Oracle® Business Intelligence Server Administration Guide

Version 10.1.3.2

December 2006

ORACLE®

Contents

1 What’s New in This Release

Oracle Business Intelligence Enterprise Edition consists of components that were formerly available from Siebel Systems as Siebel Business Analytics Platform, with a number of significant enhancements.

The Oracle Business Intelligence Server Administration Guide is part of the documentation set for Oracle Business Intelligence Enterprise Edition. This guide contains information about setting up the Oracle Business Intelligence Server. This guide contains new material and material that was previously published under the title *Siebel Business Analytics Server Administration Guide*.

Oracle recommends reading the Oracle Business Intelligence Enterprise Edition Release Notes before installing, using, or upgrading the Oracle BI Infrastructure. The Oracle Business Intelligence Enterprise Edition Release Notes are available:

- On the Oracle Business Intelligence Enterprise Edition CD-ROM.
- On the Oracle Technology Network at http://www.oracle.com/technology/documentation/bi_ee.html (to register for a free account on the Oracle Technology Network, go to http://www.oracle.com/technology/about/index.html).

What’s New in Oracle Business Intelligence Server Administration Guide, Version 10.1.3.2

Table 1 on page 11 lists changes described in this version of the documentation to support Release 10.1.3.2 of the software. These changes include the name of this guide and many of the products.

Table 1. Changed Features and Information in Oracle Business Intelligence Server Administration Guide, Version 10.1.3.2

Topic	Description
“Menus in the Administration Tool” on page 18	Updated menu items to reflect changes to product names and changed functionality.
“Keyboard Shortcuts in the Administration Tool” on page 21	Added new shortcuts.
“Icons and Symbols in the Administration Tool” on page 21	Added new icons for physical cube hierarchy types, opaque views, and aggregate objects.
“Features and Options for Oracle’s Siebel Marketing Application” on page 24	Added topic.
“Checking the Consistency of a Repository or a Business Model” on page 26	Rewrote topic to describe the new Consistency Check Manager.

Table 1. Changed Features and Information in Oracle Business Intelligence Server Administration Guide, Version 10.1.3.2

Topic	Description
“Using the Options Dialog Box—General Tab” on page 29	Updated Calculation Wizard information, added import repository option, and removed options for Time Series Wizard and Merge repository mismatch warning.
“Using the Options Dialog Box—Multiuser Tab” on page 33	Added information about new tab.
“Populating Logical Level Counts Automatically” on page 36	Added topic.
“Process of Creating the Physical Layer from Multidimensional Data Sources” on page 58	Updated section to include support for SAP/BW and new versions of Microsoft Analysis Services.
“Importing a Physical Schema from Multidimensional Data Sources” on page 59	Updated the steps to import objects to include support for SAP/BW.
“Creating a Database Object Manually in the Physical Layer” on page 61	Added field needed to create a database using a virtual private database as a source.
“Creating or Changing Connection Pools” on page 67	Updated general properties for setting up connection pools for all data sources.
“Setting Up Connection Pool Properties for Multidimensional Data Sources” on page 73	Updated general properties for setting up connection pools.
“About Physical Tables” on page 81	Updated topic.
“About Physical Alias Tables” on page 83	Added topic.
“Creating and Administering General Properties for Physical Tables” on page 85	Updated this topic to include changes for multidimensional data sources and added new properties for alias tables.
“About Measures in a Multidimensional Data Source” on page 88	Added topic.
“Creating and Administering Columns and Keys in a Physical Table” on page 87	Updated topic and procedure to include instructions for alias tables.
“Setting Up Hierarchies in the Physical Layer for a Multidimensional Data Source” on page 91	Updated topic to reflect new properties.

Table 1. Changed Features and Information in Oracle Business Intelligence Server Administration Guide, Version 10.1.3.2

Topic	Description
“Updating Member Counts” on page 94 and “Viewing Members in Physical Cube Tables” on page 94	Added topics about updating member counts and viewing member data.
“Deploying Opaque Views” on page 103	Added topics that explain how to deploy, undeploy, and remove opaque views.
“Creating Dimensions” on page 126	Updated topic.
“Creating Dimension Levels and Keys” on page 126	Updated topic and added topics about working with time dimensions and chronological keys.
“Selecting and Sorting Chronological Keys in a Time Dimension” on page 130	Added topic.
“Importing From Another Repository” on page 154	Added information about changes to the import from repository process.
“Querying and Managing Repository Metadata” on page 156	Added procedures for saving queries and deleting saved queries. Updated procedures for running queries and saving query results to an external file.
“About Extracting Metadata Subsets Into Projects” on page 167	Added an explanation about how project extract work.
“Setting Up the SA System Subject Area” on page 181	Revised topic and added caution note about authorization and authentication configuration option.
Extracting Analytics Metadata Using Dependency Tracking	This topic replaced by “Generating and Deploying a Metadata Dictionary” on page 187
Time Series Wizard	Removed topic from “Oracle BI Administration Tool Utilities and Expression Builder” on page 183.
Synchronize Aliases	Removed topic from “Oracle BI Administration Tool Utilities and Expression Builder” on page 183. Synchronization is now automatic.
“Generating and Deploying a Metadata Dictionary” on page 187	Added topic.
“Aggregate Persistence Wizard” on page 189	Added topic.
“Calculation Wizard” on page 190	Added topic.
“About Time Series Conversion Functions” on page 197	Added topic to update Expression Builder.
“About the IndexCol Conversion Function” on page 199	Added topic to update Expression Builder.

Table 1. Changed Features and Information in Oracle Business Intelligence Server Administration Guide, Version 10.1.3.2

Topic	Description
“Administering Usage Tracking” on page 219	Added information about how usage tracking works.
“Purging and Maintaining Cache Using ODBC Procedures” on page 235	Added information about the SAGetSharedRequestKey ODBC procedure.
“Storing and Purging Cache for SAP/BW Data Sources” on page 237	Added topic.
“Creating Aggregates for Oracle BI Server Queries” on page 241	Added topics.
“Understanding and Creating Session Variables” on page 286	Added description of Oracle Business Intelligence Disconnected Analytics variables.
“Creating Repository Variables” on page 285 and “Creating Session Variables” on page 289	Reorganized and updated these topics to reflect the changes in user interface design. Also added information about virtual private database parameters.
“About Using Initialization Blocks With Variables” on page 290 and “Process of Creating Initialization Blocks” on page 293	Revised all topics to conform to new user interface.
“About Authenticating Users Using Initialization Blocks” on page 293	Retitled and revised topics.
“Clustering Oracle BI Servers” on page 301	Added information about Oracle BI Scheduler to this chapter.
“Setting Parameters in the NQSConfig.INI File” on page 304 and “Setting Parameters in the NQClusterConfig.INI File” on page 304	Removed parameters from these topics. Parameters are described in the *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.
“Setting Up LDAP Authentication” on page 324 and “Maintaining Oracle BI Server User Authentication” on page 329	Expanded and reorganized topics about LDAP authentication.
“Maintaining Oracle BI Server User Authentication” on page 329	Added cross-reference to *Oracle Business Intelligence Enterprise Edition Deployment Guide*.
“Using FILTER to Compute a Conditional Aggregate” on page 362	Added topic.
“Aggregate Functions” on page 366	Added and updated several topics about time series functions.
“IndexCol” on page 400	Added conversion function.

Table 1. Changed Features and Information in Oracle Business Intelligence Server Administration Guide, Version 10.1.3.2

Topic	Description
“Description of the Usage Tracking Data” on page 406	Added data columns.
“Oracle BI Server Authentication APIs” on page 409	Added appendix.

Other Documentation Changes

- Added notes throughout the guide to identify the repository Presentation layer catalogs as subject areas.

2 Oracle BI Administration Tool Basics

This section provides an overview of the user interface components included in the Administration Tool. The Administration Tool is a Windows application that allows the Oracle BI Administrator to create and edit repositories.

NOTE: In this guide, tables of values and descriptions contain only values that need a detailed description. Self-explanatory values such as Show Toolbar do not need to be listed.

This section contains the following topics:

Administration Tool User Interface Components

This section includes a description of the following interface components:

Main Window in the Administration Tool

The main window of the Administration Tool is a graphical representation of the following three parts of a repository:

- **Physical layer**. Represents the physical structure of the data sources to which the Oracle BI Server submits queries. This layer is displayed in the right pane of the Administration Tool.
- **Business Model and Mapping layer.** Represents the logical structure of the information in the repository. The business models contain logical columns arranged in logical tables, logical joins, and dimensional hierarchy definitions. This layer also contains the mappings from the logical columns to the source data in the Physical layer. It is displayed in the middle pane of the Administration Tool.
- **Presentation layer.** Represents the presentation structure of the repository. This layer allows you to present a view different from the Business Model and Mapping layer to users. It is displayed in the left pane of the Administration Tool.

Figure 1 on page 18 shows the three layers of a repository, as described in the preceding list.

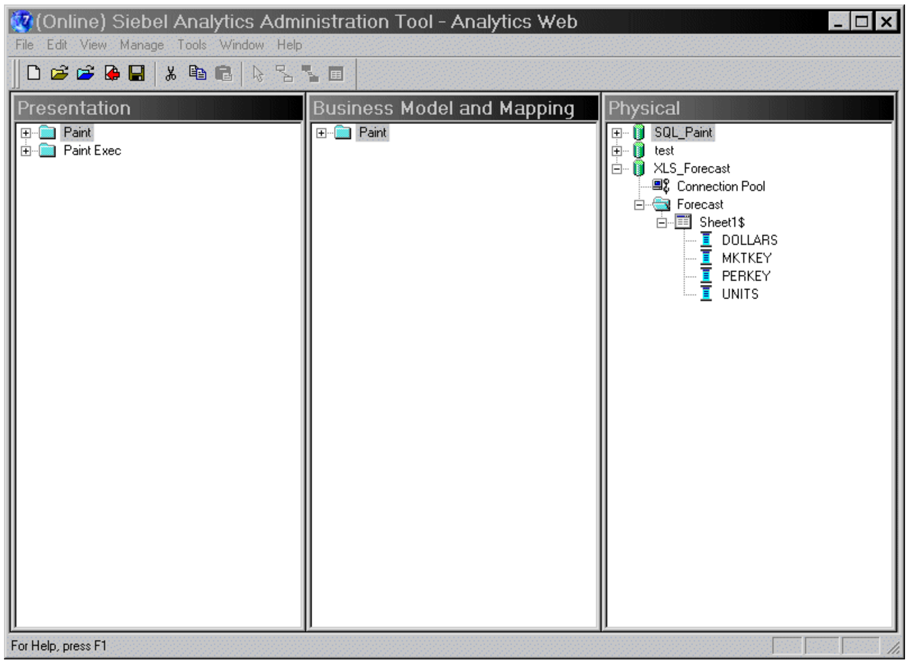


Figure 1. Example Administration Tool Main Window

Menus in the Administration Tool

The Administration Tool includes the following menus:

File

The File menu contains options to work with repositories as well as several server-related options that are active when a repository is open in online mode. Before you open a repository, the File menu has fewer commands available.

Edit

The Edit menu options allow you to perform basic editing functions (cut, copy, paste, duplicate, and delete) on objects in a repository. Additionally, you can view and edit some of the object properties.

View

The View menu options toggle the view of specific metadata panes, give you access to the Join diagrams, and refresh the repository view.

Manage

The Manage menu allows you to access the management functions described in Table 2 on page 19.

Table 2. Manage Menu Functions

Menu Option	Description
Jobs	This option is available when a repository is opened in online mode. The Job Manager is the management interface to *Oracle Business Intelligence Scheduler Guide*. For more information about using BI Scheduler, see *Oracle Business Intelligence Scheduler Guide*.
Sessions	This option is available when a repository is opened in online mode. In the Session Manager, you can monitor activity on the system, including the current values of repository and session variables.
Cache	This option is available when a repository is opened in online mode and caching is enabled. The Cache Manager allows you to monitor and manage the cache.
Clusters	This option is available when the Oracle BI Cluster Server is installed. The Cluster Manager monitors and manages the operations and activities of the cluster.
Security	The Security Manager displays security information for a repository and provides a central location for security configuration and management.
Joins	The Joins Manager allows you to work with physical and logical joins.
Variables	The Variables Manager allows you to create, edit or delete variables.
Projects	The Project Manager allows you to create, edit, or remove projects or project elements. Project elements include presentation catalogs (repository subject areas), logical fact tables, groups, users, variables, and initialization blocks. You use projects during multiuser development. For more information, refer to "Setting Up a Multiuser Development Environment (Administrator)" on page 170.
Marketing	Applies to Oracle's Siebel Marketing product. You need a separate license to use this product. For information about using Marketing with Oracle BI, refer to the administrator documentation for the Siebel Marketing application.

Tools

The Tools menu options allow you to update all row counts, open the Query Repository dialog box, set options for the Administration Tool, and work with various utilities.

Window

The Window menu options allow you to cascade or tile open layer windows and toggle among them.

NOTE: The Cascade and Tile options are only available if you clear the Tile when resizing check box in the Tools > Options dialog box, in the General tab.

Help

The Help menu allows you to obtain the following information:

- Help Topics. Access the online help system for the Administration Tool.
- Oracle on Web. Access the Oracle Web site.
- About Administration Tool. Obtain version information about Oracle BI Server Administration Tool.

Toolbar in the Administration Tool

The toolbar provides access to functions that you use frequently.

To toggle the toolbar on and off

- Select Tools > Options > Show Toolbar.

To dock the toolbar

- Place your cursor on the double bars to the left of the toolbar, and then click and drag to where you want to place the toolbar.

Keyboard Shortcuts in the Administration Tool

Table 3 on page 21 presents the keyboard shortcuts you can use in the Administration Tool to perform frequent tasks.

Table 3. Keyboard Shortcuts

Menu	Command	Shortcut
File	New	CTRL + N
	Open > Offline	CTRL + F
	Open > Online	CTRL + L
	Save	CTRL + S
	Check Global Consistency	CTRL + K
Edit	Cut	CTRL + X
	Copy	CTRL + C
	Paste	CTRL + V
	Delete	DEL
View	Refresh	F5
Tools	Show Consistency Checker	CTRL + E
	Query Repository	CTRL + Q

Icons and Symbols in the Administration Tool

For most icons, you can obtain the object type by, double-clicking the icon and reading the title bar of the dialog box. Table 4 on page 21 presents some of the icons and symbols that elaborate on the name in the title bar. Some of the items in this list are symbols used in conjunction with other icons to expand their meanings.

Table 4. Icons and Symbols

Icon or Symbol	What It Represents
🗔	Stored procedure call object, as specified by the Object Type option in the General tab of the Physical Table dialog box.
👓	View object.
👓	Opaque view object after deployed.

Table 4. Icons and Symbols

Icon or Symbol	What It Represents
🔑	A primary key for a physical or logical table (yellow).
🔑	A foreign key for a physical or logical table in the Joins Manager (gray).
🔑	A key for a logical dimension level (blue).
🔑	A key for a multidimensional data source physical level (green).
⧉	A physical or logical complex join in the Joins Manager.
⚠	Unknown hierarchy type. Typically, this icon is assigned to a hierarchy when a cube is imported. You need to assign a valid type to each hierarchy after import.
🖧	Fully Balanced hierarchy type for a multidimensional data source in the Physical layer.
🖧	Unbalanced hierarchy type for a multidimensional data source in the Physical layer.
🖧	Ragged Balanced hierarchy type for a multidimensional data source in the Physical layer.
🖧	Network hierarchy type for a multidimensional data source in the Physical layer.
▤	A level in the Business Model and Mapping layer.
▤	A level in the Business Model and Mapping layer in which a level key contains one or more columns from another level.
▤	A level for a multidimensional data source in the Physical layer.
≡	A physical or logical column.
Σ	A logical column with an aggregation rule.
fx	A derived logical column.

Table 4. Icons and Symbols

Icon or Symbol	What It Represents
🧊	A physical cube column from a multidimensional data source. This icon represents columns that are not measures.
🧊	A physical cube column from a multidimensional data source. This icon represents columns that are measures.
⚠	An invalid item. For example, a column may be invalid, if it has no physical mapping.
🚫	A collapsed business model in the Business Model and Mapping layer that is not available for queries.
🚫	An expanded business model in the Business Model and Mapping layer that is not available for queries.
A	An item that contains an attribute definition.
✳	Overlays other symbols to indicate a new item that has not been checked in (appears in online mode only). For example, this icon would appear on top of an alias table icon to indicate an alias table is new.
	A system DSN ODBC entry. Appears in the Import dialog box.
A	A measure definition.
✓	Overlays other icons to indicate an object that is checked out. For example, this icon would appear on top of a table icon to indicate that the table has been checked out.
[?]	A static repository variable.
[?]	A dynamic repository variable.
[?]	A system session variable.
[?]	A nonsystem session variable.
🧊	An initialization block.
A	A group association (appears only in the results display of the Query Repository dialog box).

Table 4. Icons and Symbols

Icon or Symbol	What It Represents
	A level-to-level relationship (appears only in the results display of the Query Repository dialog box).
	A type privilege (appears only in the results display of the Query Repository dialog box).
	A query privilege (appears only in the results display of the Query Repository dialog box).
	A privilege package (appears only in the results display of the Query Repository dialog box).
P	An object privilege (appears only in the results display of the Query Repository dialog box).
✂	Overlays other icons to indicate an object that has been cut. Appears with other symbols, for example, to indicate that a cut item is an alias table.

Online Help in the Administration Tool

Most windows and dialog boxes have help buttons that open online help topics containing information to help you complete a task.

Features and Options for Oracle’s Siebel Marketing Application

Some features and options in the Oracle BI Server Administration Tool are for use by organizations that have Siebel Marketing. For information, refer to the administrator documentation for the Siebel Marketing application.

Online and Offline Repository Modes

You can open a repository for editing in either online or offline mode. You can perform different tasks based on the mode in which you opened the repository.

This section includes the following topics:

- Opening a Repository in Offline Mode on page 25
- Opening a Repository in Online Mode on page 25
- Checking In Changes on page 26

Opening a Repository in Offline Mode

Use offline mode to view and modify a repository while it is not loaded into the Oracle BI Server. If you attempt to open a repository in offline mode while it is loaded into the Oracle BI Server, the repository opens in read-only mode. Only one Administration Tool session at a time can edit a repository in offline mode.

To open a repository in offline mode

1. In the Administration Tool, select File > Open > Offline.
2. Navigate to the repository you want to open, and then select Open.
3. In the Open Offline dialog box, type a valid user ID and password, and then click OK.

   This opens the repository for editing.

   NOTE: If the server is running and the repository you are trying to open is loaded, the repository will only open in read-only mode. If you want to edit the repository while it is loaded, you have to open it in online mode. Also, if you open a repository in offline mode and then start the server, the repository will be available to users; any changes you make will become available only when the server is restarted.

Opening a Repository in Online Mode

Use online mode to view and modify a repository while it is loaded into the Oracle BI Server. The Oracle BI Server must be running to open a repository in online mode. There are certain things you can do in online mode that you cannot do in offline mode. In online mode, you can perform the following tasks:

- Manage scheduled jobs
- Manage user sessions
- Manage the query cache
- Manage clustered servers
- Stop the Oracle BI Server

To open a repository in online mode

1. In the Administration Tool, select File > Open > Online.

   The Open Online Repository dialog box appears, from which you select a data source.

   The data sources displayed in the dialog box are all the User and System DSNs on your computer that are configured using the Oracle BI ODBC driver.

2 Select a data source, type a valid user ID and password, and then click OK.

The repository opens that contains the business model corresponding to the data source selected.

NOTE: If you expect to work extensively with the repository (for example, you plan to check out many objects), select the Load all objects option. This loads all objects immediately, rather than as selected. The initial connect time may increase slightly, but opening items in the tree and checking out items will be faster.

Checking In Changes

When you are working in a repository open in online mode, you will be prompted to perform a consistency check before checking in the changes you make to a repository.

If you have made changes to a repository and then attempt to close the repository without first checking in your changes, a dialog box opens automatically asking you to select an action to take. If you move an object from beneath its parent and then attempt to delete the parent, you will be prompted to check in changes before the delete is allowed to proceed.

Use the Check in Changes dialog box to perform the following tasks:

- Make changes available immediately for use by other applications. Applications that query the Oracle BI Server after you have checked in the changes will recognize them immediately. Applications that are currently querying the server will recognize the changes the next time they access any items that have changed.
- Specify that repository changes should be written to disk immediately. If the Oracle BI Server is shut down abnormally, using the Check Changes dialog box will make sure that checked-in changes to the repository can be recovered. Changes that are checked in but not saved to disk will be restored through the server's error recovery processing. (This processing takes place automatically whenever the Oracle BI Server has been shut down abnormally.)

To make changes available and have them saved to disk immediately

- In the Administration Tool, select File > Check in Changes.

If the Administration Tool detects an invalid change, an informational message appears to alert you to the nature of the problem. Correct the problem and perform the check-in again.

NOTE: If you make changes to a repository open in online mode, and then attempt to stop the Oracle BI Server, this option will not be available. This is because your changes will be saved automatically when the server shuts down.

Checking the Consistency of a Repository or a Business Model

Repository metadata must pass a consistency check before you can make the repository available for queries. The Consistency Check Manager allows you to enable and disable rules for consistency checks, navigate to and fix inconsistent objects, and limit the consistency check to specific objects.

The Consistency Check Manager does not check the validity of objects outside the metadata using the connection. It only checks the consistency of the metadata and not any mapping to the physical objects outside the metadata. If the connection is not working or objects have been deleted in the database, the Consistency Check Manager will not report these errors.

The consistency checker returns the following types of messages:

- **Error.** These messages describe errors that need to be fixed. Use the information in the message to correct the inconsistency, and then run the consistency checker again. The following is an example of an error message:

  ```
  [38082] Type of Hierarchy '"0RT_C41"..."0RT_C41/MDF_BW_Q02"."Product Hierarchy for
  Material MARA"' in Cube Table '"0RT_C41"..."0RT_C41/MDF_BW_Q02"' needs to be set.
  ```

  NOTE: If you disable an object and it is inconsistent, you will receive a message asking if you want to make it unavailable for queries.

- **Warning.** These messages indicate conditions that may or may not be errors, depending upon the intent of the Oracle BI Administrator. For example, a warning message about a disabled join may be the result of the Oracle BI Administrator intentionally disabling a join, such as eliminating a circular join condition. The following is an example of a warning message:

  ```
  [39024] Dimension '"Paint"."MarketDim"' has defined inconsistent values in its
  levels' property 'Number of elements'.
  ```

- **Best practices.** These messages provide information about conditions but do not indicate an inconsistency. The message appears if a condition violates a best practice recommendation. The following is an example of a best practice message:

  ```
  [89001] The Physical Table '"XLS_Forecast"."Forecast".."Sheet1$"' has no keys
  defined.
  ```

NOTE: After upgrading from a previous software version and checking the consistency of your repository, you might notice messages that you had not received in previous consistency checks. This typically indicates inconsistencies that had been undetected prior to the upgrade, not new errors.

In the Consistency Check Manager, in the Messages tab, you can sort the rows of messages by clicking the column headings. Additionally, the status bar provides a summary of all the rows displayed

Setting Up Consistency Check Manager

During installation, a default subset of rules is installed. On each workstation, users can use the default subset of rules or change the subset by adding or deleting rules. By default, all consistency check rules are enabled and all types of messages are set to display after you run a consistency check. You can disable any of the consistency check rules and change the messages list so that one or more of the messages types do not appear. However, at least one message type must be enabled.

To set up the Consistency Check Manager

1. In the Administration Tool, select Tools > Show Consistency Checker.
2. In the Consistency Check Manager, in the Messages tab, perform the following steps:

   a. Clear the check box for any message types that you do not want to display.
   b. If you want the message to show the fully qualified name of each object, select the Show Qualified Name check box.

3. Click the Options tab.
4. Expand each message type.
5. To disable an enabled rule, select the rule and click Disable.
6. To enable a disabled rule, select the rule and click Enable.
7. If you do not want to check for consistency at this time, click Close.

Checking the Consistency of a Repository

Currently you can check consistency in the following ways:

- You can check global consistency of the repository from the File menu and from the Consistency Check Manager (Check All Objects). If you disabled any rules in Consistency Check Manager, those rules will not be checked.

  NOTE: If you disable an object and it is inconsistent, You will receive a message asking if you want to make it unavailable for queries.

- You can check the consistency of a business model from the right click menu of a business model.
- You can check the consistency of some or all of the objects in the repository from the Consistency Check Manager. To limit the objects that are checked, in Consistency Check Manager, in the Options tab, you disable the rules for objects that you want to exclude.

To check the consistency of a repository

1. In the Administration Tool, from the Tools menu, select Show Consistency Checker.

   If you checked consistency in the current session, the messages from the last check appear in the Messages tab.

2. In Consistency Check Manager, click the Options tab and verify the rules are set appropriately.
3. Click the Messages tab and click Check All Objects.

   If you already have checked consistency in the current session and then change the rules, you might notice different messages.

4. To edit the repository to correct inconsistencies, perform the following steps:
   a. Double-click any cell in a row and the properties dialog box for that object opens.
   b. In the properties dialog box for the object, correct the inconsistency, and then click OK.
5. To copy the messages so that you can paste them in another file such as a spreadsheet, click Copy.

6. To check consistency again, click Check All Objects.

   When you select Show Consistency Checker from the Tools menu or Check Global Consistency from the File menu, the refresh button in the top right corner of the Consistency Check Manager dialog box checks all objects.

7. When finished, click Close.

To check the consistency of a single object in a repository

1. In the Administration Tool, right-click an object, and then select Check Consistency.

   If the object is not consistent, a list of messages appears.

2. To edit the repository to correct inconsistencies, perform the following steps:

   a. Double-click any cell in a row and the properties dialog box for that object opens.

   b. In the properties dialog box for the object, correct the inconsistency, and then click OK.

3. To copy the messages so that you can paste them in another file such as a spreadsheet, click Copy.

4. To check consistency of the object again, click the refresh button at the top right corner of the dialog box.

   NOTE: If you click Check All Objects, all objects in the repository will be checked.

Setting Preferences

You can use the Options dialog box to set preferences for the Administration Tool.

This section includes the following topics:

- Using the Options Dialog Box—General Tab on page 29
- Using the Options Dialog Box—Repository Tab on page 31
- Using the Options Dialog Box—Sort Objects Tab on page 32
- Using the Options Dialog Box—Cache Manager Tab on page 32
- Using the Options Dialog Box—Multiuser Tab on page 33
- Using the Options Dialog Box—More Tab on page 33

Using the Options Dialog Box—General Tab

Use the General tab of the Options dialog box to set general preferences for the Administration Tool.

To set general preferences

1. In the Administration Tool, select Tools > Options.

2 In the Options dialog box, select the General tab.

3 Select the options you want.

The following list contains the options that need additional explanation:

Option	Action When Selected
Tile when resizing	Automatically tiles the layer panes of the repository when you resize the Administration Tool. When this option is checked, the Cascade and Tile options are not available in the Windows menu of the Administration Tool.
Display qualified names in diagrams	Makes it easier to identify table sources.
Display original names for alias in diagrams	Makes it easier to identify the actual table referenced.
Show Calculation Wizard introduction page	Displays the Calculation Wizard introduction page. The introduction page also contains an option to suppress its display in the future. Use the Calculation Wizard to create new calculation columns that compare two existing columns and to create metrics in bulk (aggregated), including existing error trapping for NULL and divide by zero logic. You start the Calculation Wizard in the Business Model and Mapping layer by right-clicking any logical fact or dimension column of data type numeric, and then selecting the option Calculation Wizard. **NOTE:** Time Series calculation is not allowed for a logical table not identified as a time dimension. For more information, refer to topics about time dimensions in "Process of Creating and Administering Dimensions" on page 126.
Check out objects automatically	(online mode only) Automatically checks out the object when you double-click it. **NOTE:** If the option is not selected, you will be prompted to check out the object before you can edit it.

Option	Action When Selected
Show row count in physical view	Displays row counts for physical tables and columns in the Physical Layer. Row counts will not initially display until they are updated. To update the counts, select Tools > Update All Row Counts. You can also right-click on a table or column in the Physical Layer and select the option Update Row Count. Note: Row counts are not shown for items that are stored procedure calls (from the optional Object Type drop-down list in the General tab of the Physical Table dialog). Row counts are not available for XML, XML Server, or multidimensional databases. You cannot update row counts on any new objects until you check them in.
Prompt when moving logical columns	Allows you to ignore, specify an existing, or create a new logical table source for a moved column.
Remove unused physical tables after Merge	Executes a utility to clean the repository of unused physical objects. It might make the resulting repository smaller.
Allow import from repository	When checked, the Import from repository option on the File menu becomes available. **NOTE:** By default, the Import from repository option on the File menu is disabled and this option will not be supported in the future. It is recommended that you create projects in the repository that contain the objects that you wish to import, and then use repository merge to bring the projects into your current repository. For more information, see "About Extracting Metadata Subsets Into Projects" on page 167.

Using the Options Dialog Box—Repository Tab

You can set the following values in the Repository tab:

- **Show tables and dimensions only under display folders.** Oracle BI Administrators can create display folders to organize objects in the Physical and Business Model and Mapping layers. They have no metadata meaning. After you create a display folder, the selected objects appear in the folder as a shortcut and in the database or business model tree as an object. You can hide the objects so that only the shortcuts appear in the display folder.

  For more information about creating display folders, refer to "Setting Up Display Folders in the Physical Layer" on page 98 and "Setting Up Display Folders in the Business Model and Mapping Layer" on page 136.

- **Hide level-based measure.** When selected, level-based measures (columns for which the aggregation rule is other than NONE) will not appear in the Business Model and Mapping layer under the level.

To set options in the Repository tab

1. From the menu bar, choose Tools > Options.
2. In the Options dialog box, click the Repository tab.
3. In the Repository tab, select the options you wish to choose, and then click OK.

Using the Options Dialog Box—Sort Objects Tab

Use the Sort Objects tab to specify which repository objects appear in the Administration Tool in alphabetical order.

To specify which repository objects appear in alphabetical order

1. In the Administration Tool, select Tools > Options.
2. In the Options dialog box, select the Sort Objects tab.
3. Check the boxes for the objects you want to appear in alphabetical order.

   For example, if you want the database objects that appear in the Physical layer to appear in alphabetical order, select the Database check box.

Using the Options Dialog Box—Cache Manager Tab

Use the Cache Manager tab to choose the columns that should be shown in the Cache Manager and the order in which they will be displayed on the Cache tab of the Cache Manager.

To select columns to display in the Cache Manager

1. In the Administration Tool, select Tools > Options.
2. In the Options dialog box, select the Cache Manager tab.
3. Check the boxes for the columns you want display in the Cache Manager.

   Clicking on an already checked box removes the check mark. Unchecked items will not appear in the display.

4. To change the order of columns in the Cache Manager, select an item, and then use the Up and Down buttons to change its position.

Using the Options Dialog Box—Multiuser Tab

Use the Multiuser tab to specify the path to the multiuser development directory. For more information, refer to "Setting Up a Pointer to the Multiuser Development Directory" on page 172.

Using the Options Dialog Box—More Tab

Use the More tab to set the speed when scrolling through the trees in various Administration Tool dialog boxes, and to set the default window size for the join diagrams.

To set the scrolling speed and default window size

1. In the Administration Tool, select Tools > Options.
2. In the Options dialog box, select the More tab.
3. Position the cursor on the Scrolling Speed slider to set the speed.
4. In the Default diagram zoom drop-down list, you can choose a percentage or Best Fit.

   The default window size is Best Fit. If you use the Best Fit option, the following rules apply:

   - If there are five or fewer objects, the zoom level will be 50%.
   - If there are more than five objects, the zoom level changes automatically to Zoom to Fit.

Setting Permissions for Repository Objects

You can assign user and group permissions to connection pools in the Physical layer and to Presentation layer objects. Additionally, you use Security Manager to set up privileges and permissions. For more information about managing security, refer to "Oracle BI Security Manager" on page 315.

The Permissions dialog box displays all currently defined users and groups. to see all users and groups, select the check box named Show all users/groups. For each user and group, you can allow or disallow access privileges for an object by clicking in the check box to toggle among the following options:

- A check mark indicates that a permission is granted.
- An X indicates that a permission is denied.
- An empty check box indicates that a permission has not been modified.

You can access the Permissions dialog box from the following dialog boxes:

- Connection Pool—General tab
- Presentation Catalog Folder—General tab

  NOTE: The term Presentation Catalog in the Administration Tool refers to subject areas.

- Presentation Table—General tab
- Presentation Column—General tab

To add or edit permissions from a connection pool

1. Open a repository in the Administration Tool, expand a physical database, and double-click a connection pool.
2. In the Connection Pool dialog box, click Permissions.
3. In the Permissions dialog box, select the appropriate options for each user and group that you want to modify.
4. Click OK.

Sorting Columns in the Permissions Dialog box

There are six ways that you can sort the types and User/Group names. In the Permissions dialog box, there are three columns. The first column (sort on this column) has no heading and contains an icon that represents the type of user or group of users. The second column (sort on this column) contains the name of the User/Group object, and the third column (cannot sort on this column) contains the Read check box. To change the sort, click the heading of the first or second column.

There are three ways to sort by type and two ways to sort the list of user and group names. This results in a total of six possible sort results (3 x 2 = 6). The following list shows the sort results available by clicking the type column:

- Everyone, Groups, Users (ascending by name of type)
- Users, Groups, Everyone (descending by name of type)
- Type column is in no particular order (Type value is ignored as all names in User/Group column are sorted in ascending order by value in User/Group column)

The following list shows the sort results available by clicking the User/Group column:

- Ascending within the type
- Descending within the type

Examples of Sorting Columns in the Permissions Dialog Box

If you want to sort all rows first by type in ascending order and then, within type, sort the User/Group names in ascending order, use the following steps in the sequence shown:

1. Click the blank heading above the type column until the Everyone type appears at the top.

   The type column is in ascending order.

2. If the User/Group name column is in descending order within each type, click the User/Group heading once.

   The list is sorted by type in ascending order and then within type, by User/Group names in ascending order.

3. To change the sort by type to descending order, leaving the User/Group names in ascending order, click the type (blank) heading once.
4. To change the sort to ignore the type column and sort only on the names in the User/Group column in ascending order, click the type heading once more.
5. To continue ignoring the type column and change the sort for the names in the User/Group column to be in descending order, click the User/Group heading.

Editing, Deleting, and Reordering Objects in the Repository

This section contains the standard steps for editing, deleting, and reordering objects. These instructions will not be repeated for each object in the chapters discussing the layers of the repository unless the material is needed to perform a task.

- To edit objects, double-click the object and complete the fields in the dialog box that appears. In some dialog boxes, you can click Edit to open the appropriate dialog box.
- To delete objects, select the object and click Delete.
- To reorder objects, drag and drop an object to a new location. In some dialog boxes, you can use an up or down arrow to move objects to a new location.

  NOTE: Reordering is only possible for certain objects and in certain dialog boxes.

Displaying and Updating Row Counts for Tables and Columns

When you request row counts, the Administration Tool retrieves the number of rows from the physical database for all or selected tables and columns (distinct values are retrieved for columns) and stores those values in the repository. The time this process takes depends upon the number of row counts retrieved.

When updating all row counts, the Updating Row Counts window appears while row counts are retrieved and stored. If you click Cancel, the retrieve process stops after the in-process table (and its columns) have been retrieved. Row counts include all tables and columns for which values were retrieved prior to the cancel operation.

Updating all row counts for a large repository might take a long time to complete. Therefore, you sometimes might want to update only selected table and column counts.

Row counts are not available for the following:

- Stored Procedure object types
- XML data sources and XML Server databases
- Multidimensional data sources. For information about member counts, refer to "Updating Member Counts" on page 94.

- Data sources that do not support the CountDistinct function, such as Microsoft Access, Microsoft Excel
- In Online mode, Update Row Count will not work with connection pools in which the session variables :USER and :PASSWORD are set as the user name and password.

  In offline mode, the Set values for variables dialog box appears so that you can populate the session variables :USER and :PASSWORD.
- In online mode, after importing or manually creating a physical table or column, Oracle BI Server does not recognize the new objects until you check them in. Therefore, Update Row Count will not be available in the menu until you check in these objects.

To display row counts in the Physical layer

1. Select Tools > Options.
2. In the General tab of the Options dialog box, select the option Show row count in physical view, and then click OK.

To update selected row counts in the Physical layer

1. In the Physical layer, right-click a single table or column.

   You can select multiple objects and then right-click.
2. In the shortcut menu, select Update Rowcount.

To update all row counts in the Physical layer

1. Select Tools > Update All Row Counts.

   If the repository is open in online mode, the Check Out Objects window may open.
2. Click Yes to check out the objects.

   Any row counts that have changed since the last update will be refreshed.

Populating Logical Level Counts Automatically

Estimate Levels allows the Oracle BI Administrator to automatically populate level counts for one or more dimension hierarchies. Level counts are utilized by the query engine to determine the most optimal query plan and optimizes overall system performance.

The repository needs to be opened in online mode and the business model must be available for query. Then, in the Business Model and Mapping layer, the Oracle BI Administrator can select any of the following logical layer elements, and then execute the Estimate Levels command:

- Business model. Must be available for query. If you select this object, the Administration Tool will attempt to check out all objects in the business model.

- Dimension. The Oracle BI Administrator should run a consistency check on dimensions to make sure that the dimension is logically sound.
- A combination of business models and dimensions. The Oracle BI Administrator can select multiple dimensions and multiple business models individually.

When run, the Estimate Levels command also launches a consistency check on the level counts as described in the following list:

- Checks that a level key is valid. Columns in levels have referential integrity.
- Checks the parent-child relationship. If the parent level count is greater that the child level count, an error is returned.
- Generates a run report that lists all the counts that were estimated and any errors or consistency warnings.
- The queries and errors are logged to NQQuery.log on the Oracle BI Server.

  NOTE: Set the log level at 4 or higher to write this info to the log file. For more information, see the topic about setting the logging level in "Administering the Query Log" on page 214.

To populate logical level counts automatically

1. In the Administration Tool, open a repository in online mode.
2. Right-click one or more business models and dimension objects, and choose Estimate Levels.
3. In the Check Out Objects dialog box, click Yes to check out the objects that appear in the list.

   NOTE: If you click No, the action terminates because you must check out items to run Estimate Levels.

   In the Oracle BI Administration Tool dialog box, a list of the dimension level counts and any errors or warning messages appear.

When you check in the objects, you can check the global consistency of your repository.

Using the Browse Dialog Box

The Browse dialog box appears in many situations in the Administration. You use it to navigate to and select an object.

The Browse dialog box is accessible from a number of dialog boxes that allow you to make a selection from among existing objects.

- The left pane of the Browse dialog box contains the following parts:
  - A tree listing all of the objects in the Presentation Layer, Business Model and Mapping Layer and the Physical Layer of a repository.
  - Tabs at the bottom of the left pane allow you to select a layer. Only the tabs for the layers that contain objects that can be manipulated in the dialog box from which you entered the Browse dialog box will appear.
- The right pane of the Browse dialog box contains the following parts:

- Query allows you to query objects in the repository by name and type. The Name field accepts an asterisk (*) as the wild card character, so you can query for partial matches.
- The Show Qualified Names check box allows you to identify to which parents an object belongs.
- View allows you to view properties of a selected object in read-only mode.

To query for an object in the Browse dialog box

1. Select the object type from the Type drop-down list.
2. Type the name of the object or a part of the name and the wild card character (*) in the Name field. Refer to the following examples:
   - To search for logical tables that have names beginning with the letter Q, select Logical Tables from the Type drop-down list, and then type Q* in the Name field.
   - To search for logical tables that have names ending with the letters dim, type *dim in the name field.
3. Click the Query button.

   Relevant objects appear in the query results list.

To select an object in the Browse dialog box

- Select the object you want to select, and then click Select.

  The Browse dialog box closes, and the object appears in the previous dialog box.

To synchronize an object in the query results list with the tree control list

1. Select an object in the Query list.
2. Click the Synchronize Contents icon.

3 Planning and Creating an Oracle BI Repository

This chapter contains the following topics:

- Roadmap for Planning and Setting Up an Oracle BI Repository on page 39
- Process of Oracle BI Repository Planning and Design on page 39
- Creating a New Oracle BI Repository File on page 54

Roadmap for Planning and Setting Up an Oracle BI Repository

The roadmap topics are discussed in several chapters in this book. The following are the topics that you will use to plan and set up an Oracle BI repository:

- Process of Oracle BI Repository Planning and Design on page 39
- Creating a New Oracle BI Repository File on page 54
- Chapter 4, “Creating and Administering the Physical Layer in an Oracle BI Repository”
- Chapter 5, “Creating and Administering the Business Model and Mapping Layer in an Oracle BI Repository”
- Chapter 6, “Creating and Maintaining the Presentation Layer in an Oracle BI Repository”
- Chapter 7, “Completing Setup and Managing Oracle BI Repository Files”

Process of Oracle BI Repository Planning and Design

This section contains the following topics:

- About Repository Planning and Design on page 39
- Planning Your Business Model on page 40
- Identifying the Database Content For The Business Model on page 45
- Guidelines For Designing a Repository on page 50

About Repository Planning and Design

This topic is part of the “Process of Oracle BI Repository Planning and Design” on page 39.

Before you create a repository or modify the repository that was shipped with Oracle Business Intelligence, you need to plan your business model and develop a design for the repository.

An Oracle BI repository has the following layers:

- Repository Physical layer. You create this layer (schema) first from the tables in your physical databases.

  CAUTION: Before you create the physical layer of your repository, you must thoroughly analyze and understand your business requirements, business data model, and your users so that you can set up the physical layer correctly. For more information, refer to "Planning Your Business Model" on page 40.

- Repository Business Model and Mapping layer. After you set up the Physical layer, you can create the Business Model and Mapping layer by dragging and dropping the Physical layer onto the Business Model and Mapping layer. This preserves the mapping that you established in the Physical layer.
- Repository Presentation layer. After you set up the Business Model and Mapping layer, you can create the Presentation layer by dragging and dropping the Business Model and Mapping layer onto the Presentation layer. This layer provides the objects that the end user will access through an application such as Oracle's Siebel Marketing.

Customization involves modifying the Physical, Business Model and Mapping, and Presentation layers. This is a sensitive task and should be done very carefully. It is recommended that you use the information in this chapter when designing and customizing your Oracle BI repository.

Planning Your Business Model

This topic is part of the "Process of Oracle BI Repository Planning and Design" on page 39.

Planning your business model is the first step in developing a usable data model for decision support. After you have followed the planning guidelines in this section, you can begin to create your repository. This section contains the following topics:

- Analyzing Your Business Model on page 40
- Identifying the Content of The Business Model on page 41

Analyzing Your Business Model

Your first task is to thoroughly understand your business model. You have to understand what business model you want to build before you can determine what the physical layer needs to have in it.

In a decision support environment, the objective of data modeling is to design a model that presents business information in a manner that parallels business analysts' understanding of the business structure. A successful model allows the query process to become a natural process by allowing analysts to structure queries in the same intuitive fashion as they would ask business questions. This model must be one that business analysts will inherently understand and that will answer meaningful questions correctly.

This requires breaking down your business into several components to answer the following questions:

- What kinds of business questions are analysts trying to answer?
- What are the measures required to understand business performance?
- What are all the dimensions under which the business operates?
- Are there hierarchical elements in each dimension and what are the parent-child relationships that define each hierarchy?

After you have answered these questions, you can identify and define the elements of your business model. This section will help you identify the components of your business.

Identifying the Content of The Business Model

To create a business model, the data needs to be mapped logically to a business model. The Oracle BI Server can use dimensional models for this purpose. This section discusses some of the components and variants of representative dimensional models.

Businesses are analyzed by relevant dimensional criteria, and the business model is developed from these relevant dimensions. These dimensional models form the basis of the valid business models to use with the Oracle BI Server. All dimensional models build on a star schema. That is, they model some measurable facts that are viewed in terms of various dimensional attributes.

After you analyze your business model, you need to identify the specific logical tables and hierarchies that you need to include in your business model. For more information about these objects, refer to Identifying the Content of The Business Model on page 41. The following sections are discussed in this section:

- Identifying Fact Tables on page 41
- Identifying Dimension Tables on page 43
- Identifying Bridge Tables on page 43
- Identifying Dimension Hierarchies on page 43
- About Star and Snowflake Models on page 44

Identifying Fact Tables

A fact table is a table with measures. Measures must be defined in a logical fact table. Measures, or facts, are typically calculated data such as dollar value or quantity sold, and they can be specified in terms of dimensions. For example, you might want to determine the sum of dollars for a given product in a given market over a given time period.

Each measure has its own aggregation rule such as SUM, AVG, MIN, or MAX. A business might want to compare values of a measure and need a calculation to express the comparison. Also, aggregation rules can be specific to particular dimensions. The Oracle BI Server allows you to define complex, dimension-specific aggregation rules.

The Oracle BI Server recognizes that when a table in the Business Model and Mapping layer of a repository has only many-to-one (N:1) logical joins pointing to it, it is typically a fact table.

For example, you might have two tables. Sales Transactions is a fact table while Stores is a Dimension table. One store will have many sales transactions and another store might have no transactions (closed for refurbishment) resulting in 1:n and 1:0 relationships, respectively. Some transactions do not happen in the stores but at a conference, resulting in a 0:n (zero to many) relationship.

NOTE: The fact table is at the end of a 0, 1:N (zero, one-to-many relationship) join.

Figure 2 on page 42 illustrates the many-to-one joins to a fact table in a Business Model Diagram. In the diagram, all joins have the crow's feet symbol (indicating the many side) pointing into the Fact-Pipeline table and no joins pointing out of it. For an example of this in a business model, open a repository in the Administration Tool, right-click a fact table, and select Business Model Diagram > Whole Diagram.

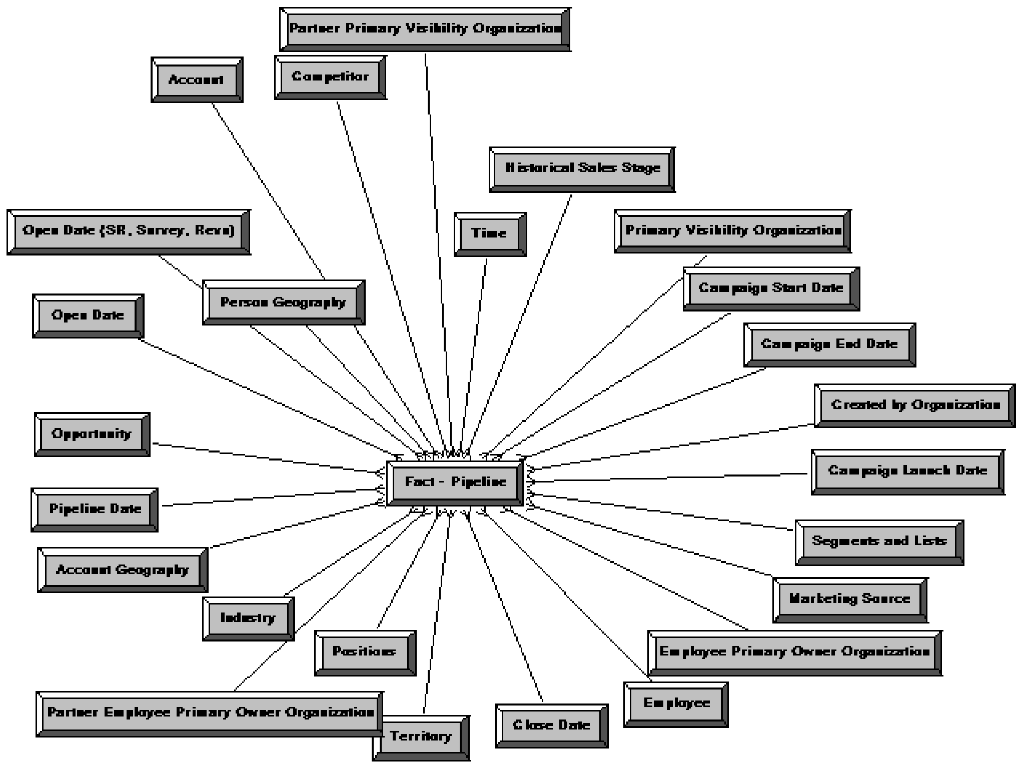


Figure 2. Diagram of Fact Table Joins

NOTE: A bridge table is an exception to this joins rule. For more information, refer to "Identifying Bridge Tables" on page 43.

Identifying Dimension Tables

A business uses facts to measure performance by well-established *dimensions*, for example, by time, product, and market. Every dimension has a set of descriptive attributes. Dimension tables contain attributes that describe business entities. For example, Customer Name, Region, Address, Country and so on. Dimension tables also contain primary keys that identify each member.

Dimension table attributes provide context to numeric data, such as being able to categorize Service Requests. Attributes stored in this dimension might include Service Request Owner, Area, Account, Priority, and so on.

The best method to identify dimensions and their attributes is to talk with the analysts in the organization who will use the data. The terminology they use and understand is important to capture.

Identifying Bridge Tables

When you need to model many-to-many relationships between dimension tables and fact tables, you can create a bridge table that resides between the fact and the dimension tables. A bridge table stores multiple records corresponding to that dimension. In the Administration Tool, General tab in the Logical Table dialog box contains a check box that you can select to specify that a table is a bridge table.

A bridge table allows you to resolve many-to-many data relationships. For example, you might have an Employees table and a Jobs table. An employee within an organization can have many jobs such as clerk, first-aid responder, and floor leader. Additionally, one job might be occupied by more than one employee such as there are 10 floor leaders in the organization. There is a many-to-many relationship between Employees and Jobs. However, a single table contains all the employees and another single table contains all the jobs within the organization.

To record and resolve this many to many relationship, you create a bridge (intermediate) table called Assignment. As a result, one employee may have many assignments and one job may also have many assignments. This assignments table records employee and job combination. The primary key is the combination of employee and job, and results in a unique record called an assignment. Therefore, Employee to Assignment is a 1:n relationship and Job to Assignment is also a 1:n relationship. The Assignment table acts as a bridging table between the Job and Employee tables, allowing the many to many relationship between Employees and Jobs to be resolved. For example, Mike and Gavin can be both first-aid responders and floor leaders.

Identifying Dimension Hierarchies

Understanding the hierarchies you need in your business is essential to provide the metadata that allows the Oracle BI Server to determine if a particular request can be answered by an aggregate that is already computed. For example, if month rolls up into year and an aggregate table exists at the month level, that table can be used to answer questions at the year level by adding up all of the month-level data for a year.

A hierarchy is a set of parent-child relationships between certain attributes within a dimension. These hierarchy attributes, called levels, roll up from child to parent; for example, months can roll up into a year. These rollups occur over the hierarchy elements and span natural business relationships.

Dimensions are categories of attributes by which the business is defined. Common dimensions are time periods, products, markets, customers, suppliers, promotion conditions, raw materials, manufacturing plants, transportation methods, media types, and time of day. Within a given dimension, there may be many attributes. For example, the time period dimension can contain the attributes day, week, month, quarter, and year. Exactly what attributes a dimension contains depends on the way the business is analyzed.

A dimensional hierarchy expresses the one-to-many relationships between attributes. Given a sample time dimension, consider the hierarchies it implies, as shown in Figure 3 on page 44.

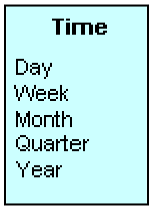


Figure 3. Sample Hierarchy

With this sample time dimension, days may aggregate, or roll up, into weeks. Months may roll up into quarters, and quarters into years. When one attribute rolls up to another, it implies a one-to-many relationship.

These hierarchy definitions have to be specific to the business model. One model may be set up where weeks roll up into a year, and another where they do not. For example, in a model where weeks roll up into a year, it is implied that each week has exactly one year associated with it; this may not hold true for calendar weeks, where the same week could span two years.

Some hierarchies might require multiple elements to roll up, as when the combination of month and year roll up into exactly one quarter. The Oracle BI Server allows you to define the hierarchy definitions for your particular business, however complex, assuring that analyses will conform to your business definitions.

You should identify as many natural hierarchies as possible. As with business questions, some hierarchies are obvious, but some are not and are only known by the end users who interact with particular aspects of the business. You should verify that you understand the hierarchies so you can define them properly using the Administration Tool.

About Star and Snowflake Models

Star and snowflake models follow the dimensional rules of one-to-many relationships. Star schemas have one-to-many relationships between the logical dimension tables and the logical fact table. Snowflake schemas have those same types of relationships, but also include one-to-many relationships between elements in the dimensions. For example, in Figure 4 on page 45 and Figure 5 on page 45, Sales Facts and Facts are fact tables and Markets, Periods, Products, Account Hierarchy, Account Region Hierarchy and Account are dimension tables. Any logical table that is not a fact table or a bridge table is a dimension table. For more information, see “Identifying Bridge Tables” on page 43.

Figure 4 on page 45 illustrates a star schema:

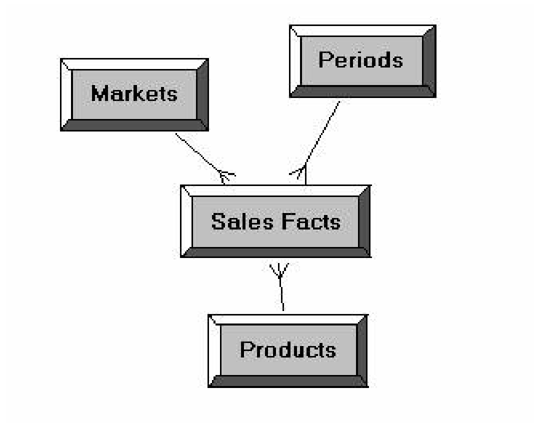


Figure 4. Diagram of a Star Schema

NOTE: It is recommended that you minimize the use of snowflake schemas.

In a snowflake schema, the server treats outer joins that are within a logical table source differently from other outer joins. Within a logical table source, the joins are always executed. When between logical tables, the joins are only performed when required. For more information about snowflake schemas, refer to "About Types of Physical Schemas" on page 46. Figure 5 on page 45 illustrates a logical snowflake schema.

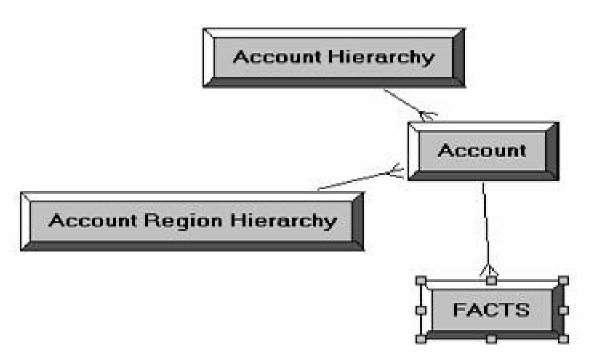


Figure 5. Diagram of a Snowflake Schema

Identifying the Database Content For The Business Model

This topic is part of the "Process of Oracle BI Repository Planning and Design" on page 39.

The Oracle BI Server provides an interface that allows you to map the Oracle BI repository to your underlying physical databases. Sometimes you can control the physical design of the underlying databases. However, sometimes the database already exists and you need to work with what is there. In either case, you need to understand the structure and content of your physical databases.

This section discusses the following topics:

- About Types of Physical Schemas on page 46
- About Primary Key-Foreign Key Relationships on page 48
- Identifying the Database Table Structure To Import on page 49

About Types of Physical Schemas

There are two types of physical schemas (models): entity-relationship (E-R) schemas and dimensional schemas. E-R schemas are designed to minimize data storage redundancy and optimize data updates. Dimensional schemas are designed to enhance understandability and to optimize query performance.

- **Entity-Relationship (E-R) Schemas.** The entity-relationship (E-R) schema is the classic, fully normalized relational schema used in many online transaction processing (OLTP) systems. The relationships between tables signify relationships between data, not necessarily business relationships.

  Typically, E-R schemas have many tables, sometimes hundreds or even thousands. There are many tables because the data has been carefully taken apart (normalized, in database terminology) with the primary goal of reducing data redundancy and bringing about fast update performance. E-R schemas are very efficient for OLTP databases. When E-R databases are queried, joins are usually predetermined and can be optimized. E-R databases are usually queried by applications that know exactly where to go and what to ask. These applications typically query small units of information at a time, such as a customer record, an order, or a shipment record.

  E-R schemas generally do not work well for queries that perform historical analysis due to two major problems: poor performance and difficulty in posing the question in SQL.

  - Performance problems persist with historical E-R queries because the queries require the database to put the data back together again; this is a slow, complex process. Furthermore, because the cost-based optimizers in database management systems are not designed for the level of complexity in such a query, they can generate query plans that result in poor performance.
  - A Database Analyst (DBA) who is very familiar with the data might be able to write a SQL query against an E-R schema that can theoretically answer a business question, but such detailed knowledge of the data is generally beyond the scope of the end-user business analyst. Even when the SQL is crafted properly, there is often an unacceptably high response time in returning results to the user.

- **Dimensional Schemas.** A dimensional schema is a denormalized schema that follows the business model. Dimensional schemas contain dimension tables and fact tables. Dimension tables contain attributes of the business, and fact tables contain individual records with a few facts and foreign keys to each of the dimension tables.

  Dimensional schemas are used for business analysis and have two major advantages over E-R schemas for decision support:

  - Better query performance
  - Easier to understand

  Dimensional schemas are not nearly as efficient as E-R schemas for updating discrete records, but they work well for queries that analyze the business across multiple dimensions.

  The following are two types of dimensional schemas:

  - **Star schema.** A star schema is a dimensional schema with a single fact table that has foreign key relationships with several dimension tables.
    - The dimension tables mirror the business model and contain columns with descriptive attributes, such as Product, Size, and Color in the sample Products dimension. Dimension tables also have a key column (or columns) that uniquely identifies each row in the table.
    - The fact table has a multipart primary key, often made up of the foreign key references to the dimension tables. The fact table also contains all the measures, or facts, used to measure business performance. The lowest level of detail stored is the granularity of the fact table. Information at higher levels of aggregation is either calculated from the detail level records or precomputed and stored in separate aggregate fact tables, resulting in a multiple-star schema. For a discussion of aggregate fact tables, read "Identifying Fact Tables" on page 41.

- **Snowflake schema.** A snowflake schema is a dimensional schema where one or more of the dimensions are normalized to some extent.

  NOTE: It is recommended that you minimize the use of snowflake schemas.

  The difference between the type of normalization in a snowflake schema and in an E-R schema is that the snowflake normalization is based on business hierarchy attributes. The tables snowflaked off the dimensions have parent-child relationships with each other that mimic the dimensional hierarchies. In a snowflake schema, multiple logical tables are considered a single logical table.

  For example, Figure 6 on page 48 contains a snowflake schema showing that Product Line > Products is a branch of the snowflake.

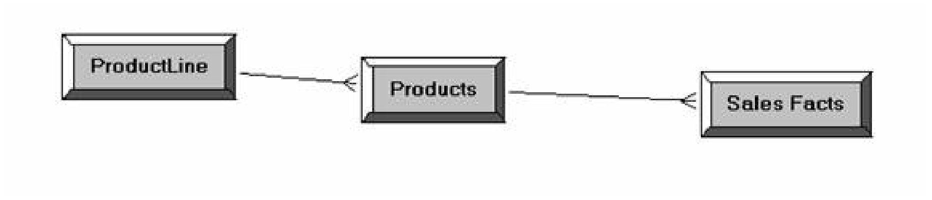


Figure 6. Diagram of a Snowflake Schema

One dimension hierarchy should be created for the Products branch. The following is a list of the minimal levels that should be created for this hierarchy:

- Grand Total Level
- Detail Level of the dimension that is not joined to the fact table. In this case, it is ProductLine.
- Detail Level of the dimension that is joined to the fact table.

Figure 7 on page 48 shows the hierarchy that should be created.

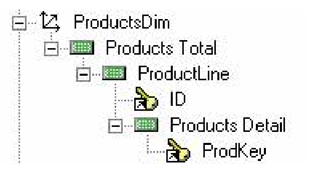


Figure 7. Hierarchy for the Products Branch in the Business Model and Mapping Layer

About Primary Key-Foreign Key Relationships

To fully understand the structure and content of your physical databases, it is important to become familiar with the concepts behind primary key-foreign key relationships.

A primary key-foreign key relationship defines a one-to-many relationship between two tables in a relational database. A foreign key is a column or a set of columns in one table that references the primary key columns in another table. The primary key is defined as a column (or set of columns) where each value is unique and identifies a single row of the table.

Consider Figure 8 on page 49, where the upper table is a fact table named Sales and the lower table is a dimension table named Date. The Sales fact table contains one row for every sales transaction, and the Date dimension table contains one row for every date the database will potentially cover.

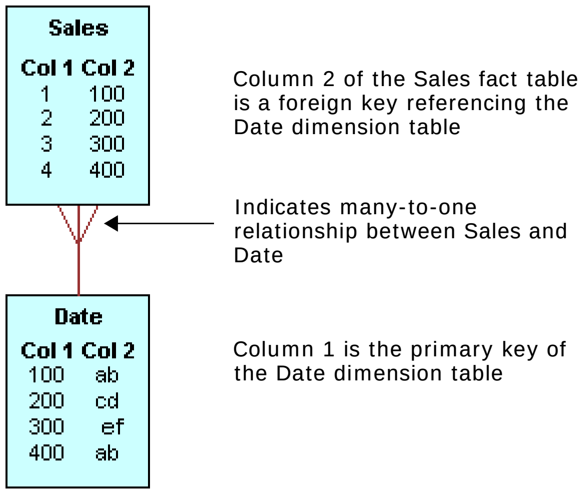

Sales
Col 1 Col 2
1 100
2 200
3 300
4 400
Column 2 of the Sales fact table is a foreign key referencing the Date dimension table
Indicates many-to-one relationship between Sales and Date
Date
Col 1 Col 2
100 ab
200 cd
300 ef
400 ab
Column 1 is the primary key of the Date dimension table

Figure 8. Primary Key-Foreign Key Sample

Because of this primary key-foreign key relationship, you can join the Sales and Date tables to combine the other attributes of the Date table with the records in the Sales table. For example, if an analyst asks for the total sales, the day of the week, the product name, and the store in which the product was sold, the information is compiled by joining the sales, date, product, and store tables through the primary key-foreign key relationships between the Sales table and the various dimension tables.

Identifying the Database Table Structure To Import

The Administration Tool provides an interface to map logical tables to the underlying physical tables in the database. Before you can map the tables, you need to identify the contents of the physical database as it relates to your business model. To do this correctly, you need to identify the following contents of the physical database:

- Identify the contents of each table
- Identify the detail level for each table
- Identify the table definition for each aggregate table. This allows you to set up aggregate navigation. The following detail is required by the Oracle BI Server:
  - The columns by which the table is grouped (the aggregation level)
  - The type of aggregation (SUM, AVG, MIN, MAX, or COUNT)

  For information on how to set up aggregate navigation in a repository, refer to "Setting Up Fragmentation Content in an Oracle BI Repository for Aggregate Navigation" on page 201.
- Identify the contents of each column

- Identify how each measure is calculated
- Identify the joins defined in the database

To acquire this information about the data, you could refer to any available documentation that describes the data elements created when the database was implemented, or you might need to spend some time with the DBA for each database to get this information. To fully understand all the data elements, you might also need to ask people in the organization who are users of the data, owners of the data, or the application developers of applications that create the data.

Guidelines For Designing a Repository

This topic is part of the "Process of Oracle BI Repository Planning and Design" on page 39.

After analyzing your business model needs and identifying the database content your business needs, you can complete your repository design. This section contains some design best practices that will help you implement a more efficient repository.

Typically, you should not make performance tuning changes until you have imported and tested your databases. This would occur during the final steps in completing the setup of your repository. For more information about these final steps, refer to "Completing Setup and Managing Oracle BI Repository Files" on page 151.

The following topics are discussed in this section:

- General Tips For Working on the Repository on page 50
- Design Tips For the Physical Layer (Schema) on page 51
- Design Tips for the Business Model and Mapping Layer on page 52
- Design Tips For the Presentation Layer on page 53

General Tips For Working on the Repository

The Oracle BI Server stores metadata in repositories. The Oracle BI Administrator uses the graphical user interface (GUI) of the Administration Tool software to create and maintain repositories. An Oracle BI repository consists of three layers. Each layer appears in a separate pane in the Administration Tool user interface and has a tree structure (similar to the Windows Explorer). These layers are not visible to the end user.

Most windows and dialog boxes have online help containing information to help you complete a task. To access a help topic, click the help button in a dialog box or select Help from some menus.

The following list contains some recommended design strategies for the Oracle BI repository structure:

- Perform the majority of repository editing in offline mode to make your results more consistent. In some environments, this might save time during the development effort.

  If you work in Online mode, save backups of the online repository before and after every completed unit of work. If needed, use Copy As on the File menu to make an offline copy containing the changes.

- For design independence, import databases for each business model separately. For example, you might have one database for each application. Some tables may exist in more than one database and you can customize each table for a different business model. This eliminates the need for cross-database joins.
- After importing a database, test it before importing another. Make sure the metadata is generating the correct record set before performing performance tuning activities.
- Use the Physical diagrams in the Administration Tool to verify sources and joins.

Design Tips For the Physical Layer (Schema)

The Physical layer contains information about the physical data sources. The most common way to create the schema in the Physical layer is by importing metadata from databases and other data sources. If you import metadata, many of the properties are configured automatically based on the information gathered during the import process. You can also define other attributes of the physical data source, such as join relationships, that might not exist in the data source metadata.

There can be one or more data sources in the Physical layer, including databases and XML documents. For more information about supported databases, refer to *System Requirements and Supported Platforms*.

For each data source, there is at least one corresponding connection pool. The connection pool contains data source name (DSN) information used to connect to a data source, the number of connections allowed, timeout information, and other connectivity-related administrative details. For more information, refer to "Setting Up Connection Pools" on page 65 and Chapter 4, "Creating and Administering the Physical Layer in an Oracle BI Repository."

The following is a list of tips to use when designing the Physical layer:

- Before importing metadata from a data warehouse into the Physical layer of your repository, eliminate all outer joins in the data warehouse. Use Extract, Transform, and Load (ETL) lookups to find them. Eliminating outer joins results in a more consistent record set, a simplified business model, and improved performance.
- To reduce problems that might be caused by importing physical joins, import the physical data without foreign keys and then create the keys as needed.
- You will probably import some tables into the Physical layer that you might not use right away but that you do not want to delete. One way to identify tables that you do want to use right away in the Business Model and Mapping layer, is to assign aliases to physical tables before mapping them to the business model layer.

  NOTE: To have the name of a table to which you assigned an alias appear, make sure you turn on the following option in the Tools > Options > General menu path: Display original names for alias in diagrams.
- Use a naming standard in the Physical Layer that identifies the logical components followed by the physical table name as shown in the following example:

  Created By Employee (W_PERSON_D)

  NOTE: The Physical layer object name is included in the Physical SQL. By using qualified Physical Layer names, debugging is made easier.

- An *opaque view* (a physical layer table that consists of a Select statement) should be used only if there is no other solution. Ideally, a physical table should be created, or alternatively a materialized view. A traditional database view is not needed because it is identical to the opaque view. For more information, refer to "Deploying Opaque Views" on page 103.

Design Tips for the Business Model and Mapping Layer

The Business Model and Mapping layer organizes information by business model. Each business model contains logical tables. Logical tables are composed of logical columns. Logical tables have relationships to each other expressed by logical joins. The relationship between logical columns can be hierarchical, as defined in business model hierarchies. Logical tables map to the source data in the Physical layer. The mapping can include complex transformations and formulas.

The Business Model and Mapping layer defines the meaning and content of each physical source in business model terms. The Oracle BI Server uses these definitions to assign the appropriate sources for each data request.

You can change the names of physical objects independently from corresponding business model object names and properties, and vice versa. Any changes in the underlying physical databases or the mappings between the business model tables and physical tables might not change the view in the end-user applications that view and query the repository.

The logical schema defined in each business model needs to contain at least two logical tables. Relationships need to be defined between all the logical tables. For more information about business model schemas, refer to "Process of Oracle BI Repository Planning and Design" on page 39. For more information about setting up the Business Model and Mapping layer, refer to Chapter 5, "Creating and Administering the Business Model and Mapping Layer in an Oracle BI Repository."

The following is a list of tips to use when designing the Business Model and Mapping layer:

- Create the business model with one-to-many complex joins between logical dimension tables and the fact tables wherever possible. The business model should ideally resemble a simple star schema in which each fact table is linked directly to its dimensions.

  NOTE: If you use the snowflake model, you might have more flexibility (for example, the ability to use outer joins) but it may create more columns in the presentation layer. However, it is recommended that you minimize the use of snowflake schemas.

- Outer joins should be avoided in the reporting SQL. They can be eliminated in the ETL via a variety of techniques, but by doing so, not only can an additional table be removed from the report SQL, but the performance will also improve.
- Combine all similar dimensional attributes into one logical dimension table. Where needed, include data from other dimension tables into the main dimension source using aliases in the Physical layer tables. This should occur during the ETL process for optimal performance.
- Every logical dimension table should have a dimensional hierarchy associated with it. Make sure that all appropriate fact sources link to the proper level in the hierarchy using aggregation content. You set up aggregation content in the Content tab of the Logical Table Source properties window.
- Aggregate sources should be created as a separate Source in a logical table. For fact aggregates, use the Content tab of the Logical Table Source properties window to assign the correct logical level to each dimension.

- It is recommended that you use table aliases frequently in the Business Model layer to eliminate extraneous joins, including the following:
  - Eliminate all physical joins that cross dimensions (inter-dimensional circular joins) by using aliases.
  - Eliminate all circular joins (intra-dimensional circular joins) in a logical table source in the Physical Model by creating physical table aliases.
- Renaming an element in the Business Model and Mapping layer will automatically create an alias.
- To prevent problems with aggregate navigation, make sure that each logical level of a dimension hierarchy contains the correct value in the field named Number of elements at this level. Fact sources are selected on a combination of the fields selected as well as the levels in the dimensions to which they link. By adjusting these values, you can alter the fact source selected by Oracle Business Intelligence.
- Outer joins in logical table sources are always included in a query, even if the table source is not used. If possible, create one logical table source without the outer join and another with the outer join. Order the logical table source with the outer join after the non-outer join so that it will be used only when necessary.

Design Tips For the Presentation Layer

You set up the user view of a business model in the Presentation layer. The names of folders and columns in the Presentation layer appear in localized language translations. The Presentation layer is the appropriate layer in which to set user permissions. In this layer, you can do the following:

- You can show fewer columns than exist in the Business Model and Mapping layer. For example, you can exclude the key columns because they have no business meaning.
- You can organize columns using a different structure from the table structure in the Business Model and Mapping layer.
- You can display column names that are different from the column names in the Business Model and Mapping layer.
- You can set permissions to grant or deny users access to individual catalogs, tables, and columns.
- You can export logical keys to ODBC-based query and reporting tools.

For more information about setting up the Presentation layer, refer to Chapter 6, “Creating and Maintaining the Presentation Layer in an Oracle BI Repository.”

The following is a list of tips to use when designing the Presentation layer:

- **Column aliases.** It is recommended that you do not use aliases for Presentation layer columns.

  NOTE: Renaming an element in the Presentation layer will automatically create an alias. This prevents reports that reference the original element from failing.

- **Single table model.** For the greatest simplicity, you could construct a subject area that consists of a single table. To create a single table model, you first create a logical dimensional model, and then present it as a single table schema in the Administration Tool's Presentation layer. The logical dimensional model is required to set up the necessary metadata for the Oracle BI Server to navigate to the proper physical tables. For information about the Presentation layer, refer to "Creating and Maintaining the Presentation Layer in an Oracle BI Repository" on page 143.

Creating a New Oracle BI Repository File

Now that you have completed your planning and design tasks, you can create a repository. The first step in creating a repository is creating a repository file. Each time you save the repository, a dialog box asks if you want to check global consistency. You have the following options:

- **Yes.** Checks global consistency and then saves the repository file.
- **No.** Does not check global consistency and then saves the repository file.
- **Cancel.** Does not check global consistency and does not save the repository file.

For more information, see "Checking the Consistency of a Repository or a Business Model" on page 26.

NOTE: In offline editing, remember to save your repository from time to time. You can save a repository in offline mode even though the business models may be inconsistent.

To create a new repository file

1. From the Administration Tool menu, select File > New or click the New button in the toolbar.
2. Type a name for the repository.

   A RPD file extension will automatically be added if you do not explicitly specify it. The default location for all repositories is in the Repository subdirectory, located in the Oracle BI software installation folder (Oracle BI\server\Repository).

   The new repository is empty. The remaining steps in the repository setup process, as outlined in "Roadmap for Planning and Setting Up an Oracle BI Repository" on page 39, describe the steps you should follow to set up and populate repositories.

After Creating a Repository File

After you create a repository file in the Administration Tool, you can import tables into the Physical layer.

NOTE: The typical order is to create the Physical layer first, the Business Model and Mapping layer next, and the Presentation layer last. However, you can work on each layer at any stage in creating a repository. You can set up security when you are ready to begin testing the repository.

For more information, refer to "Creating and Administering the Physical Layer in an Oracle BI Repository" on page 55.

4 Creating and Administering the Physical Layer in an Oracle BI Repository

This section is part of the roadmap for planning and setting up a repository. For more information, refer to "Planning and Creating an Oracle BI Repository" on page 39.

Before you create the physical layer of a repository, you need to plan, design, and then create a repository file. For more information, refer to "Process of Oracle BI Repository Planning and Design" on page 39

The Physical layer of the Administration Tool defines the data sources to which the Oracle BI Server submits queries and the relationships between physical databases and other data sources that are used to process multiple data source queries. (Your licensing options determine whether you can add databases and connection pools to the physical layer.)

The first step in creating the physical layer is to create the schema. You can import the physical schema for supported data source types. You can import schemas or portions of schemas from existing data sources. Additionally, you can create the physical layer manually but it is a labor-intensive and error-prone activity.

CAUTION: It is strongly recommended that you import physical schemas.

This section provides the following topics to help you use the Administration Tool to create the physical layer of a repository:

- Process of Creating the Physical Layer from Relational Data Sources on page 56
- Process of Creating the Physical Layer from Multidimensional Data Sources on page 58
- Setting Up Database Objects on page 60
- Setting Up Connection Pools on page 65
- About Physical Tables on page 81
- Creating and Setting Up Physical Tables on page 83
- Creating Physical Layer Folders on page 96
- About Physical Joins on page 99
- Defining Physical Foreign Keys and Joins on page 100
- Deploying Opaque Views on page 103
- Using Database Hints on page 106

Process of Creating the Physical Layer from Relational Data Sources

Importing the physical schema saves you time and effort by importing the structure for the physical layer. You can import schema for supported data sources. If you do not import the schema, you must create each table, primary key, foreign key, and any other objects against which you want to generate queries. Data from these sources can be displayed on an Oracle BI Interactive Dashboard.

The following is a list of tips to help you when importing a physical schema:

- When you import schema for most databases, the default is to import tables, primary keys, and foreign keys.

  NOTE: It is recommended that you not import foreign keys from a database because the process is lengthy when importing a large number of tables.

- When you import physical tables, be careful to limit the import to only those tables that contain data that are likely to be used in the business models you create. You can use a filter (table mask) to limit the number of tables that appear in the import list. This makes it easier to locate and select the tables that you want to import.
- You can also import database views, aliases, synonyms, and system tables. Import these objects only if you want the Oracle BI Server to generate queries against them.
- Importing large numbers of extraneous tables and other objects adds unnecessary complexity and increases the size of the repository.

NOTE: Data Warehouse (OLAP) objects are stored in and accessed through Oracle Analytic Workspaces (AWs). For information about how to extract these objects, refer to the Oracle Reference 10g Release 2 documents from the oracle.com Web site.

To create the physical layer by importing the schema, complete the following tasks in the sequence shown:

- Importing a Physical Schema from Relational Data Sources on page 56
- Setting Up Database Objects on page 60
- Setting Up Connection Pools on page 65
- About Physical Tables on page 81
- Creating and Setting Up Physical Tables on page 83

Importing a Physical Schema from Relational Data Sources

This topic is part of the "Process of Creating the Physical Layer from Relational Data Sources" on page 56.

You can import physical schema for supported data source types. When you can use importing to create the physical schema, you need to select one of the following import options and the appropriate connection type:

- **From a database.** Available in Offline and Online modes. Use this option when you have all database connections set up on your machine. You can use the following connection types with the Import option:
  - Most physical schema imports are performed using an ODBC connection type.
  - Native database gateways for schema import are supported for only DB2 (using DB2 CLI or DB2 CLI Unicode) and XML connection types. For more information about importing XML data using the Oracle BI Server XML gateway, refer to "Oracle BI Server XML Gateway Example" on page 338.
  - You can use a table mask to limit (filter) the list of tables that appear in the Import dialog box. When you have one or more specific tables that you want to import, using a table mask helps locate the tables.
- **Through the** Oracle BI Server**.** Available in Online mode. Use this option when you want to use the Oracle BI Server connections to import schema. This option allows Oracle BI Administrators to use the Data Source Name (DSN) of the Oracle BI Server machine to import physical schema information. Connectivity software and duplicate DSNs do not have to reside on the Oracle BI Administrator's machine and the Oracle BI Server machine. You can use the following connection types with the Import option:
  - Available connection types are ODBC, DB2 CLI, DB2 CLI (Unicode), and XML.
  - When it is running on a UNIX platform, the Oracle BI Server does not support importing schema using an ODBC connection type.
- **From a repository.** Available in Offline mode. By default, the Import from repository option on the File menu is disabled and this option will not be supported in the future. If you must import metadata using this method, see "Using the Options Dialog Box—General Tab" on page 29.

  NOTE: It is recommended that you create projects in the repository that contain the objects that you wish to import, and then use repository merge to bring the projects into your current repository. For more information, see "About Extracting Metadata Subsets Into Projects" on page 167.
- **From XMLA.** Used to import multidimensional data sources. For more information, refer to "Importing a Physical Schema from Multidimensional Data Sources" on page 59.

To import a physical schema from an ODBC connection type

1. In the Administration Tool, select File > Import, and then select the source type of your physical schema from the following options:
   - from Database
   - through Server
   - from Repository
2. In the Select Data Source dialog box, perform the following steps:

   a. From the Connection Type drop-down list, select the correct type.

      NOTE: It is recommended that you use ODBC 3.5 or DB2 CLI Unicode for importing schema with International characters such as Japanese table and column names.

b. In the DSN list, select a data source from which to import the schema.

   When you import through the Oracle BI Server, the DSN entries are on the Oracle BI Server, not on the local machine.

c. To import from Customer Relationship Management (CRM) tables, select the Read from CRM metadata tables check box.

d. Click OK.

3. If a logon screen appears, type a valid user ID and password for the data source, and then click OK.

   The Oracle BI Administrator must supply the user name and password for the data source. If you are performing an import on the Oracle BI Administrator's machine, the user name will be obtained from the machine’s DSN configuration.

4. In the Import dialog box, select the check boxes for the types of objects that you want to import. For example, Tables, Keys, and Foreign Keys.

   Some objects check boxes are automatically selected. These default selections depend on your data source.

5. To use a table mask, type a table mask such as F%, and then click the highest level database folder.

   Tables that meet the filter criteria appear.

6. Select the tables and columns you want to import.

   NOTE: If you select just a table without identifying the columns, all the columns in the table will be imported.

7. After you select all the objects you want to import, click Import or drag and drop them into the Physical layer.

Process of Creating the Physical Layer from Multidimensional Data Sources

This section describes how you use the Administration Tool to add a multidimensional data source to the physical layer of a repository. The ability to use multidimensional data sources allows the Oracle BI Server to connect to sources such as Microsoft Analysis Services and SAP/BW (SAP/Business Warehouse) to extract data. Data from these sources can be displayed on an Oracle BI Interactive Dashboard.

The Oracle BI Server connects to the multidimensional source using XML for Analysis (XMLA) standards protocol. This requires that the target data source have a fully functional Web services interface available. The standard dictates the various protocols that the Oracle BI Server can use to connect to the target and query data.

Importing data from a multidimensional source creates the metadata of the data source in the Oracle BI repository. The primary differences between setting up multidimensional data sources and relational data sources are in the physical layer. The setup in the business model and presentation layers for multidimensional data sources and relational data sources is almost identical.

To create a physical layer for a multidimensional data source, complete the following tasks in the sequence shown:

- Importing a Physical Schema from Multidimensional Data Sources on page 59
- Setting Up Database Objects on page 60
- Setting Up Connection Pools on page 65
- About Physical Tables on page 81
- Creating and Setting Up Physical Tables on page 83

Importing a Physical Schema from Multidimensional Data Sources

This topic is part of "Process of Creating the Physical Layer from Multidimensional Data Sources" on page 58.

CAUTION: Manually creating a physical schema from a multidimensional data source is labor-intensive and error prone. Therefore, it is strongly recommended that you import it.

The Oracle BI Server uses XMLA standards to import data from a multidimensional data source into the Oracle BI repository. During the import process, each cube in a multidimensional data source is created as a single physical cube table. The Oracle BI Server imports the cube, including its metrics, dimensions and hierarchies. After importing the cubes, you need to make sure that the physical cube columns have the correct aggregation rule and that the default member type ALL is correctly imported for a hierarchy. For more information, refer to "Adding a Hierarchy to a Physical Cube Table" on page 91.

The following list includes information that you should consider before importing data:

- Microsoft Analysis Services and SAP/BW are the only supported XMLA-compliant data sources currently available.
- The Oracle BI Server only imports the dimensions and hierarchies that are supported by Oracle BI. Therefore, if a cube has a ragged hierarchy or a parent-child hierarchy, it will not be imported. Additionally, measure hierarchies are not imported or supported.
- It is recommended that you remove hierarchies and columns from the physical layer if they will not be used in the business model. This eliminates maintaining unnecessary objects in the Administration Tool and might result in better performance.

To import a physical schema from multidimensional data sources

1. From your data source administrator, obtain the URL connection string and user name and password for the data source.
2. In the Administration Tool, select File > Import from XMLA.
3. In the Import From XMLA dialog box, complete the following fields:

   a. In the URL field, type the URL for the Web service that acts as the XMLA provider.

      Use the URL connection string obtained from your data source administrator.

   b. In the username and password fields, type a valid username and password for the data source.

      Use the user name and password that you obtained from your data source administrator.

4. Click OK.

   The Administration Tool connects to the destination source to obtain a list of the available data catalogs (databases).

