

Oracle® Fusion Middleware

Data Modeling Guide for Oracle Business Intelligence Publisher

11g Release 1 (11.1.1)

E22258-01

December 2011

Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher, 11g Release 1 (11.1.1)

E22258-01

Copyright © 2010, 2011, Oracle and/or its affiliates. All rights reserved.

Primary Author: Leslie Grumbach Studdard

Contributor: The Oracle BI Publisher development, quality assurance, and product management teams.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	vii
Intended Audience.....	vii
Documentation Accessibility	vii
Related Documents	vii
Conventions	viii
 New Features for Data Model Designers	ix
New Features for Oracle BI Publisher 11g Release 1 (11.1.1.6).....	ix
New Features for Oracle BI Publisher 11g Release 1 (11.1.1.5)	x
New Features for Oracle BI Publisher 11g Release 1 (11.1.1.3).....	xi
 1 Introduction to the Oracle Business Intelligence Publisher Data Modeling Guide	
1.1 Introduction to Oracle Business Intelligence Publisher	1-1
 2 Using the Data Model Editor	
2.1 What Is a Data Model?	2-1
2.2 Components of a Data Model	2-1
2.3 Features of the Data Model Editor	2-2
2.4 About the Data Source Options	2-2
2.5 Process Overview for Creating a Data Model	2-3
2.6 Launching the Data Model Editor	2-4
2.7 About the Data Model Editor Interface	2-4
2.8 Setting Data Model Properties	2-5
2.8.1 XML Output Options	2-6
2.8.2 Attachments to the Data Model	2-6
2.8.2.1 Attaching Sample Data	2-6
2.8.2.2 Attaching Schema	2-6
2.8.2.3 Data Files	2-7
 3 Creating Data Sets	
3.1 Overview of Creating Data Sets.....	3-1
3.2 Editing an Existing Data Set	3-3
3.3 Creating a Data Set Using a SQL Query	3-3

3.4	Using the Query Builder	3-4
3.4.1	Understanding the Query Builder Process	3-5
3.4.2	Using the Object Selection Pane	3-5
3.4.3	Selecting a Schema	3-5
3.4.4	Searching and Filtering Objects	3-5
3.4.5	Selecting Objects	3-5
3.4.6	Supported Column Types	3-5
3.4.7	Adding Objects to the Design Pane	3-6
3.4.8	Resizing the Design and Results Pane	3-6
3.4.9	Removing or Hiding Objects in the Design Pane	3-6
3.4.10	Specifying Query Conditions	3-6
3.4.11	Creating Relationships Between Objects	3-8
3.4.11.1	About Join Conditions	3-8
3.4.11.2	Joining Objects Manually	3-8
3.4.12	Saving a Query	3-9
3.4.13	Adding a Bind Variable to a Query	3-10
3.4.14	Editing a Saved Query	3-12
3.5	Creating a Data Set Using an MDX Query Against an OLAP Data Source	3-12
3.6	Creating a Data Set Using an LDAP Query	3-13
3.7	Creating a Data Set Using a Microsoft Excel File	3-14
3.7.1	About Supported Excel Files	3-14
3.7.2	Guidelines for Accessing Multiple Tables per Sheet	3-15
3.7.3	Using a Microsoft Excel File Stored in a File Directory Data Source	3-16
3.7.4	Uploading a Microsoft Excel File Stored Locally	3-17
3.7.4.1	Refreshing and Deleting an Uploaded Excel File	3-18
3.8	Creating a Data Set Using an Oracle BI Analysis	3-19
3.8.1	Additional Notes on Oracle BI Analysis Data Sets	3-20
3.9	Creating a Data Set Using a View Object	3-20
3.9.1	Additional Notes on View Object Data Sets	3-21
3.10	Creating a Data Set Using a Web Service	3-21
3.10.1	Adding a Simple Web Service: Example	3-22
3.10.2	Adding a Complex Web Service	3-24
3.10.3	Additional Information on Web Service Data Sets	3-26
3.11	Creating a Data Set Using a Stored XML File	3-26
3.11.1	Additional Information on File Data Sets	3-27
3.12	Using Data Stored as a Character Large Object (CLOB) in a Data Model	3-27
3.12.1	How the Data Is Returned	3-28
3.12.1.1	Additional Notes on Data Sets Using CLOB Column Data	3-29
3.12.2	Handling XHTML Data Stored in a CLOB Column	3-29
3.12.2.1	Retrieving XHTML Data Wrapped in CDATA	3-30
3.12.2.2	Wrapping the XHTML Data in CDATA in the Query	3-30
3.13	Creating a Data Set from an HTTP XML Feed	3-31
3.13.1	Additional Information on Data Sets Created from HTTP XML Feed	3-32
3.14	Testing Data Models and Generating Sample Data	3-32
3.15	Including User Information Stored in System Variables in Your Report Data	3-33
3.15.1	Adding the User System Variables as Elements	3-34
3.15.2	Sample Use Case: Limit the Returned Data Set by User ID	3-34

3.15.2.1	Creating Bind Variables from LDAP User Attribute Values	3-34
3.15.2.1.1	Prerequisite.....	3-34
3.15.2.1.2	How BI Publisher Constructs the Bind Variable	3-35

4 Structuring Data

4.1	Working with Data Models	4-1
4.1.1	About Multipart Unrelated Data Sets.....	4-1
4.1.2	About Multipart Related Data Sets	4-2
4.1.3	Guidelines for Working with Data Sets.....	4-3
4.2	Features of the Data Model Editor	4-4
4.3	About the Interface	4-5
4.4	Creating Links Between Data Sets.....	4-7
4.4.1	About Element-Level Links.....	4-7
4.4.2	About Group-Level Links.....	4-7
4.5	Creating Element-Level Links.....	4-8
4.5.1	Deleting Element-Level Links.....	4-9
4.6	Creating Group-Level Links.....	4-10
4.6.1	Deleting Group-Level Links.....	4-11
4.7	Creating Subgroups.....	4-11
4.8	Moving an Element Between a Parent Group and a Child Group	4-12
4.9	Creating Group-Level Aggregate Elements.....	4-13
4.10	Creating Group Filters	4-16
4.11	Performing Element-Level Functions	4-18
4.12	Setting Element Properties	4-18
4.13	Sorting Data	4-19
4.14	Performing Group-Level Functions	4-20
4.14.1	The Group Action Menu.....	4-20
4.14.2	Editing the Data Set.....	4-21
4.14.3	Removing Elements from the Group	4-21
4.14.4	Editing the Group Properties.....	4-21
4.15	Performing Global-Level Functions	4-22
4.15.1	Adding a Global-Level Aggregate Function.....	4-22
4.15.2	Adding a Group-Level or Global-Level Element by Expression.....	4-24
4.15.3	Adding a Global-Level Element by PL/SQL.....	4-25
4.16	Using the Structure View to Edit Your Data Structure	4-26
4.16.1	Renaming Elements.....	4-26
4.16.2	Adding Value for Null Elements.....	4-27
4.17	Function Reference	4-27

5 Adding Parameters and Lists of Values

5.1	About Parameters	5-1
5.2	Adding a New Parameter	5-1
5.2.1	Defining a Text Parameter.....	5-3
5.2.2	Defining a Menu Parameter	5-4
5.2.2.1	Customizing the Display of Menu Parameters	5-7
5.2.3	Defining a Date Parameter	5-7

5.3	About Lists of Values	5-8
5.4	Adding Lists of Values	5-8
5.4.1	Creating a List from a SQL Query	5-9
5.4.2	Creating a List from a Fixed Data Set	5-10

6 Adding Event Triggers

6.1	About Triggers	6-1
6.2	Adding Before Data and After Data Triggers.....	6-1
6.2.1	Order of Execution	6-2
6.3	Creating Schedule Triggers	6-2

7 Adding Flexfields

7.1	About Flexfields	7-1
7.2	Adding Flexfields.....	7-2
7.2.1	Entering Flexfield Details	7-2

8 Adding Bursting Definitions

8.1	About Bursting	8-1
8.2	What is the Bursting Definition?.....	8-2
8.3	Adding a Bursting Definition to Your Data Model	8-2
8.4	Defining the Query for the Delivery XML	8-4
8.5	Passing a Parameter to the Bursting Query	8-7
8.6	Defining the Split By and Deliver By Elements for a CLOB/XML Data Set.....	8-9
8.7	Configuring a Report to Use a Bursting Definition	8-11
8.8	Sample Bursting Query	8-11
8.9	Creating a Table to Use as a Delivery Data Source.....	8-11

Index

Preface

Welcome to Release 11g (11.1.1) of the *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher*.

Intended Audience

This book is intended for report developers and designers who build data models for Oracle BI Publisher reports.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit
<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit
<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Business Intelligence Enterprise Edition 11g Release 1 (11.1.1) documentation set:

- The Oracle Business Intelligence chapter in the *Oracle Fusion Middleware Release Notes* for your platform
- *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*
- *Oracle Fusion Middleware Developer's Guide for Oracle Business Intelligence Publisher*
- *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*
- *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*
- *Oracle Fusion Middleware Installation Guide for Oracle Business Intelligence*

System Requirements and Certification

Refer to the system requirements and certification documentation for information about hardware and software requirements, platforms, databases, and other information. Both of these documents are available on Oracle Technology Network (OTN).

The system requirements document covers information such as hardware and software requirements, minimum disk space and memory requirements, and required system libraries, packages, or patches:

http://www.oracle.com/technology/software/products/ias/files/fusion_requirements.htm

The certification document covers supported installation types, platforms, operating systems, databases, JDKs, and third-party products:

http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

New Features for Data Model Designers

This preface describes changes to Oracle BI Publisher data modeling features for Oracle Business Intelligence Publisher 11g Release 1 (11.1.1). The following information provides descriptions of the differences in features, tools, and procedures between releases.

This preface contains the following topics:

- [New Features for Oracle BI Publisher 11g Release 1 \(11.1.1.6\)](#)
- [New Features for Oracle BI Publisher 11g Release 1 \(11.1.1.5\)](#)
- [New Features for Oracle BI Publisher 11g Release 1 \(11.1.1.3\)](#)

New Features for Oracle BI Publisher 11g Release 1 (11.1.1.6)

New features for report data model developers in Oracle BI Publisher 11g Release 1 (11.1.1.6) include:

- [Schedule Triggers for Event-Driven Schedules](#)
- [Enhancements to Parameter Display Options](#)
- [Support for HTML Formatting in Data](#)

Schedule Triggers for Event-Driven Schedules

The execution of a scheduled report job can now be conditionalized based on an event. A report data model supports a new type of trigger called a Schedule Trigger. The schedule trigger that you create in the data model can then be enabled in the report job schedule. When the report job is scheduled to run, the trigger is executed. If no data is returned from the trigger, the job instance is skipped. If data is returned, the job instance runs as scheduled. See [Section 6.3, "Creating Schedule Triggers"](#) for more information.

Enhancements to Parameter Display Options

Enhancements to parameter display options include:

- Support for radio button and check box display of values
Parameters that are defined as a menu in the data model can now be configured to display the menu options as a list of radio buttons or check boxes. Parameters configured to support one value support the option to display as radio buttons. Parameters configured to support multiple values support the option to display as check boxes.
- New options for placement of parameters in the report viewer

Previously the report viewer always displayed the parameters in a horizontal region across the top of the viewer. In this release, the display of the parameter region can also be configured in one of the following ways:

- In a vertical region along the right side of the viewer
- As a dialog
- As a separate full page
- New report viewer toolbar button to show or hide parameters
This release adds a toolbar button to the report viewer to enable users to hide or show the parameter region. This enhancement complements the new parameter display options. When report parameters are configured to display as a dialog or in a separate full page, the parameter display region is dismissed when the viewer displays the report. Use the **Parameters** button to redisplay the parameter region to make new selections.
- Search added to menus
All parameter menus having more than a specified number of options provide a Search option.
- Option to remove the **Apply** button
Reports can now be configured to remove the parameter **Apply** button. In these reports the action of selecting a new parameter value automatically reruns the report.

Support for HTML Formatting in Data

You can now convert stored XHTML to XSL-FO to display the HTML formatting from your data in your generated report. The XHTML data must be extracted wrapped in a CDATA section. Specific syntax must also be used in the RTF template to render it. See [Section 3.12.2, "Handling XHTML Data Stored in a CLOB Column"](#) for more information.

New Features for Oracle BI Publisher 11g Release 1 (11.1.1.5)

New features in Oracle BI Publisher 11g Release 1 (11.1.1.5) include:

- [Support for CLOB as XML](#)
- [Upload Local Microsoft Excel File as Data Source](#)
- [Use LDAP Attributes as Bind Variables in Data Queries](#)

Support for CLOB as XML

The data engine can now extract well-formed XML data stored in a database column as a character large object (CLOB) data type and maintain its structure. This feature enables you to use XML data generated by a separate process and stored in your database as input to a BI Publisher data model. For more information, see [Section 3.12, "Using Data Stored as a Character Large Object \(CLOB\) in a Data Model."](#)

Upload Local Microsoft Excel File as Data Source

You can now upload a locally stored Excel file directly to a data model definition. This file can then be refreshed on demand from the data model definition. For more information, see [Section 3.7, "Creating a Data Set Using a Microsoft Excel File."](#)

Use LDAP Attributes as Bind Variables in Data Queries

LDAP attributes defined in the LDAP Security Model definition can be used as bind variables in data queries. For more information, see [Section 3.15.2.1, "Creating Bind Variables from LDAP User Attribute Values."](#)

New Features for Oracle BI Publisher 11g Release 1 (11.1.1.3)

New features in Oracle BI Publisher 11g Release 1 (11.1.1.3) include:

- [Major User Interface Improvements](#)
- [Shared BI Presentation Catalog](#)
- [Data Model Editor](#)
- [Data Model as a Sharable Object](#)
- [Support for Microsoft Excel File as a Data Source](#)
- [Support for View Object as a Data Source](#)

Major User Interface Improvements

The user interface has undergone major improvements in several areas, including a new Home page and redesigned editors and panes. These improvements are intended to make working with Oracle BI Publisher easier and more consistent. For information about working in the new interface, see *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*.

Shared BI Presentation Catalog

For installations of BI Publisher with the Oracle BI Enterprise Edition, BI Publisher now shares the same catalog with Oracle BI Presentation services. For information about the improved catalog, see *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*.

Data Model Editor

This release introduces the data model editor to create your report data models. The data model editor enables you to combine data from multiple data sets from different data sources, such as SQL, Excel files, Web services, HTTP feeds, and other applications into a single XML data structure. Data sets can either be unrelated or a relationship can be established between them using a data link. This guide describes in detail how to use the data model editor.

Data Model as a Sharable Object

The data model is saved as a distinct catalog object. This means that a single data model can now be used for multiple reports.

Support for Microsoft Excel File as a Data Source

A Microsoft Excel file can now be used to create a data set in a BI Publisher data model. For more information, see [Section 3.7, "Creating a Data Set Using a Microsoft Excel File."](#)

Support for View Object as a Data Source

BI Publisher enables you to connect to your custom applications built with Oracle Application Development Framework and use view objects in your applications as

data sources for reports. For more information, see [Section 3.9, "Creating a Data Set Using a View Object."](#)

Introduction to the Oracle Business Intelligence Publisher Data Modeling Guide

This chapter provides an introduction to using Oracle BI Publisher's data model editor. It covers the following topics:

- [Section 1.1, "Introduction to Oracle Business Intelligence Publisher"](#)

1.1 Introduction to Oracle Business Intelligence Publisher

Oracle BI Publisher is an enterprise reporting solution for authoring, managing, and delivering all your highly formatted documents, such as operational reports, electronic funds transfer documents, government PDF forms, shipping labels, checks, sales and marketing letters, and much more.

The *Oracle Fusion Middleware Data Modeling Guide for Oracle Business Intelligence Publisher* (this guide) describes how report developers use BI Publisher's data model editor to fetch and structure the data for use in the many different types of report layouts that BI Publisher supports. [Table 1–1](#) provides more information about using the product for other business roles.

Table 1–1 Business Roles, Sample Tasks, and Related Documentation

Role	Sample Tasks	Guide
Administrator	Configuring Security Configuring System Settings Diagnosing and Monitoring System Processes	<i>Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher</i>
Application developer or integrator	Integrating BI Publisher into existing applications using the application programming interfaces	<i>Oracle Fusion Middleware Developer's Guide for Oracle Business Intelligence Publisher</i>
Report consumer	Viewing reports Scheduling report jobs Managing report jobs	<i>Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher</i>
Report designer	Creating report definitions Designing layouts	<i>Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher</i>

Using the Data Model Editor

This chapter describes in detail how to use Oracle BI Publisher's data model editor. It covers the following topics:

- [Section 2.1, "What Is a Data Model?"](#)
- [Section 2.2, "Components of a Data Model"](#)
- [Section 2.3, "Features of the Data Model Editor"](#)
- [Section 2.4, "About the Data Source Options"](#)
- [Section 2.5, "Process Overview for Creating a Data Model"](#)
- [Section 2.6, "Launching the Data Model Editor"](#)
- [Section 2.7, "About the Data Model Editor Interface"](#)
- [Section 2.8, "Setting Data Model Properties"](#)

2.1 What Is a Data Model?

A data model is an object that contains a set of instructions for BI Publisher to retrieve and structure data for a report. Data models reside as separate objects in the catalog.

At the very simplest, a data model can be one data set retrieved from a single data source (for example, the data returned from the columns in the employees table). A data model can also be complex, including parameters, triggers, and bursting definitions as well as multiple data sets.

To build a data model, you use the data model editor.

2.2 Components of a Data Model

A data model supports the following components:

- **Data set**

A data set contains the logic to retrieve data from a single data source. A data set can retrieve data from a variety of data sources (for example, a database, an existing data file, a Web service call to another application, or a URL/URI to an external data provider). A data model can have multiple data sets from multiple sources.

- **Event triggers**

A trigger checks for an event. When the event occurs the trigger runs the PL/SQL code associated with it. The data model editor supports before data and after data

triggers. Event triggers consist of a call to execute a set of functions defined in a PL/SQL package stored in an Oracle database.

- **Flexfields**

A flexfield is a structure specific to Oracle Applications. The data model editor supports retrieving data from flexfield structures defined in your Oracle Application database tables.

- **Lists of values**

A list of values is a menu of values from which report consumers can select parameter values to pass to the report.

- **Parameters**

A parameter is a variable whose value can be set at runtime. The data model editor supports several parameter types.

- **Bursting Definitions**

Bursting is a process of splitting data into blocks, generating documents for each data block, and delivering the documents to one or more destinations. A single bursting definition provides the instructions for splitting the report data, generating the document, and delivering the output to its specified destinations.

2.3 Features of the Data Model Editor

Use the data model editor to combine data from multiple data sets from different data sources, such as SQL, Excel files, Web services, HTTP feeds, and other applications into a single XML data structure. Data sets can either be unrelated or a relationship can be established between them using a data link.

The data model editor enables you to perform the following tasks:

- **Link data** — Define master-detail links between data sets to build a hierarchical data model.
- **Aggregate data** — Create group level totals and subtotals.
- **Transform data** — Modify source data to conform to business terms and reporting requirements.
- **Create calculations** — Compute data values that are required for your report that are not available in the underlying data sources.

2.4 About the Data Source Options

BI Publisher supports a variety of data source types for creating data sets. These can be categorized into three general types:

The first type are data sets for which BI Publisher can retrieve metadata information from the source. For these data set types, the full range of data model editor functions is supported. These data set types are:

- SQL queries submitted against Oracle BI Server, an Oracle database, or other supported databases

See [Section 3.3, "Creating a Data Set Using a SQL Query."](#)

For information on supported databases, see [System Requirements and Certification](#).

- Microsoft Excel spreadsheet data sources

The Excel spreadsheet can be either stored in a file directory set up as a data source by your administrator; or you can upload it directly from a local source to the data model. See [Section 3.7, "Creating a Data Set Using a Microsoft Excel File."](#)

- Queries against your LDAP repository to retrieve user data

You can report on this data directly, or join this to data retrieved from other sources. See [Section 3.6, "Creating a Data Set Using an LDAP Query."](#)

- Multidimensional (MDX) queries against an OLAP data source

See [Section 3.5, "Creating a Data Set Using an MDX Query Against an OLAP Data Source."](#)

For the second type, BI Publisher can retrieve column names and data type information from the data source but it cannot process or structure the data. For these data set types, only a subset of the full range of data model editor functions is supported. These data set types are:

- Oracle BI Analyses

See [Section 3.8, "Creating a Data Set Using an Oracle BI Analysis."](#)

- View objects created using Oracle Application Development Framework (ADF)

See [Section 3.9, "Creating a Data Set Using a View Object."](#)

For the third type, BI Publisher retrieves data that has been generated and structured at the source and no additional modifications can be applied by the data model editor. These data set types are:

- HTTP XML feeds off the Web

See [Section 3.13, "Creating a Data Set from an HTTP XML Feed."](#)

- Web services

See [Section 3.10, "Creating a Data Set Using a Web Service."](#)

Supply the Web service WSDL to BI Publisher and then define the parameters in BI Publisher to use a Web service to return data for the report.

- Existing XML data files

See [Section 3.11, "Creating a Data Set Using a Stored XML File."](#)

2.5 Process Overview for Creating a Data Model

[Table 2–1](#) lists the process overview for creating a data model.

Table 2–1 Process of Creating a Data Model

Step	Reference
Launch the Data Model Editor.	Section 2.6, "Launching the Data Model Editor"
Set properties for the data model. (Optional)	Section 2.8, "Setting Data Model Properties"
Create the data sets for the data model.	Chapter 3, "Creating Data Sets"
Define the data output structure. (Optional)	Chapter 4, "Structuring Data"
Define the parameters to pass to the query, and define lists of values for users to select parameter values. (Optional)	Section 5, "Adding Parameters and Lists of Values"

Table 2–1 (Cont.) Process of Creating a Data Model

Step	Reference
Define Event Triggers. (Optional)	Section 6.1, "About Triggers"
(Oracle Applications Only) Define Flexfields. (Optional)	Chapter 7, "Adding Flexfields"
Test your data model and add sample data.	Section 3.14, "Testing Data Models and Generating Sample Data"
Add a bursting definition. (Optional)	Chapter 8, "Adding Bursting Definitions"

2.6 Launching the Data Model Editor

Launch the Data Model Editor from the Home page or from the global header in one of the following ways:

To launch the Data Model Editor from the global header:

1. Click **New** and then click **Data Model** to open the Data Model Editor.

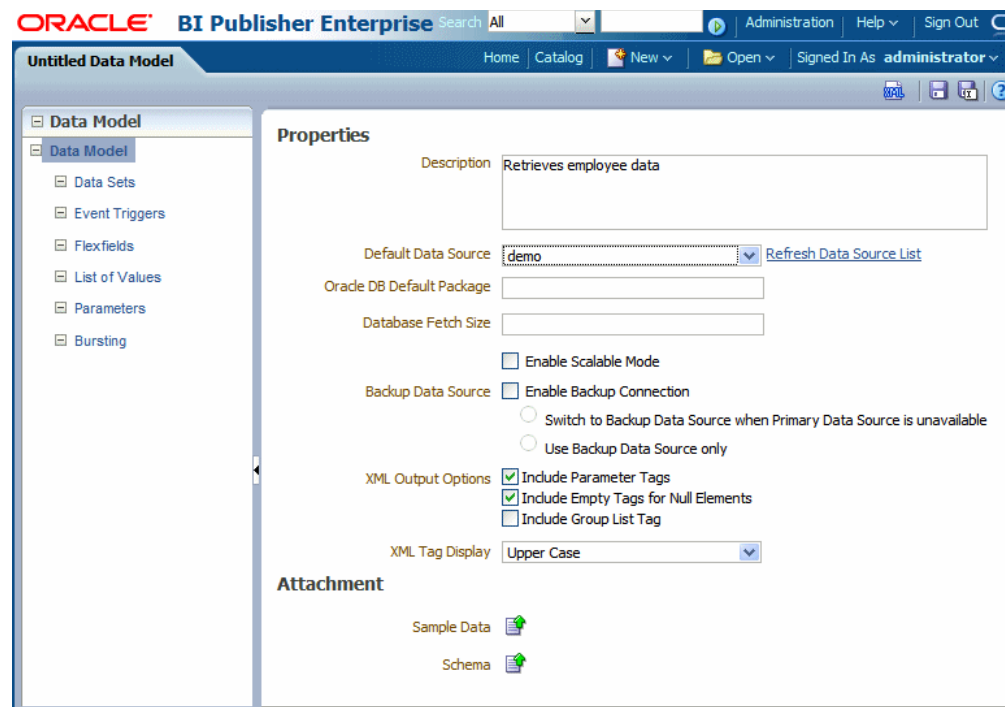
To launch the Data Model Editor from the Home page:

1. Under the **Create** region, click **Data Model**.

2.7 About the Data Model Editor Interface

Figure 2–1 shows the **Properties** pane of the data model editor interface.

Figure 2–1 Data Model Editor Interface



The Data Model Editor is designed with a component pane on the left and work pane on the right. Selecting a component on the left pane launches the appropriate fields for the component in the work area.

The toolbar, shown in [Figure 2-2](#), provides the following functions:

Figure 2-2 *Toolbar of Data Model Editor*



- **Get XML Output**

Launches the XML output page to run the data model definition and view or save the XML output.

- **Save / Save As**

Select **Save** to save your work in progress to the existing data model object or select **Save As** to save the data model as a new object in the catalog.

- **Help**

View online help for the data model editor.

2.8 Setting Data Model Properties

Enter the following properties for the data model:

Description — (Optional) The description that you enter here displays in the catalog. This description is translatable.

Default Data Source — Select the data source from the list. Data models can include multiple data sets from one or more data sources. The default data source you select here is presented as the default for each new data set you define. Select **Refresh Data Source List** to see any new data sources added since your session was initiated.

Oracle DB Default Package — If you define a query against an Oracle database, then you can include before or after data triggers (event triggers) in your data model. Event triggers make use of PL/SQL packages to execute RDBMS level functions. For data models that include event triggers or a PL/SQL group filter, you must enter a default PL/SQL package here. The package must exist on the default data source.

Database Fetch Size — Sets the number of rows fetched at a time through the JDBC connection. This value overrides the value set in the system properties. See "Setting Server Configuration Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*. If neither this value nor the server setting is defined, then a default value of 300 is used.

Enable Scalable Mode — Processing large data sets requires the use of large amounts of RAM. To prevent running out of memory, activate scalable mode for the data engine. In scalable mode, the data engine takes advantage of disk space when it processes the data.

Backup Data Source — If you have set up a backup database for this data source, select **Enable Backup Connection** to enable the option; then select when you want BI Publisher to use the backup.

- To use the backup data source only when the primary is down, select **Switch to Backup Data Source when Primary Data Source is unavailable**. Note that when

the primary data source is down, the data engine must wait for a response before switching to the backup.

- To always use the backup data source when executing this data model, select **Use Backup Data Source Only**. Using the backup database may enhance performance.

Note: This feature requires that a backup data source has been enabled for the selected data source. For more information see "About Backup Data Sources" in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

2.8.1 XML Output Options

These options define characteristics of the XML data structure. Note that any changes to these options can impact layouts that are built on the data model.

- **Include Parameter Tags** — If you define parameters for your data model, select this box to include the parameter values in the XML output file. See [Section 5, "Adding Parameters and Lists of Values"](#) for information on adding parameters to your data model. Enable this option when you want to use the parameter value in the report.
- **Include Empty Tags for Null Elements** — Select this box to include elements with null values in your output XML data. When you include a null element, then a requested element that contains no data in your data source is included in your XML output as an empty XML tag as follows: `<ELEMENT_ID\>`. For example, if the element `MANAGER_ID` contained no data and you chose to include null elements, it would appear in your data as follows: `<MANAGER_ID />`. If you do not select this option, no entry appears for `MANAGER_ID`.
- **Include Group List Tag** — (This property is for 10g backward compatibility and Oracle Report migration.) Select this box to include the rowset tags in your output XML data. If you include the group list tags, then the group list appears as another hierarchy within your data.
- **XML Tag Display** — Select whether to generate the XML data tags in upper case, in lower case, or to preserve the definition you supplied in the data structure.

2.8.2 Attachments to the Data Model

The Attachment region of the page displays data files that you have uploaded or attached to the data model.

2.8.2.1 Attaching Sample Data

After you build your data model, it is required that you attach a small, but representative set of sample data generated from your data model. The sample data is used by BI Publisher's layout editing tools. Using a small sample file helps improve performance during the layout design phase.

The Data Model Editor provides an option to generate and attach the sample data. See [Section 3.14, "Testing Data Models and Generating Sample Data."](#)

2.8.2.2 Attaching Schema

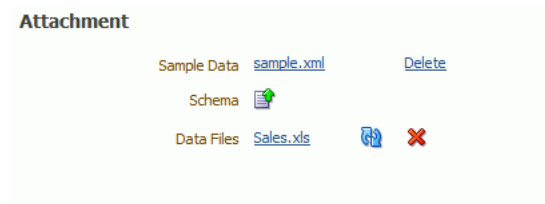
The Data Model Editor enables you to attach sample schema to the data model definition. The schema file is not used by BI Publisher, but can be attached for developer reference. The Data Model Editor does not support schema generation.

2.8.2.3 Data Files

If you have uploaded a local Microsoft Excel file as a data source for this report, the file displays here. Use the refresh button to refresh this file from the local source. For information on uploading an Excel file to use as a data source, see [Section 3.7](#), "Creating a Data Set Using a Microsoft Excel File."

