Informatica®

Informatica® PowerCenter
10.4.0

Designer ガイド

Informatica PowerCenter Designer ガイド
10.4.0
2019 年 12 月

www.openmdx.org/#FAQ、http://www.php.net/license/3_01.txt、http://srp.stanford.edu/license.txt、http://www.schneier.com/blowfish.html、http://www.jmock.org/license.html、http://xsom.java.net、http://benalman.com/about/license/、https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js、http://www.h2database.com/html/license.html#summary、http://jsoncpp.sourceforge.net/LICENSE、http://jdbc.postgresql.org/license.html、http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto、https://github.com/rantav/hector/blob/master/LICENSE、http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html、http://jibx.sourceforge.net/jibx-license.html、https://github.com/lyokato/libgeohash/blob/master/LICENSE、https://github.com/hjiang/jsonxx/blob/master/LICENSE、https://code.google.com/p/lz4/、https://github.com/jedisct1/libsodium/blob/master/LICENSE、http://one-jar.sourceforge.net/index.php?page=documents&file=license、https://github.com/EsotericSoftware/kryo/blob/master/license.txt、http://www.scala-lang.org/license.html、https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt、http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html、https://aws.amazon.com/asl/、https://github.com/twbs/bootstrap/blob/master/LICENSE、および https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt。

本製品には、Academic Free License（http://www.opensource.org/licenses/afl-3.0.php）、Common Development and Distribution License（http://www.opensource.org/licenses/cddl1.php）、Common Public License（http://www.opensource.org/licenses/cpl1.0.php）、Sun Binary Code License Agreement Supplemental License Terms、BSD License（http:// www.opensource.org/licenses/bsd-license.php）、BSD License（http://opensource.org/licenses/BSD-3-Clause）、MIT License（http://www.opensource.org/licenses/mit-license.php）、Artistic License（http://www.opensource.org/licenses/artistic-license-1.0）、Initial Developer’s Public License Version 1.0（http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/）に基づいて許諾されたソフトウェアが含まれています。

本製品には、ソフトウェア copyright (C) 2003-2006 Joe WaInes, 2006-2007 XStream Committers が含まれています。All rights reserved. 本ソフトウェアに関する許諾および制限は、http://j.org/license.html にある使用条件に従います。本製品には、Indiana University Extreme! Lab によって開発されたソフトウェアが含まれています。詳細については、http://www.extreme.indiana.edu/を参照してください。

本製品には、ソフトウェア Copyright (C) 2013 Frank Balluffi and Markus Moeller が含まれています。All rights reserved. 本ソフトウェアに関する許諾および制限は、MIT ライセンスの使用条件に従います。

特許については、https://www.informatica.com/legal/patents.html を参照してください。

NOTICES

この Informatica 製品（以下「ソフトウェア」）には、Progress Software Corporation（以下「DataDirect」）の事業子会社である DataDirect Technologies からの特定のドライバ（以下「DataDirect ドライバ」）が含まれています。DataDirect ドライバには、次の用語および条件が適用されます。

1. DataDirect ドライバは、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。
2. DataDirect または第三者は、予見の有無を問わず発生した ODBC ドライバの使用に関するいかなる直接的、間接的、偶発的、特別、あるいは結果的損害に対して責任を負わないものとします。本制限事項は、すべての訴訟原因に適用されます。訴訟原因には、契約違反、保証違反、過失、厳格責任、詐称、その他の不法行為を含みますが、これらに限るものではありません。

本マニュアルの情報は、予告なしに変更されることがあります。このドキュメントで問題が見つかった場合は、infa_documentation@informatica.com までご報告ください。

Informatica 製品は、それらが提供される契約の条件に従って保証されます。Informatica は、商品性、特定目的への適合性、非侵害性の保証等を含めて、明示的または黙示的ないかなる種類の保証をせず、本マニュアルの情報を「現状のまま」提供するものとします。

発行日: 2020-02-04

目次

序文

*PowerCenter(R) Designer ガイド*は、PowerCenter Designer の使用方法とマッピングの作成方法を学習するために使用します。ソース、ターゲットおよびトランスフォーメーションを定義してマッピングとマップレットを構築することができます。ウィンドウを使用してフォルダ、リポジトリオブジェクト、およびタスクを表示することができます。

Informatica のリソース

Informatica は、Informatica Network やその他のオンラインポータルを通じてさまざまな製品リソースを提供しています。リソースを使用して Informatica 製品とソリューションを最大限に活用し、その他の Informatica ユーザーや各分野の専門家から知見を得ることができます。

Informatica Network

Informatica Network は、Informatica ナレッジベースや Informatica グローバルカスタマサポートなど、多くのリソースへの入口です。Informatica Network を利用するには、https://network.informatica.com にアクセスしてください。

Informatica Network メンバーは、次のオプションを利用できます。

- ナレッジベースで製品リソースを検索できます。
- 製品の提供情報を表示できます。
- サポートケースを作成して確認できます。
- 最寄りの Informatica ユーザーグループネットワークを検索して、他のユーザーと共同作業を行えます。

Informatica ナレッジベース

Informatica ナレッジベースを使用して、ハウツー記事、ベストプラクティス、よくある質問に対する回答など、製品リソースを見つけることができます。

ナレッジベースを検索するには、https://search.informatica.com にアクセスしてください。ナレッジベースに関する質問、コメント、ご意見の連絡先は、Informatica ナレッジベースチーム（KB_Feedback@informatica.com）です。

Informatica マニュアル

Informatica マニュアルポータルでは、最新および最近の製品リリースに関するドキュメントの膨大なライブラリを参照できます。マニュアルポータルを利用するには、https://docs.informatica.com にアクセスしてください。

製品マニュアルに関する質問、コメント、ご意見については、Informatica マニュアルチーム（infa_documentation@informatica.com）までご連絡ください。

Informatica 製品可用性マトリックス

製品可用性マトリックス（PAM）には、製品リリースでサポートされるオペレーティングシステム、データベースなどのデータソースおよびターゲットが示されています。Informatica PAM は、https://network.informatica.com/community/informatica-network/product-availability-matrices で参照できます。

Informatica Velocity

Informatica Velocity は、Informatica プロフェッショナルサービスが開発したヒントとベストプラクティスのコレクションで、多数のデータ管理プロジェクトから得た実体験に基づいています。Informatica Velocity には、世界中の組織と連携してデータ管理ソリューションを計画、開発、デプロイ、管理する Informatica コンサルタントによる集合知を表しています。

Informatica Velocity リソースには、http://velocity.informatica.com からアクセスしてください。Informatica Velocity についての質問、コメント、またはアイデアがある場合は、ips@informatica.com から Informatica プロフェッショナルサービスにお問い合わせください。

Informatica Marketplace

Informatica Marketplace は、お使いの Informatica 製品を拡張したり強化したりするソリューションを検索できるフォーラムです。Marketplace で、Informatica デベロッパーやパートナーからの多数のソリューションを活用すれば、生産性を向上したり、プロジェクトでの実装時間を短縮したりできます。Informatica Marketplace は、https://marketplace.informatica.com からアクセスしてください。

Informatica グローバルカスタマサポート

電話または Informatica Network からグローバルサポートセンターに連絡できます。

各地域の Informatica グローバルカスタマサポートの電話番号は、Informatica Web サイト（https://www.informatica.com/services-and-training/customer-success-services/contact-us.html）を参照してください。

Informatica Network でオンラインサポートリソースを見つけるには、https://network.informatica.com にアクセスし、eSupport オプションを選択します。

第 1 章

Designer の使用

この章では、以下の項目について説明します。

Designer の使用の概要

Designer には、マッピングおよびマプレットの作成に役立つツールが用意されており、ソースとターゲットの間でデータの移動やトランスフォーメーションをどのように行うか指定できます。Designer を使用して、マッピングを構築するためのソース定義、ターゲット定義、トランスフォーメーションを作成できます。

Designer には、フォルダ、リポジトリオブジェクト、およびタスクを表示するウィンドウが含まれています。同時に複数のフォルダおよびリポジトリで作業することができます。

フォントや背景色などの一般的な Designer 設定を設定できます。 また、各 Designer ツール固有のツール設定を行うこともできます。

Designer ツール

Designer には以下のツールが用意されています。

- **フラットファイル、XML、COBOL、アプリケーション、およびリレーショナルソースのソース定義をインポートまたは作成するために使います。**
- **Target Designer。**ターゲット定義のインポートまたは作成を行います。
- **Transformation Developer。**再利用可能なトランスフォーメーションを作成します。
- **Mapplet Designer。**マップレットを作成します。

- **Mapping Designer。**マッピングを作成します。

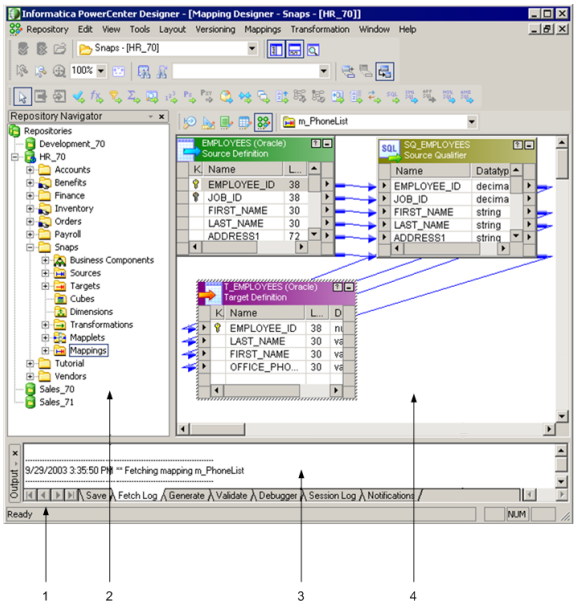
Designer ウィンドウ

Designer は次のウィンドウで構成されています。

- **ナビゲータ。**複数のリポジトリおよびフォルダに接続します。 またナビゲータを使ってオブジェクトをコピーおよび削除したりショートカットを作成したりもできます。
- **ワークスペース。**ソース、ターゲット、マップレット、トランスフォーメーション、およびマッピングを表示または編集します。 ワークスペースでは、デフォルトとワークブックという 2 つのフォーマットがある 1 つのツールを使用して作業します。ワークスペースにオブジェクトの複数バージョンを表示できます。
- **ステータスバー。**ユーザーが行う操作の状態を表示します。
- ユーザーが作業の保存やマッピングの検証などのタスクを実行する場合に、詳細情報を表示します。出力ウィンドウを右クリックすると、出力テキストの印刷、テキストのファイル保存、フォントサイズの変更などのウィンドウオプションにアクセスできます。
- **概要。**大規模なマッピングまたは多数のオブジェクトを含んだワークブックを表示します。 ［オーバービュー］ウィンドウでは、ワークスペースの表示領域がアウトライン化され、選択されているオブジェクトが色付きで強調表示されます。 オーバービューウィンドウを開くには、［表示］-［オーバービューウィンドウ］を選択してください。
- デバッガでマッピングをデバッグしているときに、トランスフォーメーションデータを表示します。
- デバッガでマッピングをデバッグしているときに、ターゲットデータを表示します。

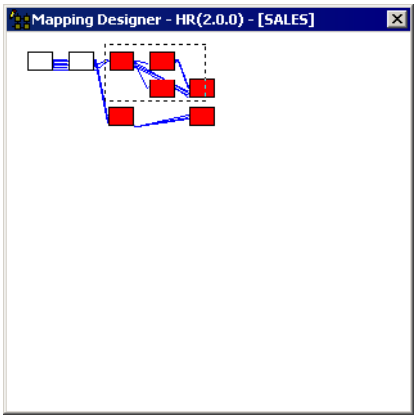
開いているウィンドウのリストを表示して、Designer のウィンドウから別のウィンドウへの切り替えができます。開いているウィンドウのリストを表示するには、［ウィンドウ］-［ウィンドウ］を選択します。

以下の図に、Designer のウィンドウを示します。



1. ステータスバー
2. ナビゲータ
3. 出力
4. ワークスペース

以下の図に、［オーバービュー］ウィンドウを示します。



ワークスペースのアウトライン表示内のオブジェクト。 色の塗られたオブジェクトはワークスペースで選択されています。

Designer オプションの設定

Designer が一般的な情報、テーブル、およびツールをどのように表示するかを設定できます。Designer では、ツールの背景色、ナビゲータウィンドウの編成、さまざまなツールで使用するフォントなどの表示オプションを指定できます。その他のオプション（Source Analyzer がプライマリキーをインポートするかどうかなど）も指定できます。変更の内容によっては、Designer を再起動して適用しなければならない場合もあります。プログラムを再起動する必要がある場合は、警告が表示されます。

Designer で設定できるオプションは次のとおりです。

- XML のインポート、ファイルのディレクトリ、ナビゲータウィンドウの構成など、一般的な表示オプションを設定できます。
- テーブル定義のカラム、サイズ、動作を設定できます。
- Designer ツールの色、フォント、その他のフォーマットオプションを設定できます。
- デバッガの表示および TCP/IP オプションを設定できます。
- **Web サービス。**Web Services Hub の下位互換性と WSDL 作成のオプションを設定できます。
- コピーウィザードおよびチェックアウトで使用する機能を設定できます。

一般的なオプションの設定

ナビゲータウィンドウ、表示ウィンドウ、およびディレクトリについて全般オプションを設定できます。

全般オプションを設定するには：

1. ［ツール］-［オプション］をクリックします。
2. ［全般］タブをクリックします。

3. 以下の全般オプションの編集ができます。

全般オプション	説明
フォルダを開く時にテーブル/マッピングを再ロードする	ツールを開いたときに前回のツールビューを再ロードします。例えば、リポジトリから切断したときにマッピングが開かれていた場合、次回開いた時には、このマッピングにはフォルダと Mapping Designer が表示されます。
テーブル/マッピングを再ロードする時に確認する	［フォルダを開く時にテーブル/マッピングを再ロードする］を選択している場合にのみ表示されます。このオプションを選択すると、フォルダを開くたびに、オブジェクトを再ロードするかどうか Designer が確認メッセージを表示します。
ツール名をワークスペースに表示する	ワークスペースまたはワークブックの左上隅にツールの名前を表示します。
オーバービューウィンドウと同期しない	デフォルトでは、［オーバービュー］ウィンドウのフォーカスをドラッグすると、ワークブックのフォーカスも同時に移動します。このオプションを選択すると、マウスボタンを離すまでワークスペースのフォーカスは変化しません。
リポジトリサービスから通知を受ける	Designer で受信した通知メッセージを、出力ウィンドウに表示できます。通知メッセージには、他のユーザーが作成、変更、または削除を行うオブジェクトに関する情報が含まれています。マッピング、マプレット、ショートカット、ソース定義、ターゲット定義、およびトランスフォーメーションに関する通知を受信します。リポジトリサービスは、ユーザーに変更を通知するので、使用中のオブジェクトが最新ではないことがわかります。Designer が通知を受信するには、オブジェクトを格納しているフォルダをナビゲータで開いておく必要があります。Designer が変更または削除に関する通知を受信するには、ワークスペースで対象オブジェクトを開いておく必要があります。リポジトリサービスを管理するユーザーが投稿したユーザー作成の通知も受信されます。デフォルトでは有効になっています。
すべての MX データを保存する	Designer にマッピングを保存する場合は、すべての MX データを保存します。サードパーティ製のリポジトリツールに MX ビューを使用するには、このオプションを選択します。 マッピングのため MX データを保存する場合、PowerCenter はマッピング内の各ターゲットフィールド用にフィールド式を作成します。これらの式は、式エディタで作成するトランスフォーメーション式とは異なります。 MX データにアクセスする前に、マッピングの保存とチェックインが必要です。 デフォルトでは無効になっています。 **注**: 注：MX データを保存すると、リポジトリのパフォーマンスに影響が出る場合があります。MX ビューを使用する場合に限り、このオプションを使用します。
ソース/ターゲットの依存性情報のみ保存する	Designer にマッピングを保存する場合、依存性に関連する MX データのみを保存します。MX ビューのフィールドに式を表示する必要がなくても Repository Manager を使用してソース/ターゲットの依存性を表示するという場合に限り、このオプションを選択します。デフォルトでは無効になっています。

全般オプション	説明
データベース別グループソース	ナビゲータに、データベースごとにグループ分けしてソースを表示します。このオプションを指定しない場合、ソースは名前のアルファベット順で表示され、データベース名はかっこ内に表示されます。
ビジネス名を使ってソースを表示する	ナビゲータで、ソースをビジネス名で表示します。ソースをビジネス名で表示するオプションを選択すると、ナビゲータはまずビジネス名を表示し、次にソースタイプ名とテーブル名をかっこで囲んで表示します。
ビジネス名を使ってターゲットを表示する	ナビゲータに、ターゲットをビジネス名で表示します。ターゲットをビジネス名で表示するオプションを選択すると、ナビゲータはまずビジネス名を表示し、次にターゲットタイプ名とテーブル名をかっこで囲んで表示します。
ワークスペースのファイル	Designer が作成したワークスペースファイルのディレクトリ。 ワークスペースファイルに保持されるのは、ユーザーが前回最後に開いたソースやターゲット、または前回最後に保存したマッピングです。ファイルの破損や複数ユーザーによる上書きを防止するため、このディレクトリはローカルディレクトリとする必要があります。デフォルトでは、Designer はインストールディレクトリにファイルを作成します。
HTML 文書パスまたは URL	ビジネス文書にリンクを入力するための HTML または URL パス。

テーブルオプションの設定

テーブル定義のカラム、サイズ、動作を設定できます。

テーブル定義のオプションを設定するには：

1. ［ツール］-［オプション］をクリックします。
2. ［テーブル］タブをクリックします。
3. ［テーブル］リストから、設定したいリポジトリオブジェクトを選択します。

   ［カラム］セクションは、各リポジトリオブジェクトに対するパラメータを一覧表示します。上向きおよび下向きの矢印を使用すると、カラムの表示順を変更できます。
4. 選択したオブジェクトタイプに対して、以下のオプションを設定できます。

テーブルオプション	ツールの使用可否	説明
デフォルトの幅（カラム）	すべてのツール	テーブルカラムのデフォルトの幅（バイト）。
利用可能	すべてのツール	リポジトリオブジェクトがフルサイズである場合に表示されるすべての利用可能なカラム。
選択済み	すべてのツール	リポジトリオブジェクトがフルサイズである場合に、Designer に表示されるカラム。カラムを表示するには、［利用可能］リストでカラムを選択し、選択したカラムを矢印をクリックして［選択済み］リストに移動します。カラムの削除や表示順の変更も可能です。
ツール	すべてのツール	設定したい Designer ツールを選択します。

テーブルオプション	ツールの使用可否	説明
プライマリキーのインポート	Source Analyzer、Target Designer	Designer は、リレーショナルソース定義またはターゲット定義のプライマリキーをインポートします。
外部キーのインポート	Source Analyzer、Target Designer	Designer は、リレーショナルソース定義またはターゲット定義の外部キーをインポートします。
ソースを開く時にソース修飾子を作成する	Mapping Designer、Mapplet Designer	Designer は、マッピングに追加する各ソースにソース修飾子トランスフォーメーションまたはノーマライザトランスフォーメーションを作成します。すべてのソース修飾子およびノーマライザを手動で作成する場合は、このオプションのチェックマークを外します。ソース修飾子を手動で作成した方が便利なのは、ソース修飾子を用いてリレーショナルテーブルを結合する場合などです。
ソース修飾子に対してビジネス名をポート名として使用する	Mapping Designer、Mapplet Designer	ソース修飾子は、ビジネス名をカラム名として使用します。
全般的なデフォルトの幅	すべてのツール	すべてのリポジトリオブジェクトのデフォルトの幅。
全般的なデフォルトの高さ	すべてのツール	すべてのリポジトリオブジェクトのデフォルトの高さ。
ツールチップを表示(S)	すべてのツール	ポインタをカラムの上またはオブジェクトのタイトルバーの上に移動させると、カラムまたはリポジトリオブジェクトの名前が表示されます。このオプションを選択した場合、オブジェクトのタイトルバーのアイコン上にポインタを移動させると、ビジネス名も表示されます。このオプションは、すべてのリポジトリオブジェクトに適用されます。

［テーブル］タブの一部のオプションは、編集するために選択したツールに基づいて無効化されています。このタブのデフォルト設定に戻すには、［全て元に戻す］をクリックします。

5. 設定するリポジトリオブジェクトタイプで、それぞれ 3 と 4 の手順を繰り返します。

フォーマットオプションの設定

Designer の各ツールに対して、色、フォント、その他のフォーマットオプションを設定できます。

フォーマットオプションを設定するには：

1. ［ツール］-［オプション］をクリックします。
2. ［フォーマット］タブをクリックします。
3. カラーテーマを適用するには、［テーマの選択］をクリックします。
4. ［ワークスペースの色］セクションでは、［ツール］メニューで Designer ツールを選択して、ワークスペースの色を設定します。
5. ワークスペースの要素の色を変更するには、ワークスペースの要素を選択してから、［色］をクリックします。

［ワークスペースの色］セクションの要素は、設定するために選択したツールによって異なります。以下の要素の色を設定できます。

要素	ツールの使用可否	説明
背景	すべてのツール	ワークスペースの背景。
フォアグラウンドテキスト	すべてのツール	ワークスペースに表示されるテキスト。
選択されたリンク	Source Analyzer、Target Designer、Mapplet Designer、Mapping Designer	ワークスペース内のリポジトリオブジェクト間で選択されたリンク。
関連するリンク	Source Analyzer、Target Designer、Mapplet Designer、Mapping Designer	2 つのテーブル間での、プライマリキーと外部キーの関係を示すリンク。
リンクのプロパゲート	Mapplet Designer、Mapping Designer	ポート属性のプロパゲートに影響されたリンク。
データフローリンク	Mapplet Designer、Mapping Designer	ワークスペース内のマプレットまたはマッピングにおける、リポジトリオブジェクトのポート間のリンク。
メタデータフローリンク;	Mapplet Designer、Mapping Designer	MQ ソース修飾子ンスフォーメーションおよび関連ソースの間のリンク。

6. ［キャプションの色］セクションの［テーブル］メニューでオブジェクトタイプを選択して、タイトルバーのテキストおよび背景の色を設定します。

   以下の色を設定できます。

オプション	説明
フォアグランド	テーブルキャプションのテキストの色。
背景	テーブルキャプションの背景色。
バックグランド（2）	テーブルキャプションの 2 番目の背景色。2 色の背景色を組み合わせて、リポジトリオブジェクトのタイトルバーを階調表示できます。階調表示しない場合は、［バックグランド］と［バックグランド（2）］に同じ色を選択してください。

7. ［フォント］セクションの［カテゴリ］メニューで Designer のコンポーネントを選択し、そのフォントを表示または変更します。

   ［現在のフォント］フィールドには、現在選択されている Designer コンポーネントのフォントが表示されます。［変更］をクリックして、［カテゴリ］メニューで選択されている Designer コンポーネントの表示フォントおよび言語スクリプトを変更します。のコードページがソースファイルのコードページと異なっている場合には、この操作を行う必要が生じる場合があります。

カラーテーマの使用

Designer ツール内のワークスペース要素の色を素早く選択するには、カラーテーマを使用します。カラーテーマを適用すると、Designer ツール内のすべてのワークスペース要素の色がトータルで更新されます。次の標準カラーテーマの中から選択できます。

- **Informatica クラシック。**これは、ワークスペース要素の標準的な色指定です。ワークスペースの背景色はグレー、ワークスペースのテキストは白、リンクの色は青、赤、ブルーグレー、濃い緑、および黒です。
- **ハイコントラストブラック。**明るいリンクの色が黒の背景に対してはっきりと表示されます。 ワークスペースの背景色は黒、ワークスペースのテキストは白、リンクの色は紫、赤、明るい青、明るい緑、および白です。
- **色付きの背景。**Designer ツールごとに、ワークスペースの背景色は異なるパステルカラーになっています。ワークスペースのテキストは黒、リンクの色は［Informatica クラシック］カラーテーマのものと同じです。

Designer ツールのカラーテーマを選択した後で、ワークスペース要素の色を個別に変更できます。個々の要素に対する変更は、［テーマセレクタ］ダイアログボックスの［プレビュー］セクションには表示されません。

Designer ツールのカラーテーマを選択するには、以下のとおりに実行します。

1. Designer で、［ツール］-［オプション］の順にクリックします。
2. ［フォーマット］タブをクリックします。
3. ［フォーマット］タブの［色のテーマ］セクションで、［テーマの選択］をクリックします。
4. ［テーマ］メニューからテーマを選択します。
5. ［プレビュー］セクションでタブをクリックし、それぞれの Designer ツールでワークスペース要素がどのように表示されるかを確認します。
6. ［OK］をクリックして、カラーテーマを適用します。

デバッグオプションの設定

［デバッグ］タブで、デバッガの表示および TCP/IP のオプションを設定できます。

デバッグオプションを設定するには：

1. ［ツール］-［オプション］をクリックします。
2. ［デバッグ］タブをクリックします。
3. デバッガに対して以下のオプションを設定できます。

オプション	説明
データ表示	ターゲットデータウィンドウとインスタンスデータウィンドウに表示する情報を選択します。
TCP/IP	Designer がに接続する際の特定の TCP/IP ポートまたはポートの範囲を選択します。

Web サービスオプションの設定

［WebService］タブで Web サービスオプションを設定できます。

Web サービスオプションを設定するには：

1. ［ツール］-［オプション］をクリックします。
2. ［WebService］タブをクリックします。

3. Web サービスに対して以下のオプションを設定できます。

オプション	説明
メッセージポートとヘッダポート用に個別のグループを作成	メッセージポートを Web サービスのソースおよびターゲットの定義に追加するときに、メッセージポートとヘッダポート用に個別の XML ビューを作成するかどうかを示します。［メッセージポートとヘッダポート用に個別のグループを作成］オプションを選択すると、Designer はメッセージ ID 用に 1 つのビューを作成し、ヘッダポート用に別のビューを作成します。 個別のビューにメッセージポートとヘッダポートを含むソース定義とターゲット定義は、Web サービスのソース定義またはターゲット定義の以前のバージョンのフォーマットと一致します。このオプションは、PowerCenter バージョン 8.1x への下位互換性のために提供されています。
フォールト用の個別ターゲットの作成	フォールト応答に個別のターゲット定義を作成するかどうかを示します。フォールト応答に個別のターゲット定義を作成する場合、Designer は出力メッセージの各フォールトにターゲット定義を作成します。出力メッセージの各フォールトに個別の定義を含む Web サービスターゲットは、Web サービスのターゲット定義の以前のバージョンのフォーマットと一致します。このオプションは、PowerCenter バージョン 8.1x への下位互換性のために提供されています。
WSDL 作成	リレーショナルまたはフラットファイルのソースまたはターゲット、トランスフォーメーション、またはマプレットから生成されたマッピングの WSDL を生成するかどうかを示します。WSDL なしで Web サービスマッピングを生成する場合、Designer ではマッピングを作成した後で WSDL を作成できます。Designer が WSDL ファイルを作成するディレクトリを選択するには、［参照］ボタンをクリックします。WSDL ファイルの名前は.wsdl の拡張子が付いたマッピング名です。

その他のオプションの設定

［詳細設定］タブでは、コピーウィザードおよびバージョン管理のオプションを設定できます。

その他のオプションを設定するには、以下のとおりに実行します。

1. ［ツール］-［オプション］をクリックします。
2. ［詳細設定］タブをクリックします。
3. 以下のオプションを設定することができます。

オプション	説明
「名前の変更」が選択されたときに一意な名前を生成する	コピーウィザードウィンドウで［名前の変更］オプションを選択すると、コピーしたオブジェクトにそれぞれ固有の名前が生成されます。例えば、ソースオブジェクト s_customers がコピー先フォルダのソースと同じ名前である場合、［名前の変更］により一意な名前 s_customers1 が生成されます。
ナビゲータにチェックアウトアイコンを表示する	チェックアウトされているオブジェクトにチェックアウトアイコンを表示します。

オプション	説明
チェックアウトせずに削除を許可	バージョン管理されたリポジトリオブジェクトを、先にチェックアウトせずに削除できます。しかし、他のユーザーがチェックアウトしたオブジェクトは削除できません。このオプションを選択した場合、ユーザーがオブジェクトを削除するとリポジトリサービスがそのユーザーに対してオブジェクトをチェックアウトします。
削除済みオブジェクトを保存した後、自動的にチェックイン	削除済みオブジェクトを、ユーザーがその変更をリポジトリに保存した後チェックインします。このオプションをクリアすると、削除されたオブジェクトはチェックアウトされたままになり、Results View からチェックインする必要があります。
全て元に戻す	［その他］オプションをすべてデフォルト値にリセットします。

ツールバーの使用

ツールバーを使用すると、ツールやタスクをすばやく選択することができます。ツールバーの表示と非表示を設定できます。また、新しいツールバーの作成や、ツールバーのボタンの追加や削除を行うこともできます。

Designer ツールバー

Designer を設定して以下のツールバーを表示できます。

- リポジトリおよびフォルダへの接続/接続解除や、ビューとカラムの切り替えを行うボタンが含まれています。
- リポジトリおよびフォルダへの接続／接続解除、オブジェクトのエクスポートとインポート、変更の保存、およびワークスペースの印刷を行うボタンが含まれています。
- ツールバーの設定、ウィンドウの切り替え、全画面表示の切り替え、ワークスペースモードの変更、プロパティの表示を行うボタンが含まれています。
- リポジトリオブジェクトのアイコン化、整列、カラムのコピー、リンクと移動、およびワークスペースの拡大縮小を行うボタンが含まれています。
- **マッピング/マップレット。**マッピングおよびマップレットを作成、編集、およびパースするためのボタンが含まれています。
- **トランスフォーメーション。**トランスフォーメーションを作成するボタンが含まれています。
- **詳細トランスフォーメーション。**詳細トランスフォーメーションを作成するボタンが含まれています。
- **バージョン管理:ツールバー**オブジェクトのチェックイン、チェックアウトの取り消し、チェックアウトの検索、履歴の表示、ラベルの取り消し、およびクエリーの管理を行うボタンが含まれています。
- **デバッガ。**デバッガを開始、停止、および継続するボタンが含まれています。
- 他の PowerCenter クライアントアプリケーションに接続するためのボタンがあります。［ツール］ボタンを使用して他の PowerCenter クライアントアプリケーションを開くと、PowerCenter は同じリポジトリ接続を使用してリポジトリに接続し、同じフォルダを開きます。

ツールバーの一部のボタンを表示したくない場合は、ツールバーを設定することができます。

Workflow Manager ツールバー

Workflow Manager では、ツールを選択したり操作をすばやく実行したりするのに便利な以下のツールバーを表示できます。

- リポジトリやフォルダとの接続/接続解除、ウィンドウのトグル、ズーム拡大/縮小、ワークスペースのパン、オブジェクトの検索を行うためのボタンが含まれています。
- 接続、サービス、およびサーバーグリッドの作成と編集を実行するためのボタンが含まれています。
- リポジトリとの接続/接続解除、リポジトリの追加、フォルダのオープン、ツールのクローズ、変更内容のリポジトリへの保存、ワークスペースの印刷を行うためのボタンが含まれています。
- ツールバーのカスタマイズ、ステータスバーとウィンドウの切り替え、全画面表示の切り替え、新規ワークブックの作成、オブジェクトのプロパティの表示を実行できます。
- ワークスペースのオブジェクトの並べ替え、オブジェクトの検索、ズーム拡大/縮小、およびワークスペースのパンを行うためのボタンが含まれています。
- **タスク。**タスクを作成するためのボタンが含まれています。
- **ワークフロー。**ワークフロープロパティを編集するためのボタンが含まれています。
- ワークフローのスケジュール/起動、タスクの起動を行うためのボタンが含まれています。
- オブジェクトのチェックイン、チェックアウトの取り消し、バージョンの比較、チェックアウトしたオブジェクトの一覧表示、およびリポジトリクエリーの一覧表示を実行するためのボタンが含まれています。
- 他の PowerCenter クライアントアプリケーションに接続するためのボタンがあります。［ツール］ボタンを使用して他の PowerCenter クライアントアプリケーションを開くと、PowerCenter は同じリポジトリ接続を使用してリポジトリに接続し、同じフォルダを開きます。デフォルトでは、PowerCenter はツールツールバーを表示します。

Workflow Monitor のツールバー

Workflow Monitor では、操作をすばやく実行するのに便利な以下のツールバーを表示できます。

- リポジトリへの接続／リポジトリからの切断、印刷、印刷プレビューの表示、ワークスペースの検索、タスクビューでのナビゲータの表示／非表示、および出力ウィンドウの表示／非表示といった各種操作を行うためのボタンが含まれています。
- **Integration Service。**Integration Service との間での接続および切断、Integration Service の ping、ワークフロー操作の実行のためのボタンが含まれます。
- 時間増分を設定したり、プロパティ、ワークフローログ、およびセッションログを表示したりするためのボタンが含まれています。
- 最新の実行を表示したり、タスク、Integration Services、およびフォルダをフィルタリングしたりするためのボタンが含まれています。

Repository Manager のツールバー

Repository Manager には、デフォルトでは以下のツールバーが表示されています。

- ここには、以下のボタンがあります。リポジトリとの間の接続および接続解除、選択したオブジェクトの依存性の表示、キーワードによる検索、オブジェクトプロパティの表示、依存性ウィンドウを閉じる、ナビゲータウィンドウおよび出力ウィンドウのトグルの切り替え。
- 他の PowerCenter クライアントアプリケーションに接続するためのボタンがあります。［ツール］ボタンを使用して他の PowerCenter クライアントアプリケーションを開くと、PowerCenter は同じリポジトリ接続を使用してリポジトリに接続し、同じフォルダを開きます。

ツールバーの表示

ツールバーを常に表示するように設定することができます。Designer、Workflow Manager、Workflow Monitor で、表示するツールバーを設定できます。

ツールバーを表示するには：

1. ［ツール］-［カスタマイズ］をクリックします。
2. ［ツールバー］タブで、表示したいツールバーを選択します。
3. ツールバーのボタンにポインタを置いたときにツールチップを表示させるには、［ツール］を選択します。
4. ［OK］をクリックします。

ツールバーの作成

新しいツールバーを作成して、新しいツールバー用のボタンを選択することができます。Designer、Workflow Manager、Workflow Monitor で、ツールバーを作成することができます。

ツールバーを新たに作成するには：

1. ［ツール］-［カスタマイズ］をクリックします。
2. ［ツールバー］タブで、［新規］をクリックします。
3. 新しいツールバーの名前を入力し、［OK］をクリックします。

   フローティングツールバーが新たに表示され、ツールバーの名前がツールバーリストに表示されます。
4. ［コマンド］タブをクリックします。
5. ［カテゴリ］リストからツールバーを選択して、利用できるボタンを表示します。
6. 必要なボタンを、［ボタン］領域から新しいフローティングツールバーにドラッグします。
7. ツールバーの作成が終わったら、［OK］をクリックします。

ツールバーの設定

ツールバーを設定するには、ツールバーにボタンを追加するか、またはツールバーのボタンを削除します。Designer、Workflow Manager、Workflow Monitor で表示するツールバーを設定することができます。

ツールバーを設定するには：

1. ［ツール］-［カスタマイズ］をクリックします。
2. 設定対象のツールバーを選択していることを確かめます。

   ツールバーが表示されます。
3. ツールバーからボタンを削除するには、このボタンをツールバーから［カスタマイズ］ダイアログボックスにドラッグします。
4. ボタンを追加するには、［コマンド］タブをクリックします。
5. ［カテゴリ］リストからツールバーを選択して、利用できるボタンを表示します。
6. 必要なボタンを、［ボタン］領域からカスタマイズしたツールバーにドラッグします。
7. ツールバーのカスタマイズが終わったら、［OK］をクリックします。

カスタムツールの追加

Designer では、［ツール］メニューにカスタムツールを追加できます。頻繁に使用するプログラムを Designer から起動できます。例えば、Designer 内から Business Objects Designer またはウェブブラウザを起動できます。

ユーザー作成のツールを追加すると、［ツール］メニューの末尾にそのツールの名前が追加されます。ツールを起動するには、［ツール］メニューをクリックしてユーザー作成のツールを選択します。

ユーザー作成のツールは追加や削除、順序の変更を行うことができます。［工具］-［カスタマイズ］の順にクリックし、［工具］タブをクリックします。

［ツールリスト］リストに、ユーザー作成のツールが表示されます。このリストに表示される順序で、［ツール］メニューの末尾にツールの名前が並べられます。

ユーザー作成のツールを追加するには：

1. ［カスタムツールの追加］をクリックします。

   最大で 9 個のツールを［ツール］メニューに追加できます。
2. ［ツール一覧］フィールドで、ユーザー作成のツールの固有の名前を入力します。この名前が［ツール］メニューに表示されます。

   ヒント: ヒント：クイックアクセスキーとして使用する文字の前には、アンパサンド（&）を置きます。
3. 矢印ボタンを使って、新しいツールを［ツール］メニュー上の適切な位置に配置します。
4. ツールについて、次の情報を入力します。

オプション	説明
コマンド	ツールの実行ファイルの名前とファイルパス。[Browse］をクリックして、実行ファイルを選択します。
引数	Designer がカスタムツールに渡す引数。ツールによって、引数が必須の場合とオプションの場合があります。 ユーザー作成のツールを起動するときに引数の入力を求めるメッセージを表示させたい場合には、［引数のプロンプト］を選択します。
初期ディレクトリ	カスタムツールが起動されるディレクトリ。初期ディレクトリを入力しなければ、ユーザー作成のツールのプログラム実行ファイルのディレクトリが使用されます。

5. ［OK］をクリックします。

ワークスペースでの操作

ワークスペースでリポジトリオブジェクトを表示または編集する場合、以下のタスクを完了すると、ワークスペースを移動しやすくなります。

- カラムまたはポートを検索する。
- リポジトリオブジェクトのサイズを変更する。
- ワークスペースを拡大縮小する。

検索ツール

Designer には、次を検索ツールおよびワークスペースを検索ツールが用意されているので、リポジトリオブジェクト内のカラムやポート、または出力ウィンドウ内の文字列を検索できます。

次を検索

次を検索ツールを使用して、以下に含まれるカラム名やポート名を検索できます。

- トランスフォーメーション
- マップレット
- ソース定義
- ターゲット定義

次を検索ツールでは、一度に 1 つのオブジェクトを検索できます。複数のオブジェクトを同時に検索することはできません。次を検索は、各 Designer ツールで使用できます。トランスフォーメーションを 1 つ選択するか、出力ウィンドウ内でクリックしてから、検索を実行します。

標準ツールバーの［次を検索］ボックスには、最後に検索した文字列から 10 個までが保存されます。

カラム名またはポート名を検索するには：

1. 該当するトランスフォーメーション、マプレット、ソースまたはターゲット定義を選択するか、出力ウィンドウをクリックします。
2. 標準ツールバーの［検索］ボックスに、検索したいテキストを入力します。例えば、address を探す場合には、「**add**」などと入力します。

   ヒント: 検索は、大文字小文字を区別しません。
3. 文字列を検索するには、［編集］-［次を検索］を選択するか、［次を検索］をクリックするか、Enter キーを押します。

   Designer は、最初に見つかった検索文字列を表示します。

   ヒント: F3 キーを押して文字列を検索することもできます。
4. Enter キーをもう一度押すと、次に見つかった検索文字列が表示されます。Designer はトランスフォーメーション内の各ポート名またはカラム名を検索し、最後までいったらトランスフォーメーションの先頭に戻り、該当するものが見つかるまで探します。

ワークスペースの検索

出力ウィンドウの［保存］、［生成］、［検証］タブの文字列を検索できます。ワークスペースを検索ツールは、ワークスペース内のすべてのトランスフォーメーションにおいてフィールド名またはトランスフォーメーション名を検索します。

ワークスペースを検索ツールによって、ワークスペース内のすべてのトランスフォーメーションにおいてポート名またはトランスフォーメーション名を検索できます。検索文字列と一致するカラム名、ポート名、またはテーブル名を検索できます。ワークスペース内のすべての名前に対して検索を行うか、テーブル、カラム、またはポートのビジネス名に対して検索を行うかを指定できます。また、検索文字列と完全に一致する語を検索するか、大文字小文字を区別して検索するかを選択することもできます。

ワークスペース内のカラム名、ポート名、トランスフォーメーション名を検索するには：

1. Designer ツールで、［ワークスペースを検索］をクリックするか、または［編集］-［ワークスペースを検索］の順にクリックします。

   ［ワークスペースを検索］ダイアログボックスが開きます。

2. 検索テキストに一致するフィールド名を検索するか、またはテーブル名を検索するかを選択します。

［検索］オプション	説明
フィールド	Designer は、検索テキストに一致するカラム名またはポート名を検索します。
テーブル	Designer は、検索テキストに一致するテーブル名を検索します。

3. ［検索］入力フィールドに検索文字列を入力するか、またはリストから検索文字列を選択します。

   Designer は、最後に検索した文字列から 10 個までをリストに保存します。

4. 検索テキストに一致するものをすべての名前の中から探すか、ビジネス名の中から探すかを指定します。

［入力］オプション	説明
名前	Designer は、ワークスペース内のテーブル、カラム、またはポートのすべての名前に対して検索を行います。
ビジネス名	Designer は、ワークスペース内のテーブル、カラム、またはポートのビジネス名に対して検索を行います。

5. 検索条件を指定します。

検索オプション	説明
すべてのテーブルを検索	このオプションを選択すると、ワークスペース内のすべてのテーブルで検索が行われます。このオプションを選択しない場合、Designer はワークスペース内で現在選択されているすべてのテーブルに対して検索を行います。検索文字列に一致するテーブル名を検索する場合、このオプションが選択されます。
単語単位で検索	このオプションを選択すると、指定した検索文字列と一致する名前またはビジネス名が検索されます。
大文字小文字を区別する(M)	指定した検索文字列と一致する語を、大文字と小文字を区別して検索したい場合に選択します。

6. ［検索］をクリックします。

   一致した語は、［ワークスペースを検索］ダイアログボックスの下部にあるテーブルにすべて表示されます。

7. ［閉じる］をクリックします。

ワークスペースオブジェクトのアイコン化とリストア

各 Designer ツールはワークスペースに各種のオブジェクトを表示します。開いているツールに応じて、ソース定義、ターゲット定義、トランスフォーメーション、マプレット、またはマッピング全体を表示することができます。Designer はリポジトリオブジェクトを次のフォーマットで表示することができます。

- Designer は、各オブジェクト内の情報をカラムに表示します。デフォルトでは、Designer は通常モードでオブジェクトを表示します。

- Designer はオブジェクトを名前付きのアイコンとして表示します。大きなマッピングを取り扱う場合には、オブジェクトのアイコン化が必要となる場合があります。ポインタをアイコンに合わせると、アイコン化されたオブジェクトの説明を表示できます。
- ズーム機能を使用すると、ワークスペース内の通常のオブジェクトおよびアイコン化されたオブジェクトの拡大レベルを変更できます。適用可能な拡大レベルは、10%単位で 30%から 100%です。

ワークスペースオブジェクトのアイコン化

ワークスペースのオブジェクトをアイコンとして表示できます。

ワークスペースオブジェクトをアイコン化するには：

1. ワークスペースでオブジェクトを選択します。

   複数のオブジェクトを選択するには、Shift キーまたは Ctrl キーを押しながらオブジェクトをクリックします。［編集］-［すべてを選択］を選択するか、マウスをドラッグして選択したいオブジェクトを矩形で囲んでも、複数のオブジェクトが選択できます。
2. ［レイアウト］-［アイコン化して整列］をクリックします。

   ヒント: 右クリックして［アイコン化して整列］を選択する方法もあります。

ワークスペースオブジェクトのリストア

アイコン化したオブジェクトを通常サイズにリストアできます。

アイコン化したワークスペースオブジェクトを通常サイズにリストアするには：

1. オブジェクトを選択します。

   複数のオブジェクトを選択するには、Shift キーまたは Ctrl キーを押しながらオブジェクトをクリックします。

   ［編集］-［すべてを選択］を選択するか、マウスをドラッグして選択したいオブジェクトを矩形で囲んでも、複数のオブジェクトが選択できます。
2. ［レイアウト］-［整列］をクリックします。

   ヒント: アイコン化したオブジェクトをダブルクリックするか、オブジェクトを右クリックして［整列］を選択しても、オブジェクトをリストアできます。

ワークスペースオブジェクトの整列

Mapping Designer で特定のパイプラインを表示することができます。

Mapping Designer で特定のパイプラインを表示するには：

1. ［レイアウト］-［整列］をクリックします。
2. ターゲットごとにパイプラインを選択します。

   ［アイコン化］オプションを選択すると、整列されたパイプラインをアイコン化した形式で表示できます。
3. ［OK］をクリックします。

ワークスペースのズーム

ワークスペースを拡大縮小表示することができます。ズームレベルを設定するには、ツールバーまたは［レイアウト］メニューオプションを使用します。ツールバーには以下のズームオプションがあります。

- **［ズームイン 10%］ボタン。**選択したポイントを中心点として、現在の倍率を 10%単位で増大させます。
- **［ズームアウト 10%］ボタン。**選択したポイントを中心点として、現在の倍率を 10%単位で減少させます。

- **矩形に基づくズームイン。**選択した矩形領域の現在の倍率を増大させます。 拡大の度合いは、選択した領域のサイズ、ワークスペースのサイズ、および現在の倍率によって決まります。
- ワークスペースの中心点を変更せずに、ズームレベルをドロップダウンリストから選択したパーセンテージに設定します。
- **全体を表示。**すべてのワークスペースオブジェクトがワークスペースに一致するように調整します。

［レイアウト］メニューには以下のズームオプションがあります。

- ワークスペースの中心点を維持し、10%単位で拡大または縮小します。
- 選択した点を中心点として、10%単位で拡大または縮小します。
- **ズーム - 範囲指定。**選択した矩形領域の現在の倍率を増大させます。 拡大の度合いは、選択した領域のサイズ、ワークスペースのサイズ、および現在の倍率によって決まります。
- **ズームを戻す。**ズームレベルを 100%に設定します。
- **全体を表示。**すべてのワークスペースオブジェクトがワークスペースに一致するように調整します。
- ワークスペースの中心点を維持し、ズームレベルを選択したパーセンテージに設定します。

オブジェクトをワークスペースに追加すると、Designer は他のオブジェクトの拡大レベルを使用します。終了時には、Designer は各 Designer ツールのズームレベルをフォルダに保存します。

ワークスペースの表示

［表示］-［全画面］を選択すると、ワークスペースウィンドウのサイズを最大化できます。全画面表示では、標準ツールバーおよびトランスフォーメーションツールバーが表示されます。メニュー、ナビゲータウィンドウ、出力ウィンドウ、およびタイトルバーは非表示になります。全画面表示でメニューを使用するには、画面の上部で任意の場所をポイントしてメニューを表示させます。

通常の表示に戻すには、［Close Full Screen］をクリックするか Esc キーを押します。

Designer タスク

各 Designer ツールで、以下のタスクを実行できます。

- リポジトリの追加。
- ワークスペースの印刷。
- オブジェクトが最後に保存された日付と時間の表示。
- フォルダを開く/閉じる。
- ショートカットの作成。
- リポジトリオブジェクトのチェックインおよびチェックアウト。
- リポジトリオブジェクトの検索。
- リポジトリオブジェクトの説明の入力。
- ワークスペース内におけるオブジェクトの旧バージョンの表示。
- 前回保存したオブジェクトのバージョンに戻す。
- オブジェクトのコピー。
- リポジトリオブジェクトのエクスポートとインポート。

- 複数のオブジェクト、ポート、カラムを扱う作業。
- ポートの名前の変更。
- オブジェクトの更新。
- ショートカットキーの使用。

オブジェクトの依存性は、Designer で表示することもできます。

リポジトリの追加

リポジトリオブジェクトを編集するには、最初にリポジトリをナビゲータに追加して、リポジトリオブジェクトへのアクセスを可能にします。ナビゲータにリポジトリを追加するには、［リポジトリ］-［リポジトリの追加］をクリックします。［リポジトリの追加］ダイアログボックスを使用して、リポジトリを追加します。

ワークスペースの印刷

ワークスペースを印刷する手順

▶ ［リポジトリ］-［印刷］をクリックします。

ワークスペースを右クリックして［印刷］を選択する方法もあります。

ワークスペースの印刷出力の設定

ワークスペースの印刷出力用のヘッダおよびフッタを指定する手順

▶ ［リポジトリ］-［ページ設定］をクリックします。

ワークスペースの印刷出力のプレビュー

印刷前にワークスペースをプレビューする手順

▶ ［リポジトリ］-［印刷プレビュー］をクリックします。

最終保存日時の表示

オブジェクトがリポジトリで最後に保存された日時を表示できます。「最終保存」日時を表示するには、ナビゲータでオブジェクトを選択し、［表示］-［プロパティ］を選択します。

注: ソースには「最終保存」日時はありません。

Windows の場合は、コントロールパネルの［地域と言語のオプション］設定を使用すると、「最終保存」日時のフォーマットを設定できます。

フォルダを開く/閉じる

フォルダをダブルクリックして、フォルダとツールを同時に開きます。フォルダをダブルクリックすると、ナビゲータでフォルダが開かれ、フォルダ内で最後にアクティブだったツールが表示されます。

フォルダを選択して、ワークスペースを開くツールを指定することもできます。［フォルダを開く時にテーブル/マッピングを再ロードする］オプションを選択している場合には、Designer はそのツールの使用時に最後に開いていたオブジェクトも表示します。

例えば Source Analyzer がアクティブな状態でフォルダを閉じた場合、次回に［Open Folder］を使用してフォルダを開くと、Designer は自動的に Source Analyzer を表示します。

フォルダを閉じる手順

- Navigator でフォルダを選択して、［切断］ボタンをクリックします。 開いているフォルダおよびツールをすべて閉じるには、［リポジトリ］-［すべてのツールを閉じる］を選択します。

ショートカットの作成

共有フォルダのオブジェクトへのショートカットを作成するには、このオブジェクトを目的のフォルダまたはマッピング内にドラッグします。例えば、ソースへのショートカットを作成するには、このソースを共有フォルダから、ワークスペースで開いているマッピング内にドラッグします。ソースを目的のフォルダ内にドロップしてもショートカットを作成できます。新しいショートカットを使用するには、それをワークスペースにドラッグします。

共有フォルダへのショートカットは同じリポジトリ内に作成できます。また、グローバルリポジトリ内の共有フォルダへのショートカットをローカルリポジトリに作成することもできます。この場合、両方のリポジトリが同じドメインにあることが条件です。

オブジェクトをコピーする際には、コピー先のフォルダが開いている必要があります。

注: 非共有フォルダのオブジェクトへのショートカットを作成することはできません。

バージョン管理されたオブジェクトのチェックアウトおよびチェックイン

バージョン管理されたオブジェクトで作業している場合、オブジェクトを変更するにはまずチェックアウトし、リポジトリに保存して変更内容をコミットする必要があります。オブジェクトをチェックインすることで、オブジェクト履歴に新しいバージョンが追加されます。他のユーザーにそのオブジェクトの変更を許可するには、オブジェクトをチェックインする必要があります。

オブジェクトの検索

オブジェクトクエリーを使用して、リポジトリにある、指定された条件を満たすオブジェクトを検索します。クエリーを実行すると、指定した条件に基づいて結果が返されます。オブジェクトクエリーを作成して、次のようなタスクを実行することができます。

- **開発中のリポジトリオブジェクトの追跡**。［Label］、［User］、［Last Saved Time］、または［Comments］パラメータをクエリーに追加して、オブジェクトを開発中に追跡することができます。
- **デプロイメントグループへのクエリーの関連付け**。動的デプロイメントグループを作成するときに、そのグループとオブジェクトクエリーとの関連付けが行えます。

オブジェクトクエリーを作成するには、［ツール］-［クエリ］をクリックしてクエリブラウザを開きます。

クエリブラウザから、クエリーを作成、編集、および削除することができます。また、クエリブラウザから各クエリーの権限を設定することもできます。クエリブラウザでは、読み込み権限のあるクエリーを実行できます。

リポジトリオブジェクトの説明の入力

各リポジトリオブジェクトについて説明およびコメントを入力できます。2,000 バイト/K までの文字を入力できます。K は、選択されているリポジトリコードページにおいて文字が含むことができる最大バイト数です。例えばリポジトリのコードページが日本語コードページ（K=2）の場合、説明フィールドおよびコメントフィールドにはそれぞれ 1,000 文字まで入力できます。

バージョン管理されたリポジトリオブジェクトの表示と比較

様々なオブジェクトを表示して、比較できます。オブジェクトに複数のバージョンがある場合、［履歴の表示］ウィンドウでそのオブジェクトのバージョンを検索できます。1 つのウィンドウのあるオブジェクトの各種バージョンを比較するだけでなく、そのワークスペース内のオブジェクトのさまざまなバージョンを、分かりやすく比較して表示できます。

バージョン管理されたリポジトリオブジェクトの表示と比較に関するルールおよびガイドライン

ワークスペース内の古いバージョンを表示するときには、次の規則とガイドラインに従ってください。

- マッピングやマプレットなど、複合オブジェクトについての複数のバージョンを同時に表示することはできません。
- 複合オブジェクトの旧バージョンには、複合オブジェクトがチェックインされたときに使用していた子オブジェクトが含まれていない場合もあります。子オブジェクトのバージョンを持ち、それがリポジトリから削除されている複合オブジェクトを開く場合は、子オブジェクトのそれ以前のバージョンは、複合オブジェクトの一部としてワークスペースに表示されます。例えば、本来はソース定義のバージョン 3 が格納されているマッピングのバージョン 5 を表示するとします。ただし、ソース定義のバージョン 3 はリポジトリからパージされています。マッピングのバージョン 5 を表示すると、ソース定義のバージョン 2 がマッピングの一部として表示されます。
- 参照するオブジェクトを変更しても、ショートカットオブジェクトは更新されません。ショートカットオブジェクトを開くと、後続のバージョンが存在する場合でも、そのショートカットが本来参照していたバージョンのオブジェクトが表示されます。

旧バージョンのリポジトリオブジェクトの表示

ワークスペースに古いバージョンのオブジェクトを開くには：

1. ワークスペースまたはナビゲータで、オブジェクトを選択し、［バージョニング］-［履歴の表示］をクリックします。
2. ワークスペースに表示するバージョンを選択し、［ツール］-［ワークスペースに開く］をクリックします。

   注: オブジェクトの古いバージョンは読み取り専用で、バージョン番号がオブジェクト名の接頭語として表示されます。ワークスペースには、非複合オブジェクトの複数のバージョンを同時に表示できます。

リポジトリオブジェクトのバージョンの比較

同じオブジェクトの 2 つのバージョンを比較するには：

1. ワークスペースまたはナビゲータで、オブジェクトを選択し、［バージョニング］-［履歴の表示］をクリックします。
2. 比較するバージョンを選択し、［比較］-［選択したバージョン］をクリックします。

   オブジェクトのバージョンを以前のバージョンと比較するには、バージョンを選択し、［比較］-［以前のバージョン］をクリックします。

   ウィンドウが開き、オブジェクトの両方のバージョンでの詳細情報が表示されます。

   注: クエリーを実行した場合、クエリー結果ウィンドウから履歴の表示ウィンドウにもアクセスできます。

以前のオブジェクトバージョンに戻す

Designer でオブジェクトを編集する場合、以前に保存したバージョンに戻り、前回保存した後に行った変更を取り消すことができます。複数のオブジェクトを、以前保存したバージョンに同時に戻すことができます。

オブジェクトを以前保存したバージョンに戻すには：

1. ワークスペースでオブジェクトを開きます。
2. オブジェクトを選択し、［編集］-［保存状態に復帰］を選択します。
3. ［はい］をクリックします。複数のオブジェクトを選択した場合、［すべてはい］をクリックします。

   Designer は、前回保存した後に実行した変更をすべて削除します。

Designer オブジェクトのコピー

Designer オブジェクトは同一のフォルダ内にコピーしたり、別のフォルダや別のリポジトリにコピーしたりできます。ソース、ターゲット、マッピング、マプレット、トランスフォーメーション、次元などあらゆる Designer オブジェクトをコピーできます。オブジェクトをコピーする前に、コピー先のフォルダを開く必要があります。

オブジェクトをコピーするには、Designer のコピーウィザードを使用します。コピーウィザードは、コピー先フォルダ内の競合を検査し、競合を解決するための選択肢を提供します。例えば、ある項目が既にコピー先フォルダに存在している場合、問題の説明が画面に表示されます。コピーウィザードは使用できる解決策を表示します。重複するオブジェクトに対して、名前の変更、再利用、上書き、およびオブジェクトのコピーのスキップを実行できます。

コピーウィザードの表示および機能を設定するには、Designer で［ツール］-［オプション］をクリックします。

Designer のインポートウィザードを使用して、XML ファイルからオブジェクトをインポートできます。

インポートウィザードは、コピーウィザードと同じ競合を解決するためのオプションを提供します。

マッピングセグメントのコピー

マッピングロジックの一部を再利用したい場合、マッピングおよびマプレットのセグメントをコピーできます。セグメントはマッピングまたはマプレットの 1 つ以上のオブジェクトにより構成されます。オブジェクトは、ソース、ターゲット、ショートカット、トランスフォーメーション、およびマプレットになります。マッピングセグメントをコピーするには、マッピングからセグメントを選択およびコピーし、コピー先のマッピングに貼り付けます。マッピングまたはマプレットのセグメントを空のマッピングまたはマプレットのワークスペースに貼り付けることができます。フォルダまたはリポジトリ間でセグメントをコピーすることもできます。

マッピングまたはマプレットのセグメントをコピーするには：

1. マッピングまたはマプレットを開きます。
2. コピーする各オブジェクトを強調表示して、セグメントを選択します。

   複数のオブジェクトを選択することができます。また、ワークスペースでマウスをドラッグしてオブジェクトを矩形で囲んでもセグメントを選択できます。
3. セグメントをクリップボードにコピーします。
4. コピー先のマッピングまたはマプレットを開きます。

   必要に応じて、空のマッピングまたはマプレットのワークスペースを開くことができます。
5. ［編集］-［貼り付け］をクリックするか、Ctrl+V キーを押します。

   コピー競合が発生した場合には、コピーウィザードが開きます。

イメージとしてのオブジェクトのコピー

Designer のワークスペースで、マッピングやトランスフォーメーションといったオブジェクトのイメージをクリップボードにコピーできます。その後、イメージファイルを、グラフィックを使用できるアプリケーションのワークスペースに貼り付けることができます。

Designer ワークスペースでオブジェクトをイメージとしてコピーするには：

1. Designer で、コピーしたいオブジェクトに関連付けられたツールを開きます。

   例えば、マッピングをイメージとしてコピーしたい場合は、Mapping Designer を開きます。

2. コピーするオブジェクトを開きます。

   アイコン化されたマッピングを表示するよう選択した場合、コピーされたイメージにマッピングのリンクは含まれません。マッピングのリンクを手動で選択してコピーできます。

3. ポインタをドラッグして、選択するオブジェクトを矩形で囲みます。

   Ctrl キーを押しながらクリックしても、個々のオブジェクトを選択できます。ただし、この方法を使用してオブジェクトを選択した場合、コピーされたイメージにマッピングのリンクは含まれません。

4. Shift+Alt+C キーを押すか、［編集］-［画像としてコピー］をクリックします。

   一度につき 1 つのイメージをクリップボードにコピーできます。

5. グラフィックファイルを使用できるアプリケーションのワークスペースにイメージを貼り付けます。

   例えば、Microsoft Word 文書にイメージを貼り付けることができます。

オブジェクトのエクスポートおよびインポート

オブジェクトを XML ファイルにエクスポートするには、Designer でオブジェクトを右クリックし、［ファイルへのエクスポート］を選択します。Designer は、そのオブジェクトをリポジトリにインポートするのに必要な情報をすべて保存します。

Designer で XML ファイルからオブジェクトをインポートするには、［リポジトリ］-［オブジェクトのインポート］をクリックします。

リポジトリオブジェクトの更新

リポジトリフォルダリストまたはフォルダを更新して最新の変更を反映させることができます。フォルダを更新すると、その内容が更新されます。

フォルダを更新するには、オープンされたフォルダを右クリックしてから、［更新］を選択します。

リポジトリフォルダリストを更新するには、リポジトリを右クリックしてから、［リポジトリフォルダリスト］を選択します。

複数のポートやカラムに関する作業

すべての Designer ツールで、複数のポートやカラムを同時に移動したり、削除したりできます。

注: Source Analyzer で COBOL ソースを編集している場合には、複数のポートまたはカラムを選択することはできません。

連続したポートまたはカラムを選択するには：

1. ソース、ターゲット、またはトランスフォーメーションをダブルクリックします。
2. ［テーブルの編集］ダイアログボックスで、［ポート］または［カラム］タブを選択します。
3. トランスフォーメーション内の行のヘッダ番号を使用して最初のポートまたはカラムを選択します。
4. 移動または削除の範囲を選択するには、SHIFT キーを押しながら、その範囲内の最後のポートまたはカラムをクリックします。 個別にポートやカラムを選択するには、［Ctrl］キーを押しながら、ヘッダ番号で移動/削除対象の各ポートまたはカラムをクリックします。

   注: 複数のポートやカラムを選択していると、追加、コピー、貼り付けが無効になります。

5. ［削除］をクリックします。

ポート名の変更

ソース、ターゲット、およびトランスフォーメーションのポートの名前を変更できます。ソース、ターゲット、またはトランスフォーメーションのポートの名前を変更するには、ポートをダブルクリックして新しい名前を入力します。Designer は、このソース、ターゲット、またはトランスフォーメーションを使用するマッピングおよびマプレットに新しい名前をプロパゲートします。以下の Designer ツールでポートの名前を変更できます。

- **Source Analyzer。**ソースのポート名を変更します。
- **Target Designer。**ターゲットのポートの名前を変更します。
- **Transformation Developer。**再利用可能なトランスフォーメーションのポート名を変更します。
- **Mapplet Designer および Mapping Designer。**再利用不可能なトランスフォーメーションのポートの名前を変更します。新しい名前を、Designer を通してこのポートにアクセスする再利用不可能なトランスフォーメーションにプロパゲートできます。

ショートカットキーの使用

リポジトリオブジェクトを編集する場合、［ポート］タブまたは［カラム］タブのショートカットを使用できます。

以下の表に、Designer のショートカットを示します。

タスク	ショートカット
新しいフィールドまたはポートを追加する	Alt+F
セル内で編集を取り消す。	Esc
ポートタイプのチェックボックスを選択またはクリアする。	スペースバー
行をコピーする。	Alt+O
セルでテキストをコピーする。	Ctrl+C
行を削除する。	Alt+C
セルのテキストを編集する。	F2 キーを押して、その後、カーソルをセル内に移動する。
すべての組み合わせを検索してボックスを一覧表示する。	リストの最初の文字を入力します。
ワークスペース内でテーブルまたはフィールドを検索する。	Ctrl+F
現在の行を下に移動させる。	Alt+W
現在の行を上に移動させる。	Alt+U
［式］フィールドから式エディタを開く。	F2 キーを押してから、F3 キーを押します。
行を貼り付ける。	Alt+P
コピーしたテキストをセルに貼り付ける。	Ctrl+V

タスク	ショートカット
セルのテキストを選択する。	F2
トランスフォーメーションのデフォルト値を有効にする。	Alt+V

データのプレビュー

Designer でソースおよびターゲットデータをプレビューすることができます。マッピングを作成する前やマッピングに関する作業を行っている場合は、ソースデータまたはターゲットデータをプレビューします。

以下の種類のデータをプレビューすることができます。

- **リレーショナルテーブルおよびビュー。**リレーショナルソースおよびターゲットをプレビューします。Designer で有効なリレーショナルソース定義またはターゲット定義のデータをプレビューすることができます。ソース定義またはソース定義へのショートカットは、ソーステーブルに一致していれば有効です。ターゲット定義は、ターゲットテーブルに一致していれば有効です。リレーショナルデータをプレビューするためには、ソースまたはターゲットデータベースに接続できなければなりません。
- **固定長フラットファイルおよび区切りフラットファイル。**バイナリデータを含まないフラットファイルソースおよびターゲットをプレビューします。 有効なフラットファイルソースまたはターゲット定義のデータを Designer でプレビューすることができます。ソース定義またはターゲット定義は、ソースファイルと一致していれば有効です。
- **XML ファイル** XML エディタで XML ファイルのデータをプレビューします。XML 定義および外部 XML ファイルを使用して、XML データをプレビューできます。データをプレビューするには、リポジトリ内に有効な XML 定義があり、定義に対して有効な外部 XML ファイルにデータが入っていることが必要です。一度にデータを表示できる XML ビューは 1 つです。

  ヒント: ソースデータをプレビューして、適切なソースからデータを抽出していることを確かめます。また、ソースデータをプレビューして、マッピングで使用するカラムを特定できます。ターゲットテーブルを切り詰める前に、または更新方式を実装する際に、ターゲットデータをプレビューできます。

リレーショナルデータのプレビュー

リレーショナルソースまたはターゲットデータをプレビューするには：

1. ワークスペース内でリレーショナルソースまたはターゲット定義を選択します。
2. ワークスペース内でソースまたはターゲット定義を右クリックし、［データのプレビュー］を選択します。
3. データソース名を選択します。
4. データベースユーザー名およびパスワードを入力します。

   オブジェクトを表示するには、ユーザー名がデータベースに対して権限を持っていなければなりません。
5. データベーステーブルオーナ名を入力します。
6. プレビューしたい行の数を入力します。

   ［データのプレビュー］ダイアログボックスでは、500 行、65,000 カラムまで表示できます。
7. ［接続］をクリックします。
8. プレビューしたい行数を変更するには、新しい数を入力し、［リフレッシュ］をクリックします。
9. ［閉じる］をクリックします。

フラットファイルデータのプレビュー

フラットファイルソースデータおよびフラットファイルターゲットデータをプレビューするには：

1. ワークスペース内でフラットファイルソースまたはターゲット定義を選択します。
2. ワークスペース内でソースまたはターゲット定義を右クリックし、［データのプレビュー］を選択します。
3. フラットファイルを選択するために、［参照］をクリックします。
4. プレビューしたいファイルを探して選択し、［開く］をクリックします。
5. ［データのプレビュー］ダイアログボックスからコードページを選択します。

   コードページは、ソースのコードページと完全に一致しなければなりません。コードページがソースのコードページと一致していない場合は、データが正しく表示されないことがあります。

   Designer には、最近選択したコードページが 5 つ表示されます。その次に、残りのコードページがアルファベット順に表示されます。
6. 開始行番号およびプレビューしたい行数を入力します。

   ［データのプレビュー］ダイアログボックスでは、500 行、65,000 カラムまで表示できます。
7. ［開く］をクリックします。

   注: ［データのプレビュー］ダイアログボックスでは、シフト依存のフラットファイル内のシフト文字がピリオド（.）として表示されます。
8. プレビューしたい行の開始番号または行数を変更するには、新しい数を入力し、［リフレッシュ］をクリックします。
9. ［閉じる］をクリックします。

XML データのプレビュー

XML データをプレビューするには：

1. ワークスペースの XML 定義をダブルクリックします。

   XML エディタが表示されます。
2. XML エディタの作業慮域でビューを選択します。
3. ［XML View］-［データのプレビュー］をクリックします。
4. ブラウズして、プレビューするデータが含まれている XML ファイルを探します。

   ［XML データのプレビュー］ダイアログボックスに、選択したビューを使用した XML ファイルのデータが表示されます。
5. 別の XML ファイルを使用したい場合は、ダイアログボックスの［ファイル選択］アイコンをクリックします。
6. ［OK］をクリックします。

メタデータエクステンションに関する作業

情報を個々のリポジトリオブジェクトに関連付けることで、リポジトリに格納されているメタデータを拡張することができます。たとえば、作成したソースと共に自分の連絡情報を格納できます。アグリゲータトランスフォーメーションを作成する場合は、そのトランスフォーメーションと共にメールアドレスを格納することもできます。情報をリポジトリオブジェクトに関連付けるには、メタデータエクステンションを使用します。

リポジトリオブジェクトは、ベンダー定義のメタデータエクステンションとユーザー定義のメタデータエクステンションの両方を含むことができます。ベンダ定義メタデータエクステンションについてはその値を表示したり変更することができますが、作成、削除、または再定義することはできません。ユーザー定義メタデータエクステンションの作成、編集、削除、表示、およびその値の変更を実行できます。

Designer では、以下のオブジェクトに対してメタデータエクステンションを作成することができます。

- ソース定義
- ターゲット定義
- トランスフォーメーション
- マッピング
- マップレット

再利用可能なメタデータエクステンションか、再利用不可能なメタデータエクステンションのどちらかを作成することができます。再利用可能なメタデータエクステンションは、特定のタイプの全リポジトリオブジェクト（たとえば、すべてのソース定義、すべての式トランスフォーメーションなど）に関連付けます。再利用不可能なメタデータエクステンションは、1 つのリポジトリオブジェクト（たとえば、1 つのターゲット定義や 1 つのマッピングなど）に関連付けます。

メタデータエクステンションの作成

Designer を使用して、リポジトリオブジェクトに対するユーザー定義の再利用可能/不可能なメタデータエクステンションを作成できます。メタデータエクステンションを作成するには、メタデータエクステンションを作成したいオブジェクトを編集し、そのメタデータエクステンションを［メタデータエクステンション］タブに追加します。

複数の再利用可能なメタデータエクステンションを作成する場合は、Repository Manager を使うと作成が容易になります。

メタデータエクステンションを作成するには：

1. 該当する Designer ツールを開きます。
2. 適切なオブジェクトをワークスペースにドラッグします。
3. オブジェクトを編集するために、オブジェクトのタイトルバーをダブルクリックします。オブジェクトがマッピングまたはマプレットの場合、［マッピング］-［メタデータエクステンション］または［マプレット］-［メタデータエクステンション］をクリックします。
4. ［メタデータエクステンション］タブをクリックします。

   このタブには、ユーザー定義およびベンダー定義のメタデータエクステンションが表示されます。ユーザー定義メタデータエクステンションは、ユーザー定義メタデータエクステンションの下に表示されます。ベンダー定義メタデータエクステンションは、専用のドメインに表示されます。
5. ［追加］ボタンをクリックします。

   ユーザー定義メタデータエクステンションの下に新しい行が表示されます。

6. 以下の情報を入力します。

フィールド	説明
エクステンション名	メタデータエクステンションの名前。メタデータエクステンション名は、ドメイン内のそれぞれのオブジェクトタイプについて一意でなければなりません。 メタデータエクステンション名は下線以外の特殊文字を含むことができず、数字で開始されてはなりません。
データタイプ	数値（整数）、文字列、または論理型を選択します。
精度	文字列型メタデータエクステンションの最大長。
値	数値型メタデータエクステンションの場合、値は-2,147,483,647 - 2,147,483,647 の範囲の整数でなければなりません。 論理型メタデータエクステンションの場合は、真または偽を選択します。 文字列型メタデータエクステンションの場合は、［値］フィールドの［開く］をクリックして、2 行以上の（最大で 2,147,483,647 バイトの）値を入力します。
再利用可能	メタデータエクステンションが再利用可能かどうかを指定します。チェックマークを付けると、このメタデータエクステンションはこのタイプのすべてのオブジェクトに適用されます（再利用可能になります）。チェックマークを外すと、このメタデータエクステンションはこのオブジェクトだけに適用されます（再利用不可能になります）。 トランスフォーメーションの再利用可能なメタデータエクステンションを作成すると、このメタデータエクステンションはこのタイプのすべてのトランスフォーメーション（例えばすべてのアグリゲータトランスフォーメーションやルータトランスフォーメーション）に適用されますが、全トランスフォーメーションに適用されるわけではありません。 **注**: メタデータエクステンションを再利用可能にすると、その操作を取り消して再利用不可能なメタデータエクステンションに戻すことはできません。Designer は、操作が確定されると同時にメタデータエクステンションを再利用可能にします。
オーバーライドの取り消し	［元に戻す］がクリックされたときにメタデータエクステンションのデフォルト値をリストアします。このカラムは、メタデータエクステンションのいずれかの値が変更されたときに表示されます。
説明	メタデータエクステンションの説明。

7. ［OK］をクリックします。

メタデータエクステンションの編集

Designer を使用すると、リポジトリオブジェクトに対するユーザー定義の再利用可能/不可能なメタデータエクステンションを編集することができます。メタデータエクステンションを編集するには、リポジトリオブジェクトを編集し、次に［メタデータエクステンション］タブで設定を変更します。

編集できる内容は、メタデータエクステンションが再利用可能かどうかによって決まります。再利用不可能なメタデータエクステンションは再利用可能なメタデータエクステンションに格上げできますが、逆に再利用可能なメタデータエクステンションを再利用不可能なメタデータエクステンションに変更することはできません。

再利用可能なメタデータエクステンションの編集

編集したいメタデータエクステンションが再利用可能で編集が可能な場合は、そのメタデータエクステンションの値を変更することができますが、プロパティは変更することができません。ただし、メタデータエクステ

ンションを作成したベンダーまたはユーザーがそのメタデータエクステンションを編集可能にしていなかった場合は、メタデータエクステンションまたはその値を編集することはできません。

再利用可能なメタデータエクステンションの値を編集するには、［メタデータエクステンション］タブをクリックし、［値］フィールドを編集します。メタデータエクステンションのデフォルト値をリストアするには、［オーバーライドの取り消し］カラムの［元に戻す］をクリックします。

再利用不可能なメタデータエクステンションの編集

編集したいメタデータエクステンションが再利用不可能な場合は、メタデータエクステンションの値とプロパティを変更することができます。さらに、メタデータエクステンションを再利用可能なメタデータエクステンションに格上げすることもできます。

再利用不可能なメタデータエクステンションを編集するには、［メタデータエクステンション］タブをクリックします。［データタイプ］、［値］、［精度］、および［説明］フィールドを更新できます。

メタデータエクステンションを再利用可能にするには、［再利用可能］を選択します。メタデータエクステンションを再利用可能にすると、その操作を取り消して再利用不可能なメタデータエクステンションに戻すことはできません。Designer は、操作が確定されると同時にメタデータエクステンションを再利用可能にします。

メタデータエクステンションのデフォルト値をリストアするには、［オーバーライドの取り消し］カラムの［元に戻す］をクリックします。

メタデータエクステンションの削除

Repository Manager で、再利用可能なメタデータエクステンションを削除します。

Designer では、再利用不可能なメタデータエクステンションを削除できます。メタデータエクステンションを削除するには、まずリポジトリオブジェクトを編集し、次に［メタデータエクステンション］タブからメタデータエクステンションを削除します。

ビジネス名の使用

ソース、ターゲット、およびカラムにビジネス名を追加することができます。ビジネス名は、ソース、ターゲット、またはカラムに使用するわかりやすい名前です。ビジネス名はナビゲータの［ビジネスコンポーネント］ソースノード、およびソースとターゲットの各ノードに表示されます。ビジネス名は、ワークスペースにソースおよびターゲット定義のカラム名としても表示できます。ソース修飾子を作成して、ビジネス名をカラム名として Mapping Designer や Mapplet Designer に表示することもできます。

ソースまたはターゲットへのビジネス名の追加

ソースおよびターゲット定義にビジネス名を追加することができます。PeopleSoft、および Siebel のソース定義をインポートする場合、Designer は自動的にビジネス名をインポートします。

ソースやターゲットにビジネス名を追加するには：

1. Source Analyzer または Target Designer で、ソース定義またはターゲット定義を開きます。
2. ［テーブルの編集］ダイアログボックスで、［名前の変更］をクリックします。
3. ［ビジネス名］フィールドにビジネス名を入力します。
4. ［OK］をクリックします。

ナビゲータでビジネス名を表示するには、［ビジネス名を使ってソースを表示する/ビジネス名を使ってターゲットを表示する］オプションを有効にします。ビジネス名がナビゲータに表示され、テーブル名はかっこ内に表示されます。

ナビゲータでのビジネス名の表示

ソース定義やターゲット定義にビジネス名が存在する場合、このビジネス名をナビゲータに表示するように Designer を設定できます。ナビゲータにビジネス名を表示すると、テーブル名はかっこ内に表示されます。

例えば、テーブル名を EMPLOYEES、ビジネス名を Employee Data としてソース定義を作成した場合には、ナビゲータはビジネス名と共にかっこで囲んだテーブル名を表示します。

ナビゲータでビジネス名が表示されるようにした場合、ソースまたはターゲットへのショートカットを作成すると、Designer はショートカットに Shortcut_To_*ビジネス名*と名前を付けます。

カラム名としてのビジネス名の表示

ソース定義およびターゲット定義でカラムビジネス名を表示できます。ワークスペースでカラム名としてビジネス名を選択するように、Designer を設定することができます。ソースおよびターゲット定義について重複したカラムビジネス名を使用できます。

ソース修飾子のポート名としてのビジネス名の使用

ソースカラムのビジネス名を、ソース修飾子トランスフォーメーションでポート名として使用します。

マッピング内の既存のソース修飾子にビジネス名を追加するには、ビジネス名を表示するオプションを有効にする必要があります。次に、既存のソースを削除し、そのソースをソース修飾子とともに再インポートします。

ソースにビジネス名がない場合は、ソース修飾子にはポート名が入ります。ビジネス名にポート名として使用できない文字が含まれている場合には、Designer は各文字をアンダスコア（_）で置き換えます。ビジネス名の場合、語と語の間に通常はスペースが入りますが、ポート名にスペースを使用することはできません。例えば、Employee Data というビジネス名は Employee_Data となります。

ヒント: ビジネス名やポート名では、DD_INSERT など、の予約語は使用しないでください。

ビジネスドキュメントの使用

ビジネス文書は、リポジトリオブジェクトまたはトランスフォーメーション式に関する詳細を提供します。Designer では、リポジトリオブジェクトについて作成したビジネス文書へのリンクを作成および編集できます。この文書は、Windows 環境で、ローカルマシン、ネットワークサーバー、または社内のイントラネットやインターネットの Web サイトに格納されている必要があります。

ビジネス文書は以下のリポジトリオブジェクトについて、HTML、PDF、またはテキストフォーマットで作成できます。

- ソーステーブルおよびターゲットテーブル、ならびにテーブルインスタンス
- すべてのトランスフォーメーションおよびトランスフォーメーションインスタンス
- マップレット
- マッピング
- ビジネスコンポーネントディレクトリ

ビジネス文書にアクセスするには、以下の作業を実行する必要があります。

- Designer で文書パスを指定する。
- リポジトリオブジェクトのリンクを作成する。
- リンクをクリックして文書を表示する。

ドキュメントパスの指定

文書ファイルを格納する文書パスまたはルートを指定します。各リポジトリオブジェクトに追加するリンクで、このパスを参照します。

ローカルマシンまたはネットワークサーバーにファイルを格納する場合は、ファイルパスを使用します。社内のイントラネットまたはインターネットの Web サイトにファイルを入れておく場合は、URL を使用します。

文書パスまたはルートとして有効なフォーマットは次のとおりです。

- ローカルドライブへのファイルパス。c:\doc\informatica\\u306a ど。
- ネットワークサーバーへのファイルパス。\\server5\doc\informatica\\u306a ど。
- URL。http://www.internal.company.com/doc/など。

関連項目：

- 「一般的なオプションの設定」 (ページ 19)

ドキュメントファイルへのリンクの作成

リポジトリオブジェクト内に文書ファイルへのリンクを作成することができます。各オブジェクトの［プロパティ］または［編集］ダイアログボックスでこのリンクをクリックすると、ビジネス文書が表示されます。

ビジネス文書を参照でするには、リンクの URL またはファイルパスが有効となっていることが必要です。以下のいずれかのフォーマットを使用します。

- **ルート変数:概要**文字列「file://$docroot」を使用して、文書パスフィールドで指定した文書ルートを参照します。

  例えば文書ルートが http://internal.company.com/doc の場合、Designer は http://internal.company.com/doc/finance/vendor_help.html を指しますが、リンクの一部に file://$docroot を表示します。
- **完全なファイルパスまたはリンク。**ファイルパスの先頭に://を付けて、file://c:\doc\help\\u307e または http://internal.company.com/doc/help/などとします。 ファイルパスを有効にするには、その先頭に file://を付ける必要があります。

リンクを作成するには、オブジェクトのプロパティを編集します。編集の方法は、文書化したいオブジェクトのタイプによって異なります。

以下の表に、リポジトリオブジェクト用のリンクを作成する方法をまとめます。

リポジトリ オブジェクト	文書リンクの作成方法
テーブル/トランスフォーメーション	- ワークスペースでテーブル/トランスフォーメーションをダブルクリックして開きます。 - ［説明］ウィンドウで、文書リンクを入力します。
マッピング/マプレット	- ワークスペースでマッピング/マプレットを開きます。 - ［マッピング］-［編集］または［マプレット］-［編集］をクリックします。 - ［説明］フィールドで、文書リンクを入力します。
ビジネスコンポーネントディレクトリ	- ナビゲータでビジネスコンポーネントディレクトリを選択します。 - ［リポジトリ］-［ビジネスコンポーネント］-［プロパティの編集］をクリックします。 - ［説明］フィールドで、文書リンクを入力します。

ヒント: 各ビジネスコンポーネントにコメントを追加するには、参照する元のオブジェクトのプロパティを編集します。

ビジネスドキュメントの表示

ビジネス文書へのリンクをクリックすると、Windows によって、そのファイルを表示するための該当アプリケーションが起動されます。

マップレットレポートおよびマッピングレポートの表示（廃止）

Designer で、マッピングとマプレットの PowerCenter リポジトリレポートを表示できます。マッピングとマプレットのソース、ターゲット、ポート、トランスフォーメーションに関する詳細を確認するには、レポートを表示します。レポートを表示する際、Designer は JasperReports Server をブラウザのウィンドウで起動し、レポートを表示します。

以下のレポートを表示できます。

- マップレットコンポジットレポート
- マッピングコンポジットレポート

マップレットコンポジットレポートの表示

マプレットコンポジットレポートには、次のマプレットに関する情報が示されます。

- **すべてのオブジェクト。**マップレット内のすべてのオブジェクトに関する情報です。
- **トランスフォーメーション。**マップレットで使用されているトランスフォーメーション。