5. In the Browse dialog box, expand the data source and catalog (database), if necessary, and select the catalogs (databases) or cubes to import.

   The Import button is unavailable until you select an object that can be downloaded.

6. After you select all the objects you want to import, click Import.

   If some objects could not be imported, a list of warning messages appears in the Oracle BI Administration Tool dialog box. For example:

   ```
   Hierarchy "[Measures]" from data source "SDCHS40I023", catalog
   "XMLA_council_interop", cube "BeverageSales" is of measure type and not imported.
   ```

7. In the Oracle BI Administration Tool dialog box, you can perform the following actions:

   - To search for specific terms, click Find and then Find Again.
   - To copy the contents of the window so that you can paste the messages in another file, click Copy.

8. When finished importing, in each dialog box, click OK and then Close.

Setting Up Database Objects

This topic is part of the "Process of Creating the Physical Layer from Relational Data Sources" on page 56 and the "Process of Creating the Physical Layer from Multidimensional Data Sources" on page 58.

Importing a schema automatically creates a database object for the schema but you need to set up the database properties.

For more information about supported databases, refer to *System Requirements and Supported Platforms*.

To create or edit database objects in the physical layer, perform the following tasks:

- About Database Types in the Physical Layer on page 61
- Creating a Database Object Manually in the Physical Layer on page 61
- Specifying SQL Features Supported by a Database on page 64

About Database Types in the Physical Layer

If you import the physical schema into the Physical layer, the database type is usually assigned automatically. The following list contains additional information about automatic assignment of database types:

- **Relational data sources.** During the import process, some ODBC drivers provide the Oracle BI Server with the database type. However, if the server cannot determine the database type, an approximate ODBC type is assigned to the database object. Replace the ODBC type with the closest matching entry from the database type drop-down menu.
- **Multidimensional data sources.** Microsoft Analysis Services and SAP/BW are the only supported XMLA-compliant data sources currently available. When you import a multidimensional data source, you need to select the appropriate database type and version.

Creating a Database Object Manually in the Physical Layer

If you create a database object manually, you need to manually set up all database elements such as connection pool, tables, and columns.

NOTE: For multidimensional data sources, if you create the physical schema in the physical layer of the repository, you need to create one database in the physical layer for each cube, or set of cubes, that belong to the same catalog (database) in the data source. A physical database can have more than one cube. However, all of these cubes must be in the same catalog in the data source. You specify a catalog in the Connection Pool dialog box.

CAUTION: It is strongly recommended that you import your physical schema.

To create a database object

1. In the Physical layer of the Administration Tool, right-click and choose New Database.

   Make sure that no object is selected when you right-click.

2 In the Database dialog box, in the General tab, complete the fields using Table 5 on page 62 as a guide.

Table 5. Fields General Tab of the Database Dialog Box

Field	Description
Allow direct database requests by default (check box)	When checked, allows everyone to execute physical queries. The Oracle BI Server will send unprocessed, user-entered, physical SQL directly to an underlying database. The returned results set can be rendered in Oracle BI Presentation Services, and then charted, rendered in a dashboard, and treated as an Oracle BI request. If you want most but not all users to be able to execute physical queries, check this option and then limit queries for specific users or groups. For more information about limiting queries, refer to "Managing Query Execution Privileges" on page 330. **CAUTION:** If configured incorrectly, this can expose sensitive data to an unintended audience. For more information, refer to "Recommendations for Allowing Direct Database Requests by Default" on page 63. For more information about executing physical SQL, refer to *Oracle Business Intelligence Answers, Delivers, and Interactive Dashboards User Guide*.
Allow populate queries by default (check box)	When checked, allows everyone to execute POPULATE SQL. If you want most but not all users to be able to execute POPULATE SQL, check this option and then limit queries for specific users or groups. For more information about limiting queries, refer to "Managing Query Execution Privileges" on page 330.
Data Source Definition CRM Metadata Tables	Not available for multidimensional data sources. When selected, the import utility will look for the table definition in Oracle's Siebel CRM specific tables. For Siebel CRM tables, the import reads the Siebel metadata dictionary to define the definition of physical tables and columns (it does not read the database data dictionary). This is for legacy Siebel Systems sources only.
Data Source Definition Database	The database type for your database. For more information about using the Features tab to examine the SQL features supported by the specified database type, refer to "Specifying SQL Features Supported by a Database" on page 64.

Table 5. Fields General Tab of the Database Dialog Box

Field	Description
Data Source Definition Virtual Private Database	Identifies the physical database source as a virtual private database (VPD). When a VPD is used, returned data results are contingent on the users authorization credentials. Therefore, it is important to identify these sources. These data results affect the validity of the query result set that will be used with Caching. For more information, see "Query Caching in the Oracle BI Server" on page 229. **NOTE:** If you select this check box, you also should select the Security Sensitive check box in the Session Variable dialog box. For more information, see "Creating Session Variables" on page 289.
Persist Connection Pool	To use a persistent connection pool, you must set up a temporary table first. For more information, refer to "Setting Up the Persist Connection Pool Property" on page 79.

Recommendations for Allowing Direct Database Requests by Default

This property allows all users to execute physical queries. If configured incorrectly, it can expose sensitive data to an unintended audience. Use the following recommended guidelines when setting this database property:

- The Oracle BI Server should be configured to accept connection requests only from a machine on which the Oracle BI Server, Oracle BI Presentation Services, or Oracle BI Scheduler Server are running. This restriction should be established at the TCP/IP level using the Oracle BI Presentation Services IP address. This will allow only a TCP/IP connection from the IP Address of the Oracle BI Presentation Services.
- To prevent users from running nQCmd.exe (a file that executes SQL script) by logging in remotely to this machine, you should disallow access by the following to the machine on which you installed the Oracle BI Presentation Services:
  - TELNET
  - Remote shells
  - Remote desktops
  - Teleconferencing software (such as Windows NetMeeting)

  If necessary, you might want to make an exception for users with Administrator permissions.
- Only users with Administrator permissions should be allowed to perform the following tasks:
  - TELNET into the Oracle BI Server and Oracle BI Presentation Services machines to perform tasks such as running nQCmd.exe for cache seeding. For more information, see "Executing the SQL Script File to Create and Delete Aggregates" on page 247.
- Access to the advanced SQL page of Answers to create reports. For more information, refer to *Oracle Business Intelligence Presentation Services Administration Guide*.

- Setup group/user-based permissions on the Oracle BI Presentation Services to control access to editing (preconfigured to allow access by the Oracle BI Presentation Services Administrators) and executing (preconfigured to not allow access by anyone) direct database requests. For more information, refer to *Oracle Business Intelligence Presentation Services Administration Guide*.

Specifying SQL Features Supported by a Database

When you import the schema or specify a database type in the General tab of the Database dialog box, the Feature table is automatically populated with default values appropriate for the database type. These are the SQL features that the Oracle BI Server uses with this data source.

You can tailor the default query features for a database. For example, if a data source supports left outer join queries but you want to prohibit the Oracle BI Server from sending such queries to a particular database, you can change this default setting in the Feature table.

CAUTION: Be very careful when modifying the Feature table. If you enable SQL features that the database does not support, your query may return errors and unexpected results. If you disable supported SQL features, the server could issue less efficient SQL to the database.

To specify SQL features supported by a database

1. In the Physical layer of the Administration Tool, double-click the database.
2. In the Database dialog box, click the Features tab.
3. In the Features tab, use the information in Table 6 on page 64 to help you specify properties for each SQL features.

Table 6. SQL Feature Tab Descriptions

Field or Button	Description
Ask DBMS	A button that is used only if you are installing and querying a database that has no Features table. It allows you to query this database for Feature table entries. For more information, refer to "Changing Feature Table Entries Using Ask DBMS" on page 65.
Default	A check box that identifies default SQL features. Default SQL features that are supported by the database type of the data source are automatically selected.
Find	A button that allows you to type a string to help you locate a feature in the list.
Find Again	A button that becomes available after you click Find. Allows you to perform multiple searches for the same string.
Revert to Defaults	A button that restores the default values.
Value	A check box that allows you to specify additional SQL features. Select to enable a query type or clear to disable a query type. It is strongly recommended that you do not disable the default SQL features.

Changing Feature Table Entries Using Ask DBMS

You should use the Ask DBMS button only if installing and querying a database that has no Features table.

NOTE: The Ask DBMS button is not available if you are using an XML or a multidimensional data source.

The Ask DBMS button on the Features tab of the Database dialog box allows you to query a database for the SQL features it supports. You can change the entries that appear in the Feature table based on your query results.

CAUTION: Be very careful when using Ask DBMS. The results of the features query are not always an accurate reflection of the SQL features actually supported by the data source. You should only use this functionality with the assistance of Oracle Technical Support.

Setting Up Connection Pools

This topic is part of the "Process of Creating the Physical Layer from Relational Data Sources" on page 56 and the "Process of Creating the Physical Layer from Multidimensional Data Sources" on page 58.

The *connection pool* is an object in the Physical layer that describes access to the data source. It contains information about the connection between the Oracle BI Server and that data source.

The Physical layer in the Administration Tool contains at least one connection pool for each database. When you create the physical layer by importing a schema for a data source, the connection pool is created automatically. You can configure multiple connection pools for a database. Connection pools allow multiple concurrent data source requests (queries) to share a single database connection, reducing the overhead of connecting to a database.

NOTE: It is recommended that you create a dedicated connection pool for initialization blocks. For additional information, refer to "Creating or Changing Connection Pools" on page 67.

For each connection pool, you must specify the maximum number of concurrent connections allowed. After this limit is reached, the Oracle BI Server routes all other connection requests to another connection pool or, if no other connection pools exist, the connection request waits until a connection becomes available.

Increasing the allowed number of concurrent connections can potentially increase the load on the underlying database accessed by the connection pool. Test and consult with your DBA to make sure the data source can handle the number of connections specified in the connection pool. Also, if the data sources have a charge back system based on the number of connections, you might want to limit the number of concurrent connections to keep the charge-back costs down.

In addition to the potential load and costs associated with the database resources, the Oracle BI Server allocates shared memory for each connection upon server startup. This raises the number of connections and increases Oracle BI Server memory usage.

Table 7 on page 66 contains a brief description of preconfigured connection pools.

Table 7. Preconfigured Connection Pools

Connection Name	Description
BBB Data Warehouse	Enterprise visibility to the Oracle Business Analytics Warehouse. Oracle's Siebel operational applications (Oracle BI customers only).
BBB OLTP	Enterprise visibility to Oracle's Siebel transactional database. Oracle's Siebel operational applications (Oracle BI customers only).
BBB XLS Data	DSN is the same as the connection pool name. Data is stored in C:\Data\Applications\BBB\BBB. Oracle's Siebel operational applications (Oracle BI customers only).
ERM OLTP	Oracle's Siebel Workforce Analytics connection to Oracle's Siebel transactional database. (Workforce Oracle BI customers only.)
Externalized Metadata Strings	Connection to Oracle's Siebel operational application database to load the translations of Metadata Strings. Financial Services customers and all customers deploying Oracle BI in a language other than English. **NOTE:** This connection pool is configured to be the same as the Oracle's Siebel transactional database.
Forecasting Oracle Business Analytics Warehouse	Connection to the Oracle Business Analytics Warehouse. (Forecasting Oracle BI customers only.)
Forecasting Siebel OLTP	Connection to Oracle's Siebel transactional database. (Real-time Forecasting Oracle BI customers only.)
Incentive Compensation Siebel OLTP	Database connection to Oracle's Siebel transactional database. (Incentive Compensation Oracle BI customers only.)
Pharma Data Warehouse	Connection to the Pharmaceutical data warehouse. (Pharmaceutical industry-specific customers only.)
Real-time OLTP	Connection to Oracle's Siebel transactional database. Real-time Oracle BI (all customers).
SIA Data Warehouse	Connection for Oracle's Siebel Industry Applications data warehouse. (For Oracle's Siebel Industry Applications customers only.)

Table 7. Preconfigured Connection Pools

Connection Name	Description
Oracle BI Usage	Usage Tracking Writer Connection Pool connects to the database where you store the usage statistics of the Oracle BI Server. (Optional for all customers.) **NOTE:** Using this requires that the Oracle BI Scheduler Server be set up to load the usage statistics into the database.
Oracle Business Intelligence Warehouse	Database connection to the Oracle Business Analytics Warehouse. (Oracle BI applications customers only.)
Siebel OLTP	**NOTE:** There are two connection pools to Oracle's Siebel transactional database. Both should be configured properly. OLTP DbAuth connects to Oracle's Siebel transactional database for Authentication and Authorization. The user name and password is preconfigured to USER and PASSWORD and should be left as such if you want to use database logons to authenticate users. (All customers.)
UQ Siebel OLTP	Connection to Oracle's Siebel transactional database. Oracle's Siebel Universal (Queuing customers only.)
Usage Accelerator Datawarehouse	Database connection to the Oracle Business Analytics Warehouse. (Oracle BI applications customers only.)

This section includes the following topics:

- Creating or Changing Connection Pools on page 67
- Setting Up Write-Back Properties on page 77
- Setting Up Connection Pool Properties for Multidimensional Data Sources on page 73
- Setting Up Additional Connection Pool Properties for an XML Data Source on page 76
- Setting Up the Persist Connection Pool Property on page 79

Creating or Changing Connection Pools

You must create a database object before you create a connection pool. Typically, the database object and connection pool are created automatically when you import the physical schema. You create or change a connection pool in the Physical layer of the Administration Tool.

CAUTION: It is strongly recommend that customers use OCI for connecting to Oracle. ODBC should only be used to import from Oracle.

This section contains the following topics:

- About Connection Pools for Initialization Blocks on page 68
- Setting Up General Properties For Connection Pools on page 68

About Connection Pools for Initialization Blocks

It is recommended that you create a dedicated connection pool for initialization blocks. This connection pool should not be used for queries.

Additionally, it is recommended that you isolate the connections pools for different types of initialization blocks. This also makes sure that authentication and login-specific initialization blocks do not slow down the login process. The following types should have separate connection pools:

- All Authentication and login-specific initialization blocks such as language, externalized strings, and group assignments.
- All initialization blocks that set session variables.
- All initialization blocks that set repository variables. These initialization blocks should always be run using the system Administrator user login.

  Be aware of the number of these initialization blocks, their scheduled refresh rate, and when they are scheduled to run. Typically, it would take an extreme case for this scenario to affect resources. For example, refresh rates set in minutes, greater than 15 initialization blocks that refresh concurrently, and a situation in which either of these scenarios could occur during prime user access time frames. To avoid straining the available resources, you might want to disable query logging for the default Oracle BI Administrator.

Initialization blocks should be designed so that the maximum number of Oracle BI Server variables may be assigned by each block. For example, if you have five variables, it is more efficient and less resource intensive to construct a single initialization block containing all five variables. When using one initialization block, the values will be resolved with one call to the back end tables using the initialization string. Constructing five initialization blocks, one for each variable, would result in five calls to the back end tables for assignment.

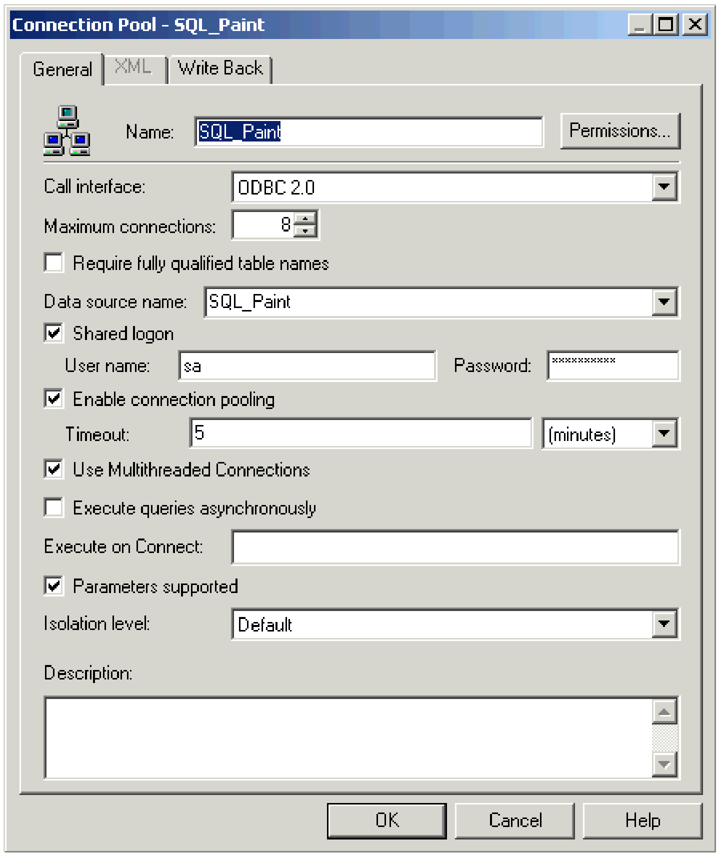
Setting Up General Properties For Connection Pools

Use this section to complete the General tab.

To set up general properties for connection pools

1. In the Physical layer of the Administration Tool, right-click a database and choose New Object > Connection Pool, or double-click an existing connection pool.

   The following is an illustration of the General tab in the Connection Pool dialog box.



Connection Pool - SQL_Paint

Name: SQL_Paint

Call interface: ODBC 2.0

Maximum connections: 8

☐ Require fully qualified table names

Data source name: SQL_Paint

☑ Shared logon

User name: sa

Password: ***********

☑ Enable connection pooling

Timeout: 5 (minutes)

☑ Use Multithreaded Connections

☐ Execute queries asynchronously

Execute on Connect:

☑ Parameters supported

Isolation level: Default

Description:

2 In the Connection Pool dialog box, click the General tab, and then complete the fields using information in Table 8 on page 70.

Properties that are specific to a multidimensional data sources can be found in Table 9 on page 75.

Table 8. Connection Pool General Properties

Field or Button	Description
Call interface	The application program interface (API) with which to access the data source. Some databases may be accessed using native APIs, some use ODBC, and some work both ways. If the call interface is XML, the XML tab is available but options that do not apply to XML data sources are not available.
Data source name	The drop-down list shows the User and System DSNs configured on your system. A data source name that is configured to access the database to which you want to connect. The data source name needs to contain valid logon information for a data source. If the information is invalid, the database logon specified in the DSN will fail.
Enable connection pooling	Allows a single database connection to remain open for the specified time for use by future query requests. Connection pooling saves the overhead of opening and closing a new connection for every query. If you do not select this option, each query sent to the database opens a new connection.
Execute on Connect	Allows the Oracle BI Administrator to specify a command to be executed each time a connection is made to the database. The command may be any command accepted by the database. For example, it could be used to turn on quoted identifiers. In a mainframe environment, it could be used to set the secondary authorization ID when connecting to DB2 to force a security exit to a mainframe security package such as RACF. This allows mainframe environments to maintain security in one central location.
Execute queries asynchronously	Indicates whether the data source supports asynchronous queries.

Table 8. Connection Pool General Properties

Field or Button	Description
Isolation level	For ODBC and DB2 gateways, the value sets the transaction isolation level on each connection to the back-end database. The isolation level setting controls the default transaction locking behavior for all statements issued by a connection. Only one of the options can be set at a time. It remains set for that connection until it is explicitly changed. The following is a list of the options: **Committed Read.** Specifies that shared locks are held while the data is read to avoid dirty reads. However, the data can be changed before the end of the transaction, resulting in non repeatable reads or phantom data. **Dirty Read.** Implements dirty read (isolation level 0 locking). When this option is set, it is possible to read uncommitted or dirty data, change values in the data, and have rows appear or disappear in the data set before the end of the transaction. This is the least restrictive of the isolation levels. **Repeatable Read.** Places locks on all data that is used in a query, preventing other users from updating the data. However, new phantom rows can be inserted into the data set by another user and are included in later reads in the current transaction. **Serializable.** Places a range lock on the data set, preventing other users from updating or inserting rows into the data set until the transaction is complete. This is the most restrictive of the four isolation levels. Because concurrency is lower, use this option only if necessary.
Maximum connections	The maximum number of connections allowed for this connection pool. The default is 10. This value should be determined by the database make and model and the configuration of the hardware box on which the database runs as well as the number of concurrent users who require access. **NOTE:** For deployments with Intelligence Dashboard pages, consider estimating this value at 10% to 20% of the number of simultaneous users multiplied by the number of requests on a dashboard. This number may be adjusted based on usage. The total number of all connections in the repository should be less than 800. To estimate the maximum connections needed for a connection pool dedicated to an initialization block, you might use the number of users concurrently logged on during initialization block execution.
Name	The name for the connection pool. If you do not type a name, the Administration Tool generates a name. For multidimensional and XML data sources, this is prefilled.
Parameters supported	If the database features table supports parameters and the connection pool check box property for parameter support is unchecked, special code executes that allows the Oracle BI Server to push filters (or calculations) with parameters to the database. The Oracle BI Server does this by simulating parameter support within the gateway/adapter layer by sending extra SQLPrepare calls to the database.

Table 8. Connection Pool General Properties

Field or Button	Description
Permissions	Assigns permissions for individual users or groups to access the connection pool. You can also set up a privileged group of users to have its own connection pool.
Require fully qualified table names	Select this check box, if the database requires it. When this option is selected, all requests sent from the connection pool use fully qualified names to query the underlying database. The fully qualified names are based on the physical object names in the repository. If you are querying the same tables from which the physical layer metadata was imported, you can safely check the option. If you have migrated your repository from one physical database to another physical database that has different database and schema names, the fully qualified names would be invalid in the newly migrated database. In this case, if you do not select this option, the queries will succeed against the new database objects. For some data sources, fully qualified names might be safer because they guarantee that the queries are directed to the desired tables in the desired database. For example, if the RDBMS supports a master database concept, a query against a table named *foo* first looks for that table in the master database, and then looks for it in the specified database. If the table named *foo* exists in the master database, that table is queried, not the table named *foo* in the specified database.
Shared logon	Select the Shared logon check box if you want all users whose queries use the connection pool to access the underlying database using the same user name and password. If this option is selected, then all connections to the database that use the connection pool will use the user name and password specified in the connection pool, even if the user has specified a database user name and password in the DSN (or in user configuration). If this option is not selected, connections through the connection pool use the database user ID and password specified in the DSN or in the user profile.

Table 8. Connection Pool General Properties

Field or Button	Description
Timeout (Minutes)	Specifies the amount of time, in minutes, that a connection to the data source will remain open after a request completes. During this time, new requests use this connection rather than open a new one (up to the number specified for the maximum connections). The time is reset after each completed connection request. If you set the timeout to 0, connection pooling is disabled; that is, each connection to the data source terminates immediately when the request completes. Any new connections either use another connection pool or open a new connection.
Use Multithreaded Connections	When the check box is select ed, Oracle BI Server terminates idle physical queries (threads). When not selected, one thread is tied to one database connection (number of threads = maximum connections). Even if threads are idle, they consume memory. The parameter DB_GATEWAY_THREAD_RANGE in the Server section of NQSConfig.ini establishes when Oracle BI Server terminates idle threads. The lower number in the range is the number of threads that are kept open before Oracle BI Server takes action. If the number of open threads exceeds the low point in the range, Oracle BI Server terminates idle threads. For example, if DB_GATEWAY_THREAD_RANGE is set to 40-200 and 75 threads are open, Oracle BI Server terminates any idle threads.

Setting Up Connection Pool Properties for Multidimensional Data Sources

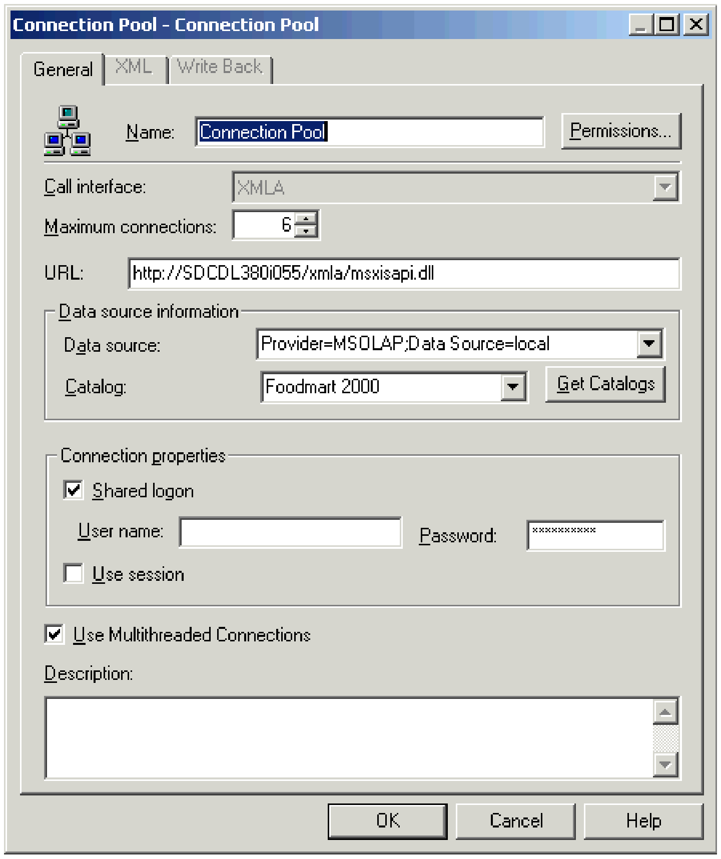
When you import an external multidimensional data source, the connection pool is automatically set up in the physical layer. You can add a connection pool manually using the instructions in this section.

Table 8 on page 70 describes some general properties that are shared among connection pools with different data sources. Table 9 on page 75 describes the properties in this dialog box that are unique to multidimensional data sources.

To set up connection pools for a multidimensional data source

1 In the Physical layer of the Administration Tool, right-click a database and select New Object > Connection Pool, or double-click an existing connection pool.

The following is an illustration of the General tab for a multidimensional data source in the Connection Pool dialog box.



Name: Connection Pool

Call interface: XMLA

Maximum connections: 6

URL: http://SDCDL380i055/xmla/msxisapi.dll

Data source information

Data source: Provider=MSOLAP;Data Source=local

Catalog: Foodmart 2000

Connection properties

☑ Shared logon

User name:

Password: **********

☐ Use session

☑ Use Multithreaded Connections

Description:

2 In the Connection Pool dialog box, in the General tab, complete the fields using information in Table 8 on page 70 and Table 9 on page 75.

NOTE: Table 9 on page 75 describes the connection pool properties or descriptions that are unique to multidimensional data sources.

Table 9. Multidimensional Data Source Connection Pool General Properties

Property	**Description**
Data Source Information: Catalog	The list of catalogs available, if you imported data from your data source. The cube tables correspond to the catalog you use in the connection pool.
Data Source Information: Data Source	The vendor-specific information used to connect to the multidimensional data source. Consult your multidimensional data source administrator for setup instructions because specifications may change. For example, if you use v1.0 of the XML for Analysis SDK then the value should be `Provider-MSOLAP;Data Source-local`. If using v1.1, then it should be `Local Analysis Server`.
Shared logon	Select the Shared logon check box if you want all users whose queries use the connection pool to access the underlying database using the same user name and password. If this option is selected, then all connections to the database that use the connection pool will use the user name and password specified in the connection pool, even if the user has specified a database user name and password in the DSN (or in the user configuration). If this option is not selected, connections through the connection pool use the database user ID and password specified in the DSN or in the user profile.
URL	The URL to connect to the XMLA provider. It points to the XMLA virtual directory of the machine hosting the cube. This virtual directory needs to be associated with msxisapi.dll (part of the Microsoft XML for Analysis SDK installation). For example, the URL might look like the following: http://SDCDL360i101/xmla/msxisap.dll
Use session	Controls whether queries go through a common session. Consult your multidimensional data source administrator to determine whether this option should be enabled. Default is Off (not checked).

Setting Up Additional Connection Pool Properties for an XML Data Source

Use the XML tab of the Connection Pool dialog box to set additional properties for an XML data source.

CAUTION: The XML tab of the Connection Pool dialog box provides the same functionality as the XML tab of the Physical Table dialog box. However, when you set the properties in the XML tab of the Physical Table dialog box you will override the corresponding settings in the Connection Pool dialog box.

To set up connection pool properties for an XML data source

1. Right-click the XML database, select New Object > Connection Pool.
2. In the Connection Pool dialog box, click the XML tab.
3. Complete the fields, using Table 10 on page 76 as a guide.

Table 10. XML Connection Pool Properties

Property	Description
Connection method Search script	Used for XML Server data source.
Connection properties Maximum connections	Default is 10.
Connection properties URL loading time out	Used for XML data source. Time-out interval for queries. The default is 15 minutes. If you specified a URL to access the data source, set the URL loading time-out as follows: ■ Select a value from the drop-down list (Infinite, Days, Hours, Minutes or Seconds). ■ Specify a whole number as the numeric portion of the interval.
Connection properties URL refresh interval	Used for XML data source. The refresh interval is analogous to setting cache persistence for database tables. The URL refresh interval is the time interval after which the XML data source will be queried again directly rather than using results in cache. The default setting is infinite, meaning the XML data source will never be refreshed. If you specified a URL to access the data source, set the URL refresh interval. ■ Select a value from the drop-down list (Infinite, Days, Hours, Minutes or Seconds). ■ Specify a whole number as the numeric portion of the interval.

Table 10. XML Connection Pool Properties

Property	Description
Query input supplements Header file/Trailer file	Used for XML Server data source.
Query output format	Choose only XML for an XML data source. Other choices are available for an XML Server data source.
XPath expression	An XPath expression is a simple XSLT transformation rule that fits into one line. It is not essential, given the support of XSLT, but it is supported for convenience. A sample entry might be */BOOK[not(PRICE>'200')]. ■ For an XML Server data source, you cannot specify an XPath expression on the XML tab of the Physical Table object.
XSLT file	An XSLT file contains formatting rules written according to the XSLT standard. It defines how an XML file may be transformed. The current implementation of the XML gateway does not support all XML files, only those that resemble a tabular form, with repeating second level elements. To increase the versatility of the XML gateway, customers can specify an XSLT file to preprocess an XML file to a form that the Oracle BI Server supports. Specifying the XSLT file in the connection pool applies it to all the XML physical tables in the connection pool. ■ For an XML data source, you can specify an XSLT file on a per-table basis on the XML tab of the Physical Table object. This overrides what you specified in the connection pool. ■ For an XML Server data source, you cannot specify an XSLT file on the XML tab of the Physical Table object.

To specify query output format settings

1. (Optional) For an XSLT file, type the path to and name of the XSLT file in the XSLT File field, or use the Browse button.
2. (Optional) For an XPath expression, type the XPath expression in the XPath Expression field, for example, //XML, or use the Browse button.

Setting Up Write-Back Properties

Use this section to complete the Write Back tab in the Connection Pool dialog box.

To set up write-back properties for connection pools

1. In the Physical layer of the Administration Tool, right-click a database and select New Object > Connection Pool, or double-click an existing connection pool.
2. In the Connection Pool dialog box, click the Write Back tab.
3. In the Write Back tab, complete the fields using Table 11 on page 78 as a guide.

Table 11. Field Descriptions for Write Back Tab

Field	Description
Bulk Insert Buffer Size (KB)	Used for limiting the number of bytes each time data is inserted in a database table.
Bulk Insert Transaction Boundary	Controls the batch size for an insert in a database table.
Temporary table Database Name	Database where the temporary table will be created. This property applies only to IBM OS/390 because IBM OS/390 requires database name qualifier to be part of the CREATE TABLE statement. If left blank, OS/390 will default the target database to a system database for which the users may not have Create Table privileges.
Temporary table Owner	Table owner name used to qualify a temporary table name in a SQL statement, for example to create the table owner.tablename. If left blank, the user name specified in the writeable connection pool is used to qualify the table name and the Shared Logon field on the General tab should also be set.
Temporary table Prefix	When the Oracle BI Server creates a temporary table, these are the first two characters in the temporary table name. The default value is TT.

Table 11. Field Descriptions for Write Back Tab

Field	Description
Temporary table Tablespace Name	Tablespace where the temporary table will be created. This property applies to OS/390 only as OS/390 requires tablespace name qualifier to be part of the CREATE TABLE statement. If left blank, OS/390 will default the target database to a system database for which the users may not have Create Table privileges.
Unicode Database Type	Select this check box when working with columns of an explicit Unicode data type, such as NCHAR, in an Unicode database. This makes sure that the binding is correct and data will be inserted correctly. Different database vendors provide different character data types and different levels of Unicode support. Use the following general guidelines to determine when to set this check box: ■ On a database where CHAR data type supports Unicode and there is no separate NCHAR data type, do not select this check box. ■ On a database where NCHAR data type is available, it is recommended to select this check box. ■ On a database where CHAR and NCHAR data type are configured to support Unicode, selecting this check box is optional. **NOTE:** Unicode and non-Unicode datatypes cannot coexist in a single non-Unicode database. For example, mixing the CHAR and NCHAR data types in a single non-Unicode database environment is not supported.

Setting Up the Persist Connection Pool Property

A persist connection pool is a database property that is used for specific types of queries (typically used to support Marketing queries). In some queries, all of the logical query cannot be sent to the transactional database because that database might not support all of the functions in the query. This issue might be solved by temporarily constructing a physical table in the database and rewriting the Oracle BI Server query to reference the new temporary physical table.

You could use the persist connection pool in the following situations:

- **Populate stored procedures.** Use to rewrite the logical SQL result set to a managed table. Typically used by Oracle's Siebel Marketing Server to write segmentation cache result sets.
- **Perform a generalized subquery.** Stores a nonfunction subquery in a temporary table and then rewrites the original subquery result against this table. Reduces data movement between the Oracle BI Server and the database and supports unlimited IN list values and might result in improved performance.

NOTE: In these situations, the user issuing the logical SQL needs to have been granted the Populate privilege on the target database.

The persist connection pool functionality designates a connection pool with write-back capabilities for processing this type of query. You can assign one connection pool in a single database as a persist connection pool. If this functionality is enabled, the User name specified in the connection pool must have the privileges to create DDL (Data Definition Language) and DML (Data Manipulation Language) in the database.

Example of Using Buffer Size and Transaction Boundary

If each row size in a result set is 1 KB and the buffer size is 20 KB, then the maximum array size will be 20 KB. If there are 120 rows, there will be 6 batches with each batch size limited to 20 rows.

If you set the Transaction boundary field to 3, the server will commit twice. The first time the server commits after row 60 (3 * 20). The second time the server commits after row 120. If there is a failure when the server commits, the server will only rollback the current transaction. For example, if there are two commits and the first commit succeeds but the second commit fails, the server only rolls back the second commit. To make sure that the array-based insert runs successfully, it is recommended that you not set the transaction boundary greater than 10 and you set the buffer size to approximately 32 KB.

To assign a persist connection pool

1. In the Physical layer, double-click the database icon.
2. In the Database dialog box, click the General tab.
3. In the Persist Connection Pool area, click Set.

   If there is only one connection pool, it appears in the Persist Connection Pool field.
4. If there are multiple connection pools, in the Browse dialog box, select the appropriate connection pool, and then click OK.

   The selected connection pool name appears in the Persist connection pool field.
5. (Optional) To set write-back properties, click the Connection Pools tab.
6. In the connection pool list, double-click the connection pool.
7. In the Connection Pool dialog box, click the Write Back tab.
8. Complete the fields using Table 11 on page 78 as a guide.
9. Click OK twice to save the persist connection pool.

To remove a persist connection pool

1. In the Physical layer, double-click the database icon.
2. In the Database dialog box, click the General tab.
3. In the Persist Connection Pool area, click Clear.

   The database name is replaced by not assigned in the Persist connection pool field.
4. Click OK.

About Physical Tables

This topic is part of the "Process of Creating the Physical Layer from Relational Data Sources" on page 56 and the "Process of Creating the Physical Layer from Multidimensional Data Sources" on page 58.

A physical table is an object in the Physical layer of the Administration Tool that corresponds to a table in a physical database. Physical tables are usually imported from a database or another data source. They provide the metadata necessary for the Oracle BI Server to access the tables with SQL requests.

In addition to importing physical tables, you can create virtual physical tables in the Physical layer, using values in the Table Type field in the Physical Table dialog box. A virtual physical table can be a stored procedure or a Select statement. Creating virtual tables can provide the Oracle BI Server and the underlying databases with the proper metadata to perform some advanced query requests.

Table Types for Physical Tables

The Table Type drop-down list in the General tab of the Physical Table dialog box allows you to specify the physical table object type. Table 12 on page 81 provides a description of the available object types.

Table 12. Table Type Descriptions for Physical Tables

Table Type	Description
Physical Table	Specifies that the physical table object represents a physical table.

Table 12. Table Type Descriptions for Physical Tables

Table Type	Description
Stored Proc	Specifies that the physical table object is a stored procedure. When you select this option, you type the stored procedure in the text box. Requests for this table will call the stored procedure. For stored procedures that are database specific, select the Use database specific SQL check box. At run time, if a stored procedure has been defined, the stored procedure will be executed; otherwise, the default configuration will be executed. **NOTE:** Stored procedures using an Oracle database do not return result sets. For more information, refer to "Using Stored Procedures with an Oracle Database" on page 82. For information about stored procedures and alias tables, see "About Physical Alias Tables" on page 83.
Select	Specifies that the physical table object is a Select statement. When you select this option, you type the select statement in the text field and you need to manually create the table columns. The column names must match the ones specified in the Select statement. Column aliases are required for advanced SQL functions, such as aggregates and case statements. Requests for this table will execute the Select statement. For Select statements that are database specific, select the Use database specific SQL check box. At run time, if a Select statement has been defined, the Select statement will be executed; otherwise, the default configuration will be executed.

Using Stored Procedures with an Oracle Database

Stored Procedures within Oracle do not return result sets. Therefore they cannot be initiated from within Oracle BI. You need to rewrite the procedure as an Oracle function, use it in a select statement in the Administration Tool initialization block, and then associate it with the appropriate Oracle BI session variables in the Session Variables dialog box.

The function uses the GET_ROLES function and takes a user Id as a parameter and returns a semi-colon delimited list of group names.

The following is an example of an initialization SQL string using the GET_ROLES function that is associated with the USER, GROUP, DISPLAYNAME variables:

```
select user_id, get_roles(user_id), first_name || ' ' || last_name

    from csx_security_table

    where user_id = ':USER' and password = ':PASSWORD'
```

Creating and Setting Up Physical Tables

This topic is part of the "Process of Creating the Physical Layer from Relational Data Sources" on page 56 and the "Process of Creating the Physical Layer from Multidimensional Data Sources" on page 58.

For all data sources, you can define general properties, columns, a primary key, and foreign keys.

About Physical Alias Tables

An *Alias table* (Alias) is a physical table with the type of Alias. It is a reference to a *logical table source*, and inherits all its column definitions and some properties from the logical table source. A logical table source shows how the logical objects are mapped to the physical layer and can be mapped to physical tables, stored procedures, and select statements. An alias table can be a reference to any of these logical table source types. For more information, see "Creating and Administering the Business Model and Mapping Layer in an Oracle BI Repository" on page 109.

Alias Tables can be an important part of designing a physical layer. The following is a list of the main reasons to create an alias table:

- To reuse an existing table more than once in your physical layer (without having to import it several times.
- To set up multiple alias tables, each with different keys, names, or joins.
- To help you design sophisticated star or snowflake structures in the business model layer. Alias tables are critical in the process of converting ER Schemas to Dimensional Schemas. For more information, see "Identifying the Database Content For The Business Model" on page 45.

You can allow an alias table to have cache properties that differ from its source table by setting an override flag in the Physical Table dialog box. In alias tables, columns cannot be added, deleted, or modified. Columns are automatically synchronized; no manual intervention is required.

NOTE: Synchronization makes sure that source tables and their related alias tables have the same column definitions. For example, if you delete a column in the source table the column is automatically removed from the alias table.

You cannot delete source tables unless you delete all its alias tables first. You can change the source table of an alias table, if the new source table is a superset of the current source table. However, this could result in an inconsistent repository if changing the source table deletes columns that are being used. If you attempt to do this, a warning message appears to let you know that this could cause a problem and allows you to cancel the action.

NOTE: Running a consistency check identifies orphaned aliases.

When you edit a physical table or column in online mode, all alias tables and columns must be checked out. The behavior of online checkout uses the following conventions:

- If a source table or column is checked out, all its alias tables and columns will be checked out.
- If an alias table or column is checked out, its source table and column will be checked out.
- The checkout option is available for online repositories (if not read-only) and for all source and alias tables and columns.

Alias tables inherit some properties from their source tables. A property that is *proxied* is a value that is always the same as the source table, and cannot be changed. If the source table changes its value for that particular property, the same will be applied on the alias table.

The following is a list of the properties that are proxied:

- IsCacheable (the inherited property can be overridden)
- CacheExpiry (the inherited property can be overridden)
- Row Count
- Last Updated
- Table Type
- External Db Specifications

The following is list of the properties that are not proxied:

- Name
- Description
- Display Folder Containers
- Foreign Keys
- Columns (tables don't share columns, ever. Aliases and sources have distinctly different columns that alias each other)
- Table Keys
- Complex Joins
- Source Connection Pool
- Polling Frequency
- All XML attributes

About Creating and Setting Up Physical Tables for Multidimensional Data Sources

Each cube from a multidimensional data source is set up as a physical cube table, a type of physical table. It has all the capabilities of a table such as physical cube columns and keys (optional) and foreign keys (optional). It also has cube-specific metadata such as hierarchies and levels.

In the Physical layer, a physical cube table looks like a regular table but will have a different icon. For more information about icons, refer to "Icons and Symbols in the Administration Tool" on page 21.

When you import the physical schema, the Oracle BI Server imports the cube, including its metrics, hierarchies, and levels. Expanding the hierarchy icon reveals the levels in the hierarchy. In the Physical Cube Table dialog box, the Hierarchies tab lists the dimensional hierarchies in the cube.

Each multidimensional catalog in the database can contain multiple physical cubes. You can import one or more of these cubes into your BI repository. You can create a cube table manually. However, it is recommended that you import cube tables and their components.

NOTE: If creating a cube manually, it is strongly recommended that you build each cube one hierarchy at a time and test each one before building another. For example, create the time hierarchy and a measure, and then test it. When it is correct, create the geography hierarchy and test it. This will help make sure you have set up each cube correctly and make it easier to identify any setup errors.

To create a physical table or a physical cube table and any necessary properties, perform the following tasks:

- Creating and Administering General Properties for Physical Tables on page 85
- Creating and Administering Columns and Keys in a Physical Table on page 87
- Setting Up Hierarchies in the Physical Layer for a Multidimensional Data Source on page 91
- Setting Physical Table Properties for an XML Data Source on page 96

Creating and Administering General Properties for Physical Tables

Use the General tab of the Physical Table dialog box to create or edit a physical table in the Physical layer of the Administration Tool.

This section contains the following topics:

- Creating or Editing Physical Tables on page 85
- Deleting a Physical Table on page 87

Creating or Editing Physical Tables

This section describes how to create or edit the general properties for a table. This includes physical cube tables and alias tables.

To create a physical table or edit general properties for tables and alias tables

1. In the Physical layer of the Administration Tool, perform one of the following steps:
   - To create a physical table, right-click the physical database and choose New Object > Physical Table.
   - To create a physical cube table for a multidimensional data source, right-click the physical database and choose New Object > Physical Cube Table.

     NOTE: It is strongly recommended that you import cube tables, not create them manually.
   - To create an alias table, right click a physical table, and choose New Object > Alias.

     NOTE: You can also create aliases on opaque views and stored procedures.

- To edit an existing physical table, double-click the physical table icon.

2 In the selected Physical Table dialog box, complete the fields using Table 13 on page 86 as a guide.

Table 13. Physical Table General Properties for Relational and XML Data Sources

Property	Description
Name	The Oracle BI Administrator assigns a name to new table.
Cacheable	To include the table in the Oracle BI Server query cache, select this check box. When you select this check box, the Cache persistence time settings become active. This is useful for OLTP data sources and other data sources that are updated frequently. Typically, you should check this option for most tables.
Cache never expires	When you select this option, cache entries do not expire. This could be useful when a table will be important to a large number of queries users might run. For example, if most of your queries have a reference to an account object, keeping it cached indefinitely could actually improve performance rather than compromise it. **CAUTION:** This is only of use on some objects. Set this option for too many objects and the cache will become manageably large or objects might begin dropping out of the cache at inefficient times.
Cache persistence time	How long table entries should persist in the query cache. The default value is Infinite, meaning that cache entries do not automatically expire. However, this does not mean that an entry will always remain in the cache. Other invalidation techniques, such as manual purging, LRU (Least Recently Used) replacement, metadata changes, and use of the cache polling table, result in entries being removed from the cache. If a query references multiple physical tables with different persistence times, the cache entry for the query will exist for the shortest persistence time set for any of the tables referenced in the query. This makes sure that no subsequent query gets a cache hit from an expired cache entry. If you change the default to minutes or seconds, type a whole number into the field on the left. For more information, refer to "Troubleshooting Problems with an Event Polling Table" on page 253.
External name	Applies to physical cube tables from a multidimensional data source. If you select a Table Type of Physical Table, the external data source name appears.
Override Source Table Caching Properties	Check box available for alias tables. When selected, the cacheable properties become available and you can clear or select the appropriate options.
Source Table	Applies to alias tables. The Select button allows you to choose the physical table from which to create an alias table.

Table 13. Physical Table General Properties for Relational and XML Data Sources

Property	Description
Table Type	Physical Table values: Physical Table, Stored Proc (stored procedure), or Select. Physical Cube Table values: Physical Table or Select.
Use Dynamic Name	Check box available for non-multidimensional data source tables when you select a Table type of Physical Table. When selected, a dialog box opens in which you can choose a session variable.
Use Database Specific SQL Default Initialization String	For non-multidimensional data source tables (not alias tables), this appears if you choose a Table Type of Stored Proc or Select. For multidimensional data source tables, this appears if you choose a Table Type of Select. When you select the check box, you can specify the database and type the SQL.

Deleting a Physical Table

When you delete a physical table, all dependent objects are deleted. For example columns, keys, and foreign keys. When you delete a physical cube table, hierarchies are also deleted.

NOTE: The deletion fails if an alias exists on the physical table.

To delete a physical table from the Physical layer

1. In the Physical layer of the Administration Tool, locate the table that you want to delete.
2. Right-click the table and choose Delete.

Viewing Data in Physical Tables or Columns

You can view the data in a physical table or an individual physical column by right-clicking the object and choosing View Data. In online editing mode, you must check in changes before you can use View Data.

View Data is not available for physical cube tables or columns. For information, refer to "Viewing Members in Physical Cube Tables" on page 94.

CAUTION: View Data will not work if you set the user name and password for connection pools to :USER and :PASSWORD. In offline mode, the Set values for variables dialog box appears so that you can populate :USER and :PASSWORD as part of the viewing process.

Creating and Administering Columns and Keys in a Physical Table

Each table in the Physical layer of the Administration Tool has one or more physical columns.

The Columns, Keys, and Foreign Keys tabs in the Physical Table dialog box allow you to view, create new, and edit existing columns, keys, and foreign keys that are associated with the table.

The following list describes the buttons that appear in the tabs:

- **New.** Opens the dialog box that corresponds to the tab.
- **Edit.** When you select an object and then click Edit, the dialog box that corresponds to the tab appears. You can then edit the object's properties.
- **Delete.** Deletes the selected object.

This section contains the following tasks:

- About Measures in a Multidimensional Data Source on page 88
- Creating and Editing a Column in a Physical Table on page 89
- Specifying a Primary Key for a Physical Table on page 90
- Deleting a Physical Column For All Data Sources on page 91

About Measures in a Multidimensional Data Source

You need to select the aggregation rule for a physical cube column carefully to make sure your measures are correct. Setting it correctly might improve performance.

Use the following guidelines to verify and assign the aggregation rule correctly:

- Verify aggregation rules after importing a cube. Typically, aggregation rules are assigned correctly when you import the cube. However, if a measure is a calculated measure, the aggregation rule will be reported as None. Therefore, you should examine the aggregation rule for all measures after importing a cube to verify that the aggregation rule has been assigned correctly.

  For all measures assigned an aggregation rule value of None, contact the multidimensional data source administrator to verify that the value of the aggregation rule is accurate. If you need to change the aggregation rule, you can change it in the Physical Cube Column dialog box.

- Setting the aggregation rule when you build the measures manually. Set the aggregation rule to match its definition in the multidimensional data source.

About Externally Aggregated Measures

In a multidimensional data source, some cubes contain very complex, multi-level based measures. If the Oracle BI Administrator assigns an aggregation rule of Aggr_External, the BI Server will bypass its internal aggregation mechanisms and use the pre-aggregated measures. When imported, these measures are assigned an aggregate value of None.

The following are some guidelines for working with pre-aggregated measures:

- External aggregation only applies to multidimensional data sources (such as MS Analysis Services and SAP/BW) that support these complex calculations.

- You cannot assign external aggregation to measures from standard data sources (relational). If the measure is supported and can be mapped to a relational database, then it is not complex and does not require external aggregation.
- You cannot mix noncomplex measures from standard data sources (relational) with complex measures from multidimensional data sources.
- You can mix noncomplex measures from standard data sources (relational) with noncomplex measures from multidimensional data sources if they are aggregated through the Oracle BI Server.

Creating and Editing a Column in a Physical Table

If the column is imported, the properties of the column are set automatically. The following list contains information about nullable and data type values for columns imported into the Physical layer.

- **Nullable.** The option Nullable in the Physical Columns dialog box indicates whether null values are allowed for the column. If null values can exist in the underlying table, you need to select this option. This allows null values to be returned to the user, which is expected with certain functions and with outer joins. It is generally safe to change a non-nullable value to a nullable value in a physical column.
- **Data type.** The data type list indicates the data type of the columns. Use caution in changing the data type values. Setting the values to ones that are incorrect in the underlying data source might cause unexpected results. If there are any data type mismatches, correct them in the repository or reimport the columns with mismatched data types.

  If you reimport columns, you also need to remap any logical column sources that reference the remapped columns. The data type of a logical column in the business model must match the data type of its physical column source. The Oracle BI Server will pass these logical column data types to client applications.

NOTE: Except when stated otherwise, the characteristics and behavior of a physical cube column are the same as for other physical columns.

About Creating and Editing a Column With an Associated Column in an Alias Table

Creating and editing a column in a physical source table that has a corresponding column in an alias table, causes the following results:

- Creating a Source Column. Creating a column in the physical source table, creates the same column on all its alias tables. Here are the steps involved:
- Deleting a Source Column. Deleting a column in the physical source table, deletes the same column on all its alias tables.
- Modifying a Source Column. Modifying a column in the physical source table, modifies the same column on all its alias tables.

To create or edit a physical column

1 In the Physical layer of the Administration Tool, perform one of the following steps:

- To create a physical column, right-click a physical table and choose New Object > Physical Column from the shortcut menu.
- To create a physical cube column for a multidimensional data source, right-click a physical cube table, and choose New Object > Physical Cube Column.
- To edit an existing physical column, double-click the physical column icon.

2 In the Physical Column dialog box, type a name for the physical column.

   For XML data sources, this field will store and display the unqualified name of a column (attribute) in an XML document.

3 In the Type field, select a data type for the physical column.

4 If applicable, specify the length of the data type.

   For multidimensional data sources, if you select VARCHAR, you need to type a value in the Length field.

5 Select the Nullable option if the column is allowed to have null values.

6 In the External Name field, type an external name.

   - Required if the same name (such as STATE) is used in multiple hierarchies.
   - Optional for XML documents. The External Name field stores and displays the fully qualified name of a column (attribute).

7 (Multidimensional data sources) When the physical cube column is a measure, in the Aggregation rule drop-down list, select the appropriate value.

   NOTE: A new physical cube column is created as a measure by default. To change this, refer to "Setting Up Hierarchies in the Physical Layer for a Multidimensional Data Source" on page 91.

8 Click OK.

Specifying a Primary Key for a Physical Table

Use the Physical Key dialog box to specify the column or columns that define the primary key of the physical table.

To specify a primary key for a physical table

1 In the Physical layer of the Administration Tool, right-click a physical table and choose Properties.

2 In the Physical Table dialog box, click the Keys tab.

3 In the Keys tab, click New.

4 In the Physical Key dialog box, type a name for the key.

5 Select the check box for the column that defines the primary key of the physical table.

6 (Optional) In the Physical Key dialog box, type a description for the key, and then click OK.

Deleting a Physical Column For All Data Sources

You delete a physical column in the same way for all data sources. The following is a list of some results that occur:

- **Multidimensional data source.** If you delete property or key columns from a level, the association is deleted and the column changes to a measure under the parent cube table.
- **Alias tables.** Deleting a column in a physical source table from which an alias table has been created, deletes the same column on all its alias tables.

To delete a physical column from the Physical layer

1. In the Physical layer of the Administration Tool, locate the column that you want to delete.
2. Right-click the column and choose Delete.

Setting Up Hierarchies in the Physical Layer for a Multidimensional Data Source

The following are some guidelines to follow when setting up hierarchies in the Physical layer.

- Hierarchies that are ragged or have a parent-child hierarchy are not imported. You can set up unbalanced hierarchies in the physical layer by changing the hierarchy type.
- To change the column from a measure to a property or a level key, you need to set up a hierarchy and associate the cube column with the hierarchy. If you delete a property or level key column from a level, the column will change back to a measure under the parent cube table.

  CAUTION: You will need to build a matching hierarchy in the Business Model and Mapping layer. If you do not do this, queries may appear to work but might not return the correct results.

To create and maintain hierarchies in the Physical Layer, perform the following tasks:

- Adding a Hierarchy to a Physical Cube Table on page 91
- Verifying Hierarchy Levels on page 93
- Updating Member Counts on page 94
- Viewing Members in Physical Cube Tables on page 94
- Adding or Removing a Cube Column in an Existing Hierarchy on page 95
- Removing a Hierarchy from a Physical Cube Table on page 95
- Associating a Physical Cube Column with a Hierarchy Level on page 96

Adding a Hierarchy to a Physical Cube Table

Most hierarchies are imported into the physical layer. Columns associated with a hierarchy that is not imported will not be imported. If users need access to columns that are not imported, first add these columns to the physical layer and then associate them with a level in a hierarchy. This section contains instructions for adding a hierarchy.

Each level in a hierarchy has a level key. The first cube column associated with (added to) the level of a hierarchy is the level key. This must match with the data source definition of the cube. The data source cube table cannot set one column as a level key and the Oracle BI physical layer table set a different column as a level key. The icon for the column that you select first changes to the key icon after it is associated with the level of a hierarchy.

When you select columns to add to a hierarchy, it is recommended that you select them in hierarchical order, starting with the highest level. If you select multiple columns and bring them into the hierarchy at the same time, the order of the selected group of columns remains the same. After adding columns to the hierarchy, you can change the order of the columns in the Browse dialog box.

If a query does not explicitly refer to a member of a hierarchy, a default member must be used. Therefore, every hierarchy must be associated with a default member, typically the ALL member. The Hierarchy dialog box contains a check box (Default member type ALL) that you use when you want to designate the ALL member as the default. The following list contains some guidelines about selecting the check box:

- If you import the cube, the Default member type ALL check box should be automatically selected. The ALL member is identified during import.
- If you build the hierarchies manually, the check box will not be automatically selected. Before selecting the check box, ask your multidimensional data source administrator if a non-ALL default member has been defined. For example, for the Year level, 1997 might be designated as the default member. In this case, you should not select the Default member type ALL check box.

To add a hierarchy to a physical cube table

1. In the Physical layer of the Administration Tool, double-click the table to which you want to add a hierarchy.
2. In the Physical Cube Table dialog box, click the Hierarchies tab and click Add.
3. In the Hierarchy dialog box, complete the fields using Table 14 on page 93 as a guide.
4. To create a level, perform the following steps:
    a. In the Hierarchy dialog box, click Add.
    b. In the Physical Level dialog box, complete the fields using Table 14 on page 93 as a guide.

       NOTE: In a hierarchy, levels should be added from the top down (you can reorder them later). Using the correct hierarchical sequence allows your queries to return accurate information and avoids errors.
5. To add one or more columns to the level, in the Physical Level dialog box, click Add.

   NOTE: You can also add columns to a physical level by dragging and dropping physical columns on the level object. The first column you add will be a key. Subsequent columns will be properties.
6. In the Browse dialog box, perform the following steps:
    a. In the Name list, locate the columns that you want to add to the hierarchy.
    b. Select the key column first, and then click Select.
    c. In the Physical Level dialog box, click OK.

7 To add more columns, repeat Step 5 on page 92 through Step 6 on page 92.

NOTE: You can add multiple columns by pressing Ctrl on your keyboard while clicking each column, and then clicking Select.

8 When finished adding columns, in the Hierarchy dialog box, click OK.

Table 14. Hierarchy and Level Properties for Physical Cube Tables

Property	Description
Default member type ALL	Check box used to designate the ALL member as the default. Should not be selected for non-ALL default members.
Dimension Name	(Dimension Unique Name) Dimension to which the hierarchy belongs.
External Name	Fully qualified name for the object.
Level Number	Identifies the order of levels in a hierarchy. Use this property to change the order of the levels.
Time Dimension	Check box that identifies a dimension as one involving time such as year, day, quarter.
Type	Type of hierarchy: Fully Balanced, Unbalanced, Ragged Balanced, and Network.

Verifying Hierarchy Levels

It is strongly recommended that after setting up a hierarchy containing more than one level, you should verify the order of the levels in the hierarchy.

To verify the levels in a hierarchy

1 In the Physical layer of the Administration Tool, double-click the table you want to verify.

2 In the Physical Cube Table dialog box, click the Hierarchies tab.

3 In the Hierarchies tab, select a hierarchy, and then click Edit.

4 In the Hierarchy dialog box, verify the levels are correct.

The Hierarchy dialog box lists all the defined levels for the selected hierarchy. The highest level in the hierarchy should be the first (highest) item in the list.

5 If you need to reorder the hierarchy levels, select a level and click Up or Down to correct the order of the levels.

There must be multiple levels and you must select a level for the buttons to be available.

6 When the levels are correct, click OK.

7 In the Physical Cube Table dialog box, click OK.

Updating Member Counts

You must open the repository in online mode to update member counts.

To determine if counts need to be updated, move your mouse over the hierarchy or level name. A message appears to let you know that the counts need to be updated or when they were last updated.

When you update member counts, the current number of members are returned from the selected hierarchy. After successfully updating the member count, the updated member count appears in a message when you move the mouse above the hierarchy or level name. The message appears in the following syntax:

```
<hierarchy name> (<x> members, last updated <time stamp>)
```

To update member counts

1. In the Administration Tool, in the Physical layer, move your cursor over a hierarchy or level.

   If the counts need to be updated, a message appears.

2. Right-click one or more hierarchies and levels.
3. In the menu, select Update Member Count.

   An updated message appears if the update was successful.

Viewing Members in Physical Cube Tables

To view members, the repository must be opened in online mode. This is available for physical cube tables from Analysis Services and SAP/BW data sources.

You can view members of hierarchies or levels in the physical layer of repositories. The list of members by level in the hierarchy can help you determine if the XMLA connection on the server is set up properly. You might want to reduce the time it takes to return data or the size of the returned data by specifying a starting point (Starting from option) and the number of rows you want returned (Show option).

To view members

1. In the Administration Tool, in the Physical layer, right-click a hierarchy or level.
2. Select View Members.

   A window opens showing the number of members in the hierarchy and a list of the levels. You might need to enlarge the window and the columns to view all the returned data.

3. Click Query to display results.
4. When finished, click Close.

Adding or Removing a Cube Column in an Existing Hierarchy

After setting up a hierarchy you may need to add or remove a column. You might want to remove a hierarchy if it has been built incorrectly and you want to start over.

If you remove a cube column from a hierarchy, it is deleted from the hierarchy but remains in the cube table and is available for selection to add to other levels.

To add a cube column to or remove a cube column from an existing hierarchy

1. In the Physical layer of the Administration Tool, double-click the table that you want to change.
2. In the Physical Cube Table dialog box, click the Hierarchies tab.
3. Select the hierarchy you want to change, and then click Edit.
4. In the Hierarchy dialog box, select the level and click Edit.
5. In the Physical Level dialog box, perform one of the following steps:
   a. To add a column, click Add.
      - ❏ In the Browse dialog box, in the Name list, select the columns that you want to add.
      - ❏ Click Select.
   b. To remove a column, select the column and click Remove.
   c. To change the sequence of the levels in a hierarchies, select the level and click Up or Down.
   d. Click OK.
6. In the Hierarchy dialog box, click OK.
7. In the Physical Cube Table dialog box, click OK.

Removing a Hierarchy from a Physical Cube Table

You might want to remove a hierarchy if it has been built incorrectly and you want to start over or if you want to remove objects that are not being used. For example, you might import an entire physical multidimensional schema and only want to keep parts of it in the business model.

NOTE: When you delete a hierarchy in the Physical layer, you remove the hierarchy and the columns that are part of the hierarchy.

To remove a hierarchy from a physical cube table

1. In the Physical layer of the Administration Tool, double-click the table that you want to change.
2. In the Physical Cube Table dialog box, click the Hierarchies tab.
3. Select the hierarchy you want to remove, and then click Remove.

Associating a Physical Cube Column with a Hierarchy Level

Attributes are used in the physical layer to represent columns that only exist at a particular level of a hierarchy. For example, if Population is an attribute that is associated with the level State in the Geography hierarchy, when you query for Population you are implicitly asking for data that is at the State level in the hierarchy.

There can be zero or more attributes associated with a level. The first physical cube column that is associated with a level becomes the level key. If you associate subsequent columns with a level, they become attributes, not level keys.

Example of Associating a Physical Cube Column with a Hierarchy

You have a level called State and you want to associate a column called Population with this level.

- Create the hierarchy and the State level.
- Create the physical cube column for Population.
- In the Physical Cube Table dialog box, in the Hierarchies tab, select the State level and click Edit.
- In the Hierarchy dialog box, click Add.
- In the Physical Level dialog box, click Add.
- In the Browse dialog box, select the Population column and click Select.

  The measure icon changes to the property icon.

Setting Physical Table Properties for an XML Data Source

Use the XML tab to set or edit properties for an XML data source. The XML tab of the Physical Table dialog box provides the same functionality as the XML tab of the Connection Pool dialog box. However, setting properties in the Physical Table dialog box will override the corresponding settings in the Connection Pool dialog box. For more information, refer to "Setting Up Additional Connection Pool Properties for an XML Data Source" on page 76.

Creating Physical Layer Folders

This section contains the following topics:

- Creating Physical Layer Catalogs and Schemas on page 97
- Using a Variable to Specify the Name of a Catalog or Schema on page 97
- Setting Up Display Folders in the Physical Layer on page 98

Creating Physical Layer Catalogs and Schemas

Catalogs are optional ways to group different schemas. A catalog contains all the schema (metadata) for a database object. A schema contains only the metadata information for a particular user or application. Model the physical layer after the way your database is structured.

A database can have either catalogs or schemas but not both. If your database has one or more schemas, you cannot create a catalog. If your database has one or more catalogs, you cannot create schemas.

NOTE: You must create a database object before you create a catalog object or a schema object.

Creating Catalogs

In the Physical layer of a large repository, Oracle BI Administrators can create catalogs that contain one or more physical schemas.

To create a catalog

1. In the Physical layer, right-click a database object, and then choose New Object > Physical Catalog.
2. In the Physical Catalog dialog box, type a name for the catalog.
3. Type a description for the catalog, and then click OK.

Creating Schemas

The schema object contains tables and columns for a physical schema. Schema objects are optional in the Physical layer of the Administration Tool.

To create a schema

1. In the Physical layer, right-click a database object, and then choose New Object > Physical Schema.
2. In the Physical Schema dialog box, type a name.
3. Type a description for the schema, and then click OK.

Using a Variable to Specify the Name of a Catalog or Schema

You can use a variable to specify the names of catalog and schema objects. For example, you have data for multiple clients and you structured the database so that data for each client was in a separate catalog. You would initialize a session variable named Client, for example, that could be used to set the name for the catalog object dynamically when a user signs on to the Oracle BI Server.

NOTE: The Dynamic Name tab is not active unless at least one session variable is defined.

To specify the session variable to use in the Dynamic Name tab

1. In the Name column of the Dynamic Name tab, click the name of the session variable that you want to use. The initial value for the variable (if any) is shown in the Default Initializer column.
2. To select the highlighted variable, click Select.

   The name of the variable appears in the dynamic name field, and the Select button toggles to the Clear button.

To remove assignment for a session variable in the Dynamic Name tab

- Click Clear to remove the assignment for the variable as the dynamic name.

  The value Not Assigned displays in the dynamic name field, and the Clear button toggles to the Select button.

To sort column entries in the Dynamic Name tab

- You can sort the entries in a column by clicking on the associated column heading, Name or Default Initializer. Clicking on a column heading toggles the order of the entries in that column between ascending and descending order, according to the column type.

  When no dynamic name is assigned, Not Assigned displays in the dynamic name field to the left of the Select button. When a dynamic name is assigned, the Select button toggles to the Clear button, and the name of the variable displays in the dynamic name field.

Setting Up Display Folders in the Physical Layer

Oracle BI Administrators can create display folders to organize table objects in the Physical layer. They have no metadata meaning. After you create a display folder, the selected tables appear in the folder as a shortcut and in the Physical layer tree as an object. You can hide the objects so that you only view the shortcuts in the display folder. For more information about hiding these objects, refer to "Using the Options Dialog Box—Repository Tab" on page 31.

NOTE: Deleting objects in the display folder deletes only the shortcuts to those objects.

To set up a physical display folder

1. In the Physical layer, right-click a database object, and choose New Object > Physical Display Folder.
2. In the Physical Display Folder dialog box, in the Tables tab, type a name for the folder.
3. To add tables to the display folder, perform the following steps:
   a. Click Add.
   b. In the Browse dialog box, select the fact or physical tables you want to add to the folder and click Select.
4. Click OK.

About Physical Joins

All valid physical joins need to be configured in the Physical layer of the Administration Tool.

NOTE: You do not create joins for multidimensional data sources.

When you import keys in a physical schema, the primary key-foreign key joins are automatically defined. Any other joins within each database or between databases have to be explicitly defined to express relationships between tables in the physical layer.

NOTE: Imported key and foreign key joins do not have to be used in metadata. Joins that are defined to enforce referential integrity constraints can result in incorrect joins being specified in queries. For example, joins between a multipurpose lookup table and several other tables can result in unnecessary or invalid circular joins in the SQL issued by the Oracle BI Server.

Multi-Database Joins

A multi-database join is defined as a table under one metadata database object that joins to a table under a different metadata database object. You need to specify multi-database joins to combine the data from different databases. Edit the Physical Table Diagram window to specify multi-database joins. The joins can be between tables in any databases, regardless of the database type, and are performed within the Oracle BI Server. While the Oracle BI Server has several strategies for optimizing the performance of multi-database joins, multi-database joins will be significantly slower than joins between tables within the same database. It is recommended to avoid them whenever possible. For more information about the Physical Table Diagram, refer to "Defining Physical Joins in the Physical Diagram" on page 101.

Fragmented Data

Fragmented data is data from a single domain that is split between multiple tables. For example, a database might store sales data for customers with last names beginning with the letter A through M in one table and last names from N through Z in another table. With fragmented tables, you need to define all of the join conditions between *each* fragment and all the tables it relates to. Figure 9 on page 99 shows the physical joins with a fragmented sales table and a fragmented customer table where they are fragmented the same way (A through M and N through Z).

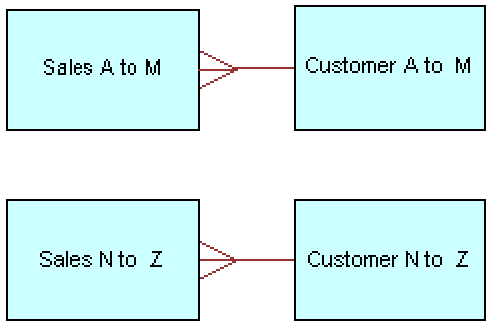


Figure 9. Fragmented Tables Example

In some cases, you might have a fragmented fact table and a fragmented dimension table, but the fragments might be across different values. In this case, you need to define all of the valid joins, as shown in Figure 10 on page 100.

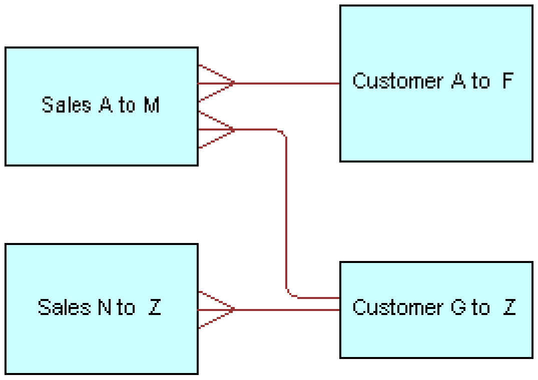


Figure 10. Joins for Fragmented Tables Example

TIP: Avoid adding join conditions where they are not necessary (for example, between Sales A to M and Customer N to Z in Figure 9 on page 99). Extra join conditions can cause performance degradations.

Primary Key and Foreign Key Relationships

A primary key and foreign key relationship defines a one-to-many relationship between two tables. A foreign key is a column or a set of columns in one table that references the primary key columns in another table. The primary key is defined as a column or set of columns where each value is unique and identifies a single row of the table. You can specify primary key and foreign keys in the Physical Table Diagram or by using the Keys tab and Foreign Keys tab of the Physical Table dialog box. Also refer to "Defining Physical Joins in the Physical Diagram" on page 101 and "Creating and Administering Columns and Keys in a Physical Table" on page 87.

Complex Joins

In the physical layer of the repository, complex joins are joins over nonforeign key and primary key columns. When you create a complex join in the physical layer, you can specify expressions and the specific columns on which to create the join. When you create a complex join in the business model layer, you do not specify expressions.

Defining Physical Foreign Keys and Joins

You can create physical foreign keys, complex joins, and logical joins using the Joins Manager or the Physical or Logical Table Diagram.

NOTE: You do not create joins for multidimensional data sources.

To define physical joins, refer to the following topics:

Defining Physical Foreign Keys or Complex Joins with the Joins Manager

You can use the Joins Manager to view join relationships and to create physical foreign keys and complex joins.

To create a physical foreign key or complex join

1. In the Administration Tool toolbar, select Manage > Joins.
2. In the Joins Manager dialog box, perform one of the following tasks:
   - Select Action > New > Complex Join.

     The Physical Complex Join dialog box appears.
   - Select Action > New > Foreign Key. In the Browse dialog box, double-click a table.
3. In the Physical Foreign Key dialog box, type a name for the foreign key.
4. Click the Browse button for the Table field on the left side of the dialog box, and then locate the table that the foreign key references.
5. Select the columns in the left table that the key references.
6. Select the columns in the right table that make up the foreign key columns.
7. If appropriate, specify a database hint.

   For more information, refer to "Using Database Hints" on page 106.
8. To open the Expression Builder, click the button to the right of the Expression pane.

   The expression displays in the Expression pane.
9. Click OK to save your work.

Defining Physical Joins in the Physical Diagram

You can define foreign keys and complex joins between tables, whether or not the tables are in the same database. If you click the Physical diagram icon on the toolbar, the Physical Table Diagram window opens and only the selected objects appear. If you right-click a physical object, several options are available. For more information about these options, refer to Table 15 on page 102.

To display the Physical Table Diagram

1. In the Administration Tool, in the Physical layer, right-click a table and choose Physical Diagram.
2. In the shortcut menu, choose an option from Table 15 on page 102.

3 To add another table to the Physical Table Diagram window, perform the following steps:

   a Leave the Physical Table Diagram window open.

   b Right-click to select a table you want to add and choose one of the Physical Diagram options described in Table 15 on page 102.

   Repeat this process until all the tables you need appear in the Physical Table Diagram window.

Table 15. Physical Diagram Shortcut Menu Options

Physical Diagram Menu	Description
Object(s) and all joins	Displays the selected objects, as well as each object that is related directly or indirectly to the selected object through some join path. If all the objects in a schema are related, then using this option diagrams every table, even if you only select one table.
Object(s) and direct joins	Displays the selected objects and any tables that join to the objects that you select.
Object(s) and direct joins in the business model	Option Not available at this time.
Selected object(s) only	Displays the selected objects only. Joins display only if they exist between the objects that you select.

To define a foreign key join or a complex join

1 In the Physical layer of the Administration Tool, select one or more tables and execute one of the Physical Diagram commands from the right-click menu.

2 Click one of the following icons on the Administration Tool toolbar:

   - New foreign key
   - New complex join

3 With this icon selected, in the Physical Table Diagram window, left-click the first table in the join (the table representing ***one*** in the one-to-many join) to select it.

4 Move the cursor to the table to which you want to join (the table representing ***many*** in the one-to-many join), and then left-click the second table to select it.

   The Physical Foreign Key or Physical Join dialog box appears.

5 Select the joining columns from the left and the right tables.

   The SQL join conditions appear in the expression pane of the window.

   NOTE: The driving table is shown on this window, but it is not available for selection because the Oracle BI Server implements driving tables only in the Business Model and Mapping layer. For more information about driving tables, refer to "Specifying a Driving Table" on page 140.

6 If appropriate, specify a database hint.

For more information, refer to "Using Database Hints" on page 106.

7 To open the Expression Builder, click the button to the right of the Expression pane.

The expression displays in the Expression pane.

8 Click OK to apply the selections.

Deploying Opaque Views

This section contains the following topics:

- About Deploying Opaque Views on page 103
- Deploying Opaque View Objects on page 103
- Undeploying a Deployed View on page 105
- Guidelines for Deleting an Opaque View or Deployed View on page 106
- Guidelines for Redeploying Opaque Views on page 106

About Deploying Opaque Views

An opaque view is a physical layer table that consists of a Select statement. When you need a new table, you should create a physical table or a materialized view. An opaque view should be used only if there is no other solution.

In the repository, opaque views appear as view tables in the physical databases but the view does not actually exist. You deploy an opaque view in the physical database using the Deploy View(s) utility. After deploying an opaque view, it is called a deployed view. Opaque views can be used without deploying them but the Oracle BI Server has to generate a more complex query when an opaque view is encountered.

NOTE: Databases such as XLS and nonrelational database do not support opaque views and, therefore, cannot run the view deployment utility.

To verify opaque views are supported by a database, make sure that the CREATE_VIEW_SUPPORTED SQL feature is selected in the Database dialog box, in the Features tab. For instructions, refer to "Specifying SQL Features Supported by a Database" on page 64.

Deploying Opaque View Objects

In offline mode, the Deploy View(s) utility is available when importing from databases with ODBC and DB2cli data sources. Oracle Native (client) drivers are also supported in the offline mode for deploying views. In online mode, view deployment is available for supported databases using Import through server (the settings on the client will be ignored).

The Create View Select Statement

The SQL statement for deploying opaque views in the physical layer of the repository is available for supported databases. To determine which of your databases support opaque views, contact your system administrator or consult your database documentation.

Only repository variables can be used in the definition. An error will generate if a session variable is used in the view definition.

Syntax:

```
    CREATE VIEW <view name> AS <select statement>,
```

where

<select statement>	The user-entered SQL in the opaque view object. If SQL is invalid, the create view statement will fail during view deployment.
<view name>	Two formats: schema.viewname, or viewname. The connection pool settings determine if the schema name is added.

For opaque view objects, the right-click menu contains the Deploy View(s) option. When you select Deploy View(s), the Create View SQL statement executes and attempts to create the deployed view objects. The following list describes the ways you can initiate view deployment and the results of each method:

- Right-click a single opaque view object. When you select Deploy View(s), the Create View SQL statement executes and attempts to create a deployed view for the object.
- Right-click several objects. If at least one of the selected objects is an opaque view object, the right-click menu contains the Deploy View(s) option. When you select Deploy View(s), the Create View SQL statement executes and attempts to create the deployed views for any qualifying objects.
- Right-click a physical schema or physical catalog. If any opaque view object exists in the schema or catalog, the right-click menu contains the Deploy View(s) option. When you select Deploy View(s), the Create View SQL statements for all qualifying objects execute and attempt to create deployed views for the qualifying objects contained in the selected schema or catalog.

During deployment, names are assigned to the views. If you change the preassigned name, the new name must be alphanumeric and no more than 18 characters. If these guidelines are not followed, the object name will be automatically transformed to a valid name using the following Name Transform algorithm:

1. All non-alphanumeric characters will be removed.
2. If there are 16 or more after Step 1 on page 104, the first 16 characters will be kept.
3. Two digits starting from 00 to 99 will be appended to the name to make the name unique in the corresponding context.

After the deployment process completes, the following occurs:

- Views that have been successfully and unsuccessfully deployed appear in a list.
- For unsuccessful deployments, a brief reason appears in the list.

- If deployment is successful, the object type of the opaque view changes from Select to None and the deployed view will be treated as a regular table.

  NOTE: If you change the type back to Select, the associated opaque views will be dropped from database or an error message will appear. For information about deleting deployed views, refer to "Guidelines for Deleting an Opaque View or Deployed View" on page 106

- In the Administration Tool, the view icon changes to the deployed view icon for successfully deployed views.

To deploy an opaque view

1. In the Physical layer of the repository, right-click the opaque view that you want to deploy.
2. In the right-click menu, choose Deploy View(s).
3. In the View Deployment - Deploy View(s) dialog box, perform the following steps:
   a. In the New Table Name column, change the new deployed view names, if you wish.

      If the change does not conform to the naming rules a new name will be assigned and the dialog box appears again so that you can accept or change it. This action will repeat until all names pass validation.
   b. If you do not wish to deploy one or more of the views at this time, in the appropriate rows, clear the check boxes.
4. If there are multiple connection pools defined for the database, in the Select Connection Pool dialog box, choose a connection pool and click Select.

   The SQL statement (CREATE VIEW) executes, and then the View Deployment Messages dialog box appears.
5. In the View Deployment Messages dialog box, you can search for views using Find and Find Again or copy the contents.
6. When you have performed the desired tasks, click OK.

Undeploying a Deployed View

Running the Undeploy View(s) utility against a deployed view deletes the views and converts the table back to an opaque view with its original SELECT statement.

To undeploy a deployed view

1. In the Physical layer of the repository, right-click a database, physical catalog, schema, or physical table.

   If a deployed view exists that is related to the selected object, the right-click menu contains the Undeploy View(s) option.
2. Choose Undeploy View(s).

   A list of views that will be undeployed appears.

3 If you do not wish to undeploy one or more of the views at this time, in the appropriate rows, clear the check boxes.

4 View Deployment - Undeploy View(s) dialog box, click OK to remove the views.

   A message appears if the undeployment was successful.

5 In the View Deployment Messages dialog box, you can search for undeployed views using Find and Find Again, or you can copy the contents.

6 When you have performed the desired tasks, click OK.

Guidelines for Deleting an Opaque View or Deployed View

Use the following guidelines to remove an opaque or deployed view object in the repository:

- **Removing an undeployed opaque view in the repository.** If the opaque view has not been deployed, you can delete it from the repository.
- **Removing a deployed view.** When you deploy an opaque view, a database view table is created physically in the back-end database and the repository. Therefore, you must undeploy the view before deleting it. You use the Undeploy View(s) utility in the Administration Tool. This removes the opaque view from the back-end database, changes the Object Type from None to Select, and restores the SELECT statement of the object in the physical layer of repository.

  CAUTION: You should not delete the physical database view in the back-end database. If deleted, the Oracle BI Server will not be able to query the view object. When you undeploy the view it is removed automatically from the back-end database.

Guidelines for Redeploying Opaque Views

After removing an opaque view, you can choose to redeploy it. The Administration Tool does not distinguish between a first-time deployment and a redeployment. Make sure that you remove a deployed view before deploying the opaque view again. Failure to do this causes the deploy operation to fail, and an error message will be returned from the database.

Using Database Hints

Database hints are instructions placed within a SQL statement that tell the database query optimizer the most efficient way to execute the statement. Hints override the optimizer’s execution plan, so you can use hints to improve performance by forcing the optimizer to use a more efficient plan.

NOTE: Hints are database specific. The Oracle BI Server supports hints only for Oracle 8i, 9i, and 10g servers.

Using the Administration Tool, you can add hints to a repository, in both online and offline modes, to optimize the performance of queries. When you add a hint to the repository, you associate it with database objects. When the object associated with the hint is queried, the Oracle BI Server inserts the hint into the SQL statement.

Table 16 on page 107 shows the database objects with which you can associate hints. It also shows the Administration Tool dialog box that corresponds to the database object. Each of these dialog boxes contains a Hint field, into which you can type a hint to add it to the repository.

Table 16. Database Objects That Accept Hints

Database Object	Dialog Box
Physical complex join	Physical Join - Complex Join
Physical foreign key	Physical Foreign Key
Physical table - object type of Alias	Physical Table - General tab
Physical table - object type of None	Physical Table - General tab

Usage Examples

This section provides a few examples of how to use Oracle hints in conjunction with the Oracle BI Server. For more information about Oracle hints, refer to the Oracle SQL Reference documentation for the version of the Oracle server that you use.

Index Hint

The Index hint instructs the optimizer to scan a specified index rather than a table. The following hypothetical example explains how you would use the Index hint. You find queries against the ORDER_ITEMS table to be slow. You review the query optimizer's execution plan and find the FAST_INDEX index is not being used. You create an Index hint to force the optimizer to scan the FAST_INDEX index rather than the ORDER_ITEMS table. The syntax for the Index hint is *index(table_name, index_name)*. To add this hint to the repository, navigate to the Administration Tool's Physical Table dialog box and type the following text in the Hint field:

```
index(ORDER_ITEMS, FAST_INDEX)
```

Leading Hint

The Leading hint forces the optimizer to build the join order of a query with a specified table. The syntax for the Leading hint is *leading(table_name)*. If you were creating a foreign key join between the Products table and the Sales Fact table and wanted to force the optimizer to begin the join with the Products table, you would navigate to the Administration Tool's Physical Foreign Key dialog box and type the following text in the Hint field:

```
leading(Products)
```

Performance Considerations

Hints that are well researched and planned can result in significantly better query performance. However, hints can also negatively affect performance if they result in a suboptimal execution plan. The following guidelines are provided to help you create hints to optimize query performance:

- You should only add hints to a repository after you have tried to improve performance in the following ways:

  - Added physical indexes (or other physical changes) to the Oracle database.
  - Made modeling changes within the server.
- Avoid creating hints for physical table and join objects that are queried often.