Figure 2–3 shows the Attachments region with sample data and data files attached:

Figure 2–3 Attachments Region with Attached Sample Data and Files



Creating Data Sets

This chapter describes how to create data sets. It covers the following topics:

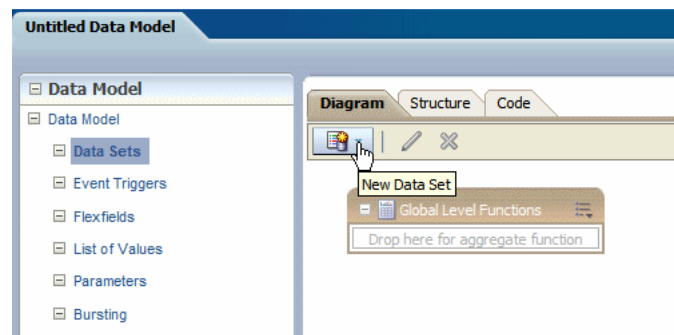
- [Section 3.1, "Overview of Creating Data Sets"](#)
- [Section 3.2, "Editing an Existing Data Set"](#)
- [Section 3.3, "Creating a Data Set Using a SQL Query"](#)
- [Section 3.4, "Using the Query Builder"](#)
- [Section 3.5, "Creating a Data Set Using an MDX Query Against an OLAP Data Source"](#)
- [Section 3.6, "Creating a Data Set Using an LDAP Query"](#)
- [Section 3.7, "Creating a Data Set Using a Microsoft Excel File"](#)
- [Section 3.8, "Creating a Data Set Using an Oracle BI Analysis"](#)
- [Section 3.9, "Creating a Data Set Using a View Object"](#)
- [Section 3.10, "Creating a Data Set Using a Web Service"](#)
- [Section 3.11, "Creating a Data Set Using a Stored XML File"](#)
- [Section 3.12, "Using Data Stored as a Character Large Object \(CLOB\) in a Data Model"](#)
- [Section 3.13, "Creating a Data Set from an HTTP XML Feed"](#)
- [Section 3.14, "Testing Data Models and Generating Sample Data"](#)
- [Section 3.15, "Including User Information Stored in System Variables in Your Report Data"](#)

3.1 Overview of Creating Data Sets

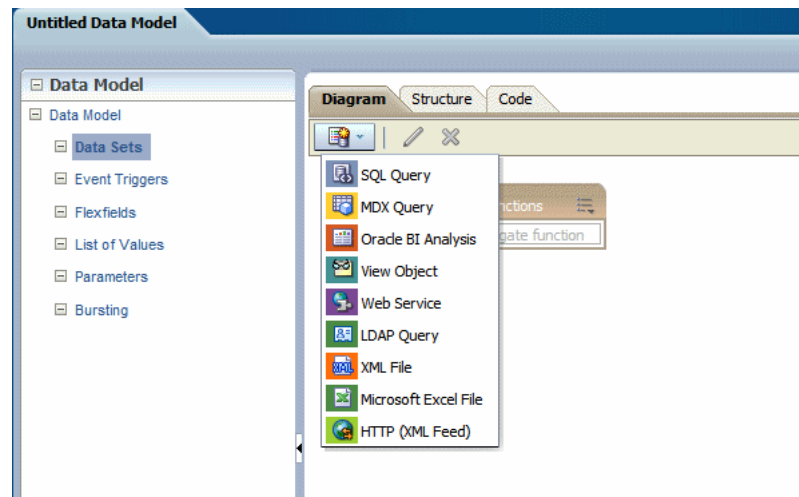
Oracle BI Publisher can retrieve data from multiple types of data sources.

to create a data set:

1. On the component pane of the data model editor click **Data Sets**.
2. Click **New Data Set** as shown in [Figure 3-1](#).

Figure 3–1 Creating a Data Set

3. Select the data set type from the list to launch the appropriate dialog, as shown in [Figure 3–2](#).

Figure 3–2 Selecting the Data Set Type

4. Complete the required fields to create the data set. See the corresponding section in this chapter for information on creating each data set type.
 - [Section 3.3, "Creating a Data Set Using a SQL Query"](#)
 - [Section 3.5, "Creating a Data Set Using an MDX Query Against an OLAP Data Source"](#)
 - [Section 3.6, "Creating a Data Set Using an LDAP Query"](#)
 - [Section 3.7, "Creating a Data Set Using a Microsoft Excel File"](#)
 - [Section 3.8, "Creating a Data Set Using an Oracle BI Analysis"](#)
 - [Section 3.9, "Creating a Data Set Using a View Object"](#)
 - [Section 3.10, "Creating a Data Set Using a Web Service"](#)
 - [Section 3.11, "Creating a Data Set Using a Stored XML File"](#)
 - [Section 3.12, "Using Data Stored as a Character Large Object \(CLOB\) in a Data Model"](#)
 - [Section 3.13, "Creating a Data Set from an HTTP XML Feed"](#)

3.2 Editing an Existing Data Set

To edit an existing data set:

1. On the component pane of the data model editor click **Data Sets**. All data sets for this data model display in the working pane.
2. Click the data set to edit.
3. Click the **Edit** toolbar button. The dialog for the data set opens. For information about each type of data set, see the corresponding section in this chapter.
4. Make changes to the data set and click **OK**.
5. Save the data model.
6. Test your edited data model and add new sample data. See [Section 3.14, "Testing Data Models and Generating Sample Data."](#)

3.3 Creating a Data Set Using a SQL Query

To create a data set using a SQL query:

1. Click the **New Data Set** icon and then click **SQL Query**. The Create Data Set - SQL dialog launches, as shown in [Figure 3-3](#).

Figure 3-3 Create Data Set - SQL Dialog

Create Data Set - SQL

* Name:

* Data Source: ☐ Default Data Source ☒ demo [Refresh Data Source List](#)

* SQL Query:

```
select  "EMPLOYEES"."FIRST_NAME" as "FIRST_NAME",
        "EMPLOYEES"."LAST_NAME" as "LAST_NAME",
        "EMPLOYEES"."HIRE_DATE" as "HIRE_DATE",
        "EMPLOYEES"."SALARY" as "SALARY",
        "EMPLOYEES"."MANAGER_ID" as "MANAGER_ID",
        "DEPARTMENTS"."DEPARTMENT_NAME" as "DEPARTMENT_NAME"
from    "OE"."DEPARTMENTS" "DEPARTMENTS",
        "OE"."EMPLOYEES" "EMPLOYEES"
where   "DEPARTMENTS"."DEPARTMENT_ID"="EMPLOYEES"."DEPARTMENT_ID"
```

2. Enter a name for this data set.
3. The **Data Source** defaults to the Default Data Source you selected on the Properties page. If you are not using the default data source for this data set, select the **Data Source** from the list.
4. Enter the SQL query or select **Query Builder**. See [Section 3.4, "Using the Query Builder"](#) for information about the Query Builder utility.
5. If you are using Flexfields, bind variables, or other special processing in your query, edit the SQL returned by the Query Builder to include the required statements.

Note: If you include lexical references for text that you embed in a SELECT statement, then you must substitute values to get a valid SQL statement.

6. After entering the query, click **OK** to save. The data model editor validates the query.

If your query includes a bind variable, you are prompted to create the bind parameter. Click **OK** to have the data model editor create the bind parameter. To edit the parameter, see [Chapter 5, "Adding Parameters and Lists of Values."](#)

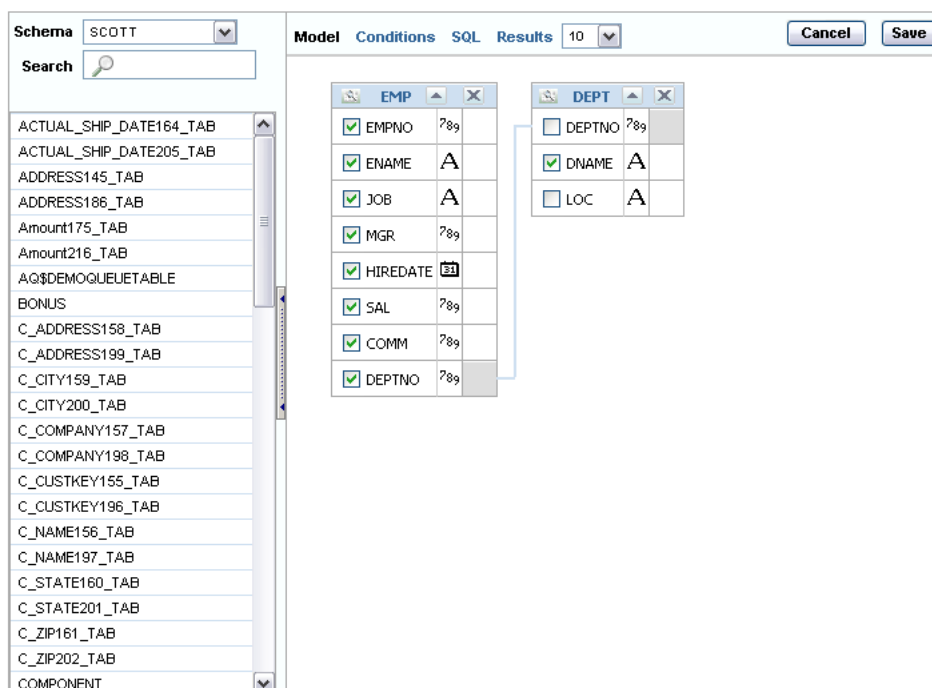
3.4 Using the Query Builder

Use the Query Builder to build SQL queries without coding. The Query Builder enables you to search and filter database objects, select objects and columns, create relationships between objects, and view formatted query results with minimal SQL knowledge.

The Query Builder page is divided into two sections:

- Object Selection pane contains a list of objects from which you can build queries. Only objects in the current schema display.
- Design and output pane consists of four tabs:
 - **Model** — Displays selected objects from the Object Selection pane.
 - **Conditions** — Enables you to apply conditions to your selected columns.
 - **SQL** — Displays the query.
 - **Results** — Displays the results of the query.

Figure 3–4 Design and Output Pane



3.4.1 Understanding the Query Builder Process

To build a query, perform the following steps:

1. Select objects from the Object Selection pane.
2. Add objects to the Design pane and select columns.
3. Optional: Establish relationships between objects.
4. Add a unique alias name for any duplicate column.
5. Optional: Create query conditions.
6. Execute the query and view results.

3.4.2 Using the Object Selection Pane

In the Object Selection pane you can select a schema and search and filter objects.

To hide the Object Selection pane, select the control bar located between it and the Design pane. Select it again to unhide it.

3.4.3 Selecting a Schema

The Schema list contains all the available schemas in the data source. Note that you may not have access to all that are listed.

3.4.4 Searching and Filtering Objects

Use the Search field to enter a search string. Note that if more than 100 tables are present in the data source, you must use the Search feature to locate and select the desired objects.

3.4.5 Selecting Objects

The Object Selection pane lists the tables, views, and materialized views from the selected schema (for Oracle databases, synonyms are also listed). Select the object from the list and it displays on the Design pane. Use the Design pane to identify how the selected objects are used in the query.

3.4.6 Supported Column Types

Columns of all types display as objects in the Design pane. Note the following column restrictions:

- You can select no more than 60 columns for each query.
- Only the following column types are selectable:
 - VARCHAR2, CHAR
 - NUMBER
 - DATE, TIMESTAMP

Note: The data type TIMESTAMP WITH LOCAL TIMEZONE is not supported.

- Binary Large Object (BLOB)

Note: The BLOB must be an image. When you execute the query in the Query Builder, the BLOB does not display in the Results pane; however, the query is constructed correctly when saved to the data model editor.

- Character Large Object (CLOB)

For more information about working with CLOB data in the data model, see [Section 3.12, "Using Data Stored as a Character Large Object \(CLOB\) in a Data Model."](#)

3.4.7 Adding Objects to the Design Pane

To add objects to the design pane:

1. Select an object.

The selected object displays in the Design pane. An icon representing the datatype displays next to each column name.

2. Select the check box for each column to include in your query.

When you select a column, it appears on the **Conditions** tab. Note that the **Show** check box on the **Conditions** tab controls whether a column is included in query results. By default, this check box is selected.

To select the first twenty columns, click the small icon in the upper left corner of the object and then select **Check All**.

3. To execute the query and view results, select **Results**.

Tip: You can also execute a query using the key strokes CTRL + ENTER.

3.4.8 Resizing the Design and Results Pane

As you select objects, you can resize the Design and Results panes by selecting and dragging the gray horizontal rule dividing the page.

3.4.9 Removing or Hiding Objects in the Design Pane

To remove an object:

1. Select the **Remove** icon in the upper right corner of the object.

To temporarily hide the columns within an object:

1. Click the **Show/Hide Columns** icon.

3.4.10 Specifying Query Conditions

Conditions enable you to filter and identify the data you want to work with. As you select columns within an object, you can specify conditions on the Conditions tab. You can use these attributes to modify the column alias, apply column conditions, sort columns, or apply functions. [Figure 3–5](#) shows the **Conditions** tab.

Figure 3–5 Conditions Tab

Model Conditions SQL Results 10

Column	Alias	Object	Condition	Sort Type	Sort Order
EMPLOYEE_ID	EMPLOYEE_ID	EMPLOYEES		ASC	
FIRST_NAME	FIRST_NAME	EMPLOYEES		ASC	
LAST_NAME	LAST_NAME	EMPLOYEES		ASC	
EMAIL	EMAIL	EMPLOYEES		ASC	
PHONE_NUMBER	PHONE_NUMBER	EMPLOYEES		ASC	
HIRE_DATE	HIRE_DATE	EMPLOYEES		ASC	
JOB_ID	JOB_ID	EMPLOYEES		ASC	
SALARY	SALARY	EMPLOYEES		ASC	
COMMISSION_PCT	COMMISSION_PCT	EMPLOYEES		ASC	
MANAGER_ID	MANAGER_ID	EMPLOYEES		ASC	
DEPARTMENT_ID	DEPARTMENT_ID	EMPLOYEES		ASC	
DEPARTMENT_NAME	DEPARTMENT_NAME	DEPARTMENTS		ASC	

Table 3–1 describes the attributes available on the **Conditions** tab.

Table 3–1 Attributes Available on the Conditions Tab

Condition Attribute	Description
Up and Down Arrows	Controls the display order of the columns in the resulting query.
Column	Displays the column name.
Alias	Specify an optional column alias. An alias is an alternative column name. Aliases are used to make a column name more descriptive, to shorten the column name, or prevent possible ambiguous references. Note that multibyte characters are not supported in the alias name.
Condition	The condition modifies the query's WHERE clause. When specifying a column condition, you must include the appropriate operator and operand. All standard SQL conditions are supported. For example: >=10 ='VA' IN (SELECT dept_no FROM dept) BETWEEN SYSDATE AND SYSDATE + 15
Sort Type	Select ASC (Ascending) or DESC (Descending).
Sort Order	Enter a number (1, 2, 3, and so on) to specify the order in which selected columns should display.

Table 3–1 (Cont.) Attributes Available on the Conditions Tab

Condition Attribute	Description
Show	<p>Select this check box to include the column in your query results. You do not need to select Show to add a column to the query for filtering only. For example, to create following query:</p> <pre>SELECT ename FROM emp WHERE deptno = 10</pre> <p>To create this query in Query Builder:</p> <ol style="list-style-type: none"> 1. From the Object list, select EMP. 2. In the Design Pane, select ename and deptno. 3. For the deptno column, in Condition enter =10 and uncheck the Show check box.
Function	<p>Available argument functions include:</p> <ul style="list-style-type: none"> ■ Number columns — COUNT, COUNT DISTINCT, AVG, MAXIMUM, MINIMUM, SUM ■ VARCHAR2, CHAR columns — COUNT, COUNT DISTINCT, INITCAP, LENGTH, LOWER, LTRIM, RTRIM, TRIM, UPPER ■ DATE, TIMESTAMP columns— COUNT, COUNT DISTINCT
Group By	Specify columns to be used for grouping when an aggregate function is used. Only applicable for columns included in output.
Delete	Deselect the column, excluding it from the query.

As you select columns and define conditions, Query Builder writes the SQL for you.

To view the underlying SQL:

1. Click the **SQL** tab.

3.4.11 Creating Relationships Between Objects

You can create relationships between objects by creating a join. A join identifies a relationship between two or more tables, views, or materialized views.

3.4.11.1 About Join Conditions

When you write a join query, you specify a condition that conveys a relationship between two objects. This condition is called a join condition. A join condition determines how the rows from one object combine with the rows from another object.

Query Builder supports inner, outer, left, and right joins. An inner join (also called a simple join) returns the rows that satisfy the join condition. An outer join extends the result of a simple join. An outer join returns all rows that satisfy the join condition and returns some or all of those rows from one table for which no rows from the other satisfy the join condition.

3.4.11.2 Joining Objects Manually

Create a join manually by selecting the Join column in the Design pane.

To join objects manually:

1. From the Object Selection pane, select the objects you want to join.
2. Identify the columns you want to join.

You create a join by selecting the Join column adjacent to the column name. The Join column displays to the right of the datatype. When your cursor is in the appropriate position, the following help tip displays:

Click here to select column for join

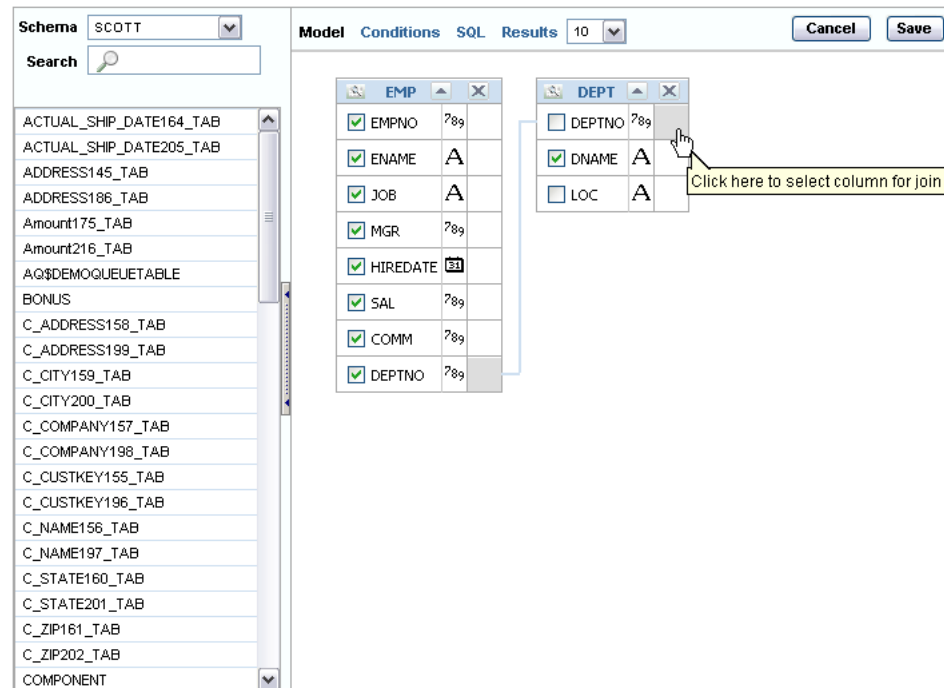
3. Select the appropriate Join column for the first object.

When selected, the Join column is darkened. To deselect a Join column, simply select it again or press ESC.

4. Select the appropriate Join column for the second object.

When joined, a line connects the two columns. An example is shown in [Figure 3–6](#).

Figure 3–6 *Joined Columns*

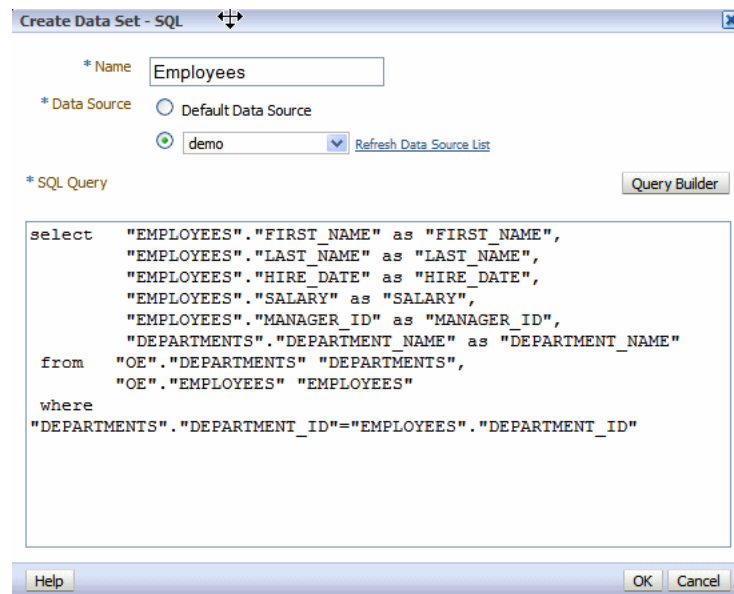


5. Select the columns to be included in your query. You can view the SQL statement resulting from the join by positioning the cursor over the join line.
6. Click **Results** to execute the query.

3.4.12 Saving a Query

Once you have built the query, click **Save** to return to the data model editor. The query appears in the SQL Query box. Click OK to save the data set.

To link the data from this query to the data from other queries or modify the output structure, see [Chapter 4, "Structuring Data."](#)

Figure 3–7 Query Displayed in SQL Query Box

3.4.13 Adding a Bind Variable to a Query

Now you have a basic query, but in the report you want users to be able to pass a parameter to the query to limit the results. For example, in the employee listing, you want users to be able to choose a specific department.

You can add the variable using either of the following methods:

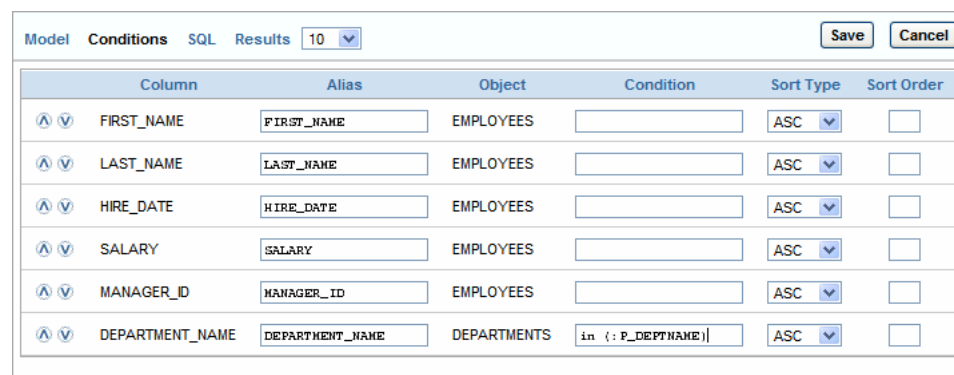
- Add the bind variable using the Query Builder Conditions tab

To add the bind variable in the Query Builder Conditions tab:

1. Add the following Condition for the column:

in (:P_DEPTNAME)

where P_DEPTNAME is the name you choose for the parameter. This is shown in [Figure 3–8](#).

Figure 3–8 Adding the Bind Variable Using the Query Builder Conditions Tab

- Update the SQL query directly in the text box.

Important: After manually editing the query, the Query Builder can no longer parse it. Any further edits must also be made manually.

To update the SQL query directly in the text box:

1. Add the following after the where clause in your query:

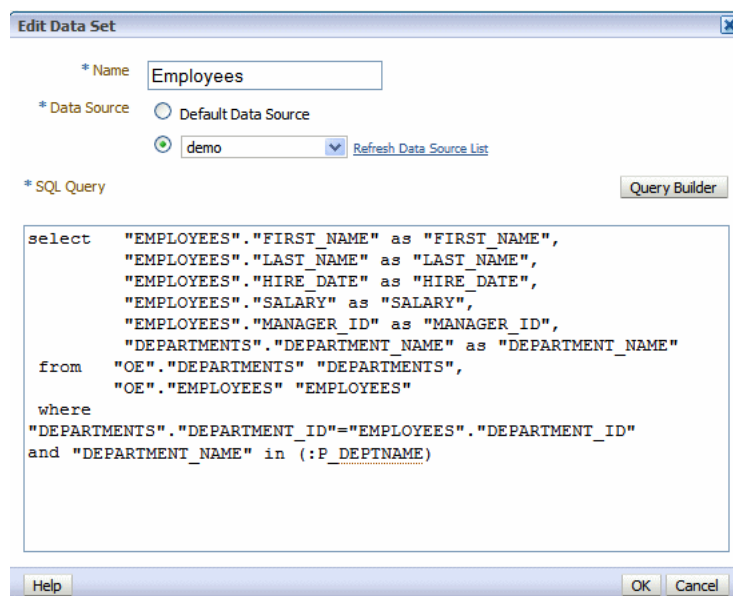
```
and "COLUMN_NAME" in (:PARAMETER_NAME)
```

for example:

```
and "DEPARTMENT_NAME" in (:P_DEPTNAME)
```

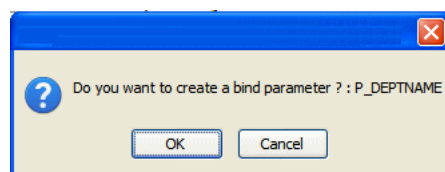
where P_DEPTNAME is the name you choose for the parameter. This is shown in [Figure 3–9](#).

Figure 3–9 *Editing the Generated SQL Query*



2. When you select **Save**, the data model editor asks whether to create the parameter that you entered with the bind variable syntax, as shown in [Figure 3–10](#).

Figure 3–10 *Create a Bind Parameter Dialog*



3. Click **OK** to have the data model editor create the parameter entry for you.

To define the parameter properties, see [Chapter 5, "Adding Parameters and Lists of Values."](#)

3.4.14 Editing a Saved Query

When you have saved the query from the Query Builder to the data model editor, you can also use the Query Builder to edit the query.

To use the Query Builder to edit the query:

1. Select the SQL data set.
2. Click the **Edit Selected Data Set** toolbar button.
3. This launches the **Edit Data Set** dialog. Click **Query Builder** to load the query to the Query Builder.

Note: If you have made modifications to the query, or did not use the Query Builder to construct it, you may receive an error when launching the Query Builder to edit it. If the Query Builder cannot parse the query, you can edit the statements directly in the text box.

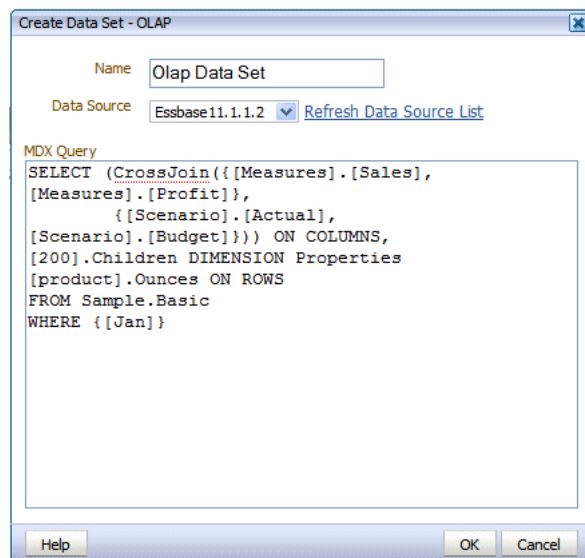
4. Edit the query and click **Save**.

3.5 Creating a Data Set Using an MDX Query Against an OLAP Data Source

BI Publisher supports Multidimensional Expressions (MDX) queries against your OLAP data sources. MDX lets you query multidimensional objects, such as cubes, and return multidimensional cellsets that contain the cube's data. See your OLAP database documentation for information about the MDX syntax and functions it supports.

Figure 3–11 shows a sample MDX query.

Figure 3–11 Sample MDX Query



To create a data set using an MDX query against an OLAP data source:

1. Click the **New Data Set** toolbar button and select OLAP. The Create Data Set - OLAP dialog launches.
2. Enter a name for this data set.

3. Select the **Data Source** for this data set. Only data sources defined as OLAP connections display in the list.
4. Enter the MDX query by direct entry or by copying and pasting from a third-party MDX editor.
5. Click **OK**.
6. To link the data from this query to the data from other queries or modify the output structure, see [Chapter 4, "Structuring Data."](#)

Note: Ensure that in your OLAP data source that you do not use Unicode characters from the range U+F900 to U+FFFE to define any metadata attributes such as column names or table names. This Unicode range includes half-width Japanese Katakana and full-width ASCII variants. Using these characters results in errors when generating the XML data for a BI Publisher report.

3.6 Creating a Data Set Using an LDAP Query

BI Publisher supports queries against Lightweight Directory Access protocol (LDAP) data sources. You can query user information stored in LDAP directories and then use the data model editor to link the user information with data retrieved from other data sources.

For example, to generate a report that lists employee salary information that is stored in the database application and include on the report employee e-mail addresses that are stored in the LDAP directory. You can create a query against each and then link the two in the data model editor to display the information in a single report. [Figure 3–12](#) shows a sample LDAP query.

Figure 3–12 Sample LDAP Query

The screenshot shows a dialog box titled "Create Data Set - LDAP". It has four main input areas: "Name" with the text "LDAP Data Set", "Data Source" with a dropdown menu showing "STAQ17" and a "Refresh Data Source List" link, "Attribute" with the text "mail, cn, givenName", and "Filter" with the text "(objectclass=person)". At the bottom of the dialog are three buttons: "Help", "OK", and "Cancel".

To create a data set using an LDAP query:

1. Click the **New Data Set** toolbar button and select LDAP. The Create Data Set - LDAP dialog launches.

2. Enter a name for this data set.
3. Select the **Data Source** for this data set. Only data sources defined as LDAP connections display in the list.
4. In the Attributes entry box, enter the attributes whose values you want to fetch from the LDAP data source.
For example:
`mail,cn,givenName`
5. To filter the query, enter the appropriate syntax in the Filter entry box. The syntax is as follows:
`(Operator (Filter)through(Filter))`
For example:
`(objectclass=person)`

LDAP search filters are defined in the Internet Engineering Task Force (IETF) Request for Comments document 2254, "The String Representation of LDAP Search Filters," (RFC 2254). This document is available from the IETF Web site at <http://www.ietf.org/rfc/rfc2254.txt>
6. To link the data from this query to the data from other queries or modify the output structure, see [Chapter 4, "Structuring Data."](#)

3.7 Creating a Data Set Using a Microsoft Excel File

To use a Microsoft Excel file as a data source, you can either:

- Place the file in a directory that your administrator has set up as a data source. See the section "Setting Up a Connection to a File Data Source" in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.
- Upload the file to the data model from a local directory.

3.7.1 About Supported Excel Files

Following are guidelines for the support of Microsoft Excel files as a data set type in BI Publisher:

- The Microsoft Excel files must be saved in the Excel 97-2003 Workbook (*.xls) format by Microsoft Excel. Files created by a third party application or library are not supported.
- The source Excel file might contain a single sheet or multiple sheets.
- Each worksheet may contain one or multiple tables. A table is a block of data that is located in the continuous rows and columns of a sheet.

In each table, BI Publisher always considers the first row to be a heading row for the table.

- The data type of the data in the table may be number, text, or date/time.
- If multiple tables exist in a single worksheet, the tables must be identified with a name for BI Publisher to recognize each one. See [Section 3.7.2, "Guidelines for Accessing Multiple Tables per Sheet."](#)

- If all tables in the Excel file are not named, only the data in the first table (the table located in the upper most left corner) is recognized and fetched.
- When the data set is created, BI Publisher truncates all trailing zeros after the decimal point for numbers in all cases. To preserve the trailing zeros in your final report, you must apply a format mask in your template to display the zeroes. For more information about format masks, see the section "Number, Date, and Currency Formatting" in *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

3.7.2 Guidelines for Accessing Multiple Tables per Sheet

If the Excel worksheet contains multiple tables that you want to include as data sources, then you must define a name for each table in Excel.

