

|                                |    |
|--------------------------------|----|
| XML ガイド - 著作権.....             | 3  |
| はじめに.....                      | 7  |
| Informatica のリソース.....         | 7  |
| XML の概念.....                   | 9  |
| XML の概念の概要.....                | 9  |
| XML ファイル.....                  | 9  |
| DTD ファイル.....                  | 12 |
| XML スキーマファイル.....              | 14 |
| XML メタデータのタイプ.....             | 15 |
| カーディナリティ.....                  | 17 |
| XML の単純型および複合型.....            | 20 |
| Any 型要素および属性.....              | 23 |
| コンポーネントグループ.....               | 26 |
| XML パス.....                    | 28 |
| コードページ.....                    | 28 |
| PowerCenter での XML の使用.....    | 29 |
| PowerCenter での XML の使用の概要..... | 29 |
| XML メタデータのインポート.....           | 30 |
| XML ビューについて.....               | 35 |
| 階層リレーションについて.....              | 37 |
| エンティティリレーションについて.....          | 40 |
| 循環参照に関する作業.....                | 46 |
| ビュー行について.....                  | 47 |
| 列のピボット化.....                   | 49 |
| XML ソースに関する作業.....             | 51 |
| XML ソースに関する作業の概要.....          | 51 |
| XML ソース定義のインポート.....           | 52 |
| XML ビューに関する作業.....             | 54 |
| エンティティリレーションの生成.....           | 55 |
| 階層リレーションの生成.....               | 56 |
| カスタム XML ビューの作成.....           | 56 |

|  |     |
|--|-----|
| XML 定義の同期.....                                     | 58  |
| XML ソース定義のプロパティの編集.....                            | 59  |
| リポジトリ定義からの XML 定義の作成.....                          | 60  |
| XML ソースのトラブルシューティング.....                           | 61  |
| XML エディタの使用.....                                   | 62  |
| XML エディタの使用の概要.....                                | 62  |
| ビューの作成と編集.....                                     | 65  |
| XPath クエリの述部の作成.....                               | 70  |
| ビューリレーションの維持.....                                  | 74  |
| スキーマコンポーネントの表示.....                                | 76  |
| XML ビューオプションの設定.....                               | 79  |
| XML エディタを使用する作業のトラブルシューティング.....                   | 87  |
| XML ターゲットに関する作業.....                               | 87  |
| XML ターゲットに関する作業の概要.....                            | 87  |
| XML ファイルからの XML ターゲット定義のインポート.....                 | 88  |
| XML ソース定義からのターゲットの作成.....                          | 89  |
| XML ターゲット定義のプロパティの編集.....                          | 89  |
| XML ターゲットの検証.....                                  | 91  |
| マッピングにおける XML ターゲットの使用.....                        | 93  |
| XML ターゲットのトラブルシューティング.....                         | 96  |
| XML Source Qualifier トランスフォーメーション.....             | 97  |
| XML ソース修飾子トランスフォーメーションの概要.....                     | 97  |
| マッピングへの XML Source Qualifier の追加.....              | 98  |
| XML ソース修飾子トランスフォーメーションの編集.....                     | 99  |
| マッピングでの XML Source Qualifier の使用.....              | 101 |
| XML Source Qualifier トランスフォーメーションのトラブルシューティング..... | 104 |
| Midstream XML トランスフォーメーション.....                    | 105 |
| Midstream XML トランスフォーメーションの概要.....                 | 105 |
| XML Parser トランスフォーメーション.....                       | 105 |
| XML Generator トランスフォーメーション.....                    | 110 |
| Midstream XML トランスフォーメーションの作成.....                 | 110 |
| Midstream XML 定義の同期.....                           | 110 |
| Midstream XML トランスフォーメーションのプロパティの編集.....           | 111 |
| パススルーポートの生成.....                                   | 115 |
| Midstream XML トランスフォーメーションのトラブルシューティング.....        | 116 |
| XML データ型リファレンス.....                                | 116 |

|                                |     |
|--------------------------------|-----|
| XML データ型とトランスフォーメーションデータ型..... | 116 |
| XPath クエリ関数の参照.....            | 119 |
| XPath クエリ関数の概要.....            | 119 |
| 関数のクイックリファレンス.....             | 120 |
| 倫理.....                        | 121 |
| ceiling.....                   | 122 |
| concat.....                    | 123 |
| contains.....                  | 124 |
| false.....                     | 125 |
| floor.....                     | 125 |
| lang.....                      | 126 |
| normalize-space.....           | 126 |
| not.....                       | 127 |
| number.....                    | 128 |
| round.....                     | 129 |
| starts-with.....               | 129 |
| string.....                    | 130 |
| string-length.....             | 131 |
| substring.....                 | 132 |
| substring-after.....           | 133 |
| substring-before.....          | 134 |
| translate.....                 | 135 |
| true.....                      | 136 |

## XML ガイド - 著作権

本ソフトウェアおよびマニュアルは、使用および開示の制限を定めた個別の使用許諾契約のもとでのみ提供されています。本マニュアルのいかなる部分も、いかなる手段（電子的複製、写真複製、録音など）によっても、Informatica LLC の事前の承諾なしに複製または転載することは禁じられています。

Informatica、Informatica ロゴ、および PowerCenter は、米国およびその他の国における Informatica LLC の商標または登録商標です。Informatica の商標の最新リストは、Web (<https://www.informatica.com/trademarks.html>) にあります。その他の企業名および製品名は、それぞれの企業の商標または登録商標です。

本ソフトウェアまたはドキュメントの一部は、次のサードパーティが有する著作権に従います（ただし、これらに限定されません）。Copyright DataDirect Technologies. All rights reserved. Copyright (C) Sun Microsystems. All rights reserved. Copyright (C) RSA Security Inc. All rights reserved. Copyright (C) Ordinal Technology Corp. All rights reserved. Copyright (C) Aandacht c.v. All rights reserved. Copyright Genivia, Inc. All rights reserved. Copyright Isomorphic Software. All rights reserved. Copyright (C) Meta Integration Technology, Inc. All rights reserved. Copyright (C) Intalio. All rights reserved.

reserved. Copyright (C) Oracle. All rights reserved. Copyright (C) Adobe Systems Incorporated. All rights reserved. Copyright (C) DataArt, Inc. All rights reserved. Copyright (C) ComponentSource. All rights reserved. Copyright (C) Microsoft Corporation. All rights reserved. Copyright (C) Rogue Wave Software, Inc. All rights reserved. Copyright (C) Teradata Corporation. All rights reserved. Copyright (C) Yahoo! Inc. All rights reserved. Copyright (C) Glyph & Cog, LLC. All rights reserved. Copyright (C) Thinkmap, Inc. All rights reserved. Copyright (C) Clearpace Software Limited. All rights reserved. Copyright (C) Information Builders, Inc. All rights reserved. Copyright (C) OSS Nokalva, Inc. All rights reserved. Copyright Edifecs, Inc. All rights reserved. Copyright Cleo Communications, Inc. All rights reserved. Copyright (C) International Organization for Standardization 1986. All rights reserved. Copyright (C) ej-technologies GmbH. All rights reserved. Copyright (C) Jaspersoft Corporation. All rights reserved. Copyright (C) International Business Machines Corporation. All rights reserved. Copyright (C) yWorks GmbH. All rights reserved. Copyright (C) Lucent Technologies. All rights reserved. Copyright (C) University of Toronto. All rights reserved. Copyright (C) Daniel Veillard. All rights reserved. Copyright (C) Unicode, Inc. Copyright IBM Corp. All rights reserved. Copyright (C) MicroQuill Software Publishing, Inc. All rights reserved. Copyright (C) PassMark Software Pty Ltd. All rights reserved. Copyright (C) LogiXML, Inc. All rights reserved. Copyright (C) 2003-2010 Lorenzi Davide, All rights reserved. Copyright (C) Red Hat, Inc. All rights reserved. Copyright (C) The Board of Trustees of the Leland Stanford Junior University. All rights reserved. Copyright (C) EMC Corporation. All rights reserved. Copyright (C) Flexera Software. All rights reserved. Copyright (C) Jinfonet Software. All rights reserved. Copyright (C) Apple Inc. All rights reserved. Copyright (C) Telerik Inc. All rights reserved. Copyright (C) BEA Systems. All rights reserved. Copyright (C) PDFlib GmbH. All rights reserved. Copyright (C) Orientation in Objects GmbH. All rights reserved. Copyright (C) Tanuki Software, Ltd. All rights reserved. Copyright (C) Ricebridge. All rights reserved. Copyright (C) Sencha, Inc. All rights reserved. Copyright (C) Scalable Systems, Inc. All rights reserved. Copyright (C) jQWidgets. All rights reserved. Copyright (C) Tableau Software, Inc. All rights reserved. Copyright (C) MaxMind, Inc. All rights reserved. Copyright (C) TMate Software s.r.o. All rights reserved. Copyright (C) MapR Technologies Inc. All rights reserved. Copyright (C) Amazon Corporate LLC. All rights reserved. Copyright (C) Highsoft. All rights reserved. Copyright (C) Python Software Foundation. All rights reserved. Copyright (C) BeOpen.com. All rights reserved. Copyright (C) CNRI. All rights reserved.

本製品には、Apache Software Foundation (<http://www.apache.org/>) によって開発されたソフトウェア、およびさまざまなバージョンの Apache License（まとめて「License」と呼んでいます）の下に許諾された他のソフトウェアが含まれます。これらのライセンスのコピーは、<http://www.apache.org/licenses/> で入手できます。適用法にて要求されないか書面にて合意されない限り、ライセンスの下に配布されるソフトウェアは「現状のまま」で配布され、明示的あるいは黙示的かを問わず、いかなる種類の保証や条件も付帯することはありません。ライセンス下での許諾および制限を定める具体的文言については、ライセンスを参照してください。

本製品には、Mozilla (<http://www.mozilla.org/>) によって開発されたソフトウェア、ソフトウェア Copyright (c) The JBoss Group, LLC, all rights reserved、ソフトウェア Copyright (c) 1999-2006 by Bruno Lowagie and Paulo Soares および GNU Lesser General Public License Agreement のさまざまなバージョン (<http://www.gnu.org/licenses/lgpl.html> で参照できる場合がある) に基づいて許諾されたその他のソフトウェアが含まれています。資料は、Informatica が無料で提供しており、一切の保証を伴わない「現状渡し」で提供されるものとし、Informatica LLC は市場性および特定の目的の適合性の黙示の保証などを含めて、一切の明示的及び黙示的保証の責任を負いません。

製品には、ワシントン大学、カリフォルニア大学アーバイン校、およびバンダービルト大学の Douglas C.Schmidt および同氏のリサーチグループが著作権を持つ ACE (TM) および TAO (TM) ソフトウェアが含まれています。Copyright (C) 1993-2006, All rights reserved.

本製品には、OpenSSL Toolkit を使用するために OpenSSL Project が開発したソフトウェア (copyright The OpenSSL Project.All Rights Reserved) が含まれています。また、このソフトウェアの再配布は、<http://www.openssl.org> および <http://www.openssl.org/source/license.html> にある使用条件に従います。

本製品には、Curl ソフトウェア Copyright 1996-2013, Daniel Stenberg, <daniel@haxx.se>が含まれます。All rights reserved. 本ソフトウェアに関する許諾および制限は、<http://curl.haxx.se/docs/copyright.html> にある使用条件に従います。すべてのコピーに上記の著作権情報とこの許諾情報が記載されている場合、目的に応じて、本ソフトウェアの使用、コピー、変更、ならびに配布が有償または無償で許可されます。

本製品には、MetaStuff, Ltd.のソフトウェアが含まれます。Copyright 2001-2005 (C) MetaStuff, Ltd. All Rights Reserved.本ソフトウェアに関する許諾および制限は、<http://www.dom4j.org/license.html> にある使用条件に従います。

本製品には、Per Bothner のソフトウェアが含まれます。Copyright (C) 1996-2006.All rights reserved. お客様がこのようなソフトウェアを使用するための権利は、ライセンスで規定されています。<http://www.gnu.org/software/kawa/Software-License.html> を参照してください。

本製品には、OSSP UUID ソフトウェアが含まれます。Copyright (C) 2002 Ralf S. Engelschall, Copyright (C) 2002 The OSSP Project Copyright (C) 2002 Cable & Wireless Deutschland.本ソフトウェアに関する許諾および制限は、<http://www.opensource.org/licenses/mit-license.php> にある使用条件に従います。

本製品には、Boost (<http://www.boost.org/>) によって開発されたソフトウェア、または Boost ソフトウェアライセンスの下で開発されたソフトウェアが含まれます。本ソフトウェアに関する許諾および制限は、[http://www.boost.org/LICENSE\\_1\\_0.txt](http://www.boost.org/LICENSE_1_0.txt) にある使用条件に従います。

本製品には、University of Cambridge のが含まれます。Copyright (C) 1997-2007.本ソフトウェアに関する許諾および制限は、<http://www.pcre.org/license.txt> にある使用条件に従います。

本製品には、The Eclipse Foundation のソフトウェアが含まれます。Copyright (C) 2007.All rights reserved. 本ソフトウェアに関する許諾および制限は、<http://www.eclipse.org/org/documents/epl-v10.php> および <http://www.eclipse.org/org/documents/edl-v10.php> にある使用条件に従います。

本製品には、<http://www.tcl.tk/software/tcltk/license.html>、<http://www.bosrup.com/web/overlib/?License>、<http://www.stlport.org/doc/license.html>、<http://www.asm.ow2.org/license.html>、<http://www.cryptix.org/LICENSE.TXT>、<http://hsqldb.org/web/hsqLicense.html>、<http://httpunit.sourceforge.net/doc/license.html>、<http://jung.sourceforge.net/license.txt>、[http://www.gzip.org/zlib/zlib\\_license.html](http://www.gzip.org/zlib/zlib_license.html)、<http://www.openldap.org/software/release/license.html>、<http://www.libssh2.org>、<http://slf4j.org/license.html>、<http://www.sente.ch/software/OpenSourceLicense.html>、<http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3-license-agreement>、<http://antlr.org/license.html>、<http://aopalliance.sourceforge.net/>、<http://www.bouncycastle.org/licence.html>、<http://www.jgraph.com/jgraphdownload.html>、<http://www.jcraft.com/jsch/LICENSE.txt>、[http://jotm.objectweb.org/bsd\\_license.html](http://jotm.objectweb.org/bsd_license.html) に基づいて許諾されたソフトウェアが含まれています。<http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231>、<http://www.slf4j.org/license.html>、<http://nanoxml.sourceforge.net/orig/copyright.html>、<http://www.json.org/license.html>、<http://forge.ow2.org/projects/javaservice/>、<http://www.postgresql.org/about/licence.html>、<http://www.sqlite.org/copyright.html>、<http://www.tcl.tk/software/tcltk/>

license.html、<http://www.jaxen.org/faq.html>、<http://www.jdom.org/docs/faq.html>、<http://www.slf4j.org/license.html>、<http://www.iodbc.org/dataspace/iodbc/wiki/iODBC/License>、<http://www.keplerproject.org/md5/license.html>、<http://www.toedter.com/en/jcalendar/license.html>、<http://www.edankert.com/bounce/index.html>、<http://www.net-snmp.org/about/license.html>、<http://www.openmdx.org/#FAQ>、[http://www.php.net/license/3\\_01.txt](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 Walnes, 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。

免責: 本文書は、一切の保証を伴わない「現状渡し」で提供されるものとし、Informatica LLC は他社の権利の非侵害、市場性および特定の目的への適合性の黙示の保証などを含めて、一切の明示的および黙示的保証の責任を負いません。Informatica LLC では、本ソフトウェアまたはドキュメントに誤りのないことを保証していません。本ソフトウェアまたはドキュメントに記載されている情報には、技術的に不正確な記述や誤植が含まれる場合があります。本ソフトウェアまたはドキュメントの情報は、予告なしに変更されることがあります。

## 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を提供され、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](mailto: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。

## はじめに

『XML ガイド』は、データウェアハウス環境で XML を使用する開発者およびソフトウェアエンジニアを対象としています。『XML ガイド』の読者は、XML の概念、オペレーティングシステム、プラットフォーム、および使用する環境内のメインフレームのシステムについて理解する必要があります。また、使用するアプリケーションのインタフェース条件についても理解していることを前提としています。

## 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](mailto: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](mailto: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](mailto: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。



# XML の概念

## XML の概念の概要

Extensible Markup Language (XML)は、共通の情報フォーマットの作成や、アプリケーション間やインターネット上でのフォーマットおよびデータの共有を行うための柔軟な方法です。

以下の種類のファイルから PowerCenter に XML 定義をインポートすることができます。

- **XML ファイル**。XML ファイルには、データおよびメタデータが含まれています。XML ファイルは文書型定義(DTD)ファイルまたは XML スキーマ定義(XSD)を参照して検証できます。
- **DTD ファイル**。DTD ファイルでは、XML ファイルの要素タイプ、属性、エンティティを定義します。DTD ファイルには、XML 文書の構造に対する制約がありますが、データは何も含まれていません。
- **XML スキーマ**。XML スキーマは、要素、属性、型定義を定義します。スキーマには単純型と複合型が指定されています。単純型とは、テキストを含む XML の要素または属性のことです。複合型とは、その他の要素や属性を含む XML 要素のことです。

スキーマは、スキーマを通じて参照できる要素、属性、置き換えグループをサポートします。置き換えグループを使用して、XML インスタンス文書内で 1 つの要素を別の要素に置き換えることができます。スキーマは、このほか要素、複合型、要素グループおよび属性グループの継承をサポートします。

## XML ファイル

XML ファイルには XML ファイルのデータを識別するタグが含まれますが、データのフォーマットは含まれません。XML ファイルの基本コンポーネントは要素（エレメント）です。XML 要素は、要素の開始タグ、要素の内容、および要素の終了タグで構成されています。XML ファイルには、ファイルの先頭と末尾にタグを 1 つ付けて必ずルート要素を定義します。ファイル中の要素はすべてルート要素で囲まれます。

XML ファイルは、階層型データベースをモデル化したものです。XML 階層内での要素の位置は、他の要素との関係を示しています。各要素の内部に複数の子要素を指定することや、複数の要素が他の要素から特徴を継承することが可能です。

たとえば、次の XML ファイルでは book を記述します。

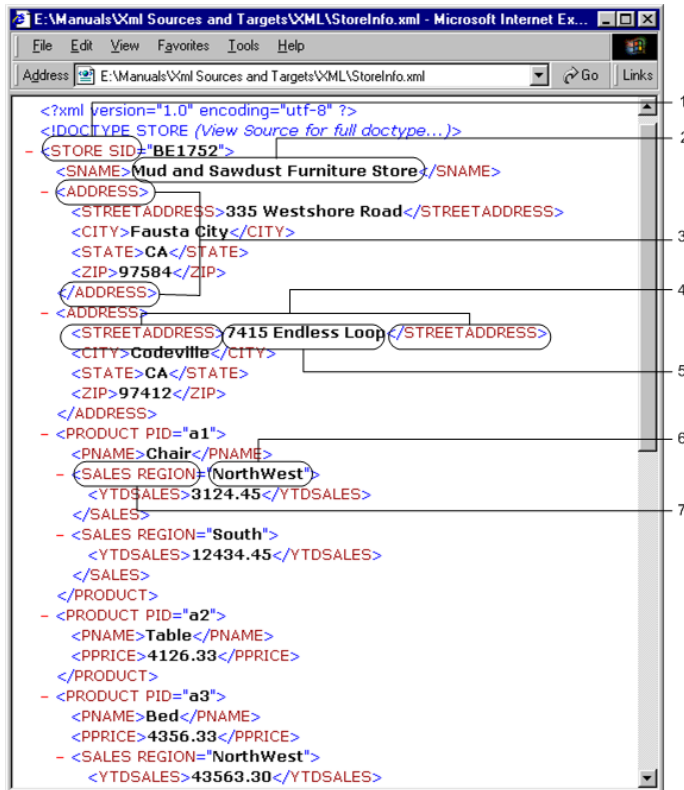
```
<book>
  <title>Fun with XML</title>
  <chapter>
    <heading>Understanding XML</heading>
    <heading>Using XML</heading>
  </chapter>
  <chapter>
    <heading>Using DTD Files</heading>
    <heading>Fun with Schemas</heading>
  </chapter>
</book>
```

book がルート要素になるので、book にはタイトルと章要素が指定されます。book が title と chapter の親要素になり、chapter が heading の親になります。title と chapter は、親が同じなので兄弟要素になります。

どの要素も、該当要素に関する追加情報を示す属性を持つことができます。次の例の場合、属性の graphic\_type が図の内容を記述しています。

```
<picture graphic_type="gif">computer.gif</picture>
```

以下の図に、XML ファイルの構成、要素、属性を示します。



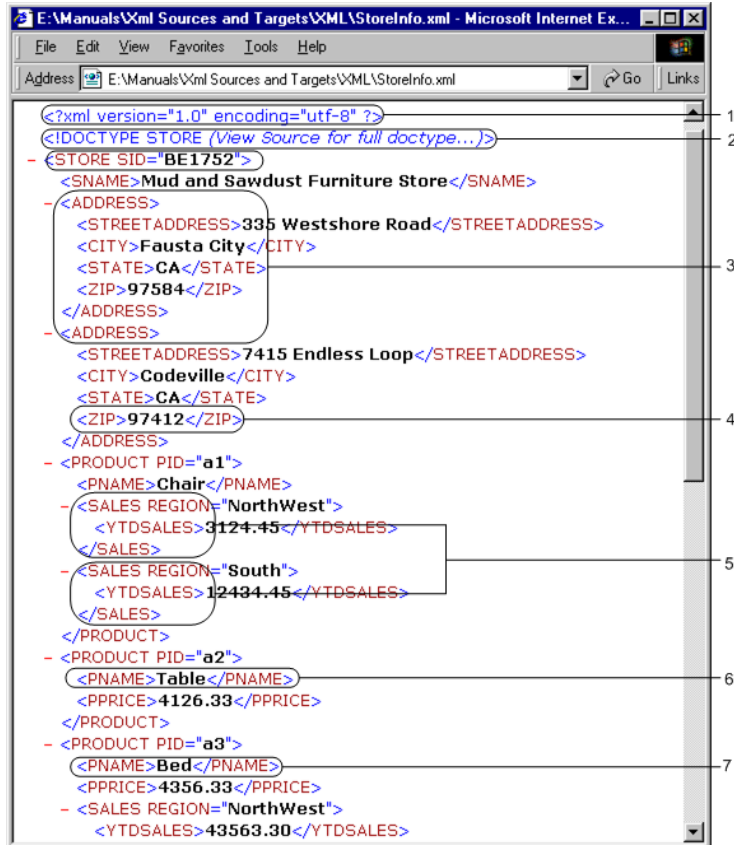
1. ルート要素
2. 要素のデータ
3. 囲み要素
4. 要素タグ
5. 要素のデータ
6. 属性値
7. 属性タグ

XML ファイルは階層化構造になっています。XML 階層には次の要素が含まれます。

- **子要素**。別の要素に含まれている要素です。
- **囲み要素**。他の要素を含んでいるが、データは含まない要素。囲み要素は、他の囲み要素を含むこともできます。
- **グローバル要素**。どの要素も、ルート要素の直接の子になります。ユーザーは、XML スキーマ全体でグローバル要素を参照することができます。
- **リーフ要素**。他の要素を含まない要素です。リーフ要素は、XML 階層において最も低いレベルにある要素です。
- **ローカル要素**。別の要素にネストされた要素です。ローカル要素は、親要素のコンテキスト内でのみ参照できます。
- **複数出現要素**。親要素内に複数回出現する要素です。囲み要素は複数出現要素になることができます。
- **親チェーン**。ある要素からルート要素までのパスをさかのぼる親子要素の連続です。

- **親要素**。他の要素を含まない要素です。
- **単独出現要素**。親要素内に 1 回のみ出現する要素です。

以下の図に、XML 階層のいくつかの要素を示します。



1. エンコード属性は、コードページを識別します。
2. DOCTYPE は、関連付けられた DTD ファイルを識別します。
3. 囲み要素: 要素 Address は、4 つの要素 StreetAddress、City、State および Zip を囲んでいます。要素 Address は、親要素でもあります。
4. リーフ要素: 要素 Zip は、他のすべての兄弟要素と同様に、要素 Address の最下位にある要素です。
5. 複数出現要素: 要素 Sales Region は、要素 Product 内で 2 回以上出現します。
6. 単独出現要素: 要素 PName は、要素 Product 内で 1 回出現します。
7. 子要素: 要素 PName は、Product の子要素で、Store の孫要素です。

## DTD またはスキーマによる XML ファイルの検証

有効な XML ファイルは、関連する DTD ファイルまたはスキーマファイルの構造に準拠しています。

DTD ファイルの位置および名前を参照するには、XML ファイルの DOCTYPE 宣言を使用します。

DOCTYPE 宣言では、XML ファイルのルート要素の名前も指定します。

たとえば、次の XML ファイルは note.dtd ファイルの位置を参照します。

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM
"http://www.w3schools.com/dtd/note.dtd">
```

```
<note>
  <body>XML Data</body>
</note>
```

スキーマを参照するには、schemaLocation 宣言を使用します。schemaLocation 宣言では、スキーマの場所と名前を指定します。

以下の XML ファイルは、外部に位置する note.xsd スキーマを参照します。

```
<?xml version="1.0"?>
<note xsi:SchemaLocation="http://www.w3schools.com note.xsd">
  <body>XML Data</body>
</note>
```

### Unicode によるエンコード

XML ファイルには、ファイルのコードページを示すエンコードの属性が含まれます。最も一般的なエンコードは、UTF-8 および UTF-16 です。UTF-8 は、Unicode の記号によって、文字を 1 - 4 バイトで表します。UTF-16 は、文字を 16 ビットワードとして表します。

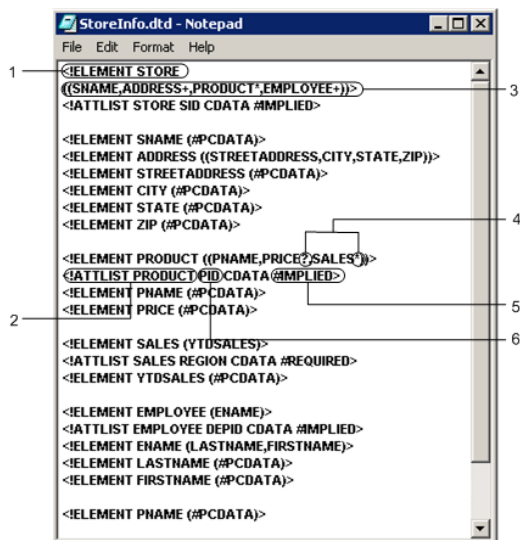
次の例では、XML ファイルの UTF-8 属性を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<note xsi:SchemaLocation="http://www.w3schools.com note.xsd">
  <body>XML Data</body>
</note>
```

## DTD ファイル

文書型定義(DTD)ファイルでは、XML ファイルの要素タイプおよび属性を定義します。また、DTD ファイルは XML ファイルの構造に制約を設けます。DTD ファイルには、データや要素のデータ型は含まれていません。

以下の図に、DTD ファイルの要素と属性を示します。



1. 要素
2. 属性
3. 要素リスト
4. 要素の出現
5. 属性値のオプション
6. 属性名

## DTD の構文

DTD ファイルでは、要素の宣言が XML 要素を定義します。要素の宣言には、次の構文が含まれています。

```
<!ELEMENT product (#PCDATA)>
```

この DTD 記述は、XML タグ<product>を定義します。この記述(#PCDATA)は、解析された文字データを意味します。解析されたデータは、XML 要素の開始タグと終了タグの間にあるテキストのことです。解析された文字データは、子要素が存在しないテキストのことです。

次の例は、2 つの子要素を持つ要素の DTD 記述を示します。

```
<!ELEMENT boat (brand, type) >  
<!ELEMENT brand (#PCDATA) >  
<!ELEMENT type (#PCDATA) >
```

ブランドおよびタイプは、boat の子要素です。それぞれの子要素には文字を指定することができます。この例では、要素 boat の中でブランドとタイプが一回ずつ出現可能です。次の DTD 記述は、boat に対してブランドが 1 回以上出現していなければならないことを指定します。

```
<!ELEMENT boat (brand+) >
```

## DTD 属性

属性では、要素についての追加情報を提供します。DTD ファイルでは、属性は要素の開始タグ内に出現します。

DTD ファイルの属性を次の構文で記述します。

```
<!ATTLIST element_name attribute_name attribute_type "default_value">
```

DTD ファイルの属性を以下のパラメータで記述します。

- **Element\_name**。この属性がある要素の名前です。
- **Attribute\_name**。この属性の名前です。
- **Attribute\_type**。属性の種類です。最もよく使う属性タイプは CDATA です。CDATA 属性は、文字データです。
- **Default\_value**。XML ファイルに属性値がない場合は属性の値です。

以下のオプションをデフォルト値と共に使用します。

- **#REQUIRED**。XML ファイルには必ず属性値を指定する必要があります。

- **#IMPLIED**。属性値はオプションです。

- **#FIXED**。XML ファイルには DTD ファイルのデフォルト値を含める必要があります。有効な XML ファイルには DTD と同じ属性値を指定できますが、XML ファイルでは属性値がない場合もあります。このオプションでは、デフォルト値を指定する必要があります。

以下の例に、固定値のある属性を示します。

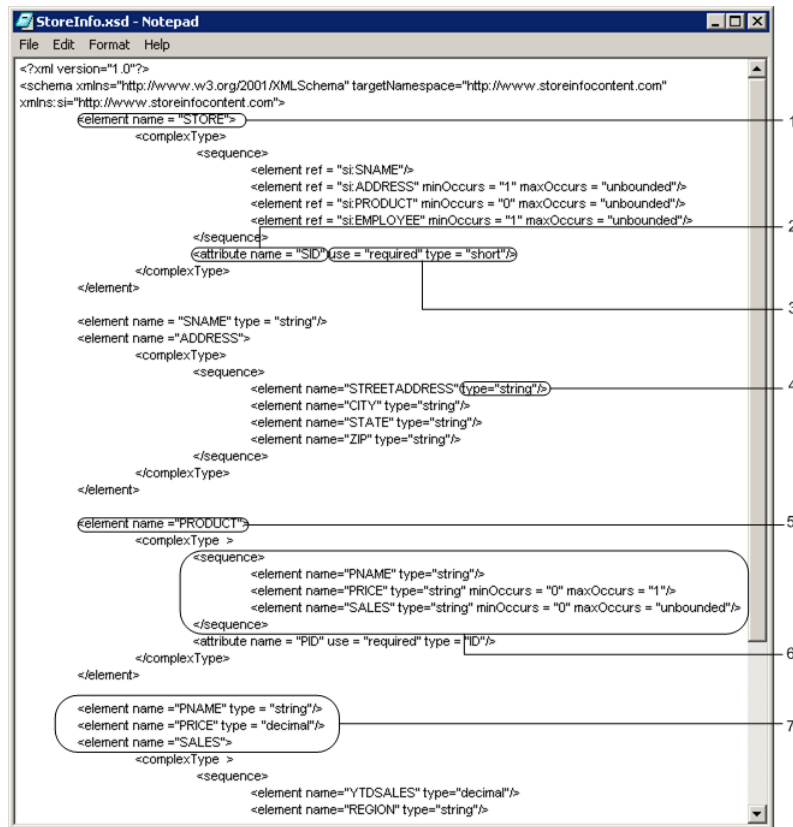
```
<!ATTLIST product product_name CDATA #FIXED "vacuum">
```

要素名は Product です。属性は product\_name です。属性には、デフォルト値の vacuum があります。

## XML スキーマファイル

XML スキーマは、XML ファイルの有効な内容を指定する文書のことです。DTD ファイルと同様に、XML スキーマファイルにはメタデータのみが含まれています。XML スキーマは、関連付けられた XML ファイルの要素および属性の構造と型を定義します。XML ファイルを定義するスキーマを使う場合、データの制限、データフォーマットの定義、およびデータ型間のデータ変換を行うことができます。XML スキーマでは複合型および型間の継承をサポートします。スキーマは、要素グループと属性グループ、ANY 内容、循環参照のそれぞれを指定する方法を提供します。

以下の図に、XML スキーマのコンポーネントを示します。



1. 要素名
2. 属性
3. 属性タイプと NULL 制約
4. 要素データ型
5. 要素のデータ
6. 要素リストと出現
7. 要素リストとデータ型

関連項目：

- [「XML の単純型および複合型」](#) (ページ 20)
- [「コンポーネントグループ」](#) (ページ 26)

## XML メタデータのタイプ

XML ファイル、DTD ファイル、または XML スキーマファイルから PowerCenter XML 定義を作成することができます。XML ファイルはデータおよびメタデータを提供します。DTD ファイルおよび XML スキーマファイルは、メタデータを提供します。



PowerCenter は、XML、DTD、および XML スキーマファイルから、以下のタイプのメタデータを抽出します。

- **名前空間**。XML ファイルの Uniform Resource Identifier (URI) 参照で識別される要素と属性名のコレクションです。ソースが異なる要素を区別するのが名前空間です。
- **名前**。要素または属性の名前を持っているタグです。
- **階層**。ある要素を XML ファイルの他の要素と比べたときの位置関係です。
- **カーディナリティ**。XML ファイルで発生する要素の回数です。
- **データ型**。数値、文字列、倫理値、時間などのデータ要素を分類したものです。XML ではデータ型と継承をサポートします。

## 名前空間

名前空間には、スキーマの位置を識別するための URI が含まれます。URI はインターネットリソースを識別するための文字列です。URI は URL を抽象化したものです。URL がリソースの位置を決めるのに対して、URI はリソースを識別します。URI で指定する位置に DTD ファイルやスキーマファイルが存在することは必要条件ではありません。

XML の名前空間は、要素のグループを識別するものです。名前空間では、異なる XML ファイルの要素や属性を識別するか、要素間の意味を判別することができます。たとえば、要素“table”の意味について、*math:table* および *furniture:table* という異なる名前空間を宣言することによって区別することができます。XML では、大文字と小文字が区別されます。名前空間の *Math:table* は、名前空間の *math:table* と区別されます。

XML ファイルのルートレベルで名前空間を宣言することも、XML 構造のすべての要素の内部で名前空間を宣言することもできます。同一の XML ファイルで複数の名前空間を宣言する場合、名前空間のプレフィックスを使用すると、要素を名前空間に関連付けることができます。名前空間の宣言は、xmlns から始まる属性として XML ファイル内に表示されます。xmlns 属性を持つ名前空間のプレフィックスを宣言します。接頭語名を作成する場合、長さは自由です。

次の例は、XML インスタンス文書内の 2 つの名前空間を示します。

```
<example>
  xmlns:math = "http://www.mathtables.com"
  xmlns:furniture = "http://www.home.com">
  <math:table>4X6</math:table>
  <furniture:table>Brueners </furniture:table>
</example>
```

1 つの名前空間には math 要素があり、その他の名前空間には furniture 要素があります。それぞれの名前空間には「table」という要素がありますが、要素には異なる型のデータが含まれます。名前空間プレフィックスは、math テーブルと furniture テーブル間を区別します。