NOTE: If you drop or rename a physical object that is associated with a hint, you must also alter the hints accordingly.

Creating Hints

The following procedure provides the steps to add hints to the repository using the Administration Tool.

To create a hint

1. Navigate to one of the following dialog boxes:
   - Physical Table—General tab
   - Physical Foreign Key
   - Physical Join—Complex Join
2. Type the text of the hint in the Hint field and click OK.

   For a description of available Oracle hints and hint syntax, refer to *Oracle8i SQL Reference*.

   NOTE: Although hints are identified by using SQL comment markers (/* or --), do not type SQL comment markers when you type the text of the hint. The Oracle BI Server inserts the comment markers when the hint is executed.

5 Creating and Administering the Business Model and Mapping Layer in an Oracle BI Repository

The following topics contain information about creating the business model and logical objects:

About Creating the Business Model and Mapping Layer

This section is part of the roadmap for planning and setting up a repository. For more information, refer to "Planning and Creating an Oracle BI Repository" on page 39.

After creating all of the elements of the Physical layer, you can drag tables or columns from the Physical layer to the Business Model and Mapping layer. For more information, refer to "Creating and Administering the Physical Layer in an Oracle BI Repository" on page 55 and "Creating the Business Model Layer for a Multidimensional Data Source" on page 110.

The Business Model and Mapping layer of the Administration Tool defines the business, or logical, model of the data and specifies the mapping between the business model and the physical layer schemas.

You create one or more business models in the logical layer and create logical tables and columns in each business model. To automatically map objects in the Business Model and Mapping layer to sources in the Physical layer, you can drag and drop Physical layer objects to a business model in the logical layer. When you drag a physical table to the Business Model and Mapping layer, a corresponding logical table is created. For each physical column in the table, a corresponding logical column is created. If you drag multiple tables at once, a logical join is created for each physical join, but only the first time the tables are dragged onto a new business model.

Creating the Business Model Layer for a Multidimensional Data Source

Setting up the Business Model and Mapping (logical) layer for multidimensional data sources is similar to setting up the logical layer for a relational data source. To create the business model layer, you can drag and drop the physical layer cube to the logical layer. However, because the contents of the physical cube are added as one logical table, you still have to reorganize the columns into appropriate logical tables and recreate the hierarchies.

Creating Business Model Objects

The Business Model and Mapping layer of the Administration Tool can contain one or more business model objects. A business model object contains the business model definitions and the mappings from logical to physical tables for the business model.

NOTE: When you work in a repository in offline mode, remember to save your repository from time to time. You can save a repository in offline mode even though the business models may be inconsistent.

To create a business model

1. Right-click in the Business Model and Mapping layer below the existing objects.
2. Select the option New Business Model from the shortcut menu.
3. Specify a name for the business model.
4. If you wish to make the corresponding presentation layer available for queries, select the option Available for queries.

   NOTE: The business model should be consistent before you make select this option.

5. (Optional) Type a description of the business model, and then click OK.

Duplicate Business Model and Presentation Catalog

This allows you to select a matching business model and presentation catalog, make a copy, and assign new names to the duplicates.

NOTE: Aliases are not copied.

To copy a business model and its presentation catalog

1. In the Business Model and Mapping layer, right-click a business model.
2. In the right-click menu, choose Duplicate with Presentation Catalog.
3. In the Copy Business Model and Presentation Catalog dialog box, select the business model to copy.

4 Specify new names for the business model and its catalog in the appropriate name fields, and then click OK.

The copied business model appears in the Business Model and Mapping layer window.

Creating and Administering Logical Tables

Logical tables exist in the Business Model and Mapping layer. The logical schema defined in each business model needs to contain at least two logical tables and you need to define relationships between them.

Each logical table has one or more logical columns and one or more logical table sources associated with it. You can change the logical table name, reorder the logical table sources, and configure the logical keys (primary and foreign).

This section includes the following topics:

- Creating Logical Tables on page 111
- Specifying a Primary Key in a Logical Table on page 112
- Reviewing Foreign Keys for a Logical Table on page 113

Creating Logical Tables

Typically, you create logical tables by dragging and dropping a physical table from the Physical layer to a business model in the Business Model and Mapping layer. If a table does not exist in your physical schema, you would need to create the logical table manually.

Drag and drop operations are usually the fastest method for creating objects in the Business Model and Mapping layer. If you drag and drop physical tables from the Physical layer to the Business Model and Mapping layer, the columns belonging to the table are also copied. After you drag and drop objects into the Business Model and Mapping layer, you can modify them in any way necessary without affecting the objects in the Physical layer.

When you drag physical tables (with key and foreign key relationships defined) to a business model, logical keys and joins are created that mirror the keys and joins in the physical layer. This occurs only if the tables that you drag include the table with the foreign keys. Additionally, if you create new tables or subsequently drag additional tables from the Physical layer to the Business Model and Mapping layer, the logical links between the new or newly dragged tables and the previously dragged tables must be created manually.

For more information about joins, refer to “Defining Logical Joins with the Joins Manager” on page 137 and “Defining Logical Joins with the Business Model Diagram” on page 139.

To create a logical table by dragging and dropping

1 Select one or more table objects in the Physical layer.

  You must include the table with the foreign keys if you want to preserve the keys and joins from the physical layer.

2 Drag and drop the table objects to a business model in the Business Model and Mapping layer.

  When you drop them, the table objects, including the physical source mappings, are created automatically in the Business Model and Mapping layer.

To create a logical table manually

1 In the Business Model and Mapping layer, right-click the business model in which you want to create the table and select New Object > Logical Table.

  The Logical Table dialog box appears.

2 In the General tab, type a name for the logical table.

3 If this is a bridge table, select the option Bridge table.

  For more information, refer to "Identifying Dimension Hierarchies" on page 43.

4 (Optional) Type a description of the table.

5 Click OK.

NOTE: After creating a logical table manually, you must create all keys and joins manually.

Adding or Editing Logical Table Sources

You can add a new logical table source, edit or delete an existing table source, create or change mappings to the table source, and define when to use logical tables sources and how content is aggregated. For instructions about how to perform these tasks, refer to "Creating and Administering Logical Table Sources (Mappings)" on page 117.

Specifying a Primary Key in a Logical Table

After creating tables in the Business Model and Mapping layer, you specify a primary key for each table. Logical dimension tables must have a logical primary key. Logical keys can be composed of one or more logical columns.

NOTE: Logical keys are optional for logical fact tables. However, it is recommended that you do not specify logical keys for logical fact tables. For more information, refer to "Reviewing Foreign Keys for a Logical Table" on page 113.

To specify a primary key in a logical table

1 In the Business Model and Mapping layer, double-click a table.

2 In the Logical Table dialog box, select the Keys tab and then click New.

3 In the Logical Key dialog box, perform the following steps:

  a Type a name for the key.

  b Select the check box for the column that defines the key of the logical table.

4 Click OK.

Reviewing Foreign Keys for a Logical Table

You can use the Foreign Keys tab to review the foreign keys for a logical table.

In fact tables, it is recommended that you use complex logical joins instead of foreign key logical joins. If complex logical joins are used, then there is more flexibility in defining the primary key. If the physical table has a primary key, then this field can be used as a logical key for the fact table. This is the method recommended for the Oracle BI repository.

CAUTION: It is recommended that you do not have foreign keys for logical tables. However, you can create logical foreign keys and logical complex joins using either the Joins Manager or the Business Model Diagram. A logical key for a fact table must be made up of the key columns that join to the attribute tables. For more information, refer to "Defining Logical Joins" on page 136.

To review foreign key information for a logical table

1 In the Business Model and Mapping layer, double-click a table.

2 In the Logical Table dialog box, select the Foreign Keys tab.

3 To review an existing foreign key, in the Foreign Keys list, select a key and click Edit.

  The Logical Foreign Key dialog box appears. For more information about changing information in this dialog box, refer to "Defining Logical Joins" on page 136.

Creating and Administering Logical Columns

Many logical columns are automatically created by dragging tables from the Physical layer to the Business Model and Mapping layer. Other logical columns, especially ones that involve calculations based on other logical columns, can be created later.

Logical columns are displayed in a tree structure expanded out from the logical table to which they belong. If the column is a primary key column or participates in a primary key, the column is displayed with the key icon. If the column has an aggregation rule, it is displayed with a sigma icon. You can reorder logical columns in the Business Model and Mapping layer.

This section includes the following topics:

- Creating and Moving a Logical Column
- Setting Default Levels of Aggregation for Measure Columns on page 115
- Associating an Attribute with a Logical Level in Dimension Tables on page 116

Creating and Moving a Logical Column

Use the General tab to create or edit the general properties of a logical column. You can create a logical column object in the Business Model and Mapping layer, and then drag and drop it to the Presentation layer.

About Sorting on a Logical Column

For a logical column, you can specify a different column on which to base the sort. This changes the sort order of a column when you do not want to order the values *lexicographically*. Lexicographical sort arranges the results in alphabetic order such as in a dictionary. In this type of sort, numbers are ordered by their alphabetic spelling and not divided into a separate group.

For example, if you sorted on month (using a column such as MONTH_NAME), the results would be returned as February, January, March, and so on, in lexicographical sort order. However, you might want months to be sorted in chronological order. Therefore, your table should have a month key (such as MONTH_KEY) with values of 1 (January), 2 (February), 3 (March), and so on. To achieve the desired sort, you set the Sort order column field for the MONTH_NAME column to be MONTH_KEY. Then a request to order by MONTH_NAME would return January, February, March, and so on.

To create a logical column

1. In the Business Model and Mapping layer, right-click a logical table.
2. From the shortcut menu, select New Object > Logical Column.
3. In the Logical Column dialog box, select the General tab.
4. In the General tab, type a name for the logical column.

   The name of the business model and the associated logical table appear in the Belongs to Table field.

5. (Optional) If you want to assign a different column on which to base the sort order for a column, perform the following steps:
   a. Next to the Sort order column field, click Set.
   b. In the Browse dialog box, select a column
   c. To view the column details, click View to open the Logical Column dialog box for that column, and then click Cancel.

      NOTE: You can make some changes in this dialog box. If you make changes, click OK to accept the changes instead of Cancel.

   d. In the Browse dialog box, Click OK.
6. (Optional) To remove the Sort order column value, click Clear.
7. (Optional) If you want the logical column to be derived from other logical columns, perform the following steps:
   a. Select the check box for Use existing logical columns as source.
   b. Click the ellipsis button next to the text box to open the Expression Builder.

c. In the Expression Builder - Derived logical column dialog box, specify the expression from which the logical column should be derived.

d. Click OK.

8. (Optional) In the Logical Column dialog box, type a description of the logical column.

   NOTE: The type and length fields are populated automatically based upon the column's source.

9. Click OK.

To move or copy logical columns

1. In the Business Model and Mapping layer, drag and drop a logical column to a different logical table.

   NOTE: You can select multiple columns to move.

2. In the Sources for moved columns dialog box, in the Action area, select an action.
3. If you select Ignore, no logical source will be added in the Sources folder of the destination table.
4. If you select Create new, a copy of the logical source associated with the logical column will be created in the Sources folder of the destination table.
5. If you select Use existing, in the Use existing drop-down list, you must select a logical source from the Sources folder of the destination table.

   The column that you moved or copied will be associated with this logical source.

Setting Default Levels of Aggregation for Measure Columns

You need to specify aggregation rules for mapped logical columns that are measures. Aggregation should only be performed on measure columns, with the possible exception of the aggregation COUNT and Count Distinct. Measure columns should exist only in logical fact tables.

NOTE: For multidimensional logical columns, you can set the default aggregation rule to Aggr_External.

You can specify an override aggregation expression for specific logical table sources. This helps the Oracle BI Server take advantage of aggregate tables when the default aggregation rule is Count Distinct. If you do not specify any override, then the default rule prevails.

By default, data is considered sparse. However, on rare occasions you might have logical table source with dense data. A logical table source is considered to have dense data if it has row for every combination of its associated dimension levels. When setting up a aggregate rules for a measure column, you can specify that data is dense only if all the logical table sources to which it is mapped are dense.

CAUTION: Specifying data is dense when any table source that is used by this column does not contain dense data will return incorrect results.

To specify a default aggregation rule for a measure column

1 In the Business Model and Mapping layer, double-click a logical column.

2 In the Logical Column dialog box, click the Aggregation tab.

3 In the Aggregation tab, complete the following fields.

   - If the aggregation rule will be based on a time dimension, select the Based on dimensions check box.

     The Data is dense check box appears.

   - If all the logical table sources to which this column is mapped are dense, you should select the Data is dense check box.

     CAUTION: Selecting this check box indicates that all sources to which this column is mapped have a row for every combination of dimension levels that they represent. Checking this box when any table source that is used by this column does not contain dense data will return incorrect results.

   - Select one of the aggregate functions from the Default Aggregation Rule drop-down list.

     The function you select is always applied when an end user or an application requests the column in a query.

4 Click OK.

Associating an Attribute with a Logical Level in Dimension Tables

Attributes can be associated with a logical level by selecting the dimensional level on the Levels tab. Measures can be associated with levels from multiple dimensions and will always aggregate to the levels specified.

Dimensions appear in the Dimensions list. If this attribute is associated with a logical level, the level appears in the Levels list.

Another way to associate a measure with a level in a dimension is to expand the dimension tree in the Business Model and Mapping layer, and then use drag-and-drop to drop the column on the target level. For more information about level-based measures, refer to "Level-Based Measure Calculations Example" on page 131.

To associate a measure with a logical level in a dimension

1 In the Business Model and Mapping layer, double-click a logical column.

2 In the Logical Column dialog box, click the Levels tab.

3 In the Levels tab, click the Logical Levels field for the dimension from which you want to select a logical level.

   NOTE: In the Levels tab, in the levels list, you can sort the rows (toggle between ascending order and descending order) by clicking a column heading.

4 In the Logical Levels drop-down list, select the level.

5 Repeat this process to associate this measure with other logical levels in other dimensions.

To remove the association between a dimension and a measure

1 In the Business Model and Mapping layer, double-click a logical column.

2 In the Logical Column dialog box, click the Levels tab.

3 In the Levels tab, click the delete button next to the Logical Levels field.

4 Click OK.

Creating and Administering Logical Table Sources (Mappings)

You can add a new logical table source, edit or delete an existing table source, create or change mappings to the table source, and define when to use logical tables sources and how content is aggregated. Additionally, you can copy aggregation content to the Windows clipboard or from another logical table source, and check the aggregation content of logical fact table sources.

You would add new logical table sources when more than one physical table could be the source of information. For example, many tables could hold information for revenue. You might have three different business units (each with its own order system) where you get revenue information. In another example, you might periodically summarize revenue from an orders system or a financial system and use this table for high-level reporting.

One logical table source folder exists for each logical table. The folder contains one or more logical table sources. These sources define the mappings from the logical table to the physical table. Complex mappings, including formulas, are also configured in the logical table sources.

Logical tables can have many physical table sources. A single logical column might map to many physical columns from multiple physical tables, including aggregate tables that map to the column if a query asks for the appropriate level of aggregation on that column.

When you create logical tables and columns by dragging and dropping from the Physical layer, the logical table sources are generated automatically. If you create the logical tables manually, you need to also create the sources manually.

For examples of how to set up fragmentation content for aggregate navigation, refer to "Specify Fragmentation Content" on page 201.

This section includes the following topics:

- Creating or Removing a Logical Table Source on page 118
- Defining Physical to Logical Table Source Mappings on page 119
- Defining Content of Logical Table Sources on page 121

Creating or Removing a Logical Table Source

Use the General tab of the Logical Table Source dialog box to define general properties for the logical table source.

To create a logical table source

1. In the Business Model and Mapping layer, right-click a logical table and choose New Object > Logical Table Source.
2. In the Logical Table Source dialog box, click the General tab, and then type a name for the logical table source and click Add.
3. In the Browse dialog box, you can view joins and select tables for the logical table source.

   When there are two or more tables in a logical table source, all of the participating tables must have joins defined between them.
4. To view the joins, in the Browse dialog box, select a table and click View.
   - In the Physical Table dialog box, review the joins, and then click Cancel.
5. To add tables to the table source, select the tables in the Name list and click Select.
6. In the Logical Table Source dialog box, click the Column Mapping tab and complete the fields using the instructions in "Defining Physical to Logical Table Source Mappings" on page 119.
7. In the Logical Table dialog box, click the Content tab and complete the fields using the instructions in "Defining Content of Logical Table Sources" on page 121.
8. Click OK.

To remove a table as a source

1. In the Business Model and Mapping layer, right-click a logical table source and choose Properties.
2. In the Logical Table Source dialog box, click the General tab.
3. In the tables list, select the table you want to remove and click Remove.
4. After removing the appropriate table, click OK.

Example of Creating Sources for Each Level of Aggregated Fact Data

In addition to creating the source for the aggregate fact table, you should create corresponding logical dimension table sources at the same levels of aggregation.

NOTE: You need to have at least one source at each level referenced in the aggregate content specification. If the sources at each level already exist, you do not need to create new ones.

For example, you might have a monthly sales table containing a precomputed sum of the revenue for each product in each store during each month. You need to have the following three other sources, one for each of the logical dimension tables referenced in the example:

- A source for the Product logical table with one of the following content specifications:

    - By logical level: ProductDimension.ProductLevel
    - By column: Product.Product_Name
- A source for the Store logical table with one of the following content specifications:
    - By logical level: StoreDimension.StoreLevel
    - By column: Store.Store_Name
- A source for the Time logical table with one of the following content specifications:
    - By logical level: TimeDimension.MonthLevel
    - By column: Time.Month

Defining Physical to Logical Table Source Mappings

Use the Column Mapping tab of the Logical Table Source dialog box to map logical columns to physical columns. The physical to logical mapping can also be used to specify transformations that occur between the Physical layer and the Business Model and Mapping layer. The transformations can be simple, such as changing an integer data type to a character, or more complex, such as applying a formula to find a percentage of sales per unit of population.

To map logical columns to physical columns

1. In the Business Model and Mapping layer, double-click a logical table source, if the Logical Table Source dialog box is not already open.
2. In the Logical Table Source dialog box, click the Column Mapping tab.
3. In the Column Mapping tab, maximize or enlarge the dialog box to show all the contents, as shown in Figure 11 on page 120.

   NOTE: In the Column Mapping tab, in the Logical column to physical column area, you can sort the rows (toggle among ascending order, descending order, and then restore original order) by clicking a column heading.

4. In the Physical Table column, select the table that contains the column you want to map.

   When you select a cell in the Physical Table column, a drop-down list appears. It contains a list of tables currently included in this logical table source.

5. In the Expression list, select the physical column corresponding to each logical column.

   When you select a cell in the Expression column, a drop-down list appears. It contains a list of tables currently included in this logical table source.

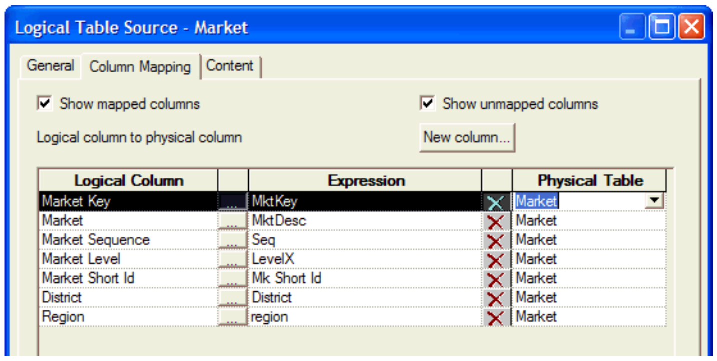
6. To open the Expression Builder, click the ellipsis button to the left of the Expression you want to view or edit.

   NOTE: All columns used in creating physical expressions must be in tables included in the logical table source. You cannot create expressions involving columns in tables outside the source.

7 To remove a column mapping, click the delete button.

  You might need to scroll to the right to locate the delete button.

8 After you map the appropriate columns, click OK.


Logical Table Source - Market

General | Column Mapping | Content

☑ Show mapped columns ☑ Show unmapped columns

Logical column to physical column [New column...]

Logical Column	Expression	Physical Table
Market Key	MktKey	Market
Market	MktDesc	Market
Market Sequence	Seq	Market
Market Level	LevelX	Market
Market Short Id	Mk Short Id	Market
District	District	Market
Region	region	Market

Figure 11. Logical Table Source Dialog Box

To remove a column mapping

1 In the Business Model and Mapping layer, right-click a logical table and choose New Object > Logical Table Source.

2 In the Logical Table Source dialog box, click the Column Mapping tab.

3 In the Column Mapping tab, maximize or enlarge the dialog box to show all the contents, as shown in Figure 11 on page 120.

4 To remove a column mapping, click the delete button next to the Physical Table cell.

5 Click OK.

Unmapping a Logical Column from Its Source

In the Logical Column dialog box, the Datatype tab contains information about the logical column. You can edit the logical table sources from which the column derives its data, or unmap it from its sources.

To unmap a logical column from its source

1 In the Business Model and Mapping layer, double-click a logical column.

2 In the Logical Column dialog box, click the Datatype tab.

3 In the Datatype tab, in the Logical Table Source list, select a source and click Unmap.

4 Click OK.

Defining Content of Logical Table Sources

To use a source correctly, the Oracle BI Server has to know what each source contains in terms of the business model. Therefore, you need to define aggregation content for each logical table source of a fact table. The aggregation content rule defines at what level of granularity the data is stored in this fact table. For each dimension that relates to this fact logical table, define the level of granularity, making sure that every related dimension is defined. For more information, refer to "Example of Creating Sources for Each Level of Aggregated Fact Data" on page 118.

If a logical table is sourced from a set of fragments, it is not required that every individual fragment maps the same set of columns. However, the server returns different answers depending on how columns are mapped.

- If all the fragments of a logical table map the same set of columns, than the set of fragmented sources is considered to be the whole universe of logical table sources for the logical table. This means that measure aggregations can be calculated based on the set of fragments.
- If the set of mapped columns differ across the fragments, than we assume that we do not have the whole universe of fragments, and therefore it would be incorrect to calculate aggregate rollups (since some fragments are missing).

  In this case we return NULL as measure aggregates.

NOTE: It is recommended that all the fragments map the same set of columns

Use the Content tab of the Logical Table Source dialog box to define any aggregate table content definitions, fragmented table definitions for the source, and Where clauses (if you want to limit the number of rows returned).

NOTE: For examples of how to set up fragmentation content for aggregate navigation, refer to "Setting Up Fragmentation Content in an Oracle BI Repository for Aggregate Navigation" on page 201.

Verify Joins Exist From Dimension Tables to Fact Table

This source content information tells the Oracle BI Server what it needs to know to send queries to the appropriate physical aggregate fact tables, joined to and constrained by values in the appropriate physical aggregate dimension tables. Be sure that joins exist between the aggregate fact tables and the aggregate dimension tables in the Physical layer.

One recommended way to verify joins is to select a fact logical table and request a Business Model Diagram (Selected Tables and Direct Joins). Only the dimension logical tables that are directly joined to this fact logical table appear in the diagram. It does not show dimension tables if the same physical table is used in logical fact and dimension sources.

Figure 12 on page 122 is an example of how the Fact - Asset fact logical table appears in a Business Model Diagram (Selected Tables and Direct Joins) view.

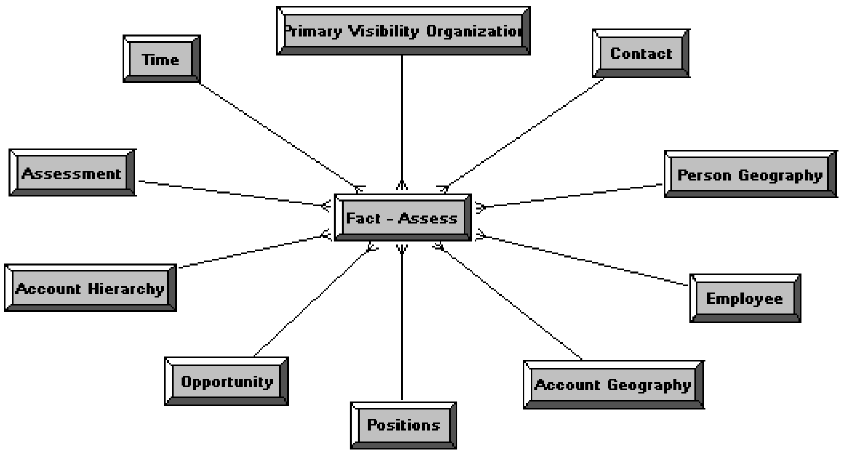


Figure 12. Diagram of Direct Joins for a Fact Table

Table 17 on page 122 contains a list of the logical level for each dimension table that is directly joined the Fact - Assess fact table shown in Figure 12 on page 122.

Table 17. Dimension and Logical Level as Shown in Content Tab

Dimension	Logical Level
Account Geography	Postal Code Detail
Person Geography	Postal Code Detail
Time	Day Detail
Account Organization	Account Detail
Opportunity	Opty Detail
Primary Visibility Organization	Detail
Employee	Detail
Assessment	Detail
Contact (W_PERSON_D)	Detail
FINS Time	Day
Positions	Details

To create logical table source content definitions

1 In the Business Model and Mapping layer, double-click a logical table source.

2. In the Logical Table Source dialog box, click the Content tab and perform the following steps using Table 18 on page 124 as a guide.
3. If a logical source is an aggregate table and you have defined logical dimensions, perform the following steps:
   a. Select Logical Level from the Aggregation content, group-by drop-down list.

      CAUTION: Although you have the option to specify aggregate content by logical level or column, it is recommended that you use logical levels exclusively.

   b. In the Logical Level drop-down list, select the appropriate level for each dimension logical table to which the fact logical table is joined.

      You should specify a logical level for each dimension, unless you are specifying the Grand Total level. Dimensions with no level specified will be interpreted as being at the Grand Total level.
4. If a logical source is an aggregate table and you want to define content for columns, do the following:
   a. Select Column from the Aggregation content, group-by drop-down list.

      CAUTION: Although you have the option to specify aggregate content by logical level or column, it is recommended that you use logical levels exclusively.

   b. In the Table pane, select each logical dimension table that defines the aggregation level of the source.
   c. In the Column pane, select the logical column for each dimension that defines how the aggregations were grouped.

      When there are multiple logical columns that could be used, select the one that maps to the key of the source physical table. For example, if data has been aggregated to the Region logical level, pick the logical column that maps to the key of the Region table.

      NOTE: Do not mix aggregation by logical level and column in the same business model. It is recommended that you use aggregation by logical level.
5. To specify fragmented table definitions for the source, use the Fragmentation content window to describe the range of values included in the source when a source represents a portion of the data at a given level of aggregation.

   You can type the formula directly into the window, or click the Expression Builder button to the right of the window. In the Fragmentation Content Expression Builder, you can specify content in terms of existing logical columns. For examples of how to set up fragmentation content for aggregate navigation, see"Specify Fragmentation Content" on page 201.
6. Select the following option:

   This source should be combined with other sources at this level

   NOTE: This option is only for multiple sources that are at the same level of aggregation.
7. (Optional) Specify Where clause filters in the Where Clause Filter window to limit the number of rows the source uses in the resultant table. For more information, refer to "About WHERE Clause Filters" on page 125.

a Click the Expression Builder button to open the Physical Where Filter Expression Builder.

b Type the Where clause and click OK.

8 Select the option Select distinct values if the values for the source are unique.

Table 18. Content Tab Fields for Logical Table Source

Field	Description
Aggregation content, group by	How the content is aggregated.
Fragmentation content	A description of the contents of a data source in business model terms. Data is fragmented when information at the same level of aggregation is split into multiple tables depending on the values of the data. A common situation would be to have data fragmented by time period. For examples of how to set up fragmentation content for aggregate navigation, see“Specify Fragmentation Content” on page 201.
More (button)	When you click More, the following options appear: ■ **Copy.** (Available only for fact tables) Copies aggregation content to the Windows clipboard. You can paste the Dimension.Level info into a text editor and use it for searching or for adding to documentation. ■ **Copy from.** (Available for fact tables and dimension tables) Copies aggregation content from another logical table source in the same business model. You need to specify the source from which to copy the aggregation content. (Multiple business models appear but only the logical table sources from the current business model are selectable.) ■ **Get Levels.** (Available only for fact tables) Changes aggregation content. If joins do not exist between fact table sources and dimension table sources (for example, if the same physical table is in both sources), the aggregation content determined by the administration tool will not include the aggregation content of this dimension. ■ **Check Levels.** (Available only for fact tables) check the aggregation content of logical fact table sources (not dimension table sources). The information returned depends on the existence of dimensions and hierarchies with logical levels and level keys, and physical joins between tables in dimension table sources and the tables in the fact table source. (If the same tables exist in the fact and dimension sources and there are no physical joins between tables in the sources, Check Levels will not include the aggregation content of this dimension.)

Table 18. Content Tab Fields for Logical Table Source

Field	Description
Select distinct values	Used if the values for the source are unique.
This source should be combined with other sources at this level (check box)	Check this box when data sources at the same level of aggregation do not contain overlapping information. In this situation, all sources must be combined to get a complete picture of information at this level of aggregation.

About WHERE Clause Filters

The WHERE clause filter is used to constrain the physical tables referenced in the logical table source. If there are no constraints on the aggregate source, leave the WHERE clause filter blank.

Each logical table source should contain data at a single intersection of aggregation levels. You would not want to create a source, for example, that had sales data at both the Brand and Manufacturer levels. If the physical tables include data at more than one level, add an appropriate WHERE clause constraint to filter values to a single level.

Any constraints in the WHERE clause filter are made on the physical tables in the source.

About Dimensions and Hierarchical Levels

In a business model, a dimension represents a hierarchical organization of logical columns (attributes) belonging to a single logical dimension table. Common dimensions might be time periods, products, markets, customers, suppliers, promotion conditions, raw materials, manufacturing plants, transportation methods, media types, and time of day. Dimensions exist in the Business Model and Mapping (logical) layer and are not visible to end users.

In each dimension, you organize attributes into hierarchical levels. These logical levels represent the organizational rules, and reporting needs required by your business. They provide the structure (metadata) that the Oracle BI Server uses to drill into and across dimensions to get more detailed views of the data.

Dimension hierarchical levels are used to perform the following actions:

- Set up aggregate navigation
- Configure level-based measure calculations (refer to "Level-Based Measure Calculations Example" on page 131)
- Determine what attributes appear when Oracle BI Presentation Services users drill down in their data requests

Process of Creating and Administering Dimensions

Each business model can have one or more dimensions, each dimension can have one or more logical levels, and each logical level has one or more attributes (columns) associated with it.

NOTE: The concept of dimensions for a multidimensional data source is less complex than for dimensions in other data sources. For example, you do not create dimension level keys. A dimension is specific to a particular multidimensional data source (cannot be used by more than one) and cannot be created and manipulated individually. Additionally, each cube in the data source should have at least one dimension and one measure in the logical layer.

The following sections explain how to create dimensions:

- Creating Dimensions
- Creating Dimension Levels and Keys on page 126
- Creating Dimensions Automatically on page 133

Creating Dimensions

After creating a dimension, each dimension can be associated with attributes (columns) from one or more logical dimension tables and level-based measures from logical fact tables. After you associate logical columns with a dimension level, the tables in which these columns exist will appear in the Tables tab of the Dimension dialog box.

To create a dimension

1. In the Business Model and Mapping Layer, right-click a business model and select New Object > Dimension.
2. In the Dimension dialog box, in the General tab, type a name for the dimension.
3. If the dimension is a time dimension, select the Time Dimension check box.

   NOTE: The Default root level field will be automatically populated after you associate logical columns with a dimension level.
4. (Optional) Type a description of the dimension.
5. Click OK.

Creating Dimension Levels and Keys

A dimension contains two or more logical levels. The recommended sequence for creating logical levels is to create a grand total level and then create child levels, working down to the lowest level. The following are the parts of a dimension:

- **Grand total level.** A special level representing the grand total for a dimension. Each dimension can have just one Grand Total level. A grand total level does not contain dimensional attributes and does not have a level key. However, you can associate measures with a grand total level. The aggregation level for those measures will always be the grand total for the dimension.
- **Level.** All levels, except the Grand Total level, need to have at least one column. However, it is not necessary to explicitly associate all of the columns from a table with logical levels. Any column that you do not associate with a logical level will be automatically associated with the lowest level in the dimension that corresponds to that dimension table. All logical columns in the same dimension table have to be associated with the same dimension.
- **Hierarchy.** In each business model, in the logical levels, you need to establish the hierarchy (parent-child levels). One model might be set up so that weeks roll up into a year. Another model might be set up so that weeks do not roll up. For example, in a model where weeks roll up into a year, it is implied that each week has exactly one year associated with it. This might not be true for calendar weeks, where the same week could span two years. Some hierarchies might require multiple elements to roll up, as when the combination of month and year roll up into exactly one quarter. You define the hierarchical levels for your particular business so that results from analyses conform to your business needs and requirements.
- **Level keys.** Each logical level (except the topmost level defined as a Grand Total level) needs to have one or more attributes that compose a level key. The level key defines the unique elements in each logical level. The dimension table logical key has to be associated with the lowest level of a dimension and has to be the level key for that level.

  A logical level may have more than one level key. When that is the case, specify the key that is the primary key of that level. All dimension sources which have an aggregate content at a specified level need to contain the column that is the primary key of that level. Each logical level should have one level key that will be displayed when an Answers or Intelligence Dashboard user clicks to drill down. This may or may not be the primary key of the level. To set the level key to display, select the Use for drill down check box on the Level Key dialog box.

  Be careful using level keys such as Month whose domain includes values January, February, and so on—values that are not unique to a particular month, repeating every year. To define Month as a level key, you also need to include an attribute from a higher level, for example, Year. To add Year, click the Add button in this dialog and select the logical column from the dialog that is presented.
- **Time dimensions and chronological keys.** You can identify a dimension (for example, Year) as a time dimension. At least one level of a time dimension must have a chronological key. The following is a list of some guidelines you should use when setting up and using time dimensions:
  - At least one level of a time dimension must have chronological key. For more information, see Selecting and Sorting Chronological Keys in a Time Dimension on page 130.
  - All time series measures using the Ago and ToDate functions must be on time levels. Ago and ToDate aggregates are created as derived logical columns. For information, refer to “About Time Series Conversion Functions” on page 197.
  - Any physical table that is part of a time logical table cannot appear in another logical table. This prevents using date fields from calendar table as measures.

- Physical tables in time sources, except the most detailed ones, cannot have joins to table outside their source. The join has to be between the time table and the fact table. The join can only be based on foreign key; it cannot be a complex join.
- Ago or ToDate functionality is not supported on fragmented logical table sources. For more information, refer to "About Time Series Conversion Functions" on page 197.

To create and administer dimension hierarchy levels, perform the following tasks:

- Creating a Logical Level in a Dimension on page 128
- Associating a Logical Column and Its Table with a Dimension Level on page 129
- Identifying the Primary Key for a Dimension Level on page 129
- Selecting and Sorting Chronological Keys in a Time Dimension on page 130
- Adding a Dimension Level to the Preferred Drill Path on page 131
- Level-Based Measure Calculations Example on page 131
- Grand Total Dimension Hierarchy Example on page 132
- Creating Dimensions Automatically on page 133

Creating a Logical Level in a Dimension

When creating a logical level in a dimension, you also create the hierarchy by identifying the type of level and defining child levels. For more information about creating hierarchies for a multidimensional data source, refer to "Creating the Business Model Layer for a Multidimensional Data Source" on page 110.

To define general properties for a logical level in a dimension

1. In the Business Model and Mapping layer, right-click a dimension and choose New Object > Logical Level.
2. In the Logical Level dialog box, in the General tab, specify a name for the logical level.
3. Specify the number of elements that exist at this logical level. If this level will be the Grand Total level, leave this field blank. The system will set to a value of 1 by default.

   This number is used by the Oracle BI Server when picking aggregate sources. The number does not have to be exact, but ratios of numbers from one logical level to another should be accurate.
4. When the following criteria is met, perform the specified action:
   - If the logical level is the grand total level, select the Grand total level check box.

     NOTE: There should be only one grand total level for a dimension.
   - If you want the logical level to roll up to its parent, select the Supports rollup to parent elements check box.
   - If the logical level is not the grand total level and does not roll up, do not select either check box.
5. To define child logical levels, click Add.

6. In the Browse dialog box, select the child logical levels and click OK.

   The child levels appear in the Child Levels pane.

7. To remove a previously defined child level, select the level in the Child Levels pane and click Remove.

   The child level and all of its child levels are deleted from the Child Levels pane.

8. (Optional) Type a description of the logical level.
9. Click OK.

Associating a Logical Column and Its Table with a Dimension Level

After you create all logical levels within a dimension, you need to drag and drop one or more columns from the dimension table to each logical level except the Grand Total level. The first time you drag a column to a dimension it associates the logical table to the dimension. It also associates the logical column with that level of the dimension. To change the logical level to be associated with that logical column, you can drag a column from one logical level to another.

NOTE: The logical column(s) comprising the logical key of a dimension table must be associated with the lowest level of the dimension.

After you associate a logical column with a dimension level, the tables in which these columns exist appear in the Tables tab of the Dimensions dialog box.

To verify tables that are associated with a dimension

1. In the Business Model and Mapping layer, double-click a dimension.
2. In the Dimensions dialog box, click the Tables tab.

   The tables list contains tables that you associated with that dimension. This list of tables includes only one logical dimension table and one or more logical fact tables (if you created level-based measures).

3. Click OK or Cancel to close the Dimensions dialog box.

Identifying the Primary Key for a Dimension Level

Use the Keys tab in the Logical Level dialog box to identify the primary key for a level.

To specify a primary key for a dimension level

1. In the Business Model and Mapping layer, expand a dimension and then expand the highest level (grand total level) of the dimension.
2. Double-click a logical level below the grand total level.
3. In the Logical Level dialog box, click the Keys tab.
4. In the Keys tab, from the Primary key drop-down list, select a level key.

   NOTE: If only one level key exists, it will be the primary key by default.

5 To add a column to the list, perform the following steps:

   a In the Logical Level dialog box, click New.

   b In the Logical Level Key dialog box, type a name for the key.

   c In the Logical Level Key dialog box, select a column or click Add.

   d If you click Add, in the Browse dialog box, select the column, and then click OK.

      The column you selected appears in the Columns list of the Logical Level Key dialog box and the check box is automatically selected.

6 If the level is in a time dimension, you can select chronological keys and sort the keys by name.

7 (Optional) Type a description for the key and then click OK.

8 Repeat Step 2 on page 129 through Step 7 on page 130 to add primary keys to other logical levels.

9 In the Logical Level dialog box, click OK.

Selecting and Sorting Chronological Keys in a Time Dimension

At least one level of a time dimension must have a chronological key. For any level, you can select one or more chronological keys and then sort keys in the level, however, only the first chronological key is used at this time.

NOTE: To use a dimension as a time dimension, you must select the Time Dimension check box in the Dimension dialog box.

To select and sort chronological keys for a time dimension

1 In the Business Model and Mapping layer, expand a time dimension and then expand the highest level (grand total level) of the dimension.

   NOTE: For a dimension to be recognized as a time dimension, you need to select the Time Dimension check box in the Dimension dialog box.

2 Double-click a logical level below the grand total level.

3 In the Logical Level dialog box, click the Keys tab.

4 To select a chronological key, in the Keys tab, select the Chronological Key check box.

5 To sort chronological keys, in the Keys tab, double-click a chronological key.

6 In the Chronological Key dialog box, select a chronological key column, click Up or Down to reorder the column, and then click OK.

Adding a Dimension Level to the Preferred Drill Path

You can use the Preferred Drill Path tab to identify the drill path to use when Oracle BI Presentation Services users drill down in their data requests. You should use this only to specify a drill path that is outside the normal drill path defined by the dimensional level hierarchy. It is most commonly used to drill from one dimension to another. You can delete a logical level from a drill path or reorder a logical level in the drill path.

To add a dimension level to the preferred drill path

1. Click the Add button to open the Browse dialog box, where you can select the logical levels to include in the drill path. You can select logical levels from the current dimension or from other dimensions.
2. Click OK to return to the Level dialog box.

   The names of the levels are added to the Names pane.

Level-Based Measure Calculations Example

A level-based measure is a column whose values are always calculated to a specific level of aggregation. For example, a company might want to measure its revenue based on the country, based on the region, and based on the city. You can set up columns to measure CountryRevenue, RegionRevenue, and CityRevenue.

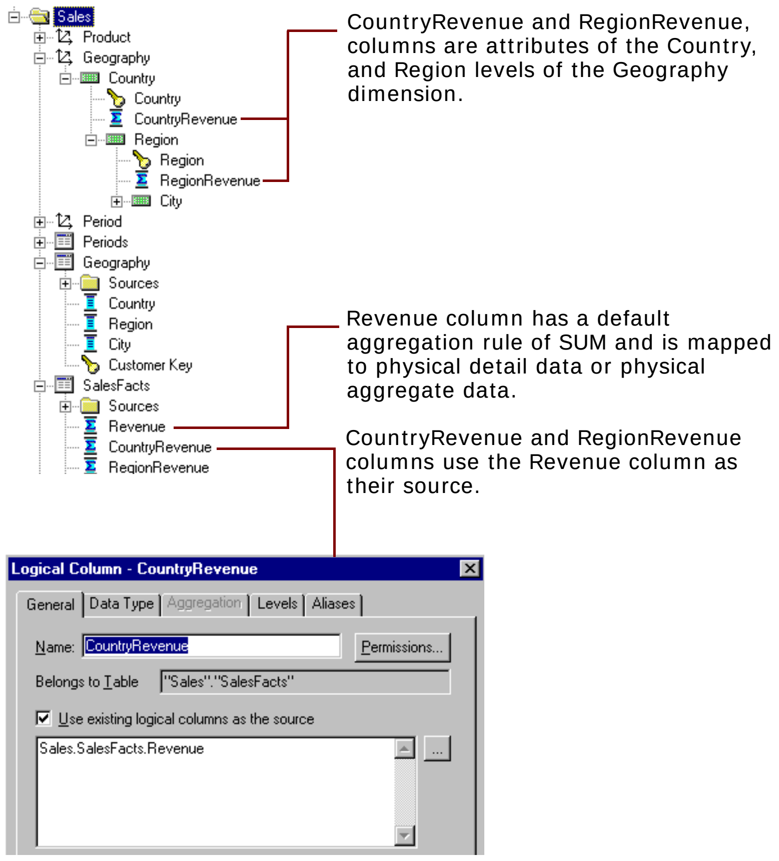
The measure AllProductRevenue is an example of a level-based measure at the Grand Total level. Level-based measures allow a single query to return data at multiple levels of aggregation. They are also useful in creating share measures, that are calculated by taking some measure and dividing it by a level-based measure to calculate a percentage. For example, you can divide salesperson revenue by regional revenue to calculate the share of the regional revenue each salesperson generates.

To set up these calculations, you need to build a dimensional hierarchy in your repository that contains the levels Grandtotal, Country, Region, and City. This hierarchy will contain the metadata that defines a one-to-many relationship between Country and Region and a one-to-many relationship between Region and City. For each country, there are many regions but each region is in only one country. Similarly, for each region, there are many cities but each city is in only one region.

Next, you need to create three logical columns (CountryRevenue, RegionRevenue, and CityRevenue). Each of these columns uses the logical column Revenue as its source. The Revenue column has a default aggregation rule of SUM and has sources in the underlying databases.

You then drag the CountryRevenue, RegionRevenue, and CityRevenue columns into the Country, Region, and City levels, respectively. Each query that requests one of these columns will return the revenue aggregated to its associated level.

Figure 13 on page 132 shows what the business model in the Business Model and Mapping layer would look like for this example.


Sales
Product
Geography
Country
Country
CountryRevenue
Region
Region
RegionRevenue
City
Period
Periods
Geography
Sources
Country
Region
City
Customer Key
SalesFacts
Sources
Revenue
CountryRevenue
RegionRevenue
CountryRevenue and RegionRevenue, columns are attributes of the Country, and Region levels of the Geography dimension.
Revenue column has a default aggregation rule of SUM and is mapped to physical detail data or physical aggregate data.
CountryRevenue and RegionRevenue columns use the Revenue column as their source.
Logical Column - CountryRevenue
General
Data Type
Aggregation
Levels
Aliases
Permissions...
...

Name: CountryRevenue

Belongs to Table "Sales"."SalesFacts"

☑ Use existing logical columns as the source

Sales.SalesFacts.Revenue

Figure 13. Example Business Model in the Business Model and Mapping Layer

Grand Total Dimension Hierarchy Example

You might have a product dimensional hierarchy with levels TotalProducts (grand total level), Brands, and Products. Additionally, there might be a column called Revenue that is defined with a default aggregation rule of Sum. You can then create a logical column, AllProductRevenue, that uses Revenue as its source (as specified in the General tab of the Logical Column dialog). Now drag AllProductRevenue to the grand total level. Each query that includes this column will return the total revenue for all products. The value is returned regardless of any constraints on Brands or Products. If you have constraints on columns in other tables, the grand total is limited to the scope of the query. For example, if the scope of the query asks for data from 1999 and 2000, the grand total product revenue is for all products sold in 1999 and 2000.

If you have three products, A, B, and C with total revenues of 100, 200, and 300 respectively, then the grand total product revenue is 600 (the sum of each product's revenue). If you have set up a repository as described in this example, the following query produces the results listed:

```
select product, productrevenue, allproductrevenue
```

```
from sales_subject_area

where product in 'A' or 'B'

PRODUCT   PRODUCTREVENUE    ALLPRODUCTREVENUE

A         100               600

B         200               600
```

In this example, the AllProductRevenue column will always return a value of 600, regardless of the products the query constrains on.

Creating Dimensions Automatically

You can set up a dimension automatically from a logical dimension table if a dimension for that table does not exist. To create a dimension automatically, the Administration Tool examines the logical table sources and the column mappings in those sources and uses the joins between physical tables in the logical table sources to determine logical levels and level keys. Therefore, it is best to create a dimension in this way after all the logical table sources have been defined for a dimension table.

The following rules apply:

- Create Dimensions is only available if the selected logical table is a dimension table (defined by 1:N logical joins) and no dimension has been associated with this table.
- An automatically created dimension uses the same name as the logical table, adding Dim as a suffix. For example, if a table is named Periods, the dimension is named Periods Dim.
- A grand total level is automatically named [*name of logical table*] Total. For example, the grand total level of the Periods Dim table is Periods Total.
- When there is more than one table in a source, the join relationships between tables in the source determine the physical table containing the lowest level attributes. The lowest level in the hierarchy is named [*name of logical table*] Detail. For example, the lowest level of the periods table is Periods Detail.
- The logical key of the dimension table is mapped to the lowest level of the hierarchy and specified as the level key. This logical column should map to the key column of the lowest level table in the dimension source.
  - If there are two or more physical tables in a source, the columns that map to the keys of those tables become additional logical levels. These additional level names use the logical column names of these key columns.
  - The order of joins determines the hierarchical arrangement of the logical levels. The level keys of these new logical levels are set to the logical columns that map to the keys of the tables in the source.
- If there is more than one logical table source, the tool uses attribute mappings and physical joins to determine the hierarchical order of the tables in the physical sources. For example, you might have three sources (A, B, C) each containing a single physical table and attribute mappings for 10, 15, and 3 attributes, respectively, (not counting columns that are constructed from other logical columns). The following is a list of the results of creating a dimension for this table automatically:

  - The Administration Tool creates a dimension containing 4 logical levels, counting the grand total and detail levels.
  - The key of the table in source B (that has the greatest number of columns mapped and contains the column mapping for the logical table key) should be the level key for the detail level.
  - The parent of the detail level should be the logical level named for the logical column that maps to the key of the physical table in source A.
  - Any attributes that are mapped to both A and B should be associated with level A.
  - The parent of level A should be the logical level named for the logical column that maps to the key of the physical table in source C.
  - Any columns that are mapped to both A and C should be associated with level C.

- Table joins in a physical source might represent a pattern that results in a split hierarchy. For example, the Product table may join to the Flavor table and a Subtype table. This would result in two parents of the product detail level, one flavor level and one subtype level.
- You cannot create a dimension automatically in the following situations:
  - If a dimension with joins and levels has already been created, Create Dimension will not appear on the right-click menu.
  - If the table is not yet joined to any other table, the option is not available because it is considered a fact table.
- In a snowflake schema, if you use a table with only one source, and create the dimension automatically the child tables will automatically be incorporated into a hierarchy. The child tables will form intermediate levels between the grand total level and detail level. If more then one child table exists for a dimension table, while creating dimension automatically hierarchy will be split hierarchy.

To create a dimension automatically

1. In the Administration Tool, open a repository.
2. In the Business Model and Mapping layer of a repository, right-click a logical table.
3. From the right-click menu, choose Create Dimension.

   A dimension appears in the Business Model and Mapping layer.

Setting Up Dimension-Specific Aggregate Rules for Logical Columns

The majority of measures have the same aggregation rule for each dimension. However, some measures can have different aggregation rules for different dimensions. For example, bank balances might be averaged over time but summed over the individual accounts. The Oracle BI Server allows you to configure dimension-specific aggregation rules. You can specify one aggregation rule for a given dimension and specify other rules to apply to other dimensions.

You need to configure dimensions in the Business Model and Mapping layer to set up dimension-specific aggregation. For more information about setting up aggregate navigation, refer to "Setting Up Fragmentation Content in an Oracle BI Repository for Aggregate Navigation" on page 201.

To specify dimension-specific aggregation rules for a single logical column

1. In the Business Model and Mapping layer, double-click a logical column.
2. In the Logical Column dialog box, click the Aggregation tab.
3. In the Aggregation tab, select the Based on dimensions check box.

   The Browse dialog box automatically opens.
4. In the Browse dialog box, click New, select a dimension over which you want to aggregate, and then click OK.
5. In the Aggregation tab, from the Formula drop-down list, select a rule.

   NOTE: After selecting rules for specified dimensions, set the aggregation rule for any remaining dimensions by using the dimension labeled Other.
6. If you need to create more complex formulas, click the ellipsis button to the right of the Formula column to open the Expression Builder.
7. If you have multiple dimensions, to change the order in which the dimension-specific rules are performed, click Up or Down.

   When calculating the measure, aggregation rules are applied in the order (top to bottom) established in the dialog box.
8. Click OK

To specify dimension-specific aggregation rules for multiple logical fact columns

1. In the Business Model and Mapping layer, select multiple logical fact columns.
2. Right-click and select Set Aggregation.

   If the fact column has an aggregation rule, Set Aggregation will not appear in the menu.
3. In the Aggregation dialog box, select or clear the All columns the same check box.

   The check box is checked by default. When checked, you can set aggregation rules that will apply to all selected columns. If you clear the check box, you can set aggregation rules separately for each selected column.
4. In the Aggregation tab, click the Use advanced options check box.
5. In the Browse dialog box, select a dimension over which you want to perform aggregation, and then click OK.

   After setting up the rule for a dimension, specify aggregation rules for any other dimensions in the entry labeled Other.
6. Click the ellipsis button to the right of the Formula column.

7 In the Expression Builder - Aggregate dialog box, from the Formula drop-down list, select the aggregation to perform over the dimension.

8 To change the order in which the dimension-specific rules are performed, click Up or Down, and then click OK.

   When calculating the measure, aggregation rules are applied in the order (top to bottom) established in the dialog box.

Setting Up Display Folders in the Business Model and Mapping Layer

Oracle BI Administrators can create display folders to organize objects in the Business Model and Mapping layer. They have no metadata meaning. After you create a display folder, the selected tables and dimensions appear in the folder as a shortcut and in the business model tree as the object. You can hide the objects so that you only view the shortcuts in the display folder. For more information about hiding these objects, refer to "Using the Options Dialog Box—Repository Tab" on page 31.

NOTE: Deleting objects in the display folder only deletes the shortcuts to those objects.

To set up a logical display folder

1 In the Business Model and Mapping layer, right-click a business model, and choose New Object > Logical Display Folder.

2 In the Logical Display Folder dialog box, in the Tables tab, type a name for the folder.

3 To add tables to the display folder, perform the following steps:

   a Click Add.

   b In the Browse dialog box, select the fact or dimension tables you want to add to the folder and click Select.

4 To add dimensions to the display folder, click the Dimensions tab and perform the following steps:

   a Click Add.

   b In the Browse dialog box, select the dimensions that you want to add to the folder and click Select.

5 Click OK.

Defining Logical Joins

Logical tables are related to each other. How they are related is expressed in logical joins. A key property of a logical join is cardinality. Cardinality expresses how rows in one table are related to rows in the table to which it is joined. A one-to-many cardinality means that for every row in the first logical dimension table there are 0, 1, or many rows in the second logical table. The Administration Tool considers a table to be a logical fact table if it is at the Many end of all logical joins that connect it to other logical tables.

Specifying the logical table joins is required so that the Oracle BI Server can have the necessary metadata to translate a logical request against the business model to SQL queries against the physical data sources. The logical join information provides the Oracle BI Server with the many-to-one relationships between the logical tables. This logical join information is used when the Oracle BI Server generates queries against the underlying databases.

The joins between the logical layer and the physical layer will be automatically created if both of the following statements are true:

- You create the logical tables by simultaneously dragging and dropping all required physical tables to the Business Model and Mapping layer.
- The logical joins are the same as the joins in the Physical layer.

However, you will probably have to create some logical joins in the Business Model and Mapping layer, because you will rarely drag and drop all physical tables simultaneously except in very simple models. In the Business Model and Mapping layer, you should create complex joins with one-to-many relationships and not key or foreign key joins.

You can create logical foreign keys and logical complex joins using either the Joins Manager or the Business Model Diagram. When you create a complex join in the Physical layer, you can specify expressions and the specific columns on which to create the join. When you create a complex join in the Business Model and Mapping layer, you cannot specify expressions or columns on which to create the join. The existence of a join in the Physical layer does not require a matching join in the Business Model and Mapping layer.

CAUTION: It is recommended that you do not have foreign keys for logical tables. However, you can create logical foreign keys and logical complex joins using either the Joins Manager or the Business Model Diagram. A logical key for a fact table must be made up of the key columns that join to the attribute tables.

To create logical joins, perform the following tasks:

- Defining Logical Joins with the Joins Manager on page 137
- Defining Logical Joins with the Business Model Diagram on page 139

Defining Logical Joins with the Joins Manager

You can use the Joins Manager to view logical join relationships and to create logical foreign keys and complex joins.

This section includes the following topics:

- Creating a Logical Foreign Key on page 138
- Creating a Logical Complex Join on page 138

Creating a Logical Foreign Key

Logical foreign key joins might be needed if the Oracle BI Server is to be used as an ODBC data source for certain third-party query and reporting tools. Typically, you should not create logical foreign keys. This capability is in the Administration Tool to provide compatibility with previous releases.

To create a logical foreign key

1. In the Administration Tool toolbar, select Manage > Joins.

   The Joins Manager dialog box appears.

2. Select Action > New > Logical Foreign Key

3. In the Browse dialog box, double-click a table.

   The Logical Foreign Key dialog box appears.

4. Type a name for the foreign key.

5. In the Table drop-down list on the left side of the dialog box, select the table that the foreign key references.

6. Select the columns in the left table that the foreign key references.

7. Select the columns in the right table that make up the foreign key columns.

8. (Optional) To specify a driving table for the key, select a table from the Driving drop-down list, and an applicable cardinality.

   This is for use in optimizing the manner in which the Oracle BI Server processes multi-database inner joins when one table is very small and the other table is very large. Do not select a driving table unless multi-database joins are going to occur. For more information about driving tables, refer to "Specifying a Driving Table" on page 140.

   CAUTION: Use extreme caution in deciding whether to specify a driving table. Driving tables are used for query optimization only under rare circumstances and when the driving table is extremely small, that is, less than 1000 rows. Choosing a driving table incorrectly can lead to severe performance degradation.

9. Select the join type from the Type drop-down list.

10. To open the Expression Builder, click the button to the right of the Expression pane.

    The expression displays in the Expression pane.

11. Click OK to save your work.

Creating a Logical Complex Join

The use of logical complex joins is recommended over logical key and foreign key joins.

To create a logical complex join

1. In the Administration Tool toolbar, select Manage > Joins.

   The Joins Manager dialog box appears.

2. Select Action > New > Logical Complex Join.

   The Logical Join dialog box appears.

3. Type a name for the complex join.
4. In the Table drop-down lists on the left and right side of the dialog box, select the tables that the complex join references.
5. (Optional) To specify a driving table for the key, select a table from the Driving drop-down list, and an applicable cardinality.

   This is for use in optimizing the manner in which the Oracle BI Server processes multi-database inner joins when one table is very small and the other table is very large. Do not select a driving table unless multi-database joins are going to occur. For more information about driving tables, refer to "Specifying a Driving Table" on page 140.

   CAUTION: Use extreme caution in deciding whether to specify a driving table. Driving tables are used for query optimization only under rare circumstances and when the driving table is extremely small, that is, less than 1000 rows. Choosing a driving table incorrectly can lead to severe performance degradation.

6. Select the join type from the Type drop-down list.
7. Click OK.

Defining Logical Joins with the Business Model Diagram

The Business Model Diagram shows logical tables and any defined joins between them. Logical foreign key joins may be needed if the Oracle BI Server is to be used as an ODBC data source for certain third-party query and reporting tools.

NOTE: It is recommended that you use the Business Model Diagram to define complex joins. It is recommended that you do not use the Diagram to define logical foreign keys.

To display the Business Model Diagram

1. In the Administration Tool, right-click a business model, and then select Business Model Diagram > Whole Diagram.
2. Click one of the following buttons on the Administration Tool toolbar:
   - New complex join (Recommended)
   - New foreign key (Not recommended. This capability exists to provide compatibility with previous releases.)
3. With one of the buttons selected, move the cursor to the first table in the join (the one of the one-to-many join).

4 Left-click and move the cursor to the table to which you want to make the join (the many of the one-to-many join), and then left-click on the second table.

   The Logical Foreign Key or Logical Join dialog box appears.

5 For a logical foreign key, select the joining columns from the left and the right tables.

6 (Optional) To specify a driving table for the key, select a table from the Driving drop-down list, and an applicable cardinality.

   This is for use in optimizing the manner in which the Oracle BI Server processes multi-database inner joins when one table is very small and the other table is very large. Do not select a driving table unless multi-database joins are going to occur. For more information about driving tables, refer to "Specifying a Driving Table" on page 140.

   CAUTION: Use extreme caution in deciding whether to specify a driving table. Driving tables are used for query optimization only under rare circumstances and when the driving table is extremely small (fewer than 1000 rows). Choosing a driving table incorrectly can lead to severe performance degradation.

7 Select the join type from the Type drop-down list.

8 To open the Expression Builder, click the button to the right of the Expression pane.

   Only columns, designated predicates, and operators are allowed in the expression. For more information, see "Expression Builder" on page 190.

9 Click OK to save your work.

Specifying a Driving Table

You can specify a driving table for logical joins from the Logical Joins window. Driving tables are for use in optimizing the manner in which the Oracle BI Server processes cross-database joins when one table is very small and the other table is very large. Specifying driving tables leads to query optimization only when the number of rows being selected from the driving table is much smaller than the number of rows in the table to which it is being joined.

CAUTION: To avoid problems, only specify driving tables when the driving table is extremely small - less than 1000 rows.

When you specify a driving table, the Oracle BI Server will use it if the query plan determines that its use will optimize query processing. The small table (the driving table) is scanned, and parameterized queries are issued to the large table to select matching rows. The other tables, including other driving tables, are then joined together.

CAUTION: If large numbers of rows are being selected from the driving table, specifying a driving table could lead to significant performance degradation or, if the MAX_QUERIES_PER_DRIVE_JOIN limit is exceeded, the query terminates.

In general, driving tables can be used with inner joins, and for outer joins when the driving table is the left table for a left outer join, or the right table for a right outer join. Driving tables are not used for full outer joins. For instructions about specifying a driving table, refer to "Defining Logical Joins" on page 136.

There are two entries in the database features table that control and tune driving table performance.

- MAX_PARAMETERS_PER_DRIVE_JOIN

  This is a performance tuning parameter. In general, the larger its value, the fewer parameterized queries that will need to be generated. Values that are too large can result in parameterized queries that fail due to back-end database limitations. Setting the value to 0 (zero) turns off drive table joins.

- MAX_QUERIES_PER_DRIVE_JOIN

  This is used to prevent runaway drive table joins. If the number of parameterized queries exceeds its value, the query is terminated and an error message is returned to the user.

Identifying Physical Tables That Map to Logical Objects

The Physical Diagram shows the physical tables that map to the selected logical object and the physical joins between each table.

One of the joins options, Object(s) and Direct Joins within Business Model, is unique to the logical layer. It creates a physical diagram of the tables that meet both of the following conditions:

- Tables in the selected objects and tables that join directly
- Tables that are mapped (exist in logical table sources in the business model) in the business model

To open the physical diagram of a logical object

1. In the Business Model and Mapping layer, right-click a business model, logical table, or logical table source.
2. Choose Physical Diagram and then one of the joins options.
3. Click and drag any object to more clearly view the relationship lines such as one-to-many.

6 Creating and Maintaining the Presentation Layer in an Oracle BI Repository

This section is part of the roadmap for planning and setting up a repository. For more information, refer to "Planning and Creating an Oracle BI Repository" on page 39.

After you have created the Business Model and Mapping layer, you can drag and drop that layer to the Presentation layer in the Administration Tool. For more information about the Business Model and Mapping layer, refer to "Creating and Administering the Business Model and Mapping Layer in an Oracle BI Repository" on page 109.

This section provides instructions for using the Administration Tool to create and edit objects in the Presentation layer of a repository. This is the fourth step in setting up a repository.

This chapter contains the following topics:

Creating the Presentation Layer in the Repository

The Presentation layer provides a way to present customized views of a business model to users. Presentation Catalogs in the Presentation layer (called Subject Area in Oracle Answers) are seen as business models by Oracle BI Presentation Services users. They appear as catalogs to client tools that use the Oracle BI Server as an ODBC data source. The following topics describe the Process of creating the Presentation layer.

NOTE: In offline editing, remember to save your repository from time to time. You can save a repository in offline mode even though the business models may be inconsistent.

Copy Business Models to Publish to Users

There are several ways to create a Presentation Catalog in the Presentation layer. The recommended method is to drag and drop a business model from the Business Model and Mapping layer to the Presentation layer, and then modify the Presentation layer based on what you want users to see. You can move columns between presentation tables, remove columns that do not need to be seen by the users, or even present all of the data in a single presentation table. You can create presentation tables to organize and categorize measures in a way that makes sense to your users.

Remove Any Unneeded or Unwanted Columns

One important reason to use a custom Presentation layer is to make the schema as easy to use and understand as possible. Therefore, users should not be able to view columns that have no meaning to them. The following columns are examples of columns that you might want to remove from the Presentation layer:

- Key columns that have no business meaning.
- Columns that users do not need to view (for example, codes, when text descriptions exist).
- Columns that users are not authorized to see.

NOTE: You can also restrict access to tables or columns in the security layer. For more information, refer to Chapter 15, “Security in Oracle BI.”

Rename Presentation Columns to User-Friendly Names

By default, presentation columns have the same name as the corresponding logical column in the Business Model and Mapping layer. However, you can specify a different name to be shown to users by changing the name in the Presentation Column dialog box. Whenever you change the name of a presentation column, an alias is automatically created for the old name, so compatibility to the old name remains.

Export Logical Keys in the Presentation Catalog

For each presentation catalog in the Presentation layer, decide whether to export any logical keys as key columns to tools that access it in the Presentation Catalog dialog box. Exporting logical keys is irrelevant to users of Oracle BI Presentation Services, but it may be advantageous for some query and reporting tools. If you decide to export logical keys, be sure the logical key columns exist in the table folders. In this situation, your business model should use logical key/foreign key joins.

When you select the option Export logical keys in the Presentation Catalog dialog box, any columns in the Presentation layer that are key columns in the Business Model and Mapping layer are listed as key columns to any ODBC client. This is the default selection. In most situations, this option should be selected.

NOTE: If you are using a tool that issues parameterized SQL queries, such as Microsoft Access, do not select the option Export logical keys. This will stop the tool from issuing parameterized queries.

Presentation Layer Objects

The Presentation layer adds a level of abstraction over the Business Model and Mapping layer. It is the view of the data seen by client tools and applications.

The Presentation layer provides a means to further simplify or customize the Business Model and Mapping layer for end users. For example, you can hide key columns or present the schema as a single table. Simplifying the view of the data for users makes it easier to craft queries based on users’ business needs.

The section provides instructions for using the Administration Tool’s Presentation layer dialog boxes to create and edit repository objects.

This section includes the following topics:

- Working with Presentation Catalogs on page 145
- Working with Presentation Tables on page 146
- Working with Presentation Columns on page 147
- Using the Alias Tab of Presentation Layer Dialog Boxes on page 149

Working with Presentation Catalogs

In the Presentation layer, presentation catalogs (subject areas) allow you to show different views of a business model to different sets of users. Presentation catalogs have to be populated with contents from a single business model. They cannot span business models.

When creating a presentation catalog, selecting the option Export logical keys causes any columns in the Presentation layer that are key columns in the Business Model and Mapping layer to be listed as key columns to any ODBC client. This is the default selection. In most situations, this option should be selected. Many client tools differentiate between key and nonkey columns, and the option Export logical keys provides client tools access to the key column metadata. Any join conditions the client tool adds to the query, however, are ignored; the Oracle BI Server uses the joins defined in the repository.

If you set an implicit fact column this column will be added to a query when it contains columns from two or more dimension tables and no measures. The column is not visible in the results. It is used to specify a default join path between dimension tables when there are several possible alternatives.

The Presentation Catalog dialog box has three tabs: General, Presentation Tables, and Aliases. The functionality provided in each tab is described in the following list:

Tab	Comment
General	Use this tab to create or edit a presentation catalog.
Presentation Table	Use this tab to reorder or sort the Presentation layer tables in the Administration Tool workspace, and to delete tables. You can also use this tab to access the Presentation Table dialog box, where you can create and edit tables.
Aliases	Use this tab to specify or delete an alias for a catalog folder.

To create a presentation catalog

1. In the Presentation layer, right-click and select New Presentation Catalog.
2. In the Presentation Catalog dialog box, in the General tab, type a name for the presentation catalog and click Permissions.
3. In the Permissions dialog box, assign user or group permissions to the catalog folder, and then click OK.

   For more information about assigning permissions to a presentation catalog, refer to "Setting Permissions for Repository Objects" on page 33.

4 In the Presentation Catalog dialog box, from the Business Model drop-down list, select a business model.

After you add columns to the presentation catalog, the drop-down list becomes inactive because you can add columns from only one business model in each presentation catalog.

5 To expose the logical keys to other applications, select the option Export logical keys.

NOTE: If you are using a tool that issues parameterized SQL queries, such as Microsoft Access, do not select the Export logical keys option. Not exporting logical keys stops the tool from issuing parameterized queries.

6 (Optional) Type a description of the catalog folder.

This description will appear in a mouse-over ToolTip for the presentation column in Oracle Business Intelligence Answers.

CAUTION: When you move columns into presentation catalog folders, be sure columns with the same name or an alias of the same name do not already exist in the catalog.

7 Set an Implicit Fact Column.

8 Click OK.

Working with Presentation Tables

You can use presentation tables to organize columns into categories that make sense to the user community. Presentation tables in the Presentation layer contain columns. A presentation table can contain columns from one or more logical tables. The names and object properties of the presentation tables are independent of the logical table properties.

The Presentation Tables dialog box has three tabs: General, Columns, and Aliases. The functionality provided in each tab is described in the following list:

Tab	Comment
General	Use this tab to create or edit a presentation table.
Columns	Use this tab to reorder or sort the Presentation layer columns in the Administration Tool workspace, and to delete columns. You can also use this tab to access the Presentation Column dialog box, where you can create and edit columns.
Aliases	Use this tab to specify or delete an alias for a presentation table.

To create a presentation table

1 Right-click a catalog folder in the Presentation layer, and then select New Presentation Table from the shortcut menu.

The Presentation Table dialog box appears.

2 In the General tab, specify a name for the table.

3. Click the Permissions button to open the Permissions dialog box, where you can assign user or group permissions to the table.

   For more information about assigning permissions to a presentation table, refer to "Setting Permissions for Repository Objects" on page 33.

4. (Optional) Type a description of the table.

   NOTE: To give the appearance of nested folders in Answers, prefix the name of the presentation folder to be nested with a hyphen and a space and place it after the folder in which it nests (- <*folder name*>). For example, to nest the Sales Facts folder in the Facts folder, place the Sales Facts folder directly after Facts in the metadata and change its name to - Sales Facts. When Answers displays the folder name in the left pane, it omits the hyphen and space from the folder name. To nest a second folder, for example Marketing Facts, in the Facts folder, change its name to - Marketing Facts and place it directly after Sales Facts. The standard preconfigured repositories provide additional examples for you to review.

To delete a presentation table

1. In the Presentation layer, right-click a catalog and select Properties.
2. In the Presentation Catalog dialog box, click the Presentation Tables tab.
3. In the Presentation Tables tab, select a table and click Remove.

   A confirmation message appears.

4. Click Yes to remove the table, or No to leave the table in the catalog.
5. Click OK.

To reorder a table or sort all tables in a presentation catalog

1. In the Presentation layer, right-click a catalog and select Properties.
2. In the Presentation Catalog dialog box, click the Presentation Tables tab.
3. To move a table, perform the following steps:
   a. In the Presentation Tables tab, in the Name list, select the table you want to reorder.
   b. Use drag-and-drop to reposition the table, or click the Up and Down buttons.
4. To sort all tables in alphanumeric order, click the Name column heading.

   This toggles the sort between ascending and descending alphanumeric order.

Working with Presentation Columns

The presentation column names are, by default, identical to the logical column names in the Business Model and Mapping layer. However, you can present a different name by clearing both the Use Logical Column Name and the Display Custom Name check boxes in the Presentation Column dialog box.

To provide a convenient organization for your end users, you can drag and drop a column from a single logical table in the Business Model and Mapping layer onto multiple presentation tables. This allows you to create categories that make sense to the users. For example, you can create several presentation tables that contain different classes of measures—one containing volume measures, one containing share measures, one containing measures from a year ago, and so on.

The Presentation Column dialog box has the following tabs:

- **General.** Use this tab to create or edit presentation columns.
- **Aliases.** Use this tab to specify or delete an alias for a presentation column.

To create a presentation column

1. Right-click a presentation table in the Presentation layer, and then choose New Presentation Column.
2. In the Presentation Column dialog box, to use the name of the logical column for the presentation column, select the Use Logical Column check box.

   The name of the column and its associated path in the Business Model and Mapping layer appears in the Logical Column Name field.
3. To specify a name that is different from the Logical Column name, clear the Use Logical Column check box, and then type a name for the column.
4. To assign user or group permissions to the column, click Permissions.
5. In the Permissions dialog box, assign permissions, and then click OK.

   For more information about assigning permissions, refer to "Setting Permissions for Repository Objects" on page 33.
6. To select the logical column, click Browse.
7. In the Browse dialog box, select the column, and then click Select.
8. (Optional) Type a description of the presentation column.
9. To define any aliases for the logical column, click the Aliases tab.

To edit a presentation column

1. In the Business Model and Mapping layer, double-click the presentation column.
2. In the Presentation Column dialog box, click Edit.
3. In the Logical Column dialog box, make any changes or review the information, and then click OK.

To delete a presentation column

1. Right-click a presentation table in the Presentation layer, and then select Properties.
2. Click the Columns tab.

3. Select the column you want to delete.
4. Click Remove, or press the Delete key, and then click Yes.

To reorder a presentation column

1. Right-click a presentation table in the Presentation layer, and then select Properties.
2. Click the Columns tab.
3. Select the column you want to reorder.
4. Use drag-and-drop to reposition the column, or click the Up and Down buttons.
5. Click OK.

Using the Alias Tab of Presentation Layer Dialog Boxes

An Alias tab appears on the Presentation Catalog, Presentation Table, and Presentation Column dialog boxes. You can use this tab to specify or delete an alias for the Presentation layer objects.

To add an alias

1. Double-click a presentation catalog.
2. In the Presentation Layer dialog box, click the Aliases tab.
3. Click the new button, and then type the text string to use for the alias.
4. Click OK.

To delete an alias

1. Double-click a presentation catalog.
2. In the Presentation Layer dialog box, click the Aliases tab.
3. In the Aliases list, select the alias you want to delete.
4. Click the delete button, and then click OK.

Generating an XML File from a Presentation Table

You can import the structure of an Oracle BI presentation table into Oracle's Siebel Tools. To do this, you create an XML file from a table in the Presentation layer of an Oracle BI repository and then import the XML file into Oracle's Siebel Tools.

For more information, refer to the documentation for Oracle's Siebel Tools.

7 Completing Setup and Managing Oracle BI Repository Files

This section is part of the roadmap for planning and setting up a repository. For more information, refer to "Planning and Creating an Oracle BI Repository" on page 39. After you have created the repository file, the Physical layer, Business Model and Mapping layer, and Presentation layer, you need to perform several tasks to complete the initial repository setup. This section contains these setup steps and topics for managing your repository files.

This section contains instructions for the following topics:

- Process of Completing the Setup for a Repository File
- Importing From Another Repository on page 154
- Querying and Managing Repository Metadata on page 156
- Constructing a Filter for Query Results on page 159
- Comparing Repositories on page 161
- Merging Oracle BI Repositories on page 163
- Exporting Oracle BI Metadata to IBM DB2 Cube Views on page 167
- About Extracting Metadata Subsets Into Projects on page 167
- Setting up and Using the Oracle BI Multiuser Development Environment on page 169
- Setting Up the Repository to Work with Delivers on page 180

Process of Completing the Setup for a Repository File

Perform the following tasks to complete the repository file setup:

- Saving the Repository and Checking Consistency on page 152
- Add an Entry in the NQSConfig.INI File on page 152
- Create the Data Source on page 153
- Start the Oracle BI Server on page 153
- Test and Refine the Repository on page 154
- Publish to User Community on page 154

Saving the Repository and Checking Consistency

In offline editing, remember to save your repository from time to time. You can save a repository in offline mode even though the business models may be inconsistent.

To determine if business models are consistent, use the Check Consistency command to check for compilation errors. You can check for errors in the whole repository with the File > Check Global Consistency command or in a particular logical business model by selecting a business model and using the Check Consistency command from the right-click menu.

The consistency check analyzes the repository for certain kinds of errors and inconsistencies. For example, the consistency check finds any logical tables that do not have logical sources configured or any logical columns that are not mapped to physical sources, checks for undefined logical join conditions, determines whether any physical tables referenced in a business model are not joined to the other tables referenced in the business model, and checks for existence of a presentation catalog for each business model.

NOTE: Passing a consistency check does not guarantee that a business model is constructed correctly, but it does rule out many common problems.

When you check for consistency, any errors or warnings that occur are displayed in a dialog box. Correct any errors and check for consistency again, repeating this process until there are no more errors. An error message indicates a problem that needs to be corrected. A warning message identifies a possible problem to the Oracle BI Administrator. Refer to "Checking the Consistency of a Repository or a Business Model" on page 26.

NOTE: The consistency check algorithm was enhanced in Siebel Business Analytics 7.8.2. After upgrading from a previous software version and checking the consistency of your repository, you might observe messages that you had not received in previous consistency checks. This typically indicates inconsistencies that had been undetected prior to the upgrade, not new errors.

Add an Entry in the NQSConfig.INI File

After you build a repository and it is consistent, you need to add an entry in the NQSConfig.INI file for the repository. The entry allows the Oracle BI Server to load the repository into memory upon startup. Therefore, if you want your changes to take effect immediately, restart the Oracle BI Server. The NQSConfig.INI file is located in the following location:

[*drive path*]:\OracleBI\Config\

If errors occur in the file, you might not be able to restart the server. You can review the log messages in the log (NQServer.log) at the following location:

[*drive path*]:\OracleBI\server\Log\

For organizations that use Oracle Application Server, Oracle recommends that you use Oracle Application Server Control to modify configuration files. For organizations that use other application servers, Oracle recommends that you use JConsole. For more information, see *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

CAUTION: Although you can use an editor such as Windows Notepad, it's not recommended.

To add an entry in the NQSConfig.INI file

1. Open the NQSConfig.INI file in an editor such as Notepad.

2. In the repository section, add an entry for your new repository in this format:

   ```
   logical_name = repository_file_name ;
   ```

   For example, if the repository file is named northwind.rpd and the logical name you assign it is star, the entry will read as follows:

   ```
   star = northwind.rpd ;
   ```

   One of the repositories should be specified as the default and using the same repository name, the entry would read as follows:

   ```
       star = northwind.rpd, default;
   ```

   The logical name is the name end users have to configure when configuring a DSN in the Oracle BI ODBC setup wizard. Filenames consisting of more than a single word should be enclosed in single quotes. Save the configuration file after adding the entry. For more information about the NQSConfig.INI file, refer to *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

3. Save the file and restart the Oracle BI Server.

   For more information, see “Starting the Oracle BI Server” on page 209.

Create the Data Source

For end user client applications to connect to the new repository, each user needs to define a data source using an ODBC driver.

NOTE: Oracle BI Presentation Services has the same relationship to the Oracle BI Server as any other client application.

You can create standard data sources, and data sources that will participate in a cluster. The steps to create a new data source are given in Chapter 12, “Connectivity and Third-Party Tools in Oracle BI Server.”

Start the Oracle BI Server

When you start the Oracle BI Server, the repository specified in the NQSConfig.INI file is loaded and is available for queries. For detailed information on starting the server, refer to “Starting the Oracle BI Server” on page 209.

Test and Refine the Repository

When the repository is created and you can connect to it, run sample queries against it to test that it is created properly. Correct any problems you find and test again, repeating this process until you are satisfied with the results.

Tips For Performance Tuning

The Physical Data Model should more closely resemble the Oracle BI metadata model (for example the star schema) instead of a transactional database system. If the Physical model is set up like the underlying transactional model, performance problems and configuration problems could likely arise. For additional tips, refer to "Guidelines For Designing a Repository" on page 50.

NOTE: Make sure the metadata is generating the correct record set first, then focus on performance tuning activities (such as adding sources).

- Accuracy of metadata is more important than improved performance.
- In general, push as much processing to the database as possible. This includes tasks such as filtering, string manipulation, and additive measures.
- Move as much of the query logic to the ETL as possible to improve system response time. Pre-calculation of additive metrics and attributes will reduce query complexity and therefore response time.
- Use base and extension tables to allow for a cleaner upgrade path. To improve runtime performance, merge the two tables into a third one, which is then mapped into Oracle BI. Although this technique requires more effort and a larger ETL batch window, it insulates the system from upgrade errors while still providing optimal performance.
- Denormalize data into _DX (dimension extension) tables using the ETL process to reduce runtime joins to other tables.

Publish to User Community

After testing is complete, notify the user community that the data sources are available for querying. Presentation Services users need only know the URL to type in their browser. Client/server users (for example, users accessing the Oracle BI Server with a query tool or report writer client application) need to know the subject area names, the machine on which the server is running, their user IDs and passwords, and they need to have the ODBC setup installed on their PCs. They may also need to know the logical names of repositories when multiple repositories are used and the data source name (DSN) being created does not point to the default repository.

Importing From Another Repository

It is recommended that you create projects in the repository that contain the objects that you wish to import, and then use repository merge to bring the projects into your current repository. For more information, see "Merging Oracle BI Repositories" on page 163.

Use the Repository Import Wizard to import a presentation catalog (called Subject Area in Answers) and its associated children business model and physical layer objects from another repository. You can also import users, groups, variables, initialization blocks, and projects.

NOTE: By default, the Import from repository option on the File menu is disabled. When this option is enabled, the Deprecated feature warning dialog box appears each time a user begins the import. If the user clicks yes in the dialog box, the import process continues. If the user clicks No, the import process terminates. To turn on this option, see "Using the Options Dialog Box—General Tab" on page 29.

Use this option when the objects you import are unrelated to objects already in the repository such as when the business model and physical layer objects do not exist. If an object of the same name and type exists, the import process overwrites the existing object with the new object. When you import objects from one repository into another, the repository from which you are importing must be consistent.

To import from another repository

1. In the Administration Tool, open the repository in offline mode, and then choose File > Import from Repository.

   This option is only available when opening a repository offline.

2. In the Repository Import Wizard, select a repository by its file name or, if it is being used by Oracle BI, by the ODBC DSN that points to the desired repository on that server, and then click Next.
3. Type the Oracle BI Administrator user ID and password.
4. In the Repository Import Wizard-Objects to Update dialog box, from the drop-down list, choose a category using Table 19 on page 156 as a guide.

   Available buttons (options) depend on the category that you select. You can add only the selected object, the selected object with its child objects, or the selected object with its parent objects.

5. Repeat Step 4 until you have added all the objects you want to import.

   If any objects need to be checked out, the Check Out Objects screen appears, notifying you that objects will be checked out.

6. Click Next to continue.

7 In the Finish screen, click finish.

Table 19. Categories of Repository Objects to Import

Category	Description
Catalogs	When you select a catalog, the Add with Children button become active. Presentation catalogs are always added with all their child objects. All associated objects, from the Presentation layer to the Physical layer, will be updated and synchronized.
Groups	When you select a group, the Add, Add with Children, and Add with Parents buttons become active. (You can view information about group membership from the Security Manager.)
Initialization Blocks	When you select an initialization block, the Add with Children button becomes active.
List Catalogs	When you select a list catalog, the Add with Children button becomes active.
Projects	When you select a project, all repository objects that you choose to update and synchronize appear.
Target levels	When you select a target level, the Add with Children button is active.
Users	When you select a user, the buttons that become active depend on the user that you select in the left pane.
Variables	When you select a variable, the Add and Add with Parents buttons are active. Defined system and session variables appear in the left pane.

Querying and Managing Repository Metadata

You can query for objects in the repository using the Query Repository tool. If you query using the All Types option, a list appears that contains the exposed object types in the repository. The list does not contain objects such as aggregate rules, logical source folders, privilege packages, and other objects that are considered internal objects.

You can use repository queries to help manage the repository metadata in the following ways:

- Examine and update the internal structure of the repository. For example, you can query a repository for objects in the repository based on name, type (such as Catalog, Complex Join, Key, and LDAP Server), or on a combination of name and type. You can then edit or delete objects that appear in the Results list. You can also create new objects, and view parent hierarchies.
- Query a repository and view reports that show such items as all tables mapped to a logical source, all references to a particular physical column, content filters for logical sources, initialization blocks, and security and user permissions.

