバー**は、ローカル・ネットワークを通じてアクセス可能な単一の XML ファイルに対応します。

XML ファイルには、**Oracle Data Integrator Driver for XML** を通じてアクセスできます。この JDBC ドライバは、メモリー内に格納された**スキーマ**のリレーショナル構造に XML ファイルの階層構造をロードし、JDBC を通じた SQL 問合せを使用可能にします。また、このドライバは、リレーショナル構造を XML ファイルにアンロードすることも可能です。

前提条件

JDBC の構成

XML ファイルには、**Oracle Data Integrator Driver for XML** を通じてアクセスできます。ドライバは、Oracle Data Integrator とともにインストールされます。

接続情報

システム管理者から次の情報を入手する必要があります。

- XML ファイルに関連付けられた **DTD ファイル**の場所
- **XML ファイル**の場所
- XML ファイルの**ルート**要素の名前

データ・サーバーの作成

XML データ・サーバーを作成する手順:

1. **トポロジ・マネージャ**に接続します。
2. ツリーで「**トポロジ**」→「**物理アーキテクチャ**」→「**テクノロジ**」→「**XML**」を選択します。
3. 右クリックし、「**データ・サーバーの挿入**」を選択します。
4. 「**定義**」タブで、次のフィールドを入力します。
   - **名前:** Oracle Data Integrator に表示されるデータ・サーバーの名前。
   - **ユーザー/パスワード:** このフィールドは使用しません。
5. 「**JDBC**」タブで、次のフィールドを入力します。
   - **JDBC ドライバ:** `com.sunopsis.jdbc.driver.xml.SnpsXmlDriver`
   - **JDBC URL:** `jdbc:snps:xml?[property=value&property=value...]`

JDBC ドライバのプロパティ

パラメータ	値	説明
f	`<XML File location>`	UNC 形式での XML ファイルの場所（相対パスまたは絶対パス）。ファイルのパス名には、バックスラッシュ（\）ではなくスラッシュ（/）を使用してください。
d	`<DTD File location>`	UNC 形式での DTD ファイルの場所（相対パスまたは絶対パス）。ファイルのパス名には、バックスラッシュ（\）ではなくスラッシュ（/）を使用してください。このパラメータが存在しない場合、ドライバは、.xml 拡張子を.dtd で置換して XML ファイルから DTD ファイルの名前を構成します。
re	`<Root element>`	スキーマのルート表として使用する要素の名前です。この値は大/小文字が区別されます。このパラメータは、WSDL ファイルから特定のメッセージ定義をリバース・エンジニアリングする場合、または XSD ファイルに複数のルート要素候補が存在するときに使用できます。

ro	`true | false`	true の場合、XML ファイルは読取り専用モードで開かれます。
s	`<schema name>`	XML ファイルがロードされるリレーショナル・スキーマの名前。このパラメータが存在しない場合、スキーマ名はファイル名から自動的に生成されます。
cs	`true | false`	大/小文字を区別するモードまたは区別しないモードで XML ファイルをロードします。大/小文字を区別しないモードの場合、DTD ファイルのすべての要素名は、それぞれ明確に区別できる必要があります（たとえば、同じファイルに Abc と abc を含めることはできません）。大/小文字を区別するパラメータは、スキーマの永続パラメータです。この設定は、スキーマ作成後に変更することはできません。大/小文字を区別しないモードで XML ファイルを開いた場合も、XML ファイルの大/小文字は維持されます。

例:

```
jdbc:snps:xml?f=../demo/xml/GEO_DIM.xml&re=GEOGRAPHY_DIM&ro=false&case_sens=true&s=GEO
```

警告: XML ファイルにアクセスするには、最初に必ずファイルのロックを解除する必要があります。ファイルがロックされているかどうかをチェックするには、XML ファイルのディレクトリの.lck ファイルを確認します。

6. **「テスト」**をクリックします。
7. **テスト接続**ウィンドウの**「テスト」**をクリックします。
8. **接続に成功**したことを示すウィンドウが表示されます。**「OK」**をクリックします。接続に失敗した場合は、「XML での一般的エラー」を参照してください。
9. **「OK」**をクリックしてデータ・サーバーの作成を承認します。

