Oracle BI 11*g* R1: Build Repositories

Activity Guide

D63514GC10
Edition 1.0
October 2010
D69307

ORACLE

Author

Jim Sarokin

Technical Contributors and Reviewers

Marla Azriel, Roger Bolsius, Bob Ertl, Alan Lee, Monica Moore, Kasturi Shekhar, Aneel Shenker, Scott Silbernick, Nick Tuson, Krishnan Viswanathan

This book was published using: Oracle Tutor

Table of Contents

Practices for Lesson 1

Lesson 1

Practices for Lesson 1

Practices Overview

There are no practices for this lesson.

Practices for Lesson 2: Repository Basics

Lesson 2

Practices for Lesson 2

Lesson Overview

In these practices, you will explore an existing Oracle Business Intelligence repository.

Practice 2-1: Exploring an Oracle BI Repository

Goal

To explore the three layers of an Oracle BI repository

Scenario

Before beginning the development of a repository, you use the Administration Tool to explore an existing repository to get a better understanding of its three layers and how the layers relate to one another, and to understand the link between physical data sources and the information presented in the Oracle BI user interface.

Time

25 minutes

Tasks

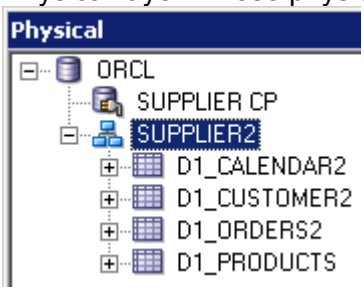
1. Copy the repository for this practice to the appropriate directory.
   a. Navigate to **D:\PracticeFiles**.
   b. Copy the **ClassStart.rpd** file.
   c. Paste the file in **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.
2. Start the Oracle BI Administration Tool and open the ClassStart repository in offline mode.
   a. Select **Start > Programs > Oracle Business Intelligence > BI Administration**.
   b. Select **File > Open > Offline**.
   c. In the **Open** dialog box, double-click **ClassStart.rpd**.
   d. Enter **welcome1** as the repository password and click **OK**. The repository opens in offline mode.
3. Examine the properties of the ORCL database object.
   a. In the Physical layer, double-click the **ORCL** database object to view its properties.
   b. Click the **General** tab.
   c. Notice that the database platform type is **Oracle 11g**.
   d. Click the **Features** tab. Each database comes with a set of features that determine the SQL that the Oracle BI Server will issue for this database. Features can have a Boolean value (on or off), integer value, or a string value. Scroll to the right to view the Value and Default columns. A check mark in the Default column indicates that the feature is supported by this database type and a check mark in the Value column indicates that the feature is enabled.
   e. Click the **Connection Pools** tab. This tab identifies all the connection pools associated with this database. In this example, there is only one connection pool, SUPPLIER CP.
   f. Click the **Display Folders** tab. Physical layer objects can be organized into display folders. When there are display folders in the Physical layer, they are listed here.
   g. Click **Cancel** to close the Properties dialog box.
4. Explore the properties of a connection pool object.
   a. In the Physical layer, expand the **ORCL** database object.
   b. Double-click the **SUPPLIER CP** connection pool object.

c. Notice that the call interface type for this connection pool is OCI 10g/11g and the data source name is ORCL. The call interface is the application programming interface (API) used to access the data source. Some databases may be accessed using native APIs, some using ODBC. In this example, the ORCL data source is accessed by the Oracle Call Interface (OCI) native API. The data source name, ORCL, is a tnsnames.ora entry.

d. Click **Cancel**.

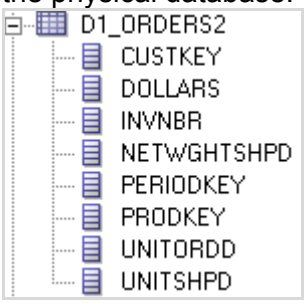
5. Examine the properties of a physical schema and its physical table objects.

a. Expand the **SUPPLIER2** schema folder to display the physical table objects in the Physical layer. These physical table objects map to tables in the physical database.

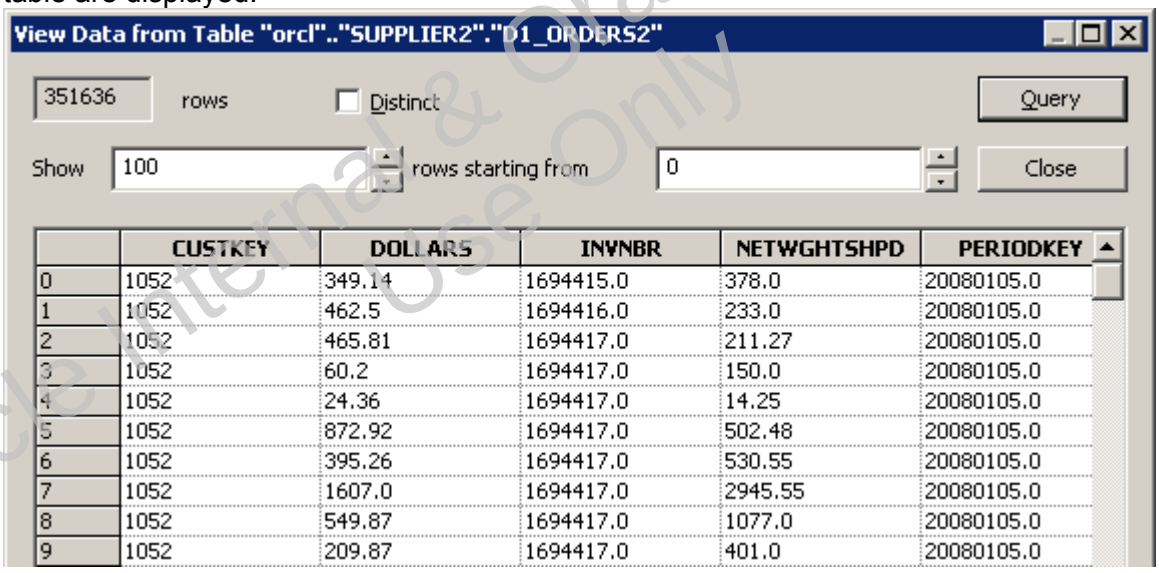


There are more tables in the physical database. The tables displayed here are the tables that have been imported into the Physical layer. You learn more about importing tables in the lesson titled “Building the Physical Layer of a Repository.”

b. Expand **D1_ORDERS2** to view the physical columns for this table. D1_ORDERS2 is the “fact” table in this business model. These columns correspond to the columns in the physical database.



c. Right-click **D1_ORDERS2** and select **View Data**. The first 100 rows of data for this table are displayed.



	CUSTKEY	DOLLARS	INVNBR	NETWGHTSHPD	PERIODKEY
0	1052	349.14	1694415.0	378.0	20080105.0
1	1052	462.5	1694416.0	233.0	20080105.0
2	1052	465.81	1694417.0	211.27	20080105.0
3	1052	60.2	1694417.0	150.0	20080105.0
4	1052	24.36	1694417.0	14.25	20080105.0
5	1052	872.92	1694417.0	502.48	20080105.0
6	1052	395.26	1694417.0	530.55	20080105.0
7	1052	1607.0	1694417.0	2945.55	20080105.0
8	1052	549.87	1694417.0	1077.0	20080105.0
9	1052	209.87	1694417.0	401.0	20080105.0

d. Click **Close**.

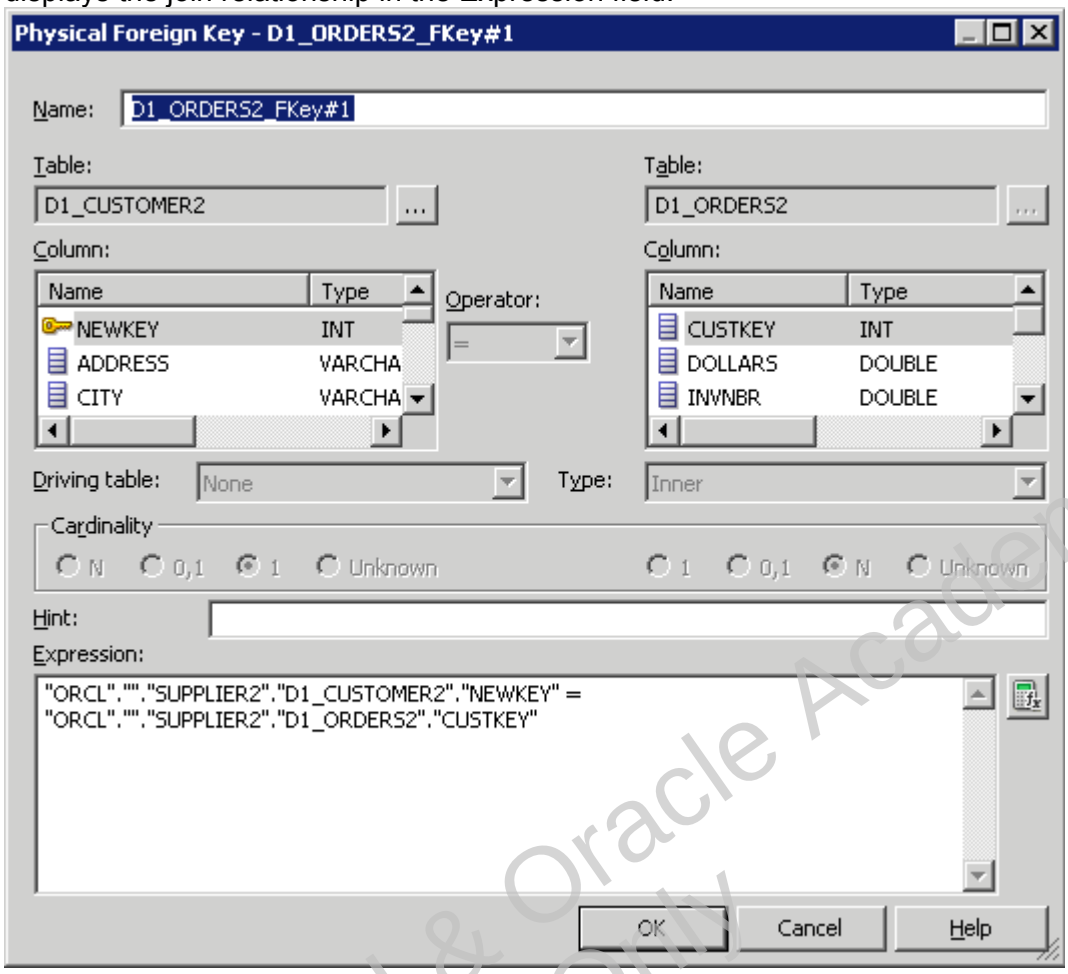
e. Double-click the **D1_ORDERS2** table object to view its properties.

f. Click the **Columns** tab to view the columns in this table. This is another way to create, view, and modify physical columns.

g. Click the **Foreign Keys** tab.

h. Notice that all three tables have join relationships with D1_ORDERS2: D1_CUSTOMER2, D1_CALENDAR2, and D1_PRODUCTS.

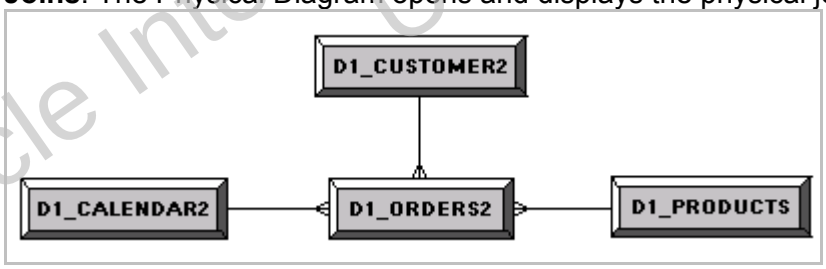
i. Double-click one of the foreign keys. The Physical Foreign Key dialog box opens and displays the join relationship in the Expression field.



j. Click **Cancel** to close the Physical Foreign Key dialog box.

k. Click **Cancel** to close the Physical Table dialog box.

l. Right-click **D1_ORDERS2** and select **Physical Diagram > Object(s) and Direct Joins**. The Physical Diagram opens and displays the physical join relationships.

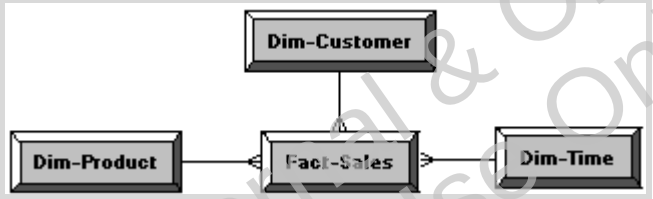


m. Double-click the **connector** between D1_CUSTOMER2 and D1_ORDERS2. The Physical Foreign Key dialog box opens and displays the join relationship in the Expression field. This is another way to view, build, and modify joins between tables in the Physical layer.

n. Click **Cancel**.

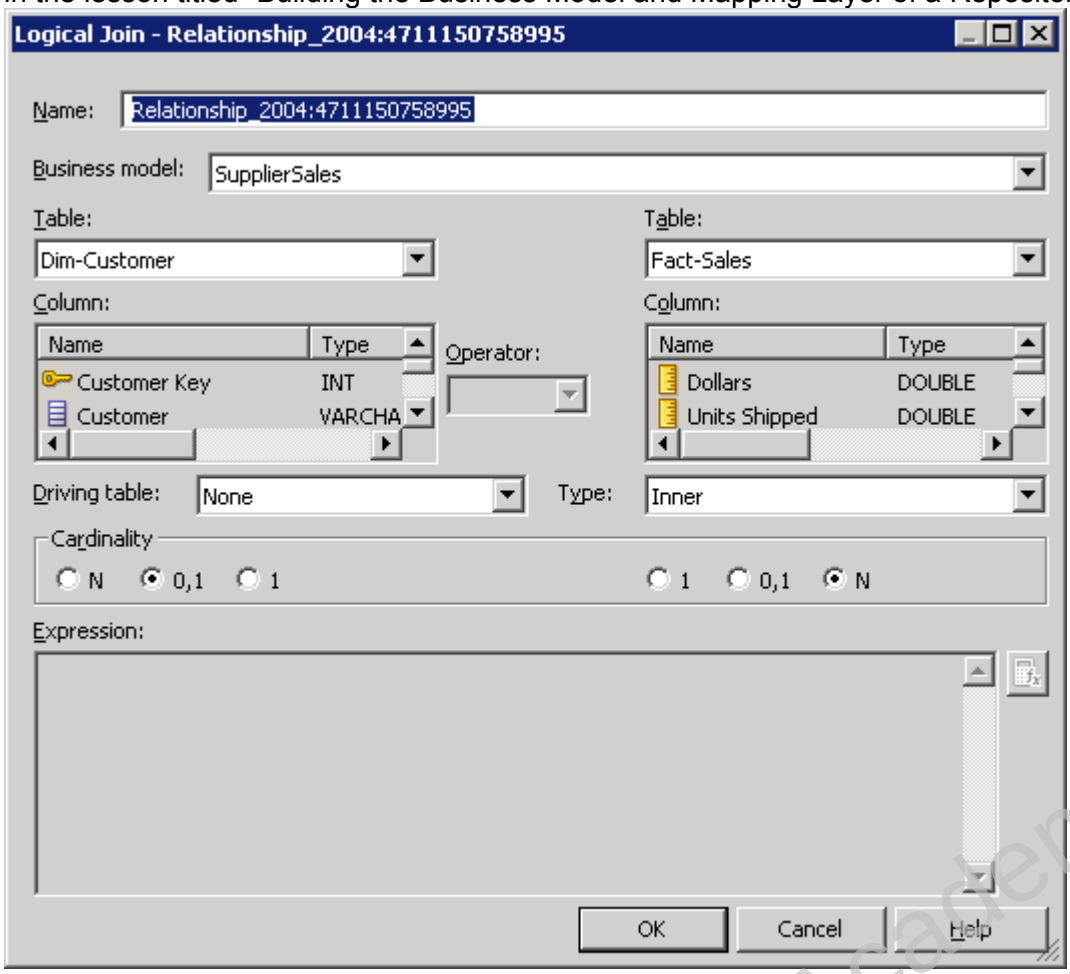
o. Close the Physical Diagram.

6. Examine the properties of a physical column object.
   a. Expand the **D1_CUSTOMER2** table object.
   b. Double-click the **Address** physical column to view the properties.
   c. Click **Cancel**.
7. Examine the properties of a logical table in the SupplierSales business model.
   a. If necessary, expand the **SupplierSales** business model in the Business Model and Mapping layer.
   b. Notice that there are four logical table objects in the SupplierSales business model: Fact-Sales, Dim-Time, Dim-Customer, and Dim-Product.
   c. Expand the **Fact-Sales** logical table to view the logical columns for this table. Fact-Sales is the logical "fact" table in this business model. These logical columns map to columns in the Physical layer.
   d. Double-click the **Fact-Sales** logical table object.
   e. Click the **General** tab. Notice that the logical columns and their corresponding properties are listed. On this tab you can change the name of the logical table, reorder the columns, and add, edit, or remove a column.
   f. Click the **Sources** tab. The source for this logical table is the D1_ORDERS2 table that you explored in the Physical layer. In a more complex business model, there may be many physical sources for a logical table.
   g. Click the **Keys** tab. Typically, no keys are defined for a logical fact table.
   h. Click the **Foreign Keys** tab. Foreign key joins are typically not used in the Business Model and Mapping layer. All joins in the Business Model and Mapping layer are logical joins that do not require primary key – foreign key relationships.
   i. Click **Cancel** to close the Logical Table properties dialog box.
   j. Right-click **Fact-Sales** and select **Business Model Diagram > Whole Diagram**. The Business Model Diagram opens and displays the logical join relationships.



   k. Double-click the **connector** between Dim-Customer and Fact-Sales. The Logical Join dialog box opens. Notice that there is no join expression in the Expression field and that there is a one-to-many relationship between the Dim-Customer logical dimension table and the Fact-Sales logical fact table. You learn more about complex logical joins

in the lesson titled “Building the Business Model and Mapping Layer of a Repository.”



l. Click **Cancel** to close the Logical Join dialog box.

m. Close the Business Model Diagram.

8. Examine the logical table source for the Fact-Sales logical table.

   a. Expand **Fact-Sales** > **Sources** to display the D1_ORDERS2 logical table source for this logical table. In this example, there is only one logical table source. However, it is possible to have many logical table sources for a single logical table.

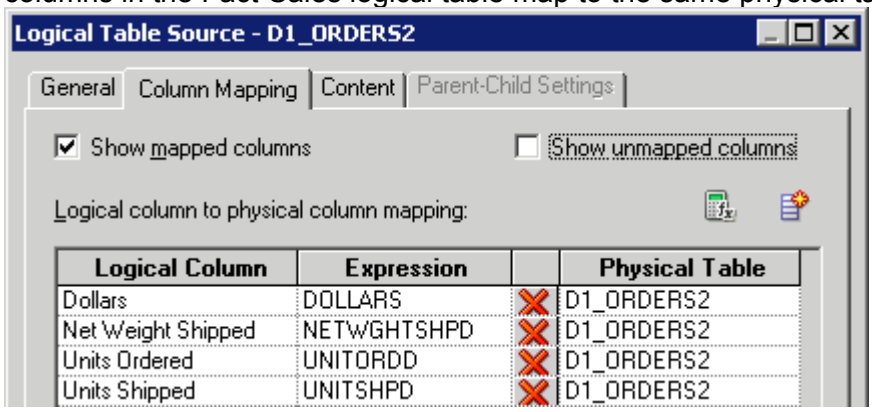
   b. Double-click the **D1_ORDERS2** logical table source to view the properties.

   c. Click the **General** tab.

   d. Notice that the D1_ORDERS2 logical table source maps to the D1_ORDERS2 physical table.

   e. Click the **Column Mapping** tab. This tab shows the mappings between the logical columns and physical columns.

f. If necessary, scroll to the right to view the Physical Table column. Notice that all columns in the Fact-Sales logical table map to the same physical table.



g. Notice that some logical columns have names that are different from the physical columns to which they map. This is because the column names were changed in the Business Model and Mapping layer. You learn how to rename logical columns in the lesson titled “Building the Business Model and Mapping Layer of a Repository.”

h. Click the **Content** tab. Currently, there is no information on the tab. You learn how to use this tab to identify aggregation content and fragmentation content in the lesson titled “Using Aggregates.”

i. Click **Cancel** to close the Logical Table Source dialog box.

9. Examine the properties of a logical column in the Fact-Sales logical table.

a. Double-click the **Dollars** logical column to open the properties window. Dollars is a measure in this business model.

b. Click the **General** tab. This tab provides general information about the column, such as the column name, the table it belongs to, and a description of the column.

c. Click the **Column Source** tab. This tab provides information about the physical table and physical column that the logical column maps to, or whether the logical column is derived from other existing logical columns.

d. Double-click the **D1_ORDERS2** logical table source in this dialog box. Notice that this is another way to access the Logical Table Source properties dialog box.

e. Click **Cancel** to close the Logical Table Source properties dialog box.

f. Click the **Aggregation** tab. Notice that the default aggregation rule is set to Sum. It is common to apply aggregation rules to measures in business models. Open the Default aggregation rule drop-down list to see the other available aggregation rules.

g. Make sure that the Sum aggregation rule is still selected and click **Cancel** to close the Logical Column dialog box.

10. Examine the properties of the SupplierSales presentation catalog object.

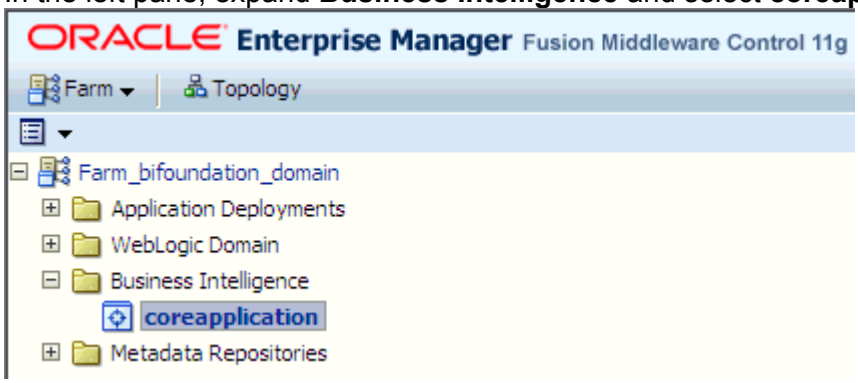
a. In the Presentation layer, double-click the **SupplierSales** subject area to open the Subject Area properties dialog box.

b. Click the **General** tab. Notice that the subject area name is the same as the business model name. This is because the subject area was created by dragging the business model from the Business Model and Mapping layer to the Presentation layer. If desired, you could use this tab to change the name of the subject area. The subject area and its description appear in the Analysis Editor user interface. Many subject areas can map to a single business model, but each subject area can map to only one business model.

c. Click **Permissions** to open the Permissions dialog box. This shows the permissions for all users and application roles in the repository. Currently, no permissions are defined for users or application groups, so the default is to give everyone read permission for this subject area. You learn more about setting object permissions in the lesson titled "Security."
d. Click **Cancel** to close the Permissions dialog box.
e. Click the **Presentation Tables** tab to display a list of presentation tables. You can use this tab to add, remove, edit, or change the display order of the presentation tables in the subject area.
f. Click the **Aliases** tab. If you change the name of a subject area, the tool automatically creates an alias based on the previous name. You can use this tab to specify or delete an alias.
g. Click **Cancel** to close the Subject Area dialog box.

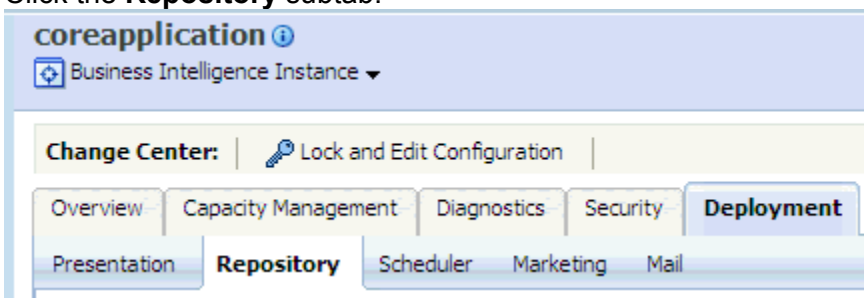
11. Examine the properties of a presentation table in the SupplierSales subject area.
    a. In the Presentation layer, expand **SupplierSales** to view the presentation tables.
    b. Double-click the **Fact-Sales** table to view the properties.
    c. Click the **General** tab. You can use this tab to change the name of the presentation table. A description would appear as a "tool tip" in the Oracle BI user interface.
    d. Click the **Columns** tab to see a list of columns and their mappings in the Fact-Sales presentation table. You can use this tab to add, remove, edit, or change the display order of the presentation columns.
    e. Double-click **Dollars** to open the Presentation Column properties dialog box. This is one method for viewing and modifying presentation column properties.
    f. Click **Cancel** to close the Presentation Column properties dialog box.
    g. Click the **Hierarchies** tab. Currently, no hierarchies are defined for this presentation table. You learn about presentation hierarchies in the lesson titled "Working with Logical Dimensions."
    h. Click **Cancel** to close the Presentation Table properties dialog box.
12. Examine the properties of a presentation column in the Fact-Sales table.
    a. Expand the **Fact-Sales** presentation table.
    b. Double-click **Dollars** to open the Presentation Column properties dialog box. Notice that this is the same dialog box that you saw earlier. This is another method for viewing and modifying presentation column properties.
    c. Click **Cancel** to close the Presentation Column properties dialog box.
    d. Click **File > Close** to close the repository without saving any changes.
    e. Leave the Administration Tool open.
13. Use Fusion Middleware Control Enterprise Manager to upload the repository.
    a. Open **Internet Explorer** and enter the following URL:
    `http://localhost:7001/em`
    b. Enter **weblogic** as the username and **welcome1** as the password.

c. In the left pane, expand **Business Intelligence** and select **coreapplication**.



d. In the right pane, click the **Deployment** tab.

e. Click the **Repository** subtab.



f. Click **Lock and Edit Configuration**.

g. Click **Close** when you receive the confirmation message "Lock and Edit configuration – Completed Successfully."

h. In the Upload BI Server Repository section, click **Browse** to open the Choose file dialog box.

i. By default, the Choose file dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.

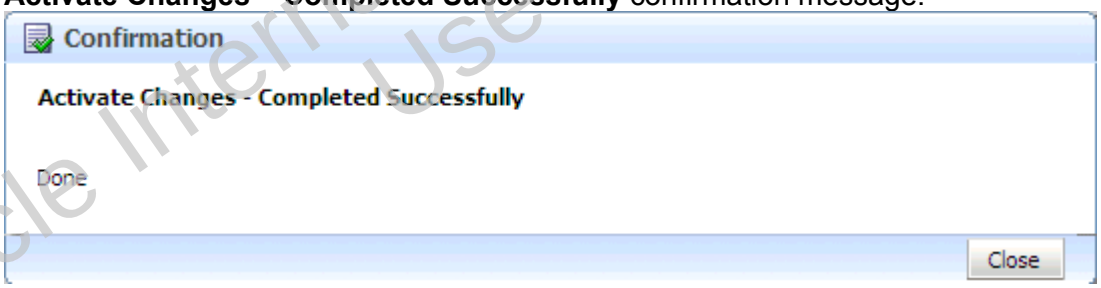
j. Select **ClassStart.rpd** and click **Open**.

k. Enter **welcome1** in the Repository Password and Confirm Password fields.

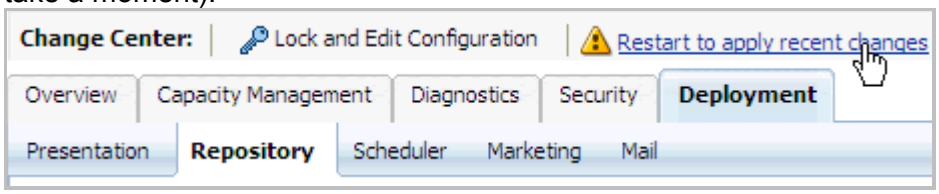
l. Click **Apply**. Notice that Default RPD now displays ClassStart with an extension (for example, ClassStart_BI0001).

m. Click **Activate Changes**.

n. Allow Active Changes processing to complete. Click **Close** when you receive the **Activate Changes – Completed Successfully** confirmation message.

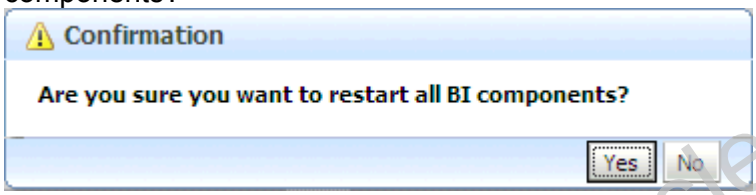


o. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).

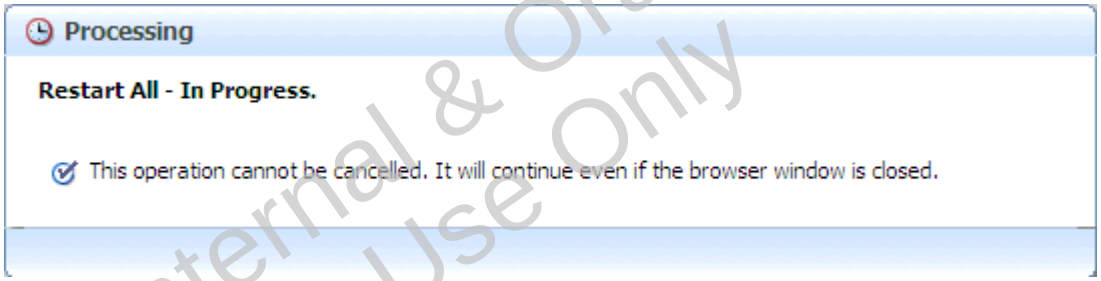


p. On the Overview page, click **Restart**.

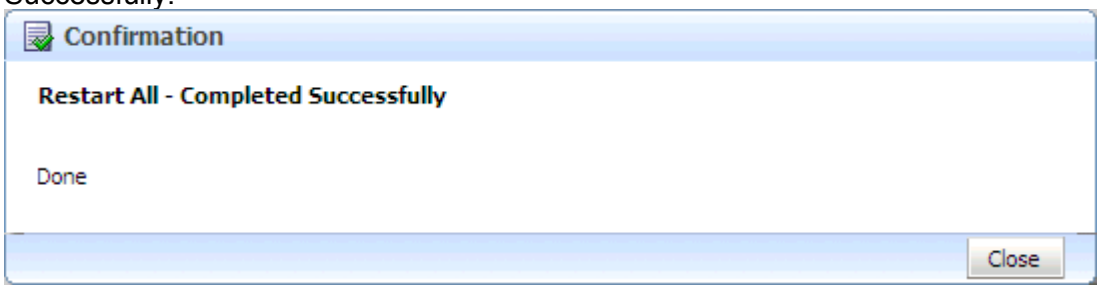
q. Click **Yes** when you receive the message "Are you sure you want to restart all BI components?"



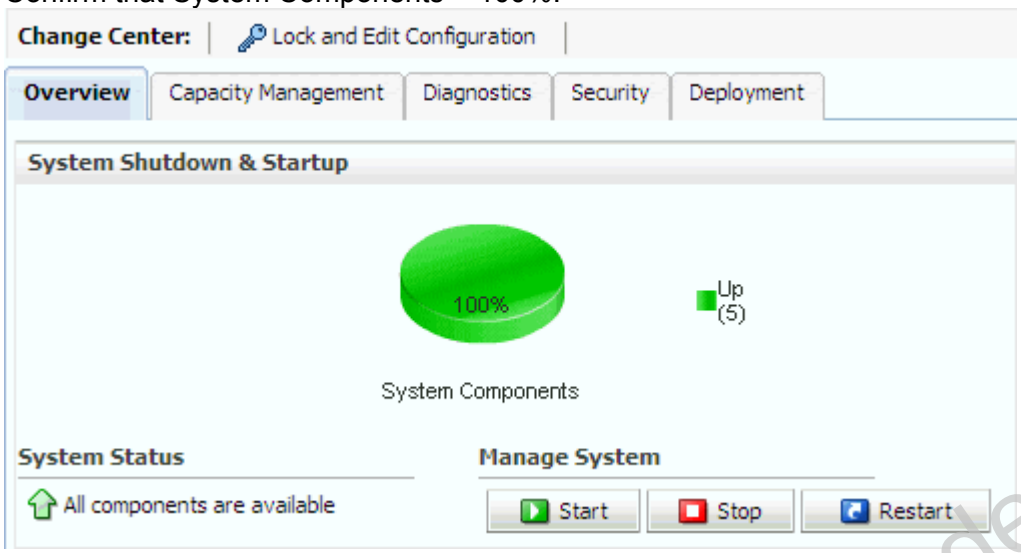
r. Allow the Restart All – In Progess processing to complete. This may take a few moments.



s. Click **Close** when you receive the confirmation message “Restart All – Completed Successfully.”



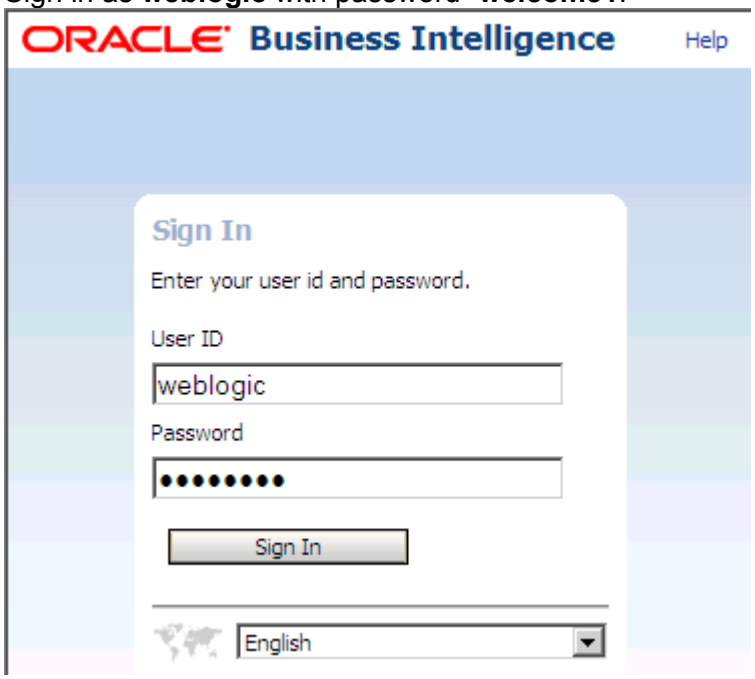
t. Confirm that System Components = 100%.



u. Leave Enterprise Manager open.

14. Examine the relationship between the columns in the Presentation layer and the columns displayed in Oracle BI Analysis Editor.
    a. Return to the Administration Tool, which should still be open.
    b. Select **File > Open > Offline**.
    c. Double-click **ClassStart.rpd** to open it in offline mode.
    d. Enter **welcome1** as the repository password and click OK. The ClassStart repository opens in offline mode.
    e. Return to Internet Explorer, which should still be open with FMW Enterprise Manager.
    f. On a new tab, enter the following URL:
    http://localhost:9704/analytics

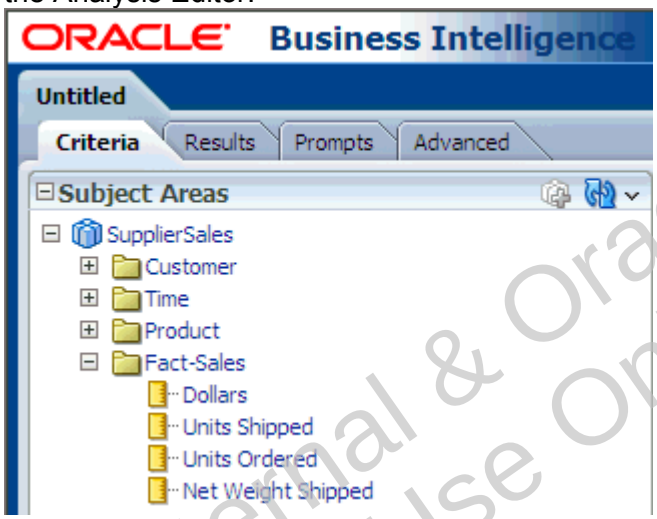
g. Sign in as **weblogic** with password **welcome1**.

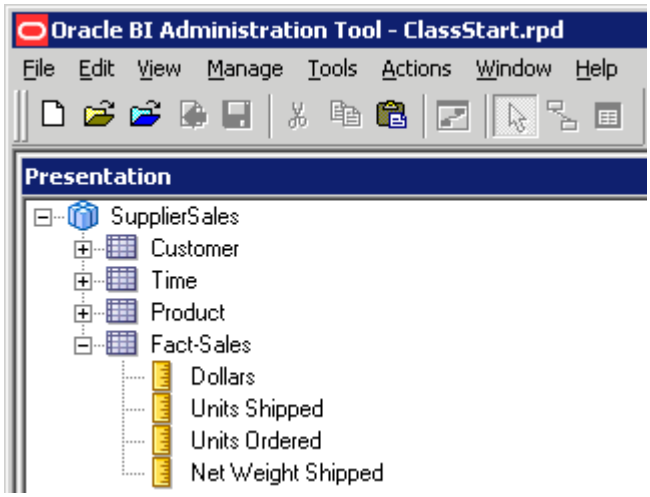


h. In the Create section on the left, click **Analysis**.

i. Click the **SupplierSales** subject area to open the Analysis Editor.

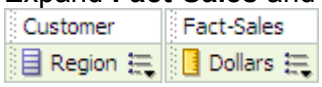
j. Size the windows of Internet Explorer and the Administration Tool so that you can see the two applications side by side. Notice that the SupplierSales subject area in the Presentation layer of the repository corresponds to the SupplierSales subject area in the Analysis Editor.





The key point to understand is this: What you see in the Analysis Editor is driven by what is defined in the Presentation layer of the repository in the Administration Tool.

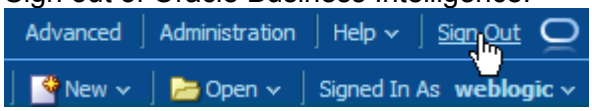
15. Create a simple request.
    a. In the Analysis Editor, expand the **Customer** table and double-click the **Region** column to add it to the analysis criteria in the right pane.
    b. Expand **Fact-Sales** and select the **Dollars** column to add it to the request.



    c. Click the **Results** tab. The table displays the total dollars for each region.

Region	Dollars
Central	$13,423,387
East	$25,460,351
West	$25,728,722

    d. Sign out of Oracle Business Intelligence.



    e. Click **OK** when you receive the message "Are you sure you want to navigate away from this page…".
    f. Leave the browser open.
    g. In the Administration Tool, close the **ClassStart** repository without saving.
    h. Leave the Administration Tool open.

Practices for Lesson 3: Building the Physical Layer of a Repository

Lesson 3

Practices for Lesson 3

Lesson Overview

In these practices, you will build the physical layer of an Oracle Business Intelligence repository.

Practice 3-1: ABC Business Scenario

Goal

To read the business scenario for the fictitious company used throughout the course

Time

10 minutes

Background

ABC sells food and nonfood items to restaurant chain customers throughout the United States. ABC's product line includes food, condiments, cookware, clothing, and other miscellaneous restaurant supplies.

ABC employees currently rely on reports generated by their IT department to analyze sales and shipment data. These reports are generated monthly and, therefore, contain static information. ABC is looking for a way to generate dynamic, interactive reports to analyze this data in order to effectively manage its orders, monitor sales performance, and increase overall customer satisfaction. ABC recognizes the importance of a business intelligence solution that will provide its employees with the data and tools that they need to query large data sets, generate reports, analyze data, identify trends, and monitor business performance.

To achieve these goals, ABC has decided to implement an Oracle Business Intelligence (BI) solution. The solution will enable its sales executives and operations managers to answer business questions that are important to running the part of the business for which they are responsible. The desired system will provide timely, up-to-date sales order data and provide sales and shipment performance monitoring, which will allow ABC to improve its customer service efforts. ABC also has the vision of making this data directly available to customers, so that they can analyze their purchase history better and compare it with what is being purchased nationally or regionally. This initiative is consistent with ABC's commitment to deliver a high-quality business intelligence solution that will set it apart from its competitors, and result in additional sales and revenue.

Business Requirements

You are a consultant hired by ABC to implement a business intelligence solution. From your initial interviews with the managers at ABC, you obtain the following information about ABC:

- Managers want to use their own information to ask and answer questions about the sales history of their products, the buying history of their customers, the order fulfillment performance of their operations group, and the selling performance of the sales force.
- The company has little experience in data analysis and does not expect to hire any data analysts in the future. Therefore, the managers want to be able to ask relevant questions and analyze the results themselves using an intuitive user interface.
- The database has approximately 350,000 invoice-level records that span the period from January 2, 2008 to April 21, 2009.

To make this kind of information widely available within the enterprise, the business intelligence solution must have a structure that is consistent with the way employees think about the business. From further interviews and an examination of the existing sample reports presenting invoice-based data, you determine the following:

- ABC employees think about their business in terms of sales, products and product hierarchies, time periods, and relationships between customers and the sales force.

- ABC employees measure product data at five levels. The product levels from the top level to the bottom (most detailed) level are:
  - Total
  - Type
  - Subtype
  - Generic Product
  - Specific Product
- ABC employees measure sales organization data at five levels. These levels mirror organizational management responsibilities. They are (in descending order):
  - Total
  - Region
  - District
  - Sales Representative
  - Customer
- ABC employees measure time data at five levels:
  - Total time
  - Year
  - Quarter
  - Month
  - Day
- Many existing reports contain data presented at these levels. An example of a report containing product data levels is:
  - Total Product sales
  - Product Type sales: Cheese
  - Product Subtype sales: American Cheese
  - Generic Product sales: American Cheese Slices
  - Specific Product sales: 2 Pack American Cheese Slices 16 Slices

In addition to these hierarchies, ABC wants to group and analyze customers using geographical attributes such as region, city, and state. They also want to analyze products by characteristics such as diet type and suppliers.

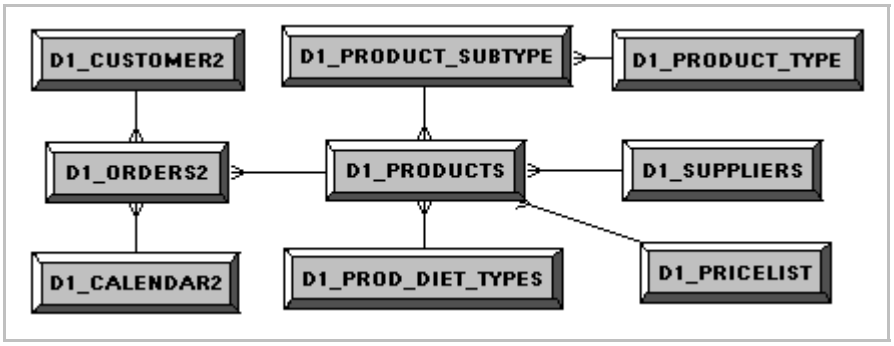
The most common measures used in reports are dollar sales, units ordered, units shipped, and net weight shipped. ABC would like to analyze these measures at all levels of the hierarchies, by the important product and customer attributes, by the various time periods, and be able to compare performance with previous years.

Source Data

By interviewing ABC's database administrators and examining the documents that they provide, you learn that the source data resides in an invoice system on an Oracle database. The core data containing the business measures is stored in an invoice (orders) table.

In addition to the invoice information, there are various tables providing information about customers, the product lines, the time periods, and the relationship between customers and the company's sales organization.

The relevant data exists in the following tables, which are part of the invoice schema:



The following are examples of data in some of the tables.

D1_PRODUCTS

PRODUCTKEY	PGCODE	GENERICDESCRIPTION	SPECIFICDESCRIPTIN	SUPPLIERCODE
1130	130 - Pepper; Pc	Black Pepper	100 Ct Single-serving Black Pepper 1/8 tsp	2300
1131	131 - Boxes	Take-out Containers	"100 Ct Foam Containers 6""x6""x3"""	1400
1132	132 - Checks	Guest Checks	10 Pak Guest Check Tablet w/Tearoff Rec...	1400
1133	133 - Wraps	Shrink Wrap	1000' Deli-style Shrink Wrap	2100
1134	134 - Containers	Refrigerator Jars	Refrigerator Jars	2200
1135	135 - Disposable Papers	Placemats	500 Ct Disposable Placemats	1400
1136	136 - Sanitary	Liquid Disinfectant	Liquid Disinfectant	1400
1137	137 - Bags	Take-out bags	Take-out Bags 200-pk	1400
1138	138 - Can Liners	Can Liners	Can Liners 2 qt 100-pak	1400
1139	139 - Dishes	Plain Dinnerware	20 Piece Setting for 4	2100
1140	140 - Cutlery/Flatware	Flatware Fancy	20 Piece Service for 5	1400

D1_CUSTOMER2

NEWKEY	NAME	ADDRESS	PHONE	CITY	STATE
1111	Midway Cafe	400 Crescent Ct	(214) 871-3240	Dallas	TX
1112	Penn Brewery	1717 N Akard St	(214) 720-5291	Dallas	TX
1113	Soulfood Seafood	1706 Commerce St	(214) 653-1900	Dallas	TX
1114	Caesar's Frozen Custard	6930 N Mesa St	(915) 833-4100	El Paso	TX
1115	Cristina's	6951 N Mesa St	(915) 585-6994	El Paso	TX
1116	Johnnie's Charcoal Broiler	5151 Fairbanks Dr # B	(915) 757-9000	El Paso	TX
1117	Aibonitos Restaurant	5480 Fm 1960 Rd W	(281) 893-2281	Houston	TX
1118	Mayflower Cuisinier	14025 Memorial Dr	(281) 293-7676	Houston	TX
1119	Peter's Pub	14715 Hempstead Rd	(713) 896-4000	Houston	TX
1120	El Burrito	4109 Fredericksburg Rd	(210) 732-3571	San Antonio	TX

D1_CALENDAR2

YYYYMMDD	FULL_GREGORIAN_DTE	MONTH_IN_YEAR	DAY_IN_MONTH	DAY_NAME	DAY_IN_YEAR
20080406	06-APR-08	4	6	Monday	96
20080407	07-APR-08	4	7	Tuesday	97
20080408	08-APR-08	4	8	Wednesday	98
20080409	09-APR-08	4	9	Thursday	99
20080410	10-APR-08	4	10	Friday	100
20080411	11-APR-08	4	11	Saturday	101
20080412	12-APR-08	4	12	Sunday	102
20080413	13-APR-08	4	13	Monday	103
20080414	14-APR-08	4	14	Tuesday	104
20080415	15-APR-08	4	15	Wednesday	105

D1_ORDERS2

PRODKEY	PERIODKEY	INVNBR	CUSTKEY	UNITSHPD	UNITORDD	DOLLARS	NETWGHTSHPD
1009	20080105	1694415	1052	14	14	349.14	378
1143	20080105	1694416	1052	21	21	462.5	233
1126	20080105	1694417	1052	33	33	465.81	211.27
1097	20080105	1694417	1052	6	6	60.2	150
1107	20080105	1694417	1052	1	1	24.36	14.25
1108	20080105	1694417	1052	24	24	872.92	502.48
1118	20080105	1694417	1052	31	31	395.26	530.55
1040	20080105	1694417	1052	100	100	1607	2945.55
1042	20080105	1694417	1052	44	50	549.87	1077
1043	20080105	1694417	1052	14	14	209.87	401

Training Objective

The primary objective of this training is to build the metadata and administer Oracle BI Server to support the business requirements of ABC. The metadata will allow ABC employees to build interactive reports and dashboards that they can use to better analyze, monitor, and manage their business, and improve overall customer satisfaction.

The recommended strategy for building metadata is to use an iterative approach. You begin by building a relatively simple repository:

- Minimize the number of source tables.
- Expose only stored measures with simple aggregation rules.
- Use the query log to check query results.
- Create presentation objects and test with Oracle BI Analysis Editor.

After the initial repository is built and tested, you expand the business model:

- Import additional physical tables needed to support the business model.
- Add calculated measures that involve operations on existing columns.
- Add more complex calculated measures (for example, level-based measures and share measures).
- Add time series calculations (for example, percentage change in a measure compared to that in the same period in the previous year).
- Add security information.
- Add aggregate table data sources to improve performance.
- Localize Oracle BI metadata and data.
- Configure many-to-many relationships.
- Configure implicit fact columns.

In the lessons that follow, you follow this strategy to build the business model to support the business requirements of ABC. At first, the repository metadata includes only the basic set of physical tables needed to support the initial business model. You add more tables and complexity in future iterations.

Practice 3-2: Gathering Information to Build an Initial Business Model

Goal

To analyze the business requirements to begin building the business model

Scenario

Before you begin building the business model, you need to gather and analyze the business requirements of the ABC company. In this practice, you use the information provided in the ABC document that you read in the previous practice to determine the structure of the initial business model.

Time

10 minutes

Tasks:

Use the information in the ABC document to help you determine the following information, which you will need to implement the initial business model.

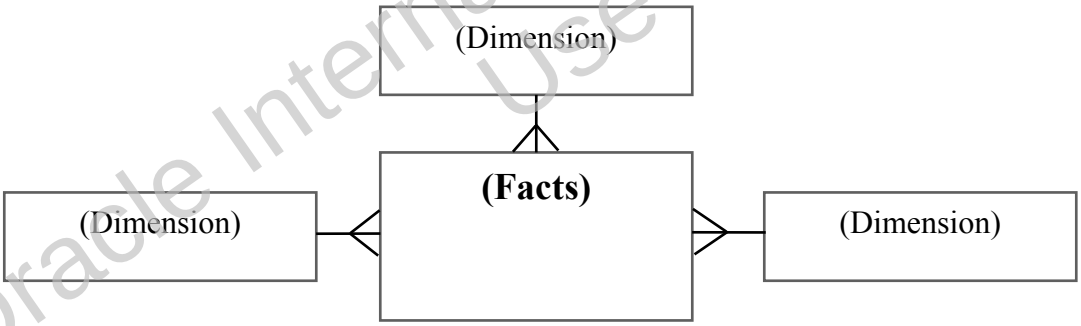
1. What measures (facts) does ABC want to report on?

2. What hierarchies can you identify?

3. Against which attributes (dimensions) does ABC want to analyze its facts?

4. By which geographical attributes does ABC want to analyze data?

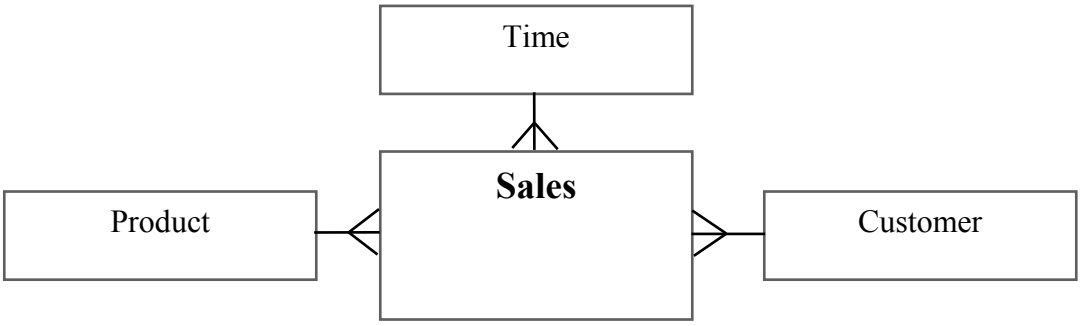
5. Complete the following diagram with your identified facts and dimensions:



Solutions 3-2: Gathering Information to Build an Initial Business Model

Answers

1. What measures (facts) does ABC want to report on?
   Dollar sales, units ordered, units shipped, net weight shipped
2. What hierarchies can you identify?
   Product, Time, Customer
3. Against which attributes (dimensions) does ABC want to analyze its facts?
   Product, Time, Customer
4. By which geographical attributes does ABC want to analyze data?
   Region, city, state
5. Complete the following diagram with your identified facts and dimensions:



Practice 3-3: Creating a Repository and Importing a Data Source

Goal

To create a new repository and import the table schema from an external data source

Scenario

You use the Import Wizard of the Administration Tool to create a new repository and import tables from the SUPPLIER2 schema into the Physical layer of the repository.

Outcome

You have a new repository file, ABC.rpd, which contains the D1_CALENDAR2, D1_CUSTOMER2, D1_ORDERS2, and D1_PRODUCTS tables in the Physical layer.

Time

15 minutes

Tasks

1. Use the Import Wizard to create a new ABC repository file.
   a. Return to the Administration Tool, which should still be open. If not, select **Start > Programs > Oracle Business Intelligence > Administration**.
   b. Select **File > New Repository** to open the Import Wizard. The Import Wizard guides you through the steps to create a new repository and import metadata.
   c. Enter **ABC** in the Name field.
   d. In the **Location** field, accept the default location to store the repository.
   e. For **Import Metadata**, accept the default selection: **Yes**. When Yes is selected, the Import Wizard continues with windows for importing metadata. When No is selected, an empty repository is saved to the selected location.
   f. Enter **welcome1** as the repository password.
   g. Reenter the password.
   h. Click **Next** to open the new ABC repository and the Select Data Source window.
2. Use the Import Wizard to import metadata.
   a. Select **OCI 10g/11g** from the Connection Type drop-down list. The window displays connection fields based on the connection type that you selected.
   b. Enter the following connection information:
      Data Source Name: **orcl**
      User Name: **supplier2**
      Password: **supplier2**
   c. Click **Next** to continue to the Select Metadata Types window.
   d. In the Select Metadata Types window, accept the defaults: **Tables**, **Keys**, and **Foreign Keys**. The check boxes allow you to select the information to import. You can import tables, keys, foreign keys, system tables, aliases, synonyms, and views. As a general rule, import only those objects that are needed to support your business model. If you do import extra objects at this point, you can delete them later if you determine that they do not support your business model.
   e. Click **Next** to open the Select Metadata Objects window.
   f. In the Data source view pane, expand the **SUPPLIER2** schema folder.

g. Scroll to view all the tables in the SUPPLIER2 schema.

h. Select the **D1_CALENDAR2** table. This automatically deselects any higher-level container objects in the tree.

i. Press and hold **Ctrl** and select the remaining three tables to import to the Physical layer to build ABC's initial business model: **D1_CUSTOMER2**, **D1_ORDERS2**, and **D1_PRODUCTS**.

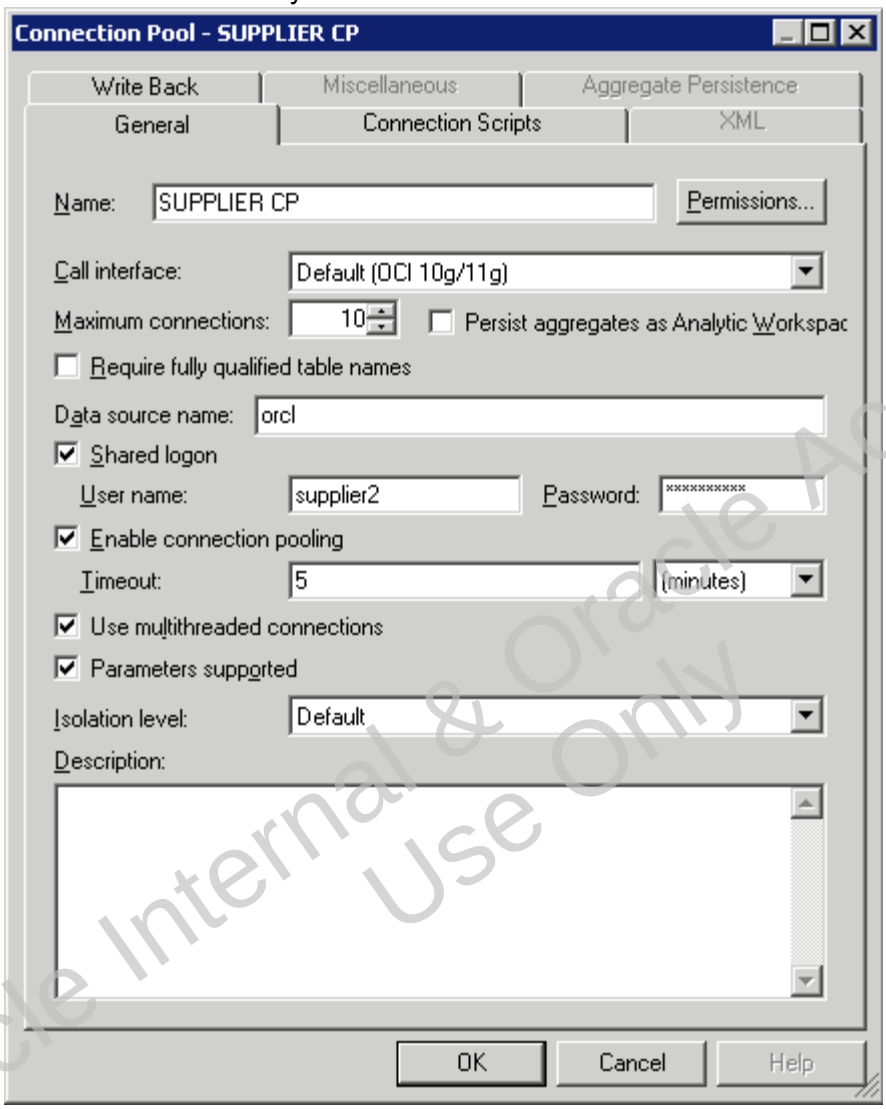
j. Scroll to ensure that no higher-level container objects are selected, and that only the four tables for import are selected.

k. Click the **Import Selected** button to move the metadata objects to the Repository View pane. The Connection Pool dialog box opens automatically.

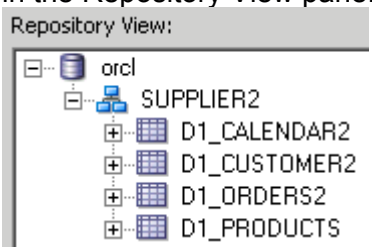
l. Change the Connection Pool name to **SUPPLIER CP**.

m. Ensure that the call interface is set to **Default (OCI 10g/11g)**.

n. Ensure that the data source name is **orcl**. The username and password fields are automatically populated with **supplier2**. The data source name, orcl, is the same as the tnsnames.ora entry for this Oracle database instance.



o. Click **OK**. The Connection Pool dialog box closes and the metadata objects are visible in the Repository View pane. Expand SUPPLIER2 to see the objects.

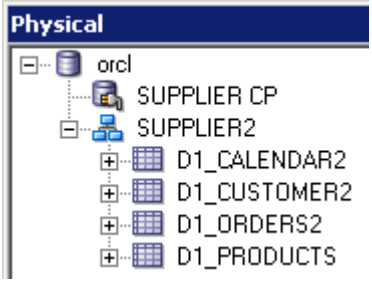


p. Click **Finish** to complete the import process. The ABC repository opens.

q. Notice that the metadata is imported into the Physical layer of the ABC repository and the orcl database object appears in the Physical layer. Recall that every repository contains three layers. The Physical layer is where information about the physical data sources is stored. The Business Model and Mapping layer is where measurements and terms used in business are mapped to the physical data sources. The Presentation layer is where the business model is customized for presentation to the user. You can work on each layer at any stage in creating a repository, but the typical order is to create the Physical layer first, then the Business Model and Mapping layer, and then the Presentation layer.

r. To display the tables, expand the **orcl** database object and then expand the **SUPPLIER2** schema folder.

s. Ensure that the four tables are imported successfully and that the connection pool name is changed to SUPPLIER CP.



Recall that D1_CALENDAR2 contains time information, D1_CUSTOMER2 contains customer information, and D1_PRODUCTS contains product information for the business model. D1_ORDERS2 contains the invoice-level information needed to create the measures in the business model. D1_CALENDAR2, D1_CUSTOMER2, and D1_PRODUCTS are the dimension tables and D1_ORDERS2 is a fact table. In the next practice, you create keys and joins so that the four tables form a star schema.

3. Examine the connection pool for the orcl data source.

a. Double-click the **SUPPLIER CP** connection pool object. Recall that you provided this name for the connection pool during the import process. Connection pools regulate access to the data source. Every data source must have at least one connection pool. A connection pool provides connections for multiple concurrent data source requests (queries), reducing the immediate overhead of connecting to a data source. Connection pools automatically queue connection requests when they exceed connection pool limits. You can create more than one connection pool for a single data source to give certain users more immediate access to data over others.

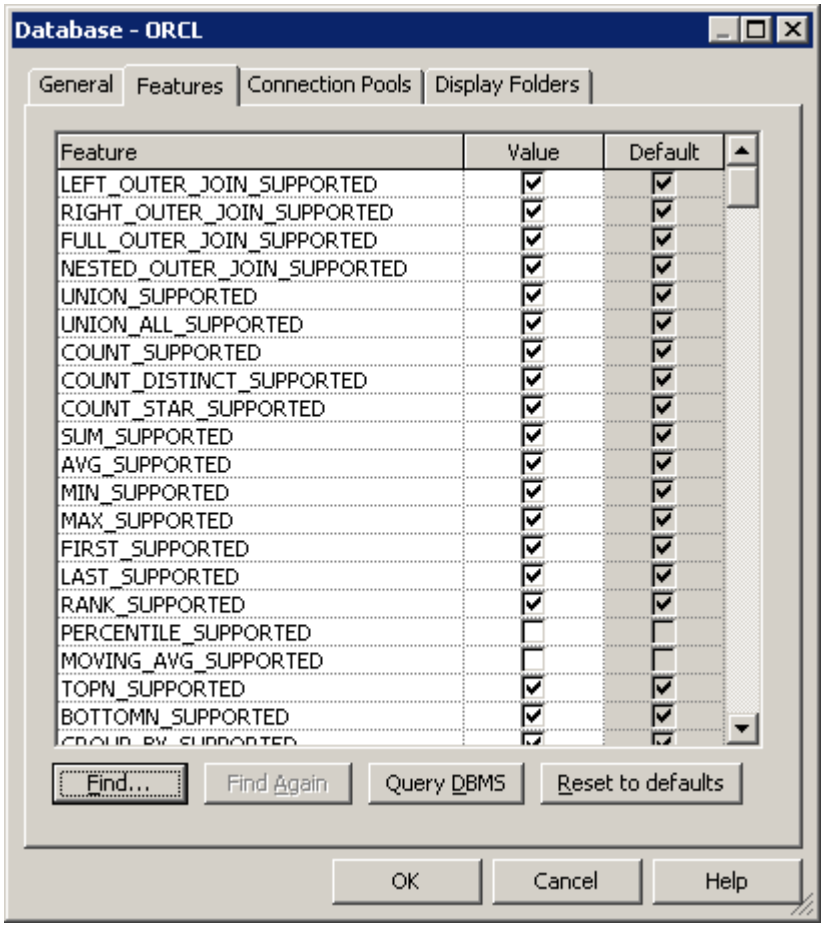
b. Call interface is the application program interface with which to access the data source; Oracle Call Interface (OCI) in this example.

c. Maximum connections is the maximum number of connections allowed for this connection pool. The default is 10. Each connection consumes about 1 MB of memory.

d. Data source name is configured to access the database to which you want to connect. This value is set automatically when you import the tables to the physical layer.
e. Username and password are also configured automatically during import depending on the parameters set for the data source.
f. Enable connection pooling allows a single database connection to remain open for a specified time (in minutes) 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.
g. For more information about connection pool parameters, consult the *System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* or click the Help button.
h. Click **OK** to close the Connection Pool dialog box.