  For example, you might want to run a report prior to making any physical changes in a database that might affect the repository. You can save the report to a file in comma-separated value (CSV) or tab-delimited format.

- You can save a query to run again in the future or save the query results to an external file. When you save to an external file, the encoding options are ANSI, Unicode, and UTF-8.

To query a repository

1. Open the Administration Tool, and then open your repository.
2. In the repository, right-click any object and choose Display Related > [*type of object on which you want to search*].
3. In the Query Repository dialog box, the type of object is prefilled.
4. In the Query Repository dialog box, complete the query information using Table 20 on page 158 as a guide.

   Table 20 on page 158 contains a description of most fields and buttons in the Query Repository dialog box.
5. Click Query.

   You can view the results, save the results to an external file, or make changes and overwrite the existing query or save as a new query.

To save a query results to an external file

1. After running the query, in the Query Repository dialog box, click Save.
2. In the Save As dialog box, type a name, choose a type of file and an Encoding value.
3. To add additional columns of information to the results, click Add Columns.
4. In the Select information to add to the report dialog box, select the columns you want from the list.
5. You can re-order the columns by selecting a checked column and clicking Up or Down.
6. Click OK
7. In the Save as dialog box, select a Save as type.
8. Click Save.

To save a query to run it again later

1. After running the query, in the Query Repository dialog box, click Save Query As.
2. In the Save Query As dialog box, type a name for the query and click Save.
3. In the Query Repository dialog box, click Close.

To delete a saved query

1. Open the Administration Tool, and then open your repository.
2. From the Tools menu, choose Query Repository.

3 In the Query Repository dialog box, click Saved Queries.

4 In the Saved Queries dialog box, scroll to the right or maximize the dialog box.

5 Click the delete button for the query that you wish to delete, and then click Close.

To run a saved query

1 Open the Administration Tool, and then open your repository.

2 From the Tools menu, choose Query Repository.

3 In the Query Repository dialog box, click Saved Queries.

4 In the Saved Queries dialog box, select the row that contains the query you wish to run, and then click Select.

5 In the Query Repository dialog box, click Query.

You can view the results, save the results to an external file, or make changes and overwrite the existing query or save as a new query.

Table 20. Query Repository Fields and Some Buttons

Field or Button	Description
Delete	After executing a query, use this button to delete an object in the list of query results.
Edit	After executing a query, use this button to edit an object in the list of query results. Not all repository objects can be edited from the results list. For example privilege objects and user database sign-on objects. If an object cannot be edited from the results list, the Edit button will not be available.
Filter	Use this button to create or edit a filter for your query. After you create a filter, the filter criteria appear in the text box on the left of the button. For more information, refer to “Constructing a Filter for Query Results” on page 159.
GoTo	After executing a query, use this button to go to the object in the Administration Tool view of the repository.
Mark	After executing a query, use this button to mark the selected objects. To unmark an object click the button again. You can mark objects to make them easier to visually identify as you develop metadata.
Name	Allows a search by object name. You can use an asterisk (*) wildcard character to specify any characters. The wildcard character can represent the first or last characters in the search string. Searches are not case sensitive.
New	Use to request a new query.

Table 20. Query Repository Fields and Some Buttons

Field or Button	Description
Parent	After executing a query, use this button to view the parent hierarchy of an object. If the object does not have a parent, a message appears. In the Parent Hierarchy dialog box, you can edit or delete objects. However, if you delete an object, any child objects of the selected object will also be deleted.
Query	Use this button when you are ready to submit your query.
Save Query As	Opens a dialog box in which you can save a query and select or delete a previously saved query. Available only after you have created a query.
Saved Queries	Opens a dialog box in which you can type a new name for the query or browse existing saved queries. This button is only available if you have previously saved queries.
Set Icon	After executing a query, use this button to select a different icon for an object. To change the icon back to this original icon, use this button and select Remove associated icon. You can set special icons for objects to make it easier to visually identify them as having common characteristics. You may, for example, want to pick a special icon to identify columns that will be used only by a certain user group.
Show Qualified Name	Use this check box to display the fully qualified name of the object(s) found by the query. For example, if you query for logical tables, the default value in the Name list is the table name. However, if you select the Show Qualified Names check box, the value in the Name list changes to *businessmodelname.logicaltablename.columnname*.
Type	Select a type from the drop-down list to narrow your search to a particular type of object.

To create a new object

1. From the Administration Tool menu bar, choose Tools > Query Repository.
2. In the Query Repository dialog box, in the Type drop-down list, select the type of object you want to create.
3. Click New.

   The dialog boxes that appear depend on the object type that you select. For more information refer to the section of this documentation that pertains to creating that object.

Constructing a Filter for Query Results

Use the Query Repository Filter dialog box to filter the results in the Results list of the Query Repository dialog box.

The Query Repository Filter dialog box contains five columns: Item column and its operator or selection column, a Value column and its operator or selection column, and a Delete column that lets you delete the highlighted filter.

To access the Query Repository Filter dialog box

1. From the Tools menu, choose Query Repository.
2. In the Query Repository dialog box, select an item in the Results list or select an item from the Type list, and then click Filter.

To construct a filter

1. From the Tools menu, choose Query Repository.
2. In the Query Repository dialog box, select an item in the Results list or select an item from the Type list, and then click Filter.
3. In the Query Repository Filter dialog box, click the Item field.

   The Item drop-down list contains the items by which you can filter.
4. In the Item drop-down list, select the filter that you want to apply to the Results or Type object you selected in Step 2 on page 160.

   Depending on what you select in the Item drop-down list, other options may become available.

To construct a filter to view all databases referenced in a business model

1. From the Tools menu, choose Query Repository.
2. In the Query Repository dialog box, select Database from the Type drop-down list, and then click Filter.
3. In the Query Repository Filter dialog box, click the Item field.

   The Item drop-down list contains the items by which you can filter.
4. In the Item drop-down list, select Related to.

   The equals sign (=) appears in the column to the right of the Item field.
5. Click the ellipsis button to the right of the Value field, and in the drop-down list, choose Select object.
6. In the Select dialog box, select the business model by which you want to filter, and then click Select.

   Your selection appears in the Value field.
7. Click OK to return to the Query Repository dialog box.

   The filter appears in the Filter text box of the Query Repository dialog box.

To construct a filter to view all Presentation layer columns mapped to a logical column

1. From the Tools menu, choose Query Repository.
2. In the Query Repository dialog box, from the Type drop-down list, select Presentation Column and then click Filter.
3. In the Query Repository Filter dialog box, click the Item field.

   The Item drop-down list contains the items by which you can filter.
4. In the Item drop-down list, select Column.

   The equals sign (=) appears in the column to the right of the Item field.
5. Click the ellipsis button to the right of the Value field, and in the drop-down list, choose Select object.
6. In the Select dialog box, select the column by which you want to filter and click Select.

   Your selection appears in the Value field.
7. Click OK to return to the Query Repository dialog box.

   The filter appears in the Filter text box of the Query Repository dialog box.

   You can construct more than one filter; when you do, the Operator field becomes active. When the Operator field is active, you can set AND and OR conditions.

TIP: If you are constructing a complex filter, you may want to click OK after adding each constraint to verify that the filter construction is valid for each constraint.

Comparing Repositories

This section explains how to use the Compare Repositories option in the Administration Tool. It allows you to compare the contents of two repositories, including all objects in the Physical, Business Model and Mapping, and Presentation layers.

If you are using an Oracle BI Applications repository and have customized its content, you can use this option to compare your customized repository to a new version of the repository received with Oracle BI Applications.

For more information about merging the contents of your customized Oracle BI Applications repository with that of a new version of the repository, refer to "Merging Oracle BI Repositories" on page 163.

To compare two repositories

1. In the Administration Tool, open a repository in offline mode.

   The repository that you open in this step is referred to as the *current repository*. For instructions on opening a repository, refer to "Online and Offline Repository Modes" on page 24.
2. From the File menu, select Compare.

3. In the Select Original Repository dialog box, select the repository you want to compare to the open repository.
4. In the Open Offline dialog box, type the password and click OK.
5. Use the Compare repositories dialog box to review the differences between the two repositories.

   The following list contains the values in the Change column and a description:

Change	Description
Created	Object was created in the current repository and does not exist in the original repository.
Deleted	Object exists in the original repository but has been deleted from the current repository.
Modified	Object exists in the original repository but has been modified in the current repository.

The following is a list of some of the buttons in the Compare repositories dialog box and a description of the functionality provided:

Button	Functionality
Diff	Differences between the current repository and the original repository.
Edit 2	Opens created objects for editing.
Equalize	This equalizes the upgrade id of the objects. If objects have the same upgrade id, they are considered to be the same object. Not available when merging repositories.
Filter	Opens the Comparison Filter dialog box to allow you to filter the objects that appear in the Compare repositories dialog box by type of change and type of object. You can specify what you want to appear and what you want to be hidden. If you select the check box (Group created and deleted objects), the tool will filter out the child objects of created and deleted objects, allowing you to view only the parent objects. By default, all items are shown.
Find	Search by an object Name and Type (such as Initialization Block).
Find Again	Search again for the most recent Find value.
Mark	Marks the object you select. Boxes appear around created and modified objects. To remove marks, from the File menu, choose Turn off Compare Mode. Not available when merging repositories.
Save	Saves a list of the differences between the two repositories.
Select	Allows you to select a repository to compare with the current repository. Not available when merging repositories.

Button	Functionality
Stats	Provides the number of changes by Change type. During a multiuser development merge, this allows you to view an overview of the merge decisions that will take place.
View 1	Opens deleted objects in read-only mode.

Turn Off Compare Mode

This option allows you to remove marks applied to objects while using the Compare Repositories and Merge Repositories options. The Turn off Compare Mode option is only available after you have clicked Mark during the File > Compare action. If no repository object is marked, this option is not available.

To enable the Turn Off Compare Mode

- From the Administration Tool toolbar, select File > Turn Off Compare Mode.

Merging Oracle BI Repositories

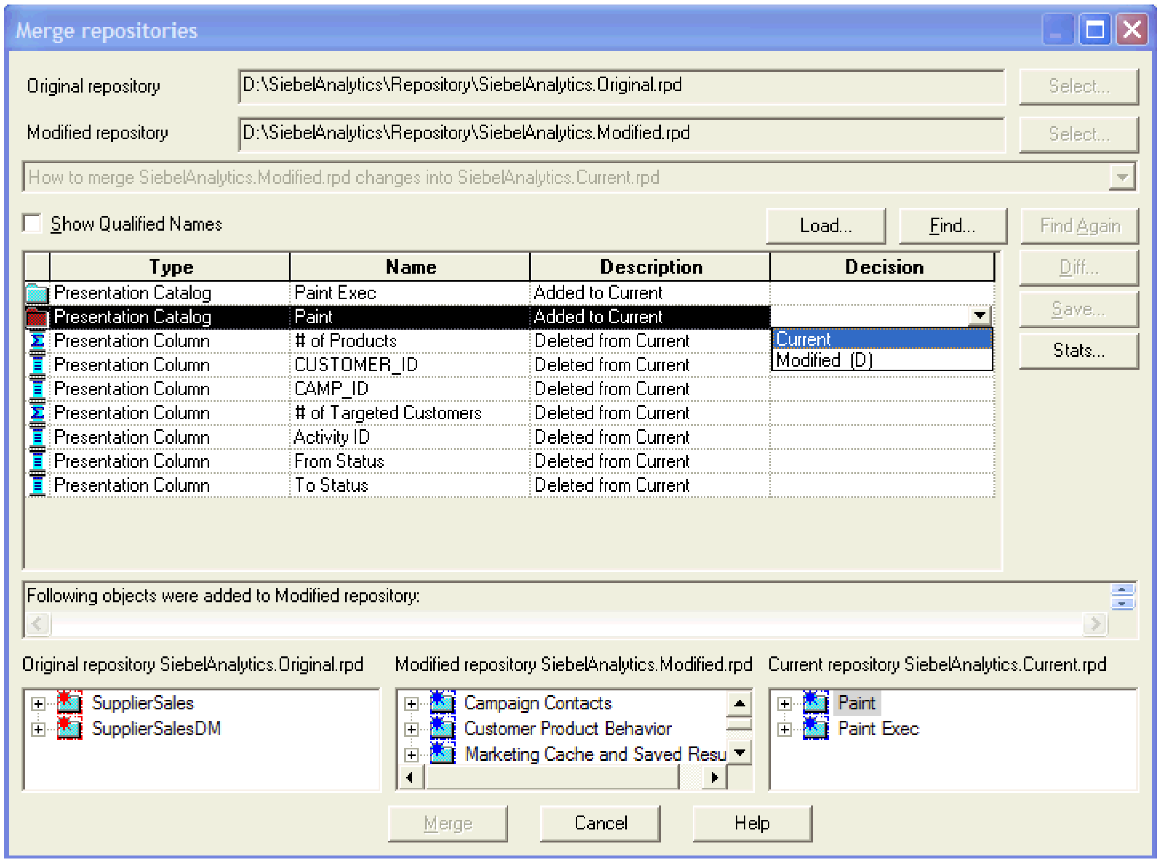
This section is intended for organizations that use an Oracle BI Applications repository. However, the Merge Repository option can also be used by customers to upgrade their custom repositories.

The merge process involves three versions of an Oracle BI repository. The terms used in the following descriptions are the terms used in the Administration Tool user interface.

- **Original repository.** The repository you received with the previous version of Oracle BI Applications. This section uses Oracle's SiebelAnalytics.Original.rpd as an example.
- **Modified repository**. The repository that contains the customizations you made to the original repository. This section uses Oracle's SiebelAnalytics.Modified.rpd as an example.
- **Current repository**. The repository that is installed with this version and is currently opened as the main repository. This section uses Oracle's SiebelAnalytics.Current.rpd as an example.

During the merge process, you can compare the original repository with the modified repository and the original repository with the current repository. The Merge Repository option allows you to decide on an object-by-object basis if you want to merge your customizations with the current repository.

Figure 14 on page 164 shows the following parts of the Merge repositories dialog box. Table 21 on page 165 contains descriptions of columns and buttons on the Merge repositories dialog box.



Merge repositories

Original repository: D:\SiebelAnalytics\Repository\SiebelAnalytics.Original.rpd

Modified repository: D:\SiebelAnalytics\Repository\SiebelAnalytics.Modified.rpd

How to merge SiebelAnalytics.Modified.rpd changes into SiebelAnalytics.Current.rpd

Show Qualified Names

Type	Name	Description	Decision
Presentation Catalog	Paint Exec	Added to Current	
Presentation Catalog	Paint	Added to Current	
Presentation Column	# of Products	Deleted from Current	
Presentation Column	CUSTOMER_ID	Deleted from Current	
Presentation Column	CAMP_ID	Deleted from Current	
Presentation Column	# of Targeted Customers	Deleted from Current	
Presentation Column	Activity ID	Deleted from Current	
Presentation Column	From Status	Deleted from Current	
Presentation Column	To Status	Deleted from Current	

Following objects were added to Modified repository:

Original repository SiebelAnalytics.Original.rpd
- SupplierSales
- SupplierSalesDM

Modified repository SiebelAnalytics.Modified.rpd
- Campaign Contacts
- Customer Product Behavior
- Marketing Cache and Saved Resu

Current repository SiebelAnalytics.Current.rpd
- Paint
- Paint Exec

Figure 14. Merge Repositories Dialog Box

- The Original repository and Modified repository fields appear at the top of the dialog box. The Select buttons at the right of the fields allow you to select repositories.
- The read-only text box describes the type of merge that you selected.

- The decision table in the middle of the dialog box dynamically changes based on the choices you make in this window. For example, if you select the option to merge repository contents, the decision table displays the following information:

Table 21. Merge Repositories Decision Table

Column Name	Description
Decision	Allows you to select an action you want to perform on the selected repository change. For examples of some results that will occur, refer to "Examples of the Results of Some Decision Choices" on page 166. ■ **Current.** This type has no suffix. Selecting this type means that you want to leave the object in the current repository as is). ■ **Modified.** This type can have an A (add), a D (delete), or an AR (after renaming) suffix. **NOTE:** AR means that the Modified version will be accepted but because it conflicts with another name in the repository, it will be renamed. For example, if both the Current and Modified repositories add the same object with the same name and the user chooses to accept both versions, both would be added and the object from the Modified repository would be renamed. ■ **Mix.** The object was not added or deleted but at least one of its properties was modified. For example, you can select the choices for the properties.
Description	Description of the changes between the original repository and the modified repository, and between the original repository and the current repository.
Diff (button)	Shows which properties of an object have been modified. Available for objects that are labelled Mix (not added or deleted).
Find (button)	Search by an object Name and Type (such as Initialization Block.
Find Again (button)	Search again for the most recent Find value.
Load (button)	Loads a saved decisions file from the Repository subdirectory so that you can continue processing a repository merge.
Name	Object name
Save (button)	Saves a file containing interim changes in the Repository subdirectory so that you can stop work on the merge and continue it later. After saving the changes (decisions) you need to close the Merge repositories dialog box by clicking Cancel.
Stats (button)	In multiuser development, this allows you to view an overview of the merge decisions that will take place.
Type	Object type

- When you select an object in the decision table, the read-only text box below the decision table describes what changes were made to that object in the current repository.

Examples of the Results of Some Decision Choices

The following examples show the results of some decision choices when the current and modified repositories are different:

- If the Description column for an object contains Added to Current, the following are choices in the Decision column and their results:
  - Selecting Current keeps the addition in the current repository.
  - Selecting Modified (D) deletes the added object from the current repository.
- If the Description column for an object contains Deleted from Modified, the following are choices in the Decision column and their results:
  - Selecting Current keeps the repository as is without deleting the object.
  - Selecting Modified (D) deletes the object from the current repository.
- If the Description column for an object contains Deleted from Current, the following are choices in the Decision column and their results:
  - Selecting Current keeps the repository as is without adding the object back into the current repository.
  - Selecting Modified (A) adds the object back into the current repository.
- If the Description column for an object contains Added to Modified, the following are choices in the Decision column and their results:
  - Selecting Current keeps the repository as is without adding the object back into the current repository.
  - Modified (A). Selecting Modified (A) adds the object back into the current repository.

NOTE: If a decision choice is Mix, the choices apply to each property level on an object. Click the ellipses button to open the Properties dialog box.

The procedure in this section explains how to use the Merge Repository option of the Administration Tool to merge your repository customizations from a prior release with a new version of the Oracle BI repository.

To merge versions of the Oracle BI repository

1. In the Administration Tool, open the newly installed Oracle BI repository (for example SiebelAnalytics.Current.rpd) in offline mode.
2. From the Administration Tool menu bar, choose File > Merge.
3. In the Select Original Repository dialog box, select the repository received with your previous version of the software (for example, SiebelAnalytics.Original.rpd).
4. Type the password and click OK.
5. Click Select for the Modified Repository field.

6. In the Select Modified Repository dialog box, select the repository that contains the customizations you made to the previous version of the repository (for example, SiebelAnalytics.Modified.rpd).
7. Click Open, type the password, and then click OK.
8. In the Decision drop-down list, select the action you want to take regarding the repository change, or accept the default action.
9. To locate subsequent rows with empty Decision fields, click the Decision header cell.

   When all rows have a value in the Decision field, the Merge button is enabled.
10. Click Merge.

    A message appears to let you know that the merge is complete.
11. From the File menu, choose Save As, and save the current repository under a new name.

Exporting Oracle BI Metadata to IBM DB2 Cube Views

You can convert Oracle BI proprietary metadata to an XML file and import the metadata into your DB2 database. For more information, refer to "Using IBM DB2 Cube Views with Oracle BI" on page 277.

NOTE: The term IBM DB2 Cube Views is a registered trademark of IBM.

About Extracting Metadata Subsets Into Projects

A project consists of a discretely-defined subset of the metadata. Projects can consist of Presentation layer catalogs (subject areas) and their associated business model logical facts, dimensions, groups, users, variables, and initialization blocks. The Oracle BI Administrator creates projects so that developers and groups of developers can work on projects in their area of responsibility. The following is a list of some of the reasons you might wish to create projects:

- **Licensing.** Prior to releasing a new software version, you want to make sure that only the metadata that is relevant to the licensed application is in a project and that everything is consistent and complete. You accomplish this by adding only the fact tables that are relevant to the application.
- **Multiuser development.** During the development process, you want to split up the work (metadata) between different teams within your company. You accomplish this by extracting the metadata into projects so that each project group can access a different part of the metadata.

Project extractions are fact table centric. This makes sure that project extracts are consistent and makes licensing much easier to manage. The following is a list of the types of fact tables:

- **Simple (base) fact table.** Fact tables with nonderived columns or with columns that are derived but whose expressions are made up of constants.

- **Compound fact table.** Fact tables with derived columns whose components come from other fact tables
- **Component fact table.** Simple (base) fact tables that are part of a compound fact table. Their nonderived columns are used in the compound fact table.

About the Project Dialog Box

Target levels, list catalogs, and presentation catalogs cannot be added to or removed from projects.

Although, in the left pane, it appears that you can add a presentation catalog, you are actually adding only the underlying fact tables. The presentation catalogs appear as choices only to make it easier for you to add the elements you want in your project. Additionally, it adds any other objects that are necessary to make the extract consistent.

The user interface reflects what is happening in the code. In the Project dialog box, the left pane contains objects that you can use to create a project. The objects in the right pane are all the objects you chose (directly or indirectly) that reflect the complete set of data that makes each addition consistent. For example, if you select a presentation catalog to add to your project, underlying fact tables of other Presentation catalogs will be automatically be added if needed to make the extract consistent.

The following describes what you will see in the left pane of the Project dialog box:

- Only simple (base) fact tables and component fact tables appear.
- Compound fact tables do not appear.
- You can choose to group objects by Catalog or Business Model.
- When grouped by business model, the left pane displays only facts that belong to the business model.

The following describes what you will see in the right pane of the Project dialog box:

- Only simple (base) fact tables and component fact tables appear.
- If you add some but not all components of a compound fact table, the compound fact table will not appear on the right pane.
- Catalogs that will be extracted when you click OK.

About Converting Older Projects During Repository Upgrade

When you upgrade to the 10.1.3.2 version of Oracle BI the project definition is upgraded. During the upgrade, the project definition, presentation catalogs, target levels, list catalogs, and existing fact tables are automatically converted into simple (base) fact tables in the following way:

- Get presentation columns related to the target levels through the qualifying keys.
- Get presentation columns related to the list catalogs through the qualifying keys.
- Get presentation columns related to the presentation catalogs.
- Get all the base logical columns from all the presentation columns.
- Get all the base logical columns from the fact tables in the project.

- Get the base fact tables from all the base logical columns.

NOTE: After the upgrade, projects contain only simple (base) fact tables. All the security objects remain unchanged.

About Using Older Repositories After Upgrading

Occasionally, you might need to use an older version of the repository. To support this, the project definition might need to be converted to the old project definition. To downgrade the project definition, the fact tables will be converted into a project definition that contains presentation catalogs, target levels, list catalogs, and compound fact tables.

When you open a repository with a version number older than the 10.1.2.3 version, the following actions occur:

- Get all the presentation catalogs related to the fact tables.
- Get all the presentation columns under the presentation catalogs.
- Add all the target levels whose qualifying key columns are contained in the presentation column.
- Add all the list catalogs whose qualifying key columns are contained in the presentation column.
- Add the presentation catalogs to the project.

Setting up and Using the Oracle BI Multiuser Development Environment

Oracle BI allows multiple developers to work on repository objects from the same repository during group development of Oracle BI applications. The following are examples of how you might use a multiuser development environment:

- Multiple developers need to work concurrently on subsets of the metadata and then merge these subsets back into a master repository without their work conflicting with other developers. For example, after completing an implementation of data warehousing at a company, an administrator might want to deploy Oracle BI to other functional areas.
- A single developer might manage all development. For simplicity and performance, this developer might want to use an Oracle BI multiuser development environment to maintain metadata code in smaller chunks instead of in a large repository.

In both examples, the administrator creates projects in the repository file in the Administration Tool, and then copies this repository file to a shared network directory. Developers can check out projects, make changes and then merge the changes into the master repository.

When developers check out projects, the Administration Tool automatically copies and overwrites files in the background. Therefore, it is important for the administrator to perform setup tasks and for the developers to perform checkout and checkin procedures carefully, paying close attention to the messages that appear.

NOTE: To perform the tasks in this section, Oracle BI Administrators must understand the metadata creation process.

This section contains the following topics:

- Setting Up a Multiuser Development Environment (Administrator) on page 170
- Making Changes in a Multiuser Development Environment (Developers) on page 172
- Checking In Multiuser Development Repository Projects on page 175
- Viewing and Deleting History for Multiuser Development on page 178

Setting Up a Multiuser Development Environment (Administrator)

To prepare for multiuser development, the administrator performs the following tasks:

- Creating Projects for a Multiuser Development Environment on page 170. In the repository, create the projects that your developers need.
- Set Up the Shared Network Directory on page 171. Identify or create a shared network directory that will be dedicated to multiuser development.
- Copy the Master Repository to the Shared Network Directory on page 172. After creating all projects, you copy the repository file in which you created the projects to the shared network directory where it will be used as your master repository for multiuser development.

  NOTE: In this section, we use the phrase master repository to refer to this copy of a repository in the shared network directory.

Creating Projects for a Multiuser Development Environment

A project consists of a discretely-defined subset of the metadata. Projects can consist of Presentation layer catalogs (subject areas) and their associated business model logical facts, dimensions, groups, users, variables, and initialization blocks.

The Oracle BI Administrator creates projects in a repository and copies this repository to a shared network directory. A best practice is to create projects of manageable size based on individual logical star schemas in the business model. For Oracle BI projects that are just beginning, the best practice is to begin with a repository containing all the necessary physical table and join definitions. In this repository, you create a logical fact table as a placeholder in the Business Model and Mapping layer and a presentation catalog as a placeholder in the Presentation layer. As you add business model and presentation catalog metadata, new projects based on individual presentation catalogs and logical facts can be created.

NOTE: Only one person at a time can create projects in a master repository.

When creating a project, the Oracle BI Administrator selects a presentation catalog or a subset of logical fact tables related to the selected presentation catalog, and the Administration Tool automatically adds any business model and physical layer objects that are related. An object can be part of multiple projects.

After you create projects, they become part of the metadata and are available to multiple developers who need to perform development tasks on the same master repository. When defined this way, projects typically become a consistent repository after a developer checks out the projects and saves them as a new repository file. For more information about creating projects, see "About Extracting Metadata Subsets Into Projects" on page 167.

To create a project for a multiuser development environment

1. In the Administration Tool menu, choose File > Open > Offline.
2. In the Open dialog box, select the repository that you want to make available for multiuser development, and then click OK.
3. In the Administration Tool menu, choose Manage > Projects.
4. In the Project Manager dialog box, in the right panel, right-click and then select New Project.

   The left pane contains the objects that are available to be placed in a project. The right pane contains the objects that you select to be part of the project.
5. In the Project dialog box, type a name for the project.
6. Perform one of the following steps to finish creating the project:
   - In the Project dialog box, expand the catalogs and select one or more logical fact tables within the business model that are related to the presentation catalog, and then click Add.

     The project is defined as explicitly containing the logical fact tables and implicitly containing all logical dimension tables that are joined to the selected logical fact tables (even though they do not appear in the right pane).
   - In the Project dialog box, select a presentation catalog and then click Add.

     The Administration Tool automatically adds all the logical fact tables.
7. To remove any fact table from the project, in the right pane, select the fact table and click Remove.
8. Add any Catalogs, Groups, Users, Variables, or Initialization Blocks needed for the project.

   NOTE: You must add all developers who need to work on the project or they will not be allowed to check out objects.
9. Click OK.

Set Up the Shared Network Directory

After defining all projects and set up the shared network directory, the Oracle BI Administrator needs to identify or create a shared network directory that all developers can access, and then upload the new master repository to that location. This shared network directory needs to be used only for multiuser development. This directory typically contains copies of repositories that need to be maintained by multiple developers.

Developers create a pointer to this shared network directory when they set up the Administration Tool on their machines.

CAUTION: The Oracle BI Administrator must set up a separate, shared network directory that is dedicated to multiuser development. If not set up and used as specified, critical repository files can be unintentionally overwritten and repository data can be lost.

Copy the Master Repository to the Shared Network Directory

Make a copy of the master repository file and paste it in the directory that you have dedicated to multiuser development. Projects from this master repository will be extracted and downloaded by the developers who will make changes and then merge these changes back into the master repository.

After you copy the repository to the shared network directory, notify developers that the multiuser development environment is ready.

Making Changes in a Multiuser Development Environment (Developers)

Before checking out projects, the developers need to set up to the Administration Tool to point to the shared network directory containing the master repository. This must be the multiuser development directory created by the Oracle BI Administrator. For more information, see "Setting Up a Multiuser Development Environment (Administrator)" on page 170

During checkout and checkin, a copy of the master repository is temporarily copied to the developer's local repository directory (\Oracle BI\Repository by default). After checking out projects and making changes in a local repository file, each developer can check in (merge) changes into the master repository or discard the changes.

To make changes in a multiuser development environment, perform the following tasks:

- Setting Up a Pointer to the Multiuser Development Directory on page 172
- Checking Out Repository Projects on page 173
- About Changing and Testing Metadata on page 174

Setting Up a Pointer to the Multiuser Development Directory

Before checking out projects, each developer must set up their Administration Tool application to point to the multiuser development directory on the network. The Administration Tool stores this path in a hidden Windows registry setting on the workstation of the developer and uses it when the developer checks out and checks in objects in the shared directory.

NOTE: Until the pointer is set up, the multiuser options will not be available in the Administration Tool.

Initially, the network directory contains the master repositories. The repositories in this location are shared with other developers.

When setting up the pointer, the developer can also complete the Full Name field. Although the field is optional, it is recommended that the developer complete this field to allow other developers to know who has locked the repository. The Full Name value is stored in HKEY_CURRENT_USER in the registry, and is unique for each login.

To set up a pointer to the multiuser default directory

1. From the Administration Tool menu, choose Tools > Options.
2. In the Options dialog box, click the Multiuser tab.
3. In the Multiuser tab, next to the Multiuser development directory field, click Browse.
4. In the Browse For Folder dialog box, locate and select the multiuser development network directory, and then click OK.
5. In the Options dialog box, verify that the correct directory appears in the Multiuser development directory field.
6. In the Full Name field, type your complete name, and then click OK.

Checking Out Repository Projects

After setting up a pointer to the multiuser development default directory, a developer can check out projects, change metadata, and test the metadata. In the File > Multiuser submenu, the Checkout option is only available when there is a multiuser development directory defined in the More tab of the Options dialog box. For more information, refer to "Setting up and Using the Oracle BI Multiuser Development Environment" on page 169.

If a developer checks out a local repository and attempts to exit the application before publishing it to the network or discarding local changes, a message appears to allow the developer to select an action. For more information, refer to "About Closing a Repository Before Publishing It to the Network" on page 177.

To check out projects

1. From the Administration Tool menu, choose File > Multiuser > Checkout.
2. In the Multiuser Development Checkout dialog box, select the repository, and then click Open.

   This dialog box will not appear if there is only one repository in the multiuser development directory.
3. If prompted for your user name and password, in the Extract from dialog box, type your user name and password, and then click OK.

   If no projects exist in the repository, a message appears and the repository does not open.
4. In the Browse dialog box, select the projects that you want to be part of your project extract, and then click OK.

   If only one project exists in the master repository, it is selected automatically and the Browse dialog box does not appear.

5 In the New Repository dialog box, type a name for the new repository and then click Save.

   A working project extract repository is saved on your local machine. The name is exactly as you specified and is opened in offline mode. A log file is also created. The extracted repository might not be consistent.

   CAUTION: A second copy of the project extract repository is saved in the same location. The name of this version contains the word Original added at the beginning of the name that you assigned to the repository extract. Do not change the Original project extract repository. It will be used when you want to compare your changes to the original projects.

About Changing and Testing Metadata

Most types of changes that can be made to standard repository files are also supported for local repository files. Developers can add new logical columns, logical tables, change table definitions, logical table sources, and so on. Developers may also work simultaneously on the same project locally. It is important to note, however, that Oracle BI assumes the individual developer understands the implications these changes might have on the master repository. For example, if a developer deletes an object in a local repository, this change will be propagated to the master repository without a warning prompt.

The following modifications should not be made in a local repository:

- Hierarchy definitions. When modified concurrently by two developers, the changes will not be merged correctly during checkin.
- Project definitions. These should only be changed by the Oracle BI Administrator in the master repository.
- Physical Connection settings. These are intentionally not propagated and developers should not test in local environments.

After making changes to a local repository, the developer can edit the local NQSConfig.INI file, enter the name of the repository as the default repository, and test the edited metadata.

NOTE: DSNs specified in the metadata need to exist on the developer's workstation.

For more information about testing the metadata, refer to "Test and Refine the Repository" on page 154.

After the local developer makes changes, tests the changes, and saves the repository locally, the local developer can perform the following tasks from the File > Multiuser submenu:

- **Compare with Original.** Compares the working extracted local repository to the original extracted repository. When this option, the Compare repositories dialog box opens and lists all the changes made to the working extracted repository since you checked out the projects. For more information, see "Comparing Repositories" on page 161.
- **Discard local changes.** Anytime after check out and before check in, you can discard your changes. When you select this option, the working repository closes without giving you an opportunity to save your work.

  CAUTION: If you select this option, there is no opportunity to change your mind. For example, no Are You Sure? dialog box appears.

- **Merge local changes.** Locks the master repository on the network multiuser directory to allow you to check in your changes. For more information, see "Checking In Multiuser Development Repository Projects" on page 175.
- **Publish to the network.** After you successfully merge your changes, the master repository opens locally and the Publish to Network submenu item is available. When you select this option, the lock is removed, the repository is published, and the repository closes. For more information, see "Checking In Multiuser Development Repository Projects" on page 175.

Checking In Multiuser Development Repository Projects

After changing and testing the metadata on a local machine, the developer needs to merge the local changes into the local master check in the projects into the master repository on the shared network directory. Only one developer at a time can merge metadata from a local repository into the master repository. Therefore, the master repository is locked at the beginning of the merge process.

About Checking-In Projects

When the check-in process begins, the following actions occur:

- The Administration Tool determines if the master repository is currently locked. If not, it locks the master repository, preventing other developers from performing a merge until the current merge is complete, and records the lock in the log file.
- For other developers, the Merge Local Changes option on the File > Multiuser menu will be unavailable until the current check-in process has been successfully completed.
- The Administration Tool automatically copies the current version of the master repository from the shared network directory to the \\Oracle BI\Repository directory on the developer's machine and updates the log files in the local and shared network directories. This is necessary because the master repository in the shared network directory might have changed after the developer checked out the projects.

About Merging Multiuser Development Metadata

The merge process involves the following files:

- **Original of the local (subset) repository.** Contains the state of the projects as originally extracted. This repository name begins with Original. An example of the file name for this copy might be OriginalDevelopment2.rpd. This version is stored in the same location as the modified (or working) version of the local repository.
- **Modified local (subset) repository.** Contains the extracted projects after being modified by the developer. This version is stored in the same location as the original version of the local repository.
- **Master repository in network shared directory.** This may have been modified by other developers before this merge. (For example, Master_SiebelAnalytics.rpd.)

During the merge, the Administration Tool checks for added objects and if found, a warning message appears. The following list describes what happens during this step:

- Warning about added objects. When a person checks out a project, they have the ability to modify that project in any way and check it back in. Deletions and modifications are ways in which the integrity of the project is maintained. However, adding objects might introduce objects into the repository that do not belong to any project. Therefore, all project related objects are checked and if a new object is found, a warning message appears.
- Aggregation of related objects. In the warning message, only the parent object is reported. The Administration tool aggregates all the objects to make the message more usable. For example, if a developer added a new business model, only the business model appears in the warning message to the user, not the tables, columns, dimensions, and so on.

When the developer closes the Administration Tool, the following actions occur:

- The master repository on the shared network directory is overwritten with the master repository containing the developer's changes.
- The [*master repository*].lck file is deleted. If another developer checks out the changed project from the master repository, the changes made by the first developer are exposed to the other developer.

  CAUTION: The Oracle BI Administrator needs to add newly created metadata to the project definition in the master repository for it to be visible in future extracted versions of the project. For example, if a developer checks out a project, adds a new object, and then checks it in, the new object will not be visible in extracted versions of the project until it is explicitly added to the project definition. For instructions, refer to "Creating Projects for a Multiuser Development Environment" on page 170.

Tracking Changes to the Master Repository

A summary of the development activities on the master repository is in the [*master_repository*].log. This log contains a record of the following activities:

- Projects that have been checked in and checked out and when these actions occurred
- The NT login name and computer name initiating the transaction
- When locks are created and removed

Differences Between the Multiuser Merge and Standard Repository Merge Processes

The multiuser development check-in process uses the same technology as the standard repository merge process with a few important differences. For more information about the standard repository merge, refer to "Merging Oracle BI Repositories" on page 163.

The following list describes the differences that occur during a multiuser development merge:

- Inserts are applied automatically. Because a subset of the master repository is being used as the original repository, most objects in the master repository appear to be new. This would result in many unnecessary prompts that the developer would have to manually approve. Therefore, inserts are applied without a prompt during a multiuser development merge.
- Conflicts that are not inserts but are resolved as a result of the automatic inserts are applied without a prompt during a multiuser development merge.

- The database and connection pool properties on the server take precedence over the same properties on the developer's machine. This precedence are applied without a prompt during a multiuser development merge.

About Closing a Repository Before Publishing It to the Network

If you attempt to close an unpublished, locked repository without selecting one of the options in the File > Multiuser submenu, the Closing MUD repository dialog box opens with the following options:

- **Publish to Network.** Publishes the merged repository to the network share as the new master, releases the lock on the master, and the event is logged. This option is available after a Merge Local Changes event occurs. This option is also available on the File > Multiuser submenu.
- **Discard Local Changes.** Releases the lock on the master repository and records the event in the log. This option is available after a Checkout or Merge Local Changes is performed. available on the File > Multiuser submenu.
- **Close repository and keep lock.** This closes the repository leaving the master repository locked.

To check in projects to the master repository

1. From the Administration Tool menu, choose File > Multiuser > Merge Local Changes.
2. In the Lock Information dialog box, in the Comment field, type a description of the changes that you made, and then click OK.
3. In the Merge repositories dialog box, verify that the Original local repository and Modified local repository file names are correct.

   In the Merge repositories dialog box, it might appear that there are no differences between the repositories. However, what this means is that there are no decisions that have to be explicitly made by the developer to check-in changes.
4. For an overview of the merge decisions that will take place, click Stats.
5. In the Results dialog box, click close.
6. Click Merge.
7. If asked if you want to check global consistency, click Yes or No.

   The changes are merged and the merged local repository file opens. In the developer's local directory, a CSV file is created that contains details of the merged changes.
8. Verify that the changes made in the modified local repository are reflected in this merged local repository.

   CAUTION: The merged repository has the same name as the shared repository, but this is still a local copy.

9 After you confirm all the changes, click Save.

 This saves the merged repository locally, and then, uploads this repository to the shared network directory with a 000 file extension. For example, Master_Sales.000.

 At this point, the changes made by the developer are still not saved to the master repository in the shared network directory.

10 To commit these changes to the master repository in the shared network directory, close the Administration Tool.

11 In the Closing MUD repository dialog box, select Publish to Network.

 The master repository on the shared network directory is overwritten with the copy of the repository containing the developer's changes. For more information about the options in this dialog box, see "About Closing a Repository Before Publishing It to the Network" on page 177

Viewing and Deleting History for Multiuser Development

You can view the development history of a multiuser development repository. In the Administration Tool, multiuser development history is only available when no repository is open and after the administrator sets up the shared network directory. This prevents the confusion that could occur if a user opened a history log that did not match an open, unrelated repository.

To view multiuser development history

1 Open the Administration Tool.

2 Without opening a repository, from the Administration Tool menu, choose File > Multiuser > History.

3 In the Multiuser Development History dialog box, select a repository.

 A list of all master repositories in the multiuser development directory appears. If directory contains only one master repository, it is selected by default, and no list appears.

4 In the Open Offline dialog box, type the password for the repository.

5 In the Multi User History dialog box, right-click a row and select one of the following items:

Action	Description
View Repository	Loads the selected master version of the repository in the Administration Tool in read-only mode.
View Projects	Loads the selected version of a modified subset repository in the Administration Tool in read-only mode.

Action	Description
View Conflict Resolution	Loads all necessary repositories of the selected version. Also shows the Merge dialog box in read-only mode with all selected decisions as they were during the Merge Local Changes activity at that time. **NOTE:** The Conflict Resolution check box has to be selected in the dialog, in order to this menu item to be enabled (otherwise, there is nothing to show - there were no decisions decided by the user).
View Details	Displays a log with details for the selected versions, or displays details for all versions if no specific versions are selected.
View Changes	Compares modified subset repository of the selected version with original subset repository and shows all changes made by the user in the selected version.
Find and Find Again	Allows you to search the list.
Select All	Selects all items displayed in the dialog.
Delete	Available only to the Administrator.

Deleting Multiuser Development History

Only the Administrator can delete history. Administrators are defined in a special hidden option file in multiuser development directory. This option file has the following properties and characteristics:

- Must have the HIDDEN flag turned on.
- Can have network access privileges assigned only to multiuser development administrators.
- Has the same base name as the master repository, but the extension is OPT. For example, for \\network\MUD\sales.rpd, the administrator might create the hidden file \\network\MUD\sales.opt.
- The OPT file is a TXT file in the format:

  ```
  [Options]

  Admin=admin1;admin2
  ```

  Administrators are defined by their network login names. When multiple administrators exist, administrator names are separated by semicolons. For example:

  ```
  [Options]

  Admin=jsmith;mramirez;plafleur
  ```

An administrator can delete the entire MUD history or the oldest 1 to n versions. It is not possible to delete versions in the middle of the range. For example, an administrator cannot delete version 3 if there are still versions 2 and 1. If an administrator deletes the entire MUD history, newly assigned version numbers restart at version 1. If one or more versions are not deleted, the newly assigned version numbers continue from.

Setting Up the Repository to Work with Delivers

The Oracle BI Presentation Services needs to deliver alerts from Delivers to entire groups and to specified email addresses, phone numbers, and so on. Delivers uses a tool called *iBot* to deliver alerts. iBots are software-based agents driven by a schedule or events that can access, filter, and perform analysis on data based upon defined criteria. For more information about iBots, refer to the chapter about using Delivers in *Oracle Business Intelligence Answers, Delivers, and Interactive Dashboards User Guide*.

You need to set up a presentation catalog (called subject area in Answers) with the name SA System in the Presentation layer of the Oracle BI repository for this to function correctly.

NOTE: For information about setting the BI Presentation Server parameters, refer to *Oracle Business Intelligence Presentation Services Administration Guide*.

This section contains the following topics:

- About the SA System Subject Area on page 180
- Setting Up the SA System Subject Area on page 181

About the SA System Subject Area

In Oracle BI, data visibility can be defined by the groups to which a user belongs. For example, when Oracle BI applications customers log on, the GROUP system session variable can be populated and the user sees certain subject areas and columns depending on the groups that exist in the variable. Although the GROUP system session variable provides Oracle BI with the groups to which each user belongs, it does not identify the users that are in each group.

SA System is a presentation catalog (called subject area in Answers) that addresses this data visibility issue by exposing users in a group to Delivers. It also allows contact information such as email addresses to be retrieved from a database and used as delivery devices in Delivers. This allows Oracle BI Administrators to set up the Oracle BI Server to automatically populate delivery devices and profiles for users instead of requiring users to update their My Account screen in Delivers.

Group membership is often maintained in an external database such as Oracle's Siebel transactional database and not in the Oracle BI repository. This information can be propagated to the Oracle BI Server and Oracle BI Presentation Services through the GROUP session variable. The SA System subject area provides group membership and external email addresses to Delivers when used in conjunction with the GROUP session variable.

Setting Up the SA System Subject Area

The Oracle BI Presentation Services is preconfigured to look for the SA System subject area on startup and will use it if it exists. You can import any external schema that holds group membership, users' primary email addresses, and users' SMTP devices (cell phones, handhelds, and so on), and then map it to the SA System subject area.

You can add columns to the SA System table (for example, adding a provider name) by extending the underlying physical tables and columns.

This section contains the following topics:

- Guidelines for Implementing the SA System Subject Area on page 181
- Setting Up the SA System Subject Area for a Stand-Alone Implementation on page 181

Guidelines for Implementing the SA System Subject Area

The name for the subject area must always be SA System. If you use the SA System subject area, the following set-up guidelines apply:

- **Users and groups.** Every user and group must be present in the data. Oracle BI does not support group membership through a mix of internal Oracle BI repository users and external users in the SA System subject area.
- **Delivery profile.** It is not recommended for the user to set up a delivery profile because there is no way for Oracle BI Administrators to control this profile. For example, the Oracle BI Administrator loses the ability to perform a mass update of email addresses. If a user does set up a delivery profile, the delivery profile will take precedence over what is shown in the SA System subject area.
- **Authorization and authentication.** This option affects what users are allowed to do in the system (authorization), not who users are (authentication). For related information about database authentication, refer to "About Oracle BI Delivers and Database Authentication" on page 328.

  CAUTION: An existing configuration option (the Alerts element) in instanceconfig.xml controls how logon names are treated before accessing the SA System subject area. For example, login names might be converted to all uppercase. For more information, refer to *Oracle Business Intelligence Presentation Services Administration Guide.*

- **Security settings.** The SA System subject area only needs to have read-access to the Administrator account and as a result, security settings are not compromised. If group membership is seen as privileged data, you can allow only the Oracle BI Administrator to have access to this subject area.

Setting Up the SA System Subject Area for a Stand-Alone Implementation

Oracle BI standalone customers also need to create and populate the SA System subject area in the Oracle BI repository.

Customers who install the Oracle BI Server in a standalone environment (without Oracle BI Applications) must perform the following tasks in the order shown:

- Create tables and columns in your data source (for example your external database).
- Create and build the subject area in the Oracle BI repository.
  - Import schema that stores the appropriate information. For instructions, refer to "Creating and Administering the Physical Layer in an Oracle BI Repository" on page 55.
  - Map the tables and columns from the Physical Layer to the Business Model and Mapping layer. For instructions, refer to "Creating and Administering the Business Model and Mapping Layer in an Oracle BI Repository" on page 109.
  - Map the tables and columns from the Business Model and Mapping layer to the Presentation layer. For instructions, refer to "Creating and Maintaining the Presentation Layer in an Oracle BI Repository" on page 143.

    The Presentation layer metadata needs to contain the SA System folder, User table, and columns.

NOTE: For Oracle BI Application customers, the SA System subject area is preconfigured in their Oracle BI repository.

8 Oracle BI Administration Tool Utilities and Expression Builder

This section describes the various utilities and wizards contained in the Administration Tool. It also describes the Expression Builder and provides instructions for creating constraints, aggregations, and other definitions within a repository.

This section contains the following topics:

- Utilities and Wizards on page 183
- Expression Builder on page 190

Utilities and Wizards

The Administration Tool provides a number of wizards and utilities to aid you in performing various tasks.

This section provides a description of the following utilities and wizards:

- Replace Column or Table Wizard on page 183
- Oracle BI Event Tables on page 184
- Externalize Strings on page 184
- Rename Wizard on page 185
- Update Physical Layer Wizard on page 185
- Generating Documentation of Repository Mappings on page 186
- Generating and Deploying a Metadata Dictionary on page 187
- Removing Unused Physical Objects on page 189
- Aggregate Persistence Wizard on page 189
- Calculation Wizard on page 190

Replace Column or Table Wizard

The Replace Wizard automates the process of replacing physical columns or tables in logical table sources by allowing the Oracle BI Administrator to select the sources from those displayed. The wizard prompts the Oracle BI Administrator to replace columns as well as tables.

To start the Replace Column or Table Wizard

1. From the Tools menu, choose Utilities > Replace Column or Table in Logical Table Sources
2. Click Execute.

Oracle BI Event Tables

This utility allows you to identify a table as an Oracle BI event polling table. An event polling table is a way to notify the Oracle BI Server that one or more physical tables have been updated. Each row that is added to an event table describes a single update event. The cache system reads rows from, or polls, the event table, extracts the physical table information from the rows, and purges cache entries that reference those physical tables. For more information about event tables, refer to "Cache Event Processing with an Event Polling Table" on page 248.

To start the Oracle BI Event Tables utility

- From the Tools menu, choose Utilities > Oracle BI Event Tables
- Click Execute.

Externalize Strings

This utility is primarily for use by translators to translate Presentation layer catalogs, tables, columns, and their descriptions into other languages. You can save these text strings to an external file with ANSI, Unicode, and UTF-8 coding options.

CAUTION: When using this utility, translators should work closely with Oracle BI contacts to verify that the correct process is followed for each situation.

To translate a string

1. In the Presentation layer, right-click a presentation catalog and choose Externalize Display Names.

   NOTE: The term Presentation Catalog in the Administration Tool refers to subject areas.

2. In the Presentation layer, right-click the same presentation catalog and choose Externalize Descriptions.

   In the right-click menu, both options appear with a check next to them.

3. From the Tools menu, choose Utilities > Externalize Strings, and then click Execute.
4. In the Externalize Strings dialog box, click a presentation catalog in the left pane.

   NOTE: You can select all catalogs at once or select them individually and create a separate string file for each one.

   In the right pane, the translated values and the original strings (names) appear. These will be placed in session variables for use by Oracle BI Presentation Services.

5. Click Save.
6. In the Save As dialog box, choose a type of file and an encoding value and click Save.
7. In the Externalized Strings dialog box, click Close.
8. (Optional) To clear the check marks next to these options, right-click the presentation catalog and click each option.

Rename Wizard

The Rename Wizard allows you to rename presentation and Business Model and Mapping layer tables and columns. It provides a convenient way to transform physical names to user-friendly names.

NOTE: Renaming the presentation layer columns will reset the Use Logical Column Name property to false. It is recommended that you rename the business model layer logical columns instead.

To start the Rename Wizard

- From the Tools menu, choose Utilities > Rename Wizard, and then click Execute.

Update Physical Layer Wizard

This wizard allows you to update database objects in the Physical layer of a repository based on their current definitions in the back-end database.

NOTE: This wizard is not available for repositories that are opened in read-only mode, because they are not available for updating.

When the wizard processes the update, the server running the Administration Tool connects to each back-end database. The objects in the Physical layer are compared with those in the back-end database. Explanatory text alerts you to differences between objects as defined in the database in the Physical layer and as defined the back-end database, such as data type-length mismatches and objects that are no longer found in the back-end database. For example, if an object exists in the database in the Physical layer of the repository but not in the back-end database, the following text is displayed:

```
Object does not exist in the database
```

NOTE: The wizard does not add columns or tables to the repository that exist in the back-end database but not in the repository. Additionally, the Wizard doesn't update column Key assignments. It checks that there is a column in the repository that matches the column in the database, and then, if the values don't match, the wizard updates the type and length of the column in the repository.

The connection pool settings for each database need to match the connection pool settings used when the objects were last imported into the Physical layer from the back-end database. For example, for Oracle, the connection pool may be set to native OCI, but an Oracle ODBC source must be used for the update. In this case, you would set the connection pool to the Oracle ODBC setting used for the import. For more information about connection pool settings, refer to "Setting Up Connection Pools" on page 65.

To update objects in the Physical layer

1. From the Tools menu, choose Utilities > Update Physical Layer, and then click Execute.

   The databases in the Physical layer of the repository are listed in the left pane of the wizard.

2. In the Update Physical Layer Wizard dialog box, select the databases that you want to update in the left pane, and then click Add.

   The databases move to the right pane.

3. To remove a database from the update list, select it and click Remove.
4. After you select the objects that you want to update in the Physical layer, click Next.
5. In the next window, select the connection pool for each database that you want to update and then click Next.

   The wizard alerts you if it needs to check any objects out.

6. Review the information about each update.

   NOTE: You can sort the rows (toggle between ascending order and descending order) by clicking the Name column heading.

7. If you decide that you do not want the wizard to update a particular object in the Physical layer, click the Back button and remove the object.
8. Click Finish.

   The wizard updates the objects in the Physical layer, and then closes automatically.

9. On the Administration Tool toolbar, click File > Save to save the updated objects in the Physical layer.

Generating Documentation of Repository Mappings

The Repository Documentation utility documents the mapping from the presentation columns to the corresponding logical and physical columns. The documentation also includes conditional expressions associated with the columns. The documentation can be saved in comma separated (CSV) or tab delimited (TXT) format.

The Repository Documentation utility allows you to extract Oracle BI metadata to a flat file so that it can be loaded into Excel and RDBMS. You can query the resulting file to answer questions such as “If I delete physical column X, what logical columns will be affected?” or “How many places in the business model refer to the physical table W_SRVREQ_F”. Then you can establish dependency relationships among elements in the repository.

Excel only allows data sets of 1,000,000 rows. You might exceed this in a large repository. Therefore, you may wish to run the Repository Documentation utility on a subset of the repository by extracting relevant business models into a new project. For more information, see “Setting up and Using the Oracle BI Multiuser Development Environment” on page 169.

The Repository Documentation utility creates a comma-separated values file or a tab-separated values file that shows the connections between the presentation and physical layers in the current repository. This file can be imported into a repository as a physical layer.

NOTE: This file excludes any information about repository variables and marketing objects.

To run the Repository Documentation utility

1. From the Tools menu, choose Utilities.
2. In the Utilities dialog box, select Repository Documentation, and then click Execute.
3. In the Save As dialog box, choose the directory where you want to save the file.
4. Type a name for the file.
5. Choose a type of file and an Encoding value and click Save.

   Current encoding options are ANSI, Unicode, and UTF-8.

Generating and Deploying a Metadata Dictionary

When using Oracle BI, you might need to obtain more information about metrics or attributes for repository objects. For example, you might need to resolve issues caused by confusing metadata object names or to obtain more details when an attribute is derived in a complicated way. A metadata dictionary can help you resolve conflicting information and understand the metrics and attributes of repository objects.

A metadata dictionary is a static set of XML documents. Each XML document describes a metadata object such as a physical table or a logical column, including its properties and relationships with other metadata objects.

These XML documents are viewed using index files in a browser. Therefore, the metadata dictionary needs to be located on the BI Presentation Server. If you know that location before you generate the dictionary, you can specify the location in the Save As dialog box before you generate the dictionary files and folders. If you obtain the location after you generate the dictionary, you can copy the files to the desired location at that time.

The dictionary does not change dynamically as repository changes are made. Therefore, you will need to generate the dictionary periodically to update the content.

Generating a Metadata Dictionary

When you generate the dictionary, you can set the output location to the final location or to a temporary location.

CAUTION: Repositories can be large (containing tens of thousands of objects). Generating a dictionary for a large repository can take a significant period of time.

You can select a destination for your dictionary in the following ways:

- Select a local or network location when you generate the dictionary. When the dictionary is generated, a subdirectory with the same name as the repository will be created in that location. The dictionary folders and files will be created in that subdirectory.

  For example, If you select J:\BI_DataDictionary\ and your repository name is demo1.rpd, the dictionary files, including the style sheets, will be located in the following location:

  J:\BI_DataDictionary\demo1\

  NOTE: If the dictionary you wish to use has been generated, copy the entire folder to the desired location.

- If you want to use an IIS virtual directory, you can create or select a virtual directory in IIS before you generate the dictionary. When you generate the dictionary, choose the physical directory associated with the IIS virtual directory.

  NOTE: If dictionary has been generated, copy the entire folder to the physical directory associated with the IIS virtual directory.

After you generate a metadata dictionary, style sheets and index files are created for that dictionary.

The related style sheets (XSL files) are created and stored in the following location:

[drive]:\[path]\[repository name]\xsl

A name index and tree index are created and stored in the [drive]:\[path]\[repository name] root folder. You can use following URLs to locate and view objects:

- http://<hostname>:<portname>/<repository name>/NameIndex.xml. Allows you to search for repository objects by name.
- http://<hostname>:<portname>/<repository name>/TreeIndex.xml. Allows you to drill down from top-level objects to the object that you want to view.

NOTE: Each index file contains a link to the other so that you can quickly switch views.

To generate the metadata dictionary

1. Open the repository in Offline mode.

   You cannot generate the metadata dictionary in Online mode.

2. From the Tools menu, choose Utilities > Generate Metadata Dictionary, and then click Execute.
3. In the Choose Directory dialog box, click Browse to locate and select the location where you want to store the dictionary.

   You should receive the following message: Metadata dictionary has been successfully created in [drive]:\[path]\

4. Click OK.

Removing Unused Physical Objects

Large repositories use more memory on the server and are harder to maintain. Additionally, development activities take longer on a large repository. This utility allows you to remove objects that you no longer need in your repository. You can remove databases, initialization blocks, physical catalogs, and variables.

To remove unused physical objects

1. From the Tools menu, choose Utilities > Remove Unused Physical Objects, and then click Execute.
2. In the Remove Unused Physical Objects dialog box, from the Type drop-down list, select the type of object.
3. In the list of objects, verify that only the objects that you want to remove are checked.

   Below the list of objects, the number of checked and the total number of objects appears.
4. To remove the checked objects, click Yes.
5. To cancel, click No.

Aggregate Persistence Wizard

Aggregates are created and persisted in the Oracle BI Server metadata as well as in the back-end databases. This section describes how you can use the Aggregate Persistence Wizard to create the SQL file that will be used to create aggregate tables and map them into the metadata.

Guidelines for Using the Aggregate Persistence Wizard

The traditional process of creating aggregates for Oracle BI Server queries is a manual process, requiring that you write complicated DDL and DML files that create and populate tables in the databases. Additionally, these tables need to be mapped into the repository metadata to be available for queries. This is a time-consuming and potentially error-prone process. The Aggregate Persistence Wizard allows the Oracle BI Administrator to automate the creation of these aggregate tables and their mappings into the metadata. The following is a list of the guidelines you should follow when using the Aggregate Persistence Wizard:

- From the Tools menu, choose Utilities > Aggregate Persistence Wizard, and then click Execute.
- In the Select file location dialog box, specify the complete path and file name of the aggregate creation script. You can specify a new or an existing file name. If you want to generate a DDL file, select the Generate DDL file check box.
- In the Select Business Model & Measures dialog box, select the items.

  NOTE: The View Script button is not available during the creation of the first aggregate table block.

- In the Select Dimensions & Levels dialog box, expand the window to view all columns. You might want to specify a surrogate key to be used for the fact-dimension join.

  The default join option between the fact and level aggregate tables is to use the primary keys from the level aggregate. If the primary key of the level is large and complex, the join to the fact table will be expensive. A surrogate key is an artificially generated key, usually a number. For example, a surrogate key in the level aggregate table would simplify this join, removing unnecessary (level primary key) columns from the fact table and resulting in a smaller fact table.

- In the Select Output Connection Pool, Container & Name, select the items. A default aggregate table name will be provided and a prefix (defined in NQSConfig.INI) is added to the file name.
- In the Aggregate Definition dialog box, the View Script button becomes available for use, the logical SQL appears for your review, and you have a choice of defining another aggregate (default) or ending the wizard.
- In the Complete Aggregate Script dialog box, the complete path and file name appears.

For information about using the SQL file to create aggregate tables, refer to "Creating Aggregates for Oracle BI Server Queries" on page 241.

Calculation Wizard

You use the Calculation Wizard to create new calculation columns that compare two existing columns and to create metrics in bulk (aggregated), including existing error trapping for NULL and divide by zero logic.

You start the Calculation Wizard in the Business Model and Mapping layer by right-clicking any logical fact or dimension column of data type numeric, and then selecting the option Calculation Wizard. For information about setting up the Calculation Wizard, refer to "Using the Options Dialog Box—General Tab" on page 29.

Expression Builder

You can use the Expression Builder dialog boxes in the Administration Tool to create constraints, aggregations, and other definitions within a repository. The expressions you create with the expression builder are similar to expressions created with SQL. Except where noted, you can use all expressions constructed with the expression builder in SQL queries against the Oracle BI Server.

For information about using SQL with the expression builder, refer to "SQL Syntax and Semantics" on page 355. For information about the SQL functions supported by the Oracle BI Server, refer to "SQL Reference" on page 366.

This section includes the following topics:

- About the Expression Builder Dialog Boxes on page 191
- Expression Builder Toolbar on page 192
- Folders in the Selection Pane on page 193
- Example of Setting Up an Expression on page 195

- Navigating Within the Expression Builder on page 196
- Building an Expression on page 196
- About Time Series Conversion Functions on page 197

About the Expression Builder Dialog Boxes

You can access the expression builder from the following dialog boxes:

- Logical Table Source—Content tab
- Logical Table Source—Column Mapping tab
- Logical Column—General tab
- Logical Column—Aggregation tab
- Logical Foreign Key
- Physical Foreign Key
- Session Variable—Variable tab
- Static Repository Variable—Variable tab

When creating expressions in the Expression Builder dialog boxes, you can search the categories pane and building blocks pane. When you type a value into the search box, it filters out the non-matching strings and only the ones that match will appear. After typing search criteria in a search box, you can move up and down the list using arrows and tab between the first search box and the second search box. To return to the full list of results, you delete the search string from the Search field.

When you locate the item you want to insert into the expression, select it and click Insert in the dialog box or press Enter on your keyboard. The item you selected will appear in the expression in the expression box.

When you first open the Expression Builder dialog box, the items are not sorted. When checked the Sort Panes check box sorts all items in the panes. As soon as you select the check box, the panes are automatically redrawn without changing the contents of the panes or your filtering criteria.

Figure 15 on page 192 shows an example of an expression builder and the dialog box contains the following sections:

- The edit pane at the top of the dialog box allows you to edit the current expression.
- The toolbar in the middle of the dialog box contains commonly used expression building blocks.
- In the lower half of the dialog box, the left pane is the Selection pane. It displays the folders that are appropriate for the dialog box from which you accessed the expression builder.
- The lower middle pane is the Categories pane. It displays the available categories for the folder you select in the Selection pane. The Search field below the middle pane allows you to search for a value in the middle pane.

- The lower right pane is the Building Blocks pane. It displays the individual building blocks for the category you select in the Category pane. The Search field below the right pane allows you to search for a value in the right pane.

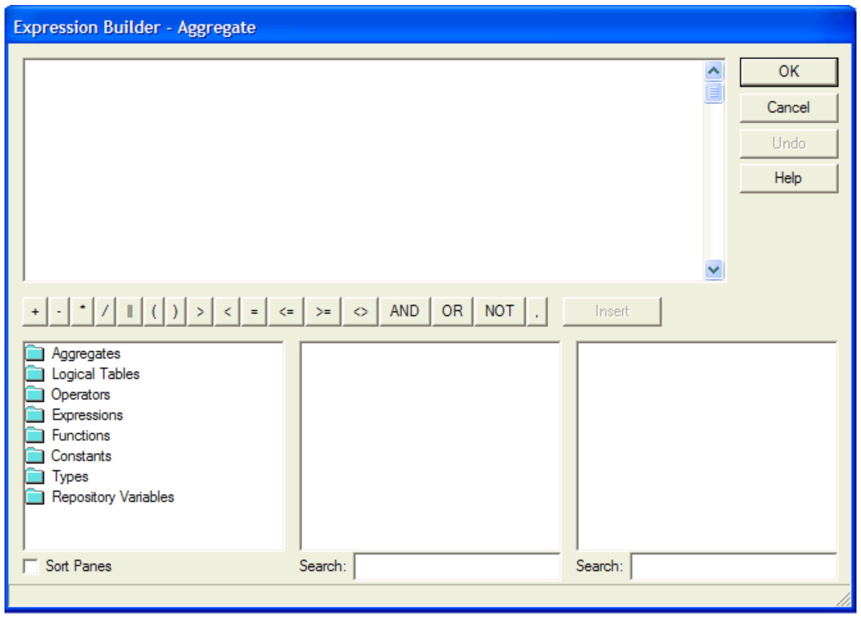


Figure 15. Example Expression Builder

Expression Builder Toolbar

The toolbar is located in the middle portion of the expression builder. Table 22 on page 192 describes each icon and its function in an expression.

Table 22. Expression Builder Toolbar

Operator	Description
+	Plus sign for addition.
-	Minus sign for subtraction.
*	Multiply sign for multiplication.
/	Divide by sign for division.
||	Character string concatenation.
(	Open parenthesis.
)	Close parenthesis.
>	Greater than sign, indicating values higher than the comparison.
<	Less than sign, indicating values lower than the comparison.
=	Equal sign, indicating the same value.

Table 22. Expression Builder Toolbar

Operator	Description
<=	Less than or equal to sign, indicating values the same or lower than the comparison.
>=	Greater than or equal to sign, indicating values the same or higher than the comparison.
<>	Not equal to, indicating values higher or lower, but not the same.
AND	AND connective, indicating intersection with one or more conditions to form a compound condition.
OR	OR connective, indicating the union with one or more conditions to form a compound condition.
NOT	NOT connective, indicating a condition is not met.
,	Comma, used to separate elements in a list.

Folders in the Selection Pane

The folders that appear in the Selection pane vary based on the dialog box from which you accessed the expression builder. This section describes the folders that may appear.

Aggregate Content

The Aggregate Content folder contains the available aggregate functions. Aggregate sources must use one of the functions listed here to specify the level of their content.

Dimensions

The Dimensions folder contains the dimension configured in the business model. If no dimension exists in a business model, or if the dimension folder is not pertinent to a particular expression builder, the Dimension folder is not displayed.

When you select the Dimensions folder, each configured dimension displays in the middle pane, and each level for the selected dimension displays in the right pane.

Logical Tables

The Logical Tables folder contains the logical tables configured in the business model. If logical tables are not pertinent to a particular expression builder, the Logical Tables folder is not displayed.

When you select the Logical Tables folder, each logical table in the business model displays in the middle pane, and each column for the selected logical table displays in the right pane.

Operators

The Operators folder contains the available SQL logical operators.

Expressions

The Expressions folder contains the available expressions.

Functions

The Functions folder contains the available functions. The functions that appear depend on the object you selected.

Constants

The Constants folder contains the available constants.

Types

The Types folder contains the available data types.

Repository Variables

This folder contains the available repository variables. If no repository variables are defined, this folder does not appear.

Session Variables

This folder contains the available system session and non system session variables. If no session variables are defined, this folder does not appear.

Example of Setting Up an Expression

Figure 16 on page 195 shows the expression builder for a derived logical column.

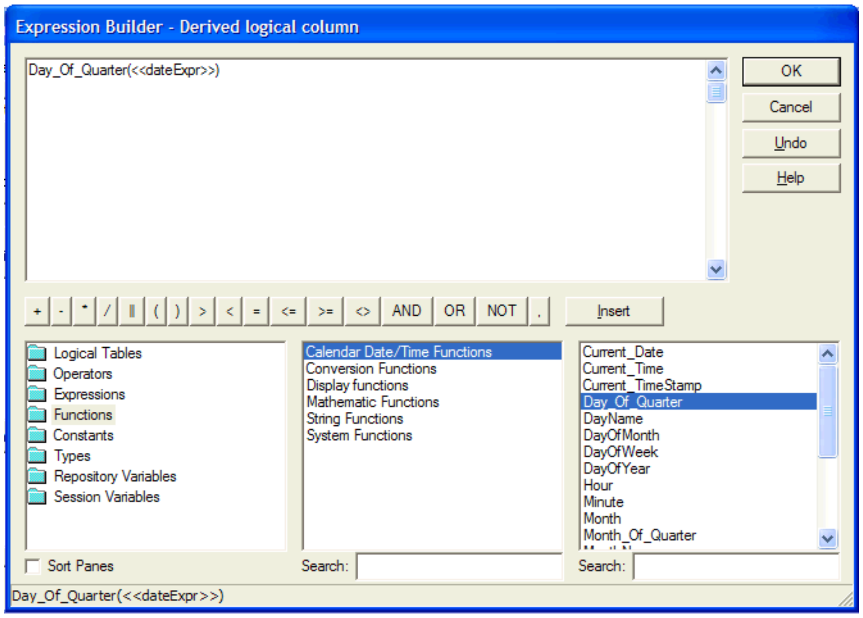
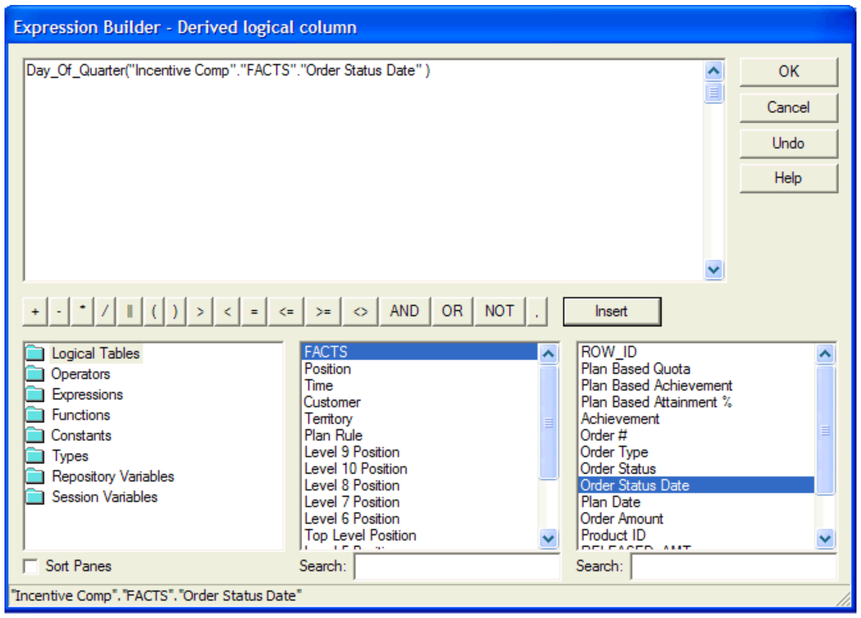


Figure 16. Expression Builder for Derived Logical Columns

Select the Functions folder in the left pane. Double-click the function in the right pane to paste the function in the edit box. In the expression builder’s edit box, click once between the parentheses of the function to select the area as the insertion point for adding the argument of the function.

Double-click the logical column to paste the logical column at the insertion point as the argument of the function. Figure 17 on page 196 shows where the expression appears in the window.



Day_Of_Quarter("Incentive Comp"."FACTS"."Order Status Date")

"Incentive Comp"."FACTS"."Order Status Date"

Figure 17. Example Logical Column Function in the Editing Pane

Navigating Within the Expression Builder

Use the following procedure to navigate within an Expression Builder dialog box.

To navigate within an Expression Builder

1. In the Selection pane, select the appropriate folder for the type of expression you want to build.

   The available categories for the folder appear in the Categories pane.

2. Select the appropriate category for the expression you want to build.

   The available building blocks for that category appear in the Building Blocks pane.

3. Double-click a building block to move it into the Editing pane.

4. To insert an operator into the expression, double-click an operator on the Expression Builder toolbar.

Building an Expression

Use this procedure to build an expression in the Expression Builder dialog box.

To build an expression

1. Navigate to the individual building blocks you want in the expression.

   The Syntax bar at the bottom of the Expression Builder dialog box shows the syntax for the expression.

2. Add the building blocks to the Editing pane.
3. Edit the building blocks to reflect the expression you want.
4. Use the Expression Builder toolbar to insert operators into the expression.
5. Repeat the preceding steps until the expression is complete, and then click OK.

   The Administration Tool displays a message for any syntax errors in the expression. When the expression is syntactically correct, the Administration Tool adds the expression to the dialog box from which you accessed the Expression Builder.

About Time Series Conversion Functions

Time series functions operate on time-oriented dimensions. To use these functions on a particular dimension, you have to designate the dimension as a Time Dimension and set one or more keys at one or more levels as Chronological keys. This identifies the dimension as having a monotonically increasing value in time (corresponds to chronological order).

NOTE: It is required that you define a chronological key at a level that can be used to answer your time series query. It is recommended that you define additional chronological keys at other relevant levels for performance reasons.

Currently, AGO and TODATE are the types of time series conversion functions. Currently, you may only enter AGO and TODATE functions in the Expression Builder in the Administration Tool. You cannot use them in coded SQL.

The Ago and ToDate functions allow you use Expression Builder to call a logical function to perform time series calculations instead of aliasing physical tables and modeling logically. The time series functions calculate Period Ago and Period to Date functions based on user supplied calendar tables, not on standard SQL date manipulation functions.

The following list describes the important grains in navigating a time query, using the following query example:

```
Select quarter, YearAgoSales:
```

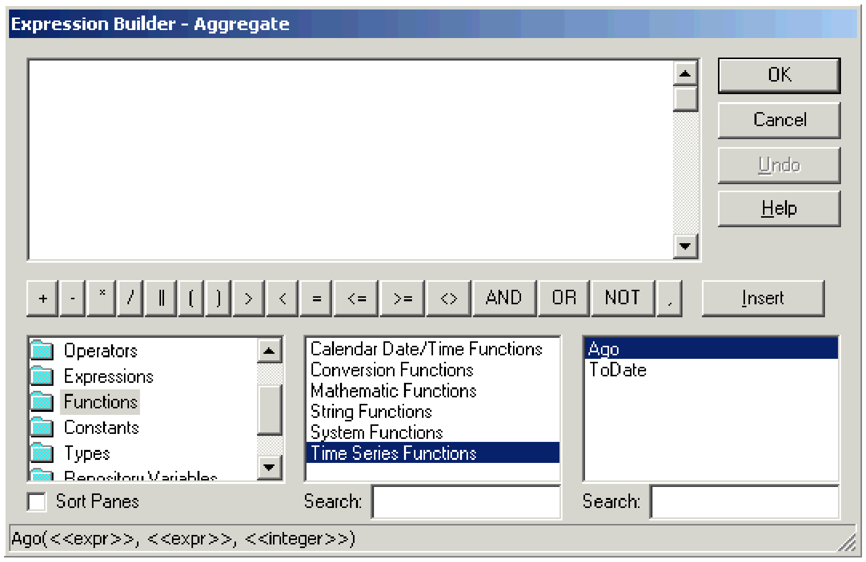
- **Query grain.** The grain of the request. In the query example, the query grain is Quarter.
- **Time Series grain.** The grain at which the aggregation is requested. In the query example, the Time Series grain is Year.

  NOTE: Time series query is valid only if the time series grain is at the query grain or higher.

- **Storage grain.** The query in the example can be computed from daily sales or from monthly sales, or from quarterly sales. The grain of the aggregate source is called aggregation grain.

  NOTE: The chronological key has to be defined at this level.

Figure 18 on page 198 shows the time series functions in the Expression Builder dialog box.



Ago(<<expr>>, <<expr>>, <<integer>>)

Figure 18. Example Time Series Functions in the Expression Builder

Ago

A time series aggregation function for relational data sources only. Calculates the aggregated value from the current time back to a specified time period. For example, Ago can produce sales for every month of the current quarter and the corresponding quarter-ago sales. Multiple Ago functions can be nested if all the Ago functions have the same level argument.

NOTE: You can nest exactly one ToDate and multiple Ago functions if they each have the same level argument.

Syntax:

```
AGO(<measure_expression>, <model_id>.<dimension_id>.<level_id>, <integer_literal>)
```

In that example, <measure_expression> is an expression that contains at least one measure, <model_id> is a model identifier, <dimension_id> is a dimension identifier, <level_id> is a level identifier, and <integer_literal> is an integer literal. The following is an example of this syntax:

```
AGO(model.sales.revenue + 5, model.time.month, 3)
```

ToDate

A time series aggregation function for relational data sources only. ToDate aggregates a measure attribute from the beginning of a specified time period to the currently displayed time. For example, this function can calculate Year to Date sales.

If unsupported metrics are requested, NULL values will be returned and a warning entry will be written to the NQQuery.log file when the logging level equals three or above. A ToDate function may not be nested within another ToDate function.

NOTE: You can nest exactly one ToDate and multiple Ago functions if they each have the same level argument.

Syntax:

```
TODATE(<measure_expression>, <model_id>.<dimension_id>.<level_id>)
```

About the IndexCol Conversion Function

The IndexCol function allows you to build a derived logical column. Selecting IndexCol automatically generates the following function template:

```
IndexCol( <<integer literal>>, <<expr1>> [, <<expr2>>, ?-] )
```

For more information, refer to "IndexCol" on page 400.

9 Setting Up Fragmentation Content in an Oracle BI Repository for Aggregate Navigation

This section contains the following topics:

- About Aggregate Navigation on page 201
- Specify Fragmentation Content on page 201

About Aggregate Navigation

Aggregate tables store precomputed results from measures that have been aggregated over a set of dimensional attributes. Each aggregate table column contains data at a given set of levels. For example, a monthly sales table might contain a precomputed sum of the revenue for each product in each store during each month. You configure this metadata in the Logical Table Source dialog box. For detailed steps showing how to create a logical source, refer to "Creating and Administering Logical Table Sources (Mappings)" on page 117.

This section includes a description of how you can use aggregate navigation and provides setup instructions.

Specify Fragmentation Content

When a logical table source does not contain the entire set of data at a given level, you need to specify the portion, or *fragment*, of the set that it does contain. You describe the content in terms of logical columns on the Content tab of the Logical Table Source dialog box, in the Fragmentation Content edit box.

The following examples illustrate techniques and rules for specifying the fragmentation content of sources.

Single Column, Value-Based Predicates

The IN predicates can be replaced with either an equality predicate or multiple equality predicates separated by the OR connective.

Fragment 1:

```
logicalColumn IN <valueList1>
```

Fragment *n*:

```
logicalColumn IN <valueListN>
```

Single Column, Range-Based Predicates

Fragment 1:

```
logicalColumn >= valueof(START_VALUE) AND logicalColumn < valueof(MID_VALUE1)
```

Fragment 2:

```
logicalColumn >= valueof(MID_VALUE1) AND logicalColumn < valueof(MID_VALUE2)
```

Fragment *n*:

```
logicalColumn >= valueof(MID_VALUEN-1) AND logicalColumn < valueof(END_VALUE)
```

Pick your start point, midpoints, and endpoint carefully.

NOTE: Notice the use of >= and < predicates to make sure the fragment content descriptions do not overlap. For each fragment, the upper value needs to be expressed as <. You will get an error if you use <=. Likewise, you cannot use the BETWEEN predicate to describe fragment range content.

The valueof referenced here is the value of a repository variable. (For more information about variables, refer to Chapter 13, "Using Variables in the Oracle BI Repository.") If you use repository values in your expression, note that the following construct will not work for Fragment 2:

```
logicalColumn >= valueof(MID_VALUE1)+1 AND logicalColumn < valueof(MID_VALUE2)
```

Use another repository variable instead of valueof(MID_VALUE1)+1.

The same variables, for example, valueof(MID_VALUE1), do not have to appear in the content of both fragments. You could set another variable, and create statements of the following form:

Fragment 1:

```
logicalColumn >= valueof(START_VALUE) AND logicalColumn < valueof(MID_VALUE1)
```

Fragment 2:

```
logicalColumn >= valueof(MID_VALUE2) AND logicalColumn < valueof(MID_VALUE3)
```

Multicolumn Content Descriptions

An arbitrary number of predicates on different columns can be included in each content filter. Each column predicate can be value-based or range-based.

Fragment 1:

```
<logicalColumn1 predicate> AND <logicalColumn2 predicate > ... AND <logicalColumnM
predicate>
```

Fragment *n*:

```
<logicalColumn1 predicate> AND <logicalColumn2 predicate > ... AND <logicalColumnM
predicate>
```

Ideally, all fragments will have predicates on the same M columns. If there is no predicate constraint on a logical column, the Oracle BI Server assumes that the fragment contains data for all values in that logical column. For exceptions using the OR predicate, refer to "Parallel Content Descriptions" on page 203.

Parallel Content Descriptions

Unfortunately, the preceding techniques are still not sufficient to handle dates because of the multiple hierarchical relationships across logical columns, such as year > year month > date; month > year month > date. For example, consider fragments delineated by different points in time, such as year and month. Constraining sufficiently far back on year should be enough to drive the selection of just the historical fragment. The parallel OR technique supports this, as shown in the next example. This example assumes that the snapshot month was April 1, 12:00 a.m. in the year 1999. The relevant OR connectives and predicates are shown in bold text.

Fragment 1 (Historical):

```
EnterpriseModel.Period."Day" < VALUEOF("Snapshot Date") OR

EnterpriseModel.Period.MonthCode < VALUEOF("Snapshot Year Month") OR

EnterpriseModel.Period."Year" < VALUEOF("Snapshot Year") OR

EnterpriseModel.Period."Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel.Period."Month in Year" < VALUEOF("Snapshot Month") OR

EnterpriseModel.Period."Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel.Period."Monthname" IN ('Mar', 'Feb', 'Jan')
```

Fragment 2 (Current):

```
EnterpriseModel.Period."Day" >= VALUEOF("Snapshot Date") OR

EnterpriseModel.Period.MonthCode >= VALUEOF("Snapshot Year Month") OR

EnterpriseModel.Period."Year" > VALUEOF("Snapshot Year") OR

EnterpriseModel.Period."Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel.Period."Month in Year" >= VALUEOF("Snapshot Month") OR

EnterpriseModel.Period."Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel.Period."Monthname" IN ('Dec', 'Nov', 'Oct', 'Sep', 'Aug', 'Jul',
'Jun', 'May', 'Apr')
```

If the logical model does not go down to the date level of detail, then omit the predicate on EnterpriseModel.Period."Day" in the preceding example.

Note the use of the OR connective to support parallel content description tracks.

Examples and Discussion

In this section, the Track n labels in the examples are shown to make it easier to relate the examples to the discussion that follows. You would not include these labels in the actual fragmentation content statement.

Fragment 1 (Historical):

```
Track 1  EnterpriseModel.Period."Day" < VALUEOF("Snapshot Date") OR

Track 2  EnterpriseModel.Period.MonthCode < VALUEOF("Snapshot Year Month") OR

Track 3  EnterpriseModel.Period."Year" < VALUEOF("Snapshot Year") OR
```

```
*Track 4*  EnterpriseModel.Period."Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel.Period."Month in Year" < VALUEOF("Snapshot Month") OR
```

```
*Track 5*  EnterpriseModel.Period."Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel.Period."Monthname" IN ('Mar', 'Feb', 'Jan')
```

For example, consider the first track on EnterpriseModel.Period."Day". In the historical fragment, the < predicate tells the Oracle BI Server that any queries that constrain on Day before the Snapshot Date fall within the historical fragment. Conversely, the >= predicate in the current fragment on Day indicates that the current fragment does not contain data before the Snapshot Date.

