

# Novell exteNd Composer™

## JDBC Connect

4.2

[www.novell.com](http://www.novell.com)

ユーザガイド



Novell®

## 保証と著作権

Copyright ©1999, 2000, 2001, 2002, 2003 SilverStream Software, LLC. All rights reserved.

SilverStream ソフトウェア製品は、SilverStream Software LLC により著作権とすべての権利が保留されています。

SilverStream は SilverStream Software, LLC の登録商標です。Novell は、Novell, Inc. の登録商標です。

ソフトウェアとマニュアルの所有権、および特許、著作権、およびそれに関連するその他のすべての財産権は常に、単独で排他的に SilverStream とそのライセンサーに保留され、当該所有権と矛盾するいかなる行為も行わないものとします。本ソフトウェアは、著作権法と国際条約規定で保護されています。ソフトウェアならびにそのマニュアルからすべての著作権に関する通知とその他の所有権に関する通知を削除してはならず、ソフトウェアとそのマニュアルのすべてのコピーまたは抜粋に当該通知を複製しなければなりません。本ソフトウェアのいかなる所有権も取得するものではありません。

Jakarta-Regexp Copyright ©1999 The Apache Software Foundation. All rights reserved. Ant Copyright ©1999 The Apache Software Foundation. All rights reserved. Xalan Copyright ©1999 The Apache Software Foundation. All rights reserved. Xerces Copyright ©1999-2000 The Apache Software Foundation. All rights reserved. Jakarta-Regexp、Ant、Xalan、Crimson、および Xerces ソフトウェアは、The Apache Software Foundation によりライセンスを付与され、Jakarta-Regexp、Ant、Xalan、Crimson、および Xerces のソースおよびバイナリ形式での再配布および使用は、変更のあるなしにかかわらず、以下の条件が満たされることを前提として許可されます。1. ソースコードの再配布に上記の著作権に関する通知、条件のリスト、および以下の権利放棄に関する通知が記載されていること。2. バイナリ形式の再配布では上記の著作権に関する通知、条件のリスト、および以下の権利放棄に関する通知がマニュアルまたは配布の際に提供されるその他の資料、あるいはその両方に記載されていること。3. エンドユーザの資料には、適宜、以下の通知を再配布の際に含めてください。「この製品には、Apache Software Foundation (<http://www.apache.org/>) により開発されたソフトウェアが含まれています」代わりに、この謝辞をソフトウェア自体に表示し、当該サードパーティに対する謝辞が通常表示される場所に表示することもできます。4. 「The Jakarta Project」、「Jakarta-Regexp」、「Xerces」、「Xalan」、「Ant」、および「Apache Software Foundation」は、書面による事前の許可なく、このソフトウェアから派生する製品を推薦したり、販売促進したりするのに使用してはなりません。書面による許可については、[apache@apache.org](mailto:apache@apache.org) <<mailto:apache@apache.org>> にお問い合わせください。5. 本ソフトウェアから派生する製品は「Apache」と呼ばれてはならず、「Apache」は The Apache Software Foundation の事前の書面による許可なくその名前に使用することはできません。本ソフトウェアは「現状のまま」提供され、いかなる明示的、暗黙の保証も行われるものではありません。販売可能性や特定の目的に対する適合性に対する暗黙の保証も行われません。いかなる場合でも、Apache Software Foundation またはその関係者はいかなる直接的、間接的、偶発的、特別な、免除的、または結果的な損害（代替品やサービスの調達、使用機会、データ、または利益の喪失、または業務の中断などを含む）についても、理論上責任がある場合でも、契約上の責任がある場合でも、厳密な責任、または瑕疵（怠慢などを含む）があった場合でも、ソフトウェアの使用の過程で生じ、当該損害の可能性を助言した場合であっても、責任をもちません。

Copyright ©2000 Brett McLaughlin & Jason Hunter. All rights reserved. ソースおよびバイナリ形式での再配布および使用は、変更のあるなしにかかわらず、以下の条件が満たされることを前提として許可されます。1. ソースコードの再配布に上記の著作権に関する通知、条件のリスト、および以下の権利放棄に関する通知が記載されていること。2. バイナリ形式の再配布では上記の著作権に関する通知、条件のリスト、および以下の権利放棄に関する通知がマニュアルまたは配布の際に提供されるその他の資料、あるいはその両方に記載されていること。3. 「JDOM」という名前は、書面による事前の許可なく、このソフトウェアから派生する製品を推薦したり、販売促進したりするのに使用してはなりません。書面による許可については、[license@jdom.org](mailto:license@jdom.org) <<mailto:license@jdom.org>> にお問い合わせください。4. 本ソフトウェアから派生する製品は「JDOM」と呼ばれてはならず、「JDOM」は JDOM Project Management ([pm@jdom.org](mailto:pm@jdom.org)) <<mailto:pm@jdom.org>> の事前の書面による許可なくその名前に使用することはできません。本ソフトウェアは「現状のまま」提供され、いかなる明示的、暗黙の保証も行われるものではありません。販売可能性や特定の目的に対する適合性に対する暗黙の保証も行われません。いかなる場合でも、Apache Software Foundation またはその関係者はいかなる直接的、間接的、偶発的、特別な、免除的、または結果的な損害（代替品やサービスの調達、使用機会、データ、または利益の喪失、または業務の中断などを含む）についても、理論上責任がある場合でも、契約上の責任がある場合でも、厳密な責任、または瑕疵（怠慢などを含む）があった場合でも、ソフトウェアの使用の過程で生じ、当該損害の可能性を助言した場合であっても、責任をもちません。

Sun Microsystems, Inc. Sun, Sun Microsystems, Sun Logo Sun、Sun のロゴ、Sun Microsystems、JavaBeans、Enterprise JavaBeans、JavaServer Pages、Java Naming and Directory Interface、JDK、JDBC、Java、HotJava、HotJava Views、Visual Java、Solaris、NEO、Joe、Netra、NFS、ONC、ONC+、OpenWindows、PC-NFS、SNM、SunNet Manager、Solaris sunburst design、Solstice、SunCore、SolarNet、SunWeb、Sun Workstation、The Network Is The Computer、ToolTalk、Ultra、Ultracomputing、Ultraserver、Where The Network Is Going、SunWorkShop、XView、Java WorkShop、Java Coffee Cup のロゴ、Visual Java、および NetBeans は、米国およびその他の国の Sun Microsystems, Inc. の商標ならびに登録商標です。

Copyright ©2001 Extreme! Lab, Indiana University License. <http://www.extreme.indiana.edu>. 同社により許可が無料で、Indiana University ソフトウェアと関連する Indiana University のドキュメントファイル (「IU Software」) のコピーを取得したすべての人に、制限なく IU Software を取り扱うために付与されます。その際に、IU Software の使用、コピー、変更、マージ、公開、配布、サブライセンス、または販売、あるいはそれらのすべてに関する権利に制限はなく、IU Software が指定した人に以下の条件に基づき権利を付与します。上記の著作権に関する通知とその許可に関する通知は、IU Software のすべてのコピーおよび主要部分に含まれる必要があります。本 IU ソフトウェアは「現状のまま」提供され、いかなる明示的、暗黙の保証も行われるものではありません。販売可能性、特定の目的に対する適合性や権利侵害がないことに対する暗黙の保証も行われません。いかなる場合でも、作成者または著作権所有者は、契約上の責任がある場合でも、厳密な責任、または瑕疵 (怠慢などを含む) があつた場合でも、UI Software に関連して、または UI Software の使用やその他の取引の過程で生じた場合であっても、クレーム、損害、その他の責任について責任をもちません。

本ソフトウェアは、著作権をもつ SSLavaTM Toolkit の一部です。Copyright ©1996-1998 by Phaos Technology Corporation. All rights reserved.

Copyright © 1994-2002 W3C® (Massachusetts Institute of Technology, Institut National de Recherche Informatique et en Automatique, Keio University), all Rights Reserved. <http://www.w3.org/consortium/legal>. この W3C の成果物 (ソフトウェア、ドキュメント、またはその他の関連品目を含む) は、以下のライセンスの下で著作権所有者により提供されています。この成果物の取得、使用、またはコピー、あるいはそれらのすべてにより、ライセンサーは以下の条件を読み、理解し、遵守することに合意するものとします。本ソフトウェアとそのドキュメントの使用、コピー、変更、および配布は、変更のあるなしにかかわらず、いかなる目的でも無料または本契約で許可された使用料をもって許可されます。ただし、変更箇所を含む本ソフトウェアとドキュメントのすべてまたはその一部に以下のとおり記述することを前提とします。1. この通知の全文は、再配布物または派生物のユーザが見やすい場所に掲示しなければなりません。2. すべての前もって存在する知的所有権の放棄、通知、または条件。存在しない場合は、以下の形式の短い通知 (ハイパーテキストが望ましい、テキストでも良い) を再配布または派生コードの本文内で使用しなければなりません。「Copyright © [Date-of-software] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All rights reserved. <http://www.w3.org/Consortium/Legal/>」3. W3C のファイルに変更または修正を加えた場合はその日付を含む通知。(コードが派生する場所への URI を示すことをお勧めします。) 本ソフトウェアは「現状のまま」提供され、いかなる明示的、暗黙の保証も行われるものではありません。販売可能性、特定の目的に対する適合性やサードパーティの特許、著作権、商標またはそのたの権利を侵害しないことに対する暗黙の保証も行われません。著作権の所有者は本ソフトウェアまたはマニュアルの使用の結果生じる、直接的、間接的、特殊な、または結果的な損害に対していかなる責任も負いません。著作権所有者の名前および商標は、特別な書面による事前の承諾なしにソフトウェアに関する広告や広報に使用してはなりません。本ソフトウェアおよび関連する資料の著作権の所有権は常に、著作権所有者に帰属するものとします。

米国 Novell, Inc.  
1800 South Novell Place  
Provo, UT 85606

[www.novell.com](http://www.novell.com)

JDBC Connect ユーザガイド  
2003 年 1 月  
000-000000-000

**オンラインマニュアル**：この製品およびその他の Novell 製品のオンラインマニュアルや更新情報については、  
<http://www.novell.com/documentation> を参照してください。

# このガイドについて

## 目的

本書では、JDBC Component Editor と呼ばれる、exteNd Connect の使用方法について説明します。JDBC Component Editor は、exteNd Composer の標準コンポーネントエディタです。

## 対象読者

本書は、exteNd Composer を使用してデータベース対応のサービスおよびコンポーネントを開発する開発者やシステムインテグレータを対象としています。

## 前提条件

本書では、exteNd Composer の作業環境および開発オプションについて理解していることを前提としています。また、Structured Query Language (SQL) を理解していることも前提としています。

## 追加のドキュメント

Novell exteNd Director の完全なマニュアルのセットは、Novell マニュアルの Web サイト (<http://www.novell.com/documentation-index/index.jsp>) を参照してください。



# 目次

このガイドについて	5
<b>1 exteNd Composer および JDBC へようこそ</b>	<b>9</b>
はじめに	9
exteNd Connects について	10
JDBC とは	10
JDBC の機能	11
exteNd の JDBC コンポーネントについて	11
JDBC Component Editor で作成できるアプリケーションの種類	12
<b>2 JDBC コンポーネントエディタを使用する</b>	<b>13</b>
JDBC 接続リソースを作成する	13
定数駆動型および式駆動型の接続パラメータについて	13
JDBC ドライバおよび接続プールについて	14
コンポーネント用の XML テンプレートの作成	18
<b>3 JDBC コンポーネントの作成</b>	<b>19</b>
JDBC コンポーネントを作成する前に	19
JDBC コンポーネントエディタウィンドウについて	22
クエリペインについて	23
<b>4 JDBC アクションを実行する</b>	<b>27</b>
アクションについて	27
SQL Statement アクション	28
プリペアドステートメント	28
SQL ステートメントを作成する	29
サンプルクエリを作成する	30
ストアドプロシージャを使用する	31
SQL ステートメントのコロン	35
SQL ステートメントを実行する	35
結果をチェックする	35
結果を出力 DOM にマップする	36
SQL Batch アクション	37
Start Batch	38
Execute Batch	39
Discard Batch	39
Batch アクションを作成する	40
JDBC 固有の Expression Builder のプロパティ	41
JDBC コンポーネントエディタで他のアクションを使用する	42
エラーおよび SQL メッセージの処理	45

<b>5</b>	<b>[Custom Result Mapping] を使用する</b>	<b>47</b>
	デフォルト結果マッピングについて .....	47
	カスタム結果マッピングについて .....	49
	カスタム結果マッピングおよびエイリアスについて .....	50
	[MapTarget] タブを使用する .....	50
	MapTarget の例を参照する .....	53
	[Detail Rows] タブを使用する .....	55
	[Detail Rows] の例を参照する .....	56
	[Declare Group/Repeat] タブを使用する .....	58
	[Declare Group/Repeat] の例を参照する .....	60
<b>6</b>	<b>ストアードプロシージャ</b>	<b>65</b>
	ストアードプロシージャのマッピングについて .....	65
	規則を作成する .....	66
	[Stored Procedure Mapping Setup] ダイアログボックスを使用する .....	66
	返される結果セット .....	68
<b>A</b>	<b>JDBC 用語集</b>	<b>69</b>
<b>B</b>	<b>予約語</b>	<b>73</b>



# 1

## exteNd Composer および JDBC へようこそ

### はじめに

『Novell exteNd Connect ユーザーガイド』へようこそ。このガイドは、Composer の全機能 (Connect コンポーネントエディタを除く) の使用方法が詳しく説明されている『exteNd Composer ユーザーガイド』に付属しています。そのため、『Composer ユーザーガイド』をご覧になっていない場合は、このガイドを使用する前に読んで内容を確認してください。

exteNd Composer には、JDBC コネクタなどの Connect ごとに異なるコンポーネントエディタが用意されています。各コンポーネントエディタの特殊な機能は、これと同じような別のガイドで説明されています。

exteNd Composer を使用しており、コアとなるコンポーネントエディタ (XML Map コンポーネントエディタ) に精通している場合は、このガイドに従って Tandem コンポーネントエディタを簡単に使用することができます。

**注記：** Component Editor を正しく使用するには、SQL ステートメントの作成および構築を理解している必要があります。

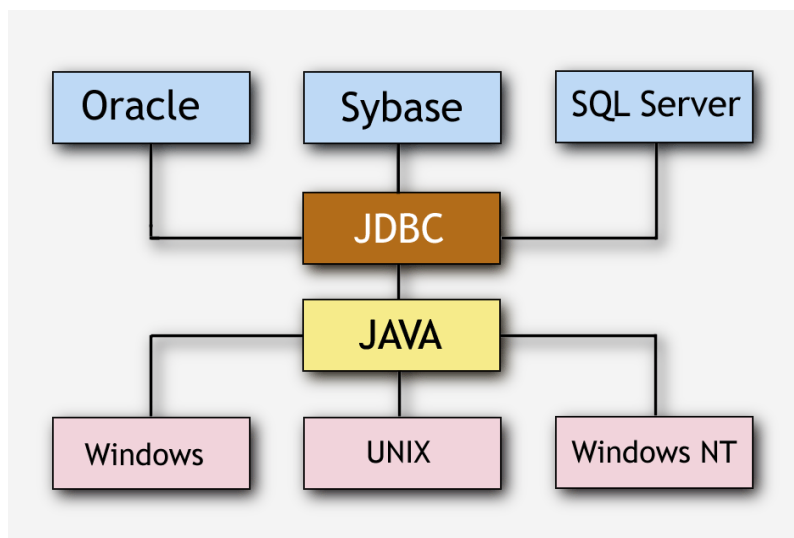
## exteNd Connects について

Novell exteNd は、単純なハブ & スポークアーキテクチャに基づいて構築されています。ハブは、XML ドキュメントを受け付けてドキュメントを処理し、XML ドキュメントを返す、強力な XML 変換エンジンです。スポーク (つまり Connect) は、XML 対応でないデータのソースを「XML に対応させる」プラグインモジュールです。これらのデータソースには、レガシー COBOL/VSAM で管理されている情報から、HTML ページに対するメッセージキューまで何でも使用できます。exteNd Connect は、情報ソースを XML に対応させるために各製品で使用されている統合方法に従って分類できます。統合方法は、インターネットベースのコンピュータアーキテクチャに対する現在のシステム設計において使用される主要な区分を反映したものです。exteNd では、B2Bi のニーズに応じて、ユーザインタフェースレベル、プログラム論理レベル、またはデータレベル、あるいはそのすべてでビジネスシステムを統合できます (図 1-1 を参照)。

## JDBC とは

JDBC は、SQL ステートメントを実行するための Java ベース API (Application Programming Interface) です。JDBC は、「Java Database Connectivity」の頭字語と間違われますが、実際は頭字語ではなく、商標名です。JDBC は、Java プログラム言語で作成されたクラスおよびインタフェースのセットで構成されています。このプログラム言語では、Oracle、Sybase、Informix などの様々なデータベースに個々のプログラムを作成しなくても、1つのプログラムを作成してこれらにアクセスできます。