4. Examine the properties of the database object in the Physical layer.
   a. Right-click the **orcl** database object and select **Properties**. You can also double-click the object.
   b. Click the **General** tab, if not selected by default. This tab provides general information about the data source, such as the database name and database type. Notice that the database type is set to **Oracle 11g**.
   c. Click the **Features** tab. This tab lists features that, when selected or deselected, determine the SQL that Oracle BI Server will issue for this database. This features table is set to the database's default values during the schema import process. You can turn off any of these features if there is a reason to do so. Oracle BI Server will adjust the SQL that it sends to the database accordingly and will compensate for the deselected features with its own functionality. Notice that turning a feature on when the default is "off" may or may not cause that feature to be used in the generated SQL. To use a feature, Oracle BI Server needs to know how the feature is implemented in that database platform. If it does not know, it will not use it, even though the feature is

selected.



d. Click the **Find** button.

e. In the Find field, enter **INTERSECT** and click **OK**.

f. Notice that the **INTERSECT_SUPPORTED** feature is supported on this database platform.

g. Click the **Connection Pools** tab. This tab displays all connection pools associated with this data source. In this example, there is only one connection pool, SUPPLIER CP.

h. Click the **Display Folders** tab. You could use this tab to create folders to organize the information in the Physical layer.

i. Click **Cancel** to close the Database properties dialog box.

5. Examine the properties of a physical table object in the Physical layer.

a. Right-click **D1_CALENDAR2** and select **Properties**. The Physical Table properties dialog box opens.

b. Click the **General** tab.

c. Notice that it is possible to rename the object by using the Name field. Do not rename the object now.

d. Notice that the Cacheable check box is selected by default. This determines that queries that hit this table will be cached. You could also select Cache persistence time and use the field and drop-down list to determine the cache persistence time. This determines how long the cached queries that include this table as a source should be used to provide information to users. Ensure that the default, **Cache never expires,** is selected. You learn more about caching in the lesson titled “Cache Management.”

e. Notice that you can select a table type from the Table Type drop-down list. Physical Table is the default, and means that the object represents a physical table. You can also select Stored Proc or Select. Stored Proc is used to call a stored procedure. Select can be used when you want to create a SQL statement to represent a physical table. Leave the table type set to Physical Table.

f. Notice that the Hint box is editable. This box is editable only when the database is Oracle. A hint specified here is included in all SQL that references this table.

g. Notice that it is possible to add a description of the physical object.

h. Click the **Columns** tab.

i. Notice that all columns and the corresponding column properties are listed on this tab. It is also possible to add new columns or delete existing columns.

j. Select the **Keys** and **Foreign Keys** tabs. Notice that no foreign keys are created yet for this table. You learn how to create foreign keys later in this practice.

k. Click **Cancel** to close the Physical Table properties dialog box.

6. Examine the properties of a physical column object in the Physical layer.

   a. Expand the **D1_CALENDAR2** table object.

   b. Double-click any column to open the Physical Column properties dialog box. The physical column properties include the data type of the columns and whether the column is nullable. (The column can contain NULL values.) The Server Administration Tool automatically selects compatible data types based on the data types of the source database.

   c. Click **Cancel** to close the Physical Column properties dialog box.

7. Update row counts and view data. It is a good idea to update row counts or view data after an import to verify connectivity. Viewing data or updating row count, if successful, tells you that you have everything configured correctly.

   a. Update the row count for all tables by selecting **Tools > Update All Row Counts**. This may take a moment. In this practice, you imported a small number of tables. Notice that updating row counts for all tables can take a long time if you have imported many large tables. It is also possible to update row count for a single physical layer object by right-clicking the object and selecting Update Row Count.

   b. When Update All Row Counts completes, move the cursor over the tables and columns and observe that row count information is now visible, including when the row count was last updated.

   c. Right-click any table and select **View Data** to view the data for the table.

   d. Close the View Data dialog box.

8. Save the repository.

   a. Select **File > Save** or click the **Save** button on the toolbar. If the toolbar is not visible, select **Tools > Options > Show Toolbar**.

   b. Click **No** when prompted to check Global Consistency. Checking Global Consistency checks for errors in the entire repository. Some of the more common checks are done in the Business Model and Mapping layer and Presentation layer. Because these layers are not defined yet, bypass this check until the other layers in the repository are built.

   c. Leave the repository open and remain logged in to the Administration Tool for the next practice.

Practice 3-4: Creating Alias Tables

Goal

To assign aliases to physical tables before mapping them to the Business Model and Mapping layer

Scenario

You create aliases for the metadata objects that you imported into the Physical layer of the repository. It is recommended that you use table aliases frequently in the Physical layer to eliminate extraneous joins and to include best practice naming conventions for physical table names.

Outcome

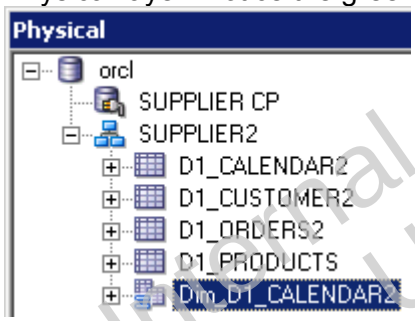
New alias tables for the D1_CALENDAR2, D1_CUSTOMER2, D1_ORDERS2, and D1_PRODUCTS tables

Time

20 minutes

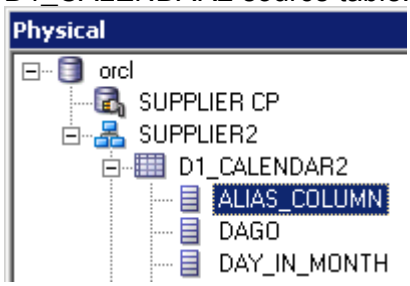
Tasks

1. Create an alias for the D1_CALENDAR2 table.
   a. Right-click **D1_CALENDAR2** and select **New Object > Alias**. The Alias Physical Dialog box opens.
   b. Name the alias **Dim_D1_CALENDAR2**. This is a simple naming convention that identifies the table as a dimension table and includes the original table name.
   c. Click the **Columns** tab. Notice that alias tables inherit all column definitions from the source table.
   d. Double-click any of the columns listed to open the Physical Column properties dialog box. Notice that the column is read-only and cannot be modified.
   e. Click **Cancel** to close the Physical Column dialog box.
   f. Click **OK** to close the Physical Table dialog box. The alias table is added to the Physical layer. Notice the green arrow icon.

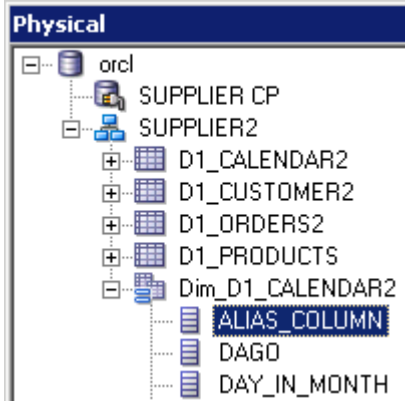


2. Create a new column in the source table and observe the results.
   a. Right-click **D1_CALENDAR2** and select **New Object > Physical Column**. The Physical Column dialog box opens.
   b. Name the column **ALIAS_COLUMN**.

c. Leave the type set to **UNKNOWN** and click **OK**. The column is added to the D1_CALENDAR2 source table.



d. Expand the **Dim_D1_CALENDAR2** alias table and ensure that the column was automatically added. Creating a new column in a source table automatically creates the same column in all its alias tables.



e. Double-click **ALIAS_COLUMN** in the Dim_D1_CALENDAR2 alias table. The Physical Column dialog box opens.

f. Confirm that the type is **UNKNOWN**.

g. Click **Cancel**.

h. Return to the D1_CALENDAR2 table and double-click **ALIAS_COLUMN**.

i. Change the type to **VARCHAR**.

j. Click **OK**.

k. Return to the Dim_D1_CALENDAR2 alias table and double-click **ALIAS_COLUMN**.

l. Confirm that the type is changed to **VARCHAR**. Modification of a source column forces the same changes to be reflected in the alias columns.

m. Click **OK**.

n. Return to the D1_CALENDAR2 table, right-click **ALIAS_COLUMN** and select **Delete**.

o. Click **Yes** to confirm the deletion.

p. Return to the Dim_D1_CALENDAR2 alias table and confirm that ALIAS_COLUMN is deleted. Deletion of a source column automatically deletes the corresponding alias columns.

3. Modify the data types for columns in D1_CALENDAR2 to improve readability of data in Oracle BI analyses.

a. Expand **D1_CALENDAR2**.

b. Double-click the **YYYYMMDD** column to open the Physical Column dialog box.

c. Change the data type from DOUBLE to **INT**.

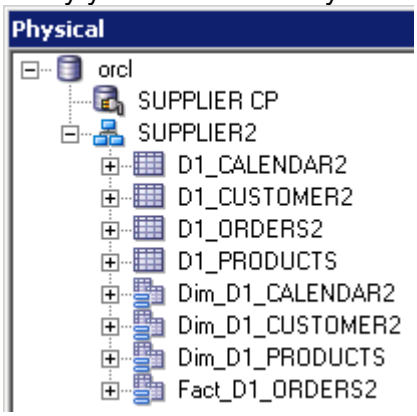
d. Click **OK** to close the Physical Column dialog box.

e. Repeat and change the data type from DOUBLE to INT for the following D1_CALENDAR2 columns:
YEAR
MONTHCODE

4. Create alias tables for the three remaining physical tables:

Original Table Name	Alias Table Name
D1_CUSTOMER2	**Dim_D1_CUSTOMER2**
D1_PRODUCTS	**Dim_D1_PRODUCTS**
D1_ORDERS2	**Fact_D1_ORDERS2**

5. Verify your work. The Physical layer should look similar to the following screenshot:



6. Click the **Save** button to save the ABC repository.
7. Click **No** when prompted to check global consistency.
8. Leave the Administration Tool open for the next practice.

Practice 3-5: Defining Keys and Joins

Goal

To define the primary keys, foreign keys, and joins in the Physical layer

Scenario

You have created a new repository, imported the initial tables from the SUPPLIER2 schema into the Physical layer of the repository, and created alias tables. Now you define keys and joins in the Physical Layer of the repository. If the imported database already had primary key-foreign key relationships defined and the primary keys and foreign keys were imported into the repository, then the join conditions would be set up automatically. But that is not always what you want, because foreign key relationships are set in a database for only one purpose, referential integrity, which may not correspond to the purpose of the Administration Tool and BI Server, which is knowing which joins to include in SQL queries. In the SUPPLIER2 schema, primary keys, foreign keys, and joins are not defined and were not imported into the repository. Therefore, you need to define the keys and join conditions manually. You can create physical keys and joins by using either the Physical Diagram or the Joins Manager.

Outcome

Keys and joins are defined on the physical tables.

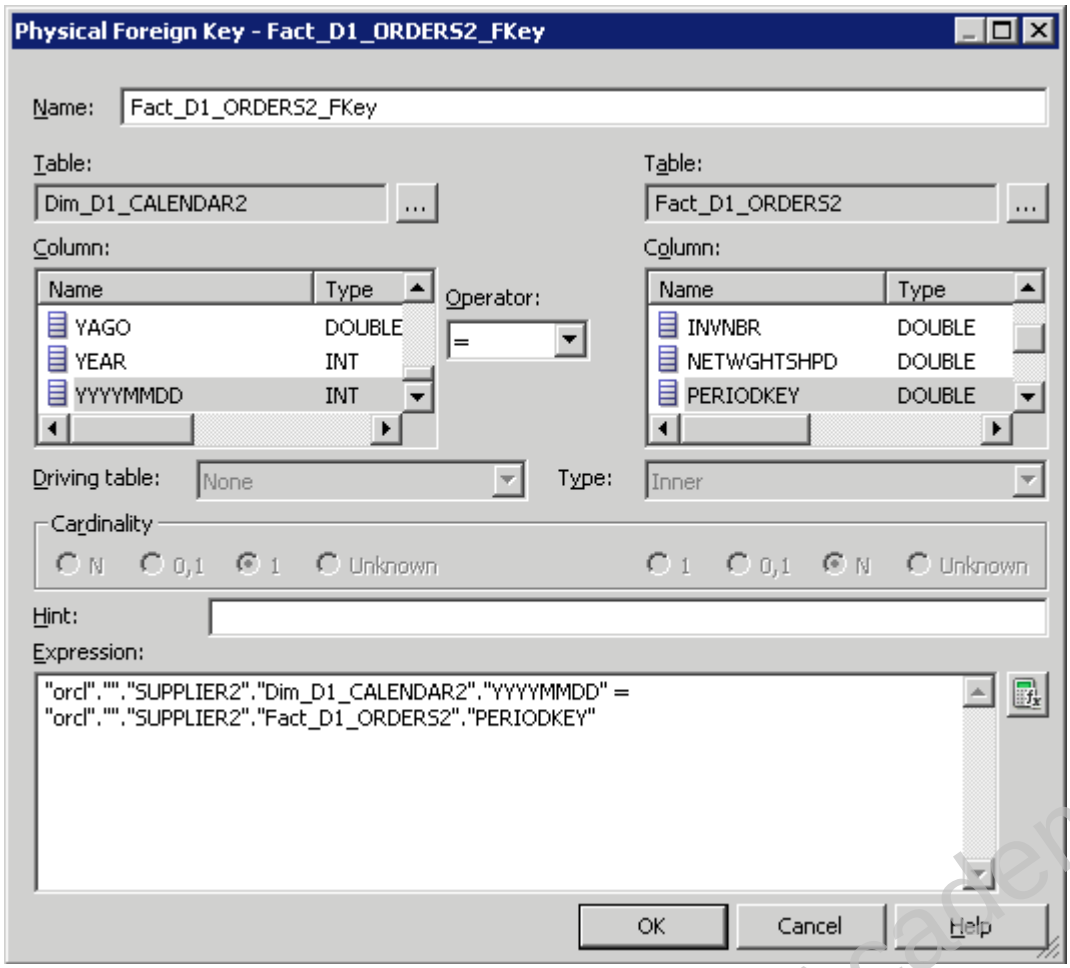
Time

20 minutes

Tasks

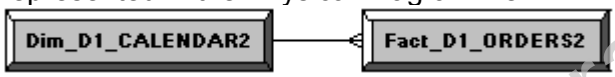
1. In this step, you define joins and keys by using the Physical Diagram feature of the Administration Tool.
   a. Expand the **orcl** database object so that all physical objects are visible.
   b. Select all four alias tables.
   c. Right-click one of the four highlighted alias tables and select **Physical Diagram > Object(s) and All Joins** to open the Physical Diagram dialog box. Alternatively, you can click the Physical Diagram button on the toolbar.
   d. Rearrange the alias table objects so that they are all visible.
   e. Click the **New Foreign Key** button on the toolbar.
   f. Click the **Dim_D1_CALENDAR2** table, and then click the **Fact_D1_ORDERS2** table. The Physical Foreign Key dialog box opens. It matters which table you click first. The join creates a one-to-many (1:N) relationship that joins the key column in the first table to a foreign key column in the second table.
   g. Select the columns that join the tables. Select the **Dim_D1_CALENDAR2.YYYYMMDD** column, and then select **Fact_D1_ORDERS2**.**PERIODKEY**. Ensure that the Expression edit box (at the bottom) contains the following expression:

```
"orcl"."".."SUPPLIER2"."Dim_D1_CALENDAR2"."YYYYMMDD" =
"orcl"."".."SUPPLIER2"."Fact_D1_ORDERS2"."PERIODKEY"
```



h. Click **OK**.

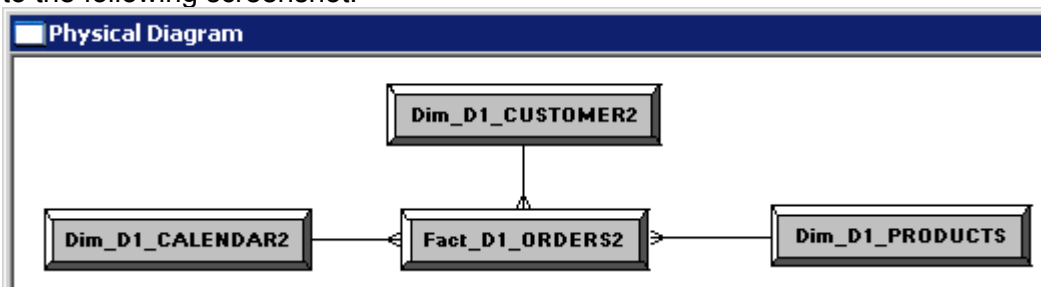
i. Observe the 1:N relationship between Dim_D1_CALENDAR2 and Fact_D1_ORDERS2 represented in the Physical Diagram view:



j. Repeat the process you to create joins and keys for the other tables. Create the joins using the following expressions as a guide:

```
"orcl"."""."SUPPLIER2"."Dim_D1_CUSTOMER2"."NEWKEY" =
"orcl"."""."SUPPLIER2"."Fact_D1_ORDERS2"."CUSTKEY"

"orcl"."""."SUPPLIER2"."Dim_D1_PRODUCTS"."PRODUCTKEY" =
"orcl"."""."SUPPLIER2"."Fact_D1_ORDERS2"."PRODKEY"
```

k. Observe the Physical Diagram and check your work. Your diagram should look similar to the following screenshot:



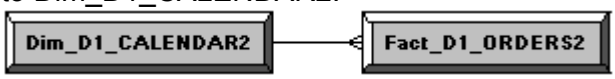
l. You may want to adjust the scale of the objects in the Physical Diagram. If so, right-click the **white space** in the Physical Diagram window, select **Zoom**, and one of the zoom options. You can also use right-click to adjust the grid properties and options.

m. Close the **Physical Diagram** window.

2. Observe additional options for viewing the physical table diagram.

a. Right-click the **Dim_D1_CALENDAR2** table and select **Physical Diagram > Object(s) and Direct Joins**.

b. Observe the Physical Diagram. The diagram shows only those objects with direct joins to Dim_D1_CALENDAR2.



c. Close the **Physical Diagram** window.

d. Repeat this process for the Fact_D1_ORDERS2 table by selecting **Physical Diagram > Object(s) and Direct Joins** for the Fact_D1_ORDERS2 table. Notice that all the tables are visible in the diagram because all the tables have a direct join relationship with Fact_D1_ORDERS2.

e. Close the Physical Diagram window.

3. Observe the changes to the physical table properties.

a. Expand **Dim_D1_CALENDAR2**. Notice that the YYYYMMDD column now has a key icon. The key was defined when you created the join in the Physical Diagram. Similar keys are defined for the Dim_D1_CUSTOMER2 and Dim_D1_PRODUCTS tables.

b. Double-click the **Fact_D1_ORDERS2** table to open the Physical Table properties dialog box.

c. Click the **Foreign Keys** tab. Notice that the foreign key information is visible. This information was created automatically when you created the join expressions in the Physical Diagram.

d. Double-click any of the foreign key expressions and notice that the Physical Foreign Key dialog box opens, displaying the join information.

e. Click **Cancel** to close the Physical Foreign Key dialog box.

f. Click **Cancel** to close the Physical Table properties dialog box.

4. Explore the Joins Manager, which allows you to examine, edit, and delete all the joins, both physical and logical, in a repository.

a. Select **Manage > Joins**. The Joins Manager opens. The joins displayed in the right pane vary depending on the leaf that you select in the left pane. You can view all joins in the repository, in a particular business model, in the Business Model and Mapping layer, in the Physical layer, in the Business Model and Mapping layer for a particular business model, and in the Physical layer for a particular business model. Joins are

further divided into logical foreign key, logical, physical foreign key, and physical complex.

b. In the left pane, select Business Model and Mapping. Notice that no joins are displayed, because you have not yet created any logical joins in the Business Model and Mapping layer. You do that later in the practices for Lesson 3.

c. In the left pane, select **Physical > Physical Foreign Key** to see all physical foreign key joins in the Physical layer. These are the joins that you created earlier in this practice. The Joins Manager displays the join name, the tables in the join, and the join expression.

d. Double-click any of the foreign key expressions and notice that the Physical Foreign Key dialog box opens, displaying the join information. Alternatively, you can right-click any join in the list and select **Properties** to open the Physical Foreign Key dialog box. You can edit the join properties by using this dialog box.

e. Click **Cancel** to close the Physical Foreign Key dialog box.

f. Click any column headings in the right pane to sort the joins by that column.

g. Select **Action > New**. Notice that you can create new joins by using the Joins Manager. However, most users tend to create joins with the physical or logical diagrams, as you do in this course.

h. Select **Action > Close** to close the Joins Manager.

5. Click the **Save** button to save the ABC repository.
6. Click **No** when prompted to check global consistency.
7. Leave the repository and the Administration Tool open for the next practice.

Congratulations! You have successfully created a new repository, imported a table schema from an external data source into the Physical layer, and defined keys and joins.

Practices for Lesson 4: Building the Business Model and Mapping Layer of a Repository

Lesson 4

Practices for Lesson 4

Lesson Overview

In these practices, you will create the Business Model and Mapping layer of a repository.

Practice 4-1: Creating the Business Model

Goal

To create a business model in the Business Model and Mapping layer of the repository

Scenario

In the previous practice, you created the Physical layer of the repository. You are now ready to begin building the business model in the Business Model and Mapping layer of the repository. The Business Model and Mapping layer of the Administration Tool defines the business model of the data and specifies the mapping between the business model and the Physical layer schemas. Business models are also referred to as logical models or dimensional models.

Business models are always dimensional, unlike objects in the Physical layer, which reflect the organization of the data sources. The Business Model and Mapping layer can contain one or more business models. Each business model contains logical tables, columns, and joins.

There are two main categories of logical tables: fact and dimension. Logical fact tables contain the measures by which ABC gauges its business operations and performance. Logical dimension tables contain the data used to qualify the facts. This practice assumes that a business model has already been designed on paper. You know what measures are important to ABC, what ABC employees compare measures to, and how the company likes to analyze its data. The goal of this practice is to capture this information in a business model in the Business Model and Mapping layer of the repository.

Outcome

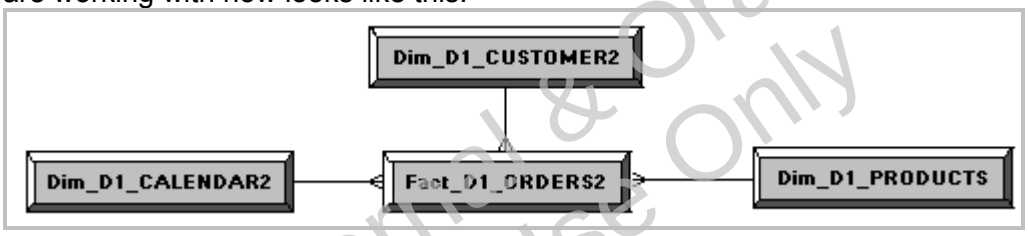
In the Business Model and Mapping layer, the SupplierSales business model is created with the following logical tables: Dim-Time, Dim-Customer, Dim-Product, and Fact-Sales.

Time

20 minutes

Tasks

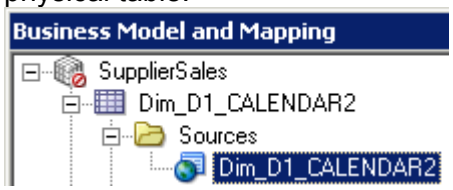
The business model that ABC has defined is based on Sales, Product, Customer, and Time data. These data elements will be used to model the initial application. The physical model you are working with now looks like this:



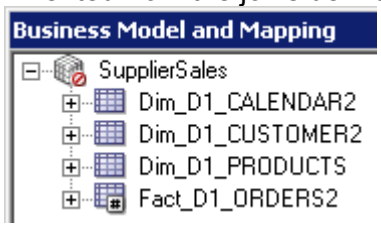
1. Return to the ABC repository, which should still be open in offline mode. If the repository is not open, follow these steps to open it.
   a. Select **Start > Programs > Oracle Business Intelligence > BI Administration**.
   b. Select **File > Open > Offline**.
   c. Double-click **ABC.rpd** to open the repository file.
   d. Enter **welcome1** as the repository password.
2. Create a business model in the Business Model and Mapping layer.

a. Right-click the white space of the Business Model and Mapping layer and select **New Business Model**.
b. In the Name field, enter **SupplierSales**.
c. Click **OK**. The new Business Model and Mapping folder appears in the Business Model and Mapping layer. The red symbol on the business model indicates that it is not yet enabled for querying. You enable the business model for querying after the Presentation layer is defined and the repository passes a global consistency check.

3. Create the logical tables.
   a. In the Physical layer, expand the **orcl** database object and expand the **SUPPLIER2** schema.
   b. Drag the following four alias tables simultaneously from the Physical layer onto the SupplierSales business model.
      Dim_D1_CALENDAR2
      Dim_D1_CUSTOMER2
      Dim_D1_PRODUCTS
      Fact_D1_ORDERS2
      This action creates logical tables in the business model with logical columns.
   c. Expand the logical tables and notice that a Sources folder is created for each logical table. In each Sources folder, there is a logical table source. For example, the logical table source for the Dim_D1_CALENDAR2 logical table is the Dim_D1_CALENDAR2 physical table.



   d. Notice also that logical tables have different table icons. In the Business Model and Mapping layer, a hash sign (#) indicates a fact table. The logical joins are automatically inherited from the joins defined in the Physical layer.



4. Rename the logical tables in the business model to make them more meaningful.
   a. Double-click the **Dim_D1_CALENDAR2** table in the SupplierSales business model. The Logical Table dialog box opens.
   b. If necessary, click the **General** tab.
   c. In the Name field, enter **Dim-Time**.
   d. Click **OK**.
   e. Repeat the process and rename the following tables. Alternative methods for renaming include right-clicking an object and selecting **Rename**, or clicking an object twice to make it editable. Notice that a logical table name is purely a business model artifact. Logical table names are not necessarily exposed to users.

Table	Rename to:
Dim_D1_CUSTOMER2	**Dim-Customer**

Dim_D1_PRODUCTS	**Dim-Product**
Fact_D1_ORDERS2	**Fact-Sales**

5. Delete the columns that are not needed for analysis.
   a. Expand the **Fact-Sales** table and delete the following columns:

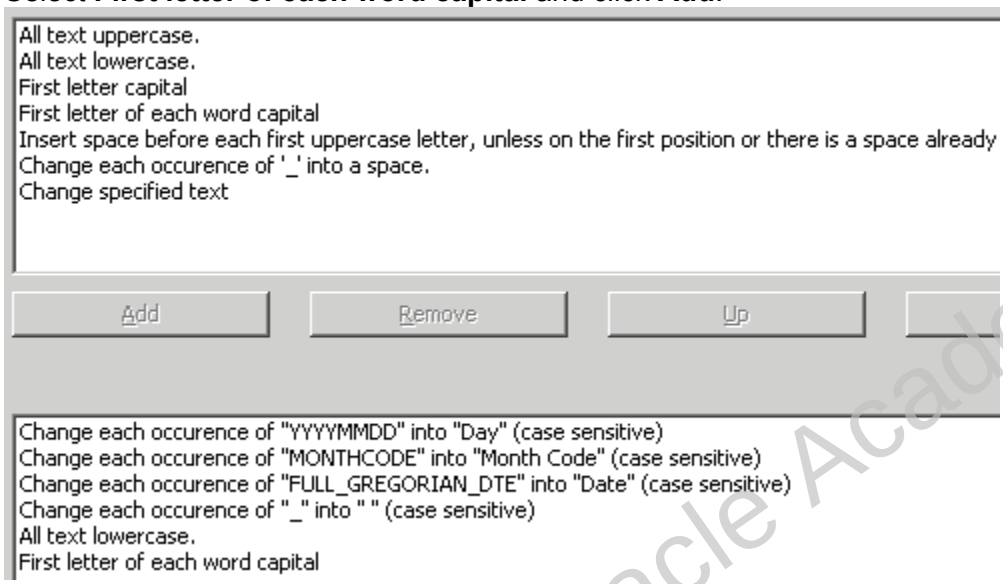
Columns
CUSTKEY
INVNBR
PERIODKEY
PRODKEY

   b. Expand the **Dim-Time** table and delete the following columns:

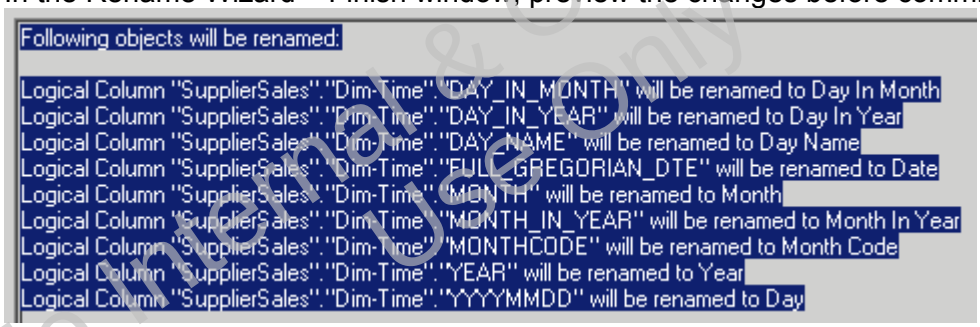
Columns
DAGO
MAGO
ORDERDAY_IN_YEAR
WEEK_NUM_IN_YEAR
YAGO

   c. Expand the **Dim-Product** table and delete the **PGCODE** column.
   d. Expand the **Dim-Customer** table and delete the **REPNO** and **FACTOR** columns.
6. Use the Rename Wizard to rename logical columns in the Dim-Time logical table. You use the Rename Wizard utility for renaming Business Model or Presentation layer objects. You can use it to replace text strings, change all letters to lowercase, use uppercase for the first letter of words, and so on. You can preview the new names before committing the changes. It is primarily used on Business Model logical columns after importing physical objects into the middle layer. The names that you give logical columns in the business model can be exposed via the Presentation layer to end-user tools such as the Analysis Editor. However, it is possible to override logical column names in the Presentation layer, as you will see later in this course.
   a. Select **Tools > Utilities > Rename Wizard** and click **Execute**.
   b. At the bottom of the middle pane, click **Business Model and Mapping**.
   c. In the middle pane, expand **SupplierSales > Dim-Time**.
   d. Use Shift + click to select all columns and click **Add** to add the columns to the right pane.
   e. Click **Next**. Notice that Logical Column is selected.
   f. Click **Next** again.
   g. Select **Change specified text**.
   h. In the Find field, enter **YYYYMMDD**.
   i. In the "Replace with field," enter **Day**.
   j. Select **Case Sensitive**.
   k. Click **Add**.
   l. Select **Change specified text**.
   m. In the Find field, enter **MONTHCODE**.

n. In the “Replace with field,” enter **Month Code**.
o. Select **Case Sensitive**.
p. Click **Add**.
q. Select **Change specified text**.
r. In the Find field, enter **FULL_GREGORIAN_DTE**.
s. In the “Replace with field,” enter **Date**.
t. Select **Case Sensitive**.
u. Click **Add**.
v. Select **Change specified text**.
w. In the Find field, enter an underscore.
x. In the “Replace with field,” enter a space.
y. Click **Add**.
z. Select **All text lowercase** and click **Add**.
aa. Select **First letter of each word capital** and click **Add**.

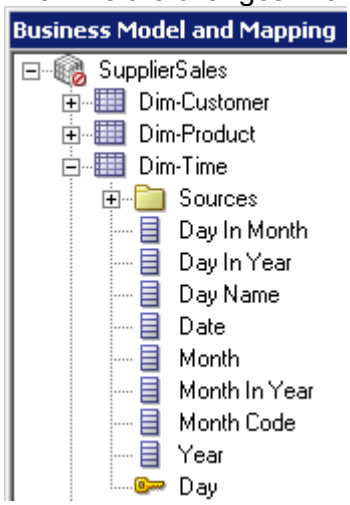


bb. Click **Next**.
cc. In the Rename Wizard – Finish window, preview the changes before committing.



dd. Click **Finish**.

ee. Examine the changes in the repository and verify that they are as expected.



ff. Save the repository.

gg. Do not check global consistency.

7. Rename columns in the remaining logical tables according to the tables below. You can use the Rename Wizard or rename the columns manually.

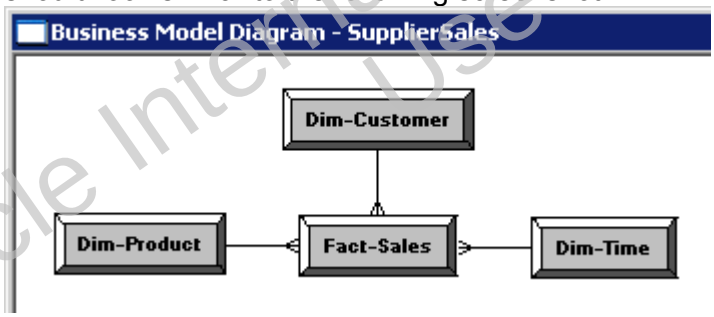
Fact-Sales columns	Rename as:
DOLLARS	**Dollars**
NETWGHTSHPD	**Net Weight Shipped**
UNITORDD	**Units Ordered**
UNITSHPD	**Units Shipped**

Dim-Customer columns	Rename as:
ADDRESS	**Address**
CITY	**City**
DISTRICT	**District**
NAME	**Customer**
NEWKEY	**Customer Key**
PHONE	**Phone**
REGION	**Region**
ROUTECODE	**Route Code**
SALESREP	**Sales Rep**
STATE	**State**
ZIP_CODE	**Zip Code**

Dim-Product Columns	Rename as:
DIETCODE	**Diet Code**
GENERICDESCRIPTION	**Generic**

PACKAGE_WEIGHT	**Package Weight**
PACKAGECODE	**Package Code**
PRODUCTKEY	**Product Key**
SPECIFICDESCRIPTIN	**Specific**
SUBTYPECODE	**Subtype Code**
SUPPLIERCODE	**Supplier Code**
TYPECODE	**Type Code**

8. In this step, you verify the logical table keys. For a business model to be valid, each logical dimension table must have a logical key. Logical keys can be composed of one or more logical columns. The logical key defines the lowest level (the most detailed level) of information of any source in the logical table. In this example, the logical table keys were built automatically when you dragged the tables from the Physical layer to the business model.
   a. Double-click the **Dim-Customer** logical table in the business model to open the Logical Table dialog box.
   b. Click the **Keys** tab.
   c. Confirm that the Customer Key column is defined as the key for this table.
   d. Click **OK** to close the Logical Table dialog box.
   e. Repeat these steps and ensure that the Day column is defined as the logical table key for the Dim-Time table and Product Key is defined as the logical table key for the Dim-Product table.
9. In this step, you verify the logical table joins. Another requirement for a valid business model is that the logical tables must be joined via logical joins. Logical joins express the cardinality relationships between the logical tables. Logical fact tables are always at the "many" end of these cardinality relationships. Logical joins help Oracle BI Server understand the relationships between the various pieces of the business model. When a query is sent to Oracle BI Server, the server figures out how to construct physical queries by examining how the logical model is structured. Examining logical joins is an integral part of this process.
   a. Right-click the **SupplierSales** business model and select **Business Model Diagram > Whole Diagram**. The Logical Table Diagram window opens.
   b. Rearrange the table icons so that they are all visible. Right-click the white space and adjust the zoom factor, if desired. Recall that the logical joins are automatically inherited from the joins defined in the Physical layer. Later in this course you learn how to build logical joins using the Business Model Diagram. The Business Model Diagram should look similar to the following screenshot.



   c. Double-click one of the join connectors to open the Logical Join dialog box.

d. Leave the default values as they are, but notice which properties you can set in the Logical Join dialog box: name, business model, tables, driving table, join type, and cardinality. Also, notice which properties you cannot set: the join expression and the join columns.

e. Do not change the default values. Click **OK**. Typically, when defining logical joins, you leave the defaults as they are and click OK.

f. Close the Business Model Diagram window. The join relationships determine which tables are the logical dimension tables and which is the logical fact table. Recall that a fact table is always on the "many" side of a logical join. You now have a logical star schema consisting of one logical fact table, Fact-Sales, and three logical dimension tables: Dim-Time, Dim-Product, and Dim-Customer.

10. Save the repository.
11. Click **No** when prompted to check global consistency.
12. Leave the repository open for the next practice.

Practice 4-2: Creating Simple Measures

Goal

To examine the logical-to-physical column mappings and to create simple measures

Scenario

The SupplierSales business model is now defined in the Business Model and Mapping layer. In this practice, you review the logical-to-physical table and column mappings to better understand the relationships that exist between logical tables and their logical table sources. You then create measures by setting aggregation rules for logical columns. Then you check the physical tables referenced by the business model.

Outcome

Measures defined in the Fact-Sales logical table

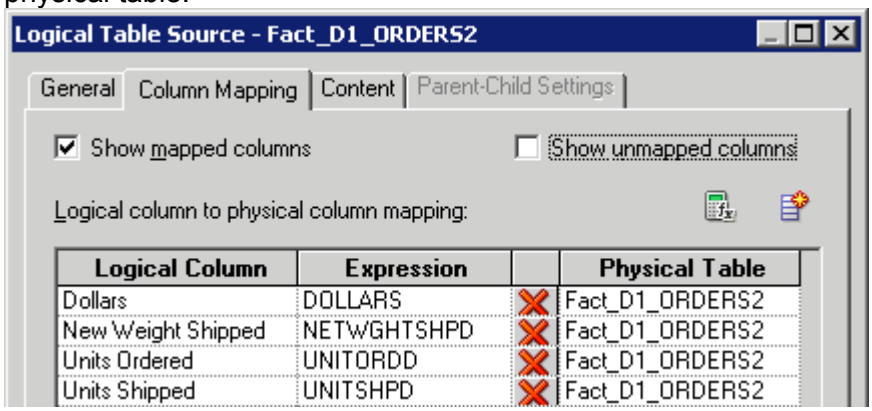
Time

5 minutes

Tasks

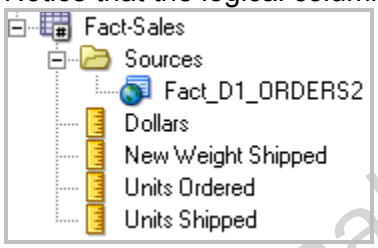
1. In this step, you examine a logical table source (LTS). Each logical table in a business model has a subfolder called Sources that contains the logical table sources. Logical table sources contain the mappings from the logical columns in the Business Model and Mapping (BMM) layer to the physical columns in the Physical layer. Every logical column maps directly (or indirectly via another logical column) to a column or columns in the Physical layer.
   a. Expand the **Sources** folder of the Fact-Sales logical table. Notice that the name of the logical table source for the Fact-Sales table is Fact_D1_ORDERS2. This logical table source was created automatically during the process of dragging the Fact_D1_ORDERS2 physical table to the business model.
   b. Double-click the **Fact_D1_ORDERS2** logical table source to open the Logical Table Source dialog box.
   c. Click the **General** tab. By default, the logical table source name corresponds to the name of the physical table that was dragged from the Physical layer, but this name could be changed to something more meaningful. For this exercise, leave the name as is. Notice also that the path to the physical table is identified in the “Map to these tables:” area.
   d. Click the **Column Mapping** tab to review how the logical columns are mapped to the Physical layer. If necessary, adjust the column widths and heights, or drag the entire dialog box window to make it larger or smaller. You should see three column headings: Logical Column, Expression, and Physical Table. For example, the Dollars logical column is mapped to the DOLLARS physical column in the Fact-D1_ORDERS2

physical table.



e. The Content tab is discussed later in this course. Click **Cancel** to close the dialog box.

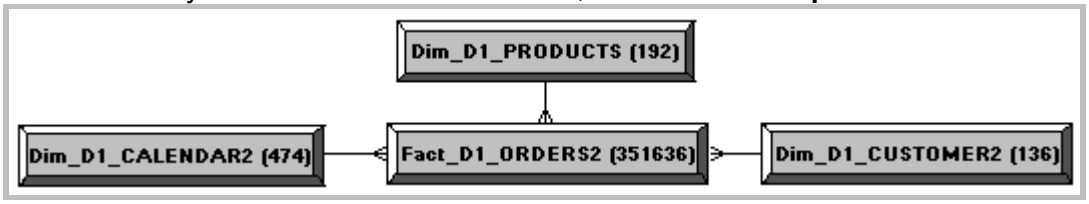
2. In this step, you create some measures by defining default aggregation rules on logical columns defined in the Fact-Sales table. Measures are typically data that is additive, such as total dollars or total quantities. The Fact-Sales logical fact table contains the measures in your business model. You aggregate some of its logical columns by summing the column data.
   a. Double-click the **Dollars** logical column in the Fact-Sales table. The Logical Column dialog box opens.
   b. Click the **Aggregation** tab.
   c. Set the default aggregation rule for Dollars to **Sum**.
   d. Click **OK**. Notice that the Dollars icon is changed to indicate that an aggregation rule is defined.
   e. Use Ctrl + click to select **Net Weight Shipped**, **Units Ordered**, and **Units Shipped**.
   f. Right-click and select **Set Aggregation**.
   g. Set the default aggregation rule to **Sum**.
   h. Click **OK**. You can use this method to set the same aggregation rule for multiple columns at once.
   i. Notice that the logical column icons are changed for all four columns.



3. In this step, you check which physical tables are referenced by a business model. In most situations, there are tables that are included in the Physical layer, but not in the Business Model and Mapping layer. Only the tables referenced in the Business Model and Mapping layer, that is, only the tables included in logical table sources, are used in queries. All other physical tables are ignored when Oracle BI Server queries the physical database.
   a. Select **Tools > Options**.
   b. Click the **General** tab.
   c. Select **Show row count in physical view**.
   d. Click **OK**.
   e. Select the **SupplierSales** business model.

f. Click the **Physical Diagram** icon on the toolbar.

g. The Physical Diagram displays all the physical tables currently referenced by the SupplierSales business model. It also displays the row count for tables with updated row counts. If you do not see the row counts, select **Tools > Update All Row Counts**.



h. Double-click the **connector** between the Dim_D1_CUSTOMER2 and the Fact_D1_ORDERS2 tables. Notice that the physical join is displayed. You can determine that this is a physical join because there is a foreign key join expression.

i. Click **Cancel** to close the join dialog box.

j. Close the Physical Diagram.

k. Select **Tools > Options**.

l. Click the **General** tab.

m. Deselect **Show row count in physical view**.

n. Click **OK**.

4. Save the repository.
5. Do not check global consistency.
6. Leave the repository open for the next practice.

Congratulations! You have successfully built a business model in the Business Model and Mapping layer of a repository and created business measures.

Practices for Lesson 5: Building the Presentation Layer of a Repository

Lesson 5

Practices for Lesson 5

Lesson Overview

In these practices, you will create the Presentation layer of an Oracle Business Intelligence repository.

Practice 5-1: Creating the Presentation Layer

Goal

To create the Presentation layer of a repository

Scenario

You have created the initial SupplierSales business model in the repository. You now create the Presentation layer of the repository. The Presentation layer exposes the business model objects in the Oracle BI Analysis Editor so that users can build analyses to analyze their data.

Outcome

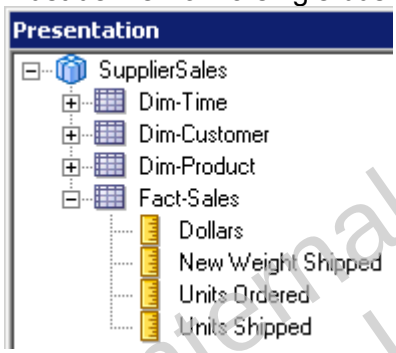
In the Presentation layer of the repository, there is a SupplierSales subject area.

Time

10 minutes

Tasks

1. In this step, you create the Presentation layer for the SupplierSales business model. The Presentation layer has three types of objects: subject area, presentation table, and presentation column. In the Oracle BI Analysis Editor the subject area appears as a subject area, the presentation table appears as a folder, and the presentation columns appear as columns in the folders. The ABC repository should still be open in the Administration Tool from the previous practice.
   a. Ensure that the Presentation layer is visible. If it is not, select **View > Presentation**.
   b. Drag the **SupplierSales** business model into the Presentation layer. A subject area appears with the name SupplierSales.
   c. Expand **SupplierSales** in the Presentation layer. When you create presentation objects by dragging a business model to the Presentation layer, the business model becomes a subject area, the logical tables become presentation tables, and the logical columns become presentation columns. Notice that all objects within a subject area must derive from a single business model.



2. In this step, you examine the properties of a subject area.
   a. Double-click **SupplierSales** in the Presentation layer to open the Subject Area dialog box.
   b. Click the **General** tab. You use this tab to create or edit a subject area.
   c. Notice that it is possible to change the name of the subject area. For the purpose of these lessons, leave the name as SupplierSales. This is the name that appears as a subject area in the Analysis Editor. Also, as you will see in later lessons, the name of

the subject area can be used in the FROM clause of a logical SQL statement. Therefore, short names are desirable. The tool prevents you from giving the same name to a subject area and a presentation table. It is also advisable to avoid using characters ($, %) that may cause illegal SQL syntax with particular client tools.

d. Click the **Permissions** button. This dialog box is used to assign user and application role permissions to this repository object. Permissions are discussed in more detail in the lesson titled “Security.”

e. Click **Cancel** to close the Permissions dialog box.

f. The custom display name is used if you are planning to present the name in local languages. You can ignore this for the purposes of this practice. You learn more about using custom display names in the lesson titled “Localizing Oracle BI Metadata”.

g. Notice that the business model is grayed out and cannot be modified. This is because all objects within a subject area derive from a single business model and cannot span multiple business models. After the business model is set, it cannot be changed and the tool prevents you from including objects from other business models.

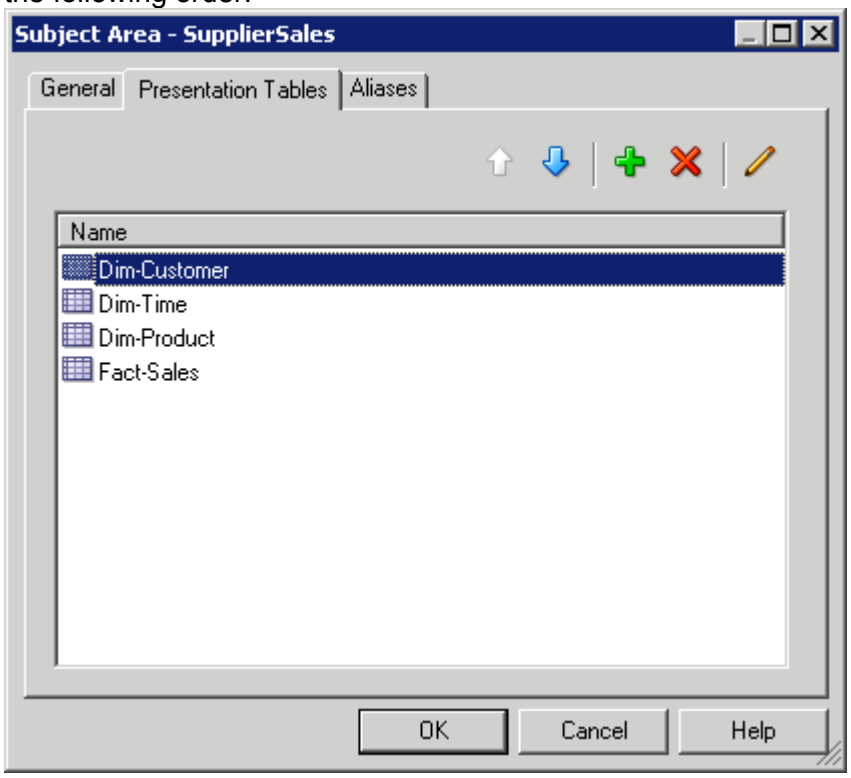
h. Notice that “export logical keys” is deselected by default. This is irrelevant to users of the Analysis Editor, but may be advantageous to some third-party query and reporting tools. If selected, columns in the Presentation layer that are key columns in the Business Model and Mapping layer will be presented as key columns to an ODBC client and will have a key icon in the Presentation layer.

i. Notice that the implicit fact column is not assigned. If you set an implicit fact column, this column is added to a query when it contains columns from two or more dimension tables and no measures. It is used to specify a default join path between dimension tables when there are several possible alternatives. You learn more about configuring implicit fact columns in the lesson titled “Setting an Implicit Fact Column”.

j. In the Description field, enter something similar to **Analyze Sales and Shipment Data**. This information is visible under the corresponding subject area in Analysis Editor. You confirm this in the next set of practices.

k. Click the **Presentation Tables** tab.

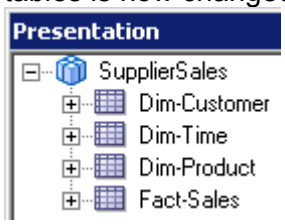
l. Use the **Up** and **Down** buttons or drag objects to rearrange the presentation tables into the following order:



m. Click the **Aliases** tab. If you change the name of a subject area, the tool automatically creates an alias using the previous name. You can use this tab to specify or delete an alias for a subject area.

n. Click **OK** to close the Subject Area dialog box.

o. Expand the **SupplierSales** subject area and notice that the order of the presentation tables is now changed in the Presentation layer.



3. In this step, you explore the properties of a presentation table. You can use presentation tables to hold columns from multiple logical tables, thereby reducing the number of apparent logical tables. You can also use presentation tables to organize logical columns into smaller groupings, thereby increasing the number of apparent logical tables. For example, you might have a hundred measures in a logical fact table. You may want to create multiple presentation tables as containers for those measures and thus present them in an organized way (for example, all dollar measures in one folder and all unit measures in another folder). These measure folders could be nested within a single folder in the Analysis Editor.

a. Double-click the **Dim-Customer** presentation table in the Presentation layer. The Presentation Table dialog box opens.

b. Click the **General** tab.

c. Notice that, by default, the presentation table name is the same as the logical table name. With Oracle BI Answers, the name can be anything, except that it should not contain single quotation marks, double quotation marks, or the "%" sign. For client tools that generate SQL, it is advisable to avoid names that might violate valid SQL syntax. For example, avoid SQL key words, spaces, single and double quotation marks, and other characters such as "$" or "%." A presentation table name cannot be the same as the subject area name or any logical column name in that catalog. As in a subject area, changing a presentation table name does not have any effect on the logical table name in the Business Model and Mapping layer. The Alias tab keeps a record of any changes.

d. Type **Customer Data** in the Description field. This will show up as a "tool tip" in the Analysis Editor when the user places the cursor over the object. You confirm this in the next set of practices.

e. Click the **Columns** tab.

f. Change the order of the columns, using the **Up** and **Down** buttons or by dragging, into the following order:

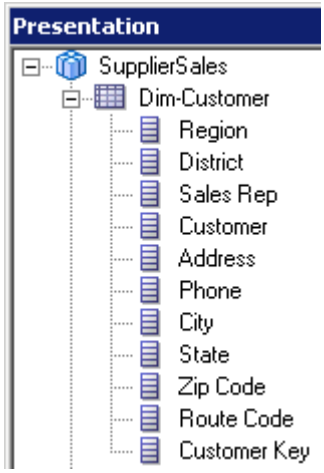
Columns
Region
District
Sales Rep
Customer
Address
Phone
City
State
Zip Code
Route Code
Customer Key

g. Click **OK** to close the Presentation Table dialog box.

4. In this step, you explore the properties of a presentation column. Presentation columns can come from multiple logical tables in a business model. By default, a presentation column uses the same name as its corresponding logical column in the Business Model and Mapping layer. If you rename the column in the Business Model and Mapping layer, corresponding presentation columns are automatically renamed wherever they appear in the Presentation layer. The reverse is not true. If you rename a presentation column, it does not impact the corresponding logical column in the Business Model and Mapping layer. However, as with subject areas and tables, the repository stores an alias for the column using the previous name.

   a. Expand the **Dim-Customer** presentation table.

b. Confirm that the columns are now in the order you specified in the previous step.



c. Double-click the **District** column. The Presentation Column dialog box opens.
d. Click the **General** tab.
e. Deselect **Use Logical Column Name**. The Name field can now be edited.
f. Change the column name by entering **Sales District** in the Name field.
g. Click the **Aliases** tab and observe that the original logical column name (alias) is stored for this presentation column.
h. Return to the **General** tab.
i. Click the **Edit** button. The Logical Column dialog box opens.
j. Click the **General** tab in the Logical Column dialog box.
k. Which logical column does this presentation column map to?

_____

l. Which business model does this presentation column map to?

_____

m. Which logical table does this presentation column map to?

_____

n. Click the **Column Source** tab.
o. Which physical table and column does this presentation column map to?

_____

p. Is this the original physical table that you imported, or the alias table that you created?

_____

q. Click **OK** to close the Logical Column window.
r. Click **OK** to close the Presentation Column window.

5. In this step, you simplify the content by deleting unnecessary presentation columns. You may not want to expose all the logical columns of a business model in a subject area. You can delete columns from the Presentation layer safely without affecting the existence of the corresponding logical columns in the Business Model and Mapping Layer. For example, key columns in presentation tables can be deleted unless the client tools require that key information be provided.

a. In the Presentation layer, in the Dim-Customer table, delete the **Customer Key** column.
b. Click **Yes** to confirm the deletion.
c. In the Dim-Product table, delete the **Product Key** column.
d. Click **Yes** to confirm the deletion.
e. In the Business Model and Mapping layer, expand **Dim-Customer** and confirm that the Customer Key logical column is not deleted.

6. Rename presentation tables.
   a. Rename the dimension presentation tables, but not the Fact-Sales table:

Presentation table	Rename as:
Dim-Customer	Customer
Dim-Time	Time
Dim-Product	Product

   b. Double-click the **Customer** presentation table to open the Presentation Table properties dialog box.
   c. Click the **Columns** tab and notice that changing the presentation table name does not impact the column mappings.
   d. Click the **Aliases** tab and notice that the previous name for the presentation table has been stored.
   e. Click **OK** to close the Presentation Table dialog box.
   f. Notice also that changing the presentation table names in the Presentation layer has no impact on the logical table names in the Business Model and Mapping layer.
7. Save the repository.
8. Click **No** when prompted to check global consistency.
9. Leave the repository open for the next practice.

Congratulations! You have successfully built the Presentation layer of a repository.

Solutions 5-1: Creating the Presentation Layer

Answers

4.k. Which logical column does this presentation column map to?
District

4.l. Which business model does this presentation column map to?
SupplierSales

4.m. Which logical table does this presentation column map to?
Dim-Customer

4.o. Which physical table and column does this presentation column map to?
Dim_D1_CUSTOMER2.DISTRICT

4.p. Is this the original physical table that you imported, or the alias table that you created?
Alias table

Practices for Lesson 6: Testing and Validating a Repository

Lesson 6

Practices for Lesson 6

Lesson Overview

In these practices, you will test and validate a repository and make it available for queries.

Practice 6-1: Testing the Repository

Goal

To test the repository by generating some queries, retrieving the results, and examining the query log

Scenario

You finished building the initial business model, and now you must test the repository before continuing. You begin by checking the repository for errors by using the consistency check option. You then test the repository by using the Analysis Editor to run queries. Finally, you examine the query log file to verify the SQL generated by Oracle BI Server.

Outcome

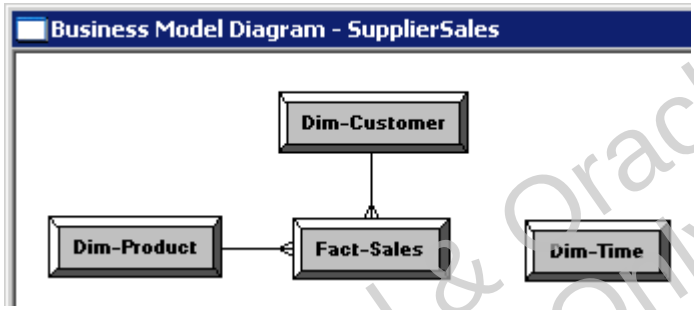
A tested and verified repository file

Time

45 minutes

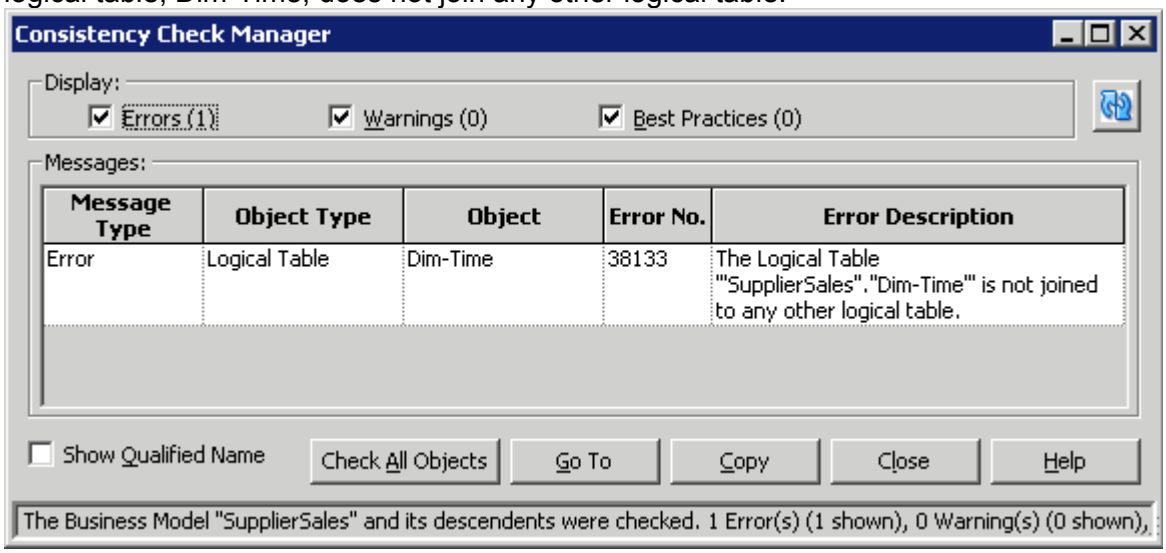
Tasks

1. Modify the business model to generate an inconsistent business model due to a missing logical join.
   a. In the Business Model and Mapping layer, right-click **SupplierSales** and select **Business Model Diagram > Whole Diagram**.
   b. Select the **join connection** between the Dim-Time and Fact-Sales logical tables.
   c. Right-click and select **Delete** to create a condition in the business model where an undefined join condition exists.
   d. Click **Yes** to confirm the delete.



   e. Close the Business Model Diagram window.
   f. In the Business Model and Mapping layer, right-click **SupplierSales** and select **Check Consistency**. The Consistency Check Manager appears and displays an error message for the SupplierSales business model. The error description explains that the

logical table, Dim-Time, does not join any other logical table.



g. The consistency check provides three types of messages:

Error messages indicate errors that must be fixed to make the repository consistent.

Warning messages indicate conditions that may or may not be errors, depending upon the intent of the Oracle BI Server administrator. For example, a warning message about a disabled join may be the result of the administrator intentionally disabling a join (for example, by eliminating a circular join condition).

Best Practices messages provide information about conditions but do not indicate an inconsistency (for example, “fact table does not contain a logical key”).

h. Notice there are buttons that allow you to perform other tasks, such as going to the object in the repository or copying the error message.

i. Click **Close** to close Consistency Check Manager.

2. Use the Business Model Diagram to repair the logical join.

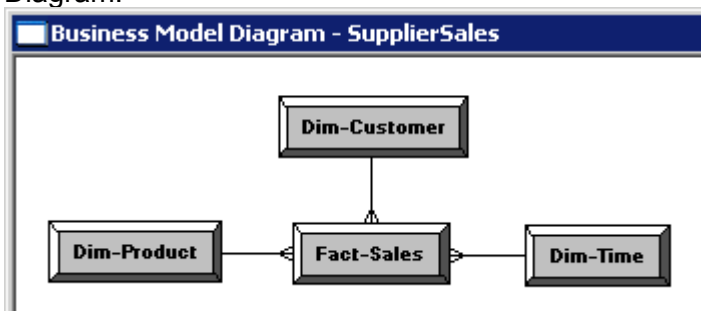
a. Right-click the **SupplierSales** business model and select **Business Model Diagram > Whole Diagram.**

b. Click the **join icon** on the tool bar.

c. Select **Dim-Time** and then **Fact-Sales** in the diagram. The order is important. The Logical Join dialog box opens. Recall that you examined this dialog box in the practices for the lesson titled “Building the Business Model and Mapping Layer of a Repository.”

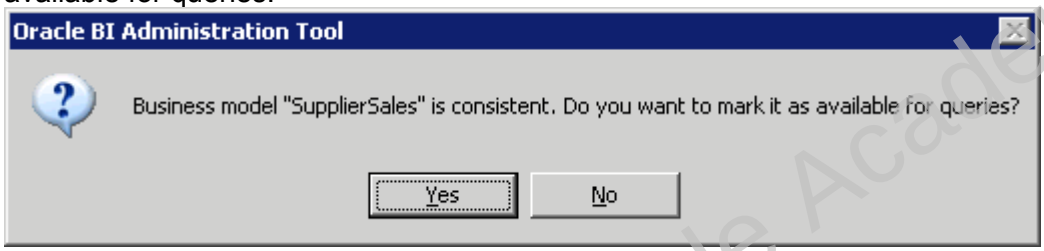
d. Click **OK** to close the Logical Join dialog box.

e. The logical join between Dim-Time and Fact-Sales is re-created in the Business Model Diagram.



3. Perform a global consistency test to check the repository for inconsistencies. A consistent repository has met the following requirements:
   - All logical columns are mapped directly or indirectly to one or more physical columns.
   - All logical dimension tables have a logical key.
   - All logical tables have a logical join relationship to another logical table.
   - There are at least two logical tables in the business model: one is a logical fact table, the other is logical dimension table. Both tables may map to the same physical table.
   - There are no circular logical join relationships.
   - A presentation catalog exists for the business model.