マプレットコンポジットレポートを表示するには：

1. Designer でマプレットを開きます。
2. ワークスペースを右クリックして、［マプレットレポートの表示］を選択します。

Designer によって、クライアントマシンのデフォルトブラウザで JasperReports Server が起動され、マプレットコンポジットレポートが実行されます。

マッピングコンポジットレポートの表示

PowerCenter マッピングのオブジェクトに関する詳細情報を得るには、マッピングレポートを表示します。マッピングコンポジットレポートには、マッピングの次のコンポーネントに関する情報が示されます。

- **ソースおよびターゲットフィールド。**マッピングソースで使用されるフィールドです。
- **ポート接続。**オブジェクト間のポートレベルの接続です。
- **トランスフォーメーションポート。**マッピング内の各トランスフォーメーションのトランスフォーメーションポートです。
- **オブジェクトレベル接続。**マッピング内のすべてのオブジェクト間の接続です。

マッピングコンポジットレポートを表示するには：

1. Designer でマプレットを開きます。
2. ワークスペースを右クリックして、［マプレットレポートの表示］を選択します。

   Designer によって、クライアントマシンのデフォルトブラウザで JasperReports Server が起動され、マッピングコンポジットレポートが実行されます。

第 2 章

ソースに関する作業

この章では、以下の項目について説明します。

ソースに関する作業の概要

ソースからデータを抽出するには、まずリポジトリでソースを定義します。Source Analyzer で次の種類のソース定義をインポートまたは作成できます。

- リレーショナルテーブル、ビュー、シノニム
- バイナリデータを含まない固定長および区切りフラットファイル。
- COBOL ファイル
- XML ファイル
- Web サービス記述言語（WSDL）。
- Metadata Exchange for Data Models（アドオン製品）を介して、特定のデータモデリングツールを使用するデータモデル

マルチバイト文字セットを使用するソースをインポートできます。 ソースコードページは、ターゲットコードページのスーパーセットであることが必要です。

ソース定義は単一グループまたは複数グループにできます。 単一グループソースには、ソース定義内に単一のグループがあります。 リレーショナルソースの場合は、単一グループのソース定義を使用します。複数グループソースには、ソース定義内に複数のグループがあります。 XML ソースなどの非リレーショナルソースでは、複数グループソース定義が使用されます。

注: ソース定義はソースと一致してする必要があるため、手動で作成するのではなく定義をインポートすることをお勧めします。

Oracle ソース

基本圧縮および OLTP 圧縮を使用する Oracle ソースをインポートできます。基本圧縮および OLTP 圧縮を使用する Oracle ソースのソース定義を手動で作成することもできます。

ソース定義内の特殊文字の処理

Designer では、スラッシュ（/）などの特殊文字を含むテーブル名やカラム名のソース定義をインポート、作成、編集できます。Source Analyzer を使用してソース定義をインポートする場合、Designer はテーブル名とフィールド名に特殊文字を残します。

ただし、特殊文字があるソース定義をマッピングに追加する場合、Designer は特殊文字を残すか、置き換えるかします。また、リレーショナルソースのソース修飾子トランスフォーメーションでデフォルトの SQL 文を生成する場合、Designer は一部の特殊文字を囲んでいる引用符を使用します。Designer による特殊文字の扱いは、リレーショナルソースと非リレーショナルソースで異なります。

以下の表に、Designer によるリレーショナルソースの特殊文字の扱い方を示します。

特殊文字	Source Analyzer **の動作**	Mapping Designer **の動作**
@#$_	- ソース定義テーブル名の文字を残します。 - ソース定義カラム名の文字を残します。	- ソースインスタンステーブル名の文字を残します。 - ソースインスタンスカラム名の文字を残します。 - Source Qualifier トランスフォーメーション名の文字を残します。 - Source Qualifier トランスフォーメーションポート名の文字を残します。 - SQL クエリーのテーブル名またはカラム名を囲んでいる引用符を使用しません。
/+-=~`!%^&*()[] {}';?,<>| <space>	- ソース定義テーブル名の文字を残します。 - ソース定義カラム名の文字を残します。	- ソースインスタンステーブル名の文字をアンダースコアに置き換えます。 - ソースインスタンスカラム名の文字を残します。 - Source Qualifier トランスフォーメーション名の文字をアンダースコアに置き換えます。 - Source Qualifier トランスフォーメーションポート名の文字をアンダースコアに置き換えます。 - 特殊文字を含む SQL クエリーのテーブル名とカラム名を引用符で区切ります。
.":\t\r\n	- Designer はリレーショナルソーステーブル名またはカラム名にあるこれらの文字を認識しません。	- Designer はリレーショナルソーステーブル名またはカラム名にあるこれらの文字を認識しません。
.	- Designer は ODBC ソースでピリオドを認識します。	- Designer は ODBC ソースインスタンスでピリオドを認識します。

注: Designer では、ソーステーブル名のスラッシュはアンダースコアに置き換えられますが、ソース定義のポート名で使用されているスラッシュはそのまま保持されます。

以下の表に、Designer による非リレーショナルソースの特殊文字の扱い方を示します。

特殊文字	Source Analyzer の動作	Mapping Designer の動作
@#$_	- ソース定義テーブル名の文字を残します。 - ソース定義カラム名の文字を残します。 **注**: 注：@文字はテーブル名またはカラム名の先頭文字としては使えません。	- ソースインスタンステーブル名の文字を残します。 - ソースインスタンスカラム名の文字を残します。 - Source Qualifier トランスフォーメーション名の文字を残します。 - Source Qualifier トランスフォーメーションポート名の文字を残します。 **注**: 注：@文字はテーブル名またはカラム名の先頭文字としては使えません。
/	- ソース定義テーブル名の文字を残します。 - ソース定義カラム名の文字を残します。	- ソースインスタンステーブル名の文字をアンダースコアに置き換えます。 - ソースインスタンスカラム名の文字を残します。 - Source Qualifier トランスフォーメーション名の文字をアンダースコアに置き換えます。 - Source Qualifier トランスフォーメーションポート名の文字をアンダースコアに置き換えます。
.+-=~`!%^&*()[]{}'" ;:?,<>|\t\r\n <space>	- Designer は、非リレーショナルソーステーブルの名前またはカラムの名前について、これらの文字を認識しません。	- Designer は、非リレーショナルソーステーブルの名前またはカラムの名前について、これらの文字を認識しません。

スラッシュ（/）を含むテーブル名やフィールド名を処理するために特殊な設定やコマンドが必要なデータベースもあります。詳細については、データベースのマニュアルを参照してください。

ソース定義の更新

ソース定義を更新すると、Designer はそのソースを使っているマッピングすべてに変更を反映します。ソース定義への変更内容によっては、マッピングが無効になる場合があります。

以下の表に、ターゲット定義の編集がマッピングにどのような影響を与えるかを示します。

変更	結果
カラムの追加	マッピングが無効になることはありません。
カラムのデータタイプの変更	マッピングが無効となることがあります。新しいデータタイプと一致しないデータタイプを使用する入力ポートにカラムを接続した場合は、マッピングは無効となります。
カラム名の変更	マッピングが無効となることがあります。カラムを追加した直後にそのカラム名を変更した場合は、マッピングは有効のままです。既存のカラムのカラム名を変更した場合は、マッピングは無効になります。
カラムの削除	削除したカラムの値をマッピングが使用すると、マッピングが無効となることがあります。

Source Analyzer でソースに新しいカラムを追加しても、そのソース定義を使用するすべてのマッピングは有効のままです。しかし新しいカラムを追加して、特定のプロパティをいくつか変更した場合は、そのソース定義を使用するマッピングは無効になります。

新しく追加したソースカラムで、次のプロパティについては変更してもマッピングは無効になりません。

- 名前
- データタイプ
- フォーマット
- 使用方法
- Redefines
- OCCURS
- キータイプ

変更したためにマッピングが無効となった場合には、マッピングを開いて編集をしなければなりません。その後で、[リポジトリ] - [保存] をクリックし、変更項目をリポジトリに保存してください。無効化されたマッピングが使用されているセッションがあれば、セッションを検証する必要があります。

セッションの作成

セッションを作成する場合、ソース定義のインポート時と異なるソースの場所を指定できます。ソースがファイルの場合は、セッションを作成するときにファイルプロパティの一部を上書きできます。

リレーショナルソースに関する作業

以下の方法で、リレーショナルソース定義を追加、管理できます。

- **ソース定義のインポート。** Source Analyzer にソース定義をインポートします。
- **ソース定義の更新。** 手動または定義の再インポートにより、ソース定義を更新します。

リレーショナルソース定義をインポートするには、ソースデータベースと PowerCenter クライアント間の接続を設定する必要があります。

リレーショナルソース定義

データベーステーブル、ビュー、およびシノニムからリレーショナルソース定義のインポートが行えます。ソース定義をインポートすると、以下のソースメタデータがインポートされます。

- ソース名
- データベースの場所
- カラム名
- データタイプ
- キー制約

注: シノニムからソース定義をインポートすると、定義内の制約を手作業で定義しなければならない場合があります。

リレーショナルソース定義をインポートした後で、必要に応じてテーブルおよびカラムに対してビジネス名を入力できます。また、手作業でキー関係を定義することもでき、この関係はデータベースに存在しないリポジトリで作成した論理的関係でもかまいません。

リレーショナルソースのための接続

ソース定義をインポートするには、正しく設定された ODBC データソースまたはゲートウェイを使用して、クライアントマシンからソースデータベースに接続できる必要があります。また、データベースオブジェクトについて読み込み権限が必要な場合もあります。

ODBC データソースを作成する際には、ODBC Driver Manager からデータベース呼び出しが送られるドライバも指定する必要があります。以下の表に、各データベースで使用する推奨 ODBC ドライバを示します。

データベース	ODBC ドライバ	必須のデータベースクライアントソフトウェア
Greenplum	DataDirect ODBC Wire Protocol driver	×
IBM DB2	DataDirect ODBC Wire Protocol driver	×
Informix	DataDirect ODBC Wire Protocol driver	×
Microsoft Access	Microsoft Access ドライバ	○
Microsoft Excel	Microsoft Excel ドライバ	○
Microsoft SQL Server	DataDirect SQL Server Wire Protocol ODBC ドライバ	×
Netezza	Netezza ODBC ドライバ	はい
Oracle	DataDirect ODBC Wire Protocol driver	×
Sybase ASE	DataDirect ODBC Wire Protocol driver	×
SAP HANA	SAP HANA ODBC ドライバ	○
Teradata	Teradata ODBC ドライバ	はい
Vertica	Vertica ODBC ドライバ	はい

サードパーティの ODBC データソースを使用してソース定義をインポートすると、サードパーティのドライバが powermart.ini のリストにないというメッセージが Designer に表示される場合があります。Designer は、PowerCenter に付属のドライバを使用してソース定義メタデータのインポートを試みます。メタデータをインポートするサードパーティ製のドライバがある場合は、powrmart.ini を設定します。

例えば、Vendor A が vendoraodbc.dll という名前のドライバを提供している場合、ODBCDLL セクションに指定のデータベースに基づいてエントリを追加します。

```
Vendor A = pmodbc.dll
Vendor A = extodbc.dll
```

この例では、Designer は ODBC データソースを使用するために、システム ODBC ドライバと直接対話します。システム ODBC ドライバは、サードパーティの ODBC ドライバ vendoraodbc.dll と内部的に対話します。

次の表に、PMODBC.ini システム ODBC ドライバで使用するエントリ例をデータベース別に示します。

データベース	エントリ
MySQL	MYSQL = PMODBC.DLL
PowerExchange	PWX = PMODBC.DLL
dBase 4	dBASE IV = PMODBC.DLL
Visual FoxPro	Visual FoxPro = PMODBC.DLL
Sybase IQ	Adaptive Server IQ = PMODBC.DLL
Lotus Notes	Lotus Notes = PMODBC.DLL

サードパーティの ODBC データソースの設定

PowerCenter は、サードパーティの ODBC ドライバの認識、サードパーティの ODBC ドライバでの ODBC データベースオブジェクトのインポート、および Designer でのデータのプレビューに powrmart.ini ファイルを使用します。powrmart.ini ファイルに含まれていない ODBC ドライバでソース定義をインポートするには、PowerCenter クライアントマシン上でこのファイルを設定する必要があります。

1. 次のディレクトリから powrmart.ini を開きます。

   <Informatica installation directory>\clients\PowerCenterClient\client\bin

2. このファイルの ODBCDLL セクションにエントリを追加し、ODBC データソースの名前を指定します。

   このエントリは、pmodbc.dll または extodbc.dll を直接指している必要があります。

3. powrmart.ini を保存して閉じます。
4. PowerCenter クライアントを再起動して、ソース定義をインポートします。

リレーショナルソース定義のインポート

リレーショナルソース定義を作成するには、Source Analyzer を使用してソースメタデータをインポートします。

リレーショナルソース定義をインポートするには：

1. Source Analyzer で、**[ソース]** > **[データベースからインポート]** をクリックします。
2. ソースデータベースへの接続に使用する ODBC データソースを選択します。

   ODBC データソースを作成または変更する必要がある場合は、**[参照]** をクリックして ODBC データソースアドミニストレータを開きます。データソースを作成し、**[OK]** をクリックします。新しい ODBC データソースを選択します。

3. データベースに接続するためにデータベースのユーザ名とパスワードを入力してください。

   注: オブジェクトを表示するには、該当するデータベース権限が必要です。

   ソースとして使用したいデータベースオブジェクトについて、オーナー名の指定が必要な場合もあります。

4. 必要に応じて、検索フィールドを使用し、表示するテーブルの数を制限します。

   注: テーブルに対し、大文字子文字を区別した検索を有効にするには、検索文字列を二重引用符で囲みます。

5. **[接続]** をクリックします。

テーブル名が 1 つも表示されない場合やインポートしたいテーブルが表示されない場合には、**[すべて]** をクリックします。

6. ソースリストをスクロールして、インポートしたいソースを探します。インポートしたいリレーショナルオブジェクトを 1 つ以上選択します。

   Shift キーを押しながら選択すると、1 つのフォルダー内で連続したソースを選択できます。また Ctrl キーを押しながら選択すると、フォルダー内で連続しない複数のソースを選択できます。また、フォルダーを選択し、**[すべて選択]** をクリックすることで、そのフォルダー内のすべてのテーブルを選択できます。**[選択の解除]** ボタンを使用して、選択され反転表示されたテーブルの選択をすべて解除することもできます。

7. **[OK]** をクリックします。

   ソース定義が **Source Analyzer** に表示されます。**ナビゲータ**では、アクティブなリポジトリフォルダーの**ソースノード**において、ソースデータベース名の下に新しいソース定義が表示されます。

SAP HANA データベースソースの使用

SAP HANA ソースからデータの読み取りおよび SAP HANA へデータの書き込みを行うには SAP HANA ライセンスが必要です。

SAP HANA は ODBC 接続を使用します。SAP テーブルおよび SAP HANA データベースモデリングビューからデータを読み取りできます。

次のタイプの SAP HANA データベースモデリングビューから読み取りできます。

- 分析ビュー
- 属性ビュー
- 計算ビュー

リレーショナルソース定義の更新

ソース定義を更新することで、ビジネス名を追加したり、新しいカラム名やデータタイプなどの変更を反映させたりできます。次の方法でソース定義を更新できます。

- インポートできないプロパティを設定する必要がある場合、またはソース定義に小さな変更を加えたい場合に、ソース定義を手動で編集します。
- **定義の再インポート。**ソースの変更が著しい場合は、ソース定義の再インポートが必要になることがあります。 これにより、既存のソース定義が上書きされるか名前が変更されます。上書き対象のソース定義内の既存のプライマリキーと外部キーの関係や説明を維持することができます。

ソース定義を更新すると、Designer はそのソースを使っているマッピングすべてに変更を反映します。ソース定義への変更内容によっては、マッピングが無効になる場合があります。

変更したためにマッピングが無効となった場合には、マッピングを開いて編集をしなければなりません。その後で、[リポジトリ] - [保存] をクリックし、変更項目をリポジトリに保存してください。無効化されたマッピングが使用されているセッションがあれば、セッションを検査する必要があります。

リレーショナルソース定義の編集

ソースからインポートできないプロパティを記録するために、ソース定義を手動で編集しなければならない場合があります。リレーショナルソース定義を編集して、キーカラムやキー関係の作成ができます。この関係は論理的関係であってもかまいません。関係はデータベースに存在する必要はありません。

ソース定義にはいつでも説明を追加したり、ビジネス文書へのリンクを指定したりできます。ソース定義に説明やビジネス文書へのリンクを追加することは、ソース定義の目的を文書化する簡便な方法です。既に作成したソース定義に説明を追加したり、説明を修正することができます。

リレーショナルソース定義の編集を行うには：

1. Source Analyzer で、該当するソース定義のタイトルバーをダブルクリックします。
2. 次の設定を編集します。

テーブル設定項目	説明
テーブルの選択	編集中のソース定義を表示します。開いている別のソース定義を選択するには、リストから選びます。
［名前の変更］ボタン	ダイアログボックスを開いて、ソース定義名の編集とビジネス名の入力を行います。
オーナ名	データベース内のテーブルオーナ。
説明	ソーステーブルの補足説明。リポジトリのコードページにおける各文字の最大バイト数を K とした場合、文字の制限は 2000 バイト/K 文字です。ビジネスドキュメントへのリンクを入力します。
データベースタイプ	ソースやデータベースタイプを示します。必要に応じて、新しいデータベースタイプを選択してください。

3. ［カラム］タブをクリックします。
4. 次の設定を編集します。

カラム設定	説明
カラム名	ソース内のカラムの名前。リレーショナルソース定義を編集するとき、実際のソースカラム名が変更された場合にのみ、カラム名を編集します。
データタイプ	ソース定義に表示されるデータタイプは、ソース定義のソースタイプによって異なります。
精度と位取り	*精度*は、数値データタイプでは最大有効桁数であり、また文字列データタイプでは最大文字数です。精度には位取りが含まれます。*位取りは、数値の小数点以下の最大桁数です。*したがって、11.47 という値の精度は 4 で、位取りは 2 です。「*Informatica*」という文字列の精度（長さ）は 11 です。 リレーショナルソースのデータタイプにはすべて、最大の精度が決められています。例えば、Integer データタイプの最大精度は 10 です。いくつかの数値データタイプには位取りについても同様の制限があって、0 より大きな値に位取りを設定できません。例えば整数は位取りが 0 ですが、それは定義上、小数点以下の値を含まないからです。 一部のデータタイプの精度および位取りは、データベースで定義されている値とは異なる値に変更できます。ただし、精度と位取りを変更すると、がソースカラムを読み込む際、数値カラムで数値データのオーバーフローが起きたり、文字カラムで文字列が切り詰められたり、日付カラムでゼロが挿入されたりすることもあります。
非 Null	ソースで NULL データを許可するかどうかを選択します。
キータイプ	［プライマリキー］、［外部キー］、［プライマリ／外部キー］、または［キー以外］を選択します。リレーショナルソースのみに適用されます。
ビジネス名	必要に応じて、各ソースカラムにビジネス名を追加できます。

5. ［OK］をクリックします。

ソース定義の再インポート

ソース定義を再インポートするには、以下の手順を実行します。上書き対象のソース定義内にある次の情報を維持できます。

- プライマリキーと外部キーの関係
- ソース定義の説明
- カラムまたはポートの説明

リレーショナルソース定義を再インポートするには：

1. Designer で、更新するソース定義を含むリポジトリに接続します。
2. Source Analyzer を開き、ソース定義を再インポートします。
3. 既存のソーステーブルの名前の変更、または既存のソーステーブルの上書きを行うよう促すメッセージが表示されます。
4. インポートするテーブルと既存のテーブルとの違いを表示するには、［比較］をクリックします。

   ダイアログボックスに、それぞれのソースの属性が左右に並べて表示されます。
5. 既存のソース定義内のプライマリキーと外部キーの情報、またはソースの説明を維持するかどうかを指定します。

   以下の表に、［テーブル名の重複］ダイアログボックスのオプションを示します。

オプション	説明
すべてのテーブルに適用	フォルダ内のすべてのテーブルに名前の変更、上書き、またはスキップを適用します。
ユーザー定義のプライマリキー - 外部キーの関係を保持する	上書き対象のソース定義内にあるプライマリキーと外部キーの関係を維持します。
ユーザー定義の説明を保持する	上書き対象のソース定義のソースの説明およびカラムやポートの説明を維持します。

6. ［上書き］、［名前の変更］、または［スキップ］をクリックします。

オプション	説明
置換	ソース定義を新しいソース定義で上書きします。
名前の変更	新しいソース定義に対して異なる名前を入力します。
スキップ	新しいソース定義のインポートを行いません。

7. ［名前の変更］をクリックした場合には、ソース定義名を入力し、［OK］をクリックします。

ターゲット定義からのソース定義の作成

ターゲット定義から、またはターゲット定義へのショートカットからソース定義を作成できます。

リレーショナルおよびフラットファイルターゲット定義を Source Analyzer にドラッグしてソース定義を作成します。Designer では、ターゲット定義コードページを一致するソース定義に使用します。

次のタイプのターゲット定義からソース定義を作成できます。

- フラットファイル
- IBM DB2
- Informix
- Microsoft SQL Server
- MQ Series
- Netezza
- ODBC
- Oracle
- Sybase
- Teradata
- Vertica

データをステージングしていて、その定義を、1 つのマッピングのターゲットと別のマッピングのソースに使用する必要がある場合、ターゲット定義からソース定義を作成すると便利です。

COBOL ソースに関する作業

メインフレームソースデータをサポートするため、Designer で COBOL ファイルをソース定義としてインポートできます。COBOL ファイルは、テキストやバイナリデータが含まれる固定長ファイルです。では、COBOL ファイル用に以下のコードページがサポートされています。

- 7 ビット ASCII
- EBCDIC-US
- 8 ビット ASCII
- 8 ビット EBCDIC
- ASCII ベース MBCS
- EBCDIC ベース MBCS

シフトキーを含まないシフト依存の COBOL ファイルをインポートできます。COBOL ソース定義の各カラムに対してシフト状態を定義します。

COBOL ソースは、データを非正規化し、別のテーブルレコードに相当するものを単一のレコードにまとめている場合があります。これらのレコードは、ノーマライザトランスフォーメーションを使用してマッピング内で正規化できます。

COBOL ソース定義をインポートした後で、COBOL ファイルを確認および設定してレコードグループを作成します。COBOL ファイルは、同一のレコードセット内で複数のソーステーブルと機能的に同等なものを表すことがあります。COBOL ファイルの構造を確認する際に、記述を修正して、単一の擬似テーブルを構成する個々のフィールドのグループをそれぞれ特定することができます。

COBOL ソースのインポート

COBOL プログラムの Data Division に格納したデータ構造を用いて、ソース定義を作成できます。COBOL ファイルをインポートすると、Designer は標準の ANSI フォーマットとは異なる固有の COBOL ファイルフォーマットを探します。

Designer は、以下の例に似た COBOL ファイルフォーマットを探します。

```
identification division.
program-id. mead.
environment division.
    select file-one assign to "fname".
data division.
file section.
fd  FILE-ONE.
 01  SCHOOL-REC.
    02  SCHOOL-ID               PIC 9(15).
    02  SCHOOL-NM               PIC X(25).
    02  CLASS-REC               OCCURS 2 TIMES.
        03  CLASS-ID            PIC 9(5).
        03  CLASS-NM            PIC X(25).
        03  STUDENT-REC       OCCURS 5 TIMES.
            04  STUDENT-ID      PIC 9(15).
            04  STUDENT-NM      PIC X(25).
            04  PARENT-REC    OCCURS 2 TIMES.
                05 PARENT-ID    PIC 9(15).
                05 PARENT-NM    PIC X(25).
         03  TEACHER-REC       OCCURS 3 TIMES.
            04  TEACHER-ID      PIC 9(15).
            04  TEACHER-NM      PIC X(25).
      02  SPORT-REC             OCCURS 2 TIMES.
         03  SPORT-TEAM         PIC X(30).
working-storage section.
procedure division.
    stop run.
```

COBOL コピーブックに関する作業

Designer は、COBOL コピーブック（.cpy ファイル）を COBOL ファイル（.cbl ファイル）として認識しません。フォーマットが適切ではないためです。Designer で COBOL コピーブックをインポートするためには、COBOL 文の「copy」を使用して COBOL ファイルテンプレートに挿入します。コピーブックファイルを COBOL ファイルテンプレートに挿入したら、.cbl ファイルとして保存して Designer にインポートすることができます。

.cbl ファイルおよび.cpy ファイルが同じローカルディレクトリにない場合は、.cpy ファイルの場所の入力が求められます。

COBOL コピーブックファイルにタブが含まれていると、Designer によりタブがスペースに展開されます。デフォルトでは、Designer はタブ文字を 8 個のスペースに展開します。このデフォルト設定は powrmart.ini で変更できます。powrmart.ini は、PowerCenter クライアントのインストールディレクトリのルートディレクトリに置かれています。

デフォルト設定を変更するには、powrmart.ini に次のテキストを追加します。

```
[AnalyzerOptions]
TabSize=n
```

n は、各タブ文字について Designer が読み出すスペースの数です。 変更を適用するには、Designer を再起動します。

たとえば、COBOL コピーブックファイルは sample.cpy と呼ばれます。以下の COBOL ファイルに、コピー文を使ってサンプルコピーブックを COBOL ファイルテンプレートに挿入する方法を示します。

```
identification division.
program-id. mead.
environment division.
    select file-one assign to "fname".
data division.
file section.
fd  FILE-ONE.
    copy “sample.cpy”.
working-storage section.
procedure division.
    stop run.
```

COBOL ソース構造体のインポート手順

COBOL ソース構造体をインポートするには、以下の手順を実行します。

1. Source Analyzer を開き、［ソース］-［COBOL ファイルからのインポート］をクリックします。
2. 分析したい COBOL ファイルを選択します。
3. COBOL ファイルのコードページを選択します。

   これはデータファイルではなく、COBOL ファイル（.cbl）のコードページです。コードページには、PowerCenter クライアントのコードページとの互換性が必要です。

   このコードページを選択すると、データファイルはデフォルトでこのコードページを使用します。COBOL ファイルをインポートしたら、ソース定義を調整する際、またはワークフローを実行する際に、ソースデータのコードページを設定することができます。

   Designer には、最近選択したコードページが 5 つ表示されます。その次に、残りのコードページがアルファベット順に表示されます。
4. ［OK］をクリックします。

COBOL ソース定義が Designer に表示されます。COBOL ファイルに FD セクションの項目が複数ある場合は、複数の定義が表示されることがあります。

COBOL ソースファイルのコンポーネント

COBOL ソースをインポートすると、Designer はファイルのスキャンを行って、以下の構成要素について調査します。

- FD セクション
- フィールド
- OCCURS
- REDEFINES

FD セクション

Designer は、FD セクションの各項目がリレーショナルソースのソーステーブルと同等なものを定義していると仮定し、各項目について別の COBOL ソース定義を作成します。例えば、COBOL ファイルに CUSTOMERS と ORDERS という 2 つの項目がある場合、Designer は CUSTOMERS という属性のフィールドを含む 1 つの COBOL ソース定義と、ORDERS という属性のフィールドを含む別の COBOL ソース定義を作成します。

フィールド

Designer はそれぞれのフィールド定義を識別し、そのデータタイプを読み込み、該当するソース定義にそのデータタイプを割り当てます。

OCCURS

COBOL ファイルでは、同一レコード内に同一種別の複数のインスタンスが含まれている場合があります。例えば、COBOL ファイルに 4 つの財務四半期についてのデータが含まれていて、それぞれが同じレコードに格納されている場合があります。Designer がこのファイルを分析する場合、COBOL ファイルのそれぞれの OCCURS 文について個別のカラムを作成します。この OCCURS 文は、同一レコード内の繰り返し情報を定義します。ノーマライザトランスフォーメーションを使用して、この情報を正規化します。

各 OCCURS 文に対して、Designer は次の項目を作成します。

- COBOL ソース定義を Target Designer にドラッグした場合は、1 つのターゲットテーブル。
- プライマリキーと外部キーの関係
- 生成カラム ID（GCID）

REDEFINES

COBOL は REDEFINES 文を使用し、あるレコードの記述を別のレコードの定義に基づいて作成します。COBOL ソースのインポートを行うと、Designer は複数の REDEFINES を含む単一のソースを作成します。

REDEFINES 文により、物理データの場所のサンプルに複数の PICTURE 句を指定できます。したがって、フィルタトランスフォーメーションを使用し、REDEFINES で作成した複数のテーブルにデータをパーティション化する必要があります。

各 REDEFINES について次の作業が行われます。

- COBOL ソース定義を Target Designer にドラッグした場合、Designer は 1 つのターゲットテーブルを作成します。
- Designer はプライマリー外部キーの関係を 1 つ作成します。
- Designer は、生成キー（GK）を作成します。
- マッピングではそれぞれ別のフィルタトランスフォーメーションが必要です。

COBOL ソース定義の設定

COBOL ソース定義をインポートした後で、ソースプロパティをいくつか設定しなければならない場合があります。COBOL ソース定義は、固定長フラットファイル定義に似ています。ただし、COBOL ファイルには、定義を設定する際に考慮する固有のプロパティがいくつかあります。

- OCCURS
- フィールド定義
- ワードまたはバイトの格納
- フィールドの属性

COBOL ソース定義を設定するときは、次のタブおよびダイアログボックスを確認してください。

- **［テーブル］タブ。**格納内容を確認します。
- **詳細プロパティ。**固定長データファイルのプロパティを確認します。
- **［カラム］タブ。**OCCURS、FD セクション、およびフィールド属性を確認します。

［テーブル］タブの設定

フラットファイル定義の設定と類似した方法で、COBOL ソース定義の［テーブル］タブを設定します。ただし、COBOL 定義では格納の種別を考慮する必要があります。

は、IEEE 754 4 バイトフォーマットの浮動小数点を持つ、ネットワークバイトオーダーでの COMP-1 ワード単位の格納をサポートしています。また、は IEEE 754 8 バイトフォーマットの浮動小数点を持つ、ネットワークバイトオーダーでの COMP-2 ワード単位の格納もサポートしています。

ソース定義で IBM 互換オプションを選択することによって、IBM VS COBOL および MicroFocus COBOL 関連のデータファイルについてバイト単位の格納に切り替えなければならない場合があります。Integration Service により、デフォルトで COMP ワード単位の格納が行われます。 COMP カラムは IBM メインフレームで 2、4、および 8 バイトです。MicroFocus COBOL を通して得られた COMP カラムは、1、2、3、4、5、6、7、8 バイトとなることがあります。バイト格納を使用する場合は、IBM 互換オプションの選択を解除します。

以下の表に、［テーブル］タブに設定可能な COBOL ファイルプロパティを示します。

テーブルオプション	説明
［名前の変更］ボタン	ソース定義名を変更してビジネス名を入力するには、［名前の変更］ボタンを使用します。
オーナ名	COBOL ファイルには適用されません。
説明	ソース定義についての付加的なコメント。
データベースタイプ	ソースの場所または種別。これは「VSAM」に設定する必要があります。
IBM COMP	格納の種別を示します。選択すると、単語単位の格納を行います。選択しなければ、バイト単位の格納を行います。
フラットファイルタイプ	固定長を選択します。
［詳細設定］ボタン	固定長オプションについてのダイアログボックスを開くには［詳細設定］ボタンを使用します。

詳細プロパティの設定

［テーブル］タブで［詳細設定］をクリックし、固定長ファイルのプロパティを設定します。

COBOL ファイルをインポートする際に、Designer がファイルを正しく読み込むことができるような COBOL ファイルのコードページを選択します。COBOL ファイルをインポートした後で、ワークフローの実行時にがデータを読み込むことができるように、ソースデータのコードページを変更することができます。データファイルのコードページは、［詳細設定］プロパティで選択できます。

関連項目：

- 「固定長ファイルのプロパティの更新」 (ページ 77)

［カラム］タブの設定

COBOL ソース定義の［カラム］タブを見ると、カラムに対する複数の属性に加えて、数種類のカラムのレベルが表示されています。次のプロパティを確認および設定することができます。

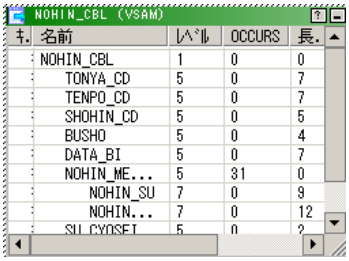
OCCURS

COBOL ソースの［カラム］タブの内容を見ると、数種類のカラムレベルが表示されています。これらのレベルは、単一の COBOL ソース内に含まれるさまざまなレコードセットに対応しています。

例えば、以下の COBOL ソースにはネストされたレコードセット HST_MTH が含まれています。それぞれのレコードセットはレベル 5 の見出しから始まり、レコードの開始を示します。各レコードセット内のカラムは、レコード見出しの下ですべて同一のレベルでなければなりません。例えば、レコードセット HST_MTH には、

HST_ACCR_REM から始まるいくつかのカラムが含まれています。24 という OCCURS の設定は、この COBOL ソースのデータを見たときに、各レコードに、HST_MTH についてネストされたレコードが 24 個含まれていることを示します。

以下の図に、OCCURS を 4 に設定した COBOL ソース定義例を示します。



この COBOL ソースでは、HST_MTH 内のカラムはすべて同じレベル 7 です。レコード HST_MTH の見出しはレベル 5 で、このソースのカラムより 2 レベル上です。

FD セクション

場合により、フィールドをグループ化するようにソース定義を設定する必要があります。Designer は COBOL ファイルの FD セクションの各項目について個別のソース定義を作成しますが、この各項目は、それぞれ複数のデータテーブルと機能的に等価な内容を表しています。ソース定義の設定を行う際に、このソース内にさまざまなフィールドレベルを作成し、データを別々の擬似テーブルへとグループ化します。

フィールドの属性

COBOL ソースの内容を見ると、各フィールド（COBOL ではカラムに該当する）には複数の属性があることがわかります。これらの属性は、COBOL ファイル内のフィールドを設定する方法を表しています。

これらの属性のうち、PICTURE 句は最も基本的なもので、COBOL ファイルでデータがどのように表現されているかを示しています。COBOL では固有の規則に従って、カラム内でのデータの書式設定を行います。例えば、ピクチャ X（32）は当該フィールドのテキストデータの長さが 32 バイトであることを示します。またピクチャ 9（7）は、当該フィールドに 7 桁以下の数値データが含まれていることを示します。2 バイト文字を含む Nstring データタイプのピクチャ N（8）は、フィールドのテキストデータが 16 バイト長であることを示します。

Source Analyzer でフィールドの定義を修正し、プロセスのピクチャを変更することが必要な場合があります。はソース定義をソースファイルでデータを探すためのマップとして使っているので、そのような調整を行うときには注意が必要です。

以下の表に、COBOL ソース定義の［カラム］タブで設定可能な属性を示します。

属性	説明
オフセット	ファイル内のフィールドのオフセット。Designer はフィールドの物理長、ピクチャ、使用法、REDEFINES の各設定項目を用いて、この設定値（表示のみ）を計算します。
長さ	このフィールドのバイト数。
カラム名	フィールドの名前。

属性	説明
レベル	同じレコードについてのデータを提供するすべてのフィールドの識別に使うインジケータ。フィールドのグループ分けをする場合には、そのカラムすべてを同じレベルに設定します。この機能を使えば、同一の COBOL ソースから複数のレコード種別（別々のデータテーブルに相当）を作成することができます。
OCCURS	このフィールドの複数のインスタンスが同一レコード内に現われることを示す COBOL 文。
データタイプ	フィールドのデータタイプ：String、Nstring、Numeric
精度	フィールド内の数値の精度。
位取り	フィールド内の数値の位取り。
PICTURE	ファイルでのデータの表現方法。
使用方法	フィールド内のデータの格納フォーマット。適用可能な規則は、すべて各フィールドの用途のリストに表示されます。COMP-1 や COMP-X などのさまざまな COBOL 規則があります。
キータイプ	このフィールドに適用されるキー制約の種類。フィールドをプライマリキーとして設定すると、COBOL ファイルをソースとして使用してワークフローを実行するときに、Integration Service は当該フィールドに一意な数値 ID を生成します。
S（符号付き）	フィールドの数値が符号付きであるかどうかを示します。
T（末尾の符号）	この属性が選択されている場合、フィールドの最後の桁に符号（+または-）があることを示します。選択されていない場合は、符号はフィールドの最初の文字です。
I（含まれる符号）	フィールドに表示される値に符号を含めるかどうかを示します。
R（実数小数点）	数値について、小数点がピリオド（.）であるか V 文字であるかを指定します。
Redefines	フィールドで COBOL の REDEFINES 文が使用され、そのフィールドの定義が別のフィールドの定義に基づいていることを示します。
シフトキー	シフトキーを含まないシフト依存の COBOL ファイルに、シフト状態を定義できます。 この属性は、固定長ファイルの［フラットファイル属性の編集］ダイアログボックスで、［ユーザー定義シフトステート］を選択した場合に表示されます。 カラムに 1 バイト文字が含まれる場合には、［Shift-In］を選択します。カラムにマルチバイト文字が含まれる場合には、［Shift-Out］を選択します。
ビジネス名	フィールドについての付加的なコメント。

関連項目：

- 「シフト依存のフラットファイルに関する作業」 (ページ 90)

Microsoft Excel のソース定義のインポート

では、Microsoft Excel ソースをフラットファイルとしてではなく、リレーショナルデータベースとして扱います。リレーショナルソースと同様、Designer は ODBC を用いて Microsoft Excel ソースのインポートを行います。Microsoft Excel ソースのインポートにあたって、データベースに対する権限は必要ありません。

Excel ソース定義をインポートする前に、以下の作業を行います。

1. システムに、Microsoft Excel ODBC ドライバをインストールします。
2. ODBC Data Source Administrator 内の各ソースファイルに対して、Microsoft Excel ODBC データソースを作成します。
3. 範囲を定義し、数値データカラムの書式設定を行って、Microsoft Excel スプレッドシートを作成します。

範囲の定義

Designer は、Microsoft Excel で定義した範囲に基づいてソース定義を作成します。Microsoft Excel シートで 1 つ以上の範囲を定義できます。シートが複数ある場合には、各シートについて 1 つ以上の範囲を定義しなければなりません。Designer でソースのインポートを行う場合には、それぞれの範囲はリレーショナルソースとして表示されます。

Excel ソース定義をインポートするには、Designer で範囲を定義する必要があります。

範囲を定義するには：

1. Microsoft Excel ファイルを開きます。
2. インポートするデータカラム（1 つまたは複数）を反転表示させます。
3. ［挿入］-［名前］-［定義］を選択します。
4. 選択範囲の名前を入力し、［OK］をクリックします。
5. シートが複数ある場合には、各シートを選択して手順 2 から 4 を繰り返し、各データセットの範囲を定義します。
6. ［ファイル］-［保存］をクリックします。

数値データのカラムのフォーマット

Microsoft Excel では、データのカラムにデータタイプを割り当てることができます。Microsoft Excel のデータタイプは、ODBC データタイプです。では、ODBC データタイプがサポートされており、トランスフォーメーションデータタイプに変換されます。Microsoft Excel でデータタイプを割り当てなければ、Designer は各カラムを VARCHAR としてインポートします。マッピングで数値計算や集計を行う場合は、スプレッドシートをインポートする前に、Microsoft Excel で数値データタイプを割り当てます。

Microsoft Excel でカラムの書式を設定するには：

1. Microsoft Excel ファイルを開きます。
2. 数値データからなるデータカラムを選択します。
3. ［書式］-［セル］をクリックします。
4. ［表示フォーマット］タブで、［数値］を選択します。

5. 小数点以下の桁数を指定。
6. ［OK］をクリックします。
7. ［ファイル］-［保存］をクリックします。

Microsoft Excel のソース定義のインポート手順

範囲を定義してセルの書式設定を行うと、Designer でこの範囲をインポートできます。ソースのインポートを行うと、範囲はソース定義として表示されます。

Microsoft Excel ソース定義をインポートするには：

1. Designer でリポジトリに接続し、該当するソース定義のフォルダを開きます。
2. Source Analyzer を開き、［ソース］-［データベースからインポート］をクリックします。
3. Excel ファイル（Microsoft Excel Driver（*.xls））フォーマットのデータソースを選択します。
4. ［参照］をクリックし、ODBC データソースを開きます。
5. データソースを作成した場所に応じて、［ユーザー DSN］または［システム DSN］タブで、Microsoft Excel ドライバをダブルクリックします。
6. ［ブックの選択］をクリックし、Microsoft Excel ファイルをリレーショナルデータベースと考えてブラウズします。
7. ［OK］を 3 回クリックして、［テーブルのインポート］ダイアログボックスに戻ります。
8. ［テーブルのインポート］ダイアログボックスでえ［接続］をクリックします。

   データベースのユーザー名やパスワードの入力は必要ありません。Microsoft Excel ファイルで定義した範囲がテーブル名として表示されます。データベースのユーザー名の入力は必要ないため、データベースオーナは［Owner がありません］です。
9. インポートするテーブルを選択します。

   複数のテーブルを選択するには、Ctrl または Shift キーを押しながら、複数のテーブルを反転表示させます。
10. ［OK］をクリックします。

    ナビゲータでは、［ソース］ノードのデータベース名の下にソース定義が表示されます。

ソース定義の手動作成

ソース定義を手動で作成することができます。

ソース定義を作成するには：

1. Source Analyzer で［ソース］-［作成］をクリックします。
2. ソースの名前とデータベースの名前、データベースのタイプを入力します。
3. ［作成］をクリックします。

   空のテーブル構造がワークスペースに表示されます。（これはダイアログボックスに表示されることもあります。）また、新しいソーステーブルがナビゲータウィンドウ内に表示されます。
4. ソース定義の作成が終了したら、［完了］をクリックします。
5. ソース定義を設定します。

ソース定義がリポジトリに保存されます。これでソース定義がマッピングで使用できるようになります。また、この定義に基づくソーステーブルをソースデータベースに作成できます。

ソースのトラブルシューティング

DB2 データベースからソースをインポートしたら、DB2 オペレーティングシステムによって SQL0954C エラーメッセージが生成されました。

Designer を使って DB2 データベースからソースをインポートする際に、DB2 システム変数 APPLHEAPSZ の値が小さすぎると、リポジトリへのアクセスに関するエラーが出力されます。ステータスバーには以下のメッセージが表示されます。

```
SQL Error:[IBM][CLI Driver][DB2]SQL0954C: Not enough storage is available in the application heap to process the statement.
```

このエラーが表示されたら、DB2 オペレーティングシステムに対する APPLHEAPSZ 変数の値を大きくしてください。APPLHEAPSZ は、データベースを使用している各プロセスに対する 4KB ページ単位のアプリケーションヒープサイズです。

第 3 章

フラットファイルに関する作業

この章では、以下の項目について説明します。

フラットファイルに関する作業の概要

ソース、ターゲット、ルックアップとしてマッピング内のフラットファイルを使用するには、リポジトリで定義をインポートまたは作成する必要があります。フラットファイルソース定義のインポートや作成には、Source Analyzer を使います。Target Designer を使用して、フラットファイルターゲット定義をインポートまたは作成できます。ルックアップトランスフォーメーションに、フラットファイルルックアップをインポートするか、既存のファイル定義を使用します。

注: ソース定義はソースに一致させる必要があるため、ファイルソース定義は手動で作成するのではなくインポートすることをお勧めします。

フラットファイルソースおよびターゲットを持つセッションの作成

ファイルソースとターゲットを使ってセッションを作成する場合、Designer で定義したプロパティをいくつか上書きできます。ファイルソースを使用するセッションを作成する場合、ファイルソース定義のインポート時と異なるソースファイルの場所を指定できます。

フラットファイルのインポート

ファイルにバイナリデータが含まれていない場合には、固定長および区切りフラットファイル定義をインポートできます。定義をインポートするには、当該ファイルがクライアントマシンのローカルディレクトリになけ

ればなりません。さらに、セッション中に PowerCenter Integration Service がすべてのソースファイルにアクセスできる必要があります。

ファイルソース定義、ファイルターゲット定義、またはファイルルックアップ定義を作成する場合には、このファイルのプロパティを正しく定義する必要があります。フラットファイルウィザードでは、以下のファイルプロパティの設定が必要です。

- ファイル名およびファイルの場所
- ファイルのコードページ
- ファイル種別
- カラムの名前とデータタイプ
- ファイル内のヘッダ行の数
- 固定長ファイルのカラムの長さと NULL キャラクタ
- 区切りファイルで使用される区切り文字のタイプ、引用符、およびエスケープ文字

特殊文字の処理

Designer でフラットファイルをインポートする場合、フラットファイルウィザードではデフォルトのフラットファイル定義の名前にファイル名が使用されます。フラットファイルウィザードを使用して、有効なファイル名を持つフラットファイルをインポートできます。しかし Designer では、フラットファイルソース名やターゲット名にあっても認識されない特殊文字があります。

フラットファイルをインポートすると、フラットファイルウィザードにより無効な文字やスペースがアンダースコア（_）に変換されます。例えば、「sample prices+items.dat」という名前のソースファイルがあるとします。Designer でこのフラットファイルをインポートすると、ファイル定義にはデフォルトで「sample_prices_items」という名前が付けられます。

関連項目：

- 「ソース定義内の特殊文字の処理」 (ページ 50)
- 「ターゲット定義内の特殊文字の処理」 (ページ 94)

コードページの選択

Designer でフラットファイルをインポートする際に、ファイルのコードページを選択することができます。Designer には、最近選択したコードページが 5 つ表示されます。その次に、残りのコードページがアルファベット順に表示されます。

コードページは、ファイルに含まれている *データ*のコードページを表します。ファイル定義を設定する際に、コードページがサポートしている区切り文字、NULL キャラクタ、およびエスケープ文字を指定します。

区切りファイルの場合、ソースファイルのコードページに含まれている区切り文字、オプションの引用符、およびエスケープ文字を指定します。固定長ファイルの場合、0 から 255 までのバイナリ値、または選択されたコードページの 1 文字を NULL キャラクタに指定します。

フラットファイル定義を設定する場合は、ワークフローを実行するのに必要なコードページ内で有効である区切り文字、エスケープ文字、および NULL キャラクタを使用します。

- **ASCII データ移動モード。**フラットファイルに定義するコードページから文字を使用します。PowerCenter の以前のバージョンで指定した 8 ビット文字はすべて有効です。
- **Unicode データ移動モード。**フラットファイルに定義するコードページから文字を使用します。

表示フォントの変更

フラットファイルウィザードを使用すると、フラットファイルに含まれるデータをプレビューできます。デフォルトでは、フラットファイルウィザードはシステムロケールを元にフォントを表示します。例えば、PowerCenter クライアントが、デフォルトの Latin-1 システムロケールを使用している Windows マシンにインストールされているとします。フラットファイルウィザードのフォントはデフォルトで Courier に設定されます。プレビューウィンドウには、Courier フォントで Latin-1 文字が表示されます。

Latin-1 ロケールと互換性のないフォントを表示する場合は、フラットファイルウィザードのフォントを変更する必要があります。例えば、Latin-1 マシンで実行されているフラットファイルウィザードのプレビューウィンドウで日本語フォントを表示する場合は、フォントを MS ゴシックなどの日本語フォントに変更する必要があります。

フラットファイルウィザードのフォントは Designer で設定できます。

フォントを設定してもまだプレビューウィンドウのテキスト表示に問題がある場合は、以下のタスクを実行します。

1. Designer のフォーマットをリセットします。
2. データの言語がマシンにインストールされていることを確認します。必要に応じて、言語をインストールします。

Designer のフォーマットのリセット

Designer のフォーマットはデフォルト設定にリセットすることができます。

Designer のフォーマットをデフォルト設定に戻します。

1. Designer で、［ツール］-［オプション］の順にクリックします。
2. ［すべて元に戻す］をクリックしてデフォルトにリセットします。

言語のインストール

Windows マシンに言語をインストールする手順

▶ ［コントロールパネル］-［地域のオプション］をクリックします。

固定長フラットファイルのインポート

固定長フラットファイルはバイト数ベースであり、フィールド長はバイト単位で数えます。また、ラインシーケンシャルでもあり、各行が改行で終わることもあります。バイナリデータや各文字が 2 バイトより大きなマルチバイト文字データを含まない固定長ファイルをインポートできます。

固定長ファイルをインポートするときに、フラットファイルウィザードを使ってカラムの区切りの作成、移動、および削除を行うことができます。カラムの区切りの配置が不適切であると、1 バイト文字とマルチバイト文字を含むファイルソースを使用するセッションを実行した際に、バイト長不揃いエラーが発生する場合があります。ファイルにおけるマルチバイトデータのバイト長の不揃いによって、ワークフローのエラーが発生する場合があります。

固定長フラットファイル定義をインポートするには：

1. ソース定義をインポートするには、Source Analyzer を開き、**［ソース］** > **［ファイルからインポート］**をクリックします。ターゲット定義をインポートするには、Target Designer を開き、**［ターゲット］** > **［ファイルからインポート］**をクリックします。

   ［フラットファイルを開く］ダイアログボックスが表示されます。
2. 使用するファイルを選択します。

3. コードページを選択します。

   フラットファイルソース定義をインポートするときは、そのファイルにあるデータのコードページに一致するコードページを選択してください。

4. **[OK]** をクリックします。

   ファイルの内容がフラットファイルウィザード下部のウィンドウ内に表示されます。

5. 次の設定を編集します。

固定長フラットファイルウィザード、手順 1/3	**説明**
フラットファイルタイプ	ファイル種別。固定長ファイルの場合は［固定長］を選択します。
ソース名	ソースの名前。リポジトリ内のソースの名前です。ファイル名またはその他の論理名を使用できます。
インポート開始行を指定	フラットファイルウィザードでファイルをインポートする際に、読み込みを開始する行番号を指定します。 たとえば行 2 から開始するように指定すると、1 行目をスキップしてから読み込みを開始します。
1 行目からフィールド名を取り込む	選択すると、Designer は最初の行のデータをカラム名として使用します。カラム名が最初の行に含まれている場合に、このオプションを選択します。「FIELD_」で始まる無効なフィールド名があります。

6. **[次へ]** をクリックします。

   表示されるウィザードの指示に従い、ファイルのプレビューウィンドウでカラム間の境界線を操作します。ドラッグすればカラム間の境界線の移動ができます。境界線を削除するには、これをダブルクリックします。

   シフト依存ファイルの場合、フラットファイルウィザードのウィンドウでは 1 バイトのシフト文字が「.」で表示されます。2 バイトシフト文字は「..」としてウィンドウに表示されるので、カラムの境界線を正確に設定できます。

7. **[次へ]** をクリックします。

   ファイルの各カラムについてのカラム情報を入力します。

カラムの選択には、**［ソース定義］**または**［ターゲット定義］**グループで新しいカラムを選択するか、ファイルプレビューウィンドウ内のカラム見出しをクリックします。

固定長フラットファイルウィザード、手順 3/3	**説明**
名前	各カラムで表示したいポート名。［最初の行からフィールド名を取得］を選択した場合には、ファイル内のカラム名が読み込まれます。
データ型	列のデータ型。［文字］、［数値］、［日時］のいずれかを選択してから、［長さ/精度］、［位取り］、［幅］を適切に入力します。 数値カラムの場合は、［長さ/精度］は有効桁数になり、［幅］はソースファイルから読み込まれるバイト数か、ターゲットファイルに書き込まれるバイト数になります。 テキストカラムの場合は、固定長ファイルの精度はバイト単位、区切りファイルの精度は文字単位です。 デフォルトでは、［長さ/精度］と［幅］には同じ値が入力されています。精度の値は有効桁数を入力して変更できますが、フィールド長は精度以上の値に設定する必要があります。 **注**: 注：数値とみなされるのは、文字 0 - 9 のみです。日本語のようなマルチバイト文字セットの数値を含むカラムはテキストとみなされます。

8. **［完了］**をクリックします。

   注: ファイルのサイズが 256KB を超えるか、または各行のデータが 16KB より大きい場合は、フラットファイルウィザードが適切なフィールド精度とフィールド長でファイルをインポートしたかどうかを確認してください。ファイルが不適切な場合は、フラットファイルウィザードまたはインポートした定義でフィールド精度とフィールド長を調整してください。

区切りフラットファイルのインポート

区切りファイルは常に文字数ベースでありラインシーケンシャルです。カラム精度は、文字列カラムの場合は常に文字数で算出され、数値カラムの場合は有効桁数で算出されます。各行は改行で終わります。バイナリデータや各文字が 2 バイトより大きなマルチバイト文字データを含まない区切りファイルをインポートできます。

ソース定義またはターゲット定義に区切りファイルをインポートするには、以下の手順を実行します。

1. ソース定義をインポートするには、Source Analyzer を開き、**［ソース］** > **［ファイルからインポート］**をクリックします。ターゲット定義をインポートするには、Target Designer を開き、**［ターゲット］** > **［ファイルからインポート］**をクリックします。

   ［フラットファイルを開く］ダイアログボックスが表示されます。

2. 使用するファイルを選択します。
3. コードページを選択します。

   フラットファイルソース定義をインポートするときは、そのファイルにあるデータのコードページに一致するコードページを選択してください。

4. **［OK］**をクリックします。

   ファイルの内容がフラットファイルウィザード下部のプレビューウィンドウ内に表示されます。

5. 次の設定を編集します。

区切りフラットファイルウィザード、手順 1/3	**説明**
フラットファイルタイプ	ファイル種別。区切りファイルの場合は［区切りファイル］を選択します。
ソース名	ソースの名前。リポジトリ内のソース定義の名前です。ファイル名またはその他の論理名を使用できます。
インポート開始行を指定	フラットファイルウィザードでファイルをインポートする際に、読み込みを開始する行番号を指定します。たとえば行 2 から開始するように指定すると、1 行目をスキップしてから読み込みを開始します。
1 行目からフィールド名を取り込む	選択すると、Designer は最初の行のデータをカラム名として使用します。カラム名が最初の行に含まれている場合に、このオプションを選択します。「FIELD_」で始まる無効なフィールド名があります。

6. **［次へ］**をクリックします。

   構文の不正なフィールドは、この画面下部のファイルプレビューウィンドウに赤字で表示されます。

7. 以下の設定を入力します。

区切りフラットファイルウィザード、手順 2/3	**説明**
区切り文字	データのカラムを区切るために使用される文字。［その他］フィールドを使用すれば、別の区切り文字が入力できます。区切り文字は印刷可能な文字でなければなりません。また、エスケープ文字や引用符を選択した場合には、それと別な文字でなければなりません。印刷できないマルチバイト文字を区切り文字として選択することはできません。
連続した区切り文字は 1 文字として扱う	選択すると、フラットファイルウィザードは 1 つ以上の連続したカラム区切り文字を１つのカラム区切り文字として読み込みます。選択しなければ、2 つの連続した区切り文字は NULL 値として読み込まれます。
複数の区切り文字を AND として扱う	選択された場合、フラットファイルウィザードでは指定した区切り文字のセットが 1 文字として読み込まれます。
エスケープ文字	引用符で囲まれていない文字列におけるカラム区切り文字の直前の文字、または引用符で囲まれた文字列内における引用符の直前の文字。エスケープ文字を指定すると、は、区切り文字を普通文字として読み込みます。エスケープ文字は、区切りの*エスケープ*または引用符文字と呼ばれています。
データからエスケープ文字列を削除	デフォルトでは、このオプションはオンになっています。エスケープ文字を出力文字列に含めるには、このオプションのチェックマークを外してください。

区切りフラットファイルウィザード、手順 2/3	説明
デフォルトのテキストの長さを使う	このオプションを選択すると、すべての文字列データ型について、入力したデフォルトのテキスト長が使用されます。
引用符	引用符はテキスト文字列の境界を定義します。[なし]、[シングルクォーテーション]、または［ダブルクォーテーション］を選択してください。引用符を選択すると、引用符で囲まれた文字列内の区切り文字が無視されます。

8. **［次へ］**をクリックします。
9. ファイルの各カラムについてのカラム情報を入力します。

   カラム間の切り替えを行うには、[ソースの定義］ペインまたは［ターゲットの定義］ペインで新しいカラムを選択するか、ファイルプレビューのカラム見出しをクリックします。

区切りフラットファイルウィザード、手順 3/3	説明
名前	各カラムで表示したいポート名。[最初の行からフィールド名を取得］を選択した場合には、ポート名ではなくファイル内のカラム名が読み込まれます。
データ型	列のデータ型。[文字]、[数値]、[日時］のいずれかを選択してから、[長さ/精度]、[位取り]、[幅］を適切に入力します。 数値カラムの場合は、[長さ/精度］は有効桁数になります。フラットファイルウィザードでは、区切りファイルの数値カラムのフィールド長は無視されます。 テキストカラムの場合は、[長さ/精度］はソースフィールドまたはターゲットフィールドに含まれる文字数の最大値になります。フラットファイルウィザードでは、区切りファイルからテキストカラムを読み出すとき、またはテキストカラムを区切りファイルに書き込むときには、この精度は無視されます。 デフォルトでは、[長さ/精度］と［幅］には同じ値が入力されています。精度または幅の値は変更できます。ただし、フラットファイルウィザードで設定できる精度の値は、幅の値以上に限定されます。 **注**: 注：数値とみなされるのは、文字 0 - 9 のみです。日本語のようなマルチバイト文字セットの数値を含むカラムはテキストとみなされます。

10. **［完了］**をクリックします。

    注: ファイルのサイズが 256KB を超えるか、または各行のデータが 16KB より大きい場合は、フラットファイルウィザードが適切なフィールド精度でファイルをインポートしたかどうかを確認してください。ファイルが不適切な場合は、フラットファイルウィザードまたはインポートした定義でフィールド精度を調整してください。

関連項目：

- 「区切りファイルの設定に関するルールおよびガイドライン」 (ページ 83)
- 「フラットファイルカラムのフォーマット」 (ページ 84)

フラットファイル定義の編集

フラットファイルのソース定義またはターゲット定義をインポートした後で、ビジネス名の追加やファイルプロパティの設定を実行する必要がある場合もあります。さらに、ファイル定義を変更した場合は、手動で定義を編集できます。

次の定義タブを使って、ソースフラットファイル定義やターゲットフラットファイル定義を編集します。

- テーブル名、ビジネス名、フラットファイルのプロパティなどの各種プロパティを編集します。
- **［カラム］タブ。**カラム名、データタイプ、精度、フォーマットなどのカラム情報を編集します。
- Source Analyzer および Target Designer における、デフォルトの数値フォーマットと日付フォーマットのプロパティをそれぞれ表示します。Mapping Designer で、マッピングに含まれる各ソースインスタンスとターゲットインスタンスについてプロパティを編集できます。
- **フラットファイル定義などのリポジトリオブジェクトに情報を関連付けることで、リポジトリに格納されているメタデータを拡張することができます。**

注: ソース定義のファイル構造が大きく変更された場合には、ファイルソース定義の再インポートが必要になる場合があります。

ソース定義やターゲット定義を更新すると、Designer はそのソースまたはターゲットを使っているマッピングすべてに変更を反映します。ソース定義やターゲット定義への変更の内容によっては、マッピングが無効になる場合もあります。変更したためにマッピングが無効となった場合には、このマッピングを検証しなければなりません。クエリの結果ウィンドウ、依存関係の表示ウィンドウ、またはナビゲータウインドウからマッピングを検証できます。また、ワークスペースで複数のオブジェクトを開くことなくこれらを検証できます。これらの場所からマッピングを検証できない場合、マッピングを開いて編集する必要があります。

ファイルソース、ファイルターゲット、またはファイルルックアップを使ってセッションを作成する場合、Designer で定義したプロパティをいくつか上書きできます。例えば、ファイルソースを使用するセッションを作成する場合、ファイルソース定義のインポート時と異なるソースファイルの場所を指定できます。

テーブルのオプションの編集

フラットファイルソース定義またはターゲット定義の［テーブル］タブでは、次のオプションを編集できます。

- **ビジネス名。**ソース定義またはターゲット定義に、説明的な名前を追加します。
- コメントやビジネス文書へのリンクを追加します。この説明は Repository Manager でソース定義やターゲット定義について表示されます。ソースやターゲットにコメントやビジネス文書のリンクを追加することで、簡単に目的を記述することができます。既存のソースやターゲットにコメントを追加したり、コメントを修正することができます。

  選択したリポジトリのコードページにおける各文字の最大バイト数を K とした場合、説明に 2000/K 文字まで入力できます。例えば、リポジトリのコードページが日本語コードページ（K=2）の場合、説明およびコメントのフィールドにはそれぞれ 1,000 文字まで入ります。
- キーワードを使用してフラットファイルターゲットを追跡します。開発作業や保守作業を進めると、ターゲット数が増大します。ターゲットがすべて同一のフォルダに表示されることがありますが、これらのターゲットの目的がそれぞれ異なることもあります。キーワードは関連ターゲットの検索に役立ちます。キーワードに含めることができるのは、開発名、マッピング、または関連スキーマです。

  Repository Manager で、キーワードを使用して検索を実行できます。
- **データベースタイプ。**ソースタイプまたはターゲットタイプを定義します。 フラットファイルソースとフラットファイルターゲットの場合は、［フラットファイル］を選択します。
- **フラットファイル情報。**データベースタイプがフラットファイルの場合、［詳細］ボタンをクリックしてフラットファイルのプロパティを定義します。

フラットファイルソース定義またはターゲット定義にオプションを追加するには：

1. ソース定義にオプションを追加するには、Source Analyzer で、ソース定義のタイトルバーをダブルクリックします。 ターゲット定義にオプションを追加するには、Target Designer で、ターゲット定義のタイトルバーをダブルクリックします。

   ［テーブルの編集］ダイアログボックスが表示されます。
2. ［名前の変更］をクリックし、ソース名またはターゲット名、ビジネス名を編集します。
3. ［データベースタイプ］フィールドで［フラットファイル］を選択します。
4. ［詳細設定］をクリックして、フラットファイルのプロパティを編集します。

   固定長ファイルおよび区切りファイル用に、それぞれ異なるダイアログボックスが表示されます。
5. 説明を追加するには、［説明］フィールドに説明を入力します。
6. ターゲット定義のキーワードを追加するには、［キーワードの編集］をクリックします。

   ［キーワードの編集］ダイアログボックスが表示されます。ボタンを使って、キーワードの作成や移動を行います。

関連項目：

- 「固定長フラットファイルのインポート」 (ページ 70)
- 「区切りフラットファイルのインポート」 (ページ 72)

カラムの編集

フラットファイルのソース定義またはターゲット定義の［カラム］タブで、以下の情報を編集できます。

- **カラム名。**フラットファイルソースまたはフラットファイルターゲットにおけるカラムの名前。
- **ファイル名カラム。**フラットファイルソースまたはターゲットにファイル名カラムを追加します。 ソースでは、［CurrentlyProcessedFileName］カラムを使用してデータ行が読み込まれるソースファイルの名前を返します。セッションでファイルリストからデータを読み込むように設定する場合は、このカラムを使用します。ターゲットでは、［FileName］カラムを使用してフラットファイルターゲットに動的に名前を指定します。
- **データタイプ。**カラムのデータタイプ。 フラットファイルの場合、bigint、datetime、double、int、nstring、number、string のいずれかを選択できます。
- **精度、位取り、およびフォーマット。**ファイル定義のインポートを行う場合に、各カラムの値の精度、位取り、フィールド長、値のフォーマットを検討する必要のある場合がしばしばあります。 ［フォーマット］カラムをクリックすると、フィールド長とフォーマットを編集できます。精度、位取り、およびフォーマットを入力します。
- ソースまたはターゲットで、NULL データを許可するかどうかを選択します。
- **キータイプ。**フラットファイルのソース定義とターゲット定義の［キー以外］を選択します。
- シフトキーを含まないシフト依存の固定長フラットファイルソースについて、シフト状態を定義できます。この属性は、固定長ファイルの［フラットファイル属性の編集］ダイアログボックスで、［ユーザー定義シフトステート］を選択した場合に表示されます。

  カラムに 1 バイト文字が含まれる場合には、［Shift-In］を選択します。カラムにマルチバイト文字が含まれる場合には、［Shift-Out］を選択します。
- 必要に応じて、各ソースフィールドまたはターゲットフィールドにビジネス名を追加できます。

注: ソース定義またはルックアップ定義のファイルカラムが大きく変更された場合には、ファイルの再インポートが必要になる場合があります。

フラットファイルのソース定義およびターゲット定義のカラムを編集するには、以下のとおりに実行します。

1. のソース定義を編集するには、Source Analyzer で、そのソース定義のタイトルバーをダブルクリックします。フラットファイルソース定義を編集するには、Target Designer で、フラットファイルターゲット定義のタイトルバーをダブルクリックします。
2. ［カラム］タブをクリックします。
3. 前述の説明に従って、ソース定義またはターゲット定義のオプションを設定します。
4. カラムを追加する場合には、カラムを選択して［追加］をクリックします。
5. カラムの名前、データタイプなどの特性を入力します。

   ソース定義またはターゲット定義に追加する各カラムについて、上記の手順を繰り返します。
6. カラムを移動する場合には、［上に移動］および［下に移動］ボタンを使うか、またはスクローリングリスト内でカラムをドラッグします。

固定長ファイルのプロパティの更新

固定長ファイルをインポートしたら、ファイルプロパティを更新できます。ソース定義またはターゲット定義のタイトルバーをダブルクリックします。テーブルおよびカラムの情報を編集します。

ファイルプロパティを編集するには、［テーブル］タブの［詳細設定］をクリックします。［フラットファイル属性の編集-固定長ファイル］ダイアログボックスが表示されます。［フラットファイル属性の編集-固定長ファイル］ダイアログボックスには、ファイルターゲットのオプションよりもファイルソースのオプションの方が多くあります。例えば、スキップする最初の行数や行間のバイト数など、がファイルを読み込むために必要な情報が含まれています。

以下の表に、ソース定義、ターゲット定義、およびルックアップ定義に設定できる固定長ファイルのプロパティを示します。

固定長ファイルの詳細設定項目	ソース定義およびルックアップ定義の場合	ターゲット定義の場合
NULL キャラクタ	NULL 値を表すためにソースファイルで使用する文字。ファイルのコードページ内の有効な文字、または 0 から 255 までのバイナリ値です。	がターゲットファイルで NULL 値を表すために使用する文字。ファイルのコードページ内の有効な文字です。
Null キャラクタを繰り返す	このオプションを選択すると、Integration Service は 1 つのフィールド内の連続した NULL キャラクタを 1 つの NULL キャラクタとして読み込みます。マルチバイト NULL キャラクタを指定して［Null キャラクタの繰り返しを許可］を選択した場合、NULL キャラクタのバイト数でフィールド長が割り切れないときは、フィールドの後ろに余分なバイトが含まれます。この場合、フィールドは NULL ではありません。常に、バイトの NULL キャラクタを指定する必要があります。	選択した場合、はターゲットフィールドに書き込める数だけ NULL キャラクタを書き込みます。このオプションを選択しない場合、Integration Service はフィールドの最初に 1 つの NULL キャラクタを入れて NULL 値を表します。 マルチバイト NULL キャラクタを指定した場合で、NULL キャラクタを書き込んだ後に余分なバイトが残っている場合、Integration Service は 1 バイト分のスペースでカラムを埋めます。カラムが NULL キャラクタとして指定されたマルチバイト文字より小さいために、NULL キャラクタをカラムに入れられない場合は、セッションは初期化で失敗します。

固定長ファイルの詳細設定項目	ソース定義およびルックアップ定義の場合	ターゲット定義の場合
コードページ	ファイル定義のコードページ。 ソース定義には、ターゲットコードページのサブセットに該当するコードページを使用します。 ルックアップファイル定義の場合、ソースコードページのスーパーセットでありターゲットコードページのサブセットにあたるコードページを使用します。	ファイル定義のコードページ。 ソースコードページのスーパーセットに該当するコードページを使用します。
改行コードあり	選択した場合、は最後のカラムの改行文字または復帰文字をこのカラムの末尾として読み込みます。ファイルで改行または復帰を使用して各行の最後のカラムを短くしている場合に、このオプションを使用します。	なし
スキップする先頭の行数	Integration Service がファイルを読み込む場合にスキップする行数を示します。この設定は、空の行またはヘッダ行をスキップするために使用します。1 行に複数のレコードが含まれている場合もあります。 0 から 2,147,483,647 の任意の整数を入力します。	なし
スキップするレコード間のバイト数	行の最後のカラムと次の行の最初のカラムの間のバイト数。Integration Service は、指定されたバイト数を行の最後でスキップして、復帰文字や改行文字を読み込まないようにします。UNIX ファイルの場合は「1」、DOS ファイルの場合は「2」を指定します。	なし
後続のスペースを削除	このオプションを選択すると、Integration Service により文字列値から末尾の空白が除去されます。	なし
ユーザー定義シフトステート	これを選択すると、[カラム] タブで、ソースカラムのシフト状態を定義できます。 ソースファイルにマルチバイトデータと 1 バイトデータの両方が含まれ、かつシフトインキーとシフトアウトキーが含まれないという場合は、[ユーザー定義シフトステート] を選択します。マルチバイトファイルソースにシフトキーが含まれない場合、フラットファイルのソース定義で各カラムのシフト状態をがそれぞれの文字を正しく読み込めるように定義する必要があります。	なし

NULL キャラクタの処理

固定長ファイルソースに、1 バイトまたはマルチバイトの NULL キャラクタを指定することができます。固定長ソースファイルを読み込む場合、はこれらの文字を使用してカラムが NULL かどうかを判断します。固定長ターゲットファイルに書き込む場合、はこれらの文字を使用して NULL 値を表します。

以下の表に、が［Null キャラクタ］および［Null の連続を許可］プロパティを使用してカラムが NULL であるかどうかを判断する方法を示します。

NULL キャラクタ	Null キャラクタを繰り返す	ソースおよびルックアップから読み込む際の Integration Service の動作	ターゲットに書き込む際の Integration Service の動作
バイナリ	無効	カラム内の最初のバイトがバイナリ NULL キャラクタである場合、そのカラムは NULL です。Integration Service は、カラムの残りの部分をテキストデータとして読み込んで、カラムのバイト長が揃っているかどうかを判断し、シフト依存のコードページについてはシフトの状態を追跡します。カラム内のデータが不揃いであれば、Integration Service はその行をスキップして、対応するエラーメッセージと共にセッションログに書き込みます。	Integration Service は、フィールドの先頭に 1 つのバイナリ NULL キャラクタを 1 つ入力して NULL 値を表します。 NULL キャラクタを書き込んだ後にバイトが余っていれば、Integration Service はカラムの空きを 1 バイトのスペースで埋めます。
非バイナリ	無効	カラム内の最初の文字が NULL キャラクタであれば、そのカラムは NULL です。Integration Service は、カラムの残りの部分を読み込んでカラムのバイト長が揃っているかどうかを判断し、シフト依存のコードページについてはシフトの状態を追跡します。カラム内のデータが不揃いであれば、Integration Service はその行をスキップして、対応するエラーメッセージと共にセッションログに書き込みます。	Integration Service は、フィールドの先頭に 1 つの NULL キャラクタを入力して NULL 値を表します。 マルチバイト NULL キャラクタを指定した場合で、NULL キャラクタを書き込んだ後に余分なバイトが残っている場合、Integration Service は 1 バイト分のスペースでカラムを埋めます。カラムが NULL キャラクタとして指定されたマルチバイト文字より小さいために、NULL キャラクタをカラムに入れられない場合は、セッションは初期化で失敗します。

NULL キャラクタ	Null キャラクタを繰り返す	ソースおよびルックアップから読み込む際の Integration Service の動作	ターゲットに書き込む際の Integration Service の動作
バイナリ	有効	指定されたバイナリ NULL キャラクタのみがカラム内に含まれている場合、そのカラムは NULL です。次のカラムは、コードページの初期のシフト状態を継承します。	Integration Service は、ターゲットフィールドに書き込めるだけのバイナリ NULL キャラクタを書き込みます。
非バイナリ	有効	連続する NULL キャラクタでカラムがちょうど埋まっていて、1 バイトも残っていなければ、そのカラムは NULL です。例えば、2 バイトの連続する NULL キャラクタを指定した場合、5 バイトのカラムは NULL ではありません。 シフト依存のコードページでは、シフトバイトはカラム内の NULL バイトに影響しません。カラムの冒頭または末端にシフトバイトが含まれ、連続する NULL キャラクタが余分なバイトを出さずにぴったりとカラムに収まっている場合、このカラムは NULL です。 連続する非バイナリ NULL キャラクタを使用している場合は、1 バイトの NULL キャラクタを指定します。そうすることにより、連続する NULL キャラクタでカラムをちょうど埋めることができます。	Integration Service は、ターゲットフィールドに書き込めるだけの NULL キャラクタを書き込みます。 マルチバイト NULL キャラクタを指定した場合で、NULL キャラクタを書き込んだ後に余分なバイトが残っている場合、Integration Service は 1 バイト分のスペースでカラムを埋めます。 カラムが NULL キャラクタとして指定されたマルチバイト文字より小さいために、NULL キャラクタをカラムに入れられない場合は、セッションは初期化で失敗します。

区切りファイルのプロパティの更新

区切りファイルをインポートした後に、ファイルプロパティを更新できます。ソース定義またはターゲット定義のタイトルバーをダブルクリックします。テーブルおよびカラムの情報を編集します。

ファイルプロパティを編集するには、［テーブル］タブの［詳細設定］をクリックします。［フラットファイル属性の編集 -区切りファイル］ダイアログボックスが表示されます。［フラットファイル属性の編集 -区切りファイル］ダイアログボックスには、ファイルターゲットのオプションよりもファイルソースのオプションの方が多くあります。例えばスキップする最初の行数やエスケープ文字など、がファイルを読み込むために必要な情報が含まれています。

以下の表に、設定可能な区切りファイルプロパティの説明を示します。

区切りファイルの詳細設定項目	ソース定義およびルックアップ定義の場合	ターゲット定義の場合
カラム区切り文字	データのカラムを区切るために使用される文字。区切り文字には、印刷できる文字か、1 バイトの印刷できない文字を使用できます。また、エスケープ文字や引用符と異なる文字にする必要があります。1 バイトの印刷できない文字を入力するには、[区切り文字] ダイアログボックスで区切り文字リストを参照できます。 印刷できないマルチバイト文字を区切り文字として選択することはできません。フラットファイルソースのカラム区切り文字として NULL 文字を選択することはできません。	データのカラムを区切るために使用される文字。区切り文字には、印刷できる文字か、1 バイトの印刷できない文字を使用できます。また、引用符と異なる文字にする必要があります。1 バイトの印刷できない文字を入力するには、[区切り文字] ダイアログボックスで区切り文字リストを参照できます。 印刷できないマルチバイト文字を区切り文字として選択することはできません。複数の区切り文字を入力すると、統合サービスは最初に指定した区切り文字を使用します。
連続した区切り文字は 1 文字として扱う	選択した場合、統合サービスは１つ以上の連続したカラム区切り文字を１つのカラム区切り文字として処理します。選択しなければ、統合サービスは 2 つの連続した区切り文字を NULL 値として読み込みます。	なし
複数の区切り文字を AND として扱う	選択された場合、統合サービスでは指定した区切り文字のセットが 1 文字として扱われます。例えば、ソースファイルには次のレコードが含まれます。abc~def|ghi~|~|jkl|~mno。デフォルトでは、統合サービスにより、8 つの区切り文字で分けられた 9 つのカラムとしてレコードが読み込まれます（abc、def、ghi、NULL、NULL、NULL、jkl、NULL、mno）。このオプションを選択して区切り文字を（~ |）として指定した場合、統合サービスにより、このレコードを 2 つの区切り文字で分けられた 3 つのカラムとして読み込まれます（abc~def|ghi、NULL、jkl|~mno）。	なし

区切りファイルの詳細設定項目	ソース定義およびルックアップ定義の場合	ターゲット定義の場合
引用符	[なし]、[シングル]、または［ダブル］を選択します。引用符はテキスト文字列の境界を定義します。デフォルトで、［ダブル］が選択されています。 選択した場合、統合サービスは引用符で囲まれたカラム区切り文字を無視します。 たとえば、ソースファイルではカンマを区切り文字として使用しており、統合サービスはソースファイルから以下の行「342-3849, 'Smith, Jenna', 'Rockville, MD', 6」を読み込む場合を想定します。 オプションとして一重引用符を選択した場合、統合サービスは引用符で囲まれた範囲内のカンマを無視して、4 つのフィールドとして行を読み込みます。 オプションとして一重引用符を選択しない場合は、統合サービスはこれを 6 つの個別のフィールドとして読み込みます。 **注**: ソースファイルに引用符やエスケープ文字を設定しないようにすれば、セッションのパフォーマンスを向上できます。	[なし]、[シングル]、または［ダブル］を選択します。引用符はテキスト文字列の境界を定義します。デフォルトで、［ダブル］が選択されています。 引用符を選択すると、統合サービスは引用符で囲まれたカラム区切り文字を区切り文字として扱いません。 たとえば、ターゲットファイルではカンマを区切り文字として使用しており、統合サービスはターゲットファイルに以下の行「342-3849, 'Smith, Jenna', 'Rockville, MD', 6」を書き込む場合を想定します。 オプションとして一重引用符を選択した場合、統合サービスは引用符で囲まれた範囲内のカンマを無視して、4 つのフィールドとして行を書き込みます。 オプションとして一重引用符を選択しない場合は、統合サービスはこれを 6 つの個別のフィールドとして書き込みます。 **注**: 統合サービスは、ターゲットファイル内の NULL 値にオプションの引用符を追加しません。例えば、ソースファイル内の入力行に、2 番目のカラムに引用符で囲まれた Null 値を持つ 3 つのカラムが含まれています。統合サービスは、引用符を省略し、次の形式でターゲットファイルに行を書き込みます。 `'<value_a>',,'<value_c>'`
コードページ	ファイル定義のコードページ。 ソース定義には、ターゲットコードページのサブセットに該当するコードページを使用します。ルックアップファイル定義の場合、ソースコードページのスーパーセットでありターゲットコードページのサブセットにあたるコードページを使用します。	ファイル定義のコードページ。 ソースコードページのスーパーセットに該当するコードページを使用します。
行区切り文字	改行文字を指定します。リストから選択するか、文字を入力します。8 進コードの前にはバックスラッシュ（\）をつけます。1 文字を使用するには、その文字を入力します。 入力したものの先頭にバックスラッシュが付いていない場合は、統合サービスは最初の文字だけを使用します。文字は 1 バイト文字とします。コードページ内のそれ以外の文字には、そのバイトが含まれていてはいけません。デフォルトは、改行、\012 LF（\n）です。	なし

区切りファイルの詳細設定項目	ソース定義およびルックアップ定義の場合	ターゲット定義の場合
エスケープ文字	引用符で囲まない文字列内で、直後に置かれた区切り文字を*エスケープ*するために使う文字。に文字列中の区切り文字を普通文字として読ませる（区切り文字の*エスケープ*）場合に使用します。 **注**: ソースファイルが引用符やエスケープ文字を含まないようにすれば、セッションのパフォーマンスを向上できます。	なし
データからエスケープ文字列を削除	デフォルトでは、このオプションはオンになっています。エスケープ文字を出力文字列に含めるには、このオプションのチェックマークを外してください。	なし
スキップする先頭の行数	統合サービスがファイルを読み込む場合にスキップする行数を示します。この設定は、空の行またはヘッダー行をスキップするために使用します。	なし

区切りファイルの設定に関するルールおよびガイドライン

区切りファイルは文字数ベースでありラインシーケンシャルです。区切りファイルを設定する場合には、以下の規則およびガイドラインに従ってください。

- 1 つのソース定義において、カラムと行の区切り文字、引用符、およびエスケープ文字はすべて異なる文字でなければなりません。これらのプロパティは、ソースまたはターゲットファイルコードページにも含まれている必要があります。
- エスケープ文字および区切り文字は、ソースファイルまたはターゲットファイルのコードページ内で有効であることが必要です。

区切りファイルのソースを設定する場合には、以下の規則およびガイドラインに従ってください。

- 引用符で囲まれた文字列内では、エスケープ文字を使用して引用符をエスケープできます。エスケープ文字が引用符の直前にない場合には、はこのエスケープ文字を普通の文字として読み込みます。
- カラムの*区切り文字*をエスケープするには、エスケープ文字を使用します。ただし、引用符で囲まれた文字列内では引用符がその役割を果たすため、*区切り文字*をエスケープするためにエスケープ文字を使用する必要はありません。エスケープ文字が区切り文字の直前にない場合、はこのエスケープ文字を普通の文字として読み込みます。
- 引用文字列内に 2 つの引用符が連続して出現した場合、はこれらを 1 つの引用符として読み込みます。例えば、Integration Service は以下の引用文字列を「I'm going tomorrow」として読み込みます。

  2353,‘I’’m going tomorrow’MD
- 選択した引用符がフィールドの最初の文字である場合に限り、はその文字列を引用符で囲まれた文字列として読み込みます。
- フィールド長がソース修飾子トランスフォーメーションで定義したカラムサイズを超える場合、はそのフィールドを切り詰めます。
- データの行が、ラインシーケンシャルバッファ長、またはソース修飾子トランスフォーメーションで定義した合計の行サイズのどちらか大きい方のサイズを超える場合、はその行を処理せずにセッションログファイルに書き込みます。ソース修飾子トランスフォーメーションで定義する行のサイズを決定するには、カラムの精度と区切り文字を追加して、その合計に１文字あたりの最大バイトを掛けます。

フラットファイルカラムのフォーマット

フラットファイル定義をインポートまたは編集する場合には、次のカラムオプションを定義する必要があります。

- 精度はデータタイプによって定義内容が異なります。
- **位取りは、数値の小数点以下の最大桁数です。**
- **フィールド長。**フィールド長は、Integration Service によりファイルから読み込まれるバイト数またはファイルに書き込まれるバイト数です。 フィールド長は、固定長ファイルソースおよびターゲットにのみ適用されます。フィールド長は精度以上の値である必要があります。