JDBC API を使用して 1つのプログラムを作成できます。このプログラムは、SQL ステートメントを該当するデータベースに送信できます。アプリケーションは、Java プログラム言語で作成されているので、異なるプラットフォームで実行する様々なアプリケーションを作成する必要はありません。Java と JDBC を併用することで、プログラムを一度作成するだけで、どこでも実行できるようになります (次の図を参照)。



## JDBC の機能

JDBC では、次のことが可能です。

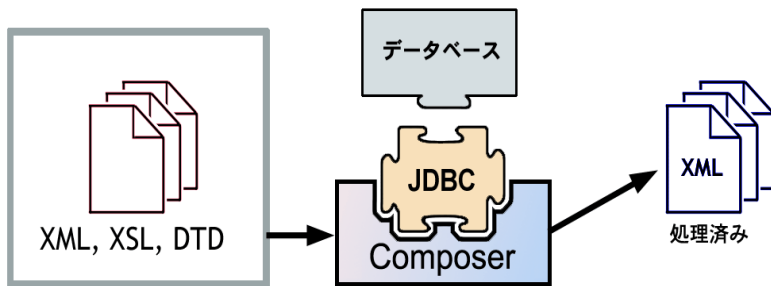
- ◆ データベースとの接続を確立する
- ◆ データベースにより処理される SQL ステートメント (またはクエリ) を送信する
- ◆ データベース操作の結果を処理する

JDBC は、SQL コマンドを直接呼び出すときに使用される低レベルインタフェースです。JDBC は、Composer に統合され、コンポーネントとデータベース間のインタフェースとして機能します。これにより、データベースとの接続の確立、SQL ステートメントの送信、および結果の処理をプログラムにより実行することができます。Composer は、必要な SQL コマンドの視覚的な構築を可能にするツールを提供します。

## exteNd の JDBC コンポーネントについて

XML Map コンポーネントと同様に、JDBC コンポーネントは、2つの異なる XML テンプレート (つまり、要求 XML ドキュメントと応答 XML ドキュメント) 間でデータをマップ、変換、および転送するために設計されています。ただし、データベースに接続して、クエリ内の DOM からの要素を使用してデータベースの SQL ステートメントを処理し、クエリの結果を DOM にマップします。

JDBC コンポーネントでは、1つのXMLドキュメントから別のXMLドキュメント、またはXMLドキュメントからデータベーステーブルへのデータのマッピングや転送など、単純なデータ操作を実行できます。また、異種データベースのデータ要求、1つ以上のドキュメントに対するデータの送受信、データベースへのSQLトランザクションの実行、およびドキュメント自体の転送など、高度な操作も実行できます。さらに、XML Map コンポーネントのように、JDBC コンポーネントでは、XSLの処理、メールの送信、およびHTTPプロトコルを使用したXMLドキュメントのポストと受信を行うXML Map コンポーネントを作成することも可能です。



JDBC Connect は、XML-ベースのデータ操作のバックプレーンとして exteNd Composer を使用し、実行時（および設計時）のデータベースアクセスを可能にします。exteNd Composer を使用すると、アクションモデルを JDBC コンポーネント内にまとめ、HTTP（オプション）を転送メカニズムとして使用して、高度なデータ変換を実行できます。また、設計時のライブデータベース接続が可能なので、SQL クエリを設計プロセスの一部として編集およびデバッグできます。

## JDBC Component Editor で作成できるアプリケーションの種類

JDBC からアクセスできるデータストア間でのデータの受け渡しが必要で、XML を交換形式として使用する、B2B アプリケーションを作成できます。たとえば、データベースから製品の説明、画像、および価格情報を取得してユーザの Web ブラウザに表示するアプリケーションを作成できます。この情報が複数のデータベースにある場合、ユーザに表示する前に個々のデータベースの情報をマージできます。

# 2

## JDBCコンポーネントエディタを使用する

### JDBC 接続リソースを作成する

JDBC コンポーネントを作成する前に、SQL データベースにアクセスするための接続リソースを作成する必要があります。JDBC コネクタを含む各接続では、その独自接続タイプが使用されます。各接続タイプは、特定の外部データソースとの接続に使用されるパラメータの数とタイプで決まります。

### 定数駆動型および式駆動型の接続パラメータについて

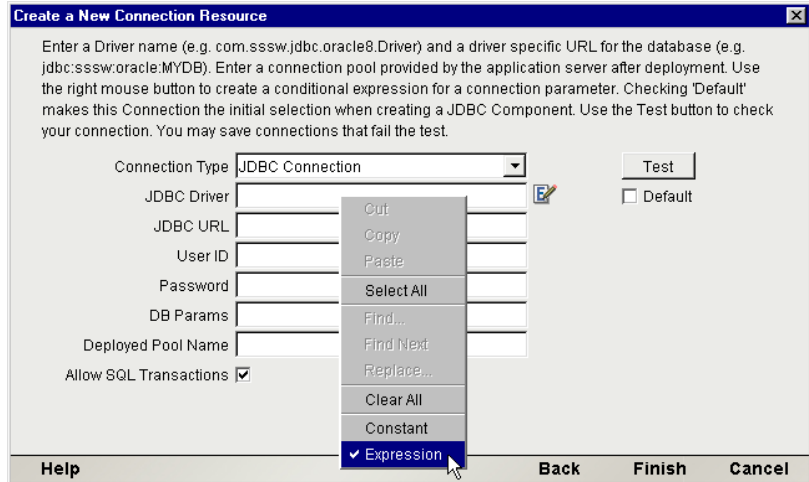
接続パラメータの値は、定数または式としての方法のうちいずれかを使用して指定できます。定数ベースのパラメータでは、接続が使用されるたびに [Connection] ダイアログボックスに入力した値を使用します。式ベースのパラメータでは、ランタイム時に接続が使用されるたびに異なる値となりえる、プログラムのな式を使用して値を設定できます。この場合、接続の動作は柔軟になり、接続の使用ごとにランタイム時の条件に対応できるようになります。

たとえば、JDBC 接続における式駆動型のパラメータの非常に単純な使用の 1 つは、ユーザ ID とパスワードを PROJECT 変数 (例: PROJECT.XPATH(“USERCONFIG/MyDeployUser”)) として定義することです。このようにすると、プロジェクトを配備する際に、Deployment Wizard で PROJECT 変数を最終配備環境に適切な値に更新できます。それとは正反対に、アプリケーションサーバで Java ビジネスオブジェクトを照会するカスタムスクリプトを使って、使用するユーザ ID とパスワードを決定することもできます。

#### ➤ 定数駆動型から式駆動型にパラメータを切り替える

- 1 変更するパラメータフィールドでマウスを右クリックします。
- 2 コンテキストメニューから [Expression] を選択すると、エディタボタンが表示されるか、または有効になります。

- 3 ボタンをクリックしてから、ランタイム時に有効なパラメータ値を返す式を作成します ( 文字列は二重引用符で囲みます )。



## JDBC ドライバおよび接続プールについて

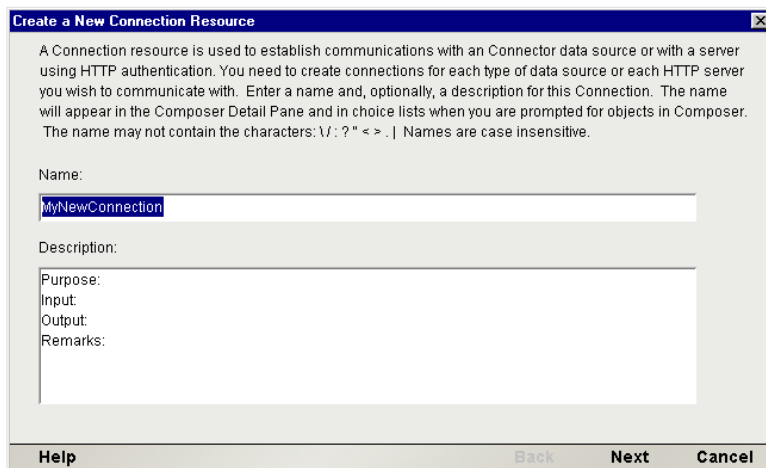
接続リソースを作成する場合、ドライバ名および接続プールを指定するように尋ねられます。

JDBC ドライバ **sun.jdbc.odbc.JdbcOdbcDriver** は、JRE (Java Runtime Environment: Composer ディレクトリにあります) の一部で、このドライバを使用すると、接続を確立できます。ただし、その他の JDBC ドライバを取得することもできます。たとえば、Novell Extend Application Server には、独自の JDBC ドライバがあります。また、使用している SQL データベースのベンダ Web サイトにアクセスして、ドライバをダウンロードすることもできます。

「接続プール」とは、アプリケーションサーバが管理する様々なアプリケーションに対して、そのアプリケーションサーバにより管理されるデータベースの接続セットのことです。接続プールを使用すると、同じアプリケーションサーバで実行している複数のアプリケーションに対して、データベースや接続リソースをより効率的に使用することができます。これにより、システム全体のパフォーマンスが向上します。アプリケーションサーバのプール名については、サーバ管理者にお尋ねください。Novell Extend Application Server 内の配備において、プール名は、**Databases/DBName/DataSource** になります。ここで、**DBName** は、データベースをサーバに追加するときを使用した名前です。たとえば、アプリケーションサーバが提供されている **TutorialBegin3** データベースに接続している場合、プール名は、**Databases/TutorialBegin3/DataSource** になります。

## ➤ JDBC 接続リソースを作成する

- 1 [File]、[New xObject]、[Resource]、[Connection] の順に選択します。「Create a New Connection Resource」ウィザードが表示されます。



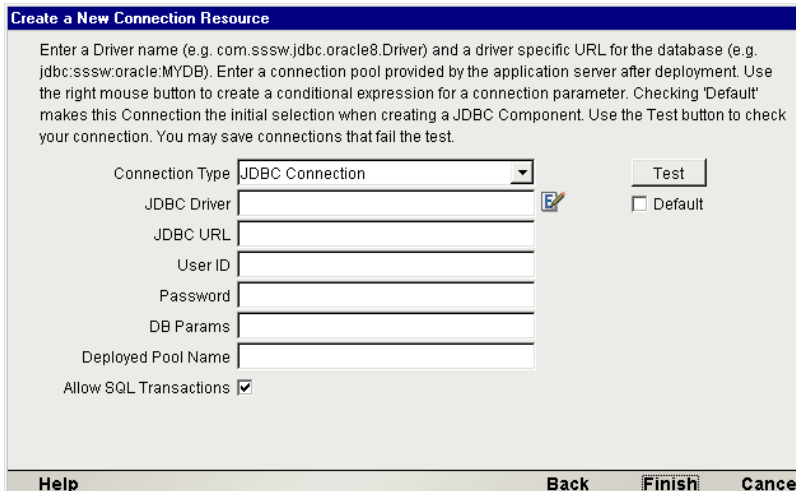
A Connection resource is used to establish communications with an Connector data source or with a server using HTTP authentication. You need to create connections for each type of data source or each HTTP server you wish to communicate with. Enter a name and, optionally, a description for this Connection. The name will appear in the Composer Detail Pane and in choice lists when you are prompted for objects in Composer. The name may not contain the characters: \/:? "< > .| Names are case insensitive.

Name:  
MyNewConnection

Description:  
Purpose:  
Input:  
Output:  
Remarks:

Help Back Next Cancel

- 2 [Name] に、接続オブジェクトの名前を入力します。
- 3 オプションとして、[Description] に説明テキストを入力します。
- 4 [次へ] をクリックします。



Enter a Driver name (e.g. com.ssw.jdbc.oracle8.Driver) and a driver specific URL for the database (e.g. jdbc:ssw:oracle:MYDB). Enter a connection pool provided by the application server after deployment. Use the right mouse button to create a conditional expression for a connection parameter. Checking 'Default' makes this Connection the initial selection when creating a JDBC Component. Use the Test button to check your connection. You may save connections that fail the test.

Connection Type: JDBC Connection

JDBC Driver

JDBC URL

User ID

Password

DB Params

Deployed Pool Name

Allow SQL Transactions

Test

Default

Help Back Finish Cancel

- 5 [Connection Type] プルダウンメニューから [JDBC Connection] を選択します。

- 6** [JDBC Driver] フィールドで、使用する JDBC ドライバの名前を入力します。たとえば、Novell exteNd ドライバの `com.sssw.jdbc.mss.odbc.AgOdbcDriver` です ( 詳細については、[14 ページ「JDBC ドライバおよび接続プールについて」](#) を参照してください)。

**注記：** このパラメータ、およびこのダイアログボックスの他のパラメータは、[Expression] を使用して動的に設定できます。この章の前半の「定数駆動型および式駆動型の接続パラメータについて」を参照してください。

- 7** [JDBC URI] フィールドで、アクセスするデータベースの位置を入力します。たとえば、「`jdbc:ssw:odbc:XTutorial`」を入力します。ここで、`jdbc:ssw:odbc:` は、ドライバにより要求されるシンタックスで、`XTutorial` は、コンポーネントが実行するコンピュータで定義されている ODBC データソース名 (Data Source Name: DSN) です ( この DSN は、[Windows Control] パネルの [ODBC Administrator] にアクセスして、Composer から外部的に定義されます)。配備については、サーバで ODBC 接続が可能な限り、上記の接続を保守できます。通常、データベースアクセスの管理にアプリケーションサーバの機能を利用するのが普通です。この場合、接続プール名を次のように指定する必要があります。

**注記：** [JDBC Driver] および [JDBC URI] フィールドでは、どちらも大文字と小文字が区別されます。

- 8** 有効な「ユーザ ID」を入力して、選択データベースにサインオンします。

- 9** 選択データベースに有効な「パスワード」を入力します。

- 10** [DB Params] フィールドに、接続に適用される任意のデータベース固有のパラメータを入力します。たとえば、Novell exteNd SQL Anywhere データベースへの更新を許可する場合、パラメータに「`S3SqlAnywhereAuth=true`」を入力します。パラメータは、「名前=値」のペアで入力する必要があります。「名前=値」のペアを複数指定する場合、`param1=true;param2=true;param3=false` のように、セミコロンを使用してペアを区切ります。

**注記：** データベース固有のパラメータを使用しない場合、このフィールドに「`false`」を入力します。

- 11** 必要な場合、「プール名」を入力します。詳細については、[14 ページ「JDBC ドライバおよび接続プールについて」](#) を参照してください。

**注記：** 接続プールは、配備環境ではオペレーショナルのみです。ここの名前を設定しても、Composer 接続には影響ありません。影響を受けるのは、配備済みプロジェクトだけです。

- 12** コンポーネントのアクションモデルのトランザクションを (SQL Begin、Commit、および Rollback バーブを使用して) 直接コントロールする場合、[Allow SQL Transactions] チェックボックスをオンにします。

[Allow SQL Transactions] チェックボックスをオンにすると、次のように様々な効果があります。



— JDBC ドライバの自動コミットが「オフ」になります (ただし、自動コミットフラグの状態は、接続がプールに戻る前に、トランザクションの最後に回復されます)。

— すべての SQL コミットおよびロールバックコマンドが、対応する JDBC 接続呼び出しに変換されます。

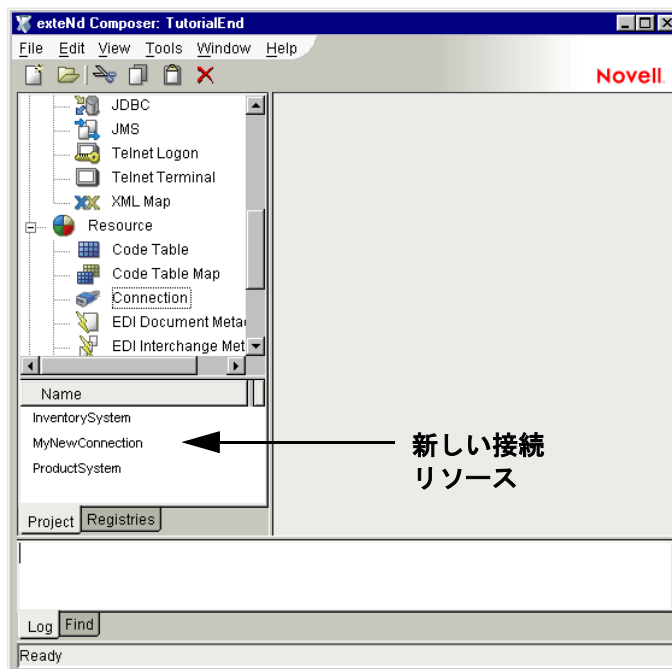
— Composer Enterprise Server により、コンポーネントの最後の Execute SQL アクションがチェックされ、最後のアクションがコミットかロールバックかが確認されます。最後のアクションが、コミットでもロールバックでもない場合、未完了の接続 (つまり、変更がコミットされていない接続) がプールに返されないように、Composer Enterprise Server はデフォルトでロールバックを実行します。