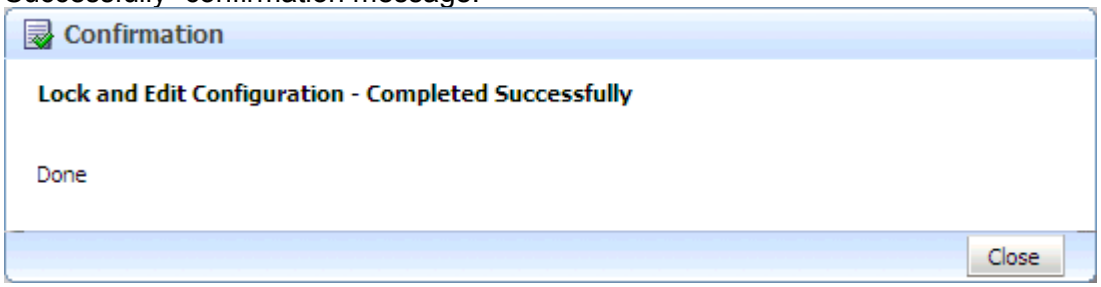
   a. Select **File > Check Global Consistency**. A message appears indicating that the SupplierSales business model is consistent and asks whether you want to mark it as available for queries.



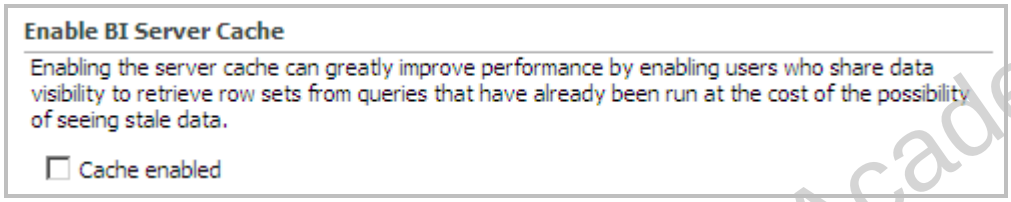
   b. Click **Yes**. You should receive the following message: "Consistency check didn't find any errors, warnings or best practice violations."

   c. Click **OK** to close the message. The SupplierSales business model folder has changed from unavailable for queries (red icon) to available for queries (green icon).
   d. Save the repository.
   e. Click **No** when prompted to check global consistency, because you just checked consistency.
   f. Select **File > Close** to close the repository.
   g. Leave the Administration Tool open.

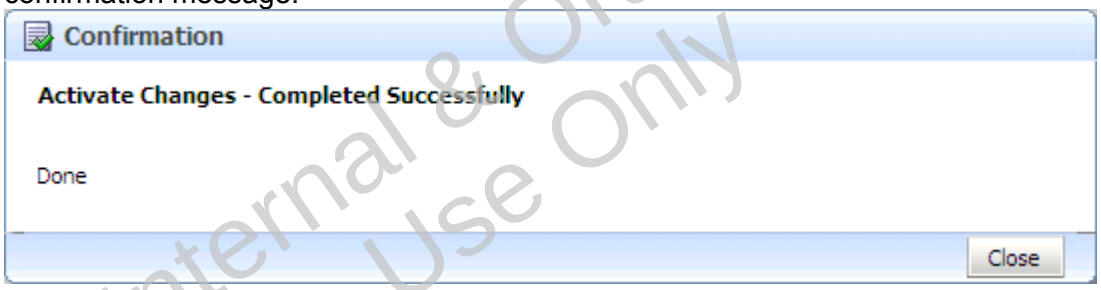
4. Use Fusion Middleware Control Enterprise Manager to disable cache. Caching is typically not used during development, except to test the cache. You learn more about caching in the lesson titled “Cache Management.”
   a. Return to **Fusion Middleware Control Enterprise Manager**, which should still be open in your browser. If it is not open, enter the following URL: **http://localhost:7001/em**.
   b. If your session has timed out, log in as **weblogic/welcome1**.
   c. In the left pane, expand **Business Intelligence** and select **coreapplication**.
   d. In the right pane, select **Capacity Management** and then the **Performance** subtab.
   e. Locate the **Enable BI Server Cache** section. Cache is enabled by default.
   f. Click **Lock and Edit Configuration**.
   g. Click **Close** when you receive the “Lock and Edit configuration – Completed Successfully” confirmation message.



   h. Deselect **Cache enabled**.

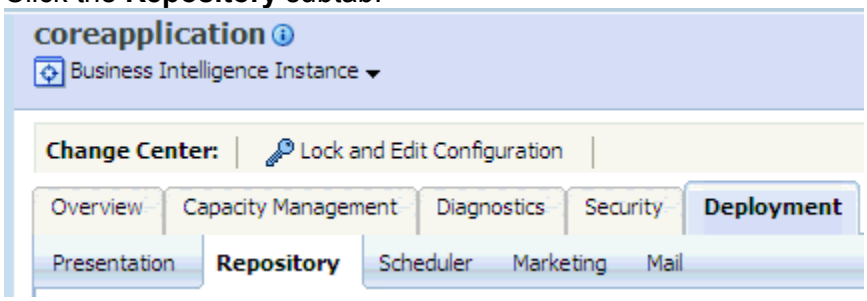


   i. Click **Apply**.
   j. Click **Activate Changes**.
   k. Allow processing to complete.
   l. Click **Close** when you receive the “Activate Changes – Completed Successfully” confirmation message.



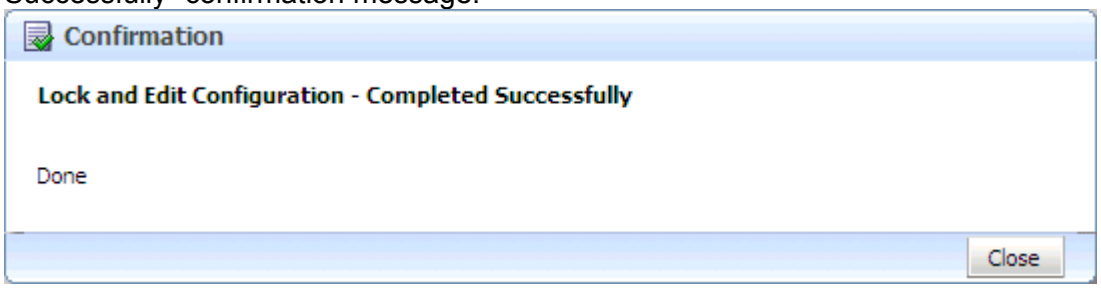
   m. Do not click **Restart to apply recent changes** yet. You do that after uploading the repository in the next set of steps.
5. Use Fusion Middleware Control Enterprise Manager to upload the repository.
   a. In the right pane, click the **Deployment** tab.

b. Click the **Repository** subtab.



c. Click **Lock and Edit Configuration**.

d. Click **Close** when you receive the "Lock and Edit configuration – Completed Successfully" confirmation message.



e. In the Upload BI Server Repository section, click **Browse** to open the "Choose file" dialog box.

f. By default, the "Choose file" dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.

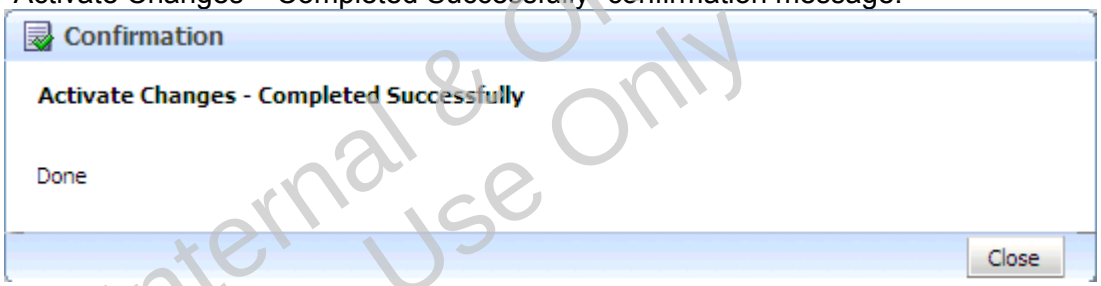
g. Select **ABC.rpd** and click **Open**. You can also double-click the repository to open it.

h. Enter **welcome1** in the Repository Password and Confirm Password fields.

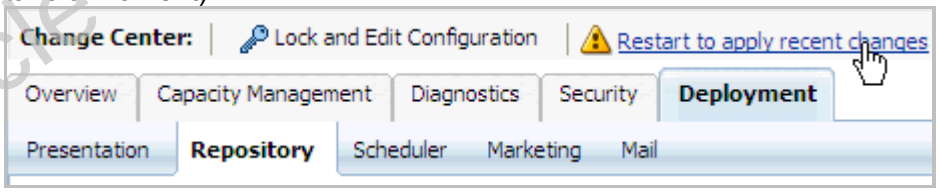
i. Click **Apply**. Notice that Default RPD now displays ABC with an extension (for example, ABC_BI0002).

j. Click **Activate Changes**.

k. Allow Active Changes processing to complete. Click **Close** when you receive the "Activate Changes – Completed Successfully" confirmation message.

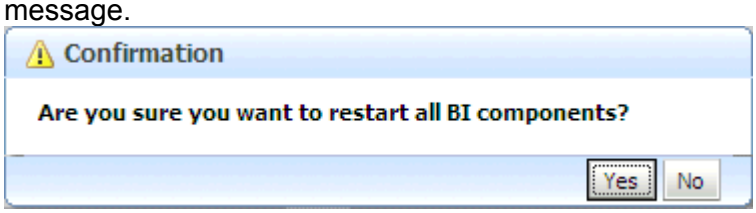


l. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).

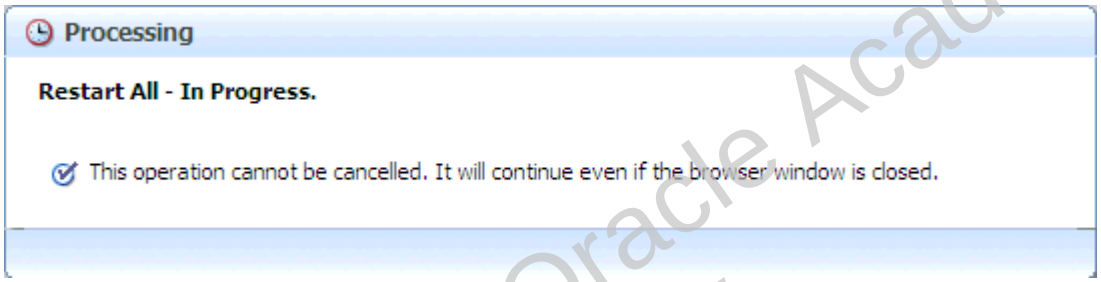


m. On the Overview page, click **Restart**.

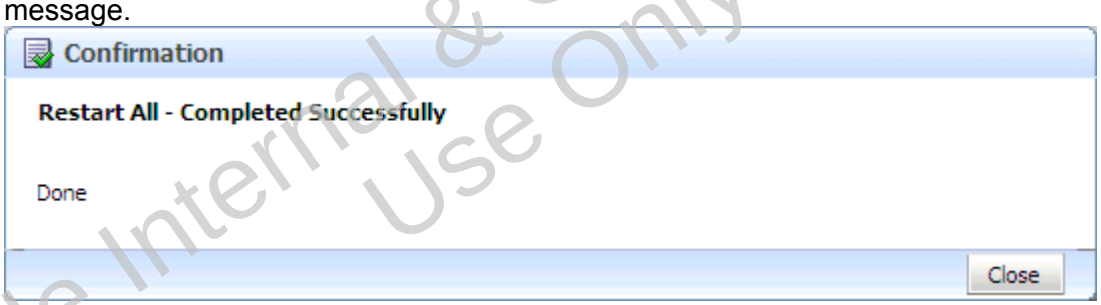
n. Click **Yes** when you receive the "Are you sure you want to restart all BI components?" message.



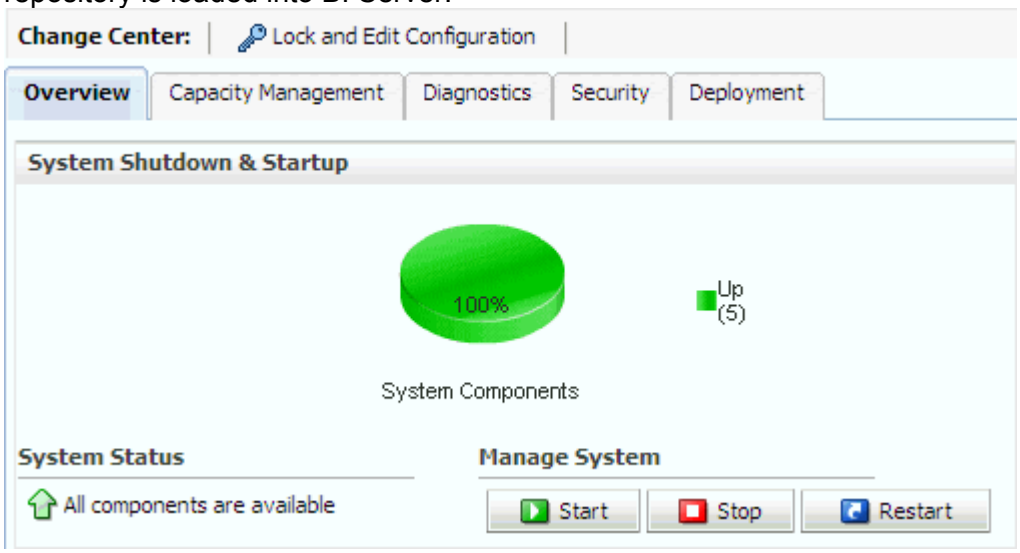
o. Allow the Restart All – In Progress processing to complete. This may take a few moments.



p. Click **Close** when you receive the "Restart All – Completed Successfully" confirmation message.



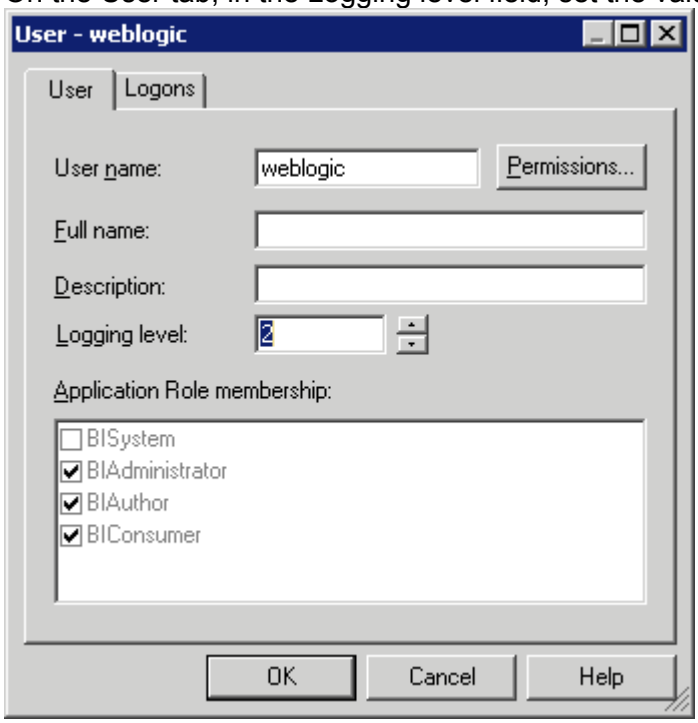
q. Confirm that System Components = 100%. Cache is now disabled and the ABC repository is loaded into BI Server.



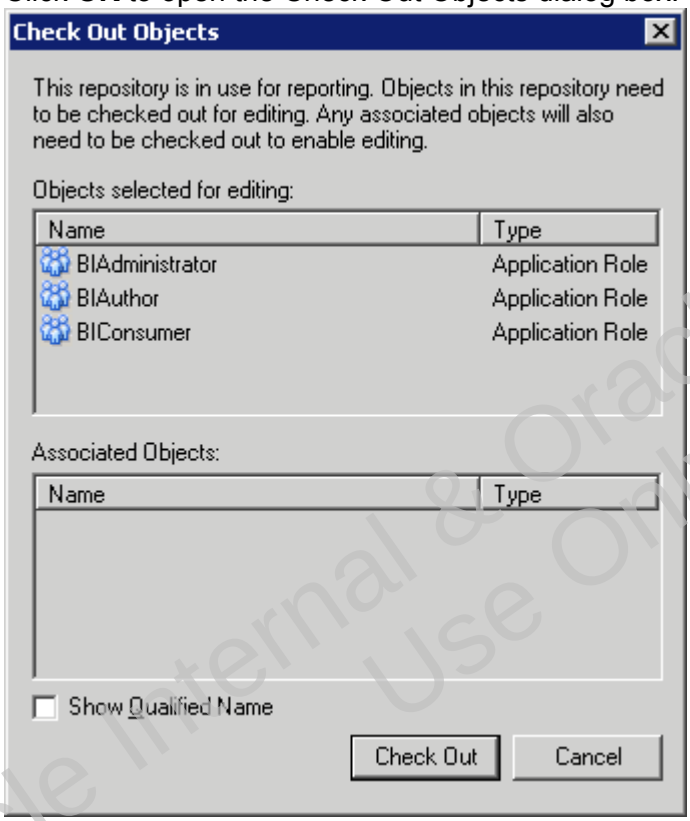
r. Leave Enterprise Manager open.

6. In this step, you enable query logging for the weblogic user. To test a repository, you must build some analyses, retrieve the results, and examine the query log. You log query activity at the individual-user level. Logging is intended for testing, debugging, and technical support. In production mode, logging is normally disabled because query logging can impact performance by producing very large log files.
   a. Return to the Administration Tool.
   b. Select **File > Open > Online** to open the ABC repository in online mode.
   c. Enter **welcome1** as the repository password.
   d. Enter **weblogic** as the username and **welcome1** as the user password.
   e. Click **Open** to open the repository in online mode.
   f. Select **Manage > Identity** to open Security Manager.
   g. In the left pane, select **Identity Management > BI Repository**.
   h. Click the **Users** tab. A list of preconfigured users appears in the right pane.
   i. In the right pane, double-click **weblogic**. The User dialog box opens.

j. On the User tab, in the Logging level field, set the value to **2**.



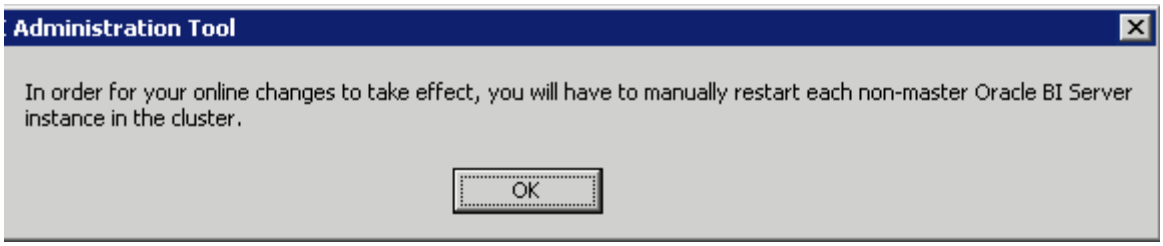
k. Click **OK** to open the Check Out Objects dialog box.



l. Click **Check Out**. When you are working in a repository open in online mode, you are prompted to check out objects when you attempt to perform various operations.

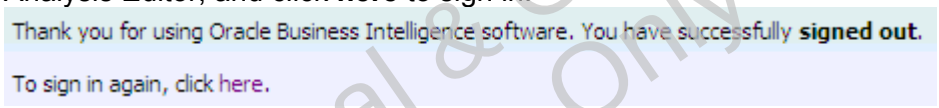
m. Select **Action > Close** to close the Security Manager window. Additional security-related topics are addressed in more detail in the lesson titled “Security.”

n. Select **File > Check In Changes** or click the **Check In Changes** icon on the toolbar.
o. Save the repository. There is no need to check consistency.
p. Select **File > Close** to close the repository.
q. Click **OK** when you receive the following message: "In order for your online changes to take effect, you will have to manually restart each non-master Oracle BI Server instance in the cluster."



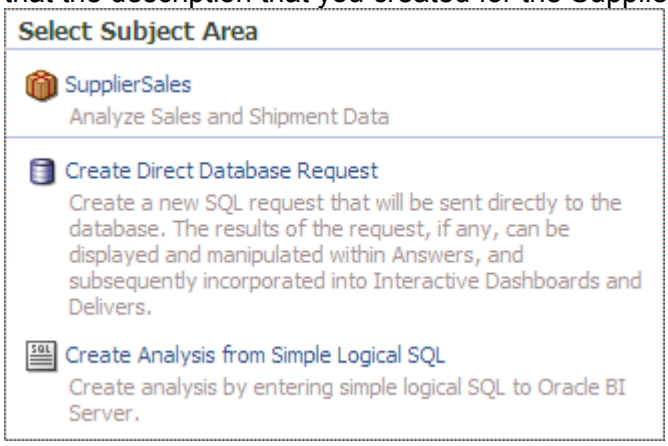
r. Leave the Administration Tool open.

7. Restart Oracle BI components.
   a. Return to **Fusion Middleware Control Enterprise Manager**, which should still be open.
   b. Navigate to **Business Intelligence > coreapplication > Overview**.
   c. Click **Restart**.
   d. Click **Yes** to confirm that you want to restart all BI components.
   e. Allow restart processing to complete. This may take a few moments.
   f. Click **Close** when you receive the "Restart All – Completed Successfully" confirmation message. **Explanation:** In Oracle BI 11*g*, user definitions and group membership are created and maintained in an identity store, not in the repository as in prior releases of the Oracle BI product. Information maintained in the identity store is often needed during repository development. To facilitate this development, a copy of objects that have properties specific to metadata is kept in the repository file and can be viewed using the Administration Tool. To make security changes visible in the repository, you must load the repository, make changes in online mode, and restart Oracle BI Server as you just did in this set of steps. You learn more about setting up users and groups in the lesson titled "Security."
8. Open Analysis Editor to execute queries and test the SupplierSales business model.
   a. Return to the Oracle Business Intelligence browser tab where you signed out of Analysis Editor, and click **here** to sign in.



   b. Sign in as **weblogic** with password **welcome1**.

c. In the Create section, click **Analysis** to open the Select Subject Area window. Notice that the description that you created for the SupplierSales subject area is visible.



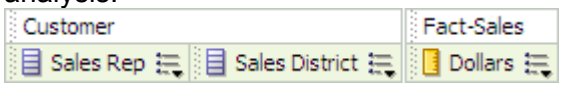
d. Click **SupplierSales** to open Analysis Editor.

e. Place the cursor over the **Customer** folder and ensure that the description you entered in an earlier practice appears as a "tool tip."

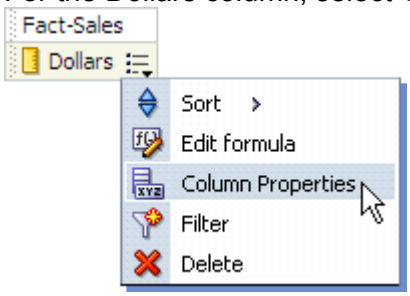
9. Create a new analysis and format the columns.

a. Expand **Customer** and double-click the **Sales Rep** and **Sales District** columns to add the columns to the analysis.

b. Expand the **Fact-Sales** table and double-click the **Dollars** column to add it to the analysis.



c. For the Dollars column, select **Column Properties**.



d. In the Column Properties dialog box, click the **Data Format** tab.

e. Click **Override Default Data Format**.

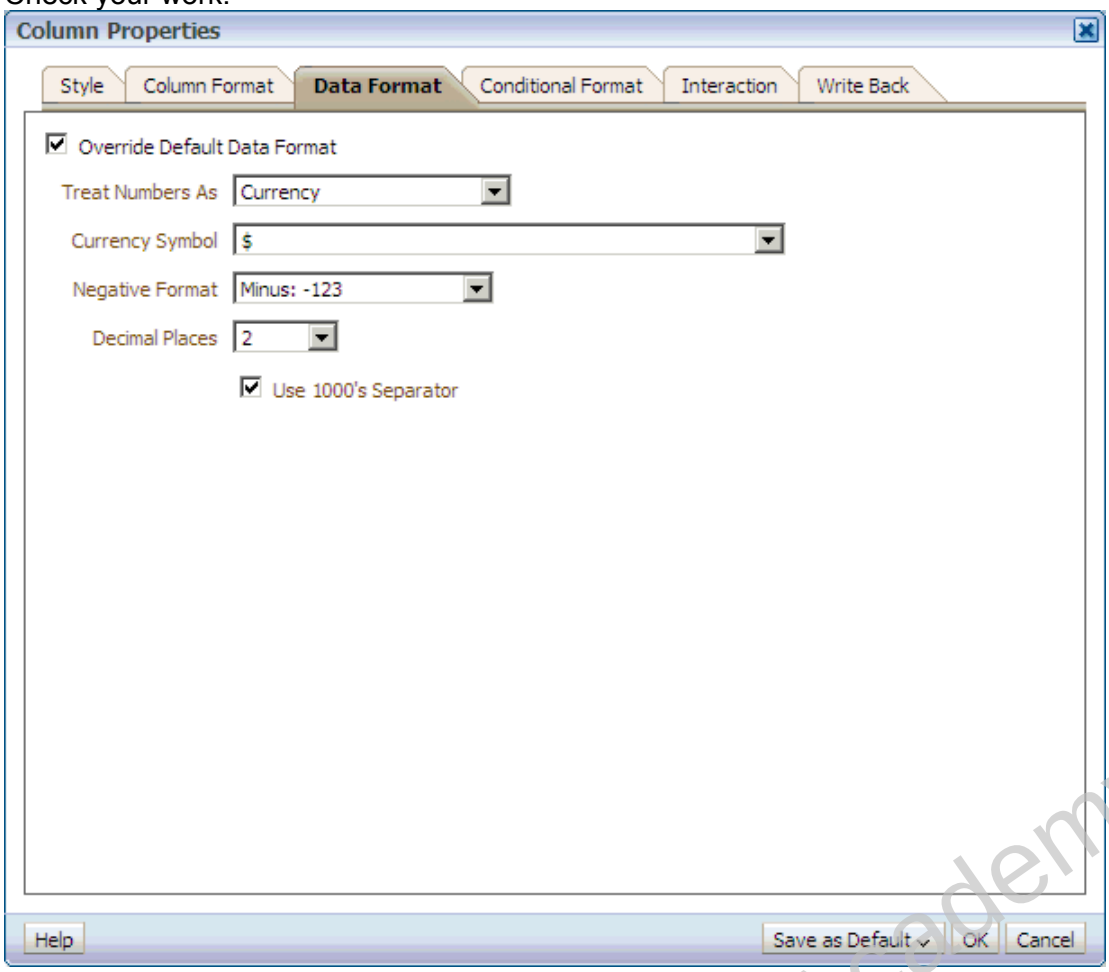
f. In the Treat Number As field, select **Currency**.

g. In the Currency Symbol field, select **$**.

h. In the Decimal Places field, select **2**.

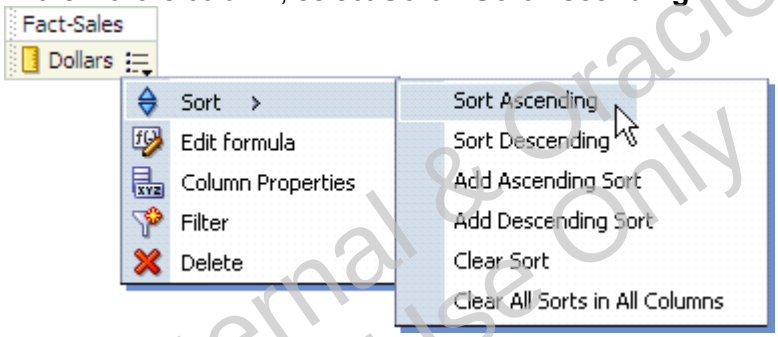
i. Select the **Use 1000's Separator** check box.

j. Check your work:



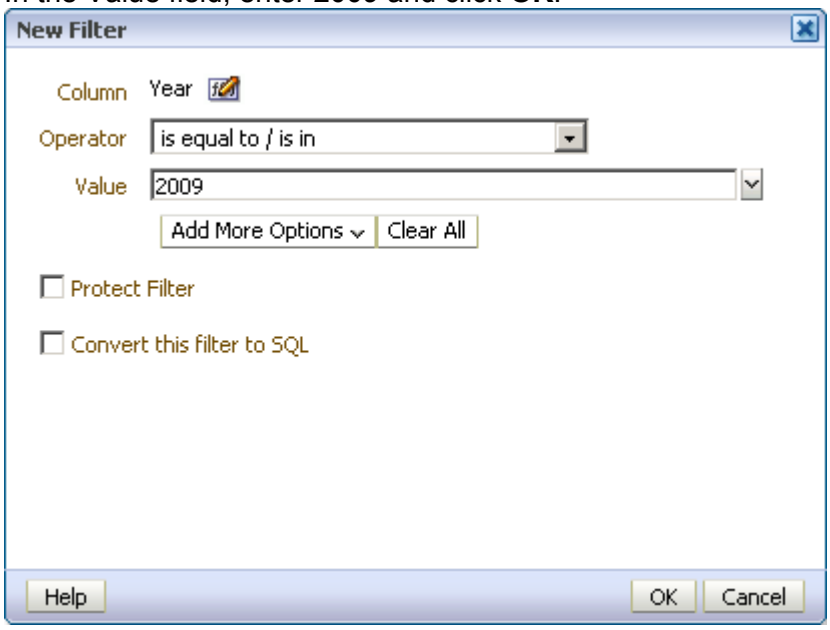
k. Select **Save as Default > Save as the system-wide default for '"Fact-Sales".Dollars'"**.

l. In the Dollars column, select **Sort > Sort Ascending**.



10. Create filters for the analysis and view the results.
    a. Expand the **Time** table.
    b. Double-click **Year** to add it to the analysis.
    c. For the Year column, select **Filter** to open the New Filter dialog box.

d. In the Value field, enter **2009** and click **OK**.



e. For the Sales District column, select **Filter**.

f. Click the **down arrow** next to the Value field.

g. Select **MidAtlantic** from the list, and click **OK**.

h. Notice that both filters are now added to the Filters area:

Year is equal to / is in 2009
AND Sales District is equal to / is in MidAtlantic

i. Delete the **Sales District** and **Year** columns from the analysis so that these columns will not be displayed in the analysis results.

j. Your analysis now includes two columns:

Customer	Fact-Sales
Sales Rep	Dollars

And two filters:

Year is equal to / is in 2009
AND Sales District is equal to / is in MidAtlantic

k. Click the **Results** tab to view the results. Verify that the Dollars column is formatted and sorted as expected. The results show total dollars for each sales rep in the MidAtlantic sales district for the year 2009.

Sales Rep	Dollars
GEORGE MASUR	$97,129.88
DALE FAIRWEATHER	$167,932.96
WALLY RAISANEN	$190,607.12
PAULA MADISON	$467,039.54

11. In this step, you create a new analysis showing the dollars for each sales district in the East region for the year 2009.

a. Select **New > Analysis > SupplierSales**.

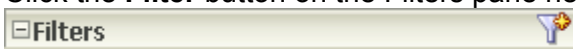
b. Click **OK** when prompted with the following message: “Are you sure you want to navigate away from this page?”.

c. Create the following request:

Customer	Fact-Sales
Sales District	Dollars

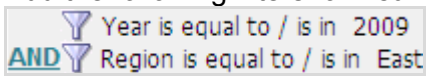
d. Sort the **Dollars** column in ascending order.

e. Click the **Filter** button on the Filters pane header.



f. Select **More Columns**. This is another method for setting filters.

g. Add the following filters for **Year** and **Region**:



h. Click **Results**.

Sales District	Dollars
Florida	$406,139.91
MidAtlantic	$922,709.50
UpperSouth	$1,513,814.30
Yankee	$3,220,998.40

i. Sign out of Oracle BI

12. Examine the query log.

a. Return to **FMW Enterprise Manager**, which should still be open.

b. Click the **Diagnostics** tab.

c. Click the **Log Messages** subtab.

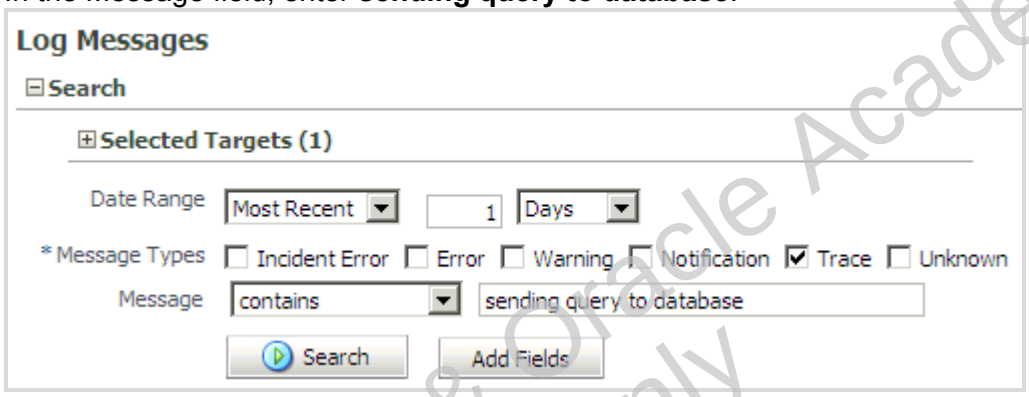
d. Scroll to the bottom of the window to the **View / Search Log Files** section.

e. Click **Server Log**.

f. In the Log Messages screen, leave the data range set to **Most Recent, 1 Days**.

g. Deselect all message types except for **Trace**.

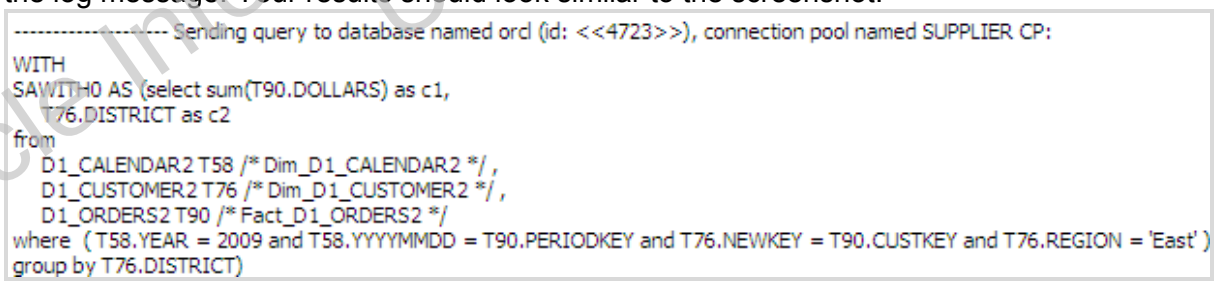
h. In the Message field, enter **sending query to database**.



i. Click **Search**.

j. There should only be one message at this point, but if there are more than one, select the last message in the list. This is the most recent query sent to the database.

k. In the bottom pane, click the **Collapse Pane button** (arrow on the right side) to view the log message. Your results should look similar to the screenshot.



l. Click the **Restore Pane button**. Throughout the course, you return to this view to check the query log.

m. Leave Enterprise Manager open.

Congratulations! You have successfully used the Consistency Check Manager, Oracle BI Analysis Editor, and the query log to test and check the repository.

Practices for Lesson 7: Managing Logical Table Sources

Lesson 7

Practices for Lesson 7

Lesson Overview

In these practices, you will manage logical table sources in and Oracle BI repository.

Practice 7-1: Enhancing the Product Dimension

Goal

To import normalized tables with additional product information into the Physical layer of the repository

Scenario

There are product tables that store detailed information about ABC's products. You want to add these tables to the Product dimension in the Business Model and Mapping layer. You import these tables into the repository and create keys and foreign key joins for the tables.

Outcome

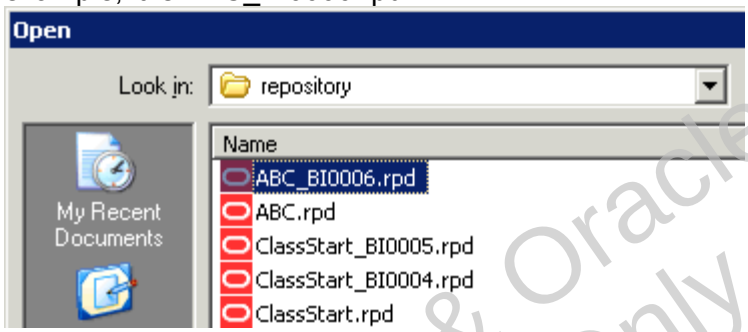
D1_PRICELIST, D1_PROD_DIET_TYPES, D1_PRODUCT_SUBTYPE, D1_PRODUCT_TYPE, and D1_SUPPLIERS tables imported into the Physical layer with associated keys and joins

Time

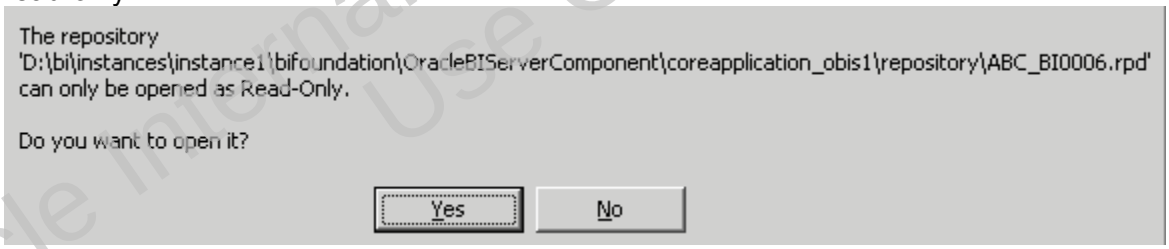
10 minutes

Tasks

1. Replace the existing ABC repository with the ABC repository version that contains the security information.
   a. Return to the Oracle BI Administration Tool, which should still be open. If not, select **Start > Programs > Oracle Business Intelligence > BI Administration**.
   b. Select **File > Open > Offline**.
   c. Select the **ABC repository with the highest number extension**. In the screenshot example, it is ABC_BI0006.rpd.

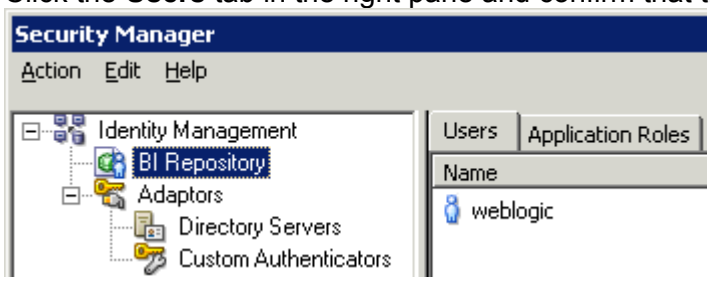


   d. Click **Open**. You should receive a message that this repository can only be opened as read-only.

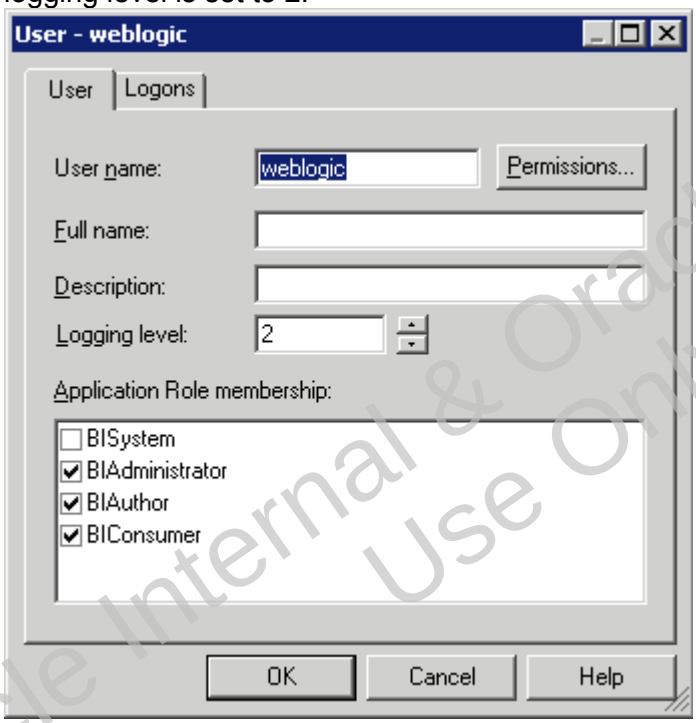


   Recall that this is the repository that is currently uploaded into Oracle BI Server memory. You want to use this version of the repository because it has the user- and logging-level information that you defined earlier.

e. Click Yes to open the Open Offline dialog box. The Repository Password dialog box opens.
f. Enter **welcome1** as the repository password.
g. Click **OK** to open the repository in read-only mode.
h. Select **File > Save As** to open the Save As dialog box.
i. Select **ABC.rpd** to enter it in the File name field.
j. Click **Save**.
k. You should receive the following message: “ABC.rpd already exists. Do you want to replace it?”.
l. Click **Yes** to open the ABC repository in offline mode. This action replaces the existing ABC repository with the ABC repository that contains the security information.
m. Select **Manage > Identity** to open Security Manager.
n. Select **BI Repository** in the left pane.
o. Click the **Users** tab in the right pane and confirm that the **weblogic** user is visible.



p. Double-click **weblogic** to open the User dialog box. On the User tab, confirm that the logging level is set to 2.

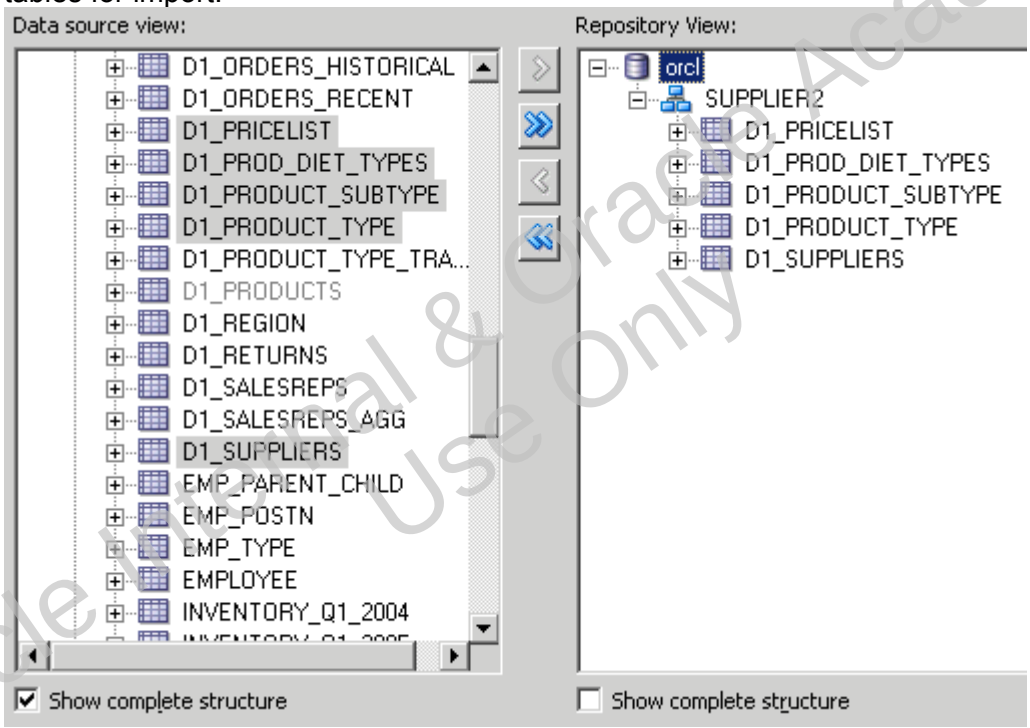


q. Click **Cancel** to close the User dialog box.
r. Select **Action > Close** to close Security Manager. The offline ABC repository now has a weblogic user with a logging level set to 2. This will allow you to check the query log as you complete the remaining practices in this course.

2. In this step, you import additional product tables that store product code, pricing, and supplier information. The product dimension is an example of information stored physically in a normalized table structure. Data warehouse design writers such as Ralph Kimball refer to this as “snowflaking a dimension.” Many database administrators regard this as good database design, so this is a very common practice. So far, you have only included the information in the root product table in the logical subject area. After import, you can include information from the other product tables.
   a. In the Physical layer, expand the **orcl** database object.
   b. Right-click the **SUPPLIER CP** connection pool and select **Import Metadata**.
   c. Accept the defaults in the Select Metadata Types window and click **Next**.
   d. In the Data source view pane, expand **SUPPLIER2**.
   e. Select the following tables (press and hold the **Ctrl** key) for import:

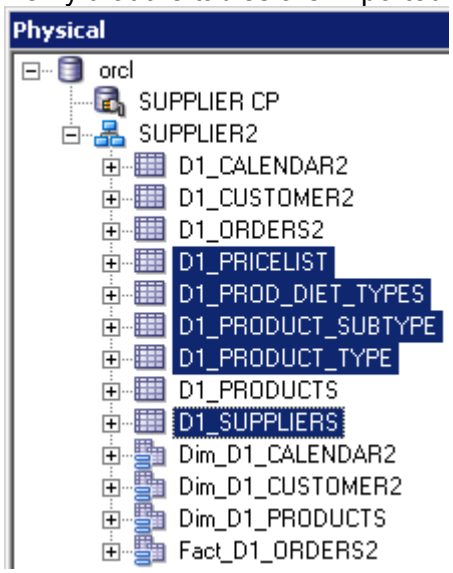
Table Name
D1_PRICELIST
D1_PROD_DIET_TYPES
D1_PRODUCT_SUBTYPE
D1_PRODUCT_TYPE
D1_SUPPLIERS

   f. Confirm that only the five tables are selected and that no higher level objects are selected.
   g. Click the **Import Selected** button to move the tables to the Repository View pane.
   h. Expand **SUPPLIER2** in the Repository View and verify that the new product tables are visible. Deselect **Show complete structure** in the Repository view to see only the tables for import.



   i. Click **Finish**.

j. Verify that the tables are imported into the Physical layer.



k. Update row counts for the new tables to confirm connectivity.

3. Create the following aliases for the imported product tables:

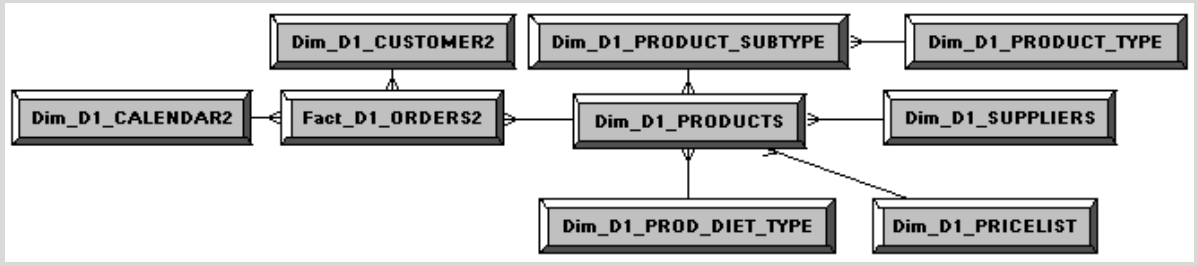
Table	Alias
D1_PRICELIST	**Dim_D1_PRICELIST**
D1_PROD_DIET_TYPES	**Dim_D1_PROD_DIET_TYPES**
D1_PRODUCT_SUBTYPE	**Dim_D1_PRODUCT_SUBTYPE**
D1_PRODUCT_TYPE	**Dim_D1_PRODUCT_TYPE**
D1_SUPPLIERS	**Dim_D1_SUPPLIERS**

4. Define joins and foreign keys by using the Physical Diagram.
   a. In the Physical layer, select all the alias tables.
   b. Right-click any one of the highlighted alias tables and select **Physical Diagram > Selected Object(s) Only** to open the Physical Diagram view.
   c. Drag the new table objects so they are all visible. Right-click the white space to use the zoom and grid features as needed.
   d. Use the **New Foreign Key** button on the toolbar to create the following join relationships:

```
"orcl"."".."SUPPLIER2"."Dim_D1_PRODUCT_SUBTYPE"."SUBTYPECODE" =
"orcl"."".."SUPPLIER2"."Dim_D1_PRODUCTS"."SUBTYPECODE"

"orcl"."".."SUPPLIER2"."Dim_D1_PROD_DIET_TYPES"."DIETCODE" =
"orcl"."".."SUPPLIER2"."Dim_D1_PRODUCTS"."DIETCODE"

"orcl"."".."SUPPLIER2"."Dim_D1_SUPPLIERS"."SUPPLIERCODE" =
"orcl"."".."SUPPLIER2"."Dim_D1_PRODUCTS"."SUPPLIERCODE"

"orcl"."".."SUPPLIER2"."Dim_D1_PRICELIST"."PRODUCTKEY" =
"orcl"."".."SUPPLIER2"."Dim_D1_PRODUCTS"."PRODUCTKEY"
```

```
"orcl"."".."SUPPLIER2"."Dim_D1_PRODUCT_TYPE"."TYPECODE" =
"orcl"."".."SUPPLIER2"."Dim_D1_PRODUCT_SUBTYPE"."TYPECODE"
```

e. Check your results.



f. Close the Physical Diagram.

g. Save the repository. Do not check consistency.

Practice 7-2: Creating Multiple Sources for a Logical Table Source (Manual)

Goal

To add the information from the price list table to the Product dimension

Scenario

You have imported the product tables that store detailed information about ABC's products into the Physical layer of the repository, and configured keys and joins for the tables. So far, the Dim-Product logical table in the Business Model and Mapping layer has only information from the root product table: Dim_D1_PRODUCTS. You now add the information from the price list table to the Dim-Product logical table. This will simplify the data structure, in effect, creating a denormalized logical table.

Outcome

In the Business Model and Mapping layer, the Dim_D1_PRICELIST physical table is added to the existing logical table source for the Dim-Product logical table. The Price logical column is added to the Dim-Product logical table and mapped to the appropriate physical column.

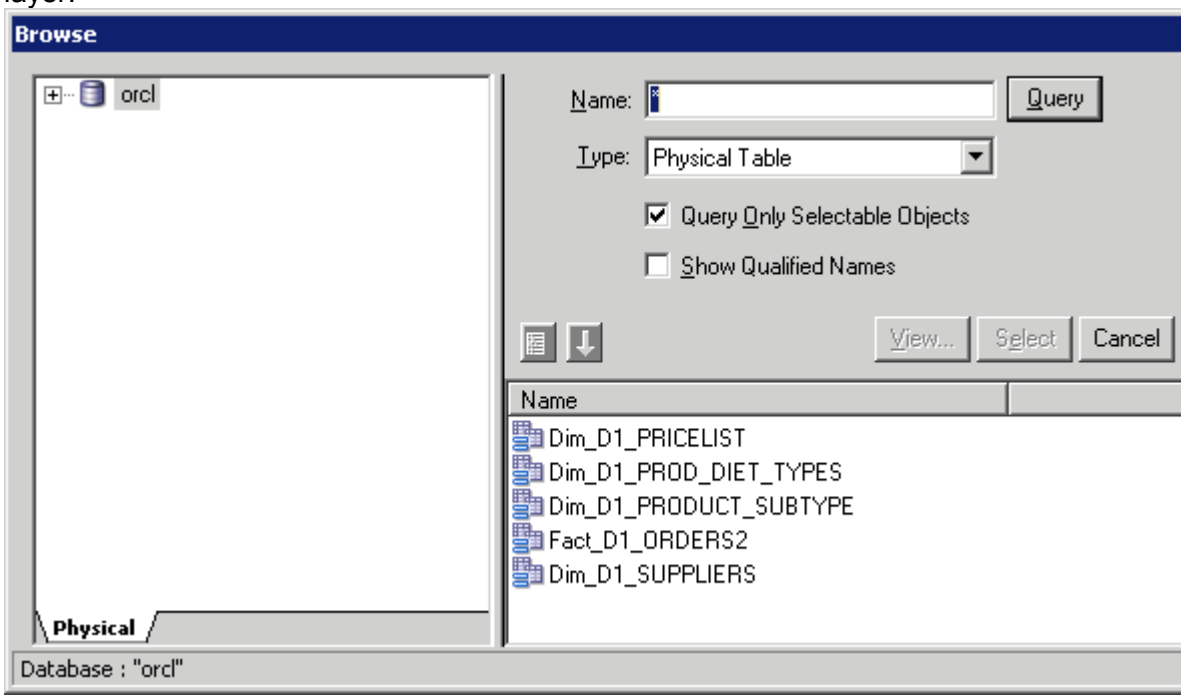
Time

10 minutes

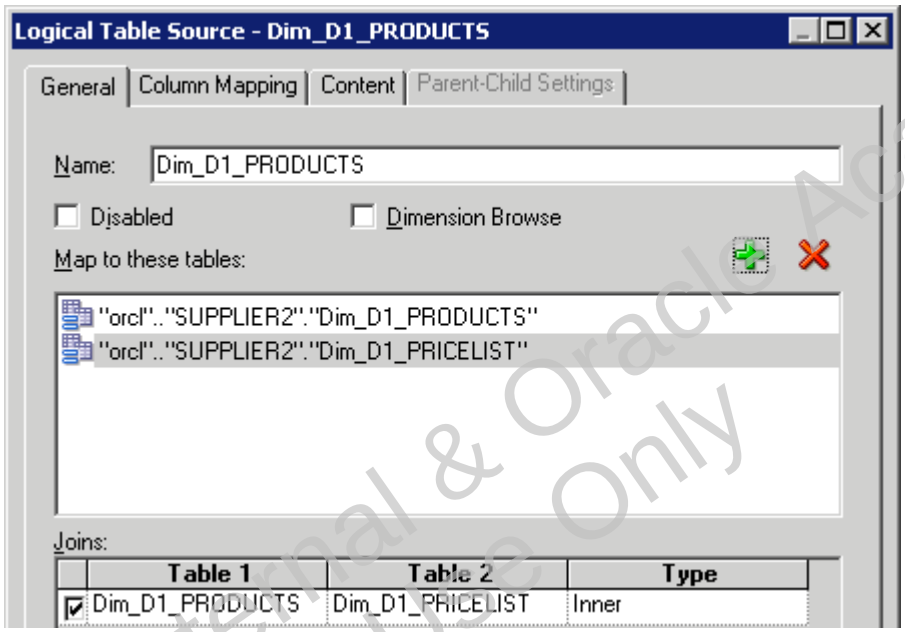
Tasks

1. In this step, you add the information from the price list table to the existing logical table source for the Dim-Product logical table. There are two methods to add multiple sources for an existing logical table source. In this practice, you use the Properties window of an existing logical table source, which is a manual process and requires several steps. In the next practice, you use a more automated process.
   a. In the Business Model and Mapping layer, expand the **Dim-Product** logical table and then the **Sources** folder, and then double-click the **Dim_D1_PRODUCTS** logical table source to view the properties.
   b. Click the **General** tab and click **Add**. The Browse window automatically includes only those tables that are joined directly to the table already in the logical table source. In this case, it includes all tables that join to Dim_D1_PRODUCTS. Only tables that join to tables included in the logical source can be added to the logical source. For example, notice that Dim_D1_PRODUCT_TYPE is not visible in the browse list. This is because it does not have a direct join relationship with Dim_D1_PRODUCTS in the Physical

layer.



c. In the Browse window, select the **Dim_D1_PRICELIST** table and click the **Select** button. The table is added to the logical table source and the join is displayed in the Joins section on the General tab.



d. Select the join in the Joins section. The View Details button becomes active.

e. Click **View Details** to open the Complex Join dialog box and view the read-only details of the join.

f. Click **Cancel** to close the dialog box.

g. To change a join to an outer join, you could use the drop-down list in the Type column. This allows you to change the join type from inner to three kinds of outer joins. For the purpose of these practices, leave the type as Inner. You can think of the tables in a logical table source as being like a database view. When it formulates physical SQL,

Oracle BI Server will leave out the tables in this “view” that are not needed to satisfy the logical query (join elimination), but only if the join type is Inner. When the join type is Outer, however, Oracle BI Server will always include the tables.

h. Click **OK** to close the Logical Table Source dialog box.

2. In this step, you create a new logical column based on the modified logical table source. In the previous step you used the manual method to add a physical table to a logical table source. That method does not add any logical columns to the logical table, nor does it change the logical-to-physical mapping of any existing column. Now that the physical table that stores the pricing information has been added to the Dim-Product logical table source, you create a new logical column and map it to the appropriate physical table and column.

   a. In the Business Model and Mapping layer, right-click **Dim-Product** and select **New Object > Logical Column**.

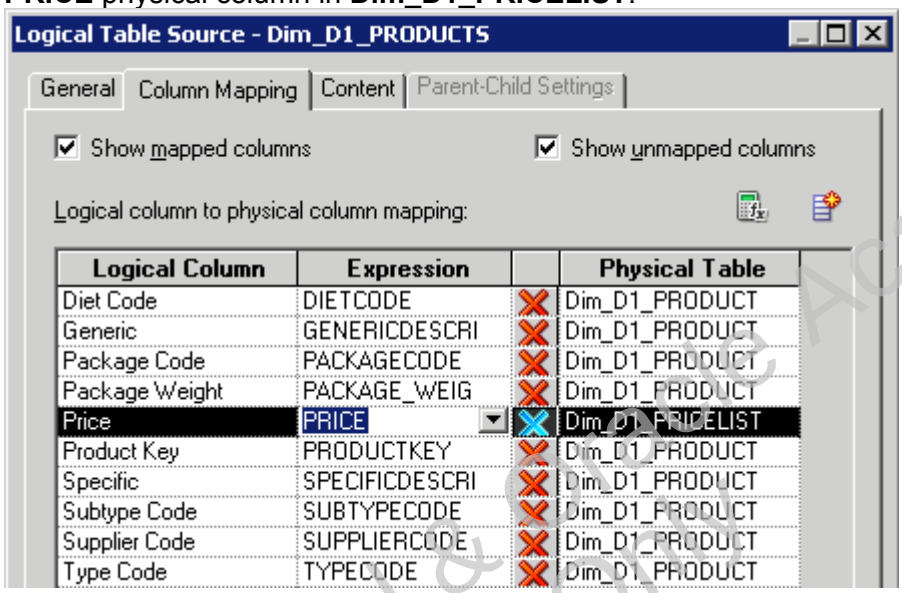
   b. Enter **Price** in the Name field and click **OK**.

   c. Double-click the **Dim_D1_PRODUCTS** logical table source to open its properties dialog box.

   d. Click the **Column Mapping** tab.

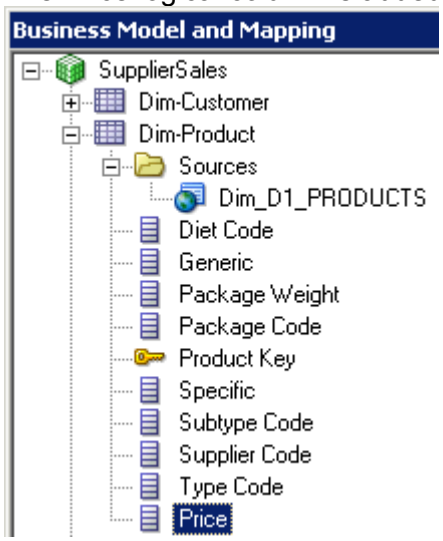
   e. If necessary, select the **Show unmapped columns** check box. Notice that the column you just created, Price, is not mapped to any physical column.

   f. Use the drop-down list in the Expression field to map the **Price** logical column to the **PRICE** physical column in **Dim_D1_PRICELIST**.



   g. Click **OK** to close the Logical Table Source dialog box.

h. The Price logical column is added to the Dim-Product logical table.



i. Save the repository without checking consistency.

Practice 7-3: Creating Multiple Sources for a Logical Table Source (Automated)

Goal

To add the information from the additional product tables to the Product dimension

Scenario

You have manually added information from the price list table to the Dim-Product logical table. Now you add information from the other product tables to the Dim-Product logical table using a more automated method. This automated method adds multiple physical tables to the existing logical table source for the Dim-Product logical table and simultaneously adds logical columns to the Dim-Product logical table.

Outcome

In the Business Model and Mapping layer, the D1_PROD_DIET_TYPES, Dim_D1_PRODUCT_SUBTYPE, Dim_D1_PRODUCT_TYPE, and D1_SUPPLIERS physical tables are added to the logical table source for the Dim-Product logical table. Also, the DIET_TYPE, ITEMSUBTYPE, ITEMTYPE, and ITEMSUPPLIER logical columns are added to the Dim-Product logical table and mapped to the appropriate physical columns.

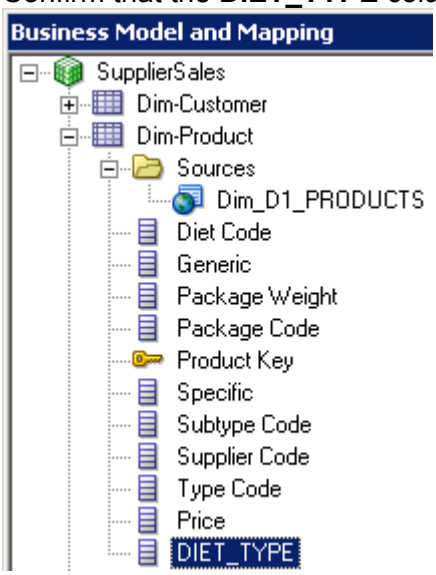
Time

10 minutes

Tasks

1. In this step, you use an automated method to add new logical table sources and logical columns to the Dim-Product logical table.
   a. In the Business Model and Mapping layer, expand **Dim-Product > Sources** so that the **Dim_D1_PRODUCTS** logical table source is visible.
   b. In the Physical layer, expand the **Dim_D1_PROD_DIET_TYPES** table and select the **DIET_TYPE** column.
   c. From the Physical layer, drag the **DIET_TYPE** column onto the **Dim_D1_PRODUCTS** logical table source in the Dim-Product logical table in the Business Model and Mapping layer.

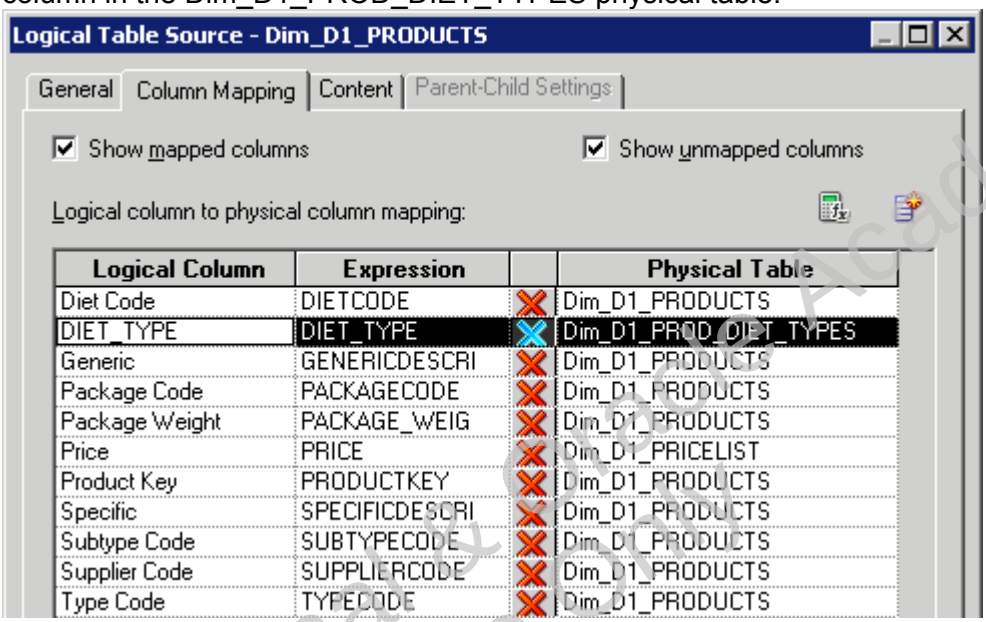
d. Confirm that the **DIET_TYPE** column is added to the Dim-Product logical table.



e. Double-click the **Dim_D1_PRODUCTS** logical table source to view its properties.

f. Click the **Column Mapping** tab.

g. Notice that the DIET_TYPE logical column is mapped to the DIET_TYPE physical column in the Dim_D1_PROD_DIET_TYPES physical table.