Important: The name that you define must begin with the prefix: "BIP_", for example, "BIP_SALARIES".

To define a name for the table in Excel:

1. Insert the table in Excel.
2. Define a name for the table as follows:

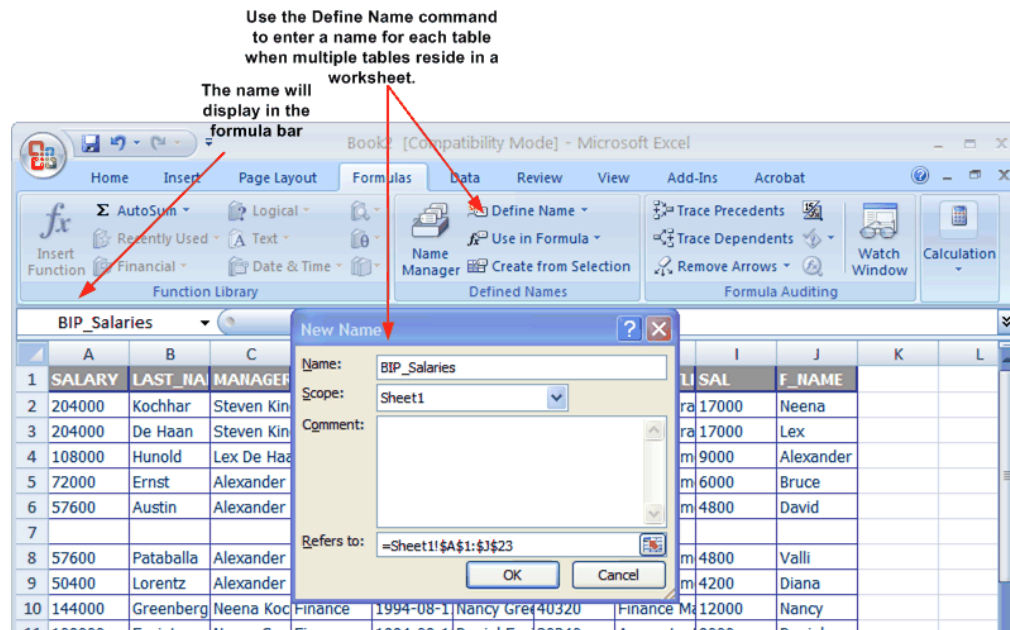
Using Excel 2003: Select the table. On the **Insert** menu, click **Name** and then **Define**. Enter a name that is prefixed with "BIP_".

Using Excel 2007: Select the table. On the **Formulas** tab, in the **Defined Names** group, click **Define Name**, then enter the name in the **Name** field. The name you enter appears on the Formula bar.

Tip: You can learn more about defined names and their usage in the Microsoft Excel 2007 document "Define and use names in formulas." at the following URL:

<http://office.microsoft.com/en-us/excel/HA101471201033.aspx>

Figure 3–13 shows how to use the Define Name command in Microsoft Excel 2007 to name a table "BIP_Salaries".

Figure 3–13 Using the Define Name Command in Microsoft Excel

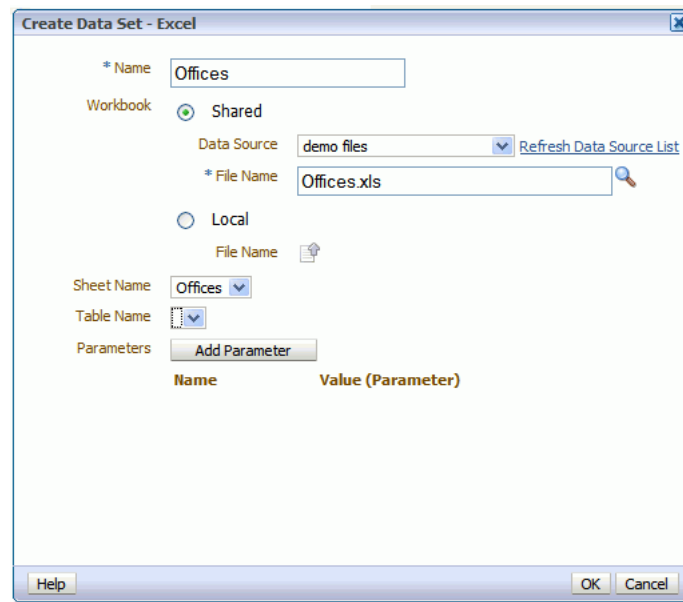
3.7.3 Using a Microsoft Excel File Stored in a File Directory Data Source

Note that to include parameters for your data set, you must define the parameters first, so that they are available for selection when defining the data set. See [Chapter 5, "Adding Parameters and Lists of Values."](#)

Important: The Excel data set type supports one value per parameter. It does not support multiple selection for parameters.

To create a data set using a Microsoft Excel file from a file directory data source:

1. Click the **New Data Set** toolbar button and select Microsoft Excel File. The Create Data Set - Excel dialog launches.
2. Enter a name for this data set.
3. Click **Shared** to enable the Data Source list.
4. Select the **Data Source** where the Excel File resides.
5. Click the browse icon to browse for the Microsoft Excel file in the data source directories. Select the file.
6. If the Excel file contains multiple sheets or tables, select the appropriate **Sheet Name** and **Table Name** for this data set, as shown in [Figure 3–14](#).

Figure 3–14 Selecting the Sheet Name

7. If you added parameters for this data set, click **Add Parameter**. Enter the **Name** and select the **Value**. The Value list is populated by the parameter Name defined in the Parameters section. See [Chapter 5, "Adding Parameters and Lists of Values."](#)
8. Click **OK**.

To link the data from this query to the data from other queries or modify the output structure, see [Chapter 4, "Structuring Data."](#)

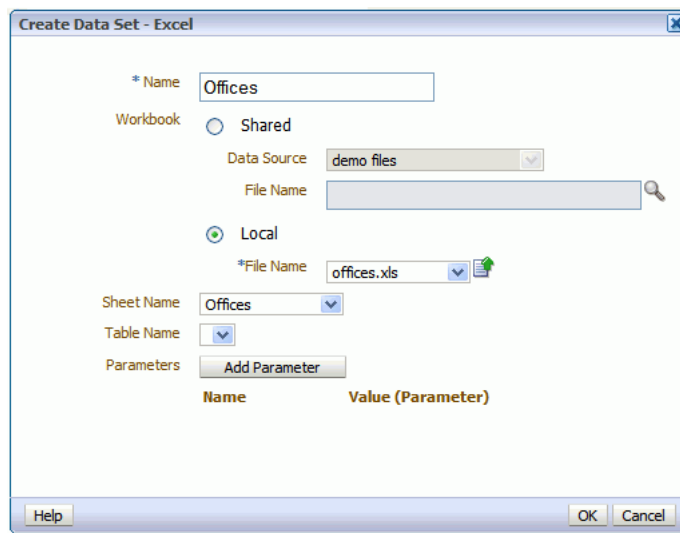
3.7.4 Uploading a Microsoft Excel File Stored Locally

Note that to include parameters for the data set, you must define the parameters first, so that they are available for selection when defining the data set. See [Chapter 5, "Adding Parameters and Lists of Values."](#)

Important: The Excel data set type supports one value per parameter. It does not support multiple selection for parameters.

To create a data set using a Microsoft Excel file stored locally:

1. Click the **New Data Set** toolbar button and select Microsoft Excel File. The Create Data Set - Excel dialog launches.
2. Enter a name for this data set.
3. Select **Local** to enable the upload button.
4. Click the **Upload** icon to browse for and upload the Microsoft Excel file from a local directory. If the file has been uploaded to the data model, then it is available for selection in the **File Name** list.
5. If the Excel file contains multiple sheets or tables, select the appropriate **Sheet Name** and **Table Name** for this data set, as shown in [Figure 3–15](#).

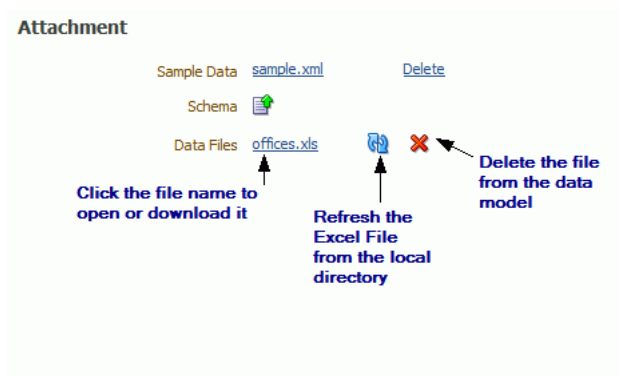
Figure 3–15 Defining Excel Spreadsheet for Data Set

6. If you added parameters for this data set, click **Add Parameter**. Enter the **Name** and select the **Value**. The Value list is populated by the parameter Name defined in the Parameters section. See [Chapter 5, "Adding Parameters and Lists of Values."](#)
7. Click **OK**.

To link the data from this query to the data from other queries or modify the output structure, see [Chapter 4, "Structuring Data."](#)

3.7.4.1 Refreshing and Deleting an Uploaded Excel File

After uploading the file, it displays on the **Properties** pane of the data model under the **Attachments** region, as shown in [Figure 3–16](#).

Figure 3–16 Attachments Region of the Properties Pane

See [Section 2.8, "Setting Data Model Properties"](#) for information about the **Properties** pane.

To refresh the local file in the data model:

1. Click **Data Model** in the component pane to view the **Properties** page.
2. In the **Attachment** region of the page, locate the file in the **Data Files** list.
3. Click **Refresh**.

4. In the **Upload** dialog, browse for and upload the latest version of the file. The file must have the same name or it will not replace the older version.
5. Save the data model.

To delete the local file:

1. Click **Data Model** in the component pane to view the **Properties** page.
2. In the **Attachment** region of the page, locate the file in the **Data Files** list.
3. Click **Delete**.
4. Click **OK** to confirm.
5. Save the data model.

3.8 Creating a Data Set Using an Oracle BI Analysis

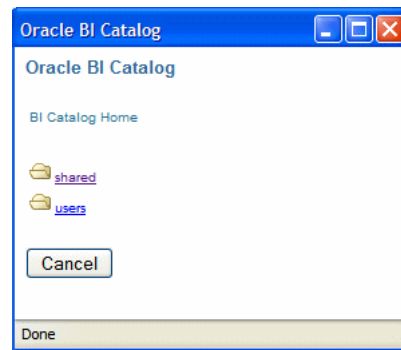
If you have enabled integration with Oracle Business Intelligence, then you can access the Oracle Business Intelligence Presentation catalog to select an Oracle BI analysis as a data source. An analysis is a query against an organization's data that provides answers to business questions. A query contains the underlying SQL statements that are issued to the Oracle BI Server.

For more information on creating analyses, see *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*.

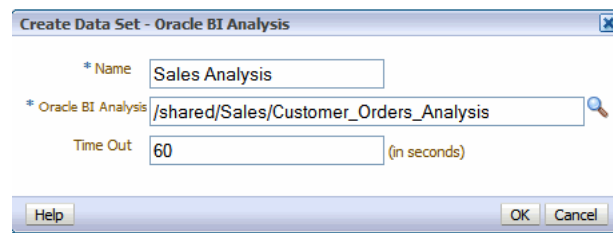
To create a data set using an Oracle BI analysis:

1. Click the **New Data Set** toolbar button and select Oracle BI Analysis. The Create Data Set - Oracle BI Analysis dialog launches.
2. Enter a name for this data set.
3. Click the browse icon to connect to the Oracle BI Presentation catalog, as shown in [Figure 3-17](#).

Figure 3-17 Connecting to the Oracle BI Presentation Catalog



4. When the catalog connection dialog launches, navigate through the folders to select the Oracle BI analysis to use as the data set for the report.
5. Enter a **Time Out** value in seconds, as shown in [Figure 3-18](#). If BI Publisher has not received the analysis data after the time specified in the time out value has elapsed, then BI Publisher stops attempting to retrieve the analysis data.

Figure 3–18 Creating a BI Analysis Data Set

6. Click OK.

3.8.1 Additional Notes on Oracle BI Analysis Data Sets

Parameters and list of values are inherited from the BI analysis and they will display at runtime.

The BI Analysis must have default values defined for filter variables. If the analysis contains presentation variables with no default values, it is not supported as a data source by BI Publisher.

If you want to structure the data based on Oracle BI Analysis Data Sets, the group breaks, data links and group-level functions are not supported.

The following are supported:

- Global level functions
- Setting the value for elements if null
- Group Filters

For more information about the above supported features, see [Chapter 4, "Structuring Data."](#)

3.9 Creating a Data Set Using a View Object

BI Publisher enables you to connect to your custom applications built with Oracle Application Development Framework and use view objects in your applications as data sources for reports.

This procedure assumes that you have created a view object in your application. For more information, see [Section 3.9, "Creating a Data Set Using a View Object."](#)

To create a data set using a view object:

1. Click the **New Data Set** toolbar button and select **View Object**. The Create Data Set - View Object dialog launches.
2. Enter a name for this data set.
3. Select the **Data Source** from the list. The data sources that you defined in the providers.xml file display.
4. Enter the fully qualified name of the application module (for example: example.apps.pa.entity.applicationModule.AppModuleAM).
5. Click **Load View Objects**.

BI Publisher calls the application module to load the view object list.

6. Select the **View Object**.

7. Any bind variables defined are retrieved. Create a parameter to map to this bind variable See [Chapter 5, "Adding Parameters and Lists of Values."](#)
8. Click OK to save your data set.

3.9.1 Additional Notes on View Object Data Sets

To structure data based on view object data sets, the group breaks, data links and group-level functions are not supported.

The following is supported: Setting the value for elements if null.

For more information about this supported feature, see [Chapter 4, "Structuring Data."](#)

3.10 Creating a Data Set Using a Web Service

BI Publisher supports Web service data sources that return valid XML data.

Important: Additional configuration may be required to access external Web services depending on your system's security. If the WSDL URL is outside the company firewall, then see the section "Configuring Proxy Settings" in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

If the Web service is protected by Secure Sockets Layer (SSL), then see the section "Configuring BI Publisher for Secure Socket Layer Communication" in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

BI Publisher supports Web services that return both simple data types and complex data types. You must make the distinction between simple and complex when you define the Web service data model. See [Section 3.10.1, "Adding a Simple Web Service: Example"](#) and [Section 3.10.2, "Adding a Complex Web Service"](#) for descriptions of setting up each type.

Note that to include parameters for the Web service method, you must define the parameters first, so that they are available for selection when setting up the data source. See [Chapter 5, "Adding Parameters and Lists of Values."](#)

Multiple parameters are supported. Ensure the method name is correct and the order of the parameters matches the order in the method. To call a method in the Web service that accepts two parameters, you must map two parameters defined in the report to those two. Note that only parameters of simple type are supported, for example, string and integer.

- Enter the WSDL URL and the Web Service Method.

Important: Only document/literal Web services are supported.

- To specify a parameter, select the Add link. Select the parameter from the list.

Note: The parameters must be set up in the Parameters section of the report definition See [Chapter 5, "Adding Parameters and Lists of Values."](#)

3.10.1 Adding a Simple Web Service: Example

This example shows how to add a Web service to BI Publisher as a data source. The Web service returns stock quote information. The Web service passes one parameter: the quote symbol for a stock.

The WSDL URL is:

<http://www.websvcex.net/stockquote.asmx?WSDL>

If you are not familiar with the available methods and parameters in the Web service to call, you can open the URL in a browser to view them. This Web service includes a method called GetQuote. It takes one parameter, which is the stock quote symbol.

To add the Web service as a data source:

1. Click the **New Data Set** toolbar button and select Web Services. The Create Data Set - Web Service dialog launches, as shown in [Figure 3–19](#).

Figure 3–19 Creating a Simple Web Service Data Set

The screenshot shows a Windows-style dialog box titled "Create Data Set - Web Services". It contains several input fields and a button. The "Name" field is filled with "Stock Quote". The "Complex Type" field is a dropdown menu set to "False". The "WSDL URL" field is filled with "http://www.websvcex.net/stockquote.asmx?WSDL". The "Method" field is filled with "GetQuote". The "Time Out" field is empty, with a label "(in seconds)" to its right. Below these fields is a "Parameters" section with an "Add Parameter" button. At the bottom of the dialog are three buttons: "Help", "OK", and "Cancel".

2. Enter a name for this data set.
3. Enter the Data Set information:
 - Select **False** for **Complex Type**.
 - Enter the **WSDL URL**: `http://www.websvcex.net/stockquote.asmx?WSDL`
 - Enter the **Method**: `GetQuote`
 - If desired, enter a **Time Out** period in seconds. If the BI Publisher server cannot establish a connection to the Web service, the connection attempt times out after the specified time out period has elapsed.
4. Define the parameter to make it available to the Web service data set.
 Select **Parameters** on the Data Model pane and click the **Create New Parameter** button. Enter the following:
 - **Identifier** — Enter an internal identifier for the parameter (for example, `Quote`).
 - **Data Type** — Select **String**.

- **Default Value** — If desired, enter a default for the parameter (for example, ORCL).
 - **Parameter Type** — Select Text
5. In the **Text Setting** region, enter the following:
- **Display label** — Enter the label you want displayed for your parameter (for example: Stock Symbol).
 - **Text Field Size** — Enter the size for the text entry field in characters.

Figure 3–20 Creating the Parameter

The screenshot shows the 'Simple Web Service' application. On the left is a 'Data Model' tree with 'Data Sets' expanded, showing 'Stock Quote'. The 'Parameters' section is active, displaying a table with one parameter: 'Quote' of type 'String' and default value 'ORCL'. Below this, the 'Text Setting' section for the 'Quote' parameter is shown, with 'Display Label' set to 'Stock Symbol' and 'Text Field Size' set to '6'. Two options are listed: 'Text field contains comma-separated values' and 'Refresh other parameters on change', both of which are unchecked.

Name	Data Type	Default Value	Parameter Type	Reorder
Quote	String	ORCL	Text	

Quote

Text Setting

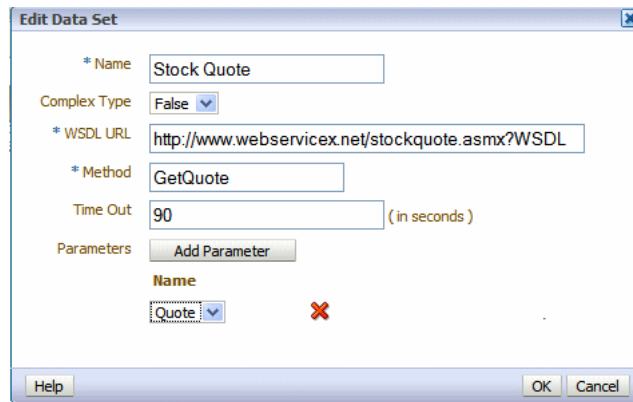
Display Label: Stock Symbol

Text Field Size: 6

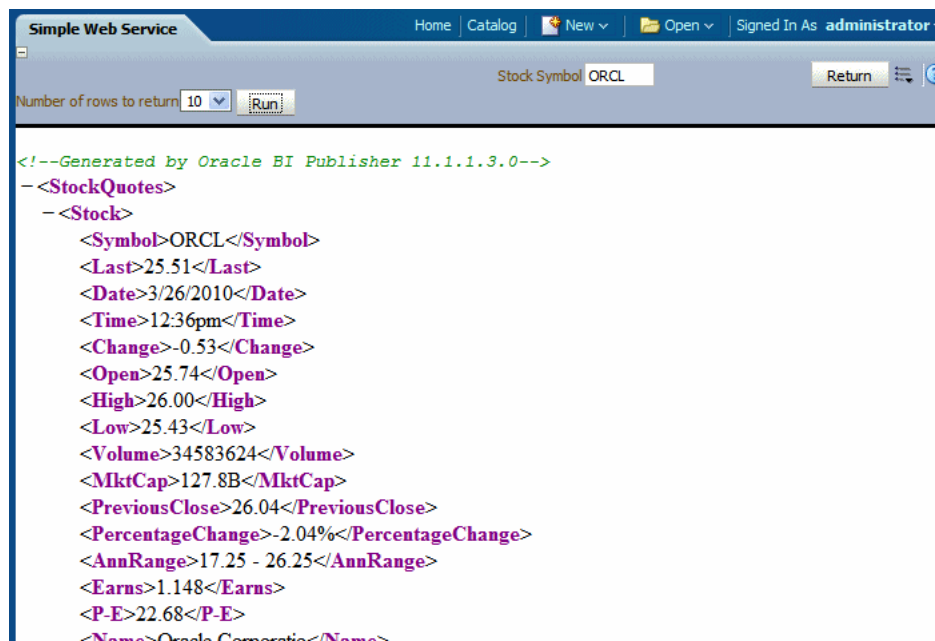
Options

- ☐ Text field contains comma-separated values
- ☐ Refresh other parameters on change

6. Select the options you want to apply:
- **Text field contains comma-separated values** — Select this option to enable the user to enter multiple comma-delimited values for this parameter.
 - **Refresh other parameters on change** — Performs a partial page refresh to refresh any other parameters whose values are dependent on the value of this one.
7. Return to your Web service data set and add the parameter.
- Click the data set name Stock Quote. Click **Add Parameter**. The Quote parameter you specified is now available from the list.
 - Click the **Edit Selected Data Set** button.
 - In the **Edit Data Set** dialog, click **Add Parameter**. The Quote parameter displays, as shown in [Figure 3–21](#).

Figure 3–21 Adding the Parameter to Web Service Data Set

- Click **OK** to close the data set.
8. Click **Save**.
 9. To view the results XML, select **Get XML Output**.
 10. Enter a valid value for your Stock Symbol parameter, select the number of rows to return, and click the **Run** button. Figure 3–22 shows the data returned from the example.

Figure 3–22 Data Returned from Stock Quote Example

3.10.2 Adding a Complex Web Service

A complex Web service type internally uses soapRequest / soapEnvelope to pass the parameter values to the destination host.

To use a complex Web service as a data source, select Complex Type equal True, then enter the WSDL URL. After loading and analyzing the WSDL URL, the Data Model Editor displays the available Web services and operations. For each selected operation,

the Data Model Editor displays the structure of the required input parameters. By choosing **Show Optional Parameters**, you can see all optional parameters as well.

If you are not familiar with the available methods and parameters in the Web service, open the WSDL URL in a browser to view them.

To add a complex Web service as a data source:

1. Enter the Data Set information:

- Enter a **Name** for the Data Set and select Web Service as the **Type**.
- Select True for **Complex Type**.
- Select a security header:
 - Disabled — Does not insert a security header.
 - 2002 — Enables the "WS-Security" Username Token with the 2002 namespace:
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd`
 - 2004 — Enables the "WS-Security" Username Token with the 2004 namespace:
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText`
- **Username and Password** — Enter the username and password for the Web service, if required.
- If desired, enter a **Time Out** period in seconds. If the BI Publisher server cannot establish a connection to the Web service, then connection attempt times out after the specified time out period has elapsed.
- Enter a **WSDL URL**. When you enter the WSDL, the Web Service list populates with the available Web services from the WSDL.
- Choose a Web Service from the list. When you choose a Web service from the list, the **Method** list populates with the available methods.
- Select the **Method**. When you select the method, the **Parameters** display. If you want to see optional parameters as well, then select **Show Optional Parameters**.
- **Response Data XPath** — If the start of the XML data for the report is deeply embedded in the response XML generated by the Web service request, then use this field to specify the path to the data to use in the BI Publisher report.

2. Define the parameter to make it available to the Web service data set.

Select **Parameters** on the **Report** definition pane and click **New** to create a parameter. Enter the following:

- **Name** — Enter an internal identifier for the parameter.
- **Data Type** — Select the appropriate data type for the parameter.
- **Default Value** — If desired, enter a default value for the parameter.
- **Parameter Type** — Select the appropriate parameter type.
- **Display label** — Enter the label you want displayed for your parameter.
- **Text Field Size** — Enter the size for the text entry field in characters.

3. Return to the Web service data set and add the parameter.
 - Select the Web service data set and then click Edit Selected Data Set to launch the Edit Data Set dialog.
 - In the entry field for the Parameter, enter the following syntax: `${Parameter_Name}` where `Parameter_Name` is the value you entered for **Name** when you defined the parameter to BI Publisher. [Figure 3–23](#) shows an example of entering the parameters.

Figure 3–23 Entering Parameters for Complex Web Service

The screenshot shows the 'Edit Data Set' dialog box with the following configuration:

- Name:** Complex Web Service
- Complex Type:** True
- WS-Security:** 2002
- Username:** (empty field)
- Password:** (empty field)
- Time Out:** (empty field) (in seconds)
- WSDL URL:** http://adc2110369.us.oracle.com:7001/xmlpserver/serv
- Web Service:** PublicReportServiceService
- Method:** getFolderContents
- Show optional parameters:** (unchecked)
- ResponseData XPath:** (empty field)
- Parameters:**
 - [name=ns:userID], [type=string] → \${userID}
 - [name=ns:folderAbsolutePath], [type=string] → \${folderAbsolutePath}
 - [name=ns:password], [type=string] → \${password}

4. To test the Web service, see [Section 3.14, "Testing Data Models and Generating Sample Data."](#)

3.10.3 Additional Information on Web Service Data Sets

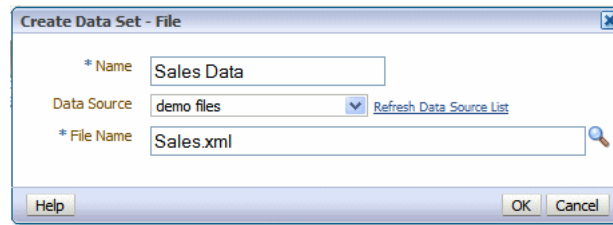
There is no metadata available from Web service data sets, therefore grouping and linking are not supported.

3.11 Creating a Data Set Using a Stored XML File

When you set up data sources, you can define a file directory as a data source. For information, see the section "Setting Up a Connection to a File Data Source" in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*. You can then place XML documents in the file directory to access directly as data sources for the reports.

To create a data set using a stored XML file:

1. Click the **Create new** toolbar button and select XML. The Create Data Set - File dialog launches, as shown in [Figure 3–24](#).

Figure 3–24 Create Data Set - File Dialog

2. Enter a name for this data set.
3. Select the **Data Source** where the XML file resides. The list is populated from the configured File Data Source connections.
4. Click **Browse** to connect to the data source and browse the available directories. Select the file to use for this report.
5. Click **OK**.

3.11.1 Additional Information on File Data Sets

There is no metadata available from XML file data sets, therefore grouping and linking are not supported.

3.12 Using Data Stored as a Character Large Object (CLOB) in a Data Model

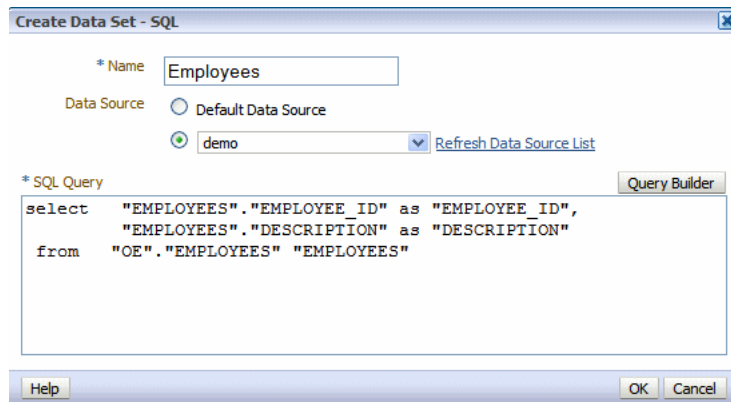
BI Publisher supports using data stored as a character large object (CLOB) data type in your data models. This feature enables you to use XML data generated by a separate process and stored in your database as input to a BI Publisher data model.

Use the Query Builder to retrieve the column in your SQL query, then use the data model editor to specify how you want the data structured. When the data model is executed, the data engine can structure the data either as:

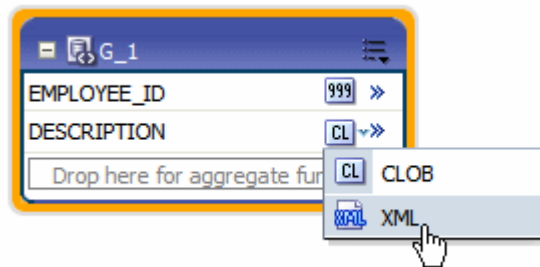
- A plain character set within an XML tag name that can be displayed in a report (for example, an Item Description)
- Structured XML

To create a data set from data stored as a CLOB:

1. Click the **New Data Set** icon and then click **SQL Query**. The Create Data Set - SQL dialog launches.
2. Enter a name for this data set.
3. If you are not using the default data source for this data set, select the **Data Source** from the list.
4. Enter the SQL query or use the **Query Builder** to construct your query to retrieve the CLOB data column. See [Section 3.4, "Using the Query Builder"](#) for information on the Query Builder utility. [Figure 3–25](#) shows an example query in which the CLOB data is stored in a column named "DESCRIPTION".

Figure 3–25 Sample Query

5. After entering the query, click **OK** to save. BI Publisher validates the query.
6. By default, the data model editor assigns the CLOB column the "CLOB" data type. To change the data type to XML, click the data type icon and select XML, as shown in [Figure 3–26](#).

Figure 3–26 Changing the Data Type to XML

3.12.1 How the Data Is Returned

When you execute the query, if the CLOB column contains well-formed XML, and you select the XML data type, the data engine returns the XML data, structured within the CLOB column tag name.

Example output when data type is XML

Note the <DESCRIPTION> element contains the XML data stored in the CLOB column, as shown in [Figure 3–27](#).

Figure 3–27 Example Data Structure When the Data Type is XML

```

<!--Generated by Oracle BI Publisher 11.1.1.4.0-->
-<DATA_DS>
  -<G_1>
    <EMPLOYEE_ID>102</EMPLOYEE_ID>
    -<DESCRIPTION>
      -<DATA_DS>
        -<G_Q1>
          <DEPTNO>10</DEPTNO>
          <DNAME>PURCHASE</DNAME>
          <LOC>HQ</LOC>
        -<G_Q2>
          <DEPTNO_1>10</DEPTNO_1>
          <EMPNO>10001</EMPNO>
          <ENAME>SCOTT</ENAME>
          <SAL>5000</SAL>
        </G_Q2>
        +<G_Q2></G_Q2>
      </G_Q1>
    -<G_Q1>
      <DEPTNO>20</DEPTNO>
      <DNAME>FINANCE</DNAME>
      <LOC>HQ</LOC>

```

Example output when data type is CLOB

If you select to return the data as the CLOB data type, the returned data is structured as shown in [Figure 3–28](#).