**注記:** [Allow SQL Transactions] チェックボックスの詳細については、アプリケーションサーバの『*exteNd Composer Application Server Guide*』でトランザクションに関する章を参照してください。

- 13 プロジェクトで作成する新しい JDBC 接続のデフォルトの接続として現在の接続を使用する場合、[Default] チェックボックスをオンにします。
- 14 [Test] をクリックして、接続が正常が確認します。「success」または「failure」メッセージが、接続に対して表示されます。*接続が失敗したとしても、リソースの作成を続けることができます。*

**注記:** 接続プール (定義されている場合) はテストされません。

- 15 [Finish] をクリックします。新しく作成されたリソース接続オブジェクトが、Composer 接続リソースの詳細ペインに表示されます。



## コンポーネント用の XML テンプレートの作成

接続リソースのほかに、JDBC コンポーネントでは、コンポーネントを設計するためのサンプルドキュメントを持つよう、XML テンプレートをすでに作成していることが必要とされます ( 詳細については、『*Composer ユーザガイド*』の第 5 章「*Creating XML Templates*」を参照してください)。

また、コンポーネント設計によって別の xObject リソース ( カスタムスクリプトやコードテーブルマップなど ) が要求される場合は、JDBC コンポーネントを作成する前にこれらのリソースを作成することが推奨されます。詳細については、『*Composer ユーザガイド*』の「*Creating Custom Scripts*」を参照してください。

# 3

## JDBC コンポーネントの作成

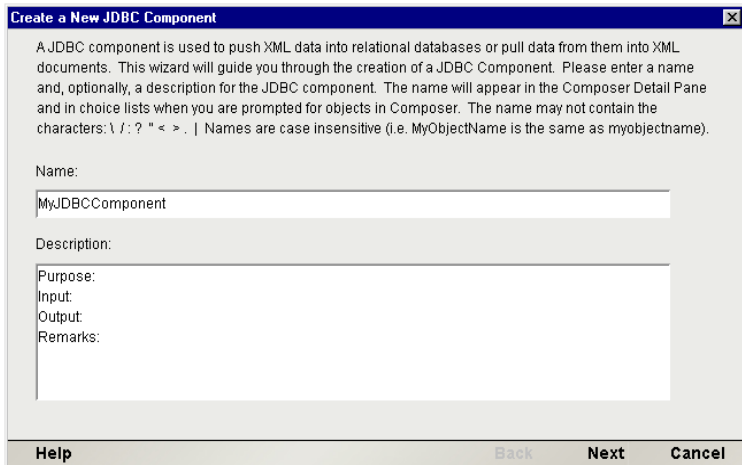
### JDBC コンポーネントを作成する前に

すべての exteNd コンポーネントと同様に、JDBC コンポーネントを作成する最初の手順は、必要な XML テンプレートを指定することです ( 詳細については、個々の『*Composer ユーザガイド*』の「*Creating a New XML Template*」を参照してください)。XML テンプレートを指定すると、コンポーネントによって処理される入力および出力を表すテンプレートのサンプルドキュメントを使用して、コンポーネントを作成できます。

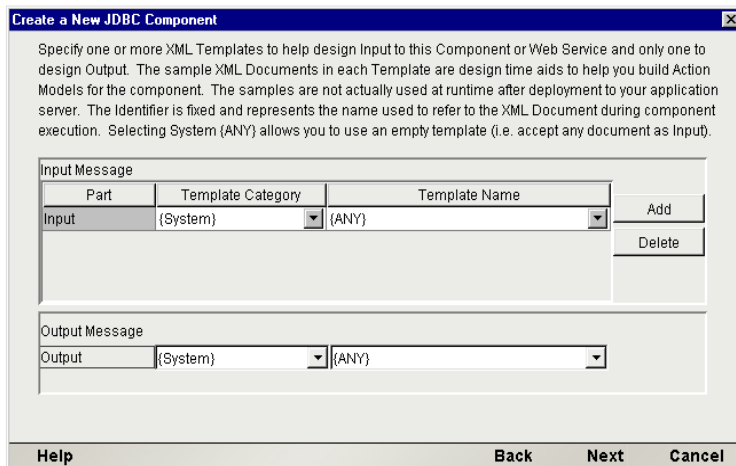
また、JDBC コンポーネントの作成プロセスの一部として、JDBC 接続を選択したり、新しい接続を作成したりすることが可能です。接続は、事前に作成した場合、新しいJDBCコンポーネントすべてに対して使用可能になります(13ページ「[JDBC 接続リソースを作成する](#)」を参照)。

#### ▶ 新しい JDBC コンポーネントを作成する

- 1 [File] > [New xObject] > [Component] > [JDBC] の順に選択します。「Create a New JDBC Component」ウィザードが表示されます。



- 2 新しい JDBC コンポーネントの「名前」を入力します。
- 3 オプションとして、[Description] に説明テキストを入力します。
- 4 [Next] をクリックします。New JDBC Component ウィザードの [XML Property Info] パネルが表示されます。



- 5 1つまたは複数の入力テンプレートを、次のように指定します。
  - ◆ デフォルトのカテゴリと異なる場合は、「テンプレートカテゴリ」を選択します。

- ◆ 選択した「テンプレートカテゴリ」にある XML テンプレートのリストから「テンプレート名」を選択します。
  - ◆ 入力 XML テンプレートをさらに追加するには、[Add] をクリックして、手順 2 から 4 を繰り返します。
  - ◆ 入力 XML テンプレートを削除するには、エントリを選択して [Delete] をクリックします。
- 6 出力に対して XML テンプレートを選択します (Output という名前の出力 DOM は 1 つしかありません)。
- 注記：** 出力テンプレートとして **{System}{ANY}** を選択すると、構造が含まれない出力 XML テンプレートを指定できます。たとえば、[Result Mapping] タブの [Custom] オプションを使用してカスタマイズされた出力 DOM を生成する場合などに、このようなテンプレートを指定します (詳細については、『Composer ユーザガイド』の XML Map コンポーネントの作成に関する章である「Creating an Output DOM without Using a Template」を参照してください)。
- 7 [Next] をクリックします。「Create a New JDBC Component」ウィザードの接続情報パネルが表示されます。

Dialog box titled "Create a New JDBC Component".

Specify which Connection you wish to use for this Component or Service. To change any connection parameters, you must change them in the Connection Resource object or create a new Connection Resource of the same type with different parameters.

Connection: InventorySystem (dropdown menu)

JDBC Driver: com.sssw.jdbc.mss.odbc.AgOdbcDriver

JDBC URL: jdbc:ssww:odbc:XCTutorial

User ID: [text input]

Password: [text input]

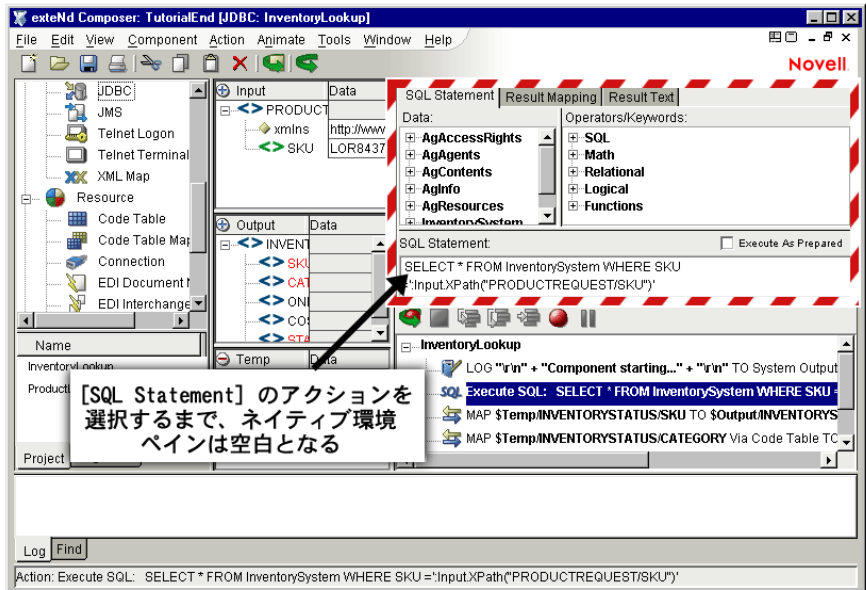
DB Params: [text input]

Deployed Pool Name: [text input]

Allow SQL Transactions:

Buttons: Test, Help, Back, Finish, Cancel

- 8 プルダウンリストで「接続」タイプを選択します。JDBC 接続の詳細については、13 ページ「JDBC 接続リソースを作成する」を参照してください。
- 9 [Finish] をクリックします。コンポーネントが作成され、JDBC コンポーネントエディタが表示されます。

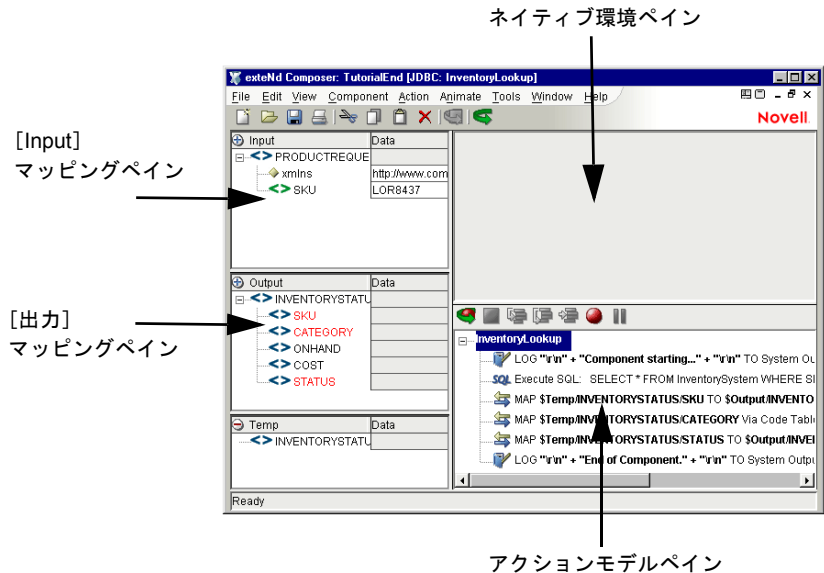


## JDBC コンポーネントエディタウィンドウについて

JDBC コンポーネントエディタには、XML Map コンポーネントエディタの機能がすべて含まれています。また、入力 XML ドキュメントと出力 XML ドキュメントのマッピングペインや、アクションペインも含まれています。

ただし、JDBC コンポーネントエディタにはすべての接続に共通なネイティブ環境ペインも含まれているという点で異なります。ネイティブ環境ペインは、SQL Statement アクションを作成するまで、グレーのペインとして表示されます。SQL ステートメントアクションを作成すると、ネイティブ環境ペインには、JDBC コネクタに固有な [Query] ペインが表示されます。

**注記：** クエリペインを表示するには、[Action] メニューから [SQL Statement] を選択して、SQL アクションを作成する必要があります。作成しない限り、ペインはグレー表示されたままの状態となります。



## クエリペインについて

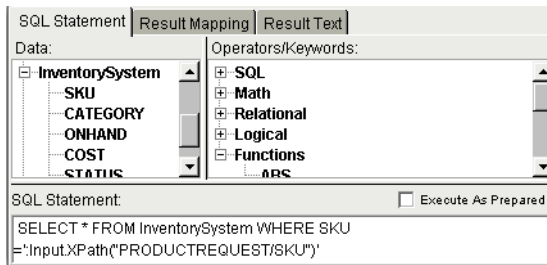
クエリペイン (アクティブなネイティブ環境ペイン) が表示されている場合、つまり、SQL Statement アクションが選択されている場合、このペインは、リアルタイムでクエリを作成およびテストするための完全に機能的な SQL 環境になります。このペインからは、次のことを実行できます。

- ◆ 入力 XML ドキュメント (または他の使用可能な DOM) からデータを取得し、そのデータを使用してリレーショナルデータソースに対して SQL クエリを作成または修正する
- ◆ クエリの結果を取得し、DOM (Temp、Output、MyDom など) に配置する

クエリペインには、[SQL Statement] タブ、[Result Mapping] タブ、および [Results Text] タブの 3 つのタブがあります。

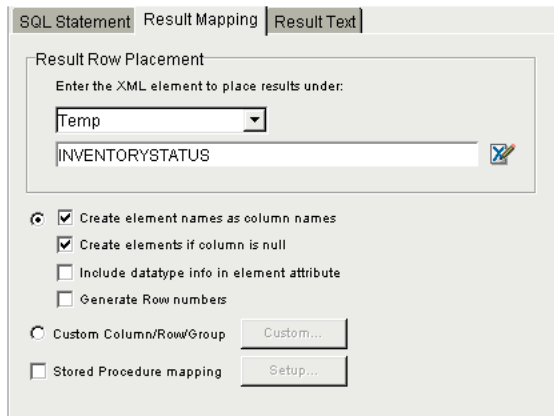
### [SQL Statement] タブ

クエリペインを開くと、ライブ SQL 環境の [SQL Statement] タブがまず表示されます。[SQL Statement] タブは、SQL コマンドを記述または作成する場所です (次の図を参照)。データツリーまたは SQL 演算子ツリー、あるいはその両方でノードをダブルクリックしたり、SQL をウィンドウの下部に直接入力したりして、ステートメントの全体または一部を作成できます。



## [Result Mapping] タブ

[Result Mapping] タブでは、データベースクエリの結果を XML ドキュメントにマップできます。また、クエリの結果を表示する正確な XML ブランチ要素を指定することもできます。[Result Mapping] タブは、次の図のとおりです。



## [Result Text] タブ

[Result Text] タブ (次を参照) には、送信された実際の SQL ステートメント、およびデータベースクエリの実行の後で返されたデータが表示されます。このタブは、一時または出力 DOM に誤ったデータが表示される場合に役立ちます。この [Result Text] タブのデータと DOM のデータを比較して、どこでエラーが発生したかを確認することができます。



SQL Statement	Result Mapping	Result Text	
SKU STATUS	CATEGORY	ONHAND	COST
---	-----	-----	----
-----			
LOR8437 Out of Stock(on re-order)	1	0	275



# 4

## JDBC アクションを実行する

### アクションについて

「アクション」は、プログラミングステートメントに類似しており、パラメータの形式で入力を受け付け、特定のタスクを実行します。『*Composer ユーザガイド*』のアクションに関する章を参照してください。

JDBC コンポーネントエディタ内では、XML ドキュメントを処理したり、非 XML データソースと通信したりするための一連の命令が、「アクションモデル」の一部として作成されます。アクションモデルは、SQL データベースと XML ドキュメント間でのすべてのデータマッピング、データ変換、データ転送、およびコンポーネントとサービス内でのデータ転送を実行します。

アクションモデルは、アクションのリストから構成されています。アクションモデル内のすべてのアクションは相互に機能します。たとえば、あるアクションモデルでは、請求書のデータをディスクから読み取り、データをインベントリデータベースから取得し、一時 XML ドキュメントに結果をマップして変換し、変換されたデータを出力 XML ドキュメントにマップする個々のアクションが含まれます。

このアクションモデルの例は、いくつかの個々のアクションから構成されています。そのアクションは次のとおりです。

- ◆ 請求書のドキュメントを開き、SQL コマンドを実行してデータベースから請求書のデータを取得する
- ◆ 結果を一時 XML ドキュメントにマップする
- ◆ コードテーブルを使用して数値コードを変換し、結果を Output XML ドキュメントにマップする

## SQL Statement アクション

SQL Statement アクションは、最も良く使用されるアクションで、既存のデータベースでクエリを実行し、結果を XML ドキュメントにマップします。しかし、SQL Data Manipulation Language (DML) ステートメントのフルセットを使用できません (データベースの挿入、削除、および更新など)。

**注記：** SQL Statement アクションで有効なクエリを作成するには、SQL データベースコマンドに精通する必要があります。

## プリペアドステートメント

JDBC Connect は、SQL コマンドを準備 (またはプリコンパイル) して、キャッシュに入れることができます。キャッシュに入れることにより、同じコマンドが何度も (たとえば、ループ内で) 実行されるたびに、必要に応じて新しい引数を挿入することで、キャッシュに入っているステートメントを再利用できます。これは、ステートメントが何度も実行される場合に、最適なパフォーマンスを実現できます。