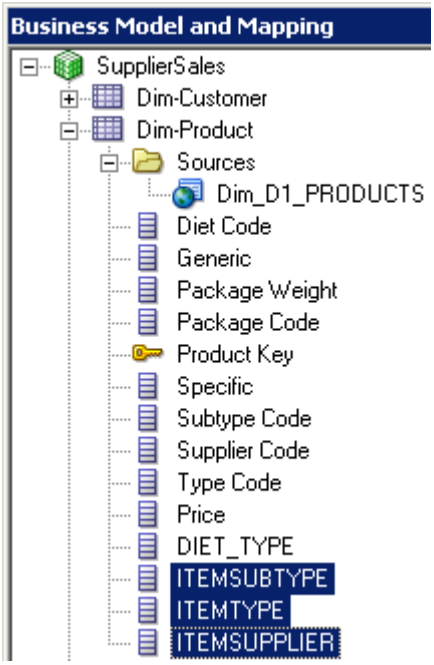
h. Click **OK** to close the Logical Table Source properties dialog box.

2. Repeat the steps to map three more tables and columns to the Dim-Product logical table.

a. Drag the following columns from the Physical layer to the Dim_D1_PRODUCTS logical table source. Be sure to drag to the Dim_D1_PRODUCTS logical table source, not the Dim-Product logical table.

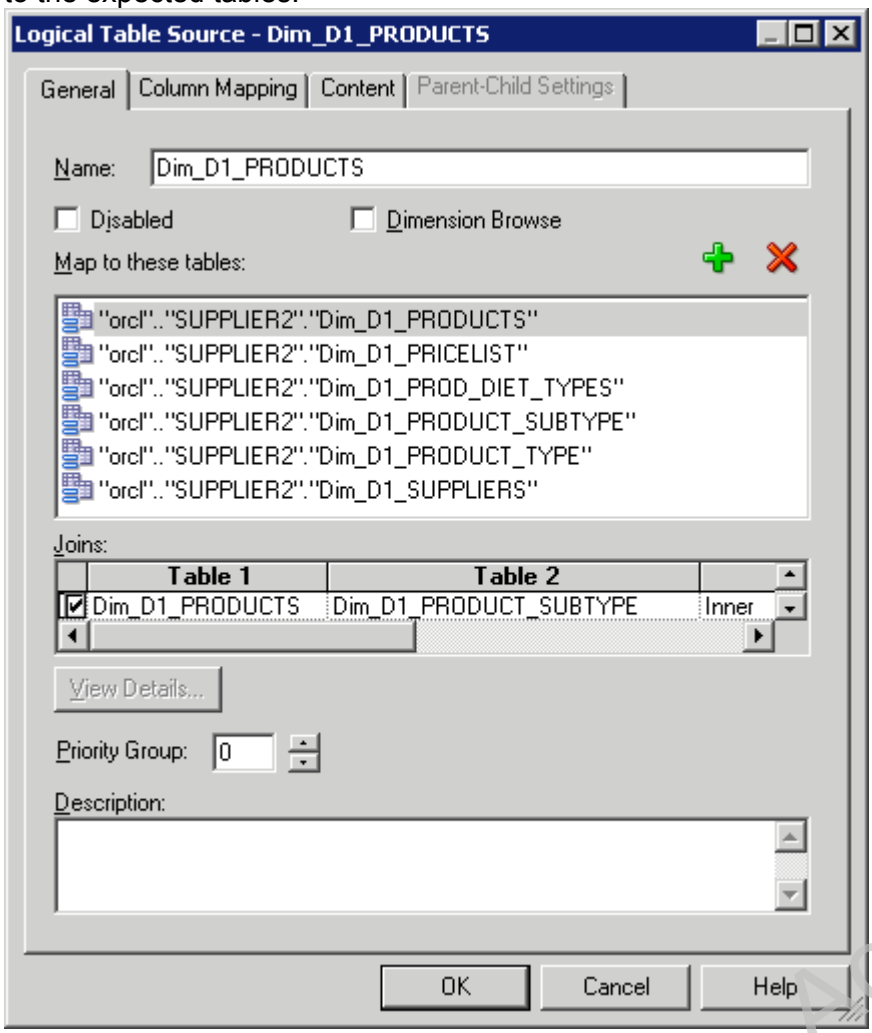
Physical Table	Physical Column
Dim_D1_PRODUCT_SUBTYPE	**ITEMSUBTYPE**
Dim_D1_PRODUCT_TYPE	**ITEMTYPE**
D1_SUPPLIERS	**ITEMSUPPLIER**

b. Confirm that the three columns are added to the Dim-Product logical table.



c. Open the **Dim_D1_PRODUCTS** logical table source properties dialog box, click the **General** tab, and verify that the Dim_D1_PRODUCTS logical table source now maps

to the expected tables.



d. Click the **Column Mapping** tab to verify the logical column to physical column mappings.

Logical Column	Expression		Physical Table
Diet Code	DIETCODE		Dim_D1_PRODUCTS
DIET_TYPE	DIET_TYPE		Dim_D1_PROD_DIET_TYPES
Generic	GENERICDESCRIPTION		Dim_D1_PRODUCTS
ITEMSUBTYPE	ITEMSUBTYPE		Dim_D1_PRODUCT_SUBTYPE
ITEMSUPPLIER	ITEMSUPPLIER		Dim_D1_SUPPLIERS
ITEMTYPE	ITEMTYPE		Dim_D1_PRODUCT_TYPE
Package Code	PACKAGECODE		Dim_D1_PRODUCTS
Package Weight	PACKAGE_WEIGHT		Dim_D1_PRODUCTS
Price	PRICE		Dim_D1_PRICELIST
Product Key	PRODUCTKEY		Dim_D1_PRODUCTS
Specific	SPECIFICDESCRIPTIN		Dim_D1_PRODUCTS
Subtype Code	SUBTYPECODE		Dim_D1_PRODUCTS
Supplier Code	SUPPLIERCODE		Dim_D1_PRODUCTS
Type Code	TYPECODE		Dim_D1_PRODUCTS

e. Click **OK** to close the Logical Table Source properties dialog box.

3. Rename the new logical columns in the Dim-Product logical table:

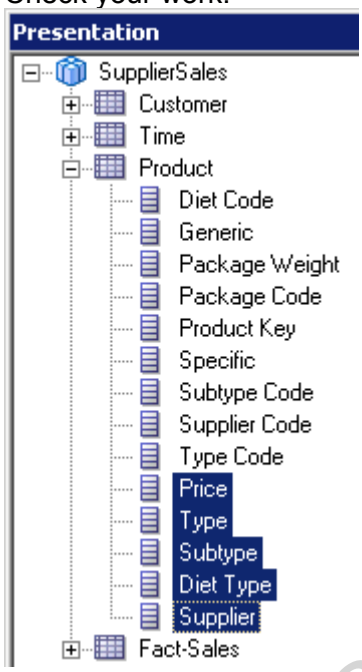
From…	To…

DIET_TYPE	Diet Type
ITEMSUBTYPE	Subtype
ITEMTYPE	Type
ITEMSUPPLIER	Supplier

4. Add the new product information to the SupplierSales presentation catalog.
   a. Drag the five new columns from the Business Model and Mapping layer onto the **Product** table in the **SupplierSales** presentation catalog and reorder the columns in the Presentation layer as follows:

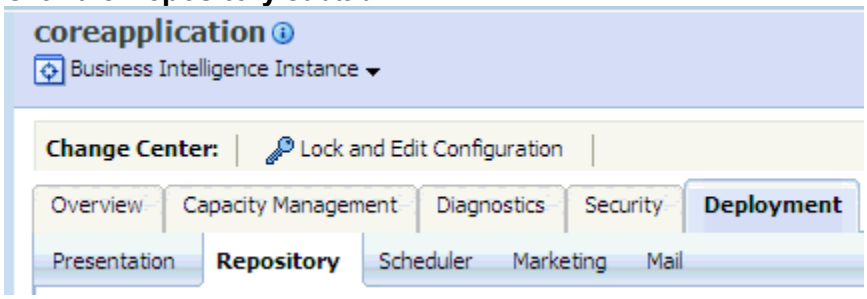
Column Name
Price
Type
Subtype
Diet Type
Supplier

   b. Check your work:

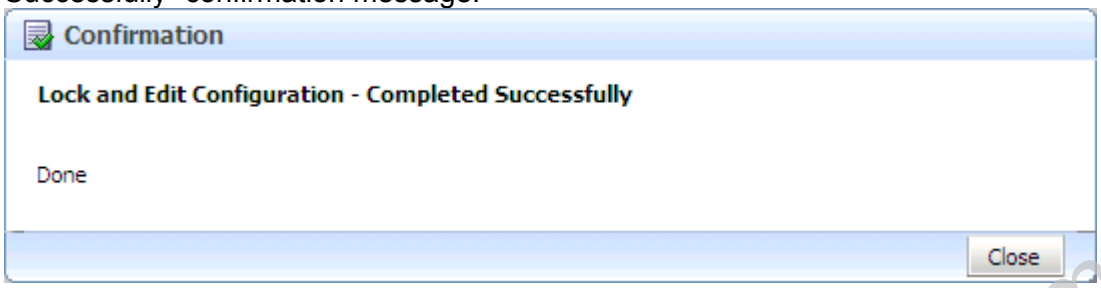


   c. Save the repository.
   d. Check consistency. You should receive a "Consistency check didn't find any errors, warnings or best practice violations." message.
   e. Click **OK** to close the Consistency Check Manager message. If you receive any error or warning messages, fix them before proceeding.
   f. Close the repository.
   g. Leave the Administration Tool open.
5. Use Fusion Middleware Control Enterprise Manager to upload the repository.
   a. Return to Fusion Middleware Control Enterprise Manager, which should still be open in your browser. If it is not open, enter the following URL: **http://localhost:7001/em**.

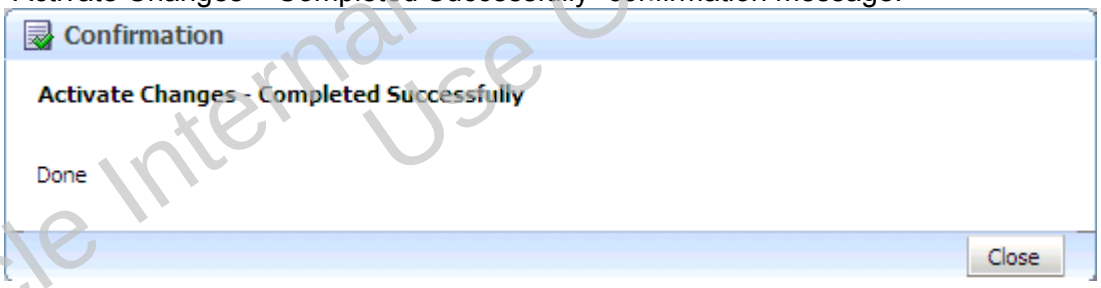
b. If your session has timed out, log in as **weblogic/welcome1**.
c. In the left pane, expand **Business Intelligence** and select **coreapplication**.
d. In the right pane, click the **Deployment** tab.
e. Click the **Repository** subtab.



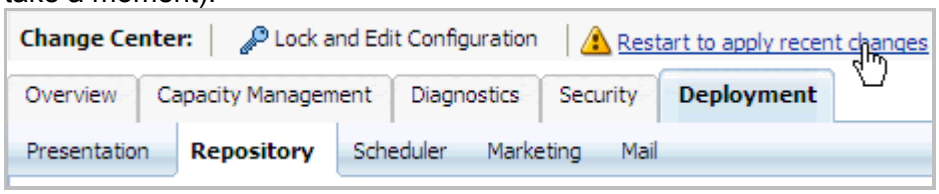
f. Click **Lock and Edit Configuration**.
g. Click **Close** when you receive the "Lock and Edit configuration – Completed Successfully" confirmation message.



h. In the Upload BI Server Repository section, click **Browse** to open the Choose file dialog box.
i. By default, the Choose file dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.
j. Select **ABC.rpd** and click **Open**. You can also double-click the repository to open it.
k. Enter **welcome1** in the Repository Password and Confirm Password fields.
l. Click **Apply**. Notice that Default RPD now displays **ABC** with an extension (for example, ABC_BI0007).
m. Click **Activate Changes**.
n. Allow Active Changes processing to complete. Click **Close** when you receive the "Activate Changes – Completed Successfully" confirmation message.

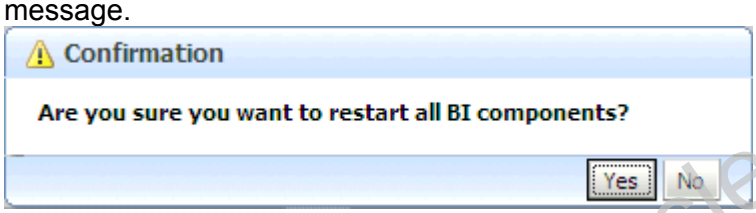


o. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).

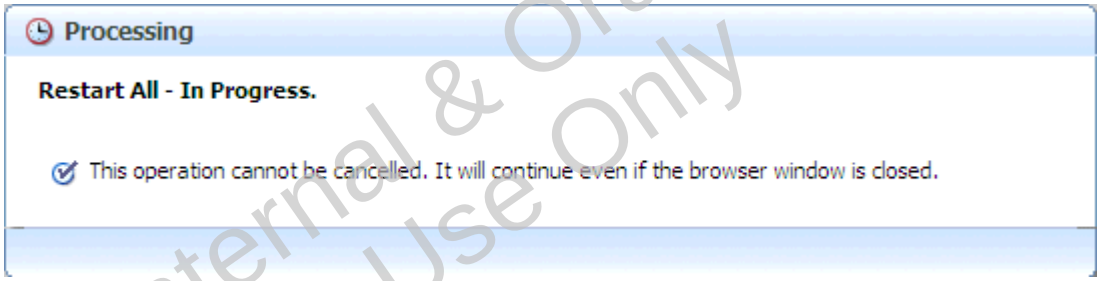


p. On the Overview page, click **Restart**.

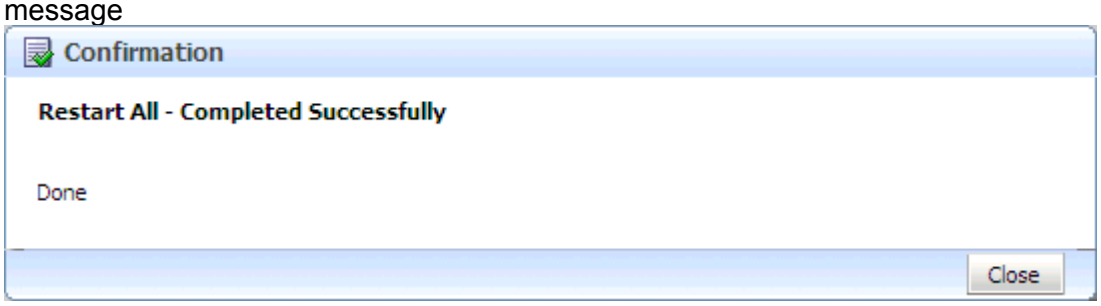
q. Click **Yes** when you receive the "Are you sure you want to restart all BI components?" message.



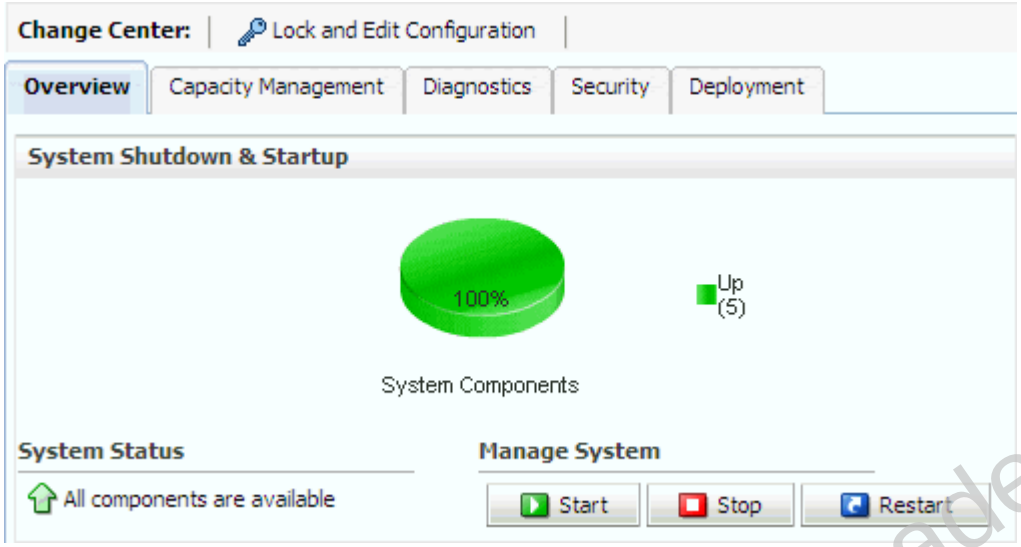
r. Allow the Restart All – In Progress processing to complete. This may take a few moments.



s. Click **Close** when you receive the “Restart All – Completed Successfully” confirmation message

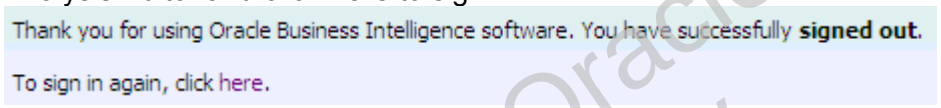


t. Confirm that System Components = 100%. The ABC repository is loaded into BI Server.



u. Leave Enterprise Manager open.

6. Open Analysis Editor to execute queries and test the SupplierSales business model.
   a. Return to the Oracle Business Intelligence browser tab where you signed out of Analysis Editor and click **here** to sign in.



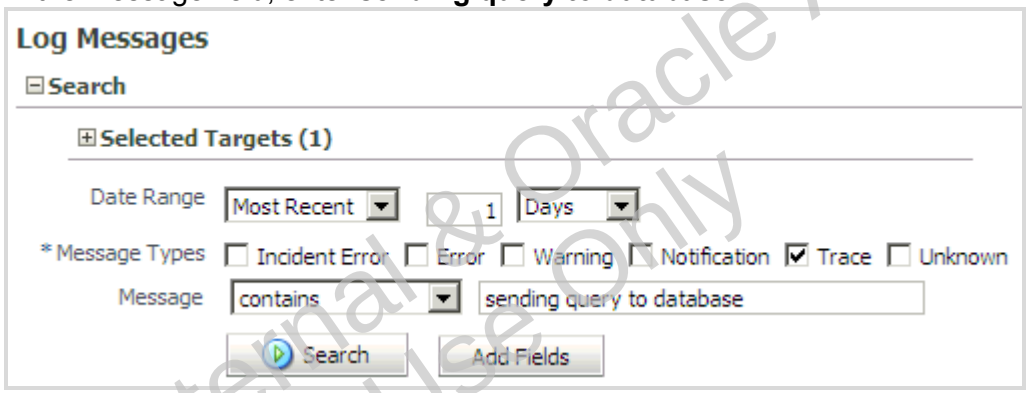
   b. Sign in as **weblogic** with password **welcome1**.
   c. In the Create section, click **Analysis** to open the Select Subject Area window.
   d. Click **SupplierSales** to open Analysis Editor.
   e. Create the following analysis:

f. Click the **Results** tab to view the results.

Type	Type Code
Baking	100
Beef	101
Beverage	102
Bread	103
Cereal	104
Cheese	105
Condiments	106
Dessert	107
Entre	108
Frozen	109
Grains	110
Lamb	111
Non-food	112
Pasta	113
Pork	114
Poultry	115
Rice	116

g. Sign out of Oracle BI.

7. Examine the query log to determine which table or tables have been accessed for this query.
   a. Return to **FMW Enterprise Manager**, which should still be open.
   b. Click the **Diagnostics** tab.
   c. Click the **Log Messages** subtab.
   d. Scroll to the bottom of the window to the **View / Search Log Files** section.
   e. Click **Server Log**.
   f. On the Log Messages screen, leave the data range set to **Most Recent**, **1 Days**.
   g. Deselect all message types except for **Trace**.
   h. In the Message field, enter **sending query to database**.



   i. Click **Search**.
   j. Select the last message in the list. This is the most recent query sent to the database.
   k. In the bottom pane, click the **Collapse Pane button** (arrow on the right side) to view the log message.
   l. Examine the query log. Notice that three tables, Dim_D1_PRODUCTS, Dim_D1_PRODUCT_SUBTYPE, and Dim_D1_PRODUCT_TYPE, have all been accessed, despite the fact that the Dim_D1_PRODUCT_TYPE table contains both

columns. The log should look similar to the following screenshot:

```
SAWITH0 AS (select distinct T99.TYPECODE as c1,
     T502.ITEMTYPE as c2
from
     D1_PRODUCTS T99 /* Dim_D1_PRODUCTS */ ,
     D1_PRODUCT_SUBTYPE T498 /* Dim_D1_PRODUCT_SUBTYPE */ ,
     D1_PRODUCT_TYPE T502 /* Dim_D1_PRODUCT_TYPE */
where  ( T99.SUBTYPECODE = T498.SUBTYPECODE and T498.TYPECODE = T502.TYPEC
select distinct 0 as c1,
     D1.c1 as c2,
     D1.c2 as c3
from
     SAWITH0 D1
order by c3, c2 NULLS FIRST
```

m. Why are all the three tables included in the query? All three tables are included because of the join conditions. The only way the query can access the Dim_D1_PRODUCT_TYPE table is through the D1_PRODUCTS table. In the next practice you learn how to instruct Oracle BI Server to directly access the Dim_D1_PRODUCT_TYPE table via a second logical table source.

n. Leave Enterprise Manager open.

Practice 7-4: Adding a New Logical Table Source

Goal

To add a second logical table source to the Product dimension

Scenario

You examine the physical sources for the Dim-Product logical table and discover that the Type Code and Type columns are mapped to different physical tables. You also discover that the information for both columns is stored in a common physical table, Dim_D1_PRODUCT_TYPE.

In order to model the most economical method for Oracle BI Server to generate queries against these two columns, you add a second logical table source to the Dim-Product logical table so that Oracle BI Server queries only one table for the Type Code and Type columns.

Outcome

In the Business Model and Mapping layer, Type is added as second logical table source for the Dim-Product logical table.

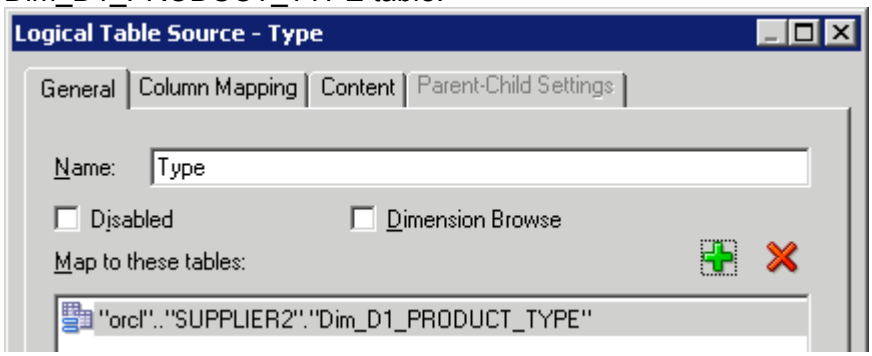
Time

10 minutes

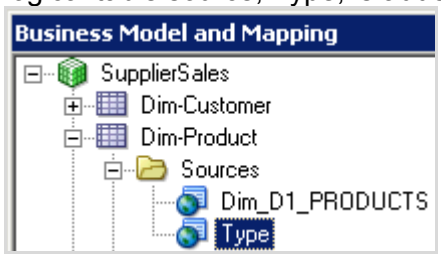
Tasks

1. Return to the Administration Tool, which should still be open, and open the **ABC** repository file in offline mode with the repository password **welcome1**.
2. Examine the existing column mappings for the Type and Type Code columns.
   a. In the Business Model and Mapping layer, expand the **Sources** folder of the Dim-Product logical table
   b. Double-click the **Dim_D1_PRODUCTS** logical table source to open the Logical Table Source dialog box.
   c. Click the **Column Mapping** tab.
   d. Notice that the Type Code logical column is mapped to the TYPECODE physical column in Dim_D1_PRODUCTS.
   e. Notice that the Type logical column is mapped to the ITEMTYPE physical column in Dim_D1_PRODUCT_TYPE.
   f. Click **Cancel** to close the Logical Table Source dialog box.
3. Determine which physical table stores information for both Type Code and Type.
   a. In the Physical layer, expand the **Dim_D1_PRODUCT_TYPE** physical table.
   b. Verify that this table stores information for ITEMTYPE and TYPECODE.
4. Model a new mapping for the Type Code logical column by creating a second logical table source for the Dim-Product logical table.
   a. In the Business Model and Mapping layer, right-click the **Dim-Product** logical table and select **New Object > Logical Table Source**.
   b. On the **General** tab, enter **Type** in the Name field.
   c. Click the **Add** button (green plus sign).
   d. In the Browse dialog box, double-click the **Dim_D1_PRODUCT_TYPE** physical table to select it. The Type logical table source is now mapped to the

Dim_D1_PRODUCT_TYPE table.



e. Click **OK** to close the Logical Table Source properties dialog box. Notice that the new logical table source, Type, is added to the Sources folder.



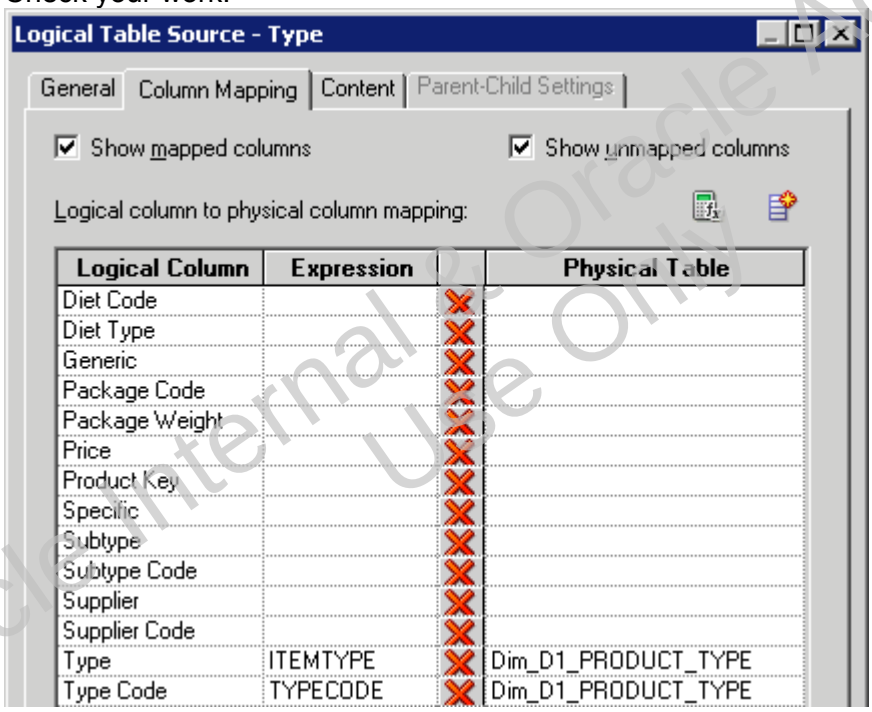
f. Double-click the **Type** logical table source to view its properties.

g. Click the **Column Mapping** tab. Make sure that **Show unmapped columns** is selected.

h. Use the Expression field to map the **Type Code** logical column to the **Dim_D1_PRODUCT_TYPE.TYPECODE** physical column.

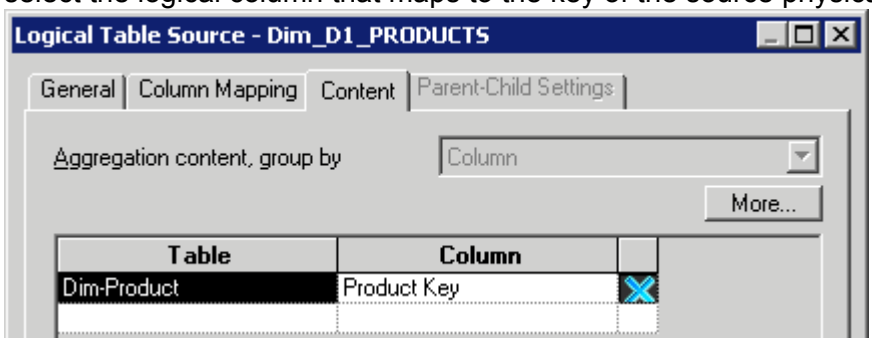
i. Repeat to map the **Type** logical column to the **Dim_D1_PRODUCT_TYPE.ITEMTYPE** physical column.

j. Check your work:



k. Click **OK** to close the Logical Table Source dialog box. There are now two logical table sources for the Dim-Product logical table and the Type Code logical column maps to physical columns in both the Dim_D1_PRODUCT_TYPE and Dim_D1_PRODUCTS tables.

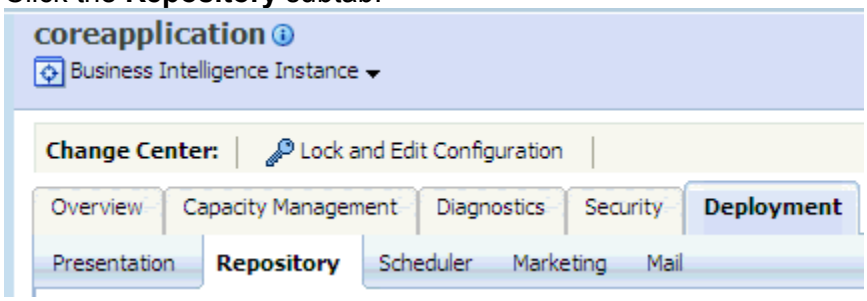
5. Specify the aggregation content for the logical table sources. To use a source correctly, Oracle BI Server has to know what each source contains in terms of the business model. Therefore, you must define aggregation content for each logical table source.
   a. Double-click the **Dim_D1_PRODUCTS** logical table source.
   b. Click the **Content** tab.
   c. Notice that for "Aggregation content, group by" column is selected and grayed out. Although you have the option to specify aggregation content by column or logical level, it is recommended that you use logical levels exclusively. However, for the purpose of demonstrating results in this practice, you define the content by columns. Later, in the "Using Aggregates" lesson, you learn how to define content using logical levels.
   d. In the Table field, select **Dim-Product**.
   e. In the Column field, select **Product Key**. When multiple logical columns could be used, select the logical column that maps to the key of the source physical table.



   f. Click **OK** to close the Logical Table Source dialog box.
   g. Double-click the **Type** logical table source.
   h. On the Content tab, set table to **Dim-Product** and column to **Type Code**.
   i. Click **OK** to close the Logical Table Source dialog box.
   j. Save the repository.
   k. Click **Yes** to check consistency. If you receive any error or warning messages, fix them before proceeding.
   l. If the repository is consistent, close the repository.
   m. Leave the Administration Tool open.

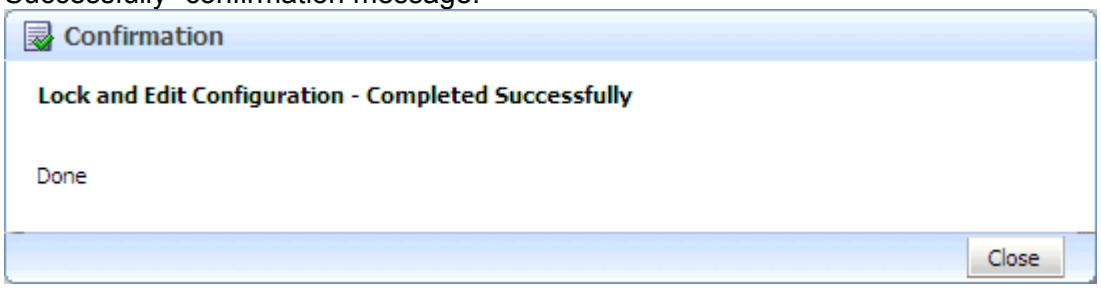
6. Use Fusion Middleware Control Enterprise Manager to upload the repository.
   a. Return to Fusion Middleware Control Enterprise Manager, which should still be open in your browser. If it is not open, enter the following URL: **http://localhost:7001/em**.
   b. If your session has timed out, log in as **weblogic/welcome1**.
   c. In the left pane, expand **Business Intelligence** and select **coreapplication**.
   d. In the right pane, click the **Deployment** tab.

e. Click the **Repository** subtab.



f. Click **Lock and Edit Configuration**.

g. Click **Close** when you receive the "Lock and Edit configuration – Completed Successfully" confirmation message.



h. In the Upload BI Server Repository section, click **Browse** to open the "Choose file" dialog box.

i. By default, the "Choose file" dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.

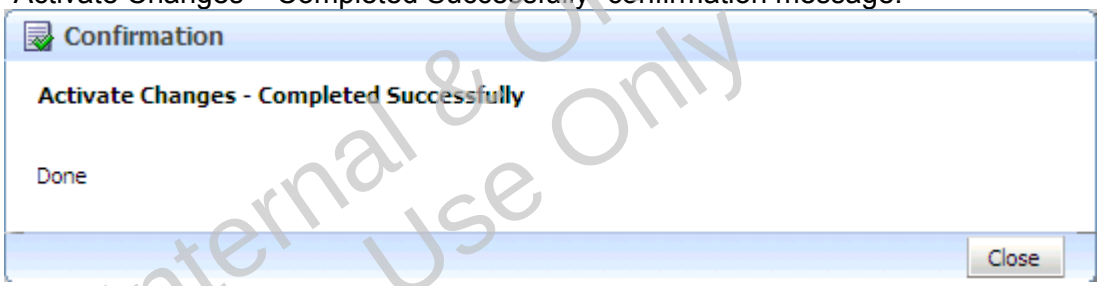
j. Select **ABC.rpd** and click **Open**. You can also double-click the repository to open it.

k. Enter **welcome1** in the Repository Password and Confirm Password fields.

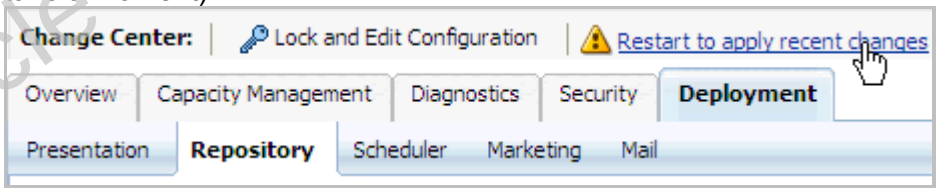
l. Click **Apply**. Notice that Default RPD now displays ABC with an extension (for example, ABC_BI0007).

m. Click **Activate Changes**.

n. Allow Active Changes processing to complete. Click **Close** when you receive the "Activate Changes – Completed Successfully" confirmation message.

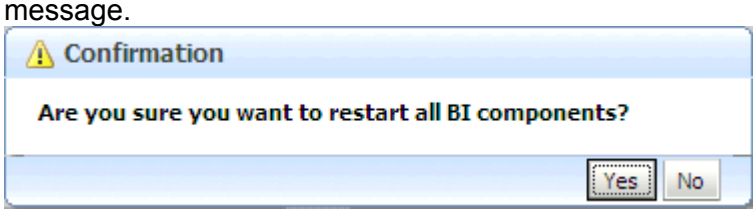


o. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).

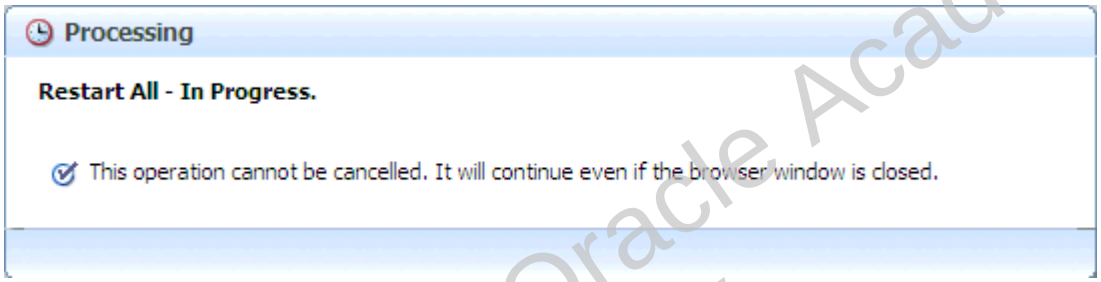


p. On the Overview page, click **Restart**.

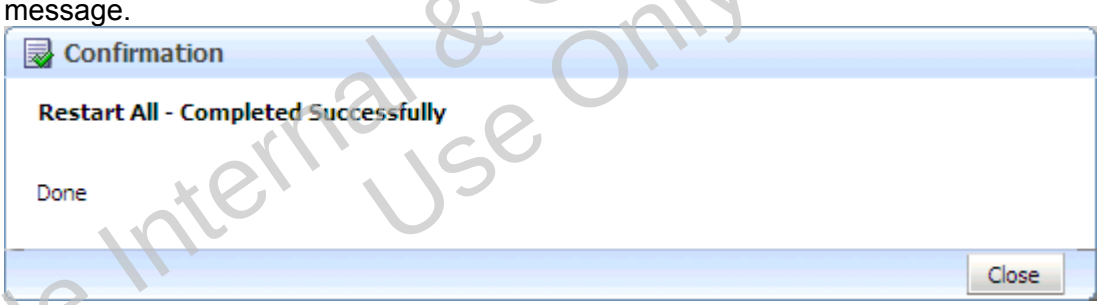
q. Click **Yes** when you receive the "Are you sure you want to restart all BI components?" message.



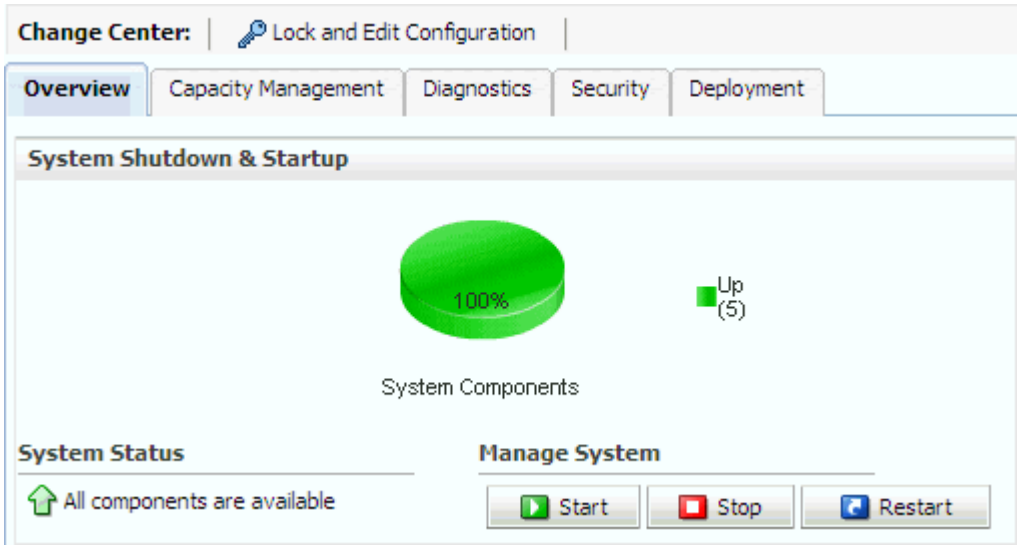
r. Allow the Restart All – In Progress processing to complete. This may take a few moments.



s. Click **Close** when you receive the "Restart All – Completed Successfully" confirmation message.

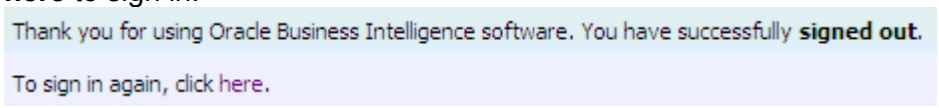


t. Confirm that System Components = 100%. The ABC repository is loaded into BI Server.

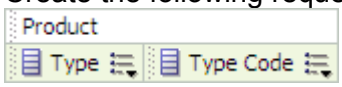


u. Leave Enterprise Manager open.

7. Open Analysis Editor to execute queries and test the SupplierSales business model.
   a. Return to the Oracle Business Intelligence browser tab where you signed out and click **here** to sign in.



   b. Sign in as **weblogic** with password **welcome1**.
   c. In the Create section, click **Analysis** to open the Select Subject Area window.
   d. Click **SupplierSales** to open Analysis Editor.
8. Create the same request as in the previous practice to check your work.
   a. Create the following request:



b. Click the **Results** tab to view the results.

Type	Type Code
Baking	100
Beef	101
Beverage	102
Bread	103
Cereal	104
Cheese	105
Condiments	106
Dessert	107
Entre	108
Frozen	109
Grains	110
Lamb	111
Non-food	112
Pasta	113
Pork	114
Poultry	115
Rice	116

c. Sign out of Oracle BI.

9. Examine the query log to determine which table or tables have been accessed for this query.
   a. Return to the **Diagnostics** tab in Enterprise Manager, which should still be open.
   b. Click the **Log Messages** subtab.
   c. Examine the query log. Verify that only one table, Dim_D1_PRODUCT_TYPE, is accessed by the query. Because you defined the aggregation content in the logical table sources for the Dim-Product logical table, Oracle BI Server executes its query against the most “economical” source, which in this example is the Dim_D1_PRODUCT_TYPE table. To provide the best performance, BI Server bypasses the physical joins and accesses the table directly rather through multiple tables as in the previous example. The log should look similar to the following screenshot:

```
SAWITH0 AS (select T502.TYPECODE as c1,
     T502.ITEMTYPE as c2
from
     D1_PRODUCT_TYPE T502 /* Dim_D1_PRODUCT_TYPE */ )
select distinct 0 as c1,
     D1.c1 as c2,
     D1.c2 as c3
from
     SAWITH0 D1
order by c3, c2
```

   d. Leave Enterprise Manager open.

Practices for Lesson 8: Adding Calculations to a Fact

Lesson 8

Practices for Lesson 8

Lesson Overview

In these practices, you will create calculation measures in an Oracle BI repository.

Practice 8-1: Creating Calculation Measures

Goal

To create calculation measures

Scenario

You use two different methods to create measures that contain calculations. First, you create a measure that is derived from other existing logical columns as a way to apply post-aggregation calculations to the measure. Then you create a measure using physical columns as a way to apply pre-aggregation calculations to the measure. In both cases, you use analyses and the query log to verify your results.

Outcome

A Dollars per Units Ordered logical column with post-aggregation calculations derived from existing logical columns. A Price x Units Ordered logical column with pre-aggregation calculations based on physical columns.

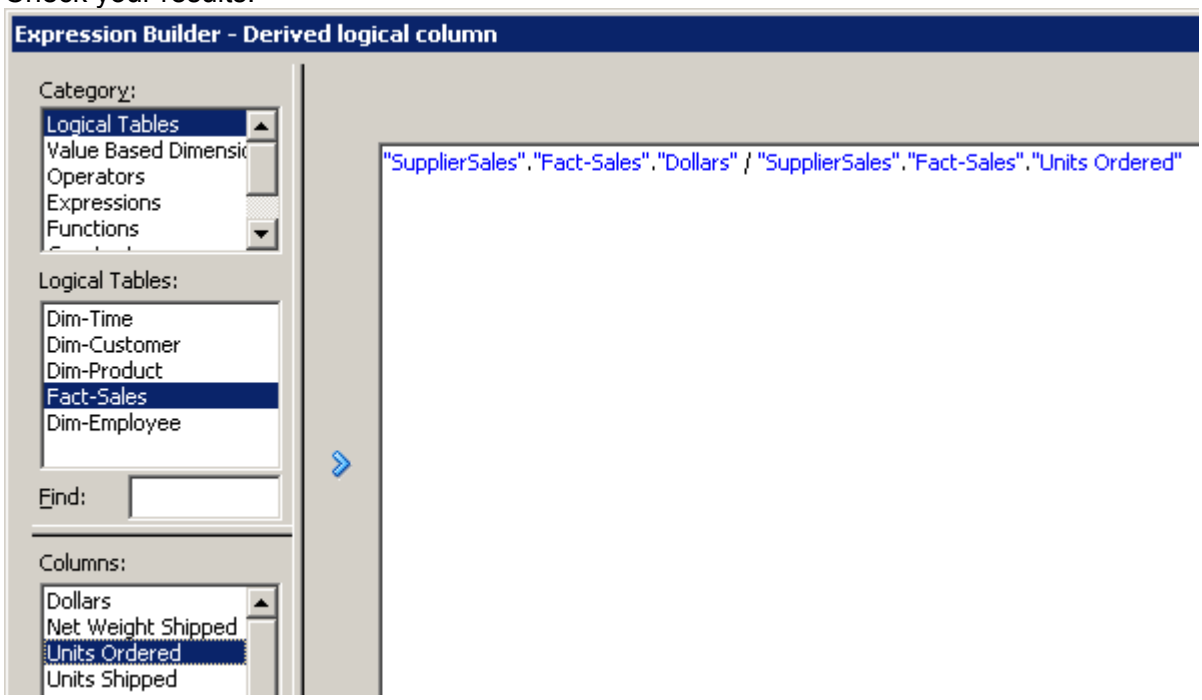
Time

35 minutes

Tasks

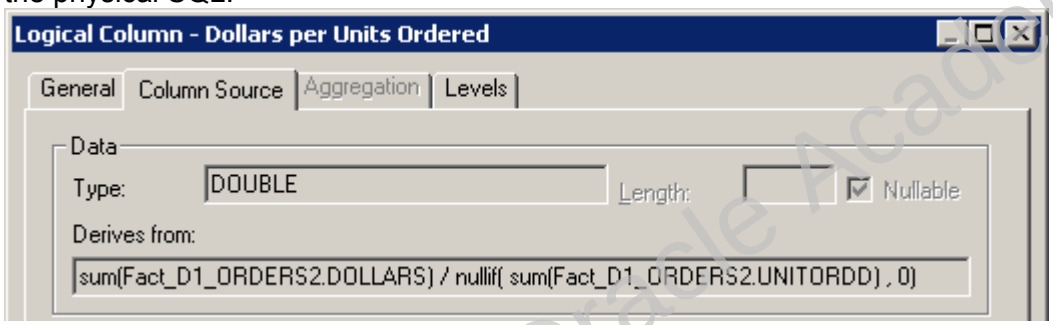
1. In this step, you define a new logical measure that is derived from other existing logical columns as a way to apply post-aggregation calculations to the measure.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with the repository password **welcome1**.
   b. In the Business Model and Mapping layer, right-click the **Fact-Sales** logical table and select **New Object > Logical Column**.
   c. On the General tab, name the column **Dollars per Units Ordered**.
   d. Click the **Column Source** tab.
   e. Select **Derived from existing columns using an expression**.
   f. Click the **Edit Expression** button to open Expression Builder.
   g. Under Category, select **Logical Tables**.
   h. Under Logical Tables, select **Fact-Sales**.
   i. Under Columns, select **Dollars**.
   j. Click the **Insert selected item** arrow to add Dollars to the expression.
   k. Click the **division sign** on the toolbar.
   l. Under Columns, double-click the **Units Ordered** logical column to insert it in the formula.

m. Check your results:



n. Click **OK** to close the Expression Builder. Notice that the expression is now displayed in the text edit box of the Logical Column dialog box.

o. In the "Derives from" field, observe the calculation that will be sent to the data source in the physical SQL.



(Notice that if the PREVENT_DIVIDE_BY_ZERO parameter is set to YES in NQSConfig.INI, the Oracle BI Server prevents errors in divide-by-zero situations, even for analysis column calculations. The Oracle BI Server creates a divide-by-zero prevention expression using nullif() or a similar function when it writes the physical SQL.)

p. Click **OK** to close the Logical Column dialog box. Dollars per Units Ordered is added to the business model.

q. Add the new **Dollars per Units Ordered** column to the **Fact-Sales** table in the **SupplierSales** presentation catalog.

2. To the Fact-Sales logical table source, add mappings that are needed to create a measure that uses physical columns to define a calculation formula.

   a. In the Business Model and Mapping layer, expand **Fact-Sales > Sources**.

   b. In the Physical layer, expand **Dim_D1_PRODUCTS**.

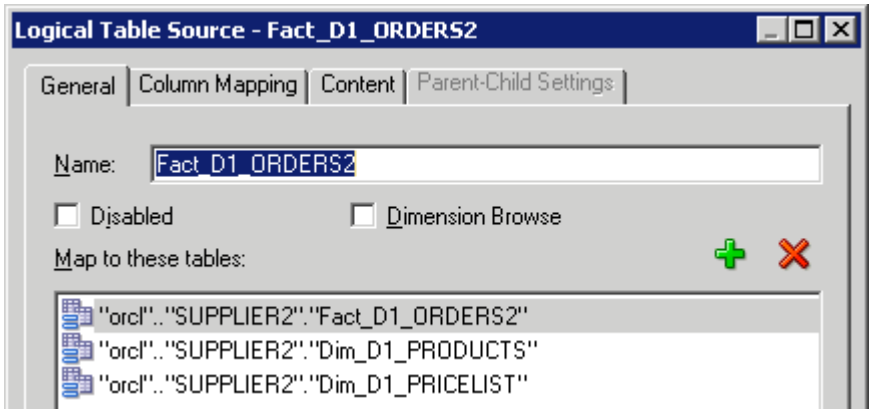
c. Drag **PRODUCTKEY** from Dim_D1_PRODUCTS to **Fact-Sales > Sources > Fact_D1_ORDERS2**. This adds a PRODUCTKEY logical column to the Fact-Sales logical table.

d. In the Physical layer, expand **Dim_D1_PRICELIST**.

e. Drag **PRICE** from Dim_D1_PRICELIST to **Fact-Sales > Sources > Fact_D1_ORDERS2**. This adds a PRICE logical column to the Fact-Sales logical table.

f. Double-click the **Fact_D1_ORDERS2** logical table source to open the Logical Table Source properties dialog box.

g. Click the **General** tab. Notice that as a result of dragging the physical columns to the logical table source, the Fact_D1_ORDERS2 logical table source now maps to three tables.



h. Why is it necessary to map to both product tables? Because of the join relationships. You are going to use the PRICE column in a calculated measure, and Fact_D1_ORDERS2 can only access Dim_D1_PRICELIST.PRICE via Dim_D1_PRODUCTS.

3. Create a measure using physical columns as a way to apply pre-aggregation calculations to the measure.

a. Click the **Column Mapping** tab.

b. Click the **Add New Column** button to open the Logical Column dialog box.

c. Name the column **Price x Units Ordered**.

d. Click the **Aggregation** tab and set the aggregation to **SUM**.

e. Click **OK** to close the logical column dialog box.

f. Click the **Edit Expression** button to open the Expression Builder.

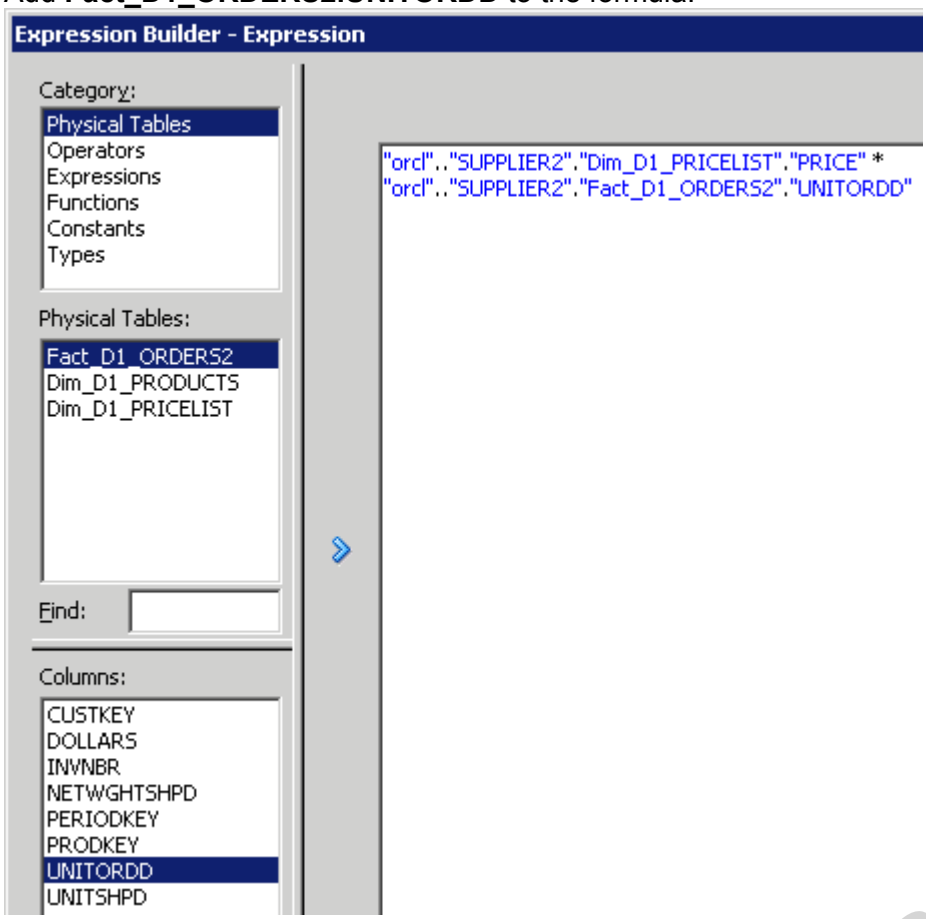
g. Under Category, select **Physical Tables**.

h. Under Physical Tables, select **Dim_D1_PRICELIST**.

i. Under Columns, double-click **PRICE** to add it to the formula.

j. Click the **multiplication icon** on the toolbar to add it to the formula.

k. Add **Fact_D1_ORDERS2.UNITORDD** to the formula.



l. Click **OK** to close the Expression Builder.

m. Notice that you can see the formula in the Expression column for Price x Units Ordered on the Column Mapping tab. You may need to adjust the column width to see the entire formula.

n. Click **OK** to close the Logical Table Source properties dialog box.

o. Double-click the **Price x Units Ordered** logical column.

p. Click the **Column Source** tab and notice that Price x Units Ordered is not derived from existing columns using an expression, but rather from physical mappings. In the Derives from field, notice that the formula multiplies the physical columns first and then sums the result.

q. Click **OK** to close the Logical Column properties dialog box.

r. Add **Price x Units Ordered** to the **Fact-Sales** presentation table in the **SupplierSales** catalog in the Presentation layer.

s. Delete the **PRICE** and **PRODUCTKEY** logical columns from the **Fact-Sales** logical table.

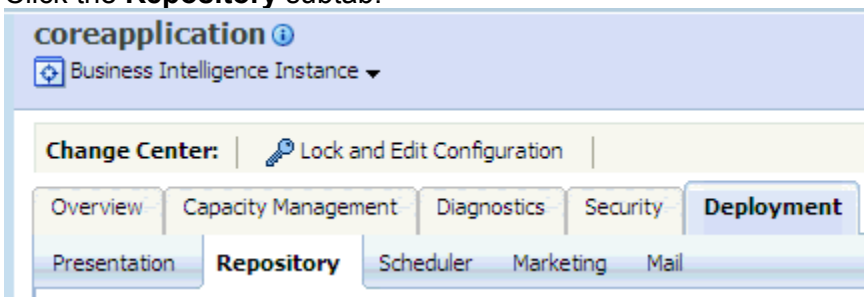
t. Save the repository.

u. Click **Yes** to check global consistency. Fix any errors or warnings before proceeding.

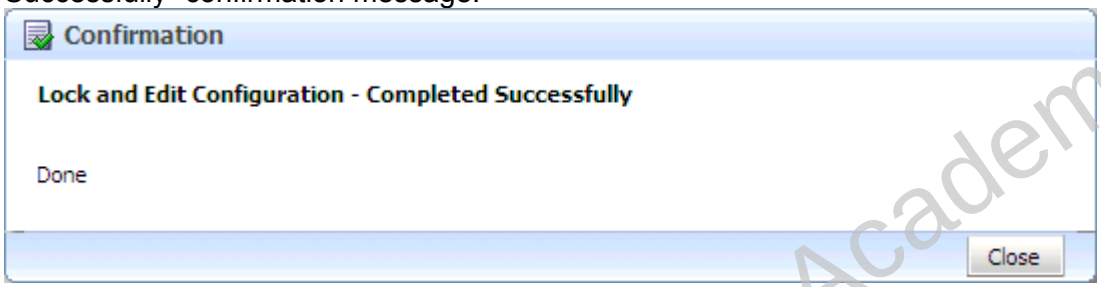
v. Close the repository.

w. Leave the Admin Tool open.

4. Use Fusion Middleware Control Enterprise Manager to upload the repository.
   a. Return to Fusion Middleware Control Enterprise Manager, which should still be open in your browser. If it is not open, enter the following URL: **http://localhost:7001/em**.
   b. If your session has timed out, log in as **weblogic/welcome1**.
   c. In the left pane, expand **Business Intelligence** and select **coreapplication**.
   d. In the right pane, click the **Deployment** tab.
   e. Click the **Repository** subtab.

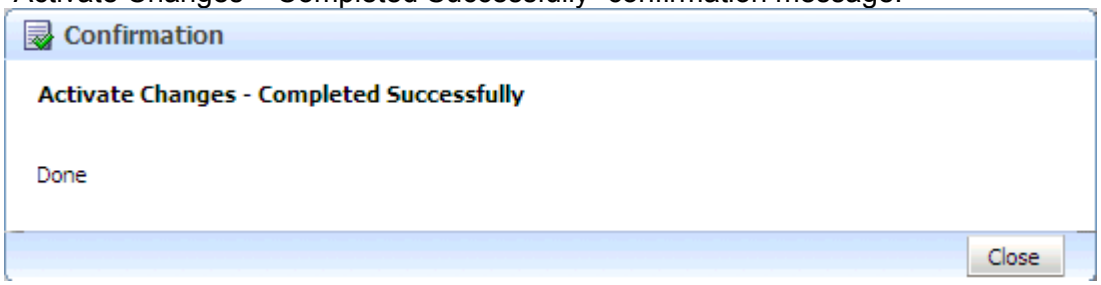


   f. Click **Lock and Edit Configuration**.
   g. Click **Close** when you receive the "Lock and Edit configuration – Completed Successfully" confirmation message.

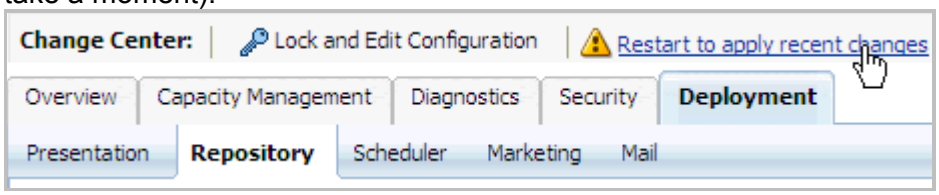


   h. In the Upload BI Server Repository section, click **Browse** to open the "Choose file" dialog box.
   i. By default, the "Choose file" dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.
   j. Select **ABC.rpd** and click **Open**. You can also double-click the repository to open it.
   k. Enter **welcome1** in the Repository Password and Confirm Password fields.
   l. Click **Apply**. Notice that Default RPD now displays **ABC** with an extension (for example, ABC_BI0007).
   m. Click **Activate Changes**.

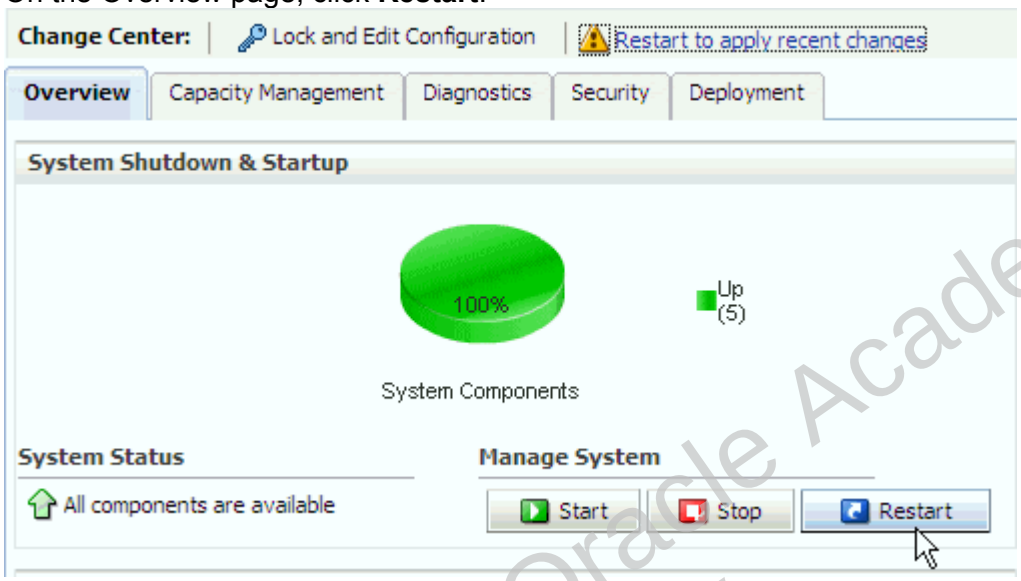
n. Allow Active Changes processing to complete. Click **Close** when you receive the "Activate Changes – Completed Successfully" confirmation message.



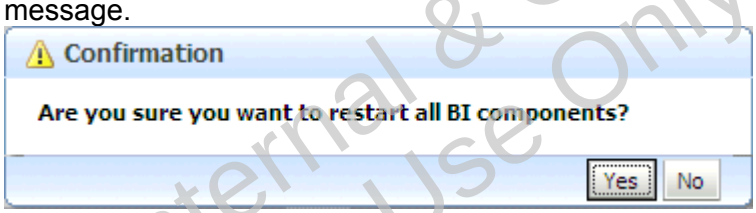
o. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).