The second track on MonthCode (for example, 199912) is similar to Day. It uses the < and >= predicates as there is a nonoverlapping delineation on month (because the snapshot date is April 1). The key rule to remember is that each additional parallel track needs to reference a different column set. Common columns may be used, but the overall column set needs to be unique. The Oracle BI Server uses the column set to select the most appropriate track.

The third track on Year (< in the historical fragment and > in the current fragment) tells the Oracle BI Server that optimal (single) fragment selections can be made on queries that just constrain on year. For example, a logical query on Year IN (1997, 1998) should only hit the historical fragment. Likewise, a query on Year = 2000 needs to hit only the current fragment. However, a query that hits the year 1999 cannot be answered by the content described in this track, and will therefore hit both fragments, unless additional information can be found in subsequent tracks.

The fourth track describes the fragment set with respect to Year and Month in Year (month integer). Notice the use of the multicolumn content description technique, described previously. Notice the use of < and >= predicates, as there is no ambiguity or overlap with respect to these two columns.

The fifth track describes fragment content in terms of Year and Monthname. It uses the value-based IN predicate technique.

As an embellishment, suppose the snapshot date fell on a specific day within a month; therefore multicolumn content descriptions on just year and month would overlap on the specific snapshot month. To specify this ambiguity, <= and >= predicates are used.

Fragment 1 (Historical):

```
EnterpriseModel.Period."Day" < VALUEOF("Snapshot Date") OR

EnterpriseModel.Period.MonthCode <= VALUEOF("Snapshot Year Month") OR

EnterpriseModel.Period."Year" < VALUEOF("Snapshot Year") OR

EnterpriseModel.Period."Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel.Period."Month in Year" <= VALUEOF("Snapshot Month") OR

EnterpriseModel.Period."Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel.Period."Monthname" IN ('Apr', 'Mar', 'Feb', 'Jan')
```

Fragment 2 (Current):

```
EnterpriseModel.Period."Day" >= VALUEOF("Snapshot Date") OR

EnterpriseModel.Period.MonthCode >= VALUEOF("Snapshot Year Month") OR

EnterpriseModel.Period."Year" > VALUEOF("Snapshot Year") OR
EnterpriseModel.Period."Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel.Period."Month in Year" >= VALUEOF("Snapshot Month") OR

EnterpriseModel.Period."Year" = VALUEOF("Snapshot Year") AND
EnterpriseModel.Period."Monthname" IN ('Dec', 'Nov', 'Oct', 'Sep', 'Aug', 'Jul',
'Jun', 'May', 'Apr')
```

Unbalanced Parallel Content Descriptions

In an order entry application, time-based fragmentation between historical and current fragments is typically insufficient. For example, records may still be volatile, even though they are historical records entered into the database before the snapshot date.

Assume, in the following example, that open orders may be directly updated by the application until the order is shipped or canceled. After the order has shipped, however, the only change that can be made to the order is to type a separate compensating return order transaction.

There are two parallel tracks in the following content descriptions. The first track uses the multicolumn, parallel track techniques described in the preceding section. Note the parentheses nesting the parallel calendar descriptions within the Shipped-or-Canceled order status multicolumn content description.

The second parallel track is present only in the Current fragment and specifies that all Open records are in the Current fragment only.

Fragment 1 (Historical):

```
Marketing."Order Status"."Order Status" IN ('Shipped', 'Canceled') AND

Marketing.Calendar."Calendar Date" <= VALUEOF("Snapshot Date") OR

Marketing.Calendar."Year" <= VALUEOF("Snapshot Year") OR

Marketing.Calendar."Year Month" <= VALUEOF("Snapshot Year Month")
```

Fragment 2 (Current):

```
Marketing."Order Status"."Order Status" IN ('Shipped', 'Canceled') AND

Marketing.Calendar."Calendar Date" > VALUEOF("Snapshot Date") OR

Marketing.Calendar."Year" >= VALUEOF("Snapshot Year") OR

Marketing.Calendar."Year Month" >= VALUEOF("Snapshot Year Month")

OR Marketing."Order Status"."Order Status" = 'Open'
```

The overlapping Year and Month descriptions in the two fragments do not cause a problem, as overlap is permissible when there are parallel tracks. The rule is that at least one of the tracks has to be nonoverlapping. The other tracks can have overlap.

Aggregate Table Fragments

Information at a given level of aggregation is sometimes stored in multiple physical tables. When individual sources at a given level contain information for a portion or fragment of the domain, the Oracle BI Server needs to know the content of the sources in order to pick the appropriate source for the query.

For example, suppose you have a database that tracks the sales of soft drinks in all stores. The detail level of data is at the store level. Aggregate information, as described in Figure 19 on page 206, is stored at the city level for the sales of Coke and Pepsi, but there is no aggregate information for the sales of 7-Up or any other of the sodas.

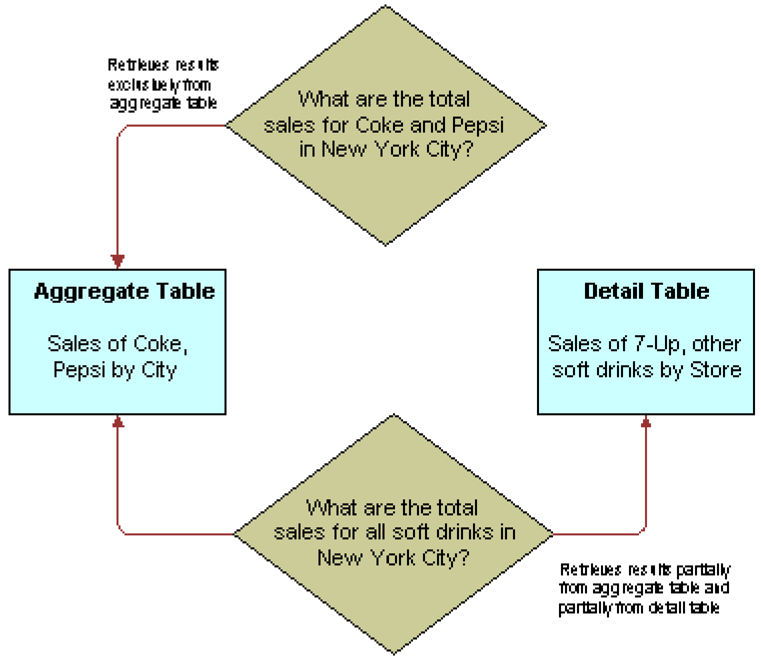

Retrieves results exclusively from aggregate table
What are the total sales for Coke and Pepsi in New York City?
Aggregate Table
Sales of Coke, Pepsi by City
Detail Table
Sales of 7-Up, other soft drinks by Store
What are the total sales for all soft drinks in New York City?
Retrieves results partially from aggregate table and partially from detail table

Figure 19. Aggregating Information

The goal of this type of configuration is to maximize the use of the aggregate table. If a query asks for sales figures for Coke and Pepsi, the data should be returned from the aggregate table. If a query asks for sales figures for all soft drinks, the aggregate table should be used for Coke and Pepsi and the detail data for the other brands.

The Oracle BI Server handles this type of partial aggregate navigation. To configure a repository to use aggregate fragments for queries whose domain spans multiple fragments, you need to define the entire domain for each level of aggregate data, even if you have to configure an aggregate fragment as being based on a less summarized physical source.

Specify the Aggregate Table Content

You configure the aggregate table navigation in the logical table source mappings. In the soft drink example, the aggregate table contains data for Coke and Pepsi sales at the city level. Its Aggregate content specification (in the Content tab of the Logical Table Source window) is similar to the following:

Group by logical level:

```
GeographyDim. CityLevel, ProductDim.ProductLevel
```

Its Fragmentation content specification (also in the Content tab of the Logical Table Source dialog) is similar to the following:

```
SoftDrinks.Products.Product IN ('Coke', 'Pepsi')
```

This content specification tells the Oracle BI Server that the source table has data at the city and product level for two of the products. Additionally, because this source is a fragment of the data at this level, you need to check the option This source should be combined with other sources at this level, in the Content tab of the Logical Table Source dialog box, to indicate that the source combines with other sources at the same level. For more information, refer to "Specify Fragmentation Content" on page 201.

Define a Physical Layer Table with a Select Statement to Complete the Domain

The data for the rest of the domain (the other types of sodas) is all stored at the store level. To define the entire domain at the aggregate level (city and product, in this example), you need to have a source that contains the rest of the domain at this level. Because the data at the store level is at a lower (that is, more detailed) level than at the city level, it is possible to calculate the city and product level detail from the store and product detail by adding up the product sales data of all of the stores in a city. This can be done in a query involving the store and product level table.

One way to do this is to define a table in the Physical layer with a Select statement that returns the store level calculations. To define the table, create a table in the Physical layer by selecting the physical schema folder that the Select statement will be querying and execute the New Table command. Choose Select from the Object Type drop-down list, and type the SQL statement in the pane to the right.

The SQL needs to define a virtual table that completes the domain at the level of the other aggregate tables. In this case, there is one existing aggregate table, and it contains data for Coke and Pepsi by city. Therefore, the SQL statement has to return all of the data at the city level, except for the Coke and Pepsi data.

Specify the SQL Virtual Table Content

Next, create a new logical table source for the Sales column that covers the remainder of the domain at the city and product level. This source contains the virtual table created in the previous section. Map the Dollars logical column to the USDollars physical column in this virtual table.

The Aggregate content specification (in the Content tab of the Logical Table Source dialog) for this source is:

Group by logical level:

```
    GeographyDim.CityLevel, ProductDim.ProductLevel
```

This tells the Oracle BI Server this source has data at the city and product level.

The Fragmentation content specification might be:

```
    SoftDrinks.Products.Product = '7-Up'
```

Additionally, because it combines with the aggregate table containing the Coke and Pepsi data at the city and product level to complete the domain, you need to check the option in the Content tab of the Logical Table Source dialog indicating that the source is combined with other sources at the same level.

Physical Joins for Virtual Table

Construct the correct physical joins for the virtual table. Notice that CityProductSales2 joins to the Cities and Products tables in Figure 20 on page 208.

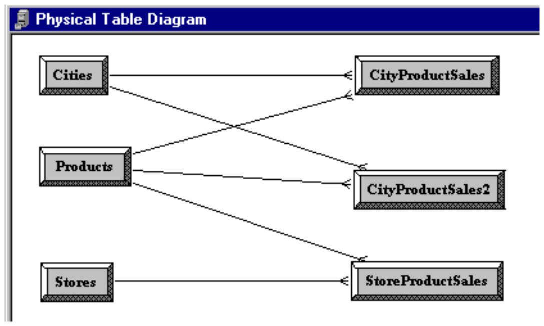


Figure 20. Example Physical Joins

In this example, the two sources comprise the whole domain for soda sales. A domain may have many sources. The sources have to all follow the rule that each level needs to contain sources that, when combined together, comprise the whole domain of values at that level. Setting up the entire domain for each level helps to make sure that queries asking for Coke, Pepsi, and 7-Up do not leave out 7-Up. It also helps to make sure that queries requesting information that has been precomputed and stored in aggregate tables can retrieve that information from the aggregate tables, even if the query requests other information that is not stored in the aggregate tables.

10 Administering the Oracle BI Server Query Environment

The Oracle BI Server is a server-based query environment and has many of the tools associated with managing server-based systems. This section describes how to use these tools to perform various administrative actions, including starting and stopping the server, checking and analyzing the log files, and other tasks related to managing a multiuser environment.

This chapter contains the following topics:

Starting the Oracle BI Server

The Oracle BI Server needs to be running before any queries can be processed. The following topics describe three ways to start the Oracle BI Server, how to change the User ID for the Oracle BI Server, and what to do if the Oracle BI Server fails to start:

Starting the Server from Windows Services

This procedure requires your user ID to be a member of the Windows Administrators group on the local machine on which the Oracle BI Server is installed.

To start the Oracle BI Server

1. On the machine in which the server is installed, choose Start > Settings > Control Panel > Services.

   You might need to open the Administrative Tools submenu.

2. Select the Oracle BI Server service and click Start.

   A message appears indicating that the service is starting. It might take a few minutes for the server to start while it loads the repository in the Repositories section of the NQSConfig.INI file.

   When startup is complete, the startup message goes away and the status of the service changes to Started. The following information is logged to the NQServer.log file, located in the Log subdirectory of the Oracle BI installation folder:

   - Startup time
   - Any business models that are loaded
   - Any errors that occurred

   If startup does not complete, check to make sure that there are no errors in the NQSConfig.INI file, such as the incorrect spelling of the repository filename. If you receive an informational message stating the server has not yet started, refresh the status periodically until the status changes to Started.

Configuring the Server for Automatic Startup in Windows

The following procedure explains how to configure the Oracle BI Server to start automatically when Windows NT or Windows 2000 starts.

To configure the Oracle BI Server for automatic startup

1. On the machine in which the server is installed, choose Start > Settings > Control Panel > Services.

   You might need to open the Administrative Tools submenu.

2. In the Services dialog box, double-click the Oracle BI Server service.
3. In the Oracle BI Server Properties dialog box, from the Startup Type drop-down list, select Automatic, and then click OK.

   The Oracle BI Server service will now start automatically when Windows starts.

Running the Server Startup Script in UNIX

Start the Oracle BI Server by running one of the following scripts:

- If you are using sh or bash:

```
run-sa.sh start
```

- If you are using csh:

  ```
  run-sa.csh start
  ```

- If you have set up your environment with sa.sh or sa.csh:

  ```
  nqscomgateway.exe &
  ```

- For the standard shell:

  ```
  nohup nqscomgateway.exe >/dev/null 2>&1 &
  ```

- For the C shell:

  ```
  nohup nqscomgateway.exe >&/dev/null &
  ```

Changing the User ID in Which the Oracle BI Server Runs

In Windows, the Oracle BI Server runs as a Windows service. It runs under the local system account by default. If the server needs to access databases on remote machines, it needs to run under a user ID that has the appropriate network privileges. Additionally, the user ID has to be a member of the Windows NT Administrators group on the local machine.

If you want to change the user ID in which the Oracle BI Server runs, you can do so from the Control Panel Services utility.

To change the user ID

1. On the machine in which the Oracle BI Server is installed, select Start > Settings > Control Panel.
2. In the Control Panel, double-click the Services icon.
3. Double-click the Oracle BI Server service and click Startup.
4. In the Oracle BI Server Properties dialog box, in the Log On As area, select This Account, and then click the button to the right of the text box.
5. In the Add User dialog box, select the user account in which you want the Oracle BI Server to run, click Add, and then click OK.
6. Type the password for the user in the Services dialog box, confirm the password, and then click OK.

   The server is now configured to run under the new user ID. The next time you start the service, it will attempt to use the new account to start the service.

If the Server Fails to Start

If the startup operation fails, look in the following log files for messages indicating why:

- In Windows NT and Windows 2000, in the Windows Event log, that you can access by selecting Start > Programs > Administrative Tools > Event Viewer.
- In Windows NT, Windows 2000, and UNIX, in the NQServer.log file. This file is located in the Log folder in the Oracle BI Server software installation folder (\OracleBI). You can use a text editor to view this file.
- In UNIX, run /usr/sbin/syslogd and look for any system and Oracle BI Server-related messages.

The log files contain messages indicating why the server startup failed. For example, if there were a syntax error in the NQSConfig.INI file, both the operating system's log and the NQServer.log file would contain messages about the syntax error. After examining the log messages, correct the problem and start the server again.

Shutting Down the Oracle BI Server

You can stop the Oracle BI Server in any of the following ways.

- Shutting Down the Server in Windows Services on page 212
- Shutting Down the Server from a Command Prompt in Windows on page 213
- Running the Server Shutdown Script in UNIX on page 213
- Shutting Down the Oracle BI Server Using the Administration Tool on page 214

When you shut down the server, any active client connections receive an error message and are immediately terminated, and any outstanding queries to underlying databases are cancelled.

Shutting Down the Server in Windows Services

The following procedure explains how to shut down the Oracle BI Server from the Windows Control Panel Services applet.

To shut down the server from the Services applet

1. On the machine in which the Oracle BI Server is installed, select Start > Settings > Control Panel.
2. Open the Services applet by double-clicking the Services icon in the Control Panel.
3. Select the Oracle BI Server service and click Stop.

   A message appears indicating that the service is shutting down.

When shutdown is complete, the status in the Services Control Panel becomes blank and a log message is written to the NQServer.log file, located in the Log folder in the software installation folder (\OracleBI\Log).

Shutting Down the Server from a Command Prompt in Windows

Use this procedure in Windows to shut down the Oracle BI Server from a Command prompt.

To shut down the Oracle BI Server from a Windows command prompt

- Type the following command in the machine in which the Oracle BI Server is running:

  `nqsshutdown {-d <data_source_name> -u <user ID> -p <password>}`

  where:

data_source_name	The name of a nonclustered ODBC data source used to connect to the server to perform the shut down operation. The data source needs to connect to the server as an Oracle BI user who is a member of the Oracle BI Administrators group. Only users defined as Oracle BI Administrators can shut down the server.

  NOTE: The nqsshutdown command is not valid for clustered DSNs. When passed a standard (nonclustered) DSN, it will shut down the targeted Oracle BI Server even if the server is participating in a cluster.

user ID	The user to connect to the Oracle BI Server to perform the shut down operation.
password	The password for the user ID connecting to the Oracle BI Server to perform the shut down operation.

  When shutdown is complete, a message indicating this is displayed in the Command window.

Running the Server Shutdown Script in UNIX

Stop the Oracle BI Server by running one of the following scripts:

- If you are using sh or bash:

  ```
  run-sa.sh stop
  ```

- If you are using csh:

  ```
  run-sa.csh stop
  ```

- If you have set up your environment with sa.sh or sa.csh:

  ```
  nqsshutdown.exe -uAdministrator
  ```

Shutting Down the Oracle BI Server Using the Administration Tool

The following procedure explains how to shut down the server using the Administration Tool. You need to open the repository in online mode using a nonclustered DSN.

NOTE: To shut down the server, the DSN has to log in as a user that has Administrator authority.

To shut down the server using the Administration Tool

1. Start the Administration Tool by selecting Start > Programs > Oracle BI > Oracle BI Administration Tool.
2. Open a repository that is loaded on the server in online mode.
3. Select File > Shut Down Server.
4. When a dialog box appears asking you to confirm the shutdown, click Yes.

   This shuts the server down and ends the Administration Tool online session. When connected using a clustered DSN, use the Cluster Manager to take individual Oracle BI Server instances offline. For more information, refer to “Using the Cluster Manager” on page 307.

Getting Users to Connect to the Server

Users need to set up a data source to connect to a business model within an Oracle BI repository. For information about setting up DSN connections, refer to “Configuring Oracle BI ODBC Data Source Names (DSNs)” on page 259.

You can also run the Administration Tool from a remote machine in online mode or offline mode.

Administering the Query Log

The Oracle BI Server provides a facility for logging query activity at the individual user level. The query log file is named the NQQuery.log file. This file is in the Log subdirectory in the Oracle BI installation folder. Logging should be used for quality assurance testing, debugging, and troubleshooting by Oracle Technical Support. In production mode, query logging is typically disabled.

Oracle BI Server query logging is tracked at a user level. It will be a resource intensive process if you track the entire user community.

NOTE: For production systems, it is recommended that query logging be enabled only for a very targeted user community. Most users should have a log level of 0 (zero). In production systems, you can use the Usage Tracking as the production level logging facility. For more information, see “Administering Usage Tracking” on page 219.

It is recommended that you only test users when the user name clearly indicates it is a test user and have verified that query logging enabled. If logging is enabled for such users, it is recommended that they be given names such as sales_admin_with_logging, sales_dev_with_logging, or sales_test_with_logging, so that you can readily identify them. Even production Oracle BI Administrator logins should not have query logging enabled because it could strain the available resources.

You should also disable query logging for the following:

- The SQL statement in the initialization string. The Initialization string text box is in the Initialization Block dialog box, in the General tab.

  NOTE: The LOGGING column references stored values for the log level.

- The logging level should be set to 0 (zero) for each production user. The Logging level field is in the User dialog box, in the User tab.

Configuring the Logging System

This section describes the logging system and includes information about setting the size of the query log, choosing a logging level, and enabling query logging for a user.

Because query logging can produce very large log files, the logging system is turned off by default. It is sometimes useful to enable logging to test that your repository is configured properly, to monitor activity on your system, to help solve performance problems, or to assist Technical Support. You need to enable logging on the system for each user whose queries you want logged.

Controlling the Size of the Log File

The parameter USER_LOG_FILE_SIZE in the User Log section of the NQSConfig.INI file determines the size of the NQQuery.log file. When the log file grows to one-half the size specified by the USER_LOG_FILE_SIZE parameter, the file is renamed to NQQuery.log.old, and a new log file is created automatically. (This helps to make sure that the disk space allocated for the log file does not exceed the size specified in the configuration file.) Only one copy of the old file is kept.

You can set the file size as high as you like, limited only by the amount of space available on the device. If you change the value of the USER_LOG_FILE_SIZE parameter, you need to restart the Oracle BI Server for the change to take effect. For the syntax of the USER_LOG_FILE_SIZE parameter, refer to *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

Setting a Logging Level

You can enable logging level for individual users; you cannot configure a logging level for a group.

NOTE: A session variable overrides a user's logging level. For example, if the Oracle BI Administrator has a logging level defined as 4 and a session variable logging level is defined as default 0 (zero) in the repository, the Oracle BI Administrator's logging level will be 0.

Set the logging level based on the amount of logging you want to do. In normal operations, logging is generally disabled (the logging level is set to 0). If you decide to enable logging, choose a logging level of 1 or 2. These two levels are designed for use by Oracle BI Administrators.

You might want to diagnose performance or data issues by setting a temporary log level for a query. You can enable query logging for a specific query by preceding your Select statement with the following:

```
Set Variable LOGLEVEL=n;
```

See the following example:

```
Set Variable LOGLEVEL=5; select year, product, sum(revenue) from time, products,
facts
```

For this query, the logging level of five is used regardless of the value of the underlying LOGLEVEL variable.

NOTE: Logging levels greater than 2 should be used only with the assistance of Technical Support.

The logging levels are described in Table 23 on page 216.

Table 23. Logging Levels

Logging Level	Information That Is Logged
Level 0	No logging.
Level 1	Logs the SQL statement issued from the client application. Logs elapsed times for query compilation, query execution, query cache processing, and back-end database processing. Logs the query status (success, failure, termination, or timeout). Logs the user ID, session ID, and request ID for each query.
Level 2	Logs everything logged in Level 1. Additionally, for each query, logs the repository name, business model name, presentation catalog (called Subject Area in Answers) name, SQL for the queries issued against physical databases, queries issued against the cache, number of rows returned from each query against a physical database and from queries issued against the cache, and the number of rows returned to the client application.
Level 3	Logs everything logged in Level 2. Additionally, adds a log entry for the logical query plan, when a query that was supposed to seed the cache was not inserted into the cache, when existing cache entries are purged to make room for the current query, and when the attempt to update the exact match hit detector fails. Do not select this level without the assistance of Technical Support.
Level 4	Logs everything logged in Level 3. Additionally, logs the query execution plan. Do not select this level without the assistance of Technical Support.

Table 23. Logging Levels

Logging Level	Information That Is Logged
Level 5	Logs everything logged in Level 4. Additionally, logs intermediate row counts at various points in the execution plan. Do not select this level without the assistance of Technical Support.
Level 6 and 7	Reserved for future use.

To set a user's logging level

1. In the Administration Tool, select Manage > Security.

   The Security Manager dialog box appears.

2. Double-click the user's user ID.

   The User dialog box appears.

3. Set the logging level by clicking the Up or Down arrows next to the Logging Level field.

To disable a user's logging level

- Set the logging level to 0.

Using the Log Viewer

Use the Oracle BI log viewer utility nQLogViewer (or a text editor) to view the query log. Each entry in the query log is tagged with the user ID of the user who issued the query, the session ID of the session in which the query was initiated, and the request ID of the individual query.

To run the nQlogViewer utility, open a Command window and type `nQlogViewer` with any combination of its arguments. The syntax is as follows:

```
nqlogviewer [-u<user_ID>] [-f<log_input_filename>]
            [-o<output_result_filename>]
            [-s<session_ID>] [-r<request_ID>]
```

where:

user_ID	The name of a user in the Oracle BI repository. This limits the scope to entries for a particular user. If not specified, all users for whom query logging is enabled are shown.
log_input_filename	The name of an existing log file. This parameter is required.
output_result_filename	The name of a file in which to store the output of the log viewer. If the file exists, results are appended to the file. If the file does not exist, a new file is created. If not specified, output is sent to the monitor screen.

session_ID	The session ID of the user session. The Oracle BI Server assigns each session a unique ID when the session is initiated. This limits the scope of the log entries to the specified session ID. If not specified, all session IDs are shown.
request_ID	The request ID of an individual query. The Oracle BI Server assigns each query a unique ID when the query is initiated. This limits the scope of the log entries to the specified request ID. If not specified, all request IDs are shown.
	NOTE: The request id will be unique among the active requests but not necessarily unique during the session. Request ids are generated in a circular manner, and if a request is closed or if the session is long enough, a request id will be reused.

You can also locate user IDs, session IDs and request IDs through the Session Manager. For more information, refer to "Using the Session Manager" on page 225.

NOTE: Oracle BI Presentation Services Administrators can view the query log using the Manage Sessions option in Presentation Services Administration.

Interpreting the Log Records

After you have logged some query information and started the log viewer, you can analyze the log. The log is divided into several sections, some of which are described in the next section. Log entries for levels 1 and 2 are generally self-explanatory. The log entries can provide insights to help DBAs in charge of the underlying databases tune them for optimum query performance. The query log can also help you check the accuracy of applications that use the Oracle BI Server.

SQL Request

This section lists the SQL issued from the client application. This can be used to rerun the query from the same application, or from a different application.

General Query Information

This section lists the repository, the business model, and the presentation catalog from which the query was run. You can use this information to provide statistics on query usage that could be used to set priorities for future application development and system management.

Database Query

This section of the log begins with an entry that reads Sending query to the database named <data_source_name>, where data_source_name is the name of the data source to which the Oracle BI Server is connecting. Multiple database queries can be sent to one or more data sources. Each query will have an entry in the log.

The database query section has several uses. It records the SQL sent to the underlying databases; you can then use the logged SQL to run queries directly against the database for performance tuning, results verification, or other testing purposes. It allows you to examine the tables that are being queried to verify that aggregate navigation is working as you expect. If you understand the structure of the underlying database, it might also provide some insights into potential performance improvements, such as useful aggregate tables or indexes to build.

Query Status

The query success entry in the log indicates if the query completed successfully or if it failed. You can search through the log for failed queries to determine why they failed. For example, all the queries during a particular time period might have failed due to a database downtime.

Administering Usage Tracking

The Oracle BI Server supports the accumulation of usage tracking statistics that can be used in a variety of ways such as database optimization, aggregation strategies, or billing users or departments based on the resources they consume. The Oracle BI Server tracks usage at the detailed query level.

When you enable usage tracking, statistics for every query are inserted into a database table or are written to a usage tracking log file. If you use direct insertion, the Oracle BI Server directly inserts the usage tracking data into a relational database table. It is recommended that you use direct insertion to write statistics to a database table.

When the Oracle BI Server starts up, Oracle BI Server validates the column names in the metadata against the list of valid columns in the usage tracking table. The lengths of varchars will not be validated. The following events occur:

- **Column names.** If metadata contains column names that are not in the table, an error will be written to the Oracle BI Server log and usage tracking will be disabled. If there are columns in tables that are not in the metadata, the columns that are in the metadata will be written to the usage tracking table.
- **Varchar length.** If the length in metadata and the set length in the table do not match, an error will be written to the NQServer.log file and usage tracking will be disabled.

This section contains the following topics:

- Setting Up Direct Insertion to Collect Information for Usage Tracking on page 219
- Setting Up a Log File to Collect Information for Usage Tracking on page 221

Setting Up Direct Insertion to Collect Information for Usage Tracking

This is the recommended method for setting up usage tracking.

To set up direct insertion for usage tracking, use the guidelines in this section. For more information, refer to *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*. To set up and administer direct insertion, use the following topics:

- Enabling Direct Insertion on page 220
- Database Table Configuration on page 220
- Connection Pool Configuration on page 220
- Buffer Size Configuration Parameter on page 221
- Buffer Time Limit Configuration Parameter on page 221
- Number of Insert Threads Configuration Parameter on page 221
- Max Inserts Per Transactions Configuration Parameter on page 221

Enabling Direct Insertion

In the Usage Tracking section of the NQSConfig.INI file, the DIRECT_INSERT parameter determines whether the query statistics are inserted directly into a database table or are written to a file for subsequent loading. The DIRECT_INSERT and ENABLE parameters must be set to YES to enable direct insertion.

NOTE: It is strongly recommended that you enable direct insertion.

Database Table Configuration

Inserting query statistic information into a table requires the configuration of the name of the table and the connection pool used to access the table.

The fully qualified physical table name consists of up to four components (database name, catalog name, schema name, and table name). Each component is surrounded by double quotes (") and separated by a period (.). The physical table name must be fully qualified. This fully qualified physical table name must match a table name in the physical layer of the loaded repository. The following is an example of a physical table name for the Usage Tracking table in the Oracle BI repository:

```
PHYSICAL_TABLE_NAME = "Oracle BI Usage"."Catalog"."dbo"."S_NQ_ACCT" ;
```

In this example, Oracle BI Usage represents the database component, Catalog represents the catalog component, dbo represents the schema component, and S_NQ_ACCT represents the table name.

Connection Pool Configuration

The fully-specified connection pool name has two parts, database name and connection pool name. Each part is surrounded by double quotes (") and separated by a period (.). The fully qualified connection pool name should match a connection pool name in the physical layer of the loaded repository. For an example, refer to the following connection pool name in the Oracle BI repository:

```
CONNECTION_POOL = "Oracle BI Usage"."Connection Pool" ;
```

In this example, Oracle BI Usage represents the database component and Connection Pool represents the connection pool name proper.

For Usage Tracking inserts to succeed, the connection pool must be configured with a user ID that has write access to the back-end database.

NOTE: It is recommended that the connectivity type supports international data.

Buffer Size Configuration Parameter

The BUFFER_SIZE configuration parameter indicates how much memory the Oracle BI Server should allocate for buffering the insert statements. Such a buffer allows the Oracle BI Server to submit multiple insert statements as part of a single transaction, improving Usage Tracking insert throughput. It also means that ordinary query requests do not have to wait on Usage Tracking inserts, improving average query response time. You may want to adjust this value based on available memory and memory utilization on the server machine.

Buffer Time Limit Configuration Parameter

The BUFFER_TIME_LIMIT_SECONDS configuration parameter indicates the maximum amount of time an insert statement will remain in the buffer before the Usage Tracking subsystem attempts to issue it. This time limit ensures that the Oracle BI Server will issue the insert statements in a timely manner even during periods of extended quiescence.

Number of Insert Threads Configuration Parameter

The NUM_INSERT_THREADS configuration parameter indicates the number of threads that will remove insert statements from the buffer and issue them to the Usage Tracking database. Assuming separate connection pools for readers and inserters, the number of insert threads should typically equal the Maximum Connections setting in the connection pool.

Max Inserts Per Transactions Configuration Parameter

The MAX_INSERTS_PER_TRANSACTION configuration parameter indicates the maximum number of insert statements the Usage Tracking subsystem attempts to issue as part of a single transaction. The larger this number, the greater potential throughput for Usage Tracking inserts. However a larger number also increases the likelihood of transactions failing due to deadlocks. Note that a small value for BUFFER_TIME_LIMIT_SECONDS may limit the number of inserts per transaction.

Setting Up a Log File to Collect Information for Usage Tracking

This is an alternate method for setting up usage tracking. It is recommended that you use direct insertion to collect information for usage tracking. For more information, refer to "Setting Up Direct Insertion to Collect Information for Usage Tracking" on page 219.

This section contains the following topics:

- Selecting an Output Location on page 222
- File Naming Conventions on page 222

Selecting an Output Location

The parameter STORAGE_DIRECTORY in the Usage Tracking section of the NQSConfig.INI file determines the location of usage tracking log files. If usage tracking is enabled, but no storage folder is specified, the files are written in the Log folder in the software installation folder (\OracleBI).

Current files are periodically written to disk, and new files are created. The parameter CHECKPOINT_INTERVAL_MINUTES controls the frequency with which usage tracking data is flushed to disk, and the parameter FILE_ROLLOVER_INTERVAL_MINUTES controls the frequency with which the current usage tracking log file is closed and a new file created.

When usage tracking is enabled, every query is logged to a usage tracking log file. This may require a large amount of available storage. For example, assume an average of 300 bytes of data output for each query and 10 queries per second over an 8 hour day. This results in approximately 83 MB of usage tracking data written to storage per day. If this example is extended to a 24 x 7 operation, the result is approximately .25 GB of storage per day.

The Oracle BI Server has no limit on the size or quantity of usage tracking log files that can exist in the specified location. It is the responsibility of the user to make sure that sufficient space is available, and to remove or archive old usage tracking files.

NOTE: Insufficient storage space may cause you to lose usage tracking data. If the Oracle BI Server encounters an error accessing a usage tracking output file, it immediately discontinues the collection of usage tracking statistics and issues an error message to the NQServer.log and, in Windows, to the Windows Event log. Even if additional storage space is made available, the collection of usage tracking statistics will not resume until the server is restarted.

File Naming Conventions

The file naming scheme for the usage tracking log files is NQAcct.yyyymmdd.hhmmss.log, where yyyy is the year, mm is the month, dd is the day, hh is the hour, mm is the minute, and ss is the second of the timestamp when the file was created. For example, if the server creates the usage tracking log file at 07:15:00 AM on February 12, 2003, the filename would be NQAcct.20030212.071500.log. After the specified rollover interval, this file is flushed to disk and closed and a new log file, with the current date and timestamp, is created.

Output File Format

The usage tracking log files are text files, in semicolon-delimited (;) format.
(A semicolon is used as the column delimiter because the logical SQL text contains commas.) A line feed delimits the end of each row of data.

The schema is described in Table 24 on page 223. For more information about the contents of each column, refer to "Description of the Usage Tracking Data" on page 406.

Table 24. Usage Tracking Output File Format

Column Number	Column Name	Data Type	Max Data Size	Nullable
1	User name	Varchar	128	No
2	Repository name	Varchar	128	No
3	Subject area name	Varchar	128	No
4	Node ID	Varchar	15	No
5	Start timestamp	Char (Timestamp)	19	No
6	Start date	Char (yyyy-mm-dd)	10	No
7	Start hourMin	Char (hh:mm)	5	No
8	End timestamp	Char (Timestamp)	19	No
9	End date	Char (yyyy-mm-dd)	10	No
10	End hourMin	Char (hh:mm)	5	No
11	Query Text	Varchar	1024	No
12	Success indicator	Integer (refer to following Note)	4	No
13	Row count	Integer (refer to following Note)	4	Yes
14	Total time (secs)	Integer (refer to following Note)	4	Yes
15	Compilation time (secs)	Integer (refer to following Note)	4	Yes
16	Number of database queries	Integer (refer to following Note)	4	Yes
17	Cumulative db time (secs)	Integer (refer to following Note)	4	Yes
18	Cumulative db rows	Integer (refer to following Note)	4	Yes
19	Cache indicator	Char	1	No
20	Query source	Varchar	30	No
21	Presentation Catalog path	Varchar	250	No
22	Interactive Dashboard name	Varchar	150	Yes

NOTE: All data in the output file is in character format. The data in columns 12 through 18 are output as text representations of integer numbers. Therefore, they behave more like Varchar(10) columns than integers. For example, if the row count is one million rows, then 1000000 appears in the output file in column 13 (Row count). This constitutes seven bytes of data, even though the data represents a 4-byte internal integer value.

- In column 12, a Success indicator value of 0 signifies a successful query. All nonzero values indicate failure. The following failure indicators are currently defined:
  - 1 indicates timeout
  - 2 indicates row limit violation
  - 3 indicates unknown error

The subsequent columns are valid only if the Success indicator signifies a successful query (value is 0):

- The Start timestamp and End timestamp columns indicate the wall clock time when the logical query started and finished. Each value is 19 bytes of character data representing a SQL-92 timestamp. The format is *yyyy-mm-dd-hh:mm:ss*. The related columns, Start date and End date, contain just the date component from the respective timestamps (in the *yyyy-mm-dd* format). Finally, the related columns, Start hourMin and End hourMin, contain just the hour and minute components from the respective timestamps (in a char *hh:mm* format).

While there is no guaranteed unique key for the usage tracking data, a combination of User name, Node ID, Start timestamp and Query text will usually be sufficient.

For information about sample scripts to help you extract data from usage tracking log files and load it to appropriately formatted relational database tables, refer to "Oracle BI Server Usage Tracking Data Descriptions and Using the Log File Method" on page 405.

Performance Considerations

When usage tracking is enabled, the Oracle BI Server collects usage tracking data for every query. This data, however, is only written to disk at user-specified intervals, known as *checkpoints*. The default setting is to checkpoint every 5 minutes.

While this value can be modified in theNQSConfig.INI file (refer to *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*), reducing the interval adds overhead and, if set low enough, could potentially impact server performance. Setting the value higher increases the amount of usage tracking data that could be lost in the unlikely event of an abnormal shutdown of the Oracle BI Server.

The Oracle BI Server periodically initiates usage tracking log file rollovers. A *rollover* consists of closing the current usage tracking log file and opening a newly created one for writing subsequent data. The frequency at which rollovers occur is called a *rollover interval*. The default rollover interval is 240 minutes (every 4 hours).

Usage tracking log files that are closed are available for analysis. Setting a lower rollover interval will make usage tracking log files available for analysis sooner, but at the cost of additional overhead.

If the checkpoint interval equals or exceeds the rollover interval, only the rollover occurs explicitly; the checkpoint only occurs implicitly when the old usage tracking log file is closed.

Server Session Management

The Session Manager is used in online mode to monitor activity. The Session Manager shows all users logged into the session, all current query requests for each user, and variables and their values for a selected session. Additionally, the Oracle BI Administrator can disconnect any users and kill any query requests with the Session Manager.

How often the Session Manager data refreshes depends on the amount of activity on the system. To refresh the display at any time, click Refresh.

Using the Session Manager

The Session Manager contains an upper and a lower pane:

- The top window, the Session window, shows users currently logged into the Oracle BI Server. To control the update speed, from the Update Speed drop-down list, choose Normal, High, or Low. Select Pause to keep the display from being refreshed.
- The bottom window contains two tabs.
  - The Request tab shows active query requests for the user selected in the Session window.
  - The Variables tab shows variables and their values for a selected session. You can click the column headers to sort the data.

    NOTE: Only 7.7 Oracle BI Server return information about variables. If the Administration Tool connects to an older-version server online, only the Requests tab will be visible in the Session Manager dialog box.

Table 25 on page 226 and Table 26 on page 226 describe the columns in the Session Manager windows.

Table 25. Fields in the Session Window

Column Name	Description
Client Type	The type of client connected to the server.
Last Active Time	The timestamp of the last activity on the session.
Logon Time	The timestamp that shows when the session initially connected to the Oracle BI Server.
Repository	The logical name of the repository to which the session is connected.
Session ID	The unique internal identifier that the Oracle BI Server assigns each session when the session is initiated.
User	The name of the user connected.

Table 26. Some Fields in the Request Tab

Column Name	Description
Last Active Time	The timestamp of the last activity on the query.
Request ID	The unique internal identifier that the Oracle BI Server assigns each query when the query is initiated.
Session ID	The unique internal identifier that the Oracle BI Server assigns each session when the session is initiated.
Start Time	The time of the individual query request.

To view the variables for a session

1. In the Administration Tool, open a repository in online mode and choose Manage > Sessions.
2. Select a session and click the Variables tab.

   For more information about variables, refer to "Using Variables in the Oracle BI Repository" on page 283.
3. To refresh the view, click Refresh.
4. To close Session Manager, click close.

To disconnect a user from a session

1. In the Administration Tool, open a repository in online mode and choose Manage > Sessions.
2. Select the user in the Session Manager top window.

3 Click Disconnect.

The user session receives a message indicating that the session was terminated by the Oracle BI Administrator. Any currently running queries are immediately terminated, and any outstanding queries to underlying databases are canceled.

4 To close Session Manager, click close.

To kill an active query

1 In the Administration Tool, open a repository in online mode and choose Manage > Sessions.

2 Select the user session that initiated the query in the top window of the Session Manager.

After the user is highlighted, any active query requests from that user are displayed in the bottom window.

3 Select the request you want to kill.

4 Click Kill Request to terminate the highlighted request.

The user receives a message indicating that the query was terminated by the Oracle BI Administrator. The query is immediately terminated, and any outstanding queries to underlying databases are canceled.

Repeat this process to kill any other requests.

5 To close Session Manager, click close.

Server Configuration and Tuning

Performance is an extremely important consideration in every decision support system, but it is particularly important in systems that allow queries over the Web. This section describes some important considerations for improving query performance with the Oracle BI Server. This is an overview topic. For details, see *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*

NQSConfig.INI File Parameters

The NQSConfig.INI file contains configuration and tuning parameters for the Oracle BI Server. There are parameters to configure disk space for temporary storage, set sort memory buffer sizes, set cache memory buffers, set virtual table page sizes, and a number of other configuration settings that allow you to take full advantage of your hardware’s capabilities.

For more information about the NQSConfig.INI file parameters, refer to *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

Aggregate Tables

You should use aggregate tables to improve query performance. Aggregate tables contain precalculated summarizations of data. It is much faster to retrieve an answer from an aggregate table than to recompute the answer from thousands of rows of detail. The Oracle BI Server uses aggregate tables automatically, if they have been properly specified in the repository.

For more information and examples of setting up aggregate navigation in a repository, refer to "Setting Up Fragmentation Content in an Oracle BI Repository for Aggregate Navigation" on page 201.

Query Caching

Enabling query caching causes the Oracle BI Server to store query results for reuse by subsequent queries. Caching can dramatically improve the apparent performance of the system for users. For more information about query caching concepts and setup, refer to Chapter 11, "Query Caching in the Oracle BI Server."

Tune and Index Underlying Databases

The Oracle BI Server sends queries to databases. For the queries to return in a timely manner, the underlying databases need to be configured, tuned, and indexed correctly. You might need to work with the DBAs of the underlying databases to help identify any problem areas where database tuning is in order.

Different database products will have different tuning considerations. If there are queries that return slowly from the underlying databases, you can capture the SQL of the queries in the query log, then provide them to the DBA for analysis. For more information about configuring query logging on your system, refer to "Administering the Query Log" on page 214.

11 Query Caching in the Oracle BI Server

Decision support queries sometimes require large amounts of database processing. The Oracle BI Server can save the results of a query in cache files and then reuse those results later when a similar query is requested. Using cache, the cost of database processing only needs to be paid once for a query, not every time the query is run.

This section explains query caching and how it is implemented in the Oracle BI Server.

NOTE: For information about how to use Delivers to seed the Oracle BI Server Cache, refer to *Oracle Business Intelligence Presentation Services Administration Guide*.

This chapter contains the following topics:

About the Oracle BI Server Query Cache

Oracle BI Administrators can configure the Oracle BI Server to maintain a local, disk-based cache of query result sets (query cache). The query cache allows the Oracle BI Server to satisfy many subsequent query requests without having to access back-end databases (such as Oracle or DB2). This reduction in communication costs can dramatically decrease query response time.

As updates occur on the back-end databases, the query cache entries can become stale. Therefore, Oracle BI Administrators need to periodically remove entries from the query cache using one of the following methods:

- **Manually.** In the Administration Tool, in the Manage menu, select Cache to open the Cache Manager. Cache Manager provides the maximum flexibility in choosing which cache entries to purge and when to purge them, but it requires direct human involvement. For more information, refer to "Using the Cache Manager" on page 255.

- **Automatically.** In the Administration Tool, you can disable cache for the system, set caching attributes for a specific physical table, and use Oracle BI event tables to purge cache automatically. For additional information about managing cache, refer to "Monitoring and Managing the Cache" on page 234.
- **Programatically.** The Oracle BI Server provides ODBC-extension functions for purging cache entries programmatically. These functions give you the choice and the timing flexibility of Cache Manager with the automation of event tables. You can write your own scripts to call these functions at times that fit your needs. For more information, refer to "Purging and Maintaining Cache Using ODBC Procedures" on page 235.

The parameters that control query caching are located in the NQSConfig.INI file described in *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

NOTE: For information about how to use Delivers to seed the Oracle BI Server Cache, refer to *Oracle Business Intelligence Presentation Services Administration Guide*.

Advantages of Caching

The fastest way to process a query is to skip the bulk of the processing and use a precomputed answer.

Aggregate tables are examples of precomputed answers. Aggregate tables contain precalculated results for a particular aggregation level. For example, an aggregate table might store sales results for each product by month, when the granularity of detail for the database is at the day level. To create this aggregate table, a process (often a query) computes the results and then stores them in a table in the database.

With query caching, the Oracle BI Server stores the precomputed results of queries in a local cache. If another query can use those results, all database processing for that query is eliminated. This can result in dramatic improvements in the average query response time.

In addition to improving performance, being able to answer a query from a local cache conserves network resources and processing time on the database server. Network resources are conserved because the intermediate results do not have to come over the network to the Oracle BI Server. Not running the query on the database frees the database server to do other work. If the database uses a charge back system, it could save money in the budget as well.

Another benefit of using the cache to answer a query is savings in processing time on the Oracle BI Server, especially if the query results are retrieved from multiple databases. Depending on the query, there might be considerable join and sort processing in the server. If the query is already calculated, this processing is avoided, freeing server resources for other tasks.

To summarize, query caching has the following advantages:

- Dramatic improvement of query performance.
- Less network traffic.
- Reduction in database processing and charge back.
- Reduction in Oracle BI Server processing overhead.

Initializing Cache Entries for User Ids

To initialize cache entries for user Ids, the Connection Pool needs to be set up for shared login with session variables VALUEOF(NQ_SESSION.PASSWORD),VALUEOF(NQ_SESSION.USER) in the login properties. If the shared login is disabled and a user specific database login is used, cache will be shared.

For more information about security, refer to Chapter 15, “Security in Oracle BI.”

Costs of Caching

Query caching has many obvious benefits, but also certain costs:

- Disk space for the cache
- Administrative costs of managing the cache
- Potential for cached results being stale
- Minor CPU and disk I/O on server machine

With proper cache management, the benefits will far outweigh the costs.

Disk Space

The query cache requires dedicated disk space. How much space depends on the query volume, the size of the query result sets, and how much disk space you choose to allocate to the cache. For performance purposes, a disk should be used exclusively for caching, and it should be a high performance, high reliability type of disk system.

Administrative Tasks

There are a few administrative tasks associated with caching. You need to set the cache persistence time for each physical table appropriately, knowing how often data in that table is updated. When the frequency of the update varies, you need to keep track of when changes occur and purge the cache manually when necessary. You can also create a cache event polling table and modify applications to update the polling table when changes to the databases occur, making the system event-driven.

The Oracle BI Server also provides ODBC-extension functions for purging cache entries programmatically. You can write your own scripts to call these functions at the appropriate times.

Keeping the Cache Up To Date

If the cache entries are not purged when the data in the underlying databases changes, queries can potentially return results that are out of date. You need to evaluate whether this is acceptable. It might be acceptable to allow the cache to contain some stale data. You need to decide what level of stale data is acceptable and then set up (and follow) a set of rules to reflect those levels.

For example, suppose your application analyzes corporate data from a large conglomerate, and you are performing yearly summaries of the different divisions in the company. New data is not going to materially affect your queries because the new data will only affect next year’s summaries. In this case, the tradeoffs for deciding whether to purge the cache might favor leaving the entries in the cache.

Suppose, however, that your databases are updated three times a day and you are performing queries on the current day's activities. In this case, you will need to purge the cache much more often, or perhaps consider not using it at all.

Another scenario is that you rebuild your data mart from scratch at periodic intervals (for example, once per week). In this example, you can purge the entire cache as part of the process of rebuilding the data mart, making sure that you never have stale data in the cache.

Whatever your situation, you need to evaluate what is acceptable as far as having noncurrent information returned to the users.

CPU Usage and Disk I/O

Although in most cases it is very minor, query caching does require a small amount of CPU time and adds to the disk I/O. In most cases, the CPU usage is insignificant, but the disk I/O might be noticeable, particularly if queries return large data sets.

Query Cache Architecture

The query cache consists of cache storage space, cache metadata, and cache detection in query compilation.

The process of accessing the cache metadata is very fast. If the metadata shows a cache hit, the bulk of the query processing is eliminated, and the results are immediately returned to the user. The process of adding the new results to the cache is independent of the results being returned to the user; the only effect on the running query is the resources consumed in the process of writing the cached results.

Configuring Query Caching

The query cache is disabled by default. To enable caching, you need to configure the cache storage and decide on a strategy for flushing outdated entries. This section includes information on the tasks necessary to configure the Oracle BI Server for query caching.

The parameters to control query caching are located in the NQSConfig.INI file, described in *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

Configuring the Cache Storage

The following items need to be set up for cache storage in the NQSConfig.INI file:

- Directories to store the cache files.
- A file to store the cache metadata.

Cache Data Storage Directories

The DATA_STORAGE_PATHS parameter in the CACHE section of the NQSConfig.INI file specifies one or more directories for query cache storage. These directories are used to store the cached query results and are accessed when a cache hit occurs. For more information, refer to "Cache Hits" on page 238.

The cache storage directories should reside on high performance storage devices, ideally devoted solely to cache storage. When the cache storage directories begin to fill up, the entries that are least recently used (LRU) are discarded to make space for new entries. The MAX_CACHE_ENTRIES parameter value specifies the maximum number of cache entries allowed in the query cache. The more space you can allocate to the cache, the less often queries will have to access the underlying databases to get the results.

For more information about this configuration parameter, refer to *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

Maximum Cache Entry Values

You can control the maximum number of rows for any cache entry and the maximum number of cache entries with the MAX_ROWS_PER_CACHE_ENTRY and MAX_CACHE_ENTRIES NQSConfig.INI file parameters, respectively. Limiting the number of rows is a useful way to avoid using up the cache space with runaway queries that return large numbers of rows. Limiting the total number of cache entries provides another parameter with which to manage your cache storage. If the number of rows a query returns is greater than the value specified in the MAX_ROWS_PER_CACHE_ENTRY parameter, the query is not cached.

For the syntax of these parameters, refer to *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

Aggregates

Typically, if a query gets a cache hit from a previously executed query, then the new query is not added to the cache. The POPULATE_AGGREGATE_ROLLUP_HITS parameter overrides this default when the cache hit occurs by rolling up an aggregate from a previously executed query.

Enabling Query Caching

After configuring the cache storage and deciding on one or more cache management strategies, as discussed in "Monitoring and Managing the Cache" on page 234, you can enable query caching.

To enable query caching

1. Set the ENABLE parameter in the CACHE section of the NQSConfig.INI file to YES.
2. Restart the Oracle BI Server.

Disabling Query Caching

This section explains how to disable query caching.

To disable query caching

1. Set the ENABLE parameter in the CACHE section of the NQSConfig.INI file to NO.
2. Restart the Oracle BI Server.

Monitoring and Managing the Cache

To manage the changes in the underlying databases and to monitor cache entries, you need to develop a *cache management strategy.* You need a process to invalidate cache entries when the data in the underlying tables that compose the cache entry have changed, as well as a process to monitor, identify, and remove any undesirable cache entries.

Choosing a Cache Management Strategy

The choice of a cache management strategy depends on the volatility of the data in the underlying databases and the predictability of the changes that cause this volatility. It also depends on the number and types of queries that comprise your cache, as well as the usage those queries receive. This section provides an overview of the various approaches to cache management.

Disable Caching for the System

You can disable caching for the whole system by setting the ENABLE parameter to NO in the NQSConfig.INI file and restarting the Oracle BI Server. Disabling caching stops all new cache entries and stops any new queries from using the existing cache. Disabling caching allows you to enable it at a later time without losing any entries already stored in the cache.

Temporarily disabling caching is a useful strategy in situations where you might suspect having stale cache entries but want to verify if they are actually stale before purging those entries or the entire cache. If you find that the data stored in the cache is still relevant, or after you have safely purged problem entries, you can safely enable the cache. If necessary, purge the entire cache or the cache associated with an entire business model before enabling the cache again.

Caching and Cache Persistence Timing for Specified Physical Tables

You can set a cachable attribute for each physical table, allowing you to specify if queries for a that table will be added to the cache to answer future queries. If you enable caching for a table, any query involving the table is added to the cache. All tables are cachable by default, but some tables may not be good candidates to include in the cache unless you use the Cache Persistence Time settings. For example, perhaps you have a table that stores stock ticker data, that is updated every minute. You could use the Cache Persistence Time settings to purge the entries for that table every 59 seconds.

You can also use the Cache persistence time field to specify how long the entries for this table should be kept in the query cache. This is useful for data sources that are updated frequently.

To set the caching attributes for a specific physical table

1. In the Physical layer, double-click the physical table.

2. In the Physical Table properties dialog box, in the General tab, make one of the following selections:
   - To enable caching, select the Make table cachable check box.
   - To prevent a table from ever being cached, clear the Make table cacheable check box.
3. To set an expiration time (maximum lifetime), perform the following steps:
   a. In the Cache persistence time drop-down list, select a value.

      If you select Infinite or until you select a different value, the Cache Persistence time field will not be available.
   b. Complete the Cache persistence time field.
4. Click OK.

Configure Oracle BI Server Event Polling Tables

Oracle BI Server event polling tables store information about updates in the underlying databases. An application (such as one that loads data into a data mart) could be configured to add rows to an event polling table each time a database table is updated. The Oracle BI Server polls this table at set intervals and invalidates any cache entries corresponding to the updated tables. Event polling tables can be your sole method of cache management or can be used in conjunction with other cache management schemes. Event tables offer less flexibility about choice of cache entries and the timing of purges. For more information about event polling tables, refer to "Setting Up Event Polling Tables on the Physical Databases" on page 249.

Purging and Maintaining Cache Using ODBC Procedures

The Oracle BI Server provides ODBC-extension functions for the Oracle BI Administrator to use for purging cache entries.

Some of these functions are particularly useful for embedding in an Extract, Transform, and Load (ETL) task. For example, after a nightly ETL is performed, all Oracle BI Server cache can be purged. If only the fact table was modified, only cache related to that table can be purged. In some cases, you might need to purge the cache entries associated with a specific database.

NOTE: Only Oracle BI Administrators have the right to purge cache. Therefore scripts that call these ODBC-extension functions must run under an Oracle BI Administrator logon ID.

The following ODBC functions affect cache entries associated with the repository specified by the ODBC connection:

- **SAPurgeCacheByQuery.** Purges a cache entry that exactly matches a specified query. For example, using the following query, you would have a query cache entry that retrieves the names of all employees earning more than $100,000:

  ```
  select lastname, firstname from employee where salary > 100000;
  ```

  The following call purges the cache entry associated with this query:

```
Call SAPurgeCacheByQuery(‘select lastname, firstname from employee where salary >
100000’ );
```

- **SAPurgeCacheByTable.** Purges all cache entries associated with a specified physical table name (fully qualified) for the repository to which the client has connected.

  This function takes up to four parameters representing the four components (database, catalog, schema and table name proper) of a fully qualified physical table name. For example, you might have a table with the fully qualified name of DBName.CatName.SchName.TabName. To purge the cache entries associated with this table in the physical layer of the Oracle BI repository, execute the following call in a script:

  ```
  Call SAPurgeCacheByTable( ‘DBName’, ‘CatName’, ‘SchName’, ‘TabName’ );
  ```

  NOTE: Wild cards are not supported by the Oracle BI Server for this function. Additionally, DBName and TabName cannot be null. If either one is null, you will receive an error message.

- **SAPurgeAllCache.** Purges all cache entries. The following is an example of this call:

  ```
  Call SAPurgeAllCache();
  ```

- **SAPurgeCacheByDatabase.** Purges all cache entries associated with a specific physical database name. A record is returned as a result of calling any of the ODBC procedures to purge the cache. This function takes one parameter that represents the physical database name and the parameter cannot be null. The following shows the syntax of this call:

  ```
  Call SAPurgeCacheByDatabase( ‘DBName’ );
  ```

About Sharing Presentation Server Cache

When users access the Intelligence Dashboard to run queries, Oracle BI Presentation Services caches the results of the queries. Oracle BI Presentation Services uses the request key and the logical SQL string to determine if subsequent queries can use cached results. If the cache can be shared, subsequent queries are not stored.

- **SAGetSharedRequestKey.** An ODBC procedure that takes a logical SQL statement from the Oracle BI Presentation Services and returns a request key value.

  The following shows the syntax of this procedure:

  ```
  SAGetSharedRequestKey(‘sql-string-literal)
  ```

About Result Records

The result record contains two columns. The first column is a result code and the second column is a short message describing result of the purge operation. The following list contains examples of result records:

Result Code	Result Message
1	SAPurgeCacheByDatabase returns successfully.

Result Code	Result Message
E_Execution_CacheNotEnabled	Operation not performed because caching is not enabled.
E_Execution_NonExistingDatabase	The database specified does not exist.

Storing and Purging Cache for SAP/BW Data Sources

In Microsoft Analysis Services, member caption name is the same as member unique name. However, in SAP/BW data sources, member caption name is different from member unique name. Therefore, the Oracle BI Server maintains a cache subsystem for SAP/BW member unique names. This subsystem is turned off by default. For configuration information, refer to the topic about the MDX Member Name Cache Section of the NQSConfig.INI file in *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

When a query is received for member unique name, the subsystem checks the cache to determine whether cache exists for this query. If cache exists, the record for the cached unique name is returned. If there is no cache that matches the query, the subsystem sends a probing query to SAP/BW.

The probing query will be logged when the log level is equal or greater than 2. The status of the subsystem, such as if the subsystem is enabled and events such as start and shutdown events, are also written to the server log.

CAUTION: With each increased logging level, performance is impacted. Use caution when increasing the log level for users.

Purging Cache for SAP/BW Data Sources

Only Oracle BI Administrators have privileges to run ODBC purge procedures. It is the responsibility of the Oracle BI Administrator to maintain the cache. Therefore, the Oracle BI Administrator should be aware of the following issues:

- The size of multidimensional cache entries can grow very large. Therefore, a limit on the size of each member set has been established in the MDX_MEMBER_CACHE section of the NQSConfig.INI file.
- The format of persisted cache might not be consistent after an upgrade. Therefore, the Oracle BI Administrator should purge all cache before a software upgrade.
- The cache will be populated the first time the query runs. The Oracle BI Administrator should arrange to populate the cache during off-peak hours, to minimize performance impact.

NOTE: In the Oracle BI Administration Tool, the Oracle BI Administrator can purge cache for an individual cube table by right-clicking the cube table, and then selecting Purge Member Cache. This must be performed in online mode and by a user with Oracle BI Administrator privileges.

The following purge procedures are specific to SAP/BW data sources:

- **SAPurgeALLMCNCache.** Purges all SAP/BW cache entries.

  The following shows the syntax of this procedure:

```
SAPurgeALLIMCNCache ()
```

- **SAPurgeMCNCacheByCube.** Purges all cache entries associated with the specified physical cube. The database name and cube name are the external names of the repository objects. The following shows the syntax of this procedure:

  ```
  SAPurgeMCNCacheByCube( ‘DBName’, ‘CubeName’)
  ```

The following messages will be returned.

Return Code	Return Message
1	SAPurgeALLMCNCache returns successfully.
1	SAPurgeMCNCacheByCube returns successfully.
E_Execution_NonExistingDatabase(*)	The database specified does not exist. **NOTE:** If the database and physical cube are both wrong, this result code will be returned.
E_Execution_NonExistingPhysicalCube	The physical cube specified does not exist.

Strategies for Using the Cache

One of the main advantages of query caching is to improve apparent query performance. It may be valuable to *seed* the cache during off hours by running queries and caching their results. A good seeding strategy requires that you know when cache hits occur.

If you want to seed the cache for all users, you might seed the cache with the following query:

```
Select User, SRs
```

After seeding the cache using `Select User, SRs`, the following queries will all be cache hits:

```
Select User, SRs where user = valueof(nq_SESSION.USER) (and the user was USER1)

Select User, SRs where user = valueof(nq_SESSION.USER) (and the user was USER2)

Select User, SRs where user = valueof(nq_SESSION.USER) (and the user was USER3)
```

Cache Hits

When caching is enabled, each query is evaluated to determine whether it qualifies for a cache hit. A cache hit means that the server was able to use cache to answer the query and did not go to the database at all.

NOTE: The Oracle BI Server can use query cache to answer queries at the same or higher level of aggregation.

A cache hit occurs only if all of the conditions described in this section are met.

- **WHERE clause semantically the same or a logical subset.** For the query to qualify as a cache hit, the WHERE clause constraints need to be either equivalent to the cached results, or a subset of the cached results.

  A WHERE clause that is a logical subset of a cached query qualifies for a cache hit if the subset meets one of the following criterion:

  - A subset of IN list values.

    Queries requesting fewer elements of an IN list cached query qualify for a cache hit. For example, the following query:

    ```
    select employeename, region
    from employee, geography
    where region in ('EAST', 'WEST')
    ```

    qualifies as a hit on the following cached query:

    ```
    select employeename, region
    from employee, geography
    where region in ('NORTH', 'SOUTH', 'EAST', 'WEST')
    ```

  - It contains fewer (but identical) OR constraints than the cached result.
  - It contains a logical subset of a literal comparison.

    For example, the following predicate:

    ```
    where revenue < 1000
    ```

    qualifies as a cache hit on a comparable query with the predicate:

    ```
    where revenue < 5000
    ```

  - There is no WHERE clause.

    If a query with no WHERE clause is cached, *queries that satisfy all other cache hit rules* qualify as cache hits regardless of their WHERE clause.

- **A subset of columns in the SELECT list.** All of the columns in the SELECT list of a new query have to exist in the cached query in order to qualify for a cache hit, or they must be able to be calculated from the columns in the query.
- **Columns in the SELECT list can be composed of expressions on the columns of the cached queries.** The Oracle BI Server can calculate expressions on cached results to answer the new query, but all the columns have to be in the cached result.

  For example, the query:

  ```
  select product, month, averageprice from sales where year = 2000
  ```

  will hit cache on the query:

  ```
  select product, month, dollars, unitsales from sales where year = 2000
  ```

  because averageprice can be computed from dollars and unitsales (averageprice = dollars/ unitsales).

- **Equivalent join conditions.** The resultant joined logical table of a new query request has to be the same as (or a subset of) the cached results to qualify for a cache hit.
- **DISTINCT attribute the same.** If a cached query eliminates duplicate records with DISTINCT processing (for example, SELECT DISTINCT...), requests for the cached columns have to also include the DISTINCT processing; a request for the same column without the DISTINCT processing will be a cache miss.
- **Compatible aggregation levels.** Queries that request an aggregated level of information can use cached results at a lower level of aggregation.

  For example, the following query:

  ```
  select supplier, region, city, qtysold
  from suppliercity
  ```

  requests the quantity sold at the supplier and region and city level, while the following query:

  ```
  select city, qtysold
  from suppliercity
  ```

  requests the quantity sold at the city level. The second query would result in a cache hit on the first query.
- **Limited additional aggregation.** For example, if a query with the column *qtysold* is cached, a request for RANK(qtysold) results in a cache miss. Additionally, a query requesting qtysold at the country level can get a cache hit from a query requesting qtysold at the country, region level.
- **ORDER BY clause made up of columns in the select list.** Queries that order by columns not contained in the select list result in cache misses.

Running a Suite of Queries to Populate the Cache

To maximize potential cache hits, one strategy is to run a suite of queries just for the purpose of populating the cache. The following are some recommendations for the types of queries to use when creating a suite of queries with which to seed the cache.

- Common prebuilt queries.

  Queries that are commonly run, particularly ones that are expensive to process, are excellent cache seeding queries. Queries whose results are embedded in Intelligence Dashboards would be good examples of common queries.
- SELECT lists with no expressions.

  Eliminating expressions on SELECT list columns expands the possibility for cache hits. A cached column with an expression can only answer a new query with the same expression; a cached column with no expressions can answer a request for that column with any expression. For example, a cached request such as:

  ```
  SELECT QUANTITY, REVENUE...
  ```

  can answer a new query such as:

```
    SELECT QUANTITY/REVENUE...
```

but not the reverse.

- No WHERE clause.

  If there is no WHERE clause in a cached result, it can be used to answer queries satisfying the cache hit rules for the select list with any WHERE clause that includes columns in the projection list.

In general, the best queries to seed cache with are queries that heavily consume database processing resources and that are likely to be reissued. Be careful not to seed the cache with simple queries that return many rows. These queries (for example, SELECT * FROM PRODUCTS, where PRODUCTS maps directly to a single database table) require very little database processing. Their expense is network and disk overhead—factors that caching will not alleviate.

NOTE: When the Oracle BI Server refreshes repository variables, it will examine business models to determine if they reference those repository variables. If they do, the Oracle BI Server purges all cache for those business models.

Creating Aggregates for Oracle BI Server Queries

Aggregate tables store precomputed results that are aggregated measures (typically summed) over a set of dimensional attributes. Using aggregate tables is a typical technique used to improve query response times in decision support systems.

If you write SQL queries or use a tool that only understands what physical tables exist and not their meaning, using aggregate tables becomes more complex as the number of aggregate tables increases. The aggregate navigation capability of the Oracle BI Server allows queries to use the information stored in aggregate tables automatically. The Oracle BI Server allows you to concentrate on asking the right business question, and then the server decides which tables provide the fastest answers.

The traditional process of creating aggregates for Oracle BI Server queries is manual. It requires writing complicated DDL and DML to create tables in the databases involved. Additionally, these tables need to be mapped into the repository metadata to be available for queries. This is a time consuming, and a potentially error-prone process. The Aggregate Persistence module allows the Oracle BI Administrator to automate the creation of aggregate tables and their mappings into the metadata.

Aggregate creation will run against the master server in a cluster. It will take some time for the metadata changes to propagate to the slaves. The cluster refresh time is a user-controlled option and results may be incorrect if a query hits a slave server before it is refreshed. It is the Oracle BI Administrator's responsibility to set an appropriate cluster refresh interval.

The NQSConfig.INI file contains the following optional parameter in the GENERAL section:

```
AGGREGATE_PREFIX = "user specified short prefix for dimension aggregates" ;
```

This parameter has a maximum of 8 characters and is specified if you want to add a prefix to automatically generated dimension (level) aggregates. If not specified, the default prefix SA_ will be used.

NOTE: Only the Oracle BI Administrators group is allowed to manage aggregates.

This section contains the following topics:

- Identifying Query Candidates for Aggregation on page 242
- About Writing the Create Aggregates Specification on page 242
- Generating the SQL Script File on page 247
- About Setting the Logging Level on page 247
- Executing the SQL Script File to Create and Delete Aggregates on page 247
- Post Creation Activities on page 247

Identifying Query Candidates for Aggregation

When creating aggregates you need to identify which queries would benefit substantially from aggregated data. You will achieve the best results by aggregating to the highest level possible. To identify slow running queries, perform the following tasks:

- Enable usage tracking in the Oracle BI Server.

  Usage tracking statistics can be used in a variety of ways, for example, database optimization, aggregation strategies, and billing users or departments based on the resources they consume. The Oracle BI Server tracks usage at the detailed query level. When you enable usage tracking, statistics for every query are written to a usage tracking log file or inserted into a database table.

  NOTE: It is strongly recommended that you use direct insertion into a database. For instructions, refer to "Administering Usage Tracking" on page 219.

- Analyze the query runtimes and identify the slowest running queries as candidates for aggregation.

  The run time for creating aggregates is dependent on the type of aggregates selected by the user. Creating aggregates from large fact tables will be slower than from smaller tables. The Oracle BI Administrator needs to carefully select the aggregates to be created.

About Writing the Create Aggregates Specification

To create the script file, you can use the Aggregate Persistence Wizard in the Administration Tool or write the file manually.

NOTE: It is recommended that you use the Aggregate Persistence Wizard. For instructions, refer to "Aggregate Persistence Wizard" on page 189.

If you do not want Oracle BI Server to modify your databases during aggregate creation, you can use the Aggregate Persistence Wizard to create the SQL file. After creating the SQL, you can use your database administration processes to create your aggregate tables.

Constraints Imposed During the Create Process

This section describes the constraints that are imposed during the create process.

Valid Measures

A valid measure must have a valid aggregation rule.

- The following constraints apply to level-based measures:
  - If the level is grand total alias, then that dimension must not be present in the list of levels for that aggregate specification.
  - Any other level defined for this measure, must be present in the list of levels for that aggregate specification.

  If the above constraints are not met, the entire aggregate specification will be discarded.
- A measure will be ignored by the create process if any of the following conditions are true:
  - Measure is mapped to a session or repository variable.
  - Measure is a derived measure.

  Measures that are ignored do not necessarily affect the aggregate specification. The remaining measures will be used to create the aggregate.

Valid Levels

A valid level must have a valid primary key.

- If a level is invalid, the aggregate specification will be discarded.
- Attributes of a level or its primary key will be ignored, if any of the following conditions are true:
  - Attribute is mapped to session or repository variables.
  - Attributes are not from the same logical table.

Valid Aggregate Specification

A valid aggregate specification has the following properties:

- Name length is between 1 and 18 characters (inclusive).
- At least one valid level must be specified.
- At least one valid measure must be specified.
- Must have a valid connection pool.
- Must have a valid output container (database/catalog/schema).
- Connection pool and container must belong to the same database.

- Only one level per dimension can be specified.
- Measures can only be from the same fact table.
- All logical components of the specification must be from the same subject area.

An aggregate specification will be ignored if the name already exists in the output container because level aggregates are scoped by the entire database. However, if different catalogs or schemas are specified for the same fact aggregate name, it is allowed to have multiple facts with the same name but different scope in the same database.

Guidelines for Writing the Create Aggregates Specification

All metadata names (except for logical fact columns) are fully qualified. There are two modes of operation: Create and Delete.

It is strongly recommended to place all aggregate specifications under a single Create Aggregates statement.

- Begin the script file with a Delete statement. It is essential to delete system generated aggregates before creating any new ones. This makes sure that data is consistent and it removes invalid or incomplete aggregates before you run the Create operation. The following statement is the syntax for deleting aggregates:

  ```
  Delete aggregates;
  ```

- The next statement should be a Create statement. The following is the syntax for creating aggregates:

  ```
  Create|Prepare aggregates

  <aggr_name_1>

  for  logical_fact_table_1 [(logical_fact_column_1, logical_fact_column_2,…)]

  at levels (level_1, level_2, …)

  using connection pool <connection_pool_name_1>

  in <schema_name_1>

  [ ,<aggr_name_2>

  for logical_fact_table_3 [(logical_fact_column_5, logical_fact_column_2,…)]

  at levels (level_3, level_2, …)

  using connection pool <connection_pool_name_2>

  in <schema_name_2>] ;
  ```

- Creating multiple aggregates. To specify more than one aggregate in a single Create Aggregates statement, use the following guidelines:
  - Each of the multiple aggregate specifications are separated by a comma, and the entire aggregate creation script is terminated with a semi-colon.

- In this file, only one Delete Aggregates statement should be specified at the beginning. The Oracle BI Administrator should make sure that only one delete is issued per ETL run (unless a reset is called for).

  CAUTION: Any aggregate scripts that are run after the first one should not have a Delete Aggregates statement or all previously created aggregates will be removed.

- Creating aggregates with surrogate keys. For detailed instructions, refer to "About Adding Surrogate Keys to Dimension Aggregate Tables" on page 245

About Adding Surrogate Keys to Dimension Aggregate Tables

The join option default between fact and level aggregate tables uses primary keys from the level aggregate. If the primary key of the level is large and complex (composite of many columns), the join to the fact table will be expensive. A *surrogate key* is an artificially generated key, usually a number. A surrogate key, in the level aggregate table, simplifies this join and removes unnecessary columns (level primary key) from the fact table, resulting in a smaller-sized fact table. Adding surrogate keys to the dimension (level) aggregate tables can simplify joins to the fact tables and might improve query performance. Additionally, a surrogate key makes sure that each aggregate table has a unique identifier.

There may be cases in which a level is shared among multiple fact tables. One fact may use surrogate keys, and another may use primary keys from the dimension aggregate. The following are some options for resolving this issue:

- Set a metadata property for levels that indicates whether to use surrogate keys or primary keys.
- Always create a surrogate key for a level aggregate (relatively low cost operation). Then decide later if the fact aggregate should join to it using a surrogate or primary key.

An alternative to specifying the join type for each dimension is to specify if surrogate keys should be used for the entire star. This would result in simpler syntax but would also restrict the available user options and slow the aggregate creation process.

Surrogate Key Input for Create/Prepare Aggregates

The Oracle BI Administrator can create the aggregate star using the following join options:

- Primary Keys (default, if no option is specified)
- Surrogate Keys

Syntax for Create/Prepare Aggregates

The following syntax for create/prepare aggregates contains the change for [Using_Surrogate_Key]. The surrogate key option can be specified for each level. If unspecified, the fact and dimension tables are joined using the primary key from the level aggregate.

```
Create|Prepare aggregates

<aggr_name_1>

[file <output_file_name>]
```

```
for  logical_fact_table_1 [(logical_fact_column_1, logical_fact_column_2,…)]
at levels (level_1 [Using_Surrogate_Key], level_2, …)
using connection pool <connection_pool_name_1>
in <schema_name_1>
[ ,<aggr_name_2>
for logical_fact_table_3 [(logical_fact_column_5, logical_fact_column_2,…)]
at levels (level_3, level_2, …)
using connection pool <connection_pool_name_2>
in <schema_name_2>] ;
```

Surrogate Key Output From Create/Prepare Aggregates

The changes to the current process are restricted to the physical metadata layer in the repository and the database.

When you use the UseSurrogateKeys join option, the following describes the results:

- For a level aggregate, the following occurs:
  - In the physical metadata, the following occurs:
    - The level aggregate table will have a new column called <level_name>_SKEY (check for collisions). This is the surrogate key column for the dimension aggregate.
    - The type of this column is UINT.
  - In the database, the following occurs:
    - The level aggregate table will also have a corresponding column called <level_name>_SKEY.
    - It can be populated using RCOUNT ().
- For a fact aggregate, the following occurs:
  - In the physical metadata, the following occurs:
    - The fact aggregate table will no longer contain columns from the level's primary keys.
    - Instead, a new column that corresponds to the level aggregate's surrogate key will be added to the table.
    - The type of this column will be identical to the level's surrogate key.
    - The column will have the same name as that in the level aggregate (check for collisions).
    - The fact table and the level table will be joined using this surrogate key only.
  - In the database, the following occurs:
    - The fact aggregate table will also have the corresponding surrogate key.
    - It is populated using new capabilities to be available through Populate.

Generating the SQL Script File

Now that you understand more about how to specify aggregates, you can write the aggregate logical SQL or generate a SQL script file using the Aggregate Persistence Wizard.

NOTE: It is strongly recommended that you use the Aggregate Persistence Wizard because it automatically enforces many of the constraints outlined in the previous section. If you choose to write the SQL script manually, use the syntax as described in "Guidelines for Writing the Create Aggregates Specification" on page 244.

About Setting the Logging Level

Trace logs will be logged to NQQuery.log if the logging level is at least 2. The logging events will include the aggregate execution plan and the order in which the aggregates are created and deleted. Higher logging levels provide more details about the query and execution plans.

Error logs will be logged to NQQuery.log if the logging level is at least 1 and to NQServer.log regardless of the logging level.

Executing the SQL Script File to Create and Delete Aggregates

After generating the SQL script file and setting the logging levels, you need to execute the SQL script. You can run nQCmd.exe from the Command Line prompt or the Job Manager utility in the Administration Tool.

NOTE: It is strongly recommended that you use Job Manager to run nQCmd.exe. For more information, refer to *Oracle Business Intelligence Scheduler Guide*. If you choose to run nQCmd.exe, a version can be found in the following Oracle BI installation folder: [*installdrivepath*]:\OracleBI\server\Bin.

After executing the SQL script, aggregates are created and persisted in the Oracle BI Server metadata as well as in the backend databases.

Post Creation Activities

This section discusses the following topics:

- About Database Index Creation on page 248
- About Error Handling on page 248

About Database Index Creation

Currently, database table indexes are not generated automatically. If required, the database Administrator has to manually create these indexes on the database tables. Since the dimension tables are automatically generated, it is useful to set the logging level to 2 or higher to view the Aggregate Creation Plan in NQQuery.log. This plan, along with the aggregate specifications, can be used as a reference to locate automatically generated tables in the database.

You can manually embed dropping and building indexes in the logical SQL script using the EXECUTE PHYSICAL capability. The following is an example of the statements that you might add to the beginning of your script:

```
EXECUTE PHYSICAL CONNECTION POOL "SQL_Paint"."SQL_Paint" DROP INDEX demo_index1;

CREATE AGGREGATES......;

EXECUTE PHYSICAL CONNECTION POOL "SQL_Paint"."SQL_Paint" CREATE INDEX demo_index1
ON table1(col1);
```

About Error Handling

The following is a list of some reasons errors can occur:

- Network failure.
- No disk space on the database.
- Bad aggregate request.

If there is an error in the creation of any aggregate, the entire aggregate request is aborted and subsequent aggregates are not created. Aggregates that are already created and checked in, remain checked in. If there are errors, you need to remove them at the time of the error or at the next ETL run in one of the following ways:

- Manually remove the aggregates from the metadata and the database.
- Automatically remove all the aggregates using the Delete Aggregates specification.

Cache Event Processing with an Event Polling Table

The use of an Oracle BI Server event polling table (event table) is a way to notify the Oracle BI Server that one or more physical tables have been updated. Each row that is added to an event table describes a single update event, such as an update occurring to the Product table in the 11308Production database. The Oracle BI Server cache system reads rows from, or *polls*, the event table, extracts the physical table information from the rows, and purges stale cache entries that reference those physical tables.

The event table is a physical table that resides on a database accessible to the Oracle BI Server. Regardless of where it resides—in its own database, or in a database with other tables—it requires a fixed schema, described in Table 27 on page 250. It is normally exposed only in the Physical layer of the Administration Tool, where it is identified in the Physical Table dialog box as being an Oracle BI Server event table.

The use of event tables is one of the most accurate ways of invalidating stale cache entries, and it is probably the most reliable method. It does, however, require the event table to be populated each time a database table is updated (refer to "Populating the Oracle BI Server Event Polling Table" on page 253). Also, because there is a polling interval in which the cache is not completely up to date, there is always the potential for stale data in the cache.

A typical method of updating the event table is to include SQL INSERT statements in the extraction and load scripts or programs that populate the databases. The INSERT statements add one row to the event table each time a physical table is modified. After this process is in place and the event table is configured in the Oracle BI repository, cache invalidation occurs automatically. As long as the scripts that update the event table are accurately recording changes to the tables, stale cache entries are purged automatically at the specified polling intervals.

Setting Up Event Polling Tables on the Physical Databases

This section describes how to set up the Oracle BI Server event polling tables on physical databases.

Polling Table Structure

You can set up a physical event polling table on each physical database to monitor changes in the database. You can also set up the event table in its own database. The event table should be updated every time a table in the database changes. The event table needs to have the structure shown in Table 27 on page 250; some columns can contain null values depending on where the event table resides.

The column names for the event table are suggested; you can use any names you want. However, the order of the columns has to be the same as shown in Table 27 on page 250. Sample CREATE TABLE statements to create an event polling table are shown in "Sample Event Polling Table CREATE TABLE Statements" on page 251.

Table 27. Event Polling Table Column Names

Event Table Column Name	Data Type	Description
CatalogName	CHAR or VARCHAR	The name of the catalog where the physical table that was updated resides. Populate the CatalogName column only if the event table does not reside in the same database as the physical tables that were updated. Otherwise, set it to the null value.
DatabaseName	CHAR or VARCHAR	The name of the database where the physical table that was updated resides. This is the name of the database as it is defined in the Physical layer of the Administration Tool. For example, if the physical database name is 11308Production, and the database name that represents it in the Administration Tool is SQL_Production, the polled rows in the event table has to contain SQL_Production as the database name. Populate the DatabaseName column only if the event table does not reside in the same database as the physical tables that were updated. Otherwise, set it to the null value.
Other	CHAR or VARCHAR	Reserved for future enhancements. This column must be set to a null value.
SchemaName	CHAR or VARCHAR	The name of the schema where the physical table that was updated resides. Populate the SchemaName column only if the event table does not reside in the same database as the physical tables being updated. Otherwise, set it to the null value.
TableName	CHAR or VARCHAR	The name of the physical table that was updated. The name has to match the name defined for the table in the Physical layer of the Administration Tool. Values cannot be null.

Table 27. Event Polling Table Column Names

Event Table Column Name	Data Type	Description
UpdateTime	DATETIME	The time when the update to the event table occurs. This needs to be a key (unique) value that increases for each row added to the event table. To make sure a unique and increasing value, specify the current timestamp as a default value for the column. For example, specify DEFAULT CURRENT_TIMESTAMP for Oracle 8i. Values cannot be null.
UpdateType	INTEGER	Specify a value of 1 in the update script to indicate a standard update. (Other values are reserved for future use.) Values cannot be null.

The Oracle BI Server needs to have read and write permission on the event polling table. The server reads the event table at specified intervals to look for changed data. Applications add rows to the event table when database tables are modified (for example, during a load operation). When there are rows in the event table, there is changed data in the underlying databases. The server then invalidates any cache entries corresponding to the changed physical tables and periodically deletes obsolete rows from the event table. The next time it checks the event table, the process repeats.

NOTE: A single event polling table cannot be shared by multiple Oracle BI Servers. When you set up multiple Oracle BI Servers, you need to create an event polling table for each one.

To allow Oracle BI Server to have write access to the event polling table but not to any other tables in a database, perform the following tasks:

- Create a separate physical database in the Physical layer of the Administration Tool with a privileged connection pool.
- Assign a user to the connection pool that has delete privileges.
- Populate the privileged database with the event table.

The Oracle BI Server will have write access to the event polling table, but not to any tables that are used to answer user queries.

Sample Event Polling Table CREATE TABLE Statements

The following are sample CREATE TABLE statements for SQL Server 7.0 and Oracle 8i. These CREATE TABLE statements create the structure required for an Oracle BI Server event polling table. In these statements, the table created is named UET. It resides in the same database as the physical tables that are being updated.