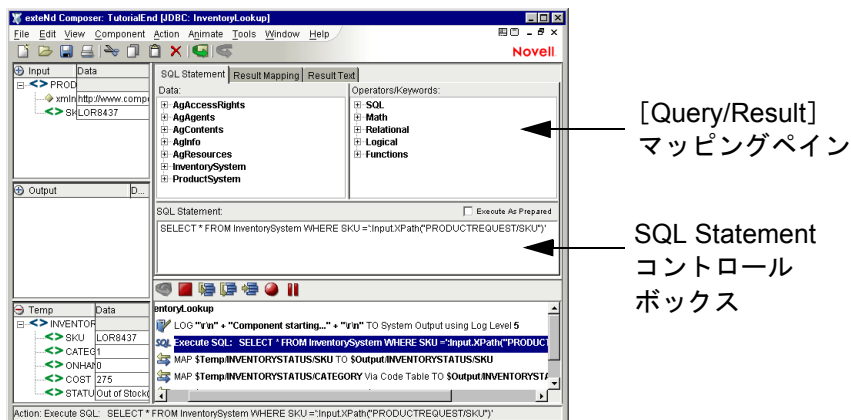
SQL 編集ボックスの上の [Query/Result Mapping] ペインにあるチェックボックスを使用して、任意のステートメントを「プリペアドステートメント」にすることができます。

Execute As Prepared

デフォルトでは、このチェックボックスはオフになっています。SQL Statement アクションがサービスのライフタイム中に一度だけ実行される場合、このチェックボックスをオフのままにしておくことをお勧めします。ループ内のステートメントでは、このチェックボックスをオンにできます。

**注記：** アプリケーションで [Execute as Prepared] チェックボックスを使用するか、またどの程度使用するかを決める基準が必要です。

JDBC コンポーネントで使用する SQL ステートメントは、[Query/Result Mapping] ペイン内で作成します。



すでに SQL Statement アクションを含むアクションモデルを作成していて、これを編集する場合、既存の SQL Statement アクションを選択（クリック）して、[Query/Result Mapping] ペインを表示することができます。作成していない場合、SQL Statement アクションを作成します。

### ➤ SQL Statement アクションを作成する

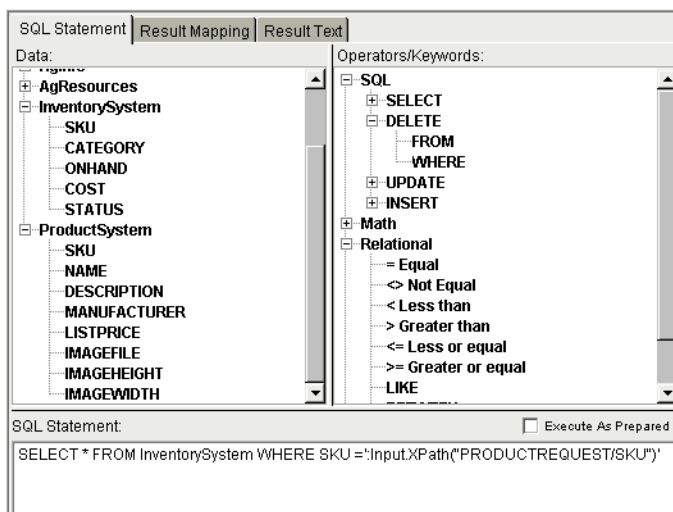
- 1 JDBC コンポーネントを作成するか、開きます。
- 2 SQL Statement を配置するアクションモデルの行を選択します。選択した行の下に新しいアクションが挿入されます。
- 3 [Action] メニューから、[New Action]、[SQL Statement] の順に選択します。[Query/Result Mapping] ペインが、[JDBC Component Editor] ウィンドウの [Native Environment] ペインに表示されます(上の図を参照)。

## SQL ステートメントを作成する

SQL ステートメントを作成するときは、データ、演算子およびキーワードをまとめます。

### ➤ SQL ステートメントを作成する

- 1 [Query/Result Mapping] ペインの [SQL Statement] コントロールボックスにカーソルを置きます。
- 2 プラス (+) 記号をクリックして、[Data] 列または [Operator/Keywords]、あるいはその両方を展開します。いくつかの親ノードが展開されている [Data and Operator/Keywords] ツリーを次の図に示します。



- 3 [SQL Statement] ボックスに追加する、各 [Data] または [Operator/Keyword]、あるいはその両方をダブルクリックします。ダブルクリックした項目は、[SQL Statement] ボックスの挿入位置に自動的に表示されます。
- 4 また、開いている DOM ツリー (たとえば、[Input DOM] ペイン) から [SQL Statement] ボックスに要素をドラッグすることもできます。
- 5 (オプション) [Execute as Prepared] チェックボックスをオンにします (詳細については、「プリペアドステートメント」を参照してください)。

## サンプルクエリを作成する

サンプルの SQL ステートメントがあります。

```
SELECT * FROM ProductSystem WHERE SKU =
':Input.XPath("PRODUCTREQUEST/SKU")';
```

このステートメントを作成するには、コンポーネントが次の条件を満たしている必要があります。

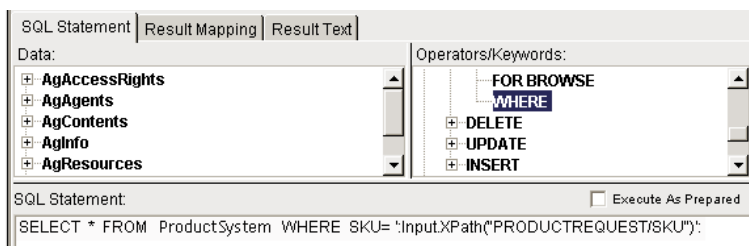
- ◆ データベースに接続する (事前に定義されている) 接続リソースを使用できる
- ◆ データベースに、SKU という名前の列のある ProductSystem という名前のテーブルがある
- ◆ SKU という名前の子要素を持つルート要素 PRODUCTREQUEST のサンプル XML ドキュメントを含むテンプレートがある

このサンプルステートメントの意味を次に示します。

「列 SKU のレコードの値と Input DOM の PRODUCTREQUEST/SKU 要素の内容が等しいデータベースの ProductSystem テーブルからすべての列を選択する」

### ▶ サンプルステートメントを作成する

- 1 Expression Builder で [SQL] ツリーを展開し、[SELECT] をダブルクリックします。
- 2 Expression Builder で [\*] をダブルクリックします。
- 3 Expression Builder で [FROM] をダブルクリックします。
- 4 「ProductSystem」を入力します。
- 5 Expression Builder で [WHERE] をダブルクリックします。
- 6 「SKU=」を入力します。
- 7 [Input DOM] で [SKU] を選択して、[SQL Statement] コントロールにドラッグします。
- 8 (オプション) [SQL Statement] の最後にセミコロン (;) を入力します。
- 9 [File]、[Save] の順に選択します。[Query/Result] ペインは次のようになります。



## ストアドプロシージャを使用する

多くの RDBMS ベンダーは、RDBMS システムに保存されているプロシージャコードを実行する機能を提供しています。これらのストアドプロシージャを使用すると、基礎となるテーブル実装とは独立した高性能インタフェースを実現できます。

ストアドプロシージャは、データアクセスのコントロールに役立ちます。ユーザのデータアクセスは、ストアドプロシージャの範囲に制限できます。ストアドプロシージャによりデータアクセスを制限すると、データが一貫した方法で入力されるので、データの整合性が保たれます。また、ストアドプロシージャを使用すると、効率性も向上されます。ストアドプロシージャは、メモリに常駐するので、実行速度が速くなります。そのため、ネットワークトラフィックも軽減されます。ストアドプロシージャは、一度だけ作成してデバッグする必要があるかもしれませんが、何度も再利用できるので、生産性も向上されます。

「プロシージャ」と「関数」という用語は、よく同じように使用されることがあるので、本書では説明のため、これらを区別します。プロシージャとは、データを返さないこともあります。呼び出しのパラメータを介して、または外部の結果セットとして使用されることもあるサブルーチンです。一方、関数は、常に「何か」を返します。プロシージャと関数は両方ともパラメータを受け渡しすることができます。

Novell exteNd Composer を使用すると、パラメータをストアドプロシージャや関数にマップし、それらを実行し、返されたデータを DOM とノードにマップすることができます。

## シンタックスの要件

プロシージャまたは関数の呼び出しを正しくまとめるため、exteNd Composer では、フォーマット規則（次の例を参照）に従う必要があります。

{ は、関数またはプロシージャへの呼び出しの開始を表します。

} は、関数またはプロシージャへの呼び出しの終了を表します。

プロシージャと関数のシンタックスは、式、プレースホルダ、または定数として使用できるパラメータをサポートしています。

**式：**プロシージャや関数の可変入力データの受け渡しに使用できます。プロシージャや関数の呼び出しでパラメータとして使用される式は、先頭にコロンを付け、単一引用符で囲みます（例：'`<variablename>`'）。

**疑問符 (?)：**パラメータとして使用され、プロシージャからのデータの送り先となるプレースホルダとして機能します。また、関数の結果で使用することもできます。

**定数：**プロシージャや関数で入力データの受け渡しに使用されますが、式やプレースホルダとは異なり、返されるデータの受け取りには使用できません。リテラル値は、単一引用符で囲みます。

## ストアドプロシージャのパラメータの規則

ストアドプロシージャには、入力パラメータ、入出力パラメータ、出力パラメータがあります。

**入力パラメータ：**データをストアドプロシージャに渡します。入力パラメータは、定数または式です。

**入出力パラメータ：**データをストアドプロシージャに渡し、ストアドプロシージャから返されるデータを受け取ります。入出力パラメータは、式でなければなりません。

**出力パラメータ：**ストアドプロシージャから返されるデータを受け取ります。出力パラメータは、プレースホルダと使用する式または疑問符です。



## JDBC コンポーネントでプロシージャおよび関数を使用する

これから示す例のため、次の手順を実行してください。

- 新しい SQL アクションを追加します。
- **[Execute as Prepared]** で「true」を設定します (チェックボックスをオンにします。19 ページ「プリペアドパラメータ」を参照してください)。

**注記:** ストアドプロシージャの結果をマップする場合、第 6 章を参照してください。

### exteNd Composer 内からプロシージャを実行する場合のシンタックス

値を返さないプロシージャ

```
{ call [<packagename>.<procedurename>]([param1,  
param2...,paramn])}
```

例:

```
{ call composerDemoPackage.sp1_withParams('12345','George') }
```

結果セットを返すプロシージャ

```
{ call [<packagename>.<procedurename>]([param1,  
param2...?....paramn])}
```

ここで、? は、結果セットが返されるパラメータです。また、結果セットは、式を含む他のパラメータに返されることもあります。

例:

```
{ call composerDemoPackage.sp_withParams('93324', ':FirstName',  
?) }
```

この例では、'93324' は定数で、':FirstName' は式で、? はプレースホルダです。

**注記:** Oracle だけは、結果セットをパラメータとして返します。Oracle 以外の RDBMS は、結果セットを返しますが、パラメータとしては返しません。

結果セットを返す Oracle プロシージャの下位互換性

バージョン 4.0 以前の exteNd Composer では、結果セットをパラメータとして返す Oracle プロシージャをサポートしていました。そのため、exteNd Composer (バージョン 4.0 以前) では、ユーザがプロシージャ呼び出し内で Oracle Cursor Position を指定する必要がありました。4.0 以前の Composer のシンタックスには、**ocp:n** がありました。ここで、**ocp** は Oracle Cursor Position で、**:n** はカーソルを含むパラメータです。このシンタックスは、4.0 以前の exteNd Composer で使用されていたので、バージョン 4.0 以降でも、下位互換性のため保持されています。

```
{ call [<packagename>.<procedurename>]([param1,
param2...ocp:x....paramn])}}
```

例:

```
{ call composerDemoPackage.sp_withParams('93324', 'Melissa',
ocp:3)}
```

**注記:** 結果セットの内容は、標準の SELECT ステートメントと同じ方法で返されます。結果は、選択された XML ドキュメントに自動的にマップされます。デフォルトでは、Output は Document で、RESULTINFO/ROW は XPath 位置です。

## Composer 内から関数を呼び出す場合のシンタックス

結果セットを返す関数

```
{ ? = [<packagename>].<functionName>([param1,
param2...,paramn])}}
```

例:

```
{ ? = call composerDemoPackage.fn_justOneReturn( ) }
```

結果セットを返す Oracle 関数の下位互換性

バージョン 4.0 以前の `exteNd Composer` との下位互換性を提供するため、4.0 以降の `exteNd Composer` でも次のシンタックスが引き続きサポートされます。

```
{ ocp:1 = [<packagename>].<functionName>([param1,
param2...,paramn])}}
```

例:

```
{ ocp:1 = call composerDemoPackage.fn_justOneReturn( ) }
```

## 特定のタスクで関数を呼び出す他の方法

SELECT ステートメント内からデータベースが更新されない任意の関数を呼び出すことがあります。

例:

```
select fn_addMin(4,6) "Sum" from dual
```

結果は返さないが、データベースを更新する関数を使用するには、結果セットを「返す」関数内から呼び出します (`fn_callAddMin` の例を参照)。

例:

```
{ ? = call composerDemoPackage.fn_callAddMin(22,44 ) }
```

## SQL ステートメントのコロン

コロンは、SQL ステートメントでは特別な文字です。これは、exteNd Composer では、コロンは、右側に ECMAScript があることを示すマーカーとして扱われるからです。上記のアクションでは、SQL ステートメントには文字列が含まれます。

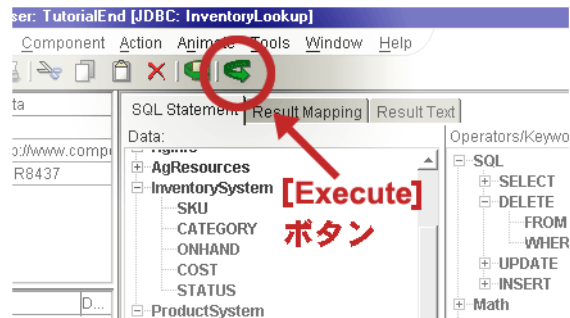
```
' :Input.XPath("PRODUCTREQUEST/SKU") '
```

このステートメントは、XPath() メソッドを含む ECMAScript 式がコロンの後に続きます。コロンがない場合、文字列は文字列リテラルとして評価されます。コロンがあると、ECMAScript 式として評価されます。

**注記：** コロンを SQL ステートメント内でリテラル値として使用する必要がある場合、バックslash をエスケープ文字として使用して、リテラルのコロンの前に付けます。このようにしないと、エラーが生じることがあります。

## SQL ステートメントを実行する

SQL ステートメントを作成したら、[Execute] ボタンをクリックします。



## 結果をチェックする

通常の行と列のフォーマットで取り出されるデータを参照して、SQL ステートメントの結果をチェックすることができます。チェックするには、[Result Text] タブをクリックします。

SQL Statement	Result Mapping	Result Text	
SKU	CATEGORY	ONHAND	COST
STATUS			
---	-----	-----	----
-----			
LOR8437	1	0	275
Out of Stock(on re-order)			

SQL ステートメントにより返されるクエリの結果が正しくないと思った場合、コンポーネントのアクションモデルの設計を続けることができます。また、[SQL Statement] タブに戻り、必要に応じて SQL をデバッグできます。

## 結果を出力 DOM にマップする

[Result Mapping] ペインを使用して、結果の行と列を配置する DOM ツリーの場所を選択します。

### ➤ [Result Mapping] を使用する

- 1 [Query/Results Mapping] ペインで [Result Mapping] タブを選択します。  
[Results Mapping] ペインが表示されます。

- 2 [Result Row Placement] で、SQL クエリの結果をマップする宛先 DOM を選択します。

3 次に、結果の各行表示する DOM 要素を選択します。該当する DOM がリストにない場合、メニューから **[File]**、**[Properties]** の順に選択して、表示される **[XML Properties Info]** ダイアログボックスを使用して別の XML テンプレートを追加します。また、**[Components]**、**[Manage]**、**[Temp Documents]** の順に選択して、Temp DOM を追加できます。

4 オプションを次のように選択します。

**[Default Result Mapping]** : 標準の列 / 行 / グループマッピングの最初のラジオボタンを選択します。

- ◆ **[Create element name as column name]**
- ◆ **[Create elements if column is null]**。返される列にデータがない場合、空の XML 要素が作成されます。
- ◆ **[Include data type info in element attribute]**。結果の列のデータタイプを示す各要素の属性が作成されます。
- ◆ **[Generate row numbers]** ( 該当する場合 )