次のテキストは、一般的なスキーマ宣言を示します。

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.w3XML.com"
  xmlns="http://www.w3XML.com"
  elementFormDefault="qualified">...
...</xs:schema>
```



以下の表に、名前空間宣言の各部分を示します。

| スキーマ宣言   | 説明   |
|--|--|
| <code>xmlns:xs="http://www.w3.org/2001/XMLSchema"</code> | ネイティブの XML スキーマおよびデータ型を含む名前空間です。この例では、各スキーマコンポーネントには、プレフィックスの「xs」があります。  |
| <code>targetNamespace="http://www.w3XML.com"</code>      | スキーマを含む名前空間です。   |
| <code>xmlns="http://www.w3XML.com"</code>                | デフォルトの名前空間宣言です。スキーマ内のすべての要素に、デフォルトの名前空間所属するプレフィックスがありません。プレフィックスのない <code>xmlns</code> 属性を使用して、デフォルトの名前空間を宣言します。 |
| <code>elementFormDefault="qualified"</code>              | スキーマ内のどの要素でも XML ファイルに名前空間があることを指定します。   |

## 名前

XML ファイルでは、各タグが要素または属性の名前になっています。DTD ファイルでは、`<!ELEMENT>` タグが要素の名前を示し、`<!ATTLIST>` タグが要素の属性を示しています。スキーマファイルでは、`<element name>` が要素の名前を示し、`<attribute name>` が属性の名前を示しています。

ユーザが XML 定義をインポートすると、デフォルトでは、要素タグが PowerCenter 定義のカラム名になります。

## 階層

XML ファイルは、階層型データベースをモデル化したものです。XML 階層内で、要素の位置は他の要素との関係を示しています。たとえば、各要素の内部に複数の子要素を指定することや、複数の要素が他の要素から特徴を継承することが可能です。

## カーディナリティ

DTD ファイルまたはスキーマファイルの場合、要素のカーディナリティとは、XML ファイル内に各要素が出現する回数のことです。ユーザが XML 定義の中にグループを構成する場合、要素のカーディナリティがその効率に影響を及ぼします。要素の絶対カーディナリティと相対カーディナリティは、XML 定義の構造に影響を及ぼします。

### 絶対カーディナリティ

要素の絶対カーディナリティとは、XML 階層で要素が親要素内に出現する回数です。DTD ファイルおよび XML スキーマファイルは、階層内の要素の絶対カーディナリティを表します。要素の絶対カーディナリティを表すために、DTD ファイルはシンボルを使用し、XML スキーマファイルは `<minOccurs>` および `<maxOccurs>` 属性を使用します。

たとえば、親要素内に 1 回だけ出現する要素では、絶対カーディナリティが 1 回(1)発生します。ただし、親要素のカーディナリティが 1 回以上(+)の場合、この要素が XML 階層内に数多く出現する可能性があります。

要素の絶対カーディナリティによって、NULL 制約が決まります。絶対カーディナリティが 1 回以上(+)の要素は NULL 値を持つことはできませんが、絶対カーディナリティが 0 回以上(\*)の要素では NULL 値を持

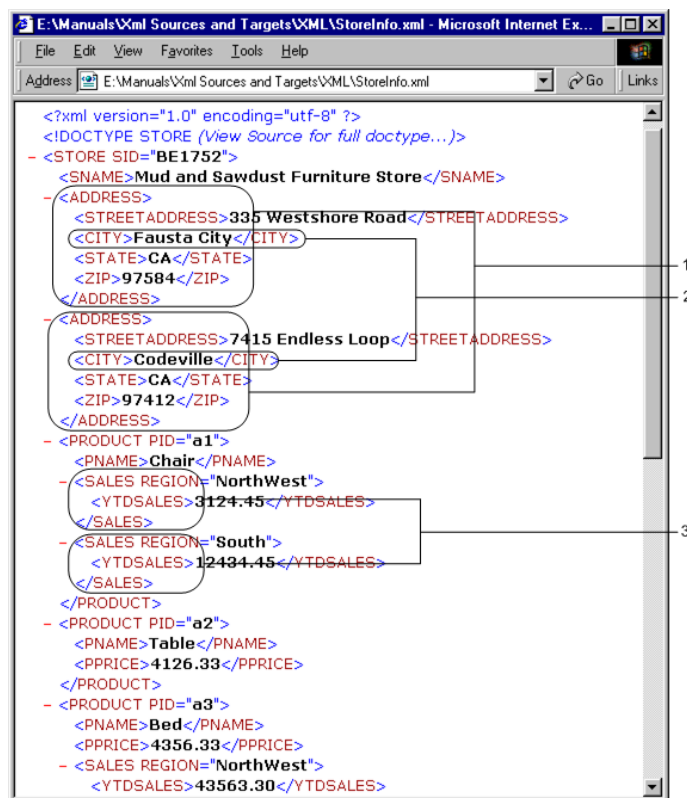
ことができます。XML スキーマまたは DTD ファイルの属性のうち、固定属性と必須属性は N U L L 値を持つことができますが、暗黙属性は N U L L 値を持つことができません。

以下の表に、DTD ファイルと XML スキーマファイルでカーディナリティを表す方法を示します。

| 絶対カーディナリティ       | DTD | スキーマ   |
|------------------|-----|--|
| ゼロまたは 1 回        | ?   | minOccurs=0 maxOccurs=1                                    |
| ゼロまたは 1 回またはそれ以上 | *   | minOccurs=0 maxOccurs=unbounded<br>minOccurs=0 maxOccurs=n |
| 1 回              | -   | minOccurs=1 maxOccurs=1                                    |
| 1 回またはそれ以上       | +   | minOccurs=1 maxOccurs=unbounded<br>minOccurs=1 maxOccurs=n |

**注:** 出現回数の最大値またはスキーマ内の無制限の出現回数を宣言することができます。

以下の図に、サンプル XML ファイル内の要素の絶対カーディナリティを示します。



1. 要素 *Address* は、*Store* 内で 1 回以上出現します。Address の絶対カーディナリティは、1 回以上 (+) です。
2. 要素 *City* は、親要素 *Address* 内で 1 回出現します。City の絶対カーディナリティは 1 回 (1) です。
3. 要素 *Sales* は、親要素 *Product* 内で 0 回以上出現します。Sales の絶対カーディナリティは、0 回以上 (\*) です。

## 相対カーディナリティ

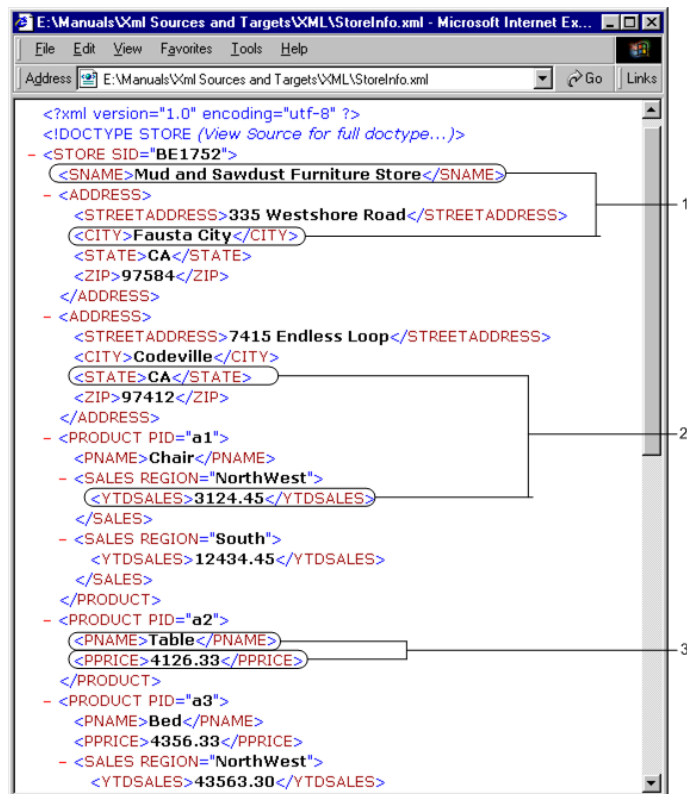
相対カーディナリティとは、XML 階層内の要素間の関係です。階層内の要素同士が持つことができる関係は、1 対 1、1 対多、または多対多の関係です。

ある要素に他の要素との 1 対 1 の関係があるということは、ある要素が 1 回出現したときに他の要素も 1 回出現する場合があるということです。たとえば、各従業員要素に対応する社会保険番号は 1 つです。したがって、従業員と社会保険番号との間には 1 対 1 の関係があります。

ある要素に他の要素との 1 対多の関係があるということは、ある要素が 1 回出現したときに他の要素は複数回出現する場合があるということです。たとえば、1 つの従業員要素に対して複数のメールアドレスが対応している場合があります。この場合、従業員とメールアドレスは、1 対多の関係になります。

要素間で多対多の関係があるとは、XML ファイルで両方の要素が複数回出現する可能性があることです。たとえば、1 人の従業員に対して複数のメールアドレスと複数の番地が対応することがあります。メールアドレスと番地は多対多関係にあります。

以下の図に、サンプル XML ファイル内の要素間の相対カーディナリティを示します。



- 1 対多関係。SNAME が出現するたびに ADDRESS が数多く出現する可能性があり、したがって CITY も数多く出現する可能性があります。
- 多対多関係。STATE が出現するたびに、YTDSALES も数多く出現する可能性があります。YTDSALES が出現するたびに、STATE も数多く出現する可能性があります。
- 1 対 1 関係。PNAME が出現するたびに、PPRICE が 1 回出現します。

## XML の単純型および複合型

XML スキーマ言語には、ビルトインのデータ型が 40 種類以上あります。たとえば、数値、文字列、時間、ML、バイナリ値などです。上記のデータ型を単純型と呼んでいます。単純型にはテキストが含まれますが、その他の要素や属性は含まれません。ユーザは基本的な XML 単純型から新規の単純型を導き出すことができます。

ユーザは XML の複合データ型を作成することができます。複合データ型とは、単純型を複数個含むことができるデータ型のことです。複合型には、他の複合型や属性を含むことができます。

XML のデータ型の詳細については、<http://www.w3.org/TR/xmlschema-2> で XML のデータ型の W3C 仕様を参照してください。

### 単純型

単純データ型とは、テキストを含む XML の要素または属性のことです。単純型は分割できません。単純型は属性を持つことができません。

PowerCenter は、以下の単純型をサポートしています。

- **Atomic 型**。倫理型、文字列、または整数などの基本的なデータ型です。
- **リスト**。Atomic 型の配列を集めたものです。
- **ユニオン**。Atomic または List を XML ファイル内で 1 個の簡単な型にマッピングし、1 個または複数個を取り出して組み合わせたものです。

#### Atomic 型

Atomic データ型とは、倫理値、文字列、整数、小数、日付といった基本的なデータ型です。カスタム Atomic データ型を定義するには、Atomic データ型に制約を加えて内容を制限します。ファセットを使用して、どの値を制約するか許可するかを定義します。

ファセットとは、有効な値の最小値や最大値、指定値、またはデータのパターンを定義する式のことです。たとえば、パターンファセットは要素をデータの値の式に制限します。列挙型ファセットでは、要素に対する有効値を一覧表示します。

以下の例には、要素を小文字の a から z に制限するパターン型ファセットが含まれます。

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-z]"/>
    </xs:restriction>
  </xs:simpleType></xs:element>
```

以下の例には、文字列を a、b、c に制限する列挙型ファセットが含まれます。

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="a"/>
      <xs:enumeration value="b"/>
      <xs:enumeration value="c"/>
    </xs:restriction>
  </xs:simpleType></xs:element>
```

## リスト

リストは、名前を表す文字列のリストのような、Atomic 型の配列のコレクションです。item タイプは、List のデータ型を定義しています。

次の例は、名前を呼び出すリストを示します。

```
<xs:simpleType name="names">
  <xs:list itemType="xs:string" />
</xs:simpleType>
```

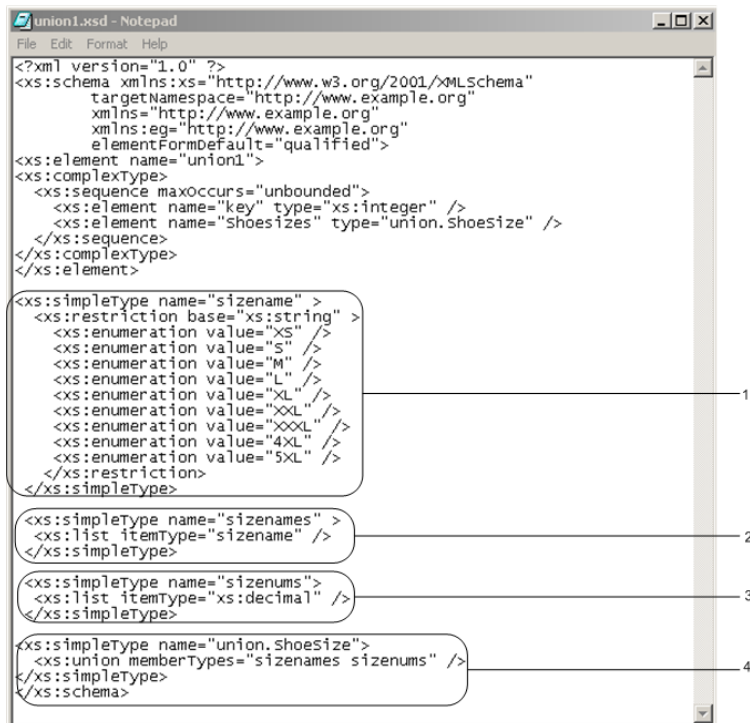
XML ファイルには、名前リスト内の以下のデータを含むことができます。

```
<names>Joe Bob Harry Atlee Will</names>
```

## ユニオン

Atomic または List を XML ファイルで 1 個の簡単な型にマッピングし、1 個または複数個を取り出して組み合わせたものを Union といいます。Union を定義する場合、組み合わせの対象になる型を指定します。たとえば、size という名前の型を作成するとします。Size には S、M、L などの文字列データを指定できますが、さらに 30、32、34 などの 10 進サイズも指定可能です。Union 型要素を定義する場合、XML ファイルでは、文字列サイズに sizename 型を指定し、数値サイズに sizenum 型を指定することができます。

以下の図に、shoesize union を含むスキーマファイルを示します。この Union には sizenames リストと sizenums リストが含まれています。



1. Sizename は制限のある文字列型です。
2. sizenames 型は文字列のリストを受け入れます。
3. sizenums 型は 10 進値の List を受け入れます。
4. shoesize union は、10 進値 List と文字列 List の両方を受け入れます。

この Union は、sizenames と sizenums を Union メンバ型として定義しています。sizenames は、文字列値の List を定義します。sizenums は、10 進値の List を定義します。

## 複合型

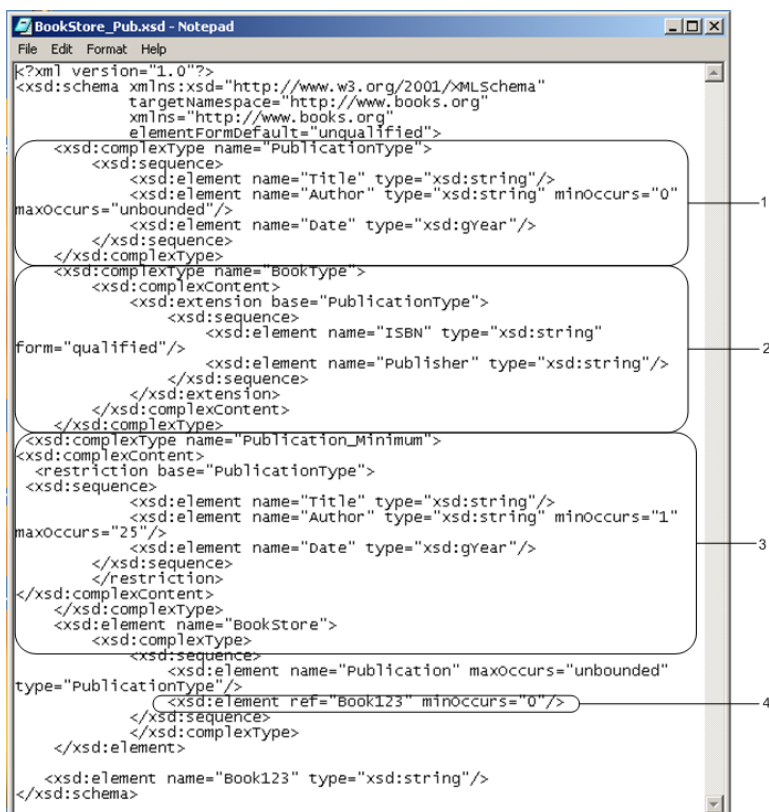
複合型は、単純型のコレクションを論理ユニットに集計します。たとえば、Customer 型には顧客番号、名前、番地、町、市、および郵便番号を指定することができます。複合型は、他の複合型を参照することも要素グループや属性グループを参照することもできます。

XML では複合型の継承をサポートしています。複合型を定義すると、基本型のコンポーネントを継承している他の複合型を作成できます。型関係の場合、ベース型が複合型になるので、ユーザはこれから別の方を導き出すことができます。派生複合型は、基本型から要素を継承します。

拡張複合型は基本型から要素を継承する派生型であり、そのため、要素が追加されます。たとえば、customer\_purchases は複合型 customer から定義を継承することもあります。要素の item、cost、date\_sold を追加することにもなります。

限定複合型は、基本型からの要素の一部を制限している派生型です。たとえば、mail\_list は customer から要素を継承しますが、minoccurs 境界と maxoccurs 境界をゼロに設定すれば、phone\_number 要素に制限を加えることができます。

以下の図に、ベース複合型を制限して拡張する派生複合型を示します。



1. ベース複合型
2. 拡張複合型
3. 限定複合型
4. 要素参照



上の図では、基本型は PublicationType です。BookType は PublicationType を拡張したもので、ISBN 要素と Publisher 要素が含まれています。Publication\_Minimum は PublicationType を限定しています。Publication\_Minimum では Authors が 1 から 25 の範囲で必要であり、日付を年に限定しています。

### 抽象要素

時には、スキーマは複合要素の基本構造を定義する基本型を含むことがあります。すべてのコンポーネントを含むわけではありません。派生複合型は、より多くのコンポーネントで基本型を拡張します。基本型が完全な定義ではないため、XML ファイルで基本型を使用しないことをお勧めします。基本型要素を抽象的に定義することができます。抽象要素は、XML ファイルで有効ではありません。派生要素のみが有効です。

抽象要素を定義するには、抽象属性に「true」の値を追加します。デフォルトは false です。

たとえば、PublicationType は抽象要素とします。BookType は PublicationType の要素を継承したものです。ISBN 要素と Publisher 要素も含まれています。PublicationType が抽象要素なので、PublicationType 要素は XML ファイルで有効ではありません。XML ファイルは、派生型の BookType のみを含むことができます。

次のスキーマは、PublicationType および BookType を含みます。

```
<xsd:complexType name="PublicationType" abstract="true">
  <xsd:sequence>
    <xsd:element name="Title" type="xsd:string"/>
    <xsd:element name="Author" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="Date" type="xsd:gYear"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="BookType">
  <xsd:complexContent>
    <xsd:extension base="PublicationType" >
      <xsd:sequence>
        <xsd:element name="ISBN" type="xsd:string"/>
        <xsd:element name="Publisher" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

### Any 型要素および属性

スキーマ要素および属性のなかには、XML ファイルに対して Any 型のデータを許可するものがあります。これらの要素や属性は、認識されていない要素型や属性型を含む XML ファイルを検証する際に利用します。

Any 型のデータを許可する、次の要素および属性を使用します。

- **anyType 要素**。関連付けられた XML ファイル内の Any データ型に対して、要素を 1 つ許可します。
- **anySimpleType 要素**。関連付けられた XML ファイル内の simpleType に対して、要素を 1 つ許可します。
- **ANY 内容要素**。スキーマで既に定義されているすべての要素に対して、要素を 1 つ許可します。
- **anyAttribute 属性**。スキーマで既に定義されているすべての属性に対して、要素を許可します。

## anyType 要素

anyType 要素は、XML インスタンス文書内のどのデータ型をとることもできます。要素に異なるデータ型を含む場合、要素を anyType として宣言します。

次のスキーマは、人物を anyType である First Name、Last Name、Age の要素で示します。

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <<xs:element name="age" type="xs:anyType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

次の XML インスタンス文書には、データ型と Age 要素の数字が含まれます。

```
<person>
  <firstname>Danny</firstname>
  <lastname>Russell</lastname>
  <age>1959-03-03</age>
</person>
<person>
  <firstname>Carla</firstname>
  <lastname>Havers</lastname>
  <age>46</age>
</person>
```

両方の型は、スキーマでは有効です。スキーマ内で要素のデータ型を宣言しない場合、Designer でスキーマをインポートした際に要素は anyType をデフォルトに設定します。

## anySimpleType 要素

anySimpleType 要素は、どの Atomic 型も含むことができます。Atomic 型とは、倫理値、文字列、整数、小数、または日付といった基本的なデータ型です。

次のスキーマは、人物を anySimpleType である First Name、Last Name、その他の要素で示します。

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <<xs:element name="other" type="xs:anySimpleType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

次の XML インスタンス文書は、anySimpleType 要素を文字列データ型に置き換えます。

```
<person>
  <firstname>Kathy</firstname>
  <lastname>Russell</lastname>
  <other>Cissy</other>
</person>
```

次の XML インスタンス文書は、anySimpleType 要素を数値データ型に置き換えます。

```
<person>
  <firstname>Kathy</firstname>
```



```

    <lastname>Russell</lastname>
    <other>34</other>
</person>

```

## ANY 内容要素

ANY 内容要素は、XML ファイルのすべてのコンテンツを許可します。スキーマで ANY 内容要素を宣言すると、XML インスタンス文書内のすべての名前および型の要素と置き換えることができます。置き換える要素は、スキーマ内に必ず存在していることが必要です。

ANY 内容を指定する場合、要素名や要素型の代わりにキーワードの ANY を使用します。

次のスキーマは、人物を ANY 内容である First Name、Last Name などの要素で示します。

```

<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="son" type="xs:string"/>
<xs:element name="daughter" type="xs:string"/>

```

スキーマには、son 要素および daughter 要素が含まれます。ANY 要素は、XML インスタンス文書内の son 要素または daughter 要素と置き換えることができます。

```

<person>
  <firstname>Danny</firstname>
  <lastname>Russell</lastname>
  <son>Atlee</son>
</person>
<person>
  <firstname>Christine</firstname>
  <lastname>Slade</lastname>
  <daughter>Susie</daughter>
</person>

```

## anyAttribute 属性

anyAttribute 属性は、XML ファイル内の属性をすべて受け入れます。属性を anyAttribute として宣言すると、anyAttribute 要素をスキーマ内のどの属性とも置き換えることができます。

次のスキーマは、人物を anyAttribute である First Name、Last Name の要素で示します。

```

<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
    <xs:anyAttribute/>
  </xs:complexType>
</xs:element>

```

次のスキーマには、gender 属性が含まれます。

```

<xs:attribute name="gender">
  <xs:simpleType>

```

```

    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

```

次の XML インスタンス文書は、gender 属性を持つ anyAttribute と置き換えることができます。

```

<person gender="female">
  <firstname>Anita</firstname>
  <lastname>Ficks</lastname>
</person>
<person gender="male">
  <firstname>Jim</firstname>
  <lastname>Geimer</lastname>
</person>

```

## コンポーネントグループ

XML スキーマには、以下のコンポーネントのグループを作成できます。

- **要素および属性のグループ**。スキーマを通じて参照できる要素または属性のグループです。
- **置き換えグループ**。同じグループ内のその他の要素と置き換えることができる要素のグループです。

### 要素および属性のグループ

要素や属性をグループにすることで、スキーマ内で参照できるようになります。グループを参照する前に、要素または属性のグループを宣言する必要があります。

次の例は、要素グループのスキーマ構文を示します。

```

<xs:group name="Songs">
  <xs:element name="songTitle" type="xs:string" />
  <xs:element name="artist" type="xs:string" />
  <xs:element name="publisher" type="xs:string" />
</xs:group>

```

次の例は、属性グループのスキーマ構文を示します。

```

<xs:attributeGroup name="Songs">
  <xs:attribute name="songTitle" type="xs:string" />
  <xs:attribute name="artist" type="xs:string" />
  <xs:attribute name="publisher" type="xs:string" />
</xs:attributeGroup>

```

次の要素グループは、XML データに制約を設けます。

- **sequence グループ**。XML ファイル内で要素が出現する場合、スキーマの要素一覧順序に従って出現する必要がある場合があります。たとえば、OrderHeader であれば、customerName、orderNumber、および orderDate をこの順序で並べる必要があります。

```

<xs:group name="OrderHeader">
  <xs:sequence>
    <xs:element name="customerName" type="xs:string" />
    <xs:element name="orderNumber" type="xs:number" />
    <xs:element name="orderDate" type="xs:date" />
  </xs:sequence>
</xs:group>

```

- **choice グループ**。XML ファイルでは、該当グループ内の要素 1 つが出現します。たとえば、CustomerInfo グループは、XML ファイルの要素の選択肢を一覧表示します。

```
<xs:group name="CustomerInfo">
  <xs:choice>
    <xs:element name="customerName" type="xs:string" />
    <xs:element name="customerID" type="xs:number" />
    <xs:element name="customerNumber" type="xs:integer" />
  </xs:choice>
</xs:group>
```

- **all グループ**。XML ファイルですべての要素が出現するか、すべての要素がまったく出現しないことが条件です。要素の出現順序は任意です。たとえば、CustomerInfo は 3 つの要素のすべてを必要とするか、またはすべてを必要としないかのいずれかです。

```
<xs:group name="CustomerInfo">
  <xs:all>
    <xs:element name="customerName" type="xs:string" />
    <xs:element name="customerAddress" type="xs:string" />
    <xs:element name="customerPhone" type="xs:string" />
  </xs:all>
</xs:group>
```

## 置き換えグループ

置き換えグループを使用して、XML ファイル内で 1 つの要素を別の要素に置き換えることができます。たとえば、カナダとアメリカからのアドレスがある場合、カナダ用のアドレス型とアメリカ用のアドレス型をそれぞれ区別して作成したい場合があります。この場合、ユーザはいずれのアドレスでも受け付けてくれる置き換えグループを作成できます。

基本型の Address と派生型の CAN\_Address および USA\_Address を次のスキーマセクションに示します。

```
<xs:complexType name="Address">
  <xs:sequence>
    <xs:element name="Name" type="xs:string" />
    <xs:element name="Street" type="xs:string"
      minOccurs="1" maxOccurs="3" />
    <xs:element name="City" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="MailAddress" type="Address" />
<xs:complexType name="CAN_Address">
  <xs:complexContent>
    <xs:extension base="Address">
      <xs:sequence>
        <xs:element name="Province" type="xs:string" />
        <xs:element name="PostalCode" type="CAN_PostalCode"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="USA_Address">
  <xs:complexContent>
    <xs:extension base="Address">
      <xs:sequence>
        <xs:element name="State" type="USPS_StateCode" />
        <xs:element name="ZIP" type="USPS_ZIP"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
```

```
</xs:complexType>
<xs:element name="AddrCAN" type="CAN_Address"
substitutionGroup="MailAddress"/>
<xs:element name="AddrUSA" type="USA_Address"
substitutionGroup="MailAddress"/>
```

CAN\_Address は Province および PostalCode を含み、USA\_Address は State および Zip を含みます。置き換えグループの MailAddress には両方のアドレス型が指定されています。

関連項目：

- [「XML 定義での置き換えグループの使用」 \(ページ 45\)](#)

## XML パス

XPath(XPath)は、XML ファイル内の項目の位置を特定する方法を記述する言語です。XPath は、ルートから要素または属性への階層でのルートに基づくアドレス構文を使用します。XML パスには、長いスキーマコンポーネントの名前を含めることができます。

XPath はスラッシュ(/)を使用して、階層内の要素を区別します。XPath 内の XML 属性は、先頭に「@」が付きます。

要素または属性の XPath でクエリを作成し、XML データをフィルタすることができます。

関連項目：

- [「XPath クエリの述部の使用」 \(ページ 48\)](#)

## コードページ

XML ファイルには、ファイル内で使用されるコードページを表すエンコーディングが宣言されています。XML のコードページで最も普及しているのは UTF-8 と UTF-16 です。すべての XML パーサはこれらのコードページをサポートします。XML での文字エンコーディングの仕様の詳細については、W3C の Web サイト (<http://www.w3c.org>) を参照してください。

PowerCenter は、リレーショナルデータベースやその他のフラットファイルに対してサポートしているのと同じコードページを XML ファイルについてサポートします。PowerCenter はユーザ定義コードページをサポートしません。

XML ソース定義またはターゲット定義の作成時に、Designer によって定義に PowerCenter クライアントのコードページが割り当てられます。コードページ割り当てを含む XML スキーマをインポートすると、XML ウィザードによってスキーマのコードページが表示されます。ただし、XML ウィザードは、そのコードページをリポジトリに作成した XML 定義には適用しません。

XML ソース定義のコードページは設定できません。XML ソースファイルは、解析時に Integration Service によって Unicode に変換されます。

ターゲット XML 定義のコードページは、Designer で設定できます。セッションプロパティで XML ターゲットインスタンスのコードページを変更することもできます。

# PowerCenter での XML の使用

## PowerCenter での XML の使用の概要

PowerCenter 用の XML 定義を作成するには、XML ファイル、DTD ファイル、XML スキーマ、フラットファイル定義、またはリレーショナルテーブルを使用します。ユーザが XML 定義を作成すると、Designer は XML メタデータを抽出し、リポジトリにスキーマを作成します。スキーマが提供する構造に基づいて、ユーザは XML 定義を編集および検証します。

XML 定義には複数のグループを指定することができます。XML の定義では、グループはビューと呼ばれます。XML 階層内の要素間の関係によって、ビュー間の関係が定義されます。

ユーザが XML 定義を作成すると、Designer はデフォルトでスキーマに複数回出現する要素および複合型に対するビューを作成します。XML 階層内の要素の相対カーディナリティは、PowerCenter が XML 定義のビューを作成するときの方法に影響します。相対カーディナリティは、要素が同一のグループに含まれるかどうかを決定します。

Designer では、XML 定義のビュー間の関係をキーにより定義します。ソース定義にはキーが不要ですが、ターゲットビューにはキーが必要になります。各ビューのプライマリキーには、XML 要素または生成キーを指定できます。

ユーザが XML 定義を作成する場合、階層モデルまたはエンティティ関係モデルを作成できます。階層モデルを作成した場合、正規化階層または非正規化階層を作成します。正規化階層に含まれるのは、複数出現要素を対象にした個々のビューです。非正規化階層には、複数出現要素を対象にした重複データのビューが 1 つ含まれています。

エンティティモデルの作成時、Designer が作成するのは複合型および複数出現要素を対象にしたビューです。Designer は、スキーマが提供する継承および循環リレーションをモデル化する XML 定義を作成します。

PowerCenter は要素数が 400 未満の XML スキーマを処理できます。PowerCenter プロファイルには 3 つ以下の階層レベルと、次の複合型要素を含めることができます。

- Sequence
- Any
- Choice

PowerCenter XML インポートウィザードは最大 400 のビューを作成できます。

## 制限

PowerCenter の XML 処理には、次の制限が適用されます。

- XML スキーマの要素数は 400 未満である必要があります。
- XML スキーマファイルのサイズは 100KB 未満である必要があります。
- XML ファイルのサイズは 10MB 以下である必要があります。
- 複雑性プロファイルの階層レベル数は 3 以下である。
- XML インポートウィザードで作成されるビューの数は最大 400 である。

PowerCenter では、以下の機能はサポートされていません。

- **連結されたカラム。**2つの要素の連結を、1つのカラムにすることはできません。たとえば、2つの要素 FIRSTNAME および LASTNAME の連結を参照する FULLNAME 列を作成することはできません。
- **複合キー。**2つの要素の連結を、1つのキーにすることはできません。たとえば、2つの要素 LASTNAME および PHONENUMBER の複合を参照するキー CUSTOMERID を作成することはできません。
- **解析リスト。**PowerCenter では、リスト型を保存する場合、配列要素のすべてを含む 1 個の文字列として保存します。PowerCenter では、文字列から個々に取り出した単純型の解析は行いません。

他の要素型を持つトランスフォーメーションを作成し、大きな XML 入力ファイルを変換するには、データプロセッサトランスフォーメーションを使用します。データプロセッサトランスフォーメーションの作成方法の詳細については、『*Informatica Data Transformation ユーザーガイド*』および『*Informatica Data Transformation 入門ガイド*』を参照してください。

## XML メタデータのインポート

ユーザが XML 定義をインポートすると、Designer は定義用のリポジトリでスキーマを作成します。ユーザが XML 定義を編集・検証するときの構造はリポジトリスキーマで規定されます。

メタデータは次のファイルタイプから作成することができます。

- XML ファイル
- DTD ファイル
- XML スキーマファイル
- リレーショナルテーブル
- フラットファイル

## XML ファイルからのメタデータのインポート

XML ファイルでは、1 組のタグによって各データ要素の始めと終わりに印を付けます。これらのタグは、PowerCenter が XML ファイルから抽出するメタデータの基盤となります。ユーザがインポートした XML ファイルに関連 DTD または XML スキーマがない場合、Designer は XML タグを読み取って要素を決定し、さらに要素の出現可能性や階層における要素の位置も決定します。Designer は、要素タグ内のデータをチェックし、データの表示に従ってデータ型を割り当てます。XML 定義の要素に関して、ユーザはデータ型を変更することができます。

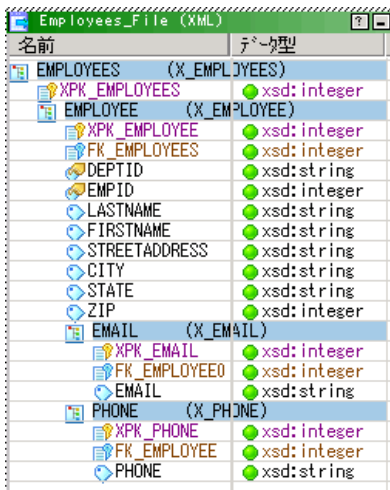
以下の図に、サンプル XML ファイルを示します。



ルート要素は Employees です。Employee は、複数出現要素です。Employee 要素には LastName、FirstName、および Address の 3 要素があります。Employee 要素には、Phone および Email などの複数出現要素も含まれます。

Designer では、この XML データからスキーマ構造を決定します。

以下の図に、ルート要素および複数出現要素に個別のビューを提供するデフォルトの XML ソース定義を示します。



XML ファイルをインポートした場合、XML データのすべてを使って XML ファイルを作成する必要はありません。XML ファイルの階層を正確に表示するには十分なデータが必要になります。

Designer は、DTD ファイルまたは XML スキーマを参照する XML ファイルから、XML 定義を作成できます。XML ファイルが別のノード上の DTD または XML スキーマへの参照を持っている場合、Designer がスキーマを読み取ることができるように、PowerCenter クライアントのホストノードからスキーマが格納



されているノードにアクセスする必要があります。XML ファイルには、DTD または XML スキーマのアドレスである Universal Resource Identifier (URI) が含まれます。

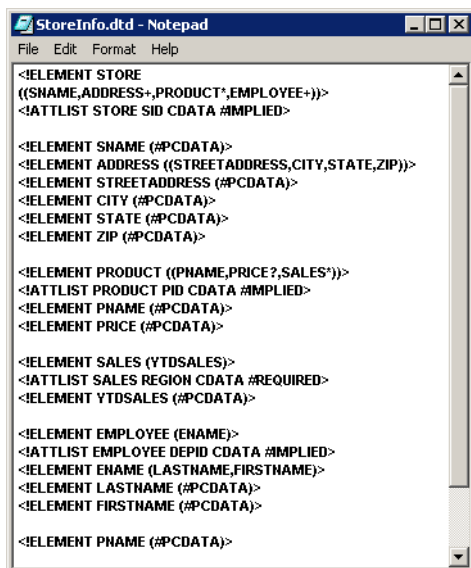
## DTD ファイルからのメタデータのインポート

DTD ファイルは XML 文書の構造に制約を設けます。DTD ファイルでは、ユーザが XML 文書で利用できる要素、属性、エンティティ、および表記を一覧表示します。また、コンポーネント間の関係を指定します。カーディナリティと NULL 制約も DTD ファイルで指定されます。ただし、DTD ファイルはデータやデータ型を含みません。

DTD ファイルをインポートすると、XML 定義の要素に対するデータ型を変更できます。ユーザは NULL 制約を変更することができますが、要素のカーディナリティを変更することはできません。

関連 DTD がある XML ファイルをユーザがインポートすると、Designer は DTD 構造に基づいた定義を作成します。

以下の図に、StoreInfo.dtd が Store 要素を含み、Product が Store の子要素のいずれかである XML ファイルの例を示します。



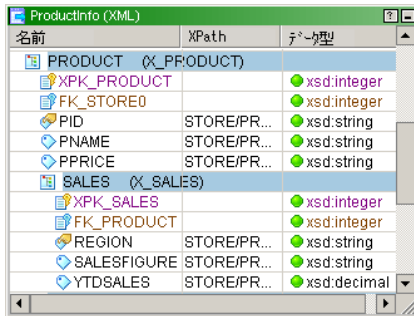
以下の図に、関連する DTD を示します。



関連する DTD の ProductInfo.xml では、StoreInfo.dtd の要素 Product を使用します。Product には複数出現要素の Sales があります。



以下の図に、Designer が作成するソース定義を示します。



| 名前                  | XPath       | データ型        |
|---------------------|-------------|-------------|
| PRODUCT (X_PRODUCT) |             |             |
| XP_K_PRODUCT        |             | xsd:integer |
| FK_STORE0           |             | xsd:integer |
| PID                 | STORE/PR... | xsd:string  |
| PNAME               | STORE/PR... | xsd:string  |
| PPRICE              | STORE/PR... | xsd:string  |
| SALES (X_SALES)     |             |             |
| XP_K_SALES          |             | xsd:integer |
| FK_PRODUCT          |             | xsd:integer |
| REGION              | STORE/PR... | xsd:string  |
| SALESFIGURE         | STORE/PR... | xsd:string  |
| YTD_SALES           | STORE/PR... | xsd:decimal |

ProductInfo 定義には Product グループと Sales グループが含まれます。XML ファイルでは、定義にどんな要素を含めるかを決定します。DTD ファイルは XML 定義の構造を決定します。

## XML スキーマからのメタデータのインポート

スキーマファイルでは XML ファイル内の要素および属性の構造を定義します。スキーマファイルには、ファイル内の要素および属性の型に対する説明が含まれます。XML スキーマをインポートすると、Designer によって要素のデータ型、精度、およびカーディナリティが判断されます。要素の定義がスキーマから継承されている場合、PowerCenter で要素の定義を変更することはできません。

XML スキーマからメタデータをインポートする場合、.xsd ファイルには、別の.xsd ファイルを参照するインポート文やインクルード文を含めることができます。別のスキーマを含むスキーマをインポートする場合、別のスキーマが同じ名前空間を参照することがないようにしてください。

例:

```
<IMPORT
schemaLocation="../../../administration/process/bo/LocationTextB0.xsd"
namespace="http://EnterpriseLibrary/com/acs/enterprise/common/program/administration/process/bo">
<IMPORT
schemaLocation="../../../administration/process/bo/LineOfBusinessB0.xsd"
namespace="http://EnterpriseLibrary/com/acs/enterprise/common/program/administration/process/bo">
<IMPORT
schemaLocation="../../../administration/process/bo/ClaimExceptionB0.xsd"
namespace="http://EnterpriseLibrary/com/acs/enterprise/common/program/administration/process/bo">
```

複数の「import schemalocation」文を 1 つの文で置き換えることができます。

```
<xsd:import schemalocation="imported.xsd" namespace="http://EnterpriseLibrary/com/acs/enterprise/
common/program/administration/process/bo"/>
```

インポートされた.xsd ファイルには、次の構文で他の XSD ファイルが含まれています。

```
<xsd:schema targetNamespace="http://EnterpriseLibrary/com/acs/enterprise/common/program/
administration/process/bo" elementFormDefault="qualified" >
  <xsd:include schemalocation="LocationTextB0.xsd" />
  <xsd:include schemalocation="LineOfBusinessB0.xsd" />
  <xsd:include schemalocation="ClaimExceptionB0.xsd" />
</xsd:schema>
```

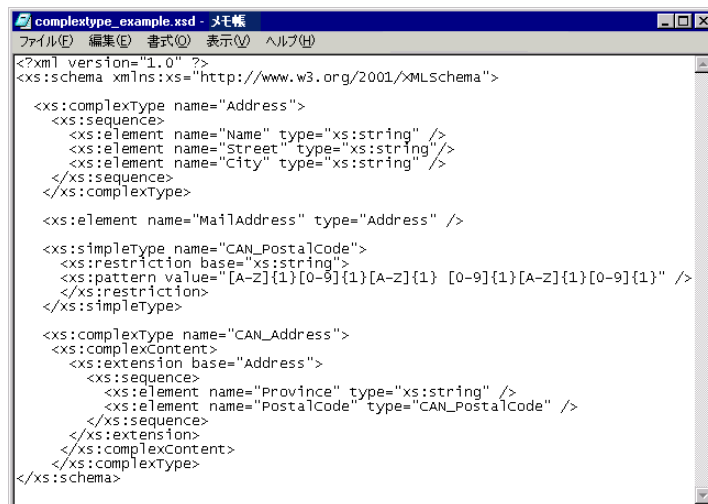
詳細については、ナレッジベースの記事 158334 を参照してください。

XML スキーマ内の各単純型定義は、論理型、文字列型、または整数型などのスキーマ Atomic データ型内にある他の単純型定義に対して制限を与え、anySimpleType データ型を制限します。XML スキーマで単

純データ型を定義すると、既存のデータ型から新しいデータ型が生成されます。たとえば、1 から 20 の数字のみを持つ制限された整数型を生成できます。基本型は整数です。

複合データ型を他のデータ型から派生させた場合、基本型の要素を含むデータ型を新しく作成します。派生型に新しい要素を追加するか、継承要素に制限を加えることができます。Designer は、継承コンポーネントを表すカラムを重複させることなく、派生型に対してビューを作成します。これによりメタデータは削除され、リポジトリ内の XML 定義の容量を減らします。

以下の図に、単純派生型および複合派生型のスキーマを示します。



MailAddress 要素は、複合型である Address タイプです。派生型の CAN\_Address は、Address タイプから Name、City、Street を継承し、Province および PostalCode を追加して Address を拡張します。PostalCode は単純型であり、CAN\_PostalCode と呼ばれています。

XML スキーマをインポートすると、すべての単純型または複合型の属性に XML 定義のカラムになる可能性があります。複合型は、ビューになります。

以下の図に、スキーマをデフォルトのオプションでインポートした場合の、スキーマの XML 定義を示します。

| 名前                          | データ型           |
|-----------------------------|----------------|
| MailAddress (X_MailAddress) |                |
| XPk_MailAddress             | xsd:integer    |
| MailAddress                 | Address        |
| Address (X_Address)         |                |
| XPk_Address                 | xsd:integer    |
| Name                        | xsd:string     |
| Street                      | xsd:string     |
| City                        | xsd:string     |
| CAN_Address (X_CAN_Address) |                |
| XPk_CAN_Address             | xsd:integer    |
| FK_Address                  | xsd:integer    |
| Province                    | xsd:string     |
| PostalCode                  | CAN_PostalCode |

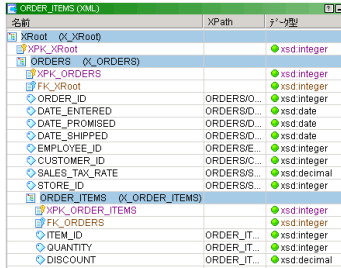
CAN\_Address ビューに含まれるのは、そのタイプに対して一意である要素です。ルート要素は MailAddress です。Address 型には Name、Street、City が含まれます。CAN\_Address には Address に対する外部キーがあります。CAN\_Address は、Province および PostalCode があります。

MailAddress から継承する Name、Street、City はこのビューに含まれません。

## リレーショナル定義からのメタデータの作成

XML 定義を作成する場合、複数のリレーショナル定義を選択して、定義間の関係を作成します。ユーザがリレーショナル定義をインポートすると、Designer はそれぞれの定義に対して XML ビューを作成します。リレーショナル定義の各カラムを変換後、Designer はプライマリキーと外部キーとの関係を生成します。root ビューを作成することを選択できます。