NOTE: The column lengths need to be large enough to represent the object names in your repository.

The following is the CREATE TABLE statement for SQL Server 7.0:

```
// SQL Server 7.0 Syntax
create table UET (
  UpdateType Integer not null,
  UpdateTime datetime not null DEFAULT CURRENT_TIMESTAMP,
  DBName     char(40) null,
  CatalogName varchar(40) null,
  SchemaName  varchar(40) null,
  TableName   varchar(40) not null,
  Other       varchar(80) null DEFAULT NULL
)
```

The following is the CREATE TABLE statement for Oracle 8i:

```
// Oracle 8i syntax
create table UET (
  UpdateType Integer not null,
  UpdateTime date DEFAULT SYSDATE not null,
  DBName     char(40) null,
  CatalogName varchar(40) null,
  SchemaName  varchar(40) null,
  TableName   varchar(40) not null,
  Other       varchar(80) DEFAULT NULL
);
```

You might need to modify these CREATE TABLE statements slightly for different versions of SQL Server and Oracle, or for other databases. Additionally, if you want to specify any explicit storage clauses, you need to add the appropriate clauses to the statements.

Making the Event Polling Table Active

After the table is created on the physical database, you can make it active in the Oracle BI Server.

To make the polling table active

1. Import the table to the Physical layer.
2. Include it in the group of Oracle BI Server event polling tables using the Tools > Utilities > Oracle Event Tables menu item, and set a polling interval.

To import the polling table into the Physical layer, perform the following steps from an open repository.

To import the table into the Physical layer

1. Select File > Import...
2. Select the data source containing the event table to import, and then click OK.

   The Import dialog box appears.
3. Check the Tables option to import the table metadata.

4. Navigate to the event polling table, select the table and either click the Import button or drag and drop it into the Physical layer.

This imports the event table to the Physical layer. If you have multiple event polling tables, repeat this procedure for each event table.

NOTE: Be sure the data source specified for the event table has read and write access to the event table. The repository will both read the table and delete rows from it, so it needs write permission. Event tables do not need to be exposed in the Business Model and Mapping layer.

After one or more polling tables are present in the Physical layer, you need to include them in the group of event polling tables.

To mark the table object as an Event Polling Table

1. Click on the Tools > Utilities menu item.
2. Select the option Oracle BI Event Tables from the list of options.
3. Click Execute.
4. Select the table to register as an Event Table and click the >> button.
5. Specify the polling frequency in minutes, and click OK.

   The default value is 60 minutes.

   NOTE: You should not set the polling frequency to less than 10 minutes. If you want a very short polling interval, consider marking some or all of the tables noncachable.

When a table has been registered as an Oracle BI Server event table, the table properties change. Registration as an event table removes the option to make the table cachable, as there is no reason to cache results from an event polling table.

Populating the Oracle BI Server Event Polling Table

The Oracle BI Server does not populate the event polling table. The event table is populated by inserting rows into it each time a table is updated. This process is normally set up by the database Administrator; typically, the load process is modified to insert a row into the polling table each time a table is modified. This can be done from the load script, using database triggers (in databases that support triggers), from an application, or manually. If the process of populating the event table is not done correctly, the Oracle BI Server cache purging will be affected; the server assumes the information in the polling table to be correct and up to date.

Troubleshooting Problems with an Event Polling Table

If you experience problems with cache polling, you can search the Oracle BI Server activity logs for any entries regarding the server’s interaction with the event table.

- The NQServer.log file logs activity automatically about the Oracle BI Server. The default location for this file is the Log folder in the Oracle BI Server software installation folder (\OracleBI\Log). Log entries are self-explanatory and can be viewed using a text editor.
- When the Oracle BI Server polls the event table, it will log the queries in the NQQuery.log file using the Oracle BI Administrator user ID unless the logging level for the Oracle BI Administrator is set to 0. You should set the logging level to 2 for the Oracle BI Administrator user ID to provide the most useful level of information. The default location for the NQQuery.log file is the Log folder in the Oracle BI Server software installation folder (\OracleBI). For more information about user-level logging, refer to "Query Caching in the Oracle BI Server" on page 229.

Making Changes to a Repository

When you modify Oracle BI repositories, the changes can have implications for entries that are stored in the cache. For example, if you change the definition of a physical object or a dynamic repository variable, cache entries that reference that object or variable may no longer be valid. These changes might result in the need to purge the cache. There are three scenarios to be aware of—when the changes occur in online mode, when they occur in offline mode, and when you are switching between repositories.

Online Mode

When you modify an Oracle BI repository in online mode, any changes you make that will affect cache entries automatically result in a purge of all cache entries that reference the changed objects. The purge occurs when you check in the changes. For example, if you delete a physical table from a repository, all cache entries that reference that table are purged upon check in. Any changes made to a business model in the Business Model and Mapping layer will purge all cache entries for that business model.

Offline Mode

When you modify an Oracle BI repository in offline mode, you might make changes that affect queries stored in the cache and render those cached results obsolete. Because the repository is not loaded by the server during offline mode edits, the server has no way of determining if the changes made affect any cached entries. The server therefore does *not* automatically purge the cache after offline changes. If you do not purge the cache, there might be invalid entries when the repository is next loaded. Unless you are sure that there are no entries in the cache that are affected by your offline changes, you should purge the cache for any business model you have modified.

Switching Between Repositories

If you intend to remove a repository from the configuration of the Oracle BI Server, make sure to purge the cache of all cache entries that reference the repository. Failure to do so will result in a corrupted cache. For information, refer to "Purging Cache" on page 256.

Using the Cache Manager

The Cache Manager provides Oracle BI Administrators the capability of viewing information about the entire query cache, as well as information about individual entries in the query cache associated with the open repository. It also provides the ability to select specific cache entries and perform various operations on those entries, such as viewing and saving the cached SQL call, or purging them.

To open the Cache Manager

- In the Administration Tool toolbar, select Manage > Cache.

Select the Cache tab on the left explorer pane to view the cache entries for the current repository, business models, and users. The associated cache entries are reflected in the right pane, with the total number of entries shown in the view-only field at the top.

The cache entry information and its display sequence is controlled by your Options settings (select Edit > Options... from the Cache Manager, or Tools > Options > Cache Manager tab from the Administration Tool). Information may include the options in Table 28 on page 255.

Table 28. Cache Options

Option	Description
Business model	The name of the business model associated with the cache entry.
Column count	The number of columns in each row of this cache entry's result set.
Created	The time the cache entry's result set was created.
Creation elapsed time	The time, in seconds, needed to create the result set for this cache entry. **NOTE:** The value stored in the cache object descriptor on disk is in units of milliseconds. The value is converted to seconds for display purposes.
Full size	Full size is the maximum size used, considering variable length columns, compression algorithm, and other factors. The actual size of the result set will be smaller than Full size.me, in seconds, needed to create the result set for this cache entry.
Last used	The last time the cache entry's result set satisfied a query. (After an unexpected shutdown of the Oracle BI Server, the last used time may temporarily have a *stale* value—a value that is older than the true value.)
Query Server	The Oracle BI Server that serviced the query.
Row count	The number of rows generated by the query.
Row size	The size of each row (in bytes) in this cache entry's result set.
SQL	The SQL statement associated with this cache entry.
Use count	The number of times this cache entry's result set has satisfied a query (since Oracle BI Server startup).
User	The ID of the user who submitted the query that resulted in the cache entry.

Expand the repository tree to display all the business models with cache entries, and expand the business models to display all users with cache entries. The right pane displays only the cache entries associated with the selected item in the hierarchical tree.

Displaying Global Cache Information

Select Action > Show Info... to display global cache information. Table 29 on page 256 describes the information that appears in the Global Cache Information window.

Table 29. Global Cache Information

Column	Description
Amount of space still available for cache storage use	The amount of space, in megabytes, still available for cache storage.
Amount of space used on disks containing cache related files	The *total* amount of space, in megabytes, used on the disk containing cache-related files (not just space used for the cache-related files).
Maximum allowable number of entries in cache	The maximum number of entries that may be in your cache, from the MAX_CACHE_ENTRIES parameter in the NQSConfig.INI file.
Maximum allowable number of rows per cache entry result set	The maximum number of rows allowed for each cache entry's result set, from the MAX_ROWS_PER_CACHE_ENTRY parameter in the NQSConfig.INI file.
Number of entries currently in cache	The current number of entries in your global cache. These entries may relate to multiple repositories.
Number of queries not satisfied from cache since startup of Oracle BI Server	Cache misses, since the last time the Oracle BI Server was started.
Number of queries satisfied from cache since startup of Oracle BI Server	Cache hits, since the last time the Oracle BI Server was started.

With the Cache Manager as the active window, press F5, or select Action > Refresh to refresh the display. This retrieves the current cache entries for the repository you have open, as well as the current global cache information. If the DSN is clustered, information about all repositories in the cluster will be displayed.

Purging Cache

Purging cache is the process of deleting entries from the query cache. You can purge cache entries in the following ways:

- Manually, using the Administration Tool Cache Manager facility (in online mode).
- Automatically, by setting the Cache Persistence Time field in the Physical Table dialog box for a particular table.
- Automatically, by setting up an Oracle BI Server event polling table.
- Automatically, as the cache storage space fills up.

To purge the cache manually with the Cache Manager facility

1. Use the Administration Tool to open a repository in online mode.
2. Select Manage > Cache to open the Cache Manager dialog box.
3. Select Cache or Physical mode by selecting the appropriate tab in the left pane.
4. Navigate the explorer tree to display the associated cache entries in the right pane.
5. Select the cache entries to purge, and then select Edit > Purge to remove them.
    - In Cache mode, select the entries to purge from those displayed in the right pane.
    - In Physical mode, select the database, catalog, schema or tables to purge from the explorer tree in the left pane.

    In Cache mode, you can purge:

    - One or more selected cache entries associated with the open repository.
    - One or more selected cache entries associated with a specified business model.
    - One or more selected cache entries associated with a specified user within a business model.

    In Physical mode, you can purge:

    - All cache entries for all tables associated with one or more selected databases.
    - All cache entries for all tables associated with one or more selected catalogs.
    - All cache entries for all tables associated with one or more selected schemas.
    - All cache entries associated with one or more selected tables.

Purging deletes the selected cache entries and associated metadata. Select Action > Refresh or press F5 to refresh your cache display.

About the Refresh Interval for XML Data Sources

This section provides information about the refresh interval for XML data sources.

For more information, refer to "Using XML as a Data Source for the Oracle BI Server" on page 335.

Typically, XML data sources are updated frequently and in real time. Setting a refresh interval for XML data sources is analogous to setting cache persistence for database tables. The refresh interval is a time interval after which the XML data sources are to be queried again directly, rather than using results in cache. This refresh interval is specified on the XML tab of the Connection Pool dialog box.

The default interval setting is Infinite, meaning that the XML data source is not automatically refreshed.

The refresh interval setting specifies the time interval after which the Oracle BI Server XML Gateway connection will be refreshed.

- For URLs that begin with http:// or https://, the gateway will refresh when it detects that the interval has expired.
- For URLs that reside on a local or network drive, the gateway will refresh when the interval has expired and the system detects that the URLs have been modified.

12 Connectivity and Third-Party Tools in Oracle BI Server

The Oracle BI Server provides the functionality for you to connect to it through many client tools and applications. This section contains the following topics:

Configuring Oracle BI ODBC Data Source Names (DSNs)

In a non-English environment, you cannot use a direct ODBC connection to Oracle BI Server. Only the Oracle BI Presentation Services client can connect directly to Oracle BI Server in a non-English environment.

This procedure applies to Windows-based operating systems.

To create a new data source

1. Open the Windows ODBC Control Panel applet by selecting Start > Settings > Control Panel, and then double-click the ODBC Data Sources icon.

   If you are running Windows 2000 or XP, the Data Sources (ODBC) icon is available under Administrative Tools.

2. In the ODBC Data Source Administrator dialog box, click the System DSN tab, and then click Add.
3. Select the Oracle BI Server driver from the Create New Data Source dialog, and then click Finish.

   The first DSN Configuration screen appears.

4. Type a name for the data source in the Name field.
5. (Optional) Type a description in the Description field.
6. If this data source will not participate in a cluster, in the Server field at the bottom of the screen, select the machine on which the Oracle BI Server is running.

   If the server name does not appear in the drop-down list, type the name in the Server field. This needs to be the NetBIOS name (computer name) of the machine.

7 If this data source is to participate in a cluster, do the following:

   a Select the option Is this a Clustered DSN?

      This causes the fields Primary Controller and Secondary Controller to become active, and the Server field to become inactive.

   b Type the name of the machine that is specified as the primary Cluster Controller (from the parameter PRIMARY_CONTROLLER in the NQClusterConfig.INI file). This needs to be the NetBIOS name (computer name) of the machine.

   c If a secondary Cluster Controller has been specified (from the parameter SECONDARY_CONTROLLER in the NQClusterConfig.INI file), type the name of the machine in the Secondary Controller field. The computer name must be unique from that of the primary Cluster Controller.

   d To test the connection to the Cluster Controller, click Test Connect.

      A message indicates if the connection was tested successfully. If the test is not successful, correct any error identified in the message and test the connection again.

8 In the next DSN Configuration screen, type a valid user ID and password for the repository to which you want the data source to connect. If you are using Windows operating system authentication, leave this field blank. You will then log into the Oracle BI Server with the logon ID of your Windows account.

9 If you want to save your logon ID in the repository, check the option Save login ID.

   If you check this option, you will not have to type your logon information each time you connect.

10 In the Port field, specify the TCP/IP port the Oracle BI Server is using for client/server communications.

   The default port is 9703. This port number should match the port number specified in the parameter RPC_SERVICE_OR_PORT in the Server section in the NQSConfig.INI file. If you change the port number in the configuration file, remember to reconfigure any affected ODBC data sources to use the new port number.

   NOTE: The default client/server communication method for the Oracle BI Server has changed from Distributed component object model (DCOM) to TCP/IP. Support for DCOM will be discontinued in a future release. For sites already running the Oracle BI Server that want to continue to use DCOM until support is discontinued, leave this field set to its default value and define a Windows system environment variable named NQUIRE_DCOM to force the usage of DCOM. Set the variable value to 1. (To define a system environment variable, select System from the Control Panel, click the Advanced tab, and then click the Environment Variables button to open the Environment Variables dialog box.)

11 If you want to connect to a repository other than the default repository, select the option Change the default repository to, and then type the logical name of the repository (as it appears in the NQSConfig.INI file) to which you want to connect in the field below the check box.

   If this option is not selected, the data source will connect to the repository marked as the default repository in the NQSConfig.INI file, or to the first repository listed if none of the entries is marked as the default.

12 Select the option Connect to Oracle BI Server to obtain default settings for additional configuration.

The setup will attempt to connect to the server to obtain information about the business models in the repository. If you do not select this option, you can still configure the DSN by manually entering the information in the next configuration screen.

13 Click Next to advance to the next window.

14 To change the default catalog, select the option Change the default catalog to, and then type the name of the catalog in the field below the check box.

The default catalog is the catalog folder that appears at the top of the Presentation layer in the Administration Tool. For the DSN used by Oracle BI Presentation Services, it is better to leave this check box clear with the drop-down box showing no entry.

You can also specify user IDs and passwords for the underlying databases to which the Oracle BI Server connects. If you specify database user IDs and passwords, those are used to connect to the databases if user-specific database logon information is configured in the connection pools, as described in "Creating or Changing Connection Pools" on page 67. The database-specific user IDs and passwords allow privileged users to connect to the underlying databases at the level of authority granted to those users in the databases.

15 At this point, you can change the password for the Oracle BI user the DSN logs in as (if the server is running in a writable mode). To change the password, you must have entered your logon information and selected the option Connect to Oracle BI in the previous screen. The new password is stored in encrypted form in the repository.

ODBC Conformance Level

The Oracle BI Server supports the following ODBC calls from client applications:

- SQLAllocConnect
- SQLAllocEnv
- SQLAllocStmt
- SQLBindCol
- SQLCancel
- SQLColumns
- SQLConnect
- SQLDescribeCol
- SQLDisconnect
- SQLDriverConnect
- SQLError
- SQLExecDirect
- SQLExecute

- SQLExtendedFetch
- SQLFetch
- SQLFreeConnect
- SQLFreeEnv
- SQLFreeStmt
- SQLGetConnectOption
- SQLGetCursorName
- SQLGetData
- SQLGetFunctions
- SQLGetInfo
- SQLGetStmtOption
- SQLGetTypeInfo
- SQLColAttributes
- SQLNumResultCols
- SQLPrepare
- SQLRowCount
- SQLSetConnectOption
- SQLSetStmtOption
- SQL Tables

Oracle BI ODBC supports full scrollable cursors with static, dynamic, forward only, and key set driven cursors.

Oracle BI ODBC supports asynchronous and synchronous processing and cancellation.

Third-Party Tools and Relational Data Source Adapters

The Oracle BI Server allows connectivity between a wide variety of client tools and a wide variety of data sources. For information, refer to *System Requirements and Supported Platforms*.

Importing Metadata

You can import metadata from a data source to an Oracle BI repository. The metadata is used to establish physical table information in the Physical layer of the Administration Tool.

Metadata imports to an Oracle BI repository have to occur through an ODBC or native database connections to the underlying data sources. Metadata can also be imported from software such as Microsoft Excel through an ODBC connection.

For the metadata import procedure, refer to "Process of Creating the Physical Layer from Relational Data Sources" on page 56.

Using Query and Reporting Tools

You can connect to the Oracle BI Server with a wide variety of ODBC-compliant query and reporting tools. Connecting with a query tool is a matter of configuring a data source using the Oracle BI ODBC driver and then using the Oracle BI DSN to connect to a repository from the query tool.

The Presentation layer allows you to configure the presentation of a business model to be consistent with the rules and conventions of your tools to take advantage of the Oracle BI Server's analytical engine and data abstraction. This makes it much easier to include columns involving complex aggregation and calculation rules in queries and reports. Also, if your organization is currently using query and reporting tools, using the Oracle BI Server as a data source will make these tools more valuable and will simplify the work entailed when using them.

Exchanging Metadata with Databases

If your organization has installed either an Oracle Database or an IBM DB2 Database, then you can use these databases to enhance the data warehouse performance and functionality of queries that run on the Oracle BI Server. By exchanging Oracle BI metadata from the Oracle BI Server with the Oracle Database or with the IBM DB2 Database, you allow the database to accelerate the performance of data warehouse queries by using the following tools:

- In the Oracle Database, Oracle Database Summary Advisor to create materialized views and index recommendations on optimizing performance.
- In the IBM DB2 Database, IBM DB2 Cube Views to create materialized query tables (MQTs).

Both of these tools preaggregate the relational data and improve query performance.

Finding Information on Metadata Exchange

The sections in the following list provide information on exchanging metadata:

- Using Materialized Views in the Oracle Database with Oracle BI on page 273
- Using IBM DB2 Cube Views with Oracle BI on page 277

The "Generating the Import File" section provides information that applies to both types of databases.

Generating the Import File

Both the Oracle Database Metadata Generator and the DB2 Cube Views Generator create the files that are needed to import metadata from Oracle BI into the Oracle Database Summary Advisor or an IBM DB2 Database. Before reading this section, read "Finding Information on Metadata Exchange" on page 263.

This section contains the following topics that are common to the two generators:

- Running the Generator on page 264
- About the Metadata Input File on page 265
- About the Output Files on page 265
- Troubleshooting Errors from the Generator on page 266
- Metadata Conversion Rules and Error Messages on page 267

Running the Generator

The Oracle Database Metadata Generator and the DB2 Cube Views Generator are invoked from the command line or embedded in a batch file. The command-line executable is named SAMetaExport.exe, and has the following syntax:

```
SAMetaExport -r "PathAndRepositoryFileName" -u <UserName> -p <Password>
-f "InputFileNameAndPath" [-t “ORACLE” or “DB2”]
```

Table 30 contains descriptions of the parameters in the command-line executable file.

Table 30. Parameters for SAMetaExport.exe

Parameter	Definition	Additional Information
-r	Repository file name and full path	Quotation marks are required for the file name and path only if the file path is in long format or has spaces. Use the full path if the file is not in the current directory.
-u	User name	User name that will allow access to the repository.
-p	Password	Password for the user name. If the repository password is empty, then do not use the password parameter.
-f	Input file name and full path	Quotation marks are required for the file name and path only if the file path is in long format or has spaces. Use the full path if the file is not in the current directory. You specify input files so that you do not have to type all the required information at the command line, and so that you can type international characters. For more information about the input file, refer to About the Metadata Input File on page 265.
-t	ORACLE or DB2	Indication for either the Oracle or IBM DB2 Database. You can omit this parameter because the database type is automatically detected based on information in the repository.

About the Metadata Input File

The input file is a text file that contains the parameters that are described in Table 31.

Table 31. Cube Metadata Input File Parameters

Input File Name	Description
BUSINESS_MODEL	The name of the business model in the logical layer of the Oracle BI repository that contains the metadata that you want to export. If the business model is not found in the repository, then an error message is displayed.
PHYSICAL_DATABASE	The name of the database in the physical layer of the Oracle BI repository that contains the metadata that you want to export. When the business model derives from more than one database, then it eliminates metadata from all databases other than the one specified here. When the physical database is not found in the repository, an error message is displayed.
RUN_AS_USER	The user name of the user whose visibility needs to be duplicated for the metadata export. This cannot be empty. If the user is not found in the repository, then an error message is displayed.
OUTPUT_FOLDER	The full path and file name of the folder to which the SQL file will be written. If the folder does not exist when you run the Oracle Database Metadata Generator, it will be created. For more information about the output files, refer to "About the Output Files" on page 265.

The following text shows a sample metadata input file:

```
BUSINESS_MODEL= "Paint"

PHYSICAL_DATABASE   ="SQL_Paint"

RUN_AS_USER =    "Administrator"

OUTPUT_FOLDER = "C:\OracleBI"
```

About the Output Files

Each Generator creates different types of output files, as described in the following list:

- Oracle Database Metadata Generator: Generates a SQL file that is encoded in UTF8 and stored in the specified output folder. The file name includes the name of the business model in the Cube Model, such as my_business_model.sql.
- DB2 Cube Views Generator: Generates the following files in the specified output folder:

- **XML files (encoded in UTF8).** One XML file is created for each specified business model. It contains all objects that were converted to cubes. Additionally, objects in the repository will be mapped to similar objects in the IBM Cube Views metadata. For a list of objects that will not be converted, refer to "Metadata Conversion Rules and Error Messages" on page 267.

  The name of the XML file will match the business model name (without spaces), followed by the XML extension. For example, SalesResults.xml. The following is the syntax of the XML file name:

  ```
  [BusinessModelNameWithNoSpaces].xml
  ```

- **A SQL file that contains the alias generation DLL.** A SQL file is created for each specified business model only if aliases exist in the physical layer databases that are referenced in the specified business model. The alias file contains SQL commands that will create the aliases in the DB2 database. The name of the SQL file will match the business model name (without spaces), followed by the SQL extension. For example, SalesResults-alias.sql. The following is the syntax of the alias-SQL file name:

  ```
  [BusinessModelNameWithNoSpaces]-alias.sql
  ```

Troubleshooting Errors from the Generator

Error messages indicate that the Generator was unable to complete some or all of its tasks. After starting the Generator, you might observe the following error messages:

- Unable to write to Log file: <log_file_name>.

  The log file specified in the NQSConfig.INI file might contain the wrong path, the user might not have write permissions to that folder, or the disk could be out-of-space.

- Run_as_user, <user_name>, is invalid.

  The user name is incorrect.

- Repository, <repository_name>, is invalid or corrupt.

  The repository name might be incorrect, it might not exist in the given path, or the user might not have permission to read it.

- Physical Database, <database_name>, is invalid.

  The physical database name is incorrect.

- Business Model, <model_name>, is invalid.

  The business model name is correct.

- Authentication information provided is invalid.

  The specified username or password is incorrect.

- Path: "<path_name>" is invalid.

  The path or file name is incorrect.

Metadata Conversion Rules and Error Messages

When the Generator creates the output files, it also maps the metadata objects in the Oracle BI repository to similar objects in the metadata of the Oracle Database or the IBM DB2 Database.

This section explains the rules used to identify Oracle BI metadata that cannot be translated (converted) into either SQL or XML format. These rules are necessary because the Oracle Database and IBM Cube Views do not support some of the metadata constructs that are allowed by Oracle BI.

Dimensional metadata in the SQL or XML file will be generated at the logical fact table source level. If a logical fact table source has an invalid logical dimension table source, then the logical dimension table source will be invalidated. If the logical fact table source is invalid, then all the logical dimension table sources that are linked to it will also be invalidated. Invalid Oracle BI repository metadata elements will not be converted to cubes in the SQL or XML file.

When a rule is violated, the Generator writes the error messages and the metadata that violated the rule to a log file. You specify the name of this log file in the LOG_FILE_NAME parameter in the NQSConfig.INI file. For information about parameters in the Cube Views section of the NQSConfig.INI file, refer to *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

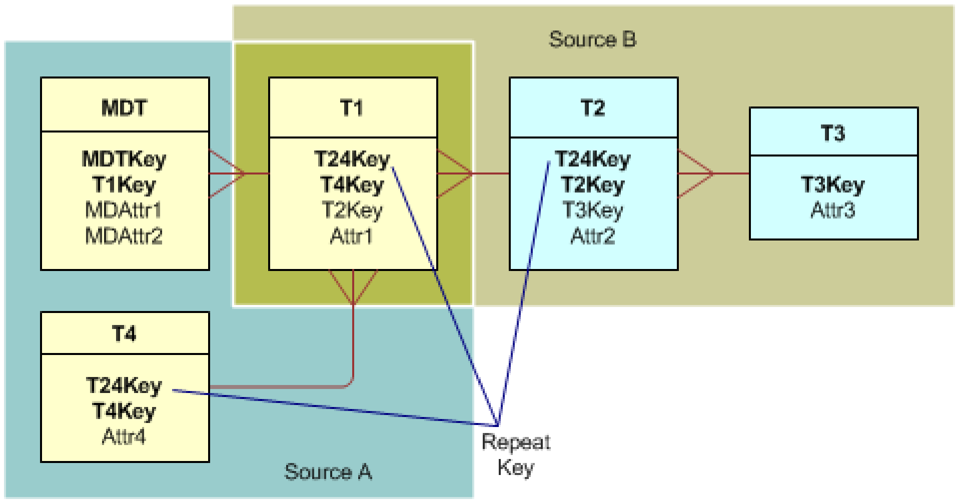
Conversion Rules for Oracle Database

The following list provides the rules for converting Oracle BI metadata into objects in the Oracle Database:

- Attributes that contain expressions in the logical table cannot be exported.
- Tables joined using complex joins are not considered.
- Tables that are opaque views are not considered.

- Columns used as part of a key in one level cannot be used as part of another level key. Consider the example that is shown in the following figure of a repository that contains a dimension called Source A and a dimension called Source B. The Source A dimension shares the T1 table with Source B. The T24Key is used in both the join to the T2 table (T24Key and T2Key) and in the join to table T4 (T24Key and T4Key).

  The Oracle Database prohibits the use of columns as keys in multiple levels. This prohibition requires the Oracle Database Metadata Generator to eliminate one of the two joins, usually the join that is encountered first. Therefore, because the T1-T4 join was encountered first, the joins to T2 and T3 are lost, which prevents the Attr2 and Attr3 attributes in tables T2 and T3 from being exported.



Conversion Rules for IBM DB2 Database

Table 32 lists the rules used to validate Oracle BI repository metadata elements, error messages that are written to the log file if the rule is violated, and an explanation of what caused the rule violation. The error messages help you determine why a particular Oracle BI metadata object was not exported to the XML file.

Table 32. Validation Rules for Metadata Elements

Rule	Message	Explanation
ComplexJoinFactsRule	[Fact Logical Table Source]Complex Physical Joins not supported %qn has a complex Join %qn between Physical Tables %qn and %qn	If the physical fact tables are connected through complex joins, the join is not supported. A complex join is defined as any join between two tables that do not have a foreign key relationship.
ComplexJoinDimsRule	[Dimension Logical Table Source]Complex Physical Joins not supported %qn has a complex Join %qn between Physical Tables %qn and %qn	If the dimension physical tables are connected through a complex join, then that join is not supported.
ComplexJoinFactDimRule	[Fact Logical Table Source -> Dimension Logical Table Source] Complex Physical Joins not supported. %qn has a complex Join %qn between Physical Tables %qn and %qn.	If a dimension physical table and a fact physical table are connected through a complex join, that join is not supported and the dimension table source is invalidated.
OpaqueViewFactRule	[Fact Logical table Source] Physical SQL Select Statements not supported. %qn uses the SQL Select Statement %qn.	When the physical fact table is generated by a SQL select statement, the logical fact table source that contains the table is invalidated. All logical dimension table sources connected to this logical fact table source are also invalidated. This construct allows subquery processing.
OpaqueViewDimRule	[Dimension Logical table Source] Physical SQL Select Statements not supported. %qn uses the SQL Select Statement %qn.	When a physical dimension table is generated by a SQL select statement, the logical dimension table source containing that table is invalidated.
OuterJoinFactRule	[Fact Logical Table Source] Physical Outer Joins not supported. %qn has an outer join %qn between physical tables %qn and %qn.	If the logical fact table source has an outer join linkage, then that logical fact table source is invalidated and all logical dimension table sources linked to this source will also be invalidated.

Table 32. Validation Rules for Metadata Elements

Rule	Message	Explanation
OuterJoinDimRule	[Dimension Logical Table Source] Physical Outer Joins not supported. %qn has an outer join %qn between physical tables %qn and %qn.	If the logical dimension table source has an outer join linkage, that logical dimension table source is invalidated.
WhereClauseFactRule	[Fact Logical Table Source] WHERE clauses are not supported. %qn has a where condition %s.	If the fact table source uses a WHERE clause to filter the data that is loaded, then this table source is invalidated.
WhereClauseDimRule	[Dimension Logical Table Source] WHERE clauses are not supported. %qn has a where condition %s.	If the dimension table source uses a WHERE clause to filter the data that is loaded, this table source is invalidated.
TwoJoinFactDimRule	[Fact Logical Table Source -> Dimension Logical Table Source] Multiple Joins between sources not supported. %qn and %qn have at least the following joins : %qn, %qn.	If a physical fact table is linked to two dimension tables from the same dimension source (if the fact table is not exclusively linked to the most detailed table in the table source), the dimension table source is invalidated.
HiddenManyManyRule	[Fact Logical Table Source -> Dimension Logical Table Source] Join between (physical or logical?) fact and dimension is not on the most detailed table. %qn between %qn and %qn is not on the most detailed table %qn {Join name, facttable, dimtable).	This is related to the TwoJoinFactDimRule. If the fact table is joined to a dimension table that is not the most detailed table in the table source, the dimension table source is invalidated.
ComplexMeasureRule	[Column] Complex Aggregation Rules not supported. %qn uses an aggregation rule of %s which is not supported.	The supported aggregations are SUM, COUNT, AVG, MIN, MAX, STDDEV, COUNT-DISTINCT, and COUNT.
CountDistMeasureRule	[Column] COUNT-DISTINCT Aggregation Rule not supported. %qn uses an aggregation rule of %s which is not supported.	COUNT-DISTINCT aggregation is not supported.
InvalidColumnLevelRule	[Level] Some columns that are part of the Primary Level Key are invalid. %qn has %qn as part of its primary key, when %qn has already been marked invalid.	COUNT-DISTINCT aggregation is not supported.

Table 32. Validation Rules for Metadata Elements

Rule	Message	Explanation
VariableBasedColumnRule	[Logical Table Source -> Column] Column uses a Variable in the Expression Column %qn uses a variable in its mapping.	COUNT-DISTINCT aggregation is not supported. The logical column uses repository and session variables in the expression.
OneFactToManyDimRule	[Fact Logical Table Source -> Dimension Logical Table Source] There must be a unique join path between the most detailed tables in the (logical or physical?) fact and the dimension. No join paths found between %qn and %qn (both physical table names). Found at least the following join paths: (%qn->%qn....), (%qn->%qn....)	Same as in TwoJoinFactDimRule or HiddenManyManyRule.
ManyMDTinFactRule	[Fact Logical Table Source] Fact Logical Table Source must have a unique most detailed table. %qn has at least the following most detailed tables : %qn,%qn.	A fact that has more than one table that is the most detailed table.
NoMeasureFactRule	[Fact Logical Table Source] Fact Logical Table Source does not have any Measures. %qn does not have any deployable measures.	A fact table does not have any measures because all the measures have been invalidated.
NoInActiveFactRule	[Fact Logical Table Source] Fact Logical Table Source is not marked Active.	A fact source is not active.
NoInActiveDimRule	[Dimension Logical Table Source] Dimension Logical Table Source is not marked Active.	A dimension source is not active.
NoAttributeInFactRule	[Fact Logical Table Source -> Column] Attribute found in Fact. %qn in a fact source %qn does not have an aggregation rule.	No attributes in the fact source.
NoMeasureInDimRule	[Dimension Logical Table Source -> Column] Measure found in Dimension. %qn in a dimension source %qn has an aggregation rule.	No measures in the dimension source.

Table 32. Validation Rules for Metadata Elements

Rule	Message	Explanation
VisibleColumnsAttrRule	[Column] -> The run_as_user does not have visibility to this Logical Column. %qn is not accessible to the run_as_user %qn due to visibility rules.	A column does not have visibility for this user.
VisibleColumnsMeasRule	[Column] -> The run_as_user does not have visibility to this Logical Column. %qn is not accessible to the run_as_user %qn due to visibility rules.	A column does not have visibility for this user.
MultiplePrimaryKeysDimRule	[Dimension Logical Table Source] A Join uses an alternate key in the Dimension Logical Table Source. %qn between %qn and %qn in %qn uses the alternate key %qn.	A dimension physical table can contain only one primary key. It is joined to another dimension physical table using a different unique key and that join is invalid. IBM Cube Views does not accept any unique keys to be used for foreign joins and always requires the primary key.
MultiplePrimaryKeysFactRule	[Dimension Logical Table Source] A Join uses an alternate key in the Dimension Logical Table Source. %qn between %qn and %qn in %qn uses the alternate key %qn.	A fact physical table can contain only one primary key. It is joined to another fact physical table using a different unique key and that join is invalid. IBM Cube Views does not accept any unique keys to be used for foreign joins and always requires the primary key.
MultiplePrimaryKeysFactDimRule	[Fact Logical Table Source -> Dim Logical Table Source] A Join uses an alternate key between the Logical Table sources. %qn between %qn and %qn for sources %qn and %qn uses the alternate key %qn.	A fact physical table can contain only one primary key. It is joined to a dimension physical table using a different unique key and is invalid. IBM Cube Views does not accept any unique keys to be used for foreign joins and always requires the primary key.
NotDB2ExpressionAttrRule	[Dimension Logical Table Source -> Column] The Column contains an Expression not supported. %qn has expression %s which is not supported.	The attribute contains an expression not supported by IBM Cube Views. This includes metadata expressions that use DateTime functions (for example, CURRENT_DATE).

Table 32. Validation Rules for Metadata Elements

Rule	Message	Explanation
NotDB2ExpressionMeasRule	[Fact Logical Table Source -> Column] The Column contains an Expression not supported. %qn has expression %s which is not supported.	A measure contains an expression not supported by IBM Cube Views. This includes metadata expressions that use DateTime functions (for example,. CURRENT_DATE).
NoAttributeDimRule	[Dimension Logical Table Source] Dimension Logical Table Source does not have any attributes visible to the run_as_user. %qn can not be queried by user %qn since none of its attributes are visible.	A dimension does not have any attributes.

Using Materialized Views in the Oracle Database with Oracle BI

This section explains how to export metadata from Oracle BI into the Oracle Database Summary Advisor and create materialized views using the Oracle Database Metadata Generator. Before reading this section, read "Exchanging Metadata with Databases" on page 263.

This section contains the following topics:

- About Using Oracle Database Summary Advisor with Materialized Views on page 273
- Process of Deploying Metadata for Oracle on page 274

About Using Oracle Database Summary Advisor with Materialized Views

This feature enhances the data warehouse performance and functionality of a database. It allows the Oracle Database Summary Advisor to store metadata about the logical relationships of the data that resides in the database. Additionally, it accelerates data warehouse queries by using more efficient Oracle materialized views. These materialized views preaggregate the relational data and improve query performance. Once the metadata is stored in the Oracle Database Summary Advisor, the database administrator can optimize the database objects and improve query performance.

When processing queries, the Oracle Database routes queries to tables that hold materialized views when possible. Because these tables of materialized views are smaller than the underlying base tables and the data has been preaggregated, the queries that are rerouted to them might run faster.

Oracle Database Metadata Generator works as a metadata bridge to convert the Oracle BI proprietary metadata into a SQL file that contains PL/SQL commands to generate dimensions in the Oracle Database Summary Advisor. After converting metadata into a SQL file, you use a tool such as

SQL*Plus to import the translated metadata into the Oracle Database Summary Advisor and store it in metadata catalog tables. After importing the metadata, you create materialized views, which are used by to optimize incoming application queries.

You can use this feature with Oracle Database 9*i* and higher. For information about platform compatibility, refer to *System Requirements and Supported Platforms*.

Process of Deploying Metadata for Oracle

NOTE: Become familiar with the Oracle Database and its tools before attempting to deploy metadata in the Oracle Database. For more information, refer to Oracle Database documentation.

Before deploying metadata, complete the steps in "Generating the Import File" on page 264. To deploy cube metadata, perform the following tasks in the order shown:

1. Executing the SQL File for Oracle on page 274
2. Defining Constraints for the Existence of Joins on page 275
3. Creating the Query Workload on page 275
4. Creating Materialized Views on page 277

Executing the SQL File for Oracle

This step is part of the "Process of Deploying Metadata for Oracle" on page 274.

Before executing the SQL file for importing into the Oracle Database Summary Advisor, ensure that you are familiar with Oracle Database import tools. Refer to Oracle Database documentation for information.

Use a tool such as SQL*Plus to execute the SQL file that Oracle Database Metadata Generator generated. You might see error messages if the dimensions already exist or if the database schema differs from that in the RPD file. When the script executes successfully, you can see the dimensions that were created by using the database web console or the Oracle Enterprise Manager Console. In the Oracle Enterprise Manager Console, expand the following nodes: Network, Databases, *database-name*, Warehouse, Summary Management, Dimensions, System.

After you execute the SQL file, you might need to perform the following actions:

- No incremental metadata changes are allowed. Schema changes require that you manually delete cube model metadata in the Oracle Database and convert the Oracle BI metadata again. For example, if you need to make a change to a dimension in a cube in the Oracle BI metadata repository, you need to delete the cube model in the Oracle Database, regenerate the SQL file from the Oracle BI repository, and import it into the Oracle Database Summary Advisor.
- You cannot delete metadata using the Oracle Database Metadata Generator. The Oracle BI Administrator must manually delete the cube model using Oracle Enterprise Manager Console.

Defining Constraints for the Existence of Joins

This step is part of the "Process of Deploying Metadata for Oracle" on page 274. For more information, refer to Oracle Database documentation.

You must ensure that the Oracle Database knows about the joins between the dimension tables and the fact tables. To do so, you create constraints in SQL*Plus or Oracle Enterprise Manager Console. In Oracle Enterprise Manager Console, you select the table on which you must create a constraint then select the Constraint tab.

You create a different type of constraint for each kind of table, as follows:

- For dimension tables, create a UNIQUE key constraint.
- For fact tables, create a FOREIGN key constraint and specify the referenced schema and referenced table. In the Constraint Definition area, include the foreign key columns in the fact table and the corresponding unique keys in the dimension table. An attempt to create a foreign key on a dimension table can fail if the foreign key column data does not match the unique key column data on the dimension table.

Creating the Query Workload

This step is part of the "Process of Deploying Metadata for Oracle" on page 274. For more information, refer to Oracle Database documentation.

A query workload is a sample set of physical queries that you want to optimize. Before you create the workload, you generate a Trace file with information on the slowest-running queries.

To generate the Trace file

You can generate the Trace file of the slowest-running queries using a tool that is appropriate to your database version, as described in the following list:

- Usage Tracking: Use this capability in Oracle BI to log queries and how long they take to run. Long running Oracle BI queries can then be executed as a script and used in conjunction with the Trace feature in the Oracle Database to capture the Oracle Database SQL code for these queries.
- Oracle Database Trace: Use this tool to identify the slowest physical query. You can enable the Trace feature either within Oracle Enterprise Manager Database Control or by entering SQL commands with the DBMS_MONITOR package. Once you enable the Trace feature, you use a script to create a Trace file to capture the SQL code for queries in a query workload table.
- Oracle Enterprise Manager: Use this tool to track slow-running queries.

TIP: The capabilities that are described in the following sections are available in the Oracle Database, rather than as part of Oracle BI.

To analyze the information in the Trace file

1. Use the following guidelines when reviewing the Trace file:
   - When you have traced many statements at once, such as in batch processes, quickly discard any statements that have acceptable query execution times. Focus on those statements that take the longest times to execute.

- Check the Query column for block visits for read consistency, including all query and subquery processing. Inefficient statements are often associated with a large number of block visits. The Current column indicates visits not related to read consistency, including segment headers and blocks that will be updated.
- Check the Disk column for the number of blocks that were read from disk. Because disk reads are slower than memory reads, the value will likely be significantly lower than the sum of the Query and Current columns. If it is not, check for issues with the buffer cache.
- Note that locking problems and inefficient PL/SQL loops can lead to high CPU time values even when the number of block visits is low.
- Watch for multiple parse calls for a single statement, because this indicates a library cache issue.

2 After identifying the problem statements in the file, check the execution plan to learn why each problem statement occurred.

To load queries into the workload

- After you use the Trace utility to learn the names of the slowest physical queries, insert them into the USER_WORKLOAD table.

  Table 33 on page 276 describes the columns of the USER_WORKLOAD table.

- Use INSERT statements to populate the QUERY column with the SQL statements for the slowest physical queries and the OWNER column with the appropriate owner names.

Table 33. Columns in USER_WORKLOAD Table

Column	Data Type	Required	Description
QUERY	Any LONG or VARCHAR type (all character types)	YES	SQL statement for the query.
OWNER	VARCHAR2 (30)	YES	User who last executed the query.
APPLICATION	VARCHAR2 (30)	NO	Application name for the query.
FREQUENCY	NUMBER	NO	Number of times that the query was executed.
LASTUSE	DATE	NO	Last date on which the query was executed.

Table 33. Columns in USER_WORKLOAD Table

Column	Data Type	Required	Description
PRIORITY	NUMBER	NO	User-supplied ranking of the query.
RESPONSETIME	NUMBER	NO	Execution time of the query in seconds.
RESULTSIZE	NUMBER	NO	Total number of bytes that the query selected.
SQL_ADDR	NUMBER	NO	Cache address of the query.
SQL_HASH	NUMBER	NO	Cache hash value of the query.

Creating Materialized Views

This step is part of the "Process of Deploying Metadata for Oracle" on page 274.

After you create the query workload table, use the appropriate tool for the Oracle Database version to create materialized views. In Oracle Database 10*g*, use the Summary Advisor on the Oracle Enterprise Manager Console and specify the query workload table that you created.

The Summary Advisor generates recommendations on improving the performance of the fact tables that you specify. The Summary Advisor displays the SQL code with which it will create the appropriate materialized views. Before indicating that the Summary Advisor should create the materialized views, review the following tips:

- The creation of a materialized view can fail if the SQL code includes a CAST statement.
- Ensure that the CREATE MATERIALIZED VIEW statement does not specify the same query that you provided as a workload table. If the statement does specify the same query, then the materialized views will likely not reflect the true performance gain. However, if the query is executed frequently, then the creation of the materialized view might still be worthwhile.
- Add a forward slash (/) to the end of the CREATE MATERIALIZED VIEW statement after the SQL statement. Otherwise, the SQL*Plus worksheet will not recognize it as a valid statement.

TIP: SQLAccess Advisor can also help determine appropriate indexing schemes.

Using IBM DB2 Cube Views with Oracle BI

This section explains how to export metadata from Oracle BI into the IBM DB2 Database using the DB2 Cube Views Generator. Before reading this section, read "Exchanging Metadata with Databases" on page 263.

This section contains the following topics:

- About Using IBM DB2 Cube Views with Oracle BI on page 278
- Process of Deploying Cube Metadata on page 278

About Using IBM DB2 Cube Views with Oracle BI

The term IBM DB2 Cube Views is a registered trademark of IBM. For information about platform compatibility, refer to *System Requirements and Supported Platforms*.

This feature enhances the data warehouse performance and functionality of a database. It allows the DB2 database to store metadata about the logical relationships of the data residing in the database. Additionally, it accelerates data warehouse queries by using more efficient DB2 materialized query tables (MQTs). These MQTs preaggregate the relational data and improve query performance.

When processing queries, the DB2 Query Rewrite functionality routes queries to the MQTs when possible. Because these tables are smaller than the underlying base tables and the data has been preaggregated, the queries that are rerouted to them might run faster.

DB2 Cube Views Generator works as a metadata bridge to convert the Oracle BI proprietary metadata into an IBM Cube Views XML file. After converting metadata into an XML file, you use IBM Cube Views to import the translated metadata into the DB2 database and store it in IBM Cube Views metadata catalog tables. After importing the metadata, you use the IBM Optimization Advisor to generate scripts to create materialized query tables (MQT) and their indexes. The deployed MQTs are used by the DB2 Query Reroute Engine to optimize incoming application queries.

NOTE: DB2 provides an API (implemented as a stored procedure) that passes XML documents as arguments to create, modify, delete, or read the metadata objects. For more information about IBM Cube Views, refer to the IBM DB2 documentation.

Process of Deploying Cube Metadata

The alias-SQL file generated by the DB2 Cube Views Generator should be executed before importing the XML file. The XML file generated by the DB2 Cube Views Generator contains the cube metadata in XML format. After importing the XML file into your DB2 database, you need to create materialized query tables.

NOTE: It is strongly recommended that you become familiar with IBM Cube Views and its tools before attempting to import the XML file. For more information, refer to IBM documentation.

Before deploying metadata, complete the steps in “Generating the Import File” on page 264. To deploy cube metadata, perform the following tasks in the order shown:

1 Executing the Alias-SQL File for IBM Cube Views on page 278

2 Importing the XML File on page 279

3 Guidelines for Creating Materialized Query Tables (MQTs) on page 280

Executing the Alias-SQL File for IBM Cube Views

This step is part of the “Process of Deploying Cube Metadata” on page 278. You must execute the alias-SQL file before you import the XML file into your DB2 database. For more information, refer to your IBM documentation.

The alias-SQL file that is generated by the DB2 Cube Views Generator needs to be executed by a SQL client on the database where the data warehouse is located. When executed, it creates aliases (synonyms) for tables in the database.

Importing the XML File

This step is part of the "Process of Deploying Cube Metadata" on page 278.

After you execute the alias-SQL. file, you can import the XML file into the database. For more information, refer to IBM documentation.

NOTE: It is strongly recommended that you become familiar with IBM Cube Views and its tools before attempting to import the XML file. For more information, refer to IBM documentation.

You can import this file using the following IBM tools:

- **IBM OLAP Center (recommended).** For more information, refer to "Guidelines for Importing the XML File Using the IBM OLAP Center" on page 279 and IBM documentation.
- **IBM command-line client utility (db2mdapiclient.exe).** IBM ships this utility with DB2. For more information about using the command-line client utility, refer to IBM documentation.
- **IBM DB2 Stored Procedure.** IBM Cube Views provides a SQL-based and XML-based application programming interface (API) that you can use to run single stored procedure to create, modify, and retrieve metadata objects. For more information, refer to IBM documentation.

Guidelines for Importing the XML File Using the IBM OLAP Center

Using the IBM OLAP Center, you can import cube metadata into the DB2 database. The IBM OLAP Center provides wizards to help you import the file. For more information, refer to IBM documentation.

To import the XML file, use the following guidelines:

- Using the IBM OLAP Center tool, connect to the DB2 database.
- In the Import Wizard, choose the XML file that you want to import.
- If metadata exists that refers to database constructs that are not in the database, then an error message is displayed.
- When the wizard asks for an import option, choose to replace existing objects.
- When you are returned to the IBM OLAP Center, a diagram of the cube model is shown.

Guidelines for Changing Cube Metadata After Importing the XML File

After you import the XML file, you might need to perform the following actions:

- Because the Oracle Business Analytics Warehouse does not store foreign keys as metadata, they will not exist in the converted metadata in the DB2 database. You need to use the IBM Referential Integrity Utility for IBM Cube Views to generate foreign key informational constraints. You can obtain this utility on the IBM Web site.
- You might encounter other issues such as foreign key join columns being nullable. You can use the following ways to solve this problem:

- If data in these columns are not null, it is recommended that you convert these columns to not-null columns.
- If data in these columns are null or you prefer not to convert the column data type even if the column data is not null, it is recommended that you modify the cube model using the following guidelines:
  - In a fact-to-dimension join, you need to manually eliminate this dimension object from the converted cube model and create a degenerated dimension object consisting of the foreign key of this join.
  - In a dimension-to-dimension join, you need to manually eliminate the dimension object that represents the primary-key side of the join from the converted cube model and create a degenerated dimension object consisting of the foreign key of this join.
  - In a fact-to-fact join, you need to manually eliminate the fact object that represents the primary-key side of the join from the converted cube model and create a degenerated dimension object consisting of the foreign key of this join.

- No incremental metadata changes will be allowed by the Cube Generator. Schema changes require that you manually delete cube model metadata in the DB2 database and convert the Oracle BI metadata again. For example, if you need to make a change to a dimension in a cube in the Oracle BI metadata repository, you need to delete the cube model in the DB2 database, regenerate the XML file from the Oracle BI repository, and import it into the DB2 database.
- You cannot delete metadata using the DB2 Cube Views Generator. The Oracle BI Administrator needs to manually delete the cube model using the IBM OLAP Center.
- The IBM Statistics tool and IBM Optimization Advisor must be run periodically.

For more information, refer to your IBM documentation.

Guidelines for Creating Materialized Query Tables (MQTs)

This step is part of the "Process of Deploying Cube Metadata" on page 278. For more information, refer to IBM documentation.

After you import the cube metadata into the database, the Oracle BI Administrator runs the IBM Optimization Advisor to generate SQL scripts and then execute those scripts to create the MQTs. The Oracle BI Administrator needs to provide certain parameters to the IBM Optimization Advisor to get optimal results from the implementation. The IBM Optimization Advisor wizard analyzes your metadata and recommends how to build summary tables that store and index aggregated data for SQL queries. Running the IBM Optimization Advisor can help you keep the MQTs current. Additionally, you must refresh your database after each ETL.

To create MQTs, use the following guidelines:

- In the IBM OLAP Center, choose the cube model that you want to optimize and open the IBM Optimization Advisor wizard.

- Follow the instructions in the wizard, using the following table as a guide.

When asked for:	Choose:
Summary Tables	Choose Deferred (or Immediate) and provide a tablespace for the tables
Limitations	Choose an appropriate value for the optimization parameters. You should turn on the Data-sampling option.
SQL Scripts	Creation of the scripts needed to run to create the Summary tables. Choose the filename and locations

- When the IBM Optimization Advisor closes, the Oracle BI Administrator must execute the SQL scripts to create the MQTs.

13 Using Variables in the Oracle BI Repository

You can use variables in a repository to streamline administrative tasks and modify metadata content dynamically to adjust to a changing data environment. The Administration Tool includes a Variable Manager for defining variables.

This section contains the following topics:

- Using the Variable Manager on page 283
- About Using Initialization Blocks With Variables on page 290
- Process of Creating Initialization Blocks on page 293

Using the Variable Manager

The Variable Manager allows you to define variables. The Variable Manager dialog box has two panes. The left pane displays a tree that shows variables and initialization blocks, and the right pane displays details of the item you select in the left pane.

There are two classes of variables: repository variables and session variables.

- A repository variable has a single value at any point in time. There are two types of repository variables: static and dynamic. Repository variables are represented by a question mark icon.
- Session variables are created and assigned a value when each user logs on. There are two types of session variables: system and nonsystem.

  System and nonsystem variables are represented by a question mark icon.

Initialization blocks are used to initialize dynamic repository variables, system session variables, and nonsystem session variables. The icon for an initialization block is a cube labeled *i*.

This section contains the following topics:

- Understanding and Creating Repository Variables on page 283
- Understanding and Creating Session Variables on page 286

Understanding and Creating Repository Variables

A repository variable has a single value at any point in time. Repository variables can be used instead of literals or constants in expression builders in the Administration Tool. The Oracle BI Server will substitute the value of the repository variable for the variable itself in the metadata.

This section includes the following topics:

- Static Repository Variables on page 284

- Dynamic Repository Variables on page 285
- Creating Repository Variables on page 285

Static Repository Variables

The value of a static repository value is initialized in the Variable dialog box. This value persists, and does not change until an Oracle BI Administrator decides to change it.

Example

Suppose you want to create an expression to group times of day into different day segments. If Prime Time were one of those segments and corresponded to the hours between 5:00 PM and 10:00 PM, you could create a CASE statement like the following:

```
CASE WHEN "Hour" >= 17 AND "Hour" < 23 THEN 'Prime Time' WHEN... ELSE...END
```

where Hour is a logical column, perhaps mapped to a timestamp physical column using the date-and-time Hour(<<timeExpr>>) function.

Rather than entering the numbers 17 and 23 into this expression as constants, you could use the Variable tab of the Variable dialog box to set up a static repository variable named prime_begin and initialize it to a value of 17, and create another variable named prime_end and initialize it to a value of 23.

Using Variables in Expression Builders

After created, variables are available for use in expression builders. In an expression builder, click on the Repository Variables folder in the left pane to display all repository variables (both static and dynamic) in the middle pane by name.

To use a repository variable in an expression, select it and double-click. The expression builder will paste it into the expression at the active cursor insertion point.

Variables should be used as arguments of the function VALUEOF( ). This will happen automatically when you double-click on the variables to paste them into the expression.

For example, the following CASE statement is identical to the one explained in the preceding example except that variables have been substituted for the constants.

```
CASE WHEN "Hour" >= VALUEOF("prime_begin")AND "Hour" < VALUEOF("prime_end") THEN
'Prime Time' WHEN ... ELSE...END
```

NOTE: You cannot use variables to represent columns or other repository objects.

Dynamic Repository Variables

You initialize dynamic repository variables in the same way as static variables, but the values are refreshed by data returned from queries. When defining a dynamic repository variable, you will create an initialization block or use a pre-existing one that contains a SQL query. You will also set up a schedule that the Oracle BI Server will follow to execute the query and periodically refresh the value of the variable.

NOTE: When the value of a dynamic repository variable changes, all cache entries associated with a business model that reference the value of that variable will be purged automatically.

Each query can refresh several variables—one variable for each column in the query. You schedule these queries to be executed by the Oracle BI Server.

Example

Dynamic repository variables are very useful for defining the content of logical table sources. For example, suppose you have two sources for information about orders. One source contains recent orders and the other source contains historical data.

You need to describe the content of these sources on the Content tab of the Logical Table Source dialog box. Without using dynamic repository variables, you would describe the content of the source containing recent data with an expression such as:

```
Orders.OrderDates."Order Date" >= TIMESTAMP '2001-06-02 00:00:00'
```

This content statement will become invalid as new data is added to the recent source and older data is moved to the historical source. To accurately reflect the new content of the recent source, you would have to modify the fragmentation content description manually. Dynamic repository values can be set up to do it automatically.

Another suggested use for dynamic repository values is in WHERE clause filters of logical table sources, that are defined on the Content tab of the Logical Table Source dialog box.

The values of dynamic repository variables are set by queries defined in Variable Initialization blocks. When defining a dynamic repository variable, you create an initialization block or use a preexisting block that contains a query. You also set up a schedule that the Oracle BI Server will follow to execute the query and periodically refresh the value of the variable.

A common use of these variables is to set filters for use in Oracle BI Presentation Services. For example, to filter a column on the value of the dynamic repository variable CurrentMonth, set the filter to the Variable CurrentMonth.

Creating Repository Variables

Use the this task to create a repository variable. For more information, see Understanding and Creating Repository Variables on page 283.

To create a repository variable

1 From the Administration Tool menu bar, choose Manage > Variables.

2 In the Variable Manager dialog box, from the menu bar, choose Action > New > Repository > Variable.

3 In the Variable dialog box, type a Variable name.

   Names for all variables should be unique. The names of system session variables are reserved and cannot be used for other types of variables.

4 In the Variables dialog box, select the type of variable: Static or Dynamic.

   The name of the dialog box changes to reflect the type of variable that you select.

5 (Dynamic repository variables) Use the Initialization Block drop-down list to select an existing initialization block that will be used to refresh the value on a continuing basis.

   To create a new initialization block, click New. For more information, refer to "Process of Creating Initialization Blocks" on page 293.

6 (Dynamic or static variables) To add a Default initializer value, perform one of the following steps:

   - To use the Expression Builder, click the ellipsis button to the right of the Default initializer work space. For more information about creating the value, refer to "SQL Logical Operators" on page 363.
   - Type the value into the Default initializer text box.

   For static repository variables, the value you specify in the Default initializer window persists. It will not change unless you change it. If you initialize a variable using a character string, enclose the string in single quotes (‘).

7 Click OK.

Understanding and Creating Session Variables

Session variables are similar to dynamic repository variables in that they obtain their values from initialization blocks. Unlike dynamic repository variables, however, the initialization of session variables is not scheduled. When a user begins a session, the Oracle BI Server creates new instances of session variables and initializes them.

Unlike a repository variable, there are as many instances of a session variable as there are active sessions on the Oracle BI Server. Each instance of a session variable could be initialized to a different value.

Session variables are primarily used when authenticating users against external sources such as database tables or LDAP servers. If a user is authenticated successfully, session variables can be used to set filters and permissions for that session. For a discussion of the use of session variables in setting up security, refer to Chapter 15, "Security in Oracle BI."

This section includes the following topics:

- Using System Session Variables on page 287
- Using Nonsystem Session Variables on page 289
- Creating Repository Variables on page 285

For more information, refer to "Creating Repository Variables" on page 285.

Using System Session Variables

System session variables are session variables that the Oracle BI Server and Oracle BI Presentation Services use for specific purposes. System session variables have reserved names, that cannot be used for other kinds of variables (such as static or dynamic repository variables and nonsystem session variables).

For information about using the GROUP system session variable in conjunction with the SA System subject area to provide group membership and external email addresses to Oracle BI Delivers, refer to "Setting Up the Repository to Work with Delivers" on page 180.

NOTE: When you use these variables for Oracle BI Presentation Services, preface their names with NQ_SESSION. For example, to filter a column on the value of the variable LOGLEVEL set the filter to the Variable NQ_SESSION.LOGLEVEL.

Table 34 on page 287 describes the available system session variables.

Table 34. System Session Variables

Variable	Description
DISPLAYNAME	Used for Oracle BI Presentation Services. It contains the name that will be displayed to the user in the greeting in the Oracle BI Presentation Services user interface. It is also saved as the author field for catalog objects. For internal Oracle BI repository users (nondatabase users), this variable is populated with the user's full name.
EMAIL	Contains the user's default email address for use with Answers. If the delivery option of Answers is enabled, an email device using this address will be created for the user upon first log in. Users can override this address by changing their account settings in Oracle BI Presentation Services.

Table 34. System Session Variables

Variable	Description
GROUP	Contains the groups to which the user belongs. These are used by both the Oracle BI Server and Oracle BI Presentation Services. When a user belongs to multiple groups, separate the group names with semicolons. Do not delimit text (for example, do not surround the text with single or double quotes). Use a Varchar column in a database table to contain the group memberships. For example, if a user belonged to groups called Sales US, Sales UK, QA and Dev, and Doc, the text entered into a Varchar data type column in a database table would be: `Sales US;Sales UK;QA and Dev;Doc` Note: The Oracle BI Presentation Services Administrator needs to make sure that the names of Presentation Services groups are different from any user IDs that will be used to log on to Oracle BI Presentation Services. If a user and a Presentation Services group share the same name, the user will receive an Invalid Account message when attempting to log on to Oracle BI Presentation Services.
LAST_SYNCH_TIME and THIS_SYNCH_TIME	These two variables are set and tracked by Oracle BI Presentation Services to manage the synchronization of Oracle BI Disconnected Analytics. For more information, refer to *Oracle Business Intelligence Disconnected Analytics Administration and Configuration Guide*.
LOGLEVEL	The value of LOGLEVEL (a number between 0 and 5) determines the Logging level that the Oracle BI Server will use for user queries. This system session variable overrides a variable defined in the Users object. If the Administrators Users object has a Logging level defined as 4 and the session variable LOGLEVEL defined in the repository has a value of 0 (zero), the value of 0 applies.
PORTALPATH	Used for Oracle BI Presentation Services. It identifies the default dashboard the user sees when logging in (the user can override this preference after logged on).
REQUESTKEY	Used for Oracle BI Presentation Services. Any users with the same nonblank request key will share the same Presentation Server cache entries. This tells Oracle BI Presentation Services that these users have identical content filters and security in the Oracle BI Server. Sharing Presentation Server cache entries is a way to minimize unnecessary communication with the Oracle BI Server.

Table 34. System Session Variables

Variable	Description
SKIN	Determines certain elements of the look and feel of the Oracle BI Presentation Services user interface. The user can alter some elements of the user interface by picking a style when logged on to Oracle BI Presentation Services. The SKIN variable points to an Oracle BI Presentation Services folder that contains the nonalterable elements (for example, graphics such as GIF files). Such directories begin with sk_. For example, if a folder were called sk_companyx, the SKIN variable would be set to companyx.
USER	Holds the value the user enters as his or her logon name.
WEBGROUPS	Specifies additional groups specific to Oracle BI Presentation Services, if any. The use of Presentation Services groups provides a mechanism for more granular content control.

Using Nonsystem Session Variables

The procedure for defining nonsystem session variables is the same as for system session variables.

A common use for nonsystem session variables is setting user filters. For example, you could define a nonsystem variable called SalesRegion that would be initialized to the name of the user's sales region.

You could then set a security filter for all members of a group that would allow them to view only data pertinent to their region.

NOTE: When you use these variables for Oracle BI Presentation Services, preface their names with NQ_SESSION. For example, to filter a column on the value of the variable SalesRegion set the filter to the Variable NQ_SESSION.SalesRegion.

Creating Session Variables

Use the this task to create a session variable.

To create a session variable

1. From the Administration Tool menu bar, choose Manage > Variables.
2. In the Variable Manager dialog box, from the menu bar, choose Action > New > Session > Variable.
3. In the Session Variable dialog box, type a variable name.

   Names for all variables should be unique. The names of system session variables are reserved and cannot be used for other types of variables.

4 For session variables, you can select the following check boxes:

Enable any user to set the value	Check box that allows you to set the session variables after the initialization block has populated the value (at user login) by calling the ODBC store procedure NQSSetSessionValue(). For example, this allows non-Oracle BI Administrators to set this variable for sampling.
Security Sensitive	Check box that identifies the variable as sensitive to security for virtual physical databases (VPD). When filtering cache table matches, the Oracle BI Server looks at the parent database of each column or table that is referenced in the logical request projection list. If the physical database source is a VPD, the Oracle BI Server matches a list of security-sensitive variables to each prospective cache hit. Cache hits would only occur on cache entries that included and matched all security-sensitive variables.

5 Use the Initialization Block drop-down list to select an initialization block that will be used to refresh the value on a continuing basis.

   To create a new initialization block, click New. For more information, refer to "Process of Creating Initialization Blocks" on page 293.

6 To add a Default initializer value, perform one of the following steps:

   - To use the Expression Builder, click the ellipsis button to the right of the Default initializer work space. For more information about creating the value, refer to "SQL Logical Operators" on page 363.
   - Type the value into the Default initializer text box.

7 Click OK.

About Using Initialization Blocks With Variables

Initialization blocks are used to initialize dynamic repository variables, system session variables, and nonsystem session variables. For example, the NQ_SYSTEM initialization block is used to refresh system session variables.

An initialization block contains the SQL that will be executed to initialize or refresh the variables associated with that block. The SQL must reference physical tables that can be accessed using the connection pool specified in the Connection Pool field in the Initialization Block dialog box.

If you want the query for an initialization block to have database-specific SQL, you can select a database type for that query. If a SQL initialization string for that database type has been defined when the initialization block is instantiated, this string will be used. Otherwise, a default initialization SQL string will be used.

CAUTION: By default, when you open the Initialization Block dialog box for editing in online mode, the initialization block object is automatically checked out. While the initialization block is checked out, the Oracle BI Server may continue to refresh the value of dynamic variables refreshed by this initialization block, depending on the refresh intervals that are set. When you check the initialization block in, the value of the dynamic variables is reset to the values shown in the Default initializer. If you do not want this to occur, use the Undo Check Out option.

Initializing Dynamic Repository Variables

The values of dynamic repository variables are set by queries defined in the Initialization string field of the Initialization Block dialog box. You also set up a schedule that the Oracle BI Server will follow to execute the query and periodically refresh the value of the variable. If you stop and restart the Oracle BI Server, the server automatically executes the SQL in repository variable initialization blocks, reinitializing the repository variables.

The Oracle BI Server logs all SQL queries issued to retrieve repository variable information in the NQQuery.log file when the Oracle BI Administrator logging level is set to 2 or higher. You should set the logging level to 2 for the Oracle BI Administrator user ID to provide the most useful level of information. The default location for the NQQuery.log file is the Log folder in the Oracle BI Server software installation folder (\OracleBI). For more information about user-level logging, refer to "Administering the Query Log" on page 214.

Initializing Session Variables

As with dynamic repository variables, session variables obtain their values from initialization blocks. Unlike dynamic repository variables, session variables are not updated at scheduled time intervals. Instead, the Oracle BI Server creates new instances of those variables whenever a user begins a new session. The values remain unchanged for the session's duration.

The Oracle BI Server logs all SQL queries issued to retrieve session variable information if Logging level is set to 2 or higher in the Security Manager User object or the LOGLEVEL system session variable is set to 2 or higher in the Variable Manager.

The default location for the NQQuery.log file is the Log folder in the Oracle BI Server software installation folder (\OracleBI). For more information about user-level logging, refer to "Administering the Query Log" on page 214.

Row-Wise Initialization

The row-wise initialization option allows you to create session variables dynamically and set their values when a session begins. The names and values of the session variables reside in an external database that you access through a connection pool. The variables receive their values from the initialization string that you type in the Initialization Block dialog box.

For example, you want to create session variables using values contained in a table named RW_SESSION_VARS. The table contains three columns: USERID, containing values that represent users' unique identifiers; NAME, containing values that represent session variable names; and VALUE, containing values that represent session variable values.

The content of the table is as follows:

USERID	NAME	VALUE
JOHN	LEVEL	4
JOHN	STATUS	FULL-TIME
JANE	LEVEL	8
JANE	STATUS	FULL-TIME
JANE	GRADE	AAA

You create an initialization block and select the Row-wise initialization check box (refer to "Process of Creating Initialization Blocks" on page 293).

For the initialization string, you type the following SQL statement:

```
select NAME, VALUE
from RW_SESSION_VARS
where USERID='VALUEOF(NQ_SESSION.USERID)'
```

NQ_SESSION.USERID has already been initialized using another initialization block.

The following session variables are created:

- When John connects to the Oracle BI Server, his session will contain two session variables from row-wise initialization: LEVEL, containing the value 4; and STATUS, containing the value FULL_TIME.
- When Jane connects to the Oracle BI Server, her session will contain three session variables from row-wise initialization: LEVEL, containing the value 8; STATUS, containing the value FULL-TIME; and GRADE, containing the value AAA.

Initializing a Variable with a List of Values

You can also use the row-wise initialization option to initialize a variable with a list of values. You can then use the SQL IN operator to test for values in a specified list.

Example: Using the table values in the previous example, you would type the following SQL statement for the initialization string:

```
select 'LIST_OF_USERS', USERID
from RW_SESSION_VARS
where NAME='STATUS' and VALUE='FULL-TIME'
```

This SQL statement populates the variable LIST_OF_USERS with a list, separated by colons, of the values JOHN and JANE; for example, JOHN:JANE. You can then use this variable in a filter, as shown in the following WHERE clause:

```
where TABLE.USER_NAME = valueof(NQ_SESSION.LIST_OF_USERS)
```

The variable LIST_OF_USERS contains a list of values, that is, one or more values. This logical WHERE clause expands into a physical IN clause, as shown in the following statement:

```
where TABLE.USER_NAME in (‘JOHN’, ‘JANE’)
```

About Authenticating Users Using Initialization Blocks

You can create a customized authentication module using initialization blocks. An *authenticator* is a DLL (or shared object on UNIX) written by a customer or developer that conforms to the Oracle BI Authenticator API Specification and can be used by Oracle BI Server to perform authentication and other tasks at run-time. The *dynamically loadable authenticator framework* (authentication module) is an Oracle BI Server module with a cache layer that uses the authenticator to perform authentication and related tasks at run-time.

Only one authenticator is allowed for each repository. The authentication for the user Administrator is always performed against the repository. Users in the repository are always authenticated against the repository.

Two sample authenticator plug-ins are installed when you install Oracle BI. One is only available for the Windows platform. The other one uses a text file for user information storage and is available to all platforms. We will provide a header file for all of the types that will be used in the dynamically loadable authenticator.

The Oracle BI Administrator asks a developer to implement a dynamically loadable authentication module according to the Oracle BI Authenticator API specification. For more information about this specification, refer to “Oracle BI Server Authentication APIs” on page 409.

After the Oracle BI Administrator creates an authentication object (authenticator plug-in) and specifies a set of parameters for the authentication module, such as configuration file path, number of cache entries, and cache expiration time. The Oracle BI Administrator then associates the authentication object with an initialization block. The Oracle BI Administrator associates the USER variable (required) and other variables with the initialization blocks.

When a user logs in, if the authentication is successful, Oracle BI Server populates a list of variables, as specified in the initialization block.

Process of Creating Initialization Blocks

It is recommended to create a dedicated connection pool for initialization blocks. For more information, refer to “Creating or Changing Connection Pools” on page 67.

For more information about initialization blocks, refer to “About Using Initialization Blocks With Variables” on page 290.

To create an initialization block, perform the following steps:

1. Assigning a Name and Schedule to Initialization Blocks on page 294
2. Selecting and Testing the Data Source and Connection Pool on page 294.
3. Associating Variables With Initialization Blocks on page 298

4 Establishing Execution Precedence on page 299

Assigning a Name and Schedule to Initialization Blocks

This task is a step in "Process of Creating Initialization Blocks" on page 293.

For repository variables, you can specify the day, date, and time for the start date and a refresh interval.

To assign a name and schedule to initialization blocks

1. From the Administration Tool menu bar, select Manage > Variables.
2. In the Variable Manager dialog box, from the Action menu, choose New > Repository (or Session) > Initialization Block.
3. In the Variable Init Block dialog box, type a name for the block. (The NQ_SYSTEM initialization block name is reserved.)
4. (Repository init blocks) In the Schedule area, select a start date and time and the refresh interval.
5. (Session init blocks) Select the following check boxes when appropriate:
   - Disabled. When selected, disables the initialization block.

     NOTE: In the Variables Manager, the right-click menu for an existing initialization block contains a Disable or Enable toggle value. This allows you to change this property without having to open the initialization block dialog box.

   - Required for authentication. Used when creating an initialization block for authenticating users.

The next step is to select the data source and connection pool.

Selecting and Testing the Data Source and Connection Pool

This task is a step in "Process of Creating Initialization Blocks" on page 293.

If you select Database as the data source type, the values returned by the database for the columns in your SQL statement will be assigned to variables that you associate with the initialization block. For session variable initialization blocks, you can select LDAP or Custom Authenticator.

If you select Database as the Data Source Type, the SQL used to refresh the variable must reference physical tables that can be accessed through the connection pool specified in the Connection Pool field. The tables do not have to be included in the physical layer of the metadata. At run time, if an initialization string for the database type has been defined, this string will be used. Otherwise, the default initialization SQL for the database type will be used. You can overtype this string.

When you create SQL and submit it directly to the database (for example when using database specific SQL in initialization blocks), the SQL bypasses Oracle BI Server. The order of the columns in the SQL statement and the order of the variables associated with the init block determine which columns are assigned to each variable.

NOTE: You should test this SQL using the Test button in the Variable Init block Data Source dialog box. If the SQL contains an error, the database will return an error message.

This following example topics contain examples of initialization strings that might be used with Delivers.

Example of an SQL Statement When Site Uses Delivers

```
select username, groupname, dbname, schemaname from users
where username=':USER'
NQS_PASSWORD_CLAUSE(and pwd=':PASSWORD')NQS_PASSWORD_CLAUSE
```

This SQL contains two constraints in the WHERE clause:

':USER' (note the colon and the single quotes) equals the ID the user types when logging in.

':PASSWORD' (again, note the colon and the single quotes) is the password the user enters. This is another system variable whose presence is always assumed when the USER system session variable is used. You do not need to set up the PASSWORD variable, and you can use this variable in a database connection pool to allow passthrough login using the user’s user ID and password. You can also use this variable in a SQL statement if you so desire.

When using external table authentication with Delivers, the portion of the SQL statement that makes up the :PASSWORD constraint needs to be embedded between NQS_PASSWORD_CLAUSE clauses.

The query will return data only if the user ID and password match values found in the specified table. You should test the SQL statement outside of the Oracle BI Server substituting valid values for the USER and PASSWORD variables and removing the NQS_PASSWORD_CLAUSE clause.

For more information, refer to “About Oracle BI Delivers and Database Authentication” on page 328.

Example of an SQL Statement When Site Does Not Use Delivers

```
select username, groupname, dbname, schemaname from users
where username=':USER'
and pwd=':PASSWORD'
```

This SQL statement contains two constraints in the WHERE clause:

':USER' (note the colon and the single quotes) is the ID the user enters when the user logged in.

':PASSWORD' (again, note the colon and the single quotes) is the password the user enters. This is another system variable whose presence is always assumed when the USER system session variable is used. You do not need to set up the PASSWORD variable, and you can use this variable in a database connection pool to allow passthrough login using the user’s user ID and password. You can also use this variable in a SQL if you so desire.

The query will return data only if the user ID and password match values found in the specified table. You should test the SQL statement outside of the Oracle BI Server, substituting valid values for the USER and PASSWORD variables.

To select a data source and connection pool for initialization blocks

1. From the Administration Tool menu bar, select Manage > Variables.
2. In the Variable Manager dialog box, double-click the variable.
3. In the Variable Initialization Block dialog box, click Edit Data Source.
4. In the Variable Initialization Block Data Source dialog box, from the Data Source Type drop-down list, select one of the following types.

Data Source Type	Description
Database	Repository and session variables.
XML	Repository and session variables.
LDAP	Session variables.
Custom Authenticator	Session variables. For more information, see "About Authenticating Users Using Initialization Blocks" on page 293.

5. If you selected Database in the Data Source Connection drop-down list, perform the following steps:

   a. Select the connection pool associated with the database where the target information is located by clicking Browse.

      CAUTION: If you do not select a connection pool before typing the initialization string, you will receive a message prompting you to select the connection pool.

   b. In the Browse dialog box, select the connection pool and click OK.

      NOTE: Select a connection pool before typing an initialization string.

      (Optional) Select the Use Database Specific SQL check box and in the Database pane, expand and select the database and its associated string.

   c. In the Initialization string text box, type the SQL initialization string needed to populate the variables.

   d. (Optional) Click Test. Tests the data source connectivity for the SQL statement.

6. If you selected XML in the Data Source Connection area, perform the following steps:

   a. Select the connection pool associated with the database where the target information is located by clicking Browse.

   b. In the Initialization string text box, type the SQL initialization string needed to populate the variables.

7. If you selected LDAP in the Data Source Connection area, perform the following steps:

   a. Click Browse to select an existing LDAP Server or click New to open the General tab of the LDAP Server dialog box and create an LDAP Server.

   b. Click OK to return to the Initialization Block dialog box.

      The LDAP server name and the associated domain identifier appear in the Name and Domain identifier columns.

8 If you selected Custom Authenticator in the Data Source Connection area, complete the fields using the following list as a guide.

Field	Description
Authenticator plug-in	Type or browse for the DLL authenticator file.
Configuration parameters	Can be used to specify a configuration file.
Cache never expires	When selected, cache never expires and has to be purged manually.
Cache persistence time	When selected, a text box and drop-down list become available, allowing you to type a number in the text box and select days, hours, minutes, or seconds as the time increment. The cache will automatically expire after this time passes.
Number of cache entries	Maximum number of cache entries.

9 Click OK.

Testing the Initialization Block

You should test the SQL using the Test button or an SQL tool such as the Oracle BI Client utility. If you use an SQL tool, be sure to use the same DSN or one set up identically to the DSN in the specified connection pool.

In Online editing mode, Initialization Block tests will not work with connection pools set to use :USER and :PASSWORD as the user name and password. In offline mode, the Set values for variables dialog box appears so that you can populate :USER and :PASSWORD.

To test the initialization block (optional)

1 From the Administration Tool menu bar, select Manage > Variables.

2 In the Variable Manager dialog box, double-click the last variable that you want to be initialized.

3 In the Variable Initialization Block dialog box, click Test.

4 In the Set value for the variables dialog box, verify the information is correct, and then click OK.

5 In the View Data from Table dialog box, type the number of rows and the starting row for your Query, and then click Query.

The Results dialog box lists the variables and their values.

The next step is to associate variables with the initialization block.

Associating Variables With Initialization Blocks

This task is a step in "Process of Creating Initialization Blocks" on page 293.

The SQL SELECT statement in the Default initializer list can contain multiple columns. The order of the columns in the SQL statement and order of the variables associated with the initialization block determine the column value that is assigned to each variable. Therefore, when you associate variables with an initialization block, the value returned in the first column will be assigned to the first variable in the list.

For more information, see "About Using Initialization Blocks With Variables" on page 290.

Repository Variables

When you open a repository in online mode, the value shown in the Default initializer field of the Initialization Block dialog box is the current value of that variable as known to the Oracle BI Server.

NOTE: The number of associated variables can be different from the number of columns being retrieved. If there are fewer variables than columns, extra column values are ignored. If there are more variables than columns, the additional variables are not refreshed (they retain their original values, whatever they may be). Any legal SQL can be executed using an initialization block, including SQL that writes to the database or alters database structures, assuming the database permits the user ID associated with the connection pool to perform these actions.

If you stop and restart the Oracle BI Server, the server automatically executes the SQL in the repository variable initialization blocks, re-initializing the repository variables.

Session Variables

For session variable initialization blocks, you can select Row-wise initialization. The Cache variables check box is automatically selected when you select the Row-wise initialization check box. Selecting the cash variables option directs the Oracle BI Server to store the results of the query in a main memory cache. For more information, refer to "Row-Wise Initialization" on page 291.

The Oracle BI Server uses the cached results for subsequent sessions. This can reduce session startup time. However, the cached results may not contain the most current session variable values. If every new session needs the most current set of session variables and their corresponding values, you clear this check box.

To associate variables with the initialization block

1 From the Administration Tool menu bar, select Manage > Variables.
2 In the Variable Manager dialog box, double-click the variable.
3 In the Variable Initialization Block dialog box, click Edit Data Target.
4 In the Variable Initialization Block Variable Target dialog box, you can select one of the following:
   - **Variables.** Associates variables with the initialization block.

- **Row-wise initialization.** Used with session init blocks only. For more information, see "Row-Wise Initialization" on page 291.

  If you select Row-wise initialization, the Use caching check box becomes available.

5. If you select the Variables option, perform one of the following steps:
   a. Click new, and in the Variable dialog box, create a new variable.

      NOTE: For the Custom Authentication data source type (Session variables), the variable USER is required.

      For information about creating variables, see "Using the Variable Manager" on page 283.

   b. Click Link, to associate an existing variable with an initialization block.
      - In the Browse dialog box, select the variable to be refreshed by this initialization block, and then click OK.
6. To reorder variables, select a variable, and then click Up or Down.
7. To remove a variable from association with this block, select the variable, and then click the remove button.
8. Click OK.

The next step is to establish execution precedence.

Establishing Execution Precedence

This task is a step in "Process of Creating Initialization Blocks" on page 293.

When a repository has more than one initialization block, you can set the order (establish the precedence) in which the blocks will be initialized.

First, you open the block that you want to be executed last and then add the initialization blocks that you want to be executed before the block you have open. For example, suppose a repository has two initialization blocks, A and B. You open initialization block B, and then specify that block A will execute before block B. This causes block A to execute according to block B's schedule, in addition to its own.

To establish execution precedence

1. From the Administration Tool menu bar, select Manage > Variables.
2. In the Variable Manager dialog box, double-click the last variable that you want to be initialized.
3. In the Variable Initialization Block dialog box, click Edit Execution Precedence.
4. In the Variable Initialization Block Execution Precedence dialog box, click Add.

   NOTE: Add is only available if unselected initialization blocks are available.

5. In the Browse dialog box, select the blocks that should be initialized before the block that you have open, and then click OK.

   CAUTION: Make sure you add the blocks in the order that you want them to be initialized.

6. To remove a block, in the Variable Initialization Block Execution Precedence dialog box, select the block you want to remove, and then click Remove.
7. Click OK.
8. If you wish the initialization block to be required, in the Variable Initialization Block dialog box, select the Required for authentication check box.
9. Click OK.

14 Clustering Oracle BI Servers

This section describes the Cluster Server and provides instructions for setting up and configuring the clustering of multiple servers.

This section contains the following topics:

About the Cluster Server

The Cluster Server allows up to 16 Oracle BI Servers in a network domain to act as a single server. Servers in the cluster share requests from multiple Oracle BI clients, including Answers and Delivers.

The Cluster Controller is the primary component of the Cluster Server. It monitors the status of resources in a cluster and performs session assignment as resources change. It also supports detection of server and Oracle Business Intelligence Scheduler failures and failover for ODBC clients of failed servers for clients of these managed services.

Components of the Cluster Server

In a clustering environment, the following components are available:

- **Two Cluster Controllers.** For more information, see "About Cluster Controllers" on page 302.
- **One or more servers.** For more information, see "About Servers Used in Clustering" on page 302.
- **One or more Schedulers.** There can only be one active Scheduler (only one Scheduler running jobs). For more information, see "About Schedulers Used in Clustering" on page 302.
- **Cluster Manager.** A utility in the Administration Tool.
- **Repository Publishing Directory.** This directory is shared by all Oracle BI Servers participating in a cluster. It holds the master copies of repositories edited in online mode. The clustered Oracle BI Servers examine this directory upon startup for any repository changes. The directory typically resides on a shared file system visible to all servers in the cluster. You must set up the following access to this publishing directory:
  - The master server must have read and write access.