**[Custom Result Mapping]** : 2 つめのボタン **[Custom Column/Row/Group]** を選択して、カスタムの列、行、またはグループマッピングを実行します ( 第 5 章参照 )。

**[Stored Procedure Mapping]** : ストアドプロシージャマッピングを選択し、ストアドプロシージャから返されるデータをマップします ( 第 6 章参照 )。

5 **[File]**、**[Save]** の順に選択します。

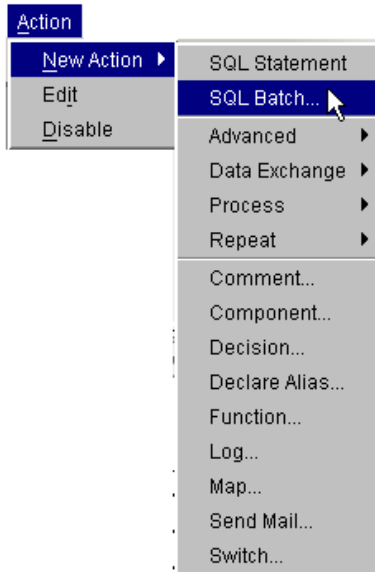
## SQL Batch アクション

ほとんどのデータベースドライバでは、接続リソースの必要性を最小化するため、SQL ステートメントをバッチで実行できます。たとえば、データベースのテーブルにデータを挿入し、別のデータベースのテーブルからデータを削除するという操作を一回で実行するとします。これは、SQL Batch アクションを使用すると、可能になります。

SQL Batch アクションを使用すると、SQL Statement アクションの特定のグループを 1 つのバッチにまとめ、一体としてデータベースに転送することができます。

**注記:** SELECT 操作は、バッチで使用できません。使用できるのは、INSERT、DELETE、および UPDATE ステートメントだけです。

SQL Batch アクションにアクセスするには、アクションペイン内を右クリックして、**[New Action]**、**[SQL Batch]** の順に選択します ( 次の図を参照 )。



[SQL Batch] コマンドは、[Start Batch]、[Execute Batch]、および [Discard Batch] の3種類あり、それぞれ新しいアクションをアクションモデルに置きます。

## Start Batch

グループ化する一連のステートメントの最初の SQL ステートメントの前に **Start Batch** ステートメントを置いて、どこからバッチが始まるかを **Composer** に示す必要があります。このコマンドは、ロールバックのチェックポイントを設定します (これは、バッチが正常に終了しなかった場合のためです)。

このコマンドの最初の指定場所から次の **Execute Batch** コマンド (次を参照) の指定場所までにある SQL ステートメントが、実行されるのではなく単にまとめられます。バッチの実行は、**Execute Batch** コマンドまで行われません。

**Map** アクションや **Function** アクションなどの通常の (SQL ではない) アクションは、**Batch** 操作の影響を受けません。**Map** アクション、**Function** アクション、またはその他の SQL 以外のアクションを、バッチされる **SQL Statement** アクションのグループの中、またはその後で置くと、これらのアクションは、バッチの SQL ステートメントの「前」で実行されます。これは、**Execute Batch** までバッチが実行されないからです。

## Execute Batch

Execute Batch コマンドを使用すると、バッチのすべての SQL ステートメントがデータベースに一体として送信されます (Execute Batch コマンドが発行されない場合、前のバッチの SQL ステートメントは実行されません)。

Execute Batch ステートメントは、SQL ステートメントアクションのバッチの直後に置くことができます。また、バッチアクション (Decision アクションのバッチ内) のダウンストリーム的位置に置くこともできます。つまり、ある場所でバッチを作成して、条件付でアクションモデルの別の場所から実行できます。

## Discard Batch

Discard Batch コマンドは、メモリの割り当てを解除するコマンドで、以前のバッチを範囲外に出します。このコマンドは、前のバッチにより使用されるメモリを解放します。

通常、SQL バッチがエラーなく実行されると、実行後にバッチは自動的に破棄されるので、明示的な破棄を発行する必要はありません。複数の連続する (それぞれに独自の Execute Batch コマンドがある) SQL バッチを含むアクションモデルを Try/On Error ステートメントで囲んでいる場合、Discard Batch を使用します。Discard Batch は、アップストリームバッチのいずれかが異常終了 (例外を生成) した場合に必要になります。別のバッチを続行するには、メモリから以前のバッチを削除する必要があります (Try アクションの On Error ブランチにある Discard Batch を使用します)。このような状況で Discard Batch を使用しないと、次の Start Batch で例外が発生します。次の図は、この場合の例を示しています。



SQL バッチが 2 つある (それぞれ Try/On Error アクションで囲まれている) 上の図の例で、Discard Batch アクションが最初の Try のエラーブランチにないと、(最初のバッチが失敗した場合) 次の Start Batch で例外が発生します。

概要：複数のバッチが連続して実行される場合、それぞれを Try/On Error アクションで囲み、「Discard Batch コマンドを各 On Error に追加します」。

SQL バッチが「1 つ」だけしかないアクションモデルには、Discard Batch は必要ありません。(1 つの) バッチの実行が正常に終了すると、そのバッチに割り当てられていたメモリが自動的に解放されます。しかし、バッチでエラーが返されると、コンポーネント自体が範囲の外に出るときに、バッチは範囲の外に出ます (ごみ箱に入ります)。

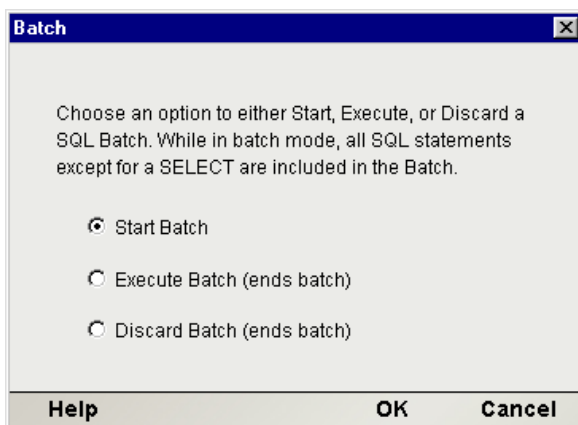
## Batch アクションを作成する

Batch アクションは、[SQL Batch] メニューコマンドを使用して作成されます。このコマンドを使用するには、JDBC Component Editor のメインメニューで [Action]、[New Action]、[SQL Batch] の順に選択するか、コンテキストメニューから [New Action]、[SQL Batch] の順に選択します。



## ➤ SQL Batch アクションを作成する

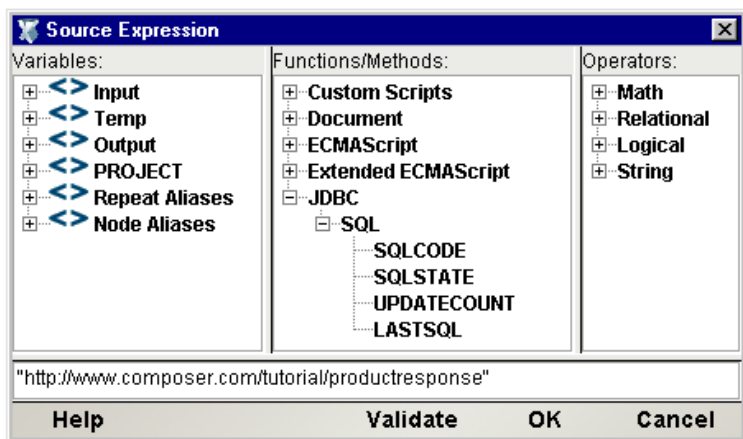
- 1 バッチする SQL Statements のグループの前にある行にカーソルを置きます。次に、マウスを右クリックして、[New Action]、[SQL Batch] の順に選択します。[Batch Setup] ダイアログボックスが表示されます。



- 2 [Start Batch] ラジオボタンをクリックして、[Start Batch] コマンドをアクションモデルに挿入します。また、必要に応じて、[Execute Batch] または [Discard Batch] を選択します。
- 3 [OK] をクリックして、ダイアログボックスを閉じます。アクションモデルに新しいアクションが表示されます。

## JDBC 固有の Expression Builder のプロパティ

SQL クエリは、特定のステータス値またはエラー値、あるいはその両方 (たとえば、Updateにより変更されたレコード数)を返すことができます。通常、ECMAScript 式でこれらの値を参照できるようにするときに役立ちます。Expression Builder ピックリスト ([Expression Editor] ウィンドウ上部) には、SQL 関連の JDBC アクション、つまり、SQLSTATE、SQLCODE、および UPDATECOUNT に固有のプロパティが表示されます (次のパネルを参照)。



## JDBC コンポーネントエディタで他のアクションを使用する

SQL Statement アクションに加えて、JDBC コンポーネントエディタでは、その他のすべてのアクションを使用できます。[Action] メニューには、基本的なアクションおよび高度なアクションの両方のリストが表示されます (次の表を参照)。

基本的なアクション	説明
コメント	アクションモデルを記録します。特に、アクションモデルに Decisions または Repeats、あるいはその両方が使用されている場合、コメントを使用して処理を明確にすることができます。
コンポーネント	別のコンポーネントを実行し、渡されるランタイム DOM を指定し、呼び出されたコンポーネントから受信します。
Decision	指定した条件に基づいて、アクションの 2 つのセットから 1 つを実行できます。コンポーネントの実行で指定した条件がどのように解決されるかによって、True または False へのパスの分岐を処理します。
Function	ECMAScript 関数または以前に作成したカスタムスクリプトのいずれかを実行します。カスタムスクリプトは、Composer のカスタムスクリプトリソースエディタを使用して作成できます (『Composer ユーザガイド』の第 9 章を参照)。
[ログ]	コンポーネントに指定されているさまざまなログファイルに情報を書き込みます。ログのタイプには、システム出力、システムログ、およびユーザログの 3 種類があります。
Map	要素のデータのある XML DOM から別の XML DOM へ転送し、オプションで変換します。

基本的なアクション	説明
メール送信	コンポーネントの実行中、指定した電子メールアドレスに自動的に電子メールを送信します。
スイッチ	入力値と大文字小文字の値との一致に基づいて、プログラムの制御をアクションの特定のブロックに分岐させることができます。これは、長く、読み取りが困難な if またはその他 (Decision アクション) のチェーンを排除するときに使用できる、基本的に便利なアクションです。

次の図の高度なアクションは、コンポーネントエディタの [Action] メニューで、[Advanced]、[Data Exchange and Repeat] の順にサブメニューを選択すると利用できます。

高度なアクション	説明
Apply Namespaces	NameSpace プリフィックスを上書きしたり、新しい NameSpace プリフィックスを宣言したり、または NameSpace 全体を無視する方法を提供します。
Raise Error	条件を評価し、true となる場合は、ERROR と呼ばれるグローバル変数に式のコンテンツを記述します。単独で使用された場合は、例外をスローしてコンポーネントを停止し、サービスに制御を返します。Try On Error アクションの Execute 分岐内で使用された場合は、評価され、On Error 分岐でアクションに制御が渡されます。
Simultaneous Components	2 つまたはそれ以上のコンポーネントを同時に (つまり、マルチスレッド方式で) 実行できるようにします。
Transaction	非コンテナ管理サービスの一部として配備されるコンポーネントで <i>User Transaction</i> コマンド (開始、コミット、およびロールバックなど) を呼び出したり、コンテナ管理 EJB 配備の一部となるコンポーネントで <i>setRollbackOnly</i> を呼び出したりできます。
Try On Error	一連のアクションを実行することで、エラーを生成するアクションに応答します。Try On Error アクションは、本質的にエラートラップおよび解決を行うアクションです。
XSLT Transform	XSL ファイルの指示に従って XML ファイルを変換します。出力は、一般的に Web ブラウザに XML ファイルを表示するために使用されます。

Data Exchange および Repeat アクションは、次の表で説明します。

Data Exchange アクション	説明
UR_/File Read	XML でないファイル形式を Composer に読み込むことができます。
UR_/File Write	ファイルを XML 以外の別の形式で書き込むことができます。
WS Interchange	WSDL リソースで定義されたメッセージおよび操作を使用して Web サービスを実行します。
XML Interchange	外部 XML ドキュメントをコンポーネントの DOM に読み込んだり、外部 XML ドキュメントにコンポーネントの DOM を書き込んだりします。読み込み / 書き込みメソッドには、ファイル、FTP、HTTP、および HTTPS プロトコルを使用した Get、Put、Post、および Post with Response が含まれます。

Repeat アクション	説明
Break	Repeat for Element、Repeat for Group、または Repeat While ループの実行を停止し、ループ外で次のアクションの実行を続行します。
続行	Repeat for Element、Repeat for Group、または Repeat While ループで現在のループ反復の実行を停止し、次の反復で同じループの一番上から続行します。
Declare Group	複数回発生する要素に基づきグループを作成して、グループに名前を付けることができます。グループは、Repeat for Group アクションで使用されます。
Repeat for Element	DOM ツリーに指定した要素が発生するごとに 1 つまたは複数のアクションを繰り返します。Repeat for Element アクションでは、複数回発生する要素に基づき、ループを作成できます。
Repeat for Group	グループの各メンバーに対して 1 つまたは複数のアクションを繰り返します。Repeat For Group アクションでは、データを再作成して、データを集約計算できます。
Repeat While	ループを作成することで、1 つまたは複数のアクションを繰り返します。Repeat While アクションでは、処理ループを任意の有効な ECMAScript 式に基づかせることができます。

## エラーおよび SQL メッセージの処理

SQL は、エラーが発生したとき (クエリでレコードが検出されなかったとき) に、または特定のアクションの結果のレポート (Update で変更されたレコード数) として、コード化された値を返します。これらの結果は、[Result Text] タブに 3 種類の特別な変数 (次を参照) として表示されます。

- ◆ SQLSTATE
- ◆ SQLCODE
- ◆ UPDATECOUNT
- ◆ LASTSQL

これらの変数は、作成する ECMAScript 関数で使用できます。また、JDBC コンポーネント内のエラー処理に使用できます。たとえば、SQL ステートメントの後に処理する Decision アクションを作成できます。UPDATECOUNT 変数で返される値に基づいて、Decision アクションの 2 つのブランチでいずれかのアクション、または他のアクションセットを選択できます。同様に、SQLSTATE または SQLCODE(標準の SQL ステータス変数)に含まれるエラー情報は、該当する回復ロジックのブランチに使用できます。

LASTSQL 変数は、該当のコンポーネントで「実際に実行される最後の SQL ステートメント」を含む exteNd 定義の文字列変数です。この変数の値を記録すると、トラブルシューティングのときに役立ちます。



# 5

## [Custom Result Mapping] を使用する

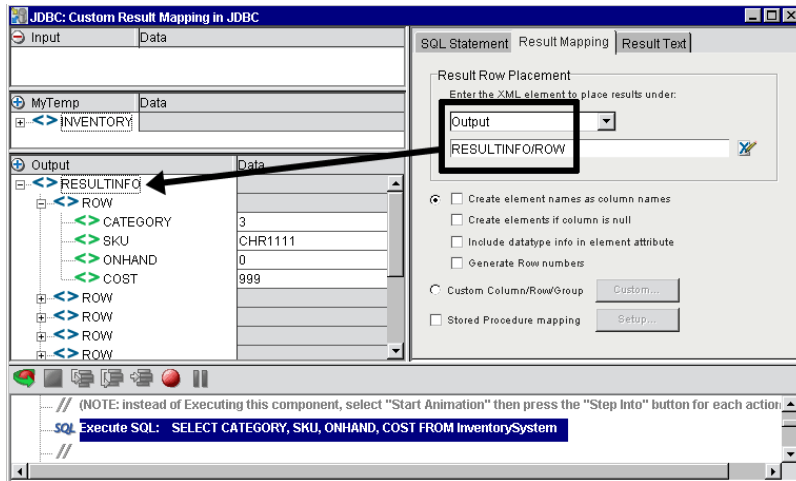
次の節では、Execute SQL アクションのデフォルト結果マッピングとカスタム結果マッピングの類似点および相違点を説明します。また、カスタムマッピング機能について詳しく説明します。

### デフォルト結果マッピングについて

Execute SQL アクションから返されるデータのマッピングは、[SQL Mapping] ペインの [Result Mapping] タブの指定により決まります。2 つの [Result Row Placement] コントロールにより、結果セットデータをターゲットドキュメントのどこに配置するかを決めることができます。ドロップダウンリストは、DOM または Repeat エイリアスコンテキストを指定し、[Expression] 編集ボックスは [Context] 内の XPath の位置を指定します。

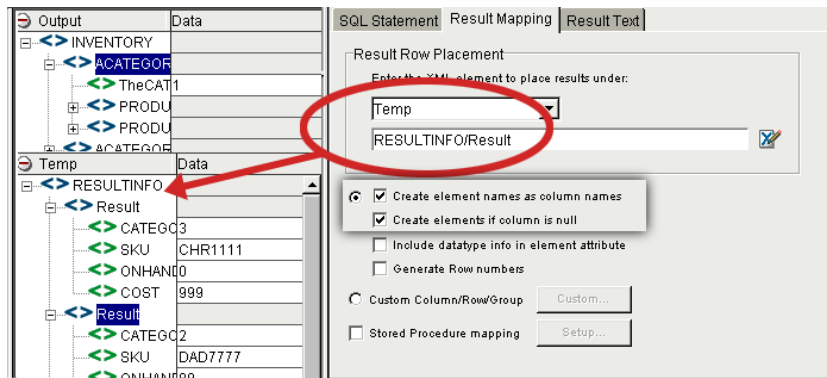
[Context] は、コンポーネントの DOM の名前、またはコンポーネントにすでに指定されている Repeat エイリアスの名前です (Repeat エイリアス自体は DOM コンテキストおよび XPath の位置を表します)。[Expression] 編集ボックスは、XPath を指定します。この最後の要素は、返される結果の親要素として機能し、データを受け取ります。データを受け取る最後の要素は、「**行ターゲット**」と呼ばれます。複数の行が返される場合、複数の行ターゲットが作成されます。行に返される各列は、各行ターゲットの子要素として表示されます。