このデータ・サーバーの最初の**物理スキーマ**に対応する作成ウィンドウが表示されます。「XML の物理スキーマの作成」を参照してください。

XML の物理スキーマの作成

物理スキーマは、XML ファイルに関連付けられた表の格納場所です。

XML の物理スキーマを作成する手順:

注意: データ・サーバーの作成直後の場合、手順 1 は無視してください。**物理スキーマ・**ウィンドウがすでに表示されているはずです。

1. 適切な XML **データ・サーバー**を選択して右クリックし、**「物理スキーマの挿入」**を選択します。**物理スキーマ・**ウィンドウが表示されます。
2. **「スキーマ」**と**「作業スキーマ」**の名前を入力します。XML **データ・サーバー**の JDBC URL の `s=<schema name>`プロパティで**スキーマ**の名前を指定している場合、ここでも同じスキーマ名を使用する必要があります。

3. このスキーマをこのデータ・サーバーのデフォルトに設定する場合、**「デフォルト」**ボックスを選択します（最初の物理スキーマは、常にデフォルトになります）。詳細は、「物理スキーマ」を参照してください。XML ファイルの場合、他の物理スキーマは必要ありません。
4. **「コンテキスト」**タブに移動します。
5. この新規**物理スキーマ**に対応する**「コンテキスト」**と既存の**「論理スキーマ」**を選択し、手順 8 に進みます。
   XML の論理スキーマが存在しない場合は、手順 6 に進みます。
6.  ボタンをクリックします。
7. 左側の列で既存の**「コンテキスト」**を選択し、右側の列に**「論理スキーマ」**の名前を入力します。この XML の論理スキーマは、自動的に作成され、このコンテキスト内の物理スキーマに関連付けられます。

警告: 特定の**コンテキスト**では、**論理スキーマ**は 1 つの**物理スキーマ**にのみ関連付けることができます。

8. **「OK」**をクリックします。

XML ファイル・モデルの作成およびリバース・エンジニアリング

XML ファイル・モデルの作成

XML ファイル・**モデル**は、各データストアが XML ファイルのエントリ・レベルを表すデータストアのセットとみなすことができます。モデルは、常に**論理スキーマ**に基づきます。特定の**コンテキスト**内で、**論理スキーマ**は単一の**物理スキーマ**に対応します。この物理スキーマに対応するデータ・スキーマには、XML モデルの表が格納されます。

XML ファイル・モデルを作成する手順:

1. **デザイナ**に接続します。
2. ツリーで**「モデル」**を選択します。
3. 右クリックし、メニューから**「モデルの挿入」**を選択します。
4. **「定義」**タブで、**「名前」**フィールドを入力します。
5. **「テクノロジ」**フィールドで、**「XML」**を選択します。
6. **「論理スキーマ」**フィールドで、モデルの基礎となる論理スキーマを選択します。
7. **「リバース」**タブに移動し、モデルのリバース・エンジニアリング時に使用する**「コンテキスト」**を選択します。**「適用」**をクリックします。

これでモデルは作成されましたが、データストアはまだ格納されていません。

XML モデルのリバース・エンジニアリング

モデルは、データストアなしで作成されます。**リバース・エンジニアリング**操作では、モデルの表の構造を収集して、適切なデータストア定義を作成します。リバース・エンジニアリングのタイプには、ドライバの機能のみを使用する**標準**リバース・エンジニアリングと、RKM（リバース・ナレッジ・モジュール）を使用してオブジェクトの構造を取得する**カスタマイズ**・リバース・エンジニアリングという 2 つのタイプがあります。

注意: デフォルトでは、XML ファイル用の RKM はありません。XML ファイルでは標準リバース・エンジニアリングを使用することをお薦めします。

標準リバース・エンジニアリング

XML ファイルで標準リバース・エンジニアリングを実行する手順:

1. XML **モデル**の**「リバース」**タブに移動します。
2. 次の**フィ**ールドを入力します。
   - **標準**
   - **コンテキスト:** リバース・エンジニアリングに使用するコンテキスト
   - **リバースエンジニアリングするオブジェクトの型:** リバース・エンジニアリング・プロセスで処理する必要のあるオブジェクト・タイプのリスト
3. **「選択的リバース」**タブに移動します。
   - **「選択的リバース」**、**「新規データストア」**および**「リバースするオブジェクト」**の各ボックスを選択します。
4. リバース・エンジニアリング対象のデータストアのリストが表示されます。リバース・**エンジニアリング**しないデータストアについては、選択を解除します。
5. **「リバース」**をクリックし、次に**「はい」**をクリックして変更を承認します。
6. Oracle Data **Integrator** により、選択したデータストアのリバース・エンジニアリングが開始され、プログレス・バーが表示されます。