以下の図に、リレーショナル定義の Orders および Order\_Items から作成された XML ターゲット定義の例を示します。



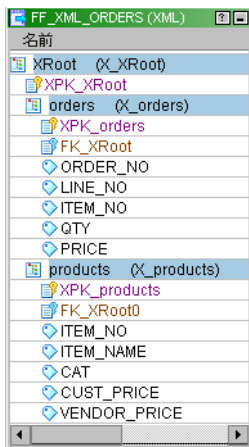
| 名前                          | XPath       | データ型        |
|-----------------------------|-------------|-------------|
| XRoot (X_XRoot)             |             |             |
| XPK_XRoot                   |             | xsd:integer |
| ORDERS (X_ORDERS)           |             | xsd:integer |
| XPK_ORDERS                  |             | xsd:integer |
| FK_XRoot                    |             | xsd:integer |
| ORDER_ID                    | ORDERS/O... | xsd:integer |
| DATE_ENTERED                | ORDERS/O... | xsd:date    |
| DATE_PROMISED               | ORDERS/O... | xsd:date    |
| DATE_SHIPPED                | ORDERS/O... | xsd:date    |
| EMPLOYEE_ID                 | ORDERS/O... | xsd:integer |
| CUSTOMER_ID                 | ORDERS/O... | xsd:integer |
| SALES_TAX_RATE              | ORDERS/O... | xsd:decimal |
| STORE_ID                    | ORDERS/O... | xsd:integer |
| ORDER_ITEMS (X_ORDER_ITEMS) |             |             |
| XPK_ORDER_ITEMS             |             | xsd:integer |
| FK_ORDERS                   |             | xsd:integer |
| ITEM_ID                     | ORDER_IT... | xsd:integer |
| QUANTITY                    | ORDER_IT... | xsd:integer |
| DISCOUNT                    | ORDER_IT... | xsd:decimal |

ルートは XRoot です。Orders と Order Items が XRoot で囲まれます。Order\_Items には、Orders を参照している外部キーがあります。

## フラットファイルからのメタデータの作成

リポジトリからフラットファイル定義をインポートすると、XML 定義を作成できます。複数のフラットファイル定義をインポートした場合、Designer は各フラットファイルに対して、ビューを含む XML 定義を作成します。XML 定義において、ビューには互いに対する関係がありません。ルートビューを作成することを選択すると、Designer はルートに対して外部キーのあるビューを作成します。

以下の図に、フラットファイルの Orders および Products から作成された XML ソース定義の例を示します。



| 名前                    | XPath | データ型 |
|-----------------------|-------|------|
| XRoot (X_XRoot)       |       |      |
| XPK_XRoot             |       |      |
| orders (X_orders)     |       |      |
| XPK_orders            |       |      |
| FK_XRoot              |       |      |
| ORDER_NO              |       |      |
| LINE_NO               |       |      |
| ITEM_NO               |       |      |
| QTY                   |       |      |
| PRICE                 |       |      |
| products (X_products) |       |      |
| XPK_products          |       |      |
| FK_XRoot0             |       |      |
| ITEM_NO               |       |      |
| ITEM_NAME             |       |      |
| CAT                   |       |      |
| CUST_PRICE            |       |      |
| VENDOR_PRICE          |       |      |

Products および Orders は、ルートビューに対する外部キーを持ち、強調表示されています。

## XML ビューについて

XML 階層内の要素間の関係によって、PowerCenter 定義内の XML ビュー間の関係が定義されます。ソース定義の場合、ビュー間で関連付ける必要はありません。したがって、ソース定義のビューにはプライマ

リキーや外部キーは必要ありません。非正規化ビューは、どのビューからも独立している場合があります。ただし、ビューが他のビューに関連付けられているときにキーカラムを指定しない場合は、Designer がキーを生成します。

ターゲット定義の各ビューは、少なくとも 1 つの他のグループと関連付けられている必要があります。したがって、各ビューには他のビューとの関係を確認するためのキーが最低 1 つは必要です。ユーザがキーを指定しなかった場合、Designer がターゲットビューにプライマリキーと外部キーを作成します。Designer には任せないで、ユーザが自分でビューと関係を XML エディターに作成した場合、ビューに対するプライマリキーと外部キーを定義することができます。

Designer はプライマリキーまたは外部キーのカラムを作成する際に、接頭語の付いたカラム名を割り当てます。XML 定義の場合、生成されたプライマリキーカラムには XPK\_、生成された外部キーカラムには XFK\_ が接頭語として付けられます。プライマリキーを参照する外部キーの場合、Designer は接頭語 FK\_ を使用します。

たとえば、Designer が Sales グループのプライマリキーカラムを生成した場合、XPK\_Sales という名前が付けられます。Sales グループを他のグループに接続する外部キーカラムを Designer が生成した場合、そのカラムには XFK\_Sales という名前が付けられます。Designer が作成するカラム名に対して、ユーザはどの名前でも変更することができます。

マッピングに XML ソースが含まれる場合、セッションを実行すると、Integration Service はソース定義内で生成されたプライマリキー列に対する値を生成します。生成キーに対して、ユーザは開始値を設定することができます。

## カスタム XML ビューの作成

カスタムビューとは、ユーザが XML ウィザードや XML エディタで作成するグループのことです。ユーザが XML ウィザードでカスタムビューを作成すると、ウィザードはスキーマ内にすべてのコンポーネントを含むビューを作成します。XML エディタを使用した場合は、各ビューを定義し、コンポーネントを選ぶことができます。

ビュー内の要素およびビュー間の関係は、ユーザが定義をインポートしたときに Designer がリポジトリに作成するスキーマに依存しています。XML エディタでは、有効なビューに関するルールを用いて XML 定義を検証します。

## XML ビューのルールおよびガイドライン

ビューキーと関係を使用する際には、以下のルールとガイドラインを考慮してください。

- PowerCenter の XML 定義は、最大 400 のビューを持つことができます。
- ビューはプライマリキーを 1 つしか持てません。
- ビューは他のいくつかのビューと関係を持つことができ、複数の外部キーを持つこともできます。
- 1 つのカラムを、プライマリキーと外部キーの両方として使用することはできません。
- ソース定義のビューには、キーが必要ありません。
- ターゲット定義のビューは、少なくとも 1 つのキーが必要です。
  - ターゲットのルートビューにはプライマリキーが必要ですが、外部キーを必要としません。
  - ターゲットのリーフビューには外部キーが必要ですが、プライマリキーは必要ありません。
- 囲み要素をキーにすることはできません。

- 外部キーは、常に他のグループのプライマリキーを参照します。自己を参照するキーは使用できません。
- 生成された外部キーカラムは、生成されたプライマリキーカラムを必ず参照します。
- XML 階層内の要素の相対カーディナリティは、PowerCenter が XML 定義のビューを作成するときの方法に影響します。以下のルールは、要素が同一のグループに含まれる可能性のある場合を決定します。
  - 1 対 1 の関係にある要素は同じビューに属することができます。
  - 1 対多の関係にある要素は、同じ正規化ビューまたは非正規化ビューに入れることができます。
  - 多対多の関係にある要素は同じビューに入れることができません。

## 階層リレーションについて

階層ビュー関係を持つ XML 定義の要素は、ビューの親要素の下にそれぞれ階層で表示されます。複数出現要素は、ビューになる可能性があります。複合型はビューにはなれず、派生複合型固有の要素はどのビューにも出現しません。

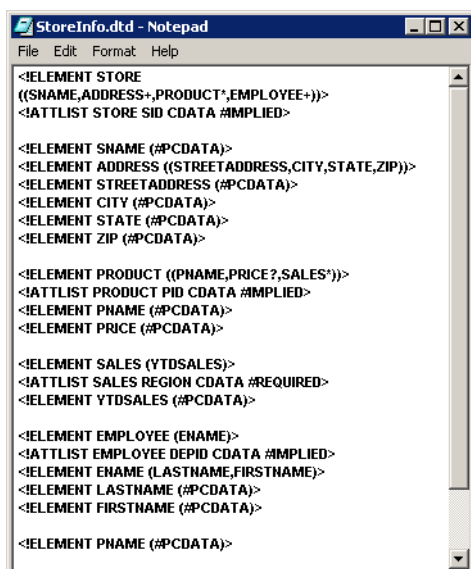
以下のタイプの階層ビューを生成できます。

- **正規化ビュー**。正規化ビューを持つ XML 定義では、複数出現データを個別のビューに分離することで冗長度を減らします。ビューは、プライマリキーおよび外部キーによって関連付けられています。
- **非正規化ビュー**。非正規化ビューを持つ XML 定義には、ビュー内の派生複合型に対して固有ではない階層の要素がすべてあります。ソース定義またはターゲット定義は、非正規化ビューを 1 つ含むことができます。

## 正規化ビュー

Designer は、XML 定義の正規化ビューを生成するときに、ルート要素および複数出現要素（XML 定義のビューになる）を確立します。

以下の図に、DTD ファイルおよび正規化 XML 定義のビューになる要素を示します。



Store はルート要素です。Address、Product、Employee、および Sales は複数出現要素です。

以下の図に、上図の DTD ファイルに基づくソース定義を示します。

| 名前                    | XPath       | データ型        |
|-----------------------|-------------|-------------|
| STORE (X_STORE)       |             |             |
| XPk_STORE             |             | xsd:integer |
| SID                   | STORE/@SID  | xsd:string  |
| SNAME                 | STORE/SN... | xsd:string  |
| ADDRESS (X_ADDRESS)   |             |             |
| XPk_AD...             |             | xsd:integer |
| FK_STO...             |             | xsd:integer |
| STREET...             | STORE/AD... | xsd:string  |
| CITY                  | STORE/AD... | xsd:string  |
| STATE                 | STORE/AD... | xsd:string  |
| ZIP                   | STORE/AD... | xsd:string  |
| PRODUCT (X_PRODUCT)   |             |             |
| XPk_PR...             |             | xsd:integer |
| FK_STO...             |             | xsd:integer |
| PID                   | STORE/PR... | xsd:string  |
| PNAME                 | STORE/PR... | xsd:string  |
| PPRICE                | STORE/PR... | xsd:string  |
| SALES (X_SALES)       |             |             |
| XPk_S...              |             | xsd:integer |
| FK_P...               |             | xsd:integer |
| REGION                | STORE/PR... | xsd:string  |
| SALES...              | STORE/PR... | xsd:string  |
| EMPLOYEE (X_EMPLOYEE) |             |             |
| XPk_EM...             |             | xsd:integer |
| FK_STO...             |             | xsd:integer |
| DEPID                 | STORE/EM... | xsd:string  |
| LASTNA...             | STORE/EM... | xsd:string  |

定義には、正規化ビューがあります。ルートビューは Store です。Address ビュー、Product ビュー、および Sales ビューには、Store に対する外部キーがあります。Sales ビューには Product ビューに対する外部キーがあります。

以下の表に、Store ビューのデータプレビュー内の行を示します。

| XPk_si_STORE | si_SID | si_NAME                         |
|--------------|--------|---------------------------------|
| 1            | BE1752 | Mud and Sawdust Furniture Store |

以下の表に、Address ビューのデータプレビュー内の行を示します。

| XPk_si_ADDRESS | FK_si_ADDRESS | si_STREETADDRESS   | si_CITY     | si_STATE | si_ZIP |
|----------------|---------------|--------------------|-------------|----------|--------|
| 1              | 1             | 335 Westshore Road | Fausta City | CA       | 95784  |
| 2              | 1             | 7415 Endless Loop  | Codeville   | CA       | 97412  |

以下の表に、Product ビューのデータプレビュー内の行を示します。

| XPk_si_PRODUCT | FK_si_PRODUCT | si_PNAME | si_PRICE | si_PID |
|----------------|---------------|----------|----------|--------|
| 1              | 1             | Chair    | 5690.00  | a1     |
| 1              | 1             | テーブル     | 1240.00  | a2     |
| 1              | 1             | Bed      | 1364.99  | a3     |

以下の表に、Sales ビューのデータプレビュー内の行を示します。

| XPK_si_SALES | FK_si_SALES | si_REGION | si_YTDSALES |
|--------------|-------------|-----------|-------------|
| 1            | a1          | Northwest | 4565.44     |
| 2            | a2          | South     | 8793.99     |
| 3            | a3          | East      | 23110.00    |
| 4            | a4          | South     | 5500.00     |
| 5            | a5          | Northwest | 10095.34    |
| 6            | a6          | East      | 200.00      |

以下の表に、Employee ビューのデータプレビュー内の行を示します。

| XPK_si_EMPLOYEE | FK_si_EMPLOYEE | si_FIRSTNAME | si_LASTNAME |
|-----------------|----------------|--------------|-------------|
| 1               | 1              | James        | Bond        |
| 2               | 1              | Austin       | Powers      |
| 3               | 1              | Indiana      | Jones       |
| 4               | 1              | Foxie        | Brown       |
| 5               | 1              | Bonnie       | Bell        |
| 6               | 1              | Laura        | Croft       |

## 非正規化ビュー

Designer は非正規化ビューを生成する場合、1つのビューを作成し、階層の要素をすべてそのビューに入れます。非正規化ビューの要素はすべて同じ親チェーンに属しています。非正規化テーブルなどの非正規化ビューは、重複したデータを生成します。

複数出現要素が 1 対多関係を持ち、すべて同じ親チェーンに属している場合、Designer は複数の複数出現要素を含む XML 定義の非正規化ビューを生成することができます。

以下の図に、複数出現要素（この場合は Product と Sales）を含む DTD ファイルを示します。

```

ProdAndSales.dtd - Notepad
File Edit Search Help
<?ELEMENT STORE (SNAME, PRODUCT*)>
<?ATTLIST STORE SID CDATA #REQUIRED>
<?ELEMENT SNAME (#PCDATA)>

<?ELEMENT PRODUCT (PNAME, PPRICE?, SALES*)>
<?ATTLIST PRODUCT PID ID #REQUIRED>
<?ELEMENT PNAME (#PCDATA)>
<?ELEMENT PPRICE (#PCDATA)>

<?ELEMENT SALES (YTDSALES)>
<?ATTLIST SALES REGION CDATA #REQUIRED>
<?ELEMENT YTDSALES (#PCDATA)>

```



Product および Sales は、複数出現要素です。複数出現要素は 1 対多関係を持っているので、Designer はすべての要素を含む非正規化ビューを 1 つ作成できます。

以下の図に、ProdAndSales.dtd に対応するソース定義の非正規化ビューを示します。

| 名前              | XPath    | データ型       |
|-----------------|----------|------------|
| SALES (X_STORE) |          |            |
| YTDsales        | YTDsales | xsd:string |
| REGION          | ./REGION | xsd:string |
| PRICE           | ./PRICE  | xsd:string |
| PNAME           | ./PNAME  | xsd:string |
| PID             | ./PID    | xsd:string |
| SNAME           | ./SNAME  | xsd:string |
| SID             | ./SID    | xsd:string |

Designer で、ProdAndSales 階層のすべての要素に対するビューが 1 つ作成されます。DTD ファイルはデータ型を定義しないため、Designer ではすべての列に文字列のデータ型が割り当てられます。この非正規化ビューには、プライマリキーまたは外部キーは必要ありません。

以下の図に、非正規化ビューのデータプレビューを示します。

| SID    | SNAME                           | PID | PNAME   | PPRICE  | REGION    | YTDsales |
|--------|---------------------------------|-----|---------|---------|-----------|----------|
| BE1752 | Mud and Sawdust Furniture Store | a1  | Chair   | NULL    | NorthWest | 3124.45  |
| BE1752 | Mud and Sawdust Furniture Store | a1  | Chair   | NULL    | South     | 12434.45 |
| BE1752 | Mud and Sawdust Furniture Store | a2  | Table   | 4126.33 | South     | 8252.66  |
| BE1752 | Mud and Sawdust Furniture Store | a3  | Bed     | 4356.33 | NorthWest | 43563.30 |
| BE1752 | Mud and Sawdust Furniture Store | a3  | Bed     | 4356.33 | South     | 21781.65 |
| BE1752 | Mud and Sawdust Furniture Store | a3  | Bed     | 4356.33 | East      | 26137.98 |
| BE1752 | Mud and Sawdust Furniture Store | a4  | Etagere | 5000.00 | South     | 20000.00 |
| BE1752 | Mud and Sawdust Furniture Store | a4  | Etagere | 5000.00 | East      | 5000.00  |
| BE1752 | Mud and Sawdust Furniture Store | a4  | Etagere | 5000.00 | NorthWest | 15000.00 |

## エンティティリレーションについて

XML スキーマからエンティティ関係を作成できます。エンティティ関係を含む XML 定義を作成する際、Designer は複数出現型、要素グループ、および複合型に対してそれぞれ個別のビューを生成します。Designer は、すべての派生複合型のビューを含みます。Designer は、型関係および階層関係に基づいてビュー間のリンクとキーを作成します。

XML スキーマを使用する際には、スキーマコンポーネント内で同じ情報を繰り返さずに、スキーマの他の部分への参照を使うことができます。コンポーネントは、他のコンポーネントの要素と属性を継承することができます。そのコンポーネントからの要素を制限または拡張することができます。たとえば、複合型を新規に作成するときに複合型をベースに使用したいことがあります。新規の型に要素を追加すれば、拡張複合型を作成することができます。あるいは、別の複合型のサブセットである限定複合型を作成することもできます。

ビューを手動で作成する場合、または XML エディタ でエンティティリレーションを再作成する場合、メタデータをどのように構造化するか選択します。継承を使用する XML スキーマに基づいて XML 定義を作成する場合、ベース型と派生型に対するビューを個別に作成することができます。XML データを正規化リレーショナルテーブルにマッピングする場合、継承リレーションを作成することができます。

XML タイプ I の継承リレーションは、2 つのビューの間のリレーションです。各ビューのルートは、グローバルな複合型です。ビューは、別のビューから導出されます。

カラムとビューの間に継承リレーションを作成できます。これは XML タイプ II の継承リレーションです。

Designer は、置き換えグループの個別ビューを生成します。



## エンティティリレーションのルールおよびガイドライン

Designer は、以下のガイドラインに基づいてエンティティを生成します。

- エンティティは、XML、DTD、または XML スキーマ階層の一部を表します。この階層は、XML ファイルのルートから始める必要はありません。
- Designer は、DTD ファイルで定義されているエンティティを使用してエンティティ関係を作成します。
- Designer は、XML スキーマで定義されている型構造を使用してエンティティリレーションを生成します。
- Designer は、親要素の下で複数出現要素を検出すると、新しいエンティティを作成します。
- Designer は、置き換えグループの各メンバに対して個別ビューを生成します。
- Designer は、プライマリキーと外部キーを生成して個々のエンティティを関連付けます。

### タイプ 1 のエンティティリレーションの例

XML タイプ 1 のエンティティリレーションは、2 つのビューの間のリレーションです。各ビューは、グローバルな複合型をルートとしていることが必要です。ひとつのビューはもう一方のビューから導出される必要があります。

次のスキーマには PublicationType、BookType、および MagazineType が含まれています。PublicationType はベース型です。パブリケーションには、Title、Author、および Date が含まれます。BookType と MagazineType は派生型であり、PublicationType を拡張します。Book には ISBN と Publisher があり、Magazine には Volume と Edition があります。

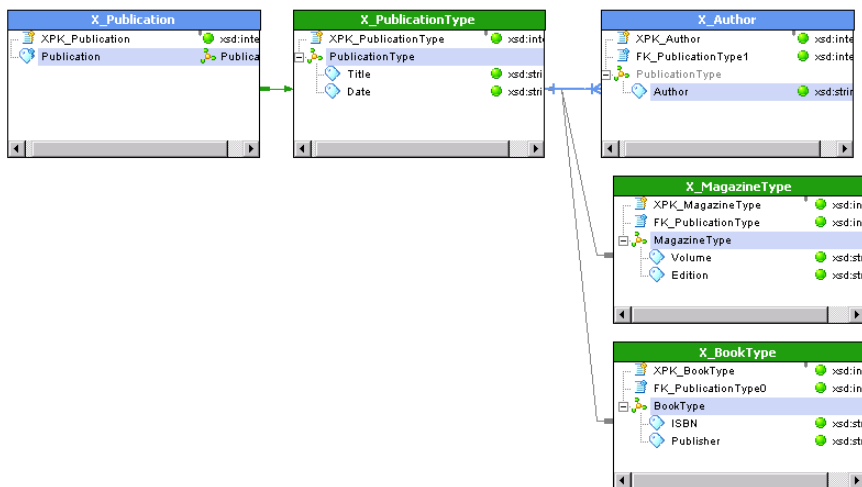
```
<xsd:complexType name="PublicationType">
  <xsd:sequence>
    <xsd:element name="Title" type="xsd:string"/>
    <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
    <xsd:element name="Date" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="Publication" type="PublicationType"/>
<xsd:complexType name="BookType">
  <xsd:complexContent>
    <xsd:extension base="PublicationType">
      <xsd:sequence>
        <xsd:element name="ISBN" type="xsd:string"/>
        <xsd:element name="Publisher" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="MagazineType">
  <xsd:complexContent>
    <xsd:extension base="PublicationType">
      <xsd:sequence>
        <xsd:element name="Volume" type="xsd:string"/>
        <xsd:element name="Edition" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
</xsd:schema>
```

XML ビューを XML 定義のエンティティとして作成した場合、Publication Type の Title および Date メタデータが BookType、または MagazineType ビューで重複することはありません。代わりに、これらのビューは、PublicationType と区別するメタデータを含みます (BookType の ISBN と Publisher、MagazineType の Volume と Edition)。これらには外部キーがあり、PublicationType とリンクしています。

この例では、基本型の要素のうち派生型で反復するものはないため、メタデータ膨張を抑制する機能を使用します。

Author は、Publication 内の複数出現要素です。Author が XML ビューになります。

以下の図に、Designer がスキーマから作成するデフォルトのビューを示します。



以下の図に、パブリケーション、雑誌、書籍が含まれる XML ファイルを示します。

```

Bookstore_TYPES.xml - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
<?xml version="1.0"?>
<bk:BookStore xmlns:bk="http://www.books.org"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="
                   http://www.books.org
                   BookStore.xsd">

  <Publication>
    <Title>Staying Young Forever</Title>
    <Author>Karin Granstrom Jordan, M.D.</Author>
    <Date>1999</Date>
  </Publication>

  <Publication xsi:type="bk:BookType">
    <Title>Illusions The Adventures of a Reluctant Messiah</Title>
    <Author>Richard Bach</Author>
    <Date>1977</Date>
    <ISBN>0-440-34319-4</ISBN>
    <Publisher>Dell Publishing Co.</Publisher>
  </Publication>

  <Publication xsi:type="bk:MagazineType">
    <Title>The First and Last Freedom</Title>
    <Author>J. Krishnamurti</Author>
    <Date>1954</Date>
    <Volume>IV</Volume>
    <Edition>Spring</Edition>
  </Publication>

  <Publication xsi:type="bk:BookType">
    <Title>working with XML</Title>
    <Author>Dell Skidmore</Author>
    <Date>1999</Date>
    <ISBN>0-444-84429-4</ISBN>
    <Publisher>Purdum Publishing Co.</Publisher>
  </Publication>

</bk:BookStore>

```

前の図に示した XML 定義を使用して XML ファイルの例を処理する場合、以下のビューでデータを作成します。

- **PublicationType ビュー。**それぞれのパブリケーションのタイトルと日付が含まれます。

以下の図に、PublicationType ビューを示します。

| XPK_boo_PublicationType | Title           | Date |
|-------------------------|-----------------|------|
| 1                       | Staying Yo...   | 1999 |
| 2                       | Illusions Th... | 1977 |
| 3                       | The First a...  | 1954 |
| 4                       | Working wi...   | 1999 |

- **BookType ビュー。**ISBN および出版社が含まれます。また、PublicationType に対する外部キーも含まれます。

以下の図に、BookType ビューを示します。

| XPK_boo_BookType | FK_boo_PublicationType1 | ISBN          | Publisher             |
|------------------|-------------------------|---------------|-----------------------|
| 1                | 2                       | 0-440-34319-4 | Dell Publishing Co.   |
| 2                | 4                       | 0-444-84429-4 | Purdum Publishing Co. |

- **MagazineType ビュー。**巻と版が含まれます。また、PublicationType に対する外部キーも含まれます。

以下の図に、MagazineType ビューを示します。

| XPK_boo_MagazineType | FK_boo_PublicationType | Volume | Edition |
|----------------------|------------------------|--------|---------|
| 1                    | 3                      | IV     | Spring  |

- **Author ビュー。**すべてのパブリケーションの作成者が含まれます。Author は複数出現要素なので、Designer はこの要素を別のビューとして生成します。それぞれの出版物に複数の Author を対応させることができます。

以下の図に、Author ビューを示します。

| XPK_Author | FK_boo_PublicationType0 | Author                       |
|------------|-------------------------|------------------------------|
| 1          | 1                       | Karin Granstrom Jordan, M.D. |
| 2          | 2                       | Richard Bach                 |
| 3          | 3                       | J. Krishnamurti              |
| 4          | 4                       | Dell Skidmore                |

## タイプ II のエンティティリレーションの例

カラムと複合型ビューとの間の継承リレーションを作成できます。カラムは、ローカルの複合型の要素である必要があります。ビューは、グローバルな複合型をルートとしている必要があります。そして、ここでのローカルな複合型は、このグローバルな複合型から導出されている必要があります。

たとえば、次のスキーマは EmployeeType という複合型を定義します。EmployeeType には、EmployeeNumber 要素および EmployeeName 要素が含まれます。

EmployeeStatusType は、EmployeeType を拡張する Employee という要素を含みます。Employee は、EmployeeStatus 要素を含みます。

```
<xs:element name="Employee_Payroll">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="EmployeeStatus" type="EmpStatusType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="EmpStatusType">
  <xs:sequence>
    <xs:element name="Employee" minOccurs="0" maxOccurs="1">
```

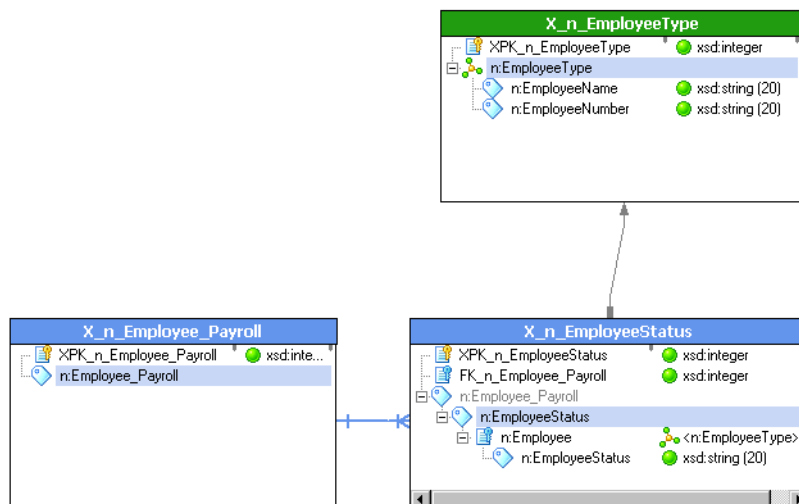
```

<xs:complexType>
<xs:complexContent>
<xs:extension base="EmployeeType">
  xs:sequence>
    <xs:element name="EmployeeStatus" type="xs:string">
      </xs:element>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="EmployeeType">
  <xs:sequence>
    <xs:element name="EmployeeName" type="xs:string"></xs:element>
    <xs:element name="EmployeeNumber" type="xs:string"></xs:element>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

スキーマをインポートすると、Designer は Employee\_Payroll、EmployeeType、および EmployeeStatus のビューを作成します。EmployeeStatus ビューには、Employee というカラムが含まれます。Employee は EmployeeType から派生します。

以下の図に、Employee\_Payroll ビュー、EmployeeType ビュー、および EmployeeStatus XML ビューを示します。



Employee\_Payroll ビューには、Employee\_Payroll 要素とプライマリキー PK\_Employee\_Payroll が含まれています。Employee\_Payroll ビューは EmployeeStatus ビューに青い線で接続されています。これは、ビューの間に 1 対多の関係があることを示しています。Employee\_Payroll には EmployeeStatus の複数の出現が含まれています。

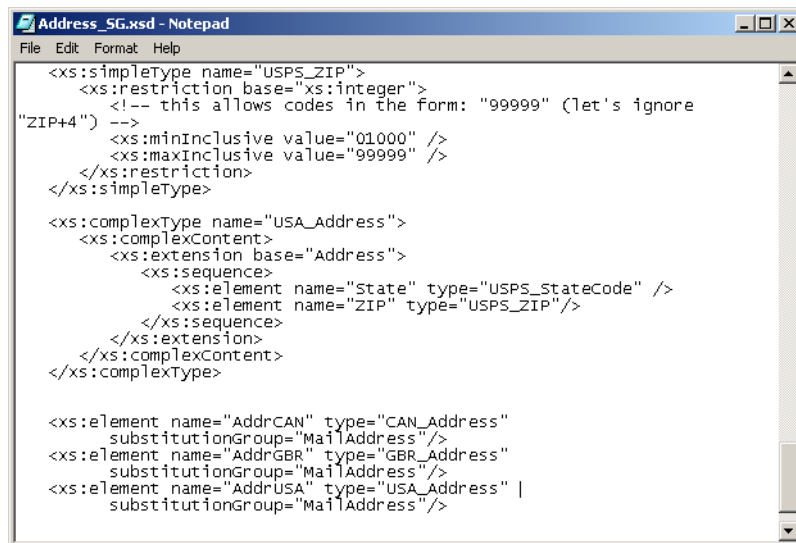
EmployeeStatus ビューには、EmployeeType タイプの Employee 要素が含まれます。Employee 要素は、EmployeeStatus 要素を含めることにより、EmployeeType を拡張します。EmployeeType ビューには Employee\_Payroll ビューに対する外部キーも含まれています。EmployeeStatus ビューは EmployeeType ビューに灰色の矢印で接続されています。この矢印は、ビュー間のリレーションを示します。

EmployeeStatus ビューには、EmployeeName と EmployeeNumber で構成される EmployeeType が含まれています。

## XML 定義での置き換えグループの使用

エンティティ関係を含む XML 定義を作成する際、Designer は要素グループ用および複合型用の別のビューを生成します。置き換えグループを使用する XML スキーマをインポートすると、Designer は置き換えグループの各メンバを別々のエンティティとしてインポートします。Designer は、各グループのビューを個別に作成します。

以下の図に、代替グループ MailAddress を含む XML スキーマサンプルの一部を示します。



以下の図に、XML 定義と代替グループの各メンバー（AddrCAN、AddrGBR、AddrUSA、ShortAddress、および Street）を示します。

| 名前                            | XPath          | データ型          |
|-------------------------------|----------------|---------------|
| AddrCAN (X_AddrCAN)           |                |               |
| XPk_Addr...                   |                | xsd:integer   |
| AddrCAN                       | AddrCAN        | xsd:integer   |
| Province                      | AddrCAN/Pr...  | xsd:string    |
| PostalCode                    | AddrCAN/P...   | CAN_Postal... |
| AddrGBR (X_AddrGBR)           |                |               |
| XPk_Addr...                   |                | xsd:integer   |
| AddrGBR                       | AddrGBR        | xsd:integer   |
| County                        | AddrGBR/C...   | xsd:string    |
| Postcode                      | AddrGBR/P...   | GBR_Postc...  |
| AddrUSA (X_AddrUSA)           |                |               |
| XPk_Addr...                   |                | xsd:integer   |
| AddrUSA                       | AddrUSA        | xsd:integer   |
| State                         | AddrUSA/St...  | USPS_State... |
| ZIP                           | AddrUSA/ZIP    | USPS_ZIP      |
| ShortAddress (X_ShortAddress) |                |               |
| XPk_Short...                  |                | xsd:integer   |
| Name                          | type(ShortA... | xsd:string    |
| City                          | type(ShortA... | xsd:string    |
| Street (X_Street2)            |                |               |
| XPk_Str...                    |                | xsd:integer   |
| FK_Shor...                    |                | xsd:integer   |
| Street                        | type(ShortA... | xsd:string    |

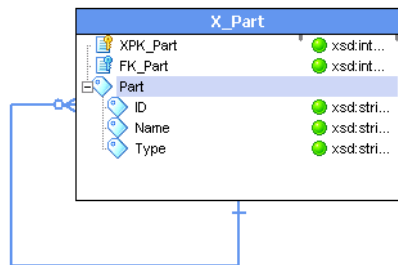
## 循環参照に関する作業

循環関係とは、XML 定義にある 2 つのビュー間の循環階層関係、または XML 定義内の単独ビューのことです。たとえば、Part と呼ばれる複合要素であれば、ID および部品名だけでなく他の部品への参照が含まれる可能性があります。

次の例は、Part 要素コンポーネントの例を示します。

```
<xs:element name="Part">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ID" type="xs:string"/>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="Type" type="xs:string"/>
      <xs:element ref="Part" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

次の図は、Part という複雑な要素がある XML エディタワークスペースの循環参照を示します。



Part の XML 定義を使用して、セッション内にある次の XML ファイル例をユーザが読み出したい場合があります。

```
<Part>
  <ID>1</ID>
  <Name>Big Part</Name>
  <Type>L</Type>
  <Part>
    <ID>1.A</ID>
    <Name>Middle Part</Name>
    <Type>M</Type>
    <Part>
      <ID>1.A.B</ID>
      <Name>Small Part</Name>
      <Type>S</Type>
    </Part>
  </Part>
</Part>
```

この XML ファイルには、Part 1 に Part 1.A が含まれ、Part 1.A に Part 1.A.B が含まれています。

以下の図に、XML ソースからのセッションで作成される可能性のあるデータとキーの例を示します。

| XPK_Part | FK_Part | ID    | Name        | Type |
|----------|---------|-------|-------------|------|
| 1        | NULL    | 1     | Big Part    | L    |
| 2        | 1       | 1.A   | Middle Part | M    |
| 3        | 2       | 1.A.B | Small Part  | S    |

**注:** セッションが制約に基づくロードで有効になっている場合、循環 XML 参照を含むセッションは実行できません。セッションはすべての行を拒否します。

## ビュー行について

XML ドキュメントからデータを抽出するには、生成する行、取り込むデータの列、および行の生成タイミングを指定する必要があります。XML エディタでビューを定義する場合は、データ行の生成で Integration Service が必要とするビュー行、要素またはグローバルな複合型を作成します。

Integration Service では、ビュー行を使用して XML ビューに対するデータを読み書きするタイミングを決定します。ユーザは、ビュー行を任意の単独出現要素または複数出現要素に設定することができます。ビュー設定後は、ユーザがビューに追加する要素が必ず、ビューと 1 対 1 の対応になります。

たとえば、Employee ビューには Employee、Name、Firstname、Lastname などの要素が含まれます。ビュー行を Employee に設定すると、Integration Service は次のアルゴリズムに従ってデータを抽出します。

```
For every (Employees/Employee)
extract ./Name/Firstname/Lastname
```

Employees の XML スキーマには次の要素が含まれることがあります。

```
EMPLOYEES
  EMPLOYEE+
    ADDRESS+
    NAME
    FIRSTNAME
    LASTNAME
    EMAIL+
```

Employee、Address、および Email は複数出現要素です。次の要素を含むビューを作成できます。

```
EMPLOYEE
  ADDRESS
  NAME
```

ビュー行を Address に設定すると、Integration Service は XML データのすべての Employee/Address に対応する Name を抽出します。Address と Email との間の多対多関係をユーザが作成することになるので、このビューに Email を追加することはできません。

このビューに Pivot 化された複数出現カラムを追加することができます。各ビュー行に Pivot 化カラムを含むことが可能です。



たとえば、1 インスタンス分の Email を Pivot 化カラムとしてビュー Employee に追加することができます。このビューには次の要素が含まれることがあります。

```
EMPLOYEE
  ADDRESS
  NAME
  EMAIL[1]
```

ビューは、EMPLOYEE/ADDRESS/EMAIL[1] のビュー行を持っている場合があります。Integration Service は、Employee/Address/Email の最初のインスタンスに対応するデータを抽出します。

## XPath クエリーの述部の使用

XML ビューのクエリーを使用して、XML ソースデータをフィルタします。Integration Service は、クエリーに基づいて、ソース XML ファイルからデータを抽出します。クエリーが True であれば、Integration Service はビューからデータを抽出します。

XML ビューにクエリーを作成するには、XML エディタに XPath クエリーの述部を作成します。XPath は、XML 文書内の項目の位置を特定する方法を記述する言語です。XPath は、ルートコンポーネントからの XML 階層のパスに基づくアドレス構文を使用します。ビュー行の要素に対して XPath クエリーの述部を作成したり、ビュー行を含む XPath のある要素および属性を作成したりすることができます。

XPath クエリーの述部には抽出する要素や属性が含まれており、クエリーの述部は基準を決定します。要素または属性の値の確認や、ソース XML データに要素または属性が存在することの確認を行うことができます。

## ビュー行を使用するためのルールおよびガイドライン

XML 定義でビュー行を使用するときは、次のルールおよびガイドラインを使用します。

- ビュー行は、型または要素である必要があります。ビュー行は属性にはなりません。
- すべてのビューにはビュー行があり、ビュー行は要素または複合型である必要があります。
- ビュールートは、ビュー内の最上位レベルの要素です。ビュールートは、ビュー内にある他のすべての要素の親に相当します。
- ビュー行は、ビューが非正規化でなければ、ビュールートと同じです。
- XML ソースまたは XML Parser トランスフォーメーションの中で、2 つのビュー行が同じビュー行を持つことができます。
- ビュー行要素は、ビュー内の最下位の複数出現要素でなければなりません。ビューに多対多関係を含むことはできません。
- ある複数出現要素を他の複数出現要素を持つビューに追加した場合、デフォルトでは、ビュー行が新規要素に変更されます。ビューに複数出現要素が既に存在する場合、別の複数出現要素を追加することはできません。
- 空のビューを作成する場合は、ビュー行を指定する必要はありません。ただし、ビューにカラムを追加すると、Designer がビュー行を作成します。これは、プライマリキーを追加しているだけの場合にも該当します。
- 後でビュー行を変更することはできますが、ビュー内にスキーマコンポーネントがない場合を除いて、ビュールートを変更することはできません。
- 下記のような Pivot 化要素で構成されているビュー行を指定することができます。

```
Product/Order[2]/Customer
```

- ビューの有効なビュー行は、階層関係の最上位からビュー内のビュー行までのビュー行のパスです。ビューは、XML 定義内に複数の階層関係を持つことができるので、複数の有効ビュー行を持つことができます。

XML エディタにオプションを指定すれば、ビュー行と有効ビュー行がデータ出力に影響するときの状況を制御することができます。

## 列のピボット化

複数回出現する要素の中には、異なる値を持つ同じ要素のセットもあります。たとえば、12 回出現する要素 Sales が、各月の売上高を含む場合があります。また、2 回出現する要素 Address が、自宅住所と勤務先住所である場合があります。

XML ソースにこのタイプの要素が含まれる場合、ピボット化を使用して要素の出現をグループの個別のカラムとして扱うようにします。XML ビューで要素の出現を Pivot 化する場合、定義内で表現したい出現のそれぞれに対してカラムを作成します。月ごとの売上の例では、12 回の出現をすべてカラムとして表したい場合、12 個の売上カラムをビューに作成します。各四半期の売上を表したい場合は、カラムを 3 つ作成します。セッションの実行時、Integration Service は、定義に指定されていない出現に対応する XML データがあればこれを無視します。

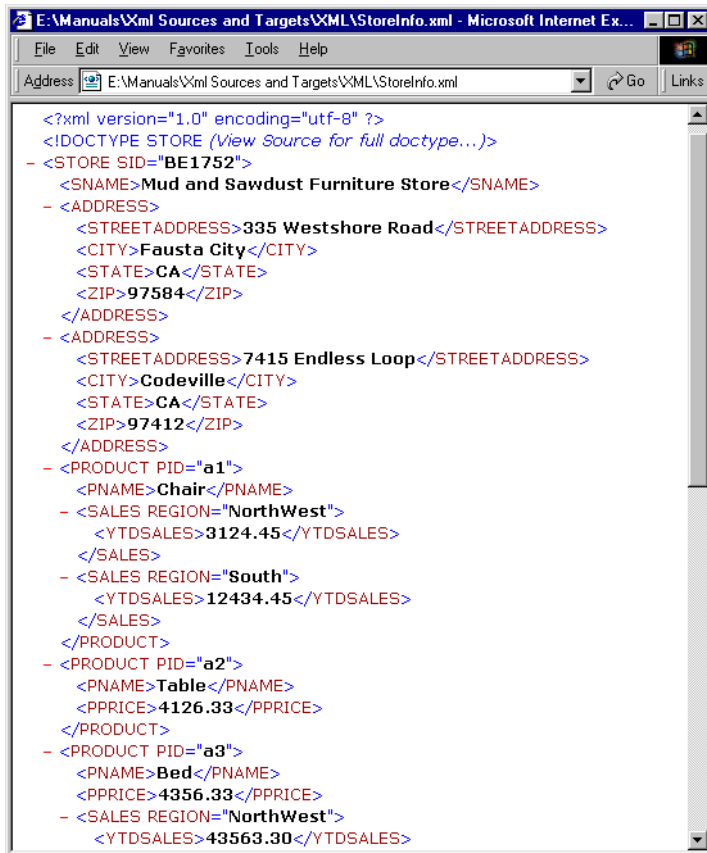
XML ソース定義でビューを追加または編集する際に、列をピボット化することができます。