Figure 3–28 Example Data Structure When Data Type Is CLOB

```

<!--Generated by Oracle BI Publisher 11.1.1.4.0-->
-<DATA_DS>
  -<G_1>
    <EMPLOYEE_ID>102</EMPLOYEE_ID>
    -<DESCRIPTION>
      <DATA_DS> <G_Q1> <DEPTNO>10</DEPTNO>
      <DNAME>PURCHASE</DNAME> <LOC>HQ</LOC> <G_Q2> <DEPTNO_1>10</DEPTNO_1>
      <EMPNO>10001</EMPNO> <ENAME>SCOTT</ENAME> <SAL>5000</SAL> </G_Q2> <G_Q2>
      <DEPTNO_1>10</DEPTNO_1> <EMPNO>10002</EMPNO> <ENAME>SMITH</ENAME>
      <SAL>3000</SAL> </G_Q2> </G_Q1> <G_Q1> <DEPTNO>20</DEPTNO>
      <DNAME>FINANCE</DNAME> <LOC>HQ</LOC> <G_Q2> <DEPTNO_1>20</DEPTNO_1>
      <EMPNO>10003</EMPNO> <ENAME>AMY</ENAME> <SAL>5500</SAL> </G_Q2> <G_Q2>
      <DEPTNO_1>20</DEPTNO_1> <EMPNO>10004</EMPNO> <ENAME>MARLIN</ENAME>
      <SAL>4000</SAL> </G_Q2> </G_Q1> <G_Q1> <DEPTNO>30</DEPTNO>
      <DNAME>CORPORATE</DNAME> <LOC>HQ</LOC> </G_Q1> </DATA_DS>
    </DESCRIPTION>
  </G_1>
</DATA_DS>

```

3.12.1.1 Additional Notes on Data Sets Using CLOB Column Data

For specific notes on using CLOB column data in a bursting query, see [Section 8.3](#), "Adding a Bursting Definition to Your Data Model."

3.12.2 Handling XHTML Data Stored in a CLOB Column

BI Publisher can retrieve data stored in the form of XHTML documents stored in a database CLOB column and render the markup in the generated report. To enable the BI Publisher report rendering engine to handle the markup tags, you must wrap the XHTML data in a CDATA section within the XML report data that is passed by the data engine.

It is recommended that you store the data in the database wrapped with the CDATA section. You can then use a simple select statement to extract the data. If the data is not wrapped in the CDATA section, then you must include in your SQL statement instructions to wrap it.

The following sections describe how to extract XHTML data in each case:

- [Retrieving XHTML Data Wrapped in CDATA](#)
- [Wrapping the XHTML Data in CDATA in the Query](#)

To display the markup in a report, you must use the syntax described in "Rendering HTML Formatted Data in a Report" in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*. This section also describes the supported HTML formats. Rendering the HTML markup in a report is supported for RTF templates only.

3.12.2.1 Retrieving XHTML Data Wrapped in CDATA

Assume you have the following data stored in a database column called "CLOB_DATA":

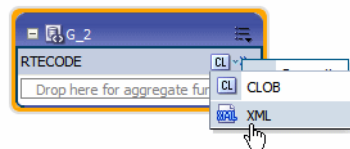
```
<![CDATA[
<font style="font-style: italic; font-weight: bold;" size="3"><a
href="http://www.oracle.com">oracle</a></font> <br/>
<font size="6"><a href="www.oracle.com">www.oracle.com</a></font><br/><br/>
]]>
```

Retrieve the column data using a simple SQL statement, for example:

```
select CLOB_DATA as "RTECODE"
  from MYTABLE
```

In the data model editor, set the data type of the RTECODE column to XML, as shown in [Figure 3–29](#).

Figure 3–29 Set Data Type to XML



3.12.2.2 Wrapping the XHTML Data in CDATA in the Query

Assume you have the following data stored in a database column called "CLOB_DATA":

```
<font style="font-style: italic; font-weight: bold;" size="3"><a
href="http://www.oracle.com">oracle</a></font> <br/>
<font size="6"><a href="www.oracle.com">www.oracle.com</a></font><br/><br/>
```

Use the following syntax in your SQL query to retrieve it and wrap it in the CDATA section:

```
select '<![CDATA' || '[' || CLOB_DATA || ']' || '>' as "RTECODE"
  from MYTABLE
```

In the data model editor, set the data type of the RTECODE column to XML, as shown in [Figure 3–29](#).

3.13 Creating a Data Set from an HTTP XML Feed

Using the HTTP (XML Feed) data set type you can create data models from RSS and XML feeds over the Web by retrieving data through the HTTP GET method.

Important: Additional configuration might be required to access external data source feeds depending on your system's security. If the RSS feed is protected by Secure Sockets Layer (SSL) then see the section "Configuring BI Publisher for Secure Sockets Layer Communication" in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Publisher*.

Note that to include parameters for the data set, you must define the parameters first, so that they are available for selection when defining the data set. See [Chapter 5, "Adding Parameters and Lists of Values."](#)

To create a data set from an HTTP XML feed:

1. Click the **New Data Set** toolbar button and select HTTP (XML Feed). The Create Data Set - HTTP dialog launches, as shown in [Figure 3–30](#).

Figure 3–30 Create Data Set - HTTP Dialog

The screenshot shows the 'Create Data Set - HTTP' dialog box. It contains the following fields and controls:

- Name:** A text box containing 'News'.
- URL:** A text box containing 'http://rss.news.yahoo.com/rss/topstories'.
- Method:** A dropdown menu set to 'GET'.
- Username:** An empty text box.
- Password:** An empty text box.
- Realm:** An empty text box.
- Parameters:** A section containing an 'Add Parameter' button and a table with two columns: 'Name' and 'Value (Parameter)'.
- Buttons:** 'Help', 'OK', and 'Cancel' buttons at the bottom.

2. Enter a name for this data set.
3. Enter the URL for the source of the RSS or XML feed.
4. Select the Method: Get.
5. Enter the Username, Password, and Realm for the URL, if required.
6. To add a parameter, click **Add Parameter**. Enter the **Name** and select the **Value**. The Value list is populated by the parameter Name defined in the Parameters section. See [Chapter 5, "Adding Parameters and Lists of Values."](#)
7. Click **OK** to close the data set dialog.

3.13.1 Additional Information on Data Sets Created from HTTP XML Feed

There is no metadata available from HTTP XML feed data sets, therefore grouping and linking are not supported.

3.14 Testing Data Models and Generating Sample Data

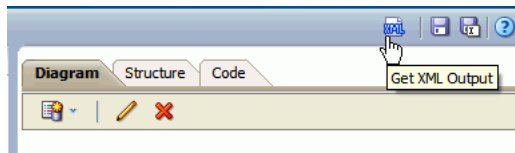
The Data Model Editor enables you to test your data model and view the output to ensure your results are as expected. After running a successful test, you can choose to save the test output as sample data for your data model, or export the file to an external location. If your data model fails to run, you can view the data engine log.

Important: For Safari browser users: The Safari browser renders XML as text. To view the XML generated by the data engine as XML, right-click inside the frame displaying the data and then click **View Frame Source**. This is a display issue only. The data is saved properly when you click **Save as Sample Data**.

To test your data model:

1. Click the **Get XML Output** toolbar button, as shown in [Figure 3–31](#).

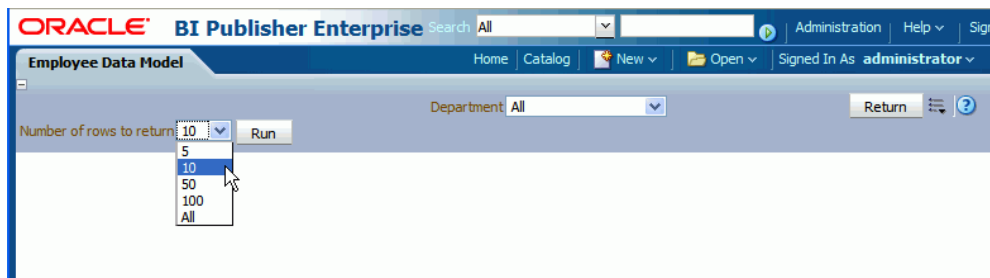
Figure 3–31 Get XML Output Button



This launches the **XML Output** page.

2. Select the number of rows to return. If you included parameters, enter the desired values for the test.

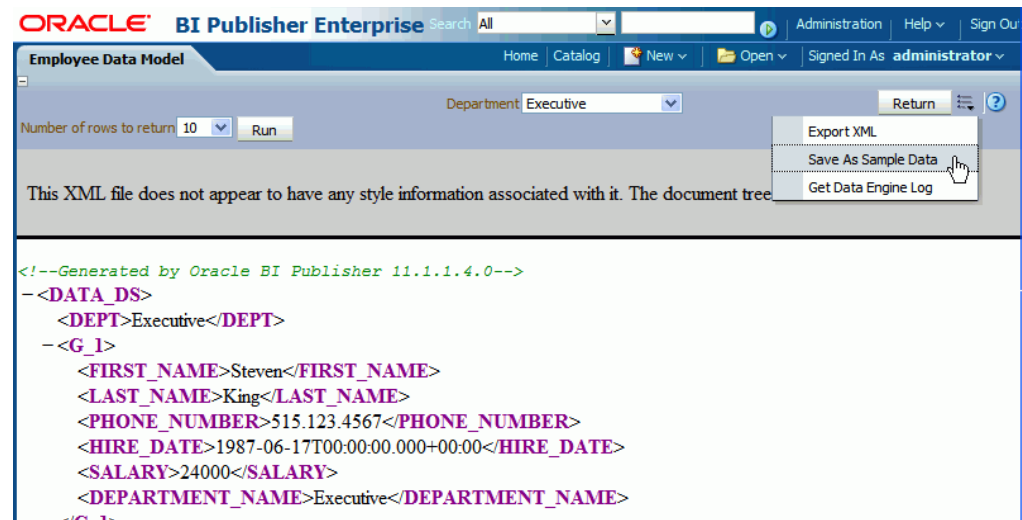
Figure 3–32 Select the Number of Rows to Return



3. Click **Run** to display the XML that is returned by the data model.

To save the test data set as sample data for the data model:

1. After the data model has successfully run, click the **Options** toolbar button and then click **Save as Sample Data**, as shown in [Figure 3–33](#). This sample data is saved to the data model. See [Section 2.8.2, "Attachments to the Data Model"](#) for more information.

Figure 3–33 Save as Sample Data

To export the test data:

1. After the data model has successfully run, select the **Options** toolbar button and then select **Export XML**. You are prompted to save the file.

To view the data engine log:

1. Select the **Options** toolbar button and then select **Get Data Engine Log**. You are prompted to open or save the file. The data engine log file is an XML file.

3.15 Including User Information Stored in System Variables in Your Report Data

BI Publisher stores information about the current user that can be accessed by your report data model. The user information is stored in system variables as described in [Table 3–2](#).

Table 3–2 User Information Stored in Variables

System Variable	Description
xdo_user_name	User ID of the user submitting the report. For example: Administrator
xdo_user_roles	Roles assigned to the user submitting the report. For example: XMLP_ADMIN, XMLP_SCHEDULER
xdo_user_report_oracle_lang	Report language from the user's account preferences. For example: ZHS
xdo_user_report_locale	Report locale from the user's account preferences. For example: en-US
xdo_user_ui_oracle_lang	User interface language from the user's account preferences. For example: US
xdo_user_ui_locale	User interface locale from the user's account preferences. For example: en-US

3.15.1 Adding the User System Variables as Elements

To add the user information to the data model, you can define the variables as parameters and then define the parameter value as an element in your data model. Or, you can simply add the variables as parameters then reference the parameter values in your report.

The following query:

```
select
:xdo_user_name as USER_ID,
:xdo_user_roles as USER_ROLES,
:xdo_user_report_oracle_lang as REPORT_LANGUAGE,
:xdo_user_report_locale as REPORT_LOCALE,
:xdo_user_ui_oracle_lang as UI_LANGUAGE,
:xdo_user_ui_locale as UI_LOCALE
from dual
```

returns the following results:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by Oracle BI Publisher -->
<DATA_DS>
<G_1>
<USER_ROLES>XMLP_TEMPLATE_DESIGNER, XMLP_DEVELOPER, XMLP_ANALYZER_EXCEL, XMLP_
ADMIN, XMLP_ANALYZER_ONLINE, XMLP_SCHEDULER </USER_ROLES>
<REPORT_LANGUAGE>US</REPORT_LANGUAGE>
<REPORT_LOCALE>en_US</REPORT_LOCALE>
<UI_LANGUAGE>US</UI_LANGUAGE>
<UI_LOCALE>en_US</UI_LOCALE>
<USER_ID>administrator</USER_ID>
</G_1>
</DATA_DS>
```

3.15.2 Sample Use Case: Limit the Returned Data Set by User ID

The following example limits the data returned by the user ID:

```
selectthroughEMPLOYEES.LAST_NAME as LAST_NAME,
      EMPLOYEES.PHONE_NUMBER as PHONE_NUMBER,
      EMPLOYEES.HIRE_DATE as HIRE_DATE,
      :xdo_user_name as USERID
from   HR.EMPLOYEES EMPLOYEES
where  lower(EMPLOYEES.LAST_NAME) = :xdo_user_name
```

Notice the use of the lower() function, the xdo_user_name is always be in lowercase format. BI Publisher does not have a USERID so you must use the user name and either use it directly in the query; or alternatively you could query against a lookup table to find a user id.

3.15.2.1 Creating Bind Variables from LDAP User Attribute Values

To bind user attribute values stored in your LDAP directory to a data query you can define the attribute names to BI Publisher to create the bind variables required.

3.15.2.1.1 Prerequisite The attributes that can be used to create bind variables must be defined in the Security Configuration page by an administrator. The attributes are defined in the Attribute Names for Data Query Bind Variables field of the LDAP Security Model definition. See the section "Configuring BI Publisher to Recognize the LDAP Server" in *Oracle Fusion Middleware Administrator's Guide for Oracle Business*

Intelligence Publisher for information about this field. Any attribute defined for users can be used (for example: `memberOf`, `sAMAccountName`, `primaryGroupID`, `mail`).

3.15.2.1.2 How BI Publisher Constructs the Bind Variable

You can reference the attribute names that you enter in the Attribute Names for Data Query Bind Variables field of the LDAP Security Model definition in the query as follows:

```
xdo_<attribute name>
```

Assume that you have entered the sample attributes: `memberOf`, `sAMAccountName`, `primaryGroupID`, `mail`. These can then be used in a query as the following bind variables:

```
xdo_memberof
xdo_SAMACCOUNTNAME
xdo_primaryGroupID
xdo_mail
```

Note that the case of the attribute is ignored; however, the "xdo_" prefix must be lowercase.

Use these in a data model as follows:

```
SELECT
:xdo_user_name AS USER_NAME ,
:xdo_user_roles AS USER_ROLES,
:xdo_user_ui_oracle_lang AS USER_UI_LANG,
:xdo_user_report_oracle_lang AS USER_REPORT_LANG,
:xdo_user_ui_locale AS USER_UI_LOCALE,
:xdo_user_report_locale AS USER_REPORT_LOCALE,
:xdo_SAMACCOUNTNAME AS SAMACCOUNTNAME,
:xdo_memberof as MEMBER_OF,
:xdo_primaryGroupID as PRIMARY_GROUP_ID,
:xdo_mail as MAIL
FROM DUAL
```

The LDAP bind variables return the values stored in the LDAP directory for the user that is logged in.

Structuring Data

This chapter describes techniques for structuring the data that is returned by BI Publisher's data engine. It covers the following topics:

- [Section 4.1, "Working with Data Models"](#)
- [Section 4.2, "Features of the Data Model Editor"](#)
- [Section 4.3, "About the Interface"](#)
- [Section 4.4, "Creating Links Between Data Sets"](#)
- [Section 4.5, "Creating Element-Level Links"](#)
- [Section 4.6, "Creating Group-Level Links"](#)
- [Section 4.7, "Creating Subgroups"](#)
- [Section 4.8, "Moving an Element Between a Parent Group and a Child Group"](#)
- [Section 4.9, "Creating Group-Level Aggregate Elements"](#)
- [Section 4.10, "Creating Group Filters"](#)
- [Section 4.11, "Performing Element-Level Functions"](#)
- [Section 4.12, "Setting Element Properties"](#)
- [Section 4.13, "Sorting Data"](#)
- [Section 4.14, "Performing Group-Level Functions"](#)
- [Section 4.15, "Performing Global-Level Functions"](#)
- [Section 4.16, "Using the Structure View to Edit Your Data Structure"](#)
- [Section 4.17, "Function Reference"](#)

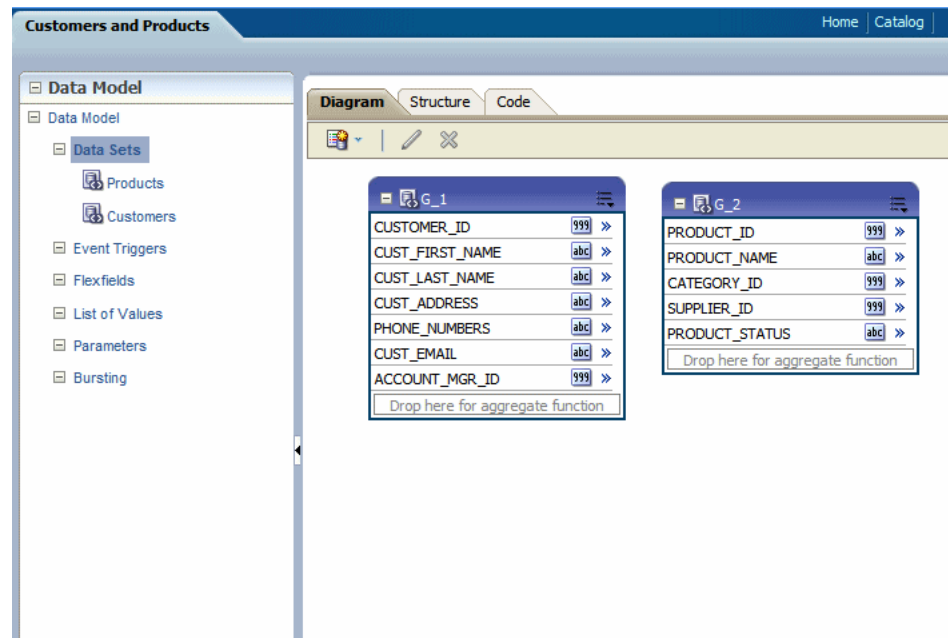
4.1 Working with Data Models

The Data Model diagram helps you to quickly and easily define data sets, break groups, and totals for a report based on multiple data sets.

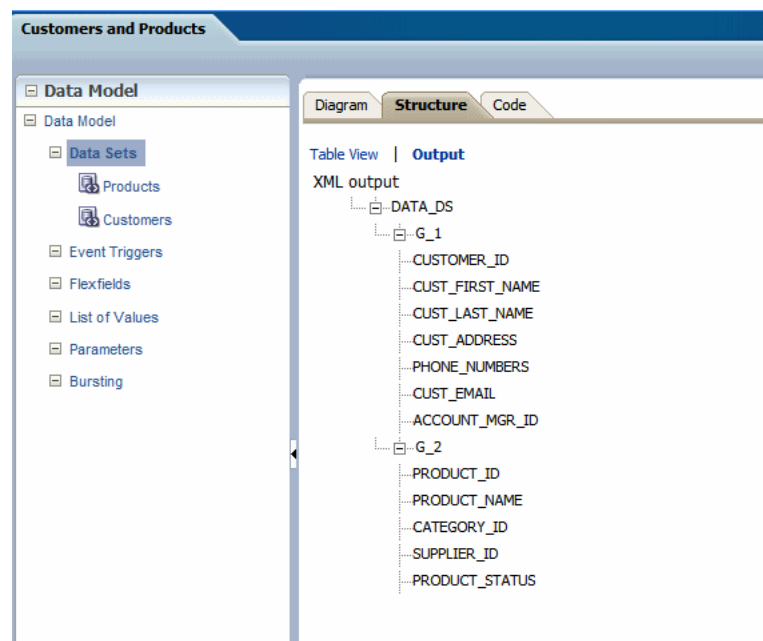
4.1.1 About Multipart Unrelated Data Sets

If you do not link the data sets (or queries) the data engine produces a multipart unrelated query data set.

For example, in the data model shown in [Figure 4-1](#), one query selects products and another selects customers. Notice that there is no relationship between the products and customers.

Figure 4–1 Multipart Unrelated Data Set

This results in the data structure shown in [Figure 4–2](#).

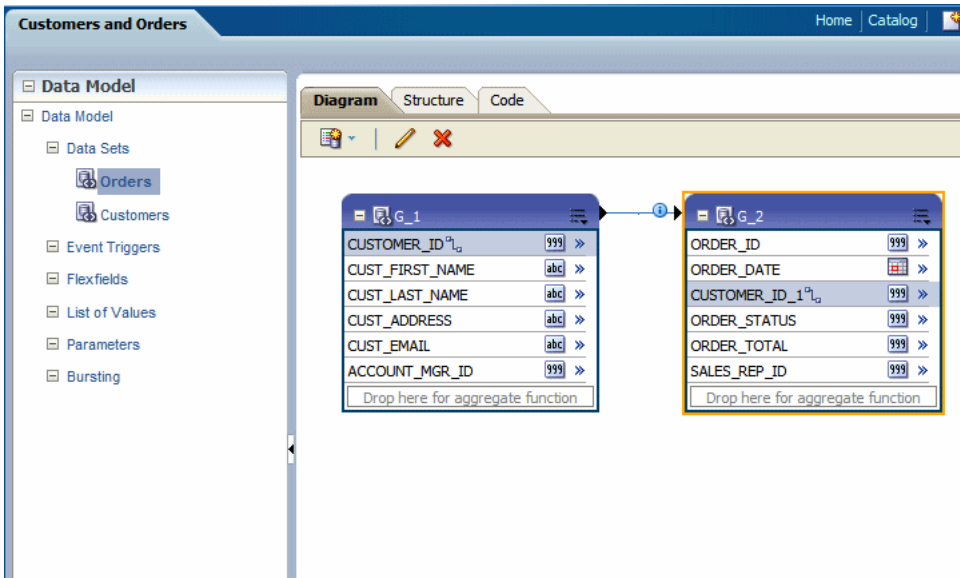
Figure 4–2 Data Structure of Multipart Unrelated Data Set

4.1.2 About Multipart Related Data Sets

In many cases, the data fetched for one part of the data set (or query) is determined by the data fetched for another part. This is often called a "master/detail," or "parent/child," relationship, and is defined with a data link between two data sets (or queries). When you run a master/detail data model, each row of the master (or parent) query causes the detail (or child) query to be executed, retrieving only matching rows.

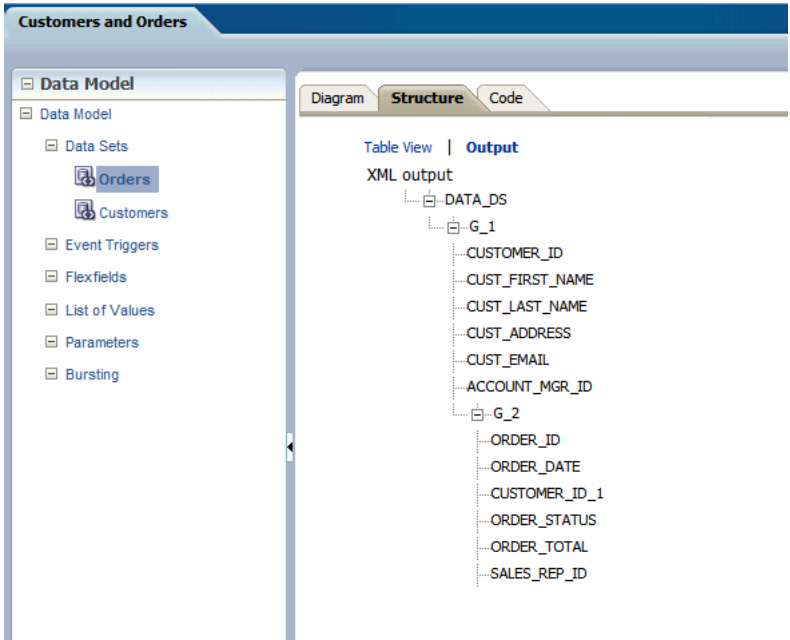
In the example shown in [Figure 4-3](#), two data sets are linked by the element Customer ID. The Orders data set a child of the Customers data set.

Figure 4-3 Multipart Related Data Sets



This produces the data structure shown in [Figure 4-4](#).

Figure 4-4 Data Structure of Multipart Related Data Set



4.1.3 Guidelines for Working with Data Sets

Following are recommended guidelines for building data models:

- Reduce the number of data sets or queries in your data model as much as possible. In general, the fewer data sets/queries you have, the faster your data model will run. While multiquery data models are often easier to understand, single-query

data models tend to execute more quickly. It is important to understand that in parent-child queries, for every parent, the child query is executed.

- You should only use multiquery data models in the following scenarios:
 - To perform functions that the query type, such as a SQL query, does not support directly.
 - To support complex views (for example, distributed queries or GROUP BY queries).
 - To simulate a view when you do not have or want to use a view.

4.2 Features of the Data Model Editor

The data model editor enables you to combine data from multiple data sets into a single XML data structure. Data sets from multiple data sources can be merged either as sequential XML or at line-level to create a single combined hierarchical XML. Using the data model editor you can easily combine data from the following data set types: SQL query, OLAP (MDX query), LDAP, and Microsoft Excel.

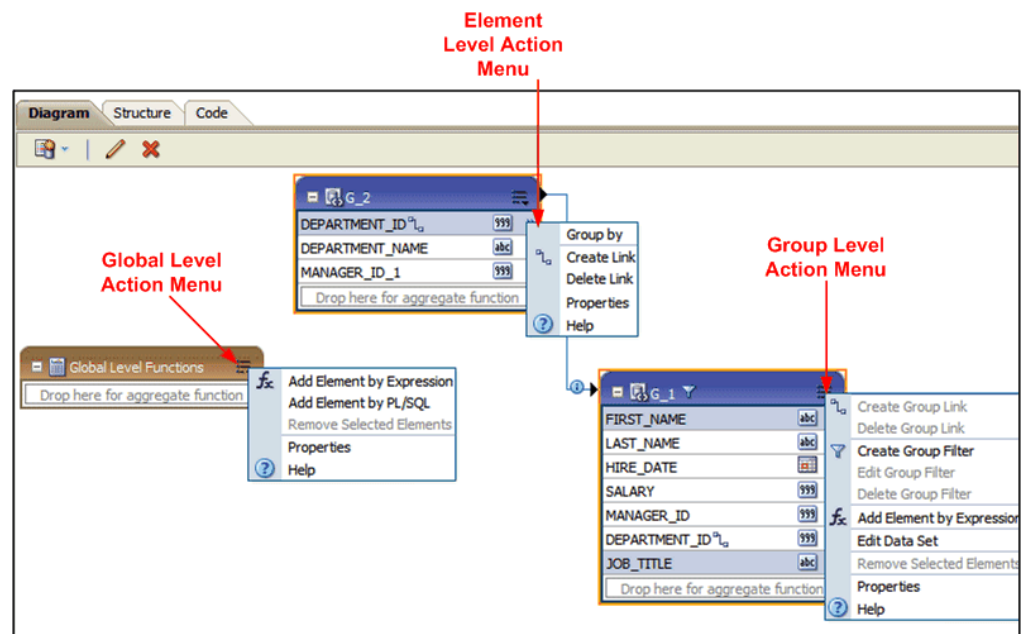
The data model editor supports the following

- **Group data**

Groups are created to organize the columns in your report. Groups can do two things: separate a query's data into sets, and filter a query's data. When you create a query, the data engine creates a group that contains the columns selected by the query; you can create groups to modify the hierarchy of the data appearing in a data model. Groups are used primarily when you want to treat some columns differently than others. For example, you create groups to produce subtotals or create breaks.
- **Link data** — Define master-detail links between data sets to group data at multiple levels.
- **Aggregate data** — Create group level totals and subtotals.
- **Transform data** — Modify source data to conform to business terms and reporting requirements.
- **Create calculations** — Compute data values that are required for your report that are not available in the underlying data sources.

The data model editor provides functions at the element level, the group level, and the global level. Note that not all data set types support all functions. See the Important Notes section that accompanies your data set type for limitations. [Figure 4-5](#) highlights some of the features and actions available in the data model editor.

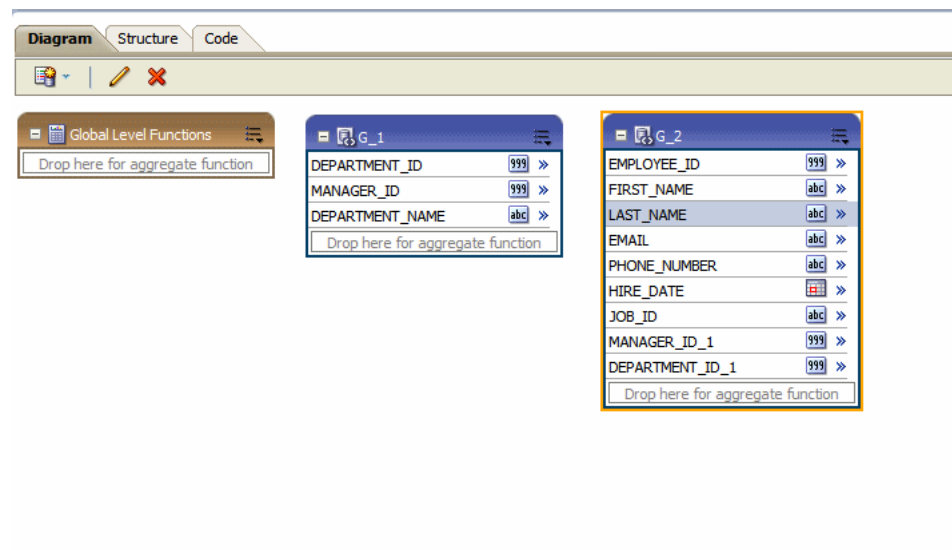
Figure 4–5 Features of Data Model Editor



4.3 About the Interface

By default, the data sets that you created are shown in the Diagram View as separate objects, as seen in Figure 4–6.

Figure 4–6 Diagram View



The data set structure builder has three views:

- **Diagram View** — (Shown in Figure 4–6) This view displays your data sets and enables you to graphically create links and filters, add elements based on expressions, add aggregate functions and global-level functions, edit element properties, and delete elements. The Diagram View is typically the view you use to build your data structure.

- **Structure View** — This view has two modes:

Table View and Output

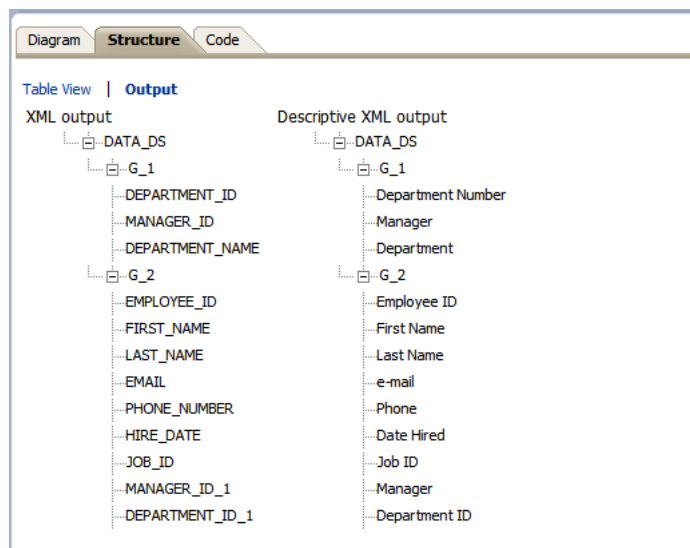
The table view displays element properties in a table and enables you to update XML element alias names, presentation names of the elements, sorting, null values, and reset options. [Figure 4–7](#) shows the structure Table View.

Figure 4–7 Structure Table View

Data Source	XML View			Business View	
	XML Tag Name	Sorting	Value If Null	Display Name	Data Type
Report Data					
Data Structure	DATA_DS				
G_Dept	G_1			G_1	
DEPARTMENT_ID	DEPARTMENT_ID			Department Number	999
MANAGER_ID	MANAGER_ID			Manager	999
DEPARTMENT_NAME	DEPARTMENT_NAME			Department	abc
G_EMP	G_2			G_2	
EMPLOYEE_ID	EMPLOYEE_ID			Employee ID	999
FIRST_NAME	FIRST_NAME			First Name	abc
LAST_NAME	LAST_NAME			Last Name	abc
EMAIL	EMAIL			e-mail	abc
PHONE_NUMBER	PHONE_NUMBER			Phone	abc
HIRE_DATE	HIRE_DATE			Date Hired	
JOB_ID	JOB_ID			Job ID	abc

The **Output** view provides a clear view of the XML structure that is generated. The Output view is not updatable. [Figure 4–8](#) shows the Output view.

Figure 4–8 Output View



- **Code View** — This view displays the data structure code created by the data structure builder that is read by the data engine. The code view is not updatable. [Figure 4–9](#) shows the code view.

Figure 4–9 Code View