  以下の表に、フラットファイル定義の精度およびフィールド長の定義を示します。

データタイプ	固定長フラットファイル	区切りフラットファイル
番号	精度は有効桁数になります。 フィールド長は、Integration Service がファイルから読み込むバイト数またはファイルに書き込むバイト数になります。デフォルトでは、フィールド長と精度は同じ値を持ちます。 フィールド長は［フォーマット］カラムを使って定義します。フィールド長を設定する場合は、1000 ごとの区切り、小数点記号、マイナス記号などの文字を考慮に入れます。例えば、「123,456」のフィールド長は 8 です。	精度は有効桁数になります。 Integration Service では、フィールド長の設定はすべて無視されます。
日時	精度の値は入力しません。 フィールド長は、Integration Service がファイルから読み込むバイト数またはファイルに書き込むバイト数になります。 カラムに指定された日付フォーマットにより、フィールド長が決まります。例えば、「MM/DD/YYYY HH24:MI:SS」というデフォルトの日付フォーマットのフィールド長は 19 バイトになります。 フォーマットとフィールド長は［フォーマット］カラムを使って定義します。	精度の値は入力しません。 Integration Service では、フィールド長の設定はすべて無視されます。
文字列	精度は、Integration Service がファイルから読み込むバイト数またはファイルに書き込むバイト数です。 文字列値にはフィールド長を入力しません。精度は、ソースフィールドまたはターゲットフィールドの総計長さになります。 **注**: 注：マルチバイトデータを固定長ファイルターゲットにロードする場合は、マルチバイトデータを収容できる精度を設定する必要があります。	精度は、Integration Service がファイルから読み込む文字またはファイルに書き込む文字の最大数です。 フィールド長は入力しません。

- フォーマットにより、数値と日付の値の表現を定義します。

  日付の値では、データの日付部分または時刻部分のみ出力するように選択できます。例えば、日付データのファイル定義を次のフォーマットで定義できます。

  03/2002

  数値については、3 桁ごとの区切りと小数点記号を選択できます。例えば、数値データのファイル定義を次のフォーマットで定義できます。

  1.000.000,95

注: 注：ソースファイル構造が大きく変更された場合には、フラットファイルソース定義の再インポートが必要になる場合があります。

次の場所で、数値カラムと日付カラムの表現を定義できます。

- ［カラム］タブでは、Source Analyzer のソースまたは Target Designer のターゲットにおける各カラムのフォーマットを定義できます。
- **マッピングのソースインスタンスまたはターゲットインスタンス。**マッピングのソースインスタンスまたはターゲットインスタンスのデフォルトの日付フォーマットと数値フォーマットは、Mapping Designer で定義できます。 個別のカラムについてのフォーマットを定義しなかった場合、はユーザーが定義したデフォルトのフォーマットを使用します。

数値カラムのフォーマット

フラットファイル定義を編集する場合、数値のフォーマットを Number データタイプで定義することができます。Bigint、double および integer データタイプには、デフォルトの精度、位取り、フォーマットがあります。 integer または double データタイプのカラムの精度を変更することができます。

フラットファイルソース定義またはターゲット定義の［カラム］タブの［フォーマット］カラムを使用して、数値のフォーマットを定義します。

［カラムフォーマット設定］ダイアログボックスでは、数値について次のフォーマットオプションを定義できます。

- 数値データ
- フィールド長

［カラムフォーマット設定］ダイアログボックスでフォーマットオプションを定義するときに、［カラム］タブの［フォーマット］カラムで定義したオプションが表示されます。

数値データ

［カラムフォーマット設定］ダイアログボックスの数値データ領域で、3 桁ごとの区切り文字と小数点記号を定義できます。小数点記号には、カンマまたはピリオドを指定できます。デフォルトではピリオドです。3 桁ごとの区切り文字には、区切り文字なし、カンマ、ピリオドを指定できます。デフォルトでは、区切り文字は使用しません。

数値区切り文字を指定するには、［区切り文字のオーバーライド］をクリックし、［小数点記号］フィールドと［桁区切り記号（t, s)］フィールドで区切り文字を選択します。区切り文字はどちらか一方だけ上書きすることも、両方上書きすることもできます。区切り文字を上書きした場合は、それぞれ異なるオプションを選択する必要があります。

例えば、ソースデータに以下のデータの数値フィールドが含まれているとします。

```
9.999.999,00
5.000.000,00
```

フラットファイルソース定義で、3 桁ごとの区切り文字にピリオドを選択し、小数点記号にカンマを選択します。

例えば、上のデータを次のフォーマットでファイルターゲットに出力するとします。

```
9,999,999.00
```

フラットファイルターゲット定義で、3 桁ごとの区切り文字にカンマを選択し、小数点記号にピリオドを選択します。

フィールド長

[カラムフォーマット設定]ダイアログボックスで、フィールド長にパディングを加えたり、固定長のバイト値を定義したりすることで、フィールド長を変更することができます。デフォルトでは、フィールド長と精度は同じ値を持ちます。

数値データを固定長フラットファイルに出力したい場合には、ターゲットフィールドの合計長さに対応できるようにターゲットフィールドのフィールド長を設定する必要があります。ターゲットフィールドのデータがフィールド長よりも大きい場合、はその行をリジェクトしてセッションログにメッセージを書き込みます。フラットファイルターゲット定義のフィールド長を設定する場合、小数点記号や負の記号など、がターゲットファイルに書き込む文字を考慮する必要があります。

フィールド長を調整するには、[幅の調節]を選択し、[パディング]フィールドにバイト数を入力します。フィールド長を調整すると、はフィールド精度にユーザーの入力したパディングを加えた値として、フィールド長を定義します。例えばフィールドの精度が 10 で、[パディング]フィールドに 5 を入力した場合、はファイルソースから 15 バイトを読み込み、ファイルターゲットに 15 バイトを書き込みます。

フィールド長を固定するには、[固定長]を選択し、[フィールドの幅]フィールドにバイト数を入力します。Designer では、フィールド長に精度以上の値を入力できます。[フィールド長]フィールドに 20 を入力すると、はファイルソースから 20 バイトを読み込み、ファイルターゲットに 20 バイトを書き込みます。

例えば、精度 4、位取り 0 のターゲットフィールドがあるとします。ターゲットファイルの内容をわかりやすくするために、ターゲットフィールドに 2 つの空白スペースを追加するとします。[幅の調節]を選択し、[パディング]フィールドに 2 を入力します。または、[固定長]を選択して[フィールドの幅]フィールドに 6 を入力します。

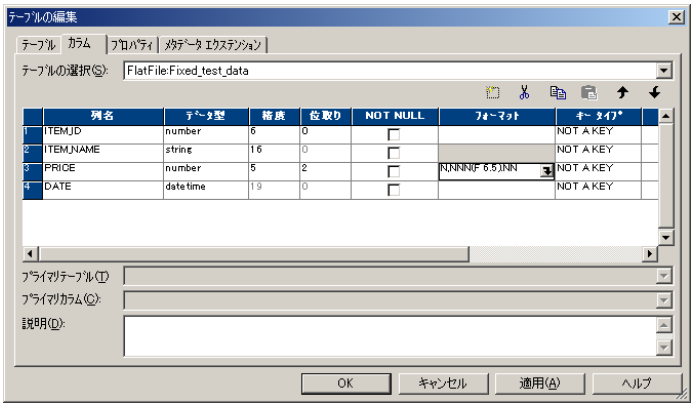
フラットファイルターゲットで数値データをパディングすると、はターゲットカラムの左側に空白スペースを追加します。

注: 他のユーザーがフィールドの精度を変更すると考えられる場合には、ターゲットフィールドのフィールド長を調整することができます。精度が変更されると、フィールド長もそれに応じて調整されます。

フォーマットカラム

数値の区切り文字を上書きしたり、フィールド長を定義したりするときに、[カラム]タブの[フォーマット]カラムで定義したオプションが表示されます。例えば、精度が 5 の数値カラムがあるとします。[フォーマット]カラムをクリックして、[カラムフォーマット設定]ダイアログボックスのオプションを定義します。

以下の図に、Designer の[フォーマット]カラムに表示されるフォーマットオプションを示します。



「PRICE」カラムの[フォーマット]カラムには、「N,NNN（F 6.5）.NN」と表示されます。「N,NNN」には、指定した 3 桁ごとの区切り文字が表示されます。 「F」は固定フィールド長を示します。 フィールド長を調整すると「A」が表示される。フィールド長と精度がバイト単位で表示される。最初の数字がフィールド長、次の数字が精度を表す。「.NN」には、指定した小数位の区切り文字が表示されます。

注: 特定のソースフィールドまたはターゲットフィールドで小数点記号と 3 桁ごとの区切り文字を定義しなかった場合、は Mapping Designer のソースインスタンスまたはターゲットインスタンスで指定した区切り文字を使用します。

関連項目：

- 「デフォルトの日付フォーマットと数値フォーマットの定義」 (ページ 89)

日時カラムのフォーマット

フラットファイル定義を編集する際に、日付の値のフォーマットを定義できます。フラットファイルソース定義またはターゲット定義の［カラム］タブの［フォーマット］カラムを使って、日付の値のフォーマットを定義します。

［カラムフォーマット設定］ダイアログボックスでは、日付値について次のフォーマットオプションを定義できます。

- フォーマット文字列
- フィールド長

［カラムフォーマット設定］ダイアログボックスでフォーマットオプションを定義するときに、［カラム］タブの［フォーマット］カラムで定義したオプションが表示されます。

フォーマット文字列

［カラムフォーマット設定］ダイアログボックスの［フォーマット文字列］フィールドに、任意の日付フォーマットを入力できます。例えば「MM/YYYY」という日付フォーマットを指定できます。あるいは、「HH24:MI」のように、時刻だけを指定することもできます。

日付フォーマットを指定するには、［フォーマット文字列］を選択してから、［フォーマット文字列］フィールドにフォーマットを入力します。リストからフォーマットを選択するか、キーボードを使ってフォーマットを入力します。デフォルトのフォーマットは「MM/DD/YYYY HH24:MI:SS」、フィールド長は 19 バイトになります。

例えば、ソースデータには以下のデータの日付フィールドが含まれているとします。

```
11/28/2002
10/15/2003
```

フラットファイルソース定義で、フォーマット「MM/DD/YYYY」を入力します。

例えば、上のデータを次のフォーマットでファイルターゲットに出力するとします。

```
28-11-2002
15-10-2003
```

フラットファイルターゲット定義で、フォーマット「DD-MM-YYYY」を入力します。

［フォーマット文字列］フィールドには、1 バイトまたはマルチバイトの任意の文字列リテラルを入力することもできます。文字列リテラルを入力する場合は、二重引用符（メモ）で囲みます。フォーマット文字列に文字列リテラルを入力すると、はセッション実行時にその文字列をファイルターゲットに書き込みます。日付を構成する各要素が何を意味しているかを表す文字列リテラルを追加することがあります。

例えば、［フォーマット文字列］フィールドに次のテキストを入力するとします。

```
“Month”MM/”Day”DD/”Year”YYYY
```

セッションが実行され、が 2002 年 10 月 21 日という日付を出力すると、ターゲットファイルには以下の文字列が書き込まれます。

```
Month10/Day21/Year2002
```

フィールド長

フォーマット文字列を定義した後に、フィールド長を定義できます。フィールド長はパディングを設定したり、固定長値をバイト単位で定義したりすることで変更できます。デフォルトでは、フィールド長と精度は同じ値を持ちます。

フォーマット文字列を入力した後でフィールド長を調整するには、[幅の調節] を選択し、[パディング] フィールドにバイト数を入力します。フィールド長を調整すると、はフィールド長を、日付フォーマットで必要なバイト数にユーザーが入力したパディングを加えた値に定義します。例えば、日付フォーマットが「MM/YYYY」で、フラットファイルソースの [パディング] フィールドに 5 を入力した場合、はファイルから 12 バイトを読み込みます。日付フォーマットが「MM/YYYY」で、フラットファイルターゲットの [パディング] フィールドに 5 を入力した場合、はファイルに 12 バイトを書き込みます。

[幅の調節] を使用するとはフォーマット文字列に基づきフィールド長を調整します。つまり、手動でフィールド長を調整せずにフォーマット文字列を変更できます。

フォーマット文字列を入力した後でフィールド長を固定するには、[固定長] を選択し、[フィールドの幅] フィールドにバイト数を入力します。固定長には、日付フォーマットに必要なバイト数以上の値を指定する必要があります。このような値を指定しないと、Integration Service によってデータが切り捨てられます。例えば、日付フォーマットが「MM/DD/YYYY HH24:MI:SS.NS」の場合、固定長値は 29 バイト以上を指定します。[フィールド長] フィールドに 21 を入力すると、はファイルソースから 21 バイトを読み込み、ファイルターゲットに 21 バイトを書き込みます。

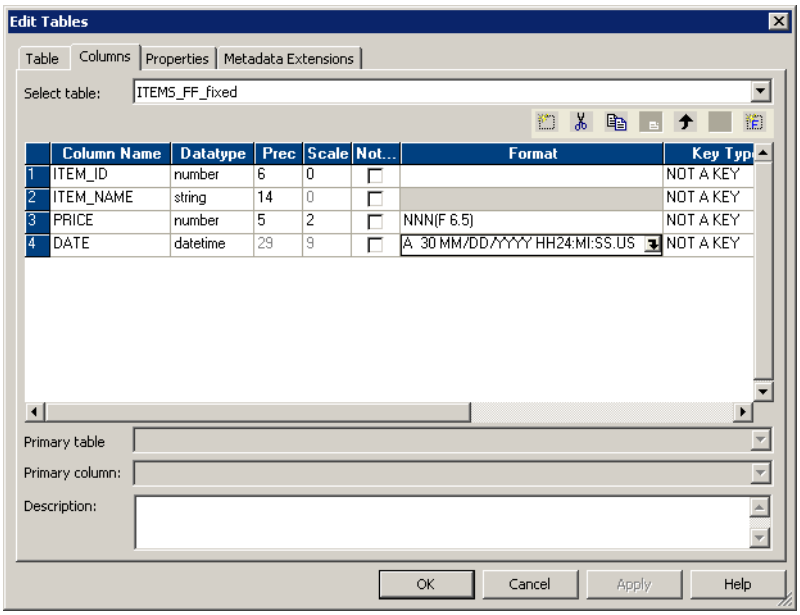
例えば、10 バイトが必要な「MM/DD/YYYY」が日付フォーマットとして設定されているターゲットファイルがあるとします。ターゲットファイルの内容をわかりやすくするために、ターゲットフィールドに 2 つの空白スペースを追加するとします。[幅の調節] を選択し、[パディング] フィールドに 2 を入力します。または、[固定長] を選択して [フィールドの幅] フィールドに 12 を入力します。

注: フラットファイルターゲットで日付データをパディングすると、はターゲットカラムの右側に空白スペースを追加します。

フォーマットカラム

日付フォーマットを選択したりフィールド長を定義したりするときに、[カラム] タブの [フォーマット] カラムで定義したオプションが表示されます。例えば、[カラムフォーマット設定] ダイアログボックスのオプションを定義します。

以下の図に、Designer の［フォーマット］カラムに表示されるオプションを示します。



Designer では、「DATE」ポートの［フォーマット］カラムに「A 30 MM/DD/YYYY HH24:MI:SS.US」と表示されます。「A」はフィールド長が調整されていることを示しています。「30」はバイト単位のフィールド数です。マイクロ秒単位の精度に 26 バイト、パディングに 4 バイトとなります。

注: ソースフィールドまたはターゲットフィールドで日付フォーマットを定義しなかった場合、はマッピングのソースインスタンスまたはターゲットインスタンスに指定された日付フォーマットを使用します。

デフォルトの日付フォーマットと数値フォーマットの定義

マッピングでフラットファイルソース定義やフラットファイルターゲット定義を使用する場合、ファイルの日付カラムや数値カラムにデフォルトのフォーマットを定義できます。個別のカラムのフォーマットを定義しなかった場合、はユーザーが定義したデフォルトフォーマットを使用します。

デフォルトのフォーマットを定義しなかった場合、は以下のフォーマットを使用します。

フォーマット	説明
日時フォーマット	日時フォーマット。デフォルトは MM/DD/YYYY HH24:MI:SS です。
1000 ごとの区切り	1000 ごとの区切り。区切り文字なし、カンマ、またはピリオドを選択します。デフォルトでは、区切り文字は使用しません。
小数位の区切り文字	小数位の区切り文字。カンマまたはピリオドを選択します。デフォルトではピリオドが使用されます。

注: セッションのプロパティの［マッピング］タブで、各ソースインスタンスまたはターゲットインスタンスのデフォルトのフォーマットを確認できます。

ファイルリストに関する作業

マッピングで 1 つのソースインスタンスに複数のソースファイルを読み込むセッションを作成できます。たとえば組織で複数の場所にデータが蓄積されていて、同じマッピングでそのデータを処理したいといった場合に、ファイルリストを作成できます。ファイルリストは、Integration Service で使用する各ソースファイルの名前とディレクトリを格納するファイルです。

セッションでファイルリストを読み込むように設定します。ファイルリストを読み込むセッションを設定すると、Integration Service はファイルリストの異なるソースファイルからデータの行を読み込みます。各ターゲット行にソースファイル名を書き込むようにマッピングを設定するには、CurrentlyProcessedFileName ポートをフラットファイルソース定義に追加します。Integration Service では、このポートをソースファイル名を返すために使用します。

CurrentlyProcessedFileName ポートは、Source Analyzer のフラットファイルソース定義に追加します。

CurrentlyProcessedFileName ポートを追加するには：

1. Source Analyzer のフラットファイルソース定義を開きます。
2. ［プロパティ］タブをクリックします。
3. ［CurrentlyProcessedFlatFileName ポートの追加］を選択します。

   Designer が CurrentlyProcessedFileName ポートを［カラム］タブの最後のカラムとして追加します。CurrentlyProcessedFileName ポートは既定精度が 256 文字の文字列ポートです。
4. ［カラム］タブをクリックして変更を確認します。

   必要に応じて CurrentlyProcessedFileName ポートの精度を変更できます。
5. CurrentlyProcessedFileName ポートを除去するには、［プロパティ］タブをクリックして ［現在処理されているフラットファイル名ポートを追加］チェックボックスをクリアします。

シフト依存のフラットファイルに関する作業

フラットファイルウィザードで、固定長モードおよび区切りモードのシフト依存のフラットファイルをインポートすることができます。

シフト依存のファイルには、マルチバイト文字と 1 バイト文字が両方とも含まれている場合があります。ファイルには、マルチバイト文字と 1 バイト文字とを区分するシフトインキーとシフトアウトキーが含まれている場合と含まれていない場合があります。

シフトインキーおよびシフトアウトキーによってマルチバイト文字をパーティション化するので、フラットファイルウィザードやは各文字を正しく読み取ることができます。シフトアウトキーは、連続したマルチバイト文字の始まりを示します。シフトインキーは、連続したマルチバイト文字の最後を示します。ファイルソースにシフトキーが含まれていない場合、がそれぞれの文字を正しく読み込めるように、ファイル内の各カラムのシフト状態を定義する必要があります。

注: 1 バイトシフトキーおよびダブルバイトシフトキーを使用します。

SHIFT キーを含むフラットファイルのインポート

フラットファイルウィザードを使用して、シフトキーを含むファイルソースをインポートします。シフトキーを含む固定長ファイルと区切りファイルのどちらもインポートできます。フラットファイルウィザードとでは、ファイル中のシフトキーを使ってソース内の各カラムのシフト状態を判断します。

フラットファイルウィザードおよび Integration Service では、連続したシフト文字を処理することができます。

次に示すのは、シフト依存のフラットファイルから得た有効な行の例です。

```
aaa-oAAA-i-oAAA-iaaaaa
```

次の表に、この例で使われている表記の説明を示します。

表記	説明
a	シングルバイト文字
A	マルチバイト文字
-o	シフトアウト文字
-i	シフトイン文字

フラットファイルウィザードのウィンドウでは 1 バイトのシフト文字が「.」で表示されます。2 バイトのシフト文字は、ウィンドウに「..」と表示されます。シフトイン文字は、緑色の背景で表示されます。シフトアウト文字は、青色の背景で表示されます。

シフト依存のフラットファイルの要件

シフトイン文字とシフトアウト文字が含まれていても、次の条件を満たしていないシフト依存のフラットファイルを分析すると、Designer はエラーを返します。

- マルチバイト文字はすべて、シフトアウト文字とシフトイン文字で囲む必要があります。1 バイト文字は、シフト文字で囲む必要はありません。
- 1 つの行で最初に出てくるシフト文字は、シフトアウト文字でなければなりません。
- ファイルには、ネストされたシフト文字を含めることはできません。例えば、次のような一連の文字を含むことはできません。

  ```
  -oAA-oAA-iaaa
  ```

- シフトアウト文字を閉じるシフトイン文字は、同じ行の中になければなりません。

1 文字が 2 バイトより大きいマルチバイト文字がファイルに存在する場合、フラットファイルウィザードは固定長オプションを無効にします。ファイル内の各行は、16KB 以下である必要があります。

フラットファイルウィザードは、データを 500 行まで、または 256KB まで読み込みます。ファイルが上記の条件を満たしていない場合、はセッションログにエラーを書き込みます。

Shift キーを含まないフラットファイルのインポート

フラットファイルウィザードを使用して、シフトキーを含まない固定長ファイルソースをインポートします。ただし、がそれぞれの文字を正しく読み込めるよう、ソース定義をインポートした後でファイルソース定義内での各カラムのシフト状態を定義する必要があります。

シフトキーを含まないシフト依存の COBOL ファイルをインポートすることもできます。この操作を行う場合も、COBOL ソース定義の各カラムについてシフト状態を定義する必要があります。

注: ユーザー定義のシフト状態を含むフラットファイルソースを使ってセッションを作成する場合、Workflow Manager のコードページが Designer で選択したコードページと同じものかどうか確認してください。Workflow Manager で異なるソースコードページを選択すると、は Designer で定義したシフトキーを使用しません。

固定長シフト依存ファイルのシフト状態を定義するには：

1. Designer で、フラットファイルソースまたは COBOL ファイルをインポートします。
2. Source Analyzer で、該当するファイルソース定義のタイトルバーをダブルクリックします。
3. ［テーブル］タブの［フラットファイル属性］セクションで、［固定長］を選択し、［詳細設定］をクリックします。

   ［フラットファイル属性の編集-固定長ファイル］ダイアログボックスが表示されます。
4. ［ユーザー定義シフトステート］を選択し、［OK］をクリックします。
5. ［カラム］タブをクリックします。

   ［シフトキー］カラムが表示されます。
6. 各カラムのシフトキーを選択します。

   カラムに 1 バイト文字が含まれる場合には、［Shift-In］を選択します。カラムにマルチバイト文字が含まれる場合には、［Shift-Out］を選択します。
7. ［OK］をクリックします。

固定長ターゲットのマルチバイトデータに関する作業

固定長フラットファイルターゲットにマルチバイトデータをロードする場合は、マルチバイトデータを許容できるように精度を設定します。固定長ファイルは、文字数ベースではなくバイト数ベースです。固定長ターゲットの精度を設定する場合は、ターゲットにロードする文字数ではなく、バイト数を考慮する必要があります。精度が小さくてマルチバイトデータを許容できない場合、は行を拒否ファイルに書き込みます。Integration Service が拒否ファイルに行を書き込むと、セッションログにメッセージを書き込みます。

注: 区切りファイルは文字数ベースなので、マルチバイトデータに対する追加の精度を設定する必要はありません。

フラットファイルのトラブルシューティング

マルチバイトデータを含むフラットファイルターゲットのセッションを実行しましたフラットファイルターゲットに格納されたデータには、一部のマルチバイト文字が含まれていません。

フラットファイルウィザードを使用してフラットファイルターゲットをインポートした時に選択したコードページがソースコードページのスーパーセットでない場合、ターゲットコードページでエンコードされなかった文字は失われる可能性があります。フラットファイルウィザードを使用してフラットファイルターゲットをインポートする時に、ソースコードページのスーパーセットであるコードページを選択してください。

第 4 章

ターゲットに関する作業

この章では、以下の項目について説明します。

ターゲットに関する作業の概要

マッピングを作成する前に、リポジトリのターゲットを定義する必要があります。ターゲット定義のインポートおよびデザインまたは作成および管理を行うには、Target Designer を使用します。ターゲット定義には、カラム名やデータタイプなどのプロパティが含まれています。

ターゲット定義の作成

Target Designer では、以下のタイプのターゲット定義を作成できます。

- 特定のデータベースプラットフォーム向けのリレーショナルターゲットを作成します。ターゲットデータベースへの外部ローダを使用する場合に、リレーショナルターゲット定義を作成します。
- **フラットファイル。**固定長および区切りフラットファイルのターゲット定義を作成します。
- **XML ファイルにデータを出力する XML ターゲット定義を作成できます。**

以下の方法で、ターゲット定義を作成できます。

- **既存ターゲット用の定義のインポート。**リレーショナルターゲットまたはフラットファイルから、ターゲット定義をインポートします。 Target Designer では、フラットファイルウィザードを使ってフラットファイルをインポートします。
- **ソース定義に基づくターゲット定義の作成。**Target Designer にソース定義をドラッグして、ターゲット定義を作成します。
- **トランスフォーメーションまたはマップレットに基づくターゲット定義の作成。**Target Designer にトランスフォーメーションをドラッグして、ターゲット定義を作成します。
- **手動によるターゲット定義の作成。**Target Designer でターゲット定義を作成します。

- **いくつかの関連ターゲット定義の設計。**関連するいくつかのターゲット定義を同時に作成します。Designer のウィザードを使用して、*スキーマ*と呼ばれる全体の関係性とターゲット定義を作成できます。キューブと次元のウィザードは、データウェアハウス設計に関する共通規則に準拠しているため、関連ターゲットの作成が簡単に行えるようになっています。

ターゲットおよびターゲット定義の管理

Target Designer では、ターゲット定義の作成だけでなく、以下のタスクを実行できます。

- ターゲット構造が大きく変更された場合に、ターゲット定義を正しく維持するためにターゲット定義を再インポートできます。
- **ターゲット定義を編集し、コメントまたはキーの関係を追加したり、それらを更新してターゲット定義の変更を反映させたりします。**
- **ターゲットデータベースでのリレーショナルテーブルの作成。**ターゲットテーブルがターゲットデータベースにない場合は、必要な SQL コードを生成および実行し、ターゲット定義に一致するターゲットテーブルを作成することができます。
- **リレーショナルおよびフラットファイルターゲットデータのプレビュー。**Designer で、リレーショナルおよびフラットファイルターゲット定義のデータをプレビューすることができます。
- **ターゲット定義の比較。**2 つのターゲット定義を比較して、違いを特定することができます。

Oracle ターゲット

基本圧縮および OLTP 圧縮を使用する Oracle ターゲットをインポートできます。基本圧縮および OLTP 圧縮を使用する Oracle ターゲットのターゲット定義を手動で作成することもできます。

ターゲットコードページ

マルチバイト文字セットを使用するターゲットを作成できます。セッションを実行する場合、ターゲットコードページはソースコードページのスーパーセットでなければなりません。

ターゲット定義内の特殊文字の処理

Designer では、スラッシュ（/）などの特殊文字を含むテーブル名やフィールド名のターゲット定義をインポート、作成、編集できます。Target Designer を使用してターゲット定義をインポート、作成、または編集する場合、Designer はテーブル名およびフィールド名の特殊文字を保持して、ターゲット定義をリポジトリに保存します。

ただし、特殊文字のあるターゲット定義をマッピングに追加する場合、Designer は特殊文字を残すか、あるいは置き換えます。また、Target Designer のターゲットインスタンスでターゲット更新のオーバーライドを生成する場合、Designer は特殊文字を含むテーブル名またはカラム名を引用符で囲みます。Designer による特殊文字の扱いは、リレーショナルソースと非リレーショナルソースで異なります。

以下の表に、Designer によるリレーショナルターゲットの特殊文字の扱い方を示します。

特殊文字	Target Designer **の動作**	Mapping Designer **の動作**
@#$_	ターゲット定義テーブル名の文字を残します。 ターゲット定義カラム名の文字を残します。	ターゲットインスタンステーブル名の文字を残します。 ターゲットインスタンスカラム名の文字を残します。 ターゲット更新のオーバーライドのテーブル名またはカラム名を囲んでいる引用符を使用しません。
/+-=~`!%^&*()[]{}'; ?,<>| <space>	ターゲット定義テーブル名の文字を残します。 ターゲット定義カラム名の文字を残します。	ターゲットインスタンステーブル名の文字をアンダースコアに置き換えます。 ターゲットインスタンスカラム名の文字を残します。 ターゲット更新のオーバーライドの、特殊文字を含むテーブル名とカラム名を引用符で区切ります。
.":\t\r\n	Designer はリレーショナルターゲットテーブル名またはカラム名にあるこれらの文字を認識しません。	Designer はリレーショナルターゲットテーブル名またはカラム名にあるこれらの文字を認識しません。

以下の表に、Designer による非リレーショナルターゲットの特殊文字の扱い方を示します。

特殊文字	Target Designer **の動作**	Mapping Designer **の動作**
@#$_	ターゲット定義テーブル名の文字を残します。 ターゲット定義カラム名の文字を残します。 **注**: 注：@文字はテーブル名またはカラム名の先頭文字としては使えません。	ターゲットインスタンステーブル名の文字を残します。 ターゲットインスタンスカラム名の文字を残します。 **注**: 注：@文字はテーブル名またはカラム名の先頭文字としては使えません。
/	ターゲット定義テーブル名の文字を残します。 ターゲット定義カラム名の文字を残します。	ターゲットインスタンステーブル名の文字をアンダースコアに置き換えます。 ターゲットインスタンスカラム名の文字を残します。
.+-=~`!%^&*()[]{}'" ;:?,<>| \t\r\n <space>	Designer は非リレーショナルターゲットテーブル名またはカラム名にあるこれらの文字を認識しません。	Designer は非リレーショナルターゲットテーブル名またはカラム名にあるこれらの文字を認識しません。

特殊文字を含むテーブル名やフィールド名を処理するために特殊な設定やコマンドが必要なデータベースもあります。詳細については、データベースのマニュアルを参照してください。

ターゲット定義のインポート

次のターゲット定義をインポートできます。

- Target Designer では、フラットファイルウィザードを使用して、フラットファイルの構造体と一致するフラットファイルからターゲット定義をインポートします。
- リレーショナルテーブルをインポートして、リレーショナルテーブルの構造に一致するターゲット定義を作成できます。
- XML、DTD、または XML スキーマファイルから、XML ターゲット定義をインポートすることができます。

リレーショナルターゲット定義をインポートするには、ターゲットデータベースと PowerCenter クライアント間の接続を設定する必要があります。

マッピングのターゲット定義は、リポジトリに追加してから使用します。

リレーショナルターゲット定義

リレーショナルテーブルからターゲット定義をインポートすると、Designer は以下のターゲット詳細をインポートします。

- **ターゲット名。**ターゲットの名前。
- **データベースの場所。**リレーショナルソースをインポートする際に、データベースの場所を指定します。Target Designer でターゲット定義を編集する際や、セッションを設定する際に、他の場所を指定できます。
- **カラム名。**カラムの名前。
- **データ型。**Designer は、各カラムのネイティブデータ型をインポートします。
- **キー制約。**ターゲット定義の制約は、きわめて重要な場合があります。というのは、がワークフローの間に制約違反を犯した場合、この制約によってターゲット内へのデータの転送ができなくなるためです。例えば、あるカラムに NOT NULL 制約が含まれていて、このカラムにデータをマッピングしない場合には、はレコードを新たにターゲットテーブルに挿入することはできません。
- **キー関係。**Target Designer をカスタマイズすると、プライマリキーと外部キーの関係を作成できます。［ツール］-［オプション］をクリックして、［フォーマット］タブを選択します。［プライマリキーのインポートおよび外部キーのインポート］にチェックを付けます。

  リポジトリに論理的関係を作成することもできます。キー関係はデータベースに存在する必要はありません。

ターゲット定義をインポートする場合、Designer はターゲットインデックスをインポートしません。このデフォルト設定は powrmart.ini で変更できます。powrmart.ini は、PowerCenter クライアントのインストールディレクトリのルートディレクトリに置かれています。

ターゲット定義インデックスをインポートするには、powrmart.ini ［Main］ セクションに以下のテキストを追加します。

```
ImportIndexes=Yes
```

注: ビューには複数のテーブルからのカラムを含めることができるので、がデータを挿入、更新、削除する際にデータベースエラーが発生する場合があります。ターゲットビューをインポートする場合は、それが単一のテーブルのビューであることを確認します。

リレーショナルターゲットのための接続

リレーショナルターゲット定義をインポートするには、正しく設定された ODBC データソースまたはゲートウェイを使用して、クライアントマシンからデータベースに接続できる必要があります。また、データベースオブジェクトについて読み込み権限が必要な場合もあります。

ODBC データソースを作成する際には、ODBC Driver Manager からデータベース呼び出しが送られるドライバも指定する必要があります。以下の表に、各データベースで使用する推奨 ODBC ドライバを示します。

データベース	ODBC ドライバ	データベースクライアントソフトウェアが必要
IBM DB2	IBM ODBC ドライバ	○
Informix	DataDirect ODBC Wire Protocol driver	×
Microsoft Access	Microsoft Access ドライバ	○
Microsoft Excel	Microsoft Excel ドライバ	○
Microsoft SQL Server	DataDirect SQL Server Wire Protocol ODBC ドライバ	×
SAP HANA	SAP HANA ODBC ドライバ	○
Oracle	DataDirect 32 ビットクローズド ODBC ドライバ	×
Sybase ASE	DataDirect 32 ビットクローズド ODBC ドライバ	×
Teradata	Teradata ODBC ドライバ	該当なし

サードパーティの ODBC データソースを使用してターゲット定義をインポートすると、サードパーティのドライバが powrmart.ini のリストにないというメッセージが Designer に表示される場合があります。Designer は、PowerCenter に付属のドライバを使用してターゲット定義メタデータのインポートを試みます。メタデータをインポートするサードパーティ製のドライバがある場合は、powrmart.ini を設定します。

例えば、Vendor A が vendoraodbc.dll という名前のドライバを提供している場合、ODBCDLL セクションに指定のデータベースに基づいてエントリを追加します。

```
Vendor A = pmodbc.dll
Vendor A = extodbc.dll
```

この例では、Designer は ODBC データソースを使用するために、システム ODBC ドライバと直接対話します。システム ODBC ドライバは、サードパーティの ODBC ドライバ vendoraodbc.dll と内部的に対話します。

次の表に、PMODBC.ini システム ODBC ドライバで使用するエントリ例をデータベース別に示します。

データベース	エントリ
MySQL	MYSQL = PMODBC.DLL
PowerExchange	PWX = PMODBC.DLL
dBase 4	dBASE IV = PMODBC.DLL
Visual FoxPro	Visual FoxPro = PMODBC.DLL
Sybase IQ	Adaptive Server IQ = PMODBC.DLL
Lotus Notes	Lotus Notes = PMODBC.DLL

SAP HANA ターゲットの一括更新/挿入

SAP HANA ターゲットにデータを更新/挿入する場合は、EnableArrayUpsert カスタムプロパティを構成してデータを一括で更新/挿入し、セッションのパフォーマンスを向上させることができます。

EnableArrayUpsert カスタムプロパティはセッションレベルまたは PowerCenter 統合サービスレベルで構成し、その値を yes に設定できます。

サードパーティの ODBC データソースの設定

PowerCenter は、サードパーティの ODBC ドライバの認識、サードパーティの ODBC ドライバでの ODBC データベースオブジェクトのインポート、および Designer でのデータのプレビューに powrmart.ini ファイルを使用します。powrmart.ini ファイルに含まれていない ODBC ドライバでターゲット定義をインポートするには、PowerCenter クライアントマシン上でこのファイルを設定する必要があります。

1. 次のディレクトリから powrmart.ini を開きます。

   `<Informatica installation directory>\clients\PowerCenterClient\client\bin`

2. このファイルの ODBCDLL セクションにエントリを追加し、ODBC データソースの名前を指定します。

   このエントリは、`pmodbc.dll` または `extodbc.dll` を直接指している必要があります。

3. powrmart.ini を保存して閉じます。
4. PowerCenter クライアントを再起動して、ターゲット定義をインポートします。

リレーショナルターゲット定義のインポート

リレーショナルターゲット定義を作成するには、Target Designer を使用してターゲットメタデータをインポートします。

リレーショナルゲット定義をインポートするには：

1. Target Designer で［ターゲット］>［データベースからインポート］をクリックします。
2. ターゲットデータベースへの接続に使用する ODBC データソースを選択します。

   最初に ODBC データソースを作成または変更する必要がある場合は、［参照］をクリックして ODBC データソースアドミニストレータを開きます。ODBC ソースの作成や変更がすんだら、以下の手順に進みます。

3. データベースへの接続を確立するのに必要なユーザー名およびパスワードを入力し、［接続］をクリックします。

   ユーザーがターゲットとして使用したいテーブルのオーナでない場合は、オーナ名を指定します。

4. データベースオブジェクトのリストをドリルダウンし、ターゲットとして使用できるテーブルを表示します。
5. リレーショナルテーブルを 1 つ以上選択し、定義をリポジトリ内にインポートします。

   Shift キーを押しながら、連続した複数のテーブルを選択できます。または Ctrl キーを押しながら、連続しない複数のテーブルを選択できます。また、［すべて選択］と［選択の解除］ボタンを用いて、使用可能なターゲットをすべて選択したり、すべての選択を解除したりできます。

6. ［OK］をクリックします。

   選択したターゲット定義が、ナビゲータの［ターゲット］アイコン下に表示されます。

ソース定義からのターゲット定義の作成

ソース定義に正確に一致したターゲット定義を作成したい場合は、ソース定義を使用してターゲット定義を作成できます。また、ソース定義へのショートカットを用いて、ターゲット定義を作成することもできます。以下のソース定義を Target Designer にドラッグして、ターゲット定義を作成できます。

- リレーショナルソース
- フラットファイルソース
- COBOL ソース
- XML ソース

一致するターゲット定義を作成した後で、ターゲットのプロパティの追加と編集、ターゲットタイプの変更を行うことができます。リレーショナルターゲット定義を作成する場合は、ターゲットデータベースにターゲットテーブルを作成できます。

リレーショナルソースからのターゲット定義の作成

リレーショナルソース定義を Target Designer のワークスペースにドラッグすると、Designer はソース定義に一致するリレーショナルターゲット定義を作成します。

定義を編集し、説明、カラム、データタイプ、ターゲットタイプなどの情報を変更することができます。

フラットファイルソースからのターゲット定義の作成

フラットファイルソース定義を Target Designer のワークスペースにドラッグすると、デフォルトでは Target Designer がそのソース定義に一致するフラットファイルターゲット定義を作成します。

フラットファイルソース定義からフラットファイルターゲット定義を作成すると、Designer はそのフラットファイルソース定義のコードページを使用します。

定義を編集し、説明、カラム、データタイプ、ターゲットタイプなどの情報を変更することができます。

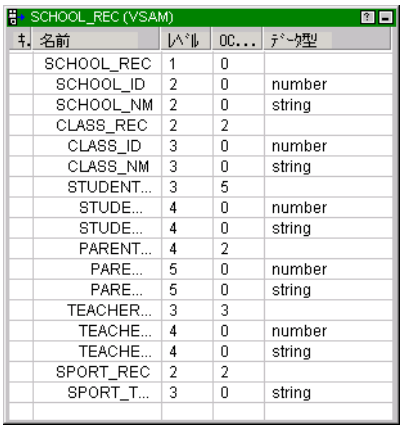
COBOL ソースからの正規化ターゲットの作成

正規化 COBOL ソースに基づいてターゲットを作成するには、まず Source Analyzer を使って COBOL 構造を分析する必要があります。

Target Designer のワークスペースに正規化された COBOL ソース定義をドラッグすると、Target Designer が以下の規則に基づいてリレーショナルターゲット定義を作成します。

- 表示されるテーブルの数は、COBOL ファイル中の OCCURS 文の数より 1 つ多くなります。
- デフォルトのターゲットテーブル名はレコード名です。
- 各テーブルに対して生成されるキー名は「GK_ターゲットテーブル名」となります。
- 生成されるキー名の数は、OCCURS 文の数からプライマリキー数を差し引いたものです。

以下の図に、5 つの OCCURS 文を含む COBOL ソース定義のサンプルを示します。



SCHOOL_REC (VSAM)

キ	名前	レベル	OC...	データ型
	SCHOOL_REC	1	0	
	SCHOOL_ID	2	0	number
	SCHOOL_NM	2	0	string
	CLASS_REC	2	2	
	CLASS_ID	3	0	number
	CLASS_NM	3	0	string
	STUDENT...	3	5	
	STUDE...	4	0	number
	STUDE...	4	0	string
	PARENT...	4	2	
	PARE...	5	0	number
	PARE...	5	0	string
	TEACHER...	3	3	
	TEACHE...	4	0	number
	TEACHE...	4	0	string
	SPORT_REC	2	2	
	SPORT_T...	3	0	string

ソースを Target Designer のワークスペースにドラッグすると、Designer は 6 つのターゲット定義を作成します。

ソース定義からターゲット定義を作成する手順

ソース定義から ターゲット定義を作成するには、次の手順に従います。

ソース定義に基づいてターゲット定義を作成するには：

1. Target Designer ツールを起動し、使用するソース定義をワークスペースにドラッグします。XML ソースの場合は、リレーショナルターゲットまたは XML ターゲットを作成するオプションを選択し、［OK］をクリックします。

   ターゲット定義が表示されます。

2. ターゲット定義を編集するには、該当するタイトルバーをダブルクリックします。
3. ターゲット名を入力し、ターゲットタイプを選択します。カラムやターゲットプロパティの追加または編集を行ってから、［OK］をクリックします。

これでターゲット定義がマッピングで使用できるようになります。リレーショナルターゲット定義に基づいて、ターゲットデータベースにターゲットテーブルを作成することもできます。

トランスフォーメーションからのターゲット定義の作成

リポジトリ内のトランスフォーメーションにほぼ一致するリレーショナルターゲット定義を作成するには、トランスフォーメーションからターゲットを作成できます。トランスフォーメーションをナビゲータから Target Designer にドラッグするか、または Mapping Designer ワークスペース内のトランスフォーメーションからターゲットを作成します。

以下のタイプのトランスフォーメーションからターゲット定義を作成します。

- 1 つの出力グループを持つトランスフォーメーションから、ターゲット定義を 1 つ作成します。
- **複数グループのトランスフォーメーション。**複数の出力グループを持つトランスフォーメーションから複数のターゲット定義を作成します。
- 1 つのソース修飾子またはパイプラインノーマライザトランスフォーメーションから、ターゲット定義を 1 つ作成します。

- マッピングのマプレットインスタンスから 1 つ以上のターゲット定義を作成します。

トランスフォーメーションからターゲット定義を作成すると、デフォルトではターゲットデータベースタイプはリポジトリデータベースと同じになります。リポジトリでターゲット定義を作成した後で、そのターゲット定義を編集できます。例えば、ターゲットタイプを変更できます。

複数のトランスフォーメーションからのカラムを含むターゲット定義を作成する必要がある場合、各トランスフォーメーションから式トランスフォーメーションやジョイナトランスフォーメーションなどのトランスフォーメーションにポートをコピーできます。そのトランスフォーメーションからターゲット定義を作成できます。

リレーショナルターゲット定義を作成する場合、ターゲットデータベースのテーブルを作成するための SQL を作成および実行する必要があります。

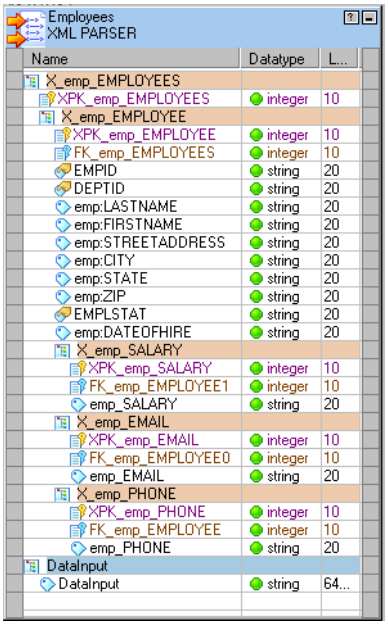
1 つの出力グループを持つトランスフォーメーションからのターゲットの作成

1 つの出力グループを持つトランスフォーメーションからターゲットを作成する場合、Designer は 1 つのターゲットを作成します。ターゲットでは、すべての出力ポートが入力ポートになります。ターゲットの名前は、トランスフォーメーションの名前と同じになります。

複数の出力グループを持つトランスフォーメーションからのターゲットの作成

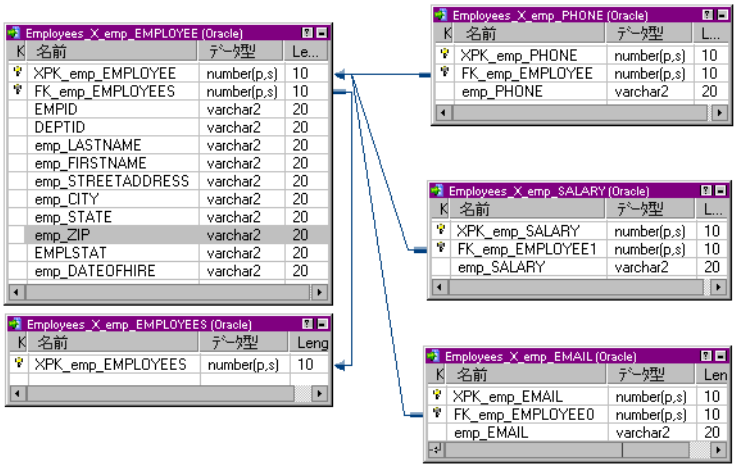
複数の出力グループを持つトランスフォーメーションからターゲットを作成する場合、Designer はトランスフォーメーション内の各出力グループに対して 1 つずつターゲットを作成します。トランスフォーメーションがプラグインまたはカスタムトランスフォーメーションの場合、Designer はターゲット定義のグループ間に存在するプライマリキーと外部キーの関係を保持します。

下記の図は、複数の出力グループを持つトランスフォーメーションを示しています。



Employees XML パーサートランスフォーメーションには、X_emp_EMPLOYEES から X_emp_PHONE まで、5 つの出力グループがあります。 各出力グループは、それぞれ異なるタイプの社員情報を表しています。DataInput は入力グループです。

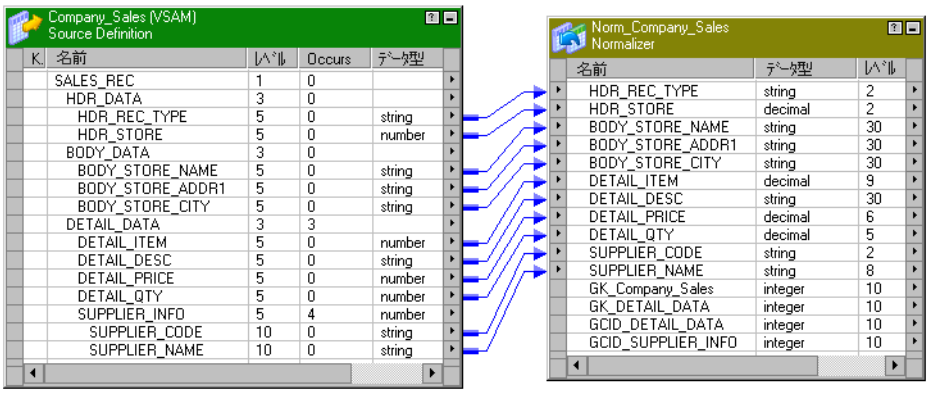
このトランスフォーメーションからターゲットを作成すると、Designer は各出力グループに対して個別のターゲットを作成します。下記の図に示されるとおり、各ターゲットにはトランスフォーメーショングループに由来するグループ名があります。



ノーマライザトランスフォーメーションからのターゲットの作成

1 つのソース修飾子またはパイプラインノーマライザトランスフォーメーションから、ターゲットを 1 つ作成できます。ノーマライザトランスフォーメーションからターゲットを作成する場合、Designer はノーマライザからのすべてのカラムを含む 1 つのターゲットを作成します。ノーマライザトランスフォーメーションのレコード階層または複数出現フィールドを表す個別ターゲットは作成しません。

下記の図は、COBOL ソース定義を持つマッピングのノーマライザトランスフォーメーションを示しています。



ノーマライザトランスフォーメーション Norm_Company_Sales は、Company_Sales COBOL ソース定義の階層データ構造を表します。 Norm_Company_Sales トランスフォーメーションからターゲットを作成すると、Designer はトランスフォーメーション内の階層を 1 つのターゲットにフラット化します。ターゲットには、ノーマライザトランスフォーメーションからの生成されたキー、および DETAIL_DATA と SUPPLIER_INFO の複数出現レコードのために生成されたカラム ID (GCID)が格納されています。

マップレットからのターゲットの作成

マッピングトランスフォーメーションインスタンスノードにあるマプレットからターゲットを作成できます。Target Designer にマプレットインスタンスをドラッグすると、Designer はマプレット内の各出力グループに対してターゲットを作成します。

注: マプレットから Target Designer にトランスフォーメーションインスタンスをドラッグした場合、ターゲットは作成できません。

トランスフォーメーションとターゲットのデータタイプ

Designer は、トランスフォーメーションのデータタイプとリポジトリデータベースの間のデータタイプに最適となるようにターゲットデータタイプを作成します。

以下の表に、トランスフォーメーションデータタイプと、各データベースに対応するデータタイプを示します。

トランスフォーメーションデータ型	IBM DB2	Microsoft SQL Server	Oracle	Sybase ASE	Teradata	Informix
bigint	bigint	bigint	number (19,0)	bigint	bigint	int8
binary	char for bit data	binary	raw	binary	byte/varbyte	byte
date/time	timestamp	datetime	timestamp	datetime	timestamp	datetime year to fraction
decimal	decimal	decimal	number(p,s)	decimal	decimal	decimal(p,s)
倍精度浮動小数点数型	浮動小数点数型	浮動小数点数型	number	浮動小数点数型	浮動小数点数型	float(p)
integer	integer	整数型	number(p,s)	整数型	integer	integer
nstring	vargraphic	nvarchar	nvarchar2	nvarchar	なし	nvarchar
ntext	long vargraphic	ntext	nclob	nvarchar	なし	なし
real	浮動小数点数型	real	number	real	なし	smallfloat
small integer	smallint	smallint	smallint	smallint	smallint	smallint
string	varchar	varchar	varchar2	varchar	varchar	varchar(m,r)
テキスト	long varchar	テキスト	長整数型	テキスト	なし	テキスト

ターゲットの作成手順

１つ以上のトランスフォーメーションをナビゲータから Target Designer にドラッグすると、ターゲットを作成できます。また、Mapping Designer 内のトランスフォーメーションインスタンスからも、ターゲットを作成できます。

Target Designer におけるターゲットの作成手順

ナビゲータ内で選択したトランスフォーメーションを Target Designer のワークスペースにドラッグすると、Target Designer 内に１つ以上のターゲットを作成できます。

別のフォルダのトランスフォーメーションからターゲットを作成する場合、Designer はターゲットフォルダにトランスフォーメーションをコピーし、そのトランスフォーメーションからターゲットを作成します。共有フォルダ内のトランスフォーメーションからターゲットを作成する場合、Designer はターゲットフォルダ内にトランスフォーメーションへのショートカットを作成し、その後でトランスフォーメーションを作成します。

以下の表に、Target Designer でターゲットの作成に使用できるオブジェクトの一覧を示します。

オブジェクト	ナビゲータの場所
トランスフォーメーション	トランスフォーメーションノード
トランスフォーメーションインスタンス	マッピングノード
マプレット	マッピングのトランスフォーメーションインスタンスノード

Target Designer でターゲット定義を作成するには、以下のとおりに実行します。

1. Target Designer を開きます。
2. ナビゲータ内のトランスフォーメーションノードまたはトランスフォーメーションインスタンスノードからワークスペースに、トランスフォーメーションをドラッグします。

   ターゲット定義が表示されます。

Designer が、ナビゲータおよび Target Designer のワークスペースに新規ターゲットを追加します。リポジトリでバージョン管理を使用している場合、デフォルトでは新規ターゲットはチェックアウトされます。

新規ターゲットの名前が既存ターゲットの名前と重複する場合、Designer は新規ターゲットの名前を変更するか既存のターゲット定義を上書きするように指示します。

Mapping Designer におけるターゲットの作成手順

Mapping Designer のトランスフォーメーションインスタンスからターゲットを作成できます。Mapping Designer でターゲットを作成する場合、マッピング内にターゲットインスタンスを作成します。Target Designer は、ターゲット定義を表示します。

Mapping Designer でターゲット定義を作成するには、以下のとおりに実行します。

1. Mapping Designer でマッピングを開きます。
2. マッピング内のトランスフォーメーションインスタンスを右クリックします。
3. ［ターゲットの作成および追加］をクリックします。

リポジトリに同じ名前のターゲット定義が格納されている場合、Mapping Designer ではなく Target Designer でターゲットを作成する必要があります。

重複する名前が存在しない場合、Designer がナビゲータおよび Mapping Designer のワークスペースに新規ターゲットを追加します。トランスフォーメーションポートをターゲットにリンクできます。

ターゲット定義の手動での作成

ターゲット定義をインポートしたり、ソース定義からターゲット定義を作成したりする代わりに、手動でターゲット定義を作成できます。

手動でターゲット定義を作成するには：

1. Target Designer で［ターゲット］-［作成］をクリックします。
2. ターゲットの名前を入力し、ターゲットタイプを入力します。

   リレーショナル定義を作成する場合、データベース固有の命名規則に従ってください。
3. ［作成］をクリックします。

   空の定義がワークスペースに表示されます。これはダイアログボックスに表示されることもあります。ナビゲータウィンドウ内にも、新しいターゲット定義が表示されます。
4. さらに別のターゲット定義を作成する場合には、新しいターゲット名とターゲットタイプを入力してから、［作成］をクリックしてください。作成対象の各ターゲットについて、この手順を繰り返します。
5. ターゲット定義の作成が終了したら、［完了］をクリックします。
6. ターゲット定義を設定します。

   新しいターゲット定義がリポジトリに保存されます。これでターゲット定義がマッピングで使用できるようになります。

リレーショナルターゲット定義に基づき、ターゲットデータベースにターゲットテーブルを作成することもできます。

注: XML ファイルのターゲット定義を手動で作成することはできません。

リレーショナルターゲット定義の保持

以下の方法で、リレーショナルターゲット定義を管理できます。

- **ターゲット定義の再インポート。**ターゲットが大幅に変更された場合には、ターゲット定義を編集するのではなく再インポートします。
- **プライマリキーと外部キーの関係の定義。**リレーショナルターゲットテーブル間の、プライマリキーと外部キーの関係を定義します。
- **ターゲット定義の編集。**ターゲット定義を編集し、コメントまたはキーの関係を追加したり、それらを更新して変更したターゲットを反映させます。

リレーショナルターゲット定義を作成すると、以下のターゲット定義タブを使用してターゲット定義を編集できます。

- リレーショナルターゲットの制約や、フラットファイルターゲットのフラットファイルプロパティなどのプロパティを編集します。
- **［カラム］タブ。**データタイプや精度などのカラム情報を編集します。
- **［インデックス］タブ。**リレーショナルターゲット定義のインデックス情報を追加します。
- **ターゲット定義などのリポジトリオブジェクトに情報を関連付けることで、リポジトリに格納されているメタデータを拡張することができます。**

ターゲット定義を変更すると、Designer はそのターゲットを使用しているマッピングすべてに変更を反映します。ターゲット定義への変更の内容によっては、マッピングが無効になる場合もあります。

以下の表に、ターゲット定義の編集がマッピングにどのような影響を与えるかを示します。

変更	結果
カラムの追加	マッピングが無効となることはありません。
カラムのデータタイプの変更	マッピングが無効となることがあります。新しいデータタイプと一致しないデータタイプを使用する入力ポートにカラムを接続した場合（例えば、Decimal から Date）は、マッピングは無効となります。
カラム名の変更	マッピングが無効となることがあります。カラムを追加した直後にそのカラム名を変更した場合は、マッピングは有効のままです。既存のカラムのカラム名を変更した場合は、マッピングは無効になります。
カラムの削除	削除したカラムの値をマッピングが使用すると、マッピングが無効となることがあります。
ターゲット定義のタイプの変更	マッピングが無効となることはありません。

Target Designer でターゲットに新しいカラムを追加しても、そのターゲット定義を使用するマッピングはすべて有効のままです。しかし新しいカラムを追加して、特定のプロパティをいくつか変更した場合は、そのターゲット定義を使用するマッピングは無効になります。

新しく追加したターゲットカラムで、次のプロパティについては変更してもマッピングは無効になりません。

- 名前
- データタイプ
- フォーマット

変更によりマッピングが無効になる場合は、マッピングおよびマッピングを使用するすべてのセッションを検証します。クエリの結果ウィンドウ、依存関係の表示ウィンドウ、またはナビゲータウィンドウからオブジェクトを検証できます。これらの場所から、ワークスペースで複数のオブジェクトを開くことなく検証できます。これらの場所の 1 つからマッピングやセッションを検証できない場合は、ワークスペースでオブジェクトを開き、編集します。

リレーショナルターゲット定義の再インポート

カラムのデータタイプの変更などのためにターゲットテーブルが変化した場合には、ターゲット定義を編集または再インポートすることができます。ターゲットを再インポートする場合、既存のターゲット定義の上書きを行うか、新しいターゲット定義の名前を変更することで、既存のターゲット定義との名前の重複が回避できます。

ターゲット定義を再インポートするには：

1. Target Designer でターゲット定義のインポートと同じ手順を実行し、インポートするターゲットを選択します。

   Designer は、リポジトリ内に同じ名前を持つターゲット定義が既に存在することを通知します。複数のテーブルをインポートして上書きする場合、［すべてのテーブルに適用］にチェックマークを付けます。
2. ［名前の変更］、［置換］、［スキップ］、［比較］のどれかをクリックします。
3. ［名前の変更］をクリックした場合には、ターゲット定義名を入力し、［OK］をクリックします。
4. リレーショナルターゲット定義があり、［上書き］をクリックした場合は、プライマリキーと外部キーの情報およびターゲットの説明を維持したいかどうかを指定します。

次の表に、リレーショナルターゲット定義の再インポートおよび上書きの際に使用できる［テーブル名の重複］ダイアログボックスのオプションの説明を示します。

オプション	説明
すべてのテーブルに適用	このオプションを選択して、フォルダ内のすべてのテーブルに名前の変更、上書き、またはスキップを適用します。
ユーザー定義のプライマリキー - 外部キーの関係を保持する	上書き対象のターゲット定義内にあるプライマリキーと外部キーの関係を維持する場合に、このオプションを選択します。ターゲット定義が非リレーショナルである場合は、このオプションは無効になります。
ユーザー定義の説明を保持する	上書き対象のターゲット定義のターゲットの説明およびカラムやポートの説明を維持する場合に、このオプションを選択します。

プライマリキーと外部キーの関係の作成

2 つのリレーショナルテーブルの関係を作成するには、［レイアウト］メニューから［カラムのリンク］モードを選択します。1 つのテーブルの「外部キー」カラムから、別のテーブルの「プライマリキー」カラムにドラッグします。Designer から、デフォルトのプライマリキーテーブルとの既存のリンクを削除するよう要求されます。

テーブルのオプションの編集

ターゲット定義の［テーブル］タブで、以下のオプションを編集できます。

- ［名前の変更］ボタンを使用してテーブルにわかりやすい名前を追加します。
- テーブルレベルの参照の整合性制約の SQL 文。リレーショナルターゲットのみに適用されます。
- **作成オプション。**テーブルストレージオプション用の SQL 文です。 リレーショナルターゲットのみに適用されます。
- **説明。**コメントやビジネス文書へのリンクを追加します。コメントやリンクは、Repository Manager でターゲットテーブルと共に表示されます。ターゲットにコメントやビジネス文書のリンクを追加することで、ターゲットの用途を簡単に文書化することができます。既存のターゲットにコメントを追加したり、コメントを修正することができます。

  選択したリポジトリのコードページにおける各文字の最大バイト数を K とした場合、説明に 2000/K 文字まで入力できます。例えば、リポジトリのコードページが日本語コードページ（K=2）の場合、説明およびコメントのフィールドにはそれぞれ 1,000 文字まで入ります。
- キーワードを使用してターゲット定義を追跡します。開発作業や保守作業を進めると、ターゲット数が増大します。ターゲットがすべて同一のフォルダに表示されることがありますが、これらのターゲットの目的がそれぞれ異なることもあります。キーワードは関連ターゲットの検索に役立ちます。キーワードに含めることができるのは、開発名、マッピング、または関連スキーマです。

  Repository Manager で、キーワードを使用して検索を実行できます。
- ターゲットタイプをリレーショナルデータベースまたはフラットファイルに定義します。リレーショナルターゲット定義をフラットファイル定義に、あるいはその逆に変更することができます。ターゲット定義のタイプを変更すると、リポジトリに変更を保存する場合に一部のメタデータが失われます。

  ターゲット定義タイプをリレーショナルからフラットファイルに変更すると、インデックス情報、制約情報、作成オプション情報が失われます。このターゲットを使用するすべてのセッションが無効になります。

ターゲット定義タイプをフラットファイルからリレーショナルに変更すると、すべてのフラットファイルプロパティ情報が失われます。ターゲット定義をフラットファイルに戻すと、フラットファイルプロパティにはデフォルト値が使用されます。このターゲットを使用するすべてのセッションが無効になります。

注: 注：ターゲットタイプをフラットファイルからリレーショナルに変更すると、そのターゲットを使用するすべてのセッションが無効になります。ただし、ターゲットタイプをリレーショナルからフラットファイルに変更する場合には、そのターゲットを使用するセッションは無効になりません。

- データベースタイプがフラットファイルの場合、［詳細設定］をクリックするとフラットファイルのプロパティを定義できます。

オプションをリレーショナルターゲット定義に追加するには：

1. Target Designer で、ターゲット定義のタイトルバーをダブルクリックします。

   ［テーブルの編集］ダイアログボックスが表示されます。
2. ［名前の変更］をクリックし、ターゲット名とビジネス名を編集します。
3. ターゲットタイプを変更するには、［データベースタイプ］フィールドで他のデータベースを選択します。

   ターゲットタイプをフラットファイルターゲットに変更するには、［フラットファイル］を選択します。
4. リレーショナルターゲット定義の、次のプロパティを編集します。
   - 制約を追加するには、［制約］フィールドに SQL 文を入力します。
   - 作成オプションを追加するには、［作成オプション］フィールドに SQL 文を入力します。
5. 説明を追加するには、［説明］フィールドに説明を入力します。
6. キーワードを追加するには、［キーワードの編集］をクリックします。

   ［キーワードの編集］ダイアログボックスが表示されます。
7. ボタンを使って、キーワードの作成や移動を行います。
8. ［OK］をクリックします。

カラムの編集

ターゲット定義の［カラム］タブで、以下の情報を編集できます。

- **カラム名。**ターゲット内のカラムの名前。 リレーショナルターゲット定義を編集するとき、リレーショナルターゲット定義を手動で作成する場合、または実際のターゲットカラム名が変更された場合に、カラム名を編集します。
- ターゲット定義に表示されるデータタイプは、ターゲット定義のターゲットタイプによって異なります。
- **精度と位取り。**リレーショナルターゲットの設計やインポートをする場合、各カラム内の値の精度および位取りを検討します。 *精度*は、数値データタイプでは最大有効桁数であり、また文字列データタイプでは最大文字数です。精度には位取りが含まれます。*位取りは、数値の小数点以下の最大桁数です。*したがって、11.47 という値の精度は 4 で、位取りは 2 です。「*Informatica*」という文字列の精度（長さ）は 11 です。

  リレーショナルターゲットのデータタイプにはすべて、最大の精度が決められています。例えば、Integer データタイプの最大精度は 10 です。いくつかの数値データタイプには位取りについても同様の制限があって、0 より大きな値に位取りを設定できません。例えば整数は位取りが 0 ですが、それは定義上、小数点以下の値を含まないからです。

  一部のデータタイプの精度および位取りは、データベースで定義されている値とは異なる値に変更できます。ただし、精度と位取りを変更すると、Integration Service がターゲットカラムに書き込む際、数値カラムで数値データのオーバーフローが起きたり、文字カラムで文字列が切り詰められたり、日付カラムでゼロが挿入されたりすることもあります。
- **非 NULL。**ターゲットで NULL データを許可するかどうかを選択します。

- **キータイプ。**［プライマリキー］、［外部キー］、［プライマリ/外部キー］、または［キー以外］を選択します。 リレーショナルターゲットのみに適用されます。
- **ビジネス名。**必要に応じて、各ターゲットカラムにビジネス名を追加できます。

リレーショナルターゲット定義のカラムを編集するには：

1. Target Designer で、ターゲット定義のタイトルバーをダブルクリックします。
2. ［カラム］タブを選択します。
3. 上記の説明に従って、ターゲット定義のオプションを設定します。
4. ターゲット定義を作成していて、カラムを追加したい場合は、カラムを選択して［追加］をクリックします。
5. カラムの名前、データタイプなどの特性を入力します。

   テーブルに追加する各カラムについて、上記の手順を繰り返します。
6. カラムを移動する場合には、［上に移動］および［下に移動］ボタンを使うか、またはスクローリングリスト内でカラムをドラッグします。
7. ［OK］をクリックします。

インデックスの定義

インデックスがあるとテーブルに対するクエリーの実行速度が向上するため、インデックスをターゲットデータベースに追加することは、ターゲットテーブルの設計での重要な作業です。リレーショナルターゲット定義にインデックス情報を追加できます。データウェアハウスに対するクエリーに応じて、インデックスを付けるべきカラムが決まります。インデックスを定義する場合は、ターゲットテーブルの作成時にインデックスを作成するオプションを選択します。

ターゲットテーブルのインデックスを作成するには：

1. Target Designer で、リレーショナルターゲット定義のタイトルバーをダブルクリックします。
2. ［インデックス］タブを選択します。
3. インデックスを追加するには、［インデックス］セクションで［追加］をクリックします。
4. インデックスの名前を入力して、Enter キーを押します。
5. インデックスにカラムを追加するには、［列］セクションで［追加］をクリックします。カラム名を選択して、［OK］をクリックします。
6. 割り当てるカラムごとに、3 から 5 までの手順を繰り返します。
7. ［OK］をクリックします。

重要：DDL を生成および実行してターゲットテーブルを作成する場合は、索引の作成を選択します。

ターゲットテーブルの作成

リポジトリにリレーショナルターゲット定義を追加すると、Designer を使用して SQL コードを生成、実行し、リレーショナルデータベースにターゲットを作成できます。XML ターゲット定義またはフラットファイルターゲット定義からリレーショナルデータベースにテーブルを作成することはできません。

Designer は、UCS-2 の文字を使用して SQL スクリプトを生成します。

データベースに当該ターゲットが既に存在する場合、そのターゲットを削除して再作成できます。Designer は SQL コードを拡張子.SQL のテキストファイルに書き込むので、このファイルを開いて DDL コマンドの確認と編集ができます。

SQL コードの生成と実行を行うには：

1. Target Designer で、データベースに作成するリレーショナルターゲット定義を選択します。複数のテーブルを作成する場合には、該当するテーブル定義をすべて選択します。
2. ［ターゲット］-［SQL 文の生成/実行］をクリックします。

   ［接続］をクリックし、ターゲットテーブルを作成するデータベースを選択します。［OK］をクリックして接続を行います。

   生成しようとする SQL スクリプトのファイル名と場所、および SQL DDL コードに含めるオプションを入力します。このテキストファイルは、リポジトリではなくローカルファイルシステムに存在します。

   選択した［生成］オプションに基づき、SQL スクリプトには選択内容に一致するすべての CREATE コマンドおよび DROP コマンドが含まれます。例えば、プライマリキーのあるターゲット定義を作成した場合には、プライマリキーのある SQL の生成を選択します。
3. SQL スクリプトを作成する場合には、［SQL ファイルの生成］をクリックします。ファイルを作成した後に直ちに実行する場合には、［生成と実行］をクリックします。

   ［SQL ファイルの生成］をクリックすると、選択したテーブル定義の SQL が生成され、選択したファイルに格納されます。ファイルが既に存在する場合には、既存のファイルに上書きするかどうかを確認するダイアログボックスが表示されます。Designer の出力ウィンドウに、生成される SQL ファイルのコピーの進行状況が表示されます。

   ファイルが作成された後で［SQL ファイルの編集］をクリックすると、テキストエディタが開き、SQL 文の変更を行えます。ターゲットデータベースの SQL ファイルが作成される際に、スラッシュを含むテーブル名やフィールド名はすべて二重引用符で囲まれます。

   ［SQL ファイルの実行］をクリックすると、テーブルを作成できます。［生成と実行］をクリックすると、選択したテーブル定義に対して SQL が生成され、選択したファイルに格納されたあと、すぐに実行されます。

   注: Designer が開いていれば、最後に開いて変更した SQL ファイルがロックされます。ファイルのロックを解除して、別のアプリケーションで表示したい場合には、Designer で別の SQL ファイルを開くか、Designer を終了させます。
4. ［閉じる］をクリックします。

   このダイアログボックスを閉じても、Designer はターゲットデータベースへの接続を維持します。このダイログボックスを再度開く場合には、ターゲットデータベースに再接続する必要はありません。

Designer の SQL DDL コマンド

Designer が SQL コードを生成する場合、プラットフォーム固有の DDL コードではなく、汎用 SQL を使用します。Designer から命令を渡された ODBC Driver Manager は、標準 SQL をプラットフォーム固有のコマンドに変換します。別のユーティリティでこの SQL ファイルを実行したり、その構文をネイティブ DDL 構文の参考に使用したりしないでください。

インデックスの削除と再作成

ある程度大きな量のデータをターゲットに挿入したら、一般に、テーブルのインデックスの削除と再作成を行って、クエリー速度を最適化する必要があります。インデックスの削除と再作成を行うには、以下のいずれかの方法を用います。

- **セッション実行前/実行後 SQL コマンドの使用。**インデックスの削除と再作成を行う場合は、ターゲットにデータをロードする前にインデックスを削除するセッション実行前 SQL 文を実行前 SQL プロパティで定義する方法をお勧めします。ターゲットにデータをロードした後で、実行後 SQL プロパティを使用してインデックスを再作成します。マッピングターゲットのプロパティか、セッションのプロパティの［マッピング］タブで、リレーショナルターゲットのセッション実行前/実行後 SQL を定義します。

- テーブルの作成時に DDL コードの生成と実行に使用したのと同じダイアログボックスを使って、インデックスの削除と再作成ができます。この方法を使う場合、ターゲットテーブルを変更するワークフローを実行するたびに、Designer を起動して、この機能を使用します。
- **ストアドプロシージャ。**ストアドプロシージャを使用してインデックスの削除と再作成をすることもできます。

ターゲットの再作成

リレーショナルターゲット定義を変更する場合、Designer を使用して該当するターゲットテーブルを削除して再作成します。

注: ターゲットテーブルを削除すると、Designer はデータベースからそのテーブルを削除します。ターゲットデータを保持する必要がある場合、テーブルを削除する前にバックアップします。

ターゲットテーブルの再作成を行うには：

1. Target Designer でリレーショナルターゲット定義を変更し、その結果を選択します。
2. ［ターゲット］-［SQL 文の生成/実行］をクリックします。

   ダイアログボックスで、該当するターゲットデータベースに接続します。当該テーブルやそのテーブルのインデックスについてチェックマークを付けた DROP オプションを選択します。
3. ［生成と実行］をクリックします。

   Designer はインデックスを含めて、テーブルの削除と再作成を行います。

ターゲットのトラブルシューティング

ターゲット定義を変更し、Designer を用いて SQL DDL コードを実行した際、ターゲットテーブル中のデータをすべて消失してしまいました。

ターゲット定義の変更の際、Designer はテーブルの削除と再作成を行うことがあります。Designer では、ALTER TABLE コマンドを発行してカラムの変更や追加を行うことはできません。テーブルの変更が必要な場合には、一時的なテーブルにデータのバックアップを取ってから、このテーブルの削除と再作成を行ってください。また、ALTER TABLE コマンドを発行することもできますが、リポジトリに格納したばかりのターゲット定義と一致させる必要があることに十分注意してください。

データベースに接続してターゲット定義のインポートを行う際に、インポートしたいテーブル、ビュー、シノニムが表示されません。

データベースに接続する際に入力したオーナ名が妥当であるか確認してください。デフォルトでは、Designer がインポート用のソースとターゲットを識別するのに用いるオーナ名は、データベースへの接続にあたって使ったデータベースユーザー名と同じです。インポートしたいターゲットを表示するためには、別のオーナ名の入力が必要な場合があります。

ターゲットをワークスペース内にドラッグすると、このターゲットが表示されず、コピーするかショートカットを作成するかを尋ねるメッセージが表示されました。

各ワークブックは、単一のフォルダからのメタデータを表すものです。別のフォルダのメタデータを編集するには、ナビゲータウィンドウ内でそのフォルダにフォーカスを合わせて、［開く］をクリックしてください。Designer は別のワークブックを開いて、当該フォルダのメタデータを表示します。

マッピングに表示されているターゲット定義を開きましたが、その編集ができません。

Mapping Designer でマッピングを作成します。Source Analyzer および Target Designer で、ソース定義とターゲット定義を作成し、変更します。Designer はソース定義、ターゲット定義、マッピングの追加プロセスを別々のモードにパーティション化して、各プロセスを明確に把握できるようにしています。ターゲット定義を変更するには、Target Designer に切り替えます。

作成済みのターゲットを含むワークフローを実行しようとしましたが、1 つ以上のターゲットテーブルが存在しない、とセッションログに表示されました。

ターゲットを作成した際には、リポジトリにターゲット定義が追加されるだけです。ターゲットテーブルを実際に作成するには、ターゲットを表示させたいデータベースで所定の SQL DDL コードを実行してください。

DB2 データベースからターゲットをインポートしたら、DB2 オペレーティングシステムによって SQL0954C エラーメッセージが生成されました。

Designer を使って DB2 データベースからターゲットをインポートする際に、DB2 システム変数 APPLHEAPSZ の値が小さすぎると、リポジトリへのアクセスに関するエラーが出力されます。ステータスバーには以下のメッセージが表示されます。

```
SQL Error:[IBM][CLI Driver][DB2]SQL0954C: Not enough storage is available in the application heap to process the statement.
```

このエラーが表示されたら、DB2 オペレーティングシステムに対する APPLHEAPSZ 変数の値を大きくしてください。APPLHEAPSZ は、データベースを使用している各プロセスに対する 4KB ページ単位のアプリケーションヒープサイズです。

第 5 章

マッピング

この章では、以下の項目について説明します。

マッピングの概要

マッピングとは、データトランスフォーメーションのルールで定義されたトランスフォーメーションオブジェクトにリンクされたソース定義とターゲット定義のセットです。マッピングは、ソースとターゲットの間のデータフローを表します。Integration Service がセッションを実行する際には、マッピングで設定した手順を使用してデータの読み込み、トランスフォーメーション、および書き込みを行います。

各マッピングには以下のコンポーネントを含める必要があります。

- **ソース定義。**ソーステーブルまたはファイルの特性を記述します。
- **トランスフォーメーション。**データをターゲットに書き込む前に変更します。 さまざまなトランスフォーメーションオブジェクトを使用して、種々の関数を実行します。
- **ターゲット定義。**ターゲットテーブルまたはファイルを定義します。
- **リンク。**ソース、ターゲット、およびトランスフォーメーションを接続して、Integration Service がトランスフォーメーション時にデータを移動できるようにします。

マッピングには、1 つ以上のマプレットを含むこともできます。マプレットは、Mapplet Designer で作成する一連のトランスフォーメーションであり、複数のマッピングで使用できます。

オブジェクトをマッピングに追加する際に、Integration Service のデータ変換方法に応じて、プロパティを設定します。 また、Integration Service のデータ移動方法に応じて、マッピングオブジェクトを接続することもできます。 オブジェクトは、ポートを介して接続することができます。

Mapping Designer は、3 つの異なるビューでオブジェクトを表示します。

- **アイコン化。**オブジェクトのアイコンを、オブジェクト名と共に表示します。
- **ノーマル。**［ポート］タブ内のカラムと、入力ポートおよび出力ポートのインジケータを表示します。 ノーマルビュー内のオブジェクトに接続することができます。
- **編集。**オブジェクトのプロパティを表示します。 タブを切り替えて、このビュー内のオブジェクトを設定できます。

オブジェクトの依存関係

マッピング内のオブジェクトの一部は、独立したオブジェクトとしてリポジトリにも格納されています。

- ソース
- ターゲット
- 再利用可能なトランスフォーメーション
- マップレット

マッピングは、これらのオブジェクトに依存しています。メタデータが変更されると、Designer や他の PowerCenter クライアントアプリケーションは、この変更がマッピングにどう影響するかを追跡します。 このような場合、マッピングを編集していなくてもマッピングが無効になることがあります。マッピングが無効になると、Integration Service がマッピングを正しく実行できず、Workflow Manager はセッションを無効にします。

マッピングのオブジェクトで、独立したリポジトリオブジェクトとして格納されない唯一のオブジェクトは、マッピング内でユーザーが作成する再利用不可能なトランスフォーメーションです。再利用不可能なトランスフォーメーションは、マッピング内にのみ格納されます。

マッピングの開発

マッピングを開発する場合、以下の手順をガイドラインとして使用します。

1. **ソース、ターゲット、および再利用可能なオブジェクトがすべて作成されたことの確認。**ソース定義およびターゲット定義を作成します。 マプレットを使用したい場合は、マプレットも作成する必要があります。再利用可能なトランスフォーメーションは、Transformation Developer で作成するか、またはマッピングの開発時に作成することができます。
2. **マッピングの作成。**ソース、ターゲット、マップレット、または再利用可能なトランスフォーメーションを Mapping Designer のワークスペースにドラッグするか、またはメニューで［マッピング］-［作成］の順にクリックしてマッピングを作成します。
3. **ソースとターゲットの追加。**ソースとターゲットをマッピングに追加します。
4. **トランスフォーメーションとトランスフォーメーションロジックの追加。**トランスフォーメーションをマッピングに追加して、トランスフォーメーションロジックをトランスフォーメーションのプロパティに構築します。
5. **マッピングの接続。**マッピングオブジェクトを接続してソースからターゲットへのデータフローを作成します。マップレットやトランスフォーメーションを通じて、このフローに沿ってデータの追加、削除、または修正が行われます。
6. **マッピングの検証。**マッピングを検証して、接続またはトランスフォーメーションのエラーを識別します。

7. **マッピングの保存。**マッピングを保存すると、Designer はそのマッピングを検証し、エラー有無を識別します。 Designer は、出力ウィンドウに検証メッセージを表示します。エラーのあるマッピングは無効であり、問題を解決するまで、このマッピングに基づいてセッションを実行することはできません。

PowerCenter では、PowerCenter マッピングのテンプレートを作成し、作成したテンプレートから複数のマッピングを作成するツールも提供されます。 Mapping Architect for Visio には、PowerCenter マッピングオブジェクトを表現する図形が含まれた、Microsoft Office Visio に対応する Informatica ステンシルが用意されています。 マッピングオブジェクトの図形を使って、Viso 描画ウィンドウでマッピングテンプレートを描きます。

マッピングに関する作業

マッピングを使用して、以下のタスクを実行できます。

- **マッピングの作成。**マッピングを作成する際に、マッピング名をリポジトリに保存します。 次に、そのマッピングを開発し、保存することができます。
- **マッピングを開く。**フォルダ内のマッピングは、一度に 1 つずつ開くことができます。
- **マッピングのコピー。**同じフォルダ内で、または別のフォルダに、マッピングをコピーすることができます。
- **マッピングセグメントのコピー。** マッピングロジックの一部を再利用する場合、マッピングおよびマップレットのセグメントをコピーできます。
- **マッピング内のオブジェクトのコピー。**マッピング内の 1 つ以上のオブジェクトをコピーし、それを別のマッピングや同じフォルダ内のマップレットに貼り付けることができます。
- **マッピングのエクスポート。**マッピングを XML ファイルにエクスポートできます。
- **マッピングのインポート。**Designer でエクスポートした XML ファイルからマッピングをインポートすることができます。
- **マッピングの編集。**マッピング内のオブジェクトを追加、変更、または削除することができます。
- **マッピングの保存。**マッピングをリポジトリに保存した場合、Designer によりマッピングの検証が実行されます。
- **マッピングのデバッグ。**Mapping Designer でデバッガを実行してマッピングロジックをテストします。
- **マッピングの削除。**使用する予定のないマッピングをリポジトリから削除します。
- **ポートへのリンクパスの表示。**マッピングでポートへのリンクパスを表示できます。 前方パス、後方パス、またはその両方を表示できます。
- **ソースカラムの依存関係の表示。**ターゲットカラムがデータを受け取るソースカラムを表示できます。
- **マッピング内のオブジェクトの接続。** マッピング内のオブジェクトを接続して、ソースからターゲットへのデータフローを定義できます。
- **ポートのリンク。** 手動で、または、名前や位置によって自動的にポートをリンクすることにより、マッピングオブジェクトを接続できます。
- **ポート属性のプロパゲート。** マッピングでポート属性をプロパゲートすることができます。属性を前方、後方、または両方向にプロパゲートすることができます。