単純型と複合型をピボット化できます。プライマリキーカラムを Pivot 化することはできません。ビューのカラムを Pivot 化する場合、結果のグループ構造が有効な正規化ビューまたは非正規化ビューのルールに従うように注意してください。ピボット化カラムがビューを無効にした場合、Designer は警告およびエラーを表示します。

ピボット化は、ピボット化した要素を含むビュー内の要素に影響を与えます。ビュー内で要素をピボット化する場合、他のビューの同じ要素は変更しません。

**注:** XML ターゲットのカラムを Pivot 化することはできません。

以下の図に、StoreInfo XML ファイルで 2 回出現する Address 要素を示します。



最初に出現する Address は、接頭辞「HOM\_」を持つ自宅住所列にピボット化されます。2 番目に出現する Address は、接頭辞「OFC\_」を持つ勤務先住所列にピボット化されます。XPath は、同じ要素に由来するカラムの 2 つのセットを示します。

以下の図に、アドレスカラムの 2 つのセットにピボット化された StoreInfo XML ファイルの要素 ADDRESS を示します。

| PowerCenter Column ... | Type            | Not ...                  | XPath                                       |
|------------------------|-----------------|--------------------------|---|
| SID                    | xsd:string (30) | <input type="checkbox"/> | <a href="#">./@SID</a>                      |
| si_SNAME               | xsd:string (30) | <input type="checkbox"/> | <a href="#">./si:SNAME</a>                  |
| HOM_STREETADDRESS      | xsd:string (30) | <input type="checkbox"/> | <a href="#">./si:ADDRESS1/STREETADDRESS</a> |
| HOM_CITY               | xsd:string (30) | <input type="checkbox"/> | <a href="#">./si:ADDRESS1/CITY</a>          |
| HOM_STATE              | xsd:string (30) | <input type="checkbox"/> | <a href="#">./si:ADDRESS1/STATE</a>         |
| HOM_ZIP                | xsd:string (30) | <input type="checkbox"/> | <a href="#">./si:ADDRESS1/ZIP</a>           |
| OFC_STREETADDRESS      | xsd:string (30) | <input type="checkbox"/> | <a href="#">./si:ADDRESS2/STREETADDRESS</a> |
| OFC_CITY               | xsd:string (30) | <input type="checkbox"/> | <a href="#">./si:ADDRESS2/CITY</a>          |
| OFC_STATE              | xsd:string (30) | <input type="checkbox"/> | <a href="#">./si:ADDRESS2/STATE</a>         |
| OFC_ZIP                | xsd:string (30) | <input type="checkbox"/> | <a href="#">./si:ADDRESS2/ZIP</a>           |

以下の図では、最初と 2 つ目のアドレスオカレンス（HOM\_および OFC\_プレフィックス）が、グループのカラムとして表示されています。

| GPX ADDRESS | FK SID | HOM_STREETADDRESS        | HOM_CITY    | HOM_STATE | HOM_ZIP | OFC_STREETADDRESS | OFC_CITY  | OFC_STATE | OFC_ZIP |
|-------------|--------|--------------------------|-------------|-----------|---------|-------------------|-----------|-----------|---------|
| 1           |        | 1 335 West Bayshore Road | Fausta city | CA        | 97584   | 7415 Endless Loop | Codeville | CA        | 97412   |

## 複数レベルピボット化の使用

ビュー内の要素を複数レベルでピボット化できます。この場合、カラムの XPath で複数出現要素に対応する固定オフセットを指定します。たとえば、ビュー内に次の要素があると仮定します。

```
STORE
  PRODUCT+
    PNAME
  ORDER+
    ORDERNAME
  CUSTOMER+
    CUSTNAME
```

XPath の STORE/PRODUCT[2]/ORDER[1]/ORDERNAME は、商店にある 2 番目の製品で最初の注文の注文名を意味します。XPath の STORE/PRODUCT[2]/ORDER/CUSTOMER[1] は、2 番目の製品に対するすべての注文で最初の顧客を意味します。

ビュー行を Pivot 化する場合、ビュー行の下側で出現する XML ビューのカラムについては、ビュー行の XPath と一致する XPath が必ず要求されます。

たとえば、ビューに次のビュー行があると仮定します。

```
Transaction/Trade[1]
```

以下のカラムでは、XPath で Trade が同時に出現します。

```
Transaction/Trade[1]/Date
Transaction/Trade[1]/Price
Transaction/Trade[1]/Person[1]/Firstname
```

ビューの場合、次の XPath でカラムを作成することはできません：

```
Transaction/Trade[2]/Date
```

## XML ソースに関する作業

### XML ソースに関する作業の概要

Designer には、リポジトリに XML 定義を作成することができる XML ウィザードが用意されています。URL またはローカルノードからファイルをインポートして、XML 定義を作成することができます。また、PowerCenter リポジトリからリレーショナルまたはフラットファイル定義をインポートすることもできます。XML 定義は次のファイルタイプから作成することができます。

- XML ファイル
- XML スキーマファイル
- DTD ファイル
- リレーショナル定義
- フラットファイル定義

XML 定義を作成する場合、XML ウィザードでファイルをインポートし、XML ビューのメタデータを編集します。XML ビューは、XML ファイル内の要素と属性を含むカラムのグループです。ウィザードでビューを生成することができますが、ユーザーがカスタムビューを作成することもできます。

XML ウィザードでは、ビュー間のリレーションを作成できます。階層リレーションまたはエンティティリレーションを作成できます。

スキーマの構造が変化する場合は、XML 定義を XML スキーマファイルに同期させることができます。

## XML ソース定義のインポート

XML スキーマまたは DTD ファイルからソース定義をインポートした場合、Designer は DTD または XML スキーマファイルに含まれる説明に基づいて、データを正しく定義することができます。関連 DTD または XML スキーマを持たない XML ファイルに基づいてソース定義をインポートした場合、XML ウィザードは XML ファイルに示されるデータに基づいてデータの型と出現を判断します。XML 定義を作成する場合は、予期しない結果が発生することがあります。たとえば、Designer が文字列カラムに不正な位取り属性を定義する可能性があります。不正な位取り属性で XML ソース定義のエクスポートやインポートを行うと、エラーが発生します。

XML ソース定義を作成すると、そのソース定義を他のソースタイプに変更することはできません。逆に、他のタイプのソース定義を XML 定義に変更することはできません。

XML ウィザードは、キーを使って XML ビューを関連付け、XML 階層を再構築します。ビューとプライマリキーの生成を選択することも、自分でビューを作成してキーを指定することもできます。カスタムビューを作成する場合は、ルートを選択して、メタデータの拡張をどのように扱うかを選択できます。

XML ウィザードは、XML 階層およびビュー情報を XML スキーマとしてリポジトリに保存します。XML 定義をインポートする場合、階層の要素のカーディナリティおよびデータ型を変更できるかどうかは、インポートするファイルのタイプによって異なります。たとえば、DTD ファイルと XML ファイルにはデータ型情報が格納されません。XML ファイルや DTD ファイルをインポートして XML 定義を作成する際は、Designer でデータ型、精度および位取りを指定することができます。XML スキーマをインポートする場合は、精度と位取りを変更することができます。

エクスポートされたリポジトリオブジェクトの XML ファイルから XML ソース定義を作成することはできません。ソース定義をインポートすると、Designer によってリポジトリ内の XML 定義にデフォルトのコードページが適用されます。コードページは、PowerCenter クライアントのコードページに基づきます。XML ソース定義のコードページは変更できませんが、XML ターゲット定義のコードページは作成後に変更できます。

XML ウィザードを使用して、XML ソース定義をインポートします。

XML ファイルをインポートするには：

1. [ソース] - [XML 定義のインポート] をクリックします。  
[XML 定義のインポート] ダイアログボックスが表示されます。
2. [Advanced Options] をクリックします。  
[XML View 生成及びネーミング変更オプション] ダイアログボックスが表示されます。Designer が XML ビューを作成して名前を付ける方法を指定するには、オプションを選択します。

以下の表に、XML ビューオプションを示します。

| オプション  | 説明  |
|--|---|
| すべての長さの infinite 指定を以下の値でオーバーライドする                         | 文字列など、長さが定義されていないコンポーネントのデフォルトの長さを指定することができます。デフォルトの長さを設定していない場合、そのコンポーネントの精度は無限に設定されます。これによって、大きなファイルを扱うセッションを実行したときに DTM バッファサイズエラーが生じる可能性があります。  |
| スタンドアロン XML の要素/属性をグローバル宣言として解析する                          | このオプションは、スタンドアロン XML 要素または属性のグローバル宣言を作成するときに選択します。グローバルな要素は、スキーマ内の別の部分で参照し、再利用できます。このオプションをクリアすると、スタンドアロン XML はローカルな宣言になります。  |
| 囲み要素の XML ビューを作成する   | 複数回出現可能な囲み要素および複数回出現可能な子要素がある場合、囲み要素に対して個別のビューを作成できます。囲み要素とは、テキスト内容や属性がないが子要素がある要素です。   |
| max Occurs が指定されていない場合に要素を Pivot しますか？                     | リーフ要素で Pivot 化できるのは出現制限のあるものだけです。Pivot 化できるのは、ソース定義内の要素のみです。  |
| fixed 要素/属性はスキーマにより固定値が指定されている要素/属性です。XMLView では上記を無視しますか？ | スキーマ内の固定値を無視し、データ内の他の要素値を許可することができます。   |
| prohibited 属性を XMLView で無視しますか？                            | XML スキーマでは、属性を禁止として宣言できます。禁止された属性は複合型を制限します。スキーマまたはファイルをインポートするときに、禁止された属性を無視するように選択できます。   |
| XMLColumn 名を生成するときのオプションを指定してください                          | <p>XML カラムに名前を付ける場合、シーケンス番号を使用するか、あるいはスキーマの要素名または属性名を使用するかを選択できます。名前を使用する場合は、以下のオプションから選択します。</p> <ul style="list-style-type: none"> <li>- XMLColumn が属性を参照する場合、要素名に接頭語を付けます。PowerCenter では XML 列の名前に次の形式を使用します。<br/>NameOfElement_NameOfAttribute</li> <li>- すべての XML カラムの XML ビュー名に接頭語を付けます。PowerCenter では XML 列の名前に次の形式を使用します。<br/>NameOfView_NameOfElement</li> <li>- すべての外部キーカラムの XML ビュー名に接頭語を付けます。PowerCenter では生成された外部キー列の名前に次の形式を使用します。<br/>FK_NameOfView_NameOfParentView_NameOfPKColumn</li> </ul> <p>カラム名の最大長は 80 文字です。81 文字を超えるカラム名は、PowerCenter によって切り詰められます。カラム名が一意でない場合は、名前を一意にするために PowerCenter によって数字の接尾語が追加されます。</p> |

3. [OK] をクリックして変更を適用します。

4. インポートするファイルのタイプを選択します。以下のオプションを選択することができます。

- **ローカル XML ファイルまたは URL から定義をインポートする。**XML ファイル、DTD ファイル、または XML スキーマファイルからソース定義を作成します。XML ファイルに関連する DTD またはスキーマと一緒にインポートする場合、XML ウィザードは DTD またはスキーマを使用して XML ドキュメントを生成します。



- **非 XML ソースまたはターゲットからの定義をインポートする。** このオプションを使用して、フラットファイルまたはリレーショナル定義からソース定義を作成します。新しいソース定義には、入力定義ごとの 1 つのグループとルート要素グループが含まれます。

5. [次へ] をクリックして XML ウィザードを終了します。

## 複数行にまたがる属性値

XML ウィザードでは、改行文字を含む属性値や、1 行を超える属性値は使用できないことになっています。改行文字がある属性値を持つ XML ファイルからソース定義またはターゲット定義をインポートすると、XML ウィザードはエラーを表示し、ファイルをインポートしません。

## XML ビューに関する作業

Designer は、ビューをソース定義のグループとして表示します。

XML ウィザードには、ビューを定義で作成するオプションがあります。また、XML エディタでビューを作成することもできます。

XML ビューを作成するために、以下のオプションのいずれかを選択できます。

- **エンティティリレーションの生成。** エンティティリレーションを作成する場合、XML ウィザードは複数出現要素、参照要素、および複合型のビューを生成します。
- **階層リレーションの生成。** 階層関係を作成すると、コンポーネントへの各参照は親要素の下で展開します。階層リレーションの中に正規化 XML ビューまたは非正規化 XML ビューを生成することができます。
  - **正規化 XML ビュー。** 正規化 XML ビューを生成する場合、要素および属性は 1 度だけ表示されます。複数出現要素または 1 対多リレーションにある要素は、キーにより関連付けられた別のビューで表示されます。
  - **非正規化 XML ビュー。** 非正規化 XML ビューを生成する場合、すべての要素および属性が 1 つのビューに表示されます。Designer では、XML 定義の要素と属性の間で多対多リレーションをモデル化することはありません。
- **カスタム XML ビューの作成。** カスタム XML ビューを作成する場合は、任意のグローバル要素をルートとして指定できます。要素、複合型、および継承複合型に関するメタデータ膨張の抑制を選択できます。
- **XML 定義の同期。** ベースとなるスキーマが変更された場合は、1 つ以上の XML 定義を更新できます。
- **XML ビューの作成のスキップ。** XML ビューを作成しないように選択した場合は、後から XML エディタで定義できます。XML エディタでビューを定義する場合は、ターゲットに合うようにビューを定義して、マッピングを簡略化できます。
- **XML ファイル内の要素および属性に対する XML ビューの作成。** 関連スキーマと同時に XML ファイルをインポートすると、スキーマの全コンポーネントに対してではなく、XML ファイルの要素や属性のみの XML ビューを作成できます。

エンティティまたは階層リレーションの生成を選択した場合、Designer はデフォルトのルートを選択し、XML ビューを作成します。XML 定義に 400 を超えるビューが必要な場合は、定義が大きすぎることを示すメッセージが表示されます。XML エディタにビューを手動で作成できます。XML 定義をインポートして、カスタムビューの作成または XML ビュー生成のスキップのいずれかを選択します。



グローバル要素を持たない XML スキーマから定義をインポートした場合、Designer は XML 定義でルートビューを作成できません。グローバル要素が存在しないことを示すメッセージが Designer に表示されます。

XML ビューを作成した後で、ビューの設定オプションを変更することはできません。たとえば、正規化 XML ビューを作成した場合、これを非正規化ビューに変更することはできません。改めて XML ソース定義をインポートし、非正規化オプションを選択する必要があります。

PowerCenter での XML サイズ設定の詳細については、『[「PowerCenter での XML の使用の概要」 \(ページ 29\)](#)』を参照してください。PowerCenter で XML 処理に適用される制限の詳細については、『[「制限」 \(ページ 29\)](#)』を参照してください。

他の要素型を持つトランスフォーメーションを作成し、大きな XML 入力ファイルを変換するには、データプロセッサトランスフォーメーションを使用します。データプロセッサトランスフォーメーションの作成方法の詳細については、『*Informatica Data Transformation ユーザーガイド*』および『*Informatica Data Transformation 入門ガイド*』を参照してください。

## XML スキーマの一部のインポート

XML スキーマをインポートすると、Designer はスキーマ全体を示す XML 定義を作成します。関連スキーマと同時に XML ファイルをインポートすると、スキーマの全コンポーネントに対してではなく、XML ファイルの要素と属性のみの XML ビューを作成することができます。Designer は、XML ファイルのコンポーネントに限定した定義を作成します。

スキーマの一部をインポートするには、スキーマを参照する XML ファイルをインポートします。XML ファイルの要素および属性のみの XML ビューを作成するオプションを選択します。

### XML スキーマの一部のインポートに関するルールとガイドライン

スキーマの一部をインポートする場合は、次のルールおよびガイドラインに従ってください。

- Designer は、XML ファイル内のメタデータのみに限定します。XML ファイルを変更してから再度 XML スキーマをインポートすると、XML 定義は変更されます。
- 要素または属性が XML ファイルの階層では 1 回のみ出現している一方で、スキーマ階層の複数の部分で出現している場合、Designer はすべての要素または属性を XML 定義に取り込みます。  
たとえば、スキーマに Store/Address および Employee/Address の 2 つの address 要素が含まれるとします。そのスキーマと共に XML ファイルをインポートした際に、XML ファイルには Store/Address のみが含まれていた場合、Designer は Store/Address 要素および Employee/Address 要素を含めたすべての Address 要素を作成します。
- XML ファイルに派生複合型が含まれている場合、Designer はすべての基本型を個別のビューとして XML 定義に含めます。Designer は派生型を展開して、同じビューに基本型を含めることはしません。
- Designer は、XML ファイルに複数レベルの循環参照がある場合、XML 定義で循環参照を展開しません。

## エンティティリレーションの生成

XML 階層をエンティティ関係モデルとして生成することができます。XML ビューのエンティティ関係として生成するときに、Designer は以下の作業を実行します。

- 複数出現要素、参照要素、および複合型のビューの生成。

- ビュー間の関係の作成。

Designer はエンティティリレーションを生成する場合、スキーマ内のリレーションに基づいて、複合型、グローバル要素、および複数出現要素に対して異なるエンティティを生成します。

デフォルトグループとは異なるグループを作成する場合、あるいは異なる複合型の要素を結合する場合は、カスタム XML ビューを作成できます。

XML エディタで XML ソース定義を表示すると、XML 階層内での各要素間の関係を見ることができます。ビュー間の関係ごとに、XML エディタはビュー間の関係の型に基づいてリンクを生成します。

エンティティ関係を生成するには：

1. Source Analyzer で、[ソース] - [XML 定義のインポート] をクリックします。  
[XML] ウィザードが開きます。
2. インポートするソースを見つけ、[開く] をクリックします。
3. XML 定義につける名前を入力して、[次へ] をクリックします。
4. [エンティティリレーションシップ] を選択して、[完了] をクリックします。  
XML ウィザードは、エンティティ関係を使用する XML 定義を生成します。

## 階層リレーションの生成

階層リレーションを作成すると、コンポーネントへの各参照は親要素の下で展開されます。XML ウィザードは、デフォルトルートを選択し、デフォルト設定を使用して XML グループを作成します。

階層リレーションを生成するには：

1. Source Analyzer で、[ソース] - [XML 定義のインポート] をクリックします。  
[XML] ウィザードが開きます。
2. インポートするソースを見つけ、[開く] をクリックします。
3. XML 定義につける名前を入力して、[次へ] をクリックします。
4. [階層リレーション] を選択します。
5. [正規化 XML View] または [非正規化 XML View] を選択して、[完了] をクリックします。  
XML ウィザードは、階層リレーションに基づいた XML ビューを生成します。

## カスタム XML ビューの作成

カスタム XML グループは、XML ウィザードを使って作成することができます。カスタマイズした XML ビューを作成する場合は、ルートを選択し、メタデータを生成する方法を指定できます。処理したいデータにルート情報が当てはまるかどうかに応じて、グローバル要素を取り込むか除外するかを選択することができます。たとえば、商店と顧客に関する情報がスキーマに含まれている場合、ユーザは該当する顧客を処理する XML 定義を作成したくなることがあります。

ビューに関連付けられるメタデータを生成する方法も指定できます。エンティティ関係を生成することにより、要素、複合型、および継承複合型に関するメタデータの膨張を抑制できます。メタデータ参照を抑制しない場合、Designer は階層リレーションを生成し、親要素のすべての子要素を展開します。

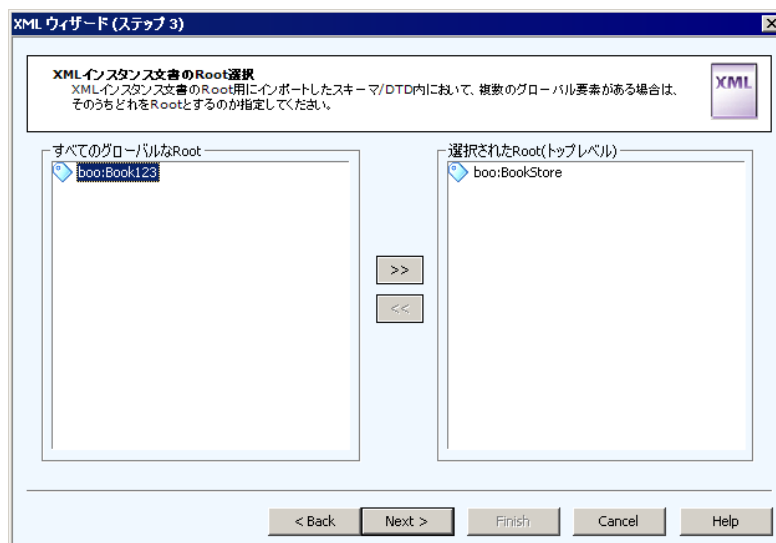
XML ウィザードでカスタムビューを作成するには：

1. Source Analyzer で、[ソース] - [XML 定義のインポート] をクリックします。  
[XML] ウィザードが開きます。
2. インポートするソースを見つけ、[開く] をクリックします。
3. XML 定義につける名前を入力して、[次へ] をクリックします。
4. [カスタマイズした XML View の作成] を選択して、[次へ] をクリックします。  
**注:** XML エディタで XML ビューのすべてを手動作成するには、[XML ビュー作成をスキップする] を選択します。XML ウィザードはリポジトリでスキーマを作成しますが、XML ビューは作成しません。
5. グローバルルート要素のリストからルート要素を選択して、[次へ] をクリックします。
6. 要素、複合型、および継承複合型に関するメタデータの抑制を選択して、[完了] をクリックします。

## ルート要素の選択

カスタムビューを作成するとき、インポートしたスキーマのグローバル要素から選択して、XML インスタンス文書のルートを設定することができます。グローバル要素は、XML スキーマ階層内で最上位のルート要素のすぐ下にある要素です。

以下の図に、[ルート選択] ページを示します。



この例では、Bookstore 要素がルートとして選択され、Book123 はルート要素としてクリアされます。

## メタデータ膨張の抑制

Designer は、継承を使用する XML スキーマに基づいて XML 定義を作成すると、ビュー内でそのメタデータを参照する参照要素または参照グループのメタデータを拡張させることができます。また、Designer は、参照されるオブジェクトについて別のビューを作成して、そのオブジェクトと他のビューとの間に関係を作成することができます。

XML スキーマ内で参照を使用した場合、参照に関連付けられるメタデータを Designer が取り込む回数を減らしたくなることがあります。XML ウィザードには、メタデータ参照を抑制する以下のオプションがあります。

- **要素の膨張の抑制。** Designer では、任意の複数出現要素に対しても、あるいは他のひとつの要素から参照された任意の要素に対してもビューを作成します。それぞれのビューには、定義内にあるその他のビューとの間で複数の階層関係を対応させることができます。
- **複合型の膨張の抑制。** Designer は、参照されている複合型または複数出現要素のそれぞれに対して XML ビューを作成します。XML ビューには、その他のビューとの間で複数の型関係を対応させることができます。スキーマが継承複合型を使用する場合には、継承複合型の膨張も抑制することができます。
- **複合型継承の膨張の抑制。** 継承型について、XML ウィザードは型関係を作成します。

メタデータ膨張を抑制すると、Designer は生成した XML ビュー間のエンティティリレーションを作成します。

## XML 定義の同期

XML 定義を使用する場合、XML 定義の作成時に使用したファイルまたはソースが変化することがあります。たとえば、XSD ファイルに新しい要素または複合型を追加することがあります。XML 定義の作成に使用した以下のリポジトリ定義またはファイルと XML 定義を同期させることができます。

- リレーショナルソース定義
- リレーショナルターゲット定義
- フラットファイル
- URL
- XML ファイル
- DTD
- スキーマファイル

XML 定義の同期を実行すると、スキーマナビゲータの XML スキーマは更新されます。しかし、XML 定義のビューは更新されません。ビューカラムは、XML 定義を同期した後で、XML エディタを使って手動で更新できます。

**ヒント:** スキーマファイルを使用して、XML 定義を同期させます。

XML ソース定義を同期させるには：

1. Source Analyzer で、[ソース] - [XML 定義のインポート] をクリックします。  
[XML] ウィザードが開きます。
2. XML 定義の作成に使用したリポジトリ定義またはファイルに移動して、[開く] をクリックします。
3. ウィザードのステップ 1 で [次へ] をクリックします。ウィザードでは、名前に対する変更は無視されます。
4. XML ウィザードのステップ 2 で、XML 定義の同期を選択して、[次へ] をクリックします。  
XML ウィザードはステップ 5 までスキップします。
5. XML ウィザードのステップ 5 で、同期させる XML 定義を選択します。  
XML ウィザードは、ソースと選択された定義を同期させます。

この方法を使って XML ターゲット定義を同期させることができます。XML ソース定義を変更した場合、ターゲット定義も同期させる必要があります。

**注:** XML 定義は、その定義の作成に使用したソースと必ず同期させてください。XML 定義を定義の作成に使用していないソースと同期させた場合、Designer は定義を同期させることができず、メタデータは失われます。[編集] - [保存状態に復帰] をクリックして、XML 定義を復元してください。

## XML ソース定義のプロパティの編集

XML ソース定義をインポートした後で、定義名などのソース定義のプロパティを編集することができます。ファイルリストを読み取るセッションを設定する場合、ソースファイル名を各ターゲット行に書き込むマッピングを設定することができます。メタデータ拡張を追加することもできます。

XML ソース定義プロパティを編集するには：

1. Source Analyzer ワークスペースで定義の上部を右クリックします。[編集] を選択します。
2. [テーブル] タブ上で、以下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。   |
| ビジネス名     | ソース定義の説明的な名前。[名前の変更] をクリックすると、ビジネス名を編集できます。  |
| オーナ名      | XML ファイルには適用されません。   |
| 説明        | ソースの説明。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 | ソースまたはデータベースタイプ。   |
| コードページ    | 読み取り専用。XML ソースファイルには適用されません。Integration Service によって、すべての XML ソースファイルが Unicode に変換されます。  |

3. [カラム] タブをクリックします。

[カラム] タブで、定義内のカラムに関する情報を表示できます。カラム名または値を変更するには XML エディタを使用します。

以下の情報を表示できます。

| カラムの設定項目 | 説明                |
|----------|-------------------|
| テーブルの選択  | 編集中のソース定義。        |
| カラム名     | カラムの名前。           |
| データ型     | PowerCenter データ型。 |

| カラムの設定項目 | 説明  |
|----------|---|
| 精度       | カラムの長さ。   |
| スケール     | 数値データの小数点以下の桁数。   |
| 非 Null   | カラムが NULL を受け入れることができるかどうかを示します。  |
| キータイプ    | プライマリキー、外部キー、またはキー以外。   |
| XPath    | XML 階層のカレントカラムによる参照先の要素へのパス。XPath は、生成されたプライマリキーまたは外部キーについては表示されません。    |
| ビジネス名    | カラムに対するユーザ定義の説明的な名前。ウィザードにビジネス名が表示されない場合は、右にスクロールして表示させるか、カラムを変更してください。 |

4. ファイルリストを読み取るセッションを設定し、ソースファイル名を各ターゲット行に書き込む場合、[プロパティ] タブをクリックして [Add Currently Processed Flat File Name Port (現在処理されているフラットファイル名ポートを追加)] を選択します。

Designer により、CurrentlyProcessedFileName ポートが [カラム] タブに追加されます。これは最初のグループの最後のカラムです。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 ポートは既定精度が 256 文字の文字列ポートです。

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] をクリアします。

5. [メタデータエクステンション] タブをクリックして、ユーザ定義のメタデータエクステンションの作成、編集、または削除を行います。
6. [OK] をクリックします。
7. [リポジトリ] - [保存] をクリックして、変更内容をリポジトリに保存します。

## リポジトリ定義からの XML 定義の作成

XML ソースまたはターゲット定義を、リポジトリのリレーショナルまたはフラットファイル定義からインポートすることができます。リポジトリ定義から XML 定義をインポートすると、XML ウィザードは選択されたオブジェクト間の関係から XML 階層を作成します。XML ウィザードは階層のルート要素を作成します。作成したグループからルートを選択できます。また、独立したルートを作成してグループを関連付けることもできます。

XML ターゲット定義を作成すると、XML ウィザードはキーを生成し、各グループをルートに関連付けます。

リポジトリのソースまたはターゲットから XML 定義を作成するには：

1. Source Analyzer で、[ソース] - [XML 定義のインポート] をクリックします。  
-または-  
Warehouse Designer で、[ターゲット] - [XML 定義のインポート] をクリックします。



2. [XML 定義のインポート] ダイアログボックスで、[Non-XML ソース] または [Non-XML ターゲット] をクリックします。

3. ソースまたはターゲットのリストから定義を選択します。矢印ボタンをクリックして、選択したソースリストに定義を追加します。

複数の入力および複数の入力タイプを選択できます。定義がプライマリキーおよび外部キーを介して関連付けられている場合、XML ウィザードは階層を生成する際にキーを使用してグループを関連付けます。

4. ルート要素に独立したグループを作成する場合、[追加 XML Root 名] を入力します。

ルート名は、デフォルトで XRoot に設定されます。ルートグループは、その他すべてのグループを囲みます。他のグループの 1 つをルートとして使用したい場合は、ルート名を削除します。XML ウィザードにより、選択した各入力ソースまたはターゲット定義のグループが作成され、各グループにプライマリキーが生成されます。XML ウィザードは、各グループに外部キーを作成します。外部キーはルートグループリンクキーを指します。

5. [開く] をクリックします。

XML ウィザードが表示されます。

6. XML ウィザードで、ソースまたはターゲットグループを生成します。

## XML ソースのトラブルシューティング

同じ親要素を持っている 2 つの複数出現要素を 1 つのビューに組み込むには、どうすればよいですか?たとえば、EMPLOYEE のすべての要素を 1 つのビューに組み込む必要がある場合は、次のように設定します。

```
<!ELEMENT EMPLOYEE (EID, EMAIL+, PHONE+)>
```

EMAIL および PHONE は同じ親要素に属していますが、同じ親チェーンには属していません。これらを同じ非正規化ビューに含めることはできません。EMPLOYEE の要素をすべて 1 つのビューに含めるには、複数出現要素の 1 つを Pivot 化する必要があります。

次の手順に従って、2 つの複数出現要素を同一ビューに追加してください。

1. EMPLOYEE ビューを作成します。
2. 要素 EID および EMAIL を EMPLOYEE ビューに追加します。
3. グループに取り込みたい EMAIL の出現を Pivot 化します。各 EMAIL の出現は、ビューの単独出現要素になります。
4. PHONE 要素を追加します。

DTD に以下の要素が含まれています。

```
<!ELEMENT EMPLOYEE (EMPNO, SALARY)+>
```

どのようにして EMPNO と SALARY を同じビューに入れることができますか？

DTD の例はあいまいです。これは、次の定義に相当します。

```
<!ELEMENT EMPLOYEE (EMPNO+, SALARY+)>
```

DTD の例では、EMPLOYEE には、複数出現要素である EMPNO および SALARY があります。同一ビューで複数回出現可能な要素を 2 つ持つことはできません。



以下の解決策のうちの一つを使用してください。

- **要素の定義をリライトして定義を明確にする。**

EMPLOYEE 要素を次のように定義することもできます。

```
<!ELEMENT EMPLOYEES (EMPLOYEE+)>
<!ELEMENT EMPLOYEE (EMPNO, SALARY)>
```

この構文を使用すると、各 EMPLOYEE に対して 1 つの EMPNO と 1 つの SALARY を定義することになります。EMPLOYEE ビューには、両方の要素が含まれます。EMPLOYEE を複数出現要素として EMPLOYEES に含める。

- **別のビューに要素を残して、マッピングでソース定義を 2 度使用する。**

EMPNO と SALARY が別々のビューに含まれていても、マッピングでデータを結合することができます。同じソース定義の 2 つのインスタンスを使用し、Joiner トランスフォーメーションを使用して結合します。

次の構造を持つ XML ファイルをインポートしました。

```
<Bookstore>
    <Book>Book Name</Book>
    <Book>Book Name</Book>
    <ISBN>051022906630</ISBN>
</Bookstore>
```

この XML ファイルをインポートすると、Designer が ISBN 要素を削除します。なぜこうなるのでしょうか?どうすれば、Designer は ISBN 要素を取り込むようになるのでしょうか?

- **スキーマを使用して XML 定義をインポートする。** XML ファイルを使って XML 定義をインポートすると、要素に子要素がないため、Designer は最初の要素を単純型の内容として読み込みます。Designer は 2 番目の Book インスタンスから ISBN 子要素を無視します。スキーマを使って定義をインポートすると、Designer はスキーマの定義を使って XML データの読み取り方法を決定します。
- **XML ファイルが関連スキーマを正確に表していることを確認する。** XML ファイルを使用してソース定義をインポートする場合は、その XML ファイルが対応する XML スキーマの構造の正確な表現であることを確認してください。

PowerCenter での XML サイズ設定の詳細については、『[「PowerCenter での XML の使用の概要」 \(ページ 29\)](#)』を参照してください。PowerCenter で XML 処理に適用される制限の詳細については、『[「制限」 \(ページ 29\)](#)』を参照してください。

他の要素型を持つトランスフォーメーションを作成し、大きな XML 入力ファイルを変換するには、データプロセッサトランスフォーメーションを使用します。データプロセッサトランスフォーメーションの作成方法の詳細については、『*Informatica Data Transformation ユーザーガイド*』および『*Informatica Data Transformation 入門ガイド*』を参照してください。

## XML エディタの使用

### XML エディタの使用の概要

XML 定義を Designer にインポートすると、XML 定義をデフォルトビュー、カスタムビュー、またはビューのない状態のいずれかで作成できます。XML 定義を作成した後は、XML エディタを使用して定義に変更を加えます。

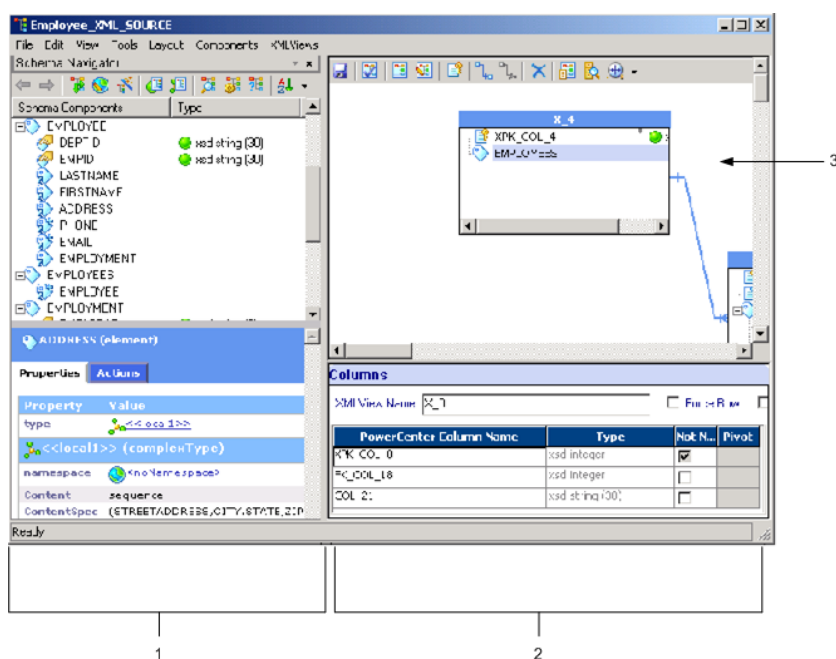
XML エディタを使用してビューを作成し、コンポーネントを変更し、カラムを追加し、ワークスペースのビュー関係を保持します。XML 定義を更新すると、Designer はそのソースを含んでいるマッピングすべてに変更を反映します。XML 定義への変更内容によっては、マッピングが無効になる場合があります。

**注:** XML 定義を作成するために使用したソースに大きな変更を加えた場合、手動で定義を編集するのではなく、XML 定義を新しいソースと同期させることができます。

XML エディタには以下のウィンドウがあります。

- ナビゲータ
- XML ワークスペース
- Columns ウィンドウ

次の図に XML エディタを示します。



1. ナビゲータ
2. Columns ウィンドウ
3. XML ワークスペース

XML エディタは、アイコンで XML のコンポーネントのタイプを表します。XML エディタのアイコンを説明する凡例を表示するには、[表示] - [凡例] の順にクリックします。

## XML ナビゲータ

ナビゲータには、階層形式でスキーマが表示され、選択したコンポーネントに関する情報が示されます。ナビゲータ内のコンポーネントを型別、階層別、または名前空間別にソートすることができます。コンポーネントを展開して、階層の下部にあるコンポーネントを参照することもできます。

ナビゲータツールバーには、ナビゲータのほとんどの機能へのショートカットが用意されています。このほかにも、クリックするだけで、速やかに前回表示されていた階層内のコンポーネントを検索できるナビゲーションの矢印があります。

ナビゲータには、次のタブがあります。

- **［プロパティ］ タブ**。選択したコンポーネントのコンポーネントタイプ、長さ、出現回数などの情報を表示します。
- **［アクション］ タブ**。選択したコンポーネントの詳細を表示するためのオプションを一覧表示します。

#### ［プロパティ］ タブ

［プロパティ］ タブには、ナビゲータで選択したコンポーネントに関する情報が表示されます。コンポーネントが複合型の要素である場合、名前空間、型、内容などのスキーマの要素のプロパティが表示されます。単純型の要素または属性を表示する場合、［プロパティ］ タブには、タイプおよび要素の長さが表示されます。［プロパティ］ タブには、注釈も表示されます。

この定義を XML ファイルからインポートした場合、［プロパティ］ タブからデータ型とカーディナリティを編集できます。DTD ファイルから定義を作成すると、コンポーネントのタイプを編集できます。

スキーマが名前空間を使用する場合、名前空間や接頭語、名前空間のスキーマの場所を変更できます。接頭語は、名前空間に属する要素または属性を表します。デフォルトの名前空間にある要素および属性には、接頭語はありません。1 つの名前空間を XML ターゲットのデフォルトの名前空間として選択できます。

#### ［アクション］ タブ

［アクション］ タブでは、選択したコンポーネントに関する詳細情報を参照するために使用できるオプションを一覧表示します。［アクション］ タブでコンポーネントに加えた変更を元に戻すこともできます。

選択したコンポーネントのプロパティに応じて、［アクション］ タブに以下のオプションが表示されます。

- **ComplexType 参照**。このタイプのスキーマコンポーネントを表示します。
- **ComplexType 階層**。選択したコンポーネントから派生した複合型を表示します。
- **SimpleType 参照**。このタイプのコンポーネントをすべて表示します。
- **SimpleType 値のプロパゲート**。SimpleType のコンポーネントすべてに対して、長さや位取りの値をプロパゲートします。
- **要素参照**。選択した要素を参照するコンポーネントを表示します。
- **子コンポーネント**。選択したコンポーネントが使用するグローバルスキーマコンポーネントを表示します。
- **simpleType の復元**。タイプ、長さ、および精度の値を変更した場合は、これらを元の値に変更します。
- **XML ビュー参照**。選択したコンポーネントを参照するすべての XML ビューおよび列を表示します。

## XML ワークスペース

XML ワークスペースには、XML ビューやビュー間の関係が表示されます。ワークスペースで XML ビューを作成したり、ビュー間の関係を定義したりすることができます。