**注記：** この節の例は、exteNd Composer インストールの Samples フォルダの「ActionExamples」プロジェクトにあります。



デフォルトでは、行ターゲットの名前は「ROW」で、ルート要素の子は「RESULTINFO」です。結果は「Output」に書き込まれます（上の図を参照）。  
 [Result Mapping] ペインのチェックボックスはオンになっていません。

結果マッピングは、任意のターゲット XPath を使用して変更できます。たとえば、  
 [Result Mapping] タブを使用して、「Temp/RESULTINFO/Result」などの行ターゲット（下の図を参照）を指定できます。



[Result Mapping] 機能のデフォルトの動作は、次のとおりです。

- ◆ ドキュメントで作成されるターゲット要素名は、結果セットで返される列名と同じ
- ◆ 結果セットで返されるすべての列は、ターゲットドキュメントにマップされる
- ◆ すべての列は、同じ親ターゲット要素にマップされる
- ◆ すべての行は 1 つのドキュメントに配置される



**注記：**XML では要素名に空白を使用できないので、列名に空白が含まれている場合、この空白は下線に変換されます。

## カスタム結果マッピングについて

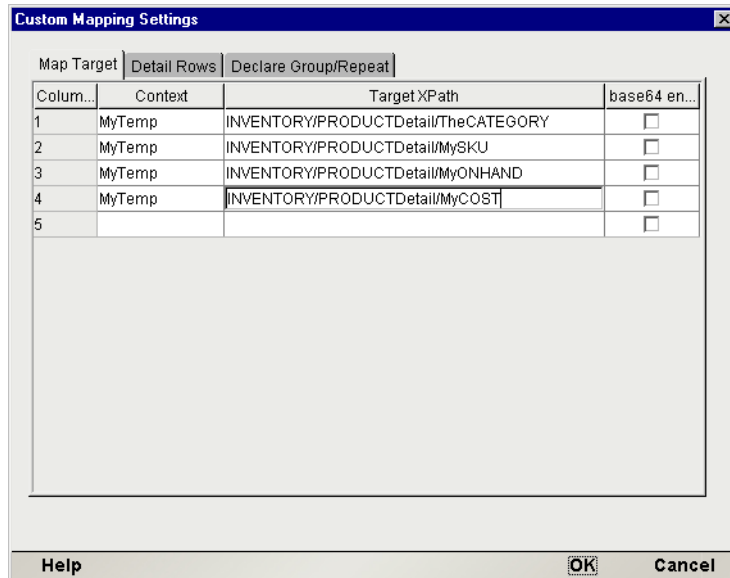
カスタム結果マッピングを使用する

- ◆ 返される列名とは異なるターゲット要素名を作成する
- ◆ 列を別の行ターゲットにマップする
- ◆ 1つ以上の列ごとに結果セットデータをグループ化する
- ◆ グループ情報だけをマップする
- ◆ グループおよび詳細情報をマップする

カスタム結果マッピングは、[Result Mapping] タブの [Custom...] ボタンからアクセスできます。



このボタンをクリックすると、[Map Target]、[Detail Rows]、および [Declare Group/Repeat] というラベルの付いた 3 種類のタブがあるダイアログボックスが表示されます。



このダイアログボックスの使用については、以下で詳しく説明します。

## カスタム結果マッピングおよびエイリアスについて

Novell exteNd Composer のデフォルトマッピング動作では、ソースドキュメントにある 1 つ以上のノード (つまり、XPath パターンで指定される要素) のリストで繰り返され、1 つのターゲットドキュメント XPath 位置にマップされます。ターゲット位置が存在しない場合、新しく作成されます。ソースリストがターゲットより大きい場合、ソースリストの各メンバー同じ物理ターゲット位置にマップする (つまり、指定の物理ターゲット位置のデータを上書きする) か、ソースリストの各メンバーの新しい物理ターゲット位置を作成する (つまり、繰り返されるソースがマップされるときに新しいターゲット位置を追加する) かを exteNd Composer に示す必要があります。ソースリストの各メンバーを同じ物理ターゲット位置にマップすることを示すには、実際の DOM 名に「Context」を指定します。ソースリストの各メンバーを「新しい」物理ターゲット位置にマップすることを示すには、別名を使用して「Context」を指定します。

**注記：** これは、Repeat for Element アクションと Repeat for Group アクションでも同じです。

SELECT ステートメントにより、XML ドキュメントの要素の繰り返しセットとして、データの行が複数返されることもあります。この場合、Declare Group アクションを使用して、Group 内で Group 要素と Detail 要素のリストを作成します。次に、Repeat for Group アクションを作成して、Group リストまたは各 Group の詳細を処理します。[Custom Map Target]、[Detail Rows]、および [Declare Group/Repeat] タブは、SQL 結果セットで行を繰り返すための別名機能を提供します。この機能は、Declare Group および Repeat for Group アクションがドキュメントで繰り返される要素に対して実行する機能と似ています。

### [MapTarget] タブを使用する

[Map Target] タブを使用すると、次のことを実行できます。

- ◆ 各結果セット列に独自のターゲット要素名を作成する
- ◆ 各結果セット列にターゲットコンテキストを作成する

[Map Target] タブは、返される「各」行の個々の列のマッピングをコントロールします。各列には、[Context] と [Target XPath] の組み合わせを指定します。[Context] と [Target XPath] の組み合わせは、Execute SQL アクションの SELECT ステートメントのプロジェクトリストに表示されている順に、各列に指定されます。[Map Target] タブに入力せずに、カスタム結果マッピングを使用することはできません。

[Context] : 列のターゲットドキュメントを指定します。[Target XPath] は、[Context] に追加され、ターゲットドキュメントの列に対する完全な XPath 位置を示します。[Context] には次のものを指定できます。

- ◆ **[Document]** : 結果セットに含まれる行が 1 つだけの場合に使用します。含まれる行が複数ある場合、行が追加されるたびに、前の行のデータが上書きされます。
- ◆ **[Detail Alias]** : **[Detail Rows]** タブで定義され、**[Document]** の名前および **[Target XPath]** の一部で構成されます。また、**[Group Alias]** ( **[Declare Group/Repeat]** タブで定義されます) と **[Target XPath]** 位置の一部で構成されることもあります。**[Detail Alias]** を使用すると、ソースリストの各メンバー (つまり結果セットの各行) に対して新しい物理ターゲット位置が作成されます。
- ◆ **[Group Alias]** : **[Declare Group/Repeat]** タブで定義され、**[Document]** の名前および **[XPath]** 位置の一部で構成されます。**[Group Alias]** を使用すると、ソースリストの各グループ (つまり結果セットの複数行) に対して新しい物理ターゲット位置が作成されます。
- ◆ **[Repeat Alias]** : Execute SQL アクションがアクションモデルの Repeat アクションで含まれる場合、そのターゲット別名を選択します。この場合、**[Context]** は、**[Document]**、および **[Target XPath]** (以下を参照) の追加先の **[XPath]** の一部になります。

詳細列データをグループ化およびマッピングする場合、**[Declare Group/Repeat]**、**[Detail Rows]**、および **[Map Target]** タブが組み合わせられて、列の完全な XPath 位置を定義します (図を参照) たとえば、**[Map Target]** タブの列は、**[Context]** と **[XPath]** で表現できます。**[Context]** は、**[Detail Rows]** タブで定義されている **[Detail Alias]** になります。次に、**[Detail Alias]** は、別の **[Context]** と **[XPath]** を表します。その **[Context]** は、**[Declare Group/Repeat]** タブで定義されている **[Detail Alias]** になります。最後に、**[Group Alias]** 自体は、別の **[Context]** と **[XPath]** を表します。

**[Group]** と **[Detail]** の別名を別々に定義すると、**[Group]** の別名をコンテキストとして使用して、複製列のデータ (グループの基本) の行を一度だけグループヘッダ要素にマップしたり、**[Context]** が **[Group Alias]** である **[Detail Alias]** を使用して、グループヘッダ内の固有のデータ (グループの詳細) で列を複数回マップすることができます。

**[Target XPath]** : 列のカスタム名を指定する XPath フラグメントです。また、任意の別の親要素が前に追加されることもあります。**[Target XPath]** には、**[Context]** が前に追加されます。これにより、ターゲットドキュメントの列に対する最終的な位置を示します。

**[Base64 encode]** : この列のチェックボックスを使用すると、DOM 要素で使用するため、バイナリデータを XML で認識できる表現に変換できます。

Map Target

Columns	Context	Target XPath	Base64 encode
1	gCATEGORY	TheCATEGORY	<input type="checkbox"/>
2	MyOutputDetail	MySKU	
3	MyOutputDetail	MyONHAND	
4	MyOutputDetail	MyCOST	
5			

Detail Rows

Detail Alias:

Representing:

Declare Group/Repeat

Group Alias	Columns	Context	Target XPath
<u>gCATEGORY</u>	CATEGORY	<u>Output</u>	<u>INVENTORY/ACATEGORYGroup</u>

[Custom Mapping Settings] ダイアログボックスの3つのタブを使用して、結果セットの項目とDOM要素間での洗練された任意のマッピングを定義できます。(それぞれ、DOMコンテキストおよびターゲットXPathを表す)ユーザー定義の別名を前のタブのコンテキストスロットで代用できる点にご注意ください。

[Map Target] タブの処理の概要は、次の表のとおりです。

SQLの結果	[Context] が [Document] の場合	[Context] が [Alias] の場合
1つの行が返された	1つの行ターゲットが、最初（および1つ）の結果行で検出または作成されます。	1つの行ターゲットが、最初（および1つ）の結果行で検出または作成されます。
複数の行が返された	1つの行ターゲットが、最初の結果行で検出または作成されます。2つめ以降の行は、「同じ」物理ターゲット位置を検出し、そこにマップします（別名を使用しないと、各行のデータは、最後の行のデータに達するまで、次の行にウツが帰されます）。	1つの行ターゲットが、「すべての」結果行で作成されます。

## MapTargetの例を参照する

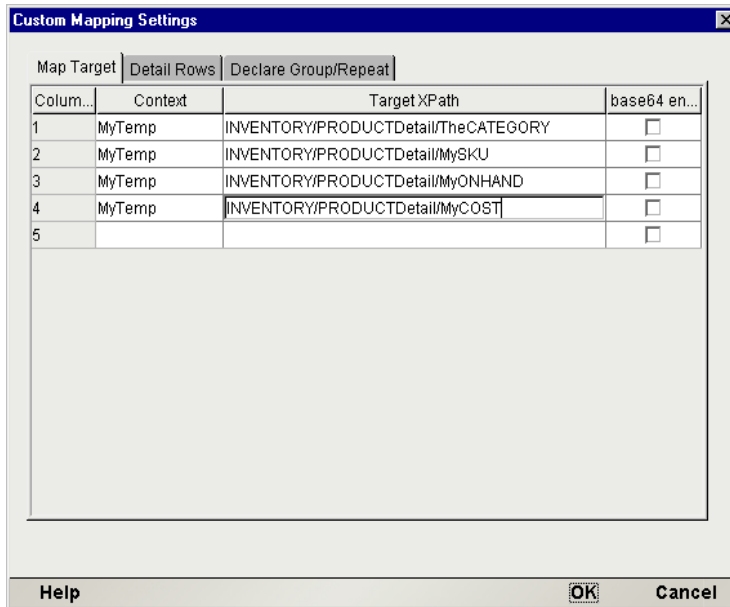
Execute SQL アクションで、次のステートメントを発行したとします。

```
SELECT CATEGORY, SKU, ONHAND, COST FROM InventorySystem
```

このステートメントにより返される行データは、次の表のとおりです。

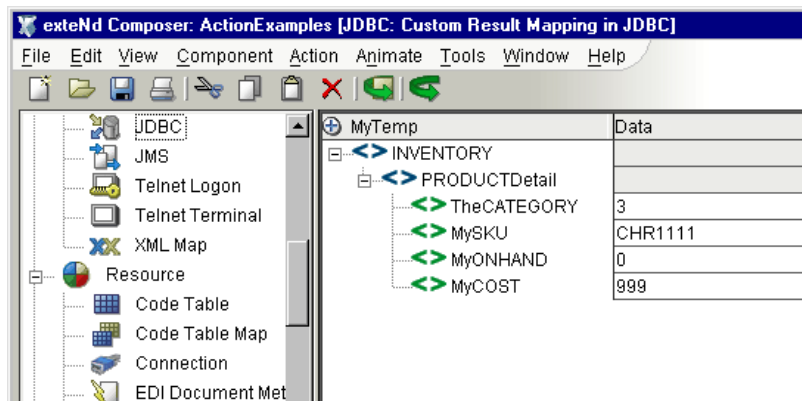
[分類]	SKU	Onhand	Cost
3	CHR1111	0	999
2	DAD7777	89	245
4	GAR1234	17	100
1	LOR8437	0	275
1	LOR8438	21	375
4	MOM4666	233	300
4	RAC4567	156	230
4	ZAC9080	4	555

[Map Target] タブは、次のように入力できます。



[Column] の [1] は、SELECT ステートメントに従って、[CATEGORY] になります。[Context] は、「MyTemp」という名前のドキュメントで、[Context] 内のターゲット XPath 位置は、「INVENTORY/PRODUCTDetail/TheCATEGORY」になります。[CATEGORY] は、[TheCATEGORY] に変更され、INVENTORY/PRODUCTDetail の親要素が前に追加されます。これと同じ理論が、残りの列には適用されます。

しかし、まだ別名を定義または使用していないので、各行の列データは、タブで指定されている同じ 4 つの物理ターゲット位置に書き込まれます。返される行が 1 行だけの場合、そのデータが、ターゲットドキュメントにマップされます。この場合、特に問題はありません。しかし、この例のように返される行が複数ある場合、最後の行のデータまで、各行のデータは、その次の行により上書きされます (これが望ましいという状況はあまりありません)。



通常、返される行が 1 行だけで、ターゲット要素の名前を列名以外に変更するだけの場合、[Map Target] タブだけを使用します。

複数の結果セット行によるデータの上書きを避けたい場合、[Detail Rows] タブで [Detail Alias] を使用して、マップされる各行に対して新しい物理ターゲット位置を作成する必要があります。

## [Detail Rows] タブを使用する

[Detail Rows] タブを使用すると、ドキュメントの [Context] または [Group/Repeat] 別名の [Context] のいずれかに結び付けられるマッピング別名を作成できます。[Detail Rows] タブは、使用しても使用しなくても構いません。

[Detail Alias] : 結果セット行のマッピング列に対して [Map Target] タブの [Context] として参照される名前です。

[Context] : 指定するドキュメント名または [Group/Repeat] 別名です。[Target XPath] は、この [Context] に追加され、ターゲットドキュメントの列に対する最終的な位置の一部を示します (残りの部分は [Map Target] タブの [Target XPath] が使用されます)。[Context] には次のものを指定できます。

- ◆ [Document] : 結果セットの各行に対して新しい物理ターゲット位置が作成されます。
- ◆ [Group Alias] : [Declare Group/Repeat] タブで定義され、[Document] の名前および [Target XPath] 位置の一部で構成されます。[Group Alias] を使用すると、各 [Group] (つまり結果セットの複数行) に属する各詳細行に対して新しい物理ターゲット位置が作成されます。

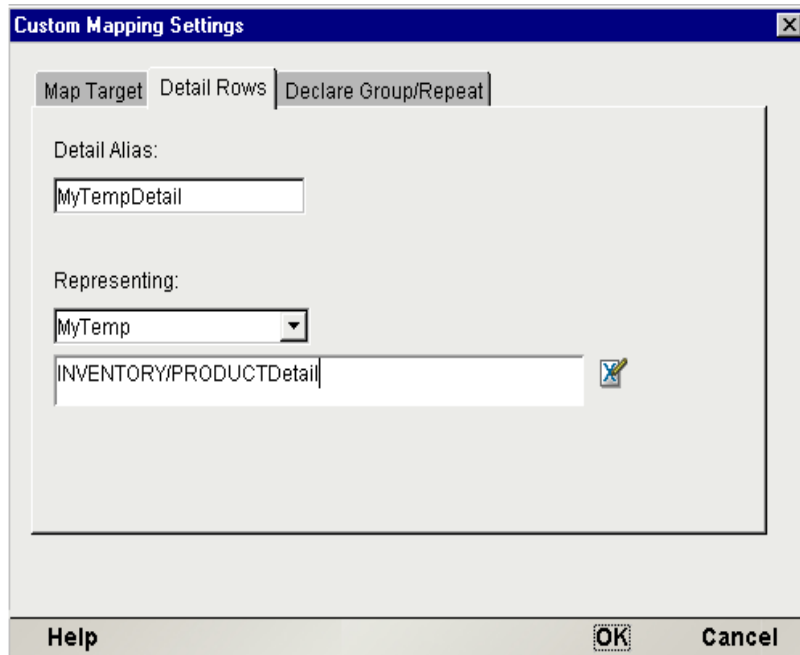
[Target XPath] : 指定する XPath フラグメントです。[Target XPath] には、このタブの [Context] が前に追加され、次に [Map Target] タブの [Target XPath] が追加されます。これにより、ターゲットドキュメントの列に対する最終的な位置を示します。