```

<link name="link_2" parentGroup="G_2" parentColumn="DEPARTMENT_ID"
childQuery="employees" childColumn='&quot;EMPLOYEES&quot;.&quot;DEPARTMENT_ID&
quot;' childColumnAlias="DEPARTMENT_ID"/>
<output type="data-structure" >
<dataStructure tagName="DATA_DS">
  <group name="G_2" source="Departments" label="G_2" >
    <element name="DEPARTMENT_ID" value="DEPARTMENT_ID" dataType="xsd:integer"
label="DEPARTMENT_ID" breakOrder="None" fieldOrder="1"/>
    <element name="DEPARTMENT_NAME" value="DEPARTMENT_NAME"
dataType="xsd:string" label="DEPARTMENT_NAME" breakOrder="None" fieldOrder="2"/>
    <element name="MANAGER_ID" value="MANAGER_ID" dataType="xsd:integer"
label="MANAGER_ID" breakOrder="None" fieldOrder="3"/>
    <group name="G_1" source="employees" label="G_1" groupFilter="{G_1.SALARY==
(2*2200)}">
      <element name="FIRST_NAME" value="FIRST_NAME" dataType="xsd:string"
label="FIRST_NAME" breakOrder="None" fieldOrder="1"/>
      <element name="LAST_NAME" value="LAST_NAME" dataType="xsd:string"
label="LAST_NAME" breakOrder="None" fieldOrder="2"/>
      <element name="HIRE_DATE" value="HIRE_DATE" dataType="xsd:date"
label="HIRE_DATE" breakOrder="None" fieldOrder="3"/>
      <element name="SALARY" value="SALARY" dataType="xsd:double"

```

4.4 Creating Links Between Data Sets

Joining and structuring data at the source into one combined data set is sometimes not possible. For example, you cannot join data at the source when data resides in disparate sources such as Microsoft SQL Server and an Oracle database. You can use the BI Publisher data engine to combine and structure data after you extract it from the data source. Even if your data is coming from the same source, if you are creating large reports or documents with potentially hundreds of thousands of rows or pages, structuring your data so that it matches the intended layout optimizes document generation.

Create a link to define a master-detail (or parent-child) relationship between two data sets. You can create links as element-level links or group-level links. The resulting, hierarchical XML data is the same. Creating links as element-level links is the preferred method. Group-level links are provided for backward compatibility with data templates from earlier versions of BI Publisher.

A data link (or parent-child relationship) relates the results of multiple queries. A data link can establish these relationships:

- Between one query's column and another query's column
- Between one query's group and another query's group (this is useful when you want the child query to know about its parent's data)

4.4.1 About Element-Level Links

Element-level links create a bind (join) between two data sets and define a master-detail (parent-child) relationship between them. This is the preferred method of defining master detail relationships between data sets. The simplest way to link data sets is by creating element-level links because they do not require you to code a join between the two data sets through a bind variable.

4.4.2 About Group-Level Links

Group level links also determine the way data sets are structured as hierarchical XML, but lack the join information that the data engine needs to execute the master and

detail queries. When you define a group-level link, you must also update your query with a link between the two data sets through a unique bind variable.

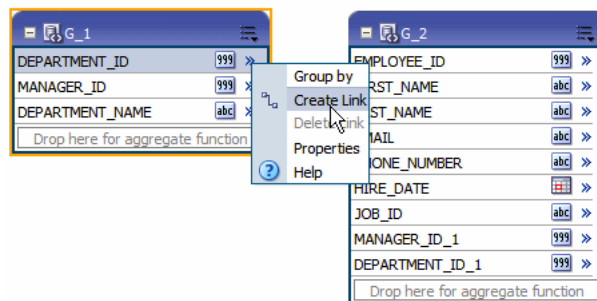
4.5 Creating Element-Level Links

Link data sets to define a master-detail (or parent-child) relationship between two data sets. Defining an element-level link enables you to establish the binding between the elements of the master and detail data sets.

To define an element-level link, do one of the following:

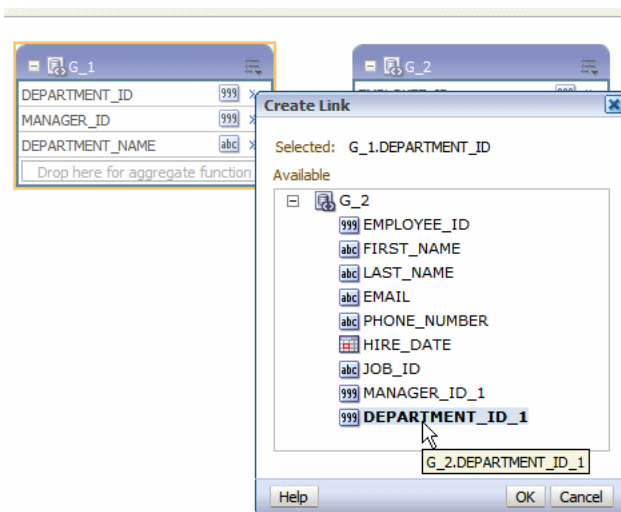
- Open the element action menu and click **Create Link**.

Figure 4–10 *Creating a Link Using the Element Action Menu*

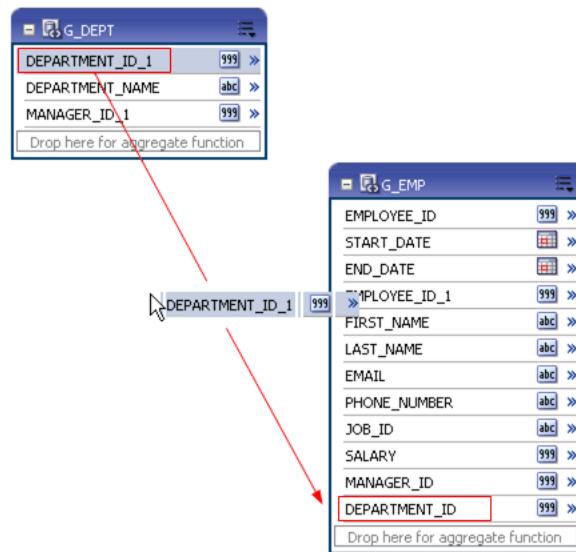


The **Create Link** dialog launches and displays the elements from the other data sets. Choose the element and click **OK** to create the link. The Create Link dialog is shown in [Figure 4–11](#).

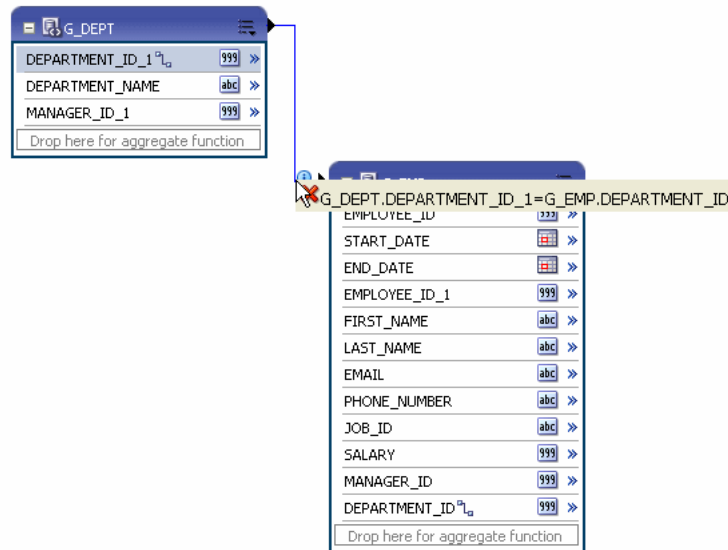
Figure 4–11 *Create Link Dialog*



- Alternatively, from the parent group, click and drag the element you want to bind to the matching element in the child group, as shown in [Figure 4–12](#).

Figure 4–12 Creating a Link by Dragging and Dropping the Bind Element

- After dropping the element from the parent data set to the matching element on the child data set, a connector displays between the data sets. Pause your cursor over the connector to display the link (as shown in [Figure 4–13](#)).

Figure 4–13 Displaying the Link

4.5.1 Deleting Element-Level Links

To delete an element link:

1. Pause your cursor over the element connector to display the linked element names and the delete button.
2. Click the delete button.

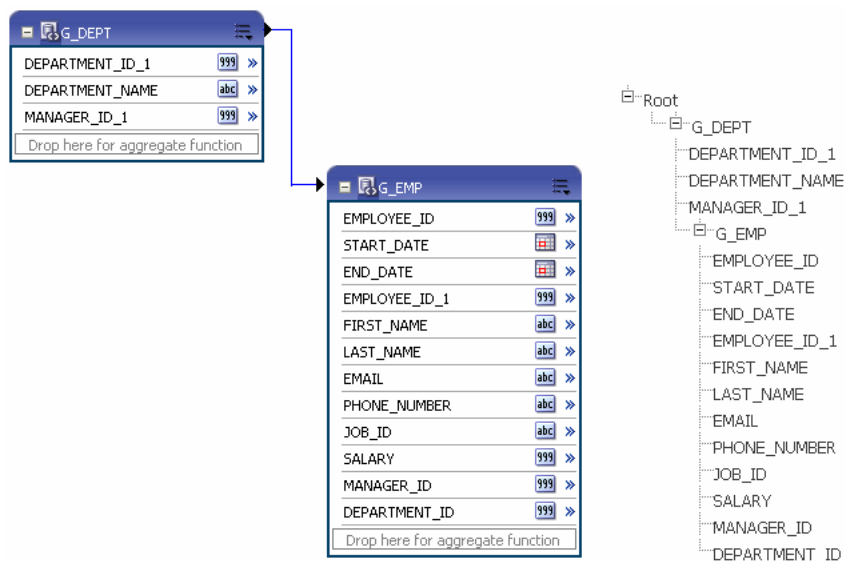
Or, alternatively:

1. Open the element action menu for either element and click **Delete Link**.

4.6 Creating Group-Level Links

A group-level link defines a master-detail relationship between two data sets. The following figure shows two data sets with a group-level link defined. Next to the data sets the resulting XML data structure is shown, as in [Figure 4-14](#).

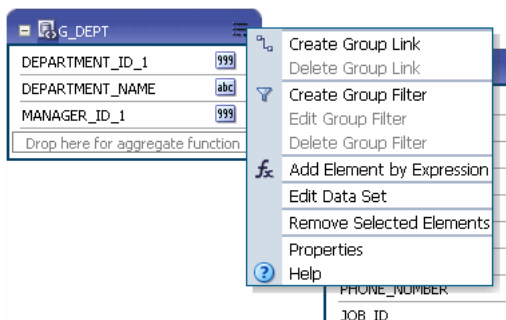
Figure 4-14 Resulting XML Data Structure



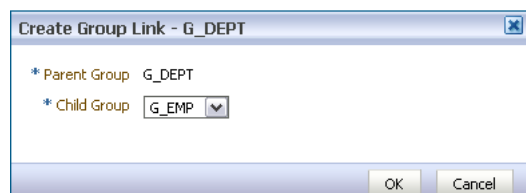
To define a group-level link:

1. In the parent group, click the **View Actions** menu (in the upper right corner of the object).
2. Click **Create Group Link** as shown in [Figure 4-15](#).

Figure 4-15 Creating a Group Link



3. In the **Create Group Link** dialog, select the Child Group from the list and click OK. The **Create Group Link** dialog is shown in [Figure 4-16](#).

Figure 4–16 Create Group Link Dialog

4. Click the **View Actions** menu and then click **Edit Data Set** to add the bind variables to your query.

An example is shown in [Table 4–1](#).

Table 4–1 Example: Edit Data Set

Data Set: DEPT	Data Set: EMP
<pre>Select DEPT.DEPTNO as DEPTID, DEPT.DNAME as DNAME, DEPT.LOC as LOC from OE.DEPT DEPT</pre>	<pre>Select EMP.EMPNO as EMPNO, EMP.ENAME as ENAME, EMP.JOB as JOB, EMP.MGR as MGR, EMP.HIREDATE as HIREDATE, EMP.SAL as SAL, EMP.COMM as COMM, EMP.DEPTNO as DEPTNO from OE.EMP EMP where DEPTNO=:DEPTID</pre>

Important: A unique bind variable must be defined in the child query.

4.6.1 Deleting Group-Level Links

To delete a group link:

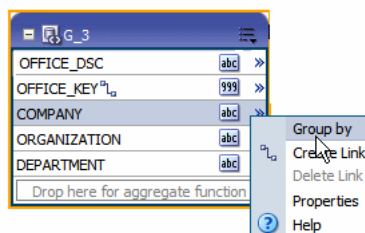
1. In the parent group, click the **View Actions** menu (in the upper right corner of the object).
2. Click **Delete Group Link**.
3. In the **Delete Group Link** dialog, select the **Child Group** from the list and click OK.

4.7 Creating Subgroups

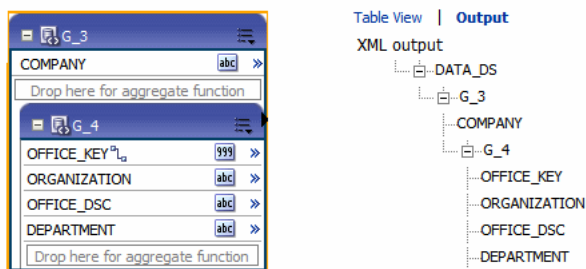
In addition to creating parent-child structures by linking two data sets, you can also group elements in the same data set by other elements. This might be helpful if your query returns data that has header data repeated for each detail row. By creating a subgroup you can shape the XML data for better more efficient document generation.

To create a subgroup:

1. Select the element by which you want to group the other elements in the data set.
2. Click the element action menu icon to open the menu and select **Group by** as shown in [Figure 4–17](#).

Figure 4–17 Creating a Subgroup

This creates a new group within the displayed data set. The following figure shows the G_3 data set grouped by the element COMPANY. This creates a new group called G_4, that contains the other four elements in the data set. [Figure 4–18](#) shows how the grouped data set is displayed in the Diagram View along with the structure.

Figure 4–18 Subgroup Data Structure

You can perform any of the group actions on the new group you have created.

To remove a subgroup:

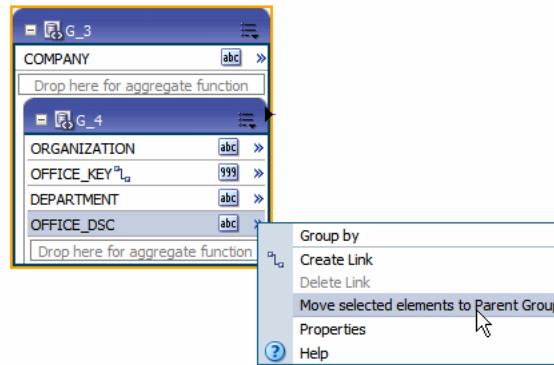
1. On the group's title bar, click **View Actions** and then click **Ungroup**.

4.8 Moving an Element Between a Parent Group and a Child Group

Once you have created a group within your data set, two new options display on the element action menu that enable you to move elements between the parent and child groups.

For the element that you want to move, click the element action icon to open the menu. If the element is in the parent group and you want to move it to the child group select **Move this element to Child Group**.

If the element is in the child group and you want to move it to the parent group select **Move this element to Parent Group**. In [Figure 4–19](#), the element action menu for OFFICE_DSC displays the option to move the element to the parent group.

Figure 4–19 *Moving Element from Child Group to Parent Group*

Important: Before moving an element be aware of any dependencies on other elements.

4.9 Creating Group-Level Aggregate Elements

You can use the data model editor to aggregate data at the group or report level. For example, if you group sales data by Customer Name, you can aggregate sales to get a subtotal for each customer's sales. Note that you can only aggregate data for at the parent level for a child element.

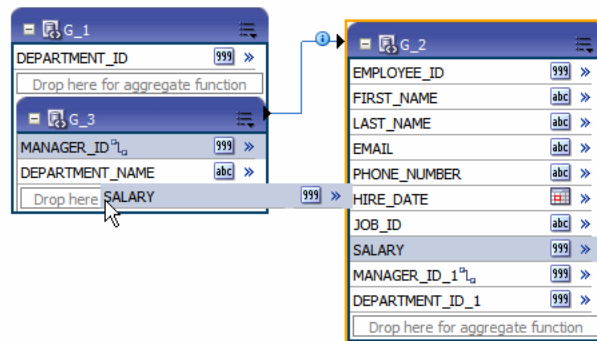
The aggregate functions are:

- **Average** — Calculates the average of all the occurrences of an element.
- **Count** — Counts the number of occurrences of an element.
- **First** — Displays the value of the first occurrence of an element in the group.
- **Last** — Displays the value of the last occurrence of an element in the group.
- **Maximum** — Displays the highest value of all occurrences of an element in the group.
- **Minimum** — Displays the lowest value of all occurrences of an element in a group.
- **Summary** — Sums the value of all occurrences of an element in the group.

To create group-level aggregate elements:

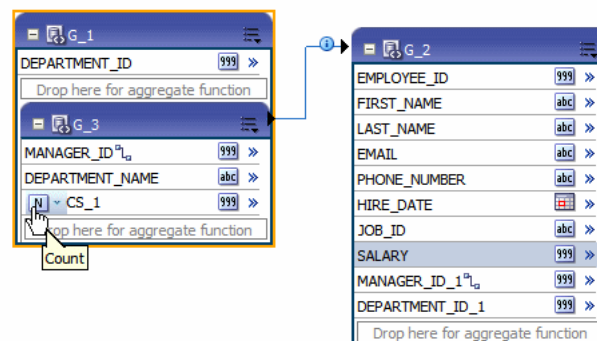
1. Drag the element to the **Drop here for aggregate function** field in the parent group.

[Figure 4–20](#) shows creating a group-level aggregate function in the G_DEPT based on the SALARY element.

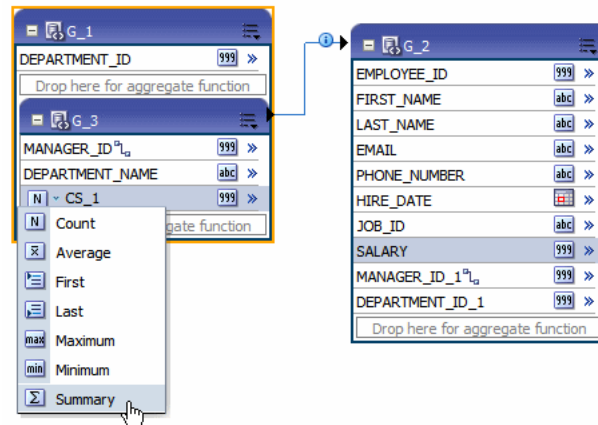
Figure 4–20 Creating a Group-Level Aggregate Function

Once you drop the element, a new element is created in the parent group. By default, the Count function is applied. The icon next to the name of the new aggregate element indicates the function. Pause your cursor over the icon to display the function.

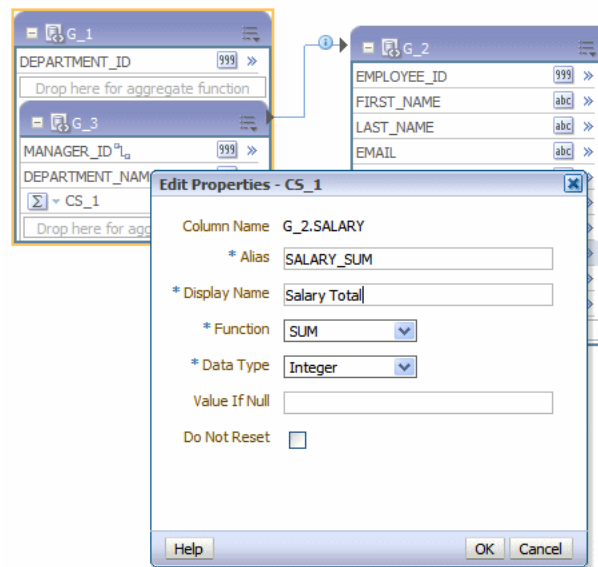
Figure 4–21 shows the new aggregate element, CS_1, with the default Count function defined.

Figure 4–21 New Element Created by Group-Level Aggregate Function

2. To change the function: Click the function icon to view a list of available functions and choose from the list, as shown in Figure 4–22.

Figure 4-22 *Choosing a Function*

3. To rename the element or update other properties, click the element's **Action** menu icon. On the menu, click **Properties**. The **Properties** dialog is shown in Figure 4-23.

Figure 4-23 *Properties Dialog*

Important: Be careful when renaming an element as it can have dependency on other elements.

Set the properties described in Table 4-2 as needed.

Table 4-2 *Element Properties*

Property	Description
Column Name	The internal name assigned to the element by the BI Publisher data model editor. This name cannot be updated.

Table 4–2 (Cont.) Element Properties

Property	Description
Alias (XML Tag Name)	BI Publisher assigns a default tag name that the element will have in the XML data file. You can update this tag name if you want a more meaningful name within the data file.
Display Name	The Display Name appears in the report design tools. Update this name to be meaningful to your business users.
Function	If you have not already selected the desired function, then you can select it from the list here.
Data Type	BI Publisher assigns a default data type of Integer or Double depending on the function. Some functions also provide the option of Float.
Value if Null	If the value returned from the function is null, you can supply a default value here to prevent having a null in your data.
Do Not Reset	By default, the function resets at the group level. For example, if your data set is grouped by DEPARTMENT_ID, and you have defined a sum function for SALARY, then the sum is reset for each group of DEPARTMENT_ID data, giving you the sum of SALARY for that department only. If instead you want the function to reset only at the global level, and not at the group level, select Do Not Reset . This creates a running total of SALARY for all departments. Note that this property is for group level functions only.

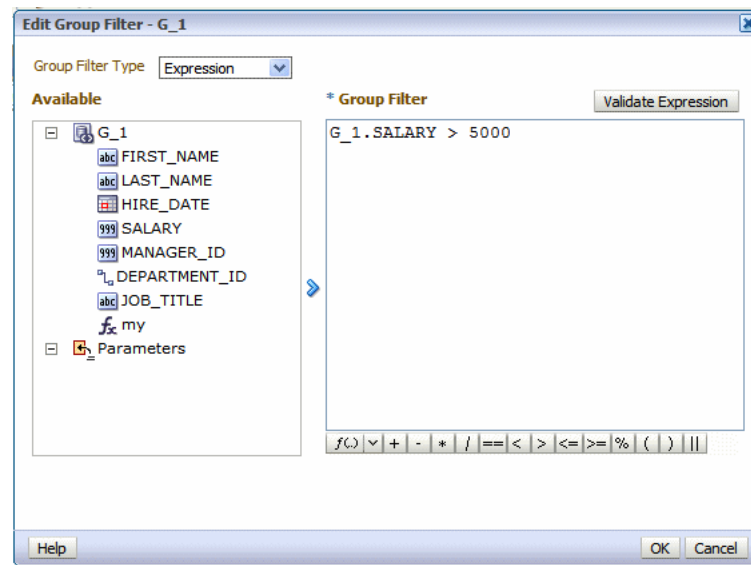
4.10 Creating Group Filters

Filters enable you to conditionally remove records selected by your queries. Groups can have two types of filters:

- Expression — Create an expression using predefined functions and operators
- PL/SQL Function — Create a custom filter

To create a group filter:

1. Click the **View Actions** menu and select **Create Group Filter**.
2. This displays the **Edit Group Filter** dialog, as shown in [Figure 4–24](#).

Figure 4–24 Edit Group Filter Dialog

3. Choose the Group Filter Type: **Expression** or **PL/SQL**.

Note: For PL/SQL filters, you must first specify the PL/SQL Package as the **Oracle DB Default Package** in the data model properties. See [Section 2.8, "Setting Data Model Properties."](#)

4. Enter the Filter:

- To enter an expression, select the elements and click the shuttle button to move the element to the Group Filter definition box. Click the predefined functions and operators to insert them in the Group Filter box.

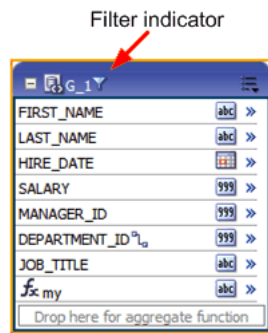
Refer to [Section 4.17, "Function Reference"](#) for a description of the available functions.

Click **Validate Expression** to ensure that the entry is valid.

- To enter a PL/SQL function, select the PL/SQL package from the Available box and click the shuttle button to move the function to the Group Filter box.

Your PL/SQL function in the default package must return a Boolean type.

After you have added the group filter, the data set object displays the filter indicator, as shown in [Figure 4–25](#).

Figure 4–25 Filter Indicator

To edit or delete a group filter:

1. Click the data set **View Actions** menu.
2. Choose the appropriate action:
 - To edit the group filter, choose **Edit Group Filter** to launch the Group Filter dialog for editing.
 - To delete the group filter, choose **Delete Group Filter**.

4.11 Performing Element-Level Functions

You can perform the following functions at the element level:

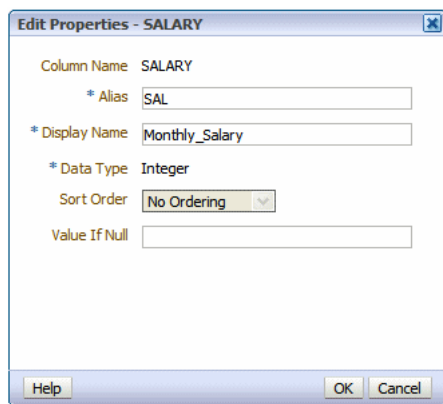
- Group by an element to create a subgroup, as described in [Section 4.7, "Creating Subgroups"](#)
- Create element-level links between data sets, as described in [Section 4.5, "Creating Element-Level Links"](#)
- Set element properties, as described in [Section 4.12, "Setting Element Properties"](#)

4.12 Setting Element Properties

You can set properties for individual elements. Note that these properties are also updatable from the Structure View. If you need to update multiple element properties, it may be more efficient to use the Structure View. See [Section 4.16, "Using the Structure View to Edit Your Data Structure."](#)

To set element-level properties using the element dialog:

1. Click the element's action menu icon. From the menu, select **Properties**. The Properties dialog is shown in [Figure 4–26](#).

Figure 4–26 Properties Dialog

2. Set the properties as needed, as described in [Table 4–3](#).

Table 4–3 Element Properties

Property	Description
Alias	BI Publisher assigns a default tag name that the element will have in the XML data file. You can update this tag name if you want a more meaningful name within the data file.
Display Name	The Display Name appears in the report design tools. Update this name to be meaningful to your business users.
Data Type	BI Publisher assigns a default data type of Integer or Double depending on the function. Some functions also provide the option of Float.
Sort Order	You can sort XML data in a group by one or more elements. For example, if in a data set employees are grouped by department and manager, you can sort the XML data by department. Within each department you can group and sort data by manager, and within each manager subgroup, employees can be sorted by salary. If the element is not in a parent group, the Sort Order property is not available.
Value if Null	If the value returned from the function is null, you can supply a default value here to prevent having a null in your data.

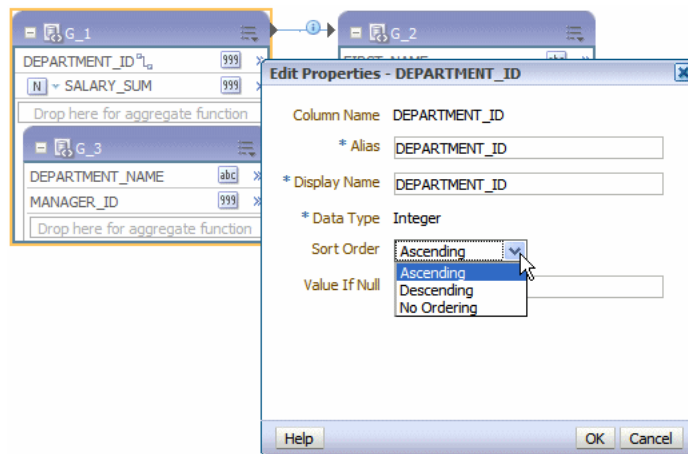
4.13 Sorting Data

Sorting is supported for parent group break columns only. For example, if a data set of employees is grouped by department and manager, you can sort the XML data by department. Within each department you can group and sort data by manager. If you know how the data should be sorted in the final report, you specify sorting at data generation time to optimize document generation.

To apply a sort order to a group:

1. Click the action menu icon of the element you want to sort by. From the menu, select **Properties**.
2. Select the **Sort Order**.

[Figure 4–27](#) shows the **Properties** dialog for the DEPARTMENT_ID element with the Sort Order list displayed.

Figure 4–27 Properties Dialog Showing Sort Order List

4.14 Performing Group-Level Functions

This section describes how to perform group-functions. It includes the following topics:

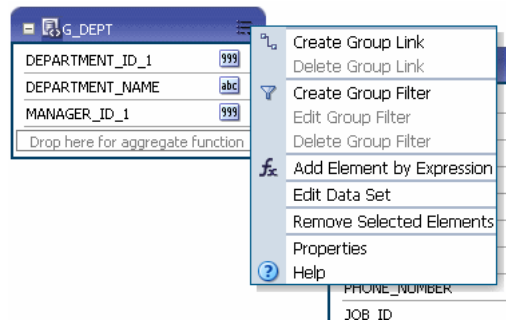
- [Section 4.14.1, "The Group Action Menu"](#)
- [Section 4.14.2, "Editing the Data Set"](#)
- [Section 4.14.3, "Removing Elements from the Group"](#)
- [Section 4.14.4, "Editing the Group Properties"](#)

4.14.1 The Group Action Menu

The **View Actions** menu available at the group level enables you to perform the following:

- Create and delete group links, as described in [Section 4.6, "Creating Group-Level Links"](#)
- Create, edit, and delete group filters, as described in [Section 4.10, "Creating Group Filters"](#)
- Add an element to the group based on an expression, as described in [Section 4.15.2, "Adding a Group-Level or Global-Level Element by Expression"](#)
- Edit the data set, as described in [Section 4.14.2, "Editing the Data Set"](#)
- Remove elements from the group, as described in [Section 4.14.3, "Removing Elements from the Group"](#)
- Edit group properties, as described in [Section 4.14.4, "Editing the Group Properties"](#)

The group-level Actions menu is shown in [Figure 4–28](#).

Figure 4–28 Group-Level Actions Menu

4.14.2 Editing the Data Set

To edit the underlying data set:

1. Click **Edit Data Set** to launch the data set editor.

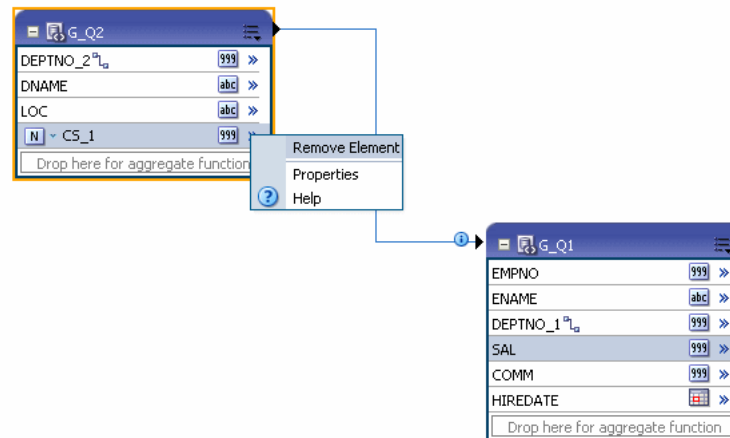
See the appropriate section for the data set type in [Chapter 3, "Creating Data Sets"](#) for more information.

4.14.3 Removing Elements from the Group

To remove an element from the group:

1. On the element row, click the menu and then click **Remove Element**. An example is shown in [Figure 4–29](#).

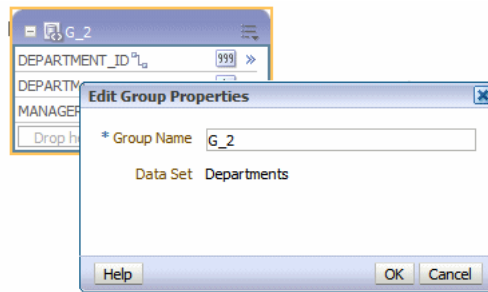
Note: You can only remove elements added as a group function (sum, count, and so on) or added as an expression.

Figure 4–29 Removing an Element

4.14.4 Editing the Group Properties

To edit the group properties:

1. Click the **View Actions** menu and select **Properties**.
2. Edit the **Group Name** and click OK, as shown in [Figure 4–30](#).

Figure 4–30 Edit the Group Name

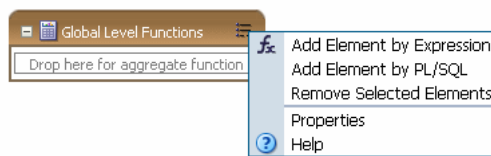
4.15 Performing Global-Level Functions

The Global Level Functions object enables you to add elements to your report data set at the top report level. You can add the following types of elements as top-level data:

- Elements based on aggregate functions
- Elements based on expressions
- Elements based on PL/SQL statements (for Oracle Database data sources)

Important: If you select a data type of Integer for any calculated element and the expression returns a fraction, the data is not truncated.

The **Global Level Functions** object is shown in [Figure 4–31](#). To add elements based on aggregate functions, drag the element to the "Drop here for aggregate function" space of the object. To add an element based on an expression or PL/SQL, choose the appropriate action from the **View Actions** menu.

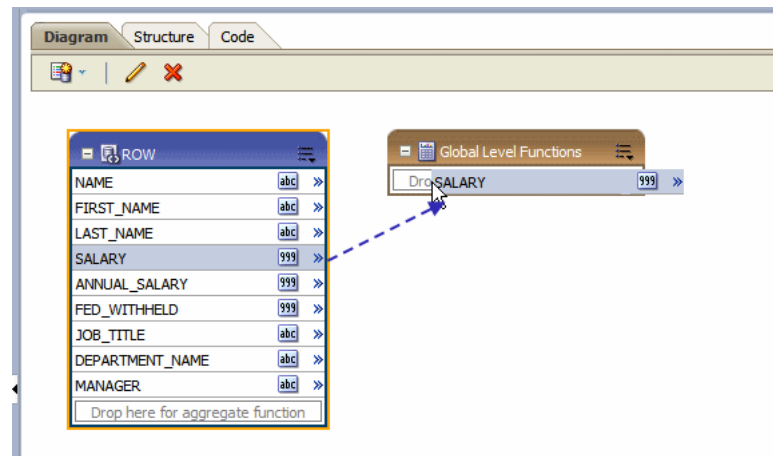
Figure 4–31 Global Level Functions Object

4.15.1 Adding a Global-Level Aggregate Function

To add a global aggregate function:

1. Drag and drop the data element from the data set to the "Drop here for aggregate function" area of the Global Level Functions object.

For example, [Figure 4–32](#) shows creating a global level aggregate function based on the Salary element.

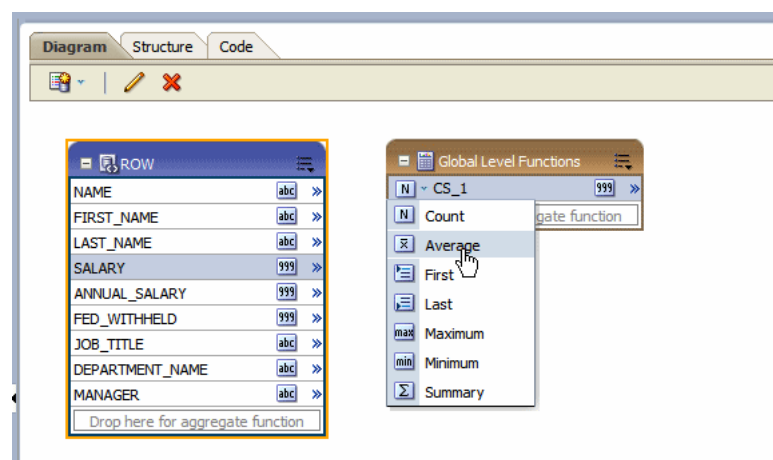
Figure 4–32 Creating Global-Level Aggregate Function

2. When you release the mouse, the data model editor assigns a default name to the aggregate element and assigns Count as the default function. Available functions are:

- Count
- Average
- First
- Last
- Maximum
- Minimum
- Summary

To change the function, click the function icon to the left of the new element name and choose the function from the list.

Figure 4–33 shows the function for the new global level element CS_1 being modified from Count to Average.

Figure 4–33 Applying a Function

- 3. To change the default name, click the actions icon to the right of the element name and click **Properties** to launch the **Edit Properties** dialog, See [Section 4.12, "Setting Element Properties."](#) for more about the properties available on this dialog.

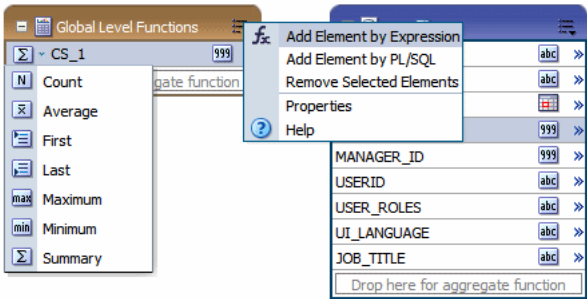
4.15.2 Adding a Group-Level or Global-Level Element by Expression

To add a group-level or global-level element by expression:

- 1. To add a group-level element: On the **Group** object, click the **View Actions** menu and select **Add Element by Expression**.

To add a global level element: On the **Global Level Functions** object, click the **View Actions** menu and select **Add Element: by Expression**, as shown in [Figure 4–34](#).

Figure 4–34 Add Element: by Expression



- 2. In the **Add Element by Expression** dialog, enter the fields, as shown in [Figure 4–35](#).

Figure 4–35 Add Element by Expression Dialog

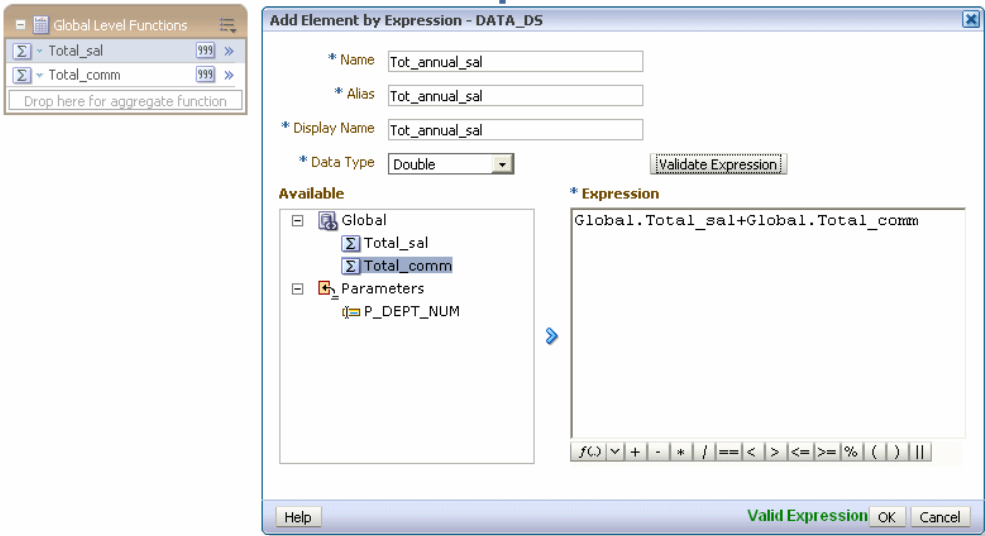


Table 4–4 Add Element by Expression Dialog Fields

Field	Description
Name	Enter a name for this element.

Table 4–4 (Cont.) Add Element by Expression Dialog Fields

Field	Description
Alias	Enter the tag name that the element has in the XML data file.
Display Name	The Display Name appears in the report design tools. Enter a name that is meaningful to your business users.
Data Type	Select from the list of data types: String, Integer, Double, Float, or Date.

3. Enter the expression.

Use the shuttle arrow to move the data elements required for the expression from the **Available** box to the **Expression** box.

Click an operator to insert it in the **Expression** box, or choose from the function list.

Refer to [Section 4.17, "Function Reference"](#) for a description of the available functions.

4. Click **Validate Expression to validate.**

4.15.3 Adding a Global-Level Element by PL/SQL

The PL/SQL function must return a VARCHAR data type.

To add a global-level element by PL/SQL:

1. On the **Global Level Functions** object, click the **View Actions** menu and then click **Add Element by PL/SQL**.
2. In the **Add Element by PL/SQL** dialog, enter the fields, as shown in [Figure 4–36](#) and as described in [Table 4–5](#).

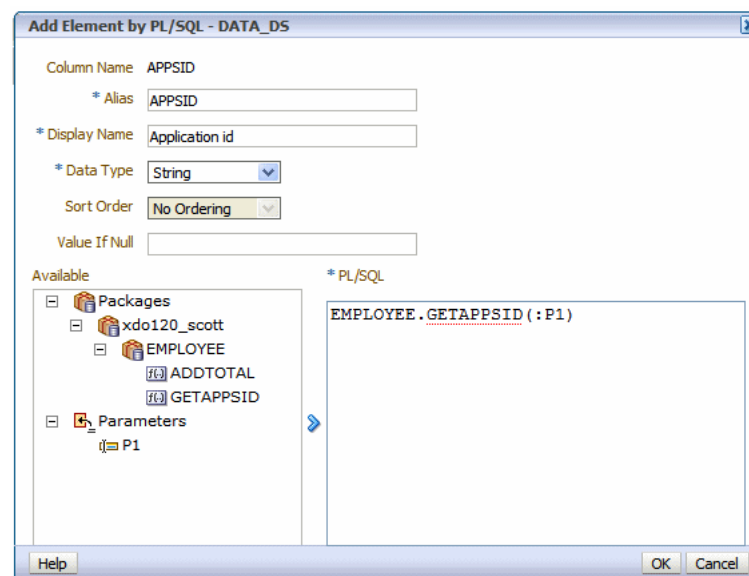
Figure 4–36 Add Element by PL/SQL Dialog