XML のワークスペースツールバーには、ワークスペースで実行できるほとんどの機能へのショートカットが用意されています。

以下の方法で、XML ワークスペースのサイズを変更できます。

- **Columns ウィンドウの非表示**。［ビュー］ - ［カラムのプロパティ］ をクリックします。

- **ナビゲータの非表示。** [ビュー] - [ナビゲータ] をクリックします。
- **ワークスペースの縮小。** [ワークスペース] ツールバーの [ズーム] ボタンをクリックします。

## Columns ウィンドウ

Columns ウィンドウには、ワークスペースのビューのカラムが表示されます。Columns ウィンドウを使用すると、追加するカラムに名前を付けることができます。Pivot 化されたカラムを使う場合は、Columns ウィンドウを使用して、複数出現要素の出現を選択し、名前を変更します。また、[非 Null] オプション、[Force Row] オプション、[階層リレーション Row] オプション、[Type Relationship Row] オプション、[Non-Recursive Row] オプションなどのオプションを指定できます。これらのオプションは、Integration Service が XML ターゲットにデータを書き込む方法に影響を与えます。

## ビューの作成と編集

XML エディタを使用してカスタム XML ビューを作成するか、XML ウィザードで作成した XML ビューを編集します。ビューを作成するには、ビューを定義して、ビューのカラムを指定します。スキーマに複数出現要素がある場合は、ビューに含める要素の出現を指定することができます。また、XML ターゲットファイル名用に専用ポートを作成することも、XML Parser トランスフォーメーションおよび XML Generator トランスフォーメーション用にパススルーポートを作成することもできます。

次のタスクを実行して、XML ビューの作成および編集を実行します。

- **XML ビューの作成。** ビューをワークスペースに追加します。
- **ビューへの列の追加。** ビューに新しい列を作成します。
- **ビューからの列の削除。** ビューから列を削除します。
- **複合型の拡張。** ビューに追加する派生複合型を選択します。
- **anyType 要素のインポート。** anyType 要素をインポートします。
- **anyAttribute 要素へのコンテンツの適用。** anyAttribute 要素のコンテンツを定義します。
- **anySimpleType 要素の使用。** XML 定義で anySimpleType 要素を使用します。
- **パススルーポートの追加。** XML トランスフォーメーションを通して非 XML データを渡すポートを追加します。
- **FileName 列の追加。** 各 XML ターゲットファイルの新しいファイル名を生成する列を追加します。

## XML ビューの作成

XML ワークスペースでビューを作成できます。ビューのない XML 定義を作成した場合、XML エディタのワークスペースは空になっています。ビューを作成して、ビューにカラムや View Row を追加できます。

ワークスペースで新しい XML ビューを作成するには：

1. XML エディタで XML 定義を開きます。
2. [XMLViews] - [XML View の作成] をクリックします。  
XML エディタはワークスペースに空のビューを作成し、Columns ウィンドウに空のカラムを表示します。
3. Columns ウィンドウ内のビューの名前を入力します。

この名前は、ワークスペース内の XML ビューに表示されます。

4. ビューを作成する場所で、スキーマナビゲータのノードを強調表示します。
5. [コンポーネント] - [XPath ナビゲータ] をクリックします。

ナビゲータウィンドウに、[XPath ナビゲータ] が表示されます。

6. XPath ナビゲータのカラムモードを View Row モードに設定すると、ビュー行が追加されます。
7. ナビゲータの要素を選択し、この要素をワークスペースのビューにドラッグします。

XML エディタはビュー行を青色で強調表示します。

初めてカラムをビューに追加した場合、Designer ではカラムがビュー行になるかどうかを検証します。たとえユーザがビュー行の追加を指定しなくても、この検証は行われます。

8. ビュー行を別のカラムに変更するには、ビュー内で該当する行を右クリックして、[View Row の設定] を選択します。

## ビューへの列の追加

XML ワークスペース内部の XML ビューにカラムを追加することができます。カラムを追加するには、XPath ナビゲータからカラムを選択します。

以下の条件が当てはまる場合は、XML ビューにカラムを追加することができます。

- コンポーネントのパスは、該当ビューのビュールートであるスキーマの中にある要素から始まります。
- コンポーネントが囲み要素ではありません。
- コンポーネントが正規化ルールにも Pivot 化ルールにも違反していません。たとえば、2 つ以上の複数出現要素をビューに追加することができません。
- 内容が混在する要素は、単純型または複合型のいずれかで追加することができます。
- 2 つのビューは同じカラムを共有でき、ビューには複数の同一のカラムが含まれます。

ビューにカラムを追加するには：

1. ワークスペースで XML ビューを選択します。
2. 追加したいコンポーネントを含んでいるナビゲータの親要素を強調表示します。
3. [コンポーネント] - [XPath ナビゲータ] をクリックします。

ナビゲータウィンドウに、[XPath ナビゲータ] が表示されます。

4. [カラムモード] をクリックし、カラムまたはビュー行を追加します。
5. [詳細設定] を選択し、ピボット化されたカラムをビューに追加します。

ピボット化されたカラムは、複数出現要素の出現を示します。単独出現列は詳細モードに追加できません。

**注:** XML ターゲット定義で Pivot 化カラムを作成することはできません。

6. カラムを [XPath ナビゲータ] から XML ワークスペースの適切なビューにドラッグします。1 度に複数のカラムが選択できます。

XML エディタは、追加するカラムを検証します。カラムがビューに対して無効である場合、カラムのドラッグ中にステータスバーにメッセージが表示されます。新しいカラムをビューに追加すると、その新しいカラムが表示されるようになります。

## ピボット化列の追加

XML 定義内のピボット化されたカラムとは、ビュー内で要素の出現に対して個別のカラムを作成する複数出現要素のことです。ピボット化されたカラムは、XML 定義内のどの複数出現要素からでも作成できます。ビュー行の要素をピボット化することもできます。

ピボット化されたカラムをビューに追加すると、Columns ウィンドウにデフォルトの出現番号が表示されます。この番号は、カラム内で使用する要素の出現回数を表示します。出現番号を変更することも、新しいカラムとしての要素の出現をさらに追加することもできます。カラムの名前を変更しない場合、XML エディタはピボット化されたカラムの名前に対してそれぞれ連続した番号を追加します。

**注:** ピボット化値がビュー行の一部である場合は、このピボット化値を変更できません。

ピボット化されたカラムをビューに追加するには、

1. ワークスペースで XML ビューを選択します。
2. ナビゲータ内でピボット化したい要素を強調表示します。  
または、ピボット化する要素を含む親チェーンの要素を強調表示します。
3. [コンポーネント] - [XPath ナビゲータ] をクリックします。
4. [詳細モード] を選択します。
5. [XPath ナビゲータ] からピボット化したいカラムを XML ワークスペースのビューにドラッグします。

Designer は、Columns ウィンドウにデフォルトの出現番号を追加します。この番号は、カラム内で使用する要素の出現回数を表示します。

次の図のビューには、3 回目に出現する Product に対して 2 回目に出現する Sales が含まれます。

Columns

XMLビュー名EMPLOYEEXMLビュー オプション

列のXPathを表示EMPLOYEE

| PowerCenter の列名 | データ型         | Null                                | XPath                 |
|-----------------|--------------|-------------------------------------|-----------------------|
| DEPTID          | integer (4)  | <input type="checkbox"/>            | /DEPTID               |
| EMPID           | integer      | <input type="checkbox"/>            | /EMPID                |
| LASTNAME        | integer (12) | <input type="checkbox"/>            | /LASTNAME             |
| FIRSTNAME       | integer (8)  | <input type="checkbox"/>            | /FIRSTNAME            |
| BASISALARY      | integer      | <input checked="" type="checkbox"/> | /EMPLOYMENT/SALARY(1) |
| BONUS           | integer      | <input type="checkbox"/>            | /EMPLOYMENT/SALARY(2) |

カラム内に初めて Sales が出現することを、First\_Half\_Sales と言います。2 回目に Sales が出現することを、Second\_Half\_Sales と言います。Region は属性です。

6. [XPath] リンクをクリックして、要素の出現番号を変更します。  
[XPath のクエリー述部を指定してください] ウィンドウが表示されます。
7. 編集する複数出現要素を選択します。
8. [ピボットの編集] ボックスで出現番号を変更して、[OK] をクリックします。

## ビューからの列の削除

ビューから列を削除できます。

ビューからカラムを削除するには：

1. ワークスペースのビューのカラムを右クリックします。
2. [このカラムを削除] を選択します。

XML エディタは、カラムを削除してよいかを確認するメッセージを表示します。



3. よい場合は、[はい] をクリックします。

XML エディタは、ビューからカラムを削除します。ただし、カラムは XPath ナビゲータの階層内に残ります。

### ピボット化列の削除

ピボット化されたカラムの出現を削除するには、Columns ウィンドウからカラムを選択して、削除します。

ピボット化されたカラムを削除するには：

1. Columns ウィンドウで削除するカラムを右クリックします。
2. [削除] - [Pivot] の順にクリックします。
3. 削除してよい場合は、[はい] をクリックします。

## 複合型の拡張

スキーマには、複数の型に対応するベース型である複合型が含まれていることがあります。たとえば、Publication が Magazine になることや Newspaper になることがあります。XML ビューを作成する場合、Publication の使用方法（Magazine 型または Newspaper 型を指定）を選択できます。

XPath ナビゲータで複合型を表示すると、その派生型を表示できます。

複合型を展開するには：

1. XPath ナビゲータ内の複合型を強調表示します。  
派生型は、[複合型の展開] リストに表示されます。
2. 使用したい型を選択します。  
コンポーネントを XML 定義に追加すると、定義には選択した型が含まれます。

## anyType 要素のインポート

anyType 要素を含む XML スキーマをインポートできます。anyType タイプの要素には、XML 文書に出現するデータ型が含まれている可能性があります。たとえば、XML スキーマの以下のセクションには、anyType である要素 Document が含まれます。

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Publication" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="Document" type="xsd:anyType"/>
</xsd:schema>
```

anyType 要素を持つスキーマをインポートした場合、anyType 要素はスキーマナビゲータでは anyType として表示されます。Designer は anyType タイプの要素に対してポートを作成しません。

PowerCenter で anyType 要素を使用するには、Designer で anyType 要素をグローバル複合型に変更する必要があります。

anyType 要素を別のグローバル複合型に変更する手順：

1. スキーマナビゲータで anyType 要素を強調表示します。  
[anyType] プロパティがスキーマナビゲータの [プロパティ] タブに表示されます。



2. [anyType] プロパティをクリックします。

スキーマにグローバル複合型が含まれていない場合、Designer によって選択するグローバル複合型がないことを通知するエラーが表示されます。

3. 複合型を選択して、[OK] をクリックします。

## anyAttribute 要素または ANY 要素へのコンテンツの適用

anyAttribute 内容の要素または ANY 内容の要素を含む XML スキーマをインポートできます。ビュー内の要素を使用するには、XML エディタで要素の内容を定義する必要があります。anyAttribute 内容の要素または ANY 内容の要素を含むスキーマをインポートした場合、要素はスキーマナビゲータにプロパティなしで表示されます。要素はビューに表示されません。

たとえば、次のスキーマ要素には ANY 内容の要素が含まれています。

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

anyAttribute 内容または ANY 内容に内容を適用するには：

1. スキーマナビゲータの要素内容のリンクをクリックします。  
[すべてまたはすべての属性タイプコンテンツの編集] ダイアログボックスが表示されます。
2. [タイプの追加] ボタンをクリックします。  
新規の行が表示されます。
3. XML 定義に対して、スキーマ内の有効要素のリストから要素を選択します。
4. カーディナリティを選択して、[OK] をクリックします。  
置き換えタイプがスキーマナビゲータに表示されます。

## XML エディタにおける anySimpleType の使用

anySimpleType 要素には、文字列、整数、小数、日付などの Atomic データ型を格納することができます。XML インスタンス文書にある要素のデータ型が分からない場合は、スキーマの anySimpleType を使用します。

要素を「anySimpleType」として定義すると、Designer はスキーマをインポートする際に、要素に対して anySimpleType カラムを作成します。マッピングでカラムを使用するとき、XML Source Qualifier はこのタイプを文字列にマッピングします。

## パススルーポートの追加

XML Parser トランスフォーメーションまたは XML Generator トランスフォーメーションにパススルーポートを追加して、非 XML データをトランスフォーメーション経由で渡すことができます。トランスフォーメーション内のポートを定義する場合、ポートを XML Parser トランスフォーメーションの DataInput グループ、または XML Generator トランスフォーメーションの DataOutput グループに追加します。

パススルーポートを生成すると、別のポートを追加してトランスフォーメーションからデータを渡すことができるようになります。このポートを、参照ポートと言います。XML Parser トランスフォーメーションでは、パススルーポートはトランスフォーメーションにデータを渡し、参照ポートはそのトランスフォーメーションからデータを渡します。XML Generator トランスフォーメーションでは、パススルーポートはトランスフォーメーションからデータを渡し、参照ポートはトランスフォーメーションにデータを渡します。

XML 定義内にパススルーポートがある場合、対応する参照ポートを決定することができます。

パススルーポートの参照ポートを決定するには、

1. パススルーポートを右クリックします。
2. [フォーカスの移動] - [参照されているカラム] を選択します。  
XML エディタは、ワークスペース内の参照カラムを強調表示します。

## FileName 列の追加

セッション実行中に、Integration Service はデータ内で新しいルート値が発生するたびに新しいターゲット XML ファイルを出力します。XML ビューに [FileName] カラムをすると、各 XML ファイルの一意的なファイル名を生成します。このファイル名は、セッションプロパティ内のデフォルトの出力ファイル名を上書きします。

[FileName] カラムを使用する場合、マッピングに Expression トランスフォーメーションまたは他のトランスフォーメーションを設定して、[FileName] カラムに渡すファイル名を生成します。

XML ビューで [Filename] カラムを作成するには：

1. XML エディタ内のビューを右クリックします。
2. [Filename カラムの作成] を選択します。  
XML エディタは、ビュー内に新しい文字列カラムを作成します。
3. Columns ウィンドウでカラム名を変更します。
4. XML エディタを終了します。  
XML 定義にカラムが表示されます。XPath は、\$Filename です。

**注:** XML ファイル名ポートを使用する場合、ファイルのディレクトリ名を指定する必要があります。

## XPath クエリの述部の作成

XML 階層のビュー行の下にあるビュー行または列に XPath クエリの述部を作成すると、セッション内の XML データをフィルタできます。XPath クエリーの述部を作成すると、XML エディタはビュー行の XPath に XPath クエリーの述部を追加します。

XML エディタで XPath クエリーの述部を作成した場合は、XML エディタがクエリーを作成するための要素や属性、演算子、関数などを用意します。クエリーに対してコンポーネントを選択したり、クエリーにコンポーネントを入力したり、コンポーネントをクエリーにコピーしたりすることができます。XML エディタは、作成した各クエリーを検証します。

要素や属性の値に対してクエリーを実行することも、要素または属性が存在することを確認することもできます。

## 属性要素の値に対するクエリの実行

XPath クエリーの述部を作成して、ビューで要素または属性の値をフィルタすることができます。たとえば、Department 100 から従業員を抽出する場合、次の XPath クエリーの述部を作成します。

```
EMPLOYEE [./DEPT = '100']
```

クエリー式はかっこで囲みます。Dept の XPath は“./”の記号で省略され、パスが Employee から続いていることが示されます。

次の XPath クエリーの述部は、姓が Smith の従業員を抽出します。

```
EMPLOYEE[./NAME/LASTNAME='SMITH']
```

Name は Employee の子要素で、Lastname の親になります。

XPath クエリーの述部では、論理演算子または数値演算子を使用します。また、クエリーでは文字列関数、数値関数、論理関数を使用できます。

### 混在コンテンツのクエリ

XML 要素に値と子要素の両方が含まれる場合、その要素は混在コンテンツを備えています。XPath クエリーの述部を作成すると、子要素で分けられた要素の値をフィルタできます。ただし、Integration Service では、混在コンテンツの最初の子要素の後に出現する述部は評価されません。

たとえば、XML ファイルに混在コンテンツを含む NAME 要素が含まれているとします。

```
<NAME>
  Kathy
  <MIDDLE> Mary </MIDDLE>
  Russell
</NAME>
```

要素 NAME には、値「Kathy」、子要素「MIDDLE」、および 2 つ目の値「Russell」が含まれています。NAME カラムの値は「KathyRussell」です。ただし、Integration Service では、NAME「Kathy」が評価されます。

以下のクエリーは False になります。

```
EMPLOYEE [./NAME = 'KathyRussell']
```

以下のクエリは True になります。

```
EMPLOYEE [./NAME = 'Kathy']
```

### 論理演算子

次の論理演算子を XPath クエリーの述部で使用します。

and or < <= > >= = !=

次の XPath クエリーの述部を使用すると、姓が Jones の従業員を Department 100 から抽出できます。

```
EMPLOYEE [./DEPT = '100' and ./ENAME/LASTNAME = 'JONES']
```

### 数値演算子

次の数値演算子を XPath クエリーの述部で使用します。

+ - \* div mod

次の XPath クエリーの述部を使用して、原価と税金を足した合計よりも価格が高い製品を抽出します。

```
PRODUCT[./PRICE > ./COST + /.TAX]]
```

## 関数

XPath クエリの述部には、以下のタイプの関数を使用します。

- **文字列。** サブストリングの値のテスト、文字列の連結、または、文字列の他の文字列への変換には、文字列関数を使用します。次の XPath クエリの述部は、従業員のフルネームが姓と名前（ファーストネーム）の結合と等しいかどうかを確認します。

```
EMPLOYEE[./FULLNAME=concat(./ENAME/LASTNAME,./ENAME/FIRSTNAME)]
```

- **数値。** 要素および属性の値には数値関数を使用します。数値関数は、数値の演算を行い、整数を返します。たとえば、次の XPath クエリーの述部は、割引額を丸め、結果が 15 より大きいかどうかを確認します。

```
ORDER_ITEMS[round(./DISCOUNT > 15)]
```

- **論理値。** 論理関数は、True または False のいずれかを返します。論理関数を使用すると、要素をテストすること、language 属性を確認すること、または True または False の結果を強制的に返すことができます。たとえば、文字列値が 0 より大きい場合、文字列は True になります。

```
boolean(string)
```

## 要素または属性のテスト

要素または属性が XML データで出現するかどうかを確認できます。次の XPath クエリーの述部は、Department に部署名の属性があるかどうかを確認します。

```
COMPANY/DEPT[@DEPTNAME]/EMPLOYEE
```

Deptname は属性です。XML クエリーの述部の式では、属性の前に“@”が付きます。

セッションを実行すると、Integration Service は、従業員の部署に部署名がある場合は、XML ソースから従業員データを抽出します。それ以外の場合、Integration Service は従業員データを抽出しません。

## XPath クエリーの述部のルールおよびガイドライン

XPath クエリーの述部を作成する場合、次のルールおよびガイドラインを使用してください。

- ビュー行のすべての要素に対して、XPath クエリーの述部を設定できます。

たとえば、行が Company/Dept の場合、次の XPath クエリーの述部を作成できます。

```
COMPANY[./DEPT=100]
```

- コンテンツを照合できます。

- XML 階層のビューにあるビュー行よりも下にカラムが出現していて、XPath のカラムにビュー行が含まれている場合は、カラムに XPath クエリーの述部を追加できます。

たとえば、ビュー行が Product/Toys[1] の場合、次の XPath クエリーの述部を作成できます。

```
Product/Toys[1][./Sales > 100]
```

次の例では、Product/Toys[1] ビュー行に対して無効な XPath クエリーの述部を示します。

```
Product/Toys[2][./Sales > 100]
```

Product/Toys[1] は、ビュー行です。Product/Toys[2] は使用できません。

- 単独出現要素または単独出現属性を使用します。複数出現要素では、XPath クエリーの述部を使用できません。
- 囲み要素には値が含まれていないため、XPath クエリーの述部を囲み要素内に作成することはできません。

## XPath クエリーの述部の作成手順

XML ソース定義内のビューに、XPath クエリーの述部を作成することができます。

XPath クエリーの述部の作成手順

1. XML エディタで XML ソース定義を開きます。
2. XML エディタのワークスペースでビューを選択します。  
Columns ウィンドウにビューの列名が表示されます。
3. [列の XPath を表示] をクリックします。  
[XPath のクエリー述部を指定してください] ウィンドウが表示されます。[XPath 述部] ウィンドウに XPath クエリーの述部を入力するか、ダイアログボックスのタブから要素、演算子、関数を選択できます。
4. [子要素/属性] タブをクリックして、XPath クエリーの述部に追加できる要素および属性を表示させます。
5. 要素または属性をダブルクリックして、XPath クエリーの述部に追加します。  
パネルにコンポーネントが表示されます。
6. [演算子] タブをクリックします。  
演算子を使用して式を作成します。値またはその他の要素に対して要素を比較したり、算術式を作成したりすることができます。

次の表は、XPath クエリーの述部に追加できる演算子を示します。

| 演算子 | 説明     |
|-----|--------|
| +   | 追加     |
| -   | 減算     |
| *   | 乗算     |
| div | 除算     |
| mod | 係数     |
| and | 論理 AND |
| or  | 論理 OR  |

| 演算子 | 説明    |
|-----|-------|
| <   | より小さい |
| <=  | 以下    |
| >   | より大きい |
| >=  | 以上    |
| =   | 等しい   |
| !=  | 等しくない |

7. 演算子をダブルクリックして、XPath クエリーの述部に演算子を追加します。
8. PowerCenter の XPath クエリーの述部の関数を追加するには、[関数] タブをクリックします。関数は引数を取り、値を返します。
9. [子要素/属性] タブで別の要素または属性を選択するか、[XPath 述部] ウィンドウに値を入力して式を完成させます。
10. [検証] をクリックして XPath クエリの述部を検証します。

## ビューリレーションの維持

次のタスクを実行して、XML エディタのビューリレーションを管理できます。

- **ビュー間のリレーションの作成。** ワークスペースでビュー間のリレーションを定義します。
- **型関係の作成。** ビューの列とワークスペースのタイプビューとの間の型関係を定義します。
- **エンティティリレーションの再作成。** XML ウィザードと同じオプションを使用して、ビューおよびリレーションを生成します。

## ビュー間のリレーションの作成

XML エディタを使用してビュー間の階層関係または継承関係を作成できます。XML ソースと非 XML ソースの間に関係を作成することはできません。

XML ビュー間のリレーションを作成するには：

1. ワークスペースで子の XML ビューの上部を右クリックします。
2. [リレーションの作成] を選択します。
3. ポインタを親のビューに移動させて、リレーションを確立します。  
ポインタを移動させると、ビュー間にリンクが表示され、XML エディタは、リレーションが有効であることを確認します。関係が無効である場合、ステータスバーにエラーメッセージが表示されます。
4. エラーメッセージが表示されない場合は、親のビューをクリックして関係を確立します。  
XML エディタは関係を作成し、適切な外部キーを追加します。

5. リレーションに関する詳細を表示するには、ビュー間のリンク上にカーソルを置きます。  
XML エディタには、リレーションの型、およびプライマリキーと外部キーが表示されます。

## 型関係の作成

ビューのカラムと型ビューとの間に型関係を作成することができます。カラムをピボット化すると、ユーザは関係に含まれる出現を選択できます。

型関係を作成するには:

1. ユーザが使用したいビューでカラムを右クリックします。
2. [リレーションの作成] を選択します。
3. カラムをピボット化した場合、ユーザは使用する出現を選択します。
4. ポインタを型ビューに移動して、リレーションを確立します。

ポインタを移動させると、ビュー間にリンクが表示され、XML エディタは、リレーションが有効であることを確認します。関係が無効である場合、ステータスバーにエラーメッセージが表示されます。

5. エラーメッセージが表示されない場合は、親のビューをクリックして関係を確立します。

XML エディタが次の関係を作成します。

## エンティティリレーションの再作成

XML 定義のエンティティ関係を再作成することができます。XML ウィザードと同じオプションを使用して、[エンティティリレーションの再作成] ダイアログボックスで新しい XML ビューを生成します。ビューを再作成する場合、既存ビューを保持することを選択できます。XML 定義にはすべてのビューが含まれます。

XML 定義のエンティティ関係を再作成するには:

1. XML エディタで XML 定義を開きます。
2. ナビゲータで XML のルートを強調表示します。  
別のコンポーネントを強調表示すると、XML エディタはそのコンポーネントをルートとして使用します。
3. [XML View] - [エンティティリレーションの作成] をクリックします。
4. 以下のオプションから選択して、メタデータの膨張を抑制します。
  - **要素の膨張の抑制。** 複数出現要素または参照要素について、XML ウィザードは複数の階層リレーションを持つ XML ビューを 1 つ作成します。
  - **複合型の膨張の抑制。** 複数出現要素または参照複合型について、XML ウィザードは複数の型関係を持つ XML ビューを 1 つ作成します。スキーマが継承複合型を使用する場合には、継承複合型の膨張も抑制することができます。
  - **複合型継承の膨張の抑制。** 継承型について、XML ウィザードは複数の型関係を使用して 1 つの XML ビューを作成します。
  - **既存の XML ビューの共有。** 既存の XML ビューを削除しないでください。
  - **共有 XML ビューのリフレッシュ。** 既存のビューを保存しますが、これらを更新します。
5. [次へ] をクリックします。  
[エンティティリレーションの再作成] ダイアログボックスが表示されます。



6. 子のコンポーネントを表示するには、共有された要素または複合型を選択して、名前をクリックします。
7. 子のコンポーネントを除外するには、[子コンポーネントを含まない] ペインで要素をクリアします。  
新しいビューを作成するには、要素または複雑型を選択します。新しいエンティティ関係を作成する場合、この要素をビュールートとしてビューを生成します。

## スキーマコンポーネントの表示

ナビゲータおよびワークスペースにコンポーネントを表示する場合は以下の手順を実行します。

- **名前空間の更新。** XML ターゲットでスキーマの場所またはデフォルトの名前空間を変更します。
- **コンポーネントへの移動。** 1 つのコンポーネントから別のコンポーネントへ、または XML エディタウインドウの領域へ移動してコンポーネントを検索します。
- **ワークスペースのビューの整列。** ワークスペースのビューを階層順に並べ替えます。ワークスペースでビューを階層の順序に従って並べ替えることができます。ワークスペースでビューを並べ替えるには、[レイアウト] - [整列] をクリックするか、ワークスペースを右クリックして [整列] を選択します。
- **コンポーネントの検索。** ナビゲータ内またはワークスペース内でコンポーネントを検索します。
- **単一型または複合型階層の表示。** XML スキーマに単一型または複合型の階層を表示します。
- **XML メタデータの表示。** XML エディタが XML 定義から作成する XML ファイル、スキーマ、または DTD を表示します。
- **XML データのプレビュー。** 外部 XML ファイルからのサンプルデータを使用して XML ビューを表示します。
- **XML 定義の検証。** XML 定義とビューのエラーを検証します。

## 名前空間の更新

XML 定義を作成すると、XML エディタで名前空間のプレフィックスとスキーマの URL を変更できます。また、名前空間にスキーマを追加することもできます。

1 つ以上の名前空間を持つターゲット XML 定義を作成した場合、デフォルトの名前空間を選択できます。セッションを実行すると、Integration Service は名前空間のプレフィックスを持たないデフォルトの名前空間から要素および属性を書き込みます。

名前空間のプレフィックスに、「xml」または「xmlns」を使用しないでください。「xml」のプレフィックスは、デフォルトでは <http://www.w3.org/XML> の名前空間を指します。「xmlns」は、XML スキーマの名前空間に要素をリンクします。

**注:** XML エディタを使用して名前空間を追加することはできません。

名前空間を更新するには、

1. ナビゲータで要素を選択します。
2. [プロパティ] タブをクリックします。
3. [名前空間] リンクをクリックします。

[名前空間のプレフィックスとスキーマ URL の編集] ダイアログボックスが表示されます。

4. プレフィックスまたはスキーマの URL を変更するには、変更したいテキストを選択して新しい値を入力します。
5. 複数のスキーマを名前空間に追加する場合は、スキーマの URL を選択して [追加] をクリックします。

名前空間の [スキーマの URL] にブランクのスキーマが表示されます。

6. スキーマを削除するには、[スキーマの URL] を強調表示して [削除] をクリックします。
7. XML ターゲットにデフォルトの名前空間を作成する場合、名前空間を選択します。  
名前空間で選択したすべてのコンポーネントは、XML ターゲットのデフォルトの名前空間に帰属します。セッションを実行すると、Integration Service は XML ターゲットファイルにデフォルトの名前空間のプレフィックスを書き込みません。

## コンポーネントへの移動

コンポーネントをすばやく検索するには、XML 定義でワークスペースコンポーネントを選択し、ナビゲーションオプションを選択します。たとえば、ビュー内で外部キーをクリックすると、関連付けられたプライマリキー、または Columns ウィンドウのカラムに移動できます。ワークスペースのコンポーネント間、Columns ウィンドウのコンポーネント間、およびナビゲータのコンポーネント間を移動することができます。

コンポーネントに移動するには：

1. ワークスペースのコンポーネントまたは Columns ウィンドウのコンポーネントを右クリックします。
2. [フォーカスの移動] を選択します。
3. 利用可能なオプションを選択します。

移動元として選択するコンポーネントに応じて、以下のオプションから選択することができます。

- **スキーマコンポーネント**。ナビゲータ内でコンポーネントを強調表示します。
- **PowerCenter 列**。Columns ウィンドウ内の列を強調表示します。
- **プライマリキー**。選択された外部キーに関連付けられたプライマリキーを強調表示します。
- **参照列**。XML Parser トランスフォーマー、または XML Generator トランスフォーマーのパススルーポートに関連付けられた参照列を強調表示します。
- **XPath ナビゲータ**。選択されたコンポーネントのパスを表示します。
- **XML ビュー**。Columns ウィンドウから選択された列を含むワークスペースのビューを強調表示します。

## コンポーネントの検索

XML 定義内でコンポーネントを検索できます。XML エディタは、コンポーネントのすべての出現を表示します。検索結果をクリックすると、XML エディタはスキーマまたはビューのコンポーネントを強調表示します。ワークスペースで XML スキーマまたは XML ビューを検索できます。

### XML スキーマの検索

コンポーネントを名前別、型別、名前空間別に検索できます。検索対象のプロパティを指定できます。たとえば、注釈、固定値（またはデフォルト値）、または長さなどのプロパティです。列挙プロパティに従って有効値を検索できます。

スキーマのコンポーネントを検索するには：

1. [編集] - [スキーマ内を検索] をクリックします。  
[XML スキーマ内のコンポーネント検索] ダイアログボックスが表示されます。
2. コンポーネントプロパティに従って検索する場合、[追加オプション] リンクをクリックし、検索可能なプロパティを表示します。
3. 検索対象の名前、型、またはプロパティを入力します。
4. 特定の名前空間内を検索したい場合は、[All] をクリックして、リストから名前空間を選択します。
5. [検索] をクリックします。  
ダイアログボックスに検索結果が表示されます。
6. 検索結果をクリックすると、検索結果がプロパティウィンドウに表示されます。

### XML ビューの検索

XML ビュー内のコンポーネントおよびカラムを検索できます。コンポーネント名で検索する場合、定義内のすべての関連するカラムを検索することができます。たとえば、コンポーネント「Number」を検索すると、結果にはコンポーネント「Number」を含むビューおよびカラムが含まれます。

断片的なキーを使用して検索する場合、カラム名またはコンポーネント名の最初の数文字を入力します。

XML ビューのコンポーネントを検索するには：

1. [編集] - [XML View の検索] をクリックします。  
[ワークスペース内での XML View とカラムの検索] ダイアログボックスが表示されます。
2. 検索条件を入力します。  
カラム型としてすべてのカラム型(All)、標準のカラム型(Regular)、生成されたキー(Generated Key)、またはその他の型(Other)を検索できます。その他の型には、FileName カラム、参照ポート、およびパススルーフィールドが含まれます。
3. [検索] をクリックします。  
ダイアログボックスに検索結果が表示されます。
4. 検索フィールドをクリアするには、[新規検索] をクリックします。

### 単純型階層または複合型階層の表示

XML 単純型の階層をスキーマ定義に表示することができます。単純型の階層を表示するには、[表示] - [単純型階層] をクリックします。ウィンドウに単純型の階層が表示されます。

スキーマ定義にある複合型の階層を表示することができます。複合型の階層を表示するには、[表示] - [複合型階層] をクリックします。ウィンドウにスキーマの複合型の階層が表示されます。[複合型階層] ウィンドウからコンポーネントを選択して、スキーマのコンポーネントに移動します。

### XML メタデータの表示

XML 定義を、XML、DTD、または XML スキーマファイルとして表示することができます。

XML メタデータを表示するには：

1. メタデータをサンプル XML 文書として表示したい場合は、ナビゲータのグローバルコンポーネントを選択します。

2. [表示] - [XML メタデータ] をクリックします。  
[XML メタデータの表示] ダイアログボックスが表示されます。
3. 選択することにより、XML 定義を XML ファイル、DTD ファイル、または XML スキーマとして表示できます。  
複数の名前空間を使用している場合、使用する名前空間を選択します。  
デフォルトのアプリケーションまたはテキストエディタにメタデータが表示されます。
4. XML ファイル、DTD ファイル、または XML スキーマファイルのコピーを保存するには、[別名で保存] をクリックします。
5. 新しいファイル名を入力します。  
デフォルトの表示用アプリケーションがテキストエディタの場合、適切なファイルの接尾語をファイル名に含める必要があります。接尾語は、作業するファイルに応じて、.xml、.dtd、または.xsd とします。

## XML 定義の検証

XML エディタで XML 定義を検証することができます。

XML 定義を検証するには：

1. XML エディタで定義を開きます。
2. ワークスペースの内部をクリックします。
3. [XML ビュー] - [XML 定義の妥当性検証] ををクリックします。  
[検証] ウィンドウに結果が表示されます。

## XML ビューオプションの設定

デフォルトでは、Integration Service はビュー行にデータを持つ各ビューに対して行を生成します。一部の XML ソース定義ビューに対しての行の生成方法を Integration Service で変更するには、[XML カラム] ウィンドウで XML ビューオプションを選択します。

以下の方法を使用すると、Integration Service が XML ソースから行を生成するタイミングを決定できます。

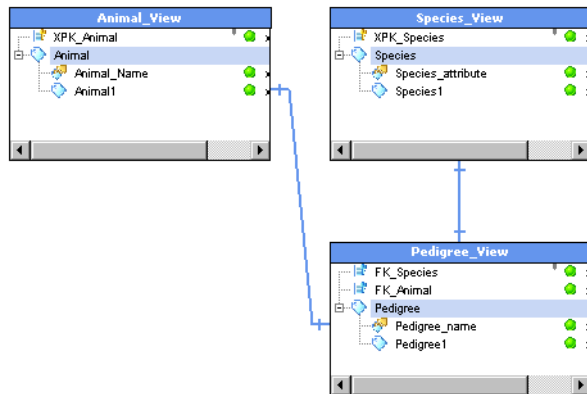
- **ビュー内のすべての外部キーを生成する。** デフォルトでは、Integration Service はビュー内の 1 つの外部キーの値を生成します。ビュー内に複数の外部キーがある場合、その他の外部キーの値は、NULL になります。すべての外部キーについて、値を生成できます。
- **循環リレーションの再帰読み込みを停止する。** デフォルトでは、Integration Service は循環リレーションのすべてのデータの出現に対し、行を生成します。最初に出現した再帰データのみに対して行を生成することができます。
- **親が存在する場合に子ビューの行を生成する。** デフォルトでは、Integration Service はビュー行内のデータを含むすべてのビューの行を作成します。親ビューにデータがある場合のみ、子ビューの行を生成します。
- **ビュー行データを含まないビューの行を生成する。** デフォルトでは、Integration Service はビュー行内にデータが含まれている場合に、ビューのデータを生成します。ビュー行にデータのないビューに対して行を生成できます。

- **型関係内の特定の複合型に対する行を生成する。** デフォルトでは、Integration Service は XML 定義内のすべてのビューに対する行を生成します。型関係の特定のビューに対して行を生成できます。

## All Hierarchy FKs の生成

デフォルトでは、Integration Service はビュー内の 1 つの外部キーの値を生成します。ビュー内に複数の外部キーがある場合、その他の外部キーの値は、NULL になります。ビュー内ですべての外部キーに対するキー値を生成することを選択できます。[All Hierarchy FKs] オプションを選択します。

以下の図は、2 つの外部キー（FK\_Species と FK\_Animal）を持つ Pedigree\_View を示しています。



Pedigree\_View で [All Hierarchy FKs] オプションを選択した場合、Integration Service は FK\_Species および FK\_Animal のキー値を生成します。

以下の図は [All Hierarchy FKs] オプションを選択した場合の Pedigree\_View のサンプルデータを示しています。

| Pedigree...   | FK_Species | FK_Animal | Pedigree1 |
|---------------|------------|-----------|-----------|
| Plains Cheeta | 1          | NULL      | 100       |
| African Lion  | 2          | NULL      | 200       |

行数: 2      OK      ヘルプ(H)

[All Hierarchy FKs] オプションを選択しない場合、Integration Service は 1 つの外部キー列に対してキー値を生成します。この例では、Species\_View が XML 階層の Pedigree\_View に最も近い親であるため、Integration Service は FK\_Species の値を生成しました。FK\_Animal 外部キーには、NULL 値があります。

以下の図は、[All Hierarchy FKs] オプションを選択しない場合の Pedigree\_View のサンプルデータを示しています。

| Pedigree...   | FK_Species | FK_Animal | Pedigree1 |
|---------------|------------|-----------|-----------|
| Plains Cheeta | 1          | NULL      | 100       |
| African Lion  | 2          | NULL      | 200       |

行数: 2      OK      ヘルプ(H)

Integration Service は、一番近い親の外部キーを生成します。

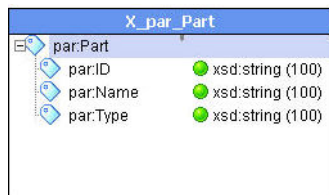
## 循環リレーションでの行の生成

デフォルトでは、Integration Service は循環リレーションのすべてのデータの出現に対し、行を生成します。最初の出現のときのみ、行の作成を選択できます。[Non-Recursive Row] オプションを選択します。

次の XML ファイルには、循環参照の Part 要素が含まれています。