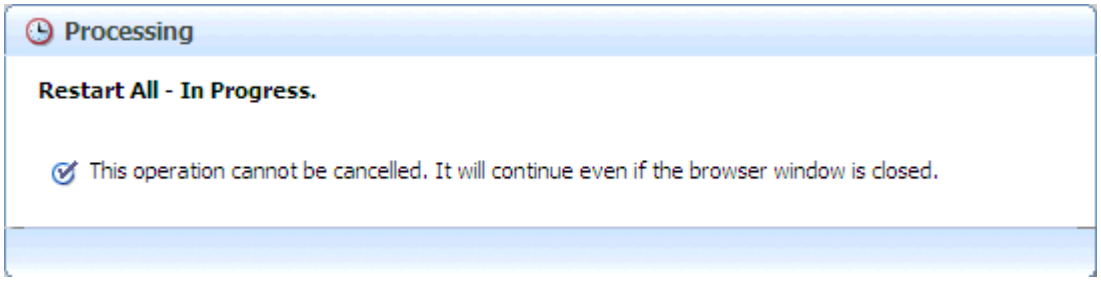
p. On the Overview page, click **Restart**.



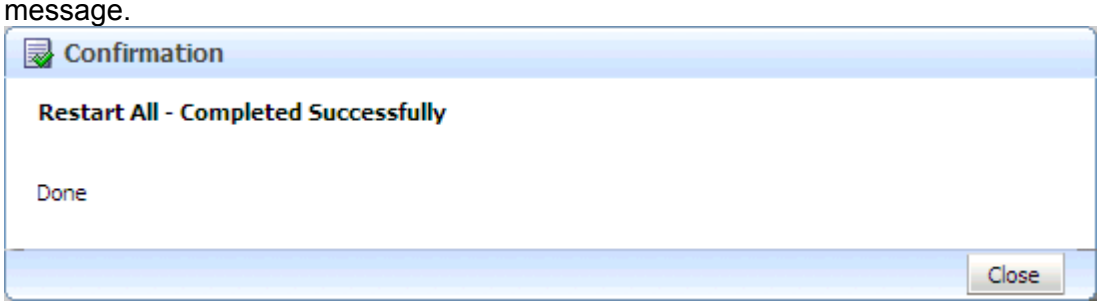
q. Click **Yes** when you receive the "Are you sure you want to restart all BI components?" message.



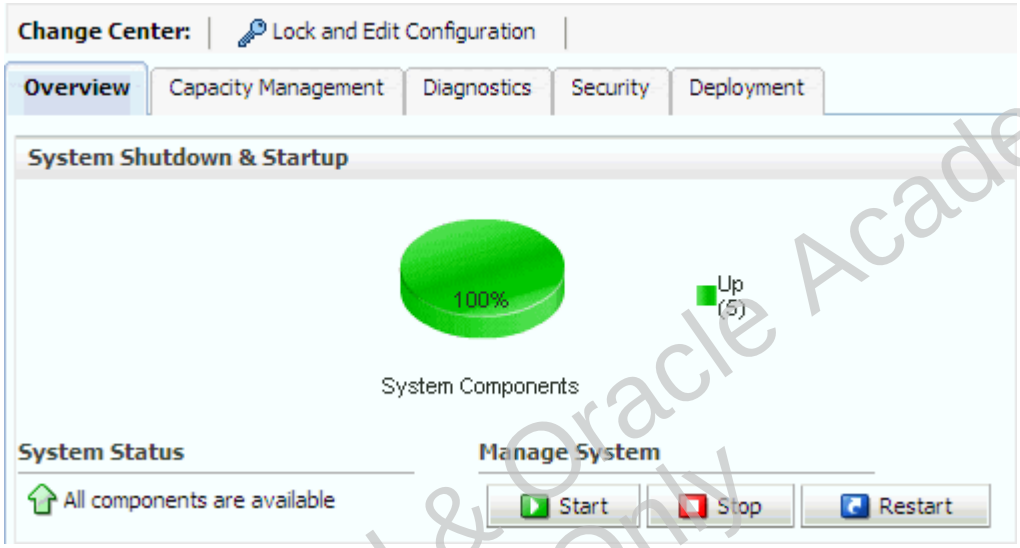
r. Allow the Restart All – In Progress processing to complete. This may take a few moments.



s. Click **Close** when you receive the "Restart All – Completed Successfully" confirmation message.

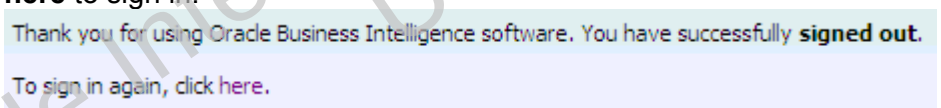


t. Confirm that System Components = 100%. The ABC repository is loaded into BI Server.



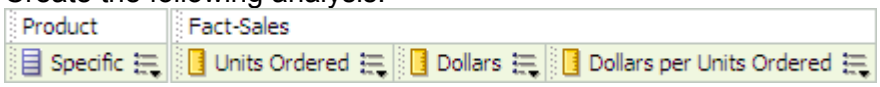
u. Leave Enterprise Manager open.

5. Open Analysis Editor to execute queries and test the SupplierSales business model.
   a. Return to the Oracle Business Intelligence browser tab where you signed out and click **here** to sign in.



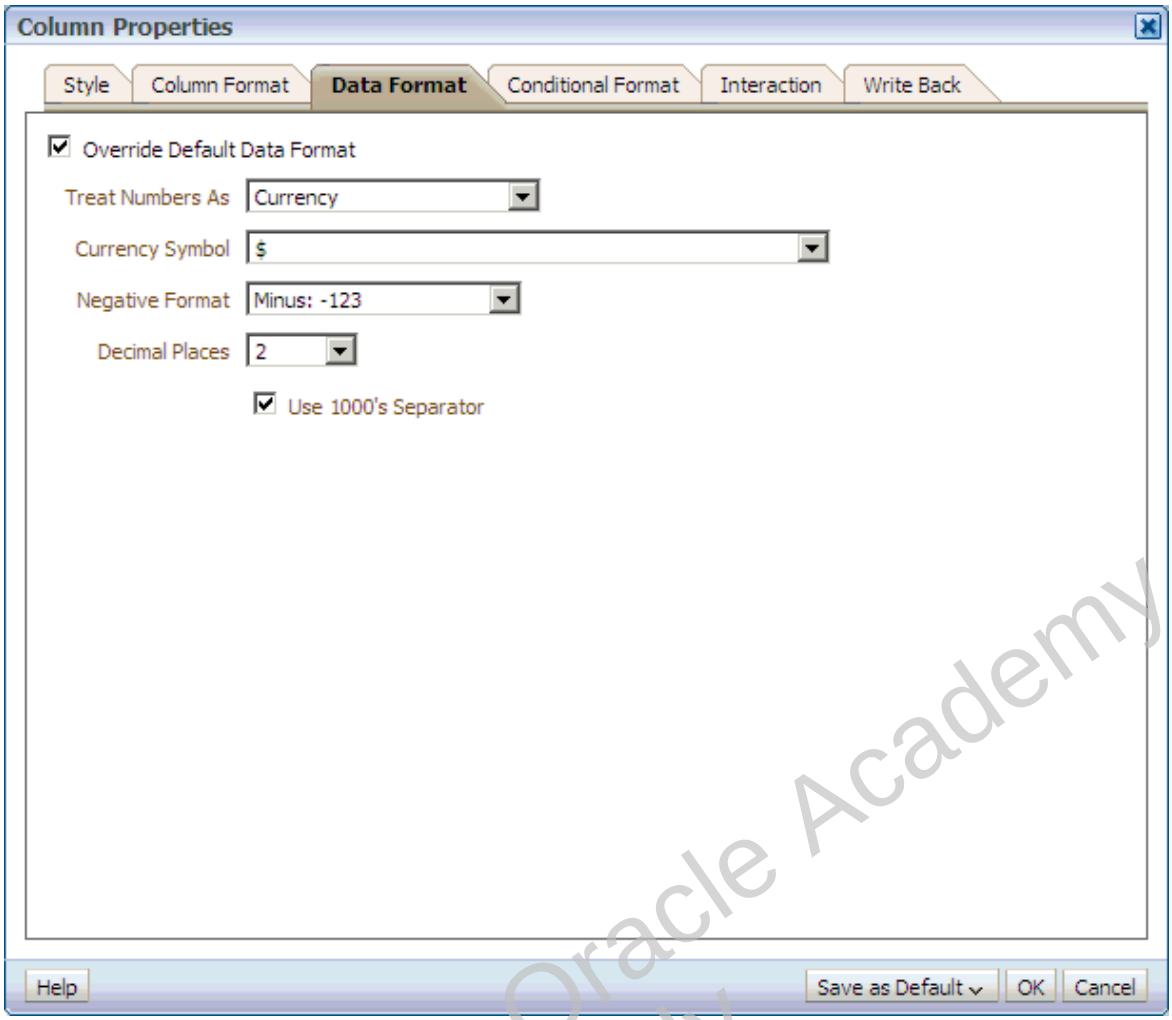
   b. Sign in as **weblogic** with password **welcome1**.
   c. In the Create section, click **Analysis** to open the Select Subject Area window.
   d. Click **SupplierSales** to open Analysis Editor.
6. Create an analysis to check your work for the Dollars per Units Ordered column.

a. Create the following analysis:



b. Select **Column Properties** for **Dollars per Units Ordered**.

c. Click the **Data Format** tab and set the data format using the following screenshot as a reference:



d. Save as the default for Dollars per Units Ordered.

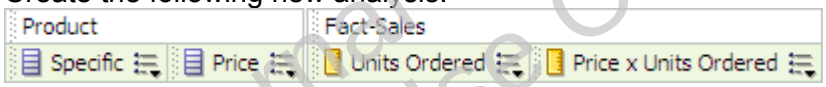
e. Click **Results**.

Specific	Units Ordered	Dollars	Dollars per Units Ordered
"100 Ct Foam Containers 6""x6""x3"""	2,531	$50,192.40	$19.83
"100 Ct Foam Plastic Plates 8"""	9	$950.07	$105.56
"12 Ct Tulip Vase 8"""	5,669	$129,295.02	$22.81
"20 Cu Ft Commercial Upright Freezer Unit w/Child Safety Latch, 4 Shelves"	3	$7,801.01	$2,600.34
"Chef's Knife 8"""	90	$595.81	$6.62
"Napkin/Straw/Condiment Holder Side-mount 16"" x 6 """	6	$7,348.60	$1,224.77
"Padded Bar Stool 36"" Swivel Base"	46	$2,344.31	$50.96
"Pizza Keeper 15"" x 15"""	272	$14,278.79	$52.50
"Push Broom 18"" Wide"	852	$4,698.00	$5.51
"Saute Pan 12"""	1,368	$13,433.06	$9.82
10 Ct Frozen Bread Dough White 2 lb Loaf	336	$12,248.16	$36.45
10 Pak Guest Check Tablet w/Tearoff Receipt 100 Sheets	822	$29,701.40	$36.13
100 Ct 1/3 lb Hamburger Patties	10,538	$1,424,403.25	$135.17

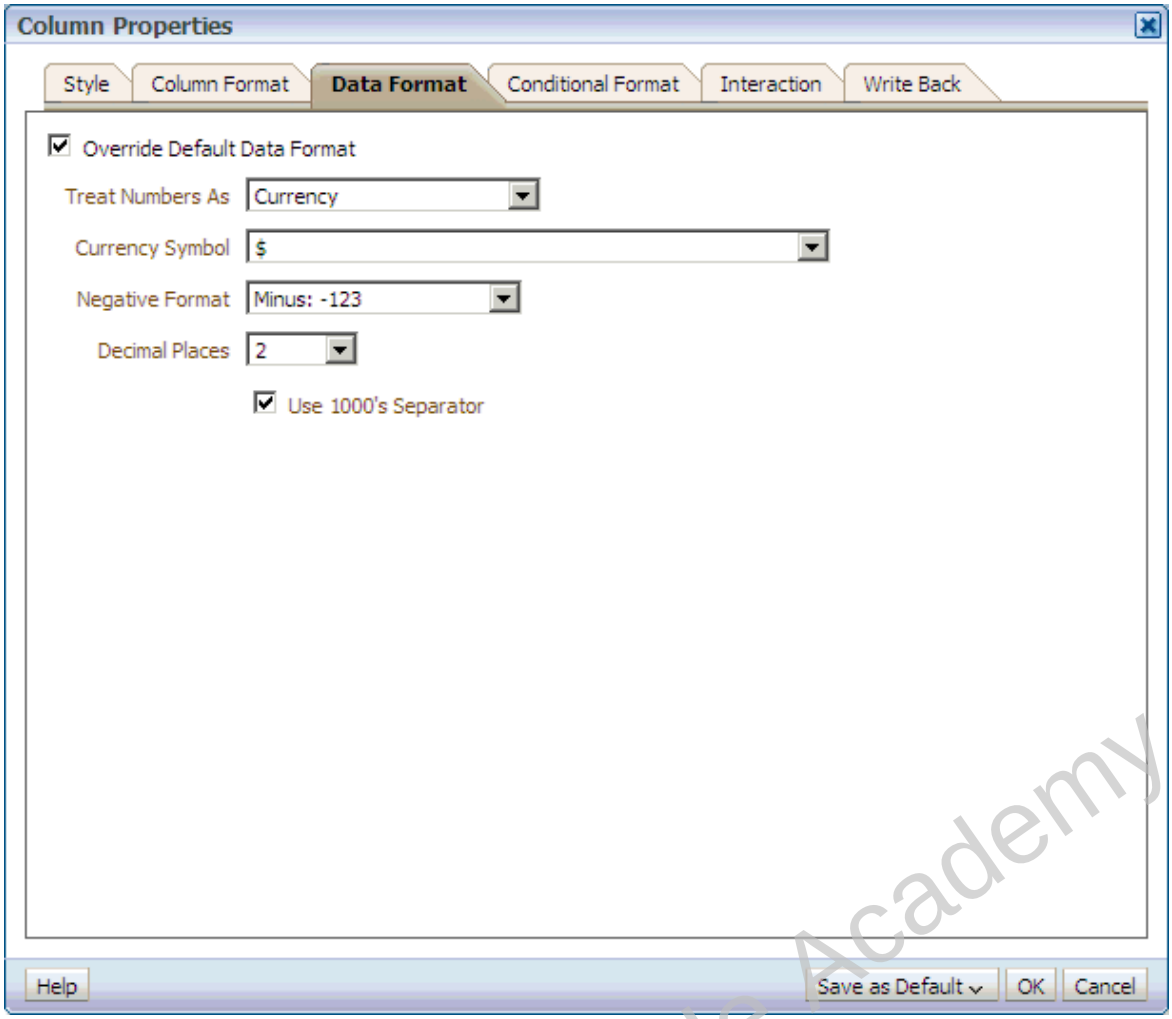
7. Examine the query log.
   a. Return to the **Diagnostics** tab in Enterprise Manager and locate your query. The query log should be similar to the following:

```
SAWITH0 AS (select sum(T90.UNITORDD) as c1,
     sum(T90.DOLLARS) as c2,
     T99.SPECIFICDESCRIPTIN as c3
from
     D1_ORDERS2 T90 /* Fact_D1_ORDERS2 */ ,
     D1_PRODUCTS T99 /* Dim_D1_PRODUCTS */
where  ( T90.PRODKEY = T99.PRODUCTKEY )
group by T99.SPECIFICDESCRIPTIN)
select distinct 0 as c1,
     D1.c3 as c2,
     D1.c2 / nullif( D1.c1, 0) as c3,
     D1.c2 as c4,
     D1.c1 as c5
from
     SAWITH0 D1
order by c2
```

   b. Notice that UNITORDD and DOLLARS are summed first in the query, and then the division of Dollars by Units Ordered is calculated in the outer query block (D1.c2 / nullif (D1.c1, 0) as c3 in this example). Because you defined the Dollars per Units Ordered calculation using logical columns, the columns are summed first and then the division is calculated.
8. Check your work for the Price x Units Ordered column.
   a. Create the following new analysis:



b. Modify the data format for both **Price** and **Price x Units Ordered** using the following screenshot as a reference:



c. Click **Results**.

Specific	Price	Units Ordered	Price x Units Ordered
10 Ct Frozen Bread Dough White 2 lb Loaf	$33.31	336	$11,192.16
100 Ct 1/3 lb Hamburger Patties	$144.58	10,538	$1,523,584.04
100 Ct Asst Children's Snacks 2 oz	$14.32	7,822	$112,011.04
100 Ct Beef Bouillon Cubes .5 oz	$17.23	950	$16,368.50
12 Pak Frozen Chicken 4 oz	$29.43	166,060	$4,887,145.80
12 Pak Frozen Peas 16 oz	$17.44	164,120	$2,862,252.80
2 Pak Coconut Cream Beverage Base Imported 14 oz	$79.10	24	$1,898.40
2 Pak Frank's Mustard 16 oz	$20.46	25,860	$529,095.60
2 Pak Frozen Chicken Wings 5 lbs	$37.18	20,198	$750,961.64
2 Pak Frozen Cornish Game Hens 8 oz	$41.74	115	$4,800.10
2 Pak Frozen Duck in Orange Sauce 6 oz	$84.21	281	$23,663.01
2 Pak Frozen Quail 8 oz	$44.04	13	$572.52
2 Pak Honey Glaze for Ham/Pork 16 oz	$25.65	350	$8,977.50

d. Sign out of Oracle BI.

9. Examine the query log.

a. Open the query log. Your results should resemble the following screenshot:

```
SAWITH0 AS (select sum(T90.UNITORDD) as c1,
     sum(T90.UNITORDD * T490.PRICE) as c2,
     T490.PRICE as c3,
     T99.SPECIFICDESCRIPTIN as c4
from
     D1_ORDERS2 T90 /* Fact_D1_ORDERS2 */ ,
     D1_PRODUCTS T99 /* Dim_D1_PRODUCTS */ ,
     D1_PRICELIST T490 /* Dim_D1_PRICELIST */
where  ( T90.PRODKEY = T99.PRODUCTKEY and T99.PRODUCTKEY = T490.PROD
group by T99.SPECIFICDESCRIPTIN, T490.PRICE)
select distinct 0 as c1,
     D1.c3 as c2,
     D1.c4 as c3,
     D1.c2 as c4,
     D1.c1 as c5
from
     SAWITH0 D1
order by c3, c2
```

b. Notice that the multiplication of units ordered and price is calculated first and then summed: (**`sum(T90.UNITORDD * T490.PRICE) as c2`** in this example). Compare these results with the query results for the calculation that used existing logical columns in the formula, where the columns were summed first and then calculated. What are the advantages and disadvantages of defining a logical column in terms of other logical columns, rather than in terms of the physical sources directly? The advantage of defining a logical column formula based on existing logical columns is that you have to define it only once. When you create formulas based on physical columns, you have to map for each physical source from which it could be derived. However, sometimes you have no choice if you have to use physical columns to apply an aggregation rule after a calculation. What would happen if you deleted a logical column that is used to define the formula of another logical column? The derived column would not be deleted automatically. However, the tool would display an icon ⚠ that warns you about this condition. Don't do this! This is just for informational purposes.

c. Why did deleting the PRICE and PRODUCTKEY logical columns have no impact on the query results for the Price x Units Ordered measure? Because Price x Units Ordered is based on physical columns. The PRICE and PRODUCTKEY columns were added to the Fact-Sales logical table only to create the mappings needed to create the Price x Units Ordered measure.

Practice 8-2: Creating Calculation Measures by Using the Calculation Wizard

Goal

To create calculation measures using the Calculation Wizard

Scenario

Using the Calculation Wizard, you create two calculation measures named Cuts and Percent Not Shipped. The Cuts measure calculates the difference between the units ordered and units shipped. The Percent Not Shipped measure calculates what percentage of the units ordered has not been shipped. The calculation measures created by the Calculation Wizard are based on existing logical columns.

Outcome

In the Business Model and Mapping layer, Cuts and Percent Not Shipped are added to the Fact-Sales logical table.

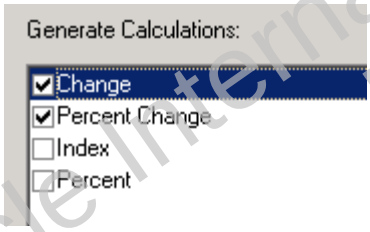
In the Presentation layer, Cuts and Percent Not Shipped are added to the Fact-Sales presentation table.

Time

20 minutes

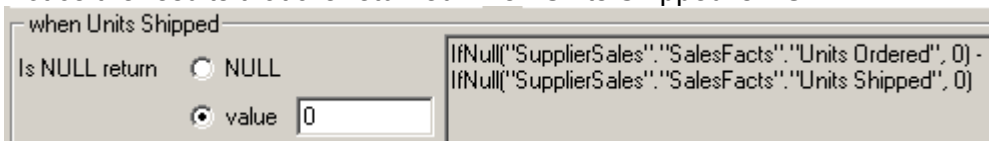
Tasks

1. In this step, you model two calculation measures by using the Calculation Wizard.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository file in offline mode with the repository password **welcome1**.
   b. In the Business Model and Mapping layer, expand the **Fact-Sales** table.
   c. Right-click the **Units Ordered** column and select **Calculation Wizard**.
   d. In the Calculation Wizard – Introduction dialog box, click **Next**.
   e. In the **Choose columns** pane, **Fact-Sales** is selected. The columns that are available to include in the calculation appear in the right pane.
   f. Select the **Units Shipped** check box.
   g. Click **Next**.
   h. In the Generate Calculations section, ensure that the **Change** and **Percent Change** check boxes are both selected.
   i. In the Generate Calculations section, ensure that **Change** is highlighted.

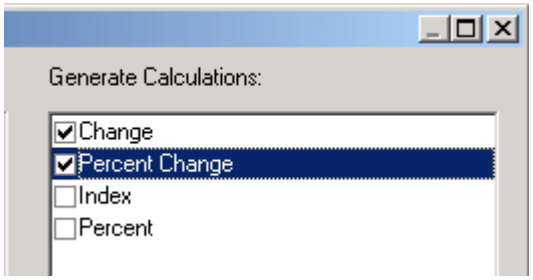


   j. In the Calculation Name field, enter **Cuts**.

k. Notice the results that are returned when Units Shipped is NULL:

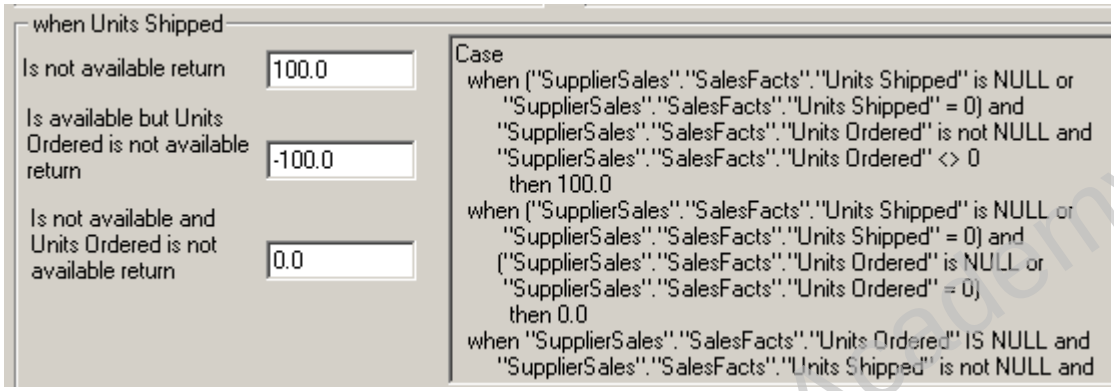


l. In the Generate Calculations section, select **Percent Change** so that it is highlighted.



m. Change the Calculation Name to **Percent Not Shipped**.

n. Notice the results that are returned when Units Shipped is not available (NULL) or is zero using the following parameters. The following screenshot shows only partial results:

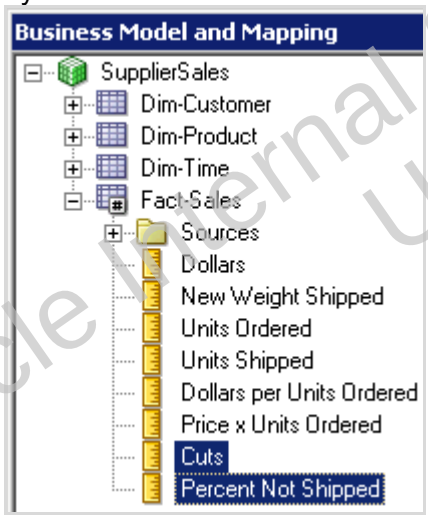


o. Click **Next** to view the Finish window.

p. Review the two calculations measures that will be created by the Wizard: **Cuts** and **Percent Not Shipped**.

q. Click **Finish**.

r. In the Business Model and Mapping layer, confirm that the two new columns created by the Calculation Wizard are visible in the Fact-Sales table.



s. Drag the **Cuts** and **Percent Not Shipped** logical columns to the **Fact-Sales** presentation table in the **SupplierSales** presentation catalog.

t. Save the repository and check global consistency. Fix any errors or warnings before proceeding.

u. Close the repository.

2. Use Fusion Middleware Control Enterprise Manager to upload the repository.

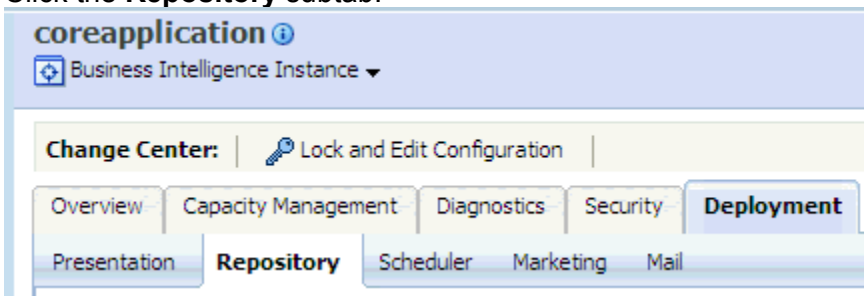
a. Return to Fusion Middleware Control Enterprise Manager, which should still be open in your browser. If it is not open, enter the following URL: **http://localhost:7001/em**.

b. If your session has timed out, log in as **weblogic/welcome1**.

c. In the left pane, expand **Business Intelligence** and select **coreapplication**.

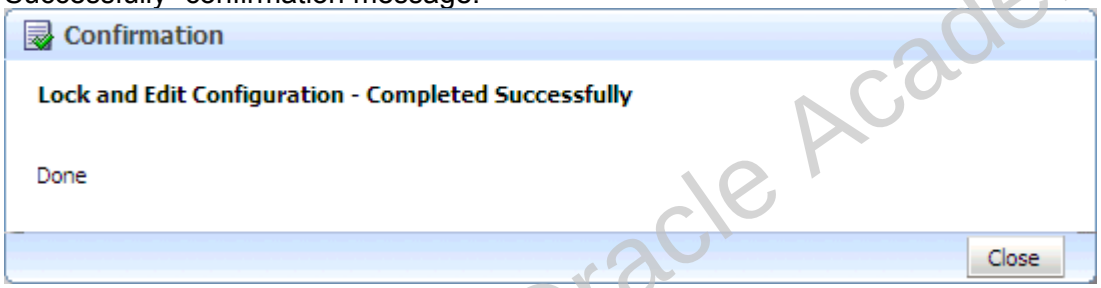
d. In the right pane, click the **Deployment** tab.

e. Click the **Repository** subtab.



f. Click **Lock and Edit Configuration**.

g. Click **Close** when you receive the “Lock and Edit configuration – Completed Successfully” confirmation message.



h. In the Upload BI Server Repository section, click **Browse** to open the “Choose file” dialog box.

i. By default, the “Choose file” dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.

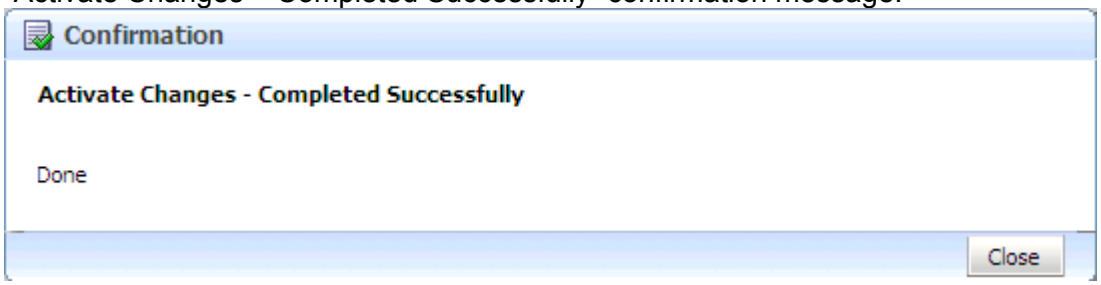
j. Select **ABC.rpd** and click **Open**. You can also double-click the repository to open it.

k. Enter **welcome1** in the Repository Password and Confirm Password fields.

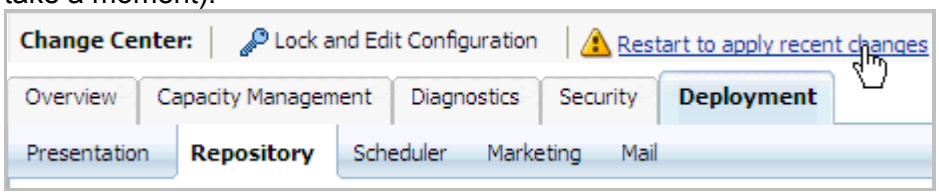
l. Click **Apply**. Notice that Default RPD now displays **ABC** with an extension (for example, ABC_BI0007).

m. Click **Activate Changes**.

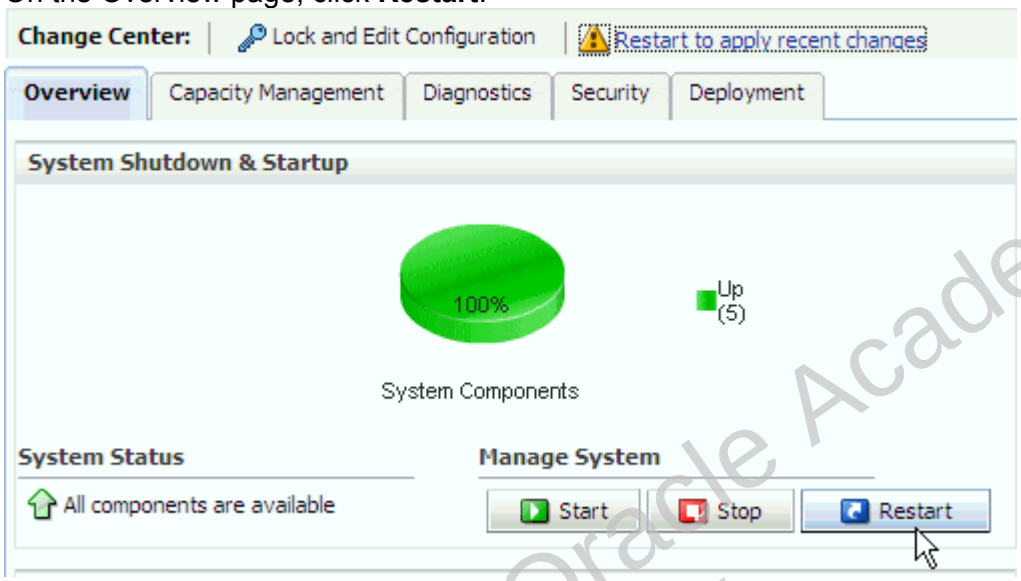
n. Allow Active Changes processing to complete. Click **Close** when you receive the "Activate Changes – Completed Successfully" confirmation message.



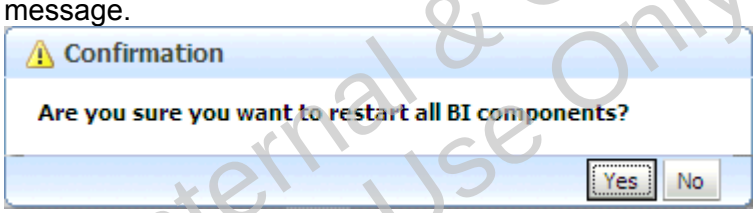
o. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).



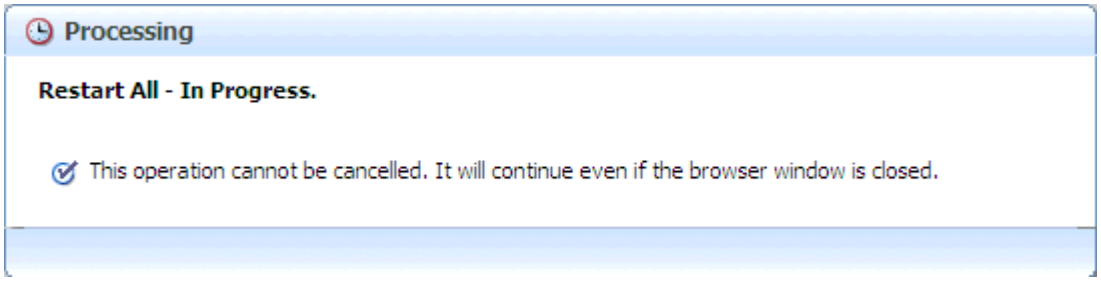
p. On the Overview page, click **Restart**.



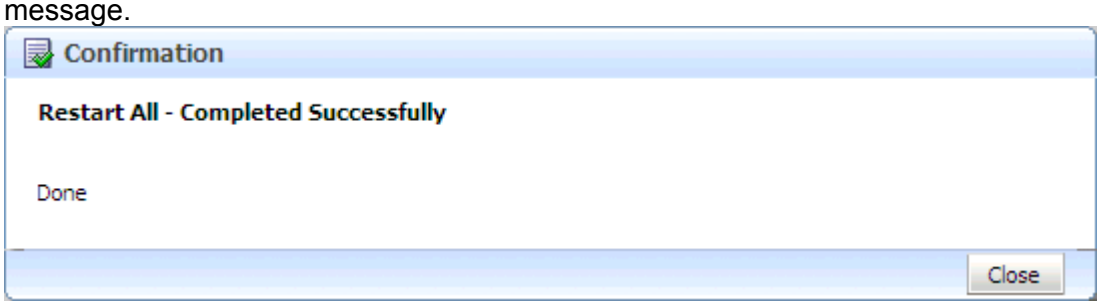
q. Click **Yes** when you receive the "Are you sure you want to restart all BI components?" message.



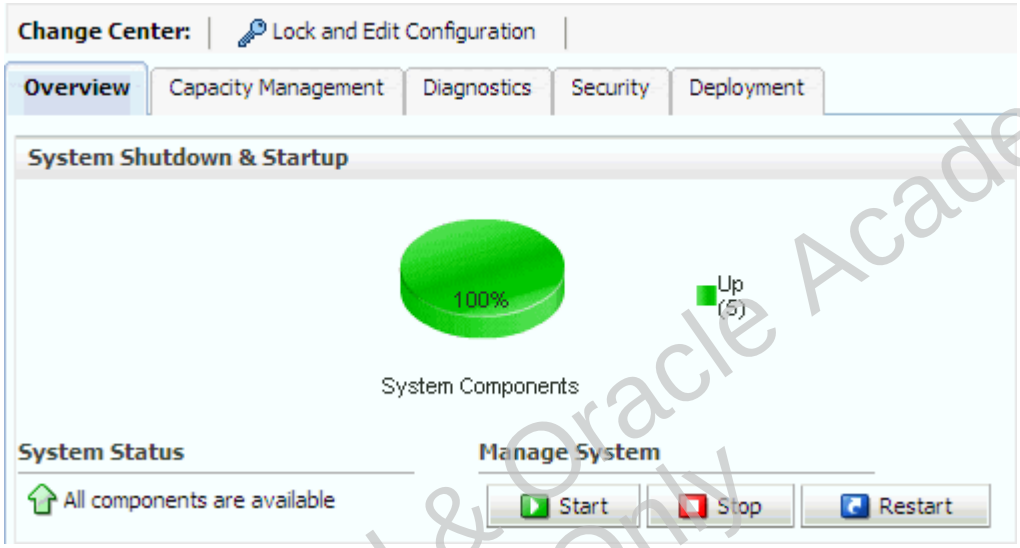
r. Allow the **Restart All – In Progress** processing to complete. This may take a few moments.



s. Click **Close** when you receive the "Restart All – Completed Successfully" confirmation message.

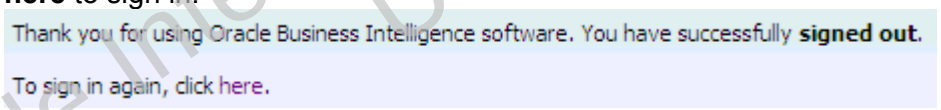


t. Confirm that System Components = 100%. The ABC repository is loaded into BI Server.



u. Leave Enterprise Manager open.

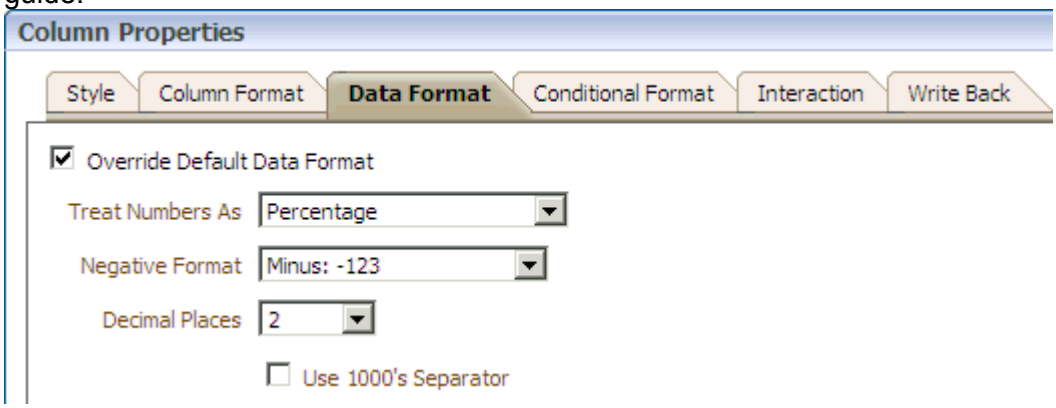
3. Open Analysis Editor to execute queries and test the SupplierSales business model.
   a. Return to the Oracle Business Intelligence browser tab where you signed out and click **here** to sign in.



   b. Sign in as **weblogic** with password **welcome1**.
   c. In the Create section, click **Analysis** to open the Select Subject Area window.
   d. Click **SupplierSales** to open Analysis Editor.
4. Create an analysis to check your work for the Calculation Wizard.

a. Create the following analysis:

b. Modify the data format for Percent Not Shipped using the following screenshot as a guide:



c. Click **Save as Default** and save these settings as the system-wide default for Percent Not Shipped.

d. Click **Results**.

Type	Units Ordered	Units Shipped	Cuts	Percent Not Shipped
Baking	301,154	299,709	1,445	0.48%
Beef	136,299	135,341	958	0.71%
Beverage	244,208	241,179	3,029	1.26%
Bread	86,365	85,490	875	1.02%
Cereal	57,392	56,737	655	1.15%
Cheese	204,358	203,023	1,335	0.66%
Condiments	522,293	523,070	-777	-0.15%
Dessert	122,389	121,282	1,107	0.91%
Entre	47,783	47,158	625	1.33%
Frozen	13	13	0	0.00%
Grains	2,581	2,570	11	0.43%

e. Sign out of Oracle BI.

5. Examine the query log.

a. Your query results should resemble the following screenshot:

```
---------- Sending query to database named orcl (id:
<<2397>>), connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T90.UNITSHPD) as c1,
     sum(T90.UNITORDD) as c2,
     T502.ITEMTYPE as c3
from
     D1_ORDERS2 T90 /* Fact_D1_ORDERS2 */ ,
     D1_PRODUCTS T99 /* Dim_D1_PRODUCTS */ ,
     D1_PRODUCT_SUBTYPE T498 /* Dim_D1_PRODUCT_SUBTYPE */ ,
     D1_PRODUCT_TYPE T502 /* Dim_D1_PRODUCT_TYPE */
where  ( T90.PRODKEY = T99.PRODUCTKEY and T99.SUBTYPECODE =
T498.SUBTYPECODE and T498.TYPECODE = T502.TYPECODE )
group by T502.ITEMTYPE)
select distinct 0 as c1,
     D1.c3 as c2,
     nvl(D1.c2 , 0) - nvl(D1.c1 , 0) as c3,
     case  when D1.c2 <> 0 and (D1.c1 in (0) or D1.c1 is
null) and not D1.c2 is null then 100.0 when (D1.c2 in (0) or
D1.c2 is null) and (D1.c1 in (0) or D1.c1 is null) then 0.0
when D1.c1 <> 0 and not D1.c1 is null and D1.c2 is null then
-100.0 else (D1.c2 - D1.c1) * 100.0 / nullif( D1.c1, 0) end
as c4,
     D1.c2 as c5,
     D1.c1 as c6
from
     SAWITH0 D1
order by c2
```

b. Notice that the Cuts and Percent Not Shipped columns are listed in the logical query.

```
-------------------- SQL Request:
SET VARIABLE QUERY_SRC_CD='Report';SELECT s_0, s_1, s_2, s_3,
s_4, s_5 FROM (SELECT
   0 s_0,
   "SupplierSales"."Product"."Type" s_1,
   "SupplierSales"."Fact-Sales"."Cuts" s_2,
   "SupplierSales"."Fact-Sales"."Percent Not Shipped" s_3,
   "SupplierSales"."Fact-Sales"."Units Ordered" s_4,
   "SupplierSales"."Fact-Sales"."Units Shipped" s_5
FROM "SupplierSales"
```

c. Notice that UNITSHPD and UNITORDD columns are summed first…

```
SAWITH0 AS (select sum(T90.UNITSHPD) as c1,
     sum(T90.UNITORDD) as c2,
```

…and then calculated:

```
select distinct 0 as c1,
     D1.c3 as c2,
     nvl(D1.c2 , 0) - nvl(D1.c1 , 0) as c3,
     case  when D1.c2 <> 0 and (D1.c1 in (0) or D1.c1 is
null) and not D1.c2 is null then 100.0 when (D1.c2 in (0) or
D1.c2 is null) and (D1.c1 in (0) or D1.c1 is null) then 0.0
when D1.c1 <> 0 and not D1.c1 is null and D1.c2 is null then
-100.0 else (D1.c2 - D1.c1) * 100.0 / nullif( D1.c1, 0) end
as c4,
     D1.c2 as c5,
     D1.c1 as c6
```

This is because the Calculation Wizard used logical columns to build the calculations.

Practice 8-3: Creating a Rank Measure

Goal

To create a rank measure

Scenario

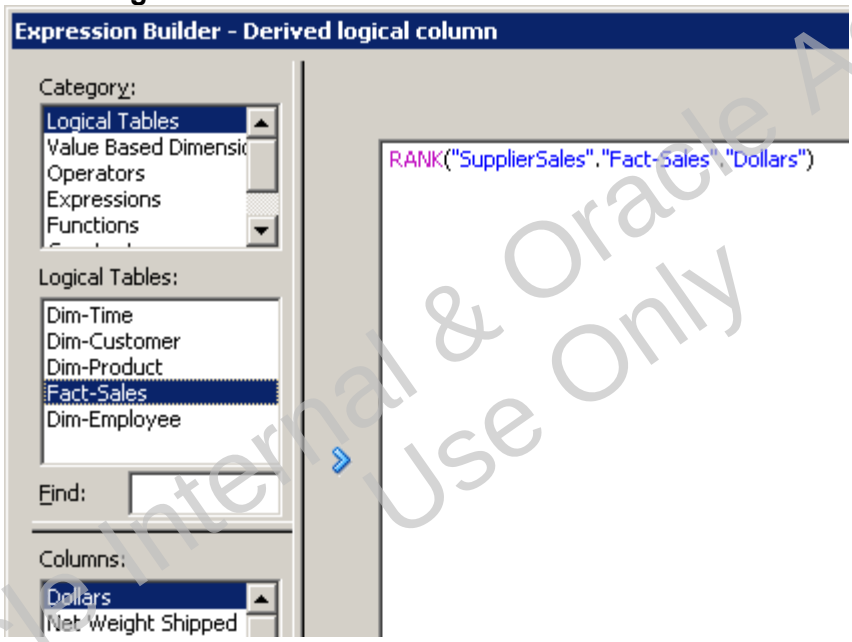
You use the RANK function to calculate rank for the Dollars logical column.

Time

20 minutes

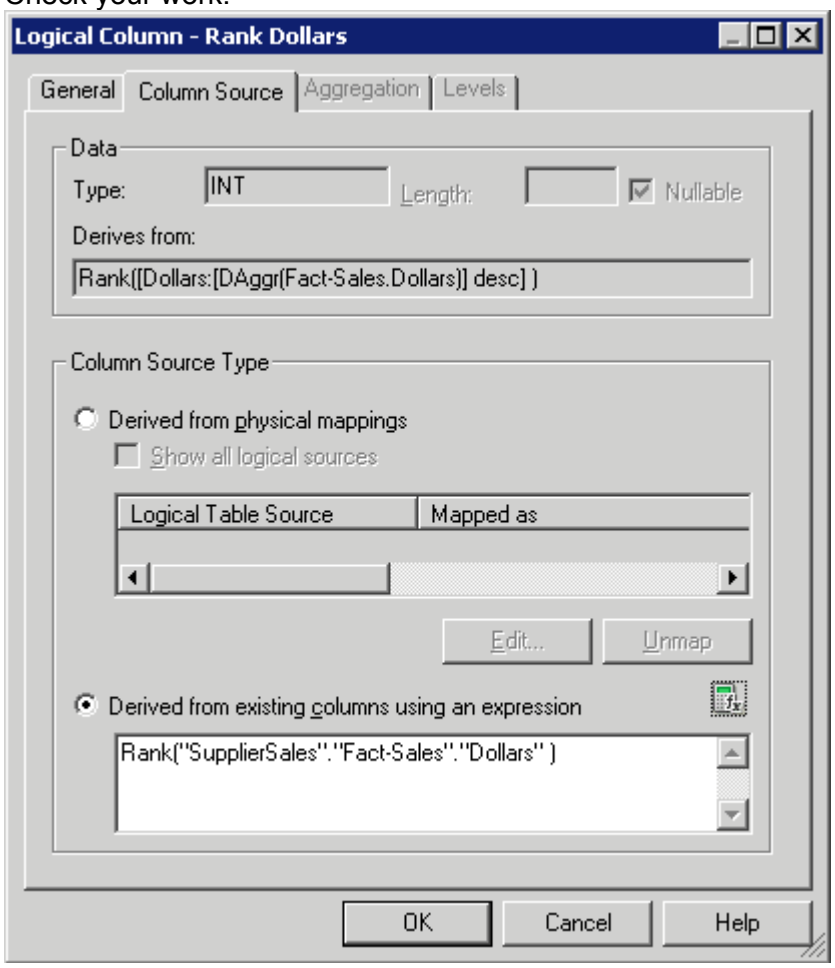
Tasks

1. Create a new rank measure referencing existing logical columns.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository file in offline mode with repository password **welcome1**.
   b. In the Business Model and Mapping layer, right-click the **Fact-Sales** table and select **New Object > Logical Column**.
   c. On the General tab, name the column **Rank Dollars**.
   d. On the Column Source tab select **Derived from existing columns using an expression**.
   e. Open the Expression Builder.
   f. Select **Functions** > **Display Functions** > **Rank**.
   g. Double-click **Rank** or click **Insert**.
   h. Click **<<numExpr>>**.
   i. Select **Logical Tables** > **Fact-Sales** > **Dollars** and add it to the formula.



   j. Click **OK** to close the Expression Builder.

k. Check your work:



l. Click **OK** to close the Logical Column dialog box. Rank Dollars is added to the business model.

m. Add the **Rank Dollars** measure to the **Fact-Sales** presentation table in the Presentation layer.

n. Save the repository and check consistency. Fix any errors or warnings before proceeding.

o. Close the repository.

p. Leave the Admin Tool open.

2. Use Fusion Middleware Control Enterprise Manager to upload the repository.

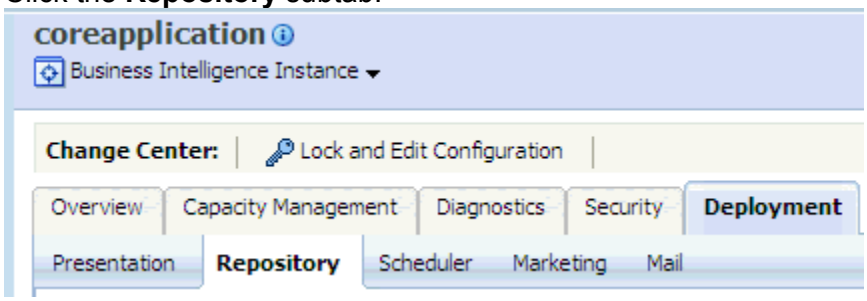
a. Return to Fusion Middleware Control Enterprise Manager, which should still be open in your browser. If it is not open, enter the following URL: **http://localhost:7001/em**.

b. If your session has timed out, log in as **weblogic/welcome1**.

c. In the left pane, expand **Business Intelligence** and select **coreapplication**.

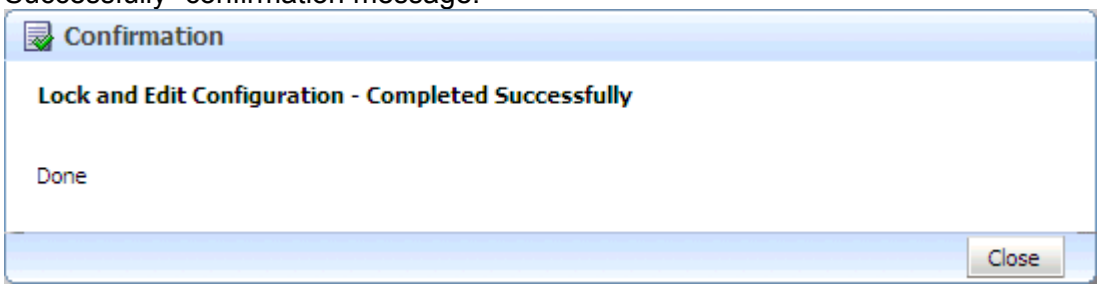
d. In the right pane, click the **Deployment** tab.

e. Click the **Repository** subtab.



f. Click **Lock and Edit Configuration**.

g. Click **Close** when you receive the "Lock and Edit configuration – Completed Successfully" confirmation message.



h. In the Upload BI Server Repository section, click **Browse** to open the "Choose file" dialog box.

i. By default, the "Choose file" dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.

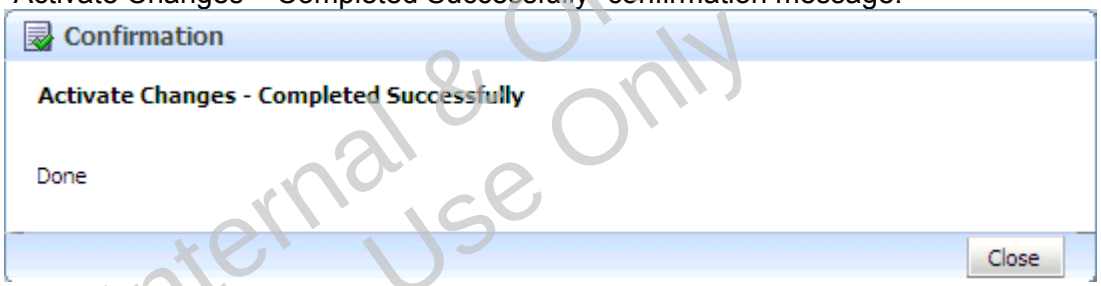
j. Select **ABC.rpd** and click **Open**. You can also double-click the repository to open it.

k. Enter **welcome1** in the Repository Password and Confirm Password fields.

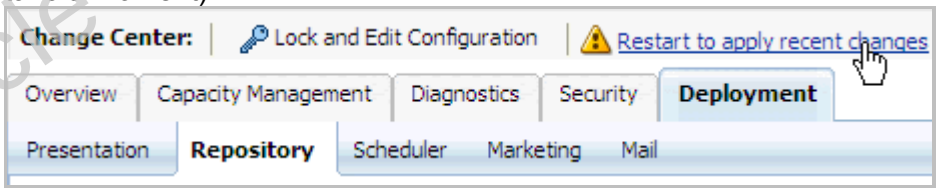
l. Click **Apply**. Notice that Default RPD now displays **ABC** with an extension (for example, ABC_BI0007).

m. Click **Activate Changes**.

n. Allow Active Changes processing to complete. Click **Close** when you receive the "Activate Changes – Completed Successfully" confirmation message.

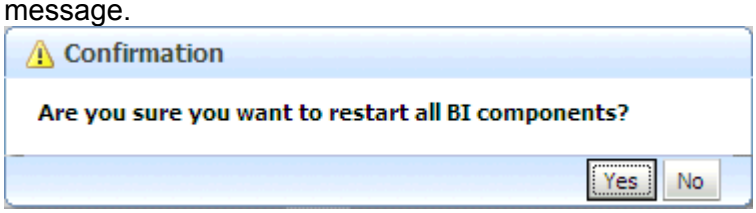


o. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).

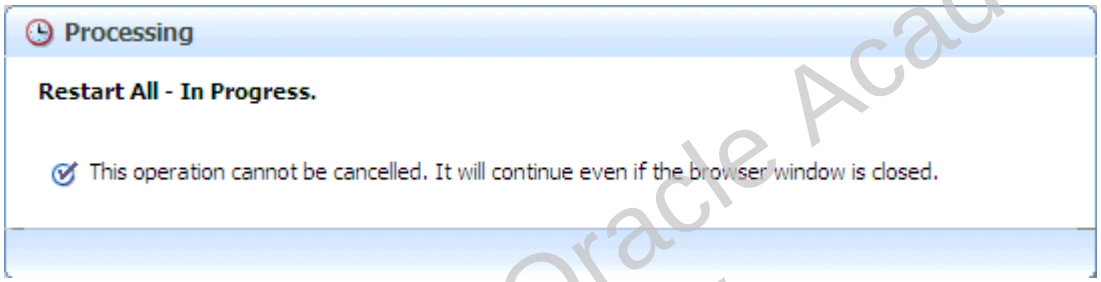


p. On the Overview page, click **Restart**.

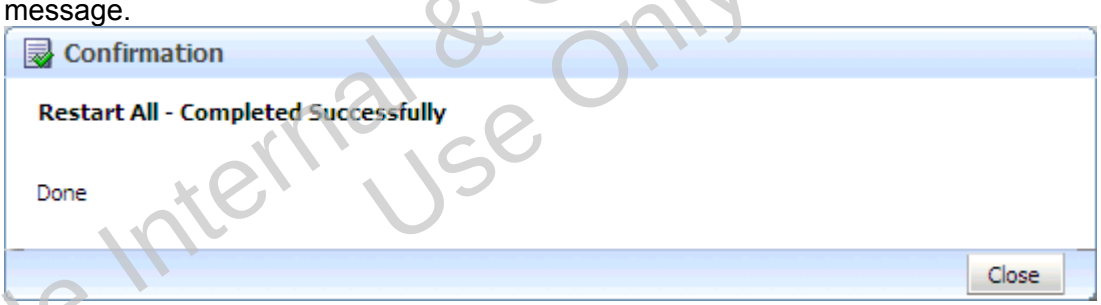
q. Click **Yes** when you receive the "Are you sure you want to restart all BI components?" message.



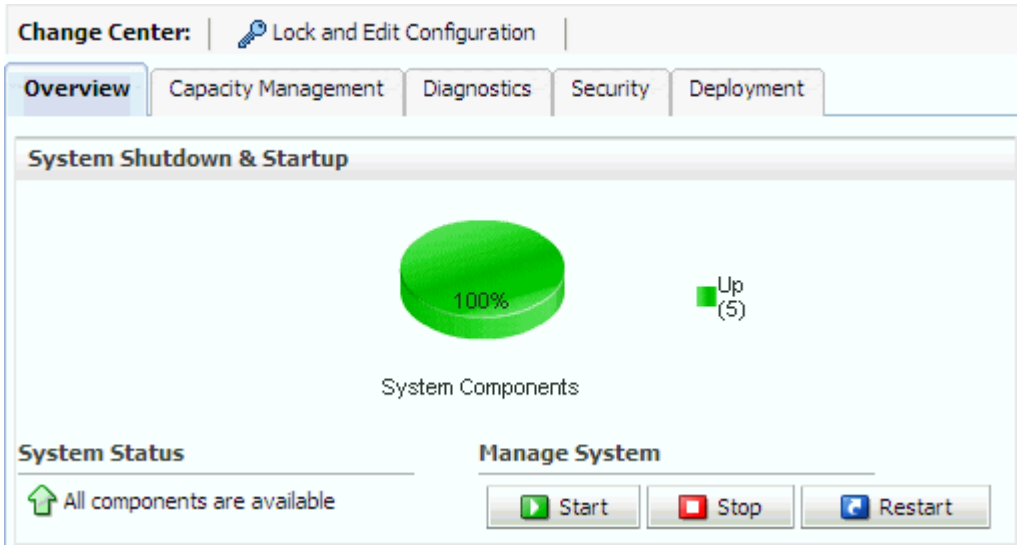
r. Allow the **Restart All – In Progress** processing to complete. This may take a few moments.



s. Click **Close** when you receive the "Restart All – Completed Successfully" confirmation message.



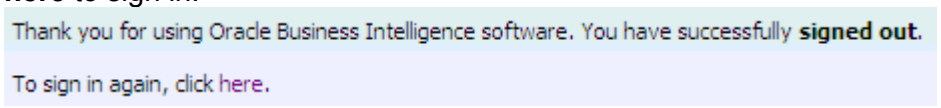
t. Confirm that System Components = 100%. The ABC repository is loaded into BI Server.



u. Leave Enterprise Manager open.

3. Open Analysis Editor to execute queries and test the SupplierSales business model.

a. Return to the Oracle Business Intelligence browser tab where you signed out and click **here** to sign in.



b. Sign in as **weblogic** with password **welcome1**.

c. In the Create section, click **Analysis** to open the Select Subject Area window.

d. Click **SupplierSales** to open Analysis Editor.

4. Create an analysis to check your work.

a. Create the following analysis with the Rank Dollars column sorted in ascending order:

b. Click **Results**:

Generic	Dollars	Rank Dollars
American Cheese Slices	$6,545,551	1
Frozen Chicken	$4,913,525	2
Frozen Peas	$3,183,209	3
Canned Tuna	$1,748,759	4
Frozen Lasagna	$1,625,759	5
White Shortening	$1,598,468	6
Fizzy Cola	$1,499,241	7
Frozen Turkey	$1,479,516	8
Hamburger Patties	$1,424,403	9
Frank's Diet Italian	$1,266,840	10

The results show total dollars for each product and how each product ranks in total sales.

c. Sign out of Oracle BI.

Practices for Lesson 9: Working with Logical Dimensions

Lesson 9

Practices for Lesson 9

Lesson Overview

In these practices, you will build logical dimensions with level-based hierarchies and parent-child hierarchies.

Practice 9-1: Creating Logical Dimension Hierarchies

Goal

To add logical dimension hierarchies to the business model

Scenario

A logical dimension represents a hierarchical organization of logical columns belonging to a single logical dimension table. Logical dimensions can exist in the Business Model and Mapping layer and in the Presentation Layer. Adding logical dimensions to the Presentation layer exposes them to users, which enables users to create hierarchy-based queries. You implement three logical dimensions for ABC: Product, Customer, and Time. Creating logical dimensions with hierarchies allows you to build level-based measures, define aggregation rules that vary by dimension, provide drilldown on charts and tables in analyses and dashboards, and define the content of aggregate sources.

Outcome

In the Business Model and Mapping layer, there are Product, Customer, and Time logical dimensions.

Time

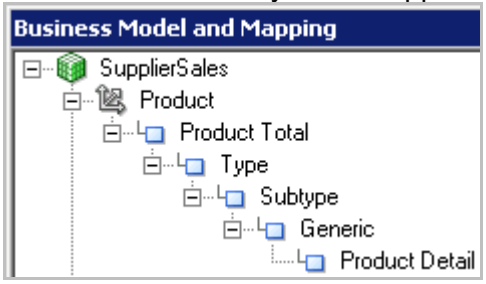
35 minutes

Tasks

1. Create a logical dimension for products.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with the repository password **welcome1**.
   b. In the Business Model and Mapping layer, right-click **SupplierSales** and select **New Object > Logical Dimension > Dimension with Level-Based Hierarchy**. The Logical Dimension dialog box opens.
   c. Name the logical dimension **Product**.
   d. Click **OK**. The new logical dimension appears in the Business Model and Mapping layer. Notice the three-arrow icon.

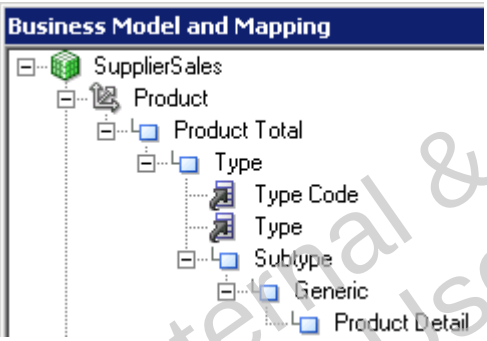
2. Add the parent level of the hierarchy.
   a. Right-click the **Product** logical dimension and select **New Object > Logical Level**.
   b. In the Name field, enter **Product Total**.
   c. Because this level represents the grand total for products, select the **Grand total level** check box. Notice that when you do this, the **Supports rollup to higher level of aggregation** field is grayed out and protected.
   d. Click **OK**. The new level appears as a child of the Product logical dimension.
3. Add the child levels of the hierarchy.

a. Right-click the **Product Total** level and select **New Object > Child Level**.
b. In the Name field, enter **Type**.
c. Click **OK**. The Type level appears as a child of the Product Total level.
d. Repeat the above steps to add further child levels: Subtype, Generic, and Product Detail. Your hierarchy should appear as follows:



4. In this step, you associate columns from the logical dimension table with levels in the logical dimension hierarchy, starting from top to bottom. Here are some guidelines for associating columns with levels:
   - Not all columns in the dimension table must be associated explicitly with a level; the ones that are not will be associated with the lowest level implicitly.
   - No columns can be associated with more than one level (although it may be part of the level key of a lower level).
   - If a column pertains to more than one level, associate it with the highest level it belongs to.
   - No level except the Grand Total level can exist without at least one column being associated with it.
   - The Detail level (lowest level) must have the column that is the logical key of the dimension table associated with it and it must be the key for that level.

   a. In the Business Model and Mapping layer, expand the **Dim-Product** table, select the **Type** column, and drag it up to the Type level.
   b. Drag the logical column **Type Code** onto the Type level in the Product logical dimension.