リバース・エンジニアリングされたデータストアが、モデルの下に表示されます。

「リバース」および**「選択的リバース」**タブのオプションを使用すると、リバース・エンジニアリングを詳細に設定できます。詳細は、「モデル」を参照してください。

リバース・エンジニアリングの結果

Oracle Data Integrator により、XML ファイルから収集された表に次の列が自動的に追加されます。

- 親子関係に対応する主キー（PK 列）
- 親子関係に対応する外部キー（FK 列）
- XML ファイルに出現するデータの順序を取得できるようにする順序識別子（ORDER 列）

これらの追加列により、スキーマに格納されたリレーショナル構造への階層 XML 構造のマッピングが可能になります。

XML ファイル用の適切な KM の選択

インタフェースにおける KM の選択により、そのインタフェースの機能およびパフォーマンスが決定されます。次に、特定の環境に適した XML ファイル用の KM を選択する際に役立つ推奨事項を示します。

KM の一般的な情報は、「ナレッジ・モジュール」を参照してください。

注意: プロジェクトのインタフェースで使用できるのは、そのプロジェクトにインポートされたナレッジ・モジュールのみです。KM のインポート方法の詳細は、「KM のインポート」を参照してください。

ファイルとメモリーの同期

XML ファイルのデータとメモリー内のデータを完全に同期させるため、次の特定のコマンドをコールする必要があります。

- データの読取りまたは更新を目的として XML モデルの表を使用する前に、XML の論理スキーマで SYNCHRONIZE FROM FILE コマンドを使用することをお薦めします。この操作により、XML の階層データがリレーショナル・メモリー・スキーマにリロードされます。最初のアクセス時にスキーマがメモリーにロードされる場合、メモリー内のデータが XML ファイルの内容と同期しない可能性があります。
- リレーショナル・メモリー・スキーマに変更を加えた後に、XML の論理スキーマで SYNCHRONIZE ALL または SYNCHRONIZE FROM DATABASE コマンドをコールし、リレーショナル・メモリー・スキーマを XML の階層データにアンロードする必要があります。

インタフェースとプロシージャで XML スキーマを操作する前（および操作した後）に、パッケージのプロシージャでこれらのコマンドを実行する必要があります。

これらのコマンドの詳細は、『Oracle Data Integrator Driver for XML』を参照してください。

警告: JDBC URL に XML ファイルのスキーマが設定されていない場合、データを処理する前にコマンド SET SCHEMA をコールする必要があります。

XML に対するインタフェース

XML モデルの表をインタフェースのターゲットとして使用する場合、XML の階層構造をリレーショナル構造にマップするため、追加の列にもデータを挿入する必要があります。たとえば、次のような XML 構造に region レコードを入力する場合を考えます。

```
<country COUNTRY_ID="6" COUNTRY_NAME="Autralia">
     <region REGION_ID="72" REGION_NAME="Queensland">
</country>
```

この場合、region 表の REGION_ID 列と REGION_NAME 列にデータを入力するだけでなく、次の列にも入力する必要があります。

- REGIONPK: この列により、各<region>タグを識別できます。
- REGIONORDER: この列により、XML ファイルで<region>タグを順序付けることができます（リレーショナル・スキーマでは、レコードの順序は設定されませんが、XML タグの順序は設定されます）。
- COUNTRYFK: この列により、<country>親要素と関連する<region>要素を配置できます。この値は、country 表の Autralia レコードに対応する COUNTRYPK の値と等しくなります。

データの整合性を確保するため、Oracle Data Integrator では、特定の**参照**制約と**主キー**制約が XML モデルに挿入されます。

XML ファイルからの、または XML ファイルへのデータのロード

XML ファイルは、インタフェースのソースまたはターゲットとして使用できます。XML ファイルと別のタイプのデータ・サーバー間でデータをロードするために使用する LKM を（「インタフェース・フロー」タブで）選択することは、インタフェースの実行にとって重要です。