マッピングの作成

ソースからターゲットにデータを移動するプロセスの初めの手順は、Mapping Designer でマッピングを作成することです。

マッピングを作成するには：

1. Mapping Designer を開きます。
2. ［マッピング］-［作成］をクリックするか、ワークスペースにリポジトリオブジェクトをドラッグします。
3. 新しいマッピングの名前を入力し、［OK］をクリックします。

   マッピングに名前を付けるにあたっては、m_ *マッピング名*（例えば m_ResearchProjects）のようにしてください。

マッピングを開く

マッピングを開くには、マッピングを Mapping Designer のワークスペースにドラッグします。同じフォルダにマッピングがあって、既に開いている場合には、Designer はこのマッピングを閉じてから続行するよう促します。［OK］をクリックし、現在開いているマッピングを閉じて、別のマッピングを開きます。

フォルダ内のマッピングは、一度に 1 つずつ開くことができます。一度に複数のフォルダを開いた場合、各フォルダでそれぞれ 1 つのマッピングを開くことができます。

ヒント: また、ナビゲータでマッピングを右クリックし、［開く］を選択して、マッピングを開くこともできます。

マッピングのコピー

Designer を使用して、マッピングをコピーできます。

- フォルダ内でのコピー
- 同じリポジトリ内のフォルダにコピー
- 別のリポジトリへのコピー

Designer では、リポジトリのオブジェクトをコピーするためのコピーウィザードが用意されています。マッピングをコピーしたときに、マッピングのコンポーネントがない場合、コピーウィザードは各コンポーネントのコピーを作成します。既にマッピングコンポーネントがある場合、コピーウィザードはそれらのコンポーネントの名前の変更、上書き、または再利用を行うように促します。ただし、オブジェクトがショートカットの場合、またはコピー先フォルダに同じ名前のショートカットが含まれている場合は、オブジェクトを上書きすることはできません。オブジェクトの名前を変更するか、またはオブジェクトを再利用することだけが可能です。マッピング内のソースが、このマッピングで使用しないソースとプライマリー外部キーの関係にある場合には、コピーウィザードはこの関連ソースをコピーするかどうか聞いてきます。

マッピングセグメントのコピー

マッピングロジックの一部を再利用したい場合、マッピングおよびマプレットのセグメントをコピーできます。セグメントはマッピングまたはマプレットの 1 つ以上のオブジェクトにより構成されます。セグメントには、ソース、ターゲット、トランスフォーメーション、マプレット、または、ショートカットを含めることができます。マッピングセグメントをコピーするには、Mapping Designer からセグメントを選択してコピーし、コピー先のマッピング、空のマッピング、またはマプレットのワークスペースに貼り付けます。フォルダまたはリポジトリ間でセグメントをコピーすることができます。

マッピングまたはマプレットのセグメントをコピーするには：

1. マッピングまたはマプレットを開きます。
2. コピーする各オブジェクトを強調表示して、セグメントを選択します。

   複数のオブジェクトを選択することができます。また、ワークスペースでマウスをドラッグしてオブジェクトを矩形で囲んでもセグメントを選択できます。

3. ［編集］-［コピー］をクリックするか、Ctrl+C キーを押して、セグメントをクリップボードにコピーします。
4. コピー先のマッピングまたはマプレットを開きます。セグメントを空のワークスペースに貼り付けることもできます。
5. ［編集］-［貼り付け］をクリックするか、Ctrl+V キーを押します。

Designer によって競合が発生したオブジェクトの名前の変更、再使用、または、置き換えをするように求められます。

［別名でコピー］コマンドの使用

マッピングを変更し、元のマッピングを上書きしない場合、［マッピング］-［別名でコピー］をクリックすることにより変更されたマッピングをコピーできます。［別名でコピー］を使用した場合、変更はマッピングのコピーに含まれ、元のマッピングにはこれらの変更は反映されません。

［別名でコピー］は、同じフォルダ内でのみ使用できます。［別名でコピー］を使用する場合、マッピングをワークスペースで開いておく必要があります。

［別名でコピー］コマンドを使用してマッピングをコピーするには：

1. Mapping Designer のワークスペースでマッピングを開きます。
2. ［マッピング］-［別名でコピー］を選択します。
3. 新しいマッピング名を入力します。
4. ［OK］をクリックします。

［別名でコピー］を使用してショートカットをコピーすることはできません。

マッピングオブジェクトのコピー

Designer では、マッピング内の 1 つ以上のオブジェクトをコピーできます。コピーしたオブジェクトは、他のマッピングまたは同じフォルダ内のマプレットに貼り付けることができます。マッピングからオブジェクトをコピーし、それを別のマッピングまたはマプレットに貼り付けると、作成したトランスフォーメーションロジックを再利用できます。

マッピングのエクスポートおよびインポート

Designer では、マッピングを XML ファイルにエクスポートしたり、XML ファイルからインポートしたりできます。エクスポートおよびインポート機能を使って、同じリポジトリ、接続されたリポジトリ、または接続できないリポジトリにマッピングをコピーすることができます。

マッピングの編集

マッピングを作成した後、オブジェクトの追加、変更、削除といった編集を加えることができます。オブジェクトとは、ソース定義、ターゲット定義、マプレット、およびトランスフォーメーションです。Designer は、マッピングのオブジェクトを削除する前に、削除するオブジェクトのリストを表示します。マッピングを保存すると、出力ウィンドウに検証メッセージが表示されます。

マッピングに加えた変更によって影響を受けるセッションまたはショートカットを確認するには、ナビゲータ内でマッピングを選択して右クリックし、［依存関係を表示］を選択します。または［マッピング］-［依存関係を表示］の順にクリックします。

以前保存したマッピングに戻す

マッピングを編集する際に、前回保存した後に行った変更を取り消して、前回保存したマッピングに戻すことができます。そのためには、［編集］-［保存状態に復帰］を選択します。［はい］をクリックすると、Designer は最後にマッピングを保存したとき以降に加えられた変更をすべて無視します。

マッピング名の変更およびマッピングへのコメントの追加

マッピングでは、いつでも名前の変更、コメントの追加、ビジネス文書へのリンクの指定を実行できます。マッピングにコメントやビジネス文書へのリンクを追加することは、マッピングの目的を文書化する簡便な方法です。メタデータの分析を容易にするため、Repository Manager および MX の画面にはこうしたコメントが表示されます。

マッピング名を変更したり、マッピングにコメントを追加するには：

1. Mapping Designer でマッピングを開き、［マッピング］-［編集］を選択します。
2. ［マッピングの編集］ダイアログボックスに、マッピングの新しい名前を入力します。
3. コメントボックスにマッピングの説明を追加します。

   最大で 2,000 文字まで入力できます。
4. ［OK］をクリックします。

セッションの無効化

マッピングを編集して保存すると、変更の内容によっては、マッピングが有効でもセッションが無効となる場合があります。Integration Service では、無効なセッションは実行されません。マッピングを編集する場合は、以下のアクションを実行すると、Designer でセッションが無効になります。

- ソースまたはターゲットを追加または削除した。
- マプレットやトランスフォーメーションを削除した。
- オブジェクトのインポートやコピーを行った際、ソース、ターゲット、マプレット、またはトランスフォーメーションを他のものと入れ替えた。
- ソース修飾子または COBOL ノーマライザを追加または削除したか、これらのトランスフォーメーションの関連ソースのリストを変更した。
- ジョイナトランスフォーメーションまたはアップデートストラテジトランスフォーメーションを追加または削除した。
- マッピング内のマプレットにトランスフォーメーションを追加したか、マプレットからトランスフォーメーションを削除した。
- ソースまたはターゲットのデータベースタイプを変更した。

マッピングのデバッグ

有効なマッピングをデバッグして、データおよびエラー条件に関するトラブルシューティング情報を取得することができます。マッピングをデバッグするには、Mapping Designer 内からデバッガを設定および実行する必要があります。デバッガは実行時にブレークポイントで一時停止するので、トランスフォーメーション出力データを表示し編集することができます。

マッピングの削除

必要でなくなったマッピングは削除することができます。マッピングを削除しても、マッピングの外部で定義されているソース、ターゲット、マプレット、および再利用可能なトランスフォーメーションは削除されません。

注: バージョン管理を有効にしている場合、削除されたマッピングはチェックインするまでチェックアウトされたままになります。削除されたマッピングをチェックインするには、［バージョニング］-［チェックアウトの検索］を選択します。削除されたマッピングを選択し、［ツール］-［チェックイン］を選択します。

ナビゲータウィンドウからマッピングを削除することができるほか、Mapping Designer のワークスペースに表示されているマッピングを削除することもできます。

- ナビゲータウィンドウからマッピングを削除するには、マッピングを選択して Delete キーを押すか、［編集］-［削除］を選択します。
- Mapping Designer のワークスペースに現在表示されているマッピングを削除するには、［マッピング］-［削除］を選択します。

ポートへのリンクパスの表示

マッピングを編集するときに、特定のポートへの前方および後方リンクパスを表示できます。リンクパスを見ると、ソース内のカラムからトランスフォーメーションのポートを経由してターゲットのポートに渡されるデータの流れを確認できます。

リンクパスを表示するには、ポートを強調表示して右クリックします。［リンクパスの選択］オプションを選択します。前方パス、後方パス、またはその両方を表示できます。選択したリンクパスにあるすべてのコネクタが Designer に表示されます。

両方のリンクパスを表示するとき、Designer はソース内の 1 つのカラムからのデータが各トランスフォーメーションを出入りしてターゲットの 1 つのポートに入るまでの流れをトレースします。Designer は、コネクトされていないトランスフォーメーションのリンクパスは表示しません。コネクトされているルックアップトランスフォーメーションの場合、Designer は、ルックアップ条件に関係する入力ポートに依存するそれぞれの出力ポートを表示します。カスタムトランスフォーメーションの場合、Designer はデフォルトで出力ポートがすべての入力ポートに依存していることを示します。ただし、カスタムトランスフォーメーションでポートの関係を定義する場合、Designer は定義する従属ポートを示します。

注: Designer でリンクパスのコネクタを表すのに使用されるカラーを設定することができます。形式オプションを設定するときに、［リンクの選択］オプションを選択します。

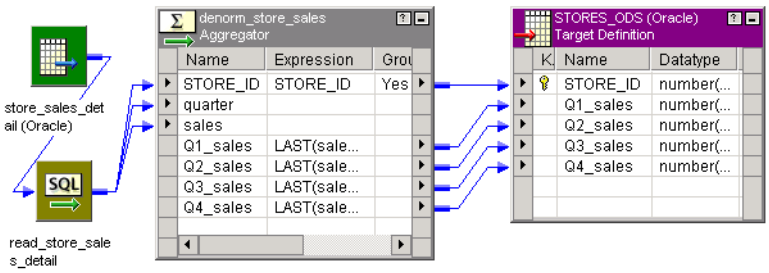
ソースカラムの依存関係の表示

マッピングを編集するときに、ターゲットカラムのソースカラム依存性を表示することができます。ソースカラムの依存性を表示すると、ターゲットカラムがデータを受け取るソースカラムを確認できます。

カラムの依存性を表示するには、マッピング内のターゲットカラムを右クリックし、［フィールド依存関係の表示］を選択します。Designer は、ターゲットカラムに接続されているすべてのソースカラムのリストを［フィールド依存関係］ダイアログボックスに表示します。

複数のソースカラムを使用して計算を実行するポート式を定義し、そのポートをターゲットカラムに接続すると、［フィールド依存関係］ダイアログボックスには式に使用されているすべてのソースカラムがリストされます。

たとえば、次のようなマッピングがあるとします。



アグリゲータトランスフォーメーションの Q3_sales ポートに以下の式を定義します。

```
LAST(sales, quarter = 3)
```

Q3_sales ターゲットカラムを右クリックし、［依存関係の表示］を選択します。

以下の図に、表示される［フィールドの依存関係］ダイアログボックスを示します。

マッピングオブジェクトの接続

マッピング内のソース、ターゲット、およびトランスフォーメーションオブジェクトを追加して設定したら、マッピングオブジェクトを接続してマッピングを完成させます。ポートを介してマッピングオブジェクトを接続できます。データは以下のポートを介してマッピングを通過します。

- **入力ポート。**データを受け取ります。
- **出力ポート。**データを渡します。
- **入力ポート/出力ポート。**データを受け取り、そのデータを無変更で渡します。

各ソースインスタンス、ターゲットインスタンス、マップレット、およびトランスフォーメーションには、ポートの集合が含まれています。 各ポートはデータのカラムを表します。

- ソースはデータを提供するものですから、ソースには出力ポートしか含まれていません。
- ターゲットはデータを受け取るものですから、ターゲットには入力ポートしか含まれていません。
- マップレットには入力ポートと出力ポートしか含まれていません。
- トランスフォーメーションは、そのトランスフォーメーションとアプリケーションに応じて、入力ポート、出力ポート、および入力ポート/出力ポートの組み合わせ含むことがあります。

ポートを接続するには、異なるマッピングオブジェクトのポート間をドラッグします。Designer は接続を検証し、リンクの検証および連結条件を満たした場合のみ接続を作成します。

ポートは未接続のままにすることができます。 Integration Service では、未接続のポートは無視されます。

ポートへのリンクオプション

トランスフォーメーションをリンクする場合、以下のいずれかのオプションを指定してリンクすることができます。

- **1 対 1。**1 つのトランスフォーメーションまたは出力グループを、1 つのトランスフォーメーション、入力グループ、またはターゲットのみにリンクします。

- **1 対多。**

  - 1 つのポートを複数のトランスフォーメーション、入力グループ、またはターゲットにリンクします。

  - 1 つのトランスフォーメーションまたは出力グループの複数のポートを、複数のトランスフォーメーション、入力グループ、またはターゲットにリンクします。

- **多対 1。**複数のトランスフォーメーションを、1 つのトランスフォーメーション、入力グループ、またはターゲットにリンクします。

1 対多のリンク

同一のデータを異なる目的で使用する場合には、このデータを提供するポートを、マッピング内の複数のポートにリンクすることができます。 例えば、アグリゲータトランスフォーメーションにより、給料情報を用いて、ある部署の平均給与を算出することができますが、各従業員の給与月額を計算するように設定した式トランスフォーメーションでも同じ情報を使用することができます。

多対 1 のリンク

往々にして、データを複数のトランスフォーメーションから単一のトランスフォーメーションまたはターゲットに統合することが必要になります。例えば、複数のアグリゲータトランスフォーメーションおよび式トランスフォーメーションから単一のファクトテーブルへのデータの統合が必要な場合があります。

マッピングオブジェクトの接続に関するルールおよびガイドライン

マッピングオブジェクトを接続する場合には、以下のルールおよびガイドラインに従います。

- 2 つのマッピングオブジェクト間でポートをリンクする時に Designer がエラーを検出すると、ポートにリンクできない旨を示す記号が表示されます。
- マッピングのデータフローのロジックに従います。以下のタイプのポートをリンクできます。

  - 受け取る側のポートは入力または入出力ポートでなければなりません。

  - 送る側のポートは、出力または入出力ポートでなければなりません。

  - 入力ポートを入力ポートにリンクしたり、出力ポートを出力ポートにリンクすることはできません。

- 少なくとも 1 つの入力グループのポートを先行するトランスフォーメーションにリンクする必要があります。
- 少なくとも 1 つの出力グループのポートを後続のトランスフォーメーションにリンクする必要があります。
- 1 つのアクティブなトランスフォーメーション、またはアクティブなトランスフォーメーションの 1 つの出力グループから、別のトランスフォーメーションの入力グループにリンクできます。
- アクティブなトランスフォーメーションとパッシブなトランスフォーメーションを同じ後続トランスフォーメーションまたはトランスフォーメーション入力グループに接続することはできません。
- 2 つ以上のアクティブなトランスフォーメーションを同じ後続トランスフォーメーションまたはトランスフォーメーション入力グループに接続することはできません。
- 任意の数のパッシブなトランスフォーメーションは、同じ後続トランスフォーメーション、トランスフォーメーション入力グループ、またはターゲットに接続できます。
- 両方の出力グループのデータがソートされている場合、同一のトランスフォーメーション内の 2 つの出力グループから、ソート済みデータに設定されている 1 つのジョイナトランスフォーメーションにポートをリンクできます。
- 互換性のあるデータタイプを持つポートのみをリンクできます。Designer は 2 つのデータタイプ間でマッピングが可能であることを検査してから、リンクします。Integration Service は、データタイプに互換性がないポート間ではデータを変換することはできません。 データタイプは同一である必要はありませんが、例えば、Char と Varchar のように互換性がなければなりません。

- ソース定義は、ソース修飾子のみにリンクする必要があります。その後で、ソース修飾子をターゲットまたは他のトランスフォーメーションにリンクします。
- カラムをマッピング内のターゲット定義にリンクすることはできますが、カラムをマッピング内のターゲット定義にコピーすることはできません。ターゲット定義にカラムを追加するには、Target Designer を使用します。
- マッピングがデータフロー検証に違反する場合、Designer によって無効マークが付けられます。

ポートへのリンク

ポートを手動でリンクしたり、またはトランスフォーメーション間でポートを自動的にリンクすることができます。ポートを自動的にリンクする場合は、位置または名前によってリンクを作成できます。名前によってポートをリンクする場合は、接頭語または接尾語を指定してポートをリンクできます。接頭語または接尾語は、マッピング内のどこにポートがあるかを示すために使用します。例えば、マッピングには Name というソース修飾子のポートと、対応する FilName というフィルタトランスフォーメーションのポートが含まれているとします。ソース修飾子とフィルタトランスフォーメーションの間でポートを自動的にリンクする場合は、接頭語「Fil」を指定します。

手動でのポートへのリンク

手動でポートをリンクするには、［レイアウト］-［カラムのリンク］を選択します。1 つのポートから別のポートにドラッグすると、Designer は接続を作成します。空のポートにポートをドラッグした場合、Designer はポートをコピーして接続を作成します。

同時に複数のポートをリンクすることもできます。別のトランスフォーメーションにリンクするポートを複数選択するには、［Ctrl］または［Shift］キーを使用します。Designer は、まず最初のペアからポートのリンクを開始します。検証条件を満たしているポートはすべてリンクされます。

位置によるポートへのリンク

位置によるリンクを実行すると、Designer は 1 番目の出力ポートを 1 番目の入力ポートにリンクし、2 番目の出力ポートを 2 番目の入力ポートに接続します。このオプションは、同じ順に関連ポートを持つトランスフォーメーションを作成する場合に使用します。位置によるリンクには、以下のオプションを使用します。

- **［オートリンク］ダイアログボックス。**［オートリンク］ダイアログボックスを使用して自動的にポートをリンクするには、［レイアウト］-［オートリンク］をクリックします。
- **［オートリンク］コマンド。**ワークスペース内でポートを選択してリンクするには、［レイアウト］-［位置によるオートリンク］を選択します。

［オートリンク］ダイアログボックスを使用したポートへのリンク

［オートリンク］ダイアログボックスを使用して、位置によるポートのリンクを実行する場合。

1. ［レイアウト］-［オートリンク］をクリックします。
2. トランスフォーメーションとターゲットを選択します。

   ［リンク先］リストで複数のトランスフォーメーションを選択して、1 つのトランスフォーメーションを複数のトランスフォーメーションにリンクすることができます。 カスタムトランスフォーメーションや XML ターゲットなどの複数の入力グループを含んでいるオブジェクトの場合は、［リンク先］リストでグループ名を選択します。

ワークスペース内のトランスフォーメーションを、リンクする順番に選択することもできます。その後、［オートリンク］ダイアログボックスでそれぞれの［リンク元］を選択します。Designer は、ワークスペース内で選択した順番に従い［リンク先］を選択します。［適用］をクリックし、次の［リンク元］を選択して［適用］をクリックします。

3. ［位置］を選択します。
4. ［OK］をクリックします。

   Designer は 1 番目の出力ポートを 1 番目の入力ポートに、2 番目の出力ポートを 2 番目の入力ポートという具合にリンクします。

［位置によるオートリンク］コマンドを使用したポートへのリンク

［位置によるオートリンク］コマンドを使用して位置を基準にポートをリンクするには、次のとおりに実行します。

1. ［レイアウト］-［位置によるオートリンク］を選択します。
2. 位置を基準にリンクするマッピングオブジェクトを選択し、選択したポートを他のマッピングオブジェクトにドラッグします。

   Designer は、オブジェクトのすべてのポートを選択します。特定のポートだけを選択するには、［オートリンク］ダイアログボックスを使用します。
3. Designer は 1 番目の出力ポートを 1 番目の入力ポートに、2 番目の出力ポートを 2 番目の入力ポートという具合にリンクします。
4. ポートのリンクが終了したら、［レイアウト］-［カラムのリンク］の順にクリックします。

名前によるポートへのリンク

Designer では、名前に基づいてポートをリンクすることができます。Designer は、同じ名前を持つ入力ポートと出力ポートの間にリンクを追加します。名前によるリンクでは大文字と小文字は区別されません。 トランスフォーメーション間で同じポート名を使用している場合に、名前によるリンクを行います。Designer は、ユーザーが定義したプレフィックスおよびサフィックスに基づき、ポートをリンクできます。 マッピングまたはマップレット内の出現位置を区別するために、ポート名でプレフィックスまたはサフィックスを使用している場合、名前およびプレフィックス/サフィックスに基づきリンクします。名前によるリンクを実行するには、以下のオプションを使用します。

- **［オートリンク］ダイアログボックス。**名前によるポートの自動リンク、名前とプレフィックスによるポートのリンク、［オートリンク］ダイアログボックスを使用した名前およびサフィックスによるポートのリンクを実行するには、［レイアウト］-［オートリンク］をクリックします。
- **［オートリンク］コマンド。**ワークスペース内でポートを選択してオブジェクトをリンクするには、［レイアウト］-［名前によるオートリンク］を選択します。

［オートリンク］ダイアログボックスを使用した名前によるポートへのリンク

［オートリンク］ダイアログボックスを使用して、名前によるポートのリンクを実行する場合。

1. ［レイアウト］-［オートリンク］をクリックします。
2. トランスフォーメーションとターゲットを選択します。

   ［リンク先］リストで複数のトランスフォーメーションを選択して、1 つのトランスフォーメーションを複数のトランスフォーメーションにリンクすることができます。 カスタムトランスフォーメーションや XML ターゲットなどの複数の入力グループを含んでいるオブジェクトの場合は、［リンク先］リストでグループ名を選択します。
3. ［名前］を選択します。

4. ［OK］をクリックします。

［オートリンク］ダイアログボックスを使用した名前およびプレフィックス/サフィックスによるポートへのリンク

［オートリンク］ダイアログボックスを使用して、名前および接頭語/接尾語によるポートのリンクを実行するには：

1. ［レイアウト］-［オートリンク］をクリックします。
2. トランスフォーメーションとターゲットを選択します。

   ［リンク先］リストで複数のトランスフォーメーションを選択して、1 つのトランスフォーメーションを複数のトランスフォーメーションにリンクすることができます。 カスタムトランスフォーメーションや XML ターゲットなどの入力グループを含んでいるオブジェクトの場合は、［リンク先］リストでグループ名を選択します。
3. ［名前］を選択します。
4. ［詳細］をクリックして接頭語と接尾語を入力するためのオプションを表示します。
5. ［リンク元］に、リンクの始点となるポートで使用されている接頭語または接尾語を入力します。
6. ［リンク先］に、リンクの終点となるポートで使用されている接頭語または接尾語を入力します。

   この例では、Designer は、SQ_CUSTOMERS 内のポートを、FIL_STATE 内で SQ_CUSTOMERS のポートと完全に同じ名前かまたは接頭語「F_」付きの同じ名前を持つポートにリンクします。
7. ［OK］をクリックします。

［名前によるオートリンク］コマンドを使用した名前によるポートのリンク

［名前によるオートリンク］コマンドを使用して、名前に基づきポートをリンクするには、以下のとおりに実行します。

1. ［レイアウト］-［名前によるオートリンク］選択します。
2. 名前に基づきリンクするマッピングオブジェクトを選択し、選択したポートを他のマッピングオブジェクトにドラッグします。

   Designer は、オブジェクトのすべてのポートを選択します。特定のポートだけを選択するには、［オートリンク］ダイアログボックスを使用します。
3. Designer は、大文字/小文字の違いを無視して同じ名前を持つ入力ポートと出力ポートの間にリンクを追加します。
4. ポートのリンクが終了したら、［レイアウト］-［カラムのリンク］をクリックします。

ポート属性のプロパゲート

トランスフォーメーション内のポート名を編集すると、デフォルトでは、Designer は、トランスフォーメーション内の式、条件、および他のポートに含まれる、このポートへの参照をプロパゲートします。変更済みの属性をマッピング全体にプロパゲートすることもできます。

Designer は、以下の要素に基づいてポート、式および条件をプロパゲートします。

- **プロパゲートする方向。**変更を前方、後方、または両方向にプロパゲートすることができます。
- **プロパゲートする属性。**ポート名、データタイプ、精度、位取り、および説明をプロパゲートできます。

- **依存関係のタイプ。**リンクパスに沿った依存関係、または、トランスフォーメーション内の暗黙の依存関係に対して変更をプロパゲートすることができます。

依存関係タイプについて

ポート属性をプロパゲートすると、Designer では、以下の依存関係を更新することができます。

- **リンクパスの依存関係。**リンクパスの依存関係とは、プロパゲートされたポートおよびリンクパス内のポートの間の依存関係です。 リンクパスの依存関係をプロパゲートすると、Designer は、リンクパス内のポートに依存する式および条件の参照に対してデフォルトの更新も実行します。
- **暗黙の依存関係。**暗黙の依存関係とは、式または条件に基づく 2 つのポート間のトランスフォーメーション内にある依存関係です。

  例えば、ルックアップ条件で使用されるポートのデータタイプを変更すると、Designer は、条件に依存する他のポートにデータタイプの変更をプロパゲートします。

リンクパスの依存関係のプロパゲート

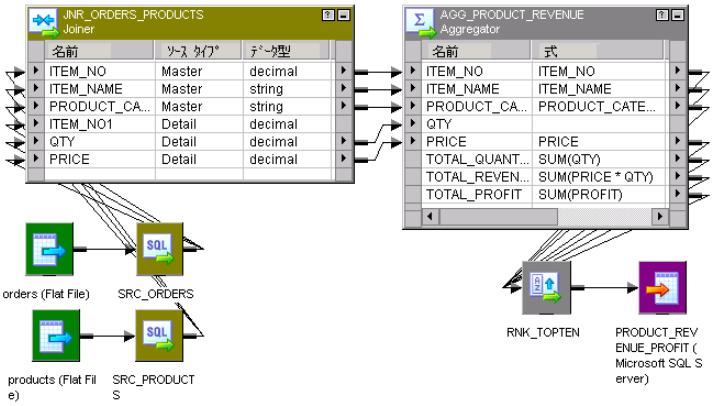
リンクパスの依存性をプロパゲートする場合、Designer は、前方リンクパスのすべての入力および入出力ポート、後方リンクパスのすべての出力および入出力ポートを更新します。Designer は、以下の更新を実行します。

- プロパゲートされたポートのリンクパスにおけるすべてのポートで、ポート名、データタイプ、精度、位取り、および説明を更新します。
- 変更されたポート名を持つプロパゲートされたポートを参照するすべての式または条件を更新します。
- 関連ポート名が変更された場合、動的ルックアップトランスフォーメーション内の関連ポートのプロパティを更新します。
- カスタムトランスフォーメーションのポート依存性のポート名を更新します。

注: ポート名をプロパゲートする際にトランスフォーメーション内に同じ名前が存在する場合は、Designer は、ポート名に「1」を付加します。

例

次のマッピングでは、ジョイナトランスフォーメーション内の QTY ポートがアグリゲータトランスフォーメーション内の QTY ポートにリンクされています。 アグリゲータトランスフォーメーションは、TOTAL_QUANTITY および TOTAL_REVENUE の式で、QTY ポートを参照します。



ジョイナトランスフォーメーションの QTY ポートに対して以下の変更を加えます。

- ポート名 QTY を QUANTITY に変更します。
- データタイプを Decimal から Integer に変更します。

属性を前方にプロパゲートすると、Designer は、アグリゲータトランスフォーメーションの以下の依存性を更新します。

- Designer は、ポート名 QTY を QUANTITY に更新します。
- Designer は、TOTAL_QUANTITY および TOTAL_REVENUE ポートのための式に含まれている QTY ポート名への参照を、QUANTITY に変更して更新します。
- Designer は、QTY ポート名のデータタイプを Integer に更新します。

暗黙の依存関係のプロパゲート

暗黙の依存性を持つポートに対して、データタイプ、精度、位取り、および説明をプロパゲートできます。［ポートのプロパゲート］ダイアログボックスの［オプション］をクリックすると、条件および式の解析を選択し、プロパゲートされたポートの暗黙の依存性を確認できます。暗黙の依存性を持つポートはすべて、出力または入出力ポートです。

条件を含める場合、Designer は、次の項目の依存プロパティを更新します。

- リンクパスの依存関係
- プロパゲートされたポートと同じルックアップ条件で使用されている任意のルックアップポート
- プロパゲートされたポートに関連付けられている動的ルックアップトランスフォーメーション内の任意のポート
- 1 つ以上の入力または入力/出力ポートとのポートの関係を定義するカスタムトランスフォーメーションが使用する任意の出力ポート
- 明細ポートと同じ結合条件で使用されている任意のマスタポート

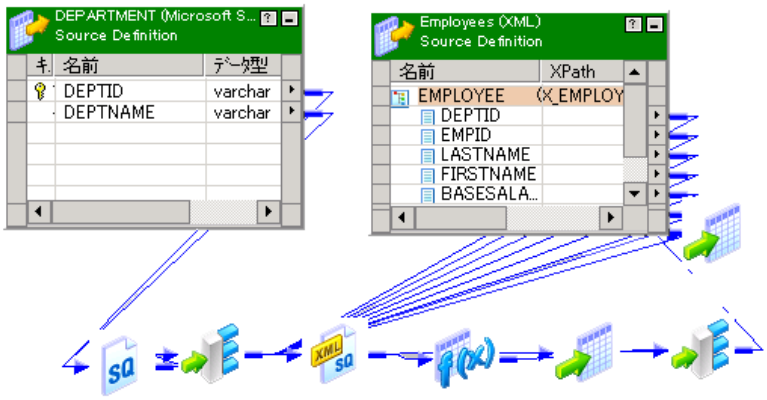
式を含める場合、Designer は、次の項目の依存プロパティを更新します。

- リンクパスの依存関係
- プロパゲートされたポートを使用する式を含む出力ポート

Designer は、同一のトランスフォーメーション内の暗黙の依存関係にプロパゲートしません。変更した属性は別のトランスフォーメーションからプロパゲートする必要があります。例えば、ルックアップ条件で使用されるポートのデータタイプを変更して、その変更をルックアップトランスフォーメーションからプロパゲートすると、Designer は、同じルックアップトランスフォーメーション内の条件に依存する他のポートにその変更をプロパゲートしません。

例

以下のようなマッピングがあるとします。



アグリゲータトランスフォーメーションの MANUFACTURER_ID ポートは、ルックアップトランスフォーメーションの IN_MANUFACTURER_ID ポートにリンクしています。ルックアップトランスフォーメーションは、以下のルックアップ条件を使用します。

```
MANUFACTURER_ID = IN_MANUFACTURER_ID
```

MANUFACTURER_ID ポートのデータタイプを、アグリゲータトランスフォーメーションで Integer から Decimal に変更します。条件を解析して依存性を推測してから、データタイプの変更をプロパゲートするように選択します。Designer では、以下の作業が実行されます。

- **リンクパスの依存関係の更新。** Designer は、リンクパスのポートを更新し、ルックアップトランスフォーメーションの IN_MANUFACTURER_ID ポートのデータタイプを Decimal に変更します。
- **依存ポートの識別。** Designer はルックアップ条件をパースし、ルックアップトランスフォーメーションの MANUFACTURER_ID ポートを従属ポートとして特定します。
- **暗黙の依存関係の更新。** Designer は、ルックアップトランスフォーメーションの MANUFACTURER_ID ポートのデータタイプを Decimal に変更します。

トランスフォーメーションでプロパゲートされる属性

以下の表に、Designer が各トランスフォーメーションに対してプロパゲートする依存関係および属性を示します。

トランスフォーメーション	依存関係	属性のプロパゲート
アグリゲータ	- リンクパス内のポート - 式 - 暗黙の依存性	- ポート名、データタイプ、精度、位取り、説明 - ポート名 - データタイプ、精度、スケール
アプリケーションソース修飾子	- リンクパス内のポート	- ポート名、データタイプ、精度、位取り、説明
カスタム	- リンクパス内のポート - ポート依存性 - 暗黙の依存性	- ポート名、データタイプ、精度、位取り、説明 - ポート名 - データタイプ、精度、スケール

トランスフォーメーション	依存関係	属性のプロパゲート
式	- リンクパス内のポート - 式 - 暗黙の依存性	- ポート名、データタイプ、精度、位取り、説明 - ポート名 - データタイプ、精度、位取り、説明
エクスターナルプロシージャ	- リンクパス内のポート	- ポート名、データタイプ、精度、位取り、説明
フィルタ	- リンクパス内のポート - Condition	- ポート名、データタイプ、精度、位取り、説明 - ポート名
入力	- リンクパス内のポート	- ポート名、データタイプ、精度、位取り、説明
ジョイナ	- リンクパス内のポート - Condition - 暗黙の依存性	- ポート名、データタイプ、精度、位取り、説明 - ポート名 - データタイプ、精度、位取り、説明
ルックアップ	- リンクパス内のポート - Condition - 関連ポート（動的ルックアップ） - 暗黙の依存性	- ポート名、データタイプ、精度、位取り、説明 - ポート名 - ポート名 - データタイプ、精度、位取り、説明
ノーマライザ（パイプライン）	- リンクパス内のポート	- ポート名、データタイプ、精度、位取り、説明
ノーマライザソース修飾子	- なし	- なし
出力	- リンクパス内のポート	- ポート名、データタイプ、精度、位取り、説明
ランク	- リンクパス内のポート - 式 - 暗黙の依存性	- ポート名、データタイプ、精度、位取り、説明 - ポート名 - データタイプ、精度、位取り、説明
ルータ	- リンクパス内のポート - Condition	- ポート名、データタイプ、精度、位取り、説明 - ポート名
SDK ソース修飾子	- なし	- なし
シーケンスジェネレータ	- なし	- なし
ソータ	- リンクパス内のポート	- ポート名、データタイプ、精度、位取り、説明
ソース修飾子	- リンクパス内のポート	- ポート名、データタイプ、精度、位取り、説明
SQL	- リンクパス内のポート	- ポート名、データタイプ、精度、位取り、説明
ストアードプロシージャ	- リンクパス内のポート	- ポート名、データタイプ、精度、位取り、説明

トランスフォーメーション	依存関係	属性のプロパゲート
トランザクションコントロール	- リンクパス内のポート - Condition	- ポート名、データタイプ、精度、位取り、説明 - ポート名
共有体	- リンクパス内のポート - 暗黙の依存性	- ポート名、データタイプ、精度、位取り、説明 - データタイプ、精度、位取り、説明
アップデートストラテジ	- リンクパス内のポート - 式 - 暗黙の依存性	- ポート名、データタイプ、精度、位取り、説明 - ポート名 - データタイプ、精度、位取り、説明
XML ジェネレータ	- リンクパス内のポート	- ポート名、データタイプ、精度、位取り、説明
XML パーサー	- リンクパス内のポート	- ポート名、データタイプ、精度、位取り、説明
XML ソース修飾子	- なし	- なし

Designer は、以下のマッピングオブジェクトには変更をプロパゲートしません。

- コネクトされていないトランスフォーメーション
- 再利用可能なトランスフォーメーション
- マップレット
- ソースおよびターゲットのインスタンス
- SDK ソース修飾子

ポートおよび属性のプロパゲートに関するルールおよびガイドライン

ポート属性をプロパゲートする場合には、以下の規則およびガイドラインに従ってください。

- Designer は、同一のトランスフォーメーション内の暗黙の依存関係にプロパゲートしません。
- ポートの説明をプロパゲートすると、Designer はマッピング内の他のトランスフォーメーションにて、そのポートの説明を上書きします。
- リンクパスに沿って後方にプロパゲートする場合、その変更が Integration Service においてセッションの原因とならないかを検証します。 例えば、ソース修飾子に変更をプロパゲートすると、Integration Service はセッション実行時に無効な SQL を生成する可能性があります。 ポート名「CUST_ID」を「CUSTOMER_ID」に変更すると、ソーステーブルが「CUST_ID」を使用している場合、Integration Service は誤ったカラム名を選択して SQL を生成する可能性があります。
- ポート属性をプロパゲートする場合、変更により Designer がマッピングを無効にしないことを検証します。例えば、ポートのデータタイプを Integer から String に変更し、データタイプを他のトランスフォーメーションにプロパゲートする場合、変更されたポートの 1 つを計算が使用していると Designer はマッピングを無効にします。ポートのプロパゲートの後にマッピングを検査します。Designer でマッピングが無効化された場合、［編集］-［保存状態に復帰］を選択して前回保存されたバージョンのマッピングに戻します。
- 複数のポート、および、複数のプロパゲートされたポートに依存する式または条件をプロパゲートすると、Designer は属性が一致しない場合、属性を暗黙の依存性にプロパゲートしません。

  例えば、式トランスフォーメーション内に以下の式があるとします。

  ```
  Item_desc_out = Substr(ITEM_NAME, 0, 6) || Substr(ITEM_DESC, 0, 6)
  ```

Item_desc_out の精度は 12、ITEM_NAME は 10、ITEM_DESC は 10 です。ITEM_DESC の精度を 15 に変更します。式を解析して依存性を推測し、ITEM_NAME および ITEM_DESC のポート属性をプロパゲートするように選択します。ITEM_NAME および ITEM_DESC ポートが異なる精度を持っているため、Designer は、式トランスフォーメーション内の Item_desc_out ポートの精度を更新しません。

ポート属性のプロパゲート手順

以下の手順でポート属性をプロパゲートします。

1. Mapping Designer でマッピングを開き、プロパゲートする 1 つ以上のポートを選択します。
2. ［マッピング］-［属性のプロパゲート］をクリックし、［属性のプロパゲート］を選択します。またはポートを右クリックし、［属性のプロパゲート］を選択します。

   Designer に、［ポート属性のプロパゲート］ダイアログボックスが表示されます。

   ［ポート属性のプロパゲート］ダイアログボックスを開くと、別のポートを選択でき、その属性をプロパゲートできます。
3. プロパゲートしたい方向および属性を選択します。
4. ［オプション］をクリックして、依存関係を推測することもできます。

   以下の表に、［ポート属性のプロパゲート］ダイアログボックスのオプションを示します。

オプション	説明
プレビュー	影響を受けるポートへのリンクを緑、影響を受けないポートへのリンクを赤で表示します。
プロパゲート	ダイアログボックスで指定したオプションに従って、ポート属性をプロパゲートします。
方向	Designer に指示して、属性を前方、後方、または両方向にプロパゲートします。
プロパゲートする属性	プロパゲートする属性を指定します。属性には、名前、データ型、精度、位取り、および説明が含まれています。
オプション	条件または式を読み込み、属性を暗黙の依存性にプロパゲートします。ポート名をプロパゲートする場合には、これらのオプションは無効になっています。これらのオプションはデフォルトでは、クリアされています。これらのオプションのいずれかを選択し、［プレビュー］をクリックすると、Designer は、従属ポートへのリンクパスを強調表示します。

5. ［プレビュー］をクリックして影響を受けるポートを表示します。

   Designer は、影響を受けるポートへのリンクを緑、影響を受けないポートへのリンクを赤で表示します。
6. ［プロパゲート］をクリックします。

   ポート属性をプロパゲートすると、出力ウィンドウにプロパゲートされた属性および影響を受けたポートが表示されます。
7. ［閉じる］をクリックします。

マッピング内のソースに関する作業

マッピングを作成する際に、1 つ以上のソース定義をマッピングに追加する必要があります。ソースを Mapping Designer のワークスペースにドラッグすると、ソース定義のインスタンスがマッピングに追加されます。

リレーショナルソースについては、ソース定義を編集し、デフォルトのソーステーブル名を上書きできます。

各マッピングには以下のソース修飾子のうち最低でも 1 つのソース修飾子が必要であり、このソース修飾子により、Integration Service でデータが読み込まれる方法が決定されます。

- **ソース修飾子トランスフォーメーション。**リレーショナルソースおよびフラットファイルソースからのデータ読み込みを表します。
- **ノーマライザトランスフォーメーション。**COBOL ソースからのデータ読み込みを表します。
- **アプリケーションソース修飾子トランスフォーメーション。**アプリケーションソースからのデータ読み込みを表します。
- **アプリケーション複数グループソース修飾子トランスフォーメーション。**複数グループアプリケーションソースからのデータ読み込みを表します。
- **XML ソース修飾子トランスフォーメーション。**XML ソースからのデータ読み込みを表します。

Designer では、デフォルトでソース修飾子が作成されます。ソースインスタンスをマッピングにドラッグするたびに、Designer はソース修飾子を追加し、このソース修飾子をソースに接続します。マッピング内の各ソースに対してそれぞれ 1 つのソース修飾子を作成する場合、ソース修飾子の自動作成を使用します。異なるリレーショナルソースから得たデータを結合したい場合は、自動作成を無効にすることができます。そして、手動で作成してソースに接続することができます。

Source Analyzer でソースを編集すると、マッピング内のソースのすべてのインスタンスがその変更を継承します。変更によっては、当該ソースを使用したマッピングが無効となる場合があります。

ただし、マッピング内のすべてのソースインスタンスに対していくつかのプロパティを指定することができます。Mapping Designer 内でソースインスタンスをダブルクリックし、［プロパティ］タブをクリックします。リレーショナルソースに対しては、テーブルオーナ名を指定できます。フラットファイルソースに対しては、デフォルトの日付/時刻形式、3 桁ごとの区切り文字、および小数位の区切り文字を指定できます。

注: 注：テーブル名にいくつか特殊文字があるソース定義をマッピングに追加する場合、Designer はソースインスタンス名にある特殊文字をアンダースコア（_）に置き換えてマッピングします。

マッピング内のリレーショナルソースに関する作業

リレーショナルソースにマッピングを追加する際、テーブルオーナー名を表示し、ソーステーブル名を上書きできます。この情報は、Mapping Designer のリレーショナルソースインスタンスの［プロパティ］タブで表示できます。

テーブルオーナー名は、データベースのソーステーブルのオーナー名を表示します。DB2 等の一部のデータベースでは、テーブルは複数のオーナーを持つことができます。セッションプロパティの各ソースインスタンスのテーブルオーナー名を上書きできます。

ソースインスタンスの［プロパティ］タブでリレーショナルソースインスタンスのソーステーブル名を上書きできます。異なるソーステーブルからデータを読み込むために単一のマッピングを使用している際に、ソーステーブル名を上書きします。ソーステーブル名にテーブル名を入力します。パラメータまたは変数も入力できます。ソーステーブル名には、マッピングパラメータ、マッピング変数、セッションパラメータ、ワークフロー変数、ワークレット変数が使用できます。例として、セッションパラメータ$ParamSrcTable をソーステーブル名として使用するとともに、パラメータファイルのソーステーブル名に設定することもできます。

注: ソースインスタンスの［プロパティ］タブでソーステーブル名を上書きし、SQL クエリを使用してソーステーブル名前を上書きした場合、SQL クエリで定義されたソーステーブル名が Integration Service で使用されます。

ソーステーブル名を上書きするには：

1. Designer で Mapping Designer ツールを開きます。
2. マッピング内におけるリレーショナルソースインスタンスのタイトルバーをダブルクリックします。
3. ［プロパティ］タブにソーステーブル名を入力します。［ソーステーブル名］フィールドにはパラメータまたは変数も入力できます。

   ユーザー定義のマッピングパラメータ、マッピング変数、ワークフロー変数、ワークレット変数を使用する場合、パラメータまたは変数を宣言する必要があります。
4. ［OK］をクリックします。
5. ソーステーブル名にパラメータまたは変数を使用する場合、パラメータまたは変数をパラメータファイルの該当するセクションで定義します。

マッピング内のトランスフォーメーションに関する作業

トランスフォーメーションは、データの生成、変更、または受け渡しを行うリポジトリオブジェクトです。ユーザーは、Integration Service によってデータの変換に使用されるロジックをトランスフォーメーション内に設定します。 Designer には、特定の関数を実行する一連のトランスフォーメーションが用意されています。たとえば、アグリゲータトランスフォーメーションはデータのグループに対して計算を実行します。

マッピング内で一度だけ使用するトランスフォーメーションか、複数のマッピング内で使用できる再利用可能なトランスフォーメーションを作成することができます。再利用可能なトランスフォーメーションをマッピングに追加する場合には、このトランスフォーメーションのインスタンスを追加してください。Transformation Developer で再利用可能なトランスフォーメーションを編集すると、マッピング内のトランスフォーメーションのすべてのインスタンスがその変更を継承します。変更によっては、再利用可能なトランスフォーメーションを使用したマッピングが無効となる場合があります。

マッピング内のマップレットに関する作業

いくつかのマッピング内の標準化された一連のトランスフォーメーションロジックを使用したい場合に、Mapplet Designer でマプレットを作成できます。マッピング内でマプレットを使用する際には、Designer がそのマプレットのインスタンスを作成します。マプレットのインスタンスは、入力トランスフォーメーションおよび出力トランスフォーメーションからのポートだけを表示します。これらのトランスフォーメーションはグループとして表示されます。マッピング内のマプレットのインスタンスに関するコメントを入力することはできますが、その他のマプレットプロパティは編集できません。

Mapping Designer でマプレットを選択し、メニューの［マッピング］-［展開］をクリックすると、マプレットを展開することができます。これにより、マッピング内のマプレットが展開されて表示されます。マプレットおよびマッピング内のトランスフォーメーションはすべて開いたりアイコン化したりできますが、そのプロパティを編集することはできません。

Mapplet Designer でマプレットを編集すると、マッピング内のマプレットのすべてのインスタンスがその変更を継承します。変更によっては、当該マプレットを使用したマッピングが無効となる場合があります。

マッピング内のターゲットに関する作業

マッピングを作成する際に、1 つ以上のターゲット定義をマッピングに追加する必要があります。ターゲットを Mapping Designer のワークスペースにドラッグすると、ターゲット定義のインスタンスが追加されます。

ターゲットをマッピングに追加する場合は、異なるタイプのターゲットを含めることができます。データベースタイプが同じで、データベース接続が異なる複数のターゲットを追加することができます。また、同じマッピング内にリレーショナルターゲットとフラットファイルターゲットの両方を含めることもできます。

Target Designer でターゲットを編集すると、マッピング内のターゲットの全インスタンスがその変更を継承します。変更によっては、当該ターゲットを使用したマッピングが無効となる場合があります。

注: 注：テーブル名に特殊文字のあるターゲット定義をマッピングに追加する場合、Designer はターゲットインスタンス名にある特殊文字をアンダースコア（_）に置き換えてマッピングします。

マッピングのリレーショナルターゲット、ファイルターゲット、および XML ターゲットのプロパティを設定できます。

マッピングのリレーショナルターゲットの設定

リレーショナルターゲットの場合、マッピング内に次のプロパティを設定することができます。

- **切り詰められたデータとオーバーフローデータの拒否。**切り詰められたデータを Integration Service によって拒否ファイルに書き込む場合に、ターゲットインスタンスの［プロパティ］タブでこのオプションを選択します。
- **更新の上書き。**SQL エディタを使用して、ターゲットインスタンスの［プロパティ］タブのデフォルト UPDATE 文をオーバーライドします。
- **テーブル名の接頭語。**ターゲットインスタンスの［プロパティ］タブで、ターゲットテーブルのオーナーを指定します。
- **セッション実行前および実行後の SQL。**マッピング内のターゲットインスタンスに対してセッション実行前の SQL コマンドを入力すると、Integration Service がソースを読み込む前にターゲットデータベースに対してコマンドを実行します。 セッション実行後の SQL コマンドを入力すると、Integration Service がターゲットに書き込んだ後にターゲットデータベースに対してコマンドを実行します。
- **ターゲットテーブル名。**デフォルトのターゲットテーブル名をオーバーライドできます。

関連項目：

- 「マッピング内のリレーショナルターゲットに関する作業」 (ページ 137)

マッピングのフラットファイルターゲットの設定

フラットファイルターゲットの場合、マッピング内に次のプロパティを設定することができます。

- **日時フォーマット。**日時の値のデフォルトの日時フォーマットを定義します。
- **3 桁ごとの区切り文字。**数値のデフォルトの 3 桁ごとの区切り文字を定義します。
- **小数位の区切り文字。**数値のデフォルトの小数位の区切り文字を定義します。

関連項目：

- 「デフォルトの日付フォーマットと数値フォーマットの定義」 (ページ 89)

マッピングの XML ターゲットの設定

XML ターゲットでは、ルート要素を変更することができます。ただし、ルートを変更すると、ターゲット XML 構造に影響を与えることになり、マッピングを無効にしてしまう可能性があります。

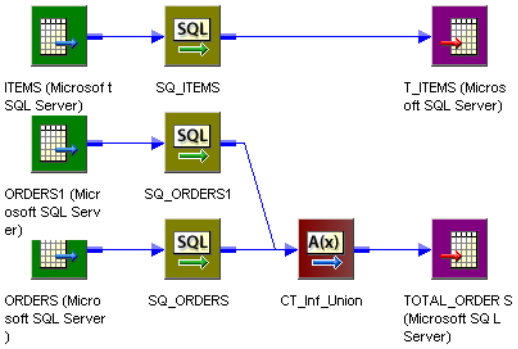
ターゲットロード順の設定

任意のタイプのターゲット定義を含むマッピングに対して、ターゲットロード順を設定することができます。Designer では、Integration Service がマッピング内の複数のターゲットロード順グループのターゲットに行を送る順序を設定できます。 ターゲットロード順グループとは、マッピング内で互いにリンクするソース修飾子、トランスフォーメーション、およびターゲットのコレクションです。プライマリキーや外部キーの制約があるテーブルでレコードの挿入、削除、または更新を行う場合に参照の整合性を維持したい場合、ターゲットロード順を設定できます。

統合サービスにより、ターゲットロード順グループのソースが同時に読み込まれ、ターゲットロード順グループはターゲットのタイプに関係なく順次処理されます。ターゲットが同じフラットファイルターゲットのコピーで、セッションのプロパティでターゲットの［存在する場合は追加］を指定している場合、統合サービスはすべてのターゲットのロード順序グループを処理します。

Integration Service がターゲットにデータを送る順序を指定するには、マッピング内の各ターゲットに対してソース修飾子を 1 つずつ作成します。 次に、ターゲットロード順を設定するために、Integration Service がマッピング内の各ソースを読み込む順序を決定します。

以下の図に、1 つのマッピング内に 2 つのターゲットロード順グループがある場合を示します。



このマッピングでは、最初のターゲットロード順グループに ITEMS、SQ_ITEMS、および T_ITEMS が含まれています。 2 つ目のターゲットロード順グループには、TOTAL_ORDERS ターゲットも含めた、マッピング内の他のすべてのオブジェクトが含まれています。 Integration Service では、最初のターゲットロード順グループを処理してから、2 つ目のターゲットロード順グループを処理します。

また、2 つ目のターゲットロード順グループを処理するときに、データを両方のソースから同時に読み込みます。

1. 複数のターゲットのロード順グループを含むマッピングを作成します。
2. ［マッピング］-［ターゲットロード プラン］を選択します。

   ［ターゲットロードプラン］ダイアログボックスに、マッピング内のすべてのソース修飾子トランスフォーメーションと、各ソース修飾子からデータを受け取るターゲットが一覧表示されます。
3. このリストからソース修飾子を選択します。

4. ［上に移動］および［下に移動］をクリックし、ソース修飾子をロード順の範囲内で移動します。
5. 順序を変更する他のソース修飾子について、3 と 4 の手順を繰り返します。
6. ［OK］をクリックします。

トランザクション別ターゲットファイルの作成

Integration Service が新規トランザクションを開始するたびに、個別の出力ファイルを生成できます。ターゲットフラットファイル名は動的に指定できます。

トランザクションごとに個別の出力ファイルを生成するには、FileName ポートをフラットファイルターゲット定義に追加します。マッピング内の FileName ポートに接続すると、Integration Service によって個別のターゲットファイルが各コミットに作成されます。Integration Service は、各トランザクションの最初の行にある FileName ポート値に基づいて出力ファイルに名前を付けます。デフォルトでは、$PMTargetFileDir に出力ファイルが書き込まれます。

ターゲットの設定

Target Designer のフラットファイルターゲット定義に［FileName］カラムを追加します。

［FileName］カラムを追加するには：

1. Target Designer でフラットファイルターゲット定義を開きます。
2. ［カラム］タブをクリックします。
3. ［FileName］カラムの追加ボタンをクリックします。

Designer は、FileName という名前の文字列ポートを作成します。ポートの精度を変更できます。

マッピングの設定

ソースベースのコミットまたはユーザー定義のコミットから出力ファイルを生成できます。ソースベースのコミットを使用すると、ソースの行数に基づいてデータをターゲットファイルにコミットできます。例えば、1,000 データ行ごとに個別の出力ファイルを作成できます。有効なトランザクションジェネレータがマッピングに含まれる場合は、ユーザー定義コミットを設定できます。例えば、都市ごとに個別の出力ファイルを作成できます。

マッピングで、ターゲットの FileName ポートを、各トランザクションの開始時に一意の値を含むトランスフォーメーションポートに接続します。トランスフォーメーションで式を作成すると、一意のファイル名を生成してそのファイル名を FileName ポートに渡すことができます。

セッションの実行

FileName ポートを設定すると、Integration Service は出力ファイル名のセッション属性を［FileName］カラムの値で上書きします。 ある行の［FileName］カラムの値が NULL の場合、その行はエラーになり Integration Service では処理されません。トランザクション境界後の［FileName］カラムが NULL の場合、Integration Service はデフォルトの出力ファイル名を使用して出力ファイルに名前を付けます。

［FileName］カラムにはトランザクションごとに一意の値が含まれる必要があります。トランザクションとトランザクションの間に［FileName］カラムの値が変更されなかった場合、Integration Service はフラットファイルターゲットを上書きします。

マッピングで FileName ポートを接続しなかった場合、Integration Service は 1 つのターゲットファイルを生成し、セッションで設定された出力ファイル名を使用します。

トランザクションによるターゲットファイルの作成に関するルールおよびガイドライン

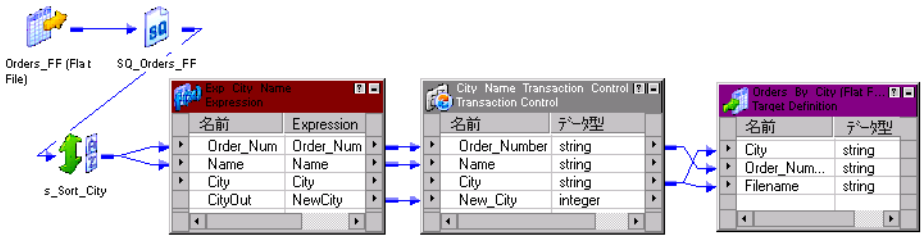
［FileName］カラムを作成する場合には、以下の規則およびガイドラインに従ってください。

- ［FileName］カラムはフラットファイルターゲットと共に使用できます。
- フラットファイルターゲット定義に［FileName］カラムを追加します。
- ［FileName］カラムはリアルタイムソースのデータと共に使用できます。
- 統合ファイル、ファイルリスト、または FTP ターゲットと共に［FileName］カラムを使用すると、セッションが失敗します。
- 複数のパーティションにあるターゲットに同じファイル名を渡すと、予期しない結果になる場合があります。
- トランスフォーメーションが入力トランザクション境界を削除してコミットを生成しなかった場合、Integration Service はすべての行を同じ出力ファイルに書き込みます。FileName ポートの初期値が出力ファイルの名前になります。

例

複数の都市の注文を含むソースがあります。都市を元に個別の出力ファイルに注文を書き込みます。

以下の図に、注文を処理するマッピングを示します。



このマッピングには、ソースとソース修飾子のほかに、以下のオブジェクトがあります。

- **ソータートランスフォーメーション。** ソースデータを都市別にソートします。
- **式トランスフォーメーション。**行に新しい都市が出現したタイミングを判断し、トランザクション制御トランスフォーメーションの New City ポートに整数を渡します。 デフォルトでは 0 を渡し、行に新しい都市が含まれる場合には 1 を渡します。
- **トランザクション制御トランスフォーメーション。**式トランスフォーメーションの New City の値を評価します。 New City が 1 の場合、トランザクション制御トランスフォーメーションはトランザクション内のすべての注文をターゲットにコミットします。トランザクション制御トランスフォーメーションは都市と注文数をフラットファイルターゲットに渡します。また、ターゲットの［FileName］カラムにも都市を渡します。
- **フラットファイルターゲット。**トランザクションごとに新しいフラットファイルを書き込みます。Integration Service は FileName の値を使用して各ターゲットに名前を付けます。

Integration Services は都市ごとにトランザクションをターゲットに渡します。この例の場合、データには以下の都市と注文数が含まれます。

```
Brisbane, 100
San Francisco, 101
San Francisco, 104
San Francisco, 105
San Francisco,107
Tiburon, 102
```

```
Tiburon, 106
Tiburon, 102
```

Integration Service は、以下の出力ファイルを生成します。

```
Brisbane
San Francisco
Tiburon
```

マッピング内のリレーショナルターゲットに関する作業

リレーショナルターゲットをマッピングに追加した場合、以下のプロパティを設定できます。

- **切り詰められたデータとオーバーフローデータの拒否。**切り詰められたデータを Integration Service によって拒否ファイルに書き込む場合に、ターゲットインスタンスの［プロパティ］タブでこのオプションを選択します。
- **更新のオーバーライド.**SQL エディタを使用して、ターゲットインスタンスの［プロパティ］タブのデフォルト UPDATE 文をオーバーライドします。
- **テーブル名の接頭語。**ターゲットインスタンスの［プロパティ］タブで、ターゲットテーブルのオーナーを指定します。
- **セッション実行前および実行後の SQL。**Integration Service がソースを読み込む前に、マッピング内のターゲットインスタンスに対してセッション実行前の SQL コマンドを入力し、ターゲットデータベースに対してコマンドを実行できます。 セッション実行後の SQL コマンドを入力すると、Integration Service がターゲットに書き込んだ後にターゲットデータベースに対してコマンドを実行します。
- **ターゲットテーブル名。**デフォルトのターゲットテーブル名をオーバーライドできます。

注: これらのプロパティは、Target Designer で設定できません。

切り詰められたデータとオーバーフローデータの拒否

Designer を使用すれば、データをポート間で転送することにより、このデータを変換できます。変換を行うと、数値データのオーバーフローが起きたり、文字列が切り詰められたりすることもあります。例えば、Decimal（28, 2）ポートから Decimal（19, 2）ポートにデータを渡すと、数値データのオーバーフローが発生します。同様に、String（28）ポートから String（10）ポートにデータを渡す場合、Integration Service により文字列が 10 文字に切り詰められます。 変換の結果オーバーフローが発生した場合、デフォルトでは Integration Service により、該当する行がスキップされます。 Integration Service では、データは拒否ファイルに書き込まれません。 文字列について、Integration Service によりその文字列が切り詰められてから次のトランスフォーメーションに渡されます。

Designer によって提供されるオプションにより、最後のトランスフォーメーションとターゲット間で切り詰められたデータやオーバーフローしたデータをすべてセッション拒否ファイルに含めることができます。［Reject Truncated/Overflow Rows］を選択した場合、Integration Service により、切り詰められた行やオーバーフローした行がすべて、セッションの設定に応じて、セッション拒否ファイルまたは行エラーログに送られます。

ターゲット更新のオーバーライドの設定

デフォルトでは、Integration Service により、キー値に基づいてターゲットテーブルが更新されます。 しかしながら、マッピングの各ターゲットについて、デフォルトの UPDATE 文を上書きすることができます。これによって、キー以外のカラムに基づいてターゲットの更新を行うことができます。

ソース、ターゲット、またはルックアップデータベースに対して SQL を実行する場合、Integration Service により Integration Service インストールディレクトリに格納されている予約語ファイルが検索されます。 一致する予約語は引用符で囲まれます。ターゲット更新のオーバーライドを使う場合は、すべての予約語を手作業によって引用符で囲む必要があります。

マッピングにアップデートストラテジトランスフォーメーション、または［アップデートストラテジ］プロパティを有効にしてあるカスタムトランスフォーメーションが含まれない場合は、セッションの設定時にソース行を「更新」に設定してください。［ターゲットの更新］オプションが影響を与えるのは更新としてマークされたソース行だけです。Integration Service により、挿入、削除、または拒否としてマークされたすべての行が通常どおりに処理されます。 セッションの設定時に、ソース行をデータドリブンとしてマークします。［ターゲットのオーバーライド］が影響を及ぼすのは、アップデートストラテジトランスフォーメーションまたはカスタムトランスフォーメーションで「更新」に設定されたソース行だけです。

例えば、あるマッピングが各販売員の総販売実績を T_SALES テーブルに送るとします。

Designer は、次のデフォルト UPDATE 文をターゲット T_SALES 用に生成します。

```
UPDATE T_SALES SET EMP_NAME = :TU.EMP_NAME, DATE_SHIPPED = :TU.DATE_SHIPPED, TOTAL_SALES = :TU.TOTAL_SALES
WHERE EMP_ID = :TU.EMP_ID
```

ターゲットポートはターゲットカラム名に一致する必要があるため、更新文にはターゲットトランスフォーメーションのポートを指定するための「:TU」というキーワードが含まれます。この文の UPDATE 部分を変更する場合には、必ず:TU を使用してポートを指定します。

WHERE 句の上書き

WHERE 句を上書きしてキー以外のカラムを含めることができます。例えば、Mike Smith という名前の従業員についてだけ、レコードの更新を行うことができます。これを行うには、WHERE 句を以下のように編集します。

```
UPDATE T_SALES SET DATE_SHIPPED = :TU.DATE_SHIPPED,

TOTAL_SALES = :TU.TOTAL_SALES WHERE :TU.EMP_NAME = EMP_NAME and

EMP_NAME = 'MIKE SMITH'
```

ターゲット更新のオーバーライドの設定に関するルールおよびガイドライン

ターゲット更新クエリーを入力する場合には、以下の規則とガイドラインに従ってください。

- ターゲット更新のオーバーライドを使う場合は、手作業ですべての予約語を引用符で囲む必要があります。
- ターゲットカラム名に次の文字のどれかが含まれている場合、デフォルトの UPDATE 文は上書きできません。
  ' , ( ) < > = + - * / \ t \ n \ 0 <space>
- ターゲットの更新クエリーではパラメータと変数を使用できます。パラメータファイルで定義可能なパラメータまたは変数タイプを使用します。パラメータまたは変数は、UPDATE 文の中に入力することも、あるいは更新クエリーとして使用することもできます。例えば、セッションパラメータ$ParamMyOverride は、更新クエリーとして入力することも、またパラメータファイル内の UPDATE 文に設定することもできます。
- マッピングの保存を行うと、Designer は参照ポート名が有効であるかどうかを検査します。SQL が正確であるかどうかは検証しません。
- ターゲットテーブルの個々の行について複数回更新を行った場合には、データベースには最後の更新データが入ります。マッピングに結果データの順序が定義されていない場合、同一の入力データに対して異なるマッピングを実行すると、ターゲットテーブルのデータが異なる場合があります。
- カラム参照を含まない WHERE 句は、ターゲットテーブルの行すべてを更新する場合もあれば、まったく更新しない場合もあります。これは、WHERE 句の内容とマッピングからのデータによって決まります。例え

ば、以下のクエリーでは、トランスフォーメーションのいずれかの行に EMP_ID > 100 がある場合、ターゲットテーブルの *すべての*行について EMP_NAME を「MIKE SMITH」に設定します。

```
UPDATE T_SALES set EMP_NAME = 'MIKE SMITH' WHERE :TU.EMP_ID > 100
```

- WHERE 句にポート参照が含まれていない場合には、マッピングの各行について同じ一連の行が更新されます。例えば、以下のクエリーでは EMP_ID > 100 のすべての従業員が更新され、マッピングの最後の行から EMP_NAME を取得します。

  ```
  UPDATE T_SALES set EMP_NAME = :TU.EMP_NAME WHERE EMP_ID > 100
  ```

- マッピングにアップデートストラテジまたはカスタムトランスフォーメーションが含まれている場合、ターゲット更新文は「更新」のマークが付いているレコードにのみ影響します。
- ［ターゲットの更新］オプションを使用する場合は、セッションを設定して、すべてのソースレコードを更新としてマークします。

ターゲット更新文の入力手順

更新文を作成するには、以下の手順を実行します。

1. ターゲットインスタンスのタイトルバーをダブルクリックします。
2. ［プロパティ］をクリックします。
3. ［更新のオーバーライド］フィールドで［開く］ボタンをクリックします。

   SQL エディタが表示されます。
4. ［SQL 文の生成］を選択します。

   デフォルトの UPDATE 文が表示されます。
5. UPDATE 文を変更します。

   WHERE 句を上書きしてキー以外のカラムを含めることができます。

   予約語はすべて引用符で囲んでください。
6. ［OK］をクリックします。

   このマッピングを保存する際に、Designer は SQL の検証を行います。

テーブル名のプレフィックスの設定

テーブル名の接頭語は、ターゲットテーブルのオーナを示すものです。DB2 など一部のデータベースでは、セッション内のターゲットテーブルが異なるオーナを持つことができます。データベース接続で指定されたデータベースユーザーがセッション内のターゲットテーブルのオーナではない場合、各ターゲットインスタンスに対するオーナを指定します。データベースユーザーがオーナでない場合にテーブルオーナ名を指定しないと、セッションが失敗する場合があります。

テーブルのオーナー名は、ターゲットインスタンスまたはセッションプロパティで指定できます。テーブルオーナ名をセッションプロパティに入力すると、トランスフォーメーションプロパティが上書きされます。

注: テーブルオーナー名を指定して接続環境 SQL 内に DB2 データベースに対する sqlid を設定すると、Integration Service ではターゲットインスタンス内のテーブルオーナー名が使用されます。SET sqlid 文に指定されているテーブルオーナー名を使用するには、［Target Name Prefix］に名前を入力しないでください。

ターゲットインスタンスレベルでターゲットオーナ名を指定するには：

1. Designer で Mapping Designer ツールを開きます。
2. マッピング内におけるリレーショナルターゲットインスタンスのタイトルバーをダブルクリックします。
3. ［プロパティ］タブで、［Table Name Prefix］の［値］フィールドにテーブルオーナ名（接頭語）を入力します。
4. ［OK］をクリックします。

セッション実行前および実行後の SQL コマンドの追加

マッピング内のターゲットインスタンスの［プロパティ］タブでは、セッション実行前/実行後 SQL コマンドを入力することができます。セッション実行前に実行される SQL とセッション実行後に実行される SQL をターゲット上で使用すると、セッションの実行前にインデックスを削除し、セッション完了後にインデックスを再作成することができます。

Integration Service は、ソースを読み込む前に、ターゲットデータベースに対してセッション実行前の SQL コマンドを実行します。 セッション実行後 SQL コマンドは、PowerCenter Server がターゲットデータベースへの書き込みを完了した後でターゲットデータベースに対して実行されます。

SQL コマンドは、セッションプロパティの［マッピング］タブを使用して上書きすることができます。セッション実行前および実行後 SQL コマンドの実行時にエラーが発生した場合に、Integration Service が処理を停止または続行するかを設定することもできます。

セッション実行前および実行後の SQL コマンドの追加に関するルールおよびガイドライン

ターゲットインスタンスにセッション実行前/実行後 SQL コマンドを入力する場合には、以下の規則およびガイドラインに従ってください。

- そのデータベースタイプで有効な任意のコマンドを使用します。ただし、Integration Service では、データベース側で許可されているとしても、ネストされたコメントは許可されません。
- ターゲットのセッション実行前/実行後 SQL コマンドではパラメータと変数を使用できます。例えば、コマンドにパラメータまたは変数が入力できます。または、セッションパラメータ$ParamMyCommand は、SQL コマンドとして使用することも、またパラメータファイル内の SQL 文に設定することもできます。
- 複数の文を区切るにはセミコロン（;）を使用します。Integration Service では各ステートメントの後にコミットが発行されます。
- Integration Service では、/* ...*/内のセミコロンは無視されます。
- コメントの外部でセミコロンを使用する必要がある場合は、バックスラッシュ（\\uff09 でセミコロンをエスケープします。
- Designer では、SQL は検証されません。

注: ソース修飾子トランスフォーメーションの［プロパティ］タブで、セッション実行前/実行後 SQL コマンドを入力することができます。

ターゲットテーブル名の上書き

マッピングのターゲットインスタンスのターゲットテーブル名を上書きできます。異なるターゲットテーブルにデータを読み込むために単一のマッピングを使用している際に、ターゲットテーブル名を上書きします。ターゲットテーブル名にテーブル名を入力します。パラメータまたは変数も入力できます。ターゲットテーブル名には、マッピングパラメータ、マッピング変数、セッションパラメータ、ワークフロー変数、ワークレット変数が使用できます。例えば、ターゲットテーブル名としてセッションパラメータ$ParamTgtTable を使用し、$ParamTgtTable をパラメータファイル内のターゲットテーブル名に設定することができます。

ターゲットテーブル名を上書きするには：

1. Designer で Mapping Designer を開きます。
2. マッピング内におけるリレーショナルターゲットインスタンスのタイトルバーをダブルクリックします。
3. ［プロパティ］タブにターゲットテーブル名を入力します。または、［ターゲットテーブル名］フィールドにパラメータ名または変数名を入力します。

   ユーザー定義のマッピングパラメータ、マッピング変数、ワークフロー変数、ワークレット変数を使用する場合、パラメータまたは変数を宣言する必要があります。

4. ［OK］をクリックします。
5. ターゲットテーブル名にパラメータまたは変数を使用する場合、パラメータまたは変数をパラメータファイルの該当するセクションで定義します。

マッピングの検証

マッピングを開発する場合、Integration Service がマッピング全体を読み込み、そして処理できるようにマッピングを設定する必要があります。 Designer により、Integration Service がマッピングに関連付けられたセッションを実行できなくなるようなエラーが検出された場合、マッピングに無効のマークが付けられます。

Designer により、以下の理由でマッピングに有効のマークが付けられます。

- **接続の検証。**必要なポートが接続され、その接続がすべて有効です。
- **式の検証。**式がすべて有効です。
- **オブジェクトの検証。**個別のオブジェクト定義がマッピング内のインスタンスに一致します。
- **データフロー検証。**ブロックトランスフォーメーションでハングすることなく、データがソースからターゲットに流れる必要があります。

接続の検証

マッピング内のポートの接続、マッピングの検証、マッピングの保存を行うたびに、Designer によって接続の検証が行われます。ポートを接続すると、Designer は有効な接続かどうかを検査します。マッピングを保存または検証すると、Designer は接続が有効かどうか、必要なポートがすべて接続されているかどうかを検証します。マッピングを保存または検証すると、Designer は以下の接続検証を行います。

- **最低でも、1 つのソースと 1 つのターゲットを接続する必要がある。**
- **ソース修飾子をターゲットにマッピングする必要がある。**
- **マップレットを接続する必要がある。**最低でも 1 つのマップレット入力ポートおよび出力ポートがマッピングに接続されていなければなりません。 SQL オーバーライドを使用したソース修飾子がマップレットに含まれている場合には、Designer によりすべてのマップレット出力ポートをマッピングに接続するよう促されます。
- **ポート間のデータタイプに互換性が必要である。**ポートのデータタイプを、接続先のポートと互換性のないものに変更した場合、Designer によってエラーが生成されてマッピングが無効になります。 例えば、2 つの日付/時刻ポートを接続し、片方のポートを Decimal に変更したとします。Designer はマッピングを無効にします。しかし、Char と Varchar のように接続したポートとの互換性が維持される場合は、データタイプを変更することができます。

関連項目：

- 「マッピングオブジェクトの接続 」 (ページ 120)

式の検証

マッピングを作成するときに、トランスフォーメーション内の式を検証することができます。エラーを修正しなかった場合、マッピングを保存または検証すると出力ウィンドウにエラーメッセージが表示されます。

式で使用した入力ポートを削除すると、Designer はマッピングに無効の印を付けます。

オブジェクトの検証

マッピングを保存または検証すると、Designer はソースやマプレットなど個々のオブジェクトの定義がマッピング内のインスタンスと一致するかどうかを検査します。マッピングを設定する際にオブジェクトを 1 つでも変更すると、マッピングにエラーが含まれる場合があります。

マッピングの設定中ではないときにオブジェクトが変更された場合、Designer やその他の PowerCenter クライアントアプリケーションは、マッピングでこれらの変更の影響を追跡します。 Repository Manager によってマッピングの状態が表示されるので、マッピングが有効かどうかを確認することができます。無効なマッピングがあった場合、そのマッピングを開いて検証を行うと、出力ウィンドウにエラーメッセージが表示されます。

データフロー検証

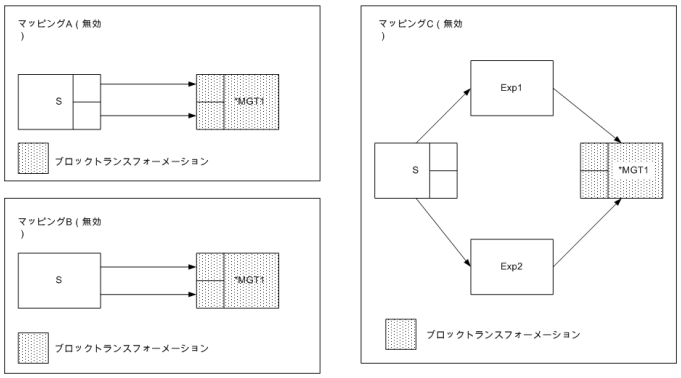
マッピングを検証または保存した場合、Designer により、Integration Service がすべてのソースをブロックすることなく、ターゲットロード順グループ内のすべてのソースからターゲットにデータが流れることが検証されます。

ブロッキングトランスフォーメーションを含むマッピングは、以下のマッピング設定のいずれかの場合、実行時に異常停止することがあります。

- 1 つのソースパイプラインを、ブロッキングトランスフォーメーションにおける複数の入力グループに接続した場合。
- ソースおよびトランスフォーメーションをターゲットのロード順グループに接続し、複数のブロッキングトランスフォーメーションがすべてのソースパイプラインをブロックできるようにした場合。セッションで使用されるソースデータに応じて、ブロッキングトランスフォーメーションは、あるソースからの行を待つ間、別のソースからのデータをブロックします。

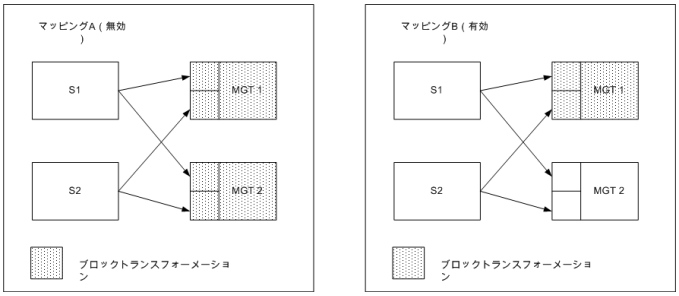
これらの設定のいずれかを持つマッピングを保存または検証すると、Designer はマッピングに無効の印を付けます。マッピングがデータフロー検証に違反しているために Designer がマッピングに無効の印を付ける場合、マッピングの設定を変更するか、可能ならば非ブロッキングトランスフォーメーションを使用する必要があります。

以下の図に、1 つのソースからブロッキングトランスフォーメーションの複数の入力グループにデータが提供されているために無効になっているマッピングを示します。



マッピングを有効にするには、MGT1 に対して非ブロッキングトランスフォーメーションを使用するか、または、同一のソースの 2 つのインスタンスを作成して、これらをブロッキングトランスフォーメーションに接続します。

以下の図に、2 つの類似するマッピングを示します。1 つは有効、1 つは無効です。



マッピング A には、データをブロックする 2 つのマルチグループのトランスフォーメーション MGT1 および MGT2 が含まれています。このセッションが実行できる場合、MGT1 は、S2 からの行を待つ間 S1 からのデータをブロックできます。また MGT2 は、S1 からの行を待つ間 S2 からのデータをブロックします。ブロッキングトランスフォーメーションは両方のソースパイプラインをブロックし、セッションはハングします。したがって、Designer はマッピングに無効の印を付けます。

マッピング B には、データをブロックするマルチグループのトランスフォーメーション、MGT1 が含まれています。ブロッキングトランスフォーメーションは、すべての入力グループをブロックすることはできません。そのため、MGT1 は S1 または S2 をブロックしますが、両方はブロックしません。MGT2 はブロッキングトランスフォーメーションではありません。そのためデータをブロックしません。したがって、このセッションは実行中にブロックによりハングしません。Designer はマッピングに有効の印をつけます。

マッピングの検証手順

Designer でマッピングに関する作業を実行している場合、そのマッピングを検証できます。また、［リポジトリ］-［保存］をクリックした場合、Designer により最後に保存してからのマッピングがすべて検証されます。 マッピングを検証または保存すると、出力ウィンドウに検証の結果が表示されます。Repository Manager にも、マッピングが有効かどうかが表示されます。

マッピングを検証するには、マッピングをチェックアウトして開き、［マッピング］-［検証］の順に選択します。

出力ウィンドウを開いていない場合、［表示］-［出力ウィンドウ］を選択します。エラーを確認し、マッピングの修正方法を決定します。

複数のマッピングの検証

ワークスペースに取り込むことなく複数のマッピングを検証できます。複数のマッピングを検証するには、クエリー結果の表示または依存性表示リストからマッピングを選択して検証する必要があります。

注: Repository Manager を使用している場合、ナビゲータから複数のマッピングを選択して検証できます。

検証の結果、ステータスが無効から有効に変わったマッピングを保存できるほか、オプションでチェックインすることもできます。

複数のマッピングを検証するには：

1. クエリーリストまたは依存性表示リストからマッピングを選択します。
2. 選択したマッピングの 1 つを右クリックし、［検証］を選択します。
   ［オブジェクトの検証］ダイアログボックスが表示されます。
3. 検証したオブジェクトを保存するか、そのオブジェクトをチェックインするかどうかを選択します。

ワークフロー生成ウィザードの使用

ワークフロー生成ウィザードを使用して、マッピングからセッションおよびワークフローを生成します。マッピングでは、リレーショナルファイルまたはフラットファイルのソースおよびターゲットを使用できます。

生成するものに応じて、以下のいずれかの場所からワークフロー生成ウィザードを起動します。

- **［メニュー］オプション。**ワークフローの生成ウィザードを使用して、1 つのマッピングに対してワークフローおよびセッションを 1 つ作成します。
- **マッピングテンプレートのインポートウィザード。**ワークフローの生成ウィザードを使用して、複数のマッピングに対して複数のワークフローおよびセッションを作成します。

ワークフロー生成ウィザードを使用する前に、マッピングが有効であり、Integration Service および接続オブジェクトが作成されていることを確認します。また、マッピングでリレーショナルファイルまたはフラットファイルのソースおよびターゲットが使用されていることを確認します。

ワークフロー生成ウィザードでは、以下の手順を実行します。

1. 生成したいセッションまたはワークフローのタイプを指定します。
2. Integration Service、接続設定、およびワークフロー名とセッション名の接頭語を指定します。この手順で指定する値は、設定するすべてのセッションおよびワークフローに適用されます。
3. ワークフロー名、セッション名、および Integration Service を変更します。接続設定を設定することもできます。この手順で指定する値は、設定するセッションおよびワークフローに適用されます。

ウィザードの最後のページには、生成されたワークフローやセッション、およびワークフローとセッションのステータスのリストが表示されます。

注: ワークフローの生成ウィザードで作成されたセッションで接続変数を使用し、ワークフローの生成ウィザードがリポジトリのセッションを作成した後に、Workflow Manager でセッションプロパティを編集します。

ワークフロー生成ウィザードの手順

ワークフロー生成ウィザードに表示されるオプションは、生成するワークフローやセッションのタイプおよびウィザードの起動方法によって異なります。

ワークフロー生成ウィザードを使用するには：

1. Mapping Designer ワークスペースでマッピングを開き、マッピングを選択します。［マッピング］-［ワークフローの生成］の順に選択します。

   また、ワークフロー生成ウィザードは、マッピングテンプレートのインポートウィザードから起動することもできます。
2. 生成したいワークフローまたはセッションのタイプを指定します。
   - 再利用可能なセッション
   - 再利用可能なセッションを使用するワークフロー
   - 再利用不可能なセッションを使用するワークフロー
3. ［次へ］をクリックします。
4. Integration Service、接続設定、およびワークフロー名とセッション名の接頭語を指定します。
5. 接続オブジェクトを変更するには、［接続オブジェクト］フィールドをクリックします。［開く］ボタンをクリックして、接続ブラウザを開き、接続オブジェクトを指定します。

   フラットファイルのファイル名を変更するには、［接続オブジェクト］フィールドでファイル名を編集します。
6. ［次へ］をクリックします。

7. ワークフロー名、セッション名、または Integration Service を変更します。
8. ［設定］ボタンをクリックして、接続設定を設定します。

   ［ワークフロー設定］ダイアログボックスには、以下のタブが表示されます。

タブ	説明
接続	ソース、ターゲット、およびトランスフォーメーションの接続情報を設定します。
プロパティ	ソースセッションおよびターゲットセッションのプロパティを設定します。
リーダー/ライター	マッピング内のソースインスタンスおよびターゲットインスタンスに対し、Reader と Writer を設定します。

9. ［次へ］をクリックします。
10. 状態を確認し、［完了］をクリックします。

マッピングのトラブルシューティング

マッピングを保存しようとしたが、Designer がこのマッピングにエラーがあることを示しました。

マッピングの保存を行うと、Designer は、エラー（ターゲットがどのソースからもデータを受け取らないなど）がないかを検査します。無効なマッピングでも保存できますが、こうしたマッピングを使用してセッションを実行することはできません。

マッピング内で 2 つのポートを接続できません。

ポートを接続すると、Designer によって検証が実行されます。入力ポートを入力ポートに接続したり、データタイプの一致しない 2 つのポートを接続したりすることはできません。Designer が接続の確立を禁止した場合には、表示されたエラーメッセージを確認してください。