```
<?xml version="1.0" encoding="utf-8"?>
<Vehicle xmlns="http://www.PartInvoice.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.PartInvoice.org part.xsd">
<VehInfo>
  <make>Honda</make>
  <model>Civic</model>
  <type>Compact</type>
</VehInfo>
<Part>
  <ID>123</ID>
  <Name>Body</Name>
  <Type>Exterior</Type>
  <Part>
    <ID>111</ID>
    <Name>DoorFL</Name>
    <Type>Exterior</Type>
    <Part>
      <ID>1112</ID>
      <Name>KnobL</Name>
      <Type>Exterior</Type>
    </Part>
    <Part>
      <ID>1113</ID>
      <Name>Window</Name>
      <Type>Exterior</Type>
    </Part>
  </Part>
</Part>
</Vehicle>
```

以下の図に、Part 要素が XML 定義内の X\_par\_Part ビューのビュー行であることを示します。



以下の図に、セッション実行時に Integration Service が Part 123 の行およびすべてのコンポーネントの part の行を生成することを示します。

☐ NonRecursive Row

| par_ID | par_Name | par_Type |
|--------|----------|----------|
| 123    | Body     | Exterior |
| 111    | DoorFL   | Exterior |
| 112    | DoorFR   | Exterior |
| 113    | DoorRL   | Exterior |

以下の図に、[NonRecursive Row] を選択した場合に Integration Service が最初に出現する Part 要素を読み取り、Part 123 のデータを 1 行生成することを示します。

☒ NonRecursive Row

| par_ID | par_Name | par_Type |
|--------|----------|----------|
| 123    | Body     | Exterior |

## 階層リレーション行の生成

デフォルトでは、Integration Service はビュー行内のデータを含むすべてのビューの行を作成します。[階層リレーション行] を選択して、親ビューの階層リレーションに対応するデータがある場合は子ビューの行を生成します。親ビューには、子ビューの行を生成するためのデータがあることが必要です。

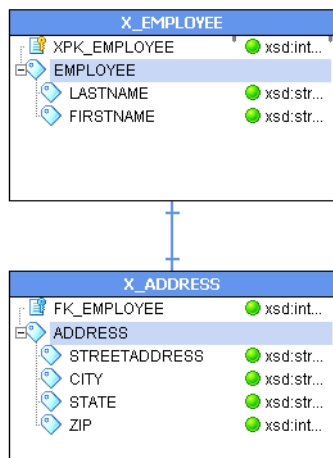
たとえば、XML 定義に、Employee ビューおよび Address ビューで構成される階層がある場合を仮定します。従業員は親のビューです。アドレスデータには、Employee\Addresses または Store\Addresses を含むことができます。Employee\Address を出力するように選択できます。

次の XML ファイルには、Store 要素内には Address があり、Employee 要素内には Address があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE STORE >
<STORE SID="BE1752">
  <SNAME>Mud and Sawdust Furniture Store</SNAME>
  <ADDRESS>
    <STREETADDRESS>335 Westshore Road</STREETADDRESS>
    <CITY>Fausta City</CITY>
    <STATE>CA</STATE>
    <ZIP>97584</ZIP>
  </ADDRESS>
  <EMPLOYEE DEPID="34">
    <ENAME>
      <LASTNAME>Bacon</LASTNAME>
      <FIRSTNAME>Allyn</FIRSTNAME>
    </ENAME>
    <ADDRESS>
      <STREETADDRESS>1000 Seaport Blvd</STREETADDRESS>
      <CITY>Redwood City</CITY>
      <STATE>CA</STATE>
      <ZIP>94063</ZIP>
    </ADDRESS>
    <EPHONE>(408)226-7415</EPHONE>
    <EPHONE>(650)687-6831</EPHONE>
  </EMPLOYEE>
</STORE>
```



以下の図に、Employee ビューと Address ビューの階層リレーションを示します。



Employee ビューは Address ビューに青い線で接続されています。これは、親と子のビューの間に 1 対 1 のリレーションを定義しています。Employee ビューにはプライマリキー XPK\_Employee、および LastName と FirstName で構成される Employee 要素があります。Address ビューには外部キー FK\_Employee、および StreetAddress、City、State、Zip で構成される Address 要素があります。

デフォルトでは、Integration Service は Address 要素の各出現に対して行を生成します。Integration Service は Store\Address および Employee\Address に対して行を 1 つずつ生成します。

以下の図は、[Hierarchy Relationship Row] オプションを選択しない場合の Address XML データを示します。

☐ 階層リレーション Row

| STREETADDRESS      | FK_EMPL... | CITY         | STATE | ZIP   |
|--------------------|------------|--------------|-------|-------|
| 335 Westshore Road | NULL       | Fausta City  | CA    | 97584 |
| 1000 Seaport Blvd  | 1          | Redwood City | CA    | 94063 |

[Hierarchy Relationship Row] オプションを選択すると、Integration Service はセッションに対して次のとおりに行を生成します。

- Integration Service は、Employee ビューのセッション内に対応するデータがある場合、Address ビューの行を生成します。
- Integration Service は、Employee\Address 階層リレーションを表す行を生成します。
- Integration Service は、Store\Address に対する行を生成しません。

以下の図は、[Hierarchy Relationship Row] オプションを選択した場合の Address データを示します。

☒ 階層リレーション Row

| STREETADDRESS     | FK_EMPL... | CITY         | STATE | ZIP   |
|-------------------|------------|--------------|-------|-------|
| 1000 Seaport Blvd | 1          | Redwood City | CA    | 94063 |

## Force Row オプションの設定

デフォルトでは、Integration Service はビュー行内にデータが含まれている場合に、ビューのデータを生成します。[Force Row] オプションを選択すると、ビュー行の要素にデータが含まれるかどうかに関係なく、XML ビューの行を生成します。たとえば、ビュー行が address\zip の場合、ビュー行に zip データがない場合でもアドレスを出力するように選択できます。

以下の図は、zip 要素をビュー行として表示しています。



zip 要素は一覧の最下部で強調表示されており、その上に street、city および state 要素行があります。

デフォルトでは、Integration Service は Address 要素内の zip 要素の各出現に対して行を生成します。

たとえば、セッションで次の XML ファイルを実行します。

```
<?xml version="1.0" ?>
<company xmlns="http://www.example.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.org forcerow.xsd" >
  <name>company1</name>
  <address>
    <street>stree1</street>
    <city>city1</city>
    <zip>1001</zip>
  </address>
  <employee>
    <name>emp1</name>
    <address>
      <street>emp1_street</street>
      <city>emp1_city</city>
    </address>
  </employee>
  <employee>
    <name>emp2</name>
    <address>
      <street>emp2_street</street>
      <city>emp2_city</city>
      <zip>2001</zip>
    </address>
  </employee>
</company>
```

Integration Service は、デフォルトでは、Address ビューの stree1 行および emp2\_street 行を生成します。

以下の図に、Address ビューの stree1 行および emp2\_street 行を示します。

☐ Force Row

| STREET      | CITY      | ZIP  |
|-------------|-----------|------|
| stree1      | city1     | 1001 |
| emp2_street | emp2_city | 2001 |

Integration Service は、ビュー行が zip であるため emp1 の行を生成せず、emp1 は zip 要素に対してデータを持っていません。

[Force Row] オプションを有効にすると、zip の有無にかかわらず、street 要素または city 要素を出力できます。セッションは、emp1 が zip 要素のデータを持っていない場合でも emp1 の行を生成します。

以下の図に、[Force Row] を有効にした場合に Integration Service が生成する行を示します。

☒ Force Row

| STREET      | CITY      | ZIP  |
|-------------|-----------|------|
| street1     | city1     | 1001 |
| emp1_street | empl_city | NULL |
| emp2_street | emp2_city | 2001 |

## 型関係のビューに対する行の生成

デフォルトでは、Integration Service は型関係内のすべてのビューに対する行を生成します。[Type Relation Row] オプションを選択して、型関係内で特定の複合型に対する行を生成します。出力したい各ビューの [Type Relation Row] オプションを設定します。

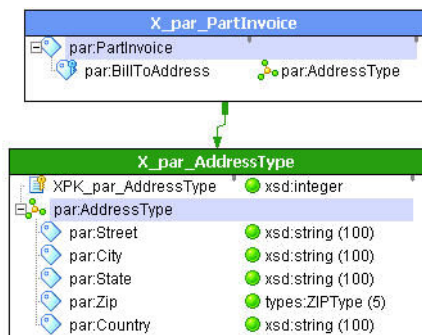
たとえば、定義に、BillToAddress および ShipToAddress を含む階層があると仮定します。BillToAddress に対して行を生成したい場合は、[Type Relation Row] オプションを使用します。

### 型関係の定義

次のスキーマは、BillToAddress および ShipToAddress を定義します。BillToAddress は AddressType で、ShipToAddress は USAddressType です。USAddressType は AddressType を拡張します。

```
<xsd:sequence>
  <xsd:element name="Street" type="xsd:string" />
  <xsd:element name="City" type="xsd:string" />
  <xsd:element name="State" type="xsd:string" />
  <xsd:element name="Zip" type="types:ZIPTYPE" />
  <xsd:element name="Country" type="xsd:string" />
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="USAddressType">
  <xsd:complexContent>
    <xsd:extension base="AddressType">
      <xsd:sequence>
        <xsd:element name="PostalCode" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="BillToAddress" type="AddressType" />
<xsd:element name="ShipToAddress" type="USAddressType" />
```

以下の図に、XML 定義での型関係を示します。X\_par\_PartInvoice XML ビューに BillToAddress 要素を示す PartInvoice ビュー、およびその下にある関連付けられた X-par\_AddressType XML ビューを使用します。



PartInvoice には、invoice データが含まれます。ビューには、BillToAddress が含まれます。XML 定義の型関係は、BillToAddress および AddressType の間にあります。

AddressType データを BillToAddress に制限するには、XML エディタのワークスペースで X\_par\_PartInvoice ビューを選択します。[Type Relationship Row] オプションを選択します。セッションを実行すると、Integration Service は Address 行を ShipToAddress ではなく BillToAddress に対して生成します。ShipToAddress は、型関係に存在しません。

### 例

この例は、型関係でデータを特定の型に限定する方法を示します。例では、PartInvoice ビューおよび AddressType ビューを使用します。

次の XML ファイルには、BillToAddress および ShipToAddress を含む invoice データが含まれます。

```
<xsd:complexType name="AddressType">
<?xml version="1.0" encoding="utf-8"?>
<Invoices xmlns="http://www.PartInvoice.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.PartInvoice.org Part.xsd">
<PartInvoice InvoiceNum="185" DateShipped="2005-01-01">
  <PartOrder>
    <PartID>HLG100</PartID>
    <PartName>Halogen Bulb</PartName>
    <Quantity>2</Quantity>
    <UnitPrice>35</UnitPrice>
  </PartOrder>
  <BillToAddress>
    <Street>2100 Seaport Blvd</Street>
    <City>Redwood City</City>
    <State>CA</State>
    <Zip>94063</Zip>
    <Country>USA</Country>
  </BillToAddress>
  <ShipToAddress xsi:type="USAddressType">
    <Street>3350 W Bayshore Rd</Street>
    <City>Palo Alto</City>
    <State>CA</State>
    <Zip>97890</Zip>
    <Country>USA</Country>
    <PostalCode>97890</PostalCode>
  </ShipToAddress>
</PartInvoice>
</Invoices>
```

セッションを実行すると、Integration Service は各 AddressType に対して X\_par\_AddressType 行を生成します。

以下の図に、Integration Service がデフォルトで生成する BillToAddress 行および ShipToAddress 行を示します。

| XPK_par_... | par_Street         | par_City    | par_State | par_Zip | par_Country |
|-------------|--------------------|-------------|-----------|---------|-------------|
| 1           | 2100 Seaport Blvd  | Redwood ... | CA        | 94063   | USA         |
| 2           | 3350 W Bayshore Rd | Palo Alto   | CA        | 97890   | USA         |

PartInvoice ビューに関連する AddressType 行を生成するには、PartInvoice ビューの [Type Relationship Row] オプションを設定します。

以下の図に、[Type Relationship Row] オプションが生成する BillToAddress 行を示します。

| XPK_par... | par_Street   | par_City    | par_State | par_Zip | par_Country |
|------------|--------------|-------------|-----------|---------|-------------|
| 1          | 2100 Seap... | Redwood ... | CA        | 94063   | USA         |

ShipToAddress は型関係ではないため、ShipToAddress の行は生成されません。

## XML エディタを使用する作業のトラブルシューティング

XML 定義を検証すると、XML 定義が大きすぎるというエラーが表示されます。どうしてでしょうか？

無限長に定義されたコンポーネントを含む XML ファイルをインポートする場合、合計カラム長の制限である 500MB を簡単に超えてしまいます。XML エディタでカラム長を変更することができます。または、オプションを設定して、すべての無限長を上書きしてファイルを再インポートすることができます。

XML メタデータを表示したときに作成した DTD または XML ファイルが見つかりません。

表示できる DTD または XML スキーマファイルは、Designer が表示のために作成する一時ファイルです。ファイルを他の目的で使いたい場合は、表示したときに別の名前を付けて別のディレクトリに保存してください。

XML ソースビューにカラムを追加した場合、ソース XML ファイルの階層は変更されません。

XML ソースビューにカラムを追加しても、そのベースとなっている階層に要素を追加したことにはなりません。ビューをどのように作成しても、ビューのカラムを階層の要素にどのようにマップしても、インポートした XML 階層は変更されません。要素のデータ型やカーディナリティを変更することはできますが、階層の構造を変更することはできません。

PowerCenter での XML サイズ設定の詳細については、『[「PowerCenter での XML の使用の概要」 \(ページ 29\)](#)』を参照してください。PowerCenter で XML 処理に適用される制限の詳細については、『[「制限」 \(ページ 29\)](#)』を参照してください。

他の要素型を持つトランスフォーメーションを作成し、大きな XML 入力ファイルを変換するには、データプロセッサトランスフォーメーションを使用します。データプロセッサトランスフォーメーションの作成方法の詳細については、『*Informatica Data Transformation ユーザーガイド*』および『*Informatica Data Transformation 入門ガイド*』を参照してください。

## XML ターゲットに関する作業

### XML ターゲットに関する作業の概要

以下の方法で、XML ターゲット定義を作成できます。

- **XML スキーマまたはファイルから定義をインポートする。**XML、DTD、または XML スキーマの各ファイルからターゲット定義を作成できます。URL またはローカルノードから XML ファイル定義をインポートすることができます。XML ファイルを関連 DTD と一緒にインポートする場合、XML ウィザードは DTD を使用して XML ファイルを生成します。

- **XML ソース定義に基づく XML ターゲット定義を作成する。** 既存の XML ソース定義を Target Designer にドラッグできます。XML ターゲット定義を作成する場合、Designer は、XML 定義の階層に基づいてターゲット定義を作成します。
- **リレーショナルファイル定義に基づいて XML ターゲットを作成する。** リレーショナルまたはフラットファイルリポジトリ定義から XML ターゲット定義をインポートすることができます。

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

- **ターゲットプロパティの編集。** XML ターゲット定義を編集してコメントを追加し、ターゲット XML、DTD、または XML スキーマファイルに変更を反映させます。
- **ターゲット定義の同期。** 変更を加える必要がある場合、ターゲット XML 定義をスキーマに同期させることができます。定義を同期させると、スキーマが変更した場合にスキーマをインポートしなくても XML 定義を更新することができます。

## XML ファイルからの XML ターゲット定義のインポート

XML スキーマ、DTD ファイル、または XML ファイルから XML 定義をインポートできます。URL を使用して、ローカルファイルまたは参照するファイルをインポートすることができます。Designer がデータを正確に定義できることを確実にするために、XML スキーマからターゲット定義をインポートします。

XML ビューを作成するために、以下のオプションのいずれかを選択できます。

- **エンティティリレーションの作成。** このオプションを使用して、複数出現要素、参照要素、および複合型のビューを作成します。1 つの大きな階層を作成する代わりにビュー間の関係を作成します。
- **階層リレーションの作成。** このオプションを使用して、ルートを作成し、そのルートの下に XML コンポーネントを展開します。正規化または非正規化ビューの作成を選択できます。正規化を選択すると、すべての要素および属性が一度に表示されます。1 対多関係は、ビューに関連付けるキーを持つ個別の XML ビューになります。非正規化 XML ビューを作成すると、すべての要素および属性は、1 つの階層グループに表示されます。
- **カスタム XML ビューの作成。** このオプションを使用して、複数のグローバル要素を XML ビューのルートとして選択し、メタデータの膨張を抑制するオプションを選択します。
- **XML ビュー作成のスキップ。** このオプションにより、ビューを作成しなくても定義をインポートすることができます。このオプションを選択すると、XML エディタを使用して、ワークスペースでの XML ビューの作成をしばらく後になってから実行することができます。
- **XML 定義の同期。** このオプションを使用して、ベースとなるスキーマが変更されたときに、1 つ以上の XML 定義を更新します。

**ヒント:** XML ファイルの代わりに DTD または XML スキーマファイルをインポートします。XML ファイルに関連 DTD と一緒にインポートする場合、XML ウィザードは DTD を使用します。

XML ターゲット定義をインポートするには：

1. Target Designer で、[ターゲット] - [XML 定義のインポート] をクリックします。  
[XML 定義のインポート] ウィンドウが開きます。デフォルトでは、ローカルフォルダのスキーマファイルが表示されます。
2. [ローカルファイル] または [URL] をクリックして、XML ファイルをブラウズします。
3. DTD または XML ファイルをブラウズするには、[ファイルの種類] リストから該当するファイル拡張子を選択します。



## XML ソース定義からのターゲットの作成

既存のソース定義に非常に似ているターゲット定義を作成する場合、ソース定義、またはソース定義へのショートカットを使用してターゲット定義を作成します。XML ソース定義を Target Designer にドラッグして、XML ターゲット定義またはリレーショナルターゲット定義を作成します。

XML ターゲット定義を作成するには、以下のガイドラインを使用します。

- XML ソース定義から XML ターゲット定義を作成すると、XML ソース定義をコピーした XML ターゲット定義が作成されます。
- 有効な XML ソース定義から、必ず有効な XML ターゲット定義が作成されるとは限りません。確実に有効なターゲット定義を作成するために、ターゲットを定義を検証してください。

**注:** XML ターゲット定義にピボット化されたカラムを収めることはできません。

リレーショナルターゲット定義を作成するには、以下のガイドラインを使用します。

- リレーショナルターゲット定義を作成すると、XML ソース定義のグループに基づいてリレーショナルターゲット定義が作成されます。XML ソース定義の各グループは、ターゲット定義になります。
- Designer は、ソース定義のグループ間の場合と同じようにリレーションをターゲット定義間に作成します。

XML 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. XML ソース定義をナビゲータから Target Designer のワークスペースにドラッグします。  
[XML エクスポート] ダイアログボックスが表示されます。
2. 選択して、リレーショナルターゲットまたは XML ターゲットを作成します。[OK] をクリックします。

Target Designer のワークスペースに、ターゲット定義が表示されます。リレーショナルターゲットを選択した場合、ソースにより複数のターゲット定義がワークスペースに表示される場合があります。

## XML ターゲット定義のプロパティの編集

XML ターゲット定義を作成した後で、ターゲットデータの変更を反映させる、ビジネス名やコメントを追加する、コードページを変更するといったプロパティの編集が実行できます。

XML ターゲット定義のプロパティを編集するには、

1. Target Designer で XML ターゲットを開きます。
2. 右クリックして、[編集] を選択します。
3. [テーブル] タブで設定を編集します。



以下の表に、テーブル設定項目を示します。

| テーブル設定項目  | 説明  |
|-----------|---|
| テーブルの選択   | ターゲット定義の名前。名前を変更するには、[名前の変更] をクリックします。  |
| ビジネス名     | ターゲットテーブルの説明的な名前。[名前の変更] を使用してビジネス名を編集します。  |
| 制約        | XML ターゲットには適用されません。すべてのエントリは無視されます。   |
| 作成オプション   | XML ターゲットには適用されません。すべてのエントリは無視されます。   |
| 説明        | ターゲットテーブルの説明。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。                |
| コードページ    | ターゲット定義で使用するコードページを選択します。   |
| データベースタイプ | ターゲット定義は XML ターゲットであることを示します。   |
| キーワード     | キーワードを使用して、ターゲットを整理し識別します。キーワードには、開発名、マッピング、または XML スキーマ名を指定できます。<br>Repository Manager で、キーワードを使用して検索を実行します。 |

4. [カラム] タブで、XML カラム定義を表示できます。

以下の表に、カラムの設定項目を示します。

| カラムの設定項目 | 説明   |
|----------|--|
| テーブルの選択  | 編集中のターゲット定義を表示します。編集する別の定義を選択するには、ワークスペースで選択可能な定義のリストから 1 つを選択します。 |
| カラム名     | カラムの名前。  |
| データ型     | カラムの PowerCenter データ型。   |
| 精度       | カラムのサイズ。文字列などの一部のデータ型に対して精度を変更できません。                               |
| スケール     | 数値の小数点以下の最大桁数です。   |
| 非 Null   | カラムが NULL 値を持つことが可能かどうかを示します。                                      |
| Key Type | ビューにリンクするために、XML ウィザードにより生成されるキーのタイプです。                            |
| XPath    | 項目の位置を表す XML ファイルの階層のパスです。   |

5. [プロパティ] タブでは、ターゲット定義のトランスフォーメーション属性を変更することができます。

ソーススペースのコミットセッション、または、XML ターゲットの Transaction Control トランスフォーメーションを使用している場合は、データをターゲットにフラッシュする方法を定義できます。

以下の表に、編集が可能な属性を示します。

| カラムの設定項目      | 説明   |
|---------------|--|
| テーブルの選択       | 編集中のソース定義を表示します。別のソース定義を選択するには、リストからソース定義を選びます。  |
| 重複グループ行の処理    | <p>以下のオプションのいずれかを選択して、ターゲット内の重複行の処理方法を選択します。</p> <ul style="list-style-type: none"> <li>- 最初の行。Integration Service は、重複行のうち最初の行をターゲットに渡します。同一のプライマリキーを持つ後続行はリジェクトされます。</li> <li>- 最後の行。Integration Service は、重複行のうち最後の行をターゲットに渡します。</li> <li>- エラー。Integration Service は、最初の行をターゲットに渡します。重複するプライマリキーを持つ行が見つかったら、エラーカウントが1つ増やされます。エラーカウントがエラーしきい値に達するとセッションが失敗します。</li> </ul> |
| DTD Reference | ターゲット XML ファイルの DTD または XML スキーマファイル名。Integration Service は XML ファイルを作成する場合、XML ファイルに対して文書型の宣言を追加します。  |
| コミット時         | <p>Integration Service は、コミット後に、複数の XML ファイルを生成することも、1つの XML 文書に書き込むこともできます。次のいずれかのオプションを使用します。</p> <ul style="list-style-type: none"> <li>- コミットの無視。Integration Service は、XML ファイルを作成し、ファイルの最後で XML ファイルに書き込みます。</li> <li>- 新しいドキュメントの作成。コミットごとに新しい XML ファイルを作成します。</li> <li>- ドキュメントへの書き込み。コミットごとに同一の XML ファイルに書き込みます。</li> </ul>  |
| キャッシュディレクトリ   | XML ターゲットキャッシュファイルのディレクトリ。デフォルトは、\$PMCacheDir サービスプロセス変数です。  |
| キャッシュサイズ      | XML ターゲットキャッシュの合計サイズ（バイト単位）。デフォルトは「自動」です。  |

6. [メタデータエクステンション] タブで、再利用不可能なメタデータエクステンションに対して、値の更新、作成、変更、削除、格上げを実行できます。再利用可能なメタデータエクステンションの値の更新も可能です。
7. [OK] をクリックします。
8. [リポジトリ] - [保存] をクリックします。

## XML ターゲットの検証

データを XML ファイルに抽出する方法を示すカスタマイズされた XML ビューを作成することができます。ただし、すべてのビュー構造、またはビュー間の関係が XML 定義内で有効であるという訳ではありません。いくつかのビュー構造は、XML ソースに対しては有効で、XML ターゲットに対しては無効である場合があります。Designer を使用することで、あいまいな定義の作成を防ぐことができます。

以下の作業を実行すると、PowerCenter はターゲット XML ビューを検証します。

- XML ターゲットをリポジトリに保存したり、リポジトリから取り出したりする場合、Designer は限定的な検証を実行します。
- XML ワークスペースで XML を編集する場合、XML エディタの各ステップで検証が行われます。
- XML エディタで編集中のターゲット定義を検証することを選択できます。
- Designer はマッピングを検証する際に、XML ターゲット接続を検証します。

Designer はルールを使用して、階層リレーション、型関係、継承関係を検証します。

**注:** Integration Service は、セッションのスキーマに対して、ターゲット XML インスタンスを検証しません。データの単純なタイプを検証するように、ターゲットの検証プロパティを設定できます。

## 階層リレーションの検証

Designer は、以下のルールを使用して階層リレーションを検証します。

- タイプにルートを持つビューは、スタンドアロンビューにはできません。継承関係の子であるか、または別のビューとの型関係を持っていることが必要です。ビューの要素にルートがない場合、XML ターゲットは無効です。
- 複数出現ビュー行を含むビューと別のビューを接続する必要があります。
- 有効なビュー行を 2 つのビューが共有することはできません。
- ビューの要素にルートがない場合、XML ターゲットは無効です。
- 親のビューと子のビューは他の要素によって分けることができます。ただし、ビューに 2 つの親がある場合、近いほうの親を選ばなければなりません。有効なビュー行のパスにより、最も近い親を判別します。パス内で一方の親が他方の親に先行しています。パス内の 2 番目のビューを選択します。
- 同じ階層で、すべてのビューを同じビューのルートに接続する必要があります。同じビューのルートで複数のツリーを定義に含むことはできません。
- 該当するビューでビュー行およびビュールートが同じである場合、XML ビューがそれ自体で階層関係を持つことができます。

## 型関係の検査

型関係とは、カラムとビューの関係です。型関係は、2 つのビューの間関係ではありません。型関係には以下のルールが適用されます。

- ビュー V1 のカラムとビュー V2 が型関係を持つことができるのは、両者のビュールートが同じ型である場合、あるいは V2 のルートの型が V1 のビュールートから派生している場合です。ビュールートは両方ともグローバルな複合型である必要があります。
- ビューのカラムが他のビューと型関係を持つ場合、カラムを展開することはできません。

## 継承の検証

XML ビューで 2 つの型の継承関係を作成することができます。

- **ビューからビューへの継承。**どのビューも他のビューに由来する派生型です。どちらのビューも、グローバル複合ビューのルートを持っている必要があります。

2 つのビュー間で継承関係が成立するのは、一方のビュールートが他方のビューのビュールート型から派生している複合型である場合です。

ビューは複数の継承関係で親になることができますが、ビューが子になることができるのは 1 つの継承関係のときです。

- **列からビューへの継承。**列はローカル複合型（タイプ 1 など）の要素ですが、ビューは、グローバル複合型（タイプ 2）にルートがあります。タイプ 1 はタイプ 2 が派生したものです。

ビューのカラムが別のビューと継承関係を持つことができるのは、カラムがローカル複合型であり、しかも型が他のビューのビュールート型から派生している場合です。

ビュー V1 のカラムがビュー V2 との継承関係を持っている場合、V2 の内容をビュー V1 に含むことはできません。

## マッピングにおける XML ターゲットの使用

XML ターゲットをマッピングに追加する場合、マルチグループのトランスフォーメーションに関するマッピングのガイドラインに従う必要があります。

以下のコンポーネントは、XML ターゲットをマッピングする方法に影響を与えます。

- アクティブソース
- ルート要素
- ターゲットポート接続
- 抽象要素
- トランザクション制御点
- FileName カラム

## アクティブソース

アクティブソースは、入力行ごとに異なる数の行を返すことができるトランスフォーメーションです。Integration Service は、異なるアクティブソースから XML ターゲットにデータをロードすることができます。ただし、XML ターゲットの 1 つのグループに属するすべてのポートは、同じアクティブソースからデータを受け取る必要があります。

以下のトランスフォーメーションはアクティブソースです。

- Aggregator
- アプリケーションソース修飾子
- Custom（アクティブなトランスフォーメーションとして設定されていること）
- Java（アクティブなトランスフォーメーションとして設定されていること）
- Joiner
- MQ Source Qualifier

- Normalizer (VSAM またはパイプライン)
- Rank
- Sorter
- Source Qualifier
- SQL
- XML Source Qualifier
- マプレット (そのマプレットに、上記のトランスフォーメーションのいずれかが含まれる場合)

## ルート要素の選択

XML 定義にルートになりうる要素が複数ある場合、ターゲットインスタンスのルート要素を指定することができます。

ルート要素を指定するには：

1. Mapping Designer でターゲット定義を右クリックして、[編集] を選択します。
2. [プロパティ] タブをクリックします。
3. [Root Element] 値カラムの矢印をクリックします。  
[Root 選択] ダイアログボックスが表示されます。
4. リストから要素を選択します。

## ターゲットポートの接続

マッピングにおいて XML ターゲットポートを正しく接続することで、Integration Service がセッションの実行中に有効な XML ファイルを作成できるようにする必要があります。XML ターゲットを含むマッピングを保存するか、または検証すると、Designer はターゲットポート接続を検証します。

マッピング内のポートを接続する場合は、以下のガイドラインを使用します。

- グループの 1 つのポートを接続する場合、グループの外部キーおよびプライマリキーのポートの両方を接続する必要があります。
- グループの外部キーポートを接続する場合、他のグループの関連付けられたプライマリキーポートを接続する必要があります。ルートグループのプライマリキーポートを接続しない場合は、他のグループの関連付けられた外部キーポートを接続する必要はありません。
- デフォルトの属性値を持つ XML スキーマを使用する場合、属性ポートを接続してターゲットにデフォルトの属性を作成する必要があります。接続ポートを介して NULL 値が渡されると、Integration Service はターゲットにデフォルトの値を書き込みます。

## 抽象要素の接続

XML インスタンスファイルで抽象要素を直接出現させることはできません。したがって、抽象要素から派生した要素を使う必要があります。デフォルトの場合、複合要素が抽象要素であれば、Designer がビューを作成します。メタデータを抑制する場合、抽象型からの要素が派生型で反復することはありません。データを抽象型にマッピングする場合、派生型（少なくとも 1 つ）に対してもデータをマッピングする必要があります。

セッションの実行中、Integration Service がデータを抽象型にロードした場合、Integration Service は抽象型ではない派生型のデータも抽象型と関連させる必要があります。派生型にデータがない場合、Integration Service が抽象要素をターゲット XML ファイルに書き込むことはありません。

## ターゲットへの XML データのフラッシュ

セッションの実行中、各コミットポイントで XML ターゲットにデータをフラッシュできます。ただし、マッピング中にそれぞれの入力グループが同一のトランザクション制御点からデータを受け取る必要があります。このマッピングに基づいてセッションを作成する場合、各コミットで XML ファイルターゲットにデータを追加するか、各コミットで新しいファイルを作成するかを選択できます。[On Commit] ターゲットプロパティでいずれかのオプションを指定することができます。

XML ターゲット入力グループを複数のトランザクション制御ポイントに接続する場合、Integration Service はすべてのソース行の処理後にデータを XML ファイルターゲットに書き込みます。

## XML ファイルの動的命名法

[FileName] 列を XML ターゲット定義に追加すると、XML ファイルに対するファイル名を動的に生成できます。Integration Service では、[FileName] 列にデータを渡すと、ターゲットプロパティの出力ファイル名を上書きします。たとえば、ユーザーが文字列「Harry」を [FileName] 列に渡すと、Integration Service は XML ファイルに Harry という名前を付けます。

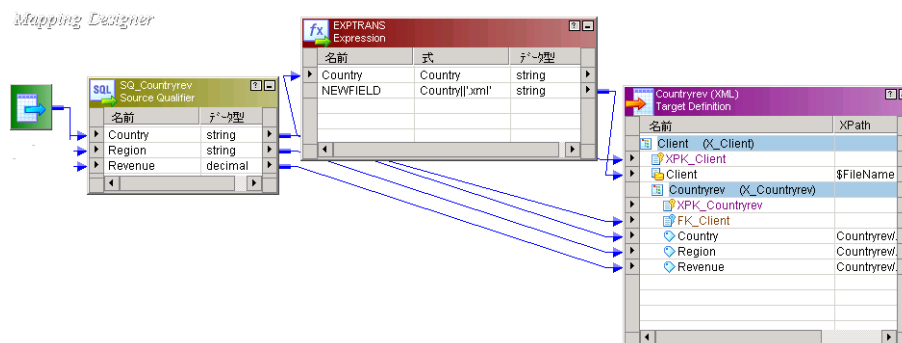
**注:** 各コミットで XML ファイルを新規作成する場合、ユーザーは作成対象の各 XML ファイルに対して動的に命名する必要があります。各 XML ファイルに対して動的な命名は行わない場合、Integration Service は前回のコミットからの XML ファイルに上書きします。

Integration Service は、ターゲットのルートグループで区別されているそれぞれのプライマリー値に対して、新しい XML ファイルを生成します。[FileName] カラムを追加し、各ファイルに対して異なる名前を設定します。それぞれの名前は、セッションプロパティの出力ファイル名を上書きします。

### 例

Expression トランスフォーメーションは、Country という XML 要素からファイル名を生成して、値を [FileName] カラムに渡します。マッピングは、国をターゲットのルート (Client) に渡します。Client の値が変化すると、Integration Service は新しい XML ファイルを作成します。Integration Service によって、各 XML ターゲットファイル名を含むリストファイルが作成されます。リスト内の各ファイルに対する絶対パスが一覧表示されます。

以下の図に、[FileName] カラムのある XML ターゲットが含まれているマッピングを示します。



Integration Service は、以下に示す行をターゲットに渡します。

```
Country,Region,Revenue
USA,region1,1000
France,region1,10
Canada,region1,100
USA,region2,200
USA,region3,300
USA,region4,400
France,region2,20
France,region3,30
France,region4,40
```

このセッションでは、国名に従って以下のファイルを作成します。

```
Canada.xml
France.xml
USA.xml
```

リストファイル名は、セッションプロパティの出力ファイル名です。

```
revenue_file.xml.lst
```

## XML ターゲットのトラブルシューティング

XML ファイルからソース定義をインポートしました。そして、同じ XML ファイルからターゲット定義をインポートしました。ソース定義とターゲット定義ではデフォルトグループが同じになりません。

ターゲットのインポート時にいくつかのオプションを変更した場合、XML ウィザードがソース定義とターゲット定義に対して常に同じグループ構造を作成するとは限りません。

たとえば、次の DTD では ContactInfo 要素が囲み要素になります。囲み要素にはテキストが含まれていませんが、maxOccurs > 1 となっています。また、子要素も maxOccurs > 1 となっています。

```
<!ELEMENT HR (EMPLOYEE+)>
<!ELEMENT EMPLOYEE (LASTNAME,FIRSTNAME,ADDRESS+,CONTACTINFO+)>
<!ATTLIST EMPLOYEE EMPID CDATA #REQUIRED>
<!ELEMENT LASTNAME (#PCDATA)>
<!ELEMENT FIRSTNAME (#PCDATA)>
<!ELEMENT ADDRESS (STREETADDRESS,CITY,STATE,ZIP)>
<!ELEMENT STREETADDRESS (#PCDATA)>
<!ELEMENT CITY (#PCDATA)>
<!ELEMENT STATE (#PCDATA)>
<!ELEMENT ZIP (#PCDATA)>
<!ELEMENT CONTACTINFO (PHONE+,EMERGCONTACT+)>
<!ELEMENT PHONE (#PCDATA)>
<!ELEMENT EMERGCONTACT (#PCDATA)>
```

ソース定義で囲み要素の XML ビューを作成しない場合、ソースの ContactInfo 要素は作成されません。



以下の図に、XML ウィザードが作成するソース定義とターゲット定義を示します。

| 名前                          | データ型        |
|-----------------------------|-------------|
| EMPLOYEE (EMPLOYEE)         |             |
| GPK_EMPLOYEE                | xsd:integer |
| LASTNAME                    | xsd:decimal |
| FIRSTNAME                   | xsd:decimal |
| ADDRESS (ADDRESS)           |             |
| GPK_ADDRESS                 | xsd:integer |
| STREETADDRESS               | xsd:decimal |
| CITY                        | xsd:decimal |
| STATE                       | xsd:decimal |
| ZIP                         | xsd:decimal |
| PHONE (PHONE)               |             |
| GPK_PHONE                   | xsd:integer |
| PHONE                       | xsd:decimal |
| EMERGCONTACT (EMERGCONTACT) |             |
| GPK_EMERGCONTACT            | xsd:integer |
| EMERGCONTACT                | xsd:decimal |

| 名前                          | データ型        |
|-----------------------------|-------------|
| HR (HR)                     |             |
| GPK_HR                      | xsd:string  |
| HR                          | xsd:string  |
| EMPLOYEE (EMPLOYEE)         |             |
| GPK_EMPLOYEE                | xsd:string  |
| EMPLOYEE                    | xsd:string  |
| LASTNAME                    | xsd:decimal |
| FIRSTNAME                   | xsd:decimal |
| ADDRESS (ADDRESS)           |             |
| GPK_ADDRESS                 | xsd:string  |
| STREETADDRESS               | xsd:decimal |
| CITY                        | xsd:decimal |
| STATE                       | xsd:decimal |
| ZIP                         | xsd:decimal |
| CONTACTINFO (CONTACTINFO)   |             |
| GPK_CONTACTINFO             | xsd:string  |
| CONTACTINFO                 | xsd:string  |
| PHONE (PHONE)               |             |
| GPK_PHONE                   | xsd:string  |
| PHONE                       | xsd:decimal |
| EMERGCONTACT (EMERGCONTACT) |             |
| GPK_EMERGCONTACT            | xsd:string  |
| EMERGCONTACT                | xsd:decimal |

ソース定義は ContactInfo 要素を含みません。ターゲット定義は ContactInfo 要素を含みます。ソースを作成した際に囲み要素のビューを作成しないことを選択したため、ウィザードは ContactInfo 要素をソース定義に含めませんでした。しかし、ターゲット定義には ContactInfo 要素を含めます。

リレーショナルソースから作成した XML ターゲット定義にすべての要素が含まれていますが、属性が含まれていません。ターゲット階層をどのように変更すれば、特定のデータに属性として印を付けられるようになりますか？

ウィザードでリレーショナルテーブルから作成した場合、コンポーネントの型を変更することはできません。ただし、ターゲット XML 階層であれば、DTD または XML スキーマファイルを表示できます。DTD または XML スキーマファイルを、新しいファイル名で保存してください。この新しいファイルを開いて階層を変更すると、属性や要素を設定できます。さらに、このファイルを利用すると、新しい階層を含んだターゲット定義をインポートできます。

## XML Source Qualifier トランスフォーメーション

### XML ソース修飾子トランスフォーメーションの概要

マッピングに XML ソース定義を追加する場合、XML ソース修飾子トランスフォーメーションに XML ソース定義を接続する必要があります。XML ソース修飾子トランスフォーメーションは、Integration Service がセッション実行中に読み込む要素を定義します。XML ソース修飾子によって、PowerCenter でのソースデータの読み込む方法が決まります。XML ソース修飾子トランスフォーメーションはアクティブなトランスフォーメーションです。

ソース修飾子トランスフォーメーションは手動で追加できます。また、マッピングにソース定義を追加する際、デフォルトでソース修飾子トランスフォーメーションを作成することもできます。

プロパティのいくつかを編集し、メタデータエクステンションを XML ソース修飾子トランスフォーメーションに追加することができます。

マッピングで XML ソース修飾子トランスフォーメーションを接続する場合には、ルールに従って有効なマッピングを作成する必要があります。

## マッピングへの XML Source Qualifier の追加

XML Source Qualifier トランスフォーメーションは、XML ソースの各カラムに対して入出力ポートを 1 つ持っています。ソース定義の XML Source Qualifier トランスフォーメーションを作成すると、Designer は XML ソース定義の各ポートを XML Source Qualifier トランスフォーメーションのポートにリンクします。リンクの削除や編集はできません。XML ソース定義をマッピングから削除すると、Designer は対応する XML Source Qualifier トランスフォーメーションも削除します。1 つの XML ソース定義を、1 つの XML Source Qualifier 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。

1 つの XML Source Qualifier グループのポートを他のトランスフォーメーションのポートにリンクし、別々のデータフローを作成することができます。ただし、XML Source Qualifier トランスフォーメーションの複数のグループのポートを同じターゲットトランスフォーメーションのポートにリンクすることはできません。

複数のグループのカラムを 1 つのトランスフォーメーションにドラッグすると、Designer はすべてのグループのカラムをトランスフォーメーションにコピーします。ただし、Designer がトランスフォーメーションの新しいカラムでの対応するポートにリンクするのは、最初のグループのポートだけです。

XML Source Qualifier トランスフォーメーションをマッピングに追加するには、XML ソース定義を Mapping Designer のワークスペースにドラッグするか、または手動で作成します。

### デフォルトでの XML Source Qualifier トランスフォーメーションの作成

XML ソース定義を Mapping Designer のワークスペースにドラッグした場合、デフォルトでは Designer が XML Source Qualifier トランスフォーメーションを作成します。

XML Source Qualifier トランスフォーメーションをデフォルトで作成する手順

1. Mapping Designer で、新しいマッピングを作成するか、既存のマッピングを開きます。
2. XML ソース定義をマッピング内にドラッグします。

Designer は XML Source Qualifier トランスフォーメーションを作成し、XML ソース定義の各ポートを XML Source Qualifier トランスフォーメーションのポートにリンクします。

### XML Source Qualifier トランスフォーメーションの手動作成

Source Qualifier を持たない XML ソース定義を含むマッピングがある場合、または XML Source Qualifier トランスフォーメーションをマッピングから削除した場合に、XML Source Qualifier トランスフォーメーションをマッピングに作成することができます。