## [Detail Rows] の例を参照する

Execute SQL アクションで、次のステートメントを発行したとします。

```
SELECT CATEGORY, SKU, ONHAND, COST FROM InventorySystem
```

[Detail Rows] タブは、次のように入力できます。



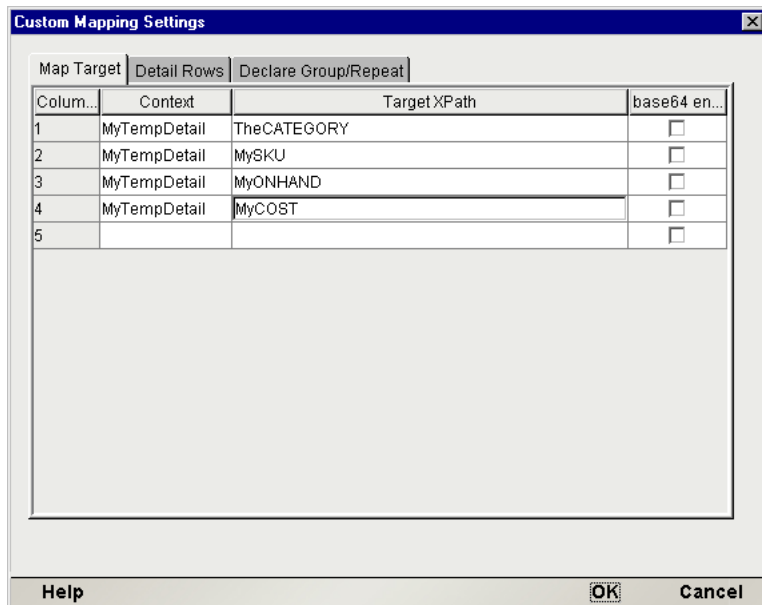
The screenshot shows a dialog box titled "Custom Mapping Settings" with three tabs: "Map Target", "Detail Rows", and "Declare Group/Repeat". The "Detail Rows" tab is selected. It contains the following fields:

- "Detail Alias:" with a text box containing "MyTempDetail".
- "Representing:" with a dropdown menu showing "MyTemp".
- A text box containing "INVENTORY/PRODUCTDetail" with a small icon to its right.

At the bottom of the dialog, there are buttons for "Help", "OK", and "Cancel".

[Context MyTemp] と [Target XPath] フラグメント [INVENTORY/PRODUCTDetail] は、[Detail Rows] タブに指定されているので (各行に新しい物理ターゲット位置が作成されます)、これらの参照は、[Map Target] タブで [Detail Alias] の「MyTempDetail」に変更する必要があります。前の節で使用した例を続けて使用し、[Map Target] タブを次のように更新します。





[Detail Rows] タブで指定された [Detail Alias] を使用すると、複数行が結果セットで返された場合、各行により、INVENTORY/PRODUCTDetail 下に新しい物理ターゲット位置が作成されます。

[Declare Group/Repeat] タブと併用しない場合、「Repeat for Row」別名の作成に [Detail Rows] タブを使用できます。[Map Target] タブの [Context for a Column] が [Detail Alias] (ドキュメントではない) の場合、行マッピングが発生するたびに、新しい [Target XPath] が作成されます。この場合、「行のデータが上書きされることなく、結果セットの複数の行により、ドキュメントの複数の行ターゲットが作成されます」。この機能は、列名の変更を除いて、[Result Mapping] タブの [Custom...] オプションで提供される機能と同じです。

→ TheCATEGORY	2
→ MySKU	DAD7777
→ MyONHAND	89
→ MyCOST	245
▶ PRODUCTDetail	
→ TheCATEGORY	4
→ MySKU	GAR1234
→ MyONHAND	17
→ MyCOST	100
▶ PRODUCTDetail	
→ TheCATEGORY	1
→ MySKU	LOR8437
→ MyONHAND	0
→ MyCOST	275
▶ PRODUCTDetail	
→ TheCATEGORY	1
→ MySKU	LOR8438
→ MyONHAND	21
→ MyCOST	375

ただし、結果セットデータは、期待通りには配置されないこともあります。たとえば、**[PRODUCTDetail]** 下のサブツリー（次の図を参照）は、製品のカテゴリ情報に関係なくリストされます。**[PRODUCTDetail/TheCATEGORY]** 以下を参照すると、2つの行がカテゴリ 1 に属し、カテゴリ 2 と 3 にはそれぞれ 1 行が属していることがわかります（この例は、Composer インストールの Sample ディレクトリ下の Action Examples プロジェクトにあります）上にスクリーンショットが作成された「JDBC でのカスタム結果マッピング」と呼ばれる JDBC コンポーネントを使用することもできます）。

カテゴリに従ってグループ化された行データを参照することもできます。この場合、**[Declare Group/Repeat]** タブから **[Group Alias]** を使用する必要があります。

## [Declare Group/Repeat] タブを使用する

**[Declare Group/Repeat]** タブを使用すると、次のことを実行できます。

- ◆ 1 つ以上の結果セット列に基いて、結果セットレコードのグループを作成する
- ◆ **[Context for Detail Rows]** として使用する **[Group Alias]** を作成する
- ◆ **[Context for Map Targets]** として使用する **[Group Alias]** を作成する（**[Group Headers]** を作成する）

[Group Alias] を宣言して、複数行の列にある固有の値で構成されるリストを作成します。この [Group Alias] を使用する任意の [Map Target] 列は、その列データを一度だけ各固有の [Group] にマップします。これにより、実質的にグループヘッダ情報が作成されます。

また、各固有のグループ値は、それに属する行のリストを参照します。[Group Alias] を使用する [Detail Rows] タブの任意の [Detail Alias] は、そのグループに対して行をまとめてマップします。

**[Group Alias]** : [Map Target] タブまたは [Detail Rows] タブ、あるいはその両方の [Context] として参照される名前です。

**[Columns]** : グループを作成する、カンマで区切られる 1 つ以上の列を指定します。2 つの列を使用すると、その 2 つの列の連結値を固有に組み合わせた場合のみ、グループが作成されます。

**注記:** 指定する列は、Execute SQL アクションの SELECT ステートメントの ORDER BY 句の基本を形成する必要があります。ORDER BY 句を省略すると、結果が予測できなくなります。

**[Context]** : コンポーネントのドキュメント名、あるいは Execute SQL アクションを含むアクションモデルにある Repeat for Group 別名または Repeat for Element 別名です。[Target XPath] は、この [Context] に追加され、ターゲットドキュメントの列に対する最終的な位置の一部を示します ( 残りの部分は [Map Target] タブの [Target XPath] が使用されます。また、[Detail Rows] タブの [Target XPath] が使用されることもあります )。[Context] には次のものを指定できます。

- ◆ [Document] : 各 [Group] に対して同じ物理ドキュメントに書き込まれます。
- ◆ [Repeat for Group Alias] : Execute SQL アクションが、アクションモデルの Repeat for Group アクション内にある場合、そのターゲット別名を各 [Group] の [Context] として使用することもできます。これにより、Execute SQL アクションのある Repeat for Group アクションで処理される各 Group に対して、新しい Group が一度作成されます。
- ◆ [Repeat for Element Alias] : Execute SQL アクションが、アクションモデルの Repeat for Element アクション内にある場合、そのターゲット別名を各 [Group] の [Context] として使用することもできます。これにより、Execute SQL アクションのある Repeat for Element アクションで処理される各繰り返し要素に対して、新しい Group が一度作成されます。

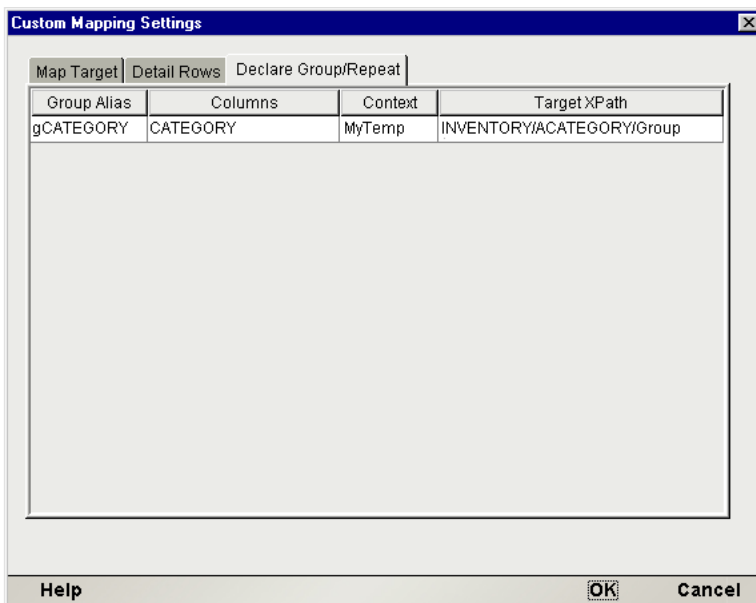
**[Target XPath]** : 指定する XPath フラグメントです。[Target XPath] には、このタブの [Context] が前に追加され、次に [Map Target] タブの [Target XPath] ( [Detail Rows] タブの [Target XPath] の場合もあります ) が追加されます。これにより、ターゲットドキュメントの列に対する最終的な位置を示します。

## [Declare Group/Repeat] の例を参照する

Execute SQL アクションで、次のステートメントを発行したとします。

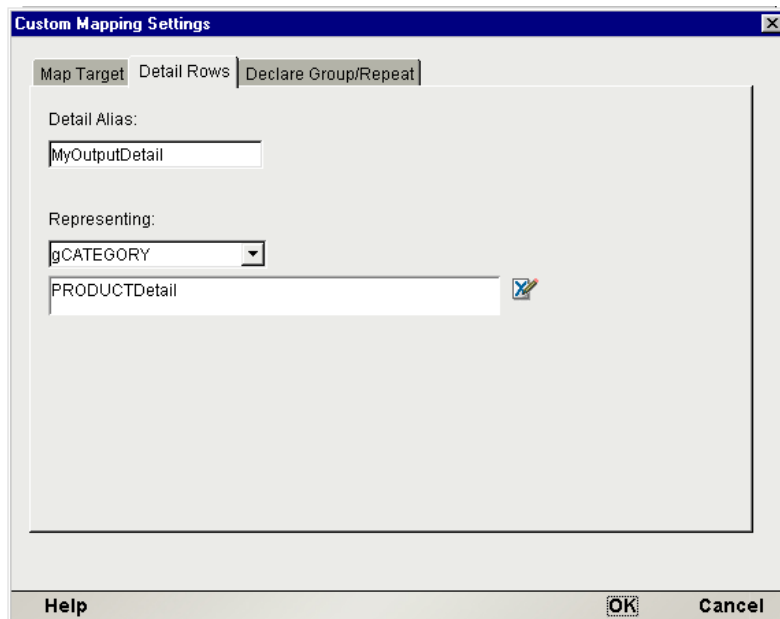
```
SELECT CATEGORY, SKU, ONHAND, COST FROM InventorySystem order  
by CATEGORY
```

[Detail Rows] タブは、次のように入力できます。



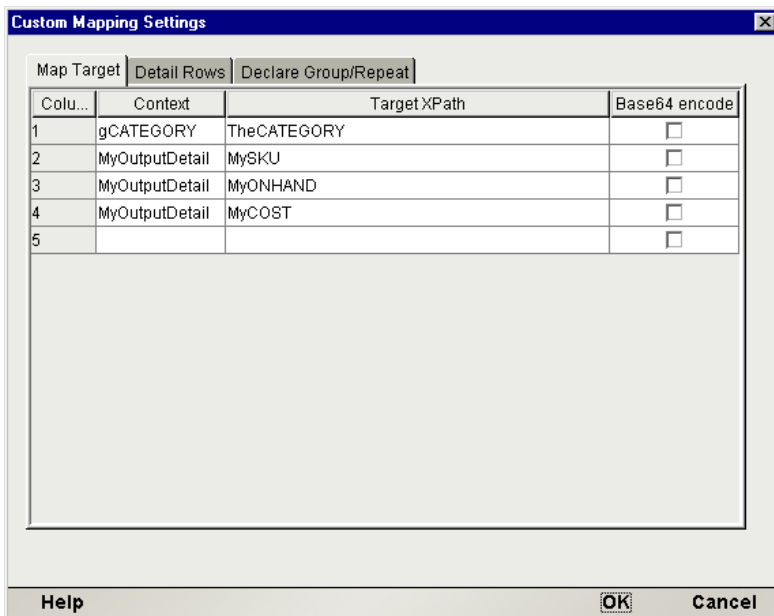
[Detail Rows] の例と同様に、[Context MyTemp] (および [Target XPath] フラグメント [INVENTORY/PRODUCT] は、[Declare Group/Repeat] タブに指定されているので、その参照は、[Detail Rows] タブで [Group Alias] の「gCATEGORY」に変更する必要があります。また、INVENTORY 下で PRODUCTDetail だけでなく、PRODUCTDetail のグループをリストしているので、新しい要素は「ACATEGORYGroup」という名前の [Group] の [Target XPath] に示されます。そのため、マップされる各 [Group] に対して、新しい ACATEGORYGroup 要素が作成されます。

前の2つの節で使用した例を続けて使用し、[Detail Rows] タブを次のように更新します。



「MyTemp」の [Context] は、MyTemp/INVENTORY/ACATEGORYGroup を表す Group Alias gCATEGORY に置き換えられているので注意してください。つまり、[Group] に属する [Detail Rows] は、すべての [Detail Rows] ではなく、マップされたものだけになります。

前の 2 つの節で使用した例を続けて使用し、[Map Target] タブを次のように更新します。



[CATEGORY] 列の [Context] は、[Group Alias] に置き換えています。つまり、[CATEGORY] は、各詳細行ではなく、各 [Group] に対して一度だけマップされます。

MyTemp	Data
INVENTORY	
ACATEGORYGroup	
TheCATEGORY	3
PRODUCTDetail	
MySKU	CHR1111
MyONHAND	0
MyCost	999
ACATEGORYGroup	
TheCATEGORY	2
PRODUCTDetail	
MySKU	DAD777
MyONHAND	89
MyCost	245
ACATEGORYGroup	
TheCATEGORY	4

[Group Alias] を宣言すると、結果セット行がスキャンされ、マッピング中にループ処理が何回発生するかを確立するグループに編成されます。結果セットの行が 8 つで、その値が 4 種類だけの場合 (たとえば、3、2、4、1、1、4、4、4)、グループマッピンググループは 4 つ (たとえば、1、2、3、4) で、その詳細ループは 8 つになります (たとえば、グループ 1 の詳細行は 2 つ、グループ 2 の詳細行は 1 つ、グループ 3 の詳細行は 1 つ、グループ 4 の詳細行は 4 つになります)。

上のグラフを使用すると、[Map Target] 列の最終コンテキストが、[Column] の [1] および [Column] の [2] に対してどのように作成されるかを追跡できます。[Column] の [1] は、結果セットの [CATEGORY] になります。DOM の名前は、[TheCATEGORY] になります。親要素は、[Declare Group/Repeat] タブの [MyTemp/INVENTORY/ACATEGORYGroup] で定義されるコンテキスト「gCATEGORY」で決まります。そのため、[CATEGORY] の最終的な [XPath] は、次のとおりです。

```
Output/INVENTORY/ACATEGORYGroup/TheCATEGORY
```

[TheCATEGORY] のコンテキストは、[Group Alias] なので、前に定義したように各グループに一度、つまり 4 回マップされます。

[Column] の [2] は、結果セットの [SKU] データになります。DOM の名前は、[MySKU] になります。親要素は、[gCATEGORY] (前の定義) および [PRODUCTDetail] に定義されるコンテキスト「MyTempDetail」で決まります。そのため、この列の最終的なコンテキストは、[MyTemp/INVENTORY/ACATEGORYGroup/PRODUCTDetail/MySKU] になります。[MySKU] のコンテキストは [Detail Alias] なので、各 [Detail Row] に対して一度マップされます。ただし、各 [Detail Row] には、[Group Alias] の [Context] があるので、マッピングが [Group] に属する詳細行にだけ制限されます。





# 6

## ストアドプロシージャ

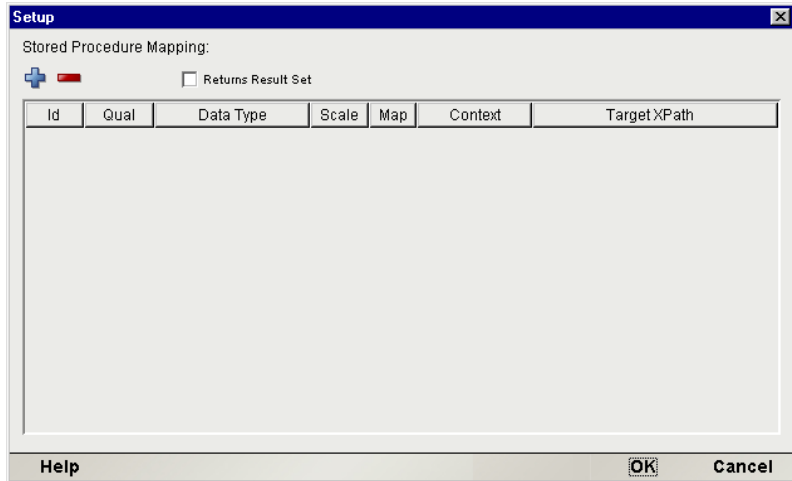
Novell exteNd Composer は、ストアドプロシージャによって返されるデータのマッピングをサポートしています。次の節では、ストアドプロシージャのマッピング機能について説明します。

### ストアドプロシージャのマッピングについて

Novell exteNd Composer では、ストアドプロシージャによって返されるデータを DOM とノードの組み合わせにマップできます。マップするには、[Query/Results Mapping] ペインの [Result Mapping] タブにある [Stored Procedure mapping] チェックボックスを選択します。

The screenshot shows the 'Result Mapping' tab in the Novell exteNd Composer interface. It features a 'Result Row Placement' section with a dropdown menu set to 'Temp' and a text input field containing 'INVENTORYSTATUS'. Below this are several checkboxes: 'Create element names as column names' (checked), 'Create elements if column is null' (checked), 'Include datatype info in element attribute' (unchecked), and 'Generate Row numbers' (unchecked). At the bottom, the 'Stored Procedure mapping' checkbox is checked, and the 'Setup...' button is highlighted.