1 つのターゲットに複数のソースを接続できません。

これは許可されていません。考えられる対策を次に示します。

1. ターゲットはすべて再利用可能です。同一のターゲットを複数回マッピングに追加することができます。それぞれのソース修飾子をそれぞれのターゲットに接続してください。
2. ソース修飾子トランスフォーメーションで複数のソースを結合します。その後、SQL クエリーから WHERE 句を削除してください。
3. ジョイナトランスフォーメーションで複数のソースを結合してください。

クリックとドラッグを行って、ポート間の接続を作成しようとしたが、Designer は作成ではなくポートのコピーを行ってしまいます。

［レイアウト］-［カラムのリンク］を選択して、作業モードを変更してください。この状態でカラム間をドラッグすると、Designer は選択したポートをコピーする代わりにリンクを行います。

マッピングの検証を行ったが、結果が確認できません。

マッピングの検証を行う際には、出力ウィンドウを必ず開いてください。［表示］-［出力ウィンドウ］を選択して、検証結果を確認してください。

ユーザー作成のクエリーを入力しましたが、ワークフローを実行してもこのユーザー作成のクエリーが動作しません。

必ずソース修飾子に対してこの設定をテストしてから、ワークフローを実行してください。ソース修飾子に戻り、ユーザー作成のクエリーを入力したダイアログを再度開いてください。データベースに接続し、［検証］をクリックすれば、SQL の確認ができます。Designer は、入力した SQL にエラーがあれば、その旨を表示します。詳細情報が必要な場合は、セッションログを確認してください。

第 6 章

マップレット

この章では、以下の項目について説明します。

マップレットの概要

マプレットとは、Mapplet Designer を使って作成できる再利用可能なオブジェクトです。マプレットには一連のトランスフォーメーションが格納されており、そのトランスフォーメーションロジックを複数のマッピングで再利用できます。

例えば、一連の次元キーが必要なファクトテーブルがいくつかある場合、一連のルックアップトランスフォーメーションが含まれるマプレットを作成して、各次元キーの検索ができます。そして、それぞれのファクトテーブルマッピングでそのマプレットを使用すれば、各マッピングで同じルックアップロジックを再作成する必要はありません。

マッピング内でマプレットを使用する際には、そのインスタンスを用います。再利用可能なトランスフォーメーションと同じく、マプレットに施した変更はそのインスタンスすべてに引き継がれます。

マプレットを使用すると、以下のようにマッピングを単純化することができます。

- **ソース定義の盛り込み。**複数のソース定義およびソース修飾子を使用して、マッピングのソースデータを提供します。
- **マッピングのソースからのデータの受け入れ。**マップレットでマッピングからデータを受け取る場合、入力トランスフォーメーションを使用してソースデータを受け取ります。
- **複数トランスフォーメーションの盛り込み。**マップレットに含めるトランスフォーメーションの数に制限はありません。
- **複数トランスフォーメーションへのデータの受け渡し。**マップレットを作成して、複数トランスフォーメーションへデータを渡すことができます。 マップレット内の各出力トランスフォーメーションは、マップレット内の 1 つの出力グループを表します。
- **未使用ポートの提供。**マッピング内のすべてのマップレット入力および出力ポートを接続する必要はありません。

マップレットの入力および出力について

マッピング内でマプレットを使用するためには、マプレットの入力と出力を設定する必要があります。ユーザーが設定したトランスフォーメーションロジックに加え、マプレットは以下のコンポーネントを持ちます。

- **マップレットの入力。** ソース定義や入力トランスフォーメーション、またはその両方を使用して、マプレットにデータを渡すことができます。入力トランスフォーメーションを使用する場合は、マッピング内のソースパイプラインに接続します。
- **マップレットの出力。**それぞれのマップレットは、マップレットからマッピングにデータを渡すために 1 つ以上の出力トランスフォーメーションを含む必要があります。
- **マップレットポート。**マップレットポートは、Mapping Designer でのみ表示されます。 マプレットポートは、入力トランスフォーメーションの入力ポートと出力トランスフォーメーションの出力ポートから構成されます。入力に関して入力トランスフォーメーションの代わりにソース定義を使用する場合、マプレットはマッピングに入力ポートを含みません。

マップレットの入力。

マプレットは、ソース定義やマプレット内の入力トランスフォーメーション、またはその両方を入力元とすることができます。1 つのマプレット内に複数のパイプラインを作成することができます。複数のソース定義およびソース修飾子、または入力トランスフォーメーションを使用します。また、ソース定義と入力トランスフォーメーションを組み合わせて使用することもできます。

マップレット入力のソース定義の使用

マプレットで 1 つ以上のソース定義を使用して、ソースデータを提供します。マッピング内でマップレットを使用する場合、マップレットはマッピングパイプラインにおける最初のオブジェクトとなり、入力ポートを含みません。

マップレット入力のための入力トランスフォーメーションの使用

マプレットがマッピング内のソースからの入力を受け取るようにしたい場合はマプレット内で入力トランスフォーメーションを使用します。マッピング内でマプレットを使用すると、入力トランスフォーメーションは入力ポートを提供し、これによりマプレット内でデータを受け渡すことが可能になります。マプレット内の別のトランスフォーメーションに接続した入力トランスフォーメーションの各ポートがマプレットの入力ポートとなります。入力トランスフォーメーションは、1 つのアクティブソースからのデータを受け取ることができます。接続されていないポートは Mapping Designer に表示されません。

入力トランスフォーメーションはマプレット内の複数のトランスフォーメーションに接続することができます。ただし、入力トランスフォーメーションの 1 つのポートをマプレット内の複数のトランスフォーメーションに接続することはできません。

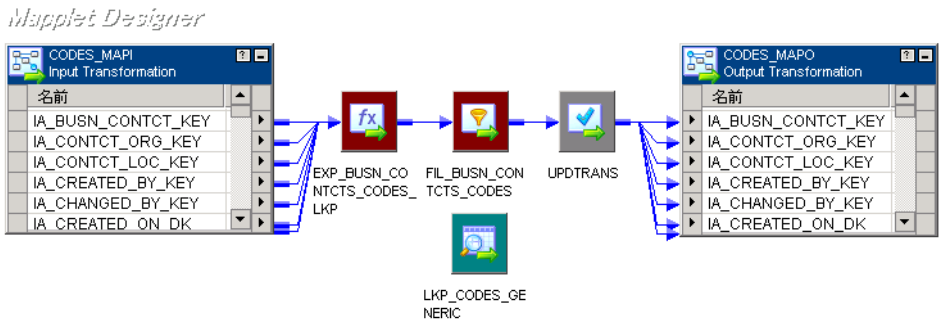
マップレットの出力。

マプレット内の出力トランスフォーメーションは、マプレットを介してマッピングにデータを渡すために使用します。マプレットは、少なくとも 1 つのポートがマプレット内で接続されている 1 つ以上の出力トランスフォーメーションを含む必要があります。出力トランスフォーメーション内で接続されている各ポートは、マッピング内でマップレットの出力ポートとして表示されます。 マプレット内の各出力トランスフォーメーションは、マッピング内で出力グループとして表示されます。出力グループは、データをマッピング内の複数のパイプラインに渡すことができます。

マップレットの入力と出力の表示

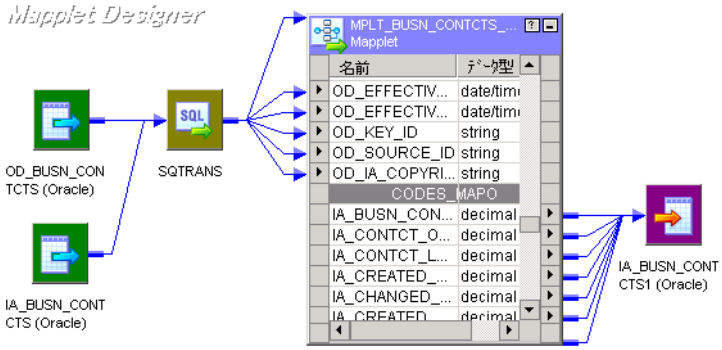
マップレットおよびマップレットポートは、Mapplet Designer と Mapping Designer で異なって表示されます。

以下の図に、入力トランスフォーメーションと出力トランスフォーメーションの両方を持つマップレットを示します。



マッピング内でマプレットを使用した場合、マプレットオブジェクトは、入力トランスフォーメーションおよび出力トランスフォーメーションからのポートだけを表示します。これらのポートは、マプレット入力ポートおよびマプレット出力ポートと呼ばれます。

以下の図に、Mapping Designer 内の同じマップレットを示します。

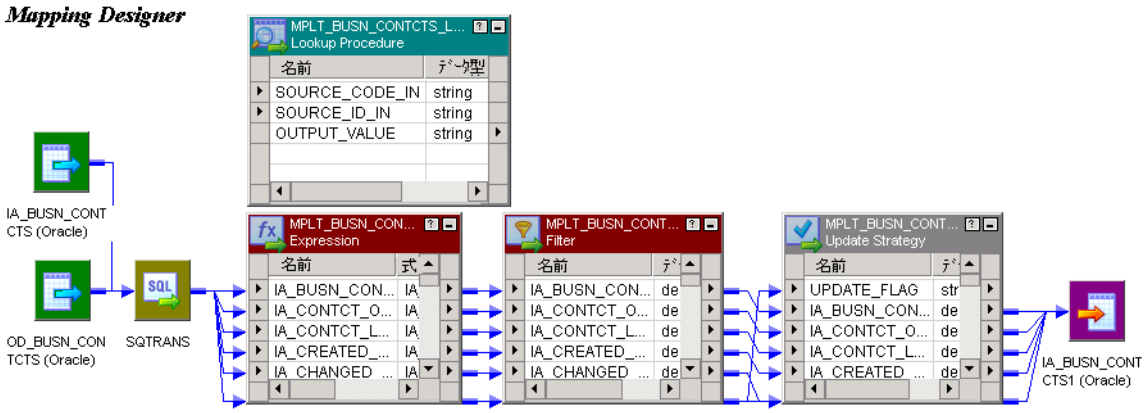


このマップレットには、入力トランスフォーメーションからの入力ポートが表示されます。 CODES_MAP0 出力トランスフォーメーションからの出力ポートは、入力ポートの下に表示されます。

Mapping Designer でマップレットを選択し、［マッピング］-［展開］をクリックすることにより、マップレットを展開することができます。 これにより、マッピング内のマプレットが展開されて表示されます。展開されたマプレット表示内のトランスフォーメーションのアイコンは淡色表示されます。

マプレットおよびマッピング内のすべてのトランスフォーメーションを開いたりアイコン化することができます。マプレットを展開した状態では、プロパティの編集、別のフォルダへの移動、またはリポジトリの保存を行うことはできません。

以下の図に、Mapping Designer 内で展開されたマップレットを示します。



展開されたマッピングでは、入力トランスフォーメーションおよび出力トランスフォーメーションは表示されません。

Mapplet Designer の使用

マプレットを作成したら、Mapplet Designer でこのマプレットの検証や編集ができます。また、Designer を使用して、マプレットのコピー、エクスポートとインポート、マプレット内のポートのリンク、マプレットへのショートカットの作成、およびリポジトリからのマプレットの削除ができます。

Mapplet Designer でマプレットを作成および設定するには、以下の手順を実行します。

1. マプレットを作成します。Mapplet Designer のメニューで、［マプレット］-［作成］を選択します。推奨されるマプレットの命名規則は、mplt_*マプレット名*です。
2. マプレットのトランスフォーメーションロジックを作成します。マッピングの場合と同じ方法でトランスフォーメーションを作成してリンクします。
3. マップレットポートを作成します。

マップレットの作成

マプレットはそのトランスフォーメーションに応じて、アクティブな場合とパッシブな場合があります。アクティブなマプレットには 1 つ以上のアクティブなトランスフォーメーションが含まれます。パッシブなマプレットにはパッシブなトランスフォーメーションしか含まれません。マッピングでマプレットを使用する場合には、マプレットの種別に応じて、トランスフォーメーションに関するすべての規則がマプレットに適用されます。例えば、アクティブなトランスフォーメーションと同様に、アクティブなマプレットからのデータを別のデータフローに連結することはできません。

マプレットにトランスフォーメーションを追加する場合には、以下の規則およびガイドラインに従ってください。

- シーケンスジェネレータトランスフォーメーションを使用する場合は、再利用可能なシーケンスジェネレータトランスフォーメーションを使用する必要があります。
- Stored Procedure トランスフォーメーションを使用する場合は、［Stored Procedure Type］を［*Normal*］に設定する必要があります。
- PowerMart 3.5 スタイルの LOOKUP 関数をマプレットに含めることはできません。

- 以下のオブジェクトをマプレットに含めることはできません。
  - ノーマライザトランスフォーメーション
  - COBOL ソース
  - XML ソース修飾子トランスフォーメーション
  - XML ソース
  - ターゲット定義
  - 他のマプレット

マプレット内の再利用可能なトランスフォーメーションおよびショートカットを使用できますが、マプレットの有効性を保護するためにも、できればトランスフォーメーションのコピーを使用してください。再利用可能なトランスフォーメーションやショートカットは、元のトランスフォーメーションの変更を引き継ぎます。これにより、マプレットとそれを使用するマッピングが無効になる場合もあります。

マップレットの検証

マプレットを保存すると、Designer はこのマプレットの検証を行います。また、［マプレット］-［検証］メニューコマンドを使用して、マプレットを検証できます。マプレットの検証を行うと、Designer は出力ウィンドウにそのマプレットについて該当するメッセージをすべて出力します。

Designer は、マッピングの検証と同じ方法でマプレットのパイプラインを検証します。さらに Designer は、以下に示すマプレット固有のチェックも行います。

- マプレットには入力トランスフォーメーションまたはソース定義を含めることができ、その少なくとも 1 つのポートがマプレット内のトランスフォーメーションに接続されている。
- マプレットに 1 つ以上の出力トランスフォーメーションが含まれ、その少なくとも 1 つのポートがマプレット内のトランスフォーメーションに接続されている。

マップレットの編集

Mapplet Designer でマプレットの編集ができます。マプレットを保存すると、Designer は変更内容を検証します。マプレットに変更を保存すると、このマプレットのインスタンスおよびマプレットへのショートカットすべてに変更が反映されます。その結果、当該マプレットを使用したマッピングが無効となる場合があります。

マプレットへの変更により影響を受けるマッピングまたはショートカットを確認するには、ナビゲータ内でマプレットを選択して右クリックし、［依存関係］を選択します。または、メニューで［マプレット］-［依存関係］を選択します。

ただし、以下の変更については、既存のマッピングやセッションの有効性に影響を及ぼすことは*ありません*。

- 入力ポートや出力ポートの追加。
- ポート名やコメントの変更。
- 入力または出力トランスフォーメーションの名前やコメントの変更。
- トランスフォーメーションの名前、コメント、またはプロパティの変更。
- マプレット内のトランスフォーメーションに対するポートのデフォルト値の変更。
- マプレット内のトランスフォーメーションの追加や削除（ただしマプレットの種別をアクティブからパッシブまたはパッシブからアクティブに変更しない場合）。

マッピングに使用されているマプレットを編集する場合には、以下の規則およびガイドラインに従ってください。

- **マップレットからポートを削除してはいけません。** 入力または出力トランスフォーメーションへのリンクを削除したり、入力または出力トランスフォーメーションに接続されているポートを削除すると、Designer はマッピング内のマップレットポートを削除します。
- **マップレットポートのデータタイプ、精度、位取りを変更してはいけません。** マプレットのポートのデータタイプ、精度、および位取りは、マプレット内でそのポートが接続されているトランスフォーメーションポートによって定義されています。そのため、マプレットを編集して、入力トランスフォーメーションまたは出力トランスフォーメーションのポートに*接続*されているポートのデータタイプ、精度、または位取りを変更すると、マプレットのポートも変更されます。
- **マップレットのタイプを変更してはいけません。**アクティブなマップレットからアクティブなトランスフォーメーションをすべて削除すると、マップレットはパッシブになります。 アクティブなトランスフォーメーションをパッシブなマップレットに追加すると、このマップレットはアクティブになります。

マップレットとマッピング

以下のマッピングタスクは、マップレットに対しても実行できます。

- **トレースレベルの設定。** マッピングの場合と同様に、マップレット内の個々のトランスフォーメーションにトレースレベルを設定することができます。
- **マップレットのコピー。**他のリポジトリオブジェクトと同様に、あるフォルダから別のフォルダにマップレットをコピーすることができます。 コピーしたマップレットは新しいフォルダの［マップレット］ノードに表示されます。

  元のマプレットを上書きせずにマッピングを変更する場合、［マプレット］-［別名でコピー］を選択してマプレットをコピーできます。
- **マップレットのエクスポートおよびインポート。** Designer では、XML ファイルへのマプレットのエクスポート、または XML ファイルからのマプレットのインポートを実行できます。エクスポートおよびインポート機能を使用して、別のリポジトリにマプレットをコピーすることができます。
- **マップレットの削除。** マプレットを削除すると、当該マプレットのすべてのインスタンスが削除されます。そうすると、マプレットのインスタンスやショートカットを含む各マッピングが無効となります。
- **マップレットの比較。** 2 つのマップレットを比較してその相違を検索することができます。例えば、異なるフォルダに同じ名前のマプレットがある場合、それらを比較することによって違いがあるかどうかを確認できます。
- **マップレット内のインスタンスの比較。** マプレット内のインスタンス同士を比較して、それらが共通の属性を持つかどうかを確認できます。例えば、ソースインスタンス同士を比較したり、トランスフォーメーション同士を比較できます。マプレット内のインスタンスは、マッピング内のインスタンスを比較するのと同じ方法で比較できます。
- **マップレットへのショートカットの作成。**マップレットが共有フォルダ内に存在する場合には、このマップレットへのショートカットを作成することができます。 マッピングでマップレットへのショートカットを使用すると、このショートカットはマップレットへの変更内容をすべて継承します。ただし、Integration Service がショートカットを使用したワークフローを実行するまでは、こうした変更は表示されない場合もあります。 そのため、マップレットの編集を予定しない場合に限り、マップレットへのショートカットを使用します。
- **説明の追加。**マッピングの場合と同様に、Mapplet Designer のマップレットに説明を追加することができます。 マッピング内のマプレットのインスタンスにも説明を追加することができます。説明を追加するときに、文書ファイルへのリンクを作成することもできます。このリンクが有効な URL またはファイルパスでなければ、ビジネス文書を参照できません。
- **ポートへのリンクの表示。**マッピング内のポートへのリンクを表示するのと同様に、マップレット内のポートへのリンクを表示できます。 前方パス、後方パス、またはその両方を表示できます。

- **ポート属性のプロパゲート。**マッピングでポート属性をプロパゲートするのと同じ方法で、マップレットでポート属性をプロパゲートできます。 属性を前方、後方、または両方向にプロパゲートすることができます。

マッピングでのマップレットの使用

マッピングでは、マプレットの入力ポートおよび出力ポートをこのマッピング内の別のトランスフォーメーションに接続することができます。マッピング内のすべてのマプレットポートを接続する必要はありません。ただし、マプレットに SQL オーバーライドが含まれている場合は、マッピング内のすべてのマプレット出力ポートを接続する必要があります。

再利用可能なトランスフォーメーションと同様に、マプレットをマッピング内にドラッグすると、Designer はこのマプレットのインスタンスを作成します。マッピング内のマプレットのインスタンスについて、コメントを入力することができます。Mapping Designer では、それ以外にマプレットを編集することはできません。

Mapplet Designer でマプレットを編集すると、マプレットのすべてのインスタンスがその変更を継承します。

PowerCenter Repository Reports には、特定のマプレットを使用しているマッピングをすべて表示するマプレットリストレポートがあります。

マプレットを使用するには、以下の手順を実行します。

1. 該当するマプレットをマッピング内にドラッグします。
2. マプレットに入力ポートがある場合、少なくとも 1 つのマプレット入力ポートをマッピング内のトランスフォーメーションに接続します。
3. マプレット出力ポートを 1 つ以上、マッピング内のトランスフォーメーションに接続します。

マップレットポートの作成と設定

マプレットのトランスフォーメーションロジックを作成したあとは、マプレットのポートを作成できます。マプレットがソース定義を含まない場合は、入力トランスフォーメーションを使用してマプレット入力ポートを定義します。また、出力トランスフォーメーションを使用して出力ポートグループを作成します。入力または出力トランスフォーメーション内で接続されているポートだけが、マッピング内でマプレット入力または出力ポートとなります。マッピングでマプレットを使用する場合、接続されていないポートは表示されません。

以下の方法で、マプレットポートを作成することができます。

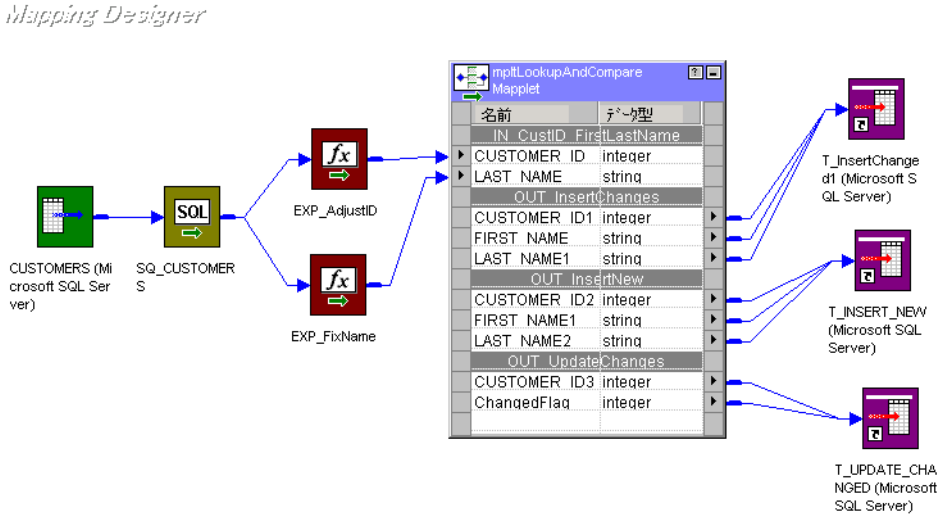
- **入力/出力トランスフォーメーション内でのポートの手動作成。**入力トランスフォーメーションおよび出力トランスフォーメーション内でポート名を作成できます。 各ポート名について、説明を入力することもできます。ポートをマプレット内のトランスフォーメーションに接続しないうちは、データタイプ、精度、位取りは定義されません。
- **別のトランスフォーメーションからのポートのドラッグ。**ポートを別のトランスフォーメーションから入力または出力トランスフォーメーションにドラッグすることにより、入力ポートまたは出力ポートを作成できます。 新しいポートは元のポートのポート名、説明、データタイプ、位取りを継承します。トランスフォーメーション内で、新しいポート名および説明を編集できます。ポートの接続を変更すると、Designer は入力または出力トランスフォーメーションのポートを更新し、新しい接続の属性に一致させます。

マッピングでマプレットを使用する場合、マプレットで利用可能な各ポートのデータタイプ、精度、位取りを確認することができます。

マップレット入力ポートへの接続

入力ポートのあるマプレットをマッピング内で使用する場合は、マプレットの入力ポートをマッピングパイプラインに接続します。データをマプレットに渡すことができるのは、そのソースが単一のアクティブなトランスフォーメーションの場合だけです。

例えば、以下の図のマプレット mpltLookupAndCompare は 2 つの式トランスフォーメーションからデータを受け取りますが、その理由は、この 2 つのトランスフォーメーションからのデータが単一のソース修飾子から発しているからです。SQ_CUSTOMERS という名前のソース修飾子は、マプレットにソースデータを提供するアクティブなトランスフォーメーションです。



マップレット出力グループへの接続

マッピング内でマプレットを使用した場合、各出力トランスフォーメーションは出力グループとして表示されます。マプレット出力ポートはマッピングパイプラインに接続します。［オートリンク］を使用して、ポートを接続します。

マッピング内でマプレット出力ポートを接続する場合は、以下の規則およびガイドラインに従ってください。

- マプレットにデフォルトの SQL キューを上書きするようなソース修飾子が含まれていると、すべてのソース修飾子出力ポートをそのマプレット内の次のトランスフォーメーションに接続する必要があります。
- マプレットに複数のソース修飾子が含まれている場合は、ジョイナトランスフォーメーションを使用して出力を 1 つのパイプラインに結合します。
- マプレットにソース修飾子が 1 つしかない場合は、マプレット出力ポートをそれぞれ別個のパイプラインに接続する必要があります。この場合、ジョイナトランスフォーメーションを使用して出力を結合することはできません。

  各パイプラインの結合が必要な場合には、マッピングを 2 つ作成して、以下のことを行ってください。

  - マプレットを第 1 のマッピングで使用し、各パイプラインのデータを別のターゲットに書き込みます。
  - 第 2 のマッピングで各ターゲットをソースとして使用してデータを結合し、そのあとで必要なトランスフォーメーションを実行します。

マプレットの表示

マッピング内でマプレットを使用した場合、Designer には、マプレットの入力ポートと出力ポートだけを含むマプレットオブジェクトが表示されます。ただし、メニューの［マッピング］-［展開］を選択すると、マプレットを展開できます。

マプレットを展開すると、マプレットトランスフォーメーションを含むマプレット全体が表示されます。入力トランスフォーメーションと出力トランスフォーメーションは表示されません。マッピングは展開表示できますが、この状態でマッピングを編集することはできません。続けてマッピングを作成するには、［マッピング］-［展開解除］を選択します。

ターゲットロードプランの設定

マッピング内でマプレットを使用する場合、Mapping Designer を使用してマプレット内のソースのターゲットロードプランを設定できます。

パイプラインのパーティション化

パーティション化オプションを使用している場合は、パイプライン内のパーティション数を増やすことにより、セッションのパフォーマンスを向上させることができます。パーティションの数を増やすことにより、Integration Service はソースへの複数の接続を作成し、ソースデータのパーティションを同時に処理することができます。

セッションを作成した場合、Workflow Manager によりマッピング内の各パイプラインがパーティション化について検証されます。Integration Service により、パーティション化されたデータが処理される際に、データの一貫性が保持される場合、パイプラインに複数のパーティションを指定することができます。

マップレットには、パーティション化に関していくつかの制限があります。

マップレットに関するルールおよびガイドライン

ここでは、この章でこれまでに紹介した規則およびガイドラインを要約します。

- 入力トランスフォーメーションはマプレット内の複数のトランスフォーメーションに接続することができます。ただし、入力トランスフォーメーションの 1 つのポートをマプレット内の複数のトランスフォーメーションに接続することはできません。
- 入力トランスフォーメーションは、1 つのアクティブソースからのデータを受け取る必要があります。
- マプレットでは、マプレット内のトランスフォーメーションに接続されているポートを少なくとも 1 つ持つ入力トランスフォーメーションまたはソース定義が、少なくとも 1 つは格納されていることが必要です。
- マプレットは、1 つ以上のポートがマッピング内の他のトランスフォーメーションに接続されている出力トランスフォーメーションを 1 つ以上含まなければなりません。
- マプレットにデフォルトの SQL キューを上書きするようなソース修飾子が含まれていると、すべてのソース修飾子出力ポートをそのマプレット内の次のトランスフォーメーションに接続する必要があります。
- マプレットに複数のソース修飾子が含まれている場合は、ジョイナトランスフォーメーションを使用して出力を 1 つのパイプラインに結合します。マプレットにソース修飾子が 1 つしかない場合は、マプレット出力ポートをそれぞれ別個のパイプラインに接続する必要があります。この場合、ジョイナトランスフォーメーションを使用して出力を結合することはできません。
- マプレットを編集してマプレットの種別をパッシブからアクティブに変更すると、マッピングが無効になる場合があります。

- マッピング内でマプレットを使用しているときにマプレット内のポートを削除すると、マッピングが無効になる場合があります。
- マッピング内でマプレットを使用している場合は、マプレットポートのデータタイプ、精度、位取りを変更してはいけません。
- シーケンスジェネレータトランスフォーメーションを使用する場合は、再利用可能なシーケンスジェネレータトランスフォーメーションを使用する必要があります。
- Stored Procedure トランスフォーメーションを使用する場合は、［Stored Procedure Type］を［*Normal*］に設定する必要があります。
- PowerMart 3.5 スタイルの LOOKUP 関数をマプレットに含めることはできません。
- 以下のオブジェクトをマプレットに含めることはできません。
  - ノーマライザトランスフォーメーション
  - COBOL ソース
  - XML ソース修飾子トランスフォーメーション
  - XML ソース
  - ターゲット定義
  - セッション実行前および実行後に起動されるストアドプロシージャ
  - 他のマプレット

マップレットに関するヒント

入力および出力トランスフォーメーションにコメントを入力してください。

マッピング内で、マプレットの入力トランスフォーメーションまたは出力トランスフォーメーションの名前の上にポインタを合わせると、そのトランスフォーメーションについてのコメントが表示されます。入力および出力トランスフォーメーションの［説明］フィールドに説明コメントや指示を記入することにより、マプレットとそのポートの用途が明確になります。ビジネス文書へのリンクを含めることもできます。

必要な各出力グループについて出力トランスフォーメーションを作成してください。

データをそれぞれのマプレット出力グループから別のマッピングデータフローに渡すことができます。必要な各出力グループについて出力トランスフォーメーションを作成してください。

既存のマッピングからマプレットを作成するには、マッピングから Mapplet Designer にオブジェクトをコピーします。

シーケンスジェネレータキャッシュを適切に設定してください。

マプレットに再利用可能なシーケンスジェネレータトランスフォーメーションを含めることができます。マプレットをいくつかのマッピングで使用していて、各マッピングでセッション中に数多くの値を使用する場合には、再利用可能なシーケンスジェネレータのキャッシュサイズを適切に設定することにより、未使用値の数を制限できます。

マッピング内で使用されているマプレットを編集するときに既存のマッピングの有効性を保つためには：

- 入力または出力トランスフォーメーション内の接続されているポートを削除してはなりません。
- 入力または出力トランスフォーメーション内の接続されているポートのデータタイプ、精度、または位取りを変更してはなりません。
- パッシブなマプレットをアクティブなマプレットに変更したり、アクティブなマプレットをパッシブなマプレットに変更したりしてはなりません。

第 7 章

マッピングパラメータおよび変数

この章では、以下の項目について説明します。

マッピングパラメータおよび変数の概要

Designer では、マッピングの柔軟性を高めるために、マッピングパラメータおよびマッピング変数を使用します。マッピングパラメータおよびマッピング変数は、マッピングおよびマップレット内の値を表します。

マッピング内でマッピングパラメータおよびマッピング変数を宣言すると、マッピングのパラメータまたは変数の値をセッションにおいて変更することによって、マッピングを再利用することができます。マッピングの再利用により、マッピングの特定の属性のみを変更する必要がある場合に、複数のマッピングを作成するというオーバーヘッドを削減できます。

マッピングパラメータまたはマッピング変数をマッピングで使用する場合、各マップレットまたはマッピング内で使用するマッピングパラメータまたは変数を最初に宣言します。次に、マッピングパラメータまたは変数の値を定義してから、セッションを実行します。

マッピングパラメータは、マップレットとマッピングで使用するために同じ名前で宣言できます。マップレット内のマッピングパラメータに値が指定されていない場合、そのパラメータはマッピング内の同名のマッピングパラメータの値を使用します。

マッピング内でマッピングパラメータおよびマッピング変数を使用して、データを差分抽出します。ソース修飾子トランスフォーメーションのソースフィルタにマッピングパラメータまたは変数を使用して、データを差分抽出するための開始タイムスタンプと終了タイムスタンプを決定します。

例えば、統合サービスによって以前のセッションで読み込まれた最後の行のタイムスタンプを保存する、ユーザー定義のマッピング変数$$LastUpdateDateTime を作成することができます。ソースフィルタで、開始タイムスタンプに$$LastUpdateDateTime を使用し、終了タイムスタンプに組み込み変数$$$SessStartTime を使用します。ソースの SALES.sales_datetime カラムに基づいてデータを差分抽出するには、以下のフィルタを使用します。

```
SALES.sales_datetime > TO_DATE ('$$LastUpdateDateTime') AND SALES.sales_datetime < TO_DATE ('$$$SessStartTime')
```

マッピングパラメータ

マッピングパラメータは、セッションを実行する前に定義することができる定数値を表します。マッピングパラメータの値は、セッション全体を通して一定です。

マッピングパラメータを使用する場合、マッピングまたはマプレット内でパラメータを宣言および使用します。そして、パラメータファイルにパラメータの値を定義します。Integration Service によってその値のパラメータへのすべての参照が評価されます。

例えば、同一セッションを使用し、各顧客の取り引きレコードを個々に抽出するとします。各顧客口座に対して別々のマッピングを作成する代わりに、1 つの顧客口座を示すマッピングパラメータを作成することができます。その後、ソースフィルタのパラメータを使用して、その顧客口座のデータのみ抽出できます。セッションを実行する前に、パラメータファイルにパラメータの値を入力します。

同じマッピングを再利用して他の顧客口座のレコードを抽出するには、パラメータファイルにパラメータの新しい値を入力し、セッションを実行します。または、各顧客口座に対してパラメータファイルを作成し、*pmcmd* を使用するたびに異なるパラメータファイルでセッションを開始することもできます。パラメータファイルを使用すると、個々の顧客口座の取り引きレコードを抽出するのに複数のマッピングおよびセッションを作成するというオーバーヘッドを削減できます。

各セッションの実行にあたって、マッピングパラメータに同じ値を使用したい場合は、全セッションに同じファイルを使用して実行します。セッションごとにマッピングパラメータの値を変更したい場合は、以下のいずれかの方法で行います。

- セッションごとにパラメータファイルを更新する。
- 異なるパラメータファイルを作成し、新しいファイルを使用するようにセッションを設定する。
- セッションプロパティからパラメータファイルを削除する。Integration Service はセッション実行前の変数割り当てにパラメータ値を使用します。セッション実行前の変数割り当てがない場合、Integration Service はマッピングでパラメータの設定済みの初期値を使用します。

マッピング変数

マッピングパラメータと異なり、マッピング変数によってセッションの実行中に変更できる値が表されます。Integration Service は、セッションが正常に実行されるたびに、そのセッションの最後にマッピング変数の値をリポジトリに保存し、その値を次回セッションを実行するときに使用します。

マッピング変数を使用する場合、マッピングまたはマップレットで変数を宣言してからマッピングで変数関数を使用して、変数の値を変更します。セッションの最初に Integration Service によって変数への参照が評価され、開始値が決定されます。 正常なセッションの最後に、Integration Service によって変数の最終値がリポジトリに保存されます。 次回セッションを実行する際に、Integration Service では保存された値の変数への参照が評価されます。 保存されている値を上書きするには、パラメータファイルの変数の開始値を定義するか、セッションプロパティのセッション実行前変数割り当てで値を割り当てます。

マッピング変数を使用して、ソースの差分読み込みを実行します。例えば、前述したマッピングパラメータ例の顧客口座に、001 から 065 までの番号が順番に付けられているとします。マッピングパラメータを作成する代わりに、初期値 001 でマッピング変数を作成することができます。マッピングでは、変数関数を使用して変数値を 1 つずつ増加させます。Integration Service はセッションの最初の実行時に、顧客口座 001 のレコードを抽出します。 このセッションの最後に、変数を 1 つ増やしその値をリポジトリに保存します。次回 Integration Service によってセッションが実行されるときに、次の顧客口座 002 のデータが抽出されます。また、その次のセッションによって顧客口座 003 のデータが抽出およびルックアップされるように変数値が増やされます。

マッピングパラメータおよび変数の使い方

マッピングパラメータおよび変数は、Mapping Designer または Mapplet Designer で作成することができます。作成したマッピングパラメータおよび変数は、式エディタの［変数］タブに表示されます。作成したマッ

ピングパラメータおよび変数は、マプレットまたはマッピングの任意の式で使用できます。Designer は、マプレットおよびマッピングの式エディタでマッピングパラメータおよび変数を検証します。

マッピングパラメータおよびマッピング変数は、マプレットまたはマッピングのソース修飾子で使用します。ソース修飾子トランスフォーメーションでマッピングパラメータおよび変数を使用する場合、Designer はそれらを展開してから、検証のためにソースデータベースにクエリーを渡します。これにより、ソースデータベースはクエリーを検証できます。

Transformation Developer で再利用可能なトランスフォーメーションを作成する場合、マッピングパラメータまたはマッピング変数を使用します。再利用可能なトランスフォーメーションはマプレットやマッピング内に含まれていないので、Designer は再利用可能なトランスフォーメーションの式のマッピングパラメータまたは変数の使用法を検証します。再利用可能なトランスフォーメーションをマッピングまたはマプレットで使用する場合、Designer は式を再度検証します。パラメータまたは変数がマプレットまたはマッピングで定義されていない場合、または再利用可能なトランスフォーメーションでそれらが正しく使用されていない場合は、Designer はマプレットまたはマッピングの検証時にログにエラーを記録します。

Designer は再利用可能なトランスフォーメーションのマッピング変数を検証する際に、変数を Integer データタイプとして扱います。

マッピングパラメータおよび変数を、マプレットとマッピングの間で入れ替えて使用することはできません。マッピングに対して宣言されたマッピングパラメータおよび変数は、マプレット内では使用できません。同様に、マプレットに対して宣言されたマッピングパラメータまたは変数は、マッピング内で使用できません。

初期値とデフォルト値

マッピングパラメータまたは変数をマッピングまたはマップレットで宣言する際に、初期値を入力することができます。パラメータファイルにパラメータが定義されていない場合、Integration Service によりマッピングパラメータに対して設定された初期値が使用されます。 同様に、パラメータファイルに変数値の定義がなく、リポジトリに変数値が保存されていない場合は、Integration Service によりマッピング変数に対して設定された初期値が使用されます。

Integration Service で初期値が必要とされる場合で、パラメータまたは変数に対する初期値が宣言されていなかった場合は、Integration Service はパラメータまたは変数のデータタイプに基づくデフォルト値を使用します。

以下の表に、Integration Service によって、異なるタイプのデータに使用するデフォルト値が示されます。

データ	デフォルト値
文字列	空の文字列。
Numeric	0
日時	Integration Service に 4.0 との互換性が設定されている場合は、1/1/1753 A.D.または 1/1/1。

例えば、Integer マッピング変数$$MiscellaneousExpenses を使用して新しいマッピングを作成するとします。変数の初期値の設定や、パラメータファイルでの定義は行っていません。このマッピングで初めてセッションを実行する際は、Integration Service により数値データタイプに対するデフォルト値 0 が使用されます。

または、マッピングパラメータ$$MiscellaneousCosts を作成し、現在はソースデータに存在しなくても今後必要になる可能性のある追加の支出を表すとします。パラメータを Decimal データタイプに設定します。追加の支出がない場合に$$MiscellaneousCosts が 0 に評価されるように、初期値を 0 に設定します。

パラメータファイルにパラメータ値を定義しない限り、Integration Service により$$MiscellaneousCosts が 0 で置き換えられます。 マッピングの計算に雑費を含めたい場合は、パラメータファイルで$$MiscellaneousExpenses をその値に設定します。

文字列パラメータおよび変数の使用

文字列データタイプのマッピングパラメータおよび変数をソース修飾子トランスフォーメーションに入力する場合、ソースデータベースに対して適切な文字列識別子を使用します。Integration Service によりソース修飾子トランスフォーメーションでパラメータまたは変数が展開される場合、Integration Service によりパラメータまたは変数がその開始値で置き換えられ、展開したクエリがソースデータベースに渡されます。 ほとんどのデータベースについては、文字列値を一重引用符で囲む必要があります。

PowerCenter トランスフォーメーション言語を使用して文字列パラメータまたは変数を入力する場合、追加引用符は使用しません。 Integration Service により、PowerCenter トランスフォーメーション言語でマッピングパラメータおよび変数の命名構文が認識されます。 例えば、ソース修飾子トランスフォーメーションのフィルタで$$State という名前のパラメータを使用して、特定の状態の行を抽出します。

```
STATE = '$$State'
```

セッション中に、Integration Service によりパラメータが文字列で置き換えられます。 パラメータファイルで $$State を MD として定義した場合、Integration Service によりパラメータが以下のように置き換えられます。

```
STATE = 'MD'
```

次のように、PowerCenter トランスフォーメーション言語を使用して、フィルタトランスフォーメーションで同様のフィルタリングを実行することができます。

```
STATE = $$State
```

フィルタトランスフォーメーションでパラメータを一重引用符で囲んだ場合、Integration Service ではその文字列をそのまま「$$State」として読み込み、パラメータを「MD」とは置き換えません。

日付パラメータおよび変数

ソース修飾子トランスフォーメーションで datetime パラメータまたは変数を使用する場合、日付フォーマットをソースで使用されているフォーマットに変更しなければならない場合があります。

コードページ緩和

Integration Service を Unicode データ移動モードで実行する場合には、コードページの検証を緩和するように Integration Service を設定することができます。 ただし、以下の状況では予期しない結果が得られる場合もあります。

- Integration Service によりリポジトリに保存されるマッピング変数値は、リポジトリのコードページと互換性がない。

  例えば、リポジトリにより ISO 8859-1 Latin-1 コードページが使用されている場合に、コードページ検証が緩和されるように Integration Service を設定します。 マッピング変数の値に JapanEUC などの日本語の文字データが含まれている場合、リポジトリに保存されているマッピング変数の値は正しくない可能性があります。JapanEUC コードページから Latin1 コードページに変換すると、データが失われることがあります。保存するマッピング変数値がリポジトリのコードページと双方向の互換性を持つことを確認してください。

  Integration Service によりリポジトリにすべてのメタデータが書き込まれるようにするには、すべてのリポジトリメタデータで 7 ビット ASCII 文字を使用するか、そのリポジトリで UTF-16LE を使用します。

- パラメータファイルに、Integration Service コードページと互換性のない文字が含まれている。

  Integration Service により、Integration Service コードページを使用して、パラメータファイル内のデータが解釈されます。 例えば、Integration Service により ISO 8859-1 Latin-1 コードページが使用されている場合に、コードページ検証を緩和するように Integration Service を設定します。 パラメータファイルを作成して ISO 8859-7 などのギリシャ語の文字データを使用した場合、Integration Service によってファイルから読み込まれる値は正しくない可能性があります。 ISO 8859-7 コードページから Latin1 コードページに変換すると、データが失われることがあります。パラメータファイル内の文字が Integration Service コードページのサブセットであることを確認します。

マッピングパラメータ

Designer で、マプレットまたはマプレットにマッピングパラメータを作成することができます。作成したパラメータは式エディタに表示されます。作成したマッピングパラメータは、マプレットまたはマッピングの任意の式で使用できます。また、ソース修飾子フィルタ、ユーザー定義結合、または抽出上書き、および再利用可能なトランスフォーメーションの式エディタで使用することもできます。

セッションを実行する前に、セッションのパラメータファイルにマッピングパラメータ値を定義します。任意の定数値を使用します。セッション中に、Integration Service によりパラメータへの参照がすべて指定された値に評価されます。 パラメータファイルにパラメータが定義されていない場合、Integration Service によりパラメータのユーザー定義初期値が使用されます。 初期値が定義されていない場合、Integration Service によりマッピングパラメータのデータタイプに基づいたデフォルト値が使用されます。

パラメータファイルを編集するか、セッションが使用するパラメータファイルを変更することによって、マッピングパラメータをセッションごとに変更することができます。

データベースルックアップの代わりに、マッピングパラメータを使用することができます。例えば、月ごとの総収入を使用して計算を実行するとします。ルックアップトランスフォーメーションを使用してその情報のデータベーステーブルに接続する代わりに、総収入のマッピングパラメータを作成し、パラメータファイルの値を毎月更新すれば、現在の収入を反映させることができます。

また、マッピングパラメータをセッションパラメータと共に使用し、マッピングおよびセッションを再利用することもできます。例えば、異なるデータベースの同じテーブルにさまざまな州からの取り引きデータが格納されている場合に、州の売上税を適宜変更し、すべてのデータに対して同じ計算を実行するとします。各州に対するマッピングおよびセッションを個々に作成する代わりに、売上税のマッピングパラメータを含むマッピングを 1 つと、ソースデータベースの接続セッションパラメータを使用するセッションを作成できます。各状態ごとに異なるパラメータファイルを作成することができます。セッションを実行する前に *pmcmd* とは異なるパラメータファイル名を入力するか、Workflow Manager でセッションを編集すると、Integration Service が使用するパラメータファイルを変更できます。

マッピングパラメータを使用するには、以下の手順を実行します。

1. マッピングパラメータを作成する。
2. マッピングパラメータを使用する。
3. パラメータ値を定義する。

手順 1. マッピングパラメータの作成

任意のマッピングまたはマプレットのマッピングパラメータを作成することができます。マッピングパラメータは必要な数だけ宣言することができます。宣言したパラメータを、マッピングまたはマプレットで使用します。

マッピングパラメータを作成するには：

1. Mapping Designer で、［マッピング］-［パラメータと変数］をクリックします。あるいは、Mapplet Designer で［マップレット］-［パラメータと変数］の順にクリックします。
2. ［追加］ボタンをクリックします。
3. 次の情報を入力して、［OK］をクリックします。

フィールド	説明
名前	パラメータ名。パラメータ名$$*ParameterName*。パラメータ名は、$$を先頭に付け、続けて任意の英数字またはアンダスコア文字を使用します。
タイプ	変数またはパラメータ。［パラメータ］を選択してください。
データタイプ	パラメータのデータタイプ。有効なトランスフォーメーションデータタイプを選択してください。［Binary］または［Raw］以外の任意のデータタイプを使用します。
精度または位取り	パラメータの精度および位取り。
集計	変数に使用。
IsExprVar	Integration Service が式文字列でパラメータを展開する方法を決定します。true の場合は、Integration Service は式の解析前にパラメータを展開します。false の場合は、Integration Service は式の解析後にパラメータを展開します。デフォルトは False です。 **注:** このフィールドを True に設定した場合、パラメータデータタイプを String に設定する必要があります。String に設定しないと、Integration Service のセッションは失敗します。
初期値	パラメータの初期値。パラメータファイルにパラメータの値が設定されていない場合、Integration Service はセッション実行時にこの値をパラメータに使用します。この値も定義されていない場合、Integration Service はマッピング変数のデータタイプに基づいたデフォルト値を使用します。 日付/時刻パラメータの初期値には、以下のフォーマットのいずれかを使用します。 - MM/DD/RR - MM/DD/RR HH24:MI:SS - MM/DD/YYYY - MM/DD/YYYY HH24:MI:SS.US
説明	パラメータに関連する説明。

手順 2. マッピングパラメータの使用

作成したパラメータは、マッピングまたはマプレット内の任意のトランスフォーメーションの式エディタで使用できます。また、ソース修飾子トランスフォーメーションや再利用可能なトランスフォーメーションで使用することもできます。

ソース修飾子トランスフォーメーションでは、マッピングパラメータは SQL Editor の［変数］タブに表示されます。ソース修飾子トランスフォーメーションでマッピングパラメータを使用する場合には、以下の規則およびガイドラインに従ってください。

- string パラメータは、ソースシステムに対して適切な文字列識別子で囲みます。

- 必要に応じて、ソースのフォーマットに一致するように datetime パラメータのフォーマットを変更します。

また、式エディタでもマッピングパラメータを使用することができます。式エディタでマッピングパラメータを使用する場合、文字列パラメータを文字列識別子で囲まないでください。Integration Service では、パラメータを他のポート識別子と同じように扱います。

再利用可能なトランスフォーメーションで、マッピングパラメータを使用します。

また、Workflow Manager のセッションプロパティのトランスフォーメーション上書きで、マッピングパラメータを使用することもできます。ソース修飾子トランスフォーメーションの、フィルタやユーザー定義結合などのプロパティを上書きすることができます。

手順 3. パラメータの定義

セッションを実行する前に、マッピングパラメータの値をパラメータファイルに定義します。パラメータファイルでパラメータを定義しない場合、Integration Service は別の場所からパラメータ値を取得します。Integration Service では、以下の順で値を探します。

1. パラメータファイルに定義されている値
2. セッション実行前の変数割り当て値
3. リポジトリに保存された初期値
4. データタイプのデフォルト値

マッピング変数

Designer で、マッピングまたはマップレット内にマッピング変数を作成することができます。 マッピング変数を作成すると、式エディタに表示されます。作成したマッピング変数は、マッピングまたはマプレットの任意の式で使用できます。また、ソース修飾子フィルタ、ユーザー定義結合、抽出上書きや、再利用可能なトランスフォーメーションの式エディタで使用することもできます。

マッピングパラメータと異なり、マッピング変数はセッションごとに変更されます。Integration Service により、マッピング変数の最終値は正常な各セッション終了時にリポジトリに保存されます。 次のセッションの実行中に、マッピング変数への参照はすべて保存された値に評価されます。保存された値は、パラメータファイルで上書きすることができます。また、Workflow Manager で、保存されたセッションの値をすべてクリアすることもできます。

マッピング変数を使用して、ソースの差分読み込みを実行することができます。例えば、タイムスタンプ付きの取り引きを含むソーステーブルがあり、その取り引きを毎日評価するとします。セッションを実行するごとにセッション上書きを手動で入力し、ソースをフィルタリングする代わりに、マッピング変数$ $IncludeDateTime を作成することができます。ソース修飾子で、取り引き日付が$$IncludeDateTime に等しい行のみを読み込むフィルタを作成することができます。例えば、次のようになります。

```
TIMESTAMP = $$IncludeDateTime
```

マッピングでは、変数関数を使用して、セッションの実行ごとに変数値が 1 日ずつインクリメントするように設定できます。$$IncludeDateTime の初期値を 8/1/2004 に設定した場合、Integration Service はセッションの最初の実行時に日付が 8/1/2004 である行のみを読み込みます。 セッション中に、Integration Service により$$IncludeDateTime が 8/2/2000 に設定されます。 PowerCenter Server は、セッションの最後に 8/2/2004 という日付をリポジトリに保存します。セッションを次に実行したとき、PowerCenter Server は日付が 8/2/2004 である行だけを読み込みます。

変数値

Integration Service によりセッションの実行中に、マッピング変数に対して 2 つの異なる値が保持されます。

- マッピング変数の開始値
- マッピング変数のカレント値

マッピング変数のカレント値は、セッションが進行すると変化します。マッピング変数のカレント値をマッピングや他のトランスフォーメーションで使用するには、SETVARIABLE 関数で次の式を作成します。

```
SETVARIABLE($$MAPVAR,NULL)
```

正常なセッションの最後に、Integration Service によってマッピング変数の最終的なカレント値がリポジトリに保存されます。

開始値

開始値とは、セッションの開始時の変数の値です。開始値は、パラメータファイルで変数に対して定義された値、セッション実行前の変数割り当てで割り当てられた値、前回のセッションの実行でリポジトリに保存された値、変数のユーザー定義初期値、または変数のデータタイプに基づいたデフォルト値です。Integration Service では、以下の順で開始値を探します。

1. パラメータファイルに定義されている値
2. セッション実行前の変数割り当て値
3. リポジトリに保存された値
4. 初期値
5. データタイプのデフォルト値

例えば、マッピングまたはマップレット内にマッピング変数を作成して、初期値を入力します。しかしその変数の値をパラメータファイルに定義しません。Integration Service はセッションの最初の実行時に、変数の開始値を設定された初期値に評価します。 次にセッションが実行されるときに、Integration Service では変数の開始値をリポジトリに保存された値に評価します。 セッションを実行する前にリポジトリに保存された値をオーバーライドする場合、その変数の値をパラメータファイルに定義する必要があります。パラメータファイルにマッピング変数の値が定義されている場合、Integration Service により、リポジトリに保存されている値や、変数に設定されている初期値ではなく、この値が使用されます。式内にマッピング変数（'$$MAPVAR'）を使用した場合、式により常にこのマッピング変数の開始値が返されます。MAPVAR の開始値が 0 の場合、$$MAPVAR は 0 を返します。

カレント値

カレント値とは、セッションが進行しているときの変数の値です。セッション開始時には、変数のカレント値は開始値と同じです。セッションが進行すると、Integration Service は変数に設定されている変数関数を使用してカレント値を計算します。 マッピング変数の開始値とは異なり、各行がマッピングをパススルーするときに Integration Service が変数のカレント値を評価すると、カレント値が変化する場合があります。 変数の最終カレント値は、正常なセッション終了時にリポジトリに保存されます。セッションの完了に失敗した場合、Integration Service はリポジトリ内の変数の値を更新しません。 Integration Service は、マッピング変数ごとにリポジトリに保存された値を、セッションログに示します。

変数のデータタイプと集計タイプ

マッピングでマッピング変数を宣言する場合、変数のデータタイプおよび集計タイプを設定する必要があります。

マッピング変数に対して選択したデータタイプによって、Integration Service でマッピング変数の適切なデフォルト値を選択することができます。 パラメータファイルやリポジトリに変数のために定義された値がなく、ユーザー定義の初期値がない場合、マッピング変数の開始値としてデフォルト値が使用されます。

Integration Service によりマッピング変数の集計タイプが使用され、マッピング変数の最終カレント値が決定されます。 パイプラインに複数のパーティションがある場合、Integration Service により各パーティションからの変数値が組み合わせられ、最終カレント変数値がリポジトリに保存されます。

以下の集計タイプで変数を作成できます。

- 合計数
- 最大
- 最小

マッピング変数が Integer または Small Integer の場合、Count 集計タイプに設定することができます。すべてのデータタイプのマッピング変数を、Max または Min 集計タイプに設定することができます。

セッション実行中における変数値の整合性を保つために、Designer は変数に使用できる変数関数を集計タイプに応じて制限しています。例えば、SetMaxVariable 関数は Max 集計タイプの変数に使用できますが、Min 集計タイプの変数には使用できません。

以下の表に、利用可能な変数関数、および各関数で使用できる集計タイプとデータタイプを示します。

変数関数	有効な集計タイプ	有効なデータタイプ
SetVariable	Max または Min	binary データ型以外のすべてのトランスフォーメーションデータ型。
SetMaxVariable	Max のみ	binary データ型以外のすべてのトランスフォーメーションデータ型。
SetMinVariable	Min のみ	binary データ型以外のすべてのトランスフォーメーションデータ型。
SetCountVariable	Count のみ	Integer および Small Integer データ型のみ。

複数のターゲットロード順グループで、ターゲットロード順グループのマッピング変数値は変数集計タイプおよび以前のターゲットロード順グループの変数値に依存します。すべてのターゲットロード順グループが実行されたあとに、Integration Service は変数集計タイプに基づいた次のターゲットロード順グループで使用するためにマッピング変数値を計算します。

例えば、セッションには 2 つのターゲットロード順グループが含まれているとします。マッピング変数の集計タイプを Max に設定しています。

最初のターゲットロード順グループが実行されるとき、マッピング変数$$MAPVAR に対し SetVariable 関数を使用して次の異なる値を設定しています:

1. SetVariable($$MAPVAR,20)
2. SetVariable($$MAPVAR,10)
3. SetVariable($$MAPVAR,60)
4. SetVariable($$MAPVAR,30)

最初のターゲットロード順グループ実行の終わりに、Integration Service は$$MAPVAR の 4 つの値の最大値を計算します。4 つの値の最大値が 60 の場合、Integration Service は次のターゲットロード順グループでマッピング変数$$MAPVAR の初期値として 60 を使用します。

変数関数

変数関数では、Integration Service の、パイプラインでのマッピング変数のカレント値の計算方法が決定されます。 変数関数を式で使用して、次のセッション実行時のマッピング変数の値を設定します。トランスフォーメーション言語は、マッピングで使用する以下の変数関数を提供します。

- **SetMaxVariable。**変数を値のグループの最大値に設定します。 更新、削除、または拒否としてマークされた行は無視されます。SetMaxVariable をマッピング変数で使用するには、マッピング変数の集計タイプを Max に設定する必要があります。
- **SetMinVariable。**変数を値のグループの最小値に設定します。 更新、削除、または拒否としてマークされた行は無視されます。SetMinVariable をマッピング変数で使用するには、マッピング変数の集計タイプを Min に設定する必要があります。
- **SetCountVariable。**変数値を 1 ずつ増やします。 つまり、行が「挿入」に設定されている場合は変数値に 1 を加え、「削除」に設定されている場合は 1 を引きます。「更新」または「拒否」に設定された行は無視します。SetCountVariable をマッピング変数で使用するには、マッピング変数の集計タイプを Count に設定する必要があります。
- **SetVariable。**変数を設定された値に設定します。 セッションの最後に、変数の最後のカレント値を変数の開始値と比較します。変数の集計タイプに基づいて、PowerCenter Server は最終カレント値をリポジトリに保存します。SetVariable 関数をマッピング変数で使用するには、マッピング変数の集計タイプを Max または Min に設定する必要があります。SetVariable 関数は、「削除」または「拒否」に設定された行を無視します。

パイプラインの各マッピング変数に対して 1 回だけ変数関数を使用します。Integration Service により、マッピングで変数関数が検出された場合、その変数関数が処理されます。Integration Service がマッピングで変数関数が検出される順序は、セッションを実行するたびに異なります。これにより、マッピングで同じ変数関数を複数回使用した場合、結果が矛盾することがあります。

以下の条件のいずれかに当てはまる場合、Integration Service によりマッピング変数の最終カレント値はリポジトリに保存されません。

- セッションの完了に失敗した。
- セッションがテストロードに設定されている。
- セッションがデバッグセッションである。
- セッションがデバッグモードで実行され、セッション出力を無視するように設定されている。

マップレットでのマッピング変数

マップレット用のマッピング変数を宣言し、そのマップレットを同じマッピング内で複数回使用する場合、同じマッピング変数値がすべてのマップレットインスタンスで共有されます。

マッピング変数の使用

マッピング変数を使用するには、以下の手順を実行します。

1. マッピング変数を作成する。
2. 変数を使用し、変数値を設定する。
3. 変数値を上書きまたはクリアする。

手順 1 マッピング変数の作成

任意のマッピングまたはマプレットのマッピング変数を作成することができます。マッピング変数は必要なだけ作成することができます。作成した変数を、マッピングまたはマプレットで使用します。

マッピング変数を作成するには：

1. Mapping Designer で、［マッピング］-［パラメータと変数］をクリックします。あるいは、Mapplet Designer で［マップレット］-［パラメータと変数］の順にクリックします。
2. ［追加］ボタンをクリックします。
3. 変数情報を指定します。

   以下の表に、［パラメータおよび変数］ダイアログボックスのオプションを示します。

フィールド	必須/ オプション	説明
名前	必須	変数名。変数には、「$$*VariableName*」のように名前を付けます。変数名の構文は、$$に続けて任意の英数字かアンダスコア文字としなければなりません。
タイプ	必須	変数またはパラメータ。変数を選択します。
データタイプ	必須	変数のデータタイプ。有効なトランスフォーメーションデータタイプを選択してください。任意のデータタイプ（[Binary] を除く）を使用します。選択したデータタイプが、選択できる集計タイプに影響を与える場合があります。例えば、文字列変数を作成した場合、その変数に Count 集計タイプを設定することはできません。
精度または位取り	必須	変数の精度および位取り。
集計	必須	変数の集計タイプ。変数で実行できる計算の種類を決定します。 - マッピング変数を使用して、ソースから読み込む行数を数えたい場合は、［集計］を［Count］に設定します。 - マッピング変数を使用して、値のグループの最大値を決める場合は、［集計］を［Max］に設定します。 - マッピング変数を使用して、値のグループの最小値を決める場合は、［集計］を［Min］に設定します。
IsExprVar	必須	Integration Service が式文字列で変数を展開する方法を決定します。true の場合は、Integration Service は式の解析前に変数を展開します。false の場合は、Integration Service は式の解析後に変数を展開します。デフォルトは False です。 **注**: このフィールドを True に設定する場合、変数データタイプを String に設定する必要があります。String に設定しないと、Integration Service のセッションは失敗します。

フィールド	必須/ オプション	説明
初期値	オプション	変数の初期値。変数値がリポジトリに保存されていない場合、またはパラメータファイルに定義されていない場合、Integration Service はこの値を変数に使用します。この値も定義されていない場合、Integration Service はマッピング変数のデータタイプに基づいたデフォルト値を使用します。 日付/時刻パラメータの初期値に以下のフォーマットのいずれかを使用します。 - MM/DD/RR - MM/DD/RR HH24:MI:SS - MM/DD/YYYY - MM/DD/YYYY HH24:MI:SS.US
説明	オプション	変数に関する説明。

4. ［OK］をクリックします。

手順 2 マッピング変数値の設定

作成したマッピング変数は、マッピングまたはマプレット内の任意の式で使用します。マッピング変数は、ソース修飾子トランスフォーメーションまたは再利用可能なトランスフォーメーションで使用することもできます。

ソース修飾子トランスフォーメーションでは、マッピング変数は SQL Editor の［変数］タブに表示されます。ソース修飾子トランスフォーメーションでマッピング変数を使用する場合、以下の規則に従ってください。

- 一重引用符などの文字列識別子で文字列変数を囲み、文字列内の変数を示します。
- 必要に応じて、ソースのフォーマットに一致するように日付/時刻変数のフォーマットを変更します。Integration Service により、PowerCenter のデフォルト日付フォーマットからソースシステムのデフォルト日付フォーマットに日付が変換されます。

マッピングまたはマッピング内のその他のトランスフォーメーションの場合、マッピング変数は式エディタに表示されます。マッピング変数を使用する式を記述する場合、文字列変数に文字列識別子は必要ありません。

再利用可能なトランスフォーメーションで、マッピング変数を使用します。式を検証する場合、Designer は変数を Integer データタイプとして扱います。

また、セッションプロパティのトランスフォーメーション上書きでマッピング変数を使用することもできます。ソース修飾子トランスフォーメーションの、フィルタやユーザー定義結合などのプロパティを上書きすることができます。

マッピング変数を使用する場合、マッピング変数の値をどのように設定するかを決める必要があります。変数関数を使用して、変数値を設定します。以下のいずれかのトランスフォーメーションで、変数関数を使用します。

- 式
- フィルタ
- ルータ
- アップデートストラテジ

手順 3. 保存された値の上書きまたはクリア

セッションが正常に完了した場合、Integration Service により各変数の最終値がリポジトリに保存されます。次回のセッション実行時にその値を使用しない場合は、パラメータファイル、またはセッションプロパティのセッション実行前の変数割り当てで値を上書きできます。

保存されている変数値をセッションに使用したくない場合は、保存されている値をすべてクリアすることができます。Workflow Manager を使用して、セッションの変数値をクリアすることができます。リポジトリから変数値を取り消した後、Integration Service は初めてセッションを実行するときと同じように動作します。

パラメータファイル内での式の文字列の定義

Integration Service は、セッションを実行すると、マッピングパラメータと変数を展開します。式でマッピングパラメータまたはマッピング変数を使用すると、Integration Service は式の解析後、パラメータまたは変数を展開します。式を作成して頻繁に変更されるビジネスルールを表す場合は、Integration Service で式の解析前にパラメータまたは変数を展開する必要が生じる場合があります。この場合は、パラメータファイルで式を定義し、ビジネスルールが変更されるたびにマッピングを変更する必要がないようにします。

例えば、次のように ID 文字列に基づいて色の名前を生成する式を作成するとします。

```
IIF(color='A0587','white')
```

翌月、次のようにこの式を変更します。

```
IIF(color='A0587','white',IIF(color='A0588','off white'))
```

ビジネスルールが変更されるたびにこの式を使用するマッピングを更新する代わりに、パラメータファイルに式を定義し、式が変更されるとファイルを更新することができます。

パラメータファイルの式を定義するには、以下のようにマッピングおよびワークフローを設定します。

1. 色の名前の式を格納するマッピングパラメータまたは変数を作成します。例えば、マッピングパラメータ $$ExpColor を作成します。
2. マッピングパラメータ$$ExpColor の IsExprVar プロパティを true に設定します。パラメータのデータタイプを String に設定する必要があります。String に設定しないと、Integration Service のセッションは失敗します。
3. 式トランスフォーメーションで次の式の出力ポートを設定します。

   ```
   $$ExpColor
   ```
4. パラメータファイルを使用するようにセッションまたはワークフローを設定します。
5. パラメータファイルで、$$ExpColor を正しい式に設定します。以下に例を示します。

   ```
   $$ExpColor=IIF(color='A0587','white')
   ```

マッピングパラメータ$$ExpColor の IsExprVar が true に設定されているので、Integration Service は式の解析前にパラメータを展開します。色 ID「A0587」の行が文字列「white」を返します。IsExprVar が false に設定されている場合、Integration Service は式の解析後にパラメータを展開します。そのため、すべての行は文字列「IIF(color='A0587','white')」を返します。

色名の式が変更されると、パラメータファイルのマッピングパラメータの値を更新できます。マッピングを変更する必要はありません。

マッピングパラメータおよび変数の使用に関するヒント

マッピングパラメータおよび変数の初期値を入力してください。

マッピングパラメータまたは変数の論理デフォルト値がわかっている場合は、パラメータまたは変数の作成時にそれを初期値として使用します。これにより、Integration Service ではデータタイプベースのデフォルト値ではなく、このデフォルト値を使用できます。

SQL エディタ、文字列および日付/時刻パラメータおよび変数を引用符で囲んでください。

文字列パラメータまたは変数をソース修飾子トランスフォーメーションで使用する場合は、一重引用符などソースシステムが認識できる文字列識別子によって変数を囲みます。

すべてのパラメータファイルを、プロセス変数ディレクトリの 1 つに保存します。

すべてのパラメータファイルをプロセス変数ディレクトリの 1 つ（$SourceFileDir など）に保存する場合、そのプロセス変数をセッションプロパティシートで使用します。ソースファイルおよびパラメータファイルを後日移動させる必要がある場合、プロセス変数を新しいディレクトリを参照するように変更すれば、すべてのセッションを更新できます。

再利用可能なパラメータのセットについて、個々にパラメータファイルを作成してください。

セッションに対してパラメータのセットを切り換えて利用する場合は、パラメータのセットそれぞれにパラメータファイルを作成します。*pmcmd* を使用してセッションを開始し、使用したいパラメータファイルの名前を指定するか、セッションを編集して異なるパラメータファイルを使用します。

マッピングパラメータおよび変数のトラブルシューティング

マッピングに対するマッピング変数を作成しましたが、各セッションの実行における変数の値が同じままです。

変数関数をマッピングに含めていないことが考えられます。セッションの実行ごとに変数の値を変更するには、変数関数を使用します。

または、パラメータファイルに変数の値が設定されている可能性があります。パラメータファイルにマッピング変数の値を定義した場合、Integration Service によりその値が変数に使用されます。 セッションログを参照し、Integration Service によりどの開始値が各変数に使用されているか確認します。

パラメータファイルで、マプレットのパラメータ値を設定しましたが、セッションで使用されません。

マプレットのマッピングパラメータおよび変数をパラメータファイルに定義する場合は、次に示すように、マプレット名の後に続ける必要があります。

```
mappletname.parameter=value
mappletname.variable=value
```

ランクまたはアグリゲータトランスフォーメーションの式エディタで変数関数を見つけられません。

ランクまたはアグリゲータトランスフォーメーションで変数関数を使用することはできません。変数関数には、別のトランスフォーメーションを使用してください。

第 8 章

ユーザー定義関数に関する作業

この章では、以下の項目について説明します。

ユーザー定義関数に関する作業の概要

ユーザー定義関数により、PowerCenter トランスフォーメーション言語が拡張されます。 Designer で PowerCenter トランスフォーメーション言語を使用して、ユーザー定義関数を作成、管理できます。 これを Designer や Workflow Manger の式に追加して、式論理を再利用したり、複雑な式を構築したりできます。ユーザー定義関数は、リポジトリ内の他のユーザーも使用できます。

例

姓の前後のスペースを削除する必要があります。RemoveSpaces という名前が付けられたユーザー定義関数を作成し、LTRIM 関数と RTRIM 関数を実行できます。 このユーザー定義関数を設定する場合は、以下の式を入力します。

```
LTRIM( RTRIM( name))
```

関数の作成後、式トランスフォーメーションで次の式を作成して、姓から先頭と末尾のスペースを削除できます。

```
:UDF.REMOVESPACES(LAST_NAME)
```

ユーザー定義関数名には、UDF が先頭についています。ポート名は LAST_NAME です。これは式の引数です。

ユーザー定義関数の作成

ユーザー定義関数は、Designer で作成します。

以下の表に、ユーザー定義関数を作成する際に設定するプロパティを示します。

プロパティ	説明
Name	関数の名前。先頭は文字にする必要があります。文字、数字、および次の特殊文字を使用できます。 _ @ $ # 名前 80 文字を超えない範囲で指定する必要があります。また、空白文字は使用できません。
タイプ	関数が任意の式から呼び出せる場合はパブリック。関数が別のユーザ定義関数からしか呼び出せない場合はプライベート。
説明	関数の説明。
戻り型	関数が返す値のデータ型。有効な関数が作成されると、Designer によってデータ型が決定されます。
引数	関数に使用する引数。引数名、トランスフォーメーションデータ型、精度、位取りを指定して、関数の入力値のデータ型、精度、位取りを決めます。
Expression	関数を定義する式。式エディタで構文を設定します。関数に設定した引数を使用します。トランスフォーメーション言語の関数、カスタム関数、または他のユーザー定義関数も使用できます。PowerCenter トランスフォーメーション言語のルールおよびガイドラインに従います。 **注**: 式ウィンドウに構文を入力した場合、［ツール］メニューでその関数を検証します。

関数タイプの設定

ユーザー定義関数を他のユーザー定義関数内に配置することができます。 また、式から呼び出せるようにユーザー定義関数を設定することもできます。 呼び出しが可能ということは、式の中にユーザー定義関数を配置できるということです。

ユーザー定義関数を設定する場合、以下のいずれかのオプションを選択します。

- **パブリック。**すべてのユーザー定義関数、トランスフォーメーション式、リンク条件式、またはタスク式から呼び出すことができます。
- **プライベート。**別のユーザー定義関数から呼び出すことができます。 より複雑な関数の一部にする場合に、プライベート関数を作成します。単純な関数は、複雑な関数と組み合わせなければ使用できない場合もあります。

パブリックのユーザー定義関数を作成した後に、その関数タイプをプライベートに変更することはできません。

ユーザー定義関数を別のユーザー定義関数内に配置することはできますが、関数はそれ自体を参照することはできません。 例えば、ユーザー定義関数 RemoveSpaces にユーザー定義関数 TrimLeadingandTrailingSpaces が含まれるとします。 TrimLeadingandTrailingSpaces に RemoveSpaces を含めることはできません。RemoveSpaces を含めた場合、RemoveSpaces は無効になります。

プライベート関数を含むパブリック関数の設定

プライベートのユーザー定義関数にポートを引数として含める場合、そのプライベート関数を含むパブリック関数にもポートを引数として含める必要があります。 この場合、プライベート関数およびパブリック関数の引数では、同じデータタイプおよび精度を使用します。

例えば、オーダー ID に「INFA」と顧客 ID を含めるための関数を定義するとします。この場合、まず ConcatCust と呼ばれる以下のプライベート関数を作成します。この関数は、「INFA」と CUST_ID ポートを連結します。

```
CONCAT (‘INFA’, CUST_ID)
```

上記のプライベート関数を作成してから、ConcatOrder と呼ばれるパブリック関数を作成します。この関数には、ConcatCust 関数が含まれています。

```
CONCAT (:UDF.CONCATCUST( CUST_ID), ORDER_ID)
```

ConcatCust 関数を ConcatOrder 関数に追加すると、そのパブリック関数と同じデータタイプおよび精度の CUST_ID 引数を追加することになります。

注: パブリック関数の構文を手動で定義する際にユーザー定義関数を入力する場合、そのユーザー定義関数に「:UDF」をプレフィックスとして追加する必要があります。

ユーザー定義関数の作成手順

以下の手順を使用して、ユーザー定義関数を作成します。

1. ［ツール］-［ユーザー定義関数］-［新規］をクリックします。

   ［ユーザー定義関数の編集］ダイアログボックスが表示されます。
2. 関数名を入力します。
3. 関数タイプを選択します。
4. 必要に応じて、そのユーザー定義関数の説明を入力します。

   最大で 2,000 文字まで入力できます。
5. そのユーザー定義関数の引数を作成します。

   引数を作成する場合、引数の名前、トランスフォーメーションデータタイプ、精度、および位取りを設定します。
6. ［エディタの起動］をクリックし、手順 5 で定義した引数を含む式を作成します。
7. ［OK］をクリックします。

   Designer が、式が返すデータのデータタイプを割り当てます。
8. ［OK］をクリックします。

ユーザー定義関数の管理

ユーザー定義関数を管理するための以下のタスクを実行することができます。

- ユーザー定義関数の編集。
- ユーザー定義関数の検証。
- ユーザー定義関数のプロパティの表示。
- ユーザー定義関数の依存関係の表示。
- ユーザー定義関数のエクスポート。
- ユーザー定義関数のバージョニングの管理。

以下の表に、ユーザー定義関数管理タスクを示し、各タスクを実行できる場所を一覧表示します。

タスク	タスクを実行する領域
ユーザー定義関数の編集	- ［ツール］-［ユーザー定義関数］をクリックします。 - ユーザー定義関数を右クリックします。
ユーザー定義関数の検証	- ［ツール］-［ユーザー定義関数］をクリックします。 - ナビゲータ、［クエリー結果］ウィンドウ、または［履歴の表示］ウィンドウでユーザー定義関数を右クリックします。
ユーザー定義関数の削除	- ［ツール］-［ユーザー定義関数］をクリックします。
ユーザー定義関数のプロパティの表示	- ［ツール］-［ユーザー定義関数］をクリックします。 - ユーザー定義関数を右クリックします。
ユーザー定義関数の依存関係の表示	- ［ツール］-［ユーザー定義関数］をクリックします。 - ユーザー定義関数を右クリックします。
ユーザー定義関数のエクスポート	- ［ツール］-［ユーザー定義関数］をクリックします。 - ユーザー定義関数を右クリックします。
ユーザー定義関数のバージョニングの管理	- ［ツール］-［ユーザー定義関数］をクリックします。 - ユーザー定義関数を右クリックします。

ユーザー定義関数の編集

ユーザー定義関数を編集し、関数プロパティを変更することができます。 加えた変更は、その関数を使用するすべてのユーザー定義関数と式にプロパゲートされます。

ユーザー定義関数を編集する際には、以下のルールおよびガイドラインを使用します。

- ユーザー定義関数名を変更する場合、Designer は名前の変更をオブジェクト内の式にはプロパゲートしません。ユーザー定義関数がある式を使用している場合は、マッピングやワークフローが無効になる場合があります。
- ユーザー定義関数の式を変更すると、その関数が式で使用された場合に返す値のデータタイプを、Designer が変更する可能性があります。
- ユーザー定義関数を、パブリックからプライベートに変更することはできません。
- ユーザー定義関数が無効な場合、ユーザー定義関数を使用するマッピングとワークフローも無効になる可能性があります。マッピングまたはワークフローを検証してください。

ユーザー定義関数の削除

ユーザー定義関数を削除すると、リポジトリサービスは、その関数を使用している他のユーザー定義関数および式からこの関数を削除します。 その結果、これらのユーザー定義関数および式が無効になる場合があります。 ユーザー定義関数を使用するマッピングおよびワークフロー、またはユーザー定義関数を使用する式も無効になる可能性があります。マッピングまたはワークフローを検証してください。

ユーザー定義関数のエクスポート

ユーザー定義関数を XML にエクスポートすることができます。 その後、それを他のリポジトリまたはリポジトリフォルダにインポートできます。

ユーザー定義関数を含んだマッピングまたはワークフローをエクスポートした場合、PowerCenter クライアントによってそのユーザー定義関数がエクスポートされます。

ユーザー定義関数の検証

ユーザー定義関数は以下の領域から検証できます。

- 式エディタ（ユーザー定義関数を作成または編集する場合）
- ［ツール］メニュー
- ［クエリー結果］ウィンドウ
- ［履歴の表示］ウィンドウ

ユーザー定義関数を検証する場合、PowerCenter クライアントではその関数を使用する他のユーザー定義関数および式は検証されません。 ユーザー定義関数が無効な場合、その関数を使用するユーザー定義関数と式も無効になります。 同様に、そのユーザー定義関数を使用するマッピングとワークフローも無効になります。

ユーザー定義関数のコピーとデプロイ

ユーザー定義関数が含まれるオブジェクトをコピーする場合、コピーウィザードによってそのユーザー定義関数もコピーされます。

ユーザー定義関数を含む静的デプロイメントグループをデプロイする場合、コピーウィザードによってそれらの関数もコピーされます。 ユーザー定義関数を含む動的デプロイメントグループをデプロイする場合、式に含まれていない関数はコピーウィザードによってデプロイされません。

ユーザー定義関数を含む式の作成

式には、ユーザー定義関数を追加できます。 式を手動で作成する際にユーザー定義関数を入力する場合は、そのユーザー定義関数に「:UDF」をプレフィックスとして追加する必要があります。 式エディタで式を作成する場合、有効なユーザー定義関数は［ユーザー定義関数］タブに表示されます。 他の関数と同じようにユーザー定義関数を使用します。

Designer でのみ使用可能なユーザー定義関数を作成した場合、その関数は Designer 内にのみ表示されます。

ユーザー定義関数を選択すると、式エディタには関数構文が以下のフォーマットで表示されます。

```
<return datatype> <function name> (<argument 1> as <datatype>,
<argument N> as <datatype>)
```

以下に例を示します。

```
NSTRING RemoveSpaces(NAMES as string)
```

［式］ウィンドウに追加された関数には、以下のように接頭語として「:UDF」が挿入されます。

```
:UDF.RemoveSpaces( )
```

式を検証する際に、ユーザー定義関数は PowerCenter によって検証されません。 式のみが検証されます。

第 9 章

デバッガの使用

この章では、以下の項目について説明します。

デバッガの使用の概要

有効なマッピングをデバッグして、データおよびエラー条件に関するトラブルシューティング情報を取得することができます。マッピングをデバッグするには、Mapping Designer 内からデバッガを設定および実行する必要があります。デバッガでは、セッションを使用してマッピングを Integration Service 上で実行します。デバッガは実行時にブレークポイントで一時停止するので、トランスフォーメーション出力データを表示して編集することができます。

以下の状況で、デバッガを実行する場合があります。

- **セッションの実行前。** マッピングを保存した後、Workflow Manager でセッションを作成および設定する前に、デバッグセッションの初期テストを実行できます。
- **セッションの実行後。**セッションが失敗した場合、またはターゲットで予想外の結果が出た場合、セッションに対してデバッガを実行できます。 また、設定されたセッションプロパティを使用してマッピングをデバッグする場合も、セッションに対してデバッガを実行できます。

デバッガセッションのタイプ

デバッガを設定する時に、3 つのタイプのデバッガセッションを選択できます。デバッガは、各セッションタイプのワークフローを実行します。デバッガ設定時に、以下のタイプのデバッガセッションから選択できます。

- **既存の再利用不可能なセッションの使用。**デバッガにより、既存のソース、ターゲット、およびセッションの設定プロパティが使用されます。 デバッガ実行時に、Integration Service は再利用不可能なセッションおよび既存ワークフローを実行します。 エラーが発生しても、デバッガはサスペンド状態になりません。

- **既存の再利用可能なセッションの使用。**デバッガにより、既存のソース、ターゲット、およびセッションの設定プロパティが使用されます。 デバッガ実行時に、Integration Service は再利用可能セッションのデバッグインスタンスを実行し、そのセッションのデバッグワークフローを作成および実行します。
- **デバッグセッションインスタンスの作成。**ソース、ターゲット、およびセッションの設定プロパティをデバッガウィザードで設定できます。 デバッガ実行時に、Integration Service はデバッグワークフローのデバッグインスタンスを実行し、そのセッションのデバッグワークフローを作成および実行します。