XML スキーマからロードする際の LKM の選択

次の場合に推奨される LKM です。

- XML ソースからステージング領域にロードする場合

複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

ステージング領域	推奨 KM	説明
Microsoft SQL Server	LKM ISO SQL to MSSQL (bulk)	SQL Server のバルク・ローダーを使用
Oracle	LKM ISO SQL to Oracle	汎用 LKM より高速（統計を使用）
Sybase	LKM ISO SQL to Sybase (bcp)	Sybase のバルク・ローダーを使用
すべて	**LKM ISO SQL to SQL**	**汎用 KM**

XML スキーマにロードする際の LKM の選択

XML スキーマはステージング領域として使用しないことをお薦めします。ただし、XML がインタフェースのターゲットであり、ターゲットをステージング領域として使用する場合を除きます。この場合、状況によっては XML スキーマにデータをロードする必要があります。

次の場合に推奨される LKM です。

- ソースから XML ステージング領域にロードする場合

複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

ソース	推奨 KM	説明
JMS	LKM JMS to SQL	
ファイル	LKM ISO File to SQL	
すべて	**LKM ISO SQL to SQL**	**汎用 KM**

XML ファイルへのデータの統合

XML ファイルでのデータ統合計画は、非常に多く存在し、複数のモードが関連します。「インタフェース・フロー」タブでの IKM の選択により、統合におけるパフォーマンスと使用可能な機能が決定されます。

XML ファイル用の IKM の選択

次の場合に推奨される IKM です。

- ステージング領域から XML ターゲットに統合する場合。
- XML ステージング領域から XML ターゲットに統合する場合。この場合、ステージング領域は XML ターゲット上に存在します。

複数のソリューションが可能である場合、優先傾向とパフォーマンスの順に各ソリューションを示します。汎用 KM は太字で示します。

モード	ステージング領域	推奨 KM	説明
更新	**XML**	**IKM ISO SQL Incremental Update**	**汎用 KM**
追加	**XML**	**IKM ISO SQL Control Append**	**汎用 KM**
追加	**すべての RDBMS**	**IKM ISO SQL to SQL Append**	**汎用 KM**

XML での一般的エラー

XML を原因とするエラーの検出

Oracle Data Integrator でのエラーは、通常、次のように表示されます。

```
java.sql.SQLException: No suitable driver
at ...
at ...
...
```

`java.sql.SQLException` というコードは、単に JDBC ドライバを通じて問合せが発行され、エラーが戻されたことを示します。このエラーは、データベースまたはドライバのエラーであることが多く、その方向で解釈する必要があります。

太字で示されたテキストの部分のみを最初に検討する必要があります。このテキストを XML ドライバのドキュメントで検索してください。ここにあるような固有のエラー・コードが含まれる場合は、エラーをすぐに識別できます。

このようなエラーが実行ログで検出された場合、データベースに送信された SQL コードを分析してエラーの原因を特定する必要があります。このコードは、エラーの発生した**タスク**の**「説明」**タブに表示されます。

次に、XML でよくあるエラーを、その主な原因とともに示します。

一般的なエラー

接続エラー

No suitable driver

Oracle Data Integrator Driver for XML が正しくインストールされていません。

インタフェースのエラー

File <XML file> is already locked by another instance of the XML driver.

XML ファイルが別のユーザーまたはアプリケーションによってロックされています。XML ファイルを使用している可能性のあるすべてのアプリケーションを終了してください。このようなアプリケーションがクラッシュしている場合は、XML ファイルのディレクトリに残っている.lck ファイルを削除します。

The DTD file "xxxxxxx.dtd" doesn't exist

この例外は、コマンド LOAD FILE により XML ファイルをロードしようとしたときに発生する可能性があります。このエラー・メッセージの原因として考えられるのは、次の 2 つです。

- DTD ファイルのパスが間違っています。
- 対応する XML ファイルがすでに別のスキーマによって開かれています（接続時など）。

Table not found: S0002 Table not found: <table name> in statement [<SQL statement>]

アクセスしようとしている表はスキーマに存在しません。

Column not found: S0022 Column not found: <column name> in statement [<SQL statement>]

アクセスしようとしている列は、文に指定された表に存在しません。