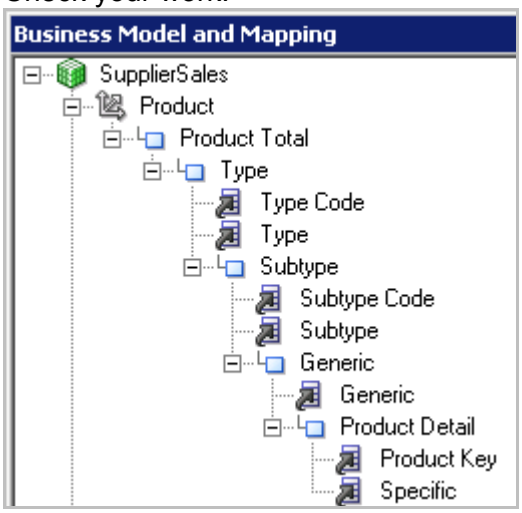


   c. Continue dragging logical columns from the Dim-Product logical table to the **Product** logical dimension levels:

Logical Column	Level
`Subtype`	**Subtype**
`Subtype Code`	**Subtype**
`Generic`	**Generic**

Specific	**Product Detail**
Product Key	**Product Detail**

d. Check your work.



Any column not associated explicitly with a level is associated implicitly with the detail level. In this example, Package Weight and Supplier (among others) are by default associated with the Product Detail level. The logical dimension table key column, Product Key in this example, must be associated explicitly with the lowest level, Product Detail.

e. Double-click the **Type** column in the Dim-Product logical table. The Logical Column dialog box opens.

f. Click the **Levels** tab. This tab identifies the logical dimensions and logical levels associated with this logical column. These values were set when you dragged the column into the Product logical dimension hierarchy.

g. Click the **drop-down list** for the **Type** logical level. Notice that this is another method for associating logical columns with dimensions and logical levels. You use this method later in the next practice. Leave the logical level set to **Type**.

h. Click **Cancel** to close the Logical Column dialog box.

5. In this step, you specify the level keys for the Type level in the hierarchy. Each logical level (except the top level defined as the Grand Total level) must have one or more attributes that compose a key level. The key level defines the unique elements in each logical level. The logical key for a logical table has to be associated with the lowest level of a logical dimension hierarchy and has to be the level key for that level.

a. In the Product logical dimension, double-click the **Type** level. The Logical Level properties dialog box opens.

b. Click the **Keys** tab.

c. Enter **Type** in the Key Name field.

d. In the Columns field, select **Dim-Product.Type**.

e. Leave the Description field blank.

f. Select **Use for Display**.

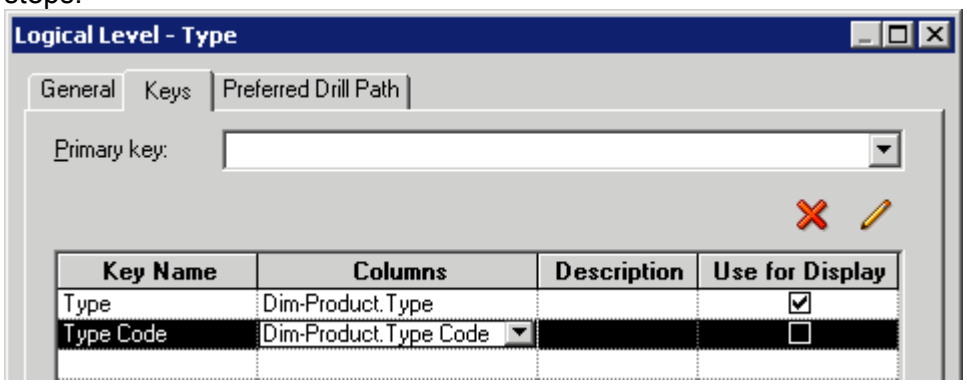
g. Select the next row.

h. Enter **Type Code** in the Key Name field.

i. In the Columns field, select **Dim-Product.Type Code**.

j. Leave the Description field blank.

k. Leave the **Use for Display** deselected. “Use for Display” is explained in the next set of steps.



l. Click **OK**. Notice that both level columns now display a key icon.

6. Set the level keys for the Subtype level using a different method.

   a. In the Product logical dimension, right-click the **Subtype** column (not the level) and select **New Logical Level Key**.

   b. Ensure that the **Subtype** check box is selected.

   c. Ensure that the **Use for Display** check box is selected.

   d. Click **OK**.

   e. Right-click the **Subtype Code** column and select **New Logical Level Key**.

   f. Ensure that the **Subtype Code** check box is selected.

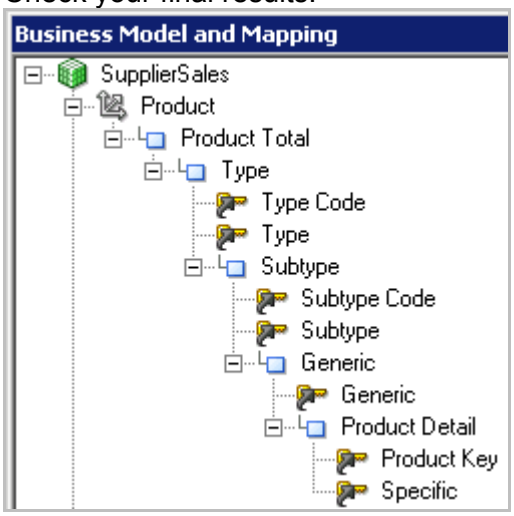
   g. Deselect the **Use for Display** check box.

   h. Click **OK**. Subtype is selected for display and Subtype Code is not. Later, when a user drills down in an analysis or a dashboard, the default drill is to the level key that has “Use for Display” selected in the next lowest level. Based on this example, when a user drills down from the Type level (the next highest level), the default is to drill down to the Subtype column, not the Subtype Code column.

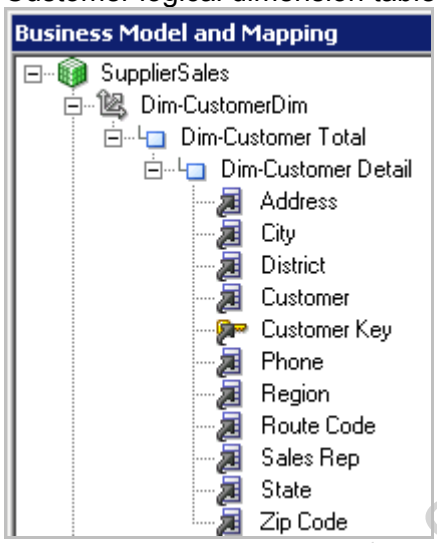
7. Continue setting the following level keys for the remaining levels:

Dimensional Level	Key	Use for Display
Generic	`Generic`	`Yes`
Product Detail	`Specific`	`Yes`
Product Detail	`Product Key`	`No`

8. Check your final results.



9. In this step, you use another method to create the Customer logical dimension and levels.
   a. Right-click the **Dim-Customer** logical table and select **Create Logical Dimension > Dimension with Level-Based Hierarchy**. A new logical dimension named Dim-CustomerDim is created with two levels, Dim-Customer Total and Dim-Customer Detail. The Dim-Customer Detail level is populated with all columns from the Dim-Customer logical dimension table.

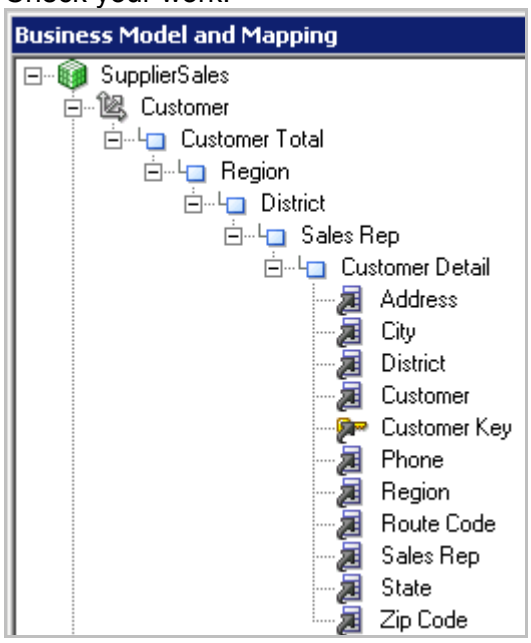


   b. Rename the logical dimension and the levels:

Old Name	New Name
Dim-CustomerDim	`Customer`
Dim-Customer Total	`Customer Total`
Dim-Customer Detail	`Customer Detail`

   c. Right-click the **Customer Detail** level and select **New Object > Parent Level**.
   d. Name the parent level **Sales Rep** and click **OK**.
   e. Create a **District** level as a parent of **Sales Rep**.
   f. Create a **Region** level as a parent of **District**.
   g. Right-click the **Customer** logical dimension and select **Expand All**.

h. Check your work.



i. Add columns to the hierarchy by dragging columns from the Customer Detail level (not the Dim-Customer logical table) to the other Customer hierarchy levels. This is a useful method when business models are large. It eliminates the need to scroll to logical tables to locate columns.

Column	Level
`Region`	**Region**
`District`	**District**
`Sales Rep`	**Sales Rep**

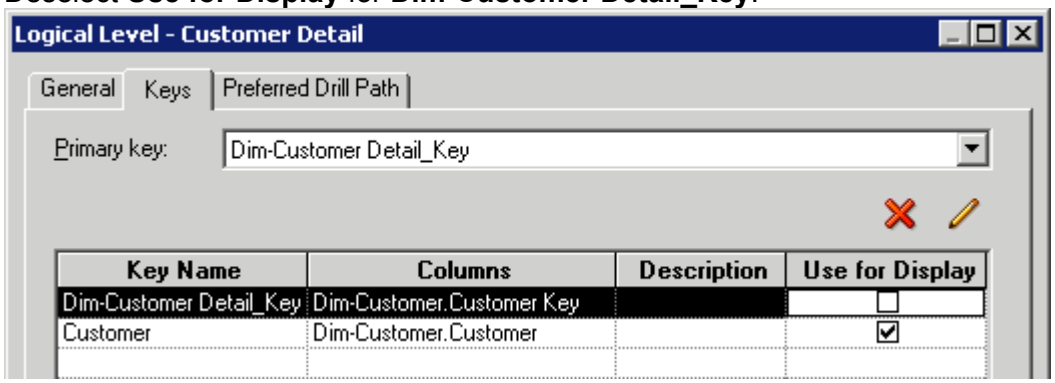
j. Create the following keys for each level. Notice that Customer Key is already identified as a key for the Customer Detail level.

Level	Key	Use for Display
Region	`Region`	`Yes`
District	`District`	`Yes`
Sales Rep	`Sales Rep`	`Yes`
Customer Detail	`Customer`	`Yes`

k. Double-click the **Customer Detail** level.

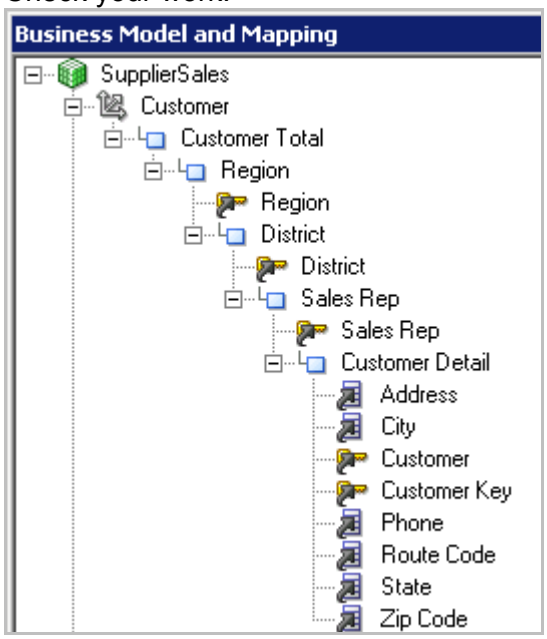
l. Click the **Keys** tab.

m. Deselect **Use for Display** for **Dim-Customer Detail_Key**.



n. Click **OK**.

o. Check your work:



10. Create a preferred drill path for the Customer Detail level. You can use a preferred drill path to identify the drill path to use when users drill down through data in analyses. 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.
    a. Double-click the **Customer Detail** level to open the Logical Level dialog box.
    b. Click the **Preferred Drill Path** tab.
    c. Click **Add**.
    d. In the Browse dialog box, double-click **Type** in the right pane to add it as the preferred drill path in the Logical Level dialog box.
    e. Click **OK**.
    f. Save the repository without checking consistency.
11. In this step, you can use either of the two methods described in the previous steps to create the Time logical dimension level-based hierarchy.
    a. Create a logical dimension named **Time** based on the Time logical dimension table.
    b. Create the following levels:

Level	Grand Total Level	Supports rollup to higher level of aggregation
Time Total	**Yes**	**Protected**
Year	**No**	**Yes**
Quarter	**No**	**Yes**
Month	**No**	**Yes**
Time Detail	**No**	**Yes**

12. There is no logical column that you can associate with the Quarter level, so you must create one and map it to the MONTH_IN_YEAR column with a formula using a CASE statement.
    a. Expand **Dim-Time > Sources** and double-click the **Dim_D1_CALENDAR2** logical table source to open the Logical Table Source dialog box.
    b. Click the **Column Mapping** tab.
    c. Click the **Add new column** button.
    d. On the General tab, name the column **Quarter**.
    e. Click **OK**.
    f. On the Column Mapping tab, ensure that **Show unmapped columns** is selected.
    g. Ensure that the **Quarter** column is selected.
    h. Click the **Edit Expression** button to open the Expression Builder for the **Quarter** column.
    i. Use fully qualified column names and build the following formula. Or, you can copy and paste this formula from the **Quarter.txt** file in D:\PracticeFiles.

```
CASE
WHEN "ORCL"."".."SUPPLIER2"."Dim_D1_CALENDAR2"."MONTH_IN_YEAR" < 4 THEN
1
WHEN "ORCL"."".."SUPPLIER2"."Dim_D1_CALENDAR2"."MONTH_IN_YEAR" < 7 THEN
2
WHEN "ORCL"."".."SUPPLIER2"."Dim_D1_CALENDAR2"."MONTH_IN_YEAR" < 10
THEN 3
ELSE 4
END
```

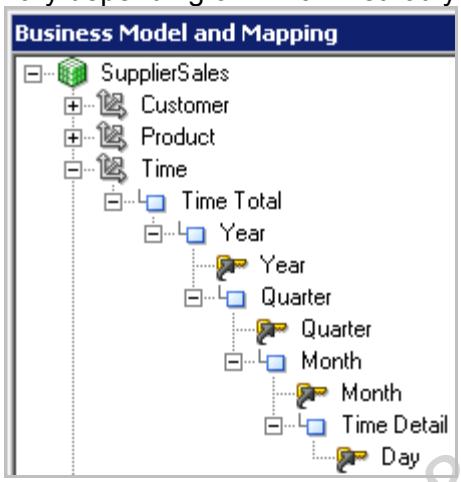
    j. Click **OK** to close the Expression Builder. The expression is displayed in the Expression field in the Logical Table Source dialog box.
    k. Click **OK** to close the Logical Table Source dialog box. The Quarter column is displayed in the business model.
13. Add logical columns to the Time logical dimension levels:

Logical Column	Level
Year	**Year**
Quarter	**Quarter**
Month	**Month**
Day	**Time Detail**

14. Create the keys for each child level in the Time logical dimension. Notice that additional configuration of the key for the Quarter level is completed in a subsequent step.

Level	Key	Use for Display
Year	**Year**	**Yes**
Quarter	**Quarter**	**Yes**
Month	**Month**	**Yes**
Time Detail	**Day**	**Yes**

15. Setting the key for Quarter involves an additional step. Quarters (1, 2, 3, 4) occur each year. However, these numbers do not uniquely identify a quarter, except when combined with Year. Therefore, you must make Year part of the Quarter level.
    a. Double-click the **Quarter** level in the Time logical dimension.
    b. Click the **Keys** tab.
    c. Select the **Quarter** key.
    d. Click the **Edit** button.
    e. Click the **Add** Button.
    f. In the Browse dialog box, expand the **Dim-Time** logical table.
    g. Select **Year** and click **OK**.
    h. Click **OK** to close the **Logical Level Key** dialog box.
    i. Click **OK** to close the **Logical Level** dialog box.
    j. Check your work. It should look similar to the following screenshot. Your results may vary depending on which method you used to build the hierarchy.



16. Set logical levels for the logical table sources.
    a. Expand **Dim-Product > Sources** and double-click the **Dim_D1_PRODUCTS** logical table source to open the Logical Table Source dialog box.
    b. Click the **Content** tab.
    c. Recall that in the practices for the "Managing Logical Table Sources" lesson you set the Aggregation content to Column. Now that you have created logical dimension hierarchies, change **Aggregation content, group by** to **Logical Level** and set the

Logical Level to **Product Detail**.

d. Click **OK** to close the Logical Table Source dialog box.

e. Repeat for the **Type** logical table source.

f. Expand **Dim-Time > Sources** and double-click the **Dim_D1_CALENDAR** logical table source to open the Logical Table Source dialog box.

g. Click the **Content** tab.

h. Depending on which method you used to create the Time logical dimension hierarchy, the logical level may or may not be set. Set it to **Time Detail** and click **OK**.

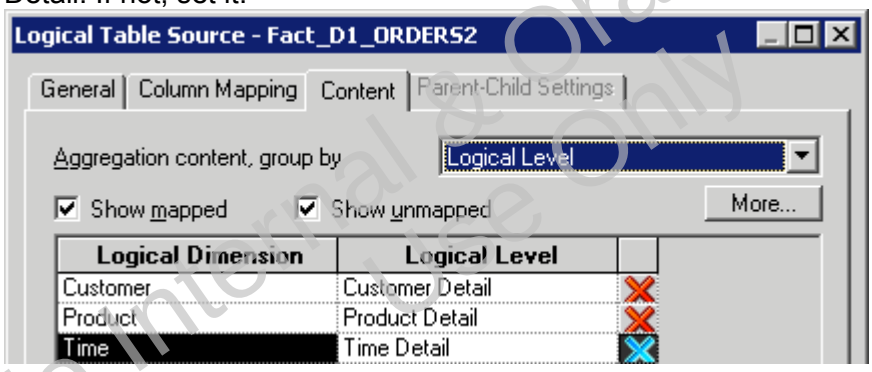
i. Expand **Fact-Sales > Sources** and double-click the **Fact_D1_ORDERS2** logical table source.

j. Click the **Content** tab.

k. Set the logical level to **Product Detail** for the **Product** logical dimension.

l. Set the logical level to **Time Detail** for the **Time** logical dimension.

m. The logical level for the Customer logical dimension should already be set to Customer Detail. If not, set it.

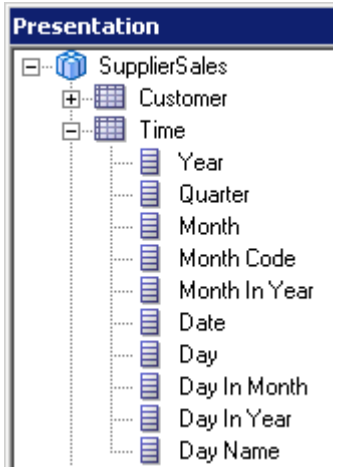


n. Click **OK** to close the Logical Table Source dialog box.

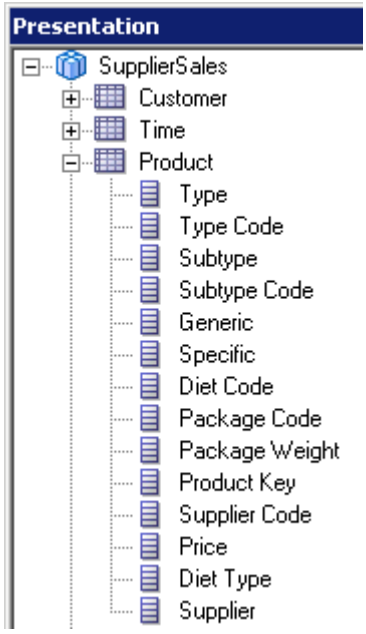
17. Modify the Presentation layer.

a. Drag the **Quarter** logical column from the **Dim-Time** logical table to the **Time** presentation table in the SupplierSales subject area to make it available for analyses.

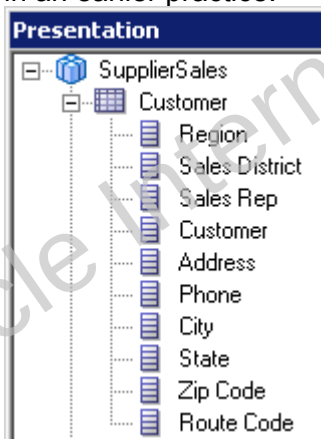
b. Reorder the columns in the Time presentation table to match the logical dimension levels.



c. Reorder the columns in the Product presentation table to match the logical dimension levels.

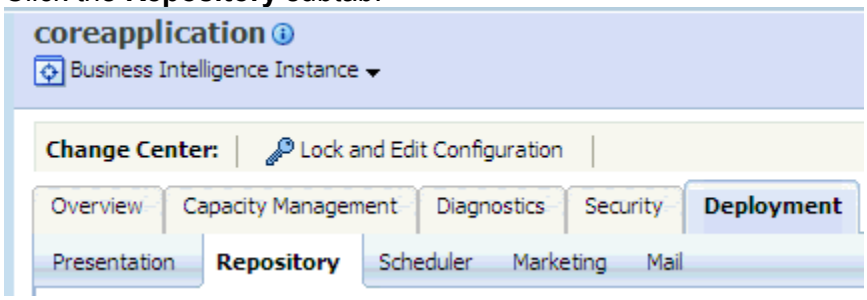


d. Verify the column order in the Customer presentation table. You modified the columns in an earlier practice.

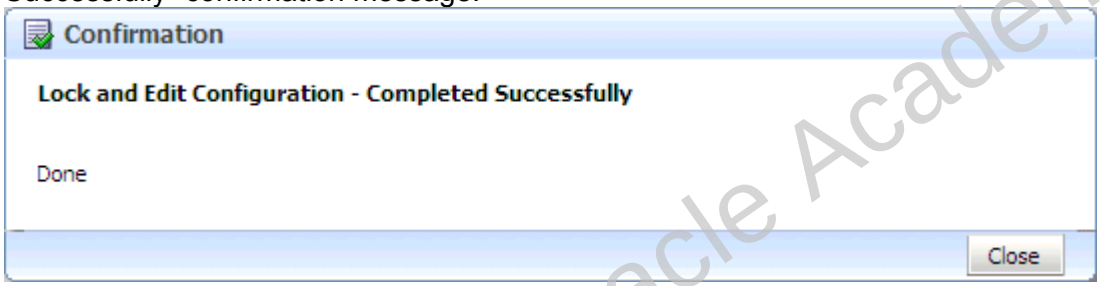


e. Save the repository.
f. Check consistency. Fix any consistency errors or warnings before proceeding.
g. Close the repository.
h. Leave the Administration Tool open for the next practice.

18. Use Fusion Middleware Control Enterprise Manager to upload the repository.
    a. Return to Fusion Middleware Control Enterprise Manager, which should still be open in your browser. If it is not open, enter the following URL: **http://localhost:7001/em**.
    b. If your session has timed out, log in as **weblogic/welcome1**.
    c. In the left pane, expand **Business Intelligence** and select **coreapplication**.
    d. In the right pane, click the **Deployment** tab.
    e. Click the **Repository** subtab.

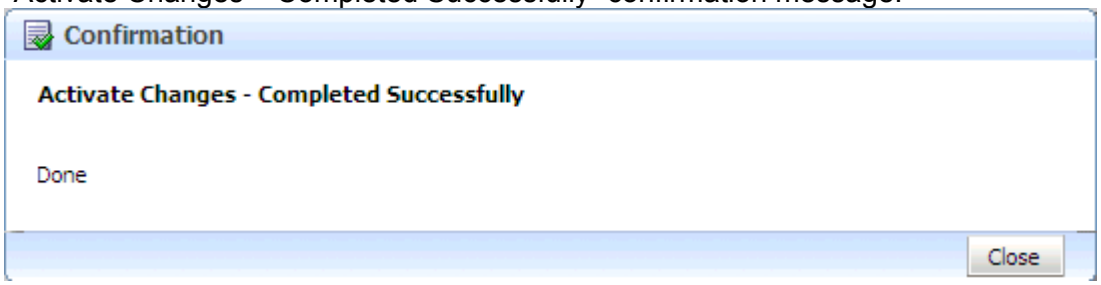


    f. Click **Lock and Edit Configuration**.
    g. Click **Close** when you receive the "Lock and Edit configuration – Completed Successfully" confirmation message.

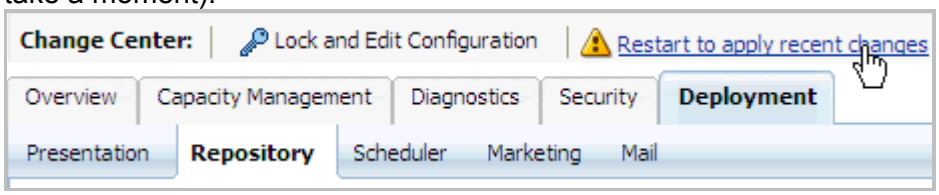


    h. In the Upload BI Server Repository section, click **Browse** to open the "Choose file" dialog box.
    i. By default, the "Choose file" dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.
    j. Select **ABC.rpd** and click **Open**. You can also double-click the repository to open it.
    k. Enter **welcome1** in the Repository Password and Confirm Password fields.
    l. Click **Apply**. Notice that Default RPD now displays **ABC** with an extension, for example, ABC_BI0007.
    m. Click **Activate Changes**.

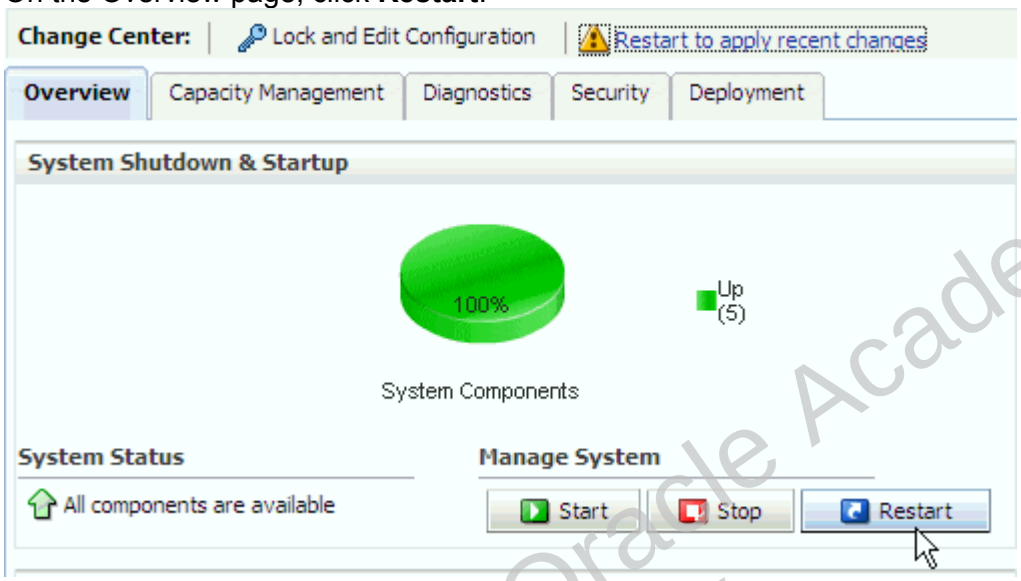
n. Allow Active Changes processing to complete. Click **Close** when you receive the "Activate Changes – Completed Successfully" confirmation message.



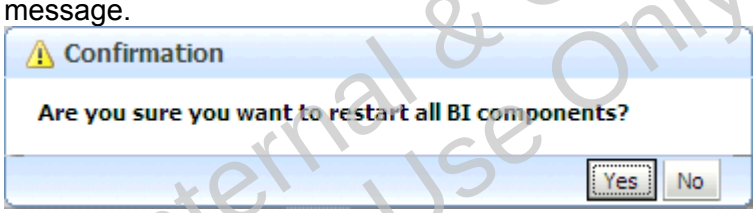
o. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).



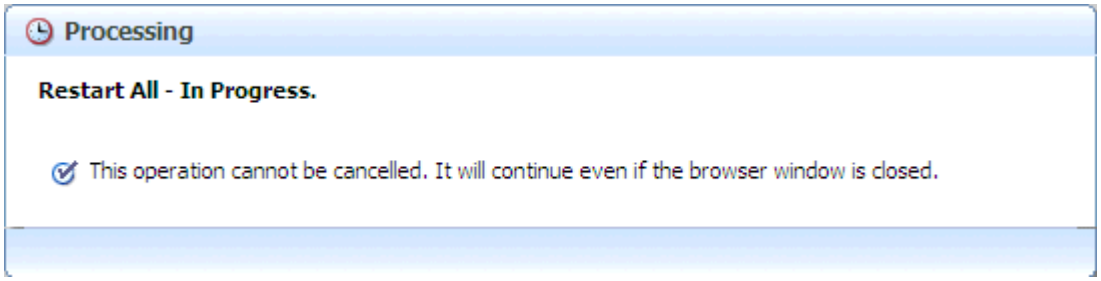
p. On the Overview page, click **Restart**.



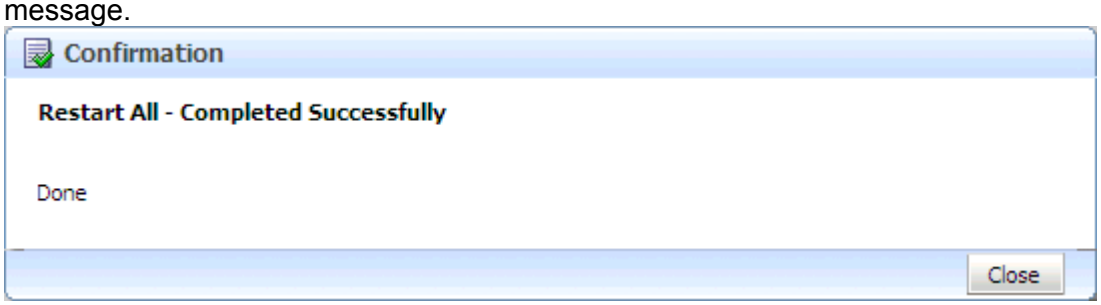
q. Click **Yes** when you receive the "Are you sure you want to restart all BI components?" message.



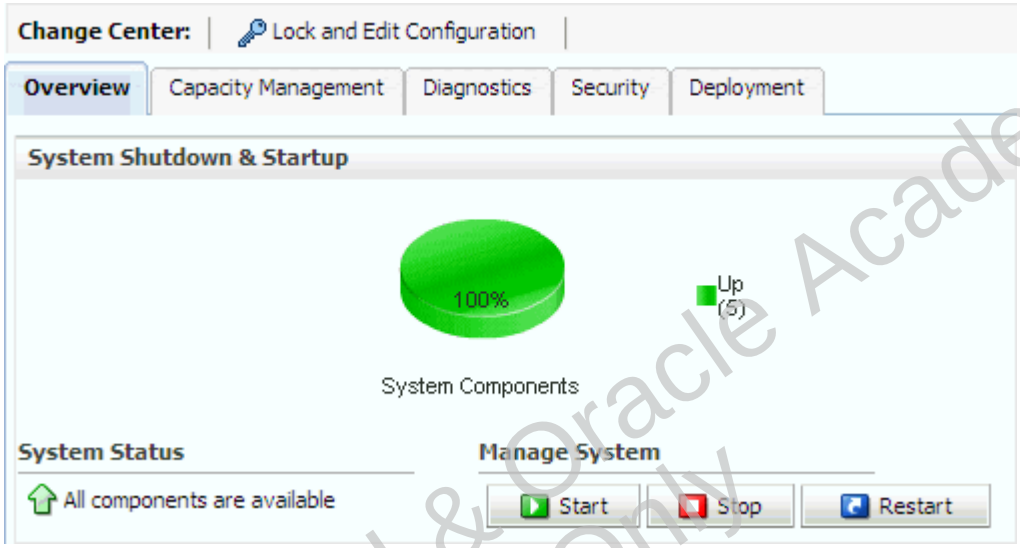
r. Allow the Restart All – In Progress processing to complete. This may take a few moments.



s. Click **Close** when you receive the "Restart All – Completed Successfully" confirmation message.

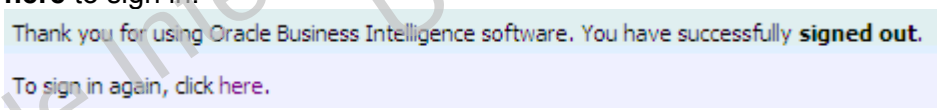


t. Confirm that System Components = 100%. The ABC repository is loaded into BI Server.



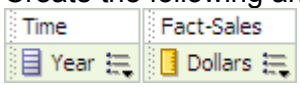
u. Leave Enterprise Manager open.

19. Open Analysis Editor to execute queries and test the SupplierSales business model.
    a. Return to the Oracle Business Intelligence browser tab where you signed out and click **here** to sign in.



    b. Sign in as **weblogic** with password **welcome1**.
    c. In the Create section, click **Analysis** to open the Select Subject Area window.
    d. Click **SupplierSales** to open Analysis Editor.
20. Create the following analysis to test the Time logical dimension:

a. Create the following analysis:



b. Click **Results**.

c. Drill down on **2008** and ensure that you can see dollars data by quarter.

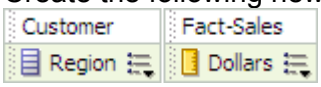
Year	Quarter	Dollars
2008	1	$11,829,308
	2	$12,207,161
	3	$12,390,129
	4	$12,601,988

d. Drill down on any quarter and ensure that you can see dollars data by month.

e. Drill down on any month and ensure that you can see dollars data by day. Your results should look similar to the following screenshot:

Year	Quarter	Month	Day	Dollars
2008	1	February	20080202	$186,903
			20080203	$231,249
			20080204	$268,629
			20080205	$95,720
			20080206	$297,776
			20080207	$120,795
			20080209	$128,238
			20080210	$189,540
			20080211	$290,734
			20080212	$254,689

21. Create the following new analysis to test the Customer logical dimension:

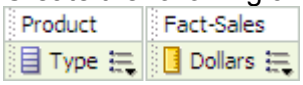


a. Click **Results**.

b. Verify that you can drill down from the Region level to Sales District > Sales Rep > Customer > Type. Recall that you set Type as the preferred drill path for the Customer Detail level. Your results should look similar to the following screenshot:

Region	Sales District	Sales Rep	Customer	Type	Dollars
Central	Gulf	MARY SILVER	Alley Dog	Baking	$33,636
				Beef	$18,100
				Beverage	$13,305
				Bread	$16,195

22. Create the following analysis to test the Product logical dimension:



a. Click **Results**.

b. Verify that you can drill down from the Type level to Subtype > Generic > Specific. Your results should look similar to the following screenshot:

Type	Subtype	Generic	Specific	Dollars
Baking	Balsamic Vinegar	Balsamic Vinegar	Balsamic Vinegar	$29,155

C. Sign out of Oracle BI.

Practice 9-2: Creating Level-Based Measures

Goal

To create level-based measures

Scenario

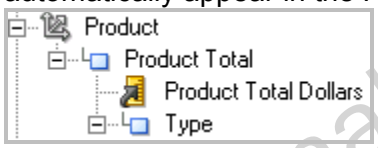
Now that you have created level-based logical dimension hierarchies, you create level-based measures that calculate total dollars at various levels in the hierarchies.

Time

30 minutes

Tasks

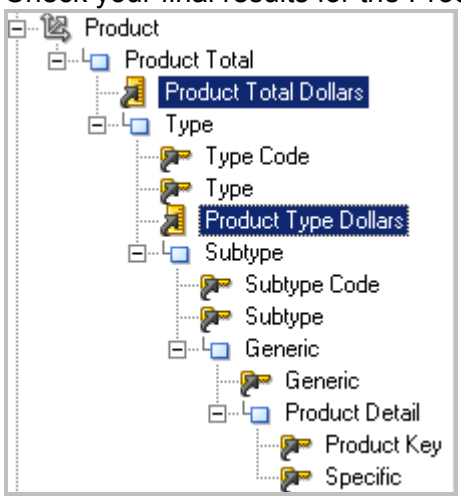
1. Create level-based measures for the Product logical dimension.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. In the Business Model and Mapping layer, right-click the **Fact-Sales** table and select **New Object > Logical Column**.
   c. On the General tab, in the Name field, enter **Product Total Dollars**.
   d. Click the **Column Source** tab.
   e. Select **Derived from existing columns using an expression**.
   f. Open the Expression Builder.
   g. In the Expression Builder, add **Logical Tables** > **Fact-Sales** > **Dollars** to the expression. Recall that the Dollars column already has a default aggregation rule of **Sum**.
   h. Click **OK**.
   i. Click the **Levels** tab.
   j. For the **Product** logical dimension, select **Product Total** from the Level drop-down list to specify that this measure should be calculated at the grand total level in the product hierarchy.
   k. Click **OK**. Notice that setting the level for the logical column causes the measure to automatically appear in the Product logical dimension.



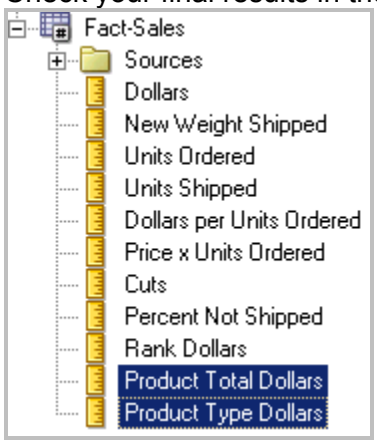
   l. Repeat the steps to create a second level-based measure:

Name	Dimension	Level
`Product Type Dollars`	`Product`	`Type`

m. Check your final results for the Product logical dimension:



n. Check your final results in the Fact-Sales logical table:



o. Expose the new columns to users by dragging **Product Total Dollars** and **Product Type Dollars** from the **Fact-Sales** logical table to the **Fact-Sales** presentation table in the SupplierSales catalog in the Presentation layer.

p. Save the repository.

q. Check consistency. Fix any errors or warnings before proceeding.

r. Leave the repository open. You check your work in Analysis Editor in Practice 8-5.

Practice 9-3: Creating Share Measures

Goal

To create a share measure using level-based measures

Scenario

Now that you have created level-based measures, you use them to create a share measure for products.

Time

40 minutes

Tasks

1. Create a new share measure derived from existing logical columns.
   a. In the Business Model and Mapping layer, right-click **Fact-Sales** and select **New Object > Logical Column**.
   b. On the General tab, name the logical column **Product Share**.
   c. On the Column Source tab, select **Derived from existing columns using an expression**.
   d. Open the Expression Builder.
   e. Select **Functions** > **Mathematic Functions** > **Round**.
   f. Click **Insert selected item**. The function appears in the edit box.
   g. Click **SourceNumber** in the formula.
   h. Enter **100*** followed by a space.
   i. Insert **Logical Tables** > **Fact-Sales > Dollars**.
   j. Using the toolbar, click the **Division** / button. Another set of angle brackets appears, <<expr>>.
   k. Click **<<expr>>**.
   l. Insert **Logical Tables > Fact-Sales > Product Total Dollars**. Recall that this is the total level-based measure for the Product logical dimension hierarchy.
   m. Click between the last set of angle brackets, **<<Digits>>,** and enter **1**. This represents the number of digits of precision with which to round the integer.
   n. Check your work:
   ```
   Round(100* "SupplierSales"."Fact-Sales"."Dollars" /
   "SupplierSales"."Fact-Sales"."Product Total Dollars"  , 1)
   ```
   o. This share measure will allow you to run an analysis to show how sales of a specific product compares to overall sales for all products.
   p. Click **OK** to close the Expression Builder.
   q. Click **OK** to close the Logical Column properties dialog box.
   r. The **Product Share** logical column is added to the business model.
   s. Add the **Product Share** measure to the Fact-Sales presentation table.
   t. Save the repository.
   u. Check consistency. If there are errors or warnings, correct them before you proceed.
   v. Leave the repository open. You check your work in Analysis Editor in Practice 8-5.

Practice 9-4: Creating Dimension-Specific Aggregation Rules

Goal

To create a measure with dimension-specific aggregation rules

Scenario

ABC wants a measure called Average Daily Dollars, which sums dollar amounts over the Customer and Product dimensions and divides by the number of days in the Time dimension. This measure can be used to compare the average daily dollar amount from month-to-month when the number of order days in each month varies.

Outcome

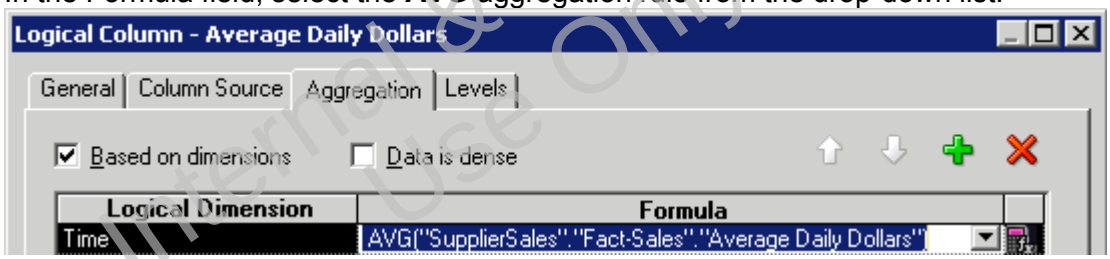
A new measure called Average Daily Dollars with dimension-specific aggregation rules

Time

15 minutes

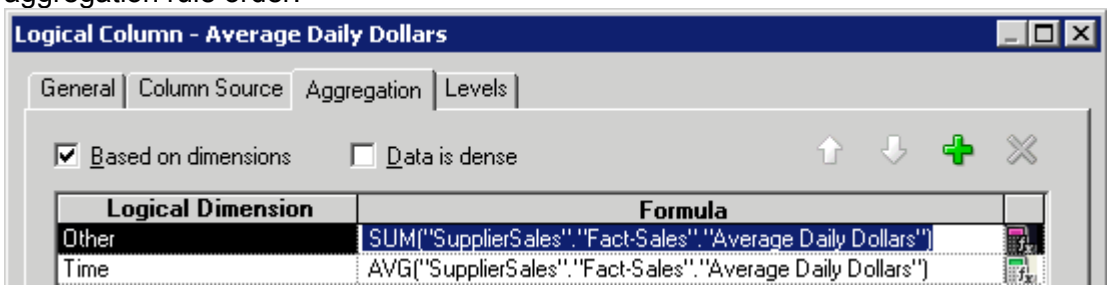
Tasks

1. Create a new measure with dimension-specific aggregation rules.
   a. In the Business Model and Mapping layer, right-click **Fact-Sales** and select **New Object > Logical Column**.
   b. On the General tab, name the logical column **Average Daily Dollars**.
   c. Click the **Column Source** tab.
   d. Select **Show all logical sources**.
   e. Double-click the **Fact_D1_ORDERS2** logical table source.
   f. Click the **Column Mapping** tab.
   g. Ensure that **Show unmapped columns** is selected.
   h. Map **Average Daily Dollars** to the **Fact_D1_ORDERS2_DOLLARS** physical column.
   i. Click **OK** to close the Logical Table Source dialog box.
   j. Click the **Aggregation** tab.
   k. Select **Based on dimensions**.
   l. In the Browse dialog box, select the **Time** dimension and click **OK**.
   m. In the Formula field, select the **AVG** aggregation rule from the drop-down list.



   n. Click the **Add** button.
   o. Select **Other** and click **OK**.
   p. Select the **SUM** aggregation rule for **Other**.

q. With **Other** selected in the Logical Dimension column, use the **Up** button to change the aggregation rule order.



r. Click **OK** to close the Logical Column dialog box.

s. Drag the **Average Daily Dollars** measure to the **Fact-Sales** presentation table.

t. Save the repository.

u. Check Global Consistency. Fix errors and warnings before you proceed.

v. Leave the repository open. You check your work in Analysis Editor in Practice 8-5.

Practice 9-5: Creating Presentation Hierarchies

Goal

To create presentation hierarchies that expose logical dimension hierarchies in Oracle BI analyses.

Scenario

Presentation hierarchies and presentation levels provide an explicit way to expose the multidimensional model in Oracle BI analyses and dashboards. When presentation hierarchies and levels are defined in the Presentation layer, roll-up information is displayed in the Analysis Editor navigation pane, providing users with important contextual information. Most importantly, users can create hierarchy-based queries using these objects.

Outcome

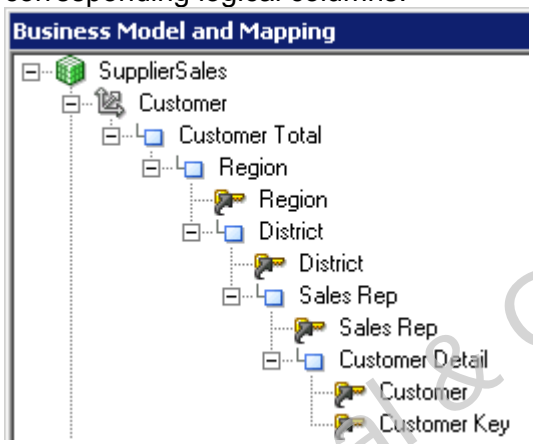
Presentation hierarchies are added to the Customer, Time, and Product presentation tables.

Time

15 minutes

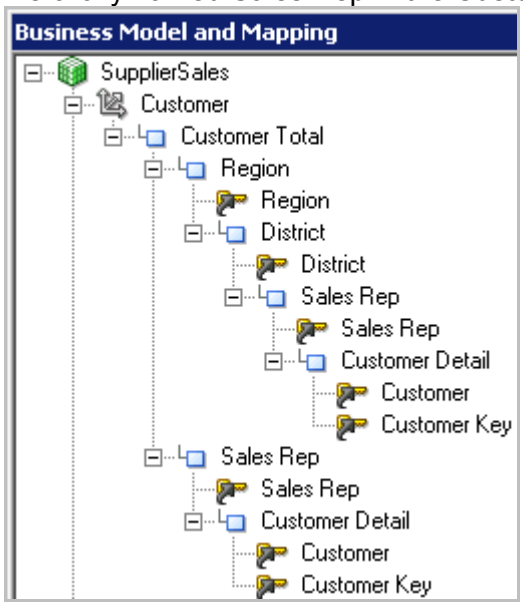
Tasks

1. Modify the customer logical dimension to include multiple hierarchies.
   a. Expand the **Customer** logical dimension.
   b. Delete all columns from the Customer Detail level except for **Customer** and **Customer Key**. Notice that deleting columns from the hierarchy does not delete the corresponding logical columns.

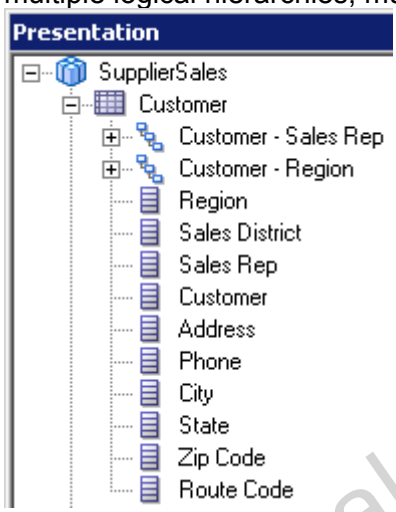


   c. Right-click the **Customer Total** level and select **New Object > Shared Level as Child**.

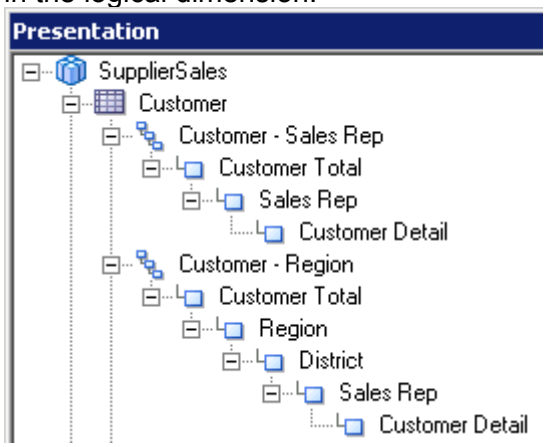
d. In the Browse dialog box, select **Sales Rep** and click **OK**. This creates a second hierarchy named Sales Rep in the Customer logical dimension:



2. Create presentation hierarchies.
   a. Drag the **Customer** logical dimension from the Business Model and Mapping layer to the **Customer** presentation table. Notice that two separate presentation hierarchies are created, one for each logical dimension hierarchy. For logical dimensions that contain multiple logical hierarchies, multiple separate presentation hierarchies are created:



b. Expand the presentation hierarchies and notice that each contains the drill path defined in the logical dimension.



c. Drag the **Product** logical dimension to the **Product** presentation table.

d. Drag the **Time** logical dimension to the **Time** presentation table. Notice that only one presentation hierarchy is created for the Product and Time logical dimensions. For logical dimensions that contain one logical hierarchy, one presentation hierarchy is created.

e. Save the repository.

f. Check consistency. Fix any errors or warnings before proceeding.

g. Close the repository.

3. Use Fusion Middleware Control Enterprise Manager to upload the repository.

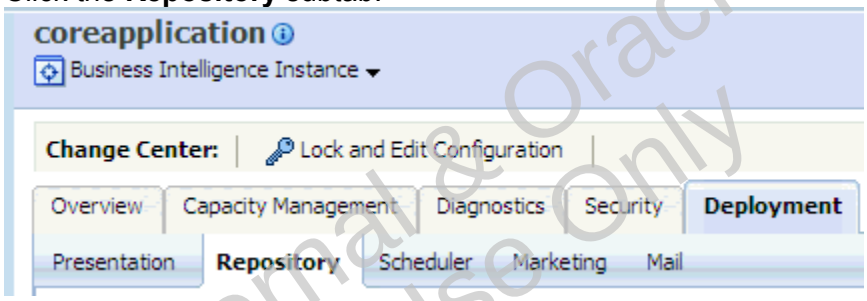
a. Return to Fusion Middleware Control Enterprise Manager, which should still be open in your browser. If it is not open, enter the following URL: **http://localhost:7001/em**.

b. If your session has timed out, log in as **weblogic/welcome1**.

c. In the left pane, expand **Business Intelligence** and select **coreapplication**.

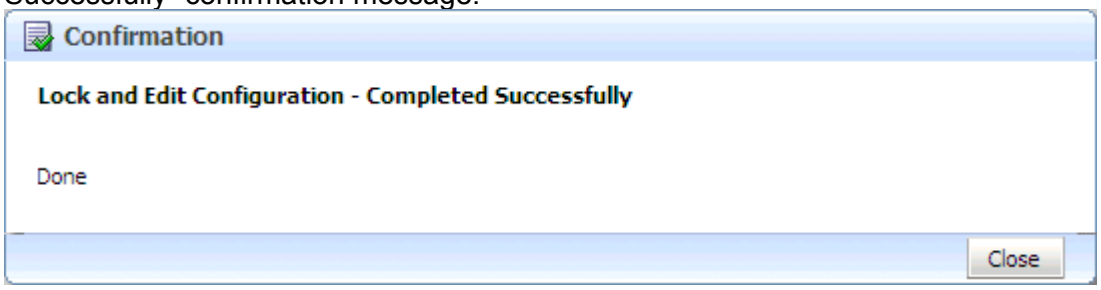
d. In the right pane, click the **Deployment** tab.

e. Click the **Repository** subtab.



f. Click **Lock and Edit Configuration**.

g. Click **Close** when you receive the “Lock and Edit configuration – Completed Successfully” confirmation message.



h. In the Upload BI Server Repository section, click **Browse** to open the “Choose file” dialog box.

i. By default, the “Choose file” dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.

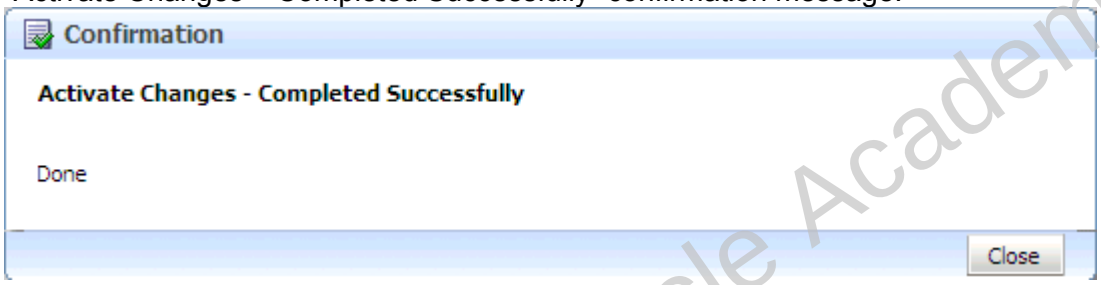
j. Select **ABC.rpd** and click **Open**. You can also double-click the repository to open it.

k. Enter **welcome1** in the Repository Password and Confirm Password fields.

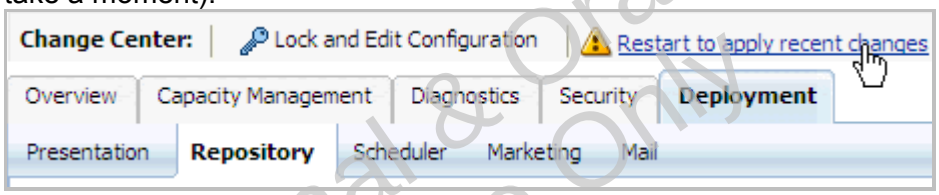
l. Click **Apply**. Notice that Default RPD now displays **ABC** with an extension, for example, ABC_BI0007.

m. Click **Activate Changes**.

n. Allow Active Changes processing to complete. Click **Close** when you receive the “Activate Changes – Completed Successfully” confirmation message.

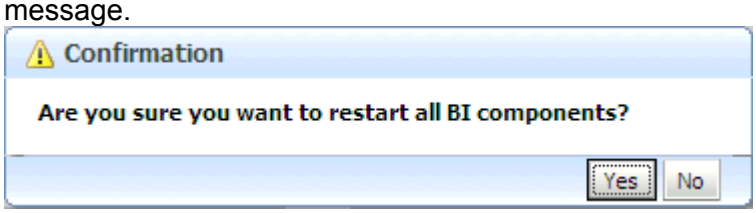


o. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).

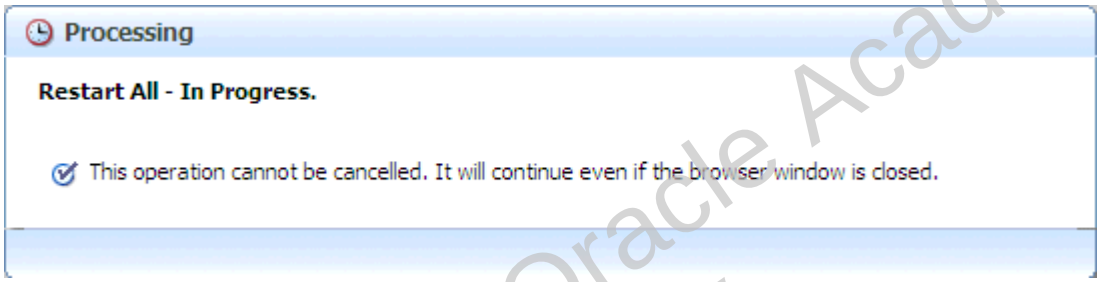


p. On the Overview page, click **Restart**.

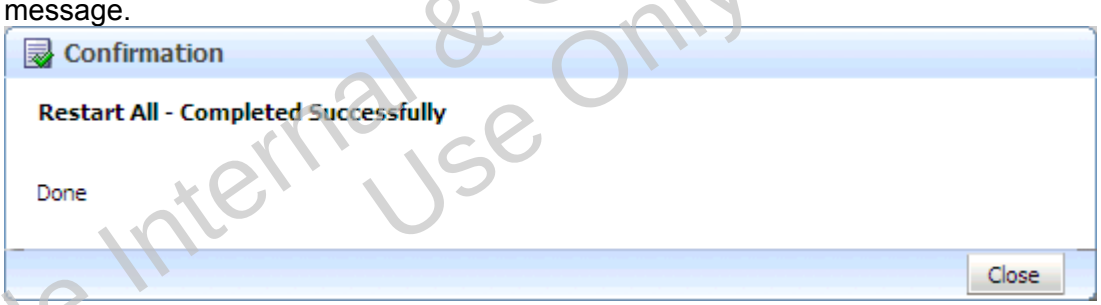
q. Click **Yes** when you receive the "Are you sure you want to restart all BI components?" message.



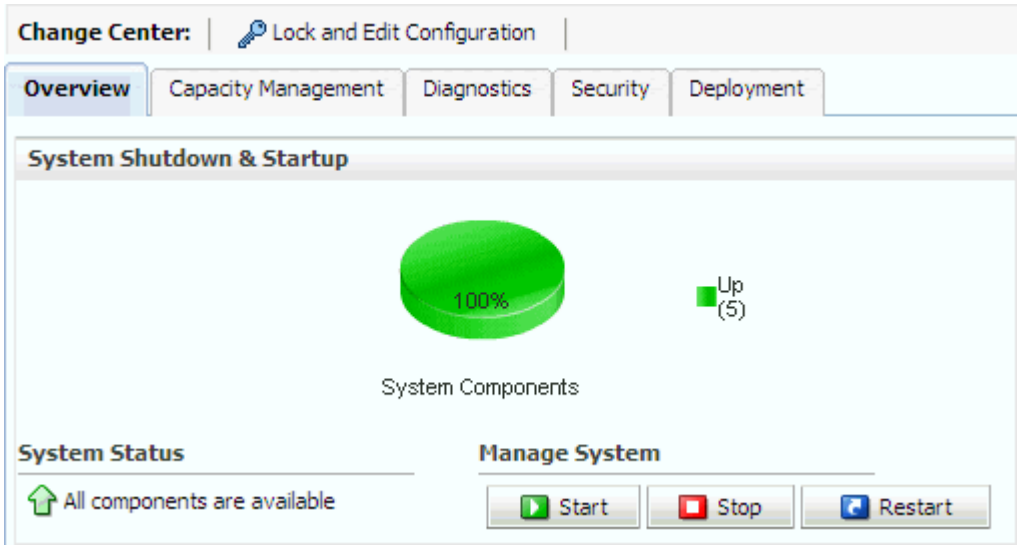
r. Allow the Restart All – In Progress processing to complete. This may take a few moments.



s. Click **Close** when you receive the "Restart All – Completed Successfully" confirmation message.



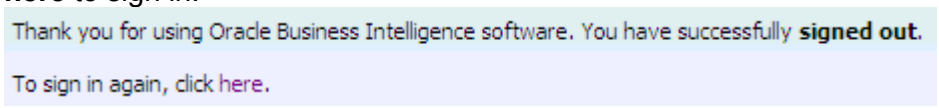
t. Confirm that System Components = 100%. The ABC repository is loaded into BI Server.



u. Leave Enterprise Manager open.

4. Open Analysis Editor to execute queries and test the SupplierSales business model.

a. Return to the Oracle Business Intelligence browser tab where you signed out and click **here** to sign in.

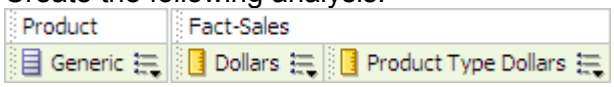


b. Sign in as **weblogic** with password **welcome1**.

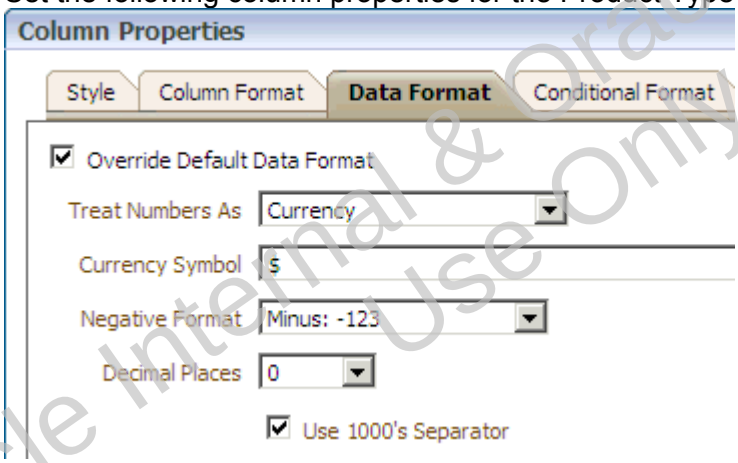
c. In the Create section, click **Analysis** to open the Select Subject Area window.

5. Create an analysis to test the level-based measures.

a. Create the following analysis:



b. Set the following column properties for the Product Type Dollars column:



c. Click **Save as Default > Save as the system-wide default for '"Fact-Sales"."Product Type Dollars"'**.

d. Click **Results**. Notice that **Product Type Dollars** returns dollars grouped by Type even though the query is at a different level than Type; Generic in this example.

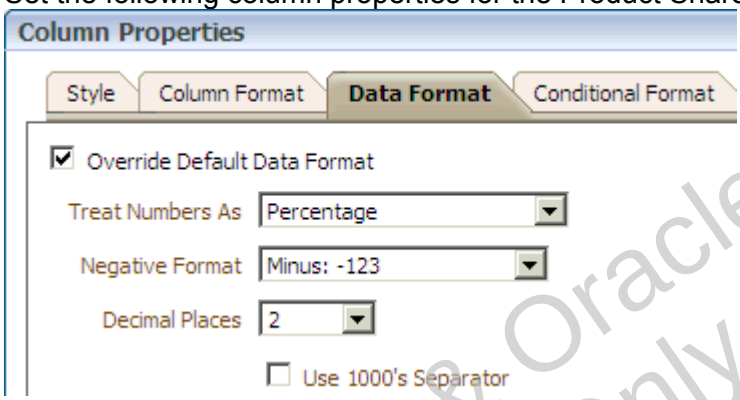
Generic	Dollars	Product Type Dollars
American Cheese Slices	$6,545,551	$7,201,383
Apple Dumplings	$23,924	$2,259,388
Apple Juice	$1,230,175	$4,470,714
Apple Sauce	$995,921	$9,123,306
Baked Beans	$513,499	$5,035,506
Balsamic Vinegar	$29,155	$5,275,980
Bar Glasses	$9,001	$4,809,399
Bar Stools	$2,344	$4,809,399
Barley	$3,607	$40,566
Beef Bouillon	$22,357	$5,275,980

e. Drill down on a value in the **Generic** column. Notice that drilldown is to the **Specific** column by default. Recall that the Specific level in the Product hierarchy had two columns associated with it: Product Key and Specific. Because you set "Use for Display" for the Specific column, and did not set it for the Product Key column, Generic drills to Specific by default. When a user drills down from the next highest level, the default is to drill down to the column with "Use for Display" selected.