XML Source Qualifier トランスフォーメーションを手動で作成する手順

1. Mapping Designer で、新しいマッピングを作成するか、既存のマッピングを開きます。  
**注:** Source Qualifier を持たない XML ソース定義がマッピングに少なくとも 1 つは必要です。
2. [トランスフォーメーション] - [作成] をクリックします。  
[トランスフォーメーションの作成] ダイアログボックスが表示されます。
3. [XML Source Qualifier] トランスフォーメーションを選択して、トランスフォーメーションの名前を入力します。  
XML Source Qualifier トランスフォーメーションの命名規則は、XSQ\_TransformationName です。
4. [作成] をクリックします。

Designer はマッピング内の対応する XML Source Qualifier トランスフォーメーションがない XML ソース定義をすべて一覧表示します。

5. ソース定義を選択して、[OK] をクリックします。

Designer はマッピングに XML Source Qualifier トランスフォーメーションを作成し、XML ソース定義の各ポートを XML Source Qualifier トランスフォーメーションのポートにリンクします。

## XML ソース修飾子トランスフォーメーションの編集

トランスフォーメーション名や説明などの XML ソース修飾子トランスフォーメーションのプロパティを編集することができます。

XML ソース修飾子トランスフォーメーション編集する手順

1. Mapping Designer で XML ソース修飾子トランスフォーメーションを開きます。
2. [トランスフォーメーション] タブでプロパティを編集します。

以下の表に、トランスフォーメーションのプロパティを示します。

| トランスフォーメーションの設定 | 説明  |
|-----------------|---|
| トランスフォーメーションの選択 | 編集中のトランスフォーメーションを表示します。編集する別のトランスフォーメーションを選択するには、リストから選びます。 |
| 名前の変更           | トランスフォーメーションの名前を編集します。                                      |
| 説明              | トランスフォーメーションの説明です。  |

3. [ポート] タブをクリックして、XML ソース修飾子トランスフォーメーションポートを表示します。  
[シーケンス] カラムを使用して、XML グループの生成されたキーの開始値を設定します。生成キーごとに、異なる値を入力することができます。シーケンスキーのデータ型は、bigint 型です。これらの値を変更すると、次にセッションを実行するときにシーケンス番号はその値からリスタートします。
4. [プロパティ] タブをクリックして、Integration Service がセッション中にマッピングを実行する方法に影響を与えるプロパティを設定します。

以下の表に、XML ソース修飾子のプロパティを示します。

| プロパティの設定        | 説明   |
|-----------------|--|
| トランスフォーメーションの選択 | 編集中のトランスフォーメーションを表示します。編集する別のトランスフォーメーションを選択するには、リストから選びます。  |
| トレースレベル         | Integration Service がワークフローを実行した場合に、このトランスフォーメーションについてセッションログに書き込む情報の量を指定します。このトレースレベルは、セッションを設定するときに上書きすることができます。 |

| プロパティの設定 | 説明   |
|----------|--|
| リセット     | セッションの終了時、Integration Service はその時点の開始値を現行セッションの開始値にリセットします。 |
| リスタート    | セッションの開始時、Integration Service は全グループの生成キーシーケンスを 1 から開始します。   |

5. [メタデータエクステンション] タブをクリックして、ユーザー定義のメタデータエクステンションを作成、編集、または削除します。

再利用不可能なメタデータエクステンションに対して、値の作成、変更、削除、格上げに加え、値の更新も実行できます。再利用可能なメタデータエクステンションの値の更新も実行できます。

6. [OK] をクリックします。

## 生成されたキーのシーケンス番号の設定

XML Source Qualifier 定義の各ビューは、プライマリキーとこのキーのシーケンス値を持ちます。セッションの実行時、Integration Service はシーケンス値からキーを生成し、シーケンス値を増加します。

セッションの終了時、Integration Service はリポジトリのシーケンス値を、カレント値より 1 大きい値に更新します。次回、Integration Service プロセスが Sequence Generator トランスフォーメーションを処理するときに、この値が開始値になります。

リポジトリでは以下のシーケンス値を保持します。

- **デフォルト値。** ソース修飾子を最初に作成したときに XML Source Qualifier で表示される値です。それぞれのキーに対して、デフォルトは 1 です。
- **開始値。** セッション開始時のキーに対するシーケンス番号値です。XML Source Qualifier トランスフォーメーションで開始値を表示してから、ワークフローを実行することができます。
- **カレント値。** セッション実行中のキーに対するシーケンス値です。

生成されたキーの開始値が XML Source Qualifier の [シーケンス] カラムに表示されます。

**注:** [ポート] タブでシーケンスの開始値を編集する場合、ワークフローを実行する前に変更内容を保存し、Designer を終了する必要があります。

### シーケンス開始値の変更

セッションの実行前または実行後にシーケンス開始値の変更が可能です。その場合、XML Source Qualifier トランスフォーメーションの [プロパティ] タブで、以下のオプションを選んでください。

- **リセット。** セッションの終了時、Integration Service はその時点の開始値を現行セッションの開始値にリセットします。たとえば、セッション開始時にキーの開始値が 2000 です。セッション終了時はカレント値が 2500 です。セッションが終了しても、リポジトリの開始値は 2000 のままになります。現在試験が進行中であり、次のセッション実行時にも同じキー番号を生成したい場合、このオプションを利用することになるかもしれません。
- **リスタート。** セッションの開始時、Integration Service はデフォルト値に従って開始値を再開します。たとえば、キーの開始値が 1005 のときに [リスタート] を選択すると、Integration Service は開始値を 1 に変更します。使用しているキーの数が増えてきていて、番号付けをリスタートしてもキーの重複が発生しない場合に、このオプションが利用できます。

## マッピングでの XML Source Qualifier の使用

XML ソース定義の各グループはリレーショナルテーブルに似ており、Designer は XML Source Qualifier トランスフォーメーション内の各グループをデータの個々のソースとして扱います。

マッピングでオブジェクトを接続するときに、Designer は結合ルールを維持します。したがって、1 つのパイプラインブランチに必要な情報をすべて各グループに含むように、XML ソース定義のグループを整理します。

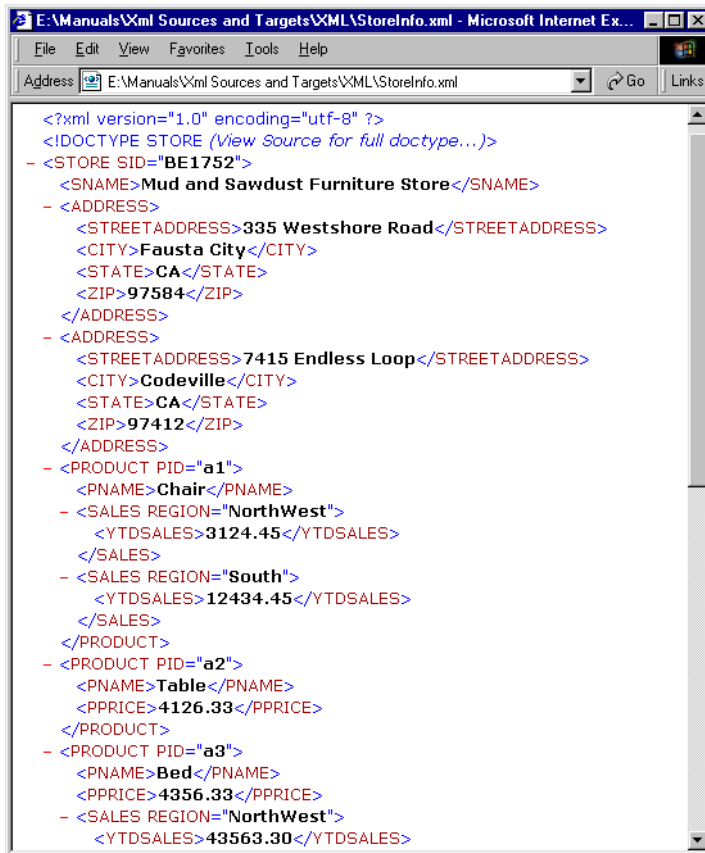
マッピングで XML Source Qualifier トランスフォーメーションを接続するときには、以下のルールを考慮してください。

- **XML Source Qualifier トランスフォーメーションの 1 つのグループのポートを、別のトランスフォーメーションの 1 つの入力グループのポートにリンクすることができます。** 複数グループの列を 1 つのトランスフォーメーションにコピーすることはできますが、トランスフォーメーションの対応するポートにリンクできるのは 1 つのグループのポートのみです。
- **XML Source Qualifier トランスフォーメーションの 1 つのグループのポートを、複数のトランスフォーメーションのポートにリンクすることができます。** XML Source Qualifier トランスフォーメーションの各グループは、複数のパイプラインブランチのデータのソースとなることができます。データは、1 つのグループから複数の異なるトランスフォーメーションに渡すことができます。
- **1 つの XML Source Qualifier トランスフォーメーションの複数のグループを、異なるトランスフォーメーションの入力グループにリンクすることができます。** 1 つの XML Source Qualifier トランスフォーメーションの複数のグループを、Joiner または Custom トランスフォーメーションなど、ほとんどの複数の入力グループトランスフォーメーション内の異なる入力グループにリンクすることができます。ただし、Joiner トランスフォーメーションで入力のスートが済んでいる場合は、1 つの XML Source Qualifier トランスフォーメーションの複数のグループを、1 つの Joiner トランスフォーメーションにリンクできます。2 つの XML Source Qualifier トランスフォーメーショングループを、未ソート入力用の Joiner トランスフォーメーションにリンクするには、同じ XML ソースのインスタンスを 2 つ作成する必要があります。

## XML ソース修飾子トランスフォーメーションの例

ここでは、マッピング内の XML ソース修飾子トランスフォーメーションの例を説明します。この例では、商店、製品、売上の情報が含まれる XML ファイルを使用します。

以下の図に、StoreInfo.xml ファイルを示します。

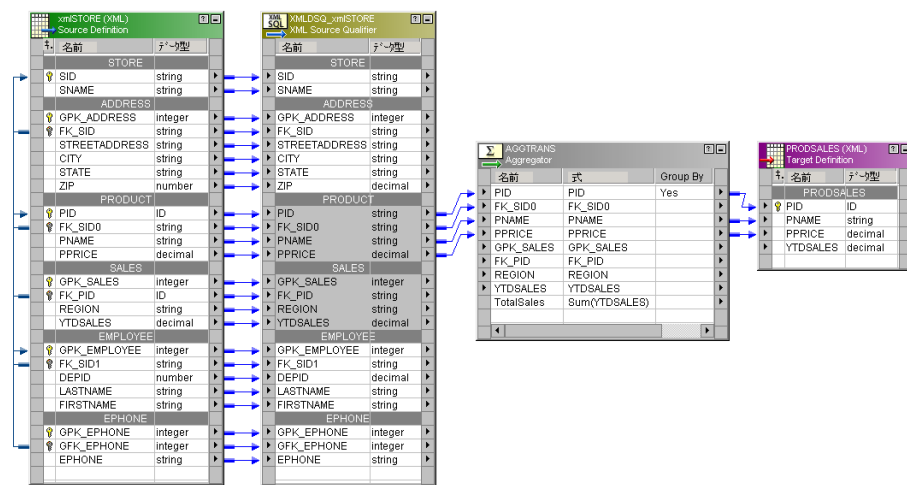


地域を限定せずに、XML ファイルの各製品の YTD 総売上を計算したい場合があります。売上のほかに、各製品の名前と価格も必要だとします。

これを行うためには、同じトランスフォーメーション内に製品と価格の情報がが必要です。ただし、StoreInfo.xml ファイルをインポートすると、Designer はデフォルトで製品情報および売上情報のグループを個別に作成します。



以下の図に、製品情報および売上情報が別々のグループに含まれている、StoreInfo ファイルのデフォルトグループを示します。



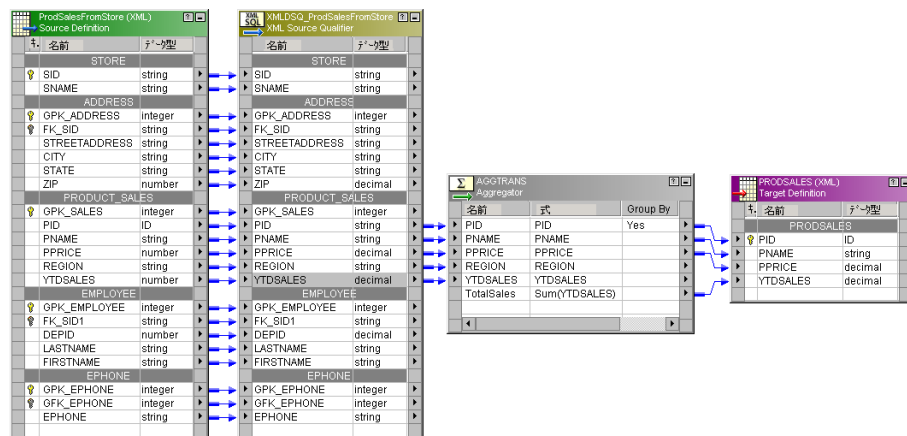
Product グループと Sales グループを同じ単一入力グループトランスフォーメーションにリンクすることはできないので、次の方法の一方でマッピングを作成することができます。

- 必要な情報をすべて含んでいる非正規化グループを使用する。
- ジョイナトランスフォーメーションを使って 2 つのグループのデータを結合する。

### 1 つの非正規化グループの使用

ソース定義のグループを再編成して、必要な情報をすべて同じグループに入れます。たとえば、Product グループと Sales グループをソース定義の 1 つの非正規化グループに統合することができます。1 つのデータフローを通じて、非正規化グループから売上集計に関するすべての情報を処理することができます。

以下の図に、Product グループの列と Sales グループの列の両方を含んでいる非正規化グループ Product\_Sales を示します。



非正規化グループを作成するには、Source Analyzer でソース定義を編集します。新しいグループを作成するか、または既存のグループを変更することができます。Aggregator トランスフォーメーションでの売上の計算を実行するには、製品および売上のカラムをグループに追加します。XML エディタを使用してグループを作成し、検証することができます。

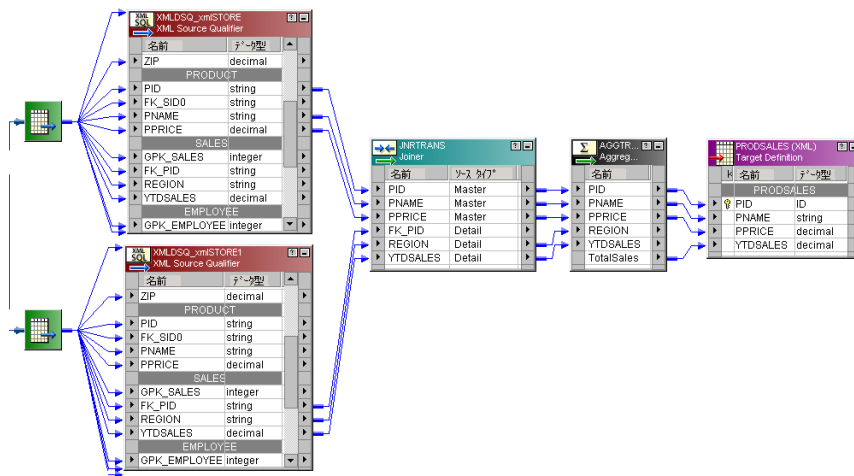


## 2つのXMLソース修飾子トランスフォーメーショングループの結合

2つのソースグループのデータを1つのデータフローに結合することができます。ジョイナトランスフォーメーションを使用して2つのグループのデータを結合します。ジョイナトランスフォーメーションをソート済み入力用に設定すると、XMLソース修飾子トランスフォーメーションの1つのインスタンスの2つのグループをJoiner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。ジョイナトランスフォーメーションを未ソート入力用に設定した場合、同じXMLソースの2つのインスタンスを使用し、XMLソース修飾子トランスフォーメーションの各インスタンスのグループをジョイナ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。

そして、ジョイナトランスフォーメーションからのデータをAggregatorトランスフォーメーションに送り、各製品のYTD Salesを計算します。

以下の図に、同じXMLソースの2つのインスタンスを作成し、2つのXMLソース修飾子トランスフォーメーションのデータを結合する方法を示します。



## XML Source Qualifier トランスフォーメーションのトラブルシューティング

XML Source Qualifier トランスフォーメーションから、2つのグループをトランスフォーメーションにドラッグしてみました。Designer はカラムをコピーしますが、すべてのポートがリンクされるわけではありません。

1つのトランスフォーメーションにリンクできるのは、XML Source Qualifier トランスフォーメーションの1つのグループです。複数のグループをトランスフォーメーションにドラッグすると、Designer はカラム名をすべてトランスフォーメーションにコピーします。ただし、Designer がリンクするのは最初のグループのカラムだけです。

XML ソース定義とそのソース修飾子との間で、リンクを解除しようとしたができませんでした。

XML Source Qualifier トランスフォーメーションのカラムは、対応するXMLソース定義のカラムと一致します。XMLソース定義とそのXML Source Qualifier トランスフォーメーションとのリンクを削除または変更することはできません。XMLソース定義を削除すると、DesignerはそのXML Source Qualifier トランスフォーメーションも削除します。

# Midstream XML トランスフォーメーション

## Midstream XML トランスフォーメーションの概要

XML 定義は、XML データの作成または読み込みを実行します。しかし、パイプラインの途中で XML の抽出や生成が必要になる場合があります。たとえば、XML 文書をデータフィールドとして含んでいる MQ ターゲットにメッセージを送りたいとします。この場合、XML 文書を生成して、その後にメッセージを MQ に送る必要があります。XML トランスフォーメーションを使用して XML を生成します。

次の種類の Midstream XML トランスフォーメーションを作成できます。

- **XML Parser トランスフォーメーション。**XML Parser トランスフォーメーションは、1 つの入力ポートから XML を読み込み、1 つ以上のグループにデータを出力します。
- **XML Generator トランスフォーメーション。**XML Generator トランスフォーメーションは、1 つ以上のソースからデータを読み込み、XML を生成します。XML Generator トランスフォーメーションには、出力ポートが 1 つあります。

Midstream XML トランスフォーメーションを使用して、TIBCO、WebSphere MQ などのメッセージングシステム、あるいはファイルやデータベースなどのその他のソースから XML データを抽出します。XML トランスフォーメーションの機能は、Midstream XML トランスフォーメーションがパイプラインの途中で XML の解析や文書の生成を行う点を除いて、XML ソースおよび XML ターゲットの機能と同様です。

Midstream XML トランスフォーメーションは、XML ウィザードおよび XML エディタがサポートする XML スキーマのコンポーネントと同じコンポーネントをサポートします。さらに、XML トランスフォーメーションは以下の機能をサポートします。

- **パススルーポート。**パススルーポートを使用して、非 XML データを Midstream トランスフォーメーション経由で渡します。XML スキーマ定義の一部ではありませんが、これらのフィールドを使用すると非正規化 XML グループを生成できます。これらのフィールドを、最上位の XML 要素と同じ方法で使用します。XML 定義内の最上位グループのプライマリキーとしてパススルーフィールドを使用することもできます。
- **リアルタイム処理。**Midstream XML トランスフォーメーションを使用して、メッセージングシステムから BLOB としてデータを処理します。
- **複数のパーティションのサポート。**パーティションごとに異なる XML ドキュメントを生成できます。

## XML Parser トランスフォーメーション

Integration Service は、XML Parser トランスフォーメーションを処理する場合、XML データの行を読み込み、XML を解析し、出力グループを介してデータを返します。XML Parser トランスフォーメーションはパススルーポート内に非 XML データを返します。JMS や IBM WebSphere MQ などのソースから XML のメッセージを解析できます。XML パーサートランスフォーメーションはアクティブなトランスフォーメーションです。

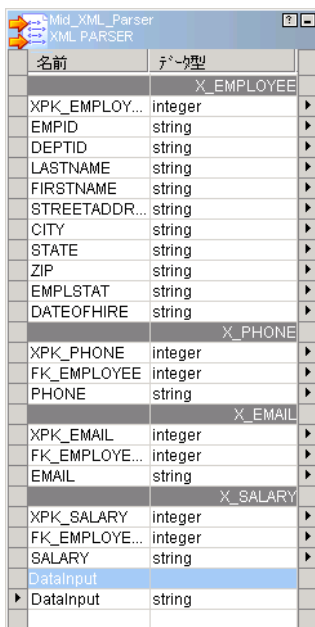
XML Parser トランスフォーメーションには 1 つの入力グループおよび 1 つ以上の出力グループがあります。入力グループには、文字列で XML ドキュメントを受け取る 1 つの入力ポート DataInput があります。

XML Parser トランスフォーメーションを作成する場合は、XML ウィザードを使用して XML、DTD、または XML スキーマファイルをインポートします。たとえば、以下の DTD ファイル Employee をインポートすることができます。

```
<!ELEMENT EMPLOYEES (EMPLOYEE+)>
<!ELEMENT EMPLOYEE (LASTNAME, FIRSTNAME, ADDRESS, PHONE+, EMAIL*, EMPLOYMENT)>
  <!ATTLIST EMPLOYEE EMPID CDATA #REQUIRED
    DEPTID CDATA #REQUIRED>
<!ELEMENT LASTNAME (#PCDATA)>
<!ELEMENT FIRSTNAME (#PCDATA)>
<!ELEMENT ADDRESS (STREETADDRESS, CITY, STATE, ZIP)>
<!ELEMENT STREETADDRESS (#PCDATA)>
<!ELEMENT CITY (#PCDATA)>
<!ELEMENT STATE (#PCDATA)>
<!ELEMENT ZIP (#PCDATA)>
<!ELEMENT PHONE (#PCDATA)>
<!ELEMENT EMAIL (#PCDATA)>
<!ELEMENT EMPLOYMENT (DATEOFHIRE, SALARY+)>
<!ATTLIST EMPLOYMENT EMPLSTAT (PF|PP|TF|TP|O) "PF">
<!ELEMENT DATEOFHIRE (#PCDATA)>
<!ELEMENT SALARY (#PCDATA)>
```

この XML パーサートランスフォーメーションは、ルートビューが X\_Employees であり、X\_Employees は X\_Employee の親になっています。X\_Employee は、X\_Salary、X\_Phone、および X\_Email の親です。

以下の図に、エンティティリレーションの作成を選択した場合に Designer が作成する XML パーサートランスフォーメーションを示します。



| 名前                | データ型    |
|-------------------|---------|
| <b>X_EMPLOYEE</b> |         |
| XPK_EMPLOY...     | integer |
| EMPID             | string  |
| DEPTID            | string  |
| LASTNAME          | string  |
| FIRSTNAME         | string  |
| STREETADDR...     | string  |
| CITY              | string  |
| STATE             | string  |
| ZIP               | string  |
| EMPLSTAT          | string  |
| DATEOFHIRE        | string  |
| <b>X_PHONE</b>    |         |
| XPK_PHONE         | integer |
| FK_EMPLOYEE       | integer |
| PHONE             | string  |
| <b>X_EMAIL</b>    |         |
| XPK_EMAIL         | integer |
| FK_EMPLOYE...     | integer |
| EMAIL             | string  |
| <b>X_SALARY</b>   |         |
| XPK_SALARY        | integer |
| FK_EMPLOYE...     | integer |
| SALARY            | string  |
| DataInput         | string  |
| DataInput         | string  |

Designer はルートビューの X\_Employees を作成します。X\_Employees は、X\_Employee の親です。X\_Employee は、X\_Salary、X\_Phone、および X\_Email の親です。

XML Parser トランスフォーメーションの各ビューは、別のビューとのリレーションを確立するためのキーを少なくとも 1 つ持っています。ユーザーが XML エディタでキーを指定しなかった場合は、Designer がビューごとにプライマリーキーと外部キーを作成します。キーのデータ型は、bigint 型です。

Integration Service が XML Parser トランスフォーメーションから行を返すたびにキー値を作成するため、キーは生成キーと呼ばれます。

Designer はプライマリキーまたは外部キーのカラムを作成する場合に、プレフィックスの付いた列名を割り当てます。XML 定義の場合、生成されたプライマリキー列には XPK\_、生成された外部キー列には XFK\_ がプレフィックスとして付けられます。外部キーは、常に他のグループのプライマリキーを参照します。生成された外部キー列は、生成されたプライマリキーカラムを必ず参照します。

たとえば、グループ X\_Employee は、XPk\_Employee プライマリキーを持ちます。Designer は、X\_Phone、X\_Email、および X\_Salary を X\_Employee グループに接続する外部キー列を作成します。各グループは、外部キー列 XFK\_Employee を持ちます。

リポジトリはキー値を格納します。リポジトリの値を変更することはできませんが、セッション後のシーケンス番号のリセットまたはリスタートを選択することができます。

## XML Parser 入力の検証

XML Parser トランスフォーメーションを設定して、XML を解析する前に検証できます。XML Parser トランスフォーメーションは、スキーマに対して XML を検証します。XML がスキーマに対して有効ではない場合、行エラーが発生します。XML Parser トランスフォーメーションは、XML メッセージと関連するエラーメッセージを個別の出力グループに返します。無効な XML とエラーメッセージターゲットに渡すことができます。

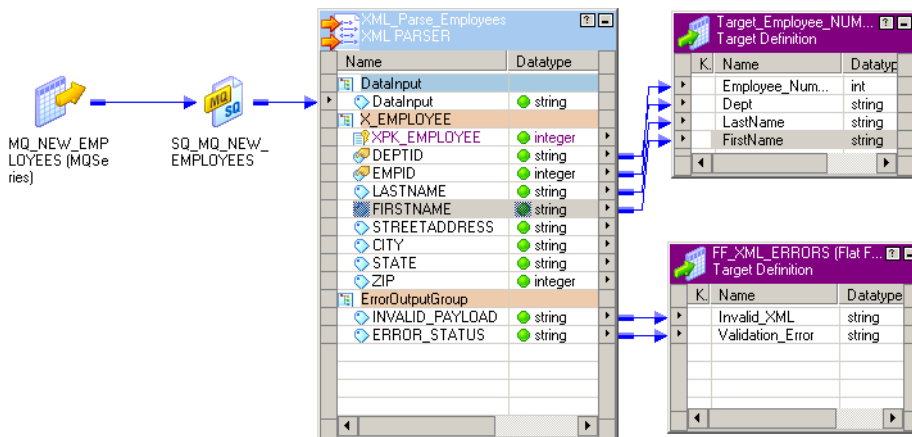
たとえば、リアルタイム PowerCenter セッションは、XML メッセージを WebSphere MQSeries ソースから読み込みます。セッションはソースベースのコミットで実行されます。コミットトランザクションのメッセージには、無効な XML ペイロードが含まれます。コミットの失敗を防止するには、有効なデータから個別の出力グループに無効な XML を返すように、XML Parser トランスフォーメーションを設定できます。XML Parser トランスフォーメーションは、有効な XML メッセージを処理してトランザクションを完了します。

セッションログには、[データフローオプションによって無効なペイロードのルーティング] を有効にするタイミングを示すメッセージが含まれます。セッションのトレースレベルをノーマルに設定すると、Integration Service は検証が正常終了したかどうかを示すセッションログにメッセージを書き込みます。このログメッセージには、XML Parser が XML を検証するためにアクセスしたスキーマの場所が含まれます。XML のストリーミングが有効で、XML が無効である場合、Integration Service は無効な XML を切り詰めて Invalid\_Payload ポートに渡します。Integration Service は、セッションログに無効な XML のログを記録します。

XML を検証するように XML Parser トランスフォーメーションを設定するには、[Midstream XML Parser] タブの [データフローオプションによって無効なペイロードのルーティング] オプションを有効にします。Designer は以下のポートを XML Parser トランスフォーメーションに追加します。

- **Invalid\_Payload.** 無効な XML メッセージをパイプラインに返します。XML ペイロードが有効な場合は、Invalid\_Payload ポートには NULL 値が含まれます。このポートの精度は、DataInput ポートの精度と同じです。
- **Error\_Status.** XML 検証から返されたエラー文字列またはステータスが含まれます。XML がカレント行に対して有効な場合は、Error\_Status には NULL 値が含まれます。このポートの精度は、DataInput ポートの精度と同じです。

次の図に、無効な XML メッセージをエラーターゲットテーブルにルーティングする XML パーサートランスフォーメーションを示します。



マッピングには以下のオブジェクトが含まれます。

- **MQSeries ソース定義。**メッセージデータフィールドに従業員の XML データが含まれます。
- **Source Qualifier トランスフォーメーション。**WebSphere MQ からデータを読み取ります。メッセージヘッダフィールドとメッセージデータフィールドを表すポートのセットが含まれます。
- **XML Parser トランスフォーメーション。**XML メッセージデータを DataInput ポートで受け取ります。XML が有効な場合、XML Parser トランスフォーメーションは従業員データを返してターゲットに渡します。XML が無効な場合、XML Parser トランスフォーメーションは Invalid\_Payload ポートに XML を返します。エラーメッセージは、Error\_Status ポートに返されます。
- **従業員ターゲット定義。**有効な従業員データの行を受け取ります。
- **XML\_Errors ターゲット定義。**無効な XML とエラーメッセージを受け取ります。

トランスフォーメーションのセッションプロパティに XML スキーマの場所属性を設定します。スキーマの名前と場所を入力して、XML を検証します。XML スキーマ定義に対して、ワークフロー、セッション、またはマッピング変数とパラメータを設定できます。複数のスキーマをセミコロンで区切ると、検証において複数のスキーマを設定できます。

DTD を入力 XML ペイロードに含めると、検証において DTD を使用できます。XML スキーマの場所属性内に DTD を設定することはできません。また、DTD を使用して無効な XML データを無効なペイロードポートにルーティングすることはできません。

XML のストリーミングを有効にする場合、Invalid\_Payload ポートの精度がメッセージの最大サイズと一致することを確認します。ポートの精度がメッセージのサイズより低い場合、XML Parser トランスフォーメーションは Invalid\_Payload ポートに切り詰めた XML を返して、セッションログにエラーを書き込みます。

## XML Parser トランスフォーメーションへの XML のストリーム

Unstructured Data transformation、JMS ソース、または WebSphere MQ ソースから XML Parser トランスフォーメーションへ XML をストリーミングするようにセッションを設定できます。PowerCenter 統合サービスでは、XML データをストリーミングするときに、XML データを複数のセグメントに分割します。



XML Parser トランスフォーメーションにより小さい入力ポートを構成して、XML Parser トランスフォーメーションで大きな XML ファイルを処理する場合に必要なメモリ量を削減します。100MB を超える XML ファイルを解析することができます。

XML のストリーミングを有効にした場合、XML Parser トランスフォーメーションは、ポートサイズ以下のセグメントでデータを受け取ります。XML ファイルがポートサイズより大きい場合、PowerCenter 統合サービスは複数行を XML パーサートランスフォーメーションに渡します。XML の各行には、ストリーミングの行タイプが含まれます。最後の行には、挿入の行タイプが含まれます。

入力ポートの精度は、XML を XML パーサートランスフォーメーションに渡すオブジェクトの出力ポートの精度以上となる必要があります。ほとんどの XML ドキュメントのサイズが小さくても、サイズの大きいメッセージがある場合、最高のパフォーマンスを得るためには、XML パーサートランスフォーメーションのポートサイズをより小さなメッセージのサイズに設定します。

XML のストリーミングを有効にする場合、XML データを XML パーサートランスフォーメーションに渡すソースまたはトランスフォーメーションに対して XML のストリーミングを有効にすることも必要です。ストリーミングを有効にしない場合、XML Parser は 1 行で XML を受け取るため、パフォーマンスが低下することがあります。

XML Parser トランスフォーメーションで XML のストリーミングを有効にするには、XML Parser トランスフォーメーションのセッションプロパティで [XML 入力ストリーミングの有効化] を選択します。ソースまたはトランスフォーメーションで XML ストリーミングを有効にし、XML Parser トランスフォーメーションに対しては有効にしない場合、XML Parser トランスフォーメーションは XML ファイルを処理できません。

XML のストリーミングを有効にして XML ドキュメントでエラーが発生する場合、PowerCenter 統合サービスはデフォルトで XML ドキュメントにセッションログを書き込みます。エラーの発生時に XML ドキュメントをエラーログファイルに書き込むように、セッションを設定できます。

ログソース行データをセッションのプロパティで有効にします。ロギングを有効にして XML ドキュメントでエラーが発生する場合、PowerCenter 統合サービスは行エラーを生成します。PowerCenter 統合サービスは XML ドキュメントをエラーログファイルに書き込み、エラーカウントを増やします。

PowerCenter での XML サイズ設定の詳細については、『[「PowerCenter での XML の使用の概要」 \(ページ 29\)](#)』を参照してください。PowerCenter で XML 処理に適用される制限の詳細については、『[「制限」 \(ページ 29\)](#)』を参照してください。

他の要素型を持つトランスフォーメーションを作成し、大きな XML 入力ファイルを変換するには、データプロセッサトランスフォーメーションを使用します。データプロセッサトランスフォーメーションの作成方法の詳細については、『*Informatica Data Transformation ユーザーガイド*』および『*Informatica Data Transformation 入門ガイド*』を参照してください。

## XML 10 進データ型

XML 10 進要素の精度を 34 桁より多く定義した場合は、Integration Service によって外部関数が呼び出され、XML Parser トランスフォーメーション内で XML 10 進データ型が Double に変換されます。関数は、Integration Service を実行しているノードに応じた精度長で Double を返します。すべてのプラットフォームで、数値が丸められる前の精度は 17 桁に保証されていますが、プラットフォームによってはより精度が高い場合もあります。

たとえば、Windows 32 ビット版では、Integration Service によって 17 桁より後の数値が丸められます。

1234.567890123456789 is converted to 1234.567890123460800.

HP-UX 32 ビット版では、Integration Service によって 34 桁より後の数値が丸められます。

## XML Generator トランスフォーメーション

XML Generator トランスフォーメーションを使用して、複数のソースからの入力を結合すると、XML 文書を作成できます。たとえば、トランスフォーメーションを使用して、2 つの TIBCO ソースからの XML データを 1 つの TIBCO ターゲットに結合できます。1 つのソースには、従業員および賃金の情報が含まれ、もう一方には従業員の電話番号およびメールの情報が含まれることがあります。XML ジェネレータトランスフォーメーションはアクティブで、接続されたトランスフォーメーションです。

XML ジェネレータトランスフォーメーションは XML ターゲット定義に似ています。Integration Service は、XML Generator トランスフォーメーションを処理する場合、XML データの行を書き込みます。Integration Service は、トランスフォーメーションで非 XML データを含むパススルーフィールドを処理することもできます。

XML Generator トランスフォーメーションには 1 つ以上の入力グループがあり、1 つの出力グループがあります。出力グループには、BLOB 文字列データの XML 文書を生成する 1 つのポート「DataOutput」があります。出力グループには、パススルーフィールドを作成するためのパススルーポートが含まれます。

## Midstream XML トランスフォーメーションの作成

Midstream XML トランスフォーメーションを作成する場合、XML ウィザードおよび XML エディタを使用して XML グループを定義します。トランスフォーメーションは、Transformation Developer および Mapping Designer で作成できます。

Midstream XML トランスフォーメーションを作成するには：

1. Transformation Developer または Mapping Designer を開きます。
2. [トランスフォーメーション] - [作成] をクリックします。  
[トランスフォーメーションの作成] ダイアログボックスが表示されます。
3. XML Parser トランスフォーメーションまたは XML Generator トランスフォーメーションのいずれかのタイプを選択します。
4. トランスフォーメーション名を入力して、[作成] をクリックします。  
[XML 定義のインポート] ダイアログボックスが表示されます。
5. ファイルを選択して、[開く] をクリックします。  
XML ウィザードが表示されます。
6. XML ウィザードを使用して XML 定義を作成します。
7. XML ウィザードの [完了] をクリックします。  
ワークスペースに Midstream XML トランスフォーメーションが表示されます。
8. Midstream XML トランスフォーメーションのプロパティを編集するには、ワークスペース内のトランスフォーメーションをダブルクリックします。

## Midstream XML 定義の同期

Midstream XML トランスフォーメーションは、異なるバージョンのスキーマか、トランスフォーメーションを作成するためにインポートしたソースファイルを使用して同期することができます。たとえば、XML Parser トランスフォーメーションを作成するためにインポートしたスキーマファイルに新規要素を



追加することがあります。XML Parser トランスフォーメーションを削除して再度作成しなくても、新しいスキーマを使用すれば XML Parser トランスフォーメーションを更新できます。

Midstream XML トランスフォーメーションを同期させるには、Transformation Developer または Mapping Designer を使用します。

Midstream XML トランスフォーメーションを同期させるには：

1. Transformation Developer または Mapping Developer を開きます。
2. Midstream XML トランスフォーメーションまたは更新したいマッピングをワークスペースにドラッグします。
3. Midstream XML トランスフォーメーションの上部を右クリックします。
4. [XML トランスフォーメーションの同期] を選択します。  
[XML 定義のインポート] ダイアログボックスが表示されます。
5. XML Parser トランスフォーメーションまたは XML Generator トランスフォーメーションの作成に使用したリポジトリ定義またはファイルを見つけます。
6. [開く] をクリックして、トランスフォーメーションを更新します。

Designer は、トランスフォーメーションを作成する際に使用しなかったファイルとトランスフォーメーションを同期させることはできません。

Source Analyzer または Target Designer を使用して、ソースとターゲット XML 定義を同期させてください。

## Midstream XML トランスフォーメーションのプロパティの編集

Midstream XML トランスフォーメーションのいくつかのプロパティを編集することができます。ただし、XML ウィザードおよび XML エディタを使用してトランスフォーメーションを定義するため、これらのツールを使用して XML 定義を変更しなければなりません。

Mapping Designer で Midstream XML トランスフォーメーションを作成する場合、以下のルールが適用されます。

- トランスフォーメーションを再利用可能にする場合、トランスフォーメーションの一部のプロパティを Mapping Designer から変更できます。パススルーポートまたはメタデータエクステンションを追加することはできません。
- 再利用不可能なトランスフォーメーションを作成する場合、Mapping Designer からトランスフォーメーションを編集することができます。

Midstream XML トランスフォーメーションを設定する場合、以下のタブでコンポーネントを設定することができます。

- **[トランスフォーメーション] タブ。** [トランスフォーメーション] タブでは、トランスフォーメーションの名前の変更、および説明の追加を行います。
- **[ポート] タブ。** [XML Parser] タブまたは [XML Generator] タブで作成するトランスフォーメーションのポートおよび属性を表示します。
- **[プロパティ] タブ。** トレースレベルを更新します。
- **[初期化プロパティ] タブ。** 初期化中に外部プロシージャが使用するランタイムプロパティを作成します。

- **【メタデータエクステンション】 タブ。**XML トランスフォーメーション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により、リポジトリに格納されているメタデータを拡張します。
- **【ポート属性定義】 タブ。**トランスフォーメーションのすべてのポートに適用されるポート属性を定義します。
- **【Midstream XML Parser】 タブまたは【XML Generator】 タブ。**このタブを使用するとパススルーポートを作成できます。パススルーポートを使用すれば、非 XML データをトランスフォーメーション経由で渡すことができます。XML Parser トランスフォーメーションで、シーケンスナンバリングを使用して XML カラム名を生成する場合には、シーケンス番号のリセットを選択できます。XML Generator トランスフォーメーションの場合、コミットでの新しい XML 文書の作成を選択できます。