Table 4–5 Add Element by PL/SQL Dialog Fields

Field	Description
Name	Enter a name for this element.
Alias	Enter the tag name that the element has in the XML data file.
Display Name	The Display Name appears in the report design tools. Enter a name that is meaningful to your business users.
Data Type	Must select String.
Sort Order	Select a sort order.
Value if Null	Enter a value to return if the value returned from the PL/SQL function is null.

3. Select the PL/SQL package from the Available box and click the shuttle button to move the function to the Group Filter box.

4.16 Using the Structure View to Edit Your Data Structure

The Structure view enables you to preview the structure of your data model. The Data Source column displays the data elements in a hierarchical tree that you can collapse and expand. Use this view to verify the accuracy of the data model structure. The Structure view is shown in [Figure 4–37](#).

Figure 4–37 Structure View

Data Source	XML View XML Tag Name	Sorting	Value If Null	Business View Display Name	Data Type
Report Data					
Data Structure	DATA_DS				
G_Dept	G_1			G_1	
DEPARTMENT_ID	DEPARTMENT_ID			Department Number	999
MANAGER_ID	MANAGER_ID			Manager	999
DEPARTMENT_NAME	DEPARTMENT_NAME			Department	abc
G_EMP	G_2			G_2	
EMPLOYEE_ID	EMPLOYEE_ID			Employee ID	999
FIRST_NAME	FIRST_NAME			First Name	abc
LAST_NAME	LAST_NAME			Last Name	abc
EMAIL	EMAIL			e-mail	abc
PHONE_NUMBER	PHONE_NUMBER			Phone	abc
HIRE_DATE	HIRE_DATE			Date Hired	
JOB_ID	JOB_ID			Job ID	abc

4.16.1 Renaming Elements

Use the Structure page to define user-friendly names for elements in the data model. You can rename both the XML element tag name (XML View) and the name that displays in the report layout tools (Business Name). [Figure 4–38](#) shows renaming the DEPARTMENT_ID element to display as Department Number.

Figure 4–38 Editing the Display Name of an Element

Diagram Structure Code

Table View | Output

Data Source	XML View	Business View
	XML Tag Name	Display Name
Report Data		
Data Structure	DATA_DS	
G_Dept	G_1	G_1
DEPARTMENT_ID	DEPARTMENT_ID	Department Number
MANAGER_ID	MANAGER_ID	Manager
DEPARTMENT_NAME	DEPARTMENT_NAME	Department
G_EMP	G_2	G_2

4.16.2 Adding Value for Null Elements

The Structure also enables you to enter a value to use for an element if the data model returns a null value for the element.

Enter the value to use in the **Value if Null** field for the element.

4.17 Function Reference

Table 4–6 describes the usage of supported functions available from the **Add Element** by Expression dialog and the **Edit Group Filter** dialog.

Table 4–6 Supported Functions from the Add Element by Expression Dialog

Function	Description
IF	operator
NOT	operator
AND	operator
OR	operator
MAX	Returns the maximum value of the element in the set.
MIN	Returns the minimum value of the element in the set.
ROUND	<p>ROUND (<i>number</i> [, <i>integer</i>]) returns <i>number</i> rounded to <i>integer</i> places right of the decimal point. If <i>integer</i> is omitted, then <i>number</i> is rounded to 0 places. <i>integer</i> can be negative to round off digits left of the decimal point. <i>integer</i> must be an integer.</p> <p>Example:</p> <p>round (2.777)</p> <p>returns</p> <p>3</p> <p>Example:</p> <p>round (2.777, 2)</p> <p>returns</p> <p>2.78</p>
FLOOR	FLOOR(<i>n</i>) returns largest integer equal to or less than <i>n</i> .
CEILING	CEILING(<i>n</i>) returns smallest integer greater than or equal to <i>n</i> .
ABS	ABS(<i>n</i>) returns the absolute value of <i>n</i> .
AVG	AVG(expr) returns average value of expr.

Table 4–6 (Cont.) Supported Functions from the Add Element by Expression Dialog

Function	Description
LENGTH	<p>The LENGTH(char) function returns the length of char. The LENGTH function calculates the length using characters as defined by the input character set. If char is null, the function returns null. If char is an array, it returns the length of the array.</p> <p>Example to return length of an array: length({1, 2, 4, 4}) returns 4.</p> <p>Example to return length of a string: length('countries') returns 9.</p>
SUM	SUM(expr) returns the sum of value of expr.
NVL	NVL(expr1, expr2) lets you replace null (returned as a blank) with a string in the results of a query. If expr1 is null, then NVL returns expr2. If expr1 is not null, then NVL returns expr1.
CONCAT	CONCAT(char1, char2) returns char1 concatenated with char2.
STRING	STRING(number) returns the number as a string data type.
SUBSTRING	<p>The substring function allows you to extract a substring from a string. The syntax for the substring function is:</p> <p>substring(string, start_position, end_position)</p> <p><i>string</i> is the source string.</p> <p><i>start_position</i> is the position to start the extraction.</p> <p><i>end_position</i> is the end position of the string to extract (optional).</p> <p>Examples:</p> <p>substring('this is a test', 5, 7)</p> <p>returns "is"</p> <p>substring('this is a test', 5)</p> <p>returns "is a test"</p>
INSTR	<p>The instr function returns the location of a substring in a string. The syntax for the instr function is:</p> <p>instr(string1, string2)</p> <p><i>string1</i> is the string to search</p> <p><i>string2</i> is the substring to search for in string1.</p> <p>Example: instr('this is a test', 'is a')</p> <p>returns 5.</p>
DATE	<p>DATE(date_str, format_str) Converts char to date data type. The format string must be a valid Java date format string.</p> <p>Example: DATE('01-01-2011', 'MM-dd-yyyy')</p>
FORMAT_DATE	<p>The FORMAT_DATE function takes a date argument in Java date format and converts it to a formatted string.</p> <p>For example:</p> <p>FORMAT_DATE(HIRE_DATE, 'MM-DD-YYYY') where the value of HIRE_DATE is 1987-09-17T00:00:00.000+00:00 would return 17-Sep-1987.</p>
FORMAT_NUMBER	<p>The FORMAT_NUMBER function takes a number argument and converts it to a string in the format specified. For example, FORMAT_NUMBER(SALES_UNITS, '9G990D000')</p>
NUMBER	NUMBER(char) converts char to a number data type.

Adding Parameters and Lists of Values

This chapter describes how to add parameters and lists of values to the data model. It covers the following topics:

- [Section 5.1, "About Parameters"](#)
- [Section 5.2, "Adding a New Parameter"](#)
- [Section 5.3, "About Lists of Values"](#)
- [Section 5.4, "Adding Lists of Values"](#)

5.1 About Parameters

Adding parameters to a data model enables users to interact with data when they view reports.

Once you have defined the parameters in the data model, you can further configure how the parameters are displayed in the report as a report-level setting. For more information about the report-level settings, see the section "Configuring Parameter Settings for the Report" in *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

BI Publisher supports the following parameter types:

- **Text** — the user enters a text string to pass as the parameter.
- **Menu** — the user makes selections from a list of values. A list of values can contain fixed data that you specify or the list can be created using a SQL query that is executed against any of the defined data sources. This option supports multiple selections, a "Select All" option, and partial page refresh for cascading parameters.

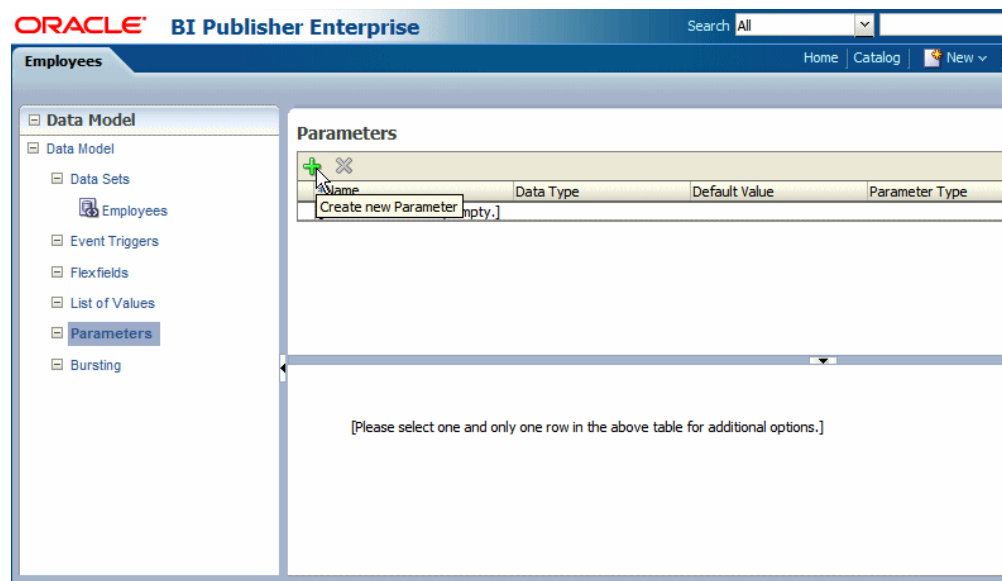
To create a menu type parameter, define the list of values first; then define the parameter and associate it to the list of values. See [Section 5.4, "Adding Lists of Values."](#)

- **Date** — the user selects a date as a parameter. Note that the data type must also be "Date" and the format must be Java date format.

5.2 Adding a New Parameter

To add a new parameter:

1. On the Data Model components pane, click **Parameters** and then click **Create new Parameter**, as shown in [Figure 5-1](#).

Figure 5–1 Create New Parameter

2. Enter a **Name** for the parameter. The name must match any references to this parameter in the data set.

Note: The parameter name you choose must not exceed the maximum length allowed for an identifier by your database. Refer to your database documentation for identifier length limitations.

3. Select the **Data Type** from the list. A **Date** data type only support a Date **Parameter Type**. The other data types support a Parameter Type of either Text or Menu:

- String
- Integer

Note: The Integer data type for parameters is a 64-bit sign integer. It has a value range of -9,223,372,036,854,775,808 to a maximum value of 9,223,372,036,854,775,807 (inclusive).

- Boolean
- Date
- Float

4. Enter a **Default Value** for the parameter. This is recommended to prevent long running queries. Default parameter values are also used to preview the report output when you design report layouts using BI Publisher Layout Editor.
5. Select the **Parameter Type**. Supported types are:
 - **Text** — Allows the user to enter a text entry to pass as the parameter. See [Section 5.2.1, "Defining a Text Parameter."](#)
 - **Menu** — Presents a list of values to the user. See [Section 5.2.2, "Defining a Menu Parameter."](#)

- **Date** — Passes a date parameter. The **Data Type** must also be Date. See [Section 5.2.3, "Defining a Date Parameter."](#)

Note: BI Publisher supports parameters that are of type text entry or menu (list of values) but not both. That is, you cannot define a "combination" parameter that enables a user to either enter a text value or choose from a menu list of values.

6. **Row Placement** - this setting configures the number of rows for displaying the parameters and in which row to place each parameter. For example, if your report has six parameters, you can assign each parameter to a separate row, 1 - 6, with one being the top row; or, you can assign two parameters each to rows 1, 2, 3. By default, all parameters are assigned to row 1.

Row placement can also be configured at the report level. The report definition supports additional display options for parameters. For more information, see "Configuring Parameter Settings for the Report" in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

5.2.1 Defining a Text Parameter

The **Text** type parameter provides a text box to prompt the user to enter a text entry to pass as the parameter to the data source. [Figure 5–2](#) shows a text parameter definition.

Figure 5–2 Text Parameter Definition

The screenshot shows the Oracle BI Publisher Enterprise web interface. The left sidebar displays the 'Data Model' tree with 'Employees' selected. The main area is titled 'Parameters' and contains a table with the following data:

Name	Data Type	Default Value	Parameter Type	Row Placement	Reorder
Dept	String	Sales	Text	1	

Below the table, the configuration for the 'Dept' parameter is shown. The 'Dept: Type: Text' section includes the following fields:

- Display Label:** Department
- Text Field Size:** 25
- Options:**
 - ☐ Text field contains comma-separated values
 - ☐ Refresh other parameters on change

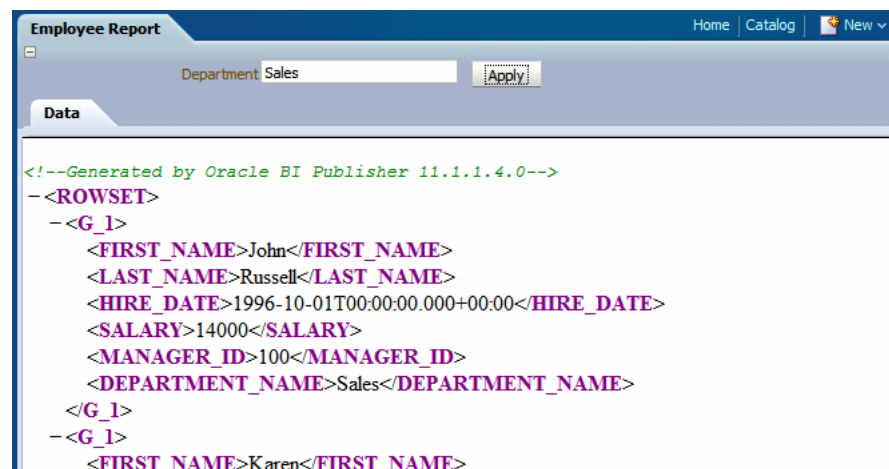
To define a Text type parameter:

1. Select Text from the Parameter Type list. The lower pane displays the appropriate fields for the selection.

2. Enter the **Display Label**. The display label is the label that displays to users when they view the report. For example: Department.
3. Enter the **Text Field Size** as an integer. This field determines the number of characters that the user can enter into the text box. For example: 25.
4. Enable the following **Options** if required:
 - **Text field contains comma-separated values** — Select this option to enable the user to enter multiple comma-delimited values for this parameter. The parameter in your data source must be defined to support multiple values.
 - **Refresh other parameters on change** — Performs a partial page refresh to refresh any other parameters whose values are dependent on the value of this one.

Figure 5–3 shows how the Department parameter displays to the report consumer.

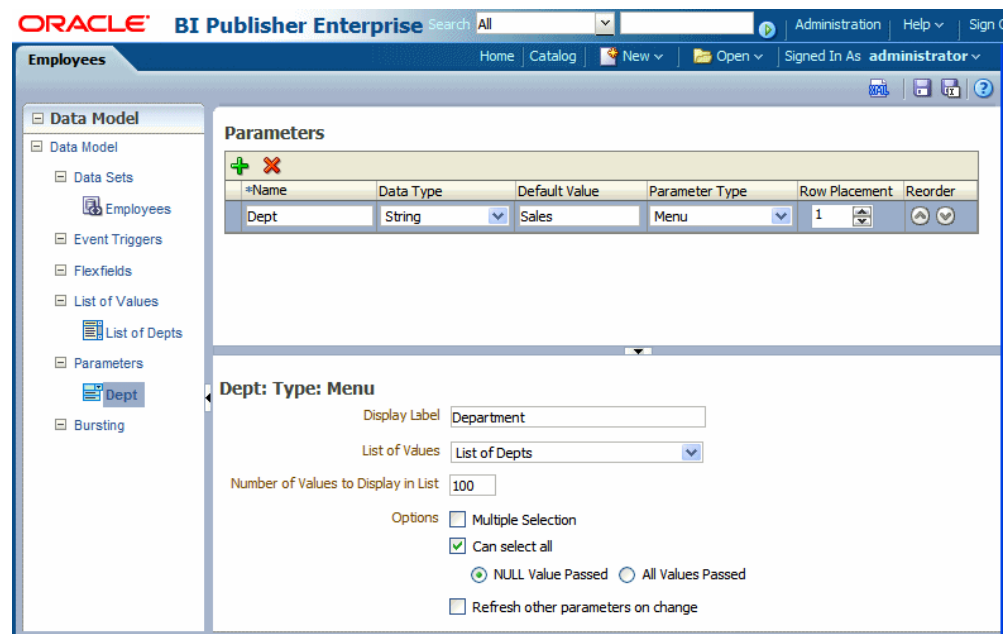
Figure 5–3 Text Type Parameter as Displayed in the Report



5.2.2 Defining a Menu Parameter

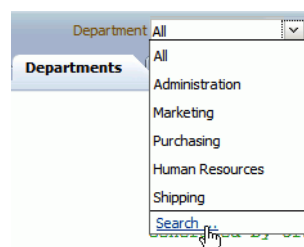
A **Menu** type parameter presents a list of values to the user. You must define the list of values first. See [Section 5.4, "Adding Lists of Values."](#) Long lists automatically enable a Search feature.

Figure 5–4 shows the menu parameter definition.

Figure 5–4 Menu Type Parameter Definition

To define a Menu type parameter:

1. Select Menu from the Parameter Type list. The lower pane displays the appropriate fields for your selection.
2. Enter the **Display Label**. The display label is the label that displays to users when they view the report. For example: Department.
3. Enter the **Number of Values to Display in List**. If the number of values in the list exceeds the entry in this field, the user must click **Search** to find a value not displayed, as shown in Figure 5–5. This field defaults to 100.

Figure 5–5 Search Feature Enabled When Number of Values Exceeds Setting

4. Select the **List of Values** that you defined for this parameter.
5. Enable the following **Options** if required:
 - **Multiple Selection** — Allows the user to select multiple entries from the list. Your data source must be able to support multiple values for the parameter. The display of a menu parameter that supports multiple selection differs. See Figure 5–6 and Figure 5–7.
 - **Can select all** — Inserts an "All" option in the list. When the user selects "All" from the list of values, you have the option of passing a null value for the parameter or all list values. Choose **NULL Value Passed** or **All Values Passed**.

Note: Using * passes a null, so you must handle the null in your data source. A method to handle the null would be the standard Oracle NVL command, for example:

where customer_id = nvl(:cstid, customer_id)

where cstid is a value passed from the LOV and when the user selects All it passes a null value.

- **Refresh other parameters on change** — Performs a partial page refresh to refresh any other parameters whose values are dependent on the value of this one.

Figure 5–6 shows how the Department menu type parameter displays to the report consumer when multiple selection is not enabled.

Figure 5–6 Department Menu Type Parameter with Multiple Selection Disabled

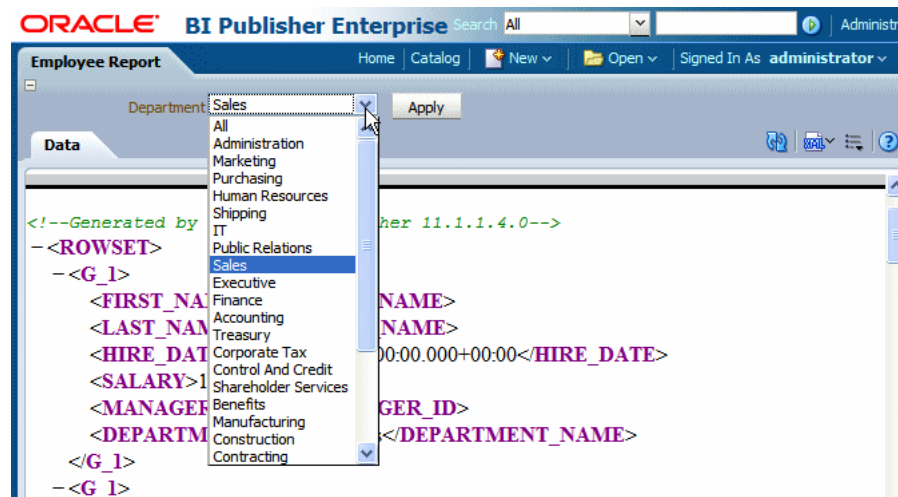
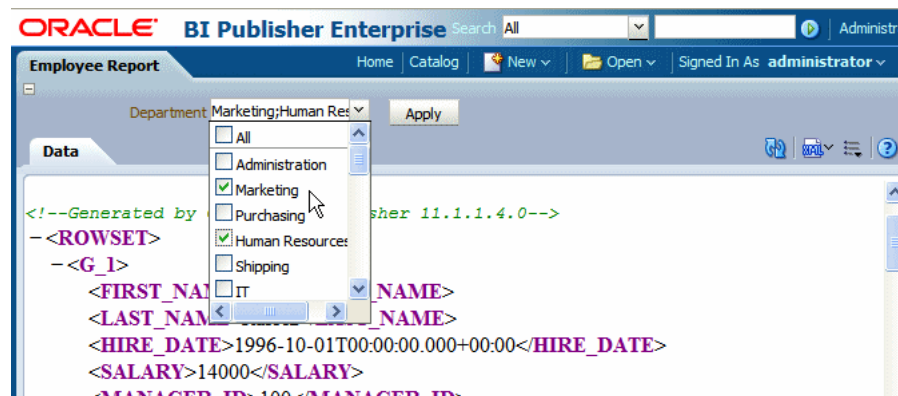


Figure 5–7 shows how the Department menu type parameter displays to the report consumer when multiple selection is enabled.

Figure 5–7 Department Menu Type Parameter with Multiple Selection Enabled



5.2.2.1 Customizing the Display of Menu Parameters

The display of menu parameters in the report can be further customized in the report definition. Menu type parameters support the additional display option as a static list of checkboxes or radio buttons. For more information, see "Configuring Parameter Settings for the Report" in the *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

5.2.3 Defining a Date Parameter

The **Date** type parameter provides a date picker to prompt the user to enter a date to pass as the parameter to the data source. Figure 5–8 shows the date parameter definition.

Figure 5–8 Date Parameter Definition

The screenshot shows the Oracle BI Publisher Enterprise web interface. On the left, a navigation pane shows the 'Data Model' structure with 'Employees' selected. The main area is titled 'Parameters' and contains a table with one parameter, 'H_DATE', of type 'Date'. Below the table, the configuration for 'H_DATE' is shown, including fields for 'Display Label', 'Text Field Size', 'Date Format String', 'Date From', and 'Date To'.

*Name	Data Type	Default Value	Parameter Type	Row Placement	Reorder
H_DATE	Date	03-04-1996	Date	1	

H_DATE: Type: Date

Display Label:

Text Field Size:

Date Format String: (must be Java date format, e.g., MM-dd-yyyy)

Date From:

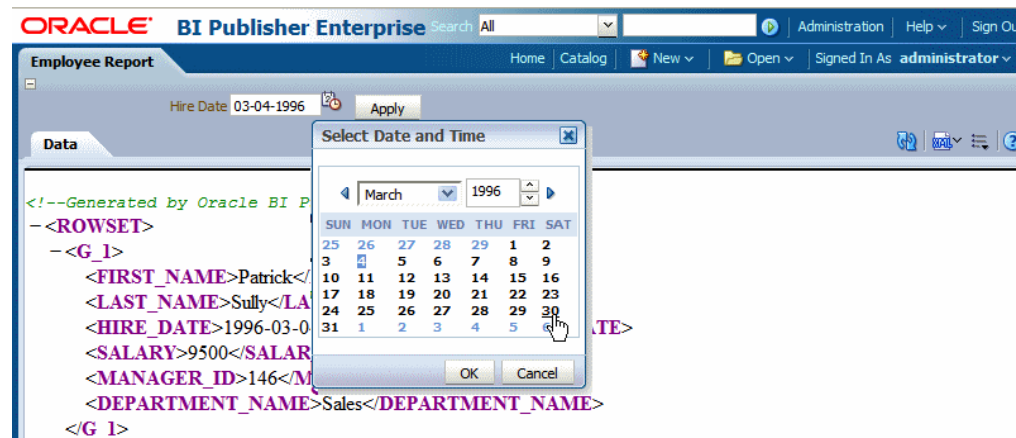
Date To:

To define a Date type parameter:

1. Select Date from the Parameter Type list. The lower pane displays the appropriate fields for your selection.
2. Enter the **Display Label**. The display label is the label that displays to users when they view the report. For example: Hire Date.
3. Enter the **Text Field Size** as an integer. This field determines the number of characters that the user can enter into the text box for the date entry. For example: 10.
4. Enter the **Date Format String**. The format must be a Java date format (for example, MM-dd-yyyy).
5. Optionally, enter a **Date From** and **Date To**. The dates entered here define the date range that are presented to the user by the date picker. For example if you enter the **Date From** as 01-01-1990, the date picker does not allow the user to select a date before 01-01-1990. Leave the **Date To** blank to enable all future dates.

Figure 5–9 shows how the Hire Date parameter displays to the report consumer.

Figure 5–9 Hire Date Parameter



5.3 About Lists of Values

A list of values is a defined set of values that a report consumer can select from to pass a parameter value to your data source. If you define a menu type or search type parameter, the list of values that you define here provides the menu of choices. You must define the list of values before you define the menu or search parameter.

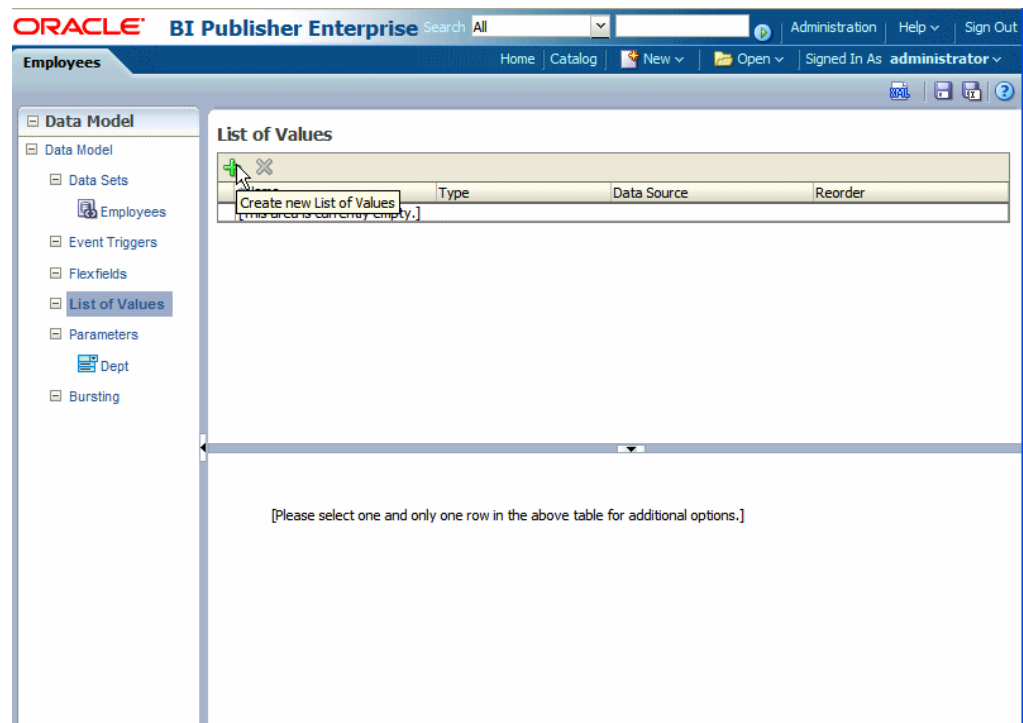
Populate the list using one of the following methods:

- **Fixed Data** — Manually enter the list of values.
- **SQL Query** — Retrieve the values from a database using a SQL query.

5.4 Adding Lists of Values

To add a List of Values:

1. On the Data Model components pane, click **List of Values** and then click **Create new List of Values**, as shown in [Figure 5–10](#).

Figure 5–10 Create New List of Values

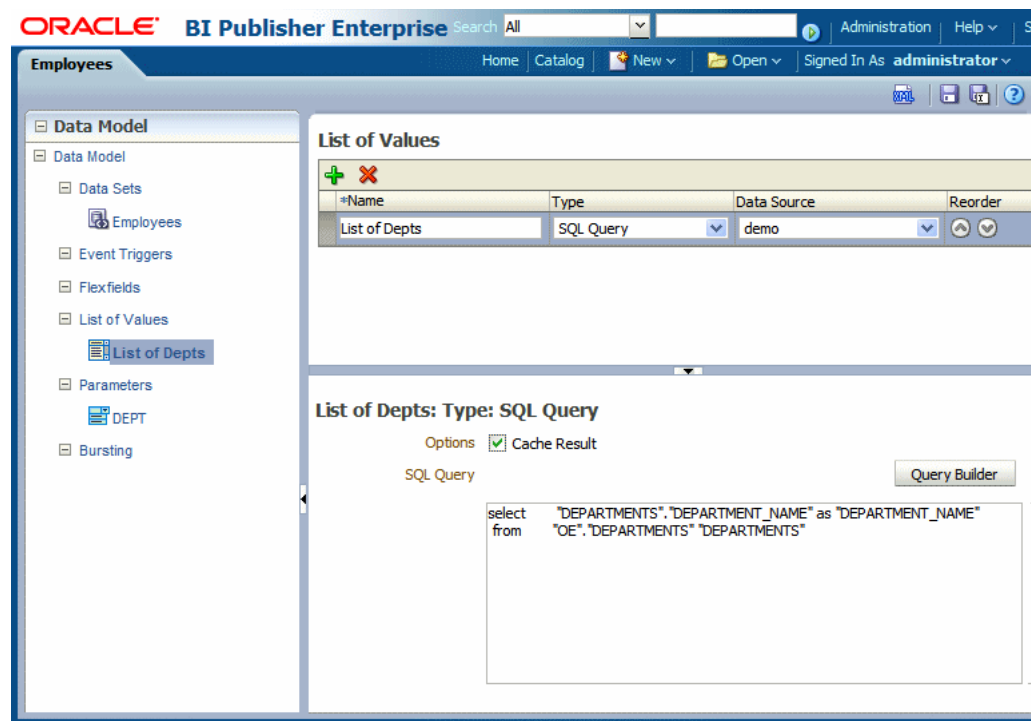
2. Enter a **Name** for the list and select a **Type**: SQL Query or Fixed Data.

5.4.1 Creating a List from a SQL Query

To create a list from a SQL query:

1. Select a **Data Source** from the list.
2. In the lower pane, select **Cache Result** (recommended) if you want the results of the query cached for the report session.
3. Enter the SQL query or use the Query Builder. See [Section 3.4, "Using the Query Builder"](#) for information on the Query Builder utility. [Figure 5–11](#) shows a SQL query type list of values.

Figure 5–11 SQL Query Type List of Values



5.4.2 Creating a List from a Fixed Data Set

To create a list from a fixed data set:

1. In the lower pane, click the **Create new List of Values** icon to add a Label and Value pair.
2. Repeat for each label-value pair required.

Figure 5–12 shows fixed data type list of values.

Figure 5–12 Fixed Data Type List of Values

The screenshot displays the Oracle BI Publisher Enterprise interface. The left-hand navigation pane shows the 'Data Model' tree with 'List of Depts' selected under 'List of Values'. The main content area is titled 'List of Values' and contains a table with the following structure:

*Name	Type	Data Source	Reorder
List of Depts	Fixed Data		

Below this, a section titled 'List of Depts: Type: Fixed Data' shows a table with two columns: '*Label' and '*Value'. The data rows are as follows:

*Label	*Value
Accounting	Accounting
Marketing	Marketing
Manufacturing	Manufacturing
Administration	Administration
Sales	Sales

Adding Event Triggers

This chapter describes how to add event triggers to the data model. It covers the following topics:

- [Section 6.1, "About Triggers"](#)
- [Section 6.2, "Adding Before Data and After Data Triggers"](#)
- [Section 6.3, "Creating Schedule Triggers"](#)

6.1 About Triggers

The BI Publisher data model supports the following types of triggers:

- Before Data — fires before the data set is executed.
- After Data — fires after the data engine executes all data sets and generates the XML.
- Schedule Trigger - Fires when a report job is scheduled to run.

An event trigger checks for an event and when the event occurs, it runs the code associated with the trigger. The BI Publisher data model supports before data and after data triggers that execute a PL/SQL function stored in a PL/SQL package in your Oracle Database. The return data type for a PL/SQL function inside the package must be a Boolean type and the function must explicitly return TRUE or FALSE.

A schedule trigger is associated with a schedule job. It is a SQL query that is executed at the time a report job is scheduled to run. If the SQL returns any data, the report job runs. If the SQL query returns no data, the job instance is skipped.

6.2 Adding Before Data and After Data Triggers

To add a before data or after data event trigger:

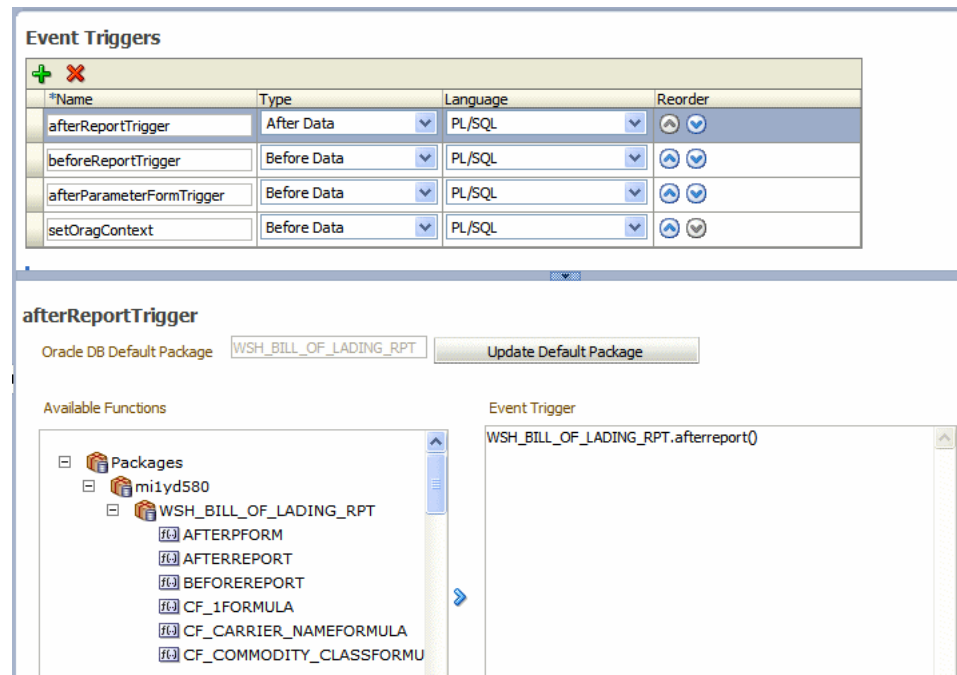
1. From the data model **Properties** pane, enter the **Oracle DB Default Package** that contains the PL/SQL function signature to execute when the trigger fires. See [Section 2.8, "Setting Data Model Properties."](#)
2. From the task pane, click **Event Triggers**.
3. From the **Event Triggers** pane, click the **Create New** icon.
4. Enter the following for the trigger:
 - **Name**
 - **Type** — Select Before Data or After Data.

- **Language** — Select PL/SQL.

The lower pane displays the available functions in the Oracle DB Default Package that you entered in the data model Properties in Step 1.

Figure 6–1 shows an event trigger.

Figure 6–1 Event Trigger



5. Select the package from the **Available Functions** box and click the arrow to move a function to the **Event Trigger** box. The name appears as PL/SQL <package name>.<function name>.

Important: If you define a default package then you must define all parameters as a global PL/SQL variable in the PL/SQL package. You can then explicitly pass parameters to your PL/SQL function trigger or all parameters are available as a global PL/SQL variable.

6.2.1 Order of Execution

If you define multiple triggers of the same type, they fire in the order that they appear in the table (from top to bottom).

To change the order of execution:

1. Use the **Reorder** arrows to place the triggers in the correct order.

6.3 Creating Schedule Triggers

A schedule trigger fires when a report job is scheduled to run. Schedule triggers are of type SQL Query. When a report job is scheduled to run, the schedule trigger executes the SQL statement defined for the trigger. If data is returned, then the report job is submitted. If data is not returned from the trigger SQL query, the report job is skipped.

The schedule trigger that you associate with a report job can reside in any data model in the catalog. You do not need to create the schedule trigger in the data model of the report for which you wish to execute it. You can therefore reuse schedule triggers across multiple report jobs.

To add a Schedule Trigger:

1. In the data model editor task pane, click **Event Triggers**.
2. From the **Event Triggers** pane, click the **Create New** icon.
3. Enter the following for the trigger:
 - **Name** - enter a name for the trigger.
 - **Type** — select Schedule.
 - **Language** — defaults to SQL Query.
4. In the lower pane, enter the following:
 - **Options** - select this box to cache the results of the trigger query.
 - **Data Source** - select the data source for the trigger query.
 - **SQL Query** - enter the query in the text area, or click **Query Builder** to use the utility to construct the SQL query. For information, see [Section 3.4, "Using the Query Builder."](#)

If the SQL query returns any results, the scheduled report job executes. [Figure 6–2](#) shows a schedule trigger to test for inventory levels.

Figure 6–2 Schedule Trigger

Event Triggers

Name	Type	Language	Reorder
Quantity	Schedule	SQL Query	

Salary: Language: SQL Query

Options ☐ Cache Result

Data Source demo

SQL Query