Generic	Specific	Dollars	Product Type Dollars
American Cheese Slices	2 Pak American Cheese Slices 16 slices	$6,545,551	$7,201,383

6. Create an analysis to test the share measure.
   a. Create the following analysis:

   b. Set the following column properties for the Product Share column:

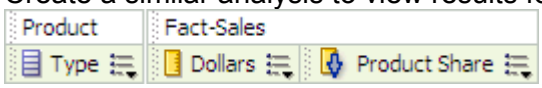


   c. Select **Save as Default > Save as the system-wide default for Fact-Sales.Product Share**.
   d. Sort Product Share in descending order.
   e. Click **Results**:

Generic	Dollars	Product Share
American Cheese Slices	$6,545,551	10.10%
Frozen Chicken	$4,913,525	7.60%
Frozen Peas	$3,183,209	4.90%
Canned Tuna	$1,748,759	2.70%
Frozen Lasagna	$1,625,759	2.50%
White Shortening	$1,598,468	2.50%
Fizzy Cola	$1,499,241	2.30%

The results show total dollars for each product and the percent of total sales for each product sorted in descending order.

f. Create a similar analysis to view results for Product Type:



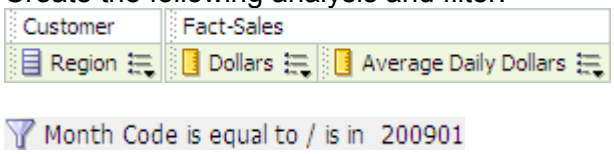
g. Click **Results**.

Type	Dollars	Product Share
Condiments	$9,123,306	14.10%
Poultry	$7,304,207	11.30%
Cheese	$7,201,383	11.10%
Baking	$5,275,980	8.20%
Beef	$5,088,480	7.90%
Vegetable	$5,035,506	7.80%
Non-food	$4,809,399	7.40%
Beverage	$4,470,714	6.90%

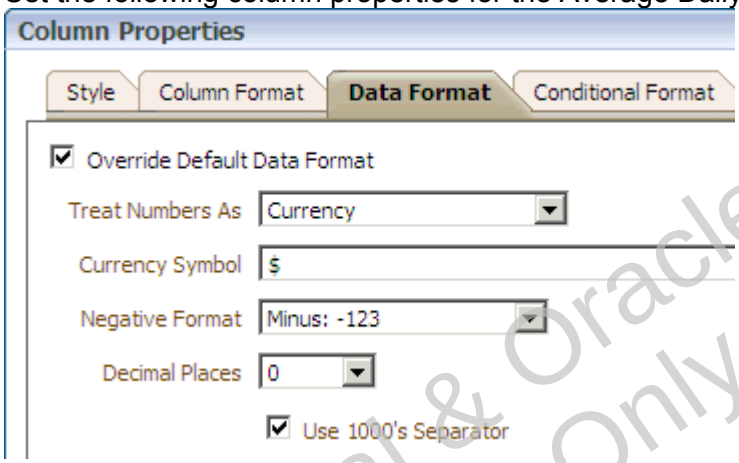
The results show total dollars for each product type and the percent of total sales for each product type sorted in descending order.

7. Create an analysis to test the Average Daily Dollars measure with a dimension-specific aggregation rule.

a. Create the following analysis and filter:



b. Set the following column properties for the Average Daily Dollars column:

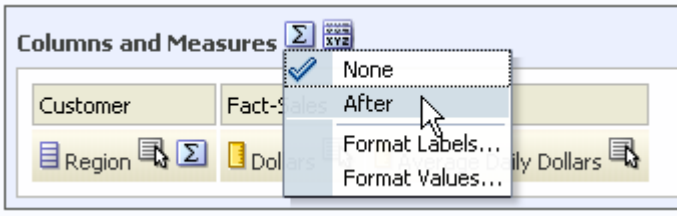


c. Click **Results**.

Region	Dollars	Average Daily Dollars
Central	$954,306	$38,172
East	$1,646,242	$65,850
West	$1,775,737	$71,029

d. Click the **Edit View** button for the Table view.

e. In the Layout area, click the Total button for “Columns and Measures” and select **After**.



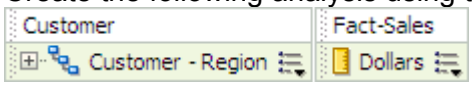
f. Check your results:

Region	Dollars	Average Daily Dollars
Central	$954,306	$38,172
East	$1,646,242	$65,850
West	$1,775,737	$71,029
Grand Total	**$4,376,285**	**$175,051**

g. Considering that there are 25 order days in the first month of 2009, use the Windows calculator to check that the results of the analysis are correct for average daily dollars.

8. Create analyses to test presentation hierarchies.

a. Create the following analysis using the Customer – Region presentation hierarchy:



b. Click **Results**.

c. Expand the hierarchy levels to view data at different levels.

	Dollars
Customer - Region	
⊟ Customer Total	$64,612,461
⊟ Central	$13,423,387
⊟ Gulf	$843,693
⊟ MARY SILVER	$843,693
Alley Dog	$304,899
Papa Pete's Pizza	$538,794
⊞ LowerMidWest	$4,753,536
⊞ MidWest	$2,452,673
⊞ Texas	$4,125,482
⊞ UpperMidWest	$1,248,003
⊞ East	$25,460,351
⊞ West	$25,728,722

d. Build another analysis using the **Customer – Sales Rep** presentation hierarchy and view the results. Notice that the drill path is different than the drill path for the Customer – Region hierarchy.

	Dollars
Customer - Sales Rep	
⊟ Customer Total	$64,612,461
⊟ ALAN ZIFF	$1,527,930
Chang's Mongolian Grill	$331,217
Globus Office	$212,182
Half-Shell Restaurant	$880,096
Times On Bay	$104,436
⊞ ANDREW TAYLOR	$594,773
⊞ ANN JOHNSON	$2,445,038
⊞ ANNE WILLIAMS	$865,118
⊞ BARBARA JENSEN	$477,633

e. Sign out of Oracle BI.

Practice 9-6: Creating Parent-Child Hierarchies

Goal

To create a parent-child hierarchy

Scenario

A parent-child hierarchy is a hierarchy of members that all have the same type. This contrasts with level-based hierarchies, where members of the same type occur only at a single level of the hierarchy. The most common real-life occurrence of a parent-child hierarchy is an organizational reporting hierarchy chart, where the following all apply:

- Each individual in the organization is an employee.
- Each employee, apart from the top-level managers, reports to a single manager.
- The reporting hierarchy has many levels.

In relational tables, the relationships between different members in a parent-child hierarchy are implicitly defined by the identifier key values in the associated base table. However, for each Oracle BI Server parent-child hierarchy defined on a relational table, you must also explicitly define the inter-member relationships in a separate parent-child relationship table.

Outcome

A parent-child hierarchy is added to the business model.

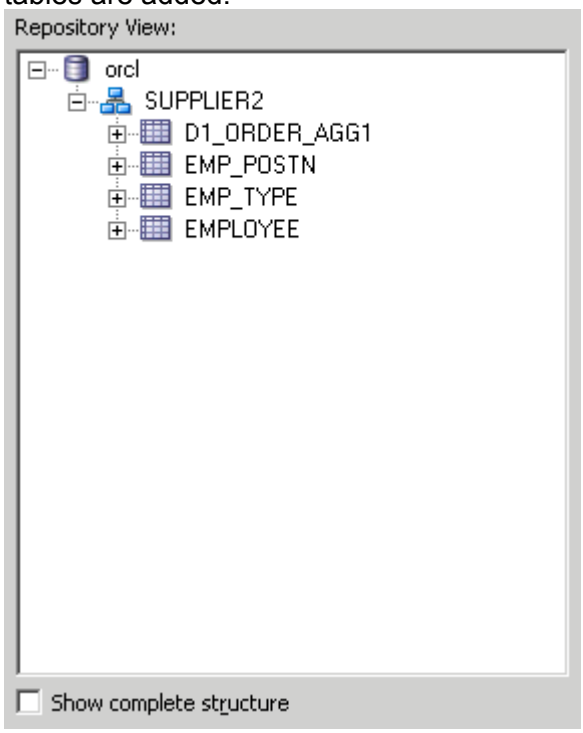
Time

40 minutes

Tasks

1. Import employee tables and an aggregate fact table.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. In the Physical layer, expand **orcl**.
   c. Right-click the **SUPPLIER CP** connection pool and select **import Metadata**.
   d. Accept the defaults in the Select Metadata Types window and click **Next**.
   e. In the Select Metadata Objects window, expand the **SUPPLIER2** schema in the Available pane.
   f. Select the following tables:
      D1_ORDER_AGG1
      EMP_POSTN
      EMP_TYPE
      EMPLOYEE
   g. Click the **Import selected** arrow to add the tables to the Selected pane.

h. When import completes, expand **SUPPLIER2** in the Selected pane to verify that the tables are added.



i. Click **Finish** to add the tables to the repository.

j. Expand **SUPPLIER2** in the Physical layer and verify that the tables were imported.

2. Explore the new tables.

a. Expand the **EMPLOYEE** table. Each row in the table contains two identifying keys, one to identify the member itself (EMPLOYEE_KEY) and the other to identify the "parent" of the member (MGR_ID). To see the data for this table, right-click the table and select **View Data**.

b. Expand the **EMP_POSTN** table. This table contains employee position information.

c. Expand the **EMP_TYPE** table. This table contains employee type information.

d. Expand the **D1_ORDER_AGG1** table. This table contains sales facts aggregated to the Sales Rep, Product Type, and Month levels.

e. Notice that the table missing from the Physical layer is the parent-child relationship table, which you create later in this practice.

3. Create aliases for the tables:

Table	Alias
D1_ORDER_AGG1	**Fact_D1_ORDER_AGG1**
EMP_POSTN	**Dim_EMP_POSTN**
EMP_TYPE	**Dim_EMP_TYPE**
EMPLOYEE	**Dim_EMPLOYEE**

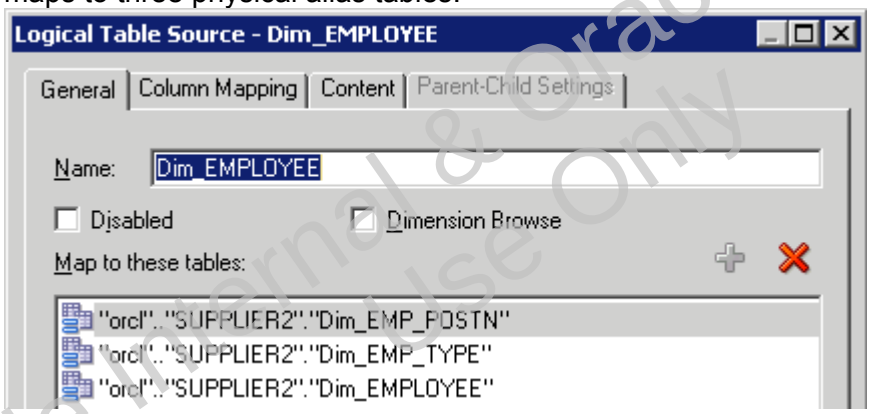
4. Use the Physical Diagram to create the following physical joins for the alias tables:

```
Dim_EMP_TYPE.TYPE_KEY = Dim_EMPLOYEE.EMP_TYPE_KEY
```

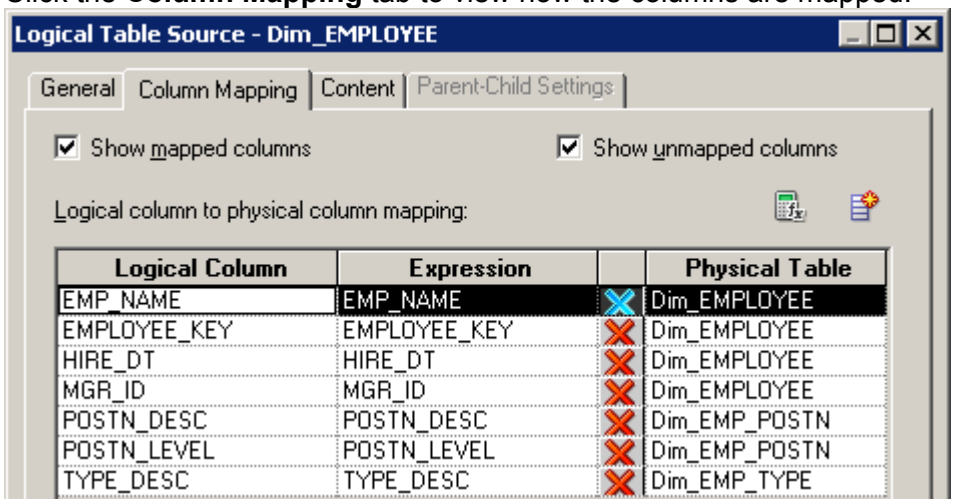
```
Dim_EMP_POSTN.POSTN_KEY = DIM_EMPLOYEE.EMP_POSTN_KEY
```

5. Add logical tables and columns for the Employee dimension to the SupplierSales business model.
   a. Right-click the **SupplierSales** business model and select **New Object > Logical Table**.
   b. Name the logical table **Dim-Employee** and click **OK**.
   c. Expand **Dim_EMPLOYEE** in the Physical layer.
   d. Drag the following columns from **Dim_EMPLOYEE** to **Dim-Employee**:
      EMP_NAME
      EMPLOYEE_KEY
      HIRE_DT
      MGR_ID
   e. Notice that this creates logical columns and a Dim_EMPLOYEE logical table source for the Dim-Employee logical table.

   f. Drag the **TYPE_DESC** column from **Dim_EMP_TYPE** to the **Dim_EMPLOYEE** logical table source (not the Dim-Employee logical table).
   g. Drag the **POSTN_DESC** and **POSTN_LEVEL** columns from **Dim_EMP_POSTN** to the **Dim_EMPLOYEE** logical table source (not the Dim-Employee logical table).
   h. Double-click the **Dim_EMPLOYEE** logical table source to open the Logical Table Source properties dialog box.
   i. Click the **General** tab and notice that the Dim_EMPLOYEE logical table source now maps to three physical alias tables.



j. Click the **Column Mapping** tab to view how the columns are mapped:



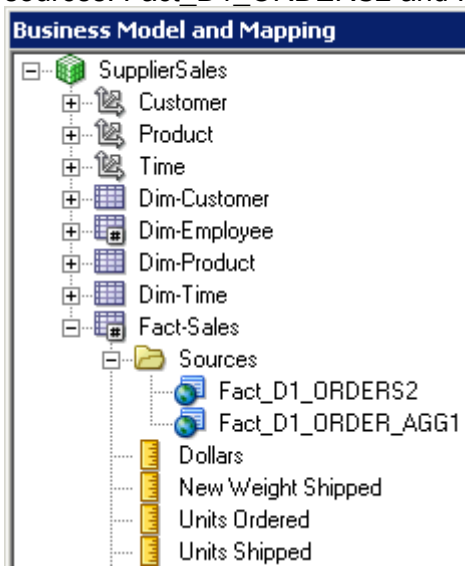
k. Click **OK** to close the Logical Table Source dialog box.

l. Rename the logical columns:

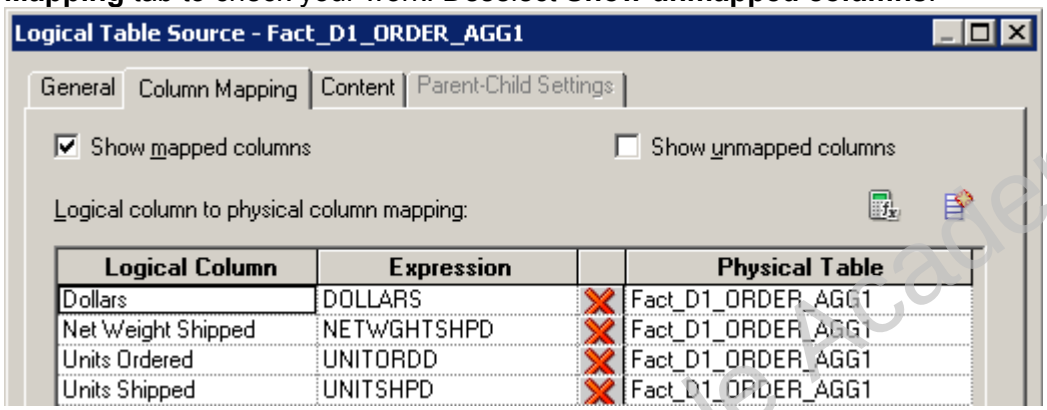
Logical column	Rename to:
EMP_NAME	**Employee Name**
EMPLOYEE_KEY	**Employee ID**
HIRE_DT	**Hire Date**
MGR_ID	**Manager ID**
TYPE_DESC	**Employee Type**
POSTN_DESC	**Position**
POSTN_LEVEL	**Position Level**

6. Set the key for the Dim-Employee logical table.
   a. Double-click the **Dim-Employee** logical table.
   b. Click the **Keys** tab.
   c. In the Key Name column, enter **Employee ID**.
   d. In the Columns field, use the drop-down list to select **Employee ID**.
   e. Click **OK**.
7. Create a new logical source and columns within the existing logical fact table.
   a. In the Physical layer, expand the **Fact_D1_ORDER_AGG1** table.
   b. In the Business Model and Mapping layer, expand **Fact-Sales**.
   c. Drag the **DOLLARS**, **NETWGHTSHPD, UNITSHPD**, and **UNITORDD** columns one at a time from Fact_D1_ORDER_AGG1, and drop each onto their corresponding Fact-Sales logical columns: **Dollars**, **Net Weight Shipped, Units Shipped**, and **Units Ordered**. This creates a new Fact_D1_ORDER_AGG1 logical table source and corresponding column mappings.

d. Expand the **Fact-Sales** > **Sources** folder. Notice that there are now two logical table sources: Fact_D1_ORDERS2 and Fact_D1_ORDER_AGG1.



e. Double-click the **Fact_D1_ORDER_AGG1** logical table source and click the **Column Mapping** tab to check your work. Deselect **Show unmapped columns**.



f. Click **OK** to close the Logical Table Source dialog box.

g. Double-click the **Fact_D1_ORDERS2** logical table source and click the **Column Mapping** tab. Notice that these four logical columns, Dollars, Units Shipped, Units Ordered, and Net Weight Shipped, now map to their corresponding columns in both the Fact_D1_ORDERS2 table and the Fact_D1_ORDER_AGG1 table.

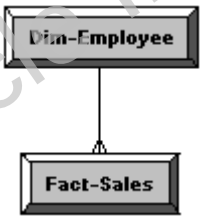
h. Click **OK** to close the Logical Table Source dialog box.

8. Create a logical join for the Dim-Employee table.

a. In the SupplierSales business model, select **Dim-Employee** and **Fact-Sales**.

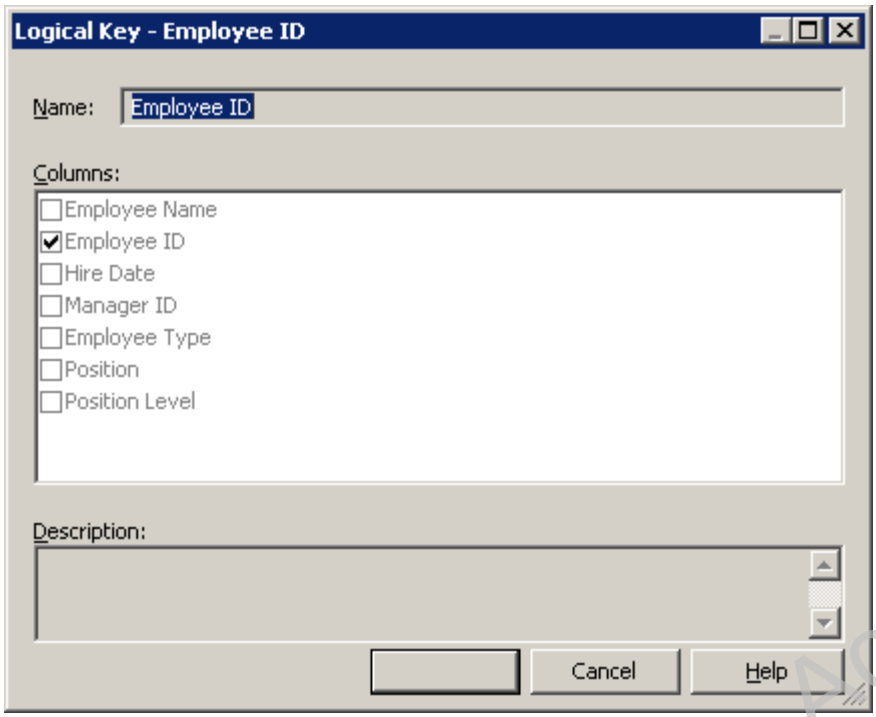
b. Right-click either of the highlighted tables and select **Business Model diagram > Selected Tables Only** to open the Business Model Diagram.

c. Create a logical join from **Dim-Employee** to **Fact-Sales**.



d. Close the Business Model Diagram.

9. Create a parent-child logical dimension based on the Dim-Employee logical table.
   a. Right-click the **Dim-Employee** logical table and select **Create Logical Dimension > Dimension with Parent-Child Hierarchy** to open the Logical Dimension dialog box.
   b. On the General tab, name the logical dimension **Employee**.
   c. Click **Browse** next to Member Key. The browse window shows the logical dimension table in the business model with its corresponding keys.
   d. Click **View** to view the key column. Verify that the key points to the **Employee ID** column.

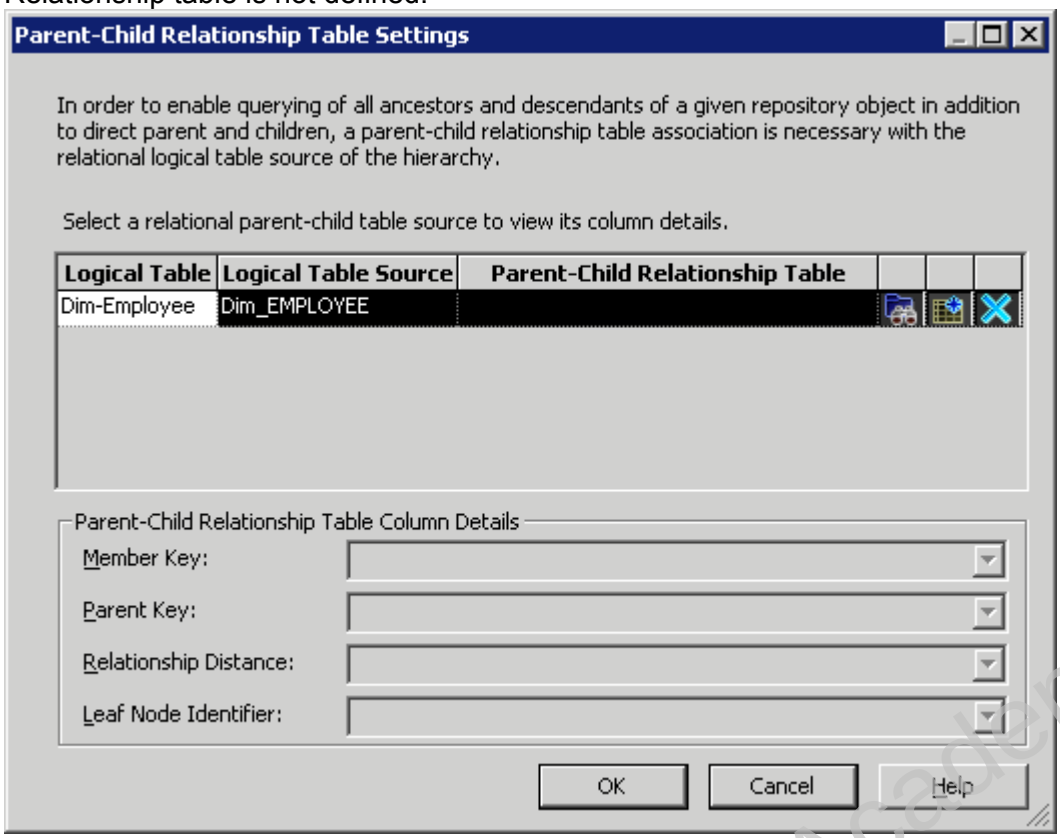


   e. Click **Cancel**.
   f. Click **OK** in the Browse window.
   g. Click **Browse** next to Parent Column.
   h. The **Browse** window shows the columns, other than the primary key, in the logical table that you selected in the previous step.
   i. Deselect **Show Qualified name**.
   j. Select **Manager ID** as the Parent Column for the parent-child hierarchy and click **OK**.
   k. Do not close the Logical Dimension dialog box.

10. Define the parent-child settings. At this point, if the logical table that you selected was not from a relational table source, you could click OK in the Logical Dimension dialog box to finish the process of creating the dimension. However, because the logical table you selected is from a relational table source, you must continue the dimension definition process to set up a parent-child relationship table for the hierarchy. For each parent-child hierarchy defined on a relational table, you must also explicitly define the inter-member relationships in a separate parent-child relationship table. In the process of creating the parent-child relationship table, you may choose one of the following options: 1. Select a previously-created parent-child relationship table. 2. Use a wizard that will generate scripts to create and populate the parent-child relationship table. In the next set of steps, you use a wizard to generate scripts to create and populate the parent-child relationship table. Later in

this practice you perform the steps to select a previously-created parent-child relationship table.

a. In the Logical Dimension dialog box, select **Parent-Child Settings** to display the Parent-Child Table Settings dialog box. Notice that at this point the Parent-Child Relationship table is not defined.



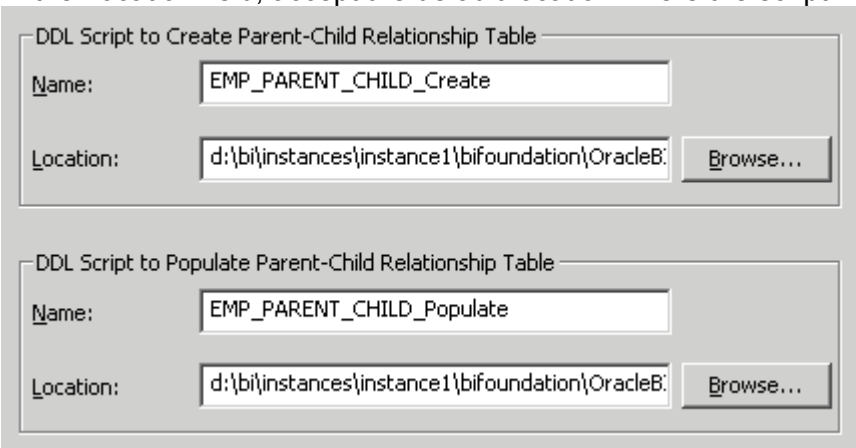
b. Click the **Create Parent-Child Relationship Table button** (blue asterisk) to start the wizard. As you will see, the wizard generates SQL scripts for creating and populating the parent-child relationship table. At the end of the wizard, Oracle BI Server stores the scripts into directories chosen during the wizard session. The scripts, when executed, will make the parent-child relationship table available to the repository.

c. In the Generate Parent-Child Relationship Table - Script Location window, enter **EMP_PARENT_CHILD_Create** as the name for the DDL script to create the parent-child table.

d. In the Location field, accept the default location where the script will be stored.

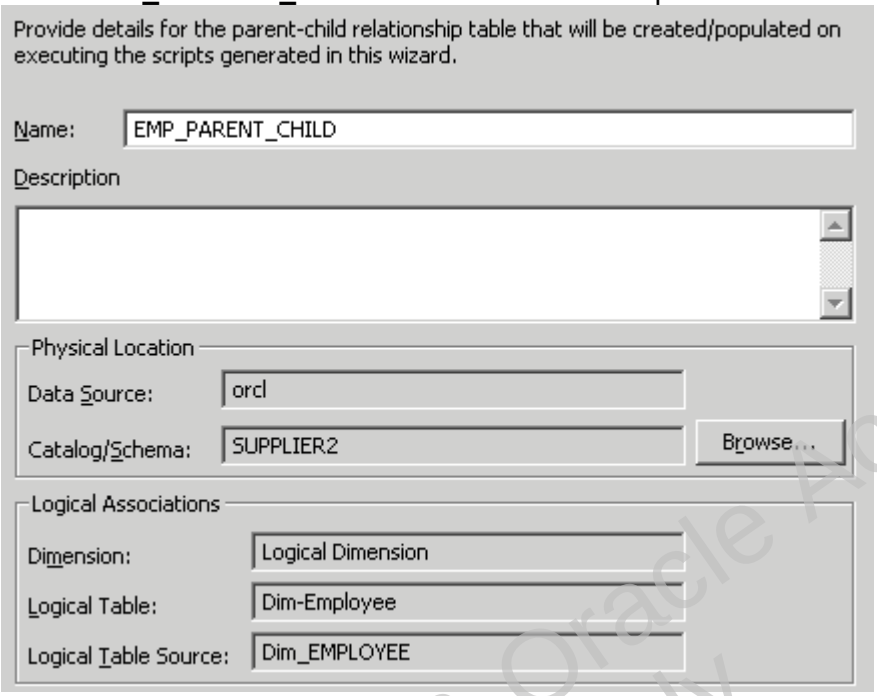
e. Enter **EMP_PARENT_CHILD_Populate** as the name for the DDL script to populate the parent-child table.

f. In the Location field, accept the default location where the script will be stored.



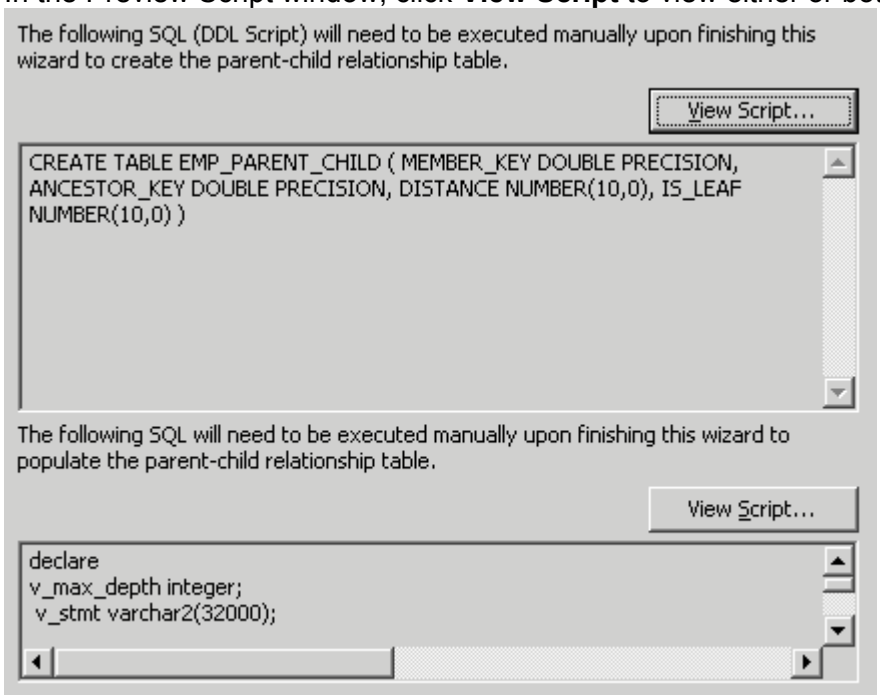
g. Click **Next**.

h. Enter **EMP_PARENT_CHILD** as the name for the parent-child relationship table.



i. Accept the defaults for the physical location and logical associations, and click **Next**.

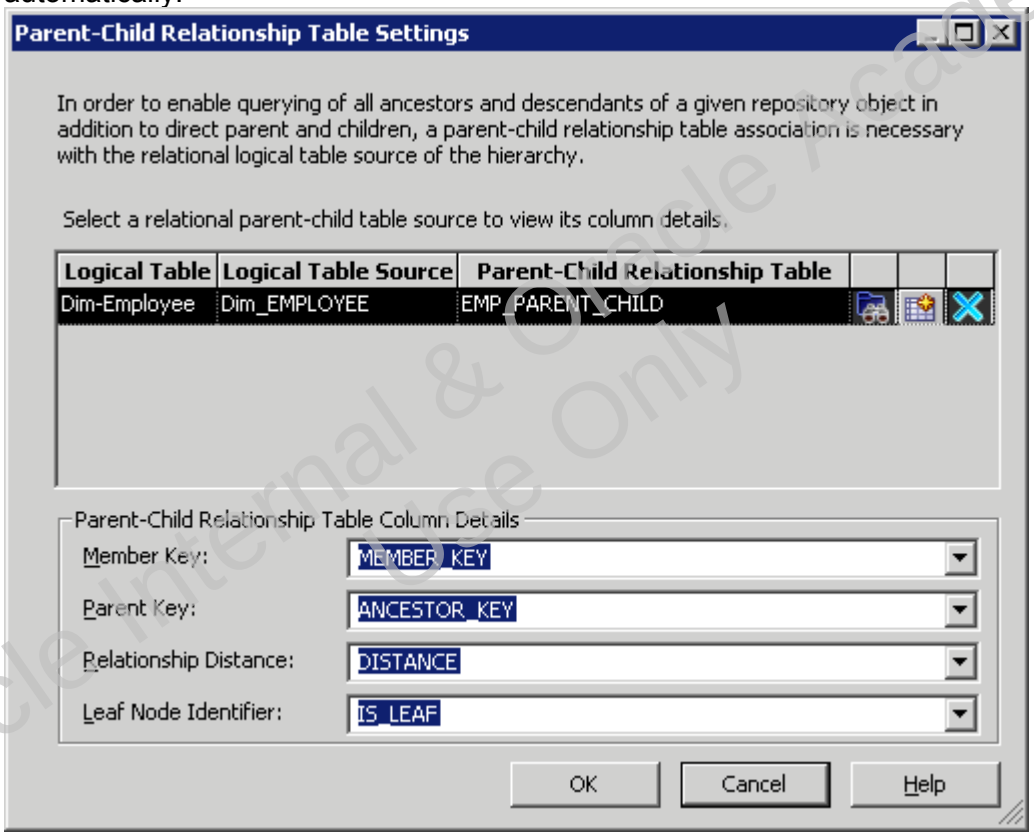
j. In the Preview Script window, click **View Script** to view either or both of the scripts.



k. Click **Finish** to close the wizard.

l. Notice that **EMP_PARENT_CHILD** is now defined as the parent-child table for the logical table source.

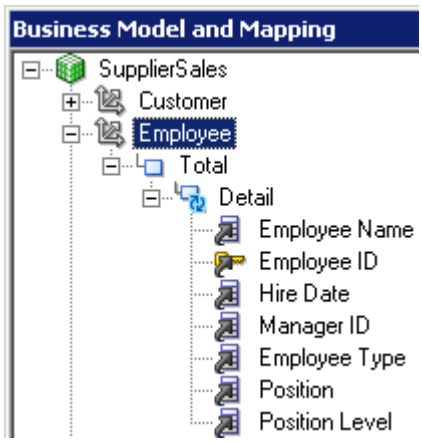
m. Notice also that the parent-child table column details have been populated automatically.



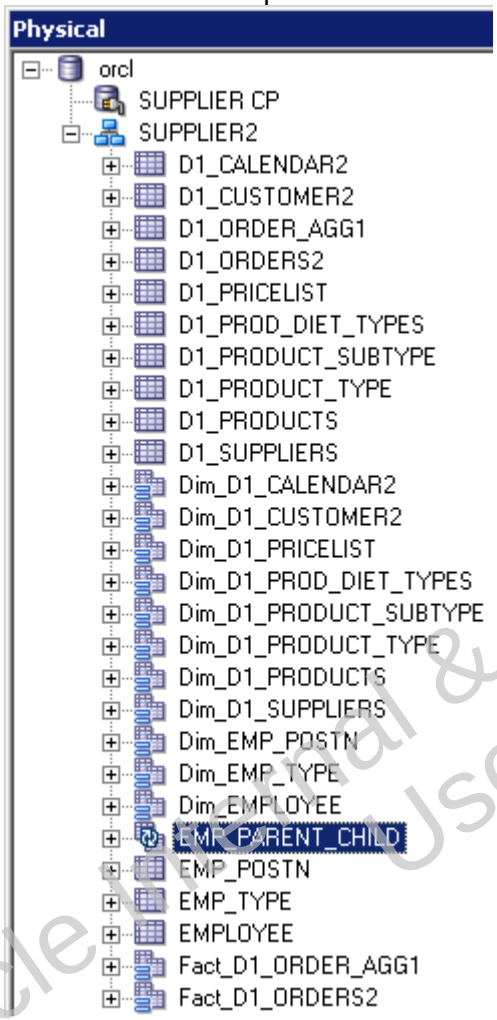
n. Click **OK** to close the Parent-Child Table Settings dialog box.

o. Click **OK** to close the Logical Dimension dialog box.

p. Notice that the Employee parent-child logical dimension hierarchy is added to the business model.



q. Notice also that the parent-child relationship table is created in the Physical layer.



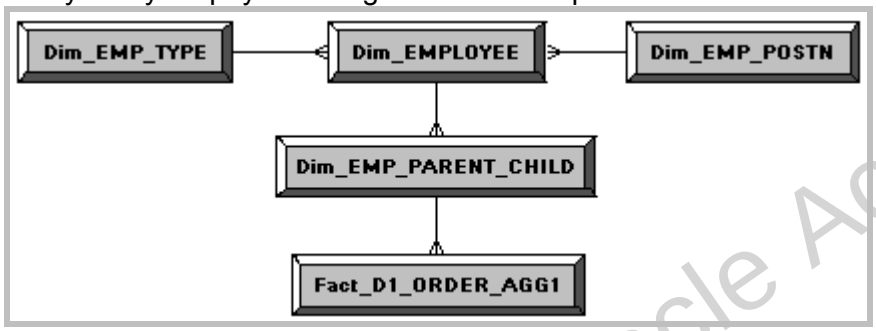
r. The wizard also saves the DDL scripts to the selected locations. The next step (not shown here) would be to run the scripts to create and populate this parent-child relationship table in the database. For the purposes of this training, the

EMP_PARENT_CHILD table has already been created and populated in the database. The EMP_PARENT_CHILD physical layer object now points to the EMP_PARENT_CHILD physical table in the database.

11. Modify Physical layer objects. After adding the parent-child relationship table to the Physical layer, you must make some modifications in both the Physical layer and the Business Model and Mapping layer.
    a. Create an alias for the **EMP_PARENT_CHILD** table named **Dim_EMP_PARENT_CHILD**.
    b. Select the following tables and open the Physical Diagram:
       Dim_EMP_TYPE
       Dim_EMPLOYEE
       Dim_EMP_POSTN
       Dim_EMP_PARENT_CHILD
       Fact_D1_ORDER_AGG1
    c. Create the following new join relationships in the Physical layer:

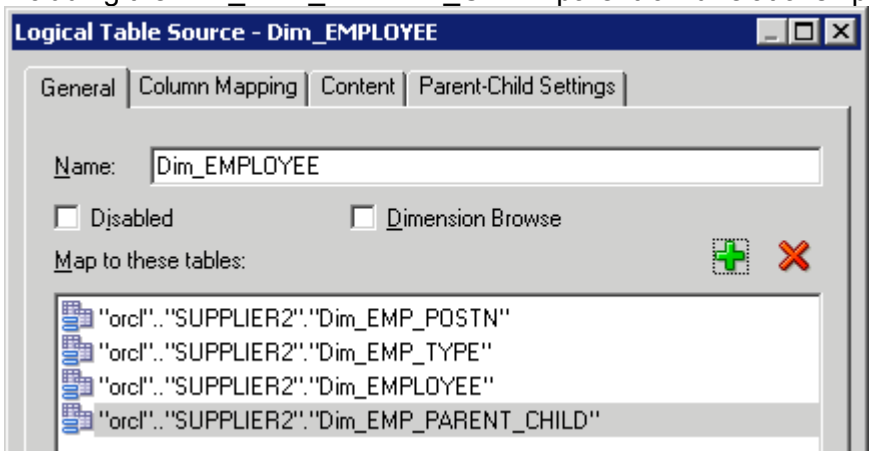
```
"orcl"."".."SUPPLIER2"."Dim_EMPLOYEE"."EMPLOYEE_KEY" =
"orcl"."".."SUPPLIER2"."Dim_EMP_PARENT_CHILD"."ANCESTOR_KEY"

"orcl"."".."SUPPLIER2"."Dim_EMP_PARENT_CHILD"."MEMBER_KEY" =
"orcl"."".."SUPPLIER2"."Fact_D1_ORDER_AGG1"."SREP_KEY"
```

    d. Verify that your physical diagram relationships look similar to the following screenshot:



    e. Close the Physical Diagram.
12. Modify business model objects.
    a. In the Physical layer, expand **Dim_EMP_PARENT_CHILD**.
    b. In the Business Model and Mapping layer, expand **Dim-Employee > Sources**.
    c. Drag the **DISTANCE** column from Dim_EMP_PARENT_CHILD to the Dim_EMPLOYEE logical table source. This action creates a new logical column and maps the Dim_EMPLOYEE logical table source to the Dim_EMP_PARENT_CHILD table.
    d. Double-click the **Dim_EMPLOYEE** logical table source.
    e. Click the **General** tab.

f. Notice that the Dim_EMPLOYEE logical table source is now mapped to four tables including the Dim_EMP_PARENT_CHILD parent-child relationship table.



g. Click **OK** to close the Logical Table Source dialog box.

h. Rename the **DISTANCE** logical column to **Distance**.

13. This next set of steps demonstrates the second technique for defining the parent-child relationship table, which is to select a previously-created, existing parent-child relationship table. In this example, you select the Dim_EMP_PARENT_CHILD alias table.

a. Double-click the **Employee** parent-child logical dimension to open the properties dialog box.

b. Click **Parent-Child Settings** to open the Parent-Child Table Settings dialog box.

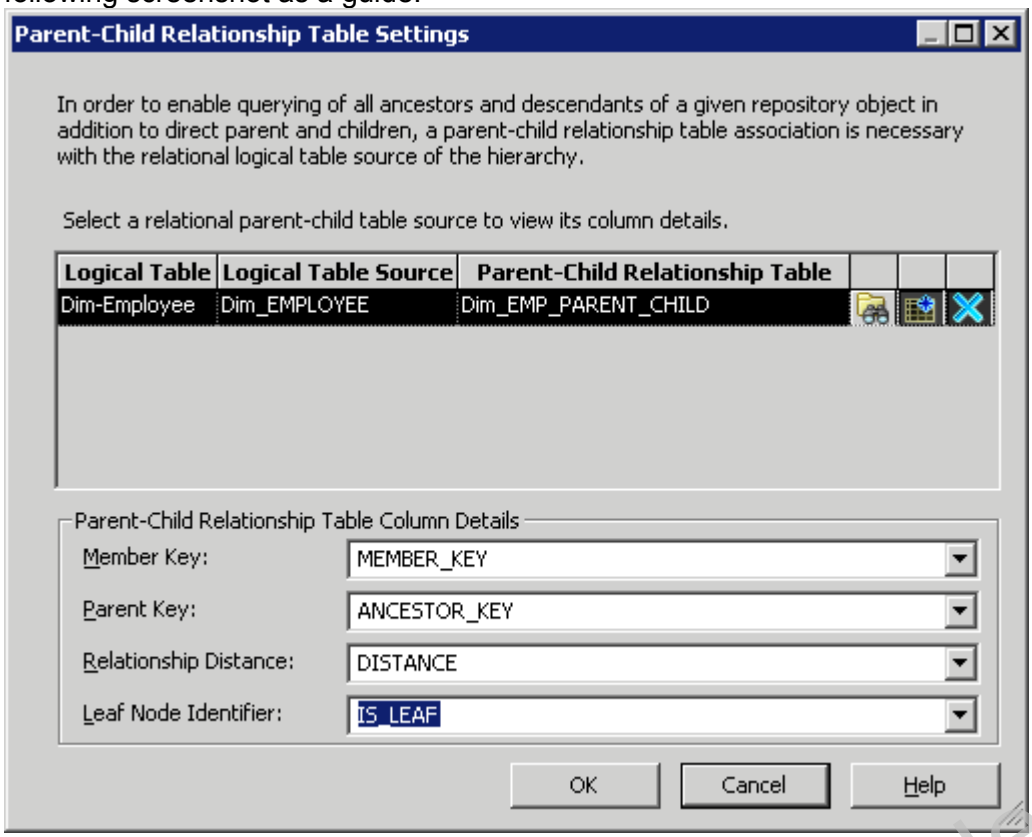
c. Notice that, as expected, the parent-child table is the table generated by the wizard: EMP_PARENT_CHILD.

d. To set the parent-child table to an existing parent-child relationship table, click the **Select Parent-Child Relationship Table** button.

e. In the Select Physical Table dialog box, select the **Dim_EMP_PARENT_CHILD** alias that you created.

f. The parent-child table is now set to the alias table instead of the original physical table generated by the wizard. Recall that you defined your physical joins using the alias table.

g. In the Parent-Child Table Column Details section, set the appropriate columns. Use the following screenshot as a guide:



h. Click **OK** to close the Parent-Child Table Settings dialog box.

i. Click **OK** to close the Logical Dimension dialog box.

14. Set aggregation content levels for the Fact-Sales logical table sources. Because you added a new logical dimension, you must set the logical levels.

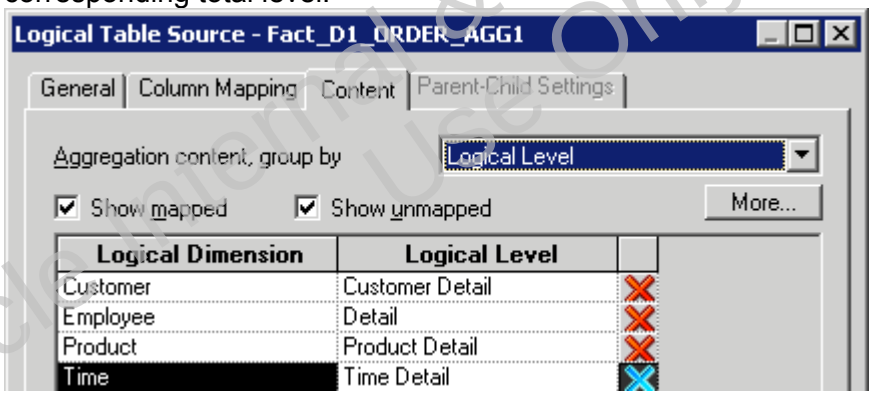
a. Expand **Fact-Sales > Sources**.

b. Double-click the **Fact_D1_ORDERS2** logical table source.

c. On the Content tab, set the logical level for the Employee logical dimension to **Total**.

d. Double-click the **Fact_D1_ORDER_AGG1** logical table source.

e. On the Content tab, set the Employee level to Detail and all other logical levels to the corresponding total level.



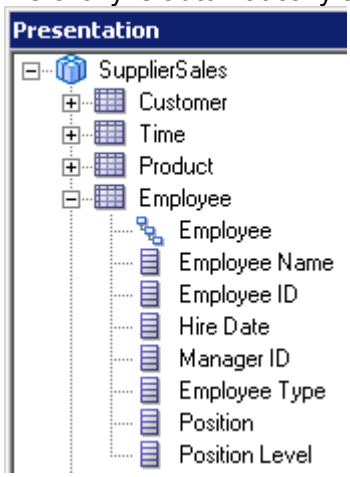
15. Make the parent-child logical dimension available for queries.

a. Drag the **Dim-Employee** logical table to the SupplierSales subject area.

b. Rename the **Dim-Employee** presentation table **Employee.**

c. Move the **Employee** presentation table above the Fact-Sales presentation table.

d. Expand the **Employee** presentation table and notice that the Employee presentation hierarchy is automatically added to the presentation table.



e. Double-click the **Employee** presentation hierarchy to open the Presentation Hierarchy dialog box.

f. On the **Display Columns** tab, select **Employee ID.**

g. Click the **red X** to remove this display column.

h. Click the **Add** button (green plus sign).

i. Select **Employee Name** as a display column.

j. Click **OK** to close the Presentation Hierarchy dialog box.

k. Save the repository.

l. Check consistency. Fix any errors or warnings before proceeding.

m. Close the repository.

16. Use Fusion Middleware Control Enterprise Manager to upload the repository.

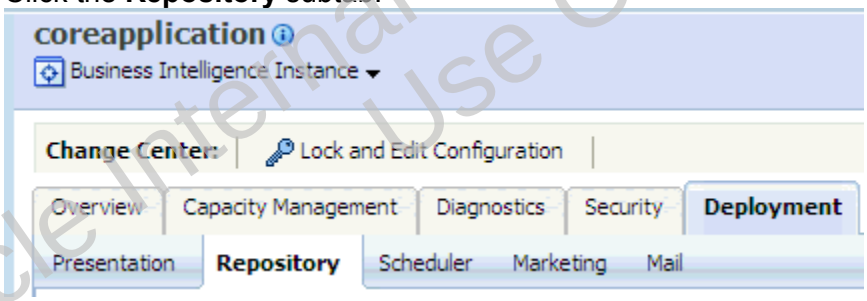
a. Return to Fusion Middleware Control Enterprise Manager, which should still be open in your browser. If it is not open, enter the following URL: **http://localhost:7001/em**.

b. If your session has timed out, log in as **weblogic/welcome1**.

c. In the left pane, expand **Business Intelligence** and select **coreapplication**.

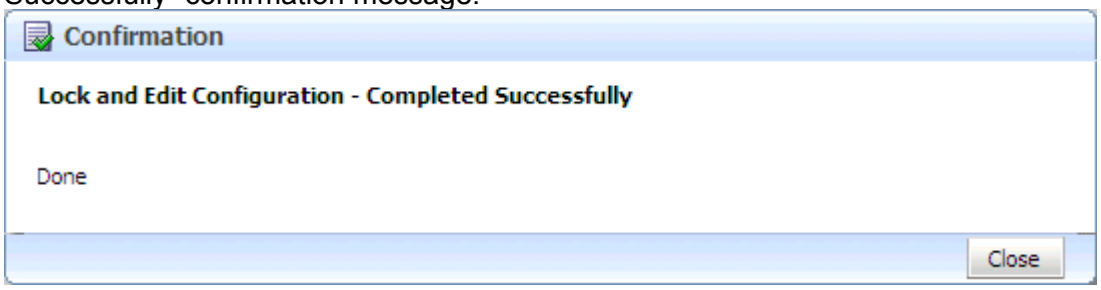
d. In the right pane, click the **Deployment** tab.

e. Click the **Repository** subtab.



f. Click **Lock and Edit Configuration**.

g. Click **Close** when you receive the “Lock and Edit configuration – Completed Successfully” confirmation message.



h. In the Upload BI Server Repository section, click **Browse** to open the “Choose file” dialog box.

i. By default, the “Choose file” dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.

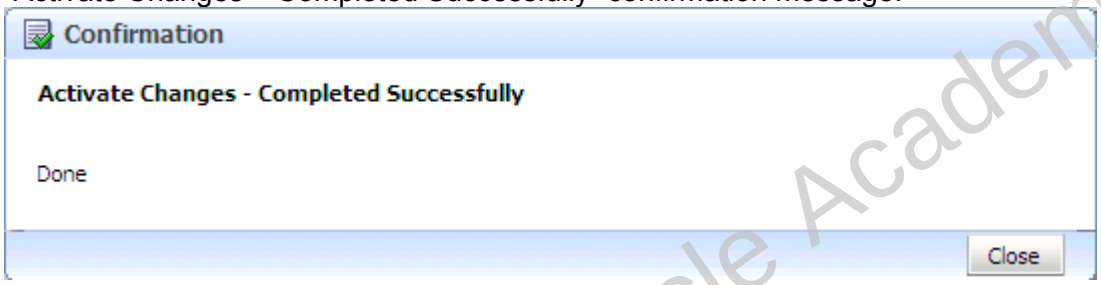
j. Select **ABC.rpd** and click **Open**. You can also double-click the repository to open it.

k. Enter **welcome1** in the Repository Password and Confirm Password fields.

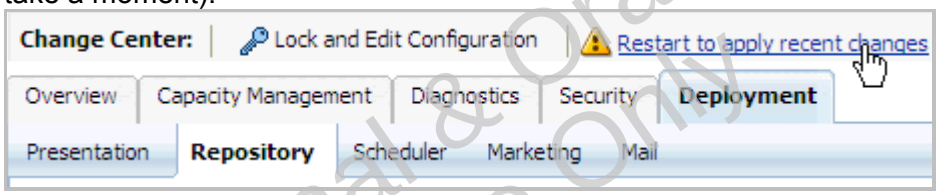
l. Click **Apply**. Notice that Default RPD now displays **ABC** with an extension, for example, ABC_BI0007.

m. Click **Activate Changes**.

n. Allow Active Changes processing to complete. Click **Close** when you receive the “Activate Changes – Completed Successfully” confirmation message.

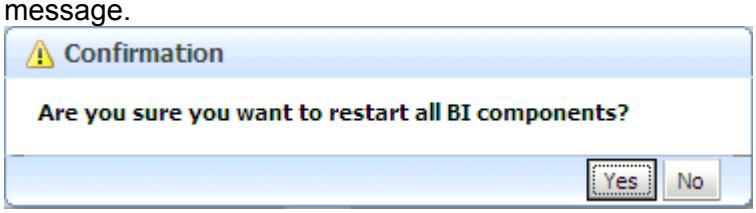


o. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).

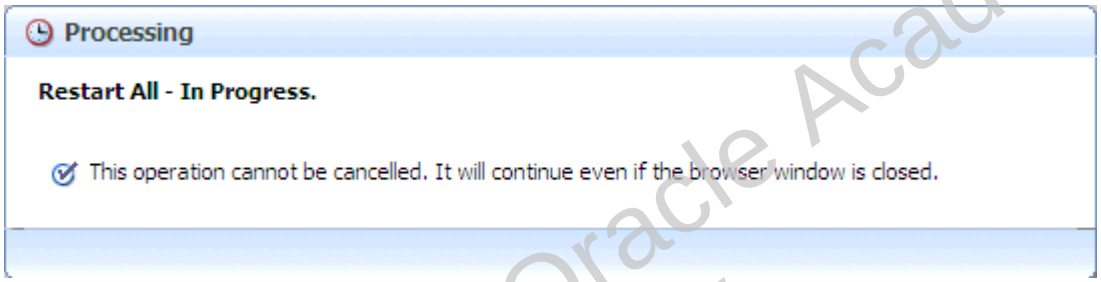


p. On the Overview page, click **Restart**.

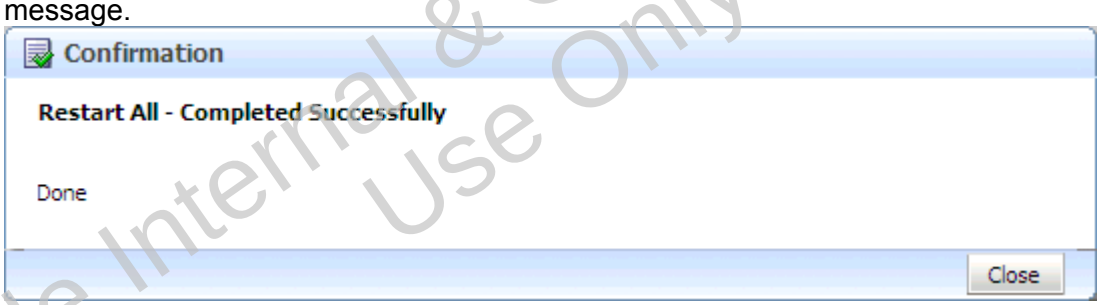
q. Click **Yes** when you receive the "Are you sure you want to restart all BI components?" message.



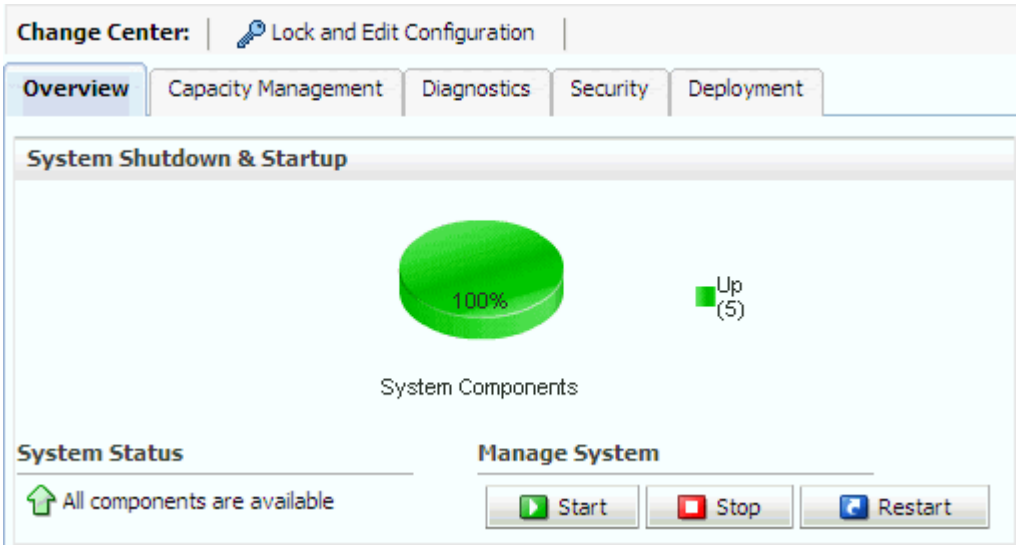
r. Allow the Restart All – In Progress processing to complete. This may take a few moments.



s. Click **Close** when you receive the "Restart All – Completed Successfully" confirmation message.



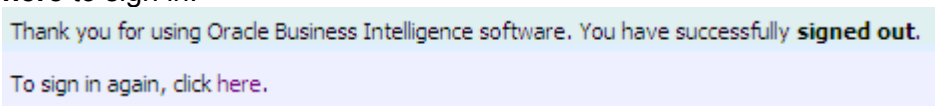
t. Confirm that System Components = 100%. The ABC repository is loaded into BI Server.



u. Leave Enterprise Manager open.

17. Open Analysis Editor to execute queries and test the SupplierSales business model.

a. Return to the Oracle Business Intelligence browser tab where you signed out and click **here** to sign in.



b. Sign in as **weblogic** with password **welcome1**.

c. In the Create section, click **Analysis** to open the Select Subject Area window.

18. Create an analysis to check your work.

a. Create the following analysis:

b. Check **Results**. Expand the pivot table to view data at different levels of the hierarchy. Notice that the Dollars measure rolls up through each level.

		Dollars
Employee	Position	
⊟ PAULA MADISON	Sales VP	$63,132,455
⊞ GEORGE MASUR	Sr. Supervisor	$601,455
⊞ LYLE IRWIN	Sr. Manager	$12,991,176
⊟ MARY SILVER	Manager	$18,355,438
⊞ JOSE CRUZ	Supervisor	$7,243,603
⊞ KATHY LOBO	Supervisor	$5,169,673
⊟ LILLIAN BAYER	Supervisor	$5,127,514
DALE AREND	Associate	$1,111,591
DONALD KIMBRIEL	Senior	$689,577
TIM ALLEN	Associate	$873,659
TRACIE BELL	Associate	$113,121
⊞ MAYNARD WAGNER	Manager	$29,367,508

c. Leave Oracle BI open for the next practice.

Practice 9-7: Using Calculated Members

Goal

To create a user-defined dimension member whose measure values are calculated at run time

Scenario

A calculated member is a user-defined dimension member whose measure values are calculated at run time. You define a calculated member within a dimension through a formula that references other members of the same dimension. Calculated members can be defined for a single dimension or across multiple dimensions.

In the Analysis Editor, there are also several places where you can issue logical SQL. In this practice, you build CALCULATEDMEMBER SQL statements and run them on the Issue SQL page on the Administration tab.

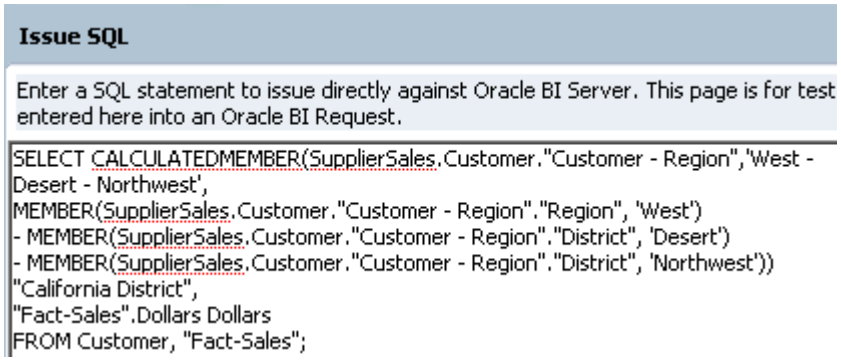
Time

30 Minutes

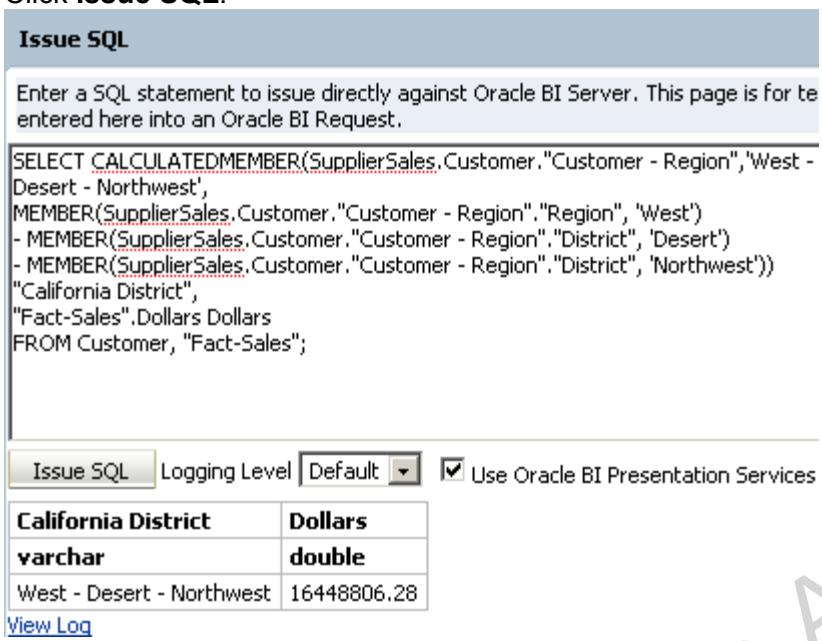
Tasks

1. Create and run a CALCULATEDMEMBER query using a single calculated member and a single dimension.
   a. Click the **Administration** link.
   b. Click **OK** to navigate away from this page.
   c. Under Maintenance and Troubleshooting, click **Issue SQL**.
   d. Enter the following SQL statement. (Alternatively, you can copy the SQL from the file CALCULATEDMEMBER_SINGLE.txt located in D:\PracticeFiles.)

```
SELECT
CALCULATEDMEMBER(SupplierSales.Customer."Customer -
Region",'West - Desert - Northwest',
MEMBER(SupplierSales.Customer."Customer - Region"."Region",
'West')
- MEMBER(SupplierSales.Customer."Customer - Region"."District",
'Desert')
- MEMBER(SupplierSales.Customer."Customer - Region"."District",
'Northwest'))
"California District",
"Fact-Sales".Dollars Dollars
FROM Customer, "Fact-Sales";
```



e. Click **Issue SQL**.