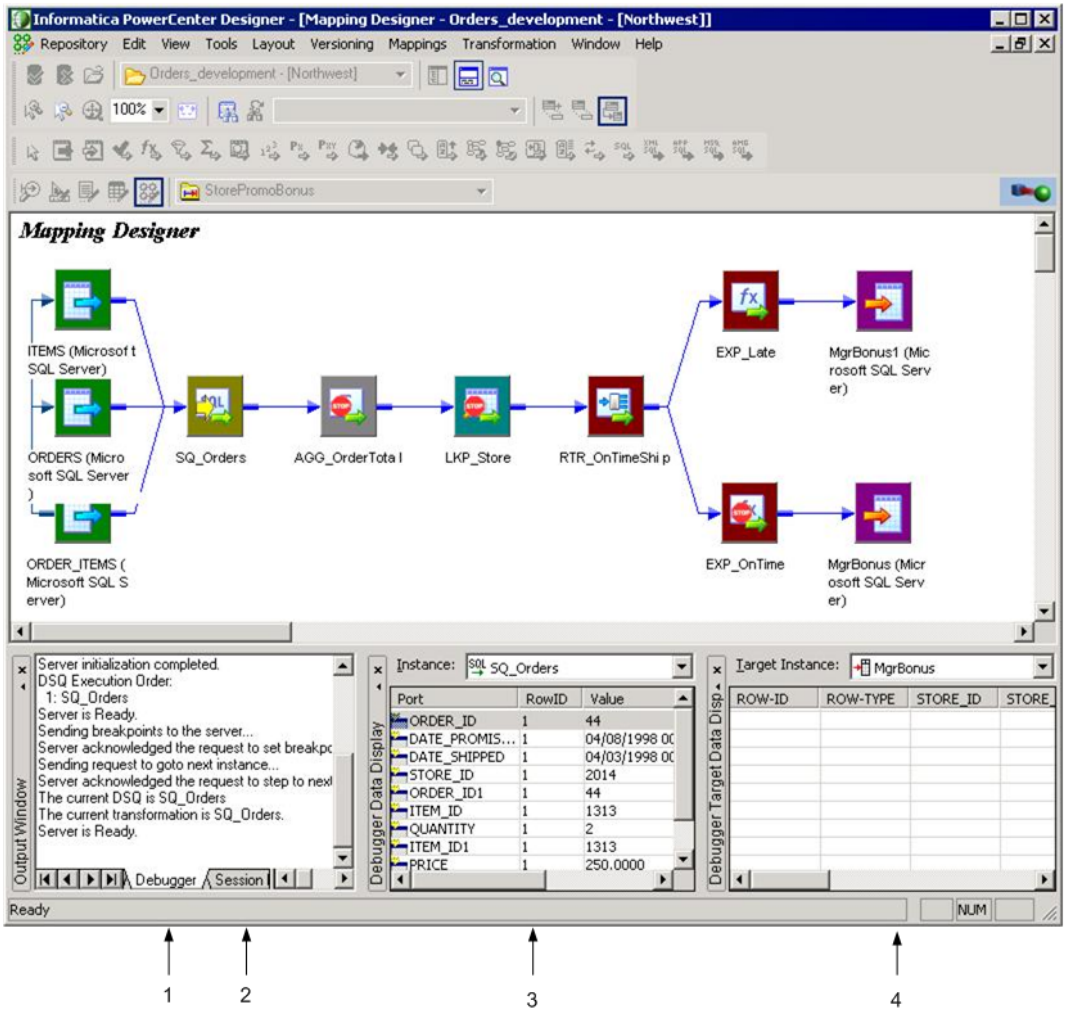
デバッグプロセス

マッピングをデバッグするには、以下の手順を実行します。

1. **ブレークポイントの作成。**Integration Service によってデータおよびエラー状態が評価されるブレークポイントを、マッピング上に作成します。
2. **デバッガの設定。** デバッガウィザードを使用して、マッピングで使用するデバッガを設定します。 Integration Service でデバッガが実行されるときに使用されるセッションタイプを選択します。 デバッグセッションを作成するときには、デバッガウィザード内にソースやターゲットの場所など、セッションプロパティのサブセットを設定します。ターゲットデータのロードまたは破棄も選択できます。
3. **デバッガの実行。**Mapping Designer 内からデバッガを実行します。 デバッガを実行すると、Designer が Integration Service に接続されます。 Integration Service によってデバッガが初期化され、デバッグセッションとワークフローが実行されます。 Integration Service でブレークポイントが読み込まれ、ブレークポイントが True と評価されるとデバッガが一時停止されます。
4. **デバッガの監視。**デバッガ実行中は、ターゲットデータ、トランスフォーメーションおよびマップレットの出力データ、デバッグログ、およびセッションログを監視できます。 デバッガを実行すると、Designer は以下のウィンドウを表示します。
   - **デバッグログ。**デバッガからのメッセージを表示します。
   - **ターゲットウィンドウ。** ターゲットデータを表示します。
   - **インスタンスウィンドウ。** トランスフォーメーションデータを表示します。
5. **データおよびブレークポイントの変更。**デバッガが一時停止した場合、データを変更して、データがパイプラインを移動する際のトランスフォーメーション、マップレット、およびターゲットへの影響を確認できます。 ブレークポイントの情報を変更することもできます。

Designer は、マッピングのブレークポイントおよびデバッガの情報をワークスペースファイルに保存します。ブレークポイント情報とデバッガ設定は、他のマッピングにコピーできます。他の PowerCenter クライアントマシンからデバッガを実行する場合、その PowerCenter クライアントマシンにブレークポイント情報とデバッガ設定をコピーできます。

以下の図に、デバッガ実行時に表示される Mapping Designer のウィンドウを示します。



1. デバッガログ
2. セッションログ
3. インスタンスウィンドウ。
4. ターゲットウィンドウ。

ブレークポイントの作成

デバッガを実行する前に、Mapping Designer の Breakpoint Editor を使用してマッピング内にブレークポイント条件を作成できます。トランスフォーメーションまたはグローバル条件のデータまたはエラーのブレークポイントを作成できます。デバッガを実行すると、ブレークポイントの評価が True の場合に Integration Service はデバッガを一時停止します。 トランスフォーメーションデータを確認および変更して、セッションを続行できます。

注: マプレットの入力トランスフォーメーションおよび出力トランスフォーメーションには、ブレークポイントを作成できません。

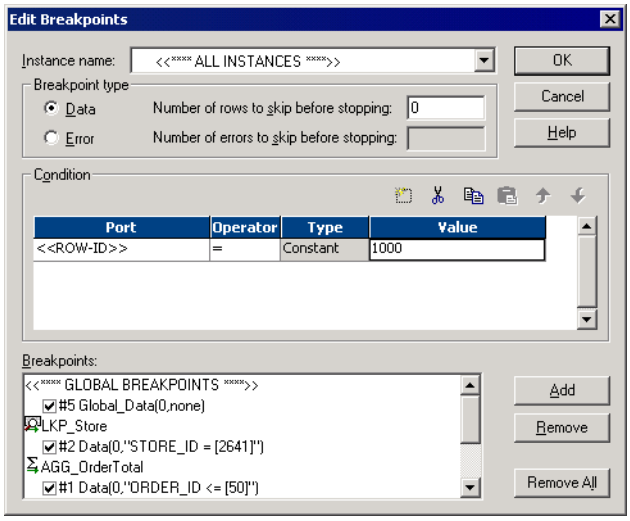
ブレークポイントには、インスタンス名、ブレークポイントタイプ、および条件を格納できます。ブレークポイントを入力したら、以下の順序でブレークポイントパラメータを設定します。

1. **インスタンス名の選択。** グローバルブレークポイントを作成するのか、あるいはマッピングまたはマップレット内の 1 つのトランスフォーメーションに対してブレークポイントを作成するのかを選択します。
2. **ブレークポイントのタイプを選択します。**エラーとデータのどちらの条件を評価するかを選択します。 ブレークポイントタイプでは、スキップする行数も選択できます。
3. **条件の入力。**グローバルまたはトランスフォーメーションデータブレークポイントのデータ条件を入力します。 エラーブレークポイントの条件は入力できません。

インスタンス名、ブレークポイントタイプ、およびオプションのデータ条件を設定した後で、Breakpoint Editor の［ブレークポイント］セクションで各パラメータを表示できます。

- **インスタンス名。**インスタンス名パラメータから取得されます。
- **ブレークポイント有効フラグ。**［ブレークポイントの表示］セクションで有効にします。
- **ブレークポイント番号。**Designer によって、マッピング内で作成されたブレークポイントの数に基づいて番号が割り当てられます。 連続した番号です。ブレークポイントを削除した場合、Breakpoint Editor を閉じて再度開いた時に Designer が番号を割り当てなおします。
- **ブレークポイントタイプ。**［ブレークポイントタイプ］セクションから取得されます。
- **スキップする行数。**［ブレークポイントタイプ］セクションから取得されます。
- **データ条件。**［条件］セクションから取得されます。

以下の図に、［ブレークポイント］セクションのパラメータを示します。



ルックアップトランスフォーメーションでは、LKP_Store のチェックマークは、ブレークポイントが有効であることを示します。 「#2」は連続ブレークポイント番号、「Data」はブレークポイントタイプ、「0」はスキップするブレークポイント行数、「STORE_ID=[2641]」はデータ条件を示します。

インスタンス名の選択

インスタンス名を選択する場合、個別のトランスフォーメーションのブレークポイントまたはグローバルブレークポイントを作成できます。

- **トランスフォーメーションインスタンス。**マッピングまたはマップレットからトランスフォーメーションを選択し、Integration Service がトランスフォーメーションを処理する場合に評価するブレークポイント条件を設定します。 マップレット内のトランスフォーメーションの名前は、*MappletName.TransformationName* に従い決定します。
- **グローバルインスタンス。**Integration Service がマッピング内の各トランスフォーメーションを処理する場合に評価するブレークポイント条件を設定するには、［グローバル］を選択します。

エラーブレークポイントの作成

エラーブレークポイントを作成した場合、Integration Service によってトランスフォーメーションエラーや ERROR 関数へのコールなどのエラー状態が検出されると、デバッガは一時停止します。

デバッガが一時停止する前に各ブレークポイントでスキップするエラーの数も設定できます。

- エラーが発生するたびにデバッガを一時停止する場合、スキップするエラーの数をゼロに設定します。
- 指定した数のエラーをスキップした後でデバッガを一時停止する場合、スキップするエラーの数をゼロより大きい数に設定します。例えばエラーの数を 5 に設定した場合、デバッガは 5 つのエラーをスキップし、6 つ目のエラーが発生した時点で一時停止します。

エラーブレークポイントの評価

Integration Service は、以下のルールを使用してエラーブレークポイントを評価します。

- Integration Service によって NULL の入力値が検出された場合、およびポートに ERROR（）のユーザー定義のデフォルト値が含まれている場合。
- Integration Service によって出力トランスフォーメーションエラーが検出された場合、およびポートに ERROR（）のシステムデフォルト値が含まれている場合。 以下のエラーは、トランスフォーメーションエラーとみなされます。

  - 日付に文字列を渡すなどのデータ変換エラー。

  - ゼロ除算などの式評価エラー。

  - 式内における ERROR 関数へのコール。
- Integration Service によって NULL の入力値またはトランスフォーメーション出力エラーが検出された場合、およびポートに ABORT（）のユーザー定義のデフォルト値が含まれている場合。
- Integration Service によって致命的なエラーが検出された場合。
- Integration Service によってトランスフォーメーション処理中にエラーが検出された場合、そのトランスフォーメーションのエラーブレークポイントおよびグローバルエラーブレークポイントが評価されます。 エラーブレークポイントが TRUE に評価された場合、デバッガは一時停止し、データブレークポイントを評価しません。

注: Integration Service によって致命的なエラーまたは ABORT が検出された場合、デバッガはスキップするエラーの設定数に達していない場合でもブレークします。

データブレークポイントの作成

データブレークポイントを作成した場合、データブレークポイントが TRUE に評価されるとデバッガが一時停止します。スキップする行数またはデータ条件、あるいは両方を設定できます。データブレークポイントの設定では、以下のオプションがあります。

- デバッガを各行で一時停止する場合、行数をゼロに設定し、データ条件は設定しません。
- トランスフォーメーションから渡された行を指定した数だけスキップした後でデバッガを一時停止する場合、行数をゼロより大きい数字に設定します。例えば行数を 3 に設定すると、デバッガはデータを 3 行スキップし、4 行目で毎回一時停止します。
- データ条件に一致するデータで毎回デバッガを一時停止する場合、データ条件を入力して行数をゼロに設定します。
- 行がデータ条件に一致する場合に指定された間隔でデバッガを一時停止する場合、データ条件を入力し、行数をゼロより大きい数字に設定します。例えば行数を 3 に設定すると、デバッガは条件に一致するブレークポイントの行を 3 行スキップし、4 行目で毎回一時停止します。

以下の表に、データブレークポイントのタイプおよび条件を設定する場合に選択できるオプションを要約して示します。

行数	データ条件	デバッガの動作
0	いいえ	各行で必ず一時停止。
>0	いいえ	トランスフォーメーションから指定した行数が *n* 回渡されるたびに一時停止。
0	はい	行がデータ条件に一致する場合、毎回一時停止。
>0	はい	行がデータ条件に一致する場合、*n* 回毎に一時停止。

データブレークポイント条件の入力

データ条件を作成する場合、グローバルパラメータまたは 1 つのトランスフォーメーションのパラメータを入力します。選択したインスタンス、および作成した条件タイプに応じたオプションを使用できます。

ポートまたは定数の値に対してトランスフォーメーションを評価する場合、以下の構文を使用します。

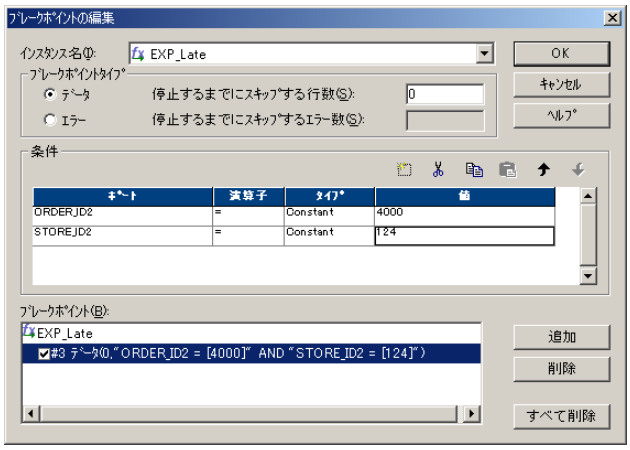
```
<port> <operator> <type> <value>
```

グローバルまたは 1 つのトランスフォーメーションで NULL およびデフォルトの値をチェックする場合、以下の構文を使用します。

```
<port> <operator>
```

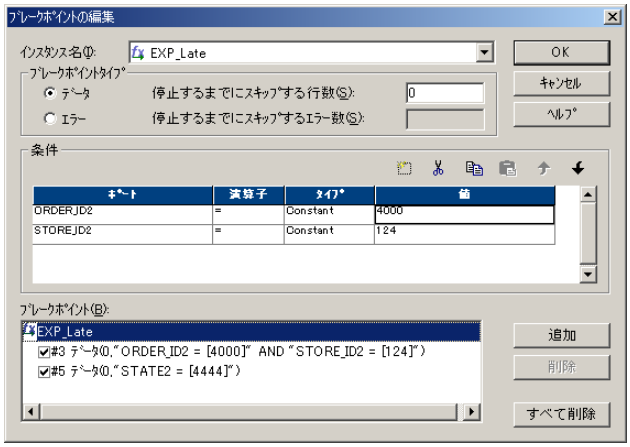
1 つのトランスフォーメーションで、1 つのブレークポイントに複数の条件を入力し、すべての条件に一致する場合にデバッガを一時停止することができます。1 つのトランスフォーメーションに複数のブレークポイントを入力し、少なくとも 1 つの条件に一致する場合にデバッガを一時停止することもできます。

以下の図に、1 つのブレークポイント内の複数の条件を示します。



この例では、両方の条件が True の場合、デバッガは一時停止します。

以下の図に、1 つのトランスフォーメーション内の複数のブレークポイントを示します。



この例では、いずれかの条件が True の場合、デバッガは一時停止します。

以下の表に、トランスフォーメーションおよびグローバルデータのブレークポイントの条件パラメータを示します。

パラメータ	トランスフォーメーションオプション	グローバルオプション
ポート	ポート名、<<ROW-ID>>、<<ROW-TYPE>>、または<<ANY-PORT>>を選択します。	<ANY-PORT>。各トランスフォーメーションおよびマプレットに対し、この条件を評価できます。
演算子	<, <=, =, >, >=, !=, ISNULL, ISDEFAULT.	ISNULL, ISDEFAULT.
タイプ	定数、ポート。定数の値、またはトランスフォーメーション内の他のポートに対して評価します。	利用できません。
値	値を入力するか、ポート名を選択します。	利用できません。

トランスフォーメーションデータ条件の入力

トランスフォーメーションデータの条件を入力する場合、トランスフォーメーション内の特定のポート、または以下のポートパラメータを選択できます。

- **行タイプ。**挿入、更新、削除、拒否、またはフィルタリング。
- **行 ID。**トランスフォーメーションから渡される行の番号。
- **任意のポート。**<<ANY-PORT>>を選択した場合、トランスフォーメーション内の各ポートで Integration Service が評価する条件を入力できます。 NULL またはエラーのデフォルト値をテストするには、ISNULL または ISDEFAULT を使用します。

注: ポート同士を比較する場合の最善の方法は、同じデータタイプのポートを使用することです。異なるデータタイプのポートを比較する場合、Integration Service は先行するポートのデータタイプを後続のポートのデータタイプに変換してからポートを比較します。ただし、これは、結果的に無効な比較となる場合もあります。

グローバルデータ条件の入力

グローバルデータ条件を入力する場合、すべてのトランスフォーメーション内の各ポートで Integration Service が NULL およびエラーのデフォルト値を評価する条件を入力します。 <ANY-PORT> <ISNULL>または <ANY-PORT> <ISDEFAULT>が選択できます。

ISNULL および ISDEFAULT の使用

トランスフォーメーションおよびグローバルデータブレークポイントでは、ISNULL および ISDEFAULT の条件を作成できます。ISNULL または ISDEFAULT の演算子を使用する場合、その条件内でタイプまたは値を使用できません。

ISNULL 条件を作成する際に、Integration Service が NULL の入力値が検出され、ポートにシステムデフォルト値が含まれている場合に、デバッガが一時停止します。

ISDEFAULT を作成すると、以下の状況でデバッガは一時停止します。

- Integration Service によって出力トランスフォーメーションエラーが検出され、定数値または定数式のユーザー定義のデフォルト値がポートに含まれる場合。
- Integration Service によって NULL の入力値が検出され、定数値または定数式のユーザー定義のデフォルト値がポートに含まれる場合。

ブレークポイントの入力手順

ブレークポイントを入力するには、次の手順を実行します。

1. マッピングで、以下のいずれかの方法で Breakpoint Editor を開きます。
   - Alt+F9 キーを押します。
   - ［マッピング］-［デバッガ］-［ブレークポイントの編集］をクリックします。
   - トランスフォーメーションまたはマプレットを右クリックし、［ブレークポイントの編集］を選択します。

   注: アイコン化されたトランスフォーメーションまたはマップレットのどの部分でも右クリックできます。 トランスフォーメーションまたはマップレットが通常ビューで表示されている場合、タイトルバーを右クリックします。
2. ［インスタンス名］セクションでインスタンス名を選択します。
3. ［追加］をクリックして新しいブレークポイントを追加します。

<<ALL INSTANCES>>を選択して［追加］をクリックすると、特定のトランスフォーメーションインスタンスまたはグローバルインスタンスを選択するように求められます。

無条件のデータブレークポイントを作成するには、［OK］をクリックします。

4. ブレークポイントのタイプを選択します。
5. データブレークポイントタイプを選択すると、以下の条件を入力できます。Integration Service で複数の条件を評価する場合、複数の条件を入力します。
6. それぞれのブレークポイントで、2 から 5 の手順を繰り返します。
7. ［OK］をクリックします。

ブレークポイントの編集

デバッガセッションの実行前または実行中に、ブレークポイントを確認する必要がある場合もあります。ブレークポイントは、Breakpoint Editor で編集、無効化、または削除できます。

ブレークポイントを編集するには、以下のとおりに実行します。

1. マッピングで、以下のいずれかの方法で Breakpoint Editor を開きます。
   - Alt+F9 キーを押します。
   - ［マッピング］-［デバッガ］-［ブレークポイントの編集］をクリックします。
   - トランスフォーメーションを右クリックし、［ブレークポイントの編集］を選択します。

   注: アイコン化されたトランスフォーメーションまたはマップレットのどの部分でも右クリックできます。 トランスフォーメーションが通常ビューで表示されている場合、タイトルバーを右クリックします。
2. ［インスタンス名］セクションで、インスタンス名を選択します。

   ヒント: マッピングのすべてのブレークポイントを表示するには、［インスタンス名］リストの<<ALL INSTANCES>>を選択します。
3. ［ブレークポイント］セクションで、ブレークポイントを選択します。
4. 条件またはブレークポイントタイプを変更します。
5. ブレークポイントを無効化するには、横に表示されたチェックボックスをクリアします。チェックボックスを選択すると、再度ブレークポイントを有効化できます。
6. 新規ブレークポイントを作成するには、上記のブレークポイント作成方法に従います。
7. 選択したブレークポイントを削除するには、［削除］をクリックします。
8. 選択したインスタンスのブレークポイントをすべて削除するには、［すべて削除］をクリックします。マッピング内のブレークポイントをすべて削除する場合、<<すべてのインスタンス>>を選択します。

デバッガの設定

ブレークポイントに加え、デバッガを設定する必要があります。Mapping Designer のデバッガウィザードを使用して、保存されたマッピングに対してデバッガを設定します。デバッガを設定する場合、Integration Service、ソース、ターゲットタイプなどのパラメータおよびメモリパラメータを入力します。 Integration Service を使用してデバッガを実行する前に、Integration Service の設定でデバッグを有効にする必要があります。

デバッガウィザードでは、以下のページが表示されます。

1. **概要。**このページでは、ウィザードの概要が表示されます。 このページでは、プロパティを設定する必要はありません。

2. **Integration Service とセッションタイプ。**セッションを実行する Integration Service を選択します。既存の再利用不可能なセッションまたは既存の再利用可能セッションに対してデバッガを実行することも、デバッグセッションインスタンスを作成することもできます。既存セッションに対してデバッガを実行すると、Integration Service はデバッグモードでセッションを実行します。 デバッグセッションを作成するときには、デバッガウィザード内にソースやターゲットの場所など、セッションプロパティのサブセットを設定します。
3. **セッション情報。**既存セッションに対してデバッガを実行する場合、セッション名を選択します。 デバッグセッションを作成した場合、セッションパラメータを設定します。
4. **セッション設定。**デバッグセッションを作成する場合、セッションを設定します。
5. **ターゲットオプション。**ターゲットデータのロードまたは破棄を選択します。

手順 1.デバッガのイントロダクション

デバッガウィザードの最初のページには、ウィザードの概略が表示されます。

手順 2 Integration Service とセッションタイプの選択

2 ページ目では、以下のオプションを選択できます。

- セッションを実行する Integration Service。 リストには、リポジトリに関連するすべての Integration Service が表示されます。 デバッグが有効になっている Integration Service を選択します。
- 既存の再利用不可能なセッションまたは既存の再利用可能セッションに対してデバッガを実行するか、デバッグセッションインスタンスを作成します。差分集計、FTP、またはセッション実行前/実行後のコマンドなどのセッションプロパティを使用しているマッピングをデバッグする場合、既存のセッションをデバッグモードで実行する必要があります。

［次へ］をクリックすると、Designer が Integration Service への接続をテストし、Integration Service でデバッグが有効になっているか確認します。 接続に失敗した場合、Designer から他の Integration Service を選択するように求められます。

手順 3. セッション情報の選択

デバッグセッションを実行するか既存セッションをデバッグモードで実行するかの選択によって、ウィザードでセッション情報が表示されるページが異なります。

デバッグモードによる既存セッションの実行

デバッグモードで既存セッションを実行することを選択した場合、デバッガウィザードにはマッピングを使用する現在のフォルダ内に格納されているすべてのセッションのリストが表示されます。使用するセッションを選択します。

複数のパーティションが設定されたセッション、またはグリッド上で実行するよう設定されたセッションに対しては、デバッガを実行できません。セッションのプロパティを変更するか、マッピングのデバッグセッションの作成を選択する必要があります。

デバッグセッションの実行

デバッグセッションの実行を選択した場合、いくつかのセッションパラメータを指定できます。デバッグセッションでユーザーが設定できないその他のパラメータすべてについて、デバッガはデフォルトのセッションパラメータを使用します。デバッガウィザードには、以下のタブを含むセッションページが表示されます。

- **接続。**ソースおよびターゲットの接続情報を選択します。

  リレーショナルソースおよびリレーショナルターゲットでは、Workflow Manager で設定されたデータベース接続を選択できます。リレーショナルターゲットの場合、ターゲットテーブルを切り詰めることができます。

  ファイルのソースおよびターゲットでは、ファイル名を上書きできます。デフォルトディレクトリは、$SourceFileDir および$TargetFileDir です。ソースファイルまたはターゲットファイルのデフォルトディレクトリを上書きするには、［セッションパラメータ］タブを使用します。そのファイル名フィールドにダイレクトパスは入力しないでください。
- **プロパティ。**ソースプロパティおよびターゲットプロパティを設定します。
- **Reader/Writer。**マッピング内のソースインスタンスおよびターゲットインスタンスに対し、Reader と Writer を設定します。

デバッグセッションを実行する場合、デバッガウィザードで設定しないすべてのプロパティにおいて、Integration Service はデフォルトのセッションプロパティを使用します。

［接続］タブのデバッグセッションで、ソースインスタンスとターゲットインスタンスのソース接続とターゲット接続をそれぞれ選択できます。

［Reader/Writer］タブで、デバッグセッションのソースインスタンスとターゲットインスタンスの Source Reader タイプと Target Writer タイプを選択できます。

［プロパティ］タブには、［Reader/Writer］タブで選択した Reader タイプまたは Writer タイプに応じたソースプロパティおよびターゲットプロパティが表示されます。［プロパティ］タブで設定してないすべてのセッションプロパティについて、Integration Service はデフォルト値を使用します。

デバッグセッションのセッションプロパティを設定できます。

手順 4. セッションの設定

デバッグセッションを設定する場合、ファイルのディレクトリ、行タイプ、およびメモリなどの情報を設定します。

以下の表に、デバッグセッションのセッションパラメータを示します。

セッションパラメータ	説明
行タイプ	Insert、Delete、Update、Driven アップデートストラテジを使用しているセッションのデフォルト行タイプ。
DTM バッファサイズ	デフォルトのメモリ割り当ては 12MB です。Unicode モードでの動作を設定された Integration Service で大量の文字を含むデータを使用したセッションを実行する場合、割り当てを 24MB に増やすこともできます。
パラメータファイル	マッピングでマッピングパラメータまたは変数が使用される場合のパラメータファイルの名前とディレクトリ。Integration Service の現在作業中のディレクトリが、パラメータファイルのデフォルトディレクトリです。

セッションパラメータ	説明
高精度を有効にする	これを選択すると、Integration Service は Decimal データタイプを精度 28 で処理します。Decimal データタイプを使用しないセッションの場合、高精度を有効化しないでください。
$Source 接続値	Integration Service で$Source 変数に対して使用するデータベース接続を入力します。
$Target 接続値	Integration Service で$Target 変数に対して使用するデータベース接続を入力します。
制約に基づくロード順序	制約に基づくロード順序を使用します。

手順 5 ターゲットオプションの設定

デバッガウィザードの最後のページで、以下のターゲットオプションを選択できます。

- **ターゲットデータの破棄。** デバッガ実行時に、ターゲットデータをロードするか破棄するかを選択できます。ターゲットデータを破棄した場合、Integration Service はターゲットに接続されません。
- **ターゲットデータの表示。**デバッグセッション実行中に、ターゲットウィンドウに表示するターゲットインスタンスを選択できます。

マッピングにマプレットが含まれる場合、[完了] をクリックするとデバッガはマプレットインスタンスのダイアログボックスを表示します。このダイアログボックスで、デバッグするマプレットを選択します。選択したマプレットをクリアする場合、Ctrl キーを押して、マプレットを選択します。

デバッグするマプレットを選択すると、デバッガ実行時に Designer はそのマプレットを展開し、個別のトランスフォーメーションを表示します。

デバッグするマプレットを選択しなかった場合、Designer はワークスペースでそのマプレットを展開しません。そのため、マプレット内で以下のタスクを完了できません。

- トランスフォーメーションデータの監視または変更。
- 式の評価。
- ブレークポイントの編集。
- トランスフォーメーションインスタンスへの進行。

デバッガの実行

デバッガウィザードを完了すると、Integration Service ではセッションが開始され、デバッガが初期化されます。 初期化されたデバッガは、Mapping Designer から発行されるブレークポイントとコマンドによって、実行と一時停止を繰り返します。デバッガは、以下のいずれかの状態にあります。

- **初期化中。**Designer が Integration Service に接続します。
- **実行中。**Integration Service がデータを処理します。
- **一時停止。**Integration Service によりブレークが検出され、デバッガが一時停止されます。

注: 複数のユーザーが同時に同じマッピングをデバッグするためには、各ユーザーは［ツール］-［オプション］-［デバッグ］タブで異なるポート番号を設定する必要があります。

デバッガは、高可用性の機能を使用しません。

初期化中の状態

デバッガを実行すると、Designer が Integration Service に接続し、Integration Service がセッションを初期化します。 初期化中、Designer はナビゲータウィンドウを閉じ、他のツールへの切り替え、リポジトリの保存、フォルダを開くなどの機能を無効化します。これらの機能は、デバッガが停止するまで無効になります。

実行状態

初期化が完了するとデバッガは一時停止の状態になり、処理を続行するためのコマンドを待ちます。続行すると、デバッガは実行中の状態になります。Integration Service によりデータが変換され、ブレークポイント条件に対してデータが評価されます。 デバッガは、Integration Service によってブレークポイントが検出される、ユーザーがブレークコマンドを発行する、またはセッションが終了するまで、実行中の状態になります。

一時停止状態

Integration Service によってブレークが検出されると、デバッガは一時停止されます。 以下のブレーク条件で、デバッガは一時停止されます。

- Integration Service によって設定済みのブレークポイント条件が検出された場合。
- ブレークポイントが関連付けられていないインスタンスへの続行を Integration Service に指示した場合。
- 手動のブレークコマンドを発行した場合。
- Integration Service によって致命的なエラーが検出された場合。
- Integration Service によるすべてのデータの評価が完了した場合。 ［デバッガ］タブには、セッション完了のメッセージが表示されます。セッションを続行すると、デバッガはターゲットウィンドウとトランスフォーメーションウィンドウをクリアします。

デバッガが一時停止している間、トランスフォーメーション出力データを確認および変更できます。デバッガは、セッションを続行するか終了するまで一時停止の状態になります。

デバッガタスク

デバッガを実行するとき、複数のタスクを実行できます。監視する情報のタイプと実行するタスクは、デバッガの状態により異なります。例えば、ログの監視はデバッガの状態にかかわらず実行できますが、データの変更はデバッガが一時停止の状態になければ実行できません。以下のタイプのタスクを完了できます。

- **セッションの監視。**Integration Service によってデバッガが実行される間、Mapping Designer にはセッションの監視に使用するインジケータとウィンドウが表示されます。
- **データおよびブレークポイントの変更。**デバッガが一時停止する際に、出力データ、行インジケータ、およびブレークポイント条件を変更できます。
- **式の評価。**デバッガが一時停止する際に、式エディタを起動し、トランスフォーメーションの現在のデータに対して式を評価できます。 デバッガは、メッセージボックスに式の結果を返します。選択したトランスフォーメーションのポートを使用して、式を入力できます。マッピング変数も評価できます。
- **Integration Service へのコマンドの発行。**ブレーク、続行、停止などのコマンドを Integration Service に対して発行できます。

以下の表に、デバッガの状態に応じて実行できるさまざまなタスクを示します。

タスク	説明	デバッガの状態	アクセス先
ログの監視	出力ウィンドウで、セッションログおよびデバッグログを監視します。	初期化中 実行中 一時停止	- 出力ウィンドウを表示
ターゲットデータの監視	デバッガが初期化中から実行中の状態になると、Designer はターゲットウィンドウを表示します。Integration Service がターゲットデータを処理するに従い、更新されたターゲットデータが表示されます。	実行中 一時停止	- ターゲットインスタンスウィンドウを表示 - ターゲットデータコマンドを表示
デバッグインジケータの監視	ブレークポイントおよびデータフローの監視に役立つマッピングオブジェクトに、デバッグインジケータが表示されます。	実行中 一時停止	- マッピングオブジェクトを表示
トランスフォーメーションデータの監視	デバッガが初期化中から実行中の状態になると、Designer はインスタンスウィンドウを表示します。デバッガが一時停止すると、インスタンスウィンドウにはブレークの原因になったトランスフォーメーションのデータが表示されます。他のトランスフォーメーションを表示するように選択することもできます。	一時停止	- インスタンス表示ウィンドウ - 現在のインスタンスコマンドを表示
データの変更	デバッガが一時停止している間、出力データおよび行パラメータを変更できます。	一時停止	- インスタンス表示ウィンドウ
式の評価	デバッガが一時停止している間、式エディタを使用してトランスフォーメーションのマッピング変数および式を評価します。	一時停止	- 式コマンドを評価
手動でのブレークコマンドの発行	トランスフォーメーションデータを表示する場合、手動のブレークコマンドを発行します。	実行中	- 現在ブレーク中のコマンド
ブレークポイントの編集	Integration Service が、変更されたブレークポイントを即座に評価し始めます。	実行中 一時停止	- ブレークポイントコマンドを編集
データのリフレッシュ	変更したデータをリフレッシュできます。データをリフレッシュすると、Intgration Service が変更の結果を返します。有効でないデータを入力した場合、再度編集してから処理を続行できます。	一時停止	- インスタンスウィンドウからのデータのリフレッシュコマンド
処理の続行	データを表示および変更した後で、デバッガを続行できます。セッションを続行する場合、いくつかのオプションがあります。	一時停止	- 「次のインスタンス」のコマンド - 「インスタンスへの移動」のコマンド - 「続行」のコマンド

タスク	説明	デバッガの状態	アクセス先
デバッガの停止	デバッガを停止します。	初期化中 実行中 一時停止	- 「デバッガの停止」のコマンド
状態リクエスト	デバッガの状態をリクエストすると、Integration Service によって出力ウィンドウにデバッガの状態が表示されます。	実行中 一時停止	- 状態リクエストコマンド

パーシステント値に関する作業

シーケンスジェネレータおよびマッピング変数を持つマッピングに対してデバッガを実行する場合、Integration Service によりパーシステント値が保存または破棄される場合があります。

- **パーシステント値の破棄。**デバッグセッションを実行した場合、またはデバッグモードでセッションを実行してターゲットデータを破棄した場合、Integration Service により、生成されたシーケンス番号またはマッピング変数の最終値がリポジトリに保存されません。
- **パーシステント値の保存。**デバッグモードでセッションを実行し、ターゲットデータを破棄しなかった場合、Integration Service により生成されたシーケンス番号およびマッピング変数の最終値がリポジトリに保存されます。 シーケンスジェネレータおよびノーマライザトランスフォーメーションの最終値は、トランスフォーメーションプロパティに表示されます。

Designer の動作

デバッガが開始されると、以下のタスクは実行できなくなります。

- フォルダを閉じる、または他のフォルダを開く。
- ナビゲータを使用する。
- ［保存］など、リポジトリ機能を実行する。
- マッピングを編集する、または閉じる。
- Designer で、Target Designer など他のツールに切り替える。
- Designer を閉じる。

注: デバッグ実行中、動的パーティションは無効になります。

上記のタスクは、デバッガが停止すると実行できます。

デバッガの監視

デバッガを実行する場合、以下の情報を監視できます。

- **セッションステータス。**セッションのステータスを監視します。
- **データ移動。**トランスフォーメーション内を移動するデータを監視します。
- **ブレークポイント。**ブレークポイント条件を満たすデータを監視します。
- **ターゲットデータ。**行ごとに、ターゲットデータを監視します。

Mapping Designer には、セッションの監視に使用するウィンドウとデバッグインジケータが表示されます。

- **デバッグインジケータ。**トランスフォーメーションのデバッグインジケータは、ブレークポイントおよびデータフローの追跡に使用します。
- **インスタンスウィンドウ。**デバッガが一時停止する際に、インスタンスウィンドウにトランスフォーメーションデータと行情報を表示できます。
- **ターゲットウィンドウ。**マッピング内の各ターゲットのターゲットデータが表示されます。
- **出力ウィンドウ。** Integration Service によって、出力ウィンドウの以下のタブにメッセージが書き込まれます。
  - **［デバッガ］タブ。**［デバッガ］タブには、デバッグログが表示されます。
  - **［セッションログ］タブ。**［セッションログ］タブには、セッションログが表示されます。
  - **［通知］タブ。**リポジトリサービスからのメッセージが表示されます。

場合によっては、デバッガを監視する間、トランスフォーメーション出力データを変更し、データフローにおける後続のトランスフォーメーションまたはターゲットへの影響を確認する必要があります。また、セッションの詳細をより詳しく監視するために、ブレークポイント情報の編集や追加を実行する場合もあります。

デバッグインジケータの監視

セッション中、マッピングはトランスフォーメーションの左上にインジケータを表示します。これは、ブレークポイント、データフロー、およびデバッガステータスの監視に活用できます。

- **ブレークポイントインジケータ。**Integration Service によって初期化が完了した後、マッピング内のブレークポイントが読み込まれ、各トランスフォーメーションに停止信号が表示されてブレークポイント条件が示されます。 デバッガが実行中または一時停止の状態にあれば、ブレークポイントのインジケータを表示できます。Mapping Designer では、グローバルブレークポイントのインジケータは表示されません。
- **現在のソース修飾子インジケータ。** デフォルトでは、Integration Service はデバッガを実行すると、ターゲットロード順グループの各ソース修飾子からのデータを同時に処理します。 Mapping Desiger は、現在のソース修飾子すべてを点滅矢印で表示します。

  注: Integration Service を、ジョイナトランスフォーメーションに接続されているソースを順番に読み込むように設定できます。
- **現在のトランスフォーメーションインジケータ。**デバッガで、Integration Service によってブレークが検出された時に処理されていたトランスフォーメーションまたはマップレットは、黄色の矢印で表示されます。 このトランスフォーメーションは、現在のトランスフォーメーションと呼ばれます。デバッガは、Integration Service によって現在のトランスフォーメーションでブレークが検出された時に処理されていたその他のトランスフォーメーションすべてが、青色の矢印で表示されます。
- **デバッガステータス。**デバッガが開始されると、Mapping Designer によってツールバーの横にデバッガの状態を示すデバッガアイコンが表示されます。 デバッガが初期化中または実行中の場合、アイコンは回転します。デバッガが一時停止している場合、アイコンも停止します。

トランスフォーメーションデータの監視

デバッガが一時停止する際に、Integration Service によってブレークが検出されたマッピングまたはマップレット内のトランスフォーメーションのインスタンスウィンドウに、データのカレント行が表示されます。 これは、現在のトランスフォーメーションです。インスタンスウィンドウに表示されたリストから他のトランスフォーメーションを選択すると、デバッガが停止した時点のトランスフォーメーションのデータを表示できます。

デバッガウィザードを完了した後でデバッグするマプレットを選択していない場合、デバッガ実行中はマプレット内のトランスフォーメーションを監視または表示できません。

［デバッガデータの表示］ウィンドウに表示するカラムを選択するには、［ツール］-［オプション］-［デバッグ］タブをクリックします。カラムの表示の詳細については、「デバッグオプションの設定」 (ページ 24)を参照してください。

注: マッピングにカスタムトランスフォーメーションが含まれている場合、インスタンスウィンドウにはすべての入力グループおよび出力グループのポート情報が表示されます。

インスタンスウィンドウに、以下の情報を表示することができます。

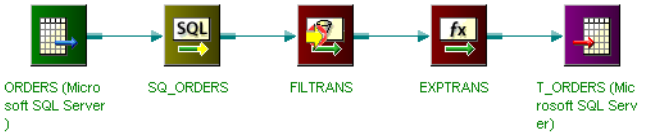
- **ポート名。**他のトランスフォーメーションまたはターゲットに接続されているポートがすべて表示されます。
- **行 ID。**トランスフォーメーションから渡される行の番号が表示されます。
- **値。**各ポートの値が表示されます。 ポートにバイナリデータが格納されている場合、この列には<raw data>が表示されます。ポートにデータが格納されていない場合、または NULL の場合、このカラムには<データがありません>が表示されます。ルータスフォーメーションでは、グループ条件に一致する入力ポートおよび出力ポートの値が表示されます。グループ条件に一致しない出力ポートの場合、値カラムには<データがありません>が表示されます。
- **データタイプ。**ポートのデータタイプが表示されます。
- **長さ/精度、および位取り。**ポートの長さ/精度、および位取りが表示されます。
- **NULL インジケータ。**列にに NULL 値が含まれる場合、NULL インジケータの列が選択され、<no data available>が表示されます。
- **行タイプ。**挿入、更新、削除、拒否、フィルタリング、または適用不可。 ポートカラムには<<行タイプ>>が、値カラムには行タイプの値が表示されます。シーケンスジェネレータトランスフォーメーションのように適用不可の場合、行タイプは表示されません。
- **ポートインジケータ。**ポート名の横に、次のいずれかのインジケータが表示されます。
  - **カレント行。**カレント行にポートがあることを示します。
  - **前の行。**前の行にポートがあることを示します。
  - **カレント行、デフォルト値。**カレント行の列の値がデフォルト値であることを示します。
  - **前の行、デフォルト値。**前の行のカラムの値がデフォルト値であることを示します。
  - **カレント行、変更可能。**カレント行で変更できる出力ポートを示します。

  ヒント: ポートインジケータに関するツールチップを表示するには、インジケータ上でポインタを移動させます。

現在のトランスフォーメーションに先立って、現在のトランスフォーメーションまたはパイプラインのトランスフォーメーションがインスタンスウィンドウに表示されると、現在の行が表示されます。パイプラインにおいて現在のインスタンスに後続するトランスフォーメーションがインスタンスウィンドウに表示されると、前の行が表示されます。ポート名の横には、現在の行または前の行を示すインジケータも表示されます。

例えば以下のマッピングでは、現在のトランスフォーメーションのインジケータが示すとおり、FILTRANS が現在のトランスフォーメーションです。FILTRANS または SQ_ORDERS のインスタンスウィンドウが表示されている場合、現在の行が表示されています。EXPTRANS に切り替えた場合、Integration Service はカレント行を EXRTRANS で処理していないため、前の行が表示されます。

以下の図に、ポート処理のインジケータと共にトランスフォーメーションを示します。



インスタンスウィンドウにデフォルト値が表示されている場合も、現在の行および前の行のインジケータが表示されます。デバッガは、インスタンスウィンドウにデフォルト値インジケータを表示する場合、ブレークポイントのデフォルト値を評価するために使用するルールと同じルールを使用します。

注: デバッガは、削除フラグが設定された行を即座に削除しません。このような行には、無効、エラー、およびフィルタ済の行が含まれています。Integration Service によって、ROW_FLAG が NULL に設定され、-3（エラー）のように負の数字で行タイプが指定されます。 冗長データトレースでのみ、セッションログにエラーインジケータが表示されます。Integration Service は、後でパイプラインから行を削除します。

デバッガの続行

データの確認または変更後に、以下のようにしてデバッガを続行できます。

- **次のブレークまで続行。** 次のブレークまで続行するには、ツールバーまたは［マッピング］-［デバッガ］のメニューオプションの［継続］をクリックします。デバッガは、次のブレークに遭遇するまで続行されます。
- **次のインスタンスまで続行。**次のインスタンスまで続行するには、ツールバーまたは［マッピング］-［デバッガ］のメニューオプションの［継続］をクリックします。 デバッガは、次のトランスフォーメーションまたはブレークに遭遇するまで続行されます。現在のインスタンスに複数のトランスフォーメーションインスタンスへの出力がある場合、デバッガは最初に処理するインスタンスで停止します。デバッガウィザードでの操作を完了した時にデバッグするマプレットを選択しなかった場合、デバッガはマプレットの後のインスタンスまで続行されます。
- **指定したインスタンスへの移動。**指定したインスタンスまで続行するには、マッピング内のトランスフォーメーションインスタンスを選択し、ツールバーまたは［マッピング］-［デバッガ］のメニューオプションの［インスタンスへの移動］をクリックします。 デバッガは、マッピング内の選択したトランスフォーメーションまたはブレークに遭遇するまで続行されます。

  関連するブレークポイントがない場合でも、マッピング内のコネクトされたトランスフォーメーションまで進むことができます。以下のインスタンスには進むことができません。

  - ソース
  - ターゲット
  - コネクトされていないトランスフォーメーション
  - デバッギングに選択されていないマプレット

ターゲットデータの監視

デバッガが実行されると、Designer はターゲットデータをキャッシュします。デバッガが実行中または一時停止の状態にある場合、ターゲットウィンドウにキャッシュされたデータが表示されます。デバッガウィザードでターゲットデータを破棄するよう設定している場合でも、キャッシュされたターゲットデータを表示できます。

ターゲットデータの各行では、以下の情報が表示されます。

- **行 ID。**キャッシュされたターゲットデータの行番号。
- **行タイプ。**挿入、更新、削除、拒否、またはフィルタリング。
- **カラム名および値。**キャッシュ内の各行のカラム名と値。

マッピングに複数のターゲットが格納されている場合、表示するターゲットを選択できます。［デバッガターゲットデータの表示］ウィンドウに示されたリストから、表示するターゲットを選択します。リストには、デバッガウィザードで選択したターゲットが表示されます。ウィザードで選択しなかったターゲットを表示するには、［マッピング］-［デバッガ］-［ターゲットデータの表示］をクリックします。

ターゲットウィンドウには、1,000 行まで表示されます。1,000 行以上になると、Designer は最初の行からデータを上書きします。［デバッガターゲットデータの表示］ウィンドウで右クリックして［データをクリア］を選択すると、キャッシュおよび［デバッガターゲットデータの表示］ウィンドウをクリアできます。

デバッグログの監視

デバッガを初期化する場合、Integration Service によって出力ウィンドウの［デバッガ］タブにデバッグログの書き込みが開始されます。 Integration Service によって、デバッグログに以下のタイプのメッセージが書き込まれます。

- セッションの初期化
- ユーザーリクエストの確認
- Integration Service によるブレークの検出
- デバッグエラー
- デバッガの状態
- セッションの停止

出力ウィンドウの最後の行に Integration Service の準備ができたことが示されている場合、デバッガは一時停止されています。 Integration Service は、［継続］、［Step to Next］、または［停止］などのコマンドを発行するまで一時停止の状態にあります。

注: デバッガが開始されない場合、または突然停止しても Integration Service によって出力ウィンドウにメッセージが書き込まれない場合、Administration Console の［ドメイン］ページで詳細を確認します。

以下の表に、出力ウィンドウに表示されるデバッグログメッセージの例を示します。

デバッガのアクション	デバッグログメッセージ例
デバッガは初期化されます。	Establishing a connection to the Integration Service dwilliam... Integration Service dwilliam acknowledged its ability to debug a mapping. Initializing debugger... Sending request to Integration Service dwilliam to debug mapping Customers... Waiting for Integration Service to initialize... Establishing communication with the DTM... Established communication with the DTM. Waiting for DTM initialization to complete... Integration Service は ASCII モードで実行されています。 Integration Service は高精度の無効で実行中です。 Integration Service の初期化が完了しました。
初期化のあと、Integration Service はマッピング内のブレークポイントを確認します。	DSQ Execution Order: Integration Service は準備が完了しました。 Sending breakpoints to the Integration Service... Integration Service はブレークポイントを設定する要求を確認しました。
Integration Service によってブレークポイントが検出されました。	Integration Service がデータブレークポイントで停止しました。 List of breakpoints that the Integration Service hit: #3 Data(2,"CUSTOMER_ID > [14]") The current transformation is Exp_Customers. Integration Service は準備が完了しました。

デバッガのアクション	デバッグログメッセージ例
［継続］コマンドを発行します。	Sending request to continue... Integration Service は続行する要求を確認しました。
次のインスタンスに進むコマンドを発行します。Integration Service は承認し、要求を実行します。	Sending request to go to next instance... Integration Service は次のインスタンスへ移動する要求を確認しました。 The current transformation is Exp_Customers. Integration Service は準備が完了しました。
インスタンスに進むコマンドを発行します。Integration Service は承認し、要求を実行します。	Sending request to step to instance... Integration Service はインスタンスへ移動する要求を確認しました。 The current transformation is SQ_CUSTOMERS. Integration Service は準備が完了しました。
インスタンスウィンドウのデータを変更します。	Sending request to modify data for transformation Exp_Customers, port NEW_CUST_ID, to value [1234566...] Integration Service はデータを修正する要求を確認しました。 Sending request to continue... Integration Service は続行する要求を確認しました。
セッションは完了しました。Integration Service は、デバッガをシャットダウンするための最終コマンドを待っています。	Response from the Integration Service: Session completed <successfully/with failure>. Last chance to look at target data. Debugger will shut down when we continue. Integration Service は準備が完了しました。
デバッガは完了しました。	Integration Service のシャットダウン中です..... Integration Service の実行が完了しました。 Debugger shutdown complete.

セッションが完了した時点でログを保存する場合、［デバッガ］タブ内で右クリックし、メニューから［出力の保存］を選択します。

Workflow Monitor の使用

Workflow Monitor には、実行中および完了済みのワークフローとセッションを表示できます。Integration Service によってデバッグワークフローが使用され、デバッグセッションが実行されます。 Workflow Monitor には、デバッガウィザードで選択したセッションタイプに応じて異なる方法で、デバッグワークフローとセッションの名前が表示されます。

以下の表に、Wokrflow Monitor に表示される各デバッグのセッションタイプに応じたワークフローとセッション名を示します。

セッションタイプ	ワークフロー名	セッション名
既存の再利用可能なセッション	DebugWorkflow_mappingname	DebugInst_sessionname
既存の再利用不可能なセッション	workflowname	sessionname
デバッグセッション	DebugWorkflow_mappingname	DebugInst_DebugSession_mappingname

Workflow Monitor は、すべてのデバッガセッションおよびワークフローの実行タイプカラムに、デバッグ実行モードを表示します。Workflow Monitor からログにアクセスすることもできます。

注: Workflow Monitor を使用して、デバッグセッション、デバッグワークフロー、またはデバッグモードで実行されているセッションを再起動、強制終了、または停止することはできません。

データの変更

デバッガが一時停止すると、インスタンスウィンドウには現在のインスタンス、マッピング内のトランスフォーメーションには現在のインスタンスのインジケータが表示されます。デバッガがデータのブレークポイントで一時停止すると、現在のインスタンスに以下の変更を加えることができます。

- **出力データの変更。**現在のトランスフォーメーションの出力データを変更できます。 セッションを続行すると、Integration Service によってデータが検証されます。 通常のセッションでポートからポートにデータを渡す場合に、実行する検証と同じ検証が実行されます。
- **NULL データの NOT NULL への変更。**NULL カラムの選択を取り消し、値カラムに値を入力して NULL データを NOT NULL に変更します。
- **NOT NULL の NULL への変更。**NULL カラムを選択し、NOT NULL データを NULL に変更します。Designer によって、この変更を実行するか確認するように求められます。
- **行タイプの変更。**アップデートストラテジ、フィルタ、またはルータトランスフォーメーション行タイプを変更します。

  ルータトランスフォーメーションの場合、行タイプを変更してユーザー定義グループのグループ条件の評価を上書きできます。例えば、グループ条件が False に評価された場合、その行は出力ポートから次のトランスフォーメーションまたはターゲットに渡されません。インスタンスウィンドウには<データがありません>が表示され、行タイプはフィルタリングになります。フィルタリングされた行を次のトランスフォーメーションまたはターゲットに渡すには、行タイプを挿入に変更します。同様に、グループ条件に一致するグループの場合、行タイプを挿入からフィルタリングに変更します。

データを変更すると、セッションを続行する前にキャッシュをリフレッシュできます。リフレッシュコマンドを発行すると、Designer が現在のトランスフォーメーションに関する要求を処理し、入力したデータが有効かどうかが表示されます。セッションを続行する前に、再度データを変更できます。

制限

以下の出力ポートでは、データを変更できません。

- **ノーマライザトランスフォーメーション。**生成されたキーと生成されたカラム ID ポート。

- **ランクトランスフォーメーション。**RANKINDEX ポート。
- **ルータトランスフォーメーション。**すべての出力ポート。
- **シーケンスジェネレータトランスフォーメーション。**CURRVAL および NEXTVAL ポート。
- **ルックアップトランスフォーメーション。**動的キャッシュの使用が設定されたルックアップトランスフォーメーションの NewLookupRow ポート。
- **カスタムトランスフォーメーション。**現在の出力グループ以外の出力グループ内のポート。
- **Java トランスフォーメーション。**現在の出力グループ以外の出力グループ内のポート。

さらに、以下に関連するデータは変更できません。

- デバッグに選択されていないマプレット
- 入力ポートまたは入出力ポート
- エラーブレークポイントでデバッガが一時停止した時の出力ポート

式の評価

デバッガが一時停止すると、式エディタを使用して、選択したトランスフォーメーション内のマッピング変数およびポートを使用する式を評価します。

以下のトランスフォーメーションで式を評価できます。

- アグリゲータトランスフォーメーション
- 式トランスフォーメーション
- フィルタトランスフォーメーション
- ランクトランスフォーメーション
- ルータトランスフォーメーション
- アップデートストラテジトランスフォーメーション

式を作成する場合、トランスフォーメーション内のポートへの参照を使用します。式エディタは、マッピング内の他のトランスフォーメーションのポートは表示しません。式を評価する場合、デバッガはメッセージボックス内に式の結果を返します。

マッピング変数を使用した式の評価

マッピングで変数を定義した場合、マッピング変数の Start Value または Current Value を評価できます。マッピング変数の Start Value を見つけるには、マッピング変数のみを評価します。

デバッガは、以下のプロセスでマッピング変数の Start Value を決定します。

1. **パラメータファイル。**パラメータファイルを使用する場合、デバッガはパラメータファイルに値を返します。
2. **リポジトリ。**パラメータファイルを使用しない場合、デバッガはリポジトリに値を返します。
3. **初期値。**リポジトリに変数の値が保存されていない場合、デバッガはマッピング変数で設定された初期値を返します。
4. **デフォルト値。**マッピング変数の初期値を設定していない場合、デバッガはデータタイプに応じたデフォルト値を返します。

マッピング変数の Current Value を見つけるには、SetMaxVariable や SetCountVariable のような変数関数でマッピング変数を評価します。Designer は、マッピング内のトランスフォーメーションに関連するマッピング

変数のみ表示します。マプレットに関連するマッピング変数を表示するには、デバッガ実行中に展開されたマプレット内のトランスフォーメーションを選択します。

式の評価手順

デバッガが一時停止している間に式を評価するには、以下の手順を実行します。

デバッガが一時停止している間に式を評価するには：

1. 式を評価するトランスフォーメーションを選択します。
2. ［マッピング］-［デバッガ］-［式の評価］をクリックします。
3. 式エディタに、選択したトランスフォーメーションのポートを参照する式を入力します。
4. 式を検証するには、［検証］をクリックします。
5. 式を評価するには、［評価］をクリックします。

   有効な式の場合、Integration Service がその式を評価します。 Integration Service は、メッセージボックス内に式の結果を返します。

ブレークポイント情報および設定のコピー

Designer は、ブレークポイントの情報およびデバッガの設定をワークスペースファイルに保存します。マッピングをコピーした場合、他のフォルダへのショートカットを作成した場合、または他の PowerCenter クライアントマシンからマッピングをデバッグする場合、デバッガメニューの［設定を保存］および［設定のロード］オプションを使用してブレークポイントと設定を転送できます。

マッピングに関連するブレークポイントおよび設定を保存するには、以下のとおりに実行します。

1. 保存するブレークポイントと設定を含むマッピングを開きます。
2. ［マッピング］-［デバッガ］-［Save Configuration］をクリックします。
3. .dcf の拡張子でファイルを保存します。
4. 既存の.dcf ファイルを選択した場合、Designer はファイルに設定情報を追加します。
5. ファイルにマッピングの設定情報が格納されている場合、Designer によって、設定を上書きするかどうかが確認されます。

ブレークポイントおよび設定の転送

あるマッピングに保存したブレークポイントおよび設定の情報は、別のマッピングに転送することができます。

他のマッピングにブレークポイントおよび設定をロードするには、以下を実行します。

1. .dcf ファイルを、設定をロードする PowerCenter クライアントマシンからアクセスできるようにします。
2. 設定をロードするには、マッピングを開きます。
3. ［マッピング］-［デバッガ］-［コンフィグレーションのロード］をクリックします。

   ［DCF ファイルを開きます］ダイアログボックスが表示されます。
4. .dcf 設定ファイルを選択し、［開く］をクリックします。

   ファイルに複数の設定が格納されている場合、または Designer が一致するマッピング名を見つけられない場合、Designer はファイル内のマッピング設定のリストを表示します。

5. マッピングを選択して、[OK] をクリックします。

   Designerd では、一致するマッピングオブジェクトのブレークポイントと設定がすべてロードされます。Integration Service やターゲットデータを破棄するためのオプションなど、マッピングオブジェクトに依存しない設定もすべてロードします。

   注: デバッガが有効な場合、ブレークポイントはロードできません。

デバッガのトラブルシューティング

デバッガセッションを実行する際に、以下のエラーが発生したとします。

```
Establishing a connection to the Integration Service <name>...
Integration Service <name> acknowledged its ability to debug a mapping.
Initializing debugger...
bind: (WSAEADDRINUSE)Address already in use
```

このエラーは、デバッガが使用しているポートが、既に他のアプリケーションで使用されている場合に発生します。使用するデバッガに他のポートを割り当てます。

デバッガに他のポートを割り当てるには：

1. [ツール] - [オプション] - [デバッグ] をクリックします。
2. [ポート] フィールドに新しいポート番号を割り当てます。または、[最小-最大の間のポート番号を自動的にピックアップ しします] を選択して、デバッガに一定範囲のポートを割り当てます。

   5001 から 32000 までのポートを選択できます。または、5001 から 32000 までの範囲を設定できます。

第 10 章

データリネージの表示

この章では、以下の項目について説明します。

データリネージの表示の概要

Metadata Manager を使用して、ソースからターゲットへの PowerCenter のデータフローを追跡できます。Metadata Manager のデータリネージでは、データの発生場所、データの変換方法とデータを変換するオブジェクトの種類、データの終了地点を分析できます。 データフローを、単一の PowerCenter リポジトリ内で、または、複数の PowerCenter リポジトリ全体で分析できます。

Designer では、PowerCenter オブジェクトのデータリネージを表示できます。 PowerCenter オブジェクトのデータリネージを表示する場合、Designer は Metadata Manager アプリケーションに接続し、Metadata Manager ウェアハウスからのデータリネージ情報を抽出します。ブラウザウィンドウに、データリネージ図が表示されます。

PowerCenter リポジトリのデータリネージ分析では、以下の PowerCenter オブジェクトが 1 つ以上表示されます。

- **リポジトリ。**複数の PowerCenter リポジトリから Metadata Manager にオブジェクトをロードできます。データリネージ内では、Metadata Manager によって各メタデータオブジェクトのリポジトリが表示されます。
- **データ構造。**PowerCenter オブジェクトのデータリネージには、ソース定義、ターゲット定義、トランスフォーメーションデータ構造が含まれます。
- **フィールド。**フィールドは、メタデータを格納するデータ構造内のオブジェクトです。 PowerCenter オブジェクトのデータリネージには、ソース、ターゲット、およびトランスフォーメーションのポートが含まれます。

データリネージのアクセスの設定

PowerCenter と Metadata Manager で PowerCenter オブジェクトのリネージを表示するように設定するには、以下のタスクを実行します。

1. **Metadata Manager が実行されていることの確認。**Informatica Administrator 内に Metadata Manager サービスを作成するか、または有効になった Metadata Manager サービスがデータリネージ分析を実行する PowerCenter リポジトリを含むドメインに存在することを確認します。
2. **PowerCenter リポジトリメタデータのロード。**Metadata Manager 内に PowerCenter リポジトリ用リソースを作成し、PowerCenter リポジトリメタデータを Metadata Manager ウェアハウスにロードします。
3. **PowerCenter リポジトリと Metadata Manager 間の接続のセットアップ。**Administrator ツールで、Metadata Manager サービスと、PowerCenter リポジトリのリソース名を設定します。
4. **Web ブラウザの設定。** PowerCenter オブジェクトでデータリネージ分析を実行する場合、Designer はシステムに設定されているデフォルトブラウザでデータリネージ分析を起動します。データリネージでは、ブラウザに Flash 9 ビューアがインストールされている必要があります。

Designer からのデータリネージの実行

データリネージには、PowerCenter Designer ツールからアクセスできます。ソース定義、ターゲット定義、トランスフォーメーション、マプレットおよびマッピングにデータリネージを表示できます。

PowerCenter Designer からデータリネージを実行する手順

1. Designer で、データリネージを実行するオブジェクトを含む PowerCenter フォルダを開きます。
2. 適切な作業領域でオブジェクトを開きます。

   例えばトランスフォーメーションでデータリネージを実行する場合、Transformation Developer でトランスフォーメーションを開きます。
3. ワークスペースで、**[データリネージ]** を右クリックして選択します。
4. Informatica Metadata Manager Business Glossary ブラウザでユーザー資格情報を入力します。

   ブラウザウィンドウに、対象オブジェクトのデータリネージが表示されます。

   注: データリネージ分析のパフォーマンスは、Metadata Manager アプリケーションを実行しているマシンで使用できるシステムリソースによって異なります。複数のオブジェクトに複数のデータリネージ分析を同時に実行すると、パフォーマンスに影響します。

データリネージにアクセスすると、データリネージ図内の各オブジェクトの詳細を表示できます。データリネージを PNG イメージファイルにエクスポートすることも、図を印刷することもできます。また、電子メールで他のユーザーにデータリネージを送信できます。

第 11 章

オブジェクトの比較

この章では、以下の項目について説明します。

オブジェクトの比較の概要

Designer では、同じ種類の 2 つのリポジトリオブジェクトを比較して、その違いを確認できます。例えば、マッピングでターゲット定義を使用したいけれども似たようなターゲット定義が 2 つあるとします。これらのターゲット定義を比較して、どちらに目的のカラムが含まれているかを調べることができます。2 つのオブジェクトを比較すると、Designer にその属性が横並びで表示されます。

以下の種類のオブジェクトを比較することができます。

- **ソース。**2 つのソース同士、2 つのソースショートカット同士、またはソースとソースショートカットを比較することができます。
- **ターゲット。**2 つのターゲット同士、2 つのターゲットショートカット同士、またはターゲットとターゲットショートカットを比較することができます。
- **トランスフォーメーション。**2 つのトランスフォーメーション同士、2 つのトランスフォーメーションショートカット同士、またはトランスフォーメーションとトランスフォーメーションショートカットを比較することができます。
- **マッピングおよびマップレット。**2 つのマッピング同士、2 つのマッピングショートカット同士、またはマッピングとマッピングショートカットを比較することができます。 2 つのマップレット同士、2 つのマップレットショートカット同士、またはマップレットとマップレットショートカットを比較することもできます。
- **インスタンス。**マッピングおよびマップレット内では、2 つのソースインスタンス同士、2 つのターゲットインスタンス同士、または 2 つのトランスフォーメーション同士を比較することができます。 マッピング内では、2 つのマップレットインスタンス同士を比較することもできます。
- **フォルダ。**Repository Manager 内では、2 つのフォルダを比較できます。
- **オブジェクトのショートカットバージョン。**Repository Manager 内では、オブジェクトの 2 つのバージョンを比較できます。

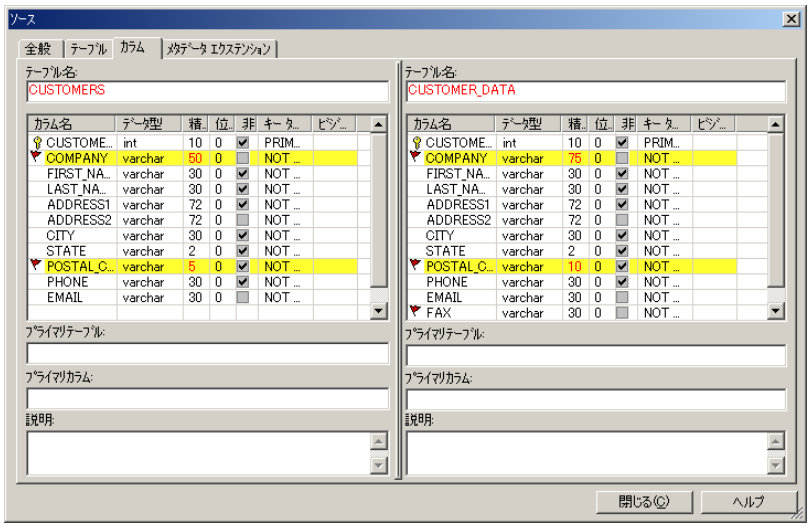
オブジェクトは、以下の状況で比較することができます。

- **開いているフォルダ内に両方のオブジェクトが存在する場合。**開いている任意のフォルダ内のオブジェクトを、開いている任意のフォルダ内の同じタイプの他のオブジェクトと比較できます。 フォルダは、異なるリポジトリ内にあってもかまいません。タイプが異なるオブジェクト同士を比較することはできません。

- **オブジェクトをインポートする場合。**同じ名前のオブジェクトが含まれているフォルダにオブジェクトをインポートする場合、Designer を使用してオブジェクトを比較し、上書きするかまたは既存のオブジェクトを使用するかを決定できます。
- **オブジェクトをコピーする場合。**同じ名前で同じ種類のオブジェクトが含まれているフォルダにオブジェクトをコピーする場合、Designer を使用してオブジェクトを比較し、上書きするかまたは既存のオブジェクトを使用するかを決定できます。
- **オブジェクトを別のリポジトリにコピーまたはデプロイする場合。** 別のリポジトリのフォルダにオブジェクトをコピーまたはデプロイする場合、コピーウィザードまたはデプロイメントのウィザードにより、デプロイまたはコピーするオブジェクトとターゲット内のオブジェクトが比較されます。
- **複数のバージョンがあるオブジェクトを表示する場合。** オブジェクトの以前のバージョンとオブジェクトの現在のバージョンを比較して、最新バージョンに行われた変更を確認できます。
- **結果ビューウィンドウでオブジェクトを表示する場合。** 結果ビューウィンドウにオブジェクトを表示する場合、オブジェクトの以前のバージョンとオブジェクトの現在のバージョンを比較して、異なるバージョン間の違いを表示することができます。

Designer では、フォルダおよびリポジトリのオブジェクトを比較できます。そのためには、両方のフォルダを開いておく必要があります。2 つのオブジェクトを比較すると、ダイアログボックスにその結果が表示されます。各ダイアログボックスには、オブジェクトの種類に応じて異なるタブが表示されます。

以下の図に、2 つのリレーショナルソースの比較を示します。



このダイアログボックスのそれぞれのタブには、2 つのカラムがあります。左カラムには 1 つ目の比較対象のオブジェクトの属性が表示され、右カラムには 2 つ目のオブジェクトの属性が表示されます。2 つのカラムは、境界線によって区切られています。境界線を左右に動かすと、カラムの大きさを変更することができます。Designer は、オブジェクト間で異なっている箇所を強調表示で示します。両方のオブジェクトに共通のポートまたはカラムは、同じ行に表示されます。

注: 結果に表示されるメタデータを変更することはできません。

ソース、ターゲット、およびトランスフォーメーションの比較

同じタイプの個々のオブジェクトを比較して、Designer を介するオブジェクト間の違いを特定することができます。

以下の表に、ソース、ターゲット、またはトランスフォーメーションの比較に使用できる方法を示します。

比較対象	使用するツール	次の手順
ソース	Source Analyzer	［ソース］-［比較］の順にクリックします。
ターゲット	Target Designer	［ターゲット］-［比較］の順にクリックします。
トランスフォーメーション	Transformation Developer	［トランスフォーメーション］-［比較］の順にクリックします。
マッピング内のインスタンス	Mapping Designer	オブジェクトを選択し、［トランスフォーメーション］-［比較］の順にクリックします。
マプレット内のインスタンス	Mapplet Designer	オブジェクトを選択し、［トランスフォーメーション］-［比較］の順にクリックします。

注: このダイアログボックスのいずれかのフィールドが空白の場合や比較しないオブジェクトの名前がフィールドに表示されている場合は、［参照］をクリックして比較するオブジェクトを選択します。

マッピングおよびマップレットの比較

2 つのマッピングまたは 2 つのマプレットを比較できます。例えば、異なるフォルダに同じ名前のマッピングがある場合、それらを比較することによって違いがあるかどうかを確認できます。

2 つのマッピングまたは 2 つのマプレットを比較すると、ダイアログボックスにその結果が表示されます。このダイアログボックスのタブには、以下の情報が表示されます。

- **概要情報。**［概要］タブの各マッピングまたはマップレット間の違いの概要を表示します。 この情報は、テキストファイルに保存できます。
- **一般的な情報。**［マッピング］タブまたは［マップレット］タブの名前、説明、および有効性などの各オブジェクトについての一般的な情報を比較します。 ショートカットに関する一般的な情報を比較することもできます。
- **インスタンス。**［インスタンス］タブの各マッピングまたはマップレットのソース、ターゲット、およびトランスフォーメーションインスタンスを比較します。 同じ種類のソース、ターゲット、トランスフォーメーション、およびマップレットインスタンスを比較することができます。 同じ名前を持つインスタンスは同じ行に表示されます。インスタンスを詳細に比較するには、各マッピングまたはマップレット内のインスタンスを選択し、［インスタンスの比較］をクリックします。
- **リンク。**［インスタンス］タブを使用して、入力リンクと出力リンクの違い、およびインスタンスを別のトランスフォーメーションインスタンスに接続するポートを比較します。 リンクを比較するには、最初に 2 つのマッピングまたはマップレットのインスタンスを比較します。 ［インスタンス］タブでインスタンスをクリックし、［リンクの比較］をクリックします。入力リンクを比較するには、［入力リンク］タブをクリックします。出力リンクを比較するには、［出力リンク］タブをクリックします。各リンクのポートを比較するには、リンクをクリックします。両方のインスタンスに存在するポートは、同じ行に表示されます。

- **パラメータおよび変数。**［変数］タブの各オブジェクトのパラメータおよび変数を比較します。
- **ターゲットロード順。**［ターゲットロード順］タブの各マッピングでのターゲットロード順を比較します。マップレットを比較する場合、Designer は表示されません。
- **メタデータエクステンション。**［メタデータエクステンション］タブの各マッピングまたはマップレットのメタデータエクステンションを比較します。

以下の表に、2 つのマッピングまたはマップレット内のマッピング、マップレット、またはインスタンスを比較する方法を示します。

比較対象	使用するツール	選択...
マッピング	Mapping Designer	［マッピング］-［比較］の順にクリックします。
マップレット	Mapplet Designer	［マプレット］-［比較］の順にクリックします。
2 つのマッピングのインスタンス	Mapping Designer	［マッピング］-［比較］の順にクリックします。
2 つのマプレットのインスタンス	Mapplet Designer	［マプレット］-［比較］の順にクリックします。

注: このダイアログボックスのいずれかのフィールドが空白の場合や目的のオブジェクト以外の名前がフィールドに表示されている場合は、［参照］をクリックしてオブジェクトを選択します。

第 12 章

ビジネスコンポーネントの管理

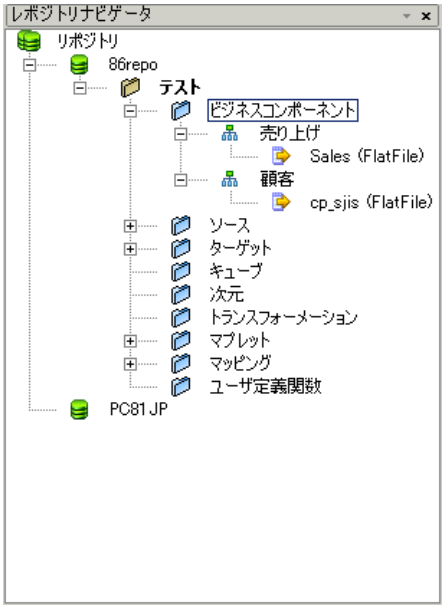
この章では、以下の項目について説明します。

ビジネスコンポーネントの管理の概要

ビジネスコンポーネントを使用して、リポジトリフォルダのソースおよびマプレットをグループ化し、表示します。Repository Manager および Designer では、ビジネスコンポーネントノードは、各リポジトリフォルダの下に表示されます。ビジネスコンポーネントツリーには任意の数のディレクトリを追加することができます。

ビジネスコンポーネントは、ソース、マップレット、またはソースやマップレットへのショートカットを参照できます。

以下の図に、ビジネスコンポーネントをグループ化するディレクトリを示します。



ビジネスコンポーネントツリー内のソースやマプレットは、繰り返し使用できます。

ビジネスコンポーネントツリーが共有フォルダにある場合、ビジネスコンポーネントツリー内のすべてのディレクトリは共有されます。

ビジネスコンポーネントのロック

Designer は、内容を編集中のビジネスコンポーネントツリーをロックします。ビジネスコンポーネントツリーがロックされている場合は、ビジネスコンポーネントをコピーできません。

ビジネスコンポーネント文書へのリンクの作成

ビジネスコンポーネントおよびディレクトリについて作成したビジネス文書に対して、リンクを作成したり編集したりすることができます。ビジネス文書は、特定のリポジトリオブジェクトまたはトランスフォーメーション式に関する詳細を提供します。

ビジネスコンポーネントおよびディレクトリの管理

Designer を使用して、以下の作業を実行します。

- ディレクトリの作成
- ビジネスコンポーネントの作成
- ディレクトリの編集
- ディレクトリまたはビジネスコンポーネントの削除
- ディレクトリまたはビジネスコンポーネントのコピー

ビジネスコンポーネントまたはディレクトリを移動するには、ビジネスコンポーネントツリー内の単一のビジネスコンポーネントまたはディレクトリ全体をドラッグします。異なるリポジトリのフォルダまたはリポジトリとの間で、単一のビジネスコンポーネントまたはディレクトリ全体をドラッグした場合、Designer により、目的の場所に元のオブジェクトのコピーを作成するように求められます。

ディレクトリの作成および編集

ディレクトリを使用して、ビジネスコンポーネントツリー内のリポジトリオブジェクトを整理できます。ツリー内の各ディレクトリには、名前に加えて識別のためのコメントを付けることもできます。ディレクトリを作成すると、ディレクトリの場所はそのプロパティの 1 つとして表示されます。ディレクトリを作成するときに別のディレクトリノードを選択することにより、別の場所にディレクトリを作成することができます。

ビジネスコンポーネントディレクトリを作成するには：

1. Designer で、リポジトリに接続してフォルダを開きます。
2. ［リポジトリ］-［ビジネスコンポーネント］-［新規ディレクトリ］を選択します。

   注: 別のディレクトリの下にビジネスコンポーネントディレクトリを作成するには、ナビゲータ内でディレクトリを選択し、［リポジトリ］-［ビジネスコンポーネント］-［新規ディレクトリ］の順に選択します。
3. ビジネスコンポーネントディレクトリの名前および説明を入力します。

ディレクトリの編集

ビジネスコンポーネントディレクトリを編集するには：

1. Designer で、リポジトリに接続してフォルダを開きます。
2. 編集したいビジネスコンポーネントディレクトリを選択します。
3. ［リポジトリ］-［ビジネスコンポーネント］-［プロパティの編集］を選択します。

ビジネスコンポーネントの作成

ビジネスコンポーネントを作成し、ディレクトリを使用してソースおよびマプレットを表示します。オブジェクトは、元の場所またはビジネスコンポーネントディレクトリから編集します。ビジネスコンポーネントツリー内のソースやマプレットは、繰り返し参照することができます。

また、ショートカットを使用するか、またはビジネスコンポーネントをコピーして、他のリポジトリフォルダまたはリポジトリのオブジェクトを持つビジネスコンポーネントを作成することもできます。

ビジネスコンポーネントを作成するには：

1. Designer で、リポジトリに接続します。
2. ビジネスコンポーネントを作成したいリポジトリフォルダを開きます。
3. このリポジトリフォルダのナビゲータで、参照先のソースまたはマプレットを選択します。
4. ソースまたはマプレットを、ビジネスコンポーネントツリー内のディレクトリにドラッグします。

コピーによるビジネスコンポーネントの作成

共有されていないビジネスコンポーネントのコピーを作成するには、リポジトリフォルダのオブジェクトをビジネスコンポーネントツリーのディレクトリにドラッグします。Designer は、ビジネスコンポーネントツリーのオブジェクトのコピーを作成するよう求めてきます。他のリポジトリフォルダからコピーされたオブジェクトは、元のオブジェクトに加えられた変更を反映しません。

ショートカットを使用したビジネスコンポーネントの作成

リポジトリ内の共有フォルダのオブジェクトのビジネスコンポーネントを作成するには、オブジェクトをビジネスコンポーネントツリーのディレクトリにドラッグします。Designer は、ローカルショートカットを作成するよう求めてきます。ローカルショートカットは、同じリポジトリ内の共有フォルダに含まれているソースまたはマプレットを参照します。

グローバルリポジトリの共有フォルダのオブジェクトのビジネスコンポーネントを作成するには、グローバルリポジトリに接続し、オブジェクトをビジネスコンポーネントツリーのディレクトリにドラッグします。Designer は、グローバルショートカットを作成するよう求めてきます。グローバルショートカットとは、グローバルリポジトリ内の共有フォルダのオブジェクトを参照し、ローカルリポジトリに作成されるショートカットです。

ディレクトリまたはビジネスコンポーネントの削除

ディレクトリまたはビジネスコンポーネントは、ビジネスコンポーネントツリーから削除できます。ディレクトリを削除すると、そのディレクトリに含まれるすべてのサブディレクトリおよびビジネスコンポーネントが削除されます。ディレクトリまたはビジネスコンポーネントを削除しても、元のリポジトリオブジェクトは削除されません。

ビジネスコンポーネントディレクトリを削除するには：

1. Designer で、リポジトリに接続してフォルダを開きます。
2. ビジネスコンポーネントディレクトリを選択し、［編集］-［削除］を選択します。

   ヒント: ナビゲータ内でディレクトリを選択した状態で Delete キーを押すことによって、ディレクトリを削除することができます。

ビジネスコンポーネントの削除

ビジネスコンポーネントを削除するには：

1. Designer で、リポジトリに接続してフォルダを開きます。

2. ビジネスコンポーネントを選択し、［編集］-［削除］を選択します。

   ヒント: ナビゲータ内でビジネスコンポーネントを選択した状態で Delete キーを押すことによって、ビジネスコンポーネントを削除することができます。

ディレクトリまたはビジネスコンポーネントのコピー

ビジネスコンポーネントディレクトリは、同じツリーの別の場所にコピーできます。また、ビジネスコンポーネントディレクトリを別のリポジトリフォルダまたはリポジトリにコピーすることもできます。他のリポジトリフォルダからコピーされたオブジェクトは、元のオブジェクトに加えられた変更を反映しません。

ディレクトリまたはビジネスコンポーネントをコピーするには：

1. Designer で、コピー元のリポジトリを含むデータベースに接続し、ディレクトリまたはビジネスコンポーネントのコピー元となるフォルダを開きます。
2. ナビゲータで、コピーしたいオブジェクトを選択します。
3. ［編集］-［コピー］をクリックするか、、Ctrl+C キーを押します。
4. オブジェクトを別のリポジトリにコピーしたい場合は、コピー先リポジトリに接続します。
5. オブジェクトを別のフォルダにコピーしたい場合は、コピー先フォルダを開きます。
6. ナビゲータで、ディレクトリまたはビジネスコンポーネントを貼り付けるビジネスコンポーネントディレクトリの位置を選択します。
7. ［編集］-［ペースト］をクリックするか、Ctrl+V キーを押します。

   ヒント: リポジトリに接続し、Ctrl キーを押しながら、コピー先のフォルダのビジネスコンポートノードにディレクトリまたはビジネスコンポーネントをドラッグして、ディレクトリまたはビジネスコンポーネントをコピーできます。

第 13 章

キューブと次元の作成

この章では、以下の項目について説明します。

キューブと次元の作成の概要

Target Designer には、キューブと次元を作成および編集するためのインタフェースが用意されています。多次元メタデータとは、オンライン分析処理（OLAP）アプリケーションで分析に使用される論理的に構成された一群のデータです。この論理構成は、一般に OLAP アプリケーションのエンドユーザーによるデータの表示およびアクセスが最も効率的になるように作成されています。以下の節では、PowerCenter の多次元機能に関連する概念について説明します。

多次元メタデータについて

多次元モデルはデータウェアハウスのデザインにとってきわめて重要なものです。次元モデルのデザインが適切であれば、大量のデータを取り扱うことができます。そもそも次元モデルは、製品や地域といったいくつかの単純な次元を使用して事業データを検討する小売業界で役立てるために作成されたものでした。この次元モデルを構成するのは、中心となる大規模なファクトテーブルと、いくつかの小規模な次元テーブルです。ファクトテーブルには、総売上高や販売品目といった計測可能な事実（ファクト）としてのデータが含まれています。そして、業界のさまざまな事業分野に関連する各種の属性が、互いに素である次元として表されます。中心となるファクトテーブルは、スキーマ内で複数の結合によって各次元テーブルに接続されている唯一のテーブルです。また、各次元テーブルには単一の結合があって、中心のファクトテーブルに接続されています。

論理スキーマの冗長度に応じて、さまざまな多次元モデルがあります。冗長度を増やせば、データアクセスの効率は向上しますが、論理スキーマの表現における正規性は低下します。最も一般的に使用される多次元スキーマは、スタースキーマと呼ばれるものです。スタースキーマは正規化された多次元モデルであって、互いに素であるそれぞれの次元が単一のテーブルで表されます。

別の正規化多次元モデルには、スノーフレークスキーマがあります。スノーフレークスキーマは論理的にはスタースキーマに類似していますが、階層的に関連し合う複数のテーブルで表される次元が少なくとも 1 つ含まれる、という点が異なります。製品次元が複数のテーブルで表現されると、上記のスタースキーマはスノーフレークスキーマとなります。例えば、主要製品属性、ブランド属性、および特定のブランド属性について、それぞれ 1 つの次元テーブルを追加できます。

非正規化多次元モデルは、次元に関連したテーブル内に重複属性を有しています。これにより、次元のテーブル間に複数の結合を実行せずに、次元のさまざまな属性を素早く取得することができます。

多次元メタデータのキー要素

以下の表に、多次元メタデータの要素を示します。

用語	定義
集計	特定のビジネスルールに基づいてデータを要約したり詳細データをグループ化したりする操作を、事前に格納したもの。ルールとしては、合計、最小値、計数、およびこれらの組み合わせなどがあります。
レベル	次元の特定のプロパティ。例えば、サイズ、種別、色などがあります。
キューブ	特定の次元分析問題について関連する実測値、集計、次元の集まり。例えば、ある地域での製品売上高などがあります。
次元	ビジネスの特定局面を記述したレベルプロパティの集まり。当該次元を使用した 1 つまたは複数のキューブの実測値を分析するために使用します。例えば、地域、時間、顧客、製品などがあります。
ドリリング	ドリリングとは、キューブの内容を詳細に見ていくという意味で使用される用語です。一般に、この操作を行う目的は、要約レベルの情報にアクセスしたり、階層中のある次元について詳細なプロパティを得ることにあります。
ファクト	ファクトとは、キューブの数量データの経時測定値です。例えば、販売数量、販売額、総利潤などがあります。
階層	階層とは、キューブの特定の次元でデータが表す粒度のレベルを表します。例えば、州、郡、地域、および市は、地理次元の階層においてそれぞれ異なる粒度を表しています。
測定	ファクトや集計値の数量データを表す手段です。例えば、年間の総売上高や販売数量があります。
正規化	さまざまな階層の関連次元テーブル中の冗長度を減少させたり、変則事項を排除するためのプロセスです。
冗長度	クエリーの処理速度を改善するために関連テーブル中のデータを重複させる度合いを示します。
スタースキーマ	正規化多次元モデルの 1 つであり、互いに素である次元をそれぞれ単一のテーブルで表します。
スノーフレークスキーマ	正規化多次元モデルの 1 つであり、少なくとも 1 つの次元が、階層的に関連し合う複数のテーブルで表されます。