```
select 'true'
from 'OE'.OC_INVENTORIES' 'OC_INVENTORIES'
where 'OC_INVENTORIES'.QUANTITY_ON_HAND' < :pQuantity
```

[Query Builder](#)

For information on implementing the schedule trigger in the report job, see "Defining the Schedule for a Job" in the *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*.

Adding Flexfields

This chapter describes how to add flexfields to your data model. It covers the following topics:

- [Section 7.1, "About Flexfields"](#)
- [Section 7.2, "Adding Flexfields"](#)

7.1 About Flexfields

Flexfields are unique to Oracle Applications. If you are reporting on data from the Oracle Applications, use this component of the data model to retrieve flexfield data.

To use a flexfield in your data model:

- Define the SELECT statement to use for the report data.
- Within the SELECT statement, define each flexfield as a lexical. Use the &LEXICAL_TAG to embed flexfield related lexicals into the SELECT statement.
- Add the flexfield to the data model.

You can use flexfield references to replace the clauses appearing after SELECT, FROM, WHERE, ORDER BY, or HAVING. Use a flexfield reference when you want the parameter to replace multiple values at runtime. The data model editor supports the following flexfield types:

- **Where** — This type of lexical is used in the WHERE section of the statement. It is used to modify the WHERE clause such that the SELECT statement can filter based on key flexfield segment data.
- **Order by** — This type of lexical is used in the ORDER BY section of the statement. It returns a list of column expressions so that the resulting output can be sorted by the flex segment values.
- **Select** — This type of lexical is used in the SELECT section of the statement. It is used to retrieve and process key flexfield (kff) code combination related data based on the lexical definition.
- **Filter** — This type of lexical is used in the WHERE section of the statement. It is used to modify the WHERE clause such that the SELECT statement can filter based on Filter ID passed from Oracle Enterprise Scheduling Service.
- **Segment Metadata** — Use this type of lexical to retrieve flexfield-related metadata. Using this lexical, you are not required to write PL/SQL code to retrieve this metadata. Instead, define a dummy SELECT statement, then use this lexical to get the metadata. This lexical should return a constant string.

After you set up the flexfield components of your data model, create a flexfield reference in the SQL query using the following syntax:

```
&LEXICAL_TAG ALIAS_NAME
```

for example:

```
&FLEX_GL_BALANCING alias_gl_balancing
```

7.2 Adding Flexfields

To add a flexfield:

1. Enter the following:
 - **Name** — Enter a name for the flexfield component.
 - **Type** — Select the flexfield type from the list. The type you select here determines the additional fields required. See [Section 7.2.1, "Entering Flexfield Details."](#)
 - **Application Short Name** — Enter the short name of the Oracle Application that owns this flexfield (for example, GL).
 - **ID Flex Code** — Enter the flexfield code defined for this flexfield in the Register Key Flexfield form (for example, GL#).
 - **ID Flex Number** — Enter the name of the source column or parameter that contains the flexfield structure information.

7.2.1 Entering Flexfield Details

Select Segment Metadata, Select, Where, Order By, Filter. Depending on the type you select, the detail pane displays the appropriate fields, described in [Table 7–1](#).

Table 7–1 Detail Fields for Segment Metadata

Field	Description
Segments	(Optional) Identifies for which segments this data is requested. Default value is "ALL". See <i>Oracle E-Business Suite Developer's Guide</i> for syntax.
Show Parent Segments	Select this box to automatically display the parent segments of dependent segments even if it is specified as not displayed in the segments attribute.
Metadata Type	Select the type of metadata to return: Above Prompt — Above prompt of segment(s). Left Prompt — Left prompt of segment(s)

[Table 7–2](#) shows the detail fields for the Select flexfield type.

Table 7–2 Detail Fields for Select

Field	Description
Multiple ID Flex Num	Indicates whether this lexical supports multiple structures. Checking this box indicates all structures are potentially used for data reporting. The data engine uses <code><code_combination_table_alias>.<set_defining_column_name></code> to retrieve the structure number.

Table 7–2 (Cont.) Detail Fields for Select

Field	Description
Code Combination Table Alias	Specify the table alias to prefix to the column names. Use TABLEALIAS if your SELECT joins to other flexfield tables or uses a self-join.
Segments	(Optional) Identifies for which segments this data is requested. Default value is "ALL". See <i>Oracle E-Business Suite Developer's Guide</i> for syntax.
Show Parent Segments	Select this box to automatically display the parent segments of dependent segments even if it is specified as not displayed in the segments attribute.
Output Type	Select from the following: <ul style="list-style-type: none"> Value — Segment value as it is displayed to user. Padded Value — Padded segment value as it is displayed to user. Number type values are padded from the left. String type values are padded on the right. Description — Segment value's description up to the description size defined in the segment definition. Full Description — Segment value's description (full size). Security — Returns Y if the current combination is secured against the current user, N otherwise.

Table 7–3 shows the detail fields for the Where flexfield type.

Table 7–3 Detail Fields for Where

Field	Description
Code Combination Table Alias	Specify the table alias to prefix to the column names. You use TABLEALIAS if your SELECT joins to other flexfield tables or uses a self-join.
Segments	(Optional) Identifies for which segments this data is requested. Default value is "ALL". See <i>Oracle E-Business Suite Developer's Guide</i> for syntax.
Operator	Select the appropriate operator.
Operand1	Enter the value to use on the right side of the conditional operator.
Operand2	(Optional) High value for the BETWEEN operator.

Table 7–4 shows the detail fields for the Order by flexfield type.

Table 7–4 Detail Fields for Order By

Field	Description
Multiple ID Flex Num	Indicates whether this lexical supports multiple structures. Selecting this box indicates all structures are potentially used for data reporting. The data engine uses <code_combination_table_alias>.<set_defining_column_name> to retrieve the structure number.
Code Combination Table Alias	Specify the table alias to prefix to the column names. You use TABLEALIAS if your SELECT joins to other flexfield tables or uses a self-join.
Segments	(Optional) Identifies for which segments this data is requested. Default value is "ALL". See <i>Oracle E-Business Suite Developer's Guide</i> for syntax.

Table 7–4 (Cont.) Detail Fields for Order By

Field	Description
Show Parent Segments	Select this box to automatically display the parent segments of dependent segments even if it is specified as not displayed in the segments attribute.

[Table 7–5](#) shows the detail fields for the Filter flexfield type.

Table 7–5 Detail Fields for Filter

Field	Description
Code Combination Table Alias	Specify the table alias to prefix to the column names. You use TABLEALIAS if your SELECT joins to other flexfield tables or uses a self-join.
Flex Filter ID	(Required) Enter the unique Key internal code of the key flexfield.
Flex Filter Comment	(Optional) Enter a comments or description.

Adding Bursting Definitions

This chapter describes how to add bursting definitions to the data model. It covers the following topics:

- [Section 8.1, "About Bursting"](#)
- [Section 8.2, "What is the Bursting Definition?"](#)
- [Section 8.3, "Adding a Bursting Definition to Your Data Model"](#)
- [Section 8.4, "Defining the Query for the Delivery XML"](#)
- [Section 8.5, "Passing a Parameter to the Bursting Query"](#)
- [Section 8.6, "Defining the Split By and Deliver By Elements for a CLOB/XML Data Set"](#)
- [Section 8.7, "Configuring a Report to Use a Bursting Definition"](#)
- [Section 8.8, "Sample Bursting Query"](#)
- [Section 8.9, "Creating a Table to Use as a Delivery Data Source"](#)

8.1 About Bursting

Bursting is a process of splitting data into blocks, generating documents for each block, and delivering the documents to one or more destinations. The data for the report is generated by executing a query once and then splitting the data based on a "Key" value. For each block of the data, a separate document is generated and delivered.

Using BI Publisher's bursting feature you can split a single report based on an element in the data model and deliver the report based on a second element in the data model. Driven by the delivery element, you can apply a different template, output format, delivery method, and locale to each split segment of the report. Example implementations include:

- Invoice generation and delivery based on customer-specific layouts and delivery preference
- Financial reporting to generate a master report of all cost centers, splitting out individual cost center reports to the appropriate manager
- Generation of pay slips to all employees based on one extract and delivered through e-mail

8.2 What is the Bursting Definition?

A bursting definition is a component of the data model. After you have defined the data sets for the data model, you can set up one or more bursting definitions. When you set up a bursting definition, you define the following:

- The **Split By** element is an element from the data that governs how the data is split. For example, to split a batch of invoices by each invoice, you may use an element called CUSTOMER_ID. The data set must be sorted or grouped by this element.
- The **Deliver By** element is the element from the data that governs how formatting and delivery options are applied. In the invoice example, it is likely that each invoice has delivery criteria determined by customer; therefore, the Deliver By element would also be CUSTOMER_ID.
- The **Delivery Query** is a SQL query that you define for BI Publisher to construct the delivery XML data file. The query must return the formatting and delivery details.

8.3 Adding a Bursting Definition to Your Data Model

Prerequisites:

- You have defined the data set for this data model
- The data set is sorted or grouped by the element by which you want to split the data in your bursting definition
- The delivery and formatting information is available to BI Publisher. The information can be provided at runtime to BI Publisher in one of the following ways:
 - The information is stored in a database table available to BI Publisher (for a dynamic delivery definition)
 - The information is hard coded in the delivery SQL (for a static delivery definition)
- The report definition for this data model has been created and includes the layouts to be applied to the report data.

To add a bursting definition:

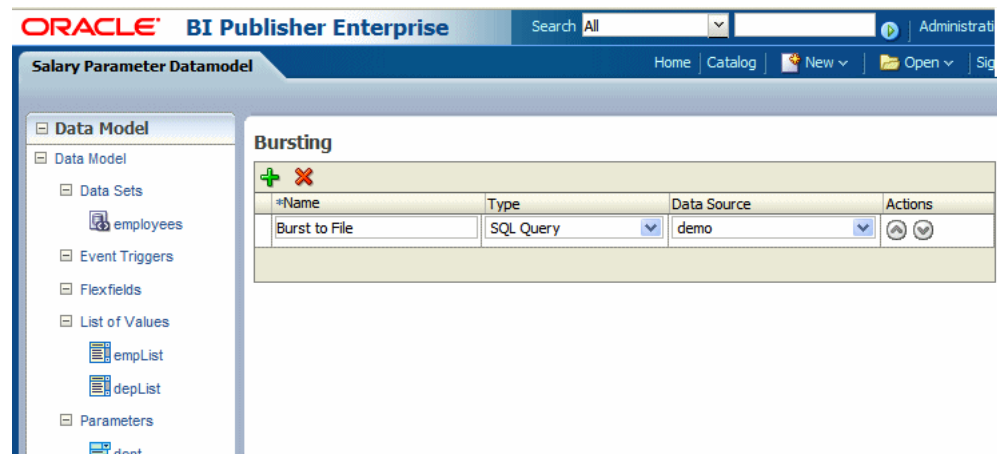
1. On the component pane of the data model editor, click **Bursting**.
2. On the Bursting definition table, click the **Create new Bursting** button.
3. Enter the following for this bursting definition:

Name — For example, "Burst to File"

Type — SQL Query is currently the only supported type

Data Source — Select the data source that contains the delivery information

Figure 8–1 shows a Bursting definition.

Figure 8–1 Bursting Definition

4. In the lower region, enter the following for this bursting definition:

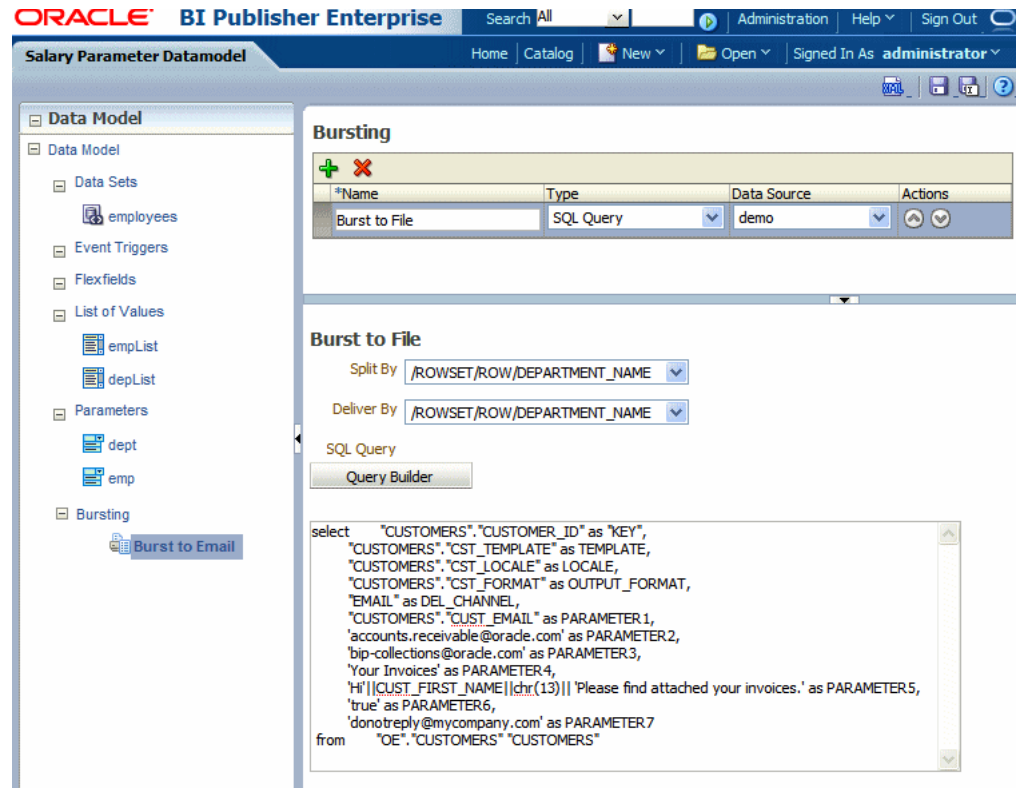
Split By — Select the element from the data set by which to split the data

Deliver By — Select the element from the data set by which to format and deliver the data

Note: If the Split By and Deliver By elements reside in an XML document stored as a CLOB in your database, you must enter the full XPATH in the Split By and Delivery By fields. For more information, see [Section 8.6, "Defining the Split By and Deliver By Elements for a CLOB/XML Data Set."](#)

SQL Query — Enter the query to construct the delivery XML. For information on how to construct the bursting query, see [Section 8.4, "Defining the Query for the Delivery XML."](#) Figure 8–2 shows a sample bursting query.

Figure 8–2 Sample Bursting Query



8.4 Defining the Query for the Delivery XML

The bursting query is a SQL query that you define to provide BI Publisher with the required information to format and deliver the report. BI Publisher uses the results from the bursting query to create the delivery XML.

The BI Publisher bursting engine uses the delivery XML as a mapping table for each Deliver By element. The structure of the delivery XML required by BI Publisher is as follows:

```
<ROWSET>
<ROW>
  <KEY></KEY>
  <TEMPLATE></TEMPLATE>
  <LOCALE></LOCALE>
  <OUTPUT_FORMAT></OUTPUT_FORMAT>
  <DEL_CHANNEL></DEL_CHANNEL>
  <TIMEZONE></TIMEZONE>
  <CALENDAR></CALENDAR>
  <OUTPUT_NAME></OUTPUT_NAME>
  <SAVE_OUTPUT></SAVE_OUTPUT>
  <PARAMETER1></PARAMETER1>
  <PARAMETER2></PARAMETER2>
  <PARAMETER3></PARAMETER3>
  <PARAMETER4></PARAMETER4>
  <PARAMETER5></PARAMETER5>
  <PARAMETER6></PARAMETER6>
  <PARAMETER7></PARAMETER7>
  <PARAMETER8></PARAMETER8>
  <PARAMETER9></PARAMETER9>
```

```
<PARAMETER10></PARAMETER10>
</ROW>
</ROWSET>
```

- **KEY** — The Delivery key and must match the **Deliver By** element. The bursting engine uses the key to link delivery criteria to a specific section of the burst data. Ensure that you use double quotes around "KEY" in the select statement, for example:

```
select d.department_name as "KEY",
```

- **TEMPLATE** — The name of the Layout to apply. Note that the value is the Layout name (for example, 'Customer Invoice'), not the template file name (for example, invoice.rtf).
- **LOCALE** — The template locale, for example, 'en-US'.
- **OUTPUT_FORMAT** — The output format. For a description of each type, see the section "Setting the Output Types" in *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*. [Table 8–1](#) shows the valid values to enter for the bursting query.

Table 8–1 Values to Enter for OUTPUT_FORMAT

Output Format	Value to Enter in Bursting Query	Template Types That Can Generate This Output Format
Interactive	N/A	Not supported for bursting
HTML	html	BI Publisher, RTF, XSL Stylesheet (FO)
PDF	pdf	BI Publisher, RTF, PDF, Flash, XSL Stylesheet (FO)
RTF	rtf	BI Publisher, RTF, XSL Stylesheet (FO)
Excel	excel	BI Publisher, RTF, Excel, XSL Stylesheet (FO)
Excel2000	excel2000	BI Publisher, RTF, Excel, XSL Stylesheet (FO)
Excel2007	xslx	BI Publisher, RTF, XSL Stylesheet (FO)
PowerPoint	ppt	BI Publisher, RTF, XSL Stylesheet (FO)
PowerPoint 2007	pptx	BI Publisher, RTF, XSL Stylesheet (FO)
MHTML	mhtml	BI Publisher, RTF, Flash, XSL Stylesheet (FO)
PDF/A	pdfa	BI Publisher, RTF, XSL Stylesheet (FO)
PDF/X	pdfx	BI Publisher, RTF, XSL Stylesheet (FO)
PDFZ	pdfz	BI Publisher, RTF, PDF, XSL Stylesheet (FO)
FO	xslfo	BI Publisher, RTF, XSL Stylesheet (FO)
Data	xml	BI Publisher, RTF, PDF, Excel, Flash, XSL Stylesheet (FO), Etext, XSL Stylesheet (HTML XML/Text)
CSV	csv	BI Publisher, RTF, PDF, Excel, Flash, XSL Stylesheet (FO), XSL Stylesheet (HTML XML/Text), Etext
XML	txml	XSL Stylesheet (HTML XML/Text)
Text	text	XSL Stylesheet (HTML XML/Text), Etext
Flash	flash	Flash

- **SAVE_OUTPUT** — Indicates whether to save the output documents to BI Publisher history tables that the output can be viewed and downloaded from the **Report Job History** page.

Valid values are 'true' (default) and 'false'. If this property is not set, the output is saved.

- **DEL_CHANNEL** — The delivery method. Valid values are:
 - EMAIL
 - FAX
 - FILE
 - FTP
 - PRINT
 - WEBDAV
- **TIMEZONE** — The time zone to use for the report. Values must be in the Java format, for example: 'America/Los_Angeles'. If time zone is not provided, then the system default time zone is used to generate the report.
- **CALENDAR** — The calendar to use for the report. Valid values are:
 - GREGORIAN
 - ARABIC_HIJRAH
 - ENGLISH_HIJRAH
 - JAPANESE_IMPERIAL
 - THAI_BUDDHA
 - ROC_OFFICIAL (Taiwan)

If not provided, the value 'GREGORIAN' is used.
- **OUTPUT_NAME** — The name to assign to the output file in the report job history.
- **Delivery parameters by channel** — The values required for the parameters depend on the delivery method chosen. The parameter values mappings for each method are shown in [Table 8–2](#). Not all delivery channels use all the parameters.

Table 8–2 Parameter Values Mapping by Method

Delivery Channel	PARAMETER Values
Email	PARAMETER1: Email address PARAMETER2: cc PARAMETER3: From PARAMETER4: Subject PARAMETER5: Message body PARAMETER6: Attachment value ('true' or 'false'). If your output format is PDF, you must set this parameter to "true" to attach the PDF to the e-mail. PARAMETER7: Reply-To PARAMETER8: Bcc (PARAMETER 9-10 are not used)

Table 8–2 (Cont.) Parameter Values Mapping by Method

Delivery Channel	PARAMETER Values
Printer	<p>PARAMETER1: Printer group</p> <p>PARAMETER2: Printer name or for a printer on CUPS, the printer URI, for example: ipp://myserver.com:631/printers/printer1</p> <p>PARAMETER3: Number of Copies</p> <p>PARAMETER4: Sides. Valid values are:</p> <ul style="list-style-type: none"> ■ "d_single_sided" for single-sided ■ "d_double_sided_l" for duplex/long edge ■ "d_double_sided_s" for tumble/short edge <p>If the parameter is not specified, single-sided is used.</p> <p>PARAMETER5: Tray. Valid values are:</p> <ul style="list-style-type: none"> ■ "t1" for "Tray 1" ■ "t2" for "Tray 2" ■ "t3" for "Tray 3" <p>If not specified, the printer default is used.</p> <p>PARAMETER6: Print range. For example "3" prints page 3 only, "2-5" prints pages 2-5, "1,3-5" prints pages 1 and 3-5 (PARAMETER 7-10 are not used)</p>
Fax	<p>PARAMETER1: Fax server name</p> <p>PARAMETER2: Fax number</p> <p>(PARAMETER 3-10 are not used)</p>
WebDAV	<p>PARAMETER1: Server Name PARAMETER2: Username</p> <p>PARAMETER3: Password PARAMETER4: Remote Directory</p> <p>PARAMETER5: Remote File Name</p> <p>PARAMETER6: Authorization type, values are 'basic' or 'digest'</p> <p>(PARAMETER 7-10 are not used)</p>
File	<p>PARAMETER1: Directory</p> <p>PARAMETER2: File Name</p> <p>(PARAMETER 3-10 are not used)</p>
FTP and SFTP	<p>PARAMETER1: Server name PARAMETER2: Username</p> <p>PARAMETER3: Password</p> <p>PARAMETER4: Remote Directory</p> <p>PARAMETER5: Remote File Name</p> <p>PARAMETER6: Secure (set this value to 'true' to enable Secure FTP)</p> <p>(PARAMETER 7-10 are not used)</p>

8.5 Passing a Parameter to the Bursting Query

You can pass the value for an element of your bursting XML using a parameter defined in the data model. For example, if you want to be able to select the template at the time of submission, you can define a parameter in the data model and use the `:parameter_name` syntax in your query. The following example demonstrates this use case of a parameter in a bursting query.

Assume your report definition includes three layouts: layout1, layout2, and layout3. At submission time you want to select the layout (or TEMPLATE, as defined in the bursting query) to use. In your data model, define a list of values with the layout names. The following figure shows a data model with the layout list of values:

Figure 8–3 Defining the List of Values

The screenshot shows the 'my data model' interface. On the left, a tree view under 'Data Model' includes 'Data Sets', 'Event Triggers', 'Flexfields', 'List of Values', 'Parameters', and 'Bursting'. The 'List of Values' item is selected, and a sub-item 'template' is highlighted. The main area displays the 'List of Values' configuration for 'template'. It shows a table with columns: *Name, Type, Data Source, and Reorder. The *Name column contains 'template', Type is 'Fixed Data', and Data Source is empty. Below this, a section titled 'template: Type: Fixed Data' shows a table with columns: *Label and *Value. The *Label column contains 'layout1', 'layout2', and 'layout3'. The *Value column contains 'layout1', 'layout2', and 'layout3'.

*Name	Type	Data Source	Reorder
template	Fixed Data		

*Label	*Value
layout1	layout1
layout2	layout2
layout3	layout3

Next create a menu type parameter, here named P1:

Figure 8–4 Defining a Parameter

The screenshot shows the 'my data model' interface. On the left, a tree view under 'Data Model' includes 'Data Sets', 'Event Triggers', 'Flexfields', 'List of Values', 'Parameters', and 'Bursting'. The 'Parameters' item is selected, and a sub-item 'P1' is highlighted. The main area displays the 'Parameters' configuration for 'P1'. It shows a table with columns: *Name, Data Type, Default Value, Parameter Type, and Reorder. The *Name column contains 'P1', Data Type is 'String', Default Value is 'layout1', and Parameter Type is 'Menu'. Below this, a section titled 'P1: Type: Menu' shows a 'Display Label' field with the value 'Template' and a 'List of Values' dropdown menu with the value 'template'. There are also three checkboxes: 'Multiple Selection' (unchecked), 'Can select all' (unchecked), and 'Refresh other parameters on change' (unchecked).

*Name	Data Type	Default Value	Parameter Type	Reorder
P1	String	layout1	Menu	

P1: Type: Menu

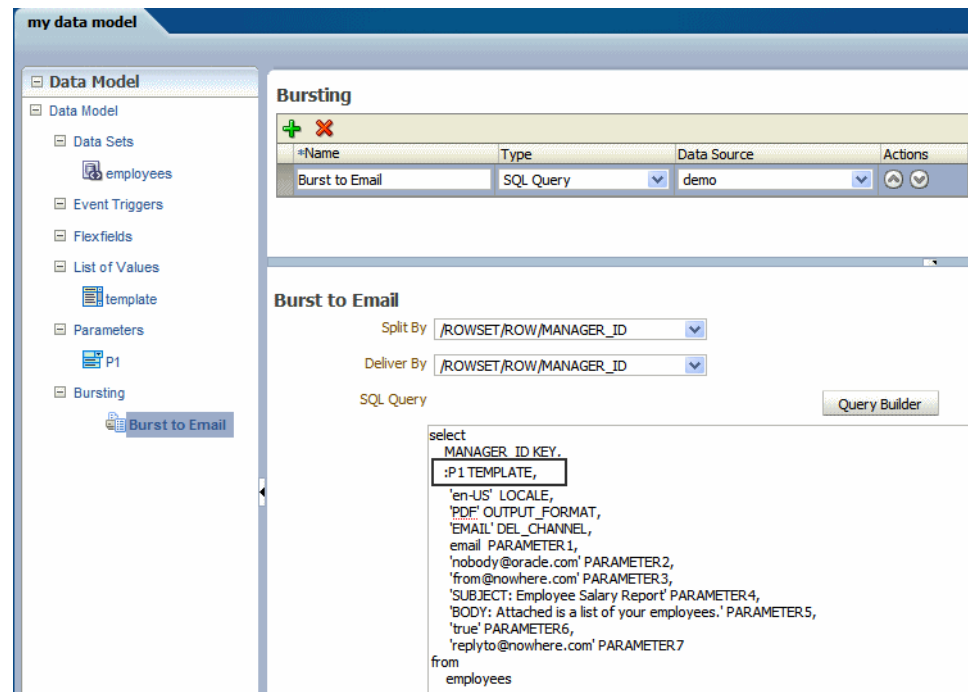
Display Label:

List of Values:

Options:

- ☐ Multiple Selection
- ☐ Can select all
- ☐ Refresh other parameters on change

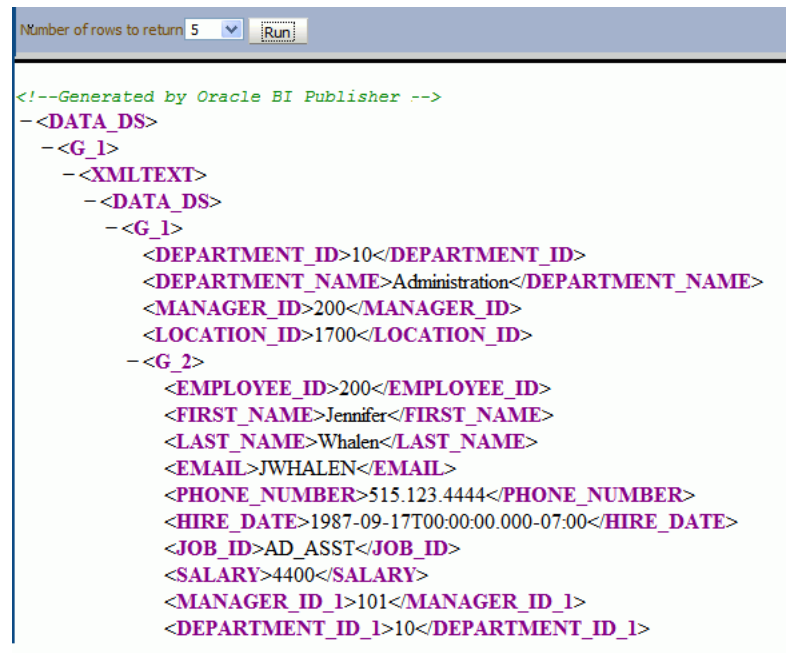
In the bursting query, pass the parameter value to the TEMPLATE field using :P1 as shown in the following figure:

Figure 8–5 Updating the Bursting Query to Accept the P1 Parameter

8.6 Defining the Split By and Deliver By Elements for a CLOB/XML Data Set

If the split-by and deliver-by elements required for your bursting definition reside in a data set retrieved from a CLOB column in a database, BI Publisher cannot parse the XML to present the elements in the **Split By** and **Deliver By** lists. You therefore must manually enter the XPath to locate each element in the retrieved XML data set. To ensure that you enter the path correctly, use the data model editor's **Get XML Output** feature to view the XML that is generated by the data engine.

For example, the sample XML code, shown in [Figure 8–6](#), was stored in a CLOB column in the database called "XMLTEXT", and extracted as an XML data set:

Figure 8–6 Sample Data Extract of Data Stored as CLOB


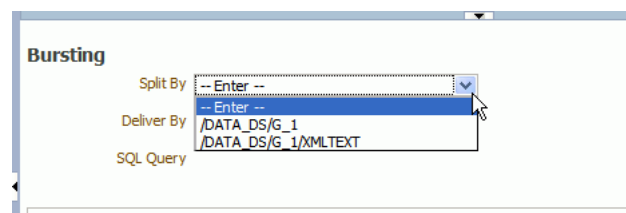
```

<!--Generated by Oracle BI Publisher -->
<DATA_DS>
  <G_1>
    <XMLTEXT>
      <DATA_DS>
        <G_1>
          <DEPARTMENT_ID>10</DEPARTMENT_ID>
          <DEPARTMENT_NAME>Administration</DEPARTMENT_NAME>
          <MANAGER_ID>200</MANAGER_ID>
          <LOCATION_ID>1700</LOCATION_ID>
        <G_2>
          <EMPLOYEE_ID>200</EMPLOYEE_ID>
          <FIRST_NAME>Jennifer</FIRST_NAME>
          <LAST_NAME>Whalen</LAST_NAME>
          <EMAIL>JWHALEN</EMAIL>
          <PHONE_NUMBER>515.123.4444</PHONE_NUMBER>
          <HIRE_DATE>1987-09-17T00:00:00.000-07:00</HIRE_DATE>
          <JOB_ID>AD_ASST</JOB_ID>
          <SALARY>4400</SALARY>
          <MANAGER_ID_1>101</MANAGER_ID_1>
          <DEPARTMENT_ID_1>10</DEPARTMENT_ID_1>

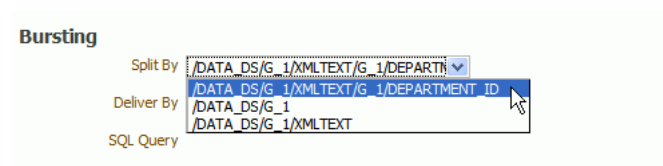
```

For this example, you want to add a bursting definition with split by and deliver by element based on the DEPARTMENT_ID, which is an element within the CLOB/XML data set.

When you add the bursting definition, the Split By and Deliver By lists cannot parse the structure beneath the XMLTEXT element. Therefore, the list does not display the elements available beneath the XMLTEXT node, as shown in [Figure 8–7](#).

Figure 8–7 Split By List Presents Only Top-Level Nodes

To use the DEPARTMENT_ID element as the Split By element, manually type the XPath into the field as shown in [Figure 8–8](#).

Figure 8–8 Manually Entering the XPath into the Split By Field

8.7 Configuring a Report to Use a Bursting Definition

Although you can define multiple bursting definitions for a single data model, you can enable only one for a report.

Enable a report to use a bursting definition on the **Report Properties** dialog of the report editor. For more information see the section "Configuring Report Properties" in *Oracle Fusion Middleware Report Designer's Guide for Oracle Business Intelligence Publisher*.

After you configure the report to use the bursting definition, when you schedule a job for this report you can choose to use the bursting definition to format and deliver the report. For more information see the section "Creating a Bursting Job" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Publisher*.

You can also opt not to use the bursting definition and choose your own output and destination as a regular scheduled report.

8.8 Sample Bursting Query

The following example is based on an invoice report. This report is to be delivered by CUSTOMER_ID to each customer's individual e-mail address.

This example assumes that the delivery and formatting preferences for each customer are contained in a database table named "CUSTOMERS". The CUSTOMERS table includes the following columns that will be retrieved to create the delivery XML dynamically at runtime:

- CST_TEMPLATE
- CST_LOCALE
- CST_FORMAT
- CST_EMAIL_ADDRESS

The SQL code to generate the delivery data set for this example is as follows:

```
select distinct
CUSTOMER_ID as "KEY",
CST_TEMPLATE TEMPLATE,
CST_LOCALE LOCALE,
CST_FORMAT OUTPUT_FORMAT,
'EMAIL' DEL_CHANNEL,
CST_EMAIL_ADDRESS PARAMETER1,
'accounts.receivable@oracle.com' PARAMETER2,
'bip-collections@oracle.com' PARAMETER3,
'Your Invoices' PARAMETER4,
'Hi' || CUST_FIRST_NAME || chr(13) || 'Please find attached your
invoices.' PARAMETER5,
'true' PARAMETER6,
'donotreply@mycompany.com' PARAMETER7
from CUSTOMERS
```

8.9 Creating a Table to Use as a Delivery Data Source

If the delivery information is not easily available in the existing data sources, then you can consider creating a table to use for the query to create the delivery XML. Following is a sample:

Important: If the JDBC driver that you use does not support column alias, when you define the bursting control table, the columns must match exactly the control XML tag name. For example, the KEY column must be named "KEY", upper case is required. PARAMETER1 must be named "PARAMETER1", not "parameter1" nor "param1", and so on.

```
CREATE TABLE "XXX"."DELIVERY_CONTROL"
( "KEY" NUMBER,
  "TEMPLATE" VARCHAR2(20 BYTE),
  "LOCALE" VARCHAR2(20 BYTE),
  "OUTPUT_FORMAT" VARCHAR2(20 BYTE),
  "DEL_CHANNEL" VARCHAR2(20 BYTE),
  "PARAMETER1" VARCHAR2(100 BYTE),
  "PARAMETER2" VARCHAR2(100 BYTE),
  "PARAMETER3" VARCHAR2(100 BYTE),
  "PARAMETER4" VARCHAR2(100 BYTE),
  "PARAMETER5" VARCHAR2(100 BYTE),
  "PARAMETER6" VARCHAR2(100 BYTE),
  "PARAMETER7" VARCHAR2(100 BYTE),
  "PARAMETER8" VARCHAR2(100 BYTE),
  "PARAMETER9" VARCHAR2(100 BYTE),
  "PARAMETER10" VARCHAR2(100 BYTE),
  "OUTPUT_NAME" VARCHAR2(100 BYTE),
  "SAVE_OUTPUT" VARCHAR2(4 BYTE),
  "TIMEZONE" VARCHAR2(300 BYTE),
  "CALENDAR" VARCHAR2(300 BYTE)
) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "EXAMPLES";
```

Tips for creating a creating bursting delivery table:

- If the split data set does not contain a DELIVERY_KEY value, then the document is neither delivered nor generated. For example, using the preceding example, if customer with ID 123 is not defined in the bursting delivery table, this customer's document is not generated.
- To enable a split data set to generate more than one document or deliver to more than one destination, duplicate the DELIVERY_KEY value and provide different sets of OUTPUT_FORMAT, DEL_CHANNEL, or other parameters. For example, customer with ID 456 wants his document delivered to two e-mail addresses. To achieve this, insert two rows in the table, both with 456 as the DELIVERY_KEY and each with its own e-mail address.

Index

A

ABS, 4-27
Actions menu
 group, 4-10
aggregate elements
 group level, 4-13
 setting properties, 4-15
analysis
 See Oracle BI analysis, 2-3
AVG, 4-27

B

backup data source
 using for data model, 2-5
bind variables
 adding, 3-4, 3-10
bursting
 setting up, 8-2
bursting definition
 defining for CLOB data set, 8-9

C

CEILING, 4-27
CLOB
 adding bursting definitions for, 8-9
 using as a data source, 3-27
code view, 4-6
CONCAT, 4-28

D

data model
 component definitions, 2-1
data model editor
 interface overview, 2-4
 launching, 2-4
data sets
 creating, 3-1
 editing, 3-3
 guidelines for multiquery data models, 4-3
 linking, 4-7
 supported types, 3-2
 what is supported for each type, 2-2
data sources

 supported types, 2-2
Database Fetch Size property, 2-5
DATE, 4-28
default data source property, 2-5
diagram view, 4-5

E

element properties
 setting, 4-18
element-level links, 4-7
 deleting, 4-9
elements
 renaming, 4-26
Excel data source files
 refreshing local files, 2-7

F

filters
 creating for groups, 4-16
 deleting, 4-18
 editing, 4-18
find variables
 user information, 3-33
FLOOR, 4-27
FORMAT_DATE, 4-28
FORMAT_NUMBER, 4-28
functions
 element level, 4-18
 global level, 4-22
 reference, 4-27

G

global-level functions, 4-22
group filters
 creating, 4-16
 deleting, 4-18
 editing, 4-18
group link
 Actions menu, 4-10
group-level links, 4-7
 creating, 4-10
 deleting, 4-11

H

HTTP XML feed
 creating data set from, 3-31
HTTP XML feed data sets
 limitations on structuring, 3-32

I

INSTR, 4-28

L

LDAP
 creating bind variables from user attributes, 3-34
 using as data source, 3-13
LENGTH, 4-28
links
 creating between data sets, 4-7
 element-level, 4-7
 group-level, 4-7
list of values
 adding to data model, 5-8

M

master-detail links, 4-7
MAX, 4-27
MDX query
 defining as a data set type, 3-12
Microsoft Excel
 creating a data set based on, 3-14
 data source from system directory, 3-16
 deleting an uploaded data source file, 3-18
 refreshing an uploaded data source file, 3-18
 uploading a local file to use as a data source, 3-17
MIN, 4-27
multiquery data models
 when to use, 4-4

N

null elements
 setting value for, 4-27
NUMBER, 4-28
NVL, 4-28

O

OLAP data sources
 limitations, 3-13
 querying, 3-12
Oracle BI analysis
 creating a data set, 3-19
 limitations as data set, 3-20
Oracle DB Default Package property, 2-5
output view, 4-6

P

parameters

 adding to data model, 5-1
 date, 5-7
 menu, 5-4
 text, 5-3
parent-child groups
 moving element between, 4-12
parent-child links, 4-7
PL/SQL
 add element, 4-22
PL/SQL filters, 4-17
properties
 aggregate elements, 4-15
 setting, 2-5
properties pane, 2-4

Q

Query Builder
 join conditions, 3-8
 joining objects, 3-8
 supported column types, 3-5
 using, 3-4

R

renaming elements, 4-26
ROUND, 4-27
RSS feed
 creating data set from, 3-31

S

Safari browser
 limitations when viewing XML, 3-32
sample data
 attaching to data model, 2-6
 exporting, 3-33
 generating and saving for a data model, 3-32
sorting
 apply to group, 4-19
 support, 4-19
SQL query
 defining as data set, 3-3
 editing, 3-12
STRING, 4-28
Structure view, 4-6, 4-26
subgroups
 creating in data models, 4-11
 removing, 4-12
SUBSTRING, 4-28
SUM, 4-28
system variables
 including in a data model, 3-33

T

toolbar, 2-5

U

user ID

- including in data model, 3-33
- user information
 - including in data model, 3-33
- user preferences
 - including in data model, 3-33
- user roles
 - including in data model, 3-33

V

- value if null, 4-27
- view object
 - creating a data set from, 3-20
 - limitations as data set, 3-21

W

- Web service
 - defining as data source, 3-21
 - limitations on structuring, 3-26
 - supported formats, 3-21

X

- XML feed
 - creating data set from over HTTP, 3-31
- XML file
 - using as a data source, 3-26
- XML file data sets
 - limitations on structuring, 3-27
- XML output
 - viewing from data model editor, 2-5
- XML output options
 - setting, 2-6
- XML tag display
 - setting upper or lower case, 2-6