[Setup...] ボタンが使用できるようになります。[Setup...] ボタンを押します。[Stored Procedure Mapping] の [Setup] ダイアログボックスが表示されます。



## 規則を作成する

定数（「abc」など）ではなく、「式」や「疑問符」プレースホルダにより表されるすべてのプレースホルダ（「:<expression>」または「?」のいずれか）と exteNd Composer の関係を理解するは重要です。

## 【Stored Procedure Mapping Setup】ダイアログボックスを使用する

【Stored Procedure Mapping Setup】ダイアログボックスは、ストアードプロシージャにより返されるデータをマップするときに使用されます。この設定ダイアログボックスを使用すると、返されるデータに [Context] と [Target XPath] の組み合わせを指定できます。

[+] および [-] コントロールを使用して、[Context] と [Target XPath] の組み合わせを追加および削除します。

Oracle RDBMS では、結果セットがパラメータとして返されます。Oracle 以外の RDBMS は、結果セットを返しますが、パラメータとしては返しません。Oracle 以外の RDBMS により結果セットが返される場合、[Returns Result Set] チェックボックスを選択します。Oracle 以外の RDBMS に対して [Returns Result Set] チェックボックスを選択すると、返される結果セットを exteNd Composer で検出できるようになります。

**注記：** 返されるデータを正しくマップするため、すべての式およびプレースホルダ (? など) を 【Stored Procedure Mapping Setup】ダイアログボックスに指定する必要があります。

返される各入出力パラメータ ( 式 )、および各出力パラメータ ( 式または ? ) に対して、次のことを実行します ( 第 4 章「ストアードプロシージャの規則」を参照してください)。

[**Id**] : SQL パラメータに基いた、戻り値の数値シーケンスです。[**Id**] を使用する場合、式 (:<ExpressionName> など) またはプレースホルダ (? など) のいずれかを含む各パラメータのシーケンス位置を明示的に指定する必要があります。たとえば、次のプロシージャ呼び出しには、定数「Process」、プレースホルダ「?」、式「:Smith」の 3 つのパラメータがあります。exteNd Composer は値にバインドしないため、値「Process」は、[**Stored Procedure Mapping**] ペインの [**Id**] で指定する必要はありません。プレースホルダ「?」および変数「:Smith」に対する [**Id**] エントリは、それぞれ 2 と 3 です。exteNd Composer は、変数やプレースホルダにはバインドするので、ストアードプロシージャにより返されるデータを正しくマップするため、これらを [**Stored Procedure Mapping**] ペインで指定する必要があります。

例:

```
{ call DemoPackage.sp_withParams('Process', ?, ':Smith') }
```

[**Qual**] : パラメータを入力パラメータ、出力パラメータ、または入出力パラメータで修飾します。

[**Data Type**] : [VARCHAR]、[DECIMAL]、[DATE]、[BINARY]、[Oracle Result Set] のオプションがあるドロップダウンリストです。[**Oracle Result Set**] が選択されている場合、[Context] および [Target XPath] は、適用されないので、使用できません。

[**Scale**] : [Scale] の値は、小数点の精度を指定します。

[**Map**] : パラメータをマップするときに選択します。

[**Context**] : 列のターゲットドキュメントを指定します。[Target XPath] は、[Context] に追加され、ターゲットドキュメントの列に対する完全な XPath 位置を示します。[Context] には次のものを指定できます。

- ◆ [**Document**] : 結果セットに含まれる行が 1 つだけの場合に使用します。含まれる行が複数ある場合、行が追加されるたびに、前の行のデータが上書きされます。
- ◆ [**Detail Alias**] : [Detail Rows] タブで定義され、[Document] の名前および [Target XPath] の一部で構成されます。また、[Group Alias] ( [Declare Group/Repeat] タブで定義されます) と [Target XPath] 位置の一部で構成されることもあります。[Detail Alias] を使用すると、ソースリストの各メンバー (つまり結果セットの各行) に対して新しい物理ターゲット位置が作成されます。

- ◆ `[Group Alias]` : `[Declare Group/Repeat]` タブで定義され、`[Document]` の名前および `[XPath]` 位置の一部で構成されます。`[Group Alias]` を使用すると、ソースリストの各グループ (つまり結果セットの複数行) に対して新しい物理ターゲット位置が作成されます。
- ◆ `[Repeat Alias]` : `Execute SQL` アクションが、アクションモデルの `Repeat` アクションに含まれる場合、そのターゲット別名を選択します。この場合、`[Context]` は、`[Document]`、および `[Target XPath]` (以下を参照) の追加先の `[XPath]` の一部になります。
- ◆ `-- via standard --` : `[Result Mapping]` タブの `[Result Row Placement]` 指定が使用されます。
- ◆ `-- via custom --` : `[Custom Mapping Settings]` ダイアログボックスの設定が使用されます。

詳細列データをグループ化およびマッピングする場合、`[Declare Group/Repeat]`、`[Detail Rows]`、および `[Map Target]` タブが組み合わされて、列の完全な `XPath` 位置を定義します (図を参照) たとえば、`[Map Target]` タブの列は、`[Context]` と `[XPath]` で表現できます。`[Context]` は、`[Detail Rows]` タブで定義されている `[Detail Alias]` になります。次に、`[Detail Alias]` は、別の `[Context]` と `[XPath]` を表します。その `[Context]` は、`[Declare Group/Repeat]` タブで定義されている `[Detail Alias]` になります。最後に、`[Group Alias]` 自体は、別の `[Context]` と `[XPath]` を表します。

`[Group]` と `[Detail]` の別名を別々に定義すると、`[Group]` の別名をコンテキストとして使用して、複製列のデータ (グループの基本) の行を一度だけグループヘッダ要素にマップしたり、`[Context]` が `[Group Alias]` である `[Detail Alias]` を使用して、グループヘッダ内の固有のデータ (グループの詳細) で列を複数回マップすることができます。

`[Target XPath]` : `XPath` フラグメントで、`[Context]` に追加され、ターゲットドキュメントの完全な `XPath` 位置を示します。

## 返される結果セット

結果セットは、その列名から作成された要素でドキュメントにマップされます。

- ◆ ドキュメントで作成されるターゲット要素名は、結果セットで返される列名と同じ
- ◆ 結果セットで返されるすべての列は、ターゲットドキュメントにマップされる
- ◆ すべての列は、同じ親ターゲット要素にマップされる
- ◆ すべての行は 1 つのドキュメントに配置される

**注記:** XML では要素名に空白を使用できないので、列名に空白が含まれている場合、この空白は下線に変換されます。

# A

## JDBC 用語集

### Custom Result Mapping

[Custom Result Mapping] ダイアログボックスは、SQL 結果セットで行を繰り返すための別名機能を提供します。この機能は、Declare Group および Repeat for Group アクションがドキュメントで繰り返される要素に対して実行する機能と似ています。

### [Declare Group/Repeat] タブ

このタブは、[Custom Results Mapping] ダイアログボックスにあり、1 つ以上の結果セット列の結果セットレコードのグループ作成、Context for Detail Rows として使用する Group Alias の作成、および Context for Map Targets として使用する Group Alias の作成 (Group Headers の作成) に使用されます。

### [Detail Rows] タブ

このタブは、[Custom Results Mapping] ダイアログボックスにあり、使用すると、ドキュメントの [Context] または [Group/Repea] 別名の [Context] のいずれかに結び付けられるマッピング別名を作成できます。[Detail Rows] タブは、使用しても使用しなくても構いません。

### DOM

ドキュメントオブジェクトモデル (DOM) とは、ソフトウェアプログラムのメモリ内でオブジェクトとして作成された XML ドキュメントです。これにより、オブジェクトを操作するための標準的な方法が提供されます。Composer では、DOM は、XML ドキュメントと同義場合があります。DOM は、単一のルートノードがある階層的なツリーとして表されます。

### DOM コンテキスト

コンポーネントで事前に定義されている DOM (入力、出力、一時など) の名前、つまり [Repeat] 別名の名前 (別名自体は、要素のノードパス階層のアップストリームを示す、DOM コンテキストを表します)。

### Execute SQL アクション

SQL Statement アクションと同じです。

## JDBC

リレーショナルデータベースのデータにアクセスする Java API の Sun の商標。通常、Java Database Connectivity と見なされます。

### [Map Target] タブ

このタブは、[Custom Results Mapping] ダイアログボックスにあり、各結果セット列のターゲット要素名の作成、および各結果セット列のターゲットコンテキストの指定に使用されます。

### [Query/Result Mapping] ペイン

(ネイティブ環境ペインと同じです。) JDBC コンポーネントエディタにあるペインで、[SQL Statement] タブ、[Result Mapping] タブ、および [Results Text] タブの 3 種類のタブがあります。

### [Result Mapping] タブ

[Query/Result Mapping] ペインのタブで、使用すると、データベースクエリの結果を XML ドキュメントにマップできます。

### [Result Text] タブ

[Query/Result Mapping] ペインにあるタブで、データベースクエリの実行に従って返される実際のデータを表示します。

## SQL Statement アクション

最も良く使用されるアクションで、既存のデータベースでクエリを実行し、結果を XML ドキュメントにマップします。

### [SQL Statement] タブ

[Query/Result Mapping] ペインにあるタブで、SQL コマンドを作成することができます。

## SQLCODE

SQL ステートメントの実行により作成されるグローバル ECMAScript 変数。データベースエンジンで生成されるステータスコードを含みます。

## SQLSTATE

SQL ステートメントの実行により作成されるグローバル ECMAScript 変数。データベースエンジンで生成される情報を含みます。

## UPDATECOUNT

SQL ステートメントの実行により作成されるグローバル ECMAScript 変数。データベースエンジンで変更された行数を含みます。

## 行ターゲット

マッピング操作での要素の受け取りは、「行ターゲット」と呼ばれます。これは、XML ファイルの DOM ツリーにおける特定の位置を表します。

## 接続プール

アプリケーションサーバが管理する様々なアプリケーションに対して、そのアプリケーションサーバにより管理されるデータベースの接続セット。

## ネイティブ環境ペイン

JDBC コンポーネントエディタにあるペインで、クエリ発行時に実際の SQL 環境をシミュレートします。





# B

## 予約語

次の用語は、JDBC Connect の exteNd Composer の予約語であるため、ユーザ作成用語またはユーザ作成オブジェクトでは使用しないでください。

- ◆ SQLCODE
- ◆ SQLSTATE
- ◆ UPDATECOUNT
- ◆ LASTSQL



# 索引

## A

[Action] メニュー 42  
Allow SQL Transactions 16

## B

base64 encode 51

## C

Custom Mapping Settings 68  
Custom Result Mapping  
  定義 69

## D

Data Exchange アクション 44  
[Data Type] 67  
DB Param 16  
[Declare Group/Repeat] タブ 58  
  定義 69  
[Declare Group/Repeat] の例 60  
[Detail Rows] タブ  
  定義 69  
Discard Batch 39

## E

ECMAScript  
  SQL ステートメント 35  
ECMAScript 関数、使用 45  
Execute Batch 39  
Execute SQL アクション  
  定義 69

## I

Id 67

## J

JDBC  
  XML テンプレートを作成する 18  
  概要 10  
  機能 11  
  定義 70  
JDBC コンポーネント  
  about 11  
  新規作成 19  
JDBC コンポーネントエディタ  
  アプリケーションを作成する 12  
  ウィンドウについて 22  
JDBC 接続プール 14  
JDBC 接続リソース 13  
JDBC ドライバ 14

## L

LASTSQL 45

## M

[Map Target] タブ 50  
  定義 70

## O

Oracle Result Set 67

## Q

Qual 67  
[Query/Result Mapping] ペイン 28  
  定義 70

## R

- Repeat アクション 44
- Result Mapping 68
  - [Result Mapping] タブ 24
    - 定義 70
  - [Result Text] タブ 24
    - 定義 70

## S

- S3SqlAnywhereAuth 16
- Scale 67
- SQL
  - トランザクションバンプ 16
  - プリペアドステートメント 28
- SQL Anywhere 16
- SQL Batch アクション 37
- SQLCODE 45
  - 定義 70
- SQLSTATE 45
  - 定義 70
- SQL Statement アクション
  - 定義 70
- [SQL Statement] タブ 23
  - 定義 70
- SQL ステートメント
  - 結果をチェックする 35
    - 作成 29
    - 実行 35
- SQL ステートメントを実行する 35
- SQL バッチの範囲 40
- SQL メッセージ 45
- Start Batch 38

## T

- Try/On Error 39

## U

- UPDATECOUNT 45
  - 定義 70

## X

- XML テンプレート
  - 作成 18

## あ

- アクション
  - 概要 27
    - 基本的なアクションおよび高度なアクションを使用する 42
  - アクションモデル 27

## え

- エラーおよび SQL メッセージ 45

## か

- カスタム結果マッピング 49, 50
- カスタムスクリプト
  - 作成 18

## き

- 基本的なアクション 42
- 行ターゲット 47, 48

## く

- クエリ / 結果マッピングペイン 23
- クエリ、サンプルクエリを作成する 30
- 繰り返し別名
  - コンテキストとして使用する 51, 68
  - 作成 58
- グループ別名
  - コンテキストとして使用する 51, 68
  - 作成 58

## け

- 結果マッピング
  - カスタムを使用する 49
  - デフォルトを使用する 47

## こ

- 高度なアクション 42, 43
- コードテーブルマップ、作成 18
- コミット 17
- コロン、SQL アクションでの特別な意味 35
- コンテキスト 50, 58
- コンポーネント
  - 新規作成 19
- コンポーネントエディタウィンドウ 22

## さ

- サンプルクエリ 30

## し

- 式 66
- 自動コミット 17
- 詳細行の例 56
- 詳細別名
  - コンテキストとして使用する 51, 67

## す

- ストアドプロシージャのマッピング 65

## せ

- 接続
  - 作成 13
  - 未完了 17
- 接続プール 14
  - 定義 71

## た

- ターゲット XPath 50, 51, 58, 68
- ターゲット要素名 50

## て

- 定数駆動型および式駆動型の接続 13
- データベース固有のパラメータ 16

- デフォルト結果マッピング 47

## と

- ドキュメント、コンテキストとして使用する 51, 67
- トランザクション
  - SQL 16
  - 自動コミットフラグ 17

## ね

- ネイティブ環境ペイン
  - 定義 71

## は

- バッチアクション (SQL Batch を参照) 37

## ふ

- プリコンパイル済み SQL 28
- プリペアド SQL ステートメント 28
- プリペアドとして実行する 28

## へ

- 別名
  - カスタム結果マッピング 50

## ま

- マップターゲット
  - 例 53

## ろ

- ロールバック 17