次元の作成

キューブを作成する前に、次元を作成する必要があります。次元を作成するには、以下の手順に従ってください。

1. 次元の説明を入力します。
2. この次元にレベルを追加します。
3. 次元に階層を追加します。
4. 階層にレベルインスタンスを追加します。

手順 1. 次元の作成

次元を作成するには：

1. Target Designer で、［ターゲット］-［次元の作成/編集］の順に選択します。

   次元エディタが表示されます。
2. ［次元の追加］をクリックします。
3. 以下の情報を入力します。
   - **名前。**次元名は、フォルダ内で一意である必要があります。
   - **説明。**次元の説明を入力します。 この説明は、Repository Manager に表示されます。
   - **データベースタイプ。**次元のデータベースタイプはキューブのデータベースタイプと一致する必要があります。

   注: 次元をいったん作成したら、データベースタイプは変更できません。
4. ［OK］をクリックします。

手順 2 次元へのレベルの追加

次元を作成した後で、必要なレベルを追加します。レベルは、ターゲットテーブルの作成に必要なプロパティを保持するものです。

レベルを次元に追加するには：

1. 次元エディタで、［レベル］を選択し、［レベルの追加］をクリックします。
2. レベルの名前と説明を入力します。

   レベル名は次元内で一意になるようにしてください。
3. ［レベルのプロパティ］を選択します。
4. ［ソースフィールドからインポート］をクリックします。

   レベルプロパティ名は次元内で一意になるようにしてください。
5. カラムをレベルにコピーしたいソーステーブルを選択します。

   カラムが［ソースフィールド］リストに表示されます。
6. レベルに追加するカラムを選択します。
7. ［カラムのコピー］をクリックしてソースカラムをレベルに追加します。
8. ［カラムの追加］をクリックして新しいカラムをレベルに追加します。
9. カラムをすべて追加したら、［OK］をクリックします。

   次元エディタに新しいレベルが表示されます。

手順 3. 次元への階層の追加

階層を次元に追加するには：

1. 次元エディタで、[階層] を選択します。
2. [階層の追加] をクリックします。
3. 階層の名前と説明を入力し、[正規化] または [非正規化] を選択します。

   正規化キューブでは、冗長データが制限されます。非正規化キューブでは冗長データが存在するため、データの取得速度が増加します。

手順 4. 階層へのレベルの追加

階層を作成したら、この階層にレベルを追加します。階層のルートレベルは 1 つに限ります。

レベルを階層に追加するには：

1. 次元エディタからドリルダウンして、次元のレベルを表示します。
2. 階層でルートレベルとして定義したいレベルをドラッグします。

   ルートレベルとは、最も細かい粒度を持つレベルです。
3. ターゲットテーブルの名前と説明を入力します。
4. [OK] をクリックします。

   ウィンドウが表示されて、新しいレベルの影響を受けるオブジェクトがすべて一覧表示されます。
5. [OK] をクリックします。

   階層の下に新しいレベルが表示されます。

キューブの作成

キューブを作成するには：

1. Target Designer で、[ターゲット] - [キューブの作成] の順に選択します。
2. 以下の情報を入力します。
   - **キューブ名。** キューブ名は、フォルダ内で一意である必要があります。
   - **キューブタイプ。** [正規化] または [非正規化] を選択します。 正規化次元では、正規化キューブとする必要があります。同様に、非正規化次元では非正規化キューブとする必要があります。
   - **データベースタイプ。** キューブのデータベースタイプは、キューブでの次元のデータベースタイプと一致する必要があります。
3. [次へ] をクリックします。
4. キューブに含める次元と階層を指定します。
5. [次へ] をクリックします。
6. メジャーをキューブに追加します。

   ソーステーブルからカラムをコピーしたり、新しいカラムを追加したりすることができます。

   メジャー名はファクト内で一意になるようにしてください。レベル名は各キューブ内で一意になるようにしてください。
7. ファクトテーブルに名前を追加します。

8. ［終了］をクリックします。

   Designer はキューブとファクトテーブルをワークスペースに追加します。

キューブの編集

Target Designer でキューブの編集ができます。ファクトテーブルや次元テーブルを直接編集することはできません。ファクトテーブルや次元テーブルのカラムを編集するには、キューブまたは次元を編集する必要があります。

キューブを編集するには：

1. ［ターゲット］-［キューブの編集］を選択します。
2. 次元の設定項目は、以下を除いて変更することができます。
   - データベースタイプ
   - 次元のタイプ（正規化または非正規化）
3. ［閉じる］をクリックします。

次元の編集

Target Designer で次元の編集ができます。ただし、次元をいったん作成したら、データベースタイプは変更できません。

次元を編集すると、Designer はその次元を使用するすべてのマッピングを無効にします。

次元を編集するには：

1. ［ターゲット］-［次元を作成または編集します］を選択します。
2. また、データベースタイプまたは次元タイプ以外の次元の設定を変更することもできます。
3. ［閉じる］をクリックします。

キューブまたは次元の削除

Designer のナビゲータからキューブまたは次元の削除ができます。ターゲットテーブルとは異なり、Target Designer ではキューブや次元を削除できません。

キューブを削除すると、このキューブに関連付けられたファクトテーブルがすべて削除されます。次元を削除すると、次元テーブルおよびこの次元に対する参照がすべて削除されます。

キューブまたは次元を削除するには：

1. Designer で、リポジトリを開きます。
2. ナビゲータで、削除するキューブまたは次元を選択します。
3. ［削除］をクリックします。

   メッセージが表示され、キューブまたは次元を削除してよいかどうか聞いてきます。

4. ［OK］をクリックし、キューブまたは次元を削除します。

キューブを開く/閉じる

Target Designer でキューブを開くことができます。

キューブを開くには：

1. リポジトリを開き、フォルダを選択します。
2. Target Designer を開きます。
3. キューブを選択し、ワークスペースまでドラッグします。

   メッセージが表示され、ワークスペースをクリアしてよいかどうか聞いてきます。
4. ［OK］をクリックし、キューブを開きます。

   Designer はワークスペースをクリアし、そのキューブに関連付けられたファクトテーブルと次元テーブルをすべて表示します。

Target Designer でキューブを閉じるには、［ターゲット］-［キューブを閉じる］の順に選択します。
Designer はキューブを閉じ、すべてのテーブルのレイアウトを保存します。

キューブと次元のメタデータの表示

Repository Manager でキューブと次元のメタデータの表示ができます。

キューブまたは次元のメタデータを表示するには：

1. Repository Manager で、フォルダを開きます。
2. 分析対象のキューブまたは次元までドリルダウンします。

   Repository Manager に各オブジェクトのメタデータが表示されます。

キューブと次元に関するヒント

キューブと次元を取り扱う際には、以下を念頭に置いてください。

- キューブをコピーするには、そのキューブを格納したフォルダをコピーする必要があります。
- キューブまたは次元のレベルを表示するには、このキューブまたは次元を編集するか、Repository Manager のナビゲータを利用します。
- キューブや次元を前バージョンに戻すことはできません。
- ナビゲータからキューブや次元の削除ができます。
- ［ターゲット］-［次元を作成または編集します］を使用して、次元を削除できます。
- Target Designer のワークスペースからは、キューブや次元を削除できません。
- ファクトテーブルや次元テーブルのカラムを変更するにあたっては、キューブまたは次元を編集する必要があります。ファクトテーブルや次元テーブルを直接編集することはできません。

- レベルを削除すると、Designer は階層内の関連レベルインスタンスを削除します。Designer はまた、関連キューブからもレベルインスタンスを削除します。
- 各ファクトテーブルや次元テーブルについて、プライマリキーが生成されます。フォーマットは、GK_*テーブル名*です。
- 次元レベルインスタンスをファクトテーブルに関連付けると、該当ファクトテーブルに外部キーが追加されます。
- キューブをワークスペースにドラッグしてから、リンクをダブルクリックしてキューブとカーディナリティを編集できます。
- ワークスペースで、ファクトテーブルと次元テーブルの間のリンクを削除したり作成したりすることはできません。リンクの削除と作成は、キューブエディタと次元エディタを使ってのみ行えます。階層内ではグラフを作成できます。

第 14 章

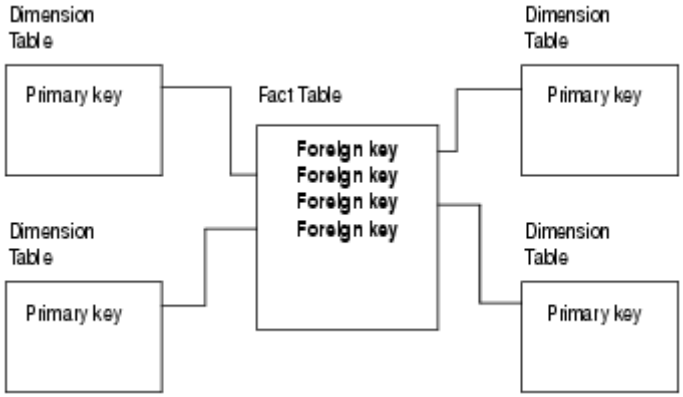
マッピングウィザードの使用

この章では、以下の項目について説明します。

スタースキーマの保持

スタースキーマをデザインする場合、購買や取引などの時系列収集情報についてファクトテーブルを作成します。その後、在庫や出荷方法などの各関連情報リストについて別々の次元テーブルを作成します。それぞれの次元テーブルには、論理的なプライマリキーや生成された複合キーがあって、次元データへのアクセスを可能としています。例えば、在庫部品番号をプライマリキーとして使用したり、部品番号やカレントフラグを用いて複合キーを作成したりできます。

以下の図に、4 つの次元テーブルと 1 つのファクトテーブルを含むスタースキーマを示します。



スタースキーマを実装する場合、ファクトテーブルおよび次元テーブルへの更新の取り扱い方法を決定します。ファクトテーブルは、情報を新たに収集することによって定期的に変化します。ファクトテーブルの既存データをすべて維持するのか、最新のバージョンまたはスナップショットのみ必要なのかを検討する必要があります。

過去のファクト情報が必要でない場合には、ワークフロー内の新しいセッションを使用する前に既存のファクトテーブルを削除したり切り詰めたりすることができます。ファクトテーブルに過去の情報を履歴として維持する場合には、一般に既存のテーブルに最新のスナップショットを追加し、ロードの日付やセッション番号などのフラグを用いて最新のスナップショットを特定します。

次元テーブルは一般には静的なリストですが、大半の次元テーブルは経時的に変化します。例えば、月に一度在庫次元を更新して、新しい部品番号や変化した部品番号を反映させることが必要な場合があります。こうした変更はファクトテーブルの変更に比べれば影響が少ないため、この次元は「緩やかに成長する」または「緩やかに変化する」次元と呼びます。

緩やかに成長する次元テーブルには、少しずつ増大する次元データが含まれ、既存の次元への更新はありません。このような次元は、新しいデータを既存のテーブルに追加することによって整備を行います。

緩やかに変化する次元テーブルには、少しずつ増大する次元データおよび既存の次元への更新が含まれます。既存の次元を更新する場合には、過去の次元データをすべて維持するのか、まったく維持しないのか、あるいは現在と変更前のバージョンの次元データのみ維持するのかを決定します。

「緩やかに成長する」または「緩やかに変化する」次元テーブルで過去の情報が必要でない場合には、ワークフロー内の新しいセッションを使用する前に既存のテーブルを削除するか切り詰めることができます。ただし、次元を新たに挿入したり既存の次元を更新したりする方が、テーブル全体の再ロードより効果的である場合もあります。

次元テーブルで過去の情報が必要な場合には、ターゲットでの現在のデータと過去のデータの区別の仕方を決定します。

- 過去のデータをすべて維持する場合は、以下の方法により新しいデータのバージョン管理を行います。
  - バージョン番号の作成とプライマリキーのバージョン管理。
  - カレントバージョンフラグを使用した複合キーの作成。
  - 有効期間の作成。
- 過去のデータを部分的に維持する場合は、カレントバージョンと前バージョンを 1 つの行に格納します。また、タイムスタンプを記入して、最新の更新を明示することができます。

マッピングウィザードについて

Designer は 2 種類のマッピングウィザードがあり、これによりマッピングの迅速かつ容易な作成が可能です。この 2 つのウィザードの目的は、中核のファクトテーブルに関連する一連の次元であるスタースキーマにデータをロードして整備するためのマッピングを作成することにあります。ただし、これらのウィザードで作成したマッピングを使用して、他の種類のターゲットにデータをロードすることもできます。

ロード対象のターゲットの種類およびこのターゲットでの過去のデータの取り扱い方法に基づいて、使用するウィザードを選択し、各ウィザードのオプションを選択します。

- **基本操作ウィザード。**静的なファクトテーブルと次元テーブル、および緩やかに成長する次元テーブルをロードするためのマッピングを作成します。
- **緩やかに変化する次元ウィザード。**保存する過去の次元データの量、および過去の次元データの処理に選択する方法に基づいて、緩やかに変化する次元テーブルにデータをロードするためのマッピングを作成します。

マッピングウィザードを使用した後、生成されたマッピングを編集して、更にカスタマイズすることができます。

基本操作ウィザードの使用

基本操作ウィザードは、静的なファクトテーブルと次元テーブル、および緩やかに成長する次元テーブルをロードするためのマッピングを作成します。

基本操作ウィザードでは 2 種類のマッピングが作成できます。

- **パススルー。**行をすべて挿入することにより、静的なファクトまたは次元テーブルをロードします。 新規データをロードする前にテーブルから既存データをすべて削除する場合に、このマッピングを使用します。
- **緩やかに成長するターゲット。**新しい行を挿入することにより、緩やかに成長するファクトルまたは次元テーブルをロードします。 既存のデータを更新する必要がない場合に、このマッピングを使用して新規データをロードします。

以下の表に、基本操作マッピングのタイプを示します。

基本操作マッピングのタイプ	ターゲットテーブルタイプ	履歴	データの処理
単純	静的なファクトまたは次元	なし	ソース行をすべて挿入します。セッションプロパティのターゲットテーブルを切り詰めるオプションまたはセッション実行前に実行されるシェルコマンドを用いて、それぞれのセッションの実行前にターゲットを削除したり切り詰めることができます。
緩やかに成長するターゲット	緩やかに成長するファクトまたは次元	完全	新しい行にフラグを設定して、既存のターゲットに挿入します。

緩やかに変化する次元ウィザードの使用

緩やかに変化する次元ウィザードでは、緩やかに変化する次元テーブルをロードするため、以下のマッピングが作成されます。

- **タイプ 1 の次元マッピング。**新しい次元を挿入し、既存の次元を上書きすることにより、緩やかに変化する次元のテーブルをロードします。 以前の次元データの履歴が必要ない場合には、このマッピングを使用します。

- **タイプ 2 の次元/バージョンデータのマッピング。**新しい次元および変更された次元を挿入することにより、緩やかに変化する次元テーブルをロードし、バージョン番号と増加したプライマリキーを使用して変更を追跡します。 次元データの全履歴を維持し、変更の軌跡を追跡する場合には、このマッピングを使用します。
- **タイプ 2 の次元/フラグカレントのマッピング。**新しい次元および変更された次元を挿入することにより、緩やかに変化する次元テーブルをロードし、フラグを使用してカレント次元データと増加したプライマリキーにマークを付け、変更を追跡します。 次元データの全履歴を維持し、現在の次元のみにフラグを設定する際に変更の軌跡を追跡する場合には、このマッピングを使用します。
- **タイプ 2 の次元/有効な日付範囲のマッピング。**新しい次元および変更された次元を挿入することにより、緩やかに変化する次元テーブルをロードし、日付の範囲を使用して、カレント次元データを定義します。次元データの全履歴を維持し、有効な日付範囲で変更を追跡する場合には、このマッピングを使用します。
- **タイプ 3 の次元マッピング。**新しい次元を挿入して既存の次元の値を更新することにより、緩やかに変化する次元テーブルをロードします。 次元テーブル内の現在の次元値および以前の次元値を維持する場合には、このマッピングを使用します。

以下の表に、緩やかに変化する次元を使ったマッピングのタイプを示します。

緩やかに変化する次元を使ったマッピング	ターゲットテーブル	履歴	データの処理
タイプ 1 の次元	緩やかに変化する次元	なし	新しい次元を挿入します。既存の次元を変更された次元で上書きします。
タイプ 2 の次元/バージョンデータ	緩やかに変化する次元	完全	新しい次元と変更された次元を挿入します。バージョン番号を作成し、プライマリキーをインクリメントすることにより、変更を追跡します。
タイプ 2 の次元/フラグ	緩やかに変化する次元	完全	新しい次元と変更された次元を挿入します。カレントバージョンにフラグを設定し、プライマリキーをインクリメントすることにより、変更を追跡します。
タイプ 2 の次元/有効日付	緩やかに変化する次元	完全	新しい次元と変更された次元を挿入します。有効期間を作成して、変更を追跡します。
タイプ 3 の次元	緩やかに変化する次元	部分的に維持	新しい次元を挿入します。既存の次元内の変更された値を更新します。オプションとしてロードの日付を使用して、変更を追跡します。

マッピングソースの選択

マッピングウィザードでは、以下のソースを使用します。

- フラットファイル
- リレーショナル
- アプリケーション
- フラットファイル、リレーショナル、アプリケーションソースへのショートカット

マッピングウィザードでは COBOL または XML ソースは使用できません。

マッピングのソースを選択すると、マッピングウィザードは利用可能なすべてのソースをソース名で表示します。Designer では、ソース名やターゲット名の代わりに、ビジネス名でソース定義やターゲット定義を表示す

ることができます。このオプションを選択した場合でも、マッピングウィザードではソース名でソースが表示されます。

マッピングウィザードでは、カラム名に SQL キーワードを使用している場合にはソースをインポートできません。マッピングウィザードは、カラム名として使用されている SQL キーワードを検出すると、別のソースを選択するように求めます。SQL キーワードを使用したカラムを含むソースを使用したい場合は、Mapping Designer でマッピングを作成してください。

さらに、マッピングウィザードでは、選択したソースに基づき以下を実行します。

- ソース名を使用して、ソース修飾子に名前を付けます。ファイルまたはリレーショナルソースの場合は SQ_*ソース名*となり、アプリケーションソースの場合は ASQ_*ソース名*となります。
- ソースのカラム名に基づいてポート名を作成します。

パススルーマッピングの作成

単純マッピングでは、すべてのソース行が挿入されます。ターゲットテーブルに過去のデータを維持する必要のない場合には、単純マッピングを用いてテーブルにデータをロードします。ソース行がターゲットに既に存在している場合には、既存のターゲットの切り詰めや削除を行ってからワークフローを実行します。単純マッピングでは、行はすべて現在のものです。

単純マッピングを用いてファクトテーブルや次元テーブルにデータをロードするのは、テーブルが一定の期間静的状態を維持したあとで大きく変化するような場合です。

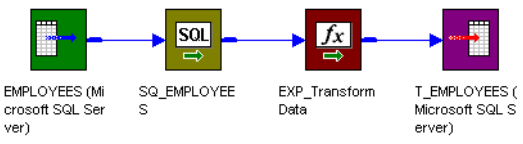
例えば、業者の次元テーブルが 1 年間同じままであるとします。年の終わりに、このテーブルのデータを再ロードして新たな業者契約や契約情報の反映を行います。この情報が著しく変化しており、過去の情報を保持しない場合、既存の次元テーブルを削除し、パススルーマッピングを使用してテーブル全体の再ロードを行うことができます。 情報が少しずつしか変化しない場合には、緩やかに変化する次元ウィザードで作成したタイプ 1 の次元マッピングを使用して、既存テーブルの更新を行うことができます。

マッピングについて

単純マッピングでは、以下のことを行います。

- すべてのソース行を選択します。
- すべての行をターゲットに挿入します。

以下の図に、パススルーマッピングを作成した場合に、基本操作ウィザードにより作成されるマッピングを示します。



1 つのデータフローがソース定義からソース修飾子および式トランスフォーメーションを経て、ターゲットに至ります。デフォルトでは、式トランスフォーメーションは変更を施すことなく、データを直接ターゲットに渡します。

トランスフォーメーションについて

以下の表に、パススルーマッピングにおける各トランスフォーメーションの機能を示します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
SQ_*SourceName*	ソース修飾子またはアプリケーションソース修飾子	マッピングウィザードで選択したソースから行をすべて選択します。
EXP_TransformData	式	すべてのソースデータを、変更を施すことなくターゲットに渡します。
T_*TargetName*	ターゲット定義	ソースデータをターゲットに挿入するためのターゲット定義です。

パススルーマッピングの作成手順

単純マッピングを作成するには：

1. Mapping Designer で、［マッピング］-［ウィザード］-［基本操作ウィザード］をクリックします。
2. マッピング名を入力し、［単純ウィザード］を選択して、［次へ］をクリックします。

   マッピング名を付けるにあたっては、「m_*マッピング名*」の形にしてください。
3. マッピングで使用するソース定義を選択します。

   ［ソーステーブルの選択］リストに利用可能なソース定義がすべて表示されます。このリストには、ショートカット、フラットファイル、リレーショナル、およびアプリケーションソースが含まれます。
4. マッピングのターゲットテーブルの名前を入力し、［完了］をクリックします。

   新たなマッピングがワークスペースに表示されます。ターゲット定義名を付けるにあたっては、「T_*ターゲット名*」の形にしてください。

   マッピングに対し、必要な編集を行います。

ワークフローを実行する前に、ターゲットデータベースでターゲットテーブルを作成します。

マッピングのカスタマイズ

ウィザードでマッピングを作成したあとで、式トランスフォーメーション EXP_TransformData を設定することができます。また、他のトランスフォーメーションを追加して、マッピングのカスタマイズが行えます。

データを、他のトランスフォーメーションを使用しないで、直接ソースからターゲットに渡す場合は、式トランスフォーメーションを削除してください。ソース修飾子を直接ターゲットへと接続することにより、単純マッピングのパフォーマンスを最適化することができます。

パススルーセッションの設定

単純マッピングでは、ターゲットにソース行が挿入されます。行が重複してプライマリキーエラーが発生しないように、ターゲットテーブルの削除または切り詰めを行ってからワークフローを実行してください。このタスクを自動的に実行するには、セッションプロパティでターゲットテーブルを切り詰めるオプション、またはセッション実行前のシェルコマンドを使用します。

緩やかに成長するターゲットのマッピングの作成

緩やかに成長するターゲットを使ったマッピングは、ユーザーが定義する比較に基づいてソース行のフィルタリングを行い、*新しい*と判明したソース行だけをターゲットに挿入します。緩やかに成長するターゲットを使ったマッピングを用いて、どのソース行が新しいかを決定し、新しいソース行を既存のターゲットテーブルにロードします。緩やかに成長するターゲットを使ったマッピングでは、行はすべて現在のものです。

緩やかに成長するファクトテーブルまたは次元テーブルで既存のデータの更新を必要としない場合には、緩やかに成長するターゲットを使ったマッピングを使用してデータをロードします。

例えば、新店舗を開店した後でのみ更新する店名および対応するサイトコードのみが格納されている次元テーブルがあるとします。テーブルに記載されている店舗には閉店するものがあるかもしれませんが、過去のデータの分析のために店コードと店名を次元テーブルに維持することにします。緩やかに成長するターゲットを使ったマッピングを使用すれば、過去のサイトのデータを削除せずに新しいソース行をサイトコード次元テーブルにロードできます。

キーの処理

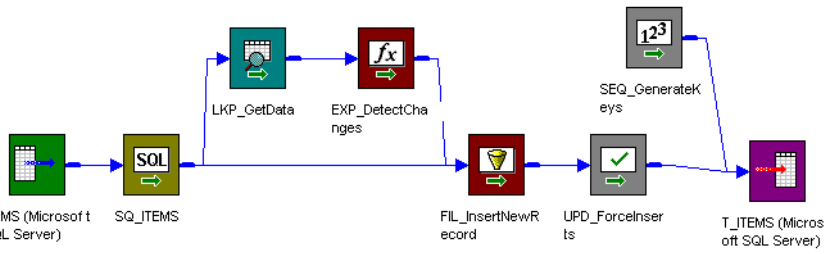
緩やかに成長するターゲットのマッピングを作成すると、Designer により、マッピングのターゲットに PM_PRIMARYKEY という追加カラムが作成されます。このカラムでは、Integration Service により、ターゲットに書き込まれる各行のプライマリキーが作成され、新しいキー値が 1 ずつ増加されます。

マッピングについて

緩やかに成長するターゲットを使ったマッピングは以下のことを行います。

- すべての行を選択します。
- 既存ターゲットをルックアップテーブルとしてキャッシュに格納します。
- ソース中の論理キーカラムとターゲットルックアップテーブル中の対応カラムとを比較します。
- フィルタリングにより既存の行を除外します。
- 新しい行のプライマリキーを作成します。
- ターゲットに新しい行を挿入します。

以下の図に、緩やかに成長するターゲットのマッピングを作成した場合に、基本操作ウィザードにより作成されるマッピングを示します。



緩やかに成長するターゲットを使ったマッピングでは、ルックアップおよび式トランスフォーメーションを用いて、ソースデータと既存のターゲットデータとを比較します。基本操作ウィザードによる作業中に、ユーザーは既存のターゲットとの比較に使用するソースの論理キーカラムを入力します。式トランスフォーメーションがターゲット中にキーカラムが一致しないソース行を検出すると、その行に「新規」のフラグを設定します。

フィルタトランスフォーメーションは、新規の行だけをアップデートストラテジトランスフォーメーションに渡します。アップデートストラテジトランスフォーメーションは新規の行に挿入のフラグを設定し、またシーケンスジェネレータはターゲットに書き込んだ各行についてプライマリキー値を新たに作成します。

トランスフォーメーションについて

以下の表に、緩やかに成長するターゲットのマッピングにおける各トランスフォーメーションの機能を示します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
SQ_*SourceName*	ソース修飾子またはアプリケーションソース修飾子	マッピングウィザードで選択したソースから行をすべて選択します。
LKP_GetData	ルックアップ	既存のターゲットテーブルをキャッシュに格納します。 ソースの論理キーカラムとターゲットの対応キーカラムとを比較します。
EXP_DetectChanges	式	以下の式を用いて、ターゲットに一致キーのないソース行にフラグを設定して、新しい行であることを明示します。 `IIF(ISNULL(PM_PRIMARYKEY),TRUE,FALSE)` NewFlag フィールドに結果を書き込みます。 行をすべて FIL_InsertNewRecord に渡します。
FIL_InsertNewRecord	フィルタ	フィルタ条件 `NewFlag` に従い、EXP_DetectChanges の結果から新規（TRUE）以外の行を除外します。新しい行を UPD_ForceInserts に渡します。
UPD_ForceInserts	アップデートストラテジ	`DD_INSERT` を使用して行をターゲットに挿入します。
SEQ_GenerateKeys	シーケンスジェネレータ	ターゲットに書き込んだ各新規行について値を生成し、各値を 1 だけインクリメントします。 値をターゲットに渡し、PM_PRIMARYKEY カラムに書き込みます。
T_*TargetName*	ターゲット定義	ターゲットに挿入する新しい行に対するターゲット定義のインスタンスです。

緩やかに成長するターゲットのマッピングの作成手順

緩やかに成長するターゲットを使ったマッピングを作成するには：

1. Mapping Designer で、［マッピング］-［ウィザード］-［基本操作ウィザード］をクリックします。
2. マッピング名を入力し、［緩やかに成長するターゲット］を選択してから、［次へ］をクリックします。

   マッピング名を付けるにあたっては、「m_*マッピング名*」の形にしてください。
3. マッピングで使用するソース定義を選択します。

   ［ソーステーブルの選択］リストに利用可能なソース定義がすべて表示されます。このリストには、ショートカット、フラットファイル、リレーショナル、およびアプリケーションソースが含まれます。
4. マッピングのターゲットテーブルの名前を入力します。［次へ］をクリックします。

   ターゲット定義名を付けるにあたっては、「T_*ターゲット名*」の形にしてください。
5. Integration Service でターゲットテーブルのデータをルックアップするときに使用する［ターゲットテーブルのフィールド］リストからカラムを、1 つまたは複数選択します。 ［追加］をクリックします。

ウィザードは選択したカラムを［論理キーフィールド］リストに追加します。

ヒント: 選択するカラムはソースのキーカラムである必要があります。

セッションを含むワークフローの実行時に、Integration Service により既存のターゲットデータのルックアップが実行されます。［論理キーフィールド］カラムが対応するターゲットカラムに一致する場合、Integration Service によりターゲットデータが返されます。

［論理キーフィールド］からカラムを削除するには、カラムを選択して［削除］をクリックします。

注: FILLER という名前を使ってポートを［論理キーフィールド］リストに追加することはできません。

6. ［終了］をクリックします。

新たなマッピングがワークスペースに表示されます。マッピングに対し、必要な編集を行います。

注: 緩やかに成長するターゲットを使ったマッピングでは、［変化を比較したいフィールド］フィールドは使用できません。

緩やかに成長するターゲットのセッションの設定

緩やかに成長するターゲットを使ったマッピングは、新しいソース行にフラグを設定してから、これを新しいプライマリキーと共にターゲットに挿入します。マッピングではアップデートストラテジトランスフォーメーションを用いて、新しい行が挿入すべき行であることを明示します。したがって、マッピングのセッションを作成する場合には、セッションを以下のように設定します。

1. セッションプロパティで、［プロパティ］タブの［全般］設定をクリックします。［ソース行の扱い］を［データドリブン］に設定します。
2. セッションプロパティで、［マッピング］タブの［ターゲットプロパティ］設定をクリックします。Integration Service により適切に行がターゲットにロードされることを確認するには、各リレーショナルターゲットに対して［挿入］を選択します。

タイプ 1 の次元マッピングの作成

タイプ 1 の次元を使ったマッピングは、ユーザーが定義する比較に基づいてソース行のフィルタリングを行い、新しい次元であると判明したソース行だけをターゲットに挿入します。既存の次元への変更を含む行は、ターゲット内で既存の次元を上書きすることにより更新されます。タイプ 1 の次元を使ったマッピングでは、すべての行にカレント次元データが含まれています。

緩やかに変化する次元テーブルを更新する場合に、テーブル中の次元の前バージョンを保持する必要がなければ、タイプ 1 の次元を使ったマッピングを使用します。

例えば、新店舗を開店した後で更新する店コード、場所、および間接費を含むサイト次元テーブルがあるとします。売上高や間接費の算定では、この次元を使用するとします。同じ店舗の以前の住所や前年の間接費を知る必要はありませんから、テーブル中に以前の次元データは必要ありません。タイプ 1 の次元を使ったマッピングを使用すれば、過去の記録のない現在のデータが確認できます。

キーの処理

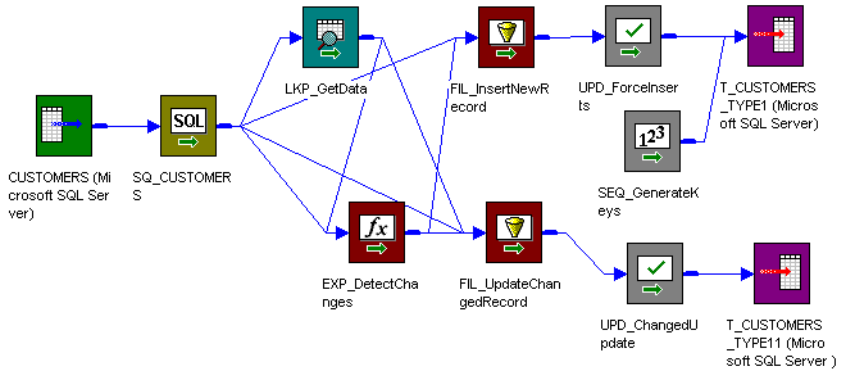
タイプ 1 の次元のオプションを使用すると、Designer により、マッピングのターゲットに PM_PRIMARYKEY という追加カラムが作成されます。このカラムでは、Integration Service により、ターゲットに書き込まれる各行のプライマリキーが作成され、新しいキー値が 1 ずつ増加されます。

マッピングについて

タイプ 1 の次元を使ったマッピングでは、以下のことを行います。

- すべての行を選択します。
- 既存ターゲットをルックアップテーブルとしてキャッシュに格納します。
- ソース中の論理キーカラムとターゲットルックアップテーブル中の対応カラムとを比較します。
- キーカラムが一致した場合には、ソースカラムと対応ターゲットカラムとを比較します。
- 新しい行と変更された行にフラグを設定します。
- 2 つのデータフローを作成します。1 つは新しい行のデータフロー、もう 1 つは変更された行のデータフローです。
- 新しい行のプライマリキーを作成します。
- ターゲットに新しい行を挿入します。
- 既存の行を上書きすることにより、ターゲット中の変更された行を更新します。

以下の図に、タイプ 1 の次元のオプションを選択した場合に、緩やかに変化する次元のウィザードにより作成されるマッピングを示します。



タイプ 1 の次元マッピングでは、ルックアップトランスフォーメーションおよび式トランスフォーメーションが使用され、ソースデータと既存のターゲットデータが比較されます。緩やかに変化する次元のウィザードの手順を行う際に、Integration Service を既存のターゲットデータと比較するルックアップ条件（ソースキーカラム）とソースカラムを入力します。

ターゲット中に一致するプライマリキーのないソース行では、式トランスフォーメーションが行に新規というフラグを設定します。ターゲット中に一致するプライマリキーのあるソース行では、式はユーザー定義のソースカラムとターゲットカラムを比較します。これらのカラムが一致しない場合には、式は行に変更があったというフラグを設定します。その後、このマッピングは 2 つの別のデータフローに分かれます。

第一のデータフローはフィルタトランスフォーメーションの FIL_InsertNewRecord を使用し、既存の行をフィルタリングして除外します。フィルタトランスフォーメーションは新しい行だけをアップデートストラテジトランスフォーメーション UPD_ForceInserts に渡します。UPD_ForceInserts は新しい行をターゲットに挿入し、シーケンスジェネレータは各行についてプライマリキーを作成します。

第二のデータフローでは、フィルタトランスフォーメーション FIL_UpdateChangedRecord により、変更を受けた行だけがアップデートストラテジトランスフォーメーション UPD_ChangedUpdate に渡されます。UPD_ChangedUpdate はターゲット中の既存の行を更新されたソース行で置換します。

トランスフォーメーションについて

以下の表に、タイプ 1 の次元マッピングにおける各トランスフォーメーションの機能を示します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
SQ_*SourceName*	ソース修飾子またはアプリケーションソース修飾子	マッピングウィザードで選択したソースから行をすべて選択します。
LKP_GetData	ルックアップ	既存のターゲットテーブルをキャッシュに格納します。 ソースのキーカラムとターゲットの対応キーカラムとを比較します。一致するキーが存在する場合、LKP_GetData は比較ターゲットから追加カラムデータを返します。 行をすべて EXP_DetectChanges に渡します。
EXP_DetectChanges	式	以下の式を使用して、ターゲット内に一致するキーがないソース行にフラグを設定します。ターゲットに一致キーが存在しない場合、この式は TRUE を返し、新しい行であることを明示します。`IIF（ISNULL（PM_PRIMARYKEY）,TRUE,FALSE）` NewFlag ポートに結果を書き込みます。 以下の式を用いて、ターゲット中に一致キーが*あり*、また指定カラムに変更を含むソース行にフラグを設定します。この式が TRUE を返すのは、ターゲット中に一致キーがあり（新しい行ではない）、さらにソースカラムとターゲットカラムとの違いを検出した場合だけです。`IIF（ISNULL（PM_PRIMARYKEY） AND（SourceColumnName<>PM_PREV_TargetColumnName） AND （その他の比較） TRUE,FALSE）` ChangedFlag ポートに結果を書き込みます。 すべての行を FIL_InsertNewRecord と FIL_UpdateChangedRecord に渡します。

新しい行のデータフロー

ソースのそれぞれの新しい行について、このデータフローはプライマリキーを作成し、有効日付範囲の開始日を設定して、行をターゲットに挿入します。

以下の表に、新しい行のデータフローを示します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_InsertNewRecord	フィルタ	フィルタ条件 `NewFlag` に従い、EXP_DetectChanges の結果から新規（TRUE）以外の行を除外します。新しい行を UPD_ForceInserts に渡します。
UPD_ForceInserts	アップデートストラテジ	`DD_INSERT` を使用して行をターゲットに挿入します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
SEQ_GenerateKeys	シーケンスジェネレータ	ターゲットに書き込んだ各新規行について値を生成し、各値を 1 だけインクリメントします。 値をターゲットに渡し、PM_PRIMARYKEY カラムに書き込みます。
T_*TargetName*	ターゲット定義	ターゲットに挿入する新しい行に対するターゲット定義のインスタンスです。

変更された行のデータフロー

ソース中の変更された各行について、このデータフローは行に更新のフラグを設定し、ターゲット中の該当行を上書きします。

以下の表に、変更された行のデータフローを示します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_UpdateChangedRecord	フィルタ	フィルタ条件 ChangedFlag に従い、EXP_DetectChanges の結果から変更（TRUE）以外の行を除外します。変更された行を UPD_ChangedUpdate に渡します。
UPD_ChangedUpdate	アップデートストラテジ	DD_UPDATE を使用してターゲット中の該当行を上書きします。
T_*ターゲット名* 1	ターゲット定義	ターゲット内の変更された行を上書きするターゲット定義のインスタンスです。

タイプ 1 の次元マッピングの作成手順

緩やかに成長するターゲットを使ったマッピングを作成するには、以下の手順を実行します。

1. Mapping Designer で、［マッピング］-［ウィザード］-［緩やかに変化する次元］をクリックします。
2. マッピング名を指定し、［タイプ 1 次元］を選択してから、［次へ］をクリックします。

   マッピング名を付けるにあたっては、m_*マッピング名*の形にしてください。
3. マッピングで使用するソース定義を選択します。

   ［ソーステーブルの選択］リストに利用可能なソース定義がすべて表示されます。このリストには、ショートカット、フラットファイル、リレーショナル、およびアプリケーションソースが含まれます。
4. マッピングのターゲットテーブルの名前を入力します。［次へ］をクリックします。

   ターゲット定義名を付けるにあたっては、「T_*ターゲット名*」の形にしてください。
5. ルックアップ条件として使用するカラムを［ターゲットテーブルのフィールド］リストから 1 つ以上選択し、［追加］をクリックします。

   ウィザードは選択したカラムを［論理キーフィールド］リストに追加します。

   ヒント: 選択するカラムはソースのキーカラムである必要があります。

セッションを含むワークフローの実行時に、Integration Service により既存のターゲットデータのルックアップが実行されます。［論理キーフィールド］カラムが対応するターゲットカラムに一致する場合、Integration Service によりターゲットデータが返されます。

［論理キーフィールド］からカラムを削除するには、カラムを選択して［削除］をクリックします。

6. Integration Service で変更を比較するカラムを 1 つ以上選択し、［追加］をクリックします。

   ウィザードにより選択したカラムが［変化を比較したいフィールド］リストに追加されます。

   セッションを含むワークフローを実行すると、Integration Service によりソース行と対応するターゲット（ルックアップ）行との間で、［変化を比較したいフィールド］リスト中のカラムが比較されます。Integration Service により変更が検出されると、行が変更のフラグに設定されます。

   リストからカラムを削除するには、カラムを選択して［削除］をクリックします。

7. ［終了］をクリックします。

   新たなマッピングがワークスペースに表示されます。マッピングに対し、必要な編集を行います。

   注: タイプ 1 の次元を使ったマッピングでは、Designer は同一のターゲット定義について 2 つのインスタンスを使用して、同じターゲットテーブルにおけるデータの挿入と更新を可能とします。ターゲットデータベースでは、ターゲットテーブルを 1 つだけ作成してください。

タイプ 1 の次元セッションの設定

タイプ 1 の次元を使ったマッピングでは、新しい行をプライマリキーと共に挿入し、既存の行を更新します。マッピングのセッションを作成する場合には、セッションを以下のように設定します。

1. セッションプロパティで、［プロパティ］タブの［全般］設定をクリックします。［ソース行の扱い］を［データドリブン］に設定します。
2. セッションプロパティで、［マッピング］タブの［ターゲットプロパティ］設定をクリックします。Integration Service により適切に行がターゲットにロードされることを確認するには、各リレーショナルターゲットに対して［挿入］および［更新として更新］を選択します。

タイプ 2 の次元/バージョンデータのマッピングの作成

タイプ 2 の次元/バージョンデータを使ったマッピングでは、ユーザーが定義する比較に基づいてソース行のフィルタリングを行い、新しい次元と変更された次元の両方をターゲットに挿入します。プライマリキーのバージョン管理を行い、またテーブル内の各次元に対してバージョン番号を作成することによって、ターゲットテーブルでの変更を追跡します。タイプ 2 の次元/バージョンデータターゲットでは、カレントバージョンの次元が最大のバージョン番号を持ち、インクリメントされる次元のプライマリキーの値も最大になります。

緩やかに変化する次元テーブルを更新する場合に、テーブル中の過去の次元データを完全に保持したければ、タイプ 2 の次元/バージョンデータを使ったマッピングを使用します。バージョン番号とバージョン管理されたプライマリキーを用いて、各次元に施された変更の順序を追跡します。

このオプションを使用すると、Designer はターゲットに 2 つの追加フィールドを作成します。

- **PM_PRIMARYKEY。**Integration Service により、ターゲットに書き込まれる各行のプライマリキーが生成されます。
- **PM_VERSION_NUMBER。**Integration Service により、ターゲットに書き込まれる各行のバージョン番号が生成されます。

キーの処理

タイプ 2 の次元/バージョンデータのマッピングでは、Integration Service により、ターゲットに挿入される新しい次元ごとに新しいプライマリキー値が生成されます。 式トランスフォーメーションにより、新しい次元に対してキー値に 1,000 単位で増加されます。

次元を更新する場合、Integration Service により、既存のプライマリキーが 1 つずつ増加されます。

例えば、Integration Service では以下の新規行のキー値を 65,000 として挿入されますが、その理由は、これがテーブルで 65 番目の次元だからです。

PM_PRIMARYKEY	ITEM	STYLES
65000	Sandal	5

次にこのセッションを含むワークフローを実行したときに、同じ項目に違うスタイル数があるとします。 Integration Service はスタイル情報が更新された新しい行を作成し、既存のキーを 1 増やして、65,001 という新しいキーを作成します。 両方の行はターゲットに存在しますが、キーバージョンの大きな方の行にカレント次元データが含まれます。

PM_PRIMARYKEY	ITEM	STYLES
65000	Sandal	5
65001	Sandal	14

このワークフローを再度実行する場合、Integration Service によりキーが再び増やされます。 最大のキーバージョンにはカレント次元データが含まれます。ターゲットでは、項目の過去のデータすべてに加えて、バージョンが発生した順序も保持されます。

PM_PRIMARYKEY	ITEM	STYLES
65000	Sandal	5
65001	Sandal	14
65002	Sandal	17

バージョンのナンバリング

プライマリキーのバージョニングに加えて、Integration Service ではターゲットに挿入され各行に一致するバージョン番号が生成されます。 バージョン番号はプライマリキーの最終桁に対応しています。新しい次元のバージョン番号は 0 です。

例えば以下のデータでは、バージョンは 0、1、および 2 です。最大のバージョン番号にカレント次元データが含まれています。

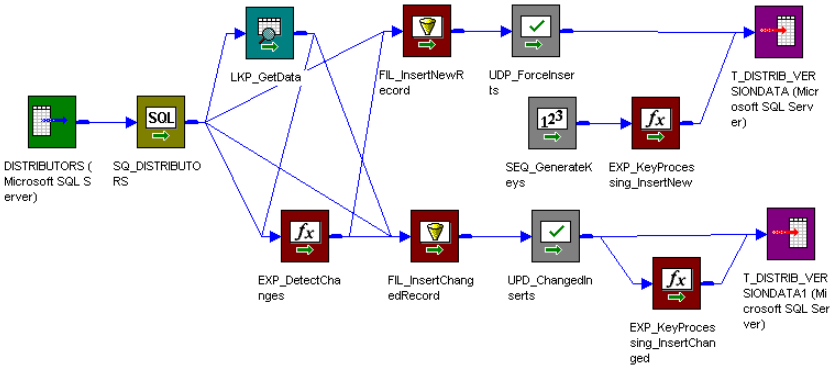
PM_PRIMARYKEY	ITEM	STYLES	PM_VERSION_NUMBER
65000	Sandal	5	0
65001	Sandal	14	1
65002	Sandal	17	2

マッピングについて

タイプ 2 の次元/バージョンデータを使ったマッピングでは、以下のことを行います。

- すべての行を選択します。
- 既存ターゲットをルックアップテーブルとしてキャッシュに格納します。
- ソース中の論理キーカラムとターゲットルックアップテーブル中の対応カラムとを比較します。
- キーカラムが一致した場合には、ソースカラムと対応ターゲットカラムとを比較します。
- 新しい行と変更された行にフラグを設定します。
- 2 つのデータフローを作成します。1 つは新しい行のデータフロー、もう 1 つは変更された行のデータフローです。
- 新しい行についてのプライマリキーとバージョン番号を作成します。
- ターゲットに新しい行を挿入します。
- 変更された行についてプライマリキーとバージョン番号をインクリメントします。
- 変更された行をターゲットに挿入します。

以下の図に、タイプ 2 の次元/バージョンデータのオプションを選択した場合に、緩やかに変化する次元のウィザードにより作成されるマッピングを示します。



タイプ 2 の次元/バージョンデータのマッピングでは、ルックアップトランスフォーメーションおよび式トランスフォーメーションが使用され、ソースデータと既存のターゲットデータが比較されます。緩やかに変化する次元のウィザードの手順を行う際に、Integration Service を既存のターゲットデータと比較するルックアップ条件（ソースキーカラム）とソースカラムを入力します。

ターゲット中に一致するプライマリキーのないソース行では、式トランスフォーメーションが行に新規というフラグを設定します。ターゲット中に一致するプライマリキーのあるソース行では、式はユーザー定義のソースカラムとターゲットカラムを比較します。これらのカラムが一致しない場合には、式は行に変更があったというフラグを設定します。その後、このマッピングは 2 つのデータフローに分かれます。

第一のデータフローはフィルタトランスフォーメーションの FIL_InsertNewRecord を使用し、既存の行をフィルタリングして除外します。フィルタトランスフォーメーションは新しい行だけをアップデートストラテジトランスフォーメーション UPD_ForceInserts に渡します。UPD_ForceInserts は新しい行をターゲットに挿入します。シーケンスジェネレータは各行のプライマリキーを作成します。式トランスフォーメーション EXP_KeyProcessing_InsertNew は、キー間の増分値を 1,000 だけ増やし、各新規行について 0 というバージョン番号を作成します。

第二のデータフローでは、フィルタトランスフォーメーション FIL_InsertChangedRecord により、変更を受けた行だけがアップデートストラテジトランスフォーメーション UPD_ChangedInserts に渡されます。UPD_ChangedInserts は変更された行をターゲットに挿入します。式トランスフォーメーション EXP_KeyProcessing_InsertChanged は、キーとバージョン番号の両方を 1 だけインクリメントします。

トランスフォーメーションについて

以下の表に、タイプ 2 の次元/バージョンデータのマッピングにおける各トランスフォーメーションの機能を示します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
SQ_*SourceName*	ソース修飾子またはアプリケーションソース修飾子	マッピングウィザードで選択したソースから行をすべて選択します。
LKP_GetData	ルックアップ	既存のターゲットテーブルをキャッシュに格納します。 ソースのキーカラムとターゲットの対応キーカラムとを比較します。一致するキーが存在する場合、LKP_GetData は比較ターゲットから追加カラムデータを返します。 行をすべて EXP_DetectChanges に渡します。
EXP_DetectChanges	式	以下の式を使用して、ターゲット内に一致するキーがないソース行にフラグを設定します。ターゲットに一致キーが存在しない場合、この式は TRUE を返し、新しい行であることを明示します。`IIF（ISNULL（PM_PRIMARYKEY）,TRUE,FALSE）` NewFlag ポートに結果を書き込みます。 以下の式を用いて、ターゲット中に一致キーがあり、また指定カラムに変更を含むソース行にフラグを設定します。この式が TRUE を返すのは、ターゲット中に一致キーがあり（新しい行ではない）、さらにソースカラムとターゲットカラムとの違いを検出した場合だけです。`IIF（ISNULL（PM_PRIMARYKEY） AND（SourceColumnName<>PM_PREV_TargetColumnName） AND （その他の比較） TRUE,FALSE）` ChangedFlag ポートに結果を書き込みます。 すべての行を FIL_InsertNewRecord および FIL_InsertChangedRecord に渡します。

新しい行のデータフロー

以下の表に、新しい行のデータフローを示します。 このデータフローでは、ソースの新しい行ごとにプライマリキーとバージョン番号が作成され、ターゲットに行が挿入されます。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_InsertNewRecord	フィルタ	フィルタ条件 NewFlag に従い、EXP_DetectChanges の結果から新規（TRUE）以外の行を除外します。新しい行を UPD_ForceInserts に渡します。
UPD_ForceInserts	アップデートストラテジ	DD_INSERT を使用して行をターゲットに挿入します。
SEQ_GenerateKeys	シーケンスジェネレータ	ターゲットに書き込んだ各新規行について値を生成し、各値を 1 だけインクリメントします。 値を EXP_KeyProcessing_InsertNew に渡します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
EXP_KeyProcessing_InsertNew	式	式 NEXTVAL * 1,000 を使用し、生成された値を 1,000 だけインクリメントします。次に、インクリメントした値をターゲットに渡して、PM_PRIMARYKEY カラムに書き込みます。 各行についてバージョン番号 0 を作成し、ターゲットの PM_VERSION_NUMBER カラムに書き込みます。
T_*TargetName*	ターゲット定義	ターゲットに挿入する新しい行に対するターゲット定義のインスタンスです。

変更された行のデータフロー

以下の表に、変更された行のデータフローを示します。 このデータフローでは、ソースの変更された行ごとに既存のプライマリキーが 1 つずつ増やされ、対応するバージョン番号が作成され、ターゲットに行が挿入されます。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_InsertChangedRecord	フィルタ	フィルタ条件 ChangedFlag に従い、EXP_DetectChanges の結果から変更（TRUE）以外の行を除外します。変更された行を UPD_ChangedInserts に渡します。
UPD_ChangedInserts	アップデートストラテジ	DD_INSERT を使用して行をターゲットに挿入します。
EXP_KeyProcessing_InsertChanged	式	式 PM_PRIMARYKEY + 1 を使用し、既存のプライマリキーを 1 だけインクリメントします。 式（PM_PRIMARYKEY + ） % 1,000 を使用し、既存のバージョン番号を 1 だけインクリメントします。
T_*ターゲット名* 1	ターゲット定義	ターゲットに変更された行を挿入するターゲット定義のインスタンスです。

タイプ 2 の次元/バージョンデータのマッピングの作成手順

タイプ 2 の次元/バージョンデータを使ったマッピングを作成するには：

1. Mapping Designer で、［マッピング］-［ウィザード］-［緩やかに変化する次元］の順にクリックします。
2. マッピング名を指定し、［タイプ 2 次元］を選択します。［次へ］をクリックします。
3. マッピングで使用するソース定義を選択します。

   ［ソーステーブルの選択］リストに利用可能なソース定義がすべて表示されます。このリストには、ショートカット、フラットファイル、リレーショナル、およびアプリケーションソースが含まれます。
4. マッピングのターゲットテーブルの名前を入力します。［次へ］をクリックします。

   ターゲット定義名を付けるにあたっては、「T_*ターゲット名*」の形にしてください。
5. ルックアップ条件として使用するカラムを［ターゲットテーブルのフィールド］リストから 1 つ以上選択し、［追加］をクリックします。

ウィザードは選択したカラムを［論理キーフィールド］リストに追加します。

ヒント: 選択するカラムはソースのキーカラムである必要があります。

セッションを含むワークフローの実行時に、Integration Service により既存のターゲットデータのルックアップが実行されます。 ［論理キーフィールド］カラムが対応するターゲットカラムに一致する場合、Integration Service によりターゲットデータが返されます。

［論理キーフィールド］からカラムを削除するには、カラムを選択して［削除］をクリックします。

6. Integration Service で変更を比較するカラムを 1 つ以上選択し、［追加］をクリックします。

   ウィザードにより選択したカラムが［変化を比較したいフィールド］リストに追加されます。

   セッションを含むワークフローを実行すると、Integration Service によりソース行と対応するターゲット行との間で、［変化を比較したいフィールド］リスト中のカラムが比較されます。 Integration Service により変更が検出されると、行が変更のフラグに設定されます。

   リストからカラムを削除するには、カラムを選択して［削除］をクリックします。

7. ［次へ］をクリックします。
8. ［別のカラムにバージョン番号を持つ］を選択します。［終了］をクリックします。

   注: タイプ 2 の次元/バージョンデータを使ったマッピングでは、Designer は同一のターゲット定義の 2 つのインスタンスを使用して、2 つの別のデータフローから同じターゲットテーブルに書き込むことができます。ターゲットデータベースでは、ターゲットテーブルを 1 つだけ作成してください。

マッピングのカスタマイズ

次元データに存在すると予期されるバージョンの数に応じて、Integration Service により生成キー間で作成される増分を加減することができます。 デフォルトでは、Integration Service によりシーケンスが 1,000 単位で増加されます。 この場合、単一の次元で使用できるバージョン数は 1,000 となります。

必要に応じて、このインクリメントの加減ができます。そのためには、マッピングを作成したあとで、式トランスフォーメーションの EXP_KeyProcessing_InsertNew を編集します。

プライマリキーのインクリメントを変更するには：

1. 式トランスフォーメーション EXP_KeyProcessing_InsertNew のタイトルバーをダブルクリックします。
2. このトランスフォーメーションには以下のポートがあります。
3. PM_PRIMARYKEY ポートの［式］フィールドの右隅をクリックします。

   式エディタが表示されます。

4. 現在の値 1000 を削除して、プライマリキーを増やす際に Integration Service で使用する値を指定します。 ［検証］をクリックします。

タイプ 2 の次元/バージョンデータのセッションの設定

タイプ 2 の次元/バージョンデータを使ったマッピングでは、新しい行と更新された行の両方を一意のプライマリキーと共に挿入します。マッピングのセッションを設定するには、以下の手順を実行します。

1. セッションプロパティで、［プロパティ］タブの［全般］設定をクリックします。［ソース行の扱い］を［データドリブン］に設定します。
2. セッションプロパティで、［マッピング］タブの［ターゲットプロパティ］設定をクリックします。Integration Service により適切に行がターゲットにロードされることを確認するには、各リレーショナルターゲットに対して［挿入］を選択します。

タイプ 2 の次元/フラグカレントのマッピングの作成

タイプ 2 の次元/フラグを使ったマッピングでは、ユーザーが定義する比較に基づいてソース行のフィルタリングを行い、新しい次元と変更された次元の両方をターゲットに挿入します。各次元のカレントバージョンにフラグを設定し、またプライマリキーのバージョン管理を行うことにより、ターゲットテーブルの変更を追跡します。タイプ 2 の次元/フラグターゲットでは、カレントバージョンの次元においてカレントフラグが 1 に設定され、インクリメントされるプライマリキーの値も最大になります。

緩やかに変化する次元テーブルを更新する場合に、テーブル中の過去の次元データを完全に保持し、最新のデータにフラグを設定したければ、タイプ 2 の次元/フラグを使ったマッピングを使用します。バージョン管理されたプライマリキーを用いて、各次元の変更の順序を追跡します。

このオプションを使用すると、Designer はターゲットに 2 つの追加フィールドを作成します。

- **PM_CURRENT_FLAG。**Integration Service により、カレント行に「1」というフラグが設定され、前のバージョンのすべてに「0」というフラグが設定されます。
- **PM_PRIMARYKEY。**Integration Service により、ターゲットに書き込まれる各行のプライマリキーが生成されます。

カレント値のフラグ設定

Integration Service により、ターゲットに書き込まれる各行に対し、1 というカレントフラグが生成されます。 このフラグは、次元が新しいか新たに更新されたことを示します。行が既存の次元に対する更新である場合には、Integration Service により既存の次元のカレントフラグが 0 にリセットされます。

その結果、カレントバージョンの次元はすべてカレントフラグを 1 としてターゲットに表示され、以前のバージョンはすべてカレントフラグが 0 となります。

例えば、以下の次元データは、カレントフラグが 1 に設定されているので、現在のデータです。

ITEM	STYLES	PM_CURRENT_FLAG
Sandal	5	1
Boot	25	1

これらの次元が変化すると、Integration Service により、カレントフラグの 1 を設定して更新されたバージョンが挿入されます。 Integration Service により、ターゲット内の既存の行の更新も行われます。 以前のカレントバージョン（カレントフラグが 1 に設定されている）が検出され、カレントフラグが 0 に更新されます。

ITEM	STYLES	PM_CURRENT_FLAG
Sandal	5	0
Boot	25	0
Sandal	12	1
Boot	15	1

キーの処理

フラグカレントオプションを使用すると、Integration Service ではターゲットに書き込まれる各行のプライマリキー値が作成され、キー値が 1 ずつ増加されます。 式トランスフォーメーションにより、新しい次元に対してキー値に 1,000 単位で増加されます。

既存の次元を更新する場合、Integration Service により、既存のプライマリキーが 1 ずつ増加されます。

例えば、以下の次元はカレント次元データです（カレントフラグは 1 に設定されています）。そのプライマリキーは 1,000 の倍数です。つまり、両方とも次元の最初のバージョンということになります。

PM_PRIMARYKEY	ITEM	STYLES	PM_CURRENT_FLAG
3000	Sandal	5	1
4000	Boot	25	1

Integration Service により、これらの次元の更新バージョンがターゲット内に挿入されると、カレントフラグが 1 に設定されます。 また、既存の次元のキーが 1 ずつ増加されることにより、更新行に対し新しいプライマリキーも作成されます。Integration Service により、カレントフラグが 0 にリセットされることで、既存の次元がもはや現在のものではないことが示されます。

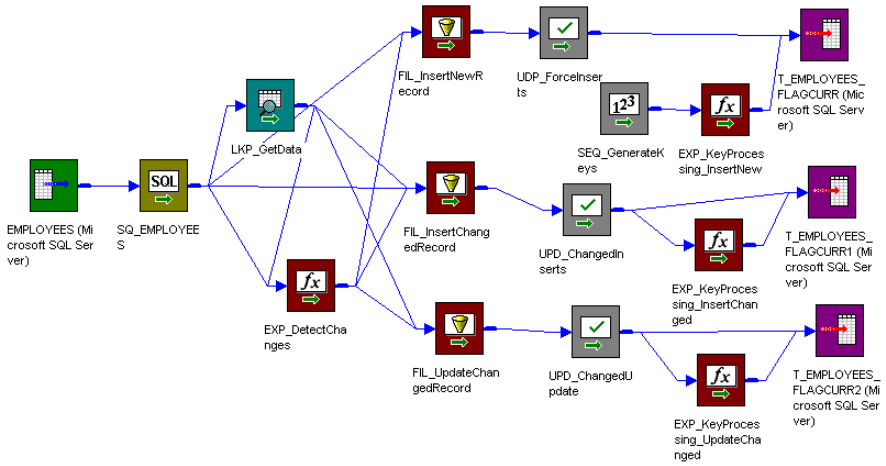
PM_PRIMARYKEY	ITEM	STYLES	PM_CURRENT_FLAG
3000	Sandal	5	0
4000	Boot	25	0
3001	Sandal	12	1
4001	Boot	15	1

マッピングについて

タイプ 2 の次元/フラグを使ったマッピングでは、以下のタスクを行います。

- すべての行を選択します。
- 既存ターゲットをルックアップテーブルとしてキャッシュに格納します。
- ソース中の論理キーカラムとターゲットルックアップテーブル中の対応カラムとを比較します。
- キーカラムが一致した場合には、ソースカラムと対応ターゲットカラムとを比較します。
- 新しい行と変更された行にフラグを設定します。
- 2 つのデータフローを作成します。1 つは新しい行のデータフロー、もう 1 つは変更された行のデータフローです。
- 新しい行についてのプライマリキーとカレントフラグを作成します。
- ターゲットに新しい行を挿入します。
- 既存のプライマリキーをインクリメントし、また変更された行についてカレントフラグを設定します。
- 変更された行をターゲットに挿入します。
- ターゲット中の変更された行の既存のバージョンを更新し、カレントフラグをリセットして、当該行がもはや現在のものではないことを明示します。

以下の図に、緩やかに変化する次元ウィザードのタイプ 2 の次元/フラグカレントオプションにより作成されるマッピングを示します。



タイプ 2 の次元/フラグカレントのマッピングでは、ルックアップトランスフォーメーションおよび式トランスフォーメーションが使用され、ソースデータと既存のターゲットデータが比較されます。緩やかに変化する次元のウィザードの手順を行う際に、Integration Service を既存のターゲットデータと比較するルックアップ条件（ソースキーカラム）とソースカラムを入力します。

ターゲット中に一致するプライマリキーのないソース行では、式トランスフォーメーションが行に新規というフラグを設定します。ターゲット中に一致するプライマリキーのあるソース行では、式はユーザー定義のソースカラムとターゲットカラムを比較します。これらのカラムが一致しない場合には、式は行に変更があったというフラグを設定します。その後、このマッピングは 3 つのデータフローに分かれます。

第一のデータフローはフィルタトランスフォーメーションの FIL_InsertNewRecord を使用し、既存の行をフィルタリングして除外します。フィルタトランスフォーメーションは新しい行だけをアップデートストラテジトランスフォーメーション UPD_ForceInserts に渡します。UPD_ForceInserts は新しい行をターゲットに挿入します。シーケンスジェネレータは各新規行のプライマリキーを作成します。式トランスフォーメーション EXP_KeyProcessing_InsertNew は、キー間の増分値を 1,000 だけ増やし、各新規行について 1 というカレントフラグを作成します。

第二のデータフローでは、フィルタトランスフォーメーション FIL_InsertChangedRecord により、変更を受けた行だけがアップデートストラテジトランスフォーメーション UPD_ChangedInserts に渡されます。UPD_ChangedInserts は変更された行をターゲットに挿入します。式トランスフォーメーション EXP_KeyProcessing_InsertChanged は、プライマリキーを 1 だけインクリメントし、1 というカレントフラグを作成して、更新された行にカレント次元データが含まれていることを明示します。

第三のデータフローでは、ターゲットに書き込まれた各変更行について、フィルタトランスフォーメーション FIL_UpdateChangedRecord が前バージョンのプライマリキーをアップデートストラテジトランスフォーメーション UPD_ChangedUpdate に渡します。UPD_ChangedUpdate はターゲット中の次元を更新します。式トランスフォーメーション EXP_UpdateChanged はカレントフラグを 0 に設定します。これにより、前次元の状態が現在のものから過去のものへと変化します。

トランスフォーメーションについて

以下の表に、タイプ 2 の次元/フラグカレントのマッピングにおける各トランスフォーメーションの機能を示します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
SQ_*SourceName*	ソース修飾子またはアプリケーションソース修飾子	マッピングウィザードで選択したソースから行をすべて選択します。
LKP_GetData	ルックアップ	既存のターゲットテーブルをキャッシュに格納します。 ソースのキーカラムとターゲットの対応キーカラムとを比較します。一致するキーが存在する場合、LKP_GetData は比較ターゲットから追加カラムデータを返します。 行をすべて EXP_DetectChanges に渡します。
EXP_DetectChanges	式	以下の式を使用して、ターゲット内に一致するキーが*ない*ソース行にフラグを設定します。ターゲットに一致キーが存在しない場合、この式は TRUE を返し、新しい行であることを明示します。`IIF（ISNULL（PM_PRIMARYKEY）,TRUE,FALSE）` NewFlag ポートに結果を書き込みます。 以下の式を用いて、ターゲット中に一致キーが*あり*、また指定カラムに変更を含むソース行にフラグを設定します。この式が TRUE を返すのは、ターゲット中に一致キーがあり（新しい行ではない）、さらにソースカラムとターゲットカラムとの違いを検出した場合のみです。 `IIF(ISNULL(PM_PRIMARYKEY) AND (SourceColumnName<>PM_PREV_TargetColumnName) AND (other comparisons) TRUE,FALSE)` ChangedFlag ポートに結果を書き込みます。 すべての行を FIL_InsertNewRecord、FIL_InsertChangedRecord、および FIL_UpdateChangedRecord に渡します。

新しい行のデータフロー

以下の表に、新しい行のデータフローを示します。 ソースのそれぞれの新しい行について、このデータフローはプライマリキーを作成し、1,000 だけインクリメントさせます。また、当該行にカレントフラグを設定してターゲットに挿入します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_InsertNewRecord	フィルタ	フィルタ条件 NewFlag に従い、EXP_DetectChanges の結果から新規（TRUE）以外の行を除外します。新しい行を UPD_ForceInserts に渡します。
UPD_ForceInserts	アップデートストラテジ	DD_INSERT を使用して行をターゲットに挿入します。
SEQ_GenerateKeys	シーケンスジェネレータ	ターゲットに書き込んだ各新規行について値を生成し、各値を 1 だけインクリメントします。 値を EXP_KeyProcessing_InsertNew に渡します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
EXP_KeyProcessing_InsertNew	式	式 NEXTVAL * 1,000 を使用し、生成された値を 1,000 だけインクリメントします。次に、インクリメントした値をターゲットに渡して、PM_PRIMARYKEY カラムに書き込みます。 各行について 1 というカレントフラグを作成し、ターゲットの PM_CURRENT_FLAG カラムに書き込みます。
T_*TargetName*	ターゲット定義	ターゲットに挿入する新しい行に対するターゲット定義のインスタンスです。

変更された行のデータフロー

以下の表に、変更された行のデータフローを示します。 このデータフローでは、ソースの変更された行ごとに既存のプライマリキーが 1 つずつ増やされ、行にカレントフラグが設定され、ターゲットが挿入されます。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_InsertChangedRecord	フィルタ	フィルタ条件 ChangedFlag に従い、変更以外の行を除外します。変更された行を UPD_ChangedInserts に渡します。
UPD_ChangedInserts	アップデートストラテジ	DD_INSERT を使用して行をターゲットに挿入します。
EXP_KeyProcessing_InsertChanged	式	式 PM_PRIMARYKEY + 1 を使用し、既存のプライマリキーを 1 だけインクリメントします。 各行について 1 というカレントフラグを作成し、ターゲットの PM_CURRENT_FLAG カラムに書き込みます。
T_*ターゲット名* 2	ターゲット定義	ターゲットに変更された行を挿入するターゲット定義のインスタンスです。

既存の行を更新するデータフロー

以下の表に、既存の行のデータフローを示します。 このデータフローでは、ソースの変更された行ごとに、ターゲットの対応する行の終了日が更新され、既存の行がもはや現在のものではないことが示されます。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_UpdateChangedRecord	フィルタ	フィルタ条件 ChangedFlag に従い、EXP_DetectChanges の結果から変更（TRUE）以外の行を除外します。 各変更行について、前バージョンのプライマリキーを UPD_ChangedUpdate に渡します。
UPD_ChangedUpdate	アップデートストラテジ	DD_UPDATE を用いて、ターゲットの既存の行を更新します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
EXP_KeyProcessing_UpdateChanged	式	各変更行で、ターゲットの前バージョンについて PM_CURRENT_FLAG を 0 に設定し、このバージョンがもはや現在のものではないことを示します。
T_*ターゲット名* 3	ターゲット定義	ターゲット内の既存の行を更新するターゲット定義のインスタンスです。

タイプ 2 の次元/フラグカレントのマッピングの作成手順

タイプ 2 の次元/フラグを使ったマッピングを作成するには：

1. Mapping Designer で、［マッピング］-［ウィザード］-［緩やかに変化する次元］の順にクリックします。
2. マッピング名を指定し、［タイプ 2 次元］を選択します。［次へ］をクリックします。

   マッピング名を付けるにあたっては、m_ *マッピング名*の形にしてください。
3. マッピングで使用するソース定義を選択します。

   ［ソーステーブルの選択］リストに利用可能なソース定義がすべて表示されます。このリストには、ショートカット、フラットファイル、リレーショナル、およびアプリケーションソースが含まれます。
4. マッピングのターゲットテーブルの名前を入力します。［次へ］をクリックします。

   ターゲット定義名を付けるにあたっては、「T_*ターゲット名*」の形にしてください。
5. ルックアップ条件として使用するカラムを［ターゲットテーブルのフィールド］リストから 1 つ以上選択し、［追加］をクリックします。

   ウィザードは選択したカラムを［論理キーフィールド］リストに追加します。

   ヒント: 選択するカラムはソースのキーカラムである必要があります。

   セッションの実行時に、Integration Service により既存のターゲットデータのルックアップが実行されます。 ［論理キーフィールド］カラムが対応するターゲットカラムに一致する場合、Integration Service によりターゲットデータが返されます。

   ［論理キーフィールド］からカラムを削除するには、カラムを選択して［削除］をクリックします。
6. Integration Service で変更を比較するカラムを 1 つ以上選択し、［追加］をクリックします。

   ウィザードにより選択したカラムが［変化を比較したいフィールド］リストに追加されます。

   セッションを実行すると、Integration Service によりソース行と対応するターゲット（ルックアップ）行との間で、［変化を比較したいフィールド］リスト中のカラムが比較されます。 Integration Service により変更が検出されると、行が変更のフラグに設定されます。

   リストからカラムを削除するには、カラムを選択して［削除］をクリックします。
7. ［次へ］をクリックします。［フラグで現在の次元に印を付けます］を選択します。
8. ［終了］をクリックします。

   新たなマッピングがワークスペースに表示されます。マッピングに対し、必要な編集を行います。

   注: タイプ 2 の次元/フラグを使ったマッピングでは、Designer は同一のターゲット定義の 3 つのインスタンスを使用して、3 つの別のデータフローから同じターゲットテーブルに書き込むことができます。ターゲットデータベースでは、ターゲットテーブルを 1 つだけ作成してください。

タイプ 2 の次元/フラグカレントのセッションの設定

タイプ 2 の次元/フラグを使ったマッピングでは、新しい行と更新された行の両方を一意のプライマリキーと共に挿入します。また、ターゲット中の既存の行の更新も行います。マッピングのセッションを設定するには、以下の手順を実行します。

1. セッションプロパティで、［プロパティ］タブの［全般］設定をクリックします。［ソース行の扱い］を［データドリブン］に設定します。
2. セッションプロパティで、［マッピング］タブの［ターゲットプロパティ］設定をクリックします。Integration Service により適切に行がターゲットにロードされることを確認するには、各リレーショナルターゲットに対して［挿入］および［更新として更新］を選択します。

タイプ 2 の次元/有効な日付範囲のマッピングの作成

タイプ 2 の次元/有効日付を使ったマッピングでは、ユーザーが定義する比較に基づいてソース行のフィルタリングを行い、新しい次元と変更された次元の両方をターゲットに挿入します。ターゲット内の各次元の各バージョンに対する有効期間を保持することにより、ターゲットテーブルの変更を追跡します。タイプ 2 の次元/有効日付ターゲットにおいて、カレントバージョンの次元には開始日があるだけで対応する終了日がありません。

緩やかに変化する次元テーブルを更新する場合に、テーブル中の過去の次元データを完全に保持したければ、タイプ 2 の次元/有効日付を使ったマッピングを使用します。有効期間により、各次元の変更の履歴を時間順に追跡します。

このオプションを使用すると、Designer はターゲットに以下の追加フィールドを作成します。

- **PM_BEGIN_DATE。**ターゲットに書き込まれた新しい次元および変更された次元のそれぞれについて、Integration Service ではシステムの日付を使用して、次元の有効な日付範囲の開始日が示されます。
- **PM_END_DATE。**更新される各次元について、Integration Service ではシステムの日付を使用して、次元の有効な日付範囲の終了日が示されます。
- **PM_PRIMARYKEY。**Integration Service により、ターゲットに書き込まれる各行のプライマリキーが生成されます。

有効な日付範囲の保持

Integration Service では、現在のシステムの日付を使用して、ターゲットに挿入する新しい次元と変更された次元のそれぞれについて開始日が生成されます。 これらの次元の終了日は NULL です。

Integration Service により変更された次元が挿入されるたびに、ターゲット内の以前のバージョンの次元が更新され、現在のシステムの日付を使用して、以前は NULL だった終了日カラムが埋められます。

その結果、タイプ 2 の次元/有効期間ターゲットのカレント次元データはすべて、その PM_END_DATE カラムに NULL 値が入っています。前バージョンの次元はすべて、PM_END_DATE にシステム日付が入り、各バージョンの有効期間の終わりを表します。

例えば、以下の次元はカレント次元データですが、その理由は終了日カラムが NULL だからです。

PM_PRIMARYKEY	ITEM	STYLES	PM_BEGIN_DATE	PM_END_DATE
4325	Sock	13	9/1/98	-
5401	Boot	20	10/1/98	-

Integration Service により、これらの次元の更新バージョンがソース内で検出されると、これがターゲット内に挿入され、システムの日付を使用して有効な日付範囲の開始日が示されると共に、終了日が NULL のままとされます。

また、Integration Service により、ターゲット内の既存のバージョンが更新され、システムの日付が有効な日付範囲の終了日に入力されます。

PM_PRIMARYKEY	ITEM	STYLES	PM_BEGIN_DATE	PM_END_DATE
4325	Sock	13	9/1/98	6/1/99
5401	Boot	20	10/1/98	6/1/99
6345	Sock	18	6/1/99	-
6346	Boot	25	6/1/99	-

キーの処理

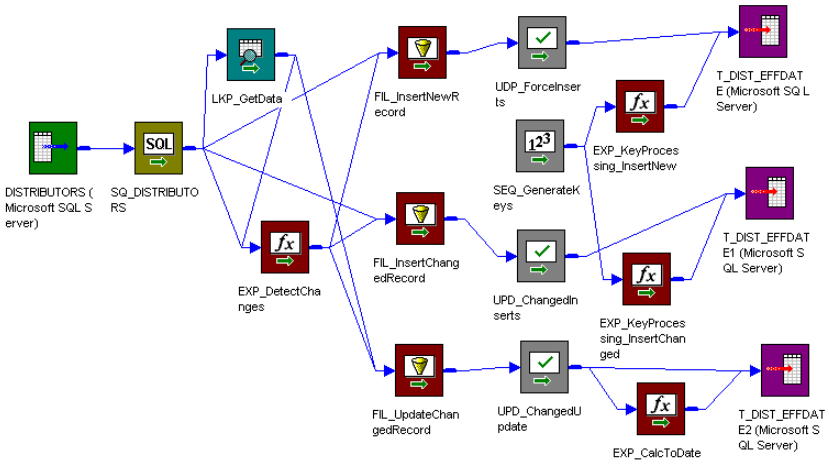
有効な日付範囲オプションを使用すると、Integration Service ではターゲットに書き込まれる各行のプライマリキー値が作成され、キー値が 1 ずつ増加されます。

マッピングについて

タイプ 2 の次元/有効日付を使ったマッピングでは、以下のことを行います。

- すべての行を選択します。
- 既存ターゲットをルックアップテーブルとしてキャッシュに格納します。
- ソース中の論理キーカラムとターゲットルックアップテーブル中の対応カラムとを比較します。
- キーカラムが一致した場合には、ソースカラムと対応ターゲットカラムとを比較します。
- 新しい行と変更された行にフラグを設定します。
- 3 つのデータフローを作成します。新しい行のデータフロー、変更された行のデータフロー、および既存の行を更新するデータフローです。
- プライマリキーを生成し、新しい行に対して有効期間の開始日を生成します。
- ターゲットに新しい行を挿入します。
- プライマリキーを生成し、変更された行に対して有効期間の開始日を生成します。
- 変更された行をターゲットに挿入します。
- ターゲット中の変更された行の既存のバージョンを更新し、有効期間の終了日を明示して、当該行がもはや現在のものではないことを示します。

以下の図に、緩やかに変化する次元ウィザードのタイプ 2 の次元/有効な日付範囲オプションにより作成されるマッピングを示します。



タイプ 2 の次元/有効な日付範囲のマッピングでは、ルックアップトランスフォーメーションおよび式トランスフォーメーションが使用され、ソースデータと既存のターゲットデータが比較されます。緩やかに変化する次元のウィザードの手順を行う際に、Integration Service を既存のターゲットデータと比較するルックアップ条件（ソースキーカラム）とソースカラムを入力します。

ターゲット中に一致するプライマリキーのないソース行では、式トランスフォーメーションが行に新規というフラグを設定します。ターゲット中に一致するプライマリキーのあるソース行では、式はユーザー定義のソースカラムとターゲットカラムを比較します。これらのカラムが一致しない場合には、式は行に変更があったというフラグを設定します。その後、このマッピングは 3 つのデータフローに分かれます。

第一のデータフローはフィルタトランスフォーメーションの FIL_InsertNewRecord を使用し、既存の行をフィルタリングして除外します。フィルタトランスフォーメーションは新しい行だけをアップデートストラテジトランスフォーメーション UPD_ForceInserts に渡します。UPD_ForceInserts は新しい行をターゲットに挿入します。シーケンスジェネレータは各行のプライマリキーを作成します。式トランスフォーメーション EXP_KeyProcessing_InsertNew は、システム日付を用いて、有効期間の開始日を表します。このトランスフォーメーションは終了日を NULL のままとし、新しい行にカレント次元データが含まれていることを示します。

第二のデータフローでは、フィルタトランスフォーメーション FIL_InsertChangedRecord により、変更を受けた行だけがアップデートストラテジトランスフォーメーション UPD_ChangedInserts に渡されます。UPD_ChangedInserts は変更された行をターゲットに挿入します。式トランスフォーメーション EXP_KeyProcessing_InsertChanged は、システム日付を用いて、有効期間の開始日を表します。このトランスフォーメーションは終了日を NULL のままとし、変更された行にカレント次元データが含まれていることを示します。

第三のデータフローでは、ターゲットに書き込まれた各変更行について、フィルタトランスフォーメーション FIL_UpdateChangedRecord が前バージョンのプライマリキーをアップデートストラテジトランスフォーメーション UPD_ChangedUpdate に渡します。UPD_ChangedUpdate はターゲット中の行を更新します。式トランスフォーメーション EXP_UpdateChanged は、システム日付を用いて終了日カラムを更新します。これにより、行の状態がカレントバージョンから前バージョンに変化します。

トランスフォーメーションについて

以下の表に、タイプ 2 の次元/有効な日付範囲のマッピングにおける各トランスフォーメーションの機能を示します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
SQ_*SourceName*	ソース修飾子またはアプリケーションソース修飾子	マッピングウィザードで選択したソースから行をすべて選択します。
LKP_GetData	ルックアップ	既存のターゲットテーブルをキャッシュに格納します。 ソースのキーカラムとターゲットの対応キーカラムとを比較します。一致するキーが存在する場合、LKP_GetData は比較ターゲットから追加カラムデータを返します。 行をすべて EXP_DetectChanges に渡します。
EXP_DetectChanges	式	以下の式を使用して、ターゲット内に一致するキーが*ない*ソース行にフラグを設定します。ターゲットに一致キーが存在しない場合、この式は TRUE を返し、新しい行であることを明示します。`IIF（ISNULL（PM_PRIMARYKEY）,TRUE,FALSE）` NewFlag ポートに結果を書き込みます。 以下の式を用いて、ターゲット中に一致キーが*あり*、また指定カラムに変更を含むソース行にフラグを設定します。この式が TRUE を返すのは、ターゲット中に一致キーがあり（新しい行ではない）、さらにソースカラムとターゲットカラムとの違いを検出した場合だけです。`IIF（ISNULL（PM_PRIMARYKEY） AND （SourceColumnName<>PM_PREV_TargetColumnName） AND （その他の比較） TRUE,FALSE）` ChangedFlag ポートに結果を書き込みます。 すべての行を FIL_InsertNewRecord、FIL_InsertChangedRecord、および FIL_UpdateChangedRecord に渡します。

新しい行のデータフロー

以下の表に、新しい行のデータフローを示します。 このデータフローでは、ソースの新しい行ごとにプライマリキーが作成され、有効な日付範囲の開始日が設定され、行がターゲットに挿入されます。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_InsertNewRecord	フィルタ	フィルタ条件 `NewFlag` に従い、EXP_DetectChanges の結果から新規（TRUE）以外の行を除外します。新しい行を UPD_ForceInserts に渡します。
UPD_ForceInserts	アップデートストラテジ	`DD_INSERT` を使用して行をターゲットに挿入します。
SEQ_GenerateKeys	シーケンスジェネレータ	ターゲットに書き込んだ各新規行について値を生成し、各値を 1 だけインクリメントします。 値を EXP_KeyProcessing_InsertNew に渡します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
EXP_KeyProcessing_InsertNew	式	生成した値をターゲットに渡し、ターゲットの PM_PRIMARYKEY カラムに書き込みます。 SYSDATE を使用し、ターゲットの PM_BEGIN_DATE カラムに書き込みを行って、有効期間の開始日を明示します。
T_*TargetName*	ターゲット定義	ターゲットに挿入する新しい行に対するターゲット定義のインスタンスです。