## 【プロパティ】 タブ

【プロパティ】 タブでは、Midstream XML トランスフォーメーションのプロパティを設定します。

以下の表に、【プロパティ】 タブで変更できるオプションを示します。

| トランスフォーメーション   | 説明  |
|----------------|---|
| 実行時位置          | <p>DLL または共有ライブラリの格納場所。デフォルトは \$PMExtProcDir です。XML セッションを実行する Integration Service ノードへの相対パスを入力します。</p> <p>このプロパティが空白の場合、Integration Service は、Integration Service ノードで定義されている環境変数を使用して DLL または共有ライブラリの位置を探します。</p> <p>Integration Service ノードで定義されている実行時位置または環境変数に、すべての DLL または共有ライブラリをコピーする必要があります。DLL、共有ライブラリ、または参照されるファイルが見つからない場合、Integration Service は手続きのロードに失敗します。</p>  |
| トレースレベル        | トランスフォーメーションのセッションログに表示される情報の詳細度。デフォルトは [Normal] です。  |
| トランスフォーメーション範囲 | <p>Integration Service が入力データにトランスフォーメーションロジックを適用する方法を示します。XML Parser トランスフォーメーションに対して、以下のトランスフォーメーション範囲の値のいずれかを選択できます。</p> <ul style="list-style-type: none"> <li>- 行。トランスフォーメーションロジックを、データの 1 つの行ごとに適用します。次の行を処理する前に、すべての出力グループで生成された行をフラッシュします。</li> <li>- Transaction。トランスフォーメーションロジックをトランザクションのすべての行に適用します。トランザクション境界で生成された行をフラッシュします。出力ブロックが満たされると、ファイルの最後でフラッシュします。</li> <li>- All Input。トランスフォーメーションロジックをすべての入力データに適用します。出力ブロックが満たされるときのみ、ファイルの最後で生成された行をフラッシュします。</li> </ul> <p>XML Generator トランスフォーメーションに対して、[コミット時] 設定を [コミットを無視する] に設定した場合、Designer はトランスフォーメーション範囲をすべての入力に設定します。[コミット時] を [新規ドキュメントを作成する] に設定した場合、Designer はトランスフォーメーション範囲をトランザクションレベルに設定します。</p> |

| トランスフォーメーション                         | 説明  |
|--------------------------------------|---|
| Output is Repeatable                 | <p>出力データの順序をセッションの実行ごとに一致させるかどうかを指定します。</p> <ul style="list-style-type: none"> <li>- Never。出力データの順序はセッションの実行ごとに異なります。</li> <li>- Based On Input Order。入力データの順序がセッションの実行ごとに一致している場合、出力順序をセッションの実行ごとに一致させます。</li> <li>- Always。入力データの順序がセッションの実行ごとに異なる場合でも、出力データの順序は常に同じです。</li> </ul> <p>XML Parser トランスフォーメーションの場合、デフォルトは [Based On Input Order(入力順による)] です。XML Generator トランスフォーメーションの場合、デフォルトは [Always(常に)] です。</p> |
| Requires Single Thread per Partition | Integration Service が各パーティションを 1 つのスレッドで処理するかどうかを示します。  |
| Output is Deterministic              | トランスフォーメーションはセッション実行ごとに同じ出力データを生成するかどうかを示します。このトランスフォーメーションを使用するセッションでリカバリを実行するには、このプロパティを有効にする必要があります。デフォルトでは有効になっています。  |

**警告:** トランスフォーメーションを繰り返し可能で一意に定まるものとして設定する場合は、データが繰り返し可能で一意に定まることを保証する必要があります。セッションとリカバリで同じデータが生成されないトランスフォーメーションを使用してセッションをリカバリしようとする、リカバリプロセスを実行した結果、データが破損する可能性があります。

## [Midstream XML Parser] タブ

[Midstream XML Parser] タブを使用して、DataInput ポートのサイズを変更します。このタブで、パススルーポートを追加することもできます。

[Midstream XML Parser] タブから XML エディタにアクセスすることができます。[XML エディタ] をクリックします。

**注:** XML エディタにアクセスする場合、XML エディタを終了するまで [トランスフォーメーションの編集] を更新することはできません。

以下の表に、[Midstream XML Parser] タブで変更できるオプションを示します。

| トランスフォーメーション | 説明   |
|--------------|--|
| 精度           | カラムの長さ。DataInput ポート精度のデフォルトは、64K です。パススルーポートのデフォルトの精度は 20 です。精度を上げることもできます。 |
| リスタート        | 生成キーシーケンスは、常に 1 から開始します。セッションを実行する際は、XML 定義にある全グループのキーシーケンス値を 1 から開始します。     |
| リセット         | セッションの最後に、全グループのすべての生成キーの値シーケンスをリセットします。シーケンス番号をセッション前の番号にリセットします。           |

| トランスフォーメーション                   | 説明  |
|--------------------------------|---|
| データフローオプションによって無効なペイロードのルーティング | XML をスキーマに対して検証します。XML がスキーマに対して有効ではない場合、行エラーが発生します。XML Parser トランスフォーメーションは、XML メッセージと関連するエラーメッセージを個別の出力グループに返します。 |
| 説明                             | トランスフォーメーションの説明です。  |

**注:** [リセット] または [リスタート] を選択しない場合、生成されたキーのシーケンス番号はセッションごとに大きくなります。[リスタート] または [リセット] オプションを選択すると、[初期化プロパティ] タブに表示される [リスタート] または [リセット] プロパティが更新されます。ただし、これらのオプションを [初期化プロパティ] タブで変更することはできません。

## [Midstream XML Generator] タブ

[Midstream XML Generator] タブを使用して、DataOutput ポートのサイズを変更します。このタブで、パススルーポートを追加することもできます。

[Midstream XML Generator] タブから XML エディタにアクセスすることができます。[XML エディタ] をクリックします。XML エディタにアクセスする場合、XML エディタを終了するまでトランスフォーメーションプロパティを編集することはできません。

以下の表に、[XML Generator トランスフォーメーション] タブで変更できるオプションを示します。

| トランスフォーメーションの設定 | 説明  |
|-----------------|---|
| 精度              | カラムの長さ。DataOutput ポート精度のデフォルトは、64K です。パススルーポートのデフォルトの精度は 20 です。精度を上げることもできます。   |
| コミット時           | Integration Service は、コミット後に複数の XML ドキュメントを生成できます。次のいずれかのオプションを使用します。 <ul style="list-style-type: none"> <li>- コミットを無視する。Integration Service は、XML ドキュメントを作成し、ファイルの最後で XML ドキュメントにデータを書き込みます。2 つの異なるソースが XML Generator トランスフォーメーションに接続されている場合、このオプションを使用します。</li> <li>- 新規ドキュメントを作成する。コミットごとに新しい XML 文書を作成します。リアルタイムセッションを実行する場合、このオプションを使用します。</li> </ul> セッションで複数のパーティションが使用されている場合、Integration Service は、[コミット時] 設定に関係なく、各パーティションに対して個別の XML ドキュメントを生成します。[新規ドキュメントを作成する] を選択すると、Integration Service は、各パーティションに対して新規ドキュメントを作成します。 |
| 説明              | トランスフォーメーションの説明です。  |

**注:** [コミット時] 設定を [コミットを無視する] に設定した場合、Designer はトランスフォーメーション範囲を [All Input] に設定します。[コミット時] を [新規ドキュメントを作成する] に設定した場合、Designer はトランスフォーメーション範囲を [Transaction] レベルに設定します。

## パススルーポートの生成

パススルーポートは、Midstream XML トランスフォーメーションを介して非 XML データを渡すカラムです。たとえば、MQSeries のソースおよびターゲットの XML と一緒にメッセージ ID を渡すことができます。メッセージ ID を使用して、入力メッセージおよび出力メッセージを要求および応答と関連付けることができます。

Midstream トランスフォーメーションのパススルーポートを定義する場合、ポートを XML Parser トランスフォーメーションの DataInput グループ、または XML Generator トランスフォーメーションの DataOutput グループに追加します。

ポート作成後、XML エディタを使用して、XML 定義の別のビューに参照ポートを追加します。XML Parser トランスフォーメーションの場合、パススルーポートは入力ポートであり、対応する参照ポートは出力ポートです。XML Generator トランスフォーメーションの場合、パススルーポートは出力ポートであり、関連付けられた参照ポートは入力ポートです。

Midstream XML トランスフォーメーションのパススルーポートを作成するには：

1. Transformation Developer または Mapping Designer でトランスフォーメーションを開きます。
2. トランスフォーメーションをダブルクリックして、[トランスフォーメーションの編集] を開きます。
3. [Midstream XML Generator] タブまたは [Midstream XML Parser] タブをクリックします。  
トランスフォーメーションのタイプに応じて、DataInput ポートまたは DataOutput ポートが表示されます。
4. [追加] をクリックして、パススルーポートの出力ポートを追加します。  
[フィールド名] カラムにデフォルトのフィールドが表示されます。
5. フィールド名を修正します。定義を作成するのに使用したファイルに応じて、型、精度、位取りも変更することができます。
6. [XML エディタ] をクリックして、トランスフォーメーションの XML 定義を開きます。  
定義内の XML ビューがワークスペースに表示されます。
7. ビューの最上部を右クリックして、参照ポートを追加します。
8. [参照ポートの追加] をクリックして、参照ポートを追加します。  
[追加したい参照ポートを選択してください。] ダイアログボックスが表示されます。  
トランスフォーメーションに追加したパススルーポートがダイアログボックスに一覧表示されます。
9. ビューの新規参照ポートに対応するパススルーポートを選択して、[OK] をクリックします。  
対応する出力参照ポートがビューに表示されます。Columns ウィンドウで、ポートの名前をわかりやすいものに変更できます。
10. [ファイル] - [変更の適用] を選択して、XML エディタを終了します。
11. トランスフォーメーションで [OK] をクリックします。

非 XML データは、入力ポート「Pass\_thru\_fields」を介して入力され、対応する参照出力ポート「COL\_0」を介して渡されます。

## Midstream XML トランスフォーメーションのトラブルシューティング

XML CLOB を含むデータベーステーブルから XML ファイルを抽出する必要があります。XML ファイルのサイズは、最大で 2 GB です。XML Parser トランスフォーメーションを作成する場合、CLOB カラムに対して最大固定長を定義する必要があります。しかし、CLOB データ型の最大長は 104 MB です。

その XML データは大きすぎるため、テーブルから XML Parser トランスフォーメーションに直接渡すことができません。CLOB テーブルデータをフラットファイルに用意し、そのファイルから XML ソース定義を作成する必要があります。

PowerCenter での XML サイズ設定の詳細については、『[「PowerCenter での XML の使用の概要」 \(ページ 29\)](#)』を参照してください。PowerCenter で XML 処理に適用される制限の詳細については、『[「制限」 \(ページ 29\)](#)』を参照してください。

他の要素型を持つトランスフォーメーションを作成し、大きな XML 入力ファイルを変換するには、データプロセッサトランスフォーメーションを使用します。データプロセッサトランスフォーメーションの作成方法の詳細については、『*Informatica Data Transformation ユーザーガイド*』および『*Informatica Data Transformation 入門ガイド*』を参照してください。

## XML データ型リファレンス

### XML データ型とトランスフォーメーションデータ型

PowerCenter では、W3C May 2, 2001 Recommendation で指定されている XML データ型をすべてサポートしています。以下の表は、XML データ型を表示し、XML データ型と XML Source Qualifier 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データ型とを比較したものです。XML データ型の詳細については、<http://www.w3.org/TR/xmlschema-2> で XML データ型の W3C 仕様を参照してください。

XML ファイルをインポートして定義を作成する場合は、XML 定義でのデータ型および Midstream XML トランスフォーメーションでのデータ型を変更できます。XML スキーマからインポートする場合は、XML データ型を変更できません。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型を変更することはできません。

以下の表に、XML データ型および対応するトランスフォーメーションデータ型を示します。

| データ型          | トランスフォーメーション  | 範囲                  |
|---------------|---------------|---------------------|
| anySimpleType | String        | 1 - 104,857,600 文字  |
| anyURI        | String        | 1 - 104,857,600 文字  |
| base64Binary  | Binary        | 1 - 104,857,600 バイト |
| boolean       | Small Integer | 精度 5、位取り 0          |
| byte          | Small Integer | 精度 5、位取り 0          |



| データ型       | トランスフォーマー<br>ション | 範囲   |
|------------|------------------|--|
| date       | Date/Time        | 西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 (精度<br>はナノ秒まで)                 |
| dateTime   | Date/Time        | 西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 (精度<br>はナノ秒まで)                 |
| decimal    | Decimal          | 精度 1 - 28、位取り 0 - 28   |
| double     | Double           | 精度 15、位取り 0  |
| duration   | String           | 1 - 104,857,600 文字   |
| ENTITIES   | String           | 1 - 104,857,600 文字   |
| ENTITY     | String           | 1 - 104,857,600 文字   |
| float      | Double           | 精度 15、位取り 0  |
| gDay       | Integer          | -2,147,483,648 - 2,147,483,647<br>精度 10、位取り 0                          |
| gMonth     | Integer          | -2,147,483,648 - 2,147,483,647<br>精度 10、位取り 0                          |
| gMonthDay  | Date/Time        | 西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 (精度<br>はナノ秒まで)                 |
| gYear      | Integer          | 精度 10、位取り 0  |
| gYearMonth | Date/Time        | 西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日 (精度<br>はナノ秒まで)                 |
| hexBinary  | Binary           | 1 - 104,857,600 バイト  |
| ID         | String           | 1 - 104,857,600 文字   |
| IDREF      | String           | 1 - 104,857,600 文字   |
| IDREFS     | String           | 1 - 104,857,600 文字   |
| int        | Bigint           | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807<br>精度 19、位取り 0 |
| integer    | Bigint           | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807<br>精度 19、位取り 0 |
| language   | String           | 1 - 104,857,600 文字   |
| long       | Bigint           | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807<br>精度 19、位取り 0 |
| Name       | String           | 1 - 104,857,600 文字   |



| データ型               | トランスフォーマー<br>ション | 範囲   |
|--------------------|------------------|--|
| Ncname             | String           | 1 - 104,857,600 文字   |
| negativeInteger    | Bigint           | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807<br>精度 19、位取り 0 |
| NMTOKEN            | String           | 1 - 104,857,600 文字   |
| NMTOKENS           | String           | 1 - 104,857,600 文字   |
| nonNegativeInteger | Bigint           | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807<br>精度 19、位取り 0 |
| nonPositiveInteger | Bigint           | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807<br>精度 19、位取り 0 |
| normalizedString   | String           | 1 - 104,857,600 文字   |
| NOTATION           | String           | 1 - 104,857,600 文字   |
| positiveInteger    | Bigint           | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807<br>精度 19、位取り 0 |
| QName              | String           | 1 - 104,857,600 文字   |
| short              | Small Integer    | 精度 5、位取り 0   |
| string             | String           | 1 - 104,857,600 文字   |
| time               | Date/Time        | 西暦 0001 年 1 月 1 日 - 西暦 9999 年 12 月 31 日（精度<br>はナノ秒まで）                  |
| token              | String           | 1 - 104,857,600 文字   |
| unsignedByte       | Small Integer    | 精度 5、位取り 0   |
| unsignedInt        | Bigint           | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807<br>精度 19、位取り 0 |
| unsignedLong       | Bigint           | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807<br>精度 19、位取り 0 |
| unsignedShort      | Integer          | -2,147,483,648 - 2,147,483,647<br>精度 10、位取り 0                          |

## XML の日付形式

PowerCenter は、以下の日付、時刻、日時 of データ型の形式をサポートします。

CCYY-MM-DDThh:mm:ss:sss

このフォーマットをそのまま使用することも、一部だけを XML ファイルで使用することもできます。  
PowerCenter は、Datetime フォーマットの負の日付をサポートしません。

セッション内の以下のフォーマットで、Date、Time、Datetime 要素を使用します。

CCYY-MM

-または-

CCYY-MM-DD/Thh

XML ファイルに出現する最初の日付/時刻要素のフォーマットによって、以降の要素の値のすべてのフォーマットが決定されます。Integration Service が異なる形式を使用した同じ日付、時刻、日時要素の値を読み込んだ場合、Integration Service はその行を拒否します。

たとえば、日付要素の最初の値が次のフォーマットを使用したとします。

CCYY-MM-DDThh:mm:ss

Integration Service は、次の形式の要素を含む行を拒否します。

CCYY-MM-DD

XML パーサは、入力 XML の日時の値を、Integration Service をホストするマシンがある現地のタイムゾーンの値に変換します。Windows で、夏時間を適用して時計を調整するオプションを有効にすると、XML パーサーが日時の値に 1 時間を追加します。一貫して日時の値を変換するために、Windows で、夏時間を適用して時計を調整するオプションを有効にしないでください。

## XPath クエリ関数の参照

### XPath クエリ関数の概要

XPath は、XML 文書内の項目の位置を特定する方法を記述する言語です。XPath は、ルートコンポーネントからの XML 階層のパスに基づくアドレス構文を使用します。ビュー行の要素に対して XPath クエリーの述部を作成したり、ビュー行を含む XPath のあるカラムを作成したりすることができます。

XML ビューで XPath クエリーの述部を使用し、XML ソースデータをフィルタすることができます。セッション中に、Integration Service はクエリに基づいて、ソース XML ファイルからデータを抽出します。すべてのクエリーが TRUE を返した場合、Integration Service はビューからデータを抽出します。

XPath クエリーの述部には抽出する要素または属性が含まれており、クエリーが基準を決定します。要素または属性の値の確認や、ソース XML データに要素または属性が存在することの確認を行うことができます。

この付録では、XPath クエリーの述部で使用される各関数について説明します。関数は引数を取り、値を返します。関数を作成する場合、XML ビューの要素や属性のコンポーネントを含めることができ、リテラル値を追加できます。リテラルを指定する場合、一重引用符または二重引用符でリテラルを囲んでください。

## 関数のクイックリファレンス

XPath クエリの述部には、以下のタイプの関数を使用します。

- **文字列**。サブストリングの値のテスト、文字列の連結、または、文字列の他の文字列への変換には、文字列関数を使用します。たとえば、次の XPath クエリの述部は従業員のフルネームが姓と名前（ファーストネーム）の結合と等しいかどうかを判断します。

```
EMPLOYEE[./FULLNAME=concat(./ENAME/LASTNAME,./ENAME/FIRSTNAME)]
```

- **数値**。要素および属性の値には数値関数を使用します。数値関数は、数値の演算を行い、整数を返します。たとえば、次の XPath クエリーの述部は、割引額を丸め、結果が 15 より大きいかどうかを確認します。

```
ORDER_ITEMS[round(./DISCOUNT) > 15]
```

- **論理型**。論理関数を使用して、要素のテスト、言語属性の確認、または強制的に True または False の結果を返すことができます。たとえば、次の XPath クエリの述部は値が 0 よりも大きい場合は True を返します。

```
boolean(string)
```

以下の表に、XPath クエリの述部の文字列関数を示します。

| 機能               | 構文   | 説明                                  |
|------------------|--|-------------------------------------|
| concat           | concat ( string1, string2 )                | 2 つの文字列を連結します。                      |
| contains         | contains( string, substring )              | 文字列に他の文字列が含まれているかどうかを確認します。         |
| normalize-space  | normalize-space ( string )                 | 文字列から先頭や末尾の空白を除去します。                |
| starts-with      | starts-with ( string, substring )          | string1 が string2 から開始するかどうかを確認します。 |
| string           | string ( value )                           | 数値または論理値を文字列に変換します。                 |
| string-length    | string-length ( string )                   | 文字列内の文字数を返します。文字列の末尾の空白も含めます。       |
| substring        | substring ( string, start<br>[,length] )   | 文字列内の指定した位置以降の部分を変換します。             |
| substring-after  | substring-after ( string,<br>substring )   | 文字列内の指定した位置以降の部分を変換します。             |
| substring-before | substring-before ( string,<br>substring )  | サブストリングの前に出現する文字列に対して、文字を返します。      |
| translate        | translate ( string1, string2,<br>string3 ) | 文字列内の文字を他の文字に変換します。                 |

以下の表に、XPath クエリの述部の数値関数を示します。

| 機能      | 構文                 | 説明                       |
|---------|--------------------|--------------------------|
| ceiling | ceiling ( number ) | 数値を、渡された数値以上の最小の整数に丸めます。 |
| floor   | floor( number )    | 数値を、渡された数値以下の最大の整数に丸めます。 |
| number  | number ( value )   | 文字列または論理値を数値に変換します。      |
| round   | round ( number )   | 数値を最も近い整数に丸めます。          |

以下の表に、XPath クエリの述部の論理関数を示します。

| 機能      | 構文                 | 説明   |
|---------|--------------------|--|
| boolean | boolean ( object ) | オブジェクトを論理値に変換します。  |
| false   | false()            | 常に FALSE を返します。  |
| lang    | lang ( code )      | 要素に対して、コードの引数と一致する <code>xml:lang</code> 属性を持っているかどうかを確認します。 |
| not     | not ( condition )  | 論理条件が FALSE の場合は TRUE を返し、TRUE の場合は FALSE を返します。             |
| true    | true()             | 常に TRUE を返します。   |

## 論理

値を論理型に変換します。

### 構文

`boolean ( object )`

以下の表に、論理型の引数を示します。

| 引数            | 説明                                  |
|---------------|-------------------------------------|
| <i>object</i> | 数値データ型または文字列データ型。テストに数値または文字列を渡します。 |

### 戻り値

論理値。

次のとおり、関数は論理値を返します。

- 文字列は長さが 0 でない場合に TRUE を返し、それ以外の場合は FALSE を返します。
- 数値は 0 または `number(NaN)` でない場合に FALSE を返し、それ以外の場合は TRUE を返します。

## 例

次の例では、名前に文字が含まれることを確認します。

`boolean ( NAME )`

以下の表に、引数および戻り値の例を示します。

| NAME  | RETURN VALUE |
|-------|--------------|
| Lilah | TRUE         |
| -     | FALSE        |

次の例では、ZIP CODE が数字であることを確認します。

`boolean ( ZIP_CODE )`

以下の表に、引数および戻り値の例を示します。

| ZIP_CODE | RETURN VALUE |
|----------|--------------|
| 94061    | TRUE         |
| 94005    | TRUE         |
| 9400g    | FALSE        |

## ceiling

数値を、渡された数値以上の最小の整数に丸めます。

### 構文

`ceiling ( number )`

以下の表に、この関数の引数を示します。

| 引数            | 説明          |
|---------------|-------------|
| <i>number</i> | 数値。丸めて返す数値。 |

### 戻り値

整数。

## 例

次の式は、価格を最も小さい整数に丸めて返します。

`ceiling ( PRICE )`

以下の表に、引数および戻り値の例を示します。

| PRICE | RETURN VALUE |
|-------|--------------|
| 39.79 | 40           |

| PRICE   | RETURN VALUE |
|---------|--------------|
| 125.12  | 126          |
| 74.24   | 75           |
| NULL    | NULL         |
| -100.99 | -100         |
| 100     | 100          |

## concat

2 つの文字列を連結します。

### 構文

`concat ( string1, string2 )`

以下の表に、この関数の引数を示します。

| 引数             | 説明                          |
|----------------|-----------------------------|
| <i>string1</i> | 文字列データ型。結合する最初の文字列を渡します。    |
| <i>string2</i> | 文字列データ型。結合する 2 番目の文字列を渡します。 |

### 戻り値

文字列。

いずれか一方の文字列が NULL である場合には、CONCAT はその文字列を無視して、もう一方の文字列を返します。

### 例

次の式は、FIRSTNAME および LASTNAME を結合します。

`concat( FIRSTNAME, LASTNAME )`

以下の表に、引数および戻り値の例を示します。

| FIRSTNAME | LASTNAME | RETURN VALUE |
|-----------|----------|--------------|
| John      | Baer     | JohnBaer     |
| NULL      | Campbell | Campbell     |
| Greg      | NULL     | Greg         |
| NULL      | NULL     | NULL         |

## ヒント

CONCAT では文字列の間にスペースが追加されません。2 つの文字列の間にスペースを追加するには、ネストした CONCAT 関数を含む式を記述します。たとえば、次の式は名前（ファーストネーム）の末尾にスペースを追加してから姓を連結します。

```
concat ( concat ( FIRST_NAME, " " ), LAST_NAME )
```

以下の表に、引数および戻り値の例を示します。

| FIRST_NAME | LAST_NAME | RETURN VALUE                               |
|------------|-----------|--|
| John       | Baer      | John Baer                                  |
| NULL       | Campbell  | Campbell ( <i>includes leading space</i> ) |
| Greg       | NULL      | Greg                                       |

## contains

文字列に他の文字列が含まれているかどうかを確認します。

### 構文

```
contains( string, substring )
```

以下の表に、この関数の引数を示します。

| 引数               | 説明   |
|------------------|--|
| <i>string</i>    | 文字列データ型。検査する文字列を渡します。引数では、大文字と小文字が区別されます。      |
| <i>substring</i> | 文字列データ型。文字列内で検索する文字列を渡します。引数では、大文字と小文字が区別されます。 |

### 戻り値

論理値。

### 例

次の式は、NAME に SHORTNAME が含まれる場合は TRUE を返します。

```
contains( NAME, SHORTNAME )
```

以下の表に、引数および戻り値の例を示します。

| NAME       | SHORTNAME | RETURN VALUE |
|------------|-----------|--------------|
| John       | Baer      | FALSE        |
| SuzyQ      | Suzy      | TRUE         |
| WorldPeace | World     | TRUE         |



| NAME           | SHORTNAME | RETURN VALUE |
|----------------|-----------|--------------|
| CASE_SENSITIVE | case      | FALSE        |

## false

常に FALSE を返します。この関数を使用して、論理値を FALSE に設定します。

### 構文

false ( )

FALSE 関数は引数を受け付けません。

### 戻り値

FALSE

### 例

FALSE 関数を他の関数と組みあわせて、強制的に FALSE の結果を出力させることができます。

以下の表に、FALSE を返す式を示します。

| EXPRESSION                          | RETURN VALUE |
|-------------------------------------|--------------|
| ( salary ) = false()                | FALSE        |
| A/B = false ( )                     | FALSE        |
| starts-with ( name, 'T' ) = false() | FALSE        |

## floor

数値を、渡された数値以下の最大の整数に丸めます。

### 構文

floor( *number* )

以下の表に、この関数の引数を示します。

| 引数            | 説明            |
|---------------|---------------|
| <i>number</i> | 数値。数値式を使用します。 |

### 戻り値

整数。

関数に NULL 値を渡した場合は NULL です。

### 例

次の式は、BANK\_BALANCE の値以下で最大の整数を返します。

floor( BANK\_BALANCE )

以下の表に、引数および戻り値の例を示します。

| BANK_BALANCE | RETURN VALUE |
|--------------|--------------|
| 39.79        | 39           |
| NULL         | NULL         |
| -100.99      | -101         |
| 5            | 5            |

## lang

要素は、コード引数と同じ言語を持つ xml:lang 属性が含まれる場合は TRUE を返します。lang 関数を使用して、XML を言語として選択します。xml:lang 属性は、要素内容の言語を確認するコードです。要素はいくつかの言語のテキストを含む可能性があります。

### 構文

lang ( *code* )

以下の表に、この関数の引数を示します。

| 引数          | 説明                       |
|-------------|--------------------------|
| <i>code</i> | 文字列データ型。要素内容の言語コードを渡します。 |

### 戻り値

論理値。

### 例

次の式は、要素内容の言語コードを検査します。

lang ( 'es' )

以下の表に、引数および戻り値の例を示します。

| XML  | RETURN VALUE |
|--|--------------|
| <Phrase xml:lang="es"><br>El perro esta en la casa.<br></Phrase> | FALSE        |
| <Phrase xml:lang="en"><br>The dog is in the house.<br></Phrase>  | TRUE         |

## normalize-space

文字列から先頭や末尾の空白を削除します。空白には、スペース文字やタブ文字などの、表示されない文字が含まれます。この関数は、空白のシーケンスを 1 バイトの空白に置き換えます。

## 構文

normalize-space ( *string* )

以下の表に、この関数の引数を示します。

| 引数            | 説明                     |
|---------------|------------------------|
| <i>string</i> | 文字列データ型。空白を含む文字列を渡します。 |

## 戻り値

文字列。

文字列が NULL の場合は、NULL を返します。

## 例

次の式は、名前から余分な空白を削除します。

normalize-space ( NAME )

以下の表に、引数および戻り値の例を示します。

| NAME         | RETURN VALUE |
|--------------|--------------|
| Jack    Dog  | Jack Dog     |
| Harry    Cat | Harry Cat    |

## not

論理条件の逆を返します。関数は条件が FALSE の場合は TRUE を返し、TRUE の場合は FALSE を返します。

## 構文

not ( *condition* )

以下の表に、この関数の引数を示します。

| 引数               | 説明               |
|------------------|------------------|
| <i>condition</i> | Boolean 式または論理値。 |

## 戻り値

論理値。

条件が NULL の場合は、NULL を返します。

## 例

次の式は、論理条件の逆を返します。

not ( EMPLOYEE = concat ( FIRSTNAME, LASTNAME ) )

以下の表に、引数および戻り値の例を示します。

| EMPLOYEE | FIRSTNAME | LASTNAME | RETURN |
|----------|-----------|----------|--------|
| Fullname | Full      | Name     | FALSE  |
| Lastname | Lastname  | First    | TRUE   |

## number

文字列または論理値を数値に変換します。

### 構文

number ( *value* )

以下の表に、この関数の引数を示します。

| 引数           | 説明                |
|--------------|-------------------|
| <i>value</i> | 論理値または文字列値を使用します。 |

### 戻り値

関数は、次のデータに対して数値を返します。

- 文字列に数値が含まれる場合、文字列を数字に変換します。文字列は空白を含めることも、数字に続けてマイナス記号を含めることもできます。空白は、文字列内の数字の後に続けて入力できます。その他の文字列は、Not a Number (NaN)です。
- 論理値が TRUE の場合は、1 に変換します。論理値が FALSE の場合は、0 に変換します。

関数に引数として渡した値が数字ではない場合、関数は Not A Number (NaN)を返します。

### 例

次の式は、Payment を Number に変換します。

number ( PAYMENT )

以下の表に、引数および戻り値の例を示します。

| PAYMENT | RETURN VALUE |
|---------|--------------|
| 850.00  | 850.00       |
| TRUE    | 1            |
| FALSE   | 0            |
| AB      | NaN          |

## round

数値を最も近い整数に丸めます。数値が 2 つの整数の間にあれば、丸めて返される数字は高い方の数値になります。

### 構文

round ( *number* )

以下の表に、この関数の引数を示します。

| 引数            | 説明                            |
|---------------|-------------------------------|
| <i>number</i> | 数値。数値データ型を渡すか、結果に数値が出る式を渡します。 |

### 戻り値

整数。

### 例

次の式は、BANK\_BALANCE を丸めます。

round( BANK\_BALANCE )

以下の表に、引数および戻り値の例を示します。

| BANK_BALANCE | RETURN VALUE |
|--------------|--------------|
| 12.34        | 12           |
| 12.50        | 13           |
| -18.99       | -19          |
| NULL         | NULL         |

## starts-with

最初の文字列が 2 番目の文字列で開始する場合は、TRUE を返します。そうでない場合は、FALSE を返します。

### 構文

starts-with ( *string*, *substring* )

以下の表に、この関数の引数を示します。

| 引数               | 説明   |
|------------------|--|
| <i>string</i>    | 文字列データ型。検索する文字列を渡します。文字列では、大文字と小文字が区別されます。         |
| <i>substring</i> | 文字列データ型。文字列内で検索するサブストリングを渡します。引数では、大文字と小文字が区別されます。 |

## 戻り値

論理型。

## 例

次の式では、NAME が FIRSTNAME で開始するかどうかを判断します。

starts-with ( NAME, FIRSTNAME )

以下の表に、引数および戻り値の例を示します。

| NAME          | FIRSTNAME | RETURN VALUE |
|---------------|-----------|--------------|
| Kathy Russell | Kathy     | TRUE         |
| Joe Abril     | Mark      | FALSE        |

## string

数値または論理値を文字列に変換します。

## 構文

string ( *value* )

以下の表に、この関数の引数を示します。

| 引数           | 説明                     |
|--------------|------------------------|
| <i>value</i> | 数値または論理値。数値または論理を返します。 |

## 戻り値

文字列。

値が渡されなかった場合は、空白の文字列を返します。NULL 値が渡された場合は、NULL を返します。

文字列関数は、次のとおりに数値を文字列に変換します。

- 数値が整数の場合、関数は小数点がなく先頭にゼロが付いていない小数点表示形式で文字列を返します。
- 数値が整数ではない場合、関数は小数点の前に少なくとも 1 けたの数値を含み、小数点の後に少なくとも 1 けたの数値を含む形式で、小数点を含む文字列を返します。
- 数値が負の数の場合、関数はマイナス記号(-)を付けて文字列を返します。

論理関数は、次のとおりに論理値を文字列に変換します。

- 論理値が FALSE の場合、関数は文字列「false」を返します。
- 論理値が TRUE の場合、関数は文字列「true」を返します。”

## 例

次の式は、数値の引数である SPEED から文字列を返します。

string( SPEED )

以下の表に、引数および戻り値の例を示します。

| SPEED    | RETURN VALUE |
|----------|--------------|
| 10.99    | '10.99'      |
| 15.62567 | '15.62567'   |
| 0        | '0'          |
| 10       | '10'         |
| 50       | '50'         |
| 1.3      | '1.3'        |

次の式は、論理型の引数である STATUS から文字列を返します。

string( STATUS )

以下の表に、引数および戻り値の例を示します。

| STATUS | RETURN VALUE |
|--------|--------------|
| TRUE   | 'true'       |
| FALSE  | 'false'      |
| NULL   | NULL         |

## string-length

文字列内の文字数を返します。文字列の末尾の空白も含めます。

### 構文

string-length ( *string* )

以下の表に、この関数の引数を示します。

| 引数            | 説明                     |
|---------------|------------------------|
| <i>string</i> | 文字列データ型。評価したい文字列を渡します。 |

### 戻り値

整数。

関数に NULL 値を渡した場合は NULL です。

### 例

次の式は、Customer Name の長さを返します。

string-length ( CUSTOMER\_NAME )



以下の表に、引数および戻り値の例を示します。

| CUSTOMER_NAME | RETURN VALUE |
|---------------|--------------|
| Bernice Davis | 13           |
| NULL          | NULL         |
| John Baer     | 9            |

## substring

文字列内の指定した位置以降の部分を返します。サブストリングは、空白を文字として文字列に含めます。

### 構文

substring ( *string*, *start* [ , *length* ] )

以下の表に、この関数の引数を示します。

| 引数            | 説明  |
|---------------|---|
| <i>string</i> | 文字列データ型。検索する文字列。  |
| <i>start</i>  | Integer データ型。カウントを開始する文字列の位置を渡します。開始位置が正の数である場合、サブストリングは文字列の先頭からその数だけ数えた位置を開始位置とします。最初の文字は 1 です。開始位置が負の数である場合、サブストリングは文字列の最後から数えた位置を開始位置とします。 |
| <i>length</i> | Integer データ型。0 より大きな値を指定する必要があります。文字列内の文字数を返します。length 引数を省略すると、サブストリングは開始位置から文字列の終わりまで、すべての文字を返します。  |

### 戻り値

文字列。

文字列に数値が含まれる場合、関数は文字列を返します。

負の整数またはゼロを渡すと、関数は空の文字列を返します。

関数に NULL 値を渡した場合は NULL です。

### 例

次の式は、PHONE の市外局番を返します。

substring( PHONE, 1, 3 )

| PHONE        | RETURN VALUE |
|--------------|--------------|
| 809-555-3915 | 809          |
| NULL         | NULL         |

次の式は、市外局番を含まない PHONE を返します。

```
substring ( phone, 5, 8 )
```

以下の表に、市外局番を含まない引数および戻り値の例を示します。

| PHONE        | RETURN VALUE |
|--------------|--------------|
| 808-555-0269 | 555-0269     |
| NULL         | NULL         |

負の開始位置を渡すと、文字列の右側から開始できます。式は、ソース文字列の左から右に、*length* 引数の数だけ読み込みます。

```
substring ( PHONE, -8, 3 )
```

以下の表に、式がソース文字列を左から右に読み込む場合の引数および戻り値の例を示します。

| PHONE        | RETURN VALUE |
|--------------|--------------|
| 808-555-0269 | 555          |
| 809-555-3915 | 555          |
| 357-687-6708 | 687          |
| NULL         | NULL         |

*length* 引数が文字列より長い場合、サブストリングは開始位置から文字列の末尾まで、すべての文字を返します。以下に例を示します。

```
substring ( 'abcd', 2, 8 )
```

「bcd」を返します。

```
substring ( 'abcd', -2, 8 )  
returns âcd.â
```

## substring-after

サブストリングの後に出現する文字列に対して、文字を返します。

### 構文

```
substring-after ( string, substring )
```

以下の表に、この関数の引数を示します。

| 引数               | 説明                             |
|------------------|--------------------------------|
| <i>string</i>    | 文字列データ型。検索する文字列を渡します。          |
| <i>substring</i> | 文字列データ型。文字列内で検索するサブストリングを渡します。 |

## 戻り値

文字列。

サブストリングが見つからない場合は空の文字列です。

関数に NULL 値を渡した場合は NULL です。

## 例

次の式は、市外局番(415)の後に出現する PHONE の文字列を返します。

substring-after ( PHONE, (415) )

以下の表に、引数および戻り値の例を示します。

| PHONE         | RETURN VALUE |
|---------------|--------------|
| (415)555-1212 | 555-1212     |
| (408)368-4017 | -            |
| NULL          | NULL         |
| (415)366-7621 | 366-7621     |

## substring-before

サブストリングの前に出現する文字列の一部を返します。

## 構文

substring-before ( *string*, *substring* )

以下の表に、この関数の引数を示します。

| 引数               | 説明                             |
|------------------|--------------------------------|
| <i>string</i>    | 文字列データ型。検索する文字列を渡します。          |
| <i>substring</i> | 文字列データ型。文字列内で検索するサブストリングを渡します。 |

## 戻り値

文字列。

サブストリングが見つからない場合は空の文字列です。

関数に NULL 値を渡した場合は NULL です。

## 例

次の式は、Third Street Address で出現する数値を返します。

substring-before ( ADDRESS, Third Street )

以下の表に、属性および戻り値の例を示します。

| ADDRESS          | RETURN VALUE |
|------------------|--------------|
| 100 Third Street | 100          |
| 250 Third Street | 250          |
| 600 Third Street | 600          |
| NULL             | NULL         |

## translate

文字列内の文字を他の文字に変換します。関数は、その他 2 つの文字列を変換するペアとして使用します。

### 構文

translate ( *string1*, *string2*, *string3* )

以下の表に、この関数の引数を示します。

| 引数             | 説明   |
|----------------|--|
| <i>string1</i> | 文字列データ型。変換する文字列を渡します。  |
| <i>string2</i> | 文字列データ型。どの文字を変換するかを定義する文字列を返します。translate は、 <i>string1</i> 内の各文字を、その文字の <i>string2</i> 内での位置を示す数字に置き換えます。   |
| <i>string3</i> | 文字列データ型。暗号化された <i>string1</i> の文字を、何に変換するかを定義する文字列を渡します。translate は、暗号化された <i>string1</i> 内の各文字を、 <i>string2</i> の位置番号の位置にある <i>string3</i> 内の文字で置き換えます。 |

### 戻り値

文字列。

### 例

次の式では、その他 2 つの文字列を使用して文字列を変換します。

translate ( EXPRESSION, STRING2, STRING3 )

以下の表に、引数および戻り値の例を示します。

| EXPRESSION      | STRING2 | STRING3 | RETURN VALUE    |
|-----------------|---------|---------|-----------------|
| A Space Odissei | i       | y       | A Space Odyssey |
| rats            | tras    | TCAS    | CATS            |
| bar             | abc     | ABC     | BAr             |

文字が *string2* に出現しない場合、変換は EXPRESSION の文字を変更しません。文字が EXPRESSION および *string2* で出現し、*string3* では出現しない場合、文字は返された文字列では出現しません。

## true

常に TRUE を返します。この関数を使用して、論理値を TRUE に設定します。

### 構文

true ()

TRUE 関数は引数を受け付けません。

### 戻り値

論理値は TRUE です。

### 例

以下の表に、TRUE を返す式の例を示します。

| EXPRESSION                         | RETURN VALUE |
|------------------------------------|--------------|
| ( decision ) = true ()             | TRUE         |
| A/B = true ()                      | TRUE         |
| ( starts-with ( name, 'T' ))= true | TRUE         |