- All slave servers must have read access.

In the NQSConfig.INI file, the REPOSITORY_PUBLISHING_DIRECTORY parameter specifies the location of the repository publishing directory.

About Cluster Controllers

The following are the types of Cluster Controllers with their descriptions:

- **Primary Cluster Controller.** The role of the primary Cluster Controller is to monitor the operation of the servers and Schedulers in the cluster and to assign sessions within the cluster. The primary Cluster Controller can reside on the same machine as an Oracle BI Server in the cluster or on another machine that is on the same subnet as the cluster. A machine can host one Oracle BI Server, one Cluster Controller, one Scheduler, or one of each. The primary controller also determines the active Scheduler in the cluster and notifies Scheduler instances when the active instance changes.

  In the NQClusterConfig.INI file, the parameter PRIMARY_CONTROLLER specifies the machine that hosts the primary Cluster Controller.

- **Secondary Cluster Controller.** The secondary Cluster Controller assumes the role of the primary Cluster Controller if the primary is unavailable. The secondary Cluster Controller can reside on the same machine as an Oracle BI Server in the cluster or on another machine that is on the same subnet as the cluster.

  In the NQClusterConfig.INI file, the parameter SECONDARY_CONTROLLER specifies the machine that will host the secondary Cluster Controller. It must be different from the machine that hosts the primary Cluster Controller. Specifying a secondary Cluster Controller is optional. However, if the primary Cluster Controller is unavailable and the secondary Cluster Controller has not been configured, the cluster will not operate.

  NOTE: In the NQClusterConfig.ini file, you must not use fully-qualified machine names because all servers are required to run on the same LAN. Use the syntax `machinename` not `machinename.domain`.

About Servers Used in Clustering

The following are descriptions of the types of servers used in clustering:

- **Master server.** A master server is a clustered Oracle BI Server to which the Administration Tool connects for online repository changes. In the NQClusterConfig.INI file, the parameter MASTER_SERVER specifies the Oracle BI Server that functions as the master server.

- **Slave server.** A slave server is a clustered Oracle BI Server that does not allow online repository changes. It is used in load balancing of ODBC sessions to the Oracle BI Server cluster. If the master server is ever down, the Administration Tool will connect to an available slave server, but in read-only mode.

About Schedulers Used in Clustering

The following are the types of Schedulers used in clustering with their descriptions:

- **Active Scheduler.** An active Scheduler is a clustered Oracle BI Scheduler instance which actively processes Scheduler jobs. The Cluster Controller determines the active instance at run time and notifies the Scheduler cluster of this instance.
- **Inactive Scheduler.** An inactive Scheduler is a clustered Oracle BI Scheduler instance which is not actively processing Scheduler jobs but is ready to take over in the event of an active Scheduler failure. An inactive scheduler is idle at all other times.

About the Cluster Manager

The Cluster Manager is a utility that is available in the Administration Tool when a repository is open in online mode. It allows the Oracle BI Administrator to monitor and manage the operations and activities of the cluster.

You cannot start, stop, or restart services from the Cluster Manager. Use the mechanism provided with your operating system for stopping and restarting an Oracle BI service. For Oracle BI Scheduler instances, the only option from the right-click menu is Activate.

For Oracle BI Server instances, the only options from the right-click menu are Quiesce (stop taking new sessions) or Enable (take new sessions). If an Oracle BI Server instance is running, you can select Quiesce. If it is queisced, you can select Enable.

Implementing the Cluster Server

These are the high-level steps to implement the Cluster Server:

1. Install Oracle BI components, including the Cluster Controller component.
2. Set parameters in the NQSConfig.INI file.
3. Set parameters in the NQClusterConfig.INI file.
4. Set up the Oracle BI ODBC data source for clustering.
5. Configure the Schedulers to be participants in the cluster.
6. Configure Oracle BI Presentation Services to point to the Scheduler Cluster Controllers.
7. Copy the NQClusterConfig.INI file to the Cluster Controllers and Oracle BI Server in the cluster.
8. Start the machines in the cluster.

For information about installing and configuring the Cluster Server, refer to *Oracle Business Intelligence Enterprise Edition Deployment Guide*.

Installing the Cluster Server Component

To install this component, see *Oracle Business Intelligence Enterprise Edition Deployment Guide*.

Setting Parameters in the NQSConfig.INI File

In the Server section of the NQSConfig.INI file, you need to set parameters for any Oracle BI Server that will participate in a cluster. The NQSConfig.INI file is located in the Config directory in the Oracle BI software installation folder. For more information about this file and its parameters, refer to *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

Setting Parameters in the NQClusterConfig.INI File

The NQClusterConfig.INI file contains the cluster configuration parameters. The Oracle BI Server and Scheduler read this file after it reads the NQSConfig.INI file (when CLUSTER_PARTICIPANT is set to YES in the NQSConfig.INI file). Cluster Controllers also read this file.

The NQClusterConfig.INI file is located in the Config directory in the Oracle BI installation folder. For more information about this file and its parameters, refer to *Oracle Business Intelligence Enterprise Edition Deployment Guide*.

Configuring the Oracle BI ODBC Data Source Name

All clients, including Oracle BI Presentation Services clients, need to have a clustered data source name (DSN) configured in order to communicate with a cluster. You set up a DSN by using the Oracle BI DSN Configuration wizard, described in "Connectivity and Third-Party Tools in Oracle BI Server" on page 259.

Copying the NQClusterConfig.INI File

A configured NQClusterConfig.INI file needs to reside in the Config directory of every Oracle BI Server and Cluster Controller that is to participate in the cluster.

For detailed instructions on configuring the NQClusterConfig.INI file, refer to *Oracle Business Intelligence Enterprise Edition Deployment Guide*.

Starting Cluster Controllers and Oracle BI Servers

When you are using the Administration Tool and have a repository open in online mode, you can use the Cluster Manager to monitor and manage the operations of the cluster Oracle BI Server. However, opening a repository in online mode does not automatically start a clustered Oracle BI Server or a Cluster Controller; therefore, you must start one Oracle BI Server and one Cluster Controller manually. You can then use the Cluster Manager to start additional clustered servers.

NOTE: On all platforms, you need to manually start each Oracle BI Server, Scheduler, and Cluster Controller.

To start the Oracle BI Server, Scheduler, and Cluster Controller from the Command window on Windows

- Open a Command window and type the following:

```
net start "Oracle BI SERVER"

net start "Oracle BI CLUSTER"
net start "Oracle BI SCHEDULER"
```

NOTE: You can also use a third-party tool designed for remote service manipulation.

After you start the services and Cluster Controller, use a text editor to examine the log files NQServer.log, NQScheduler.log, and NQCluster.log in the Log directories, and verify that all processes started without errors and joined the operational cluster configuration successfully. If the log files indicate errors, correct the errors and restart the servers.

To start the Oracle BI Server, Scheduler, and Cluster Controller from the command line on UNIX

1. Navigate to a command window (xterm).
2. In the Command window, type the following commands (the commands will be slightly different if using csh):

   ```
   cd INSTALLDIR/setup

   ./run-sa.sh start

   ./run-ccs.sh start

   ./run-sasch.sh start
   ```

Chronology of a Cluster Operation

This section provides an overview of the Oracle BI Cluster Server startup process.

1. As each Oracle BI Server starts, it reads its NQSConfig.INI file. If a server detects a syntax error while reading the file, it logs the error to its NQServer.log file in its Log directory. All syntax errors have to be corrected for startup to continue.
2. Each Oracle BI Server reads its NQClusterConfig.INI file and Scheduler file when CLUSTER_PARTICIPANT is set to YES in the NQSConfig.INI file. Cluster Controllers also read this file.
   - If a Oracle BI Server detects a syntax error while reading the file, it logs the error to its NQServer.log file.
   - If a Cluster Controller detects an error while reading the file, it logs the error to its NQCluster.log.
   - If a Scheduler detects an error while reading the file, it logs to NQScheduler.log.
   - If a machine is hosting both an Oracle BI Server and a Cluster Controller, messages will be written to both logs.
   - All syntax errors have to be corrected for startup to continue.
3. When an instance of Oracle BI Server or Scheduler starts up, it waits for a connection from the primary and secondary cluster controllers.

- Oracle BI Server can run without the presence of a Cluster Controller service instance. However, no clustered ODBC connection can be made to Oracle BI Server if there is no running Cluster Controller service.
- The Scheduler service will start but will remain in an inactive state until a Cluster Controller service instance comes online and notifies the Scheduler of a state change.

4 The primary and secondary Cluster Controllers begin to exchange heartbeat messages. (This step is omitted when no secondary Cluster Controller is defined.)

5 The Oracle BI Server verifies whether the repository publishing directory is available. If the repository publishing directory is not available, the action each server takes depends on the setting for the REQUIRE_PUBLISHING_DIRECTORY parameter in its NQSConfig.INI file.

- When set to YES, if the publishing directory is not available at startup or if an error is encountered while the server is reading any of the files in the directory, an error message is logged in the NQServer.log file and the server shuts down.
- When set to NO, the server joins the cluster, and a warning message is logged in the NQServer.log file. However, any online repository updates are not reflected in the server's Repository directory.

6 The primary and secondary Cluster Controllers begin to exchange heartbeat messages with each participant in the cluster.

- The connection status is logged in the log files of the appropriate clustered instance (Scheduler or Oracle BI Server). Messages are also logged in the NQCluster.log file of the Cluster Controller.
- Any participants with connection problems are not allowed to join the cluster.
- If the server defined as the MASTER_SERVER for an online repository is not available, you cannot edit the repository in online mode.

7 As each Oracle BI Server in the cluster starts, it examines the repository publishing directory for any updated repositories. This is done by comparing the date and timestamps.

NOTE: The Oracle BI Server administrator is responsible for making sure that the time-of-day clocks are synchronized across all Oracle BI Servers and Cluster Controllers.

- If a server detects a newer version of an existing repository, it copies the repository to its own Repository directory.
- A server will not detect the presence of any new repositories. A new repository must be manually propagated to all clustered servers when it is created. After that, online changes are detected at subsequent startups of each server.

8 When the Cluster Controller assigns a session to a particular Oracle BI Server, the server communicates with the back-end database using the connection defined in the Connection Pool dialog box for the database. Clustered servers do not share a common connection pool. An ODBC session maintains affinity to one Oracle BI Server session for the lifetime of that session.

9 If an Oracle BI Server determines it can satisfy all or a portion of a query from its cache file, it will do so. Clustered servers do not share a common cache. Clustered servers do not share common ad-hoc query cache. They can share cache for seeded queries, if each server instance is configured to do so.

Using the Cluster Manager

The Cluster Manager allows you to monitor, analyze, and manage the operations of a cluster. It provides status, cache, and session information about the servers and controllers that make up a cluster. It is available only when the Administration Tool is connected to a clustered DSN.

NOTE: If all Cluster Controllers or Oracle BI Servers in the cluster are currently stopped or offline, you cannot access the Cluster Manager to start them. You must manually start one Cluster Controller (generally, the primary) and one Oracle BI Server.

The Cluster Manager Graphical User Interface (GUI) has two panes: the Explorer pane on the left side and the Information pane on the right side. The Explorer pane displays hierarchical information about the servers, schedulers, and controllers that make up a cluster. The Information pane shows detailed information about an item selected in the Explorer pane.

The Cluster Manager window refreshes every minute by default. You can change the interval.

To set the refresh interval for the display

1. In the Administration Tool, open a repository in online mode.
2. Select Manage > Clusters.
3. To change this value, select Refresh > Every and choose another value from the list.
4. To refresh the display at any time, make sure the Cluster Manager is the active window and press F5, or select Refresh > Now.

   This retrieves the most current information for the cluster.

To activate an inactive Scheduler instance

1. In the Administration Tool, open a repository in online mode.
2. Select Manage > Clusters.
3. In the Cluster Manager dialog box, right-click a Scheduler instance.
4. If the Scheduler instance selected is inactive, select Activate.

   This allows the administrator to dictate the active Scheduler instance at run time.

Viewing and Managing Cluster Information

The section describes how to view status, cache, and session information about a cluster and the meaning of the information provided.

Status Information

The Status view is automatically displayed when you first open the Cluster Manager window. You can also access the Status view by selecting View > Status in the Cluster Manager window.

The categories of information displayed in the Information pane may vary depending on the server to which Administration Tool is connected. Table 35 on page 308 describes categories that may appear.

Table 35. Status Columns

Column	Description
Last Reported Time	The time the Cluster Controller or Oracle BI Server communicated with the Controlling Cluster Controller. If the server or controller is offline, this field may be blank.
Name	The name of the machine hosting the Oracle BI Server or Cluster Controller.
Role	The role of the object in the cluster: ■ **Controlling.** A Cluster Controller that is currently assigned the responsibility for control of the cluster. ■ **Primary.** The primary Cluster Controller. This role is not displayed if the primary Cluster Controller is currently the controlling Cluster Controller. ■ **Secondary.** The secondary Cluster Controller. This role is not displayed if the secondary Cluster Controller is currently the controlling Cluster Controller. ■ **Clustered server.** An Oracle BI Server that is a member of the cluster. This role is not displayed for the clustered server defined as the master server. ■ **Master.** The clustered server that the Administration Tool connects to for editing repositories in online mode. ■ **Active.** The Scheduler is active.
Sessions	This field is available when either Servers or an individual server is selected in the Explorer pane. It shows the number of sessions currently logged on to a clustered server.
Start Time	The timestamp showing when the Cluster Controller or Oracle BI Server was last started. This field will be blank if the Cluster Controller or clustered server is offline.

Table 35. Status Columns

Column	Description
Status	The status of the object in the cluster: ■ **Online.** The Cluster Controller or Oracle BI Server is online. For Cluster Controllers, this means the controller can accept session requests and assign them to available servers within the cluster. For clustered servers, this means that the server may be assigned sessions by the Cluster Controller. ■ **Quiesce.** This status is applicable to clustered servers only. This means that any activity in progress on outstanding sessions will be allowed to complete before the server transitions to Offline status. ■ **Offline.** The Cluster Controller or Oracle BI Server is offline. For Cluster Controllers, this means the controller cannot accept session requests or assign sessions to available servers within the cluster. For clustered servers, this means that the server is not communicating with the controlling Cluster Controller and cannot accept sessions assigned by the controlling Cluster Controller. If the server subsequently becomes available, it will be allowed to participate in the cluster. If you want to stop the Cluster Controller or clustered server after quiescing it, you need to issue the Stop command. ■ **Forced Offline.** This status applies to clustered servers only. The Oracle BI Server has been stopped. This is identical to the offline status, except that if the Oracle BI Server comes back online, it will not be assigned requests. The server will remain in this state until the Start command is issued against this server from the Administration Tool Cluster Manager or both Cluster Controllers are shut down and restarted. ■ **Online: Active.** The Scheduler instance is online, running, and the one to which Scheduler clients will connect. This instance will execute jobs. ■ **Online: Inactive.** The Scheduler is online but not running. This instance is ready to take over for the active instance if the active instance becomes unavailable. ■ **Online: Inactive Pending.** The Scheduler was active and is trying to go into an inactive state. This might take a few minutes, for example, if a multiple jobs are running.
Type	When Clusters is selected in the Explorer pane, this field is available. There are two types: ■ **Controller.** The object is a Cluster Controller. ■ **Server.** The object is an Oracle BI Server. ■ **Scheduler.** The object is a Scheduler Server.

Cache Information

The Cache view is available in the Cluster Manager window if caching is enabled.

The categories of information and their display sequence are controlled by your Options settings. Table 36 on page 310 describes categories that may appear.

Table 36. Cache View Columns

Column	Description
Business Model	Name of the business model associated with the cache entry.
Column count	Number of columns in each row of this cache entry's result set.
Created	Time the cache entry's result set was created.
Creation elapsed time	Time, in milliseconds, needed to create the result set for this cache entry.
Full size	Full size is the maximum size used, considering variable length columns, compression algorithm, and other factors. The actual size of the result set will be smaller than Full size.me, in seconds, needed to create the result set for this cache entry.
Last used	Last time the cache entry's result set satisfied a query. (After an unexpected shutdown of an Oracle BI Server, the Last used time may temporarily have a stale value, that is, older than the true value.)
Row count	Number of rows generated by the query.
Row size	Size of each row (in bytes) in this cache entry's result set.
SQL	Text of the SQL that generated the cache entry.
Use count	Number of times this cache entry's result set has satisfied a query (since Oracle BI Server startup).
User	ID of the user who submitted the query that resulted in the cache entry.

To view cache information

- Click an individual server in the Explorer pane, and then select View > Cache.

Session Information

The Session view is available for Oracle BI Servers. The information is arranged in two windows, described in Table 37 on page 311.

- Session window—Appears on the top. Shows users currently logged on to the Oracle BI Server.
- Request window—Appears on the bottom. Shows active query requests for the user selected in the Session window.

Table 37 on page 311 describes the information that appears in the Session window.

Table 37. Session Window Columns (Top Window)

Column	Description
Catalog	Name of the Presentation layer catalog to which the session is connected.
Client Type	Type of client session. The client type of Administration is reserved for the user logged in with the Oracle BI Administrator user ID.
Last Active Time	Timestamp of the last activity on the session or the query.
Logon Time	Timestamp when the session logged on to Oracle BI Server.
Repository	Logical name of the repository to which the session is connected.
Session ID	Unique internal identifier that the Oracle BI Server assigns each session when the session is initiated.
User	Name of the user connected.

Table 38 on page 311 describes the information that appears in the Request window.

Table 38. Request Window Columns (Bottom Window)

Column	Description
Last Active Time	Timestamp of the last activity on the session or the query.
Request ID	Unique internal identifier that the Oracle BI Server assigns each query when the query is initiated.
Session ID	Unique internal identifier that Oracle BI Server assigns each session when the session is initiated.

Table 38. Request Window Columns (Bottom Window)

Column	Description
Start Time	Time of the initial query request.
Status	These are the possible values. Due to the speed at which some processes complete, not all values for any given request or session may appear. ■ **Idle.** There is presently no activity on the request or session. ■ **Fetching.** The request is being retrieved. ■ **Fetched.** The request has been retrieved. ■ **Preparing.** The request is being prepared for processing. ■ **Prepared.** The request has been prepared for processing and is ready for execution. ■ **Executing.** The request is currently running. To kill a request, select it and click the Kill Request button. The user will receive an informational message indicating that the Oracle BI Administrator cancelled the request. ■ **Executed.** The request has finished running. ■ **Succeeded.** The request ran to completion successfully. ■ **Canceled.** The request has been canceled. ■ **Failed.** An error was encountered during the processing or running of the request.

To manage clustered servers

1. In the Explorer pane, click the plus sign (+) to the left of the Server icon to display the servers in the cluster.
2. In the Information pane, select a server.
3. Select Action, and then select one of the available options.

   When the operation finishes, the status of the clustered server will be refreshed automatically.

To view session information

- Select a server in the Explorer pane, and then select View > Sessions.

  Session information for the server is displayed in the Information pane. It shows all users logged into the server and all current query requests for each user.

To disconnect a session

- In the Session view, right-click the session in the Session window (top window) and click Disconnect.

To kill a query request

- In the Session view, right-click the request in the Request window (bottom window) and click Kill Request.

Server Information

Selecting Server info from the View menu provides information about the cluster server such as server version number.

Performance Considerations

This section describes characteristics of the Cluster Server that may influence the performance of clustered Oracle BI Servers. You should consider these points when implementing the Cluster Server.

- Sessions are assigned to an Oracle BI Server when the session is established. A session is assigned to the server with the fewest sessions. As a result, the load on Oracle BI Servers can vary and Oracle BI Servers brought online into an operational cluster may not receive work for some time.
- Because each Oracle BI Server maintains its own local query results cache, back-end databases may receive the same query from multiple Oracle BI Servers even though the result is cached. If you are using cluster aware caching, queries that are explicitly seeded through a cache seeding iBot are shared across nodes in the cluster. For more information, see *Oracle Business Intelligence Enterprise Edition Deployment Guide*.
- Because each Oracle BI Server maintains its own local query results cache, Oracle BI Servers that are brought online into an operational cluster may respond to queries more slowly while their local cache is being populated.
- Because each Oracle BI Server has an independent copy of each repository and hence its own back-end connection pools, back-end databases may experience as many as N*M connections, where N is the number of active servers in the cluster and M is the maximum sessions allowed in the connection pool of a single repository. Therefore, it may be appropriate to reduce the maximum number of sessions configured in session pools.

15 Security in Oracle BI

The Oracle BI Server provides secure access control at any level. This section contains the following topics:

- Oracle BI Security Manager on page 315
- Authentication Options on page 324
- Managing Query Execution Privileges on page 330

Oracle BI Security Manager

The Oracle BI Security Manager displays all security information for a repository. You can use the Security Manager to set up users and groups, synchronize LDAP users and groups, set access privileges for objects such as tables and columns, set filters on information, and set up a managed query environment in which you have a great deal of control over when users can access data.

NOTE: You should read this section to understand the basics about security and setting up authentication. After reading this section, refer to details about configuring security for Oracle BI applications in *Oracle Business Intelligence Applications Installation and Administration Guide*.

The Oracle BI Server and Oracle BI Presentation Services client support industry-standard security for login and password encryption. When an end user enters a login and password in the Web browser, the Oracle BI Server uses the Hyper Text Transport Protocol Secure (HTTPS) standard to send the information to a secure port on the Oracle BI Presentation Services. From the Oracle BI Presentation Services, the information is passed through ODBC to the Oracle BI Server, using Triple DES (Data Encryption Standard). This provides a high level of security (168 bit), preventing unauthorized users from accessing data or Oracle BI metadata.

At the database level, Oracle BI Administrators can implement database security and authentication. Finally, a proprietary key-based encryption provides security to prevent unauthorized users from accessing the Oracle BI metadata repository.

This section includes the following topics:

- Working with Users on page 315
- Working with Groups on page 317
- Importing Users and Groups from LDAP on page 321

Working with Users

User accounts can be defined explicitly in an Oracle BI repository or in an external source (such as a database table or an LDAP server). However user accounts are defined, users need to be authenticated by the Oracle BI Server for a session to take place.

Users defined explicitly in a repository can access business models in that repository, but they cannot span repositories.

The Oracle BI Administrator user account is created automatically when a repository is created and cannot be deleted. For more information about the Oracle BI Administrator user account, refer to "About the Oracle BI Administrator Account" on page 317.

This section includes the following topics:

- Adding a New User to a Repository on page 316
- About the Oracle BI Administrator Account on page 317

Adding a New User to a Repository

Use this procedure to add a new user to a repository.

To add a new user to a repository

1. Open a repository in the Administration Tool.
2. Display the security manager by selecting Manage > Security.
3. Select Action > New > User to open the User dialog box.
4. Type a name and password for the user.
5. If you want to log queries for this user in the query log, change the query logging level to 1 or 2.

   For more information about query logging, refer to "Setting a Logging Level" on page 215.
6. Click OK.

   This creates a new user with default rights granted to it. In the NQSConfig.INI file, the default rights are specified by the entry DEFAULT_PRIVILEGES.
7. To modify the user's permissions, open the User dialog by double-clicking on the user icon you want to modify. If you click Permissions, you can change permissions for multiple columns.
8. Specify the password expiration option.
   - If the user's password should never expire, select the option Password Never Expires.
   - If you want the user's password to expire, use the Days drop-down list to select the number of days to elapse before the user's password will expire. The maximum interval is 365 days.

     When the specified number of days passes after the password is created or changed, the password expires and must be changed.
9. You can grant rights to the user individually, through groups, or a combination of the two. To grant membership in a group, check as many groups as you want the user to be a part of in the Group Membership portion of the dialog box.

10 To specify specific database logon IDs for one or more databases, type the appropriate user IDs and passwords for the user in the Logons tab of the User dialog box.

NOTE: If a user specifies database-specific logon IDs in the DSN used to connect to the Oracle BI Server, the logon IDs in the DSN are used if the Oracle BI Administrator has configured a connection pool with no default database-specific logon ID and password. For information about configuring the connection pools to support database-specific logon IDs, refer to "Creating or Changing Connection Pools" on page 67.

11 Set up any query permissions for the user. For information, refer to "Managing Query Execution Privileges" on page 330.

About the Oracle BI Administrator Account

The Oracle BI Administrator account (user ID of Administrator) is a default user account in every Oracle BI repository. This is a permanent account. It cannot be deleted or modified other than to change the password and logging level. It is designed to perform all administrative tasks in a repository, such as importing physical schemas, creating business models, and creating users and groups.

NOTE: The Oracle BI Administrator account is not the same as the Windows NT and Windows 2000 Administrator account. The administrative privileges granted to this account function only within the Oracle BI Server environment.

When you create a new repository, the Oracle BI Administrator account is created automatically and has no password assigned to it. You should assign a password for the Oracle BI Administrator account as soon as you create the repository. The Oracle BI Administrator account created during the installation of the Oracle BI repository, that is, the repository shipped with Oracle BI, has the default password SADMIN.

The Oracle BI Administrator account belongs to the Administrators group by default and cannot be deleted from it. The person logged on using the Oracle BI Administrator user ID or any member of the Oracle BI Administrators group has permissions to change anything in the repository. Any query issued from the Oracle BI Administrator account has complete access to the data; no restrictions apply to any objects.

NOTE: You can set the minimum length for passwords in the NQSConfig.INI file using the MINIMUM_PASSWORD_LENGTH setting.

Working with Groups

The Oracle BI Server allows you to create *groups* and then grant membership in them to users or other groups.

You can think of a group as a set of security attributes. The Oracle BI Server groups are similar to groups in Windows NT and Windows 2000, and to groups or roles in database management systems (DBMS). Like Windows NT and Windows 2000, and database groups or roles, Oracle BI Server groups can allow access to objects. Additionally, Oracle BI Server groups can explicitly deny particular security attributes to its members.

Groups can simplify administration of large numbers of users. You can grant or deny sets of privileges to a group and then assign membership in that group to individual users. Any subsequent modifications to that group will affect all users who belong to it. Externally defined users can be granted group membership by use of the GROUP session variable. For more information about session variables, refer to "Using System Session Variables" on page 287.

This section includes the following topics:

- Predefined Administrators Group on page 318
- Defined Groups on page 318
- Group Inheritance on page 318
- Adding a New Group on page 320
- Viewing Member Hierarchies on page 320

Predefined Administrators Group

The Oracle BI Server has one predefined group, the Oracle BI Administrators group. Members of this group have the authority to access and modify any object in a repository. The predefined Oracle BI Administrator user ID is automatically a member of the Oracle BI Administrators group.

Use caution in granting membership in the Oracle BI Administrators group to users or other groups. Membership in the Oracle BI Administrators group supersedes all privileges granted to a user, either through groups or explicitly through the user privileges. Any user who is a member of the Oracle BI Administrators group has all of the privileges of the Oracle BI Administrator user.

Defined Groups

You can create an unlimited number of groups in an Oracle BI repository. Each group can contain explicitly granted privileges or privileges granted implicitly using membership in another group. For more information about setting up a group, refer to "Adding a New Group" on page 320.

For example, you can create one group that denies access to the repository on Mondays and Wednesdays (Group1), another group that denies access on Saturdays and Sundays (Group2), and another that denies access on Tuesdays, Thursdays, and Fridays (Group3). Users who are members of Group2 can access the system only during weekdays, users who are members of Group1 and Group3 can access the system only on weekends, and so on.

Group Inheritance

Users can have explicitly granted privileges. They can also have privileges granted through membership in groups, that in turn can have privileges granted through membership in other groups, and so on. Privileges granted explicitly to a user have precedence over privileges granted through groups, and privileges granted explicitly to the group take precedence over any privileges granted through other groups.

If there are multiple groups acting on a user or group *at the same level* with conflicting security attributes, the user or group is granted the least restrictive security attribute. Any explicit permissions acting on a user take precedence over any privileges on the same objects granted to that user through groups.

Example 1

Suppose you have a user (User1) who is explicitly granted permission to read a given table (TableA). Suppose also that User1 is a member of Group1, that explicitly denies access to TableA. The resultant privilege for User1 is to read TableA, as shown in Figure 21 on page 319.

Because privileges granted directly to the user take precedence over those granted through groups, User1 has the privilege to read TableA.

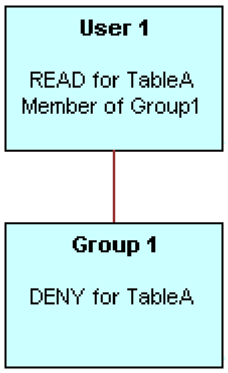


Figure 21. User Privileges and Group Privileges

Example 2

Consider the situation shown in Figure 22 on page 319.

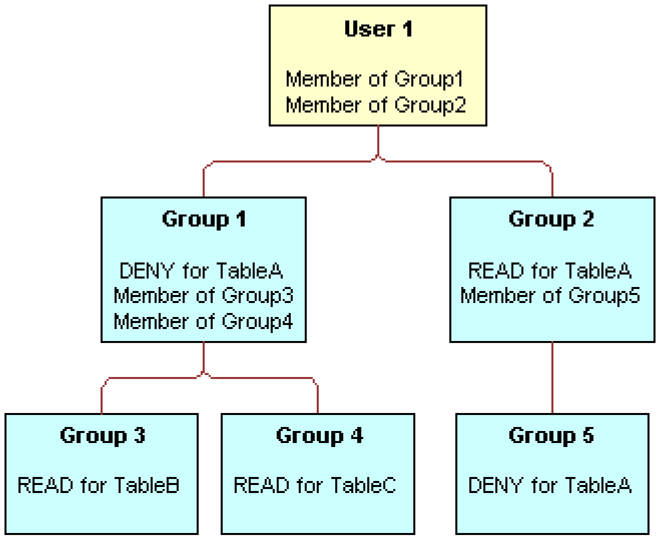


Figure 22. Privileges Example

These are the resulting privileges:

- User1 is a direct member of Group1 and Group2, and is an indirect member of Group3, Group4, and Group5.
- Because Group5 is at a lower level of precedence than Group2, its denial of access to TableA is overridden by the READ privilege granted through Group2. The result is that Group2 provides READ privilege on TableA.
- The resultant privileges from Group1 are DENY for TableA, READ for TableB, and READ for TableC.
- Because Group1 and Group2 have the same level of precedence and because the privileges in each cancel the other out (Group1 denies access to TableA, Group2 allows access to TableA), the less restrictive level is inherited by User1; that is, User1 has READ access to TableA.
- The total privileges granted to User1 are READ access for TableA, TableB, and TableC.

Adding a New Group

The following procedure explains how to add a new group to a repository.

To add a new group to a repository

1. Open a repository in the Administration Tool. (The repository can be opened in either online or offline mode.)
2. Display the security window by selecting Manage > Security.
3. Select Action > New > Group from menu.

   The Group dialog box appears.

   NOTE: You can also select the Group icon in the left pane, and then right-click on white space in the left pane and select New Security Group from the right-click menu.
4. Type a name for the group and click OK.

   This creates a new group with no rights granted to it.
5. To modify the group's permissions, open the Group dialog by double-clicking on the group icon you want to modify. If you click on Permissions, you can change permissions for multiple columns.
6. You can grant rights to the group by adding other groups, by explicit configuration for the group, or a combination of the two. To grant membership to a group, click Add and select any users or groups you want to grant membership. Click OK after you have selected the groups and users.
7. Set up any query permissions for the group. For information, refer to "Managing Query Execution Privileges" on page 330.

   NOTE: Unlike the User dialog box, the Group dialog box does not allow you to select a logging level. The logging level is a user attribute and cannot be specified for a group.

Viewing Member Hierarchies

Use the following procedures to view member hierarchies.

To view member hierarchies in the Security Manager

- Click the hierarchy icon in the left pane of the Security Manager, and then expand the tree in the right pane.

To view member hierarchies in the Query Repository dialog box

1. Select Tools > Query Repository from the main menu of the Administration Tool.
2. To view all groups, select Security Groups from the Type drop-down list and click Query.
3. To view all users, select Users from the Type drop-down and click Query.
4. To view the groups that a group is a member of, select the group and click Parent. For example, to view what groups Group1 is a member of, select Group1 and click the Parent button.

Importing Users and Groups from LDAP

If your organization uses Lightweight Directory Access Protocol (LDAP), you can import your existing LDAP users and groups to a repository. After imported, all normal Oracle BI Server user and group functions are available. You can resynchronize your imported list at any time.

You can also authenticate against LDAP as an external source. When you do this, users are not imported into the repository. Users are authenticated, and their group privileges determined, when they log on. For more information about using LDAP authentication, refer to "Setting Up LDAP Authentication" on page 324.

NOTE: If you create a variable for the same user in both the repository and in a LDAP server, the local repository user definition takes precedence and LDAP authentication will not occur. This allows the Oracle BI Administrator to reliably override users that exist in an external security system.

This section includes the following topics:

- Setting Up an LDAP Server on page 321
- Importing Users from LDAP on page 323
- Synchronizing Users and Groups with LDAP on page 323

Setting Up an LDAP Server

This section explains how to set up LDAP authentication for the repository.

NOTE: For information about the basics of security and setting up authentication, refer to "Oracle BI Security Manager" on page 315.

For instances of Oracle BI that use ADSI as the authentication method, the following options should be used when setting up the AD instance:

- In Log On To, check All Computers or, if you list some computers, include the AD server as a Logon workstation.
- The following option must not be checked:

  User must change password at next logon

In the Administration Tool, the CN user used for the BIND DN of the LDAP Server section must have both ldap_bind and ldap_search authority.

NOTE: The Oracle BI Server uses clear text passwords in LDAP authentication. Make sure your LDAP Servers are set up to allow this.

To set up LDAP authentication for the repository

1. Open a repository in the Administration Tool in offline or online mode.
2. From the application menu, choose Manage > Security.
3. From the Security Manager menu, choose Action > New > LDAP Server.
4. In the LDAP Server dialog box, in the General tab, complete the necessary fields. The following list of fields (or buttons) and descriptions contain additional information to help you set up the LDAP server:
   - **Host name.** The name of your LDAP server.
   - **Port number.** The default LDAP port is 389.
   - **LDAP version.** LDAP 2 or LDAP 3 (versions). The default is LDAP 3.
   - **Base DN.** The base distinguished name (DN) identifies the starting point of the authentication search. For example, if you want to search all of the entries under the o=`Oracle.com` subtree of the directory, o=`Oracle.com` is the base DN.
   - **Bind DN and Bind Password.** The optional DN and its associated user password that are required to bind to the LDAP server.

     If these two entries are blank, *anonymous binding* is assumed. For security reasons, not all LDAP servers allow anonymous binding.

     These fields are optional for LDAP V3, but required for LDAP V2, because LDAP V2 does not support anonymous binding.

     These fields are required if you select the ADSI check box. If you leave these fields blank, a warning message appears asking if you want to leave the password empty anyway. If you click Yes, anonymous binding is assumed.
   - **Test Connection.** Use this button to verify your parameters by testing the connection to the LDAP server.
5. Click the Advanced tab, and type the required information. The following list of fields and descriptions contain additional information to help you set up the LDAP server:

   NOTE: The Oracle BI Server maintains an authentication cache in memory that improves performance when using LDAP to authenticate large numbers of users. Disabling the authentication cache can slow performance when hundreds of sessions are being authenticated.

   - **Connection timeout.** When the Administration Tool attempts to connect to an LDAP server for import purposes or the Oracle BI Server attempts to connect to an LDAP server for user authentication, the connection will time out after the specified interval.

- **Domain identifier.** Typically, the identifier is a single word that uniquely identifies the domain for which the LDAP object is responsible. This is especially useful when you use multiple LDAP objects. If two different users have the same user ID and each is on a different LDAP server, you can designate domain identifiers to differentiate between them. The users log in to the Oracle BI Server using the following format:

  ```
  domain_id/user_id
  ```

  If a user enters a user id without the domain identifier, it will be authenticated against all available LDAP servers in turn. If there are multiple users with the same ID, only one user can be authenticated.
- **ADSI.** (Active Directory Service Interfaces) A type of LDAP server. If you select the ADSI check box, Bind DN and Bind password are required.
- **SSL.** (Single Socket Layer) Check this box to enable this.
- **User Name Attribute Type.** This uniquely identifies a user. In many cases, this is the RDN (relative distinguished name). Typically, you accept the default value. For most LDAP servers, you would use the user ID. For ADSI, use sAMAccountName.

NOTE: For information about configuring cache settings and SSL, refer to *Oracle Business Intelligence Enterprise Edition Deployment Guide*.

Importing Users from LDAP

You can import selected users or groups, or you can import all users or groups. If you have previously performed an import, you can choose to synchronize the repository with the LDAP server.

To import LDAP users and groups to a repository

1. Open a repository in the Administration Tool in offline or online mode.
2. From the application menu, choose Manage > Security.
3. In the Security Manager, select LDAP Servers in the left pane to display existing LDAP servers in the right pane. Select the LDAP server from which you want to import users or groups, and select Import... from the right-click menu. (You can also select the server and then select LDAP > Import.)

   You can choose to import selected users or groups, or you can import all users and groups. If you have previously done an import, you can choose to synchronize the repository with the LDAP server.
4. Select the users you want to import and click Import.

   You can import groups by selecting Groups from the drop down list instead of Users.

Synchronizing Users and Groups with LDAP

You can refresh the repository users and groups with the current users and groups on your LDAP server. After selecting the appropriate LDAP server, select LDAP > Synchronize (or choose Synchronize from the right-click menu).

Synchronization updates your list of repository users and groups to mirror your current LDAP users and groups. Users and groups that do not exist on your LDAP server are removed from the repository. The special user Administrator and the special group Administrators always remain in your repository and are never removed.

Properties of users already included in the repository are not changed by synchronization. If you have recycled a login name for another user, drop that name from your repository prior to synchronization. This assures that the process will import the new LDAP user definition.

NOTE: With external LDAP authentication (discussed in the next section), import and synchronization are not really necessary. The primary use for import is to make it easy to copy LDAP users as Oracle BI users for testing.

Authentication Options

Authentication is the process by which a system uses a user ID and password ti verify that a user has the necessary permissions and authorizations to log in and access data. The Oracle BI Server authenticates each connection request it receives.

The Oracle BI Server supports the following authentication types:

- Setting Up LDAP Authentication on page 324
- Setting Up External Table Authentication on page 326
- Setting Up Database Authentication on page 327
- About Oracle BI Delivers and Database Authentication on page 328
- Maintaining Oracle BI Server User Authentication on page 329

Setting Up LDAP Authentication

Instead of storing user IDs and passwords in an Oracle BI repository, you can set up the Oracle BI Server to take the user ID and password typed by a user and pass them to an LDAP server for authentication. The server uses clear text passwords in LDAP authentication. Make sure your LDAP servers are set up to allow this.

In addition to basic user authentication, the LDAP server can also provide the Oracle BI Server with other information, such as the user display name (used by Oracle BI Presentation Services) and the name of any groups to which the user belongs. The LDAP server can also provide the names of specific database catalogs or schemas to use for each user when querying data. This information is contained in LDAP variables that get passed to Oracle BI *session variables* during the process of user authentication. For more information about session variables, refer to “Understanding and Creating Session Variables” on page 286.

LDAP authentication uses Oracle BI session variables, that you define using the Variable Manager of the Administration Tool. For more information about the Variable Manager, refer to “Using the Variable Manager” on page 283.

You need to perform the following steps to set up LDAP authentication:

1 Create an LDAP Server using the Administration Tool menu path: Manage > Security. For instructions, see "Setting Up an LDAP Server" on page 321.

2 Create an LDAP initialization block and associate it with an LDAP server. Setting up an LDAP initialization block is explained in "Process of Creating Initialization Blocks" on page 293.

3 Define a system variable named USER and map the USER variable to an LDAP attribute (uid or sAMAccountName).

   Session variables get their values when a user begins a session by logging on. Certain session variables, called *system session variables*, have special uses. The variable USER is a system variable that is used with LDAP authentication. For more information about the USER system variable, refer to "Using System Session Variables" on page 287 and "Defining a USER Session Variable for LDAP Authentication" on page 325.

4 If applicable, delete users from the Oracle BI repository file.

5 Associate the USER system variable with the LDAP initialization block. For more information, see "About Authenticating Users Using Initialization Blocks" on page 293.

Defining a USER Session Variable for LDAP Authentication

To set up LDAP authentication, you define a system variable called USER and associate it with an LDAP initialization block that is associated with an LDAP server. When a user logs into the Oracle BI Server, the user ID and password will be passed to the LDAP server for authentication. After the user is authenticated successfully, other session variables for the user could also be populated from information returned by the LDAP server.

NOTE: If you create a variable for the same user in both the repository and in a LDAP server, the local repository user definition takes precedence and LDAP authentication will not occur.

The information in this section assumes that an LDAP initialization block has already been defined.

For users not defined in the repository, the presence of a defined session system variable USER determines that external authentication is performed. Associating USER with an LDAP initialization block determines that the user will be authenticated by LDAP. To provide other forms of authentication, associate the USER variable with an initialization block associated with an external database or XML source. For more information, refer to "Setting Up External Table Authentication" on page 326.

To define the USER session system variable for LDAP authentication

1 Select Manage > Variables from the Administration Tool menu.

2 Select the System leaf of the tree in the left pane.

3 Right-click on the right pane and select New USER.

4 In the Session Variable - USER dialog box, select the appropriate LDAP initialization block from the Initialization Block drop-down list.

   The selected initialization block provides the USER session system variable with its value.

5 Click OK to create the USER variable.

Setting the Logging Level

Use the system variable LOGLEVEL to set the logging level for users who are authenticated by an LDAP server. Refer to "Setting a Logging Level" on page 215 for more information.

Setting Up External Table Authentication

Instead of storing user IDs and passwords in an Oracle BI repository, you can maintain lists of users and their passwords in an external database table and use this table for authentication purposes. The external database table contains user IDs and passwords, and could contain other information, including group membership and display names used for Oracle BI Presentation Services users. The table could also contain the names of specific database catalogs or schemas to use for each user when querying data.

NOTE: If a user belongs to multiple groups, the group names should be included in the same column separated by semicolons.

External table authentication can be used in conjunction with database authentication. If external table authentication succeeds, then database authentication is not performed. If external table authentication fails, then database authentication is performed.

Refer to "Setting Up Database Authentication" on page 327, and "Order of Authentication" on page 330 for additional details.

External table authentication uses Oracle BI session variables that you define using the Variable Manager of the Administration Tool. For more information about the Variable Manager, refer to "Using the Variable Manager" on page 283.

Session variables get their values when a user begins a session by logging on. Certain session variables, called *system variables*, have special uses. The variable USER is a system variable that is used with external table authentication.

To set up external table authentication, you define a system variable called USER and associate it with an *initialization block* that is associated with an external database table. Whenever a user logs in, the user ID and password will be authenticated using SQL that queries this database table for authentication. After the user is authenticated successfully, other session variables for the user could also be populated from the results of this SQL query. For more information about session variables, refer to "Understanding and Creating Session Variables" on page 286.

The presence of a defined system variable USER determines that external authentication is done. Associating USER with an external database table initialization block determines that the user will be authenticated using the information in this table. To provide other forms of authentication, associate the USER system variable with an initialization block associated with a LDAP server or XML source. For more information, refer to "Setting Up LDAP Authentication" on page 324.

To set up external table authentication

1. Import information about the external table into the Physical layer. In this illustration, the database sql_nqsecurity contains a table named securitylogons and has a connection pool named External Table Security.
2. Select Manage > Variables to open the Variable Manager.

3. Select Initialization Blocks on the left tree pane.
4. Right-click on white space in the right pane, and then click on New Initialization Block from the right-click menu.
5. In the Initialization Block dialog box, type the name for the initialization block.
6. Select Database from the Data Source Connection drop-down list.
7. Click Browse to search for the name of the connection pool this block will use.
8. In the Initialization String area, type the SQL statement that will be issued at authentication time.

   The values returned by the database in the columns in your SQL will be assigned to variables. The order of the variables and the order of the columns will determine which columns are assigned to which variables. Consider the SQL in the following example:

   ```
   select username, grp_name, SalesRep, 2 from securitylogons where username =
   ':USER' and pwd = ':PASSWORD'
   ```

   This SQL contains two constraints in the WHERE clause:

   - :USER (note the colon) equals the ID the user entered when logging on.
   - :PASSWORD (note the colon again) equals the password the user typed.

   The query will return data only if the user ID and password match values found in the specified table.

   You should test the SQL statement outside of the Oracle BI Server, substituting valid values for :USER and :PASSWORD to verify that a row of data returns.
9. If this query returns data, the user is authenticated and session variables will be populated. Because this query returns four columns, four session variables will be populated. Create these variables (USER, GROUP, DISPLAYNAME, and LOGLEVEL) by clicking New in the dialog's Variables tab.

   If a variable is not in the desired order, click on the variable you want to reorder and use the Up and Down buttons to move it.
10. Click OK to save the initialization block.

Setting Up Database Authentication

The Oracle BI Server can authenticate users through database logons. If a user has read permission on a specified database, the user will be trusted by the Oracle BI Server. Unlike operating system authentication, this authentication can be applied to Oracle BI Presentation Services users. For information, refer to "About Oracle BI Delivers and Database Authentication" on page 328.

Database authentication can be used in conjunction with external table authentication. If external table authentication succeeds, then database authentication is not performed. If external table authentication fails, then database authentication is performed.

Refer to "Setting Up External Table Authentication" on page 326 and "Order of Authentication" on page 330 for additional details.

Database authentication requires the user ID to be stored in the Oracle BI repository.

To set up database authentication

1. Create users in the repository named identically to the users in a database. Passwords are not stored in the repository.
2. Assign the permissions (including group memberships, if any) you want the users to have.
3. Specify the authentication database in the Security section of the NQSConfig.INI file.

   For more information, see *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.
4. Create a DSN for the database.
5. Import the database into the Physical layer. You do not need to import the physical table objects. The database name in the Physical layer has to match the database name in the NQSConfig.INI file (as specified in Step 3).
6. Set up the connection pool without a shared logon.

When a user logs on to the Oracle BI Server, the server attempts to use the logon name and password to connect to the authentication database using the first connection pool associated with it. If this connection succeeds, the user is considered to be authenticated successfully.

If the logon is denied, the Oracle BI Server issues a message to the user indicating an invalid user ID or password.

About Oracle BI Delivers and Database Authentication

In Oracle BI Applications, users are always created in the operational application database, never in the Oracle BI repository. The Oracle BI repository is preconfigured for database authentication.

Oracle BI Scheduler Server runs Delivers jobs for users without accessing or storing their passwords. Using a process called impersonation, the Scheduler uses one user ID and password with Oracle BI Administrator privileges that can act on behalf of other users. The Scheduler initiates an iBot by logging on to Oracle BI Presentation Services with that Oracle BI Administrator ID and password.

For Delivers to work, all database authentication must be performed in only one connection pool, and that connection pool can only be selected in an initialization block for the USER system session variable. This is typically called the Authentication Initialization Block. When impersonation is used, this initialization block is skipped. All other initialization blocks must use connection pools that do not use database authentication.

CAUTION: Using an authentication initialization block is the only initialization block in which it is acceptable to use a connection pool in which :USER and :PASSWORD are passed to a physical database.

For other initialization blocks, SQL statements can use :USER AND :PASSWORD. However, because Oracle BI Scheduler Server does not store user passwords, the WHERE clause must be constructed as shown in the following example:

```
select username, groupname, dbname, schemaname from users
where username=':USER'
NQS_PASSWORD_CLAUSE(and pwd=':PASSWORD')NQS_PASSWORD_CLAUSE
```

NOTE: When impersonation is used, everything in the parentheses is extracted from the SQL statement at runtime.

For more information, refer to the Oracle BI Delivers examples in "Selecting and Testing the Data Source and Connection Pool" on page 294.

Maintaining Oracle BI Server User Authentication

You can maintain lists of users and their passwords in the Oracle BI repository using the Administration Tool. The Oracle BI Server will attempt to authenticate users against this list when they log on unless another authentication method has already succeeded, or database authentication has been specified in the NQSConfig.INI file.

Refer to "Order of Authentication" on page 330 for additional information.

The Oracle BI Server user IDs are stored in nonencrypted form in an Oracle BI repository and are caseinsensitive. Passwords are stored in encrypted form and are casesensitive. The Oracle BI Server user IDs can be used to access any business model in a repository provided that the users have the necessary access privileges. User IDs are valid only for the repository in which they are set up. They do not span multiple repositories.

NOTE: If you are using LDAP or external table authentication, passwords are not stored in the Oracle BI repository.

For information about configuring user authentication, refer to *Oracle Business Intelligence Enterprise Edition Deployment Guide*.

Changing Oracle BI Server User Passwords

You can change user passwords in the Administration Tool.

To change a user password

1. Select Manage > Security.
2. In the Security Manager dialog box, select Users in the left pane.
3. In the right pane, right-click the user whose password you want to change.
4. Select Properties from the shortcut menu.
5. In the User tab, type the new password.
6. In the Confirm Password text box, type the password again, and then click OK.

Order of Authentication

If the user does not type a logon name, then OS authentication is triggered, unless OS authentication is explicitly turned off in the NQSConfig.INI file. For more information, refer to *Oracle Business Intelligence Enterprise Edition Deployment Guide*. Additionally, OS authentication is not used for Oracle BI Presentation Services users.

The Oracle BI Server populates session variables using the initialization blocks in the desired order that are specified by the dependency rules defined in the initialization blocks. If the server finds the session variable USER, it performs authentication against an LDAP server or an external database table, depending on the configuration of the initialization block with which the USER variable is associated.

Oracle BI Server internal authentication (or, optionally, database authentication) occurs only after these other possibilities have been considered.

Managing Query Execution Privileges

The Oracle BI Server allows you to exercise varying degrees of control over the repository information that a user can access.

Controlling query privileges allows you to manage the query environment. You can put a high level of query controls on users, no controls, or somewhere in between. The following list contains some types of activities you may want to limit:

- Restricting query access to specific objects, including rows and columns, or time periods
  - **Objects.** If you explicitly deny access to an object that has child objects, the user will be denied access to the child objects. For example, if you explicitly deny access to a particular physical database object, you are implicitly denying access to all of the physical tables and physical columns in that catalog.

    If a user or group is granted or disallowed privileges on an object from multiple sources (for example, explicitly and through one or more groups), the privileges are used based on the order of precedence, as described in "Group Inheritance" on page 318.

    You can grant or disallow the ability to execute direct database requests for a user or group.
  - **Time periods.** If you do not select a time period, access rights remain unchanged. If you allow or disallow access explicitly in one or more groups, the user is granted the least restrictive access for the defined time periods. For example, suppose a user is explicitly allowed access all day on Mondays, but belongs to a group that is disallowed access during all hours of every day. This means that the user will have access on Mondays only.
- Controlling runaway queries by limiting queries to a specific number of rows or maximum run time
- Limit queries by setting up filters for an object

All restrictions and controls can be applied at the user level, at the group level, or a combination of the two.

To limit queries by objects for a user or group

1 From the Administration Tool menu bar, choose Manage > Security.

2 In the Security Manager dialog box, in the tree pane, select Users or Groups.

3 In the right pane, right-click the name that you want to change and select Properties.

4 In the User or Group dialog box, click Permissions.

5 In the User/Group Permissions dialog box, click the General tab and perform the following steps:

   a In the General tab, to explicitly allow or disallow access to one or more objects in the repository, click Add.

   b In the Browse dialog box, in the Name list, select the objects you want to change, and then click Select.

   c In the User/Group Permissions dialog box, assign the permissions by selecting or clearing the Read check box for each object.

   (Default is a check) If the check box contains a check, the user has read privileges on the object. If the check box contains an X, the user is disallowed read privileges on the object. If it is blank, any existing privileges (for example, through a group) on the object apply.

   For more information about assigning permissions, refer to "Setting Permissions for Repository Objects" on page 33.

6 To explicitly allow or disallow populate privilege or the ability to execute direct database requests for specific database objects, perform the following steps:

   a Click the Query Limits tab and select the database.

   b In the Populate Privilege drop-down list, select Allow or Disallow.

   NOTE: For the selected user or group, this overrides the database property Allow populate queries for all.

   c To explicitly allow or disallow the ability to execute direct database requests for specific database objects, in the Execute Direct Database Requests drop-down list, select Allow or Disallow.

   NOTE: For the selected user or group, this overrides the database property Allow direct database requests for all.

7 Click OK twice to return to the Security Manager dialog box.

To limit queries by number of rows received by a user or group

1 From the Administration Tool menu bar, choose Manage > Security.

2 In the Security Manager dialog box, in the tree pane, select Users or Groups.

3 In the right pane, right-click the name that you want to change and select Properties.

4 In the User or Group dialog box, click the Permissions tab.

5 In the User/Group Permissions dialog box, click the Query Limits tab and expand the dialog box to view all columns.

6 To specify or change the maximum number of rows each query can retrieve from a database, in the Query Limits tab, perform the following steps:

   a In the Max Rows column, type the maximum number of rows.

   b In the Status Max Rows field, select a status using Table 39 on page 333 as a guide.

7 Click OK twice to return to the Security Manager dialog box.

To limit queries by maximum run time or to time periods for a user or group

1 From the Administration Tool menu bar, choose Manage > Security.

2 In the Security Manager dialog box, in the tree pane, select Users or Groups.

3 In the right pane, right-click the name that you want to change and select Properties.

4 In the User or Group dialog box, click the Permissions tab.

5 In the User/Group Permissions dialog box, click the Query Limits tab and expand the dialog box to view all columns.

6 To specify the maximum time a query can run on a database, in the Query Limits tab, perform the following steps:

   a In the Max Time column, select the number of minutes.

   b From the Status Max Time drop-down list, select a status using Table 39 on page 333 as a guide.

7 To restrict access to a database during particular time periods, in the Restrict column, click the ellipsis button.

8 In the Restrictions dialog box, perform the following steps:

   a To select a time period, click the start time and drag to the end time.

   b To explicitly allow access, click Allow.

   c To explicitly disallow access, click Disallow.

9 Click OK twice to return to the Security Manager dialog box.

To limit queries by setting up a filter on an object for a user or group

1 From the Administration Tool menu bar, choose Manage > Security.

2 In the Security Manager dialog box, in the tree pane, select Users or Groups.

3 In the right pane, right-click the name that you want to change and select Properties.

4 In the User or Group dialog box, click Permissions.

5 In the User/Group Permissions dialog box, click the Filters tab.

6 In the Filters tab, to add an object to filter, perform the following steps:

   a Click Add.

   b In the Browse dialog box, in the Names list, locate and double-click the object on which you want to filter.

   c. Select the object and click Select.

7. In the User/Group Permissions Filters dialog box, perform the following steps:

   a. Scroll to the right to view the Business Model Filter column.

   b. Click the Business Model Filter ellipsis button for the selected object.

8. In the Expression Builder dialog box, create a logical filter, and then click OK.

9. In the User/Group Permissions Filters dialog box, from the Status drop-down list, select a status using Table 39 on page 333 as a guide.

10. Click OK twice to return to the Security Manager dialog box.

Table 39. Query Privileges Status Fields

Status	Description
Disable	■ **Status Max Rows or Status Max Time.** When selected, disables any limits set in the Max Rows or Max Time fields. ■ **Filter.** The filter is not used and no other filters applied to the object at higher levels of precedence (for example, through a group) are used.
Enable	■ **Status Max Rows or Status Max Time.** This limits the number of rows or time to the value specified. If the number of rows exceeds the Max Rows value, the query is terminated. ■ **Filter.** The filter is applied to any query that accesses the object.
Ignore	■ **Status Max Rows or Status Max Time.** Limits will be inherited from the parent group. If there is no row limit to inherit, no limit is enforced. ■ **Filter.** The filter is not in use, but any other filters applied to the object (for example, through a group) are used. If no other filters are enabled, no filtering will occur.

Assigning Populate Privilege to a User or Group

When a criteria block is cached, the Populate Stored procedure writes the Cache/Saved Result Set value to the database.

NOTE: Any Marketing user who writes a cache entry or saves a result set needs to be assigned the POPULATE privilege for the target database. All Marketing segmentation users and groups need to be assigned this privilege. Typically, all Marketing users are associated with a group and this group is granted the privilege. For more information about marketing cache, refer to the topic about setting up cache for target levels in the documentation for Oracle's Siebel Marketing application.

To assign Populate privilege to a user or group

1. From the Administration Tool menu bar, choose Manage > Security.

2. In the Security Manager dialog box, in the tree pane, select Users or Groups.

3. In the right pane, right-click the name that you want to change and select Properties.

4. In the User or Group dialog box, click Permissions.
5. In the User/Group Permissions dialog box, select the Query Limits tab.
6. In the Query Limits list, expand the dialog box to view all columns.
7. From the Populate Privilege drop-down list, select Allow or Disallow.

   NOTE: For all Marketing data warehouses, set Populate Privilege to Allow.

8. Click OK twice to return to the Security Manager dialog box.

16 Using XML as a Data Source for the Oracle BI Server

This section describes the use of the Extensible Markup Language (XML) as a data source. XML is the universal format for structured documents and data on the Web. It can also be used as a database to store structured data.

The Oracle BI Server supports various XML access modes, including access through the Oracle BI Server XML Gateway and its extension, the Data Mining Adapter; and access through an XML ODBC driver.

This section includes the following topics:

Locating the XML URL

The Oracle BI Server supports the use of XML data as a data source for the Physical layer in the repository. Depending on the method used to access XML data sources, a data source may be represented by a URL pointing to one of the following sources.

- A static XML file or HTML file that contains XML data islands on the Internet (including intranet or extranet). For example,
  tap://216.217.17.176/[DE0A48DE-1C3E-11D4-97C9-00105AA70303].XML
- Dynamic XML generated from a server site. For example,
  tap://www.aspserver.com/example.asp
- An XML file or HTML file that contains XML data islands on a local or network drive. For example,
  d:/xmldir/example.xml
  d:/htmldir/island.htm

  You can also specify a directory path for local or network XML files, or you can use the asterisk (*****) as a wildcard with the filenames. If you specify a directory path without a filename specification (like d:/xmldir), all files with the XML suffix are imported. For example,
  d:/xmldir/
  d:/xmldir/exam*.xml
  d:/htmldir/exam*.htm
  d:/htmldir/exam*.html
- An HTML file that contains tables, defined by a pair of <*table*> and </*table*> tags. The HTML file may reside on the Internet (including intranet or extranet) or on a local or network drive. Refer to "Accessing HTML Tables" on page 344 for more information.

URLs may include repository or session variables, providing support for HTTP data sources that accept user IDs and passwords embedded in the URL; for example: http://somewebserver/cgi.pl?userid=valueof(session_variable1)&password= valueof(session_variable2). (This functionality also allows the Oracle BI Administrator to create an XML data source with a location that is dynamically determined by some runtime parameters.) For more information about variables, refer to Chapter 13, "Using Variables in the Oracle BI Repository."

The Oracle BI Server also supports the use of XSL transformation files (XSLT) or XPath expressions for transforming the XML files or XML data islands in an HTML page.

XSLT is a generalized form of the Cascaded Style Sheet (CSS) for HTML documents as applied to XML documents or text fragments. XPath is a simplified version of XSLT that may be expressed in a one-line statement. For example, //xml is an XPath expression instructing the XML processor to extract all elements under the root element xml. An XSLT file can also contain an XPath expressions.

NOTE: If the Oracle BI Server needs to access any nonlocal files (network files or files on the Internet, for example), you need to run the Oracle BI Server using a valid user ID and password with sufficient network privileges to access these remote files. In Windows NT and Windows 2000, this user ID also needs to have Windows Administrator privileges on the local machine. To change the account under which the server runs, follow the steps described in "Changing the User ID in Which the Oracle BI Server Runs" on page 211.

Using the Oracle BI Server XML Gateway

Using the Oracle BI Server XML Gateway, the metadata import process flattens the XML document to a tabular form using the stem of the XML filename (that is, the filename less the suffix) as the table name and the second level element in the XML document as the row delimiter. All leaf nodes are imported as columns belonging to the table. The hierarchical access path to leaf nodes is also imported.

The Oracle BI Server XML Gateway uses the metadata information contained in an XML schema. The XML schema is contained within the XML document or is referenced within the root element of the XML document. Support is currently available for the version of XML schema defined by Microsoft and implemented in its Internet Explorer 5 family of browsers.

Where there is no schema available, all XML data is imported as text data. In building the repository, you may alter the data types of the columns in the Physical layer, overriding the data types for the corresponding columns defined in the schema. The gateway will convert the incoming data to the desired type as specified in the Physical layer. You can also map the text data type to other data types in the Business Model and Mapping layer of the Administration Tool, using the CAST operator.

At this time, the Oracle BI Server XML Gateway does not support:

- Resolution of external references contained in an XML document (other than a reference to an external XML schema, as demonstrated in the example file in the section "Oracle BI Server XML Gateway Example" on page 338).
- Element and attribute inheritance contained within the Microsoft XML schema.

- Element types of a mixed content model (such as XML elements that contain a mixture of elements and CDATA, such as <p> hello <b> Joe</b>, how are you doing?</p>).

NOTE: The Oracle BI Server XML Gateway includes a Data Mining Adapter. It allows you to access data sources by calling an executable file or DLL for each record retrieved. For more information, refer to "Using the Data Mining Adapter" on page 345.

To import XML data using the Oracle BI Server XML Gateway

1. From the Administration Tool toolbar, select File > Import.

   The Select ODBC Data Source dialog box appears.

2. Select XML from the Connection Type drop-down list.

   The Type In Uniform Resource Locator dialog box appears, with the Connection Type set to XML.

3. In the URL field, specify the XML data source URL.

   The Oracle BI Server XML Gateway supports all data sources described in the section "Locating the XML URL" on page 335.

   URLs can include repository or session variables. If you click the Browse button, the Select XML File dialog box appears, from which you can select a single file. For more information about variables, refer to Chapter 13, "Using Variables in the Oracle BI Repository."

4. Optionally, type either an Extensible Stylesheet Language Transformations (XSLT) file or XPath expression.

   Use the Browse button to browse for XSLT source files.

5. Type an optional user ID and password in the appropriate fields for connections to HTTP sites that employ the HTTP Basic Authentication security mode.

   In addition to HTTP Basic Authentication security mode, the Oracle BI Server XML Gateway also supports Secure HTTP protocol and Integrated Windows Authentication (for Windows 2000), formerly called NTLM or Windows NT Challenge/Response authentication.

6. Click OK to open the Import dialog box.

7. Select the tables and columns and check the type of metadata you want to import.

   The default setting imports all objects and all metadata.

8. Click Import to begin the import process.

9. In the Connection Pool dialog box, type a name and optional description for the connection on the General tab. Refer to "Setting Up Connection Pools" on page 65 for additional details.

10. Click the XML tab to set additional connection properties, including the URL refresh interval and the length of time to wait for a URL to load before timing out.

    Because XML data sources are typically updated frequently and in real time, the Oracle BI Server XML Gateway allows users to specify a refresh interval for these data sources.

    For more information, refer to "About the Refresh Interval for XML Data Sources" on page 257.

    The default time-out interval for queries (URL loading time-out) is 15 minutes.

11 Click OK to complete the import.

12 For additional control over the XML data sources, you can specify an XSLT file or an XPath expression for individual tables in the data sources from the Physical Table dialog box. If specified, these entries are used to overwrite corresponding XSLT or XPath entries in the Connection Pool for the respective physical tables.

Oracle BI Server XML Gateway Example

The following sample XML data document (mytest.xml) references an XML schema contained in an external file. The schema file is shown following the data document. The generated XML schema information available for import to the repository is shown at the end.

```
<?xml version="1.0"?>
<test xmlns="x-schema:mytest_sch.xml">

<row>
<p1>0</p1>
<p2 width="5">
      <p3>hi</p3>
      <p4>
         <p6>xx0</p6>
         <p7>yy0</p7>
      </p4>
      <p5>zz0</p5>
</p2>
</row>

<row>
<p1>1</p1>
<p2 width="6">
      <p3>how are you</p3>
      <p4>
         <p6>xx1</p6>
         <p7>yy1</p7>
      </p4>
      <p5>zz1</p5>
</p2>
</row>

<row>
<p1>a</p1>
<p2 width="7">
      <p3>hi</p3>
      <p4>
         <p6>xx2</p6>
         <p7>yy2</p7>
      </p4>
      <p5>zz2</p5>
</p2>
</row>
```

```
<row>
<p1>b</p1>
<p2 width="8">
      <p3>how are they</p3>
      <p4>
         <p6>xx3</p6>
         <p7>yy3</p7>
      </p4>
      <p5>zz2</p5>
</p2>
</row>
</test>
```

The corresponding schema file follows:

```
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
      xmlns:dt="urn:schemas-microsoft-com:datatypes">
      <ElementType name="test" content="eltOnly" order="many">
         <element type="row"/>
      </ElementType>
      <ElementType name="row" content="eltOnly" order="many">
      <element type="p1"/>
         <element type="p2"/>
      </ElementType>
      <ElementType name="p2" content="eltOnly" order="many">
         <AttributeType name="width" dt:type="int" />
         <attribute type="width" />
         <element type="p3"/>
         <element type="p4"/>
         <element type="p5"/>
      </ElementType>
      <ElementType name="p4" content="eltOnly" order="many">
         <element type="p6"/>
         <element type="p7"/>
      </ElementType>
      <ElementType name="p1" content="textOnly" dt:type="string"/>
      <ElementType name="p3" content="textOnly" dt:type="string"/>
      <ElementType name="p5" content="textOnly" dt:type="string"/>
      <ElementType name="p6" content="textOnly" dt:type="string"/>
      <ElementType name="p7" content="textOnly" dt:type="string"/>
</Schema>
```

The name of the table generated from the preceding XML data document (mytest.xml) would be mytest and the column names would be p1, p3, p6, p7, p5, and width.

In addition, to preserve the context in which each column occurs in the document and to distinguish between columns derived from XML elements with identical names but appearing in different contexts, a list of fully qualified column names is generated, based on the XPath proposal of the World Wide Web Consortium, as follows:

```
//test/row/p1
//test/row/p2/p3
//test/row/p2/p4/p6
```

```
//test/row/p2/p4/p7
//test/row/p2/p5
//test/row/p2@width
```

The following example is a more complex example that demonstrates the use of nested table structures in an XML document. Note that you may optionally omit references to an external schema file, in which case all elements would be treated as being of the Varchar character type.

```
===Invoice.xml===
<INVOICE>
   <CUSTOMER>
      <CUST_ID>1</CUST_ID>
      <FIRST_NAME>Nancy</FIRST_NAME>
      <LAST_NAME>Fuller</LAST_NAME>
      <ADDRESS>
         <ADD1>507 - 20th Ave. E.,</ADD1>
         <ADD2>Apt. 2A</ADD2>
         <CITY>Seattle</CITY>
         <STATE>WA</STATE>
         <ZIP>98122</ZIP>
      </ADDRESS>
      <PRODUCTS>
           <CATEGORY>
              <CATEGORY_ID>CAT1</CATEGORY_ID>
              <CATEGORY_NAME>NAME1</CATEGORY_NAME>
              <ITEMS>
                  <ITEM>
                     <ITEM_ID>1</ITEM_ID>
                     <NAME></NAME>
                     <PRICE>0.50</PRICE>
                     <QTY>2000</QTY>
                  </ITEM>
                  <ITEM>
                     <ITEM_ID>2</ITEM_ID>
                     <NAME>SPRITE</NAME>
                     <PRICE>0.30</PRICE>
                     <QTY></QTY>
                  </ITEM>
              </ITEMS>
         </CATEGORY>
           <CATEGORY>
              <CATEGORY_ID>CAT2</CATEGORY_ID>
              <CATEGORY_NAME>NAME2</CATEGORY_NAME>
              <ITEMS>
                  <ITEM>
                     <ITEM_ID>11</ITEM_ID>
                     <NAME>ACOKE</NAME>
                     <PRICE>1.50</PRICE>
                     <QTY>3000</QTY>
                  </ITEM>
                  <ITEM>
                     <ITEM_ID>12</ITEM_ID>
                     <NAME>SOME SPRITE</NAME>
                     <PRICE>3.30</PRICE>
                     <QTY>2000</QTY>
                  </ITEM>
              </ITEMS>
         </CATEGORY>
      </PRODUCTS>
   </CUSTOMER>
   <CUSTOMER>
      <CUST_ID>2</CUST_ID>
      <FIRST_NAME>Andrew</FIRST_NAME>
      <LAST_NAME>Carnegie</LAST_NAME>
      <ADDRESS>
         <ADD1>2955 Campus Dr.</ADD1>
         <ADD2>Ste. 300</ADD2>
         <CITY>San Mateo</CITY>
         <STATE>CA</STATE>
         <ZIP>94403</ZIP>
      </ADDRESS>
      <PRODUCTS>
           <CATEGORY>
              <CATEGORY_ID>CAT22</CATEGORY_ID>
              <CATEGORY_NAME>NAMEA1</CATEGORY_NAME>
              <ITEMS>
                  <ITEM>
                     <ITEM_ID>122</ITEM_ID>
                     <NAME>DDDCOKE</NAME>
                     <PRICE>11.50</PRICE>
                     <QTY>2</QTY>
                  </ITEM>
                  <ITEM>
                     <ITEM_ID>22</ITEM_ID>
                     <NAME>PSPRITE</NAME>
                     <PRICE>9.30</PRICE>
                     <QTY>1978</QTY>
                  </ITEM>
              </ITEMS>
         </CATEGORY>
           <CATEGORY>
              <CATEGORY_ID>CAT24</CATEGORY_ID>
              <CATEGORY_NAME>NAMEA2</CATEGORY_NAME>
              <ITEMS>
                  <ITEM>
                     <ITEM_ID>19</ITEM_ID>
                     <NAME>SOME COKE</NAME>
                     <PRICE>1.58</PRICE>
                     <QTY>3</QTY>
                  </ITEM>
                  <ITEM>
                     <ITEM_ID>15</ITEM_ID>
                     <NAME>DIET SPRITE</NAME>
                     <PRICE>9.30</PRICE>
                     <QTY>12000</QTY>
                  </ITEM>
              </ITEMS>
         </CATEGORY>
      </PRODUCTS>
   </CUSTOMER>
   <CUSTOMER>
      <CUST_ID>3</CUST_ID>
      <FIRST_NAME>Margaret</FIRST_NAME>
      <LAST_NAME>Leverling</LAST_NAME>
      <ADDRESS>
         <ADD1>722 Moss Bay Blvd.</ADD1>
         <ADD2> </ADD2>
         <CITY>Kirkland</CITY>
         <STATE>WA</STATE>
         <ZIP>98033</ZIP>
      </ADDRESS>
      <PRODUCTS>
           <CATEGORY>
              <CATEGORY_ID>CAT31</CATEGORY_ID>
              <CATEGORY_NAME>NAMEA3</CATEGORY_NAME>
              <ITEMS>
                  <ITEM>
                     <ITEM_ID>13</ITEM_ID>
                     <NAME>COKE33</NAME>
                     <PRICE>30.50</PRICE>
                     <QTY>20033</QTY>
                  </ITEM>
                  <ITEM>
                     <ITEM_ID>23</ITEM_ID>
                     <NAME>SPRITE33</NAME>
                     <PRICE>0.38</PRICE>
                     <QTY>20099</QTY>
                  </ITEM>
              </ITEMS>
         </CATEGORY>
           <CATEGORY>
              <CATEGORY_ID>CAT288</CATEGORY_ID>
              <CATEGORY_NAME>NAME H</CATEGORY_NAME>
              <ITEMS>
                  <ITEM>
                     <ITEM_ID>19</ITEM_ID>
                     <NAME>COLA</NAME>
                     <PRICE>1.0</PRICE>
                     <QTY>3</QTY>
                  </ITEM>
                  <ITEM>
                     <ITEM_ID>18</ITEM_ID>
                     <NAME>MY SPRITE</NAME>
                     <PRICE>8.30</PRICE>
                     <QTY>123</QTY>
                  </ITEM>
              </ITEMS>
         </CATEGORY>
      </PRODUCTS>
   </CUSTOMER>
</INVOICE>
```

The generated XML schema shown next consists of one table (INVOICE) with the following column names and their corresponding fully qualified names.

Column	Fully Qualified Name
ADD1	//INVOICE/CUSTOMER/ADDRESS/ADD1
ADD2	//INVOICE/CUSTOMER/ADDRESS/ADD2
CITY	//INVOICE/CUSTOMER/ADDRESS/CITY
STATE	//INVOICE/CUSTOMER/ADDRESS/STATE
ZIP	//INVOICE/CUSTOMER/ADDRESS/ZIP
CUST_ID	//INVOICE/CUSTOMER/CUST_ID
FIRST_NAME	//INVOICE/CUSTOMER/FIRST_NAME
LAST_NAME	//INVOICE/CUSTOMER/LAST_NAME
CATEGORY_ID	//INVOICE/CUSTOMER/PRODUCTS/CATEGORY/CATEGORY_ID
CATEGORY_NAME	//INVOICE/CUSTOMER/PRODUCTS/CATEGORY/CATEGORY_NAME
ITEM_ID	//INVOICE/CUSTOMER/PRODUCTS/CATEGORY/ITEMS/ITEM/ITEM_ID
NAME	//INVOICE/CUSTOMER/PRODUCTS/CATEGORY/ITEMS/ITEM/NAME
PRICE	//INVOICE/CUSTOMER/PRODUCTS/CATEGORY/ITEMS/ITEM/PRICE
QTY	//INVOICE/CUSTOMER/PRODUCTS/CATEGORY/ITEMS/ITEM/QTY

Only tags with values are extracted as columns. An XML query generates fully qualified tag names, to help make sure that appropriate columns are retrieved.

These are the results of a sample query against the INVOICE table.

```
select first_name, last_name, price,  qty, name from invoice
------------------------------------------------------------
FIRST_NAME   LAST_NAME         PRICE   QTY   NAME
------------------------------------------------------------
Andrew      Carnegie          1.58     3    SOME COKE
Andrew      Carnegie         11.50     2    DDDCOKE
Andrew      Carnegie          9.30   12000  DIET SPRITE
Andrew      Carnegie          9.30    1978  PSPRITE
Margar      Leverling         0.38   20099  SPRITE33
Margar      Leverling         1.0      3    COLA
Margar      Leverling        30.50   20033  COKE33
Margar      Leverling         8.30    123   MY SPRITE
Nancy       Fuller            0.30          SPRITE
Nancy       Fuller            0.50    2000
Nancy       Fuller            1.50    3000  ACOKE
Nancy       Fuller            3.30    2000  SOME SPRITE
------------------------------------------------------------
Row count: 12
```

Accessing HTML Tables

The Oracle BI Server XML Gateway also supports the use of tables in HTML files as a data source. The HTML file may be identified as a URL pointing to a file on the internet (including intranet or extranet) or as a file on a local or network drive.

Even though tables, defined by the <*table*> and </*table*> tag pair, are native constructs of the HTML 4.0 specification, they are often used by Web designers as a general formatting device to achieve specific visual effects rather than as a data structure. The Oracle BI Server XML Gateway is currently the most effective in extracting tables that include specific column headers, defined by <*th*> and </*th*> tag pairs.

For tables that do not contain specific column headers, the Oracle BI Server XML Gateway employs some simple heuristics to make a best effort to determine the portions of an HTML file that appear to be genuine data tables.

The following is a sample HTML file with one table.

```
<html>
   <body>
      <table border=1 cellpadding=2 cellspacing=0>
         <tr>
            <th colspan=1>Transaction</th>
            <th colspan=2>Measurements</th>
         </tr>
         <tr>
            <th>Quality</th>
            <th>Count</th>
            <th>Percent</th>
         </tr>
         <tr>
            <td>Failed</td>
            <td>66,672</td>
            <td>4.1%</td>
         </tr>
         <tr>
            <td>Poor</td>
            <td>126,304</td>
            <td>7.7%</td>
         </tr>
         <tr>
            <td>Warning</td>
            <td>355,728</td>
            <td>21.6%</td>
         </tr>
         <tr>
            <td>OK</td>
            <td>1,095,056</td>
            <td>66.6%</td>
         </tr>
         <tr>
            <td colspan=1>Grand Total</td>
            <td>1,643,760</td>
            <td>100.0%</td>
         </tr>
      </table>
   </body>
</html>
```

The table name is derived from the HTML filename, and the column names are formed by concatenating the headings (defined by the <*th*> and </*th*> tag pairs) for the corresponding columns, separated by an underscore.

Assuming that our sample file is named *18.htm*, the table name would be 18_0 (because it is the first table in that HTML file), with the following column names and their corresponding fully qualified names.

Column	Fully Qualified Name
Transaction_Quality	\\18_0\Transaction_Quality
Measurements_Count	\\18_0\Measurements_Count
Measurements_Percent	\\18_0\Measurements_Percent

If the table column headings appear in more than one row, the column names are formed by concatenating the corresponding field contents of those header rows.

For tables without any heading tag pairs, the Oracle BI Server XML Gateway assumes the field values (as delimited by the <*td*> and </*td*> tag pairs) in the first row to be the column names. The columns are named by the order in which they appear (c0, c1, and so on).

For additional examples of XML, refer to "XML Examples" on page 350.

Using the Data Mining Adapter

The Data Mining Adapter is an extension of the Oracle BI Server XML Gateway. It allows you to selectively access external data sources by calling an executable file or DLL API for each record retrieved.

The Data Mining Adapter can only be used for a table in a logical join with another table acting as the driving table. The table with the Data Mining Adapter receives parameterized queries from a driving table through some logical joins. The table with the Data Mining Adapter is not a table that physically exists in a back-end database. Instead, the adapter uses the column values in the WHERE clauses of the parameterized queries as its input column parameters, and generates values for those columns (the output columns) not in the WHERE clauses. For information about how to set up the logical joins, refer to "Specifying a Driving Table" on page 140.

The Data Mining Adapter operates in the following ways:

- **Calls a DLL file.** The Data Mining Adapter allows you to specify a DLL, a shared object, or a shared library that implements the Data Mining Adapter API. At run time, the adapter loads the DLL and calls the API that retrieves records one row at a time. The query results are returned to the XML gateway through an API parameter.

- **Calls an executable file.** The Data Mining Adapter allows you to specify an executable file. At run time, the adapter executes the file and retrieves records from it one row at a time. You also specify the delimiters that demarcate the column values in the output file.

You specify one executable file or DLL for each table.

Using a DLL File to call the Data Mining Adapter API

The API currently consists of only one function. It takes in the values of the input columns in the parameterized queries, plus the meta information of both the input and the output columns. On return, the API places the values of the output columns in the outputColumnValueBuffer. All buffers are allocated by the caller.

Refer to IterativeGatewayDll.h for the definition of the datatype and structures used in this API. You can find this file at the following path:

[*installation root*]\Sample\TestExternalGatewayDll\IterativeGatewayDll.h

Table 40 on page 346 provides a description of the API elements.

Table 40. API Elements

Element	Description
inputColumnCount	The number of input columns.
inputColumnValueBuffer	A buffer of bytes containing the value of the input columns. The actual size of each column value is specified in the columnWidth field of the OracleBIColumnMetaInfo. The column values are placed in the buffer in the order in which the columns appear in the pInputColumnMetaInfoArray.
modelId	An optional argument that you can specify in the Search Utility field in the XML tab of the Physical Table dialog box.
OutputColumnCount	The number of output columns.
outputColumnValueBuffer	A buffer of bytes containing the value of the output columns. The actual size of each column value is specified in the columnWidth field of the OracleBIColumnMetaInfo. The column values must be placed in the buffer in the order in which the columns appear in the pOutputColumnMetaInfoArray.
pInputColumnMetaInfoArray	An array of meta information for the input columns. OracleBIColumnMetaInfo is declared in the public header file IterativeGatewayDll.h (installed with Oracle BI).
pOutputColumnMetaInfoArray	An array of meta column information for the output column. OracleBIColumnMetaInfo is declared in the public header file IterativeGatewayDll.h (installed with Oracle BI). The caller of the API provides the column name, and the callee sets the data type of the column (currently only VarCharData is supported) and the size of the column value.

Sample Implementation

A sample implementation of the Data Mining Adapter API is provided for all supported platforms in the Sample subdirectory of the Oracle BI installation folder (\OracleBI). The following files are included in the example:

- hpacc.mak (a HPUX make file for building the sample)
- IterativeGatewayDll.h (a header file to be included in your DLL)
- ReadMe.txt (a text file that describes the Data Mining Adapter API)
- StdAfx.cpp (a Windows-specific file)
- StdAfx.h (a Windows-specific header file)
- sunpro.mak (a Solaris make file for building the sample)
- TestExternalGatewayDll.cpp (the sample implementation of the DLL)
- TestExternalGatewayDll.dsp (a Microsoft Visual C++ project file for building the sample)
- TestLibraryUnix.cpp (a test drive that load up the DLL on the UNIX platforms)
- xlC50.mak (an AIX make file for building the sample)

Using ValueOf() Expressions

You can use ValueOf() expressions in the command line arguments to pass any additional parameters to the executable file or DLL API.

The following example shows how to pass a user ID and password to an executable file:

```
executable_name valueof(USERID) valueof(PASSWORD)
```

Specifying Column Values (Executable File)

When you specify an executable file, you can pass in the column values to the executable file by bracketing the column names with the $() marker.

For example, suppose there is a table containing the columns Car_Loan, Credit, Demand, Score, and Probability. The values of the input columns Car_Loan, Credit, and Demand come from other tables through join relationships. The values of the output columns Score and Probability are to be returned by the executable file. The command line would look like the following:

```
executable_name $(Car_Loan) $(Credit) $(Demand)
```

Each time the executable file is called, it returns one row of column values. The column values are output in a single-line demarcated by the delimiter that you specify.

By default, the executable is expected to output to the stdout. Alternatively, you can direct the Data Mining Adapter to read the output from a temporary output file passed to the executable as an argument by specifying a placeholder, $(NQ_OUT_TEMP_FILE) to which the executable outputs the result line. When the Data Mining Adapter invokes the executable, the placeholder $(NQ_OUT_TEMP_FILE) is substituted by a temporary filename generated at runtime. This is demonstrated in the following example:

```
executable_name $(Car_Loan) $(Credit) $(Demand)  $(NQ_OUT_TEMP_FILE)
```

The values of the columns that are not inputs to the executable file will be output first, in the unsorted order in which they appear in the physical table. In the preceding example, the value of the Score column will be followed by the value of the Probability column.