変更された行のデータフロー

以下の表に、変更された行のデータフローを示します。 このデータフローでは、ソースの変更された行ごとに新しいプライマリキーが作成され、有効な日付範囲の開始日が設定され、行がターゲットに挿入されます。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_InsertChangedRecord	フィルタ	フィルタ条件 ChangedFlag に従い、EXP_DetectChanges の結果から変更（TRUE）以外の行を除外します。変更された行を UPD_ChangedInserts に渡します。
UPD_ChangedInserts	アップデートストラテジ	DD_INSERT を使用して行をターゲットに挿入します。
SEQ_GenerateKeys（上記のシーケンスジェネレータと同じ）	シーケンスジェネレータ	ターゲットに書き込んだ各変更行について値を生成し、各値を 1 だけインクリメントします。 値を EXP_KeyProcessing_InsertChanged に渡します。
EXP_KeyProcessing_InsertChanged	式	生成した値をターゲットに渡し、ターゲットの PM_PRIMARYKEY カラムに書き込みます。 SYSDATE を使用し、ターゲットの PM_BEGIN_DATE カラムに書き込みを行って、有効期間の開始日を明示します。
T_*ターゲット名* 2	ターゲット定義	ターゲットに変更された行を挿入するターゲット定義のインスタンスです。

既存の行を更新するデータフロー

以下の表に、既存の行のデータフローを示します。 このデータフローでは、ソースの変更された行ごとに、ターゲットの対応する行の終了日が更新され、既存の行がもはや現在のものではないことが示されます。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_UpdateChangedRecord	フィルタ	フィルタ条件 ChangedFlag に従い、EXP_DetectChanges の結果から変更（TRUE）以外の行を除外します。 各変更行について、前バージョンのプライマリキーを UPD_ChangedUpdate に渡します。
UPD_ChangedUpdate	アップデートストラテジ	DD_UPDATE を用いて、ターゲットの既存の行を更新します。
EXP_CalcToDate	式	SYSDATE を使用し、既存のターゲット行の PM_END_DATE カラムを更新し、有効期間の終了日を明示します。
T_*ターゲット名* 3	ターゲット定義	ターゲット内の既存の行を更新するターゲット定義のインスタンスです。

タイプ 2 の次元/有効な日付範囲のマッピングの作成手順

タイプ 2 の次元/有効日付を使ったマッピングを作成するには：

1. Mapping Designer で、［マッピング］-［ウィザード］-［緩やかに変化する次元］の順にクリックします。
2. マッピング名を指定し、［タイプ 2 次元］を選択します。［次へ］をクリックします。

   マッピング名を付けるにあたっては、m_ *マッピング名*の形にしてください。
3. マッピングで使用するソース定義を選択します。

   ［ソーステーブルの選択］リストに利用可能なソース定義がすべて表示されます。このリストには、ショートカット、フラットファイル、リレーショナル、およびアプリケーションソースが含まれます。
4. マッピングのターゲットテーブルの名前を入力します。［次へ］をクリックします。

   ターゲット定義名を付けるにあたっては、「T_*ターゲット名*」の形にしてください。
5. ルックアップ条件として使用するカラムを［ターゲットテーブルのフィールド］リストから 1 つ以上選択し、［追加］をクリックします。

   ウィザードは選択したカラムを［論理キーフィールド］リストに追加します。

   ヒント: 選択するカラムはソースのキーカラムである必要があります。

   セッションの実行時に、Integration Service により既存のターゲットデータのルックアップが実行されます。 ［論理キーフィールド］カラムが対応するターゲットカラムに一致する場合、Integration Service によりターゲットデータが返されます。

   ［論理キーフィールド］からカラムを削除するには、カラムを選択して［削除］をクリックします。
6. Integration Service で変更を比較するカラムを 1 つ以上選択し、［追加］をクリックします。

   ウィザードにより選択したカラムが［変化を比較したいフィールド］リストに追加されます。

セッションを実行すると、Integration Service によりソース行と対応するターゲット行との間で、［変化を比較したいフィールド］リスト中のカラムが比較されます。 Integration Service により変更が検出されると、行が変更のフラグに設定されます。

リストからカラムを削除するには、カラムを選択して［削除］をクリックします。

7. ［次へ］をクリックします。
8. ［有効期間を使って次元レコードに印を付けます］を選択してください。［終了］をクリックします。

新たなマッピングがワークスペースに表示されます。マッピングに対し、必要な編集を行います。

注: タイプ 2 の次元/有効日付を使ったマッピングでは、Designer は同一のターゲット定義の 3 つのインスタンスを使用して、3 つの別のデータフローから同じターゲットテーブルに書き込むことができます。ターゲットデータベースでは、ターゲットテーブルを 1 つだけ作成してください。

タイプ 2 の次元/有効な日付範囲のセッションの設定

タイプ 2 の次元/有効日付を使ったマッピングでは、新しい行と更新された行の両方を一意のプライマリキーと共に挿入します。また、ターゲット中の既存の行の更新も行います。マッピングのセッションを設定するには、以下の手順を実行します。

1. セッションプロパティで、［プロパティ］タブの［全般］設定をクリックします。［ソース行の扱い］を［データドリブン］に設定します。
2. セッションプロパティで、［マッピング］タブの［ターゲットプロパティ］設定をクリックします。Integration Service により適切に行がターゲットにロードされることを確認するには、各リレーショナルターゲットに対して［挿入］および［更新として更新］を選択します。

タイプ 3 の次元マッピングの作成

タイプ 3 の次元を使ったマッピングは、ユーザーが定義する比較に基づいてソース行のフィルタリングを行い、新しい次元であると判明したソース行だけをターゲットに挿入します。既存の次元への変更を含む行は、ターゲット内で更新されます。既存の次元を更新する際に、Integration Service によって同じ行の異なるカラム内に既存のデータが保存され、既存のデータが更新されたデータに置き換えられます。 オプションとして、Integration Service により挿入または更新される各行のタイムスタンプとしてシステムの日付が入力されます。 タイプ 3 の次元ターゲットでは、各次元にカレント次元データが含まれています。

緩やかに変化する次元テーブルを更新する場合に、テーブル中のカラムデータのカレントバージョンと前バージョンだけを保持したければ、タイプ 3 の次元を使ったマッピングを使用します。特定のカラムのカレントバージョンと前バージョンは両方とも同じ行に保存されます。

このオプションを使用すると、Designer はターゲットに追加フィールドを作成します。

- **PM_PREV_*ColumnName*。** Designer により、履歴データを必要とする各カラムに対応している*以前の*カラムが生成されます。 Integration Service により、これらのカラム内に以前のバージョンの次元データが保存されます。
- **PM_PRIMARYKEY。** Integration Service により、ターゲットに書き込まれる各行のプライマリキーが生成されます。
- **PM_EFFECT_DATE。** オプションのフィールド。 Integration Service では、システムの日付を使用して、次元の作成と更新を行う時期が示されます。

以前の値の保存

緩やかに変化する次元ウィザードによる作業中に、ユーザーは変化の検出対象のカラムを選択します。Designer は選択された各カラムに対して追加のカラムを作成し、元のカラムの名前を使って「PM_PREV_*カラム名*」という名前を付けます。Integration Service では、これらのカラムが使用され、以前の次元値が格納されます。

Integration Service が新しい次元をターゲットに書き込むときには、この*以前の*カラムは NULL のままです。Integration Service が次元を更新するたびに、対応する*以前の*カラムに既存データが書き込まれた後、更新データを元のカラムに書き込みます。 その結果、タイプ 3 の次元ターゲットでは各行にカレント次元データが含まれることになります。また、次元が変更されている場合には、各行には前バージョンの次元データも含まれることになります。

例えば、Integration Service が以下の次元をターゲットに初めて書き込む際には、*以前の*カラムの PM_PREV_STYLES は NULL のままです。

PM_PRIMARYKEY	ITEM	STYLES	PM_PREV_STYLES
6345	Sock	20	-
6346	Boot	25	-

Integration Service によりこれらの行が更新される場合、STYLES カラムの値（20 と 25）が PM_PREV_STYLES に書き込まれた後、スタイルデータが新しいスタイルデータ（14 と 31）で置き換えられます。

PM_PRIMARYKEY	ITEM	STYLES	PM_PREV_STYLES
6345	Sock	14	20
6346	Boot	31	25

キーの処理

タイプ 3 の次元マッピングでは、Integration Service によりターゲットに書き込まれる各新規行のプライマリキー値が作成され、キー値が 1 ずつ増加されます。 更新された行には元のキー値が保持されます。

有効な日付の記録

オプションとして、タイプ 3 の次元マッピングでは、Integration Service により次元の作成や更新が行われた日付を書き留めることができます。 このオプションを選択した場合、Designer により PM_EFFECT_DATE カラムが作成されます。Integration Service では、新しい行の作成や行の更新を行うたびに、このカラムにシステムの日付が入力されます。

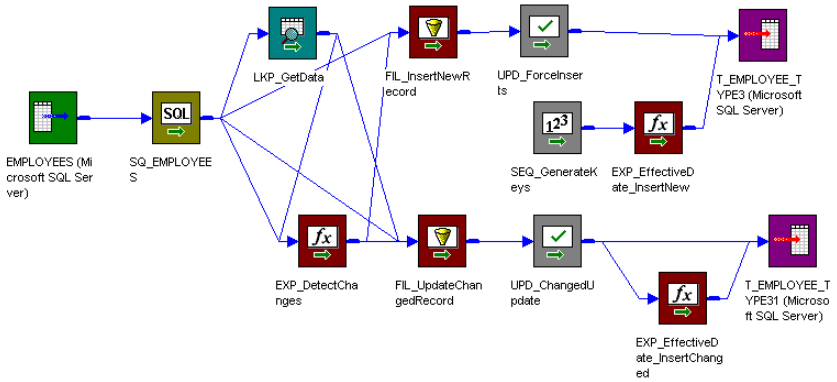
マッピングについて

タイプ 3 の次元を使ったマッピングでは、以下のことを行います。

- すべての行を選択します。
- 既存ターゲットをルックアップテーブルとしてキャッシュに格納します。
- ソース中の論理キーカラムとターゲットルックアップテーブル中の対応カラムとを比較します。
- キーカラムが一致した場合には、ソースカラムと対応ターゲットカラムとを比較します。
- 新しい行と変更された行にフラグを設定します。

- 2 つのデータフローを作成します。1 つは新しい行のデータフロー、もう 1 つは変更された行を更新するデータフローです。
- プライマリキーを生成し、オプションとして新しい行の有効日を記録します。
- ターゲットに新しい行を挿入します。
- 変更された行のそれぞれについて、変更前の値を *前バージョン*カラムに書き込み、前の値を更新された値で置換します。
- オプションとして、システム日付を使用して挿入値や更新値の有効日を記録します。
- ターゲット内の変更された行を更新します。

以下の図に、緩やかに変化する次元ウィザードのタイプ 3 の次元オプションにより作成されるマッピングを示します。



タイプ 3 の次元マッピングでは、ルックアップトランスフォーメーションおよび式トランスフォーメーションが使用され、ソースデータと既存のターゲットデータが比較されます。緩やかに変化する次元のウィザードの手順を行う際に、Integration Service を既存のターゲットデータと比較するルックアップ条件（ソースキーカラム）とソースカラムを入力します。 Designer は変更されたカラムに対して追加のカラムを作成し、過去のデータを保持します。

ターゲット中に一致するプライマリキーのないソース行では、式トランスフォーメーションが行に新規というフラグを設定します。ターゲット中に一致するプライマリキーのあるソース行では、式はユーザー定義のソースカラムとターゲットカラムを比較します。これらのカラムが一致しない場合には、式は行に変更があったというフラグを設定します。その後、このマッピングは 2 つのデータフローに分かれます。

第一のデータフローはフィルタトランスフォーメーションの FIL_InsertNewRecord を使用し、行をフィルタリングして除外します。フィルタトランスフォーメーションは新しい行だけをアップデートストラテジトランスフォーメーション UPD_ForceInserts に渡します。UPD_ForceInserts は新しい行をターゲットに挿入します。シーケンスジェネレータは各行のプライマリキーを作成します。マッピングウィザードで［有効期間の作成］オプションを選択した場合には、Designer は式トランスフォーメーション EXP_EffectiveDate_InsertNew を作成します。Integration Service では、システムの日付を使用して、新しい行を作成する時期が示されます。

第二のデータフローでは、フィルタトランスフォーメーション FIL_UpdateChangedRecord により、変更を受けた行だけがアップデートストラテジトランスフォーメーション UPD_ChangedInserts に渡されます。さらに、フィルタトランスフォーメーションは変更された行を更新します。新しいバージョンのデータをソース修飾子から取得し、既存のバージョンの次元データ（ルックアップトランスフォーメーションによって渡される）を使用して *前バージョン*カラムフィールドに書き込みます。UPD_ChangedInserts は変更された行をターゲットに挿入します。マッピングウィザードで［有効期間の作成］オプションを選択した場合には、Designer は式トランスフォーメーション EXP_EffectiveDate_InsertChanged を作成します。Integration Service では、システムの日付を使用して、行を更新する時期が示されます。

トランスフォーメーションについて

以下の表に、タイプ 3 の次元マッピングにおける各トランスフォーメーションの機能を示します。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
SQ_*SourceName*	ソース修飾子またはアプリケーションソース修飾子	マッピングウィザードで選択したソースから行をすべて選択します。
LKP_GetData	ルックアップ	既存のターゲットテーブルをキャッシュに格納します。 ソースのキーカラムとターゲットの対応キーカラムとを比較します。一致するキーが存在する場合、LKP_GetData は比較ターゲットから追加カラムデータを返します。 行をすべて EXP_DetectChanges に渡します。
EXP_DetectChanges	式	以下の式を使用して、ターゲット内に一致するキーが*ない*ソース行にフラグを設定します。ターゲットに一致キーが存在しない場合、この式は TRUE を返し、新しい行であることを明示します。`IIF（ISNULL（PM_PRIMARYKEY）,TRUE,FALSE）` NewFlag ポートに結果を書き込みます。 以下の式を用いて、ターゲット中に一致キーが*あり*、また指定カラムに変更を含むソース行にフラグを設定します。この式が TRUE を返すのは、ターゲット中に一致キーがあり（新しい行ではない）、さらにソースカラムとターゲットカラムとの違いを検出した場合だけです。`IIF（ISNULL（PM_PRIMARYKEY） AND（SourceColumnName<>PM_PREV_TargetColumnName） AND （その他の比較） TRUE,FALSE）` ChangedFlag ポートに結果を書き込みます。 すべての行を FIL_InsertNewRecord と FIL_UpdateChangedRecord に渡します。

新しい行のデータフロー

以下の表に、新しい行のデータフローを示します。 このデータフローでは、ソースの新しい行ごとにプライマリキーが作成され、オプションでロード日付が書きとめられ、行がターゲットに挿入されます。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_InsertNewRecord	フィルタ	フィルタ条件 `NewFlag` に従い、EXP_DetectChanges の結果から新規（TRUE）以外の行を除外します。新しい行を UPD_ForceInserts に渡します。
UPD_ForceInserts	アップデートストラテジ	`DD_INSERT` を使用して行をターゲットに挿入します。
SEQ_GenerateKeys	シーケンスジェネレータ	ターゲットに書き込んだ各新規行について値を生成し、各値を 1 だけインクリメントします。 マッピングウィザードで［有効期間の作成］オプションを選択した場合は、値を EXP_KeyProcessing_InsertNew に渡します。選択しない場合には、値をターゲットに渡して、PM_PRIMARYKEY カラムに書き込みます。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
EXP_KeyProcessing_InsertNew	式	マッピングウィザードで［有効期間の作成］オプションを選択した場合に限って作成されます。 生成した値をターゲットに渡し、ターゲットの PM_PRIMARYKEY カラムに書き込みます。 SYSDATE を用いて、ターゲットの PM_EFFECT_DATE カラムに書き込みを行い、行の作成日を明示します。
T_*TargetName*	ターゲット定義	ターゲットに挿入する新しい行に対するターゲット定義のインスタンスです。

既存の行を更新するデータフロー

以下の表に、既存の行のデータフローを示します。 このデータフローでは、ソースの変更された行ごとに、ターゲットの対応する行が更新され、既存データが*以前の*カラムにコピーされ、新しいデータが更新され、オプションで行の変更日が書きとめられます。

トランスフォーメーション名	トランスフォーメーションのタイプ	説明
FIL_UpdateChangedRecord	フィルタ	フィルタ条件 ChangedFlag に従い、EXP_DetectChanges の結果から変更（TRUE）以外の行を除外します。 LKP_GetData から返された値を用い、*前バージョン*カラムに書き込みを行います。 変更された行を UPD_ChangedUpdate に渡します。
UPD_ChangedUpdate	アップデートストラテジ	DD_UPDATE を用いて、ターゲットの既存の行を更新します。 マッピングウィザードで［有効期間の作成］オプションを選択した場合は、更新された行を EXP_EffectiveDate_InsertChanged に渡します。選択しない場合には、更新された行をターゲットに渡します。
EXP_EffectiveDate_InsertChanged	式	マッピングウィザードで［有効期間の作成］オプションを選択した場合に限って作成されます。 SYSDATE を用いて、ターゲットの PM_EFFECT_DATE カラムに書き込みを行い、行の更新日を明示します。
T_*ターゲット名* 2	ターゲット定義	ターゲット内の既存の行を更新するターゲット定義のインスタンスです。

タイプ 3 の次元マッピングの作成手順

タイプ 3 の次元を使ったマッピングを作成するには：

1. Mapping Designer で、［マッピング］-［ウィザード］-［緩やかに変化する次元］の順にクリックします。

2. マッピング名を指定し、［タイプ 3 次元］を選択します。［次へ］をクリックします。

   マッピング名を付けるにあたっては、m_ *マッピング名*の形にしてください。

3. マッピングで使用するソース定義を選択します。

   ［ソーステーブルの選択］リストに利用可能なソース定義がすべて表示されます。このリストには、ショートカット、フラットファイル、リレーショナル、およびアプリケーションソースが含まれます。

4. マッピングのターゲットテーブルの名前を入力します。［次へ］をクリックします。

   ターゲット定義名を付けるにあたっては、「T_*ターゲット名*」の形にしてください。

5. ルックアップ条件として使用するカラムを［ターゲットテーブルのフィールド］リストから 1 つ以上選択し、［追加］をクリックします。

   ウィザードは選択したカラムを［論理キーフィールド］リストに追加します。

   ヒント: 選択するカラムはソースのキーカラムである必要があります。

   セッションの実行時に、Integration Service により既存のターゲットデータのルックアップが実行されます。 ［論理キーフィールド］カラムが対応するターゲットカラムに一致する場合、Integration Service によりターゲットデータが返されます。

   ［論理キーフィールド］からカラムを削除するには、カラムを選択して［削除］をクリックします。

6. Integration Service で変更を比較するカラムを 1 つ以上選択し、［追加］をクリックします。

   ウィザードにより選択したカラムが［変化を比較したいフィールド］リストに追加されます。

   セッションを実行すると、Integration Service によりソース行と対応するターゲット（ルックアップ）行との間で、［変化を比較したいフィールド］リスト中のカラムが比較されます。 Integration Service により変更が検出されると、行が変更のフラグに設定されます。

   注: 前の値を保持したいカラムを選択してください。ターゲット中の前の値の保持にあたって、Designer はこのリストの各カラムについて追加のカラムを作成します。カラムの名前は「PM_PREV_*カラム名*」となります。

   リストからカラムを削除するには、カラムを選択して［削除］をクリックします。

7. ［次へ］をクリックします。

8. Integration Service で新しい行および変更された行にタイムスタンプを記録する場合、［有効な日］を選択します。

   ウィザードにより、Integration Service が比較するカラムと、過去の値を保持するカラムの名前が表示されます。

9. ［終了］をクリックします。

   注: タイプ 3 の次元を使ったマッピングでは、Designer は同一のターゲット定義の 2 つのインスタンスを使用して、2 つの別のデータフローから同じターゲットテーブルに書き込むことができます。ターゲットデータベースでは、ターゲットテーブルを 1 つだけ作成してください。

タイプ 3 の次元セッションの設定

タイプ 3 の次元マッピングでは、新しい行が挿入され、ターゲット内の既存の行が更新されます。セッションを設定する場合には、以下の手順を実行します。

1. ［プロパティ］タブの［全般オプション］設定をクリックします。 ［ソース行の扱い］を［データドリブン］に設定します。

2. ［マッピング］タブの［ターゲットプロパティ］設定をクリックします。 Integration Service により適切に行がターゲットにロードされることを確認するには、各リレーショナルターゲットに対して［挿入］および［更新として更新］を選択します。

ターゲットデータベースでのターゲットの作成

マッピングウィザードはソース定義のカラムと選択されたオプションに基づいて、新しいターゲット定義を作成します。これにより、生成されたターゲットテーブルにはマッピングに必要なカラムが確実に含まれることになります。デフォルトでは、生成されたターゲット定義には以下のようになります。

- リポジトリと同一のデータベースタイプ。
- ソース定義と同一のカラム名、定義、説明、およびキー制約
- 変更の追跡や更新に必要な追加カラム

ウィザードによってマッピングのターゲット定義が作成された後で、セッション実行前に Target Designer を使用すると、ターゲットデータベースにターゲットテーブルが作成されます。

リポジトリとターゲットデータベースのタイプが異なる場合、ターゲットを生成する前に、必ずターゲット定義のデータベースタイプを変更してターゲットデータベースに一致させてください。

セッションおよびワークフローのスケジュール設定

スタースキーマをロードするマッピングを作成した後、マッピングを実行するためのセッションおよびワークフローを作成します。Integration Service により各テーブルに正しくデータがロードされることを確認するには、作成したマッピングタイプに従って各セッションを設定します。

ファクトテーブルに最新のデータが含まれることを確認するには、次元テーブルをすべてリフレッシュしてからファクトテーブルをロードします。 このためには、ファクトセッションを実行する前に次元セッションを実行するワークフローを作成します。

次元セッションがすべて正常に完了してから Integration Service によりファクトセッションが実行されることを確認するには、以下のワークフローロジックを使用します。

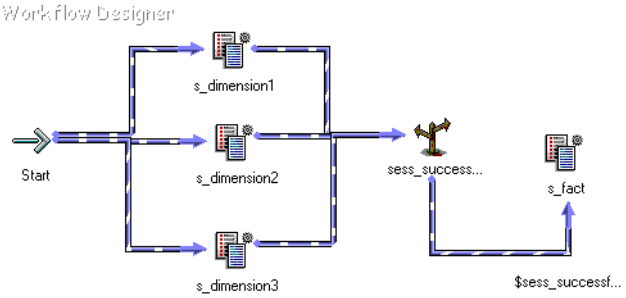
1. ワークフロー内のすべての次元セッションをファクトセッションの前に配置します。
2. すべての次元セッションをディシージョンタスクにリンクします。
3. 以下のディシージョン条件を定義します。

   $<*session_name1*>.Status = SUCCEEDED AND $<*session_name2*>.Status = SUCCEEDED AND ...
   $<*session_nameN*>.Status = SUCCEEDED
4. ディシージョンタスクのあとにファクトセッションを配置し、ディシージョンタスクをファクトセッションにリンクします。
5. ディシージョンタスクからファクトセッションへのリンク条件を以下のように設定します。

   $<*Decision_name*>.Condition = TRUE

例えば、次元テーブルにデータをロードする 3 つのセッションと、ファクトテーブルにデータをロードする 1 つのセッションを作成するとします。

以下の図に、作成できるワークフローを示します。



sess_successful ディシジョンタスクに対して、以下のディシジョン条件を定義します。

```
$s_dimension1.Status = SUCCEEDED AND $s_dimension2.Status = SUCCEEDED AND $s_dimension3.Status = SUCCEEDED
```

ディシジョンタスクから s_fact ファクトセッションへのリンクに対する以下のリンク条件を定義します。

```
$sess_successful.Condition = TRUE
```

ワークフローのパフォーマンスを改善するには、同時に次元セッションを実行します。同時に開始するよう、ワークフロー内にセッションを配置します。既存セッションの負荷が原因で、Integration Service がすべての次元セッションを同時に開始および実行できない場合、ワークフロー内で次元セッションを順次配置することにより、順次実行できます。

ワークフローの実行のたびに再ロードする必要のない次元テーブルがある場合は、そのセッションを無効にできます。

Informatica マッピングテンプレートからのマッピングの作成

Informatica マッピングテンプレートは、共通のデータウェアハウスのパターンに対応する定義済みのマッピングテンプレートです。 緩やかに変化する次元、重複の削除、増分ロードは、Informatica によりマッピングテンプレートに用意されているデータウェアハウスのパターンです。 これらのテンプレートにより、データウェアハウスの設計上の一般的な問題に対するソリューションが提供されます。 定義済みのマッピングテンプレートを使用して、データのマッピングまたは処理の方法を文書化できます。

マッピングテンプレートのインポートウィザードを使用して、Informatica マッピングテンプレートから作成する各マッピング用のマッピング名、説明、およびパラメータ値を指定します。

注: パラメータファイルを再利用する場合は、パラメータファイルとマッピングテンプレートのパラメータタイプが同じであることを確認します。

マッピングをマッピングテンプレートウィザードから作成するには、以下の手順を実行します。

1. マッピングテンプレートを選択します。
2. マッピング詳細およびパラメータ値を指定します。
3. マッピングを作成し、パラメータ値を保存します。
4. マッピングをリポジトリにインポートします。

手順 1. マッピングテンプレートを選択します。

マッピングテンプレートウィザードの最初の手順で、使用するマッピングテンプレートを選択します。

マッピングテンプレートウィザードを開始する手順

1. Designer を開いて、リポジトリに接続します。
2. インポートされたマッピングで使用するソースオブジェクトおよびターゲットオブジェクトが含まれるフォルダを開くか、これらのオブジェクトへのショートカットを開きます。
3. ［マッピング］-［マッピングテンプレートウィザード］を選択して、使用するマッピングテンプレートを選択します。

   マッピングテンプレートのインポートウィザードが表示されます。
4. 既存のマッピングテンプレートパラメータファイルを使用するには、［既存の使用］をクリックします。パラメータファイルが格納されている場所に移動し、ファイルを選択します。

   デフォルトでは、パラメータファイルは以下のディレクトリに格納されています。

   `C:\Documents and Settings\<current user>\Local Settings\Temp`

手順 2 マッピング詳細およびパラメータ値の指定

生成するマッピングごとにパラメータ値を指定することができます。

マッピング詳細およびパラメータ値を指定する手順

1. マッピングテンプレートのインポートウィザードの最初のページで、［追加］ボタンをクリックします。
2. ［MappingName］フィールドにマッピング名を入力します。
3. 必要に応じて、説明を入力します。
4. ［パラメータ/値］フィールドで［開く］ボタンをクリックします。
5. ［マッピングのパラメータ］ダイアログボックスに以下の情報を入力します。

フィールド	説明
ソーステーブル	現在の作業用ディレクトリのすべての使用可能なソース定義です。
差分抽出条件	差分抽出条件。 例えば、差分抽出条件として「UPDATE-TS>SYSDATE-1」と入力します。
ターゲットテーブルの作成	既存のターゲットテーブルを使用する代わりに、新たに作成します。 ターゲットデータベースのタイプは、ソースデータベースのタイプと同じです。
ターゲットテーブル	リポジトリフォルダのすべての使用可能なターゲット定義です。
代理キー	ターゲットテーブルのプライマリキーです。
論理キーフィールド	特定のエンティティを特定するソーステーブルのカラムです。
比較キーフィールド	ソーステーブルとターゲットテーブル間で変更された行を特定するフィールドのセットです。

   ［データベースタイプ］フィールドには、選択したデータベースのタイプが表示されます。 このオプションを変更することはできません。
6. 既存のターゲットテーブルを使用する場合は、フィールドの関連付けを設定します。 フィールドの関連付けを設定するには、［フィールドの関連付けの設定］をクリックします。
7. ［フィールドの関連付け］ダイアログボックスで、［追加］ボタンをクリックします。

8. ［ソースフィールド］リストからソースフィールドを選択します。
9. ［ターゲットフィールド］リストからターゲットフィールドを選択します。
10. ［フィールドの関連付け］ダイアログボックスで［OK］をクリックします。
11. ［マッピングのパラメータ］ダイアログボックスで［OK］をクリックします。
12. マッピングテンプレートのインポートウィザードで［次へ］をクリックします。

    マッピングテンプレートのインポートウィザードの 2 ページ目には、設定したマッピングのリストが表示されます。

手順 3. マッピングの作成とパラメータ値の保存

生成するマッピングを選択できます。選択したマッピングに対してパラメータ値を保存するには、マッピングテンプレートパラメータファイルを再利用してさらにマッピングを作成します。 ウィザードにより、選択した各マッピングのパラメータ値が、マッピングテンプレートパラメータファイルに保存されます。

注: パラメータ値はマッピングテンプレートパラメータファイルに保存します。マッピングに対してパラメータファイルを保存し、マッピング生成時にエラーが発生した場合、パラメータ設定値を取得することができます。

マッピングを作成してパラメータ値を保存する手順

1. 表示されたマッピングのリストから、生成するマッピングを選択します。
2. 複数のパイプラインを持つマッピングを 1 つ作成するには、［単一のマッピングに複数のパイプラインを作成］をクリックして、マッピング用の名前を入力します。
3. ［選択されたマッピングのパラメータ値を保存］をクリックしてから、［参照］をクリックしてパラメータファイルの保存するフォルダに移動します。

   デフォルトでは、Publish Template 関数によって、マッピングテンプレートファイルと同じ場所にマッピングテンプレートパラメータファイルが作成されます。既存ファイルの上書きか、マッピングテンプレートパラメータファイルの作成を選択することができます。
4. ［次へ］をクリックします。

   テーブル定義をエクスポートするように促されます。
5. ［はい］をクリックします。

   ［オブジェクトのエクスポート］ダイアログボックスが表示され、テーブル定義がデフォルトの場所にエクスポートされます。
6. ［オブジェクトのエクスポート］ダイアログボックスで［閉じる］をクリックします。

手順 4. マッピングのリポジトリへのインポート

マッピングは、次の手順を使用してリポジトリにインポートすることができます。

マッピングをリポジトリにインポートする手順

1. マッピングのリストを確認して、ウィザードにより正しい個数のマッピングが生成されていることを確認します。

   デフォルトでは、マッピングに対してワークフローおよびセッションを作成するオプションが選択されています。
2. ワークフローの生成ウィザードを起動するには、［次へ］をクリックします。ワークフローの生成ウィザードの詳細については、「ワークフロー生成ウィザードの使用」 (ページ 144)を参照してください。

   マッピングに対してワークフローおよびセッションを作成するオプションを無効にするには、［終了］をクリックします。

   生成されたマッピングは、選択したリポジトリフォルダのマッピングノードに表示されます。

付録 A

データタイプリファレンス

この付録では、以下の項目について説明します。

データタイプリファレンスの概要

マッピング作成時に、ソーステーブルからのデータ読み込み、変換、ターゲットテーブルへの書き込みを行う、Integration Service に対する指示セットを作成します。 Integration Service は、マッピング内の最初のトランスフォーメーションから始まるマッピング内のデータフロー、およびマッピング内の各ポートに割り当てられているデータタイプに基づいて、データを変換します。

Designer では 2 種類のデータタイプを表示します。

- **ネイティブデータタイプ。** ソースおよびターゲットデータベース、フラットファイル、あるいは ERP システムに固有のデータタイプです。 ネイティブデータタイプは、Source Analyzer および Target Designer に表示されます。ネイティブデータタイプは、Mapping Designer のソース定義やターゲット定義、および Mapplet Designer のソース定義でも表示されます。

- **トランスフォーメーションデータタイプ。**トランスフォーメーションで表示されるデータタイプのセットです。 これは ANSI SQL-92 汎用データタイプに基づく内部データタイプで、Integration Service がプラットフォーム間でデータを移動するために使用します。 トランスフォーメーションデータタイプは汎用型であるため、ソースプラットフォームとターゲットプラットフォームで異なるプラットフォームを使用できます。例えば、Oracle ソースから情報を読み込んで、Sybase ターゲットに書き込むことができます。同様に、フラットファイルから情報を読み込んで、Microsoft SQL Server データベースに書き込むこともできます。トランスフォーメーションデータタイプはマッピングのすべてのトランスフォーメーション内に表示されます。

Integration Service は、ソースデータからの読み込み時に、ネイティブデータタイプを同等のトランスフォーメーションデータタイプに変換してからデータのトランスフォームを実行します。Integration Service は、ターゲットへの書き込み時に、トランスフォーメーションデータタイプを同等のネイティブデータタイプに変換します。

マルチバイト文字セットを指定した場合、データタイプは 3 バイトまで文字を格納する追加スペースをデータベース内に割り当てます。

トランスフォーメーションデータ型

以下の表に、トランスフォーメーションのデータ型を示します。

データ型	サイズ（バイト数）	説明
Array	文字数に制限はありません。	複合データ型。 複雑なソースとターゲットを持つ配列を使用できます。
Bigint	8 バイト	-9,223,372,036,854,775,808～9,223,372,036,854,775,807 精度 19、位取り 0。 整数値。
Binary	精度	1～104,857,600 バイト COBOL またはフラットファイルソースでバイナリデータは使用できません。 フラットファイルソースでバイナリデータは使用できません。
Date/Time	16 バイト	西暦 0001 年 1 月 1 日～西暦 9999 年 12 月 31 日 精度 29、位取り 9 （精度はナノ秒まで） 日付値と時刻値が結合されたデータ型。

データ型	サイズ（バイト数）	説明
Decimal	8 バイト（高精度がオフまたは 28 より大きい場合） 8 バイト（高精度がオフまたは 38 より大きい場合） 16 バイト（精度が 18 以下で高精度がオンの場合） 20 バイト（精度が 18 より大きく、28 以下の場合） 24 バイト（精度が 28 より大きく、38 以下の場合）	宣言された精度と位取りを持つ 10 進型の値。位取りは、精度以下にする必要があります。 精度 1 から 28 桁、位取り 0 から 28 最大 38 桁の精度をサポートするトランスフォーメーションでは、精度が 1〜38 桁、スケールが 0〜38 になります。 最大 28 桁の精度をサポートするトランスフォーメーションでは、精度が 1〜28 桁、スケールが 0〜28 になります。 この最大桁数を上回る精度を指定した場合、データ統合サービスは 10 進値を高精度モードの倍精度浮動小数点数に変換します。 最大 38 桁の精度をサポートするトランスフォーメーションでは、精度が 1〜38 桁、スケールが 0〜38 になります。 最大 28 桁の精度をサポートするトランスフォーメーションでは、精度が 1〜28 桁、スケールが 0〜28 になります。 この最大桁数を上回る精度を指定した場合、データ統合サービスは 10 進値を高精度モードの倍精度浮動小数点数に変換します。
Double	8 バイト	倍精度の浮動小数点数値。 精度と位取りは編集できます。スケールは精度以下にする必要があります。
Integer	4 バイト	-2,147,483,648〜2,147,483,647 精度 10、位取り 0 整数値。
Map	文字数に制限はありません。	複合データ型。 複雑なソースとターゲットを持つマップを使用できます。
Nstring	Unicode モード: （精度+ 1）* 2 ASCII モード: 精度+ 1	1〜104,857,600 文字 固定長または可変長文字列。
Ntext	Unicode モード: （精度+ 1）* 2 ASCII モード: 精度+ 1	1〜104,857,600 文字 固定長または可変長文字列。
Real	8 バイト	精度 7、位取り 0 倍精度の浮動小数点数値。
Small Integer	4 バイト	-32,768 および 32,767 精度 5、位取り 0 整数値。
String	Unicode モード: （精度+ 1）* 2 ASCII モード: 精度+ 1	1〜104,857,600 文字 固定長または可変長文字列。

データ型	サイズ（バイト数）	説明
構造	文字数に制限はありません。	複合データ型 複雑なソースとターゲットを持つ構造を使用できます。
Text	Unicode モード: （精度+ 1）* 2 ASCII モード: 精度+ 1	1〜104,857,600 文字 固定長または可変長文字列。
timestampWithTZ	40 バイト	西暦 1947 年 8 月 1 日〜西暦 2040 年 12 月 31 日 -12:00〜+14:00 精度 36、スケール 9。 （精度はナノ秒まで） Timestamp with Time Zone データ型では、次のタイムゾーン地域はサポートされません。 - AFRICA_CAIRO - AFRICA_MONROVIA - エジプト - AMERICA_MONTREAL

Integer データ型

ソースからターゲットに整数データを渡して、整数データでトランスフォーメーションを実行できます。トランスフォーメーション言語は、Bigint、Integer、および Small Integer のデータ型をサポートします。

ソースからターゲットに整数データを渡して、整数データでトランスフォーメーションを実行できます。トランスフォーメーション言語は、Bigint および Integer のデータ型をサポートします。

トランスフォーメーションの integer データ型は正確な値を表します。

計算における整数値

計算に整数値を使用する場合、PowerCenter 統合サービスでは、計算の前に整数値が浮動小数点数に変換されることがあります。 例えば、MOD(12.00, 5)を評価する場合、PowerCenter 統合サービスは、除算を実行する前に整数値「5」を浮動少数点数に変換します。 PowerCenter 統合サービスでは、高精度が有効にされているかどうかに応じて、整数値が Double 値または Decimal 値に変換されます。

計算に整数値を使用する場合、データ統合サービスでは、計算の前に整数値が浮動小数点数に変換されることがあります。 例えば、MOD(12.00, 5)を評価する場合、データ統合サービスは、除算を実行する前に整数値「5」を浮動少数点数に変換します。 データ統合サービスでは、高精度が有効にされているかどうかに応じて、整数値が Double 値または Decimal 値に変換されます。

計算に整数値を使用する場合、データ統合サービスでは、計算の前に整数値が浮動小数点数に変換されることがあります。 例えば、MOD(12.00, 5)を評価する場合、データ統合サービスは、除算を実行する前に整数値「5」を浮動少数点数に変換します。 データ統合サービスでは、高精度が有効にされているかどうかに応じて、整数値が Double 値または Decimal 値に変換されます。

PowerCenter 統合サービスは、次の算術演算の整数値を変換します。

データ統合サービスは、次の算術演算の整数値を変換します。

データ統合サービスは、次の算術演算の整数値を変換します。

算術演算	高精度の無効	高精度の有効
小数点を使用できない関数と計算。 たとえば、整数の加算、減算、乗算、および CUME、MOVINGSUM、SUM などの関数。	変換なし[1]	Decimal
小数点を使用できる非科学関数と計算。 たとえば、整数の除算、および AVG、MEDIAN、PERCENTILE などの関数。	Double	Decimal
すべての科学関数、EXP 関数、LN 関数、LOG 関数、POWER 関数、および SQRT 関数。	Double	ダブル

1. 計算が範囲外の結果を算出した場合、統合サービスは行エラーを書き込みます。

トランスフォーメーションの Double データ型は最大 15 桁の精度をサポートし、Bigint データ型は最大 19 桁の精度をサポートします。 このため、精度が 15 桁より大きい Bigint 値となる計算では、精度の損失が発生することがあります。

例えば、式トランスフォーメーションには以下の計算が含まれています。

```
POWER( BIGINTVAL, EXPVAL )
```

PowerCenter 統合サービスでは、計算を開始する前に、POWER 関数への入力を Double 値に変換します。 BIGINTVAL ポートに Bigint 値 9223372036854775807 が含まれている場合、PowerCenter 統合サービスはこの値を 9.22337203685478e+18 に変換し、最後の 4 桁の精度が失われます。 EXPVAL ポートに値 1.0 が含まれており、結果ポートが Bigint の場合、計算結果の 9223372036854780000 が最大 Bigint 値を超えているため、行エラーが作成されます。

結果が Decimal 値となる計算で Integer データ型を使用し、高精度を有効にする場合、PowerCenter 統合サービスでは、整数値が Decimal 値に変換されます。

データ統合サービスでは、計算を開始する前に、POWER 関数への入力を Double 値に変換します。 BIGINTVAL ポートに Bigint 値 9223372036854775807 が含まれている場合、データ統合サービスはこの値を 9.22337203685478e+18 に変換し、最後の 4 桁の精度が失われます。 EXPVAL ポートに値 1.0 が含まれており、結果ポートが Bigint の場合、計算結果の 9223372036854780000 が最大 Bigint 値を超えているため、行エラーが作成されます。

結果が Decimal 値となる計算で Integer データ型を使用し、高精度を有効にする場合、データ統合サービスでは、整数値が Decimal 値に変換されます。

データ統合サービスでは、計算を開始する前に、POWER 関数への入力を Double 値に変換します。 BIGINTVAL ポートに Bigint 値 9223372036854775807 が含まれている場合、データ統合サービスはこの値を 9.22337203685478e+18 に変換し、最後の 4 桁の精度が失われます。 EXPVAL ポートに値 1.0 が含まれており、結果ポートが Bigint の場合、計算結果の 9223372036854780000 が最大 Bigint 値を超えているため、行エラーが作成されます。

結果が Decimal 値となる計算で Integer データ型を使用し、高精度を有効にする場合、データ統合サービスでは、整数値が Decimal 値に変換されます。

最大 28 桁精度の Decimal データ型をサポートするトランスフォーメーションの場合、結果で高精度モードの 28 桁を超える値が出ない限り計算の精度は失われません。 この場合、PowerCenter 統合サービスは Double 型で結果を保存します。ポート精度が 28 桁以下で、結果が高精度モードの 28 桁より大きい場合、PowerCenter 統合サービスは行を拒否します。

最大 28 桁精度の Decimal データ型をサポートするトランスフォーメーションの場合、結果で高精度モードの 28 桁を超える値が出ない限り計算の精度は失われません。 この場合、データ統合サービスは結果を Double 型

として保存します。ポート精度が 28 桁以下で、結果が高精度モードの 28 桁より大きい場合、データ統合サービスは行を拒否します。

最大 38 桁精度の Decimal データ型をサポートするトランスフォーメーションの場合、結果で高精度モードの 38 桁を超える値が出ない限り計算の精度は失われません。 この場合、データ統合サービスは結果を Double 型として保存します。ポート精度が 38 桁以下で、結果が高精度モードの 38 桁より大きい場合、データ統合サービスは行を拒否します。

最大 38 桁精度の Decimal データ型をサポートするトランスフォーメーションの場合、結果で高精度モードの 38 桁を超える値が出ない限り計算の精度は失われません。 この場合、データ統合サービスは結果を Double 型として保存します。ポート精度が 38 桁以下で、結果が高精度モードの 38 桁より大きい場合、データ統合サービスは行を拒否します。

式の整数定数

Integration Service では、計算結果が整数の場合でも、式の定数を浮動小数点数として解釈します。例えば、式 INTVALUE+1000 では、高精度が有効でない場合、Integration Service は整数値「1000」を Double 値に変換します。 高精度が有効な場合は、値「1000」を 10 進数値に変換します。値 1000 を整数値として処理するには、Integer データ型で変数ポートを作成して定数を保持し、式を変更して 2 つのポートを追加します。

NaN 値

NaN（非数）とは、通常、特に浮動小数点数の計算で、無効な入力オペランドでの演算の結果として返される値のことです。例えば、ゼロをゼロで除算する演算の場合、結果として NaN が返されます。

NaN の表示は、オペレーティングシステムおよびプログラミング言語によって異なります。例えば、以下のリストのような NaN の表示があります。

```
nan
NaN
NaN%
NAN
NaNQ
NaNS
qNaN
sNaN
1.#SNAN
1.#QNAN
```

Integration Service は、Win64EMT プラットフォームでは QNAN 値を 1.#QNAN に変換します。1.#QNAN は、NaN の有効な表示です。

文字列値の整数値への変換

統合サービスは、文字列値の整数値への暗黙の変換を実行する場合、最初の数値以外の文字でデータを切り捨てます。例えば、値「9,000,000,000,000,000,000.777」を含む文字列ポートを Bigint ポートにリンクするとします。統合サービスは、文字列を bigint 値 9,000,000,000,000,000,000 に変換します。

フラットファイルへの整数値の書き込み

整数値を固定長のフラットファイルに書き込む場合、ファイルライターではデータが範囲内であることが確認されません。例えば、ターゲットカラムのフィールド長が少なくとも 13 の場合、ファイルライターによりターゲットの Integer カラムに結果 3,000,000,000 が書き込まれます。ファイル writer では、結果が整数値の有効範囲外であるため、行は拒否されません。

Binary データ型

マッピングにバイナリデータが含まれている場合、統合サービスがデータをソースからターゲットに移動するために必要なメモリを割り当てることができるように、トランスフォーメーションの Binary データ型の精度を設定します。

COBOL またはフラットファイルソースで Binary データ型は使用できません。

フラットファイルソースで Binary データ型は使用できません。

Date/Time データ型

Date/Time データ型は、グレゴリオ暦の西暦 1 年から西暦 9999 年までの年を扱います。西暦 9999 より先の年はエラーになります。

Date/Time データ型は、ナノ秒単位の精度で日付をサポートします。データ型の精度は 29 で、位取りは 9 です。ネイティブデータ型の中には、精度が低いものがあります。日付の値を含むソースをインポートする際、インポートプロセスはソースカラムから正しい精度をインポートします。例えば、Microsoft SQL Server の Datetime データ型の精度は 23 で、位取りは 3 です。日時の値を含む Microsoft SQL Server ソースをインポートする場合、マッピングソースの日付カラムは精度が 23 で、位取りが 3 になります。

Integration Service は、マッピングソースに指定されている精度で、ソースから日時の値を読み込みます。Integration Service は、日時の値を変換する際、29 桁までの精度をサポートします。例えば、ミリ秒の精度で日時の値をインポートする場合、式トランスフォーメーションの ADD_TO_DATE 関数を使用して日付にナノ秒を追加することができます。

より短い精度をサポートするターゲットカラムに日付/時刻値を書き込むと、Integration Service はターゲットカラムの精度まで値を切り詰めます。より長い精度をサポートするターゲットカラムに日付/時刻値を書き込むと、Integration Service は日時の値でサポートされていない部分にゼロを挿入します。

Decimal および Double データ型

10 進型データおよび倍精度浮動小数点数型データをソースからターゲットに渡して、10 進型データおよび倍精度浮動小数点数型データに対するトランスフォーメーションを実行することができます。

トランスフォーメーション言語では以下のデータ型をサポートしています。

Decimal

精度 1 から 28 桁、位取り 0 から 28。位取りが精度より大きい 10 進型の値、および負の精度を持つ 10 進型の値は使用できません。トランスフォーメーションには 10 進型のデータ型に割り当てられている範囲がすべて表示されますが、PowerCenter 統合サービスは最大 28 桁までの精度しかサポートしません。

最大 38 桁の精度をサポートするトランスフォーメーションでは、精度が 1～38 桁、スケールが 0～38 になります。 最大 28 桁の精度をサポートするトランスフォーメーションでは、精度が 1～28 桁、スケールが 0～28 になります。 精度より大きなスケールや負の精度を持つ 10 進型の値は使用できません。 トランスフォーメーションには 10 進型のデータ型に割り当てた任意の範囲が表示されますが、データ統合サービスでは、トランスフォーメーションに応じて、最大 38 桁または 28 桁の精度のみサポートします。

高精度を有効にし、ポート精度が 38 桁または 28 桁（トランスフォーメーションに依存）より高い場合、データ統合サービスは結果を Double 型として保存します。

高精度を有効にし、ポート精度が 28 桁より高い場合、PowerCenter 統合サービスは結果を Double 型として保存します。

最大 38 桁の精度をサポートするトランスフォーメーションでは、精度が 1～38 桁、スケールが 0～38 になります。 最大 28 桁の精度をサポートするトランスフォーメーションでは、精度が 1～28 桁、スケールが 0～28 になります。 精度より大きなスケールや負の精度を持つ 10 進型の値は使用できません。 トラン

スフォーメーションには 10 進型のデータ型に割り当てた任意の範囲が表示されますが、データ統合サービスでは、トランスフォーメーションに応じて、最大 38 桁または 28 桁の精度のみサポートします。

高精度を有効にし、ポート精度が 38 桁または 28 桁（トランスフォーメーションに依存）より高い場合、データ統合サービスは結果を Double 型として保存します。

Double

倍精度の浮動小数点数値。

精度と位取りは編集できます。スケールは精度以下にする必要があります。

計算における Decimal 値と Double 値

以下の表に、PowerCenter 統合サービスで高精度設定に基づいて 10 進値がどのように処理されるかを示します。

以下の表に、データ統合サービスで高精度設定に基づいて 10 進値がどのように処理されるかを示します。

以下の表に、データ統合サービスで高精度設定に基づいて 10 進値がどのように処理されるかを示します。

ポートのデータ型	精度	高精度オフ	高精度オン
Decimal	0〜15	Decimal	Decimal
Decimal	15〜28 最大 38 桁の精度を持つ Decimal データ型をサポートするトランスフォーメーションの場合、15〜38。 最大 28 桁の精度を持つ Decimal データ型をサポートするトランスフォーメーションの場合、15〜28。 最大 38 桁の精度を持つ Decimal データ型をサポートするトランスフォーメーションの場合、15〜38。 最大 28 桁の精度の Decimal データ型をサポートするトランスフォーメーションの場合、15〜 28。	Double	Decimal
Decimal	28 より大きい 最大 38 桁の精度を持つ Decimal データ型をサポートするトランスフォーメーションの場合、38 より大きい。 最大 28 桁の精度を持つ Decimal データ型をサポートするトランスフォーメーションの場合、28 より大きい。 最大 38 桁の精度を持つ Decimal データ型をサポートするトランスフォーメーションの場合、38 より大きい。 最大 28 桁の精度を持つ Decimal データ型をサポートするトランスフォーメーションの場合、28 より大きい。	Double	Double

高精度を有効にすると、PowerCenter 統合サービスは式関数の数値定数を Decimal 型に変換します。高精度を有効にしないと、PowerCenter 統合サービスは数値定数を Double 型に変換します。

高精度を有効にすると、データ統合サービスは式関数の数値定数を Decimal 型に変換します。高精度を有効にしないと、データ統合サービスは数値定数を Double 型に変換します。

高精度を有効にすると、データ統合サービスは式関数の数値定数を Decimal 型に変換します。高精度を有効にしないと、データ統合サービスは数値定数を Double 型に変換します。

数値の最大精度を、トランスフォーメーションに応じて、28 または 38 桁よりも大きくすることができます。トランスフォーメーション関数を使用して計算またはトランスフォーメーションを実行する前に、大きい数字を切り詰めるか、丸めます。

最大 38 桁の精度の Decimal データ型をサポートするトランスフォーメーションでは、Decimal データ型を使用し、高精度を有効にして、最大 38 桁の精度を保証します。

最大 38 桁の精度の Decimal データ型をサポートするトランスフォーメーションでは、Decimal データ型を使用し、高精度を有効にして、最大 38 桁の精度を保証します。

最大 28 桁の精度の Decimal データ型をサポートするトランスフォーメーションでは、Decimal データ型を使用し、高精度を有効にして、最大 28 桁の精度を保証します。

ルックアップ条件や結合条件など、等価条件で使用するデータに倍精度浮動小数点数型のデータ型を使用しないでください。

ルックアップ条件や結合条件など、等価条件で使用するデータに倍精度浮動小数点数型のデータ型を使用しないでください。

ルックアップ条件や結合条件など、等価条件で使用するデータに倍精度浮動小数点数型のデータ型を使用しないでください。

Double 値の丸め処理方式

システムの実行時ライブラリと、データベースが double データタイプ計算を行うコンピュータシステムの違いにより、予想外の結果になる場合があります。double データタイプは IEEE 794 標準に準拠します。データベースクライアントライブラリへの変更、データベースの異なるバージョン、システム実行時ライブラリへの変更は、算術的に同等な値のバイナリ表現に影響します。また、多くのシステム実行時ライブラリは偶数丸め方法や対称算術法を実装しています。偶数丸め方法は、数字が次の上下の数の中間にある場合、偶数の最小位ビットの最も近い値に丸められます。例えば、偶数丸め方法では、0.125 は 0.12 に丸められます。対称算術法では末尾桁が 5 以上の場合は次の桁に丸められます。例えば、対称算術法では 0.125 は 0.13 に丸められ、0.124 は 0.12 に丸められます。

プラットフォームの違いによる計算結果への影響を少なくするため、Integration Service は有効桁数 15 桁の Double データタイプの値を格納しています。 例えば、Windows で計算すると、数値 1234567890.1234567890 が返され、同じ計算を UNIX で実行すると、1234567890.1234569999 が返される場合、Integration Service はこの数値を 1234567890.1234600000 に変換します。

String データ型

トランスフォーメーションデータ型には、次の string データ型が含まれます。

- Nstring
- Ntext
- String
- Text

Nstring、Ntext、String、および Text データ型は 104,857,600 文字までの同精度をサポートしますが、統合サービスは、ソースからターゲットに文字列データを移動する場合には String を、テキストデータを移動する場合には Text を使用します。一部のデータベースではテキストデータと文字列データの格納方法が異なるため、Integration Service ではこの 2 つの文字データタイプを区別する必要があります。 ソース修飾子で String を表示する場合は、ターゲットカラムを String に設定します。同様に、Text を表示する場合は、ターゲットカラムを Text、Long、または Long Varchar（ソースデータベースに応じたデータタイプ）に設定します。

String および Text データ型は 104,857,600 文字までの同精度をサポートしますが、統合サービスは、ソースからターゲットに文字列データを移動する場合には String を、テキストデータを移動する場合には Text を使用します。一部のデータベースではテキストデータと文字列データの格納方法が異なるため、Integration Service ではこの 2 つの文字データタイプを区別する必要があります。通常、Char や Varchar などのサイズの小さい string データ型は、トランスフォーメーション内で String として表示されますが、Text、Long、および Long Varchar などのサイズの大きな文字列データ型は Text として表示されます。

一般には、サイズの小さい文字列データ型（Char や Varchar など）はソース修飾子、ルックアップおよびストアドプロシージャトランスフォーメーションでは String として表示され、サイズの大きなテキストデータ型（Text、Long、Long Varchar など）はソース修飾子トランスフォーメーションでは Text として表示されます。

トランスフォーメーション内では、Nstring、Ntext、String、および Text を相互に入れ替えて使用できます。しかし、ソース修飾子、ルックアップ、およびストアドプロシージャトランスフォーメーションではターゲットデータ型は一致していなければなりません。データベースドライバは、データが正確に渡されるように、string データ型とトランスフォーメーションデータ型を一致させる必要があります。例えば、ルックアップテーブルの Nchar は、ルックアップトランスフォーメーションの Nstring と一致しなければなりません。

トランスフォーメーション内では、String および Text を相互に入れ替えて使用できます。ただし、ルックアップトランスフォーメーションでは、対象のデータ型が一致している必要があります。データベースドライバは、データが正確に渡されるように、string データ型とトランスフォーメーションデータ型を一致させる必要があります。例えば、ルックアップテーブルの Varchar は、ルックアップトランスフォーメーションの String と一致しなければなりません。

IBM DB2 データタイプとトランスフォーメーションデータタイプ

以下の表で、IBM DB2 データタイプとトランスフォーメーションデータタイプを比較します。

データタイプ	範囲	トランスフォーメーション	範囲
Bigint	-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807 精度 19、位取り 0
Blob	1 - 2,147,483,647 バイト	バイナリ	1 - 104,857,600 バイト
Char（L）	1 - 254 文字	文字列	1 - 104,857,600 文字
ビットデータの Char	1 - 254 バイト	バイナリ	1 - 104,857,600 バイト
Clob	1 - 2,447,483,647 バイト	テキスト	1 - 104,857,600 文字
日付	西暦 0001 年 - 9999 年 精度 19、位取り 0（精度は日付まで）	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Dbclob	1GB まで	Ntext	1 - 104,857,600 文字
Decimal(P,S)	精度 1 - 31、位取り 0 - 31	Decimal	精度 1 - 28、位取り 0 - 28

データタイプ	範囲	トランスフォーメーション	範囲
Float	精度 1 - 15	Double	精度 15
Graphic	1 - 127 バイト	Nstring	1 - 104,857,600 文字
Integer	-2,147,483,648 - 2,147,483,647	Integer	-2,147,483,648 - 2,147,483,647 精度 10、位取り 0
Long Varchar	1 - 32,700 文字	文字列	1 - 104,857,600 文字
Long Vargraphic	16,350 バイトまで	Ntext	1 - 104,857,600 文字
Numeric（P,S）	精度 1 - 31、位取り 0 - 31	Decimal	精度 1 - 28、位取り 0 - 28
Smallint	-32,768 - 32,767	Small Integer	精度 5、位取り 0
時間	24 時間制の時刻 精度 19、位取り 0 （精度は秒まで）	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
タイムスタンプ	26 バイト 精度 26、位取り 6 （精度はマイクロ秒まで）	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Varchar	4,000 文字まで	文字列	1 - 104,857,600 文字
ビットデータの Varchar	4,000 バイトまで	バイナリ	1 - 104,857,600 バイト
Vargraphic	16,336 バイトまで	文字列	1 - 104,857,600 文字

Informix データタイプとトランスフォーメーションデータタイプ

以下の表で、Informix データタイプと、ソース修飾子、ルックアップ、およびストアドプロシージャトランスフォーメーションに表示されるトランスフォーメーションデータタイプを比較します。 トランスフォーメーションデータタイプはマッピング内で変更することができ、Integration Service がそのデータを変換します。 ただし、ソース修飾子、ルックアップ、および Stored Procedure トランスフォーメーション内のデータタイプとソースデータタイプは、必ず一致させる必要があります。データタイプが一致しない場合、マッピングは無効になります。

注: ソース修飾子、ルックアップ、および Stored Procedure トランスフォーメーションで Text データタイプとして表示されるのは、Informix の Text データタイプだけです。ただし、トランスフォーメーションデータタイプの Text と String は相互に入れ替えて使用できます。

Informix	範囲	トランスフォーメーション	範囲
Byte	1 バイト以上。最大サイズなし。	バイナリ	1〜104,857,600 バイト
Char（L）	1〜32,767 文字	String	1〜104,857,600 文字
日付	西暦 0001 年 1 月 1 日〜西暦 9999 年 12 月 31 日 精度 19、位取り 0 （精度は日付まで）	日付/時刻	西暦 0001 年 1 月 1 日〜西暦 9999 年 12 月 31 日 （精度は秒まで）
Datetime year to fraction	西暦 0001 年 1 月 1 日〜西暦 9999 年 12 月 31 日 精度 21〜25、位取り 1〜5 （精度はミリ秒まで）	日付/時刻	西暦 0001 年 1 月 1 日〜西暦 9999 年 12 月 31 日 （精度は秒まで）
Decimal(P,S)	精度 1〜32、位取り 0〜32	Decimal	精度 1〜28、位取り 0〜28
Float（P）	精度 1〜14	Double	精度 15
Int8	-9,223,372,036,854,775,807 〜 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808〜 9,223,372,036,854,775,807 精度 19、位取り 0
Integer	-2,147,483,647〜2,147,483,647	Integer	-2,147,483,648〜2,147,483,647 精度 10、位取り 0
Money（P,S）	精度 1〜32、位取り 2	Decimal	精度 1〜28、位取り 0〜28
Nvarchar	1〜255 文字	Nstring	1〜104,857,600 文字
Serial（L）	-2,147,483,647〜2,147,483,647	Integer	-2,147,483,648〜2,147,483,647 精度 10、位取り 0
Smallfloat	精度 1〜7、位取り 0	Real	精度 7、位取り 0
Smallint	-32,767〜32,767	Small Integer	精度 5、位取り 0
Text	最大長 2e31	テキスト	1〜104,857,600 文字
Varchar（M,R）	1〜255 文字	String	1〜104,857,600 文字

データタイプシノニム

以下の表で、Informix シノニムとトランスフォーメーションデータタイプを比較します。

シノニム	トランスフォーメーション
Bigint (Extended Parallel Server)	Bigint
文字（L）	文字列
Character Varying（m,r）	文字列
Dec	Decimal
倍精度	Double
Int	Integer
Numeric	Decimal
Real	Real

Microsoft SQL Server データタイプとトランスフォーメーションデータタイプ

以下の表で、Microsoft SQL Server データタイプとトランスフォーメーションデータタイプを比較します。

Microsoft SQL	範囲	トランスフォーメーション	範囲
Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 精度 19、位取り 0
Binary（L）	1～8,000 バイト	バイナリ	1～104,857,600 バイト
Bit	1 ビット	String	1～104,857,600 文字
Char（L）	1～8,000 文字	String	1～104,857,600 文字
Datetime	西暦 1753 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 精度 23、位取り 3 （精度は 3.33 ミリ秒まで）	Date/Time	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Datetime2	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 精度 27、位取り 7 （精度は 100 ナノ秒まで）	Date/Time	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）

Microsoft SQL	範囲	トランスフォーメーション	範囲
Decimal(P,S)	精度 1〜38、位取り 0〜38	Decimal	精度 1〜28、位取り 0〜28
Float	-1.79E+308〜1.79E+308	Double	精度 15
Image	1〜2,147,483,647 バイト	バイナリ	1〜104,857,600 バイト
Int	-2,147,483,648〜2,147,483,647	Integer	-2,147,483,648〜2,147,483,647 精度 10、位取り 0
通貨	-922,337,203,685,477.5807〜 922,337,203,685,477.5807	Decimal	精度 1〜28、位取り 0〜28
Nchar	1〜4,000 文字	Nstring	1〜104,857,600 文字
Ntext	1〜1,073,741,823 文字	Ntext	1〜104,857,600 文字
Numeric（P,S）	精度 1〜38、位取り 0〜38	Decimal	精度 1〜28、位取り 0〜28
Numeric Identity	1〜9,999	Integer	-2,147,483,648〜2,147,483,647 精度 10、位取り 0
Nvarchar	1〜4,000 文字	Nstring	1〜104,857,600 文字
Real	-3.40E+38〜3.40E+38	Real	精度 7、位取り 0
Smalldatetime	西暦 1900 年 1 月 1 日 - 西暦 2079 年 6 月 6 日 精度 19、位取り 0 （精度は分まで）	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日（精度はナノ秒まで）
Smallint	-32,768〜32,768	Small Integer	精度 5、位取り 0
Smallmoney	-214,748.3648〜214,748.3647	Decimal	精度 1〜28、位取り 0〜28
Sysname	1〜128 文字	Nstring	1〜104,857,600 文字
Text	1〜2,147,483,647 文字	テキスト	1〜104,857,600 文字
タイムスタンプ	8 バイト	バイナリ	1〜104,857,600 バイト
Tinyint	0〜255	Small Integer	精度 5、位取り 0
Varbinary（L）	1〜8,000 バイト	バイナリ	1〜104,857,600 バイト
Varchar（L）	1〜8,000 文字	String	1〜104,857,600 文字

データタイプシノニム

以下の表で、Microsoft SQL Server シノニムとトランスフォーメーションデータタイプを比較します。

シノニム	トランスフォーメーション
Binary Varying	バイナリ
文字	文字列
Character Varying	文字列
Dec	Decimal
倍精度	Double
Integer	Integer

uniqueidentifier データタイプ

PowerCenter は、Microsoft SQL Server uniqueidentifier データタイプを 38 文字の Microsoft SQL Server Varchar データタイプとしてインポートします。

Oracle データタイプとトランスフォーメーションデータタイプ

以下の表で、Oracle データ型とトランスフォーメーションデータ型を比較します。

Oracle	範囲	トランスフォーメーション	範囲
Blob	4GB まで	バイナリ	1～104,857,600 バイト
Char（L）	1～2,000 バイト	String	1～104,857,600 文字
Clob	4GB まで	テキスト	1～104,857,600 文字
日付	紀元前 4712 年 1 月 1 日 - 西暦 4712 年 12 月 31 日 精度 19、位取り 0	Date/Time	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Long	2GB まで	Text	1～104,857,600 文字 マッピングに Long データが含まれている場合、Integration Service はそれをトランスフォーメーションの String データ型に変換し、104,857,600 文字に切り詰めます。
Long Raw	2GB まで	バイナリ	1～104,857,600 バイト

Oracle	範囲	トランスフォーメーション	範囲
Nchar	1～2,000 バイト	Nstring	1～104,857,600 文字
Nclob	4GB まで	Ntext	1～104,857,600 文字
番号	精度 1～38	ダブル	精度 15
Number (P,S)	精度 1～38、 位取り 0～38	Decimal	精度 1～28、位取り 0～28
Nvarchar2	1～4,000 バイト	Nstring	1～104,857,600 文字
Raw（L）	1～2,000 バイト	バイナリ	1～104,857,600 バイト
タイムスタンプ	紀元前 4712 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 精度 19～29、位取り 0～9 （精度はナノ秒まで）	Date/Time	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Varchar（L）	1～4,000 バイト	String	1～104,857,600 文字
Varchar2（L）	1～4,000 バイト	String	1～104,857,600 文字
CLOB ストレージに基づく XMLType	4GB まで	テキスト	1～104,857,600 文字

Number(P,S)データタイプ

PowerCenter は、負の位取りを持つ Oracle Number(P,S)値をサポートします。 ただし、精度 28 より大きい位取り、または負の精度を持つ Number(P,S)値はサポートしていません。

負の位取りの Oracle Number をインポートした場合、Mapping Designer には Decimal データタイプとして表示されますが、Integration Service では Double データタイプに変換します。

Char データタイプ、Varchar データタイプ、Clob データタイプ

Integration Service では、Unicode データ移動モードを使用する場合、Oracle データベースのカラムに設定した長さのセマンティクスに応じて、Char カラム、Varchar カラム、および Clob カラムの精度が読み込まれます。バイトセマンティクスを使用してカラム長を決定する場合は、Integration Service では精度がバイト数として読み込まれます。Char セマンティクスを使用する場合は、Integration Service では精度が文字数として読み込まれます。

SAP HANA データタイプとトランスフォーメーションデータタイプ

以下の表に、SAP HANA データタイプおよびトランスフォーメーションデータタイプの比較を示します。

SAP HANA データ型	範囲	トランスフォーメーションデータ型	範囲
英数字	精度 1 - 127	Nstring	1〜104,857,600 文字
Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 精度 19、位取り 0
バイナリ	バイナリデータのバイトの格納に使用	バイナリ	1〜104,857,600 バイト
Blob	2GB まで	バイナリ	1〜104,857,600 バイト
Clob	2GB まで	Text	1〜104,857,600 文字
日付	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 精度 10、位取り 0	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Decimal（精度, 位取り）または Dec (p, s)	精度 1 - 34	Decimal	精度 1〜28、位取り 0〜28
ダブル	単精度 64 ビット浮動小数点数を指定します	Double	精度 15
Float	精度 1 - 53	Double	精度 15
Integer	-2,147,483,648〜2,147,483,647	Integer	-2,147,483,648〜2,147,483,647 精度 10、位取り 0
NClob	2GB まで	Ntext	1〜104,857,600 文字
Nvarchar	精度 1 - 5000	Nstring	1〜104,857,600 文字
Real	単精度 32 ビット浮動小数点数を指定します	Real	精度 7、位取り 0
Seconddate	0001-01-01 00:00:01 - 9999-12-31 24:00:00	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Shorttext	可変長文字列を指定します。このデータ型ではテキスト検索および文字列検索機能がサポートされます	Nstring	1〜104,857,600 文字
Smalldecimal	精度 1 - 16	Decimal	精度 1〜28、位取り 0〜28

SAP HANA データ型	範囲	トランスフォーメーションデータ型	範囲
Smallint	-32,768〜32,767	Small Integer	精度 5、位取り 0
Text	可変長文字列を指定します。このデータ型ではテキスト検索機能がサポートされます	テキスト	1〜104,857,600 文字
Time	24 時間制の時刻	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
タイムスタンプ	0001-01-01 00:00:00.0000000 - 9999-12-31 23:59:59.9999999	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Tinyint	0〜255	Small Integer	精度 5、位取り 0
Varchar	精度 1 - 5000	String	1〜104,857,600 文字
Varbinary	1〜5000 バイト	バイナリ	1〜104,857,600 バイト

Sybase データタイプとトランスフォーメーションデータタイプ

以下の表で、Sybase ASE および IQ のデータタイプとトランスフォーメーションデータタイプを比較します。

Sybase	範囲	トランスフォーメーション	範囲
Bigint	-9,223,372,036,854,775,808〜9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 精度 19、位取り 0
バイナリ（n）	- Sybase ASE:サーバーの論理ページサイズ - Sybase IQ:1 - 255 バイト	Binary	1〜104,857,600 バイト
Bit	0 または 1	String	1〜104,857,600 文字
Char（n）	ASE サーバ論理ページサイズ	String	1〜104,857,600 文字
Date	西暦 0001 年 1 月 1 日〜西暦 9999 年 12 月 31 日 精度 10、位取り 0	Date/Time	西暦 0001 年 1 月 1 日〜西暦 9999 年 12 月 31 日 （精度はナノ秒まで）

Sybase	範囲	トランスフォーメーション	範囲
Datetime	西暦 1753 年 1 月 1 日～西暦 9999 年 12 月 31 日 精度 23、位取り 3 （精度は 1/300 秒まで）	Date/Time	西暦 0001 年 1 月 1 日～西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Decimal(P,S)	精度 1～38、位取り 0～38	Decimal	精度 1～28、位取り 0～28
Float	マシン依存	Double	精度 15
Image	1～2,147,483,647 バイト	Binary	1～32768 バイト
Int	-2,147,483,648～2,147,483,647	Integer	-2,147,483,648～2,147,483,647 精度 10、位取り 0
通貨	-922,337,203,685,477.5808～922,337,203,685,477.5807	Decimal	精度 1～28、位取り 0～28
Nchar (n)	ASE サーバ論理ページサイズ	Nstring	1～104,857,600 文字
Numeric (P,S)	精度 1～38、位取り 0～38	Decimal	精度 1～28、位取り 0～28
Nvarchar (n)	ASE サーバ論理ページサイズ	Nstring	1～104,857,600 文字
Real	マシン依存	Real	精度 7、位取り 0
Smalldatetime	西暦 1900 年 1 月 1 日～西暦 2079 年 6 月 6 日 精度 19、位取り 0 （精度は分まで）	Date/Time	西暦 0001 年 1 月 1 日～西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Smallint	-32,768～32,767	Small Integer	精度 5、位取り 0
Smallmoney	-214,748.3648～214,748.3647	Decimal	精度 1～28、位取り 0～28
Text	1～2,147,483,647 文字	Text	1～16384 文字。 バルクモードで 1～32768 文字。
時間	12:00:00AM～11:59:59:999PM 精度 8、位取り 0	Date/Time	西暦 0001 年 1 月 1 日～西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
タイムスタンプ	8 バイト	Binary	1～104,857,600 バイト
Timestamp (Sybase IQ)	0001-01-01 00:00:00.000000～9999-12-31 23:59:59.999999 精度 26、位取り 6	Date/Time	西暦 0001 年 1 月 1 日～西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Tinyint	0～255	Small Integer	精度 5、位取り 0

Sybase	範囲	トランスフォーメーション	範囲
Unichar	ASE サーバ論理ページサイズ	Nstring	1〜104,857,600 文字
Univarchar	ASE サーバ論理ページサイズ	Nstring	1〜104,857,600 文字
Varbinary (n)	- Sybase ASE:サーバーの論理ページサイズ - Sybase IQ:1 - 32,767 バイト	Binary	1〜104,857,600 バイト
Varchar（n）	ASE サーバ論理ページサイズ	String	1〜104,857,600 文字

以下のデータタイプは Sybase IQ ではサポートされません。

- イメージ
- Nchar (n)
- Nvarchar (n)
- テキスト
- Unichar
- Univarchar

データタイプシノニム

以下の表で、Sybase シノニムとトランスフォーメーションデータタイプを比較します。

シノニム	トランスフォーメーション
Char Varying	文字列
文字（L）	文字列
Character Varying	文字列
Dec	Decimal
Integer	Small Integer
National Char	Nstring
National Char Varying	Nstring
National Character	Nstring
National Character Varying	Nstring
Nchar Varying	Nstring

Sybase IQ のバイナリデータタイプと Varbinary データタイプ

Sybase IQ の ASE_Binary_Display データベースオプションが OFF に設定されている場合、Sybase IQ ではバイナリデータタイプおよび Varbinary データタイプをサポートします。 デフォルトでは、このオプションは ON になっています。 詳細については、Sybase のマニュアルを参照してください。

Teradata データタイプとトランスフォーメーションデータタイプ

以下の表で、Teradata データタイプとトランスフォーメーションデータタイプを比較します。

Teradata	範囲	トランスフォーメーション	範囲
Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 精度 19、位取り 0
Byte	1〜64,000 バイト	バイナリ	1〜104,857,600 バイト
Byteint	-128〜127	Small Integer	精度 5、位取り 0
文字	1〜64,000 バイト	String	1〜104,857,600 文字
日付	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 精度 19、位取り 0	Date/Time	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Decimal	精度 1〜18、位取り 0〜18	Decimal	精度 1〜28、位取り 0〜28
Float	-2.226E+308〜1.797E+308	Double	精度 15
Integer	-2,147,483,648〜2,147,483,647	Integer	-2,147,483,648〜2,147,483,647 精度 10、位取り 0
Smallint	-32768〜32768	Small Integer	精度 5、位取り 0
Time	00:00:00.000000〜23:59:61.999999 精度 8、位取り 0	Date/Time	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度は秒まで）
タイムスタンプ	1〜19 文字 精度 19〜26、位取り 0〜6	Date/Time	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 （精度はナノ秒まで）
Varbyte	1〜64,000 バイト	バイナリ	1〜104,857,600 バイト
Varchar	1〜64,000 バイト	String	1〜104,857,600 文字

データタイプシノニム

以下の表で、Teradata シノニムとトランスフォーメーションデータタイプを比較します。

シノニム	トランスフォーメーション
Double Precision	Double
Numeric	Decimal
Real	Double

ODBC データタイプとトランスフォーメーションデータタイプ

以下の表で、Microsoft Access や Excel などの ODBC データタイプとトランスフォーメーションデータタイプを比較します。

データタイプ	トランスフォーメーション	範囲
Bigint	Bigint	-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807 精度 19、位取り 0
バイナリ	バイナリ	1 - 104,857,600 バイト
Bit	文字列	1 - 104,857,600 文字
Char（L）	文字列	1 - 104,857,600 文字
日付	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日（精度はナノ秒まで）
Decimal(P,S)	Decimal	精度 1 - 28、位取り 0 - 28
Double	Double	精度 15
Float	Double	精度 15
Integer	Integer	-2,147,483,648 - 2,147,483,647 精度 10、位取り 0
Long Varbinary	バイナリ	1 - 104,857,600 バイト
Nchar	Nstring	1 - 104,857,600 文字
Nvarchar	Nstring	1 - 104,857,600 文字
Ntext	Ntext	1 - 104,857,600 文字

データタイプ	トランスフォーメーション	範囲
Numeric	Decimal	精度 1 - 28、位取り 0 - 28
Real	Real	精度 7、位取り 0
Smallint	Smallint	精度 5、位取り 0
テキスト	テキスト	1 - 104,857,600 文字
時間	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日（精度はナノ秒まで）
タイムスタンプ	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日（精度はナノ秒まで）
Tinyint	Small Integer	精度 5、位取り 0
Varbinary	バイナリ	1 - 104,857,600 バイト
Varchar（L）	文字列	1 - 104,857,600 文字

注: Integration Service が Unicode データ移動モードで実行されている場合、ODBC データタイプ用に指定したカラム精度によって、Integration Service で読み書きされる文字数が決定されます。

COBOL データタイプとトランスフォーメーションデータタイプ

以下の表で、COBOL データタイプとトランスフォーメーションデータタイプを比較します。

COBOL	トランスフォーメーション	範囲
N 文字列	文字列	1 - 104,857,600 文字
番号	Decimal	精度 1 - 28、位取り 0 - 28
文字列	文字列	1 - 104,857,600 文字

フラットファイルデータタイプとトランスフォーメーションデータタイプ

以下の表で、フラットファイルデータタイプとトランスフォーメーションデータタイプを比較します。

フラットファイル	トランスフォーメーション	範囲
Bigint	Bigint	精度 19 桁、位取り 0
日時	日付/時刻	西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日（精度はナノ秒まで）
Double	Double	15 桁精度
Integer	Integer	-2,147,483,648 - 2,147,483,647
Nstring	Nstring	1 - 104,857,600 文字
番号	Decimal	精度 1 - 28、位取り 0 - 28
文字列	文字列	1 - 104,857,600 文字

Integration Service がフラットファイルの数値カラムから数値以外のデータを読み込むと、その行を削除し、セッションログにメッセージを書き込みます。 また、Integration Service がフラットファイルの日時カラムから日時以外のデータを読み込むと、その行を削除し、セッションログにメッセージを書き込みます。

Number データタイプ

フラットファイルの場合、Integration Service はファイルに表示される Number データタイプのデータを読み込みます。 10 進精度および位取りを指定する場合、式トランスフォーメーションを使用して数値ポートを 10 進ポートに変換します。また、セッションの高精度を有効にします。

例えば、フラットファイルに Price という 5 桁の数値フィールドがあり、この Price を小数点以下 2 桁で出力する場合、以下の式を設定します。

```
PRICE / 100
```

この式によって、次の例のように小数点を 2 桁左に移動します。

番号	戻り値
34500	345.00
12340	123.40
23450	234.50

XML データタイプとトランスフォーメーションデータタイプ

PowerCenter は、W3C が 2001 年 5 月 2 日の勧告で指定した XML データタイプをすべてサポートしています。XML データタイプの詳細については、http://www.w3.org/TR/xmlschema-2 で XML データタイプの W3C 仕様を参照してください。

ターゲットにデータを渡す場合、Integration Service がターゲット XML ファイルに正しく書き込めるような正しい形式のデータであることを確認します。

ソースおよびターゲット定義の XML データタイプは変更できます。 Transformation Developer で Midstream XML データタイプを変更することができます。XML スキーマからインポートする場合は、XML データタイプを変更することはできません。また、マッピング内の XML ソースのトランスフォーメーションデータタイプを変更することはできません。

データの変換

次の方法によって、データのデータタイプを変換することができます。

- 異なるデータタイプのポート間でデータを受け渡す（ポート対ポート変換）。
- トランスフォーメーション関数を使ってデータを変換する。
- トランスフォーメーション算術演算子を使ってデータを変換する。

ポートからポートへのデータ変換

Integration Service は、ポートのデータタイプに基づいてデータを変換します。 データがポートを通過するたびに、Integration Service はポートに割り当てられているデータタイプを参照し、必要であればデータを変換します。

同じ数値データタイプのポート間でデータを渡し、データがトランスフォーメーション間で転送される場合、Integration Service はデータの転送先のポートの位取りと精度にデータを変換しません。例えば、マッピングで 2 つのトランスフォーメーション間でデータを転送するとします。データを精度 5 の Decimal ポートから精度 4 の Decimal ポートに渡す場合、Integration Service は値を内部に格納し、データを丸めません。

異なるデータのポート間でデータを受け渡すことで、データを変換することができます。例えば、文字列を Integer ポートに渡すことで、数値に変換することができます。

Integration Service はポート対ポート変換を、トランスフォーメーション間、およびデータフロー内の最後のトランスフォーメーションとターゲットとの間で行います。

以下の表に、Integration Service で実行されるポート対ポート変換を示します。

データタイプ	Bigint	Integer、Small Integer	Decimal	Double、Real	String、Text	Nstring、Ntext	**日付/時刻**	**バイナリ**
Bigint	いいえ	はい	はい	はい	はい	はい	いいえ	いいえ
Integer、Small Integer	はい	いいえ	はい	はい	はい	はい	いいえ	いいえ
Decimal	はい	はい	いいえ	はい	はい	はい	いいえ	いいえ
Double、Real	はい	はい	はい	いいえ	はい	はい	いいえ	いいえ
String、Text	はい	はい	はい	はい	はい	はい	はい	いいえ

データタイプ	Bigint	Integer、Small Integer	Decimal	Double、Real	String、Text	Nstring、Ntext	日付/時刻	バイナリ
Nstring、Ntext	はい	はい	はい	はい	はい	はい	はい	いいえ
日付/時刻	いいえ	いいえ	いいえ	いいえ	はい	はい	はい	いいえ
バイナリ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい

文字列から日付への変換

文字列を日付/時刻ポートに渡すことで、文字列を日付の値に変換することができます。ただし、文字列は［Datetime フォーマット文字列］セッションプロパティで定義されている日付フォーマットで格納される必要があります。

文字列から数値への変換

Integration Service は、ASCII 数値フォーマットに基づいて文字列を数値に変換します。

以下の表に、文字列を文字列ポートから数値ポートに渡したときに、Integration Service で文字列がどのように数値に変換されるかの例を示します。

文字列	Decimal への変換	Double、Real	Bigint、Integer、Small Integer
‘18e-3’	18	0.018	18
‘18-e-3abc’	18	0.018	18
‘123abc’	123	123	123
‘A123cb’	0	0	0
‘abc’	0	0	0

付録 B

Web ブラウザの設定

Web ブラウザの設定

PowerCenter Client ブラウザ機能によっては、Internet Explorer で次のオプションを設定する必要があります。

Adobe Flash Player **プラグイン**

Metadata Manager では、PowerCenter パラメータファイルをアップロードし、データリネージを表示するために、Adobe Flash Player プラグインのバージョン 9 以降が必要です。 PowerCenter パラメータファイルをアップロードするには、または Metadata Manager や Designer からデータリネージ分析を実行するには、Flash Player プラグインをダウンロードして Web ブラウザにインストールします。 以下の Web サイトから Flash Player プラグインを入手します。www.adobe.com/products/flashplayer

スクリプトと ActiveX

Microsoft Internet Explorer で次のコントロールを有効にします。

- アクティブスクリプト
- プログラム的なクリップボード アクセスの許可
- ActiveX コントロールとプラグインの実行
- スクリプトを実行しても安全だとマークされている ActiveX コントロールのスクリプトの実行

コントロールを設定するには、**[ツール]** > **[インターネットオプション]** > **[セキュリティ]** > **[レベルのカスタマイズ]** をクリックします。

注: PowerCenter Client のスタートページを表示するようにスクリプトおよび ActiveX コントロールを設定してください。

索引

F

I

M

N

O

P

Q

R

S

T

U

W

X

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

ね

の

は

ひ

ふ

へ

ほ

ま

む

め