If the executable file outputs more column values than the number of noninput columns, the Data Mining Adapter will attempt to read the column values according to the unsorted column order of the physical table. If these are in conflict with the values of the corresponding input columns, the values returned from the executable file will be used to override the input columns.

The data length of each column in the delimited query output must not exceed the size specified for that column in the physical table.

Configuring the Data Mining Adapter

Use this procedure to configure the Data Mining Adapter.

To configure the Data Mining Adapter

1 In the Administration Tool, create a database, select XML Server as the database type, and then click OK.

  For information about creating a database, refer to "Creating a Database Object Manually in the Physical Layer" on page 61.

2 Configure the connection pool.

  NOTE: Do not type information into any field in the XML tab of the Connection Pool dialog box. The empty fields indicate to the Oracle BI Server that the Data Mining Adapter functionality will be invoked.

  a Right-click the database you created in Step 1, and then select New Object > Connection Pool.

  b In the General tab, type a name for the connection pool.

    The call interface defaults to XML.

  c Type a data source name, and then click OK.

3 Right-click the database you created in Step 1, and then select New Object > Table.

4 In the Physical Table dialog box, click the XML tab.

5 In the XML tab, complete one of the following tasks:

  - Select Executable, type the path to the executable file in the Search Utility field, and specify the delimiter for the output values.
  - Select DLL and type the path to the DLL in the Search Utility field.

    To include spaces in the path, enclose the path in quotation marks. For example:

```
“C:\Program Files\OracleBI\Bin\SADataMining.dll”
```

All characters appearing after the DLL path are passed to the API as a modelid string. You can use the modelid string to pass static or dynamic parameters to the DLL through the API. For example:

```
“C:\Program Files\OracleBI\Bin\SADataMining.dll” VALUEOF(Model1) VALUEOF(Model2)
```

Using XML ODBC

Using the XML ODBC database type, you can access XML data sources through an ODBC interface. The data types of the XML elements representing physical columns in physical tables are derived from the data types of the XML elements as defined in the XML schema. In the absence of a proper XML schema, the default data type of string is used. Data Type settings in the Physical layer will not override those defined in the XML data sources. When accessing XML data without XML schema, use the CAST operator to perform data type conversions in the Business Model and Mapping layer of the Administration Tool.

To import XML data using ODBC

1. To access XML data sources through ODBC, you need to license and install an XML ODBC driver.
2. Next, create ODBC DSNs that point to the XML data sources you want to access, making sure you select the XML ODBC database type.
3. From the File menu, choose Import > from Database.
4. Follow the instructions in the dialog boxes to import the ODBC DSNs into the repository.

   CAUTION: Make sure you select the Synonyms option in the Import dialog box.

XML ODBC Example

This is an example of an XML ODBC data source in the Microsoft ADO persisted file format. Note that both the data and the schema could be contained inside the same document.

```
<xml xmlns:s='uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882'
   xmlns:dt='uuid:C2F41010-65B3-11d1-A29F-00AA00C14882'
   xmlns:rs='urn:schemas-microsoft-com:rowset'
   xmlns:z='#RowsetSchema'>
<s:Schema id='RowsetSchema'>
   <s:ElementType name='row' content='eltOnly' rs:CommandTimeout='30'
rs:updatable='true'>
      <s:AttributeType name='ShipperID' rs:number='1' rs:writeunknown='true'
rs:basecatalog='Northwind' rs:basetable='Shippers'
          rs:basecolumn='ShipperID'>
         <s:datatype dt:type='i2' dt:maxLength='2' rs:precision='5'
rs:fixedlength='true' rs:maybenull='false'/>
      </s:AttributeType>
      <s:AttributeType name='CompanyName' rs:number='2' rs:writeunknown='true'
rs:basecatalog='Northwind' rs:basetable='Shippers'
```

```
          rs:basecolumn='CompanyName'>
         <s:datatype dt:type='string' rs:dbtype='str' dt:maxLength='40'
rs:maybenull='false'/>
      </s:AttributeType>
      <s:AttributeType name='Phone' rs:number='3' rs:nullable='true'
rs:writeunknown='true' rs:basecatalog='Northwind'
          rs:basetable='Shippers' rs:basecolumn='Phone'>
         <s:datatype dt:type='string' rs:dbtype='str' dt:maxLength='24'
rs:fixedlength='true'/>
      </s:AttributeType>
      <s:extends type='rs:rowbase'/>
   </s:ElementType>
</s:Schema>

<rs:data>
   <z:row ShipperID='1' CompanyName='Speedy Express' Phone='(503) 555-9831          '/>
   <z:row ShipperID='2' CompanyName='United Package' Phone='(503) 555-3199          '/>
   <z:row ShipperID='3' CompanyName='Federal Shipping' Phone='(503) 555-9931        '/>
</rs:data>
</xml>
```

XML Examples

The following XML documents provide examples of several different situations and explain how the Oracle BI Server XML access method handles those situations.

- The XML documents *83.xml* and *8_sch.xml* demonstrate the use of the same element declarations in different scope. For example, <p3> could appear within <p2> as well as within <p4>.

  Because the element <p3> in the preceding examples appears in two different scopes, each element is given a distinct column name by appending an index number to the second occurrence of the element during the Import process. In this case, the second occurrence becomes p3_1. If <p3> occurs in additional contexts, they become p3_2, p3_3.

- XML documents *83.xml* and *84.xml* demonstrate that multiple XML files can share the same schema (*8_sch.xml*).

- Internet Explorer version 5 and higher supports HTML documents containing embedded XML fragments called XML islands.

  The XML document island2.htm demonstrates a simple situation where multiple XML data islands, and therefore multiple tables, could be generated from one document. One table is generated for each instance of an XML island. Tables are distinguished by appending an appropriate index to the document name. For *island2.htm*, the two XML tables generated would be island2_0 and island2_1.

83.xml

```
===83.xml===
<?xml version="1.0"?>
<test xmlns="x-schema:8_sch.xml">|
<row>
<p1>0</p1>
<p2 width="5" height="2">
   <p3>hi</p3>
   <p4>
      <p3>hi</p3>
      <p6>xx0</p6>
      <p7>yy0</p7>
   </p4>
   <p5>zz0</p5>
</p2>
</row>


<row>
<p1>1</p1>
<p2 width="6" height="3">
   <p3>how are you</p3>
   <p4>
      <p3>hi</p3>
      <p6>xx1</p6>
      <p7>yy1</p7>
   </p4>
   <p5>zz1</p5>
</p2>
</row>
</test>
```

8_sch.xml

```
===8_sch.xml===

<Schema xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-
com:datatypes">
         <AttributeType name="height" dt:type="int" />
   <ElementType name="test" content="eltOnly" order="many">
      <AttributeType name="height" dt:type="int" />
      <element type="row"/>
   </ElementType>
   <ElementType name="row" content="eltOnly" order="many">
         <element type="p1"/>
      <element type="p2"/>
   </ElementType>
   <ElementType name="p2" content="eltOnly" order="many">
         <AttributeType name="width" dt:type="int" />
      <AttributeType name="height" dt:type="int" />
         <attribute type="width" />
      <attribute type="height" />
      <element type="p3"/>
      <element type="p4"/>
      <element type="p5"/>
   </ElementType>
   <ElementType name="p4" content="eltOnly" order="many">
      <element type="p3"/>
      <element type="p6"/>
      <element type="p7"/>
   </ElementType>
   <ElementType name="test0" content="eltOnly" order="many">
      <element type="row"/>
   </ElementType>
      <ElementType name="p1" content="textOnly" dt:type="string"/>
      <ElementType name="p3" content="textOnly" dt:type="string"/>
      <ElementType name="p5" content="textOnly" dt:type="string"/>
      <ElementType name="p6" content="textOnly" dt:type="string"/>
      <ElementType name="p7" content="textOnly" dt:type="string"/>
</Schema>
```

84.xml

```
===84.xml===

<?xml version="1.0"?>
<test0 xmlns="x-schema:8_sch.xml">
<row>
<p1>0</p1>
<p2 width="5" height="2">
   <p3>hi</p3>
   <p4>
      <p3>hi</p3>
      <p6>xx0</p6>
      <p7>yy0</p7>
   </p4>
   <p5>zz0</p5>
</p2>
</row>


<row>
<p1>1</p1>
<p2 width="6" height="3">
   <p3>how are you</p3>
   <p4>
      <p3>hi</p3>
      <p6>xx1</p6>
      <p7>yy1</p7>
   </p4>
   <p5>zz1</p5>
</p2>
</row>
</test0>
```

Island2.htm

```
===island2.htm===

<HTML>
   <HEAD>
<TITLE>HTML Document with Data Island</TITLE>
</HEAD>
   <BODY>
<p>This is an example of an XML data island in I.E. 5</p>
   <XML ID="12345">
   test>
      <row>
         <field1>00</field1>
         <field2>01</field2>
   </row>
      <row>
         <field1>10</field1>
         <field2>11</field2>
   </row>
      <row>
         <field1>20</field1>
         <field2>21</field2>
      </row>
   </test>
</XML>
<p>End of first example.</p>
<XML ID="12346">
   <test>
      <row>
         <field11>00</field11>
         <field12>01</field12>
      </row>
      <row>
         <field11>10</field11>
         <field12>11</field12>
      </row>
      <row>
         <field11>20</field11>
         <field12>21</field12>
      </row>
   </test>
</XML>
<p>End of second example.</p>
</BODY>
</HTML>
```

17 Oracle BI Server SQL Reference

The Oracle BI Server accepts SQL SELECT statements from client tools. Additionally, the Administration Tool allows you to define logical tables with complex expressions. This section explains the syntax and semantics for the SELECT statement and for the expressions you can use in the Administration Tool to define logical tables.

SQL Syntax and Semantics

This section explains the syntax and semantics for the SELECT statement. The following topics are included:

- SELECT Query Specification Syntax on page 355
- SELECT Usage Notes on page 356
- SELECT List Syntax on page 357
- Rules for Queries with Aggregate Functions on page 358

NOTE: The syntax descriptions throughout this guide are not comprehensive. They cover only basic syntax and features unique to the Oracle BI Server. For a more comprehensive description of SQL syntax, refer to a third-party reference book on SQL or to a reference manual on SQL from your database vendors.

SELECT Query Specification Syntax

The SELECT statement is the basis for querying any structured query language (SQL) database. The Oracle BI Server accepts logical requests to query objects in a repository, and users (or query tools) make those logical requests with ordinary SQL SELECT statements. The server then translates the logical requests into physical queries against one or more data sources, combines the results to match the logical request, and returns the answer to the end user.

The SELECT statement, or *query specification* as it is sometimes referred to, is the way to query a decision support system through the Oracle BI Server. A SELECT statement returns a table to the client that matches the query. It is a table in the sense that the results are in the form of rows and columns.

The following is the basic syntax for the SELECT statement. The individual clauses are defined in the subsections that follow.

```
SELECT [DISTINCT] select_list

FROM from_clause

[WHERE search_condition]
```

```
[GROUP BY column {, column}
     [HAVING search_condition]]
[ORDER BY column {, column}]
```

where:

select_list	The list of columns specified in the query. Refer to "SELECT List Syntax" on page 357.
from_clause	The list of tables in the query, or a catalog folder name. Optionally includes certain join information for the query. Refer to "FROM Clause Syntax" on page 357.
search_condition	Specifies any combination of conditions to form a conditional test. Refer to "WHERE Clause Syntax" on page 357.
column	A column (or alias) belonging to a table defined in the data source.

SELECT Usage Notes

The Oracle BI Server treats the SELECT statement as a logical request. If aggregated data is requested in the SELECT statement, a GROUP BY clause is automatically assumed by the server. Any join conditions supplied in the query are ignored; the join conditions are all predefined in the repository.

The Oracle BI Server accepts the following SQL syntaxes for comments:

- /* */ C-style comments
- // Double slash for single-line comments
- # Number sign for single-line comments

The Oracle BI Server supports certain subqueries and UNION, UNION ALL, INTERSECT, and EXCEPT operations in logical requests. This functionality increases the range of business questions that can be answered, eases the formulation of queries, and provides some ability to query across multiple business models.

The Oracle BI Server supports the following subquery predicates in any conditional expression (for example, within WHERE, HAVING, or CASE statements):

- IN, NOT IN
- >Any, >=Any, =Any, <Any, <=Any, <>Any
- >Some, >=Some, =Some, <Some, <=Some, <>Some
- >All, >=All, =All, <All, <=All, <>All
- EXISTS, NOT EXISTS

SELECT List Syntax

The SELECT list syntax lists the columns in the query.

Syntax:

```
SELECT [DISTINCT] select_list
```

where:

select_list	The list of columns specified in the query. All columns need to be from a single business model. Table names can be included (as *Table.Column*), but are optional unless column names are not unique within a business model. If column names contain spaces, enclose column names in double quotes. The DISTINCT keyword does not need to be included as the Oracle BI Server will always do a distinct query. Columns that are being aggregated do not need to include the aggregation function (such as SUM), as aggregation rules are known to the server and aggregation will be performed automatically.

FROM Clause Syntax

The Oracle BI Server accepts any valid SQL FROM clause syntax. You can specify the name of a catalog folder instead of a list of tables to simplify FROM clause creation. The Oracle BI Server determines the proper tables and the proper join specifications based on the columns the query asks for and the configuration of the Oracle BI repository.

WHERE Clause Syntax

The Oracle BI Server accepts any valid SQL WHERE clause syntax. There is no need to specify any join conditions in the WHERE clause because the joins are all configured within the Oracle BI repository. Any join conditions specified in the WHERE clause are ignored.

The Oracle BI Server also supports the following subquery predicates in any conditional expression (WHERE, HAVING or CASE statements):

- IN, NOT IN
- >Any, >=Any, =Any, <Any, <=Any. <>Any
- >All, >=All, =All, <All, <=All, <>All
- EXISTS, NOT EXISTS

GROUP BY Clause Syntax

With auto aggregation on the Oracle BI Server, there is no need to submit a GROUP BY clause. When no GROUP BY clause is specified, the GROUP BY specification defaults to all of the nonaggregation columns in the SELECT list. If you explicitly use aggregation functions in the select list, you can specify a GROUP BY clause with different columns and the Oracle BI Server will compute the results based on the level specified in the GROUP BY clause. For additional details, and some examples of using the GROUP BY clause in queries against the Oracle BI Server, refer to "Rules for Queries with Aggregate Functions" on page 358.

ORDER BY Clause Syntax

The Oracle BI Server accepts any valid SQL ORDER BY clause syntax, including referencing columns by their order in the select list (such as ORDER BY 3, 1, 5).

Rules for Queries with Aggregate Functions

The Oracle BI Server simplifies the SQL needed to craft aggregate queries. This section outlines the rules that the Oracle BI Server follows with respect to whether or not a query contains a GROUP BY clause and, if a GROUP BY clause is specified, what results you should expect from the query. The rules outlined in this section apply to all aggregates used in SQL statements (SUM, AVG, MIN, MAX, COUNT(*), and COUNT).

Computing Aggregates of Baseline Columns

A *baseline column* is a column that has no aggregation rule defined in the Aggregation tab of the Logical Column dialog in the repository. Baseline columns map to nonaggregated data at the level of granularity of the logical table to which they belong. If you perform aggregation (SUM, AVG, MIN, MAX, or COUNT) on a baseline column through a SQL request, the Oracle BI Server calculates the aggregation at the level based on the following rules:

- If there is no GROUP BY clause specified, the level of aggregation is grouped by all of the nonaggregate columns in the SELECT list.
- If there is a GROUP BY clause specified, the level of aggregation is based on the columns specified in the GROUP BY clause.

For example, consider the following query, where the column *revenue* is defined in the repository as a baseline column (no aggregation rules specified in the Logical Column > Aggregation tab):

```
select year, product, sum(revenue)

from time, products, facts

YEAR     PRODUCT      SUM(REVENUE)

1998     Coke          500

1998     Pepsi         600

1999     Coke          600
```

```
1999    Pepsi        550

2000    Coke         800

2000    Pepsi        600
```

This query returns results grouped by *year* and *product*; that is, it returns one row for each product and year combination. The sum calculated for each row is the sum of all the sales for that product in that year. It is logically the same query as the following:

```
select year, product, sum(revenue)

from time, products, facts

group by year, product
```

If you change the GROUP BY clause to only group by *year*, then the sum calculated is the sum of all products for the year, as follows:

```
select year, product, sum(revenue)

from time, products, facts

group by year


YEAR    PRODUCT      SUM(REVENUE)
1998    Coke         1100
1998    Pepsi        1100
1999    Coke         1150
1999    Pepsi        1150
2000    Coke         1400
2000    Pepsi        1400
```

In this query result set, the sum of *revenue* is the same for each row corresponding to a given year, and that sum represents the total sales for that year. In this case, it is the sales of Coke plus the sales of Pepsi.

If you add a column to the query requesting the COUNT of *revenue*, the Oracle BI Server calculates the number of records used to calculate the results for each group. In this case, it is a year, as shown in the following example:

```
select year, product, sum(revenue), count(revenue)
from time, products, facts
group by year

YEAR    PRODUCT      SUM(REVENUE)  COUNT(REVENUE)
1998    Coke         1100          6000
1998    Pepsi        1100          6000
1999    Coke         1150          6500
1999    Pepsi        1150          6500
2000    Coke         1400          8000
2000    Pepsi        1400          8000
```

Computing Aggregates of Measure Columns

A *measure column* is a column that has a default aggregation rule defined in the Aggregation tab of the Logical Column dialog in the repository. Measure columns always calculate the aggregation with which they are defined. If you perform explicit aggregation (SUM, AVG, MIN, MAX, or COUNT) on a measure column through a SQL request, you are actually asking for an aggregate of an aggregate. For these *nested* aggregates, the Oracle BI Server calculates the aggregation based on the following rules:

- A request for a measure column without an aggregate function defined in a SQL statement is always grouped at the level of the nonaggregate columns in the SELECT list, regardless of whether the query specifies a GROUP BY clause.
- If there is no GROUP BY clause specified, the nested aggregate is a grand total of each group determined by all of the nonaggregate columns in the SELECT list.
- If there is a GROUP BY clause specified, the nested aggregation calculates the total for each group as specified in the GROUP BY clause.

For example, consider the following query, where the column SumOfRevenue is defined in the repository as a measure column with a default aggregation rule of SUM (SUM aggregation rule specified in the Aggregation tab of the Logical Column dialog):

```
select year, product, SumOfRevenue, sum(SumOfRevenue)
from time, products, facts

YEAR    PRODUCT      SUMofREVENUE   SUM(SUMofREVENUE)
1998    Coke         500            3650
1998    Pepsi        600            3650
1999    Coke         600            3650
1999    Pepsi        550            3650
2000    Coke         800            3650
2000    Pepsi        600            3650
```

This query returns results grouped by *year* and *product*; that is, it returns one row for each product and year combination. The sum calculated for each row in the SumOfRevenue column is the sum of all the sales for that product in that year because the measure column is always at the level defined by the nonaggregation columns in the query. It is logically the same query as the following:

```
select year, product, SumOfRevenue, sum(SumOfRevenue)
from time, products, facts
group by year, product
```

If you change the GROUP BY clause to only group by *year*, then the sum calculated in the SumOfRevenue column is the sum of each product for the year, and the sum calculated in the SUM(SumOfRevenue) column is total sales of all products for the given year, as follows:

```
select year, product, SumOfRevenue, sum(SumOfRevenue)
from time, products, facts
group by year

YEAR    PRODUCT      SUMofREVENUE   SUM(SUMofREVENUE)
1998    Coke         500            1100
1998    Pepsi        600            1100
```

```
1999    Coke        600          1150
1999    Pepsi       550          1150
2000    Coke        800          1400
2000    Pepsi       600          1400
```

In this result set, the sum calculated for each row in the SumOfRevenue column is the sum of all the sales for that product in that year because the measure column is always at the level defined by the nonaggregation columns in the query. The SUM(SumOfRevenue) is the same for each row corresponding to a given year, and that sum represents the total sales for that year. In this case, it is the sales of Coke plus the sales of Pepsi.

Display Function Reset Behavior

A *display function* is a function that operates on the result set of a query. The display functions the Oracle BI Server supports (RANK, TOP*n*, BOTTOM*n*, PERCENTILE, NTILE, MAVG, MEDIAN, and varieties of standard deviation) are specified in the SELECT list of a SQL query. Queries that use display functions conform to the following rules:

- If no GROUP BY clause is specified, the display function operates across the entire result set; that is, the grouping level for the display function is the same as for the query.
- If there is a GROUP BY clause specified, the display function resets its values for each group as specified in the GROUP BY clause.

For example, in the following query, SumOfRevenue is defined as a measure column with the default aggregation rule of SUM:

```
select year, product, SumOfRevenue, rank(SumOfRevenue)
from time, products, facts

YEAR    PRODUCT     SUMOFREVENUE RANK(SUMOFREVENUE)
1998    Coke        500          6
1998    Pepsi       600          2
1999    Coke        600          2
1999    Pepsi       550          5
2000    Coke        800          1
2000    Pepsi       600          2
```

In this query result set, there is no GROUP BY clause specified, so the rank is calculated across the entire result set. The query is logically the same query as the following:

```
select year, product, SumOfRevenue, rank(SumOfRevenue))
from time, products, facts
group by year, product
```

If you change the GROUP BY clause to only group by *year*, then the rank is reset for each year, as follows:

```
select year, product, sum(revenue), rank(sum(revenue))
from time, products, facts
group by year
```

```
YEAR    PRODUCT      SUM(REVENUE) RANK(SUM(REVENUE))
1998    Coke         500          2
1998    Pepsi        600          1
1999    Coke         600          1
1999    Pepsi        550          2
2000    Coke         800          1
2000    Pepsi        600          2
```

In this result set, the rank is reset each time the year changes, and because there are two rows for each year, the value of the rank is always either
1 or 2.

Alternative Syntax

When using an aggregate function, you can calculate a specified level of aggregation using BY within the aggregate function. If you do this, you do not need a GROUP BY clause.

For example, the query:

```
select year, product, revenue, sum(revenue by year) as year_revenue from softdrinks
```

will return the column year_revenue that displays revenue aggregated by year.

The same syntax can be used with display functions. The query:

```
select year, product, revenue, rank(revenue), rank(revenue by year) from softdrinks
order by 1, 5
```

will calculate overall rank of revenue for each product for each year (each row in the entire result set) and also the rank of each product's revenue within each year.

Using FILTER to Compute a Conditional Aggregate

In SQL query language, traditional aggregates, such as SUM, COUNT, MIN, and MAX are evaluated on a group of tuples (an ordered list of objects, each of a specified type), determined by the GROUP BY clause. All the aggregates specified in the SELECT clause of a query are evaluated over the same subset of tuples. Conditional aggregates extend SQL by restricting their input using a predicate.

FILTER is an operator that restricts the set of rows used to compute its aggregate argument to rows that satisfy the USING condition. The FILTER operator is a logical SQL construct. It may be used in logical queries referring to the metadata, or in logical columns that use existing logical columns as the source.

Syntax

Conditional aggregates are only notational concepts and they do not represent executable operators. Conditional aggregates are expressed in the form of a function as shown in the following statement:

```
FILTER(<measure_expression> USING <boolean_expression>)
```

Where:

- <measure_expression> is an expression that contains at least one measure. The following is a list of examples:
  - The expression Sales + 1 is allowed if Sales is a measure.
  - The expression productid is not allowed if productid is a scalar attribute.
- <boolean_expression> is a boolean expression (evaluates to TRUE or FALSE) that does not contain any measures. This expression may not contain any nested queries.

Example of the FILTER Function

The following is a simple example of the FILTER function:

```
SELECT year,

FILTER(sales USING product = 'coke'),

FILTER(sales USING product = 'pepsi')

FROM logBeverages
```

After navigation, this query is executed as follows:

```
SELECT year,

SUM(CASE WHEN product = 'coke' THEN sales),

SUM(CASE WHEN product = 'pepsi' THEN sales)

FROM physBeverages

WHERE product = 'coke' OR product = 'pepsi'

GROUP BY year
```

Error handling

In the example FILTER(X USING Y), error messages will be returned in the following situations:

- The Y expression is not a boolean expression.
- The Y expression contains measures.
- FILTER is used in outer query block.
- Explicit aggregates are used in the X (measure) expression. For example, FILTER(COUNT(product), C).

SQL Logical Operators

The following SQL logical operators are used to specify comparisons between expressions.

Between: Used to determine boundaries for a condition. Each boundary is an expression, and the bounds do not include the boundary limits, as in less than and greater than (as opposed to less than or equal to and greater than or equal to). BETWEEN can be preceded with NOT to negate the condition.

In: Specifies a comparison of a column value with a set of values.

Is Null: Specifies a comparison of a column value with the null value.

Like: Specifies a comparison to a literal value. Often used with wildcard characters to indicate any character string match of zero or more characters (%) or a any single character match (_).

Conditional Expressions

The Expressions folder contains building blocks for creating conditional expressions that use CASE, WHEN, THEN and ELSE statements.

CASE (Switch)

This form of the Case statement is also referred to as the CASE (Lookup) form.

Syntax:

```
CASE expression1
     WHEN expression2 THEN expression3
     {WHEN expression... THEN expression...}
     ELSE expression...
END
```

The value of expression1 is examined, and then the WHEN expressions are examined. If expression1 matches any WHEN expression, it assigns the value in the corresponding THEN expression.

If none of the WHEN expressions match, it assigns the default value specified in the ELSE expression. If no ELSE expression is specified, the system will automatically add an ELSE NULL.

If expression1 matches an expression in more than one WHEN clause, only the expression following the first match is assigned.

NOTE: In a CASE statement, AND has precedence over OR.

CASE

Starts the CASE statement. Has to be followed by an expression and one or more WHEN and THEN statements, an optional ELSE statement, and the END keyword.

WHEN

Specifies the condition to be satisfied.

THEN

Specifies the value to assign if the corresponding WHEN expression is satisfied.

ELSE

Specifies the value to assign if none of the WHEN conditions are satisfied. If omitted, ELSE NULL is assumed.

END

Ends the Case statement.

CASE (If)

This form of the CASE statement has the following syntax:

```
CASE
     WHEN search_condition1 THEN expression1
     {WHEN search_condition2 THEN expression2}
     {WHEN search_condition... THEN expression...}
     ELSE expression
END
```

This evaluates each WHEN condition and if satisfied, assigns the value in the corresponding THEN expression.

If none of the WHEN conditions are satisfied, it assigns the default value specified in the ELSE expression.

If no ELSE expression is specified, the system will automatically add an ELSE NULL.

NOTE: In a CASE statement, AND has precedence over OR.

Unlike the Switch form of the CASE statement, the WHEN statements in the If form allow comparison operators; a WHEN condition of WHEN < 0 THEN 'Under Par' is legal.

CASE

Starts the CASE statement. Has to be followed by one or more WHEN and THEN statements, an optional ELSE statement, and the END keyword.

WHEN

Specifies the condition to be satisfied.

THEN

Specifies the value to assign if the corresponding WHEN expression is satisfied.

ELSE

Specifies the value to assign if none of the WHEN conditions are satisfied. If omitted, ELSE NULL is assumed.

END

Ends the Case statement.

SQL Reference

SQL functions perform various calculations on column values. This section explains the syntax for the functions supported by the Oracle BI Server. It also explains how to express literals. There are aggregate, string, math, calendar date/time, conversion, and system functions.

The following topics are included:

Aggregate Functions

Aggregate functions perform work on multiple values to create summary results. The aggregate functions cannot be used to form nested aggregation in expressions on logical columns that have a default aggregation rule defined in the Aggregation tab of the Logical Column dialog box. To specify nested aggregation, you need to define a column with a default aggregation rule and then request the aggregation of the column in a SQL statement.

Avg

Calculates the average (mean) value of an expression in a result set. Has to take a numeric expression as its argument.

Syntax:

```
AVG (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

AvgDistinct

Calculates the average (mean) of all distinct values of an expression. Has to take a numeric expression as its argument.

Syntax:

```
AVG (DISTINCT n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

BottomN

Ranks the lowest n values of the expression argument from 1 to n, 1 corresponding to the lowest numerical value. The BOTTOMN function operates on the values returned in the result set.

Syntax:

```
BOTTOMN (n_expression, n)
```

where:

n_expression	Any expression that evaluates to a numerical value.
n	Any positive integer. Represents the bottom number of rankings displayed in the result set, 1 being the lowest rank.

NOTE: A query can contain only one BOTTOMN expression.

Count

Calculates the number of rows having a nonnull value for the expression. The expression is typically a column name, in which case the number of rows with nonnull values for that column is returned.

Syntax:

```
COUNT (expression)
```

where:

expression	Any expression.

CountDistinct

Adds distinct processing to the COUNT function.

Syntax:

```
COUNT (DISTINCT expression)
```

where:

expression	Any expression.

Count (*) (CountStar)

Counts the number of rows.

Syntax:

```
COUNT(*)
```

For example, if a table named Facts contained 200,000,000 rows, the following query would return the following results:

```
SELECT COUNT(*) FROM Facts
```

```
COUNT(*)
```

```
200000000
```

First

Selects the first returned value of the expression argument. The FIRST function is limited to defining dimension-specific aggregation rules in a repository. You cannot use it in SQL statements.

The FIRST function operates at the most detailed level specified in your explicitly defined dimension. For example, if you have a time dimension defined with hierarchy levels day, month, and year, the FIRST function returns the first day in each level.

You should not use the FIRST function as the first dimension-specific aggregate rule. It might cause queries to bring back large numbers of rows for processing in the Oracle BI Server causing poor performance.

Syntax:

FIRST (expression)

where:

expression	Any expression.

GroupByColumn

For use in setting up aggregate navigation. It specifies the logical columns that define the level of the aggregate data existing in a physical aggregate table.

For example, if an aggregate table contains data grouped by store and by month, specify the following syntax in the content filter (General tab of Logical Source dialog):

```
GROUPBYCOLUMN(STORE, MONTH)
```

The GROUPBYCOLUMN function is only for use in configuring a repository; you cannot use it to form SQL statements.

GroupByLevel

For use in setting up aggregate navigation. It specifies the dimension levels that define the level of the aggregate data existing in a physical aggregate table.

For example, if an aggregate table contains data at the store and month levels, and if you have defined dimensions (Geography and Customers) containing these levels, specify the following syntax in the content filter (General tab of Logical Source dialog):

```
GROUPBYLEVEL (GEOGRAPHY.STORE, CUSTOMERS.MONTH)
```

The GROUPBYLEVEL function is only for use in configuring a repository; you cannot use it to form SQL statements.

Last

Selects the last returned value of the expression. The LAST function is limited to defining dimension-specific aggregation rules in a repository. You cannot use it in SQL statements.

The LAST function operates at the most detailed level specified in your explicitly defined dimension. For example, if you have a time dimension defined with hierarchy levels day, month, and year, the LAST function returns the last day in each level.

You should not use the LAST function as the first dimension-specific aggregate rule. It might cause queries to bring back large numbers of rows for processing in the Oracle BI Server causing poor performance.

Syntax:

```
LAST (expression)
```

where:

expression	Any expression.

Max

Calculates the maximum value (highest numeric value) of the rows satisfying the numeric expression argument.

Syntax:

```
MAX (expression)
```

where:

expression	Any expression.

The MAX function resets its values for each group in the query, according to the rules outlined in "Display Function Reset Behavior" on page 361.

Median

Calculates the median (middle) value of the rows satisfying the numeric expression argument. When there are an even number of rows, the median is the mean of the two middle rows. This function always returns a double.

Syntax:

```
MEDIAN (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

The MEDIAN function resets its values for each group in the query, according to the rules outlined in "Display Function Reset Behavior" on page 361.

Min

Calculates the minimum value (lowest numeric value) of the rows satisfying the numeric expression argument.

Syntax:

```
MIN (expression)
```

where:

expression	Any expression.

The MIN function resets its values for each group in the query, according to the rules outlined in "Display Function Reset Behavior" on page 361.

NTile

The NTILE function determines the rank of a value in terms of a user-specified range. It returns integers to represent any range of ranks. In other words, the resulting sorted data set is broken into a number of tiles where there are roughly an equal number of values in each tile.

Syntax:

```
NTILE (n_expression, n)
```

where:

n_expression	Any expression that evaluates to a numerical value.
n	A positive, nonnull integer that represents the number of tiles.

If the n_expression argument is not null, the function returns an integer that represents a rank within the requested range.

NTile with n=100 returns what is commonly called the *percentile* (with numbers ranging from 1 to 100, with 100 representing the high end of the sort). This value is different from the results of the Oracle BI Server percentile function, that conforms to what is called *percent rank* in SQL 92 and returns values from 0 to 1.

Percentile

Calculates a percent rank for each value satisfying the numeric expression argument. The percent rank ranges are from 0 (1st percentile) to 1 (100th percentile), inclusive.

The PERCENTILE function calculates the percentile based on the values in the result set of the query.

Syntax:

```
PERCENTILE (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

The PERCENTILE function resets its values for each group in the query according to the rules outlined in "Display Function Reset Behavior" on page 361.

PeriodAgo

A time series aggregation function for relational data sources only. Calculates the aggregated value from the current time back to a specified time period. For example, PeriodAgo can produce sales for every month of the current quarter and the corresponding quarter-ago sales.

If unsupported metrics are requested, NULL values will be returned and a warning entry will be written to the NQQuery.log file when the logging level equals three or above. Multiple PeriodAgo functions can be nested if all the PeriodAgo functions have the same level argument.

You can nest exactly one PeriodToDate and multiple PeriodAgo functions if they each have the same level argument.

Syntax:

PeriodAgo(<time-level>, <offset>, <measure>)

PeriodToDate

A time series aggregation function for relational data sources only. PeriodToDate aggregates a measure attribute from the beginning of a specified time period to the currently displayed time. For example, this function can calculate Year to Date sales.

If unsupported metrics are requested, NULL values will be returned and a warning entry will be written to the NQQuery.log file when the logging level equals three or above. A PeriodToDate function may not be nested within another PeriodToDate function.

You can nest exactly one PeriodToDate and multiple PeriodAgo functions if they each have the same level argument.

Syntax:

PeriodToDate(<time-level>, <measure>)

Rank

Calculates the rank for each value satisfying the numeric expression argument. The highest number is assigned a rank of 1, and each successive rank is assigned the next consecutive integer (2, 3, 4,...). If certain values are equal, they are assigned the same rank (for example, 1, 1, 1, 4, 5, 5, 7...).

The RANK function calculates the rank based on the values in the result set of the query.

Syntax:

```
    RANK (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

The RANK function resets its values for each group in the query according to the rules outlined in "Display Function Reset Behavior" on page 361.

Calculating Absolute Calendar Fields

To calculate absolute calendar fields such as abs_month, you use the prebuilt Rank operator.

Syntax:

```
    Rank(<ordering key>, <partitioning key>, <at_distinct key>)
```

Where:

<ordering key>	Key that is used to order the input for the purpose of assigning rank.
<partitioning key>	Key that is used to partition data. Ranking is restarted from zero at the beginning of every partition.
<at_distinct key>	Dictates that ranking is increased only across rows with differing values in this key. The rows with the same value for this key have the same rank.

The following is a list of examples:

- `Rank(<chronological key>, null, <year key columns>)` returns abs_year.
- `Rank(<chronological key>, null, <month key columns>)` returns abs_month.
- `Rank(<chronological key>, <year key columns>, <month key columns>)` returns month_in_year.

StdDev

The STDDEV function returns the standard deviation for a set of values. The return type is always a double.

Syntax:

```
STDDEV([ALL | DISTINCT] n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

- If ALL is specified, the standard deviation is calculated for all data in the set.
- If DISTINCT is specified, all duplicates are ignored in the calculation.
- If nothing is specified (the default), all data is considered.

There are two other functions that are related to STDDEV:

STDDEV_POP([ALL | DISTINCT] n_expression)

STDDEV_SAMP([ALL | DISTINCT] n_expression)

STDDEV and STDDEV_SAMP are synonyms.

The STDDEV function resets its values for each group in the query according to the rules outlined in "Display Function Reset Behavior" on page 361.

Sum

Calculates the sum obtained by adding up all values satisfying the numeric expression argument.

Syntax:

```
SUM (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

The SUM function resets its values for each group in the query according to the rules outlined in "Display Function Reset Behavior" on page 361.

SumDistinct

Calculates the sum obtained by adding all of the distinct values satisfying the numeric expression argument.

Syntax:

```
SUM(DISTINCT n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

TopN

Ranks the highest n values of the expression argument from 1 to n, 1 corresponding to the highest numerical value.

The TOPN function operates on the values returned in the result set.

Syntax:

```
    TOPN (n_expression, n)
```

where:

n_expression	Any expression that evaluates to a numerical value.
n	Any positive integer. Represents the top number of rankings displayed in the result set, 1 being the highest rank.

A query can contain only one TOPN expression.

The TOPN function resets its values for each group in the query according to the rules outlined in "Display Function Reset Behavior" on page 361.

Running Aggregate Functions

Running aggregate functions are similar to functional aggregates in that they take a set of records as input, but instead of outputting the single aggregate for the entire set of records, they output the aggregate based on records encountered so far.

This section describes the running aggregate functions supported by the Oracle BI Server.

Mavg

Calculates a moving average (mean) for the last n rows of data in the result set, inclusive of the current row.

Syntax:

```
    MAVG (n_expression, n)
```

where:

n_expression	Any expression that evaluates to a numerical value.
n	Any positive integer. Represents the average of the last n rows of data.

The MAVG function resets its values for each group in the query, according to the rules outlined in "Display Function Reset Behavior" on page 361.

The average for the first row is equal to the numeric expression for the first row. The average for the second row is calculated by taking the average of the first two rows of data. The average for the third row is calculated by taking the average of the first three rows of data, and so on until you reach the nth row, where the average is calculated based on the last n rows of data.

MSUM

This function calculates a moving sum for the last n rows of data, inclusive of the current row.

The sum for the first row is equal to the numeric expression for the first row. The sum for the second row is calculated by taking the sum of the first two rows of data. The sum for the third row is calculated by taking the sum of the first three rows of data, and so on. When the nth row is reached, the sum is calculated based on the last n rows of data.

This function resets its values for each group in the query according to the rules described in "Display Function Reset Behavior" on page 361.

Syntax:

```
MSUM (n_expression, n)
```

Where:

n_expression	Any expression that evaluates to a numerical value.
n	Any positive integer. Represents the sum of the last n rows of data.

Example:

The following example shows a query that uses the MSUM function and the query results.

```
select month, revenue, MSUM(revenue, 3) as 3_MO_SUM from sales_subject_area

MONTH      REVENUE    3_MO_SUM
JAN        100.00     100.00
FEB        200.00     300.00
MAR        100.00     400.00
APRIL      100.00     400.00
MAY        300.00     500.00
JUNE       400.00     800.00
JULY       500.00     1200.00
AUG        500.00     1400.00
SEPT       500.00     1500.00
OCT        300.00     1300.00
NOV        200.00     1000.00
DEC        100.00     600.00
```

RSUM

This function calculates a running sum based on records encountered so far. The sum for the first row is equal to the numeric expression for the first row. The sum for the second row is calculated by taking the sum of the first two rows of data. The sum for the third row is calculated by taking the sum of the first three rows of data, and so on.

This function resets its values for each group in the query according to the rules described in "Display Function Reset Behavior" on page 361.

Syntax:

```
RSUM (n_expression)
```

Where:

n_expression	Any expression that evaluates to a numerical value.

Example:

The following example shows a query that uses the RSUM function and the query results.

```
select month, revenue, RSUM(revenue) as RUNNING_SUM from sales_subject_area

MONTH      REVENUE    RUNNING_SUM
JAN        100.00     100.00
FEB        200.00     300.00
MAR        100.00     400.00
APRIL      100.00     500.00
MAY        300.00     800.00
JUNE       400.00     1200.00
JULY       500.00     1700.00
AUG        500.00     2200.00
SEPT       500.00     2700.00
OCT        300.00     3000.00
NOV        200.00     3200.00
DEC        100.00     3300.00
```

RCOUNT

This function takes a set of records as input and counts the number of records encountered so far.

This function resets its values for each group in the query according to the rules described in "Display Function Reset Behavior" on page 361.

Syntax:

```
RCOUNT (Expr)
```

Where:

Expr	An expression of any data type.

Example:

The following example shows a query that uses the RCOUNT function and the query results.

```
select month, profit, RCOUNT(profit) from sales_subject_area where profit > 200.

MONTH      PROFIT     RCOUNT (profit
MAY         300.00      2
JUNE        400.00      3
JULY        500.00      4
AUG         500.00      5
SEPT        500.00      6
OCT         300.00      7
```

RMAX

This function takes a set of records as input and shows the maximum value based on records encountered so far. The specified data type must be one that can be ordered.

This function resets its values for each group in the query according to the rules described in "Display Function Reset Behavior" on page 361.

Syntax:

```
RMAX (expression)
```

Where:

expression	An expression of any data type. The data type must be one that has an associated sort order.

Example:

The following example shows a query that uses the RMAX function and the query results.

```
select month, profit, RMAX(profit) from sales_subject_area

MONTH       PROFIT      RMAX (profit)
JAN         100.00      100.00
FEB         200.00      200.00
MAR         100.00      200.00
APRIL       100.00      200.00
MAY         300.00      300.00
JUNE        400.00      400.00
JULY        500.00      500.00
AUG         500.00      500.00
SEPT        500.00      500.00
OCT         300.00      500.00
NOV         200.00      500.00
DEC         100.00      500.00
```

RMIN

This function takes a set of records as input and shows the minimum value based on records encountered so far. The specified data type must be one that can be ordered.

This function resets its values for each group in the query according to the rules described in "Display Function Reset Behavior" on page 361.

Syntax:

```
RMIN (expression)
```

Where:

expression	An expression of any data type. The data type must be one that has an associated sort order.

Example:

The following example shows a query that uses the RMIN function and the query results.

```
select month, profit, RMIN(profit) from sales_subject_area

MONTH       PROFIT      RMIN
                        (profit)
JAN         400.00      400.00
FEB         200.00      200.00
MAR         100.00      100.00
APRIL       100.00      100.00
MAY         300.00      100.00
JUNE        400.00      100.00
JULY        500.00      100.00
AUG         500.00      100.00
SEPT        500.00      100.00
OCT         300.00      100.00
NOV         200.00      100.00
DEC         100.00      100.00
```

String Functions

String functions perform various character manipulations, and they operate on character strings.

ASCII

Converts a single character string to its corresponding ASCII code, between 0 and 255.

Syntax:

```
ASCII (character_expression)
```

where:

character_expression	Any expression that evaluates to an ASCII character.

If the character expression evaluates to more than one character, the ASCII code corresponding to the first character in the expression is returned.

Bit_Length

Returns the length, in bits, of a specified string. Each Unicode character is 2 bytes in length (equal to 16 bits).

Syntax:

```
BIT_LENGTH (character_expression)
```

where:

character_expression	Any expression that evaluates to character string.

Char

Converts a numerical value between 0 and 255 to the character value corresponding to the ASCII code.

Syntax:

```
CHAR (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value between 0 and 255.

Char_Length

Returns the length, in number of characters, of a specified string. Leading and trailing blanks are not counted in the length of the string.

Syntax:

```
CHAR_LENGTH (character_expression)
```

where:

character_expression	Any expression that evaluates to a numerical value between 0 and 255.

Concat

There are two forms of this function. The first form concatenates two character strings. The second form uses the character string concatenation character to concatenate more than two character strings.

Form 1 Syntax:

```
CONCAT (character_expression1, character_expression2)
```

where:

character_expression	Expressions that evaluate to character strings.

Form 2 Syntax:

```
CONCAT (string_expression1 || string_expression2 || ... string_expressionxx)
```

where:

string_expression	Expressions that evaluate to character strings, separated by the character string concatenation operator || (double vertical bars). The first string is concatenated with the second string to produce an intermediate string, and then this string is concatenated with the next string, and so on.

Insert

Inserts a specified character string into a specified location in another character string.

Syntax:

```
INSERT(character_expression, n, m, character_expression)
```

where:

character_expression	Any expression that evaluates to a character string.
n	Any positive integer representing the number of characters from the start of the first string where a portion of the second string is inserted.
m	Any positive integer representing the number of characters in the first string to be replaced by the entirety of the second string.

Left

Returns a specified number of characters from the left of a string.

Syntax:

```
LEFT(character_expression, n)
```

where:

character_expression	Any expression that evaluates to a character string.
n	Any positive integer representing the number of characters from the left of the first string that are returned.

Length

Returns the length, in number of characters, of a specified string. The length is returned excluding any trailing blank characters.

Syntax:

```
LENGTH(character_expression)
```

where:

character_expression	Any expression that evaluates to a character string.

Locate

Returns the numerical position of the character_expression1 in a character expression. If the character_expression1 is not found in the character expression, the Locate function returns a value of 0. If you want to specify a starting position to begin the search, use the LocateN function instead.

Syntax:

```
LOCATE(character_expression1, character_expression2)
```

where:

character_expression1	Any expression that evaluates to a character string. This is the expression to search for in the character expression.
character_expression2	Any expression that evaluates to a character string. This is the expression to be searched.

LocateN

Returns the numerical position of the character_expression1 in a character expression. This is identical to the Locate function, except that the search for the pattern begins at the position specified by an integer argument. If the character_expression1 is not found in the character expression, the LocateN function returns a value of 0. The numerical position to return is determined by counting the first character in the string as occupying position 1, regardless of the value of the integer argument.

Syntax:

```
LOCATE(character_expression1, character_expression2, n)
```

where:

character_expression1	Any expression that evaluates to a character string. This is the expression to search for in the character expression.
character_expression2	Any expression that evaluates to a character string. This is the expression to be searched.
n	Any positive, nonzero integer that represents the starting position to being to look for the locate expression.

Lower

Converts a character string to lower case.

Syntax:

```
LOWER (character_expression)
```

where:

character_expression	Any expression that evaluates to a character string.

Octet_Length

Returns the bits, in base 8 units (number of bytes), of a specified string.

Syntax:

```
OCTET_LENGTH (character_expression)
```

where:

character_expression	Any expression that evaluates to a character string.

Position

Returns the numerical position of the character_expression1 in a character expression. If the character_expression1 is not found, the function returns 0.

Syntax:

```
POSITION(character_expression1 IN character_expression2)
```

where:

character_expression1	Any expression that evaluates to a character string. Used to search in the second string.
character_expression2	Any expression that evaluates to a character string.

Repeat

Repeats a specified expression n times, where n is a positive integer.

Syntax:

```
REPEAT(character_expression, n)
```

where:

character_expression	Any expression that evaluates to a character string.
n	Any positive integer.

Replace

Replaces specified characters from a specified character expression with other specified characters.

Syntax:

```
REPLACE(character_expression, change_expression, replace_with_expression)
```

where:

character_expression	Any expression that evaluates to a character string. This first string is the original string.
change_expression	Any expression that evaluates to a character string. This second string specifies characters from the first string that will be replaced.
replace_with_expression	Any expression that evaluates to a character string. This third string specifies the characters to substitute into the first string.

Right

Returns a specified number of characters from the right of a string.

Syntax:

```
RIGHT(character_expression, n)
```

where:

character_expression	Any expression that evaluates to a character string.
n	Any positive integer representing the number of characters from the right of the first string that are returned.

Substring

Creates a new string starting from a fixed number of characters into the original string.

Syntax:

```
SUBSTRING (character_expression FROM starting_position)
```

where:

character_expression	Any expression that evaluates to a character string.
starting_position	Any positive integer representing the number of characters from the start of the string where the result begins.

TrimBoth

Strips specified leading and trailing characters from a character string.

Syntax:

```
TRIM (BOTH 'character' FROM character_expression)
```

where:

character	Any single character. If the character part of the specification (and the single quotes) are omitted, a blank character is used as a default.
character_expression	Any expression that evaluates to a character string.

TrimLeading

Strips specified leading characters from a character string.

Syntax:

```
TRIM (LEADING 'character' FROM character_expression)
```

where:

character	Any single character. If the character part of the specification (and the single quotes) are omitted, a blank character is used as a default.
character_expression	Any expression that evaluates to a character string.

TrimTrailing

Strips specified trailing characters from a character string.

Syntax:

```
TRIM (TRAILING 'character' FROM character_expression)
```

where:

character	Any single character. If the character part of the specification (and the single quotes) are omitted, a blank character is used as a default.
character_expression	Any expression that evaluates to a character string.

Upper

Converts a character string to uppercase.

Syntax:

```
UPPER (character_expression)
```

where:

character_expression	Any expression that evaluates to a character string.

Math Functions

The math functions perform mathematical operations.

Abs

Calculates the absolute value of a numerical expression.

Syntax:

```
ABS (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Acos

Calculates the arc cosine of a numerical expression.

Syntax:

```
ACOS (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Asin

Calculates the arc sine of a numerical expression.

Syntax:

```
ASIN (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Atan

Calculates the arc tangent of a numerical expression.

Syntax:

```
ATAN (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Atan2

Calculates the arc tangent of y/x, where y is the first numerical expression and x is the second numerical expression.

Syntax:

```
    ATAN2 (n_expression1, n_expression2)
```

where:

n_expression (1 and 2)	Any expression that evaluates to a numerical value.

Ceiling

Rounds a noninteger numerical expression to the next highest integer. If the numerical expression evaluates to an integer, the Ceiling function returns that integer.

Syntax:

```
    CEILING (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Cos

Calculates the cosine of a numerical expression.

Syntax:

```
    COS (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Cot

Calculates the cotangent of a numerical expression.

Syntax:

```
    COT (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Degrees

Converts an expression from radians to degrees.

Syntax:

```
    DEGREES (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Exp

Sends the value e to the power specified.

Syntax:

```
EXP (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Floor

Rounds a noninteger numerical expression to the next lowest integer. If the numerical expression evaluates to an integer, the FLOOR function returns that integer.

Syntax:

```
FLOOR (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Log

Calculates the natural logarithm of an expression.

Syntax:

```
LOG (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Log10

Calculates the base 10 logarithm of an expression.

Syntax:

```
LOG10 (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Mod

Divides the first numerical expression by the second numerical expression and returns the remainder portion of the quotient.

Syntax:

```
MOD (n_expression1, n_expression2)
```

where:

n_expression (1 and 2)	Any expression that evaluates to a numerical value.

Pi

Returns the constant value of pi (the circumference of a circle divided by the diameter of a circle).

Syntax:

```
PI()
```

Power

Takes the first numerical expression and raises it to the power specified in the second numerical expression.

Syntax:

```
POWER(n_expression1, n_expression2)
```

where:

n_expression (1 and 2)	Any expression that evaluates to a numerical value.

Radians

Converts an expression from degrees to radians.

Syntax:

```
RADIANS (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Rand

Returns a pseudo-random number between 0 and 1.

Syntax:

```
RAND()
```

RandFromSeed

Returns a pseudo-random number based on a seed value. For a given seed value, the same set of random numbers are generated.

Syntax:

```
RAND (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Round

Rounds a numerical expression to n digits of precision.

Syntax:

```
ROUND (n_expression, n)
```

where:

n_expression	Any expression that evaluates to a numerical value.
n	Any positive integer representing the number of digits of precision with which to round.

Sign

Returns a value of 1 if the numerical expression argument evaluates to a positive number, a value of -1 if the numerical expression argument evaluates to a negative number, and 0 if the numerical expression argument evaluates to zero.

Syntax:

```
SIGN (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Sin

Calculates the sine of a numerical expression.

Syntax:

```
SIN (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Sqrt

Calculates the square root of the numerical expression argument. The numerical expression has to evaluate to a nonnegative number.

Syntax:

```
SQRT (n_expression)
```

where:

n_expression	Any expression that evaluates to a nonnegative numerical value.

Tan

Calculates the tangent of a numerical expression.

Syntax:

```
TAN (n_expression)
```

where:

n_expression	Any expression that evaluates to a numerical value.

Truncate

Truncates a decimal number to return a specified number of places from the decimal point.

Syntax:

```
TRUNCATE (n_expression, n)
```

where:

n_expression	Any expression that evaluates to a numerical value.
n	Any positive integer representing the number of characters from the right of the decimal place that are returned.

Calendar Date/Time Functions

The calendar date/time functions manipulate data of the data types DATE and DATETIME.

Current_Date

Returns the current date. The date is determined by the system in which the Oracle BI Server is running.

Syntax:

```
CURRENT_DATE
```

Current_Time

Returns the current time. The time is determined by the system in which the Oracle BI Server is running.

Syntax:

```
CURRENT_TIME (n)
```

where:

n	Any integer representing the number of digits of precision with which to display the fractional second. The argument is optional; the function returns the default precision when no argument is specified.

Current_TimeStamp

Returns the current date/timestamp. The timestamp is determined by the system in which the Oracle BI Server is running.

Syntax:

```
CURRENT_TIMESTAMP (n)
```

where:

n	Any integer representing the number of digits of precision with which to display the fractional second. The argument is optional; the function returns the default precision when no argument is specified.

Day_Of_Quarter

Returns a number (between 1 and 92) corresponding to the day of the quarter for the specified date.

Syntax:

```
DAY_OF_QUARTER (date_expression)
```

where:

date_expression	Any expression that evaluates to a date.

DayName

Returns the day of the week for a specified date.

Syntax:

```
DAYNAME (date_expression)
```

where:

date_expression	Any expression that evaluates to a date.

DayOfMonth

Returns the number corresponding to the day of the month for a specified date.

Syntax:

```
DAYOFMONTH (date_expression)
```

where:

date_expression	Any expression that evaluates to a date.

DayOfWeek

Returns a number between 1 and 7 corresponding to the day of the week, Sunday through Saturday, for a specified date. For example, the number 1 corresponds to Sunday, and the number 7 corresponds to Saturday.

Syntax:

DAYOFWEEK (date_expression)

where:

date_expression	Any expression that evaluates to a date.

DayOfYear

Returns the number (between 1 and 366) corresponding to the day of the year for a specified date.

Syntax:

```
DAYOFYEAR (date_expression)
```

where:

date_expression	Any expression that evaluates to a date.

Hour

Returns a number (between 0 and 23) corresponding to the hour for a specified time. For example, 0 corresponds to 12 a.m. and 23 corresponds to 11 p.m.

Syntax:

```
HOUR (time_expression)
```

where:

time_expression	Any expression that evaluates to a time.

Minute

Returns a number (between 0 and 59) corresponding to the minute for a specified time.

Syntax:

```
MINUTE (time_expression)
```

where:

time_expression	Any expression that evaluates to a time.

Month

Returns the number (between 1 and 12) corresponding to the month for a specified date.

Syntax:

```
MONTH (date_expression)
```

where:

date_expression	Any expression that evaluates to a date.

Month_Of_Quarter

Returns the number (between 1 and 3) corresponding to the month in the quarter for a specified date.

Syntax:

```
MONTH_OF_QUARTER (date_expression)
```

where:

date_expression	Any expression that evaluates to a date.

MonthName

Returns the name of the month for a specified date.

Syntax:

```
MONTHNAME (date_expression)
```

where:

date_expression	Any expression that evaluates to a date.

Now

Returns the current timestamp. The NOW function is equivalent to the CURRENT_TIMESTAMP function.

Syntax:

```
    NOW ()
```

Quarter_Of_Year

Returns the number (between 1 and 4) corresponding to the quarter of the year for a specified date.

Syntax:

```
    QUARTER_OF_YEAR (date_expression)
```

where:

date_expression	Any expression that evaluates to a date.

Second

Returns the number (between 0 and 59) corresponding to the seconds for a specified time.

Syntax:

```
    SECOND (time_expression)
```

where:

time_expression	Any expression that evaluates to a time.

TimestampAdd

The TimestampAdd function adds a specified number of intervals to a specified timestamp. A single timestamp is returned.

Syntax:

```
TimestampAdd (interval, integer-expression, timestamp-expression)
```

where:

interval	The specified interval. Valid values are: SQL_TSI_SECOND SQL_TSI_MINUTE SQL_TSI_HOUR SQL_TSI_DAY SQL_TSI_WEEK SQL_TSI_MONTH SQL_TSI_QUARTER SQL_TSI_YEAR
integer_expression	Any expression that evaluates to an integer.
timestamp_expression	The timestamp used as the base in the calculation.

A null integer-expression or a null timestamp-expression passed to this function will result in a null return value.

In the simplest scenario, this function merely adds the specified integer value (integer-expression) to the appropriate component of the timestamp, based on the interval. Adding a week translates to adding seven days, and adding a quarter translates to adding three months. A negative integer value results in a subtraction (going back in time).

An overflow of the specified component (such as more than 60 seconds, 24 hours, twelve months, and so on) necessitates adding an appropriate amount to the next component. For example, when adding to the day component of a timestamp, this function considers overflow and takes into account the number of days in a particular month (including leap years when February has 29 days).

When adding to the month component of a timestamp, this function verifies that the resulting timestamp has a sufficient number of days for the day component. For example, adding 1 month to 2000-05-31 does not result in 2000-06-31 because June does not have 31 days. This function reduces the day component to the last day of the month, 2000-06-30 in this example.

A similar issue arises when adding to the year component of a timestamp having a month component of February and a day component of 29 (that is, last day of February in a leap year). If the resulting timestamp does not fall on a leap year, the function reduces the day component to 28.

These actions conform to the behavior of Microsoft's SQL Server and Oracle's native OCI interface.

The following queries are examples of the TimestampAdd function and its results:

The following query asks for the resulting timestamp when 3 days are added to 2000-02-27 14:30:00. Since February, 2000 is a leap year, the query returns a single timestamp of 2000-03-01 14:30:00.

```
Select TimestampAdd(SQL_TSI_DAY, 3,

TIMESTAMP‘2000-02-27 14:30:00’)
```

```
From Employee where employeeid = 2;
```

The following query asks for the resulting timestamp when 7 months are added to 1999-07-31 0:0:0. The query returns a single timestamp of 2000-02-29 00:00:00. Notice the reduction of day component to 29 because of the shorter month of February.

```
Select TimestampAdd(SQL_TSI_MONTH, 7,

TIMESTAMP‘1999-07-31 00:00:00’)

From Employee where employeeid = 2;
```

The following query asks for the resulting timestamp when 25 minutes are added to 2000-07-31 23:35:00. The query returns a single timestamp of 2000-08-01 00:00:00. Notice the propagation of overflow through the month component.

```
Select TimestampAdd(SQL_TSI_MINUTE, 25,

TIMESTAMP‘2000-07-31 23:35:00’)

From Employee where employeeid = 2;
```

CAUTION: The TIMESTAMPADD function is turned on by default for Microsoft SQL Server, ODBC, IBM DB2, and Oracle databases. Because DB2 and Oracle semantics do not fully support this function, the answers from this function might not match exactly with what the Oracle BI Server computes.

TimeStampDiff

The TimestampDiff function returns the total number of specified intervals between two timestamps.

Syntax:

```
TimestampDiff (interval, timestamp-expression1, timestamp-expression2)
```

where:

interval	The specified interval. Valid values are: SQL_TSI_SECOND SQL_TSI_MINUTE SQL_TSI_HOUR SQL_TSI_DAY SQL_TSI_WEEK SQL_TSI_MONTH SQL_TSI_QUARTER SQL_TSI_YEAR
timestamp_expression1	The timestamp used in the function.
timestamp_expression2	The first timestamp used in the function.

A null timestamp-expression parameter passed to this function will result in a null return value.

This function first determines the timestamp component that corresponds to the specified interval parameter. For example, SQL_TSI_DAY corresponds to the day component and SQL_TSI_MONTH corresponds to the month component.

The function then looks at the higher order components of both timestamps to calculate the total number of intervals for each timestamp. For example, if the specified interval corresponds to the month component, the function calculates the total number of months for each timestamp by adding the month component and twelve times the year component.

Finally, the function subtracts the first timestamp's total number of intervals from the second timestamp's total number of intervals.

The TimestampDiff function rounds up to the next integer whenever fractional intervals represent a crossing of an interval boundary. For example, the difference in years between 1999-12-31 and 2000-01-01 is one year because the fractional year represents a crossing from one year to the next (that is, 1999 to 2000). By contrast, the difference between 1999-01-01 and 1999-12-31 is zero years because the fractional interval falls entirely within a particular year (that is, 1999).

Microsoft's SQL Server exhibits the same rounding behavior. IBM DB2 always rounds down. Oracle does not implement a generalized timestamp difference function.

When calculating the difference in weeks, the function calculates the difference in days and divides by seven before rounding. Additionally, the function takes into account how the Oracle BI Administrator has configured the start of a new week in the NQSConfig.INI file using the parameter FIRST_DAY_OF_THE_WEEK (defaults to Sunday).

For example, with Sunday as the start of the week, the difference in weeks between 2000-07-06 (a Thursday) and 2000-07-10 (the following Monday) results in a value of one week. With Tuesday as the start of the week, however, the function would return zero weeks since the fractional interval falls entirely within a particular week.

When calculating the difference in quarters, the function calculates the difference in months and divides by three before rounding.

IBM DB2 provides a generalized timestamp difference function (TIMESTAMPDIFF) but it simplifies the calculation by always assuming a 365-day year, 52-week year, and 30-day month.

TimestampDiff Function and Results Example

The following query asks for a difference in days between timestamps 1998-07-31 23:35:00 and 2000-04-01 14:24:00. It returns a value of 610. Notice that the leap year in 2000 results in an additional day.

```
Select TimestampDIFF(SQL_TSI_DAY, TIMESTAMP‘1998-07-31 23:35:00’, TIMESTAMP‘2000-04-01
14:24:00’) From Employee where employeeid = 2;
```

CAUTION: The TIMESTAMPDIFF function is turned on by default for Microsoft SQL Server, ODBC, IBM DB2, and Oracle databases. Because DB2 and Oracle semantics do not fully support this function, the answers from this function might not match exactly with what the Oracle BI Server computes.

Week_Of_Quarter

Returns a number (between 1 and 13) corresponding to the week of the quarter for the specified date.

Syntax:

```
WEEK_OF_QUARTER (date_expression)
```

where:

date_expression	Any expression that evaluates to a date.

Week_Of_Year

Returns a number (between 1 and 53) corresponding to the week of the year for the specified date.

Syntax:

```
WEEK_OF_YEAR (date_expression)
```

where:

date_expression	Any expression that evaluates to a date.

Year

Returns the year for the specified date.

Syntax:

```
YEAR (date_expression)
```

where:

date_expression	Any expression that evaluates to a date.

Conversion Functions

The conversion functions convert a value from one form to another.

Cast

Changes the data type of an expression or a null literal to another data type. For example, you can cast a customer_name (a data type of Char or Varchar) or birthdate (a datetime literal). The following are the supported data types to which the value can be changed:

```
CHARACTER, VARCHAR, INTEGER, FLOAT, SMALLINT, DOUBLE PRECISION, DATE, TIME,
TIMESTAMP, BIT, BIT VARYING
```

NOTE: Depending on the source data type, some destination types are not supported. For example, if the source data type is a BIT string, the destination data type has to be a character string or another BIT string.

The following describes unique characteristics of the CHAR and VARCHAR data types:

- **Casting to a CHAR data type.** You must use a size parameter. If you do not add a size parameter, a default of 30 will be added. Syntax options appear in the following list:
  - It is recommended that you use the following syntax:

    ```
    CAST (expression|NULL AS CHAR(n) )
    ```

    For example, `CAST (companyname AS CHAR(35) )`
  - You can use the following syntax:

    ```
    CAST (expression|NULL AS datatype )
    ```

    For example, `CAST (companyname AS CHAR )`

    NOTE: If you use this syntax, Oracle BI Server will explicitly convert and store as CAST (expression|NULL AS CHAR(30))
- **Casting to a VARCHAR data type.** The Administration Tool requires that you use a size parameter. If you omit the size parameter, you cannot can save the change.

Choose

Takes an arbitrary number of parameters and returns the first item in the list that the user has permission to see. However, the Oracle BI Administrator must model the column permissions in the Administration Tool to enable this behavior. For a alternate method, refer to "IndexCol" on page 400.

Syntax:

```
CHOOSE (expression1, expression2, ..., expressionN)
```

For example, a single query can be written to return security-based revenue numbers for the entire organization. The function could look like the following:

```
choose(L1-Revenue, L2-Revenue, L3-Revenue, L4-Revenue)
```

If the user issuing this function has access to the column L1-Revenue, then that column value would be returned. If the user does not have visibility to the column L1-Revenue but does have visibility to L2-Revenue, then L2-Revenue is returned.

IfNull

Tests if an expression evaluates to a null value, and if it does, assigns the specified value to the expression.

Syntax:

```
IFNULL (expression, value)
```

IndexCol

IndexCol can use external information to return the appropriate column for the logged-in user to see. The Oracle BI Server handles this function in the following ways:

- **ODBC Procedures.** NQSGetLevelDrillability and NQSGenerateDrillDownQuery return the context-specific drill-down information based on the expression translated from IndexCol. This applies to both IndexCol expressions specified in the logical SQL query and IndexCol expressions specified in a derived logical column.
- **Query Log and cache.** The logical SQL with IndexCol function appears in the SQL string in the query log. But the logical request will not show the IndexCol function because Oracle BI Server will translate IndexCol to one of the expressions in its expression list in the logical request generator.

  NOTE: The query cache will use the resulting translated expression for cache hit detection.
- Usage Tracking. Usage tracking will insert the logical SQL query string with the IndexCol function.
- Security. As long as the user has the privileges to access the column(s) in the expression translated from IndexCol, then the query will execute.

  When the first argument to IndexCol is a session variable and if a default expression is expected to be returned even if the init block fails, then the Oracle BI Administrator should set a default value for the session variable. Otherwise, the query will fail because the session variable has no value definition.

Syntax:

```
IndexCol( integer literal, expression_list )
```

Where:

expression_list equals expr1 [, expression_list]

The IndexCol function takes in an integer literal value as its first argument, followed by a variable length expression list and translates to a single expression from the expression list. The literal value is the 0-based index of the expression in the expression list to translate to. Consider the following expression:

```
IndexCol( integer literal, expr1, expr2, … )
```

If the literal value is 0, the above expression is the same as expr1. If the literal value is 1, then the value is the same as expr2, and so on.

NOTE: The primary use case for IndexCol is for the first argument to contain a session variable. Specifying a constant literal would result in IndexCol always choosing the same expression.

Example With Hierarchy Levels

Company ABC has a geography dimension with the hierarchy Country of State, City. The CEO can access the Country level down to the City level, and the sales manager can access the State and City levels, and the sales people can only access the City level. Table 41 on page 401 shows the backend database for Company ABC.

Table 41. IndexCol Example With Hierarchy Levels

USER_NAME	TITLE	GEO LEVEL	CURRENCY	CURRENCY_COL
Bob	CEO	0	US Dollars	0
Harriet	Sales Manager	1	Japanese Yen	1
Jackson	Sales Manager	1	Japanese Yen	1
Mike	Sales Person	2	Japanese Yen	1
Jim	Sales Person	2	US Dollars	0

The following steps illustrate one way to create a single query where each user sees the top level to which they have access:

- The Oracle BI Administrator creates a new session variable GEOOGRAPHY_LEVEL that is populated by the initialization block: SELECT GEO_LEVEL from T where USER_NAME = ':USER'.

  This assume that the Oracle BI Server instance has the same user names.

- Using SELECT IndexCol(VALUEOF(NQ_SESSION.GEOGRAPHY_LEVEL), Country, State, City), Revenue from Sales, the following occurs:

  - Bob logs in and IndexCol translates to the Country column because the GEOGRAPHY_LEVEL session variable is 0. He will get the same result and be able to drill down on Country to State as if he had used SELECT Country, Revenue from Sale.
  - Jackson logs in and IndexCol translates to the State column because the GEOGRAPHY_LEVEL session variable for Jackson is 1. He will get the same result and be able to drill down on State to City as if he had used SELECT State, Revenue from Sales.
  - Mike logs in and IndexCol translates to the City column because the GEOGRAPHY_LEVEL session variable for Mike is 2. He will get the same result and won't be able to drill down on City as if he had used SELECT City, Revenue from Sales.

VALUEOF( )

Use the VALUEOF function in an expression builder or filter to reference the value of a repository variable defined using the Server Administration Tool. You can also use the VALUEOF function when you edit the SQL for a request from the Advanced tab in Oracle BI Answers.

Variables should be used as arguments of the VALUEOF function. Refer to static repository variables by name. For example, to use the value of a static repository variables named prime_begin and prime_end:

```
CASE WHEN "Hour" >= VALUEOF("prime_begin")AND "Hour" < VALUEOF("prime_end") THEN
'Prime Time' WHEN ... ELSE...END
```

You need to refer to a dynamic repository variable by its fully qualified name. If you are using a dynamic repository variable, the names of the initialization block and the repository variable need to be enclosed in double quotes (“), separated by a period, and contained within parentheses. For example, to use the value of a dynamic repository variable named REGION contained in an initialization block named Region Security, this is an example of the proper syntax to use:

```
SalesSubjectArea.Customer.Region =

VALUEOF("Region Security"."REGION")
```

The names of session variables need to be preceded by NQ_SESSION, separated by a period, and contained within parentheses. If the variable name contains a space, enclose the name in double quotes (“). For example, to use the value of a session variable named REGION, this is an example of the proper syntax to use in an expression builder (filter):

```
"SalesSubjectArea"."Customer"."Region" = VALUEOF(NQ_SESSION.REGION)
```

NOTE: Although using initialization block names with session variables (just as with other repository variables) may work, you should use NQ_SESSION. NQ_SESSION acts like a wild card that matches all initialization block names. This allows the Oracle BI Administrator to change the structure of the initialization blocks in a localized manner without impacting reports.

System Functions

The system functions return values relating to the session.

User

Returns the user ID for the Oracle BI repository to which you are logged in.

Syntax:

```
USER ()
```

Database

Returns the name of the Oracle BI Presentation Catalog to which you are logged in.

Syntax:

```
DATABASE ()
```

Expressing Literals

A literal is a nonnull value corresponding to a given data type. Literals are typically constant values; that is, they are values that are taken literally *as is*, without changing them at all. A literal value has to comply with the data type it represents.

SQL provides mechanisms for expressing literals in SQL statements. This section describes how to express each type of literal in SQL.

Character Literals

A character literal represents a value of CHARACTER or VARCHAR data type. To express a character literal, surround the character string with single quotes ('). The length of the literal is determined by the number of characters between the single quotes.

Datetime Literals

The SQL 92 standard defines three kinds of typed datetime literals, in the following formats:

DATE 'yyyy-mm-dd'

TIME 'hh:mm:ss'

TIMESTAMP 'yyyy-mm-dd hh:mm:ss'

These formats are fixed and are not affected by the format specified in the NQSConfig.INI file for the parameters DATE_DISPLAY_FORMAT, TIME_DISPLAY_FORMAT, or DATE_TIME_DISPLAY_FORMAT. To express a typed datetime literal, use the keywords DATE, TIME, or TIMESTAMP followed by a datetime string enclosed in single quote marks. Two digits are required for all nonyear components even if the value is a single digit.

Numeric Literals

A numeric literal represents a value of a numeric data type (for example, INTEGER, DECIMAL, and FLOAT). To express a numeric literal, type the number as part of a SQL statement.

Do not surround numeric literals with single quotes; doing so expresses the literal as a character literal.

Integers

To express an integer constant as a literal, type the integer as part of a SQL statement (for example, in the SELECT list). The integer can be preceded with either a plus sign (+) or minus sign (-) to indicate the integer is a positive or negative number, respectively.

Decimal

To express a decimal literal, type a decimal number. The decimal can be preceded with either a plus sign (+) or minus sign (-) to indicate the integer is a positive or negative number, respectively.

Floating Point

To express floating point numbers as literal constants, type a decimal literal followed by the letter 'E' (either uppercase or lowercase) and followed by the plus sign (+) or the minus sign (-) to indicate a positive or negative exponent. No spaces are allowed between the integer, the letter 'E', and the sign of the exponent.

A Oracle BI Server Usage Tracking Data Descriptions and Using the Log File Method

The Oracle BI Server supports the collection of usage tracking data. When usage tracking is enabled, the Oracle BI Server collects usage tracking data for each query and writes statistics to a usage tracking log file or inserts them directly to a database table.

NOTE: It is strongly recommended that you use Direct Insert instead of writing to a log file.

For more information, refer to "Administering Usage Tracking" on page 219.

If you are upgrading from previous versions of Usage Tracking, see the usage tracking topics in *Oracle Business Intelligence Infrastructure Upgrade Guide*.

Create Table Scripts for Usage Tracking Data

The OracleBI\server\Schema folder contains the following Create Table scripts for Oracle, DB2, SQL Server and Teradata:

- SAACCT.Oracle.sql for Oracle
- SAACCT.DB2.sql for DB2
- SAACCT.MSSQL.sql for SQL Server
- SAACCT.Teradata.sql

The sample scripts set the usage tracking table name to S_NQ_ACCT. For sites that have Oracle BI Applications, this is the name used in the Oracle BI repository. Sites that build their own repositories may change the name of the usage tracking table. The table name must match the name used in the corresponding repository.

Loading Usage Tracking Tables with Log Files

It is strongly recommended that you load the Usage Tracking table using the Direct-Insert option. For those customers who have to load the Usage Tracking table from log files, Oracle BI provides the following sample JavaScript files located in the OracleBI\server\Scripts\Common subdirectory:

- sblAcctLoaderMSSQL for SQL Server
- sblAcctLoaderOCL.js for Oracle
- sblAcctLoaderADO.js for other database servers like DB2

These JavaScript files need to be modified for your environment. Comments at the beginning of these files should be used as a guide.

Before extracting usage tracking log file data, you need to create a table to store the data. For more information, refer to "Create Table Scripts for Usage Tracking Data" on page 405.

NOTE: UNIX installations cannot use these JavaScript files because UNIX operating systems typically do not support some of the advanced functionality used in these scripts. For UNIX installations, please ask your DBA to write scripts that will load your usage tracking log files.

Description of the Usage Tracking Data

Table 42 on page 406 describes each column in the usage tracking table. Where appropriate, the data type and length is also included.

Table 42. Usage Tracking Data

Column	Description
CACHE_IND_FLG	Default is N. Y indicates a cache hit for the query, N indicates a cache miss.
COMPILE_TIME_SEC	The time in seconds required to compile the query.
CUM_DB_TIME_SEC	The total amount of time in seconds that the Oracle BI Server waited for back-end physical databases on behalf of a logical query.
CUM_NUM_DB_ROW	The total number of rows returned by the back-end databases.
END_DT	The date the logical query was completed.
END_HOUR_MIN	The hour and minute the logical query was completed.
END_TS	The date and time the logical query finished. The start and end timestamps also reflect any time the query spent waiting for resources to become available.
ERROR_TEXT	Default is Null. Varchar(250) Error message from the back-end database. This column is only applicable if the SUCCESS_FLG is set to a value other than 0 (zero). Multiple messages will concatenate and will not be parsed by Oracle BI Server.
NODE_ID	Reserved for future use.
NUM_CACHE_HITS	Default is Null. Number(10,0). **NOTE:** For DB2, the data type and length is Decimal(10,0). Indicates the number of times existing cache was returned.
NUM_CACHE_INSERTED	Default is Null. Number(10,0). **NOTE:** For DB2, the data type and length is Decimal(10,0). Indicates the number of times query generated cache was returned.

Table 42. Usage Tracking Data

Column	Description
NUM_DB_QUERY	The number of queries submitted to back-end databases in order to satisfy the logical query request. For successful queries (SuccessFlag = 0) this number will be 1 or greater.
PRESENTATION_NAME	Default is Null. Varchar(128) Name of the Presentation Catalog in Oracle BI Presentation Services.
QUERY_SRC_CD	The source of the request, for example, Drill or Report.
QUERY_TEXT	The SQL submitted for the query.
REPOSITORY_NAME	The name of the repository the query accesses.
ROW_COUNT	The number of rows returned to the query client.
RUNAS_USER_NAME	Default is Null. Varchar(128) User Id of impersonated user. If the request is not run as an impersonated user, the value will be NULL.
SAW_DASHBOARD	Path of the dashboard. If the query was not submitted through an Interactive Dashboard, the value will be NULL.
SAW_DASHBOARD_PG	Default is Null. Varchar(150) Page name in the Interactive Dashboard. If the request is not a dashboard request, the value will be NULL.
SAW_SRC_PATH	The path name in the Oracle BI Presentation Catalog for the request.
START_DT	The date the logical query was submitted.
START_HOUR_MIN	The hour and minute the logical query was submitted.
START_TS	The date and time the logical query was submitted.
SUBJECT_AREA_NAME	The name of the business model being accessed.
SUCCESS_FLG	The completion status of the query: 0 - The query completed successfully with no errors. 1 - The query timed out. 2 - The query failed because row limits were exceeded. 3 - The query failed due to some other reason.
TOTAL_TIME_SEC	The time in seconds that the Oracle BI Server spent working on the query while the client waited for responses to its query requests.
USER_NAME	The name of the user who submitted the query.

B Oracle BI Server Authentication APIs

The Oracle BI Administrator requests that the developer create a dynamically loadable authentication module according to the Oracle BI Authenticator API specification. This section contains the APIs that the authenticator needs to implement. For more information about implementation, refer to "About Authenticating Users Using Initialization Blocks" on page 293.

All of the APIs take SAChar, which is defined as follows:

```
typedef wchar_t SAChar;
```

Other related type definitions are defined as follows:

```
typedef  unsigned int  SAUInt32;

enum

{

    SAAuthenticatorTrue,

    SAAuthenticatorFalse,

    SAAuthenticatorNotSupported

} SAReturnType;
```

A header file is provided for all of the types that will be used in the dynamically loadable authenticator. All of the following APIs need to be thread-safe:

- SAUInt32 SAAuthenticatorGetVersion()

  Description: This returns the current version of the authenticator. The version number is predefined as 1.

  - Arguments

    None

  - Return value

    The version number is predefined as 1.

- void SAAuthenticateFreeString(SAChar *p)

  This function is a utility function that frees up memory pointed by p.

  - Argument

Input	p	A pointer to a SAChar string

  - Return value

    None.

- void SAAuthenticateFreeStringArray(SAChar **pp)

  This function is a utility function. It frees up memory as pointed by pp.

  - Argument

Input	pp	A pointer to a SAChar array

  - Return value

    None

- SAReturnType SAAuthenticatorInit(const SAChar * pConfigParams, SAChar **ppErrorMessage)

  This function performs some basic initialization to the authentication module. It will be called from Oracle BI Server only once.

  - Arguments

Input	pConfigParams	This points to the parameters to be passed to the authenticator. The string is a concatenation of the Configuration parameter and decrypted version of the Encrypted parameter.
Output	ppErrorMessage	This contains the error message, if an error occurs. The authenticator writer needs to allocate memory for the error message.

  - Return value

    - If the initialization is successful, the return value is SAAuthenticatorTrue.
    - If the initialization fails, the return value is SAAuthenticatorFalse and the error message is stored in *ppErrorMessage.

    NOTE: The authenticator framework is responsible for freeing up the memory allocated for ppErrorMessage.

- SAReturnType SAAuthenticatorLogin(const SAChar * pUserID, const SAChar *pPassword, SAChar **ppErrorMessage)

  This function authenticates a user.

  - Arguments

Input	pUserID	User's login name
Input	pPassword	User's password
Output	ppErrorMessage	This contains the error message, if an error occurs. The authenticator writer needs to allocate the memory for the error message.

  - Return value

    - If the authentication is successful, the return value is SAAuthenticatorTrue.

- If the authentication is unsuccessful, the return value is SAAuthenticatorFalse and the error message is stored in **ppErrorMessage.

The authenticator framework is responsible for freeing up the memory allocated for ppErrorMessage. This API cannot return SAAuthenticatorNotSupported.

- bool SAAuthenticatorIsAValidUser(const SAChar * pUserID, SAChar **ppErrorMessage)

  This function determines whether the specified user id is a valid.

  - Arguments

Input	pUserID	User's login name
Output	ppErrorMessage	This contains the error message, if an error occurs. The authenticator writer needs to allocate the memory for the error message.

  - Return value
    - If the user id is valid, the return value is SAAuthenticatorTrue.
    - If the user id is invalid, the return value is SAAuthenticatorFalse and the error message is stored in *ppErrorMessage.

The authenticator framework is responsible for freeing up the memory allocated for ppErrorMessage.

NOTE: This API cannot return SAAuthenticatorNotSupported.

- SAReturnType SAAuthenticatorGetUserProps(const SAChar * pUserID, const SAChar **ppKeys, SAChar *** pppValues, SAChar **ppErrorMessage)

  This function contains a list of property values. When the key is GROUP and the user belongs to multiple groups, values need to be semicolon delimited.

  - Arguments

Input	pUserID	User's login name
Input	ppKeys	It points a null-terminated array of strings indicating which properties' values to return.
Output	pppValues	It points to a null-terminated array of strings which has the properties' values corresponding to ppKeys. The authenticator writer needs to allocate the memory for the array.
Output	ppErrorMessage	This contains the error message, if an error occurs. The authenticator writer needs to allocate the memory for the error message.

  - Return value
    - If the function call is successful, the return value is SAAuthenticatorTrue. The Oracle BI Server authenticator framework is responsible for freeing up the memory allocated for pppValues.

    - If function call fails, the return value is SAAuthenticatorFalse and the error message is stored in *ppErrorMessage. The Oracle BI Server authenticator framework is responsible for freeing up the memory allocated for ppErrorMessage.

  NOTE: This API cannot return SAAuthenticatorNotSupported.

- SAReturnType SAAuthenticatorGetAllGroups(SAChar *** pppGroups, SAChar **ppErrorMessage)

  This function gets a list of all groups.

  - Arguments

Output	pppGroups	It points to a null-terminated array of strings that are the groups to which the user belongs. The authenticator writer needs to allocate the memory for the array.
Output	ppErrorMessage	This contains the error message, if an error occurs. The authenticator writer needs to allocate the memory for the error message.

  - Return value
    - If the function call is successful, the return value is SAAuthenticatorTrue. The Oracle BI Server authenticator framework is responsible for freeing up the memory allocated for pppGroups.
    - If function call fails, the return value is SAAuthenticatorFalse and the error message is stored in *ppErrorMessage. The Oracle BI Server authenticator framework is responsible for freeing up the memory allocated for ppErrorMessage.

  NOTE: This API can return SAAuthenticatorNotSupported if it is not supported.

- void SAAuthenticatorShutdown()

  This function performs some cleanup up when Oracle BI Server shuts down. It will be called from Oracle BI Server only once.

  - Arguments

    None

  - Return value

    None

Index