2. Syntax and results explanation:
   a. **`SupplierSales.Customer."Customer - Region"`** identifies the fully-qualified presentation hierarchy in the presentation layer on which the calculated member is based.
   b. **`'West - Desert - Northwest'`** is the string that identifies the calculated member.
   c. **`MEMBER(SupplierSales.Customer."Customer - Region"."Region", 'West') - MEMBER(SupplierSales.Customer."Customer - Region"."District", 'Desert') - MEMBER(SupplierSales.Customer."Customer - Region"."District", 'Northwest'))`** is the calculated member formula, which consists of one or more examples of a "member clause" connected by standard arithmetic operators. In this example you are calculating total dollars for the West region, then subtracting dollars for the Desert and Northwest districts. Since there are only three districts in the West region, the remainder is total dollars for the California district.
   d. **`"California District", "Fact-Sales".Dollars Dollars`** provides formatting for the SQL results.

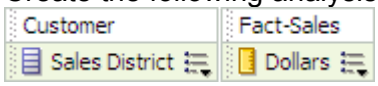
e. `FROM Customer, "Fact-Sales";` identifies the logical tables accessed by the SQL statement.

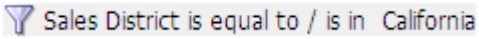
f. Notice the Dollars amount returned by the calculated member statement: **16,448,806.28**. In the next set of steps, you run an analysis to verify that this result is correct.

3. Use an analysis to verify your results.

a. Click **Home**.

b. Create the following analysis and filter:





c. Click **Results**. Verify that you results match those returned by the calculated member statement.

Sales District	Dollars
California	$16,448,806

4. Create and run a CALCULATEDMEMBER query using a multiple calculated member and multiple dimensions.

a. Click the **Administration** link.

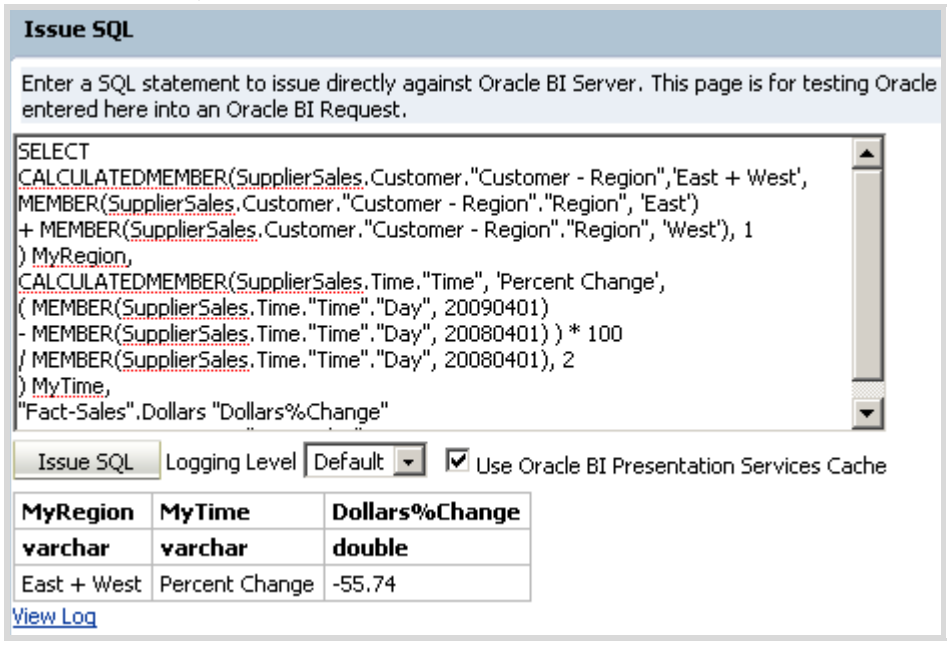
b. Click **OK** to navigate away from this page.

c. Under Maintenance and Troubleshooting, click **Issue SQL**.

d. Enter the following SQL statement. (Alternatively, you can copy the SQL from the CALCULATEDMEMBER_MULTIPLE.txt file located in D:\PracticeFiles.)

```
SELECT
CALCULATEDMEMBER(SupplierSales.Customer."Customer -
Region",'East + West',
MEMBER(SupplierSales.Customer."Customer - Region"."Region",
'East')
+ MEMBER(SupplierSales.Customer."Customer - Region"."Region",
'West'), 1
) MyRegion,
CALCULATEDMEMBER(SupplierSales.Time."Time", 'Percentage
Increase',
( MEMBER(SupplierSales.Time."Time"."Time Detail", 20090401)
- MEMBER(SupplierSales.Time."Time"."Time Detail", 20080401) ) *
100
/ MEMBER(SupplierSales.Time."Time"."Time Detail", 20080401), 2
) MyTime,
"Fact-Sales".Dollars DollarsPerCent
FROM Customer, Time, "Fact-Sales";
```

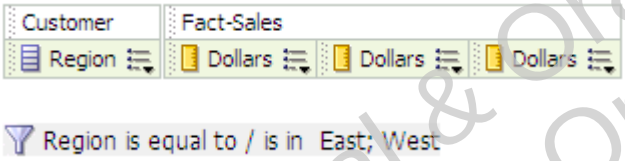
e. Click **Issue SQL**.



The requirement in this example is to determine the percentage change in dollars over time for two regions. This SQL statement calculates the percent dollar change between April 01 2008 and April 01 2009 for the East and West regions. To achieve the correct results, the solve order is significant. You must first add dollars for the two regions across the time periods, and then perform the percentage calculation. The solve order is determined by the 1 after the first CALCULATEDMEMBER clause and the 2 after the second CALCULATEDMEMBER clause. Notice that two dimensions, Customer-Region and Time, are accessed by this query.

f. Notice the **Dollars%Change** amount returned by the calculated member statement: **-55.74**. In the next set of steps you run an analysis to verify that this result is correct.

5. Create an analysis to verify your results.
   a. Click **Home**.
   b. Create the following analysis and filter:



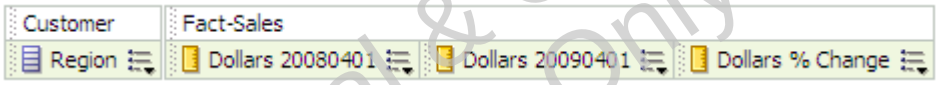
6. Create a bin for the Region column.
   a. For the Region column, select **Edit Formula**.
   b. Click the **Bins** tab.
   c. Click **Add Bin**.
   d. In the Value field, enter **East;West**.
   e. Click **OK**.
   f. Enter **East + West Regions** as the bin name and click **OK**.
   g. Click **OK** to close the Edit Column Formula dialog box.
7. Edit the first Dollars column to return data only for April 01, 2008.
   a. Click **Edit Formula** for the first Dollars column.

b. Click **Filter**.
c. In the left panel, expand **Time** and double-click **Day** to add it to the filter expression.
d. In the New Filter dialog box, enter **20080401** in the Value field and click **OK**.
e. Click **OK** again to insert the filter.
f. Click **Custom Headings**.
g. Change the column heading to **Dollars 20080401**.
h. Click **OK** to close the Edit Column Formula dialog box.

8. Edit the second Dollars column to return data only for April 01, 2009.
   a. Click **Edit Formula** for the second Dollars column.
   b. Click **Filter**.
   c. In the left panel, expand **Time** and double-click **Day** to add it to the filter expression.
   d. In the New Filter dialog box, enter **20090401** in the value field and click **OK**.
   e. Click **OK** again to insert the filter.
   f. Click **Custom Headings**.
   g. Change the column heading to **Dollars 20090401**.
   h. Click **OK** to close the Edit Column Formula dialog box.
9. Edit the third Dollars column to create a formula to calculate the percent change in dollars from April 01, 2008 to April 01, 2009.
   a. Click **Edit Formula** for the third Dollars column.
   b. Delete **"Fact-Sales"."Dollars"** from the formula window.
   c. Use the **Column** button and **operators** to create the following formula to calculate percent dollar change between April 01 2008 and April 01 2009 for the East and West regions. You can also copy the formula from **CALCULATEDMEMBER_FORMULA.txt** in D:\PracticeFiles.

```
((FILTER("Fact-Sales".Dollars USING (Time.Day = 20080401)) -
FILTER("Fact-Sales".Dollars USING (Time.Day = 20090401))) /
FILTER("Fact-Sales".Dollars USING (Time.Day = 20080401))) * 100
```

   d. Click **Custom Headings**.
   e. Change the column heading to **Dollars % Change**.
   f. Click **OK**.



   g. Modify the column properties for the **Dollars 20080401** and **Dollars 20090401** columns so that the data displays as currency.
   h. Modify the column properties for the **Dollars % Change** column so that data displays as a percentage.
   i. Click **Results**. Verify that you results match those returned by the calculated member statement.

Region	Dollars 20080401	Dollars 20090401	Dollars % Change
East + West Regions	$149,293	$66,081	55.74%

Practices for Lesson 10: Using Aggregates

Lesson 10

Practices for Lesson 10

Lesson Overview

In these practices, you will add aggregate tables to the business model to improve performance.

Practice 10-1: Using Aggregate Tables

Goal

To use aggregated data tables to improve performance of summarized queries

Scenario

ABC wants to add aggregate tables that store pre-computed results that are aggregated measures over a set of dimensional attributes. You must specify the level of aggregation for each source using logical levels. The necessary aggregate tables are already in the database:

- **MONTHS** contains one row for each year and month combination, which can be considered an aggregation of the Time dimension to the Month level.
- **D1_PRODUCT_TYPE** is already part of your model and contains one row for each product type, which can be considered an aggregation of the Product dimension to the Product Type level.
- **D1_ORDER_AGG1** is already part of your model and contains sales facts aggregated to the Product Type, and Month levels.

Outcome

In the Physical layer, there are new physical sources for the aggregate tables. In the Business Model and Mapping layer, aggregate content is defined for the logical table sources.

Time

45 minutes

Tasks

1. Import the MONTHS physical aggregate table from the database into the Physical layer of the repository. You need both aggregate fact and aggregate dimension tables because you must create logical dimension sources at the same level of detail as the fact sources.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. In the Physical layer, expand **orcl**.
   c. Right-click the **SUPPLIER CP** connection pool and select **Import Metadata**.
   d. Accept the defaults in the Select Metadata Types screen and click **Next** to open the Select Metadata Objects screen.
   e. Scroll to the **SUPPLIER2** schema and expand it.
   f. In the **Data source view** pane, select the **MONTHS** table for import.
   g. Click the **Import selected** button to add the table to the Repository View pane.
   h. Expand **SUPPLIER2** in the Repository View pane and verify that **MONTHS** is added.
   i. Click **Finish** to add the tables to the repository.
   j. Expand **SUPPLIER2** in the Physical Layer and confirm that **MONTHS** is added to the repository.
   k. Expand **MONTHS.**
   l. Double-click the **YEAR** column to open the Physical Column dialog box.
   m. Change the column type to **INT**.
   n. Repeat for **MONTHCODE**.
   o. Create a **Dim_MONTHS** alias tables for MONTHS.

2. View the data in the aggregate tables.
   a. Right-click **Fact_D1_ORDER_AGG1** and select **View Data**. Fact_D1_ORDER_AGG1 contains sales facts aggregated to the Product Type (TYPEKEY), and Month (PERKEY) levels. It contains 10212 rows, compared to Fact_D1_ORDERS2, which contains 351636 rows.

DOLLARS	NETWGHTSHPD	PERKEY	SALESREP	SREP_KEY	TYPEKEY
751.22	461.09	200902.0	BARBARA JENSEN	5.0	112
1676.12	1758.03	200903.0	BARBARA JENSEN	5.0	112
5691.81	1100.25	200904.0	BARBARA JENSEN	5.0	112
44.01	53.0	200802.0	BARBARA JENSEN	5.0	113
30.65	52.0	200803.0	BARBARA JENSEN	5.0	113
39.17	43.0	200804.0	BARBARA JENSEN	5.0	113
10.97	21.0	200805.0	BARBARA JENSEN	5.0	113
21.94	42.0	200806.0	BARBARA JENSEN	5.0	113
8.71	10.0	200807.0	BARBARA JENSEN	5.0	113

   b. Close the **View Data** window.
   c. View data for the **Dim_MONTHS** table. The Dim_MONTHS table contains one row for each year and month combination, which is an aggregation of the Time dimension to the Month level. It contains 16 rows, compared to Dim_D1_CALENDAR2, which contains 474 rows.

MAGO	MONTH_IN_YEAR	MONTHCODE	MONTHNAME	QUARTER	QUARTERDESC	YAGO
200712.0	1	200801.0	January	1.0	Q1_2008	200701.0
200801.0	2	200802.0	February	1.0	Q1_2008	200702.0
200802.0	3	200803.0	March	1.0	Q1_2008	200703.0
200803.0	4	200804.0	April	2.0	Q2_2008	200704.0
200804.0	5	200805.0	May	2.0	Q2_2008	200705.0
200805.0	6	200806.0	June	2.0	Q2_2008	200706.0
200806.0	7	200807.0	July	3.0	Q3_2008	200707.0
200807.0	8	200808.0	August	3.0	Q3_2008	200708.0
200808.0	9	200809.0	September	3.0	Q3_2008	200709.0

   d. View data for the **Dim_D1_PRODUCT_TYPE** table. Dim_D1_PRODUCT_TYPE contains one row for each product type, which is an aggregation of the Product dimension to the Type level. It contains 21 rows, compared to Dim_D1_PRODUCTS, which contains 192 rows.

ITEMTYPE	TYPECODE
Baking	100
Beef	101
Beverage	102
Bread	103
Cereal	104
Cheese	105
Condiments	106
Dessert	107
EntrTe	108
Frozen	109

3. Create physical joins between the aggregate fact and aggregate dimension tables.
   a. In the Physical layer, select the following three tables:
      Dim_D1_PRODUCT_TYPE
      Dim_MONTHS
      Fact_D1_ORDER_AGG1
   b. Click the **Physical Diagram** icon on the toolbar.
   c. Rearrange the tables to make them visible in the Physical Diagram.

d. Use the **New foreign key** button and create the following joins:
```
Dim_MONTHS.MONTHCODE = Fact_D1_ORDER_AGG1.PERKEY
Dim_D1_PRODUCT_TYPE.TYPECODE = Fact_D1_ORDER_AGG1.TYPEKEY
```

e. Check your work:

f. Close the Physical Diagram.

4. Specify the aggregation content of the logical table source for the Fact-Sales logical table so that Oracle BI Server knows what level of data is stored in the aggregate tables.

a. In the Business Model and Mapping layer, expand **Fact-Sales > Sources**.

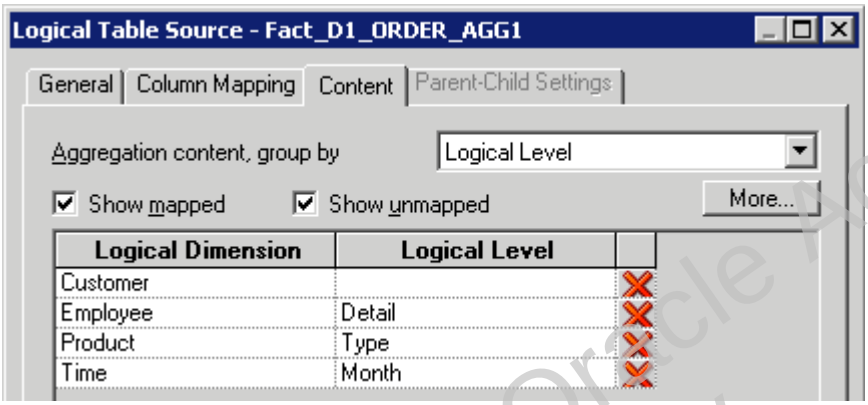
b. Double-click the **Fact_D1_ORDER_AGG1** logical table source to open its properties dialog box. Recall that you created this logical table source in the practices for the previous lesson.

c. Click the **Column Mapping** tab. Recall that the four measure columns, Dollars, Units Ordered, Units Shipped, and New Weight Shipped, map to both the Fact_D1_ORDERS2 and Fact_D1_ORDER_AGG1 tables.

d. Click the **Content** tab.

e. In the "Aggregation content, group by" field, ensure that the value is **Logical Level**. This is the default.

f. Use the drop-down lists in the Logical Level field to specify the aggregation content as follows:

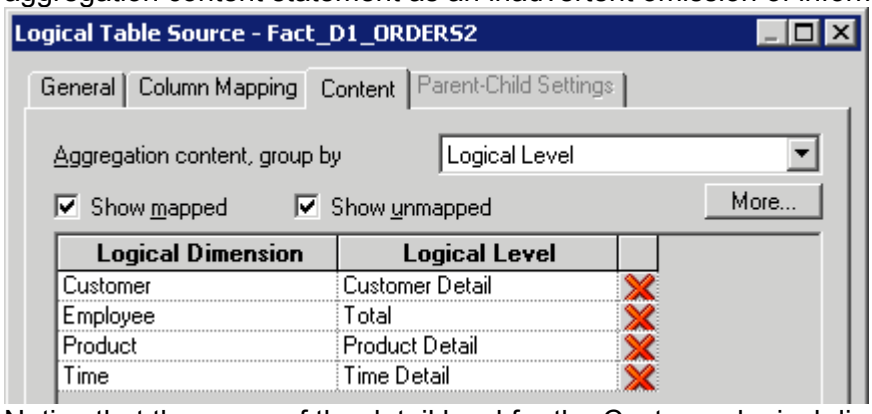


You are setting aggregation content for the fact table to the corresponding levels in the dimension hierarchies. In a subsequent step, you set similar levels for the dimension table aggregate sources. Later, when a user queries against a particular level, Oracle BI Server will "know" to access the aggregate tables instead of the detail tables. For example, if a user queries for total sales by month, the server will access the Fact_D1_ORDER_AGG1 aggregate fact table and the corresponding aggregate dimension table, Dim_MONTHS. If a user queries for a level lower than the levels specified here (for example, day instead of month), the server will access the detail tables (Fact_D1_ORDERS2, and Dim_D1_CALENDAR2). If a user queries for higher level (year instead of month), the aggregate tables will be used as well, because when a query is run against a logical level or above, the aggregate tables are used.

g. Click **OK** to close the Logical Table Source dialog box for Fact_D1_ORDER_AGG1.

h. Specify the content for the remaining fact logical table source, Fact_D1_ORDERS2. You are doing this because it is good practice to set the levels for the detail source to the lowest levels in the hierarchies. This is because you want the server to access the

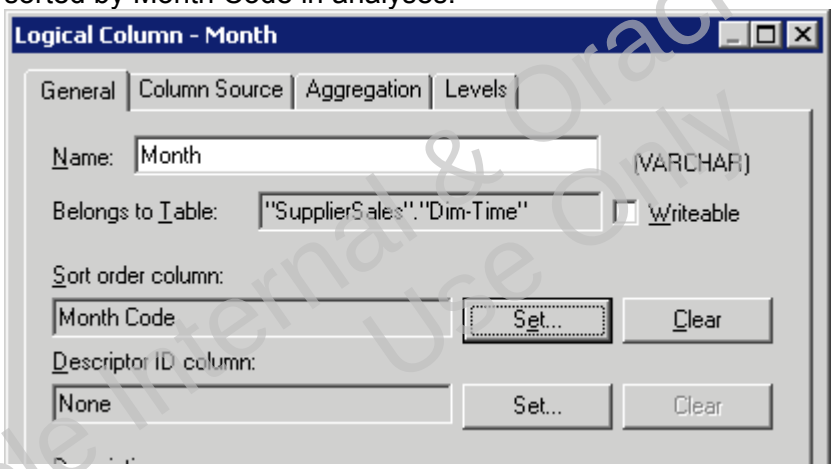
detail tables when queries are against levels lower than those specified for the aggregate tables. It is also a good practice to specify the content of all sources for documentation purposes, as another administrator could interpret the lack of an aggregation content statement as an inadvertent omission of information.



Notice that the name of the detail level for the Customer logical dimension may be different from the screenshot, depending on which method you used to create the Customer logical dimension in the previous set of practices. Notice also that Employee is a parent-child logical dimension, so the logical level is set to Total.

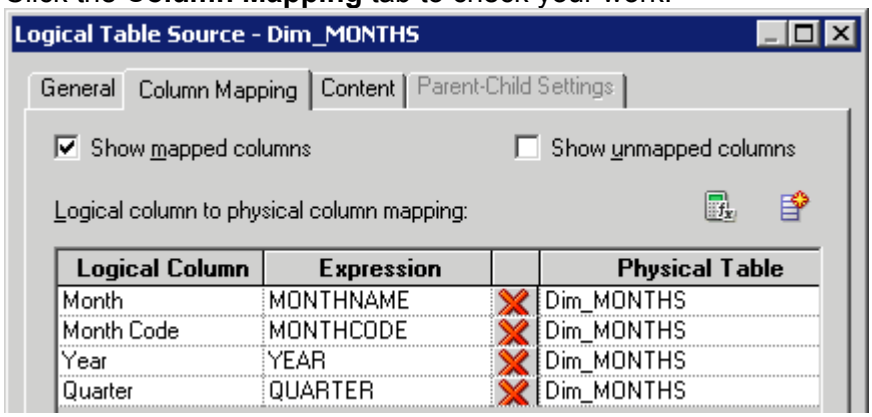
i. Click **OK** to close the Logical Table Source dialog box.

5. Create a new source within the Dim-Time logical table that points to the Dim_MONTHS aggregate table.
   a. In the Physical layer, expand the **Dim_MONTHS** table.
   b. In the Business Model and Mapping layer, expand **Dim-Time > Sources**.
   c. Drag the **MONTHCODE**, **MONTHNAME, QUARTER**, and **YEAR** columns from the Dim_MONTHS aggregate table onto the corresponding Dim-Time logical columns to create a new logical table source, Dim_MONTHS, and the corresponding column mappings (drag MONTHNAME to MONTH).
   d. Double-click the **Month** column to open the Logical Column dialog box.
   e. On the General tab, set the sort-order column to **Month Code**, so that Month is always sorted by Month Code in analyses.



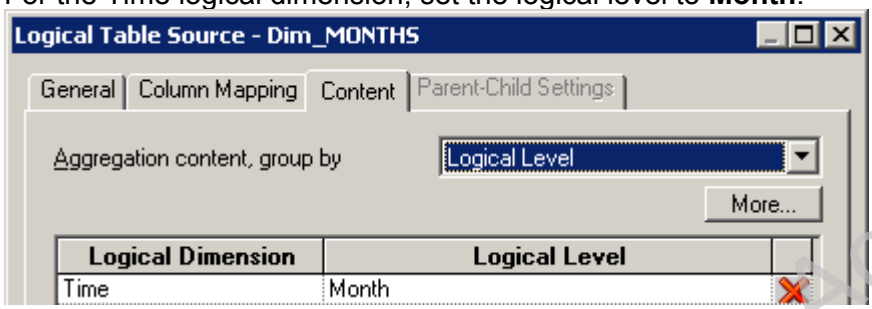
   f. Click **OK** to close the Logical Column dialog box.
   g. Double-click the **Dim_MONTHS** logical table source to open the Logical Table Source dialog box.

h. Click the **Column Mapping** tab to check your work.



i. Leave the Logical Table Source dialog box open.

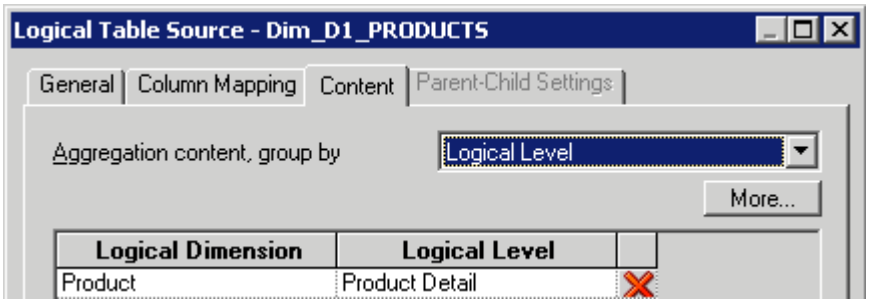
6. Specify the aggregation content for the Dim_MONTHS logical table source for the Dim-Time logical table so that Oracle BI Server knows what level of data is stored in the aggregate table. Recall that the Dim_MONTHS table contains data aggregated at the month level within the Time logical dimension hierarchy.
   a. Select the **Content** tab in the Dim_MONTHS logical table source.
   b. In the "Aggregation content, group by" field, ensure that the value is **Logical Level**.
   c. For the Time logical dimension, set the logical level to **Month**:

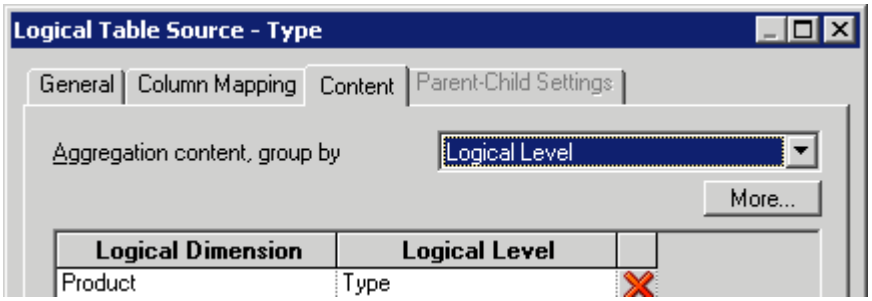


   d. Click **OK**.
   e. Confirm that the logical level is set to **Time Detail** for the remaining logical table source, Dim_D1_CALENDAR2. Again, it is best practice to set the levels for the detail source to the lowest logical level in the hierarchy.

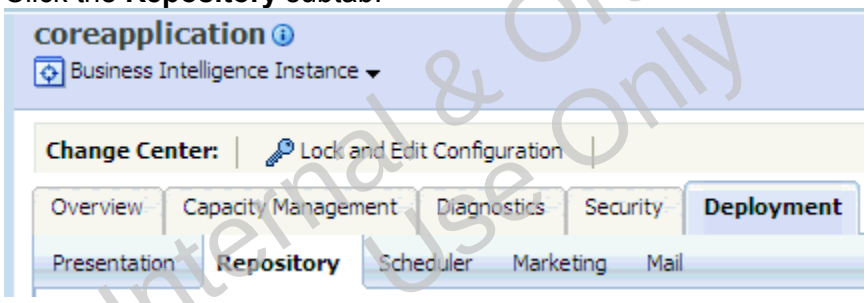
   f. Click **OK** to close the Logical Table Source dialog box.

7. Apply a similar process to set the aggregation content for the logical table sources for the Dim-Product logical table. Recall that you already added Type as a second logical table source. Be sure to change "Aggregation content, group by" from Column to Logical Level for both logical table sources. Use the following screenshots as a guide.



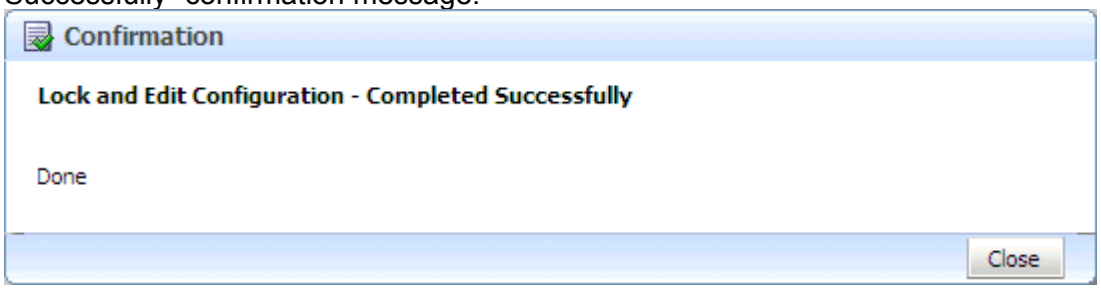


8. Save the repository.
9. Check global consistency. Fix any errors or warnings before you continue.
10. Close the repository. Leave the Administration Tool open.
11. Notice that you do not need to change the Presentation layer. You made changes in the business model that impact how the queries are processed and which sources are accessed. However, the user interface remains the same, so there is no need to change the Presentation layer. It will automatically use the new sources.
12. Use Fusion Middleware Control Enterprise Manager to upload the repository.
    a. Return to Fusion Middleware Control Enterprise Manager, which should still be open in your browser. If it is not open, enter the following URL: **http://localhost:7001/em**.
    b. If your session has timed out, log in as **weblogic/welcome1**.
    c. In the left pane, expand **Business Intelligence** and select **coreapplication**.
    d. In the right pane, click the **Deployment** tab.
    e. Click the **Repository** subtab.



    f. Click **Lock and Edit Configuration**.

g. Click **Close** when you receive the “Lock and Edit configuration – Completed Successfully” confirmation message.



h. In the Upload BI Server Repository section, click **Browse** to open the “Choose file” dialog box.

i. By default, the “Choose file” dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.

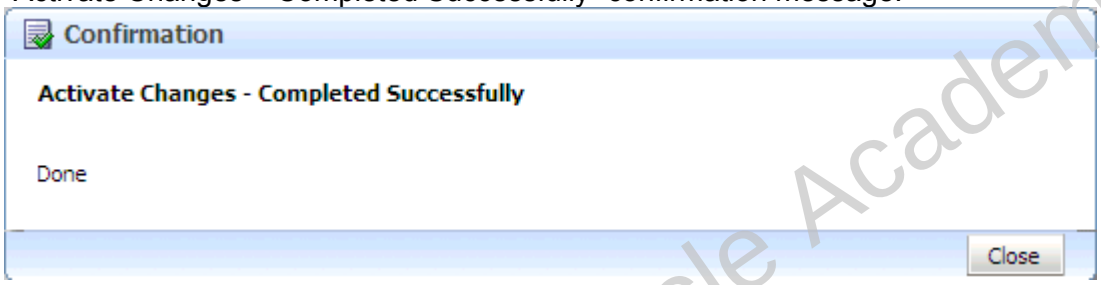
j. Select **ABC.rpd** and click **Open**. You can also double-click the repository to open it.

k. Enter **welcome1** in the Repository Password and Confirm Password fields.

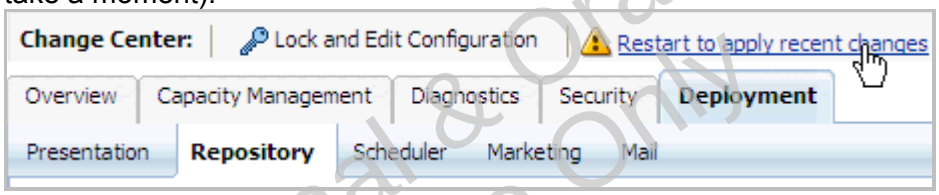
l. Click **Apply**. Notice that Default RPD now displays **ABC** with an extension (for example, ABC_BI0007).

m. Click **Activate Changes**.

n. Allow Active Changes processing to complete. Click **Close** when you receive the “Activate Changes – Completed Successfully” confirmation message.

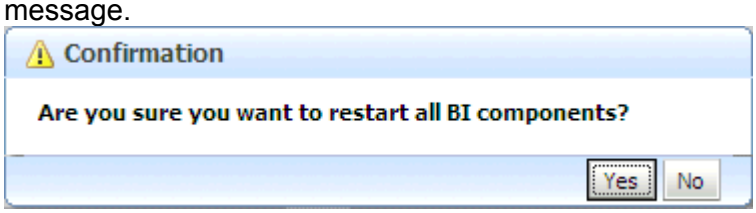


o. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).

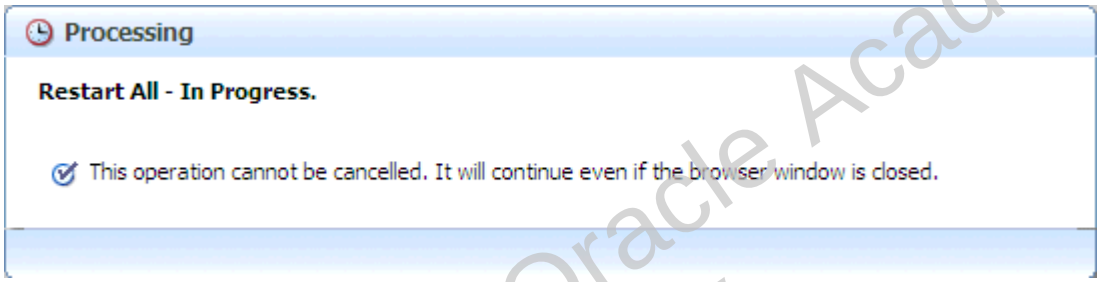


p. On the Overview page, click **Restart**.

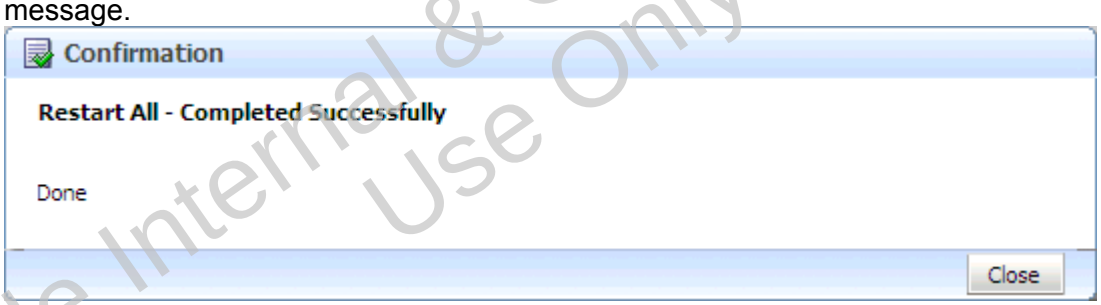
q. Click **Yes** when you receive the "Are you sure you want to restart all BI components?" message.



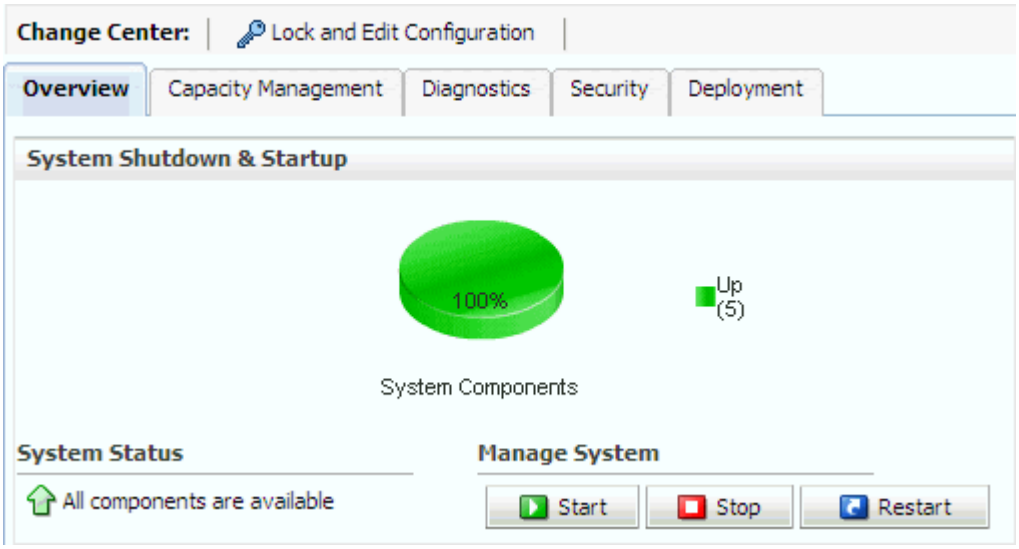
r. Allow the Restart All – In Progress processing to complete. This may take a few moments.



s. Click **Close** when you receive the "Restart All – Completed Successfully" confirmation message.



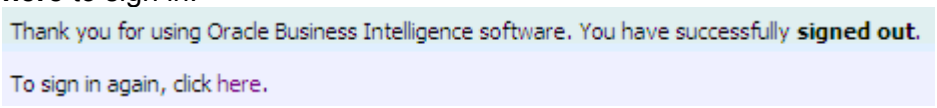
t. Confirm that System Components = 100%. The ABC repository is loaded into BI Server.



u. Leave Enterprise Manager open.

13. Open Analysis Editor to execute queries and test the SupplierSales business model.

a. Return to the Oracle Business Intelligence browser tab where you signed out and click **here** to sign in.



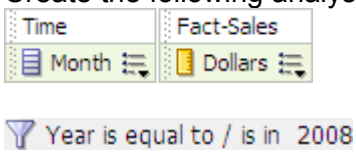
b. Sign in as **weblogic** with password **welcome1**.

c. In the Create section, click **Analysis** to open the Select Subject Area window.

d. Click **SupplierSales** to open Analysis Editor.

14. Create and run analyses to test your work.

a. Create the following analysis and filter:



b. Click **Results**.

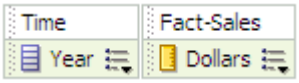
Month	Dollars
January	$3,595,669
February	$3,945,187
March	$3,975,774
April	$3,907,292
May	$4,061,558
June	$3,994,531
July	$4,062,212
August	$4,242,611
September	$3,810,263
October	$4,596,372
November	$3,655,169
December	$3,997,616

c. Leave Analysis Editor open.

d. Inspect the query log and confirm that the query uses the Fact_D1_ORDER_AGG1 aggregate fact table and the related D1_MONTHS aggregate dimension table.

```
------- Sending query to database named orcl (id: <<3116>>),
connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T744.DOLLARS) as c1,
     T862.MONTHNAME as c2
from
     MONTHS T862 /* Dim_MONTHS */ ,
     D1_ORDER_AGG1 T744 /* Fact_D1_ORDER_AGG1 */
where  ( T744.PERKEY = T862.MONTHCODE and T862.YEAR = 2008 )
group by T862.MONTHNAME)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c1 as c3
from
     SAWITH0 D1
order by c2
```

e. Return to Analysis Editor and create a new analysis.



f. Click **Results**.

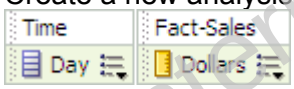
Year	Dollars
2008	$47,844,253
2009	$15,288,202

g. Inspect the query log.

```
--------- Sending query to database named orcl (id: <<3099>>),
connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T744.DOLLARS) as c1,
     T862.YEAR as c2
from
     MONTHS T862 /* Dim_MONTHS */ ,
     D1_ORDER_AGG1 T744 /* Fact_D1_ORDER_AGG1 */
where  ( T744.PERKEY = T862.MONTHCODE )
group by T862.YEAR)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c1 as c3
from
     SAWITH0 D1
order by c2 NULLS FIRST
```

Notice that the query still uses the same aggregate tables. This is because Year is at a higher level than Month in the Time logical dimension hierarchy, so the aggregate tables are still used.

h. Create a new analysis.



i. Click **Results**.

Day	Dollars
20080102	$26,036
20080103	$12,210
20080105	$140,140
20080106	$85,079
20080107	$399,278
20080108	$151,171
20080109	$107,925
20080110	$110,789

j. Inspect the query log.

```
--------- Sending query to database named orcl (id: <<3381>>),
connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T90.DOLLARS) as c1,
     T58.YYYYMMDD as c2
from
     D1_CALENDAR2 T58 /* Dim_D1_CALENDAR2 */ ,
     D1_ORDERS2 T90 /* Fact_D1_ORDERS2 */
where  ( T58.YYYYMMDD = T90.PERIODKEY )
group by T58.YYYYMMDD)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c1 as c3
from
     SAWITH0 D1
order by c2 NULLS FIRST
```

Notice that the detail fact table, Fact_D1_ORDERS2, and the detail dimension table, Dim_D1_CALENDAR2, are accessed instead of the aggregate tables. This is because the requested data is at a lower level than what is contained in the aggregate tables. Therefore, the aggregate tables do not contain the data and the detail tables are used in the query.

k. Sign out of Oracle BI.

Practice 10-2: Setting the Number of Elements

Goal

To set the number of elements for logical dimension levels

Scenario

In this practice, you set the number of elements for logical dimension levels and observe the results. The number of elements is used by Oracle BI Server when picking aggregate sources. Setting the number of elements is only necessary when there are two or more aggregate sources that could be accessed by an Oracle BI query. Aggregate fact sources are accessed based on a combination of the fields selected as well as the number of elements of the levels in the logical dimensions to which they map. The number does not have to be exact, but ratios of numbers from one logical level to another should be accurate. By adjusting the number of elements, you can alter the aggregate fact source selected by the Oracle BI Server.

Time

30 minutes

Tasks

1. Import an additional aggregate table into the Physical layer of the repository.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. In the Physical layer, expand **orcl**.
   c. Right-click the **SUPPLIER CP** connection pool and select **Import Metadata**.
   d. Accept the defaults in the Select Metadata Types screen and click **Next** to open the Select Metadata Objects screen.
   e. Scroll to the **SUPPLIER2** schema and expand it.
   f. In the **Data source view** pane, select the **D1_ORDER_AGG2** table for import.
   g. Click the **Import selected** button to add the table to the Repository View pane.
   h. Expand **SUPPLIER2** in the Repository View pane and verify that **D1_ORDER_AGG2** is added.
   i. Click **Finish** to add the table to the repository.
   j. Expand **SUPPLIER2** in the Physical Layer and confirm that the **D1_ORDER_AGG2** table is added to the repository.
   k. Create a **Fact_D1_ORDER_AGG2** alias table for **D1_ORDER_AGG2.**
2. View the data in the aggregate tables.
   a. View the data for **Fact_D1_ORDER_AGG2**. Fact_D1_ORDER_AGG2 contains sales facts aggregated to the District (DISTKEY), Product (PRODKEY), and Month (PERKEY) levels. It has 25,373 rows.

DISTKEY	DOLLARS	NETWGHTSHPD	PERKEY	PRODKEY
California	454.79	1505.0	200803.0	1105
California	563.82	2123.0	200804.0	1105
California	550.16	1896.0	200805.0	1105
California	404.96	1371.0	200806.0	1105
California	504.82	1574.38	200807.0	1105
California	411.98	1744.0	200808.0	1105
California	227.76	630.88	200809.0	1105
California	395.52	1466.0	200810.0	1105
California	342.02	1494.0	200811.0	1105
California	377.93	1318.5	200812.0	1105
California	511.49	1744.0	200901.0	1105
California	349.39	1083.06	200902.0	1105

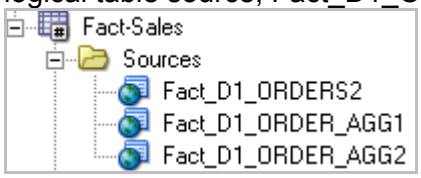
Compare this to Fact_D1_ORDER_AGG1, which contains sales facts aggregated to the Product Type (TYPEKEY) and Month (PERKEY) levels.

DOLLARS	NETWGHTSHPD	PERKEY	SALESREP	SREP_KEY	TYPEKEY
751.22	461.09	200902.0	BARBARA JENSEN	5.0	112
1676.12	1758.03	200903.0	BARBARA JENSEN	5.0	112
5691.81	1100.25	200904.0	BARBARA JENSEN	5.0	112
44.01	53.0	200802.0	BARBARA JENSEN	5.0	113
30.65	52.0	200803.0	BARBARA JENSEN	5.0	113
39.17	43.0	200804.0	BARBARA JENSEN	5.0	113
10.97	21.0	200805.0	BARBARA JENSEN	5.0	113
21.94	42.0	200806.0	BARBARA JENSEN	5.0	113
8.71	10.0	200807.0	BARBARA JENSEN	5.0	113

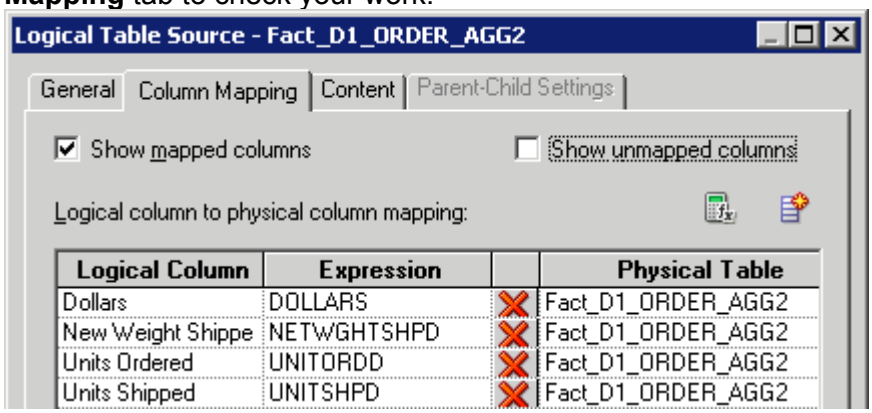
3. Create physical joins between the aggregate fact and dimension tables.
   a. In the Physical layer, select the following three tables:
      Dim_D1_PRODUCTS
      Dim_MONTHS
      Fact_D1_ORDER_AGG2
   b. Click the **Physical Diagram** icon on the toolbar.
   c. Rearrange the tables to make them visible in the Physical Diagram.
   d. Use the **New foreign key** button and create the following joins:
      Dim_MONTHS.MONTHCODE = Fact_D1_ORDER_AGG2.PERKEY
      Dim_D1_PRODUCTS.PRODUCTKEY = Fact_D1_ORDER_AGG2.PRODKEY
   e. Check your work:

   f. Close the Physical Diagram.
4. Create a new logical source and columns within the existing logical fact table.
   a. In the Physical layer, expand the **Fact_D1_ORDER_AGG2** table.
   b. In the Business Model and Mapping layer, expand **Fact-Sales**.
   c. Drag the **DOLLARS**, **NETWGHTSHPD, UNITSHPD**, and **UNITORDD** columns one at a time from Fact_D1_ORDER_AGG2 and drop each onto their corresponding Fact-Sales logical columns: **Dollars**, **Net Weight Shipped, Units Shipped**, and **Units Ordered**. This creates a new Fact_D1_ORDER_AGG2 logical table source and corresponding column mappings.
   d. Expand **Fact-Sales** > **Sources**. Notice that there are now two aggregate logical table sources, Fact_D1_ORDER_AGG1 and Fact_D1_ORDER_AGG2**,** and one detail

logical table source, Fact_D1_ORDERS2.

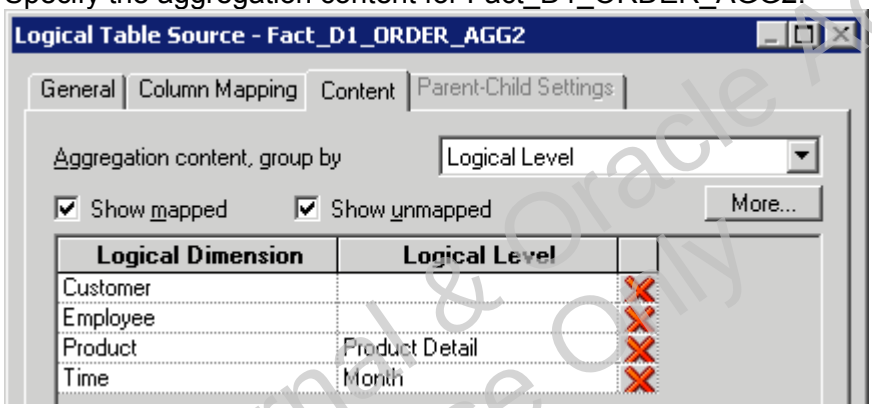


e. Double-click the **Fact_D1_ORDER_AGG2** logical table source and click the **Column Mapping** tab to check your work.

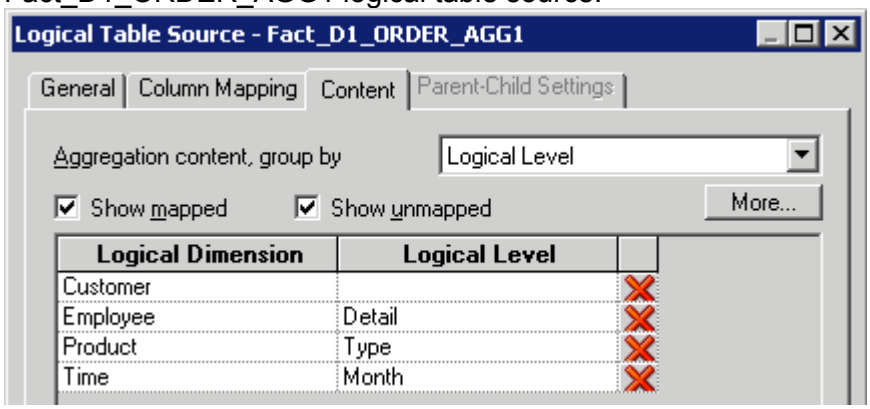


Notice that these four logical columns, Dollars, Units Shipped, Units Ordered, and Net Weight Shipped, now map to their corresponding columns in three tables:
Fact_D1_ORDERS2
Fact_D1_ORDER_AGG1
Fact_D1_ORDER_AGG2

5. Specify the aggregation content of the new logical table source so that Oracle BI Server knows what level of data is stored in the aggregate tables.
   a. Click the **Content** tab.
   b. Specify the aggregation content for Fact_D1_ORDER_AGG2.



c. Compare the aggregation content for Fact_D1_ORDER_AGG2 with that of the Fact_D1_ORDER_AGG1 logical table source.

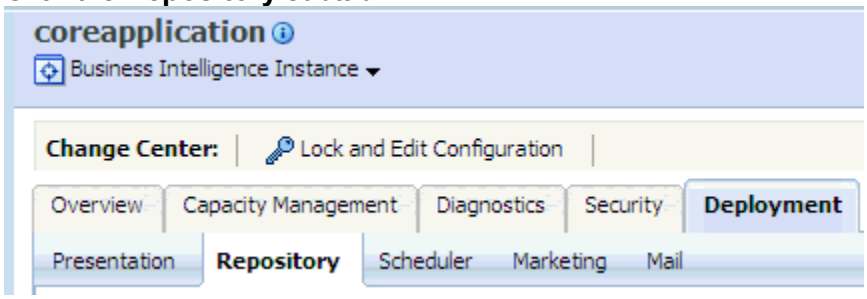


Notice that both aggregate logical table sources have aggregation content defined for the same logical dimensions, Product and Time, but at different levels.

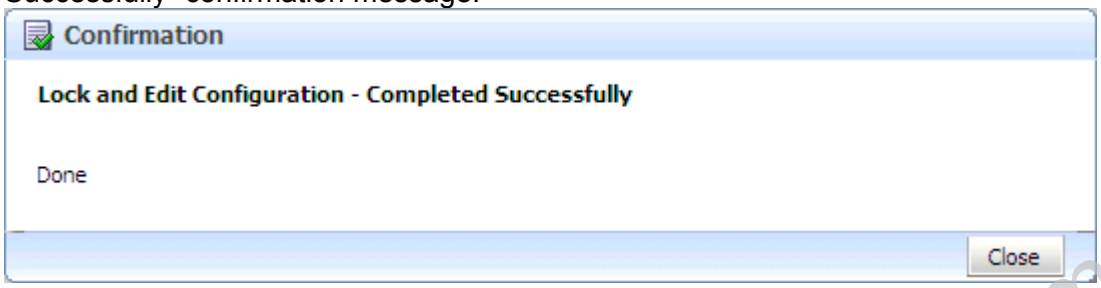
d. Click **OK** to close the Logical Table Source dialog box.

6. In this step you set the number of elements in the logical dimensions. Aggregate fact sources are selected based on a combination of the fields selected as well as the number of elements of the levels in the logical dimensions to which they map. This number is used by the Oracle BI Server when picking aggregate sources. Setting the number of elements is only necessary when there are two or more aggregate sources that could be accessed by an Oracle BI query. The number does not have to be exact, but ratios of numbers from one logical level to another should be accurate.
   a. Expand the **Product** logical dimension.
   b. Double-click the **Product Total** level and select the **General** tab. Because this is the grand total level, the number of elements is automatically set to one.
   c. Click **Cancel**.
   d. Double-click the **Type** level and select the **General** tab.
   e. Set the number of elements at this level to **22**. One way to determine the number of elements at each level is by updating row counts and then observing the row count number for the corresponding column in the Physical layer.
   f. Continue to set the number of elements for the levels in the **Product** logical dimension:
      SubType: 159
      Generic: 186
      Product Detail: 192
   g. Set the number of elements for the levels in the **Time** logical dimension:
      Year: 2
      Quarter: 6
      Month: 16
      Time Detail: 474
   h. Save the repository
   i. Check consistency. Fix any errors or warnings before proceeding.
   j. Close the repository. Leave the Admin Tool open.
7. Use Fusion Middleware Control Enterprise Manager to upload the repository.
   a. Return to Fusion Middleware Control Enterprise Manager, which should still be open in your browser. If it is not open, enter the following URL:
      http://localhost:7001/em.

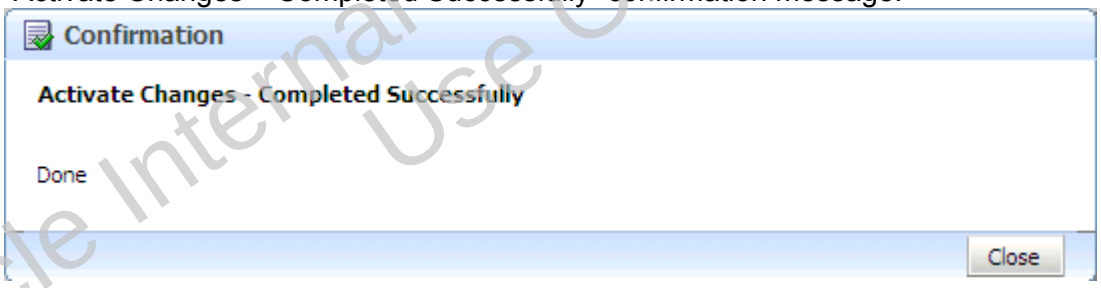
b. If your session has timed out, log in as **weblogic/welcome1**.
c. In the left pane, expand **Business Intelligence** and select **coreapplication**.
d. In the right pane, click the **Deployment** tab.
e. Click the **Repository** subtab.



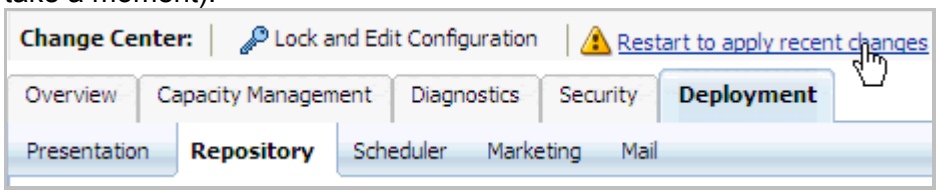
f. Click **Lock and Edit Configuration**.
g. Click **Close** when you receive the "Lock and Edit configuration – Completed Successfully" confirmation message.



h. In the Upload BI Server Repository section, click **Browse** to open the "Choose file" dialog box.
i. By default, the "Choose file" dialog box should open to the default repository directory. If not, browse to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.
j. Select **ABC.rpd** and click **Open**. You can also double-click the repository to open it.
k. Enter **welcome1** in the Repository Password and Confirm Password fields.
l. Click **Apply**. Notice that Default RPD now displays **ABC** with an extension (for example, ABC_BI0007).
m. Click **Activate Changes**.
n. Allow Active Changes processing to complete. Click **Close** when you receive the "Activate Changes – Completed Successfully" confirmation message.

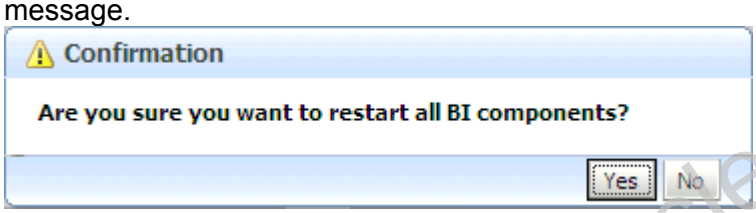


o. Click **Restart to apply recent changes** to navigate to the Overview page (this may take a moment).

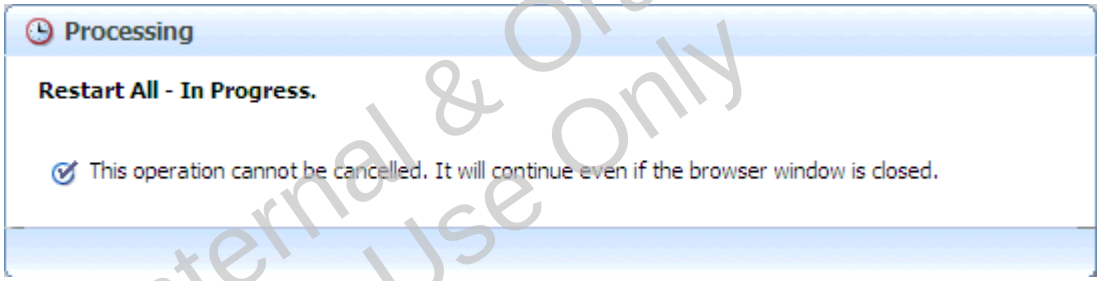


p. On the Overview page, click **Restart**.

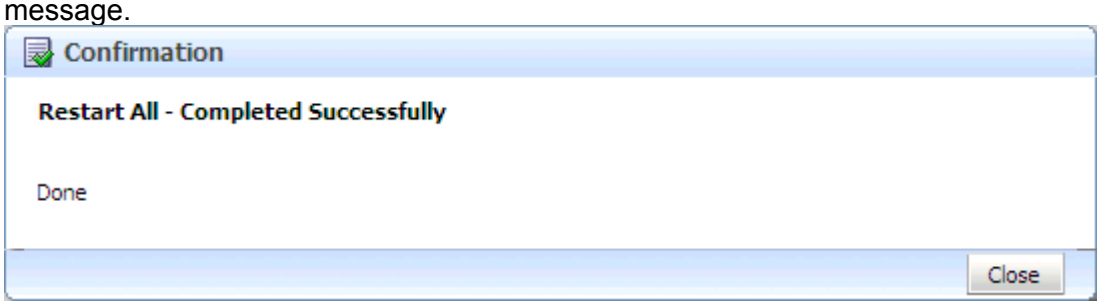
q. Click **Yes** when you receive the "Are you sure you want to restart all BI components?" message.



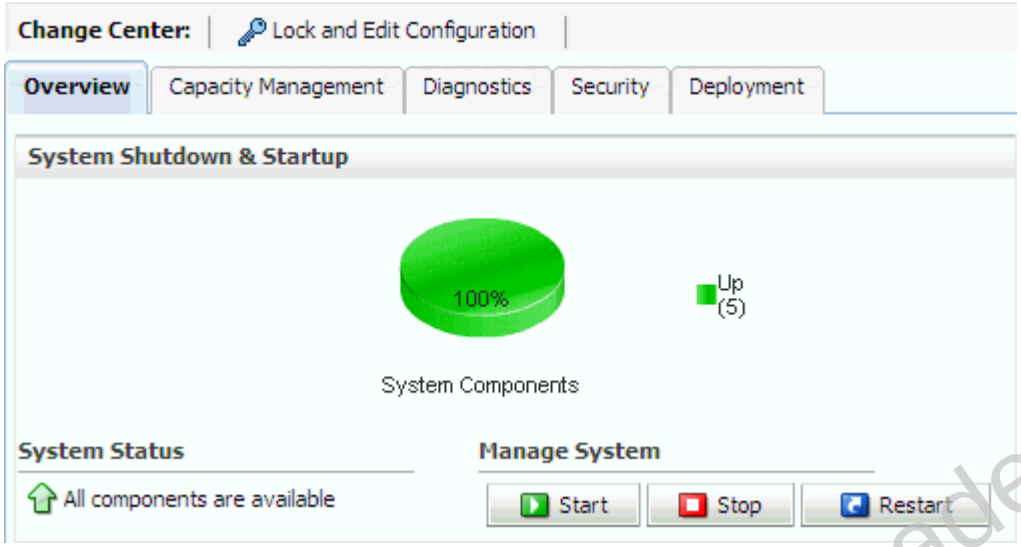
r. Allow the Restart All – In Progress processing to complete. This may take a few moments.



s. Click **Close** when you receive the “Restart All – Completed Successfully” confirmation message.

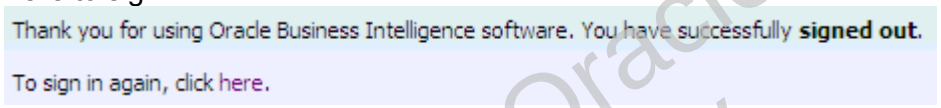


t. Confirm that System Components = 100%. The ABC repository is loaded into BI Server.

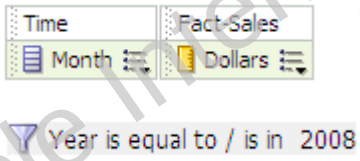


u. Leave Enterprise Manager open.

8. Open Analysis Editor to execute queries and test the SupplierSales business model.
   a. Return to the Oracle Business Intelligence browser tab where you signed out and click **here** to sign in.



   b. Sign in as **weblogic** with password **welcome1**.
   c. In the Create section, click **Analysis** to open the Select Subject Area window.
   d. Click **SupplierSales** to open Analysis Editor.
9. Create and run analyses to check your work.
   a. Create the following analysis and filter.



b. Click **Results**

Month	Dollars
January	$3,595,669
February	$3,945,187
March	$3,975,774
April	$3,907,292
May	$4,061,558
June	$3,994,531
July	$4,062,212
August	$4,242,611
September	$3,810,263
October	$4,596,372
November	$3,655,169
December	$3,997,616

c. Sign out of Oracle BI.

d. Inspect the query log.

```
---------- Sending query to database named orcl (id:
<<2579>>), connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T744.DOLLARS) as c1,
     T862.MONTHNAME as c2,
     T862.MONTHCODE as c3
from
     MONTHS T862 /* Dim_MONTHS */ ,
     D1_ORDER_AGG1 T744 /* Fact_D1_ORDER_AGG1 */
where  ( T744.PERKEY = T862.MONTHCODE and T862.YEAR = 2008 )
group by T862.MONTHCODE, T862.MONTHNAME)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c3 as c3,
     D1.c1 as c4
from
     SAWITH0 D1
order by c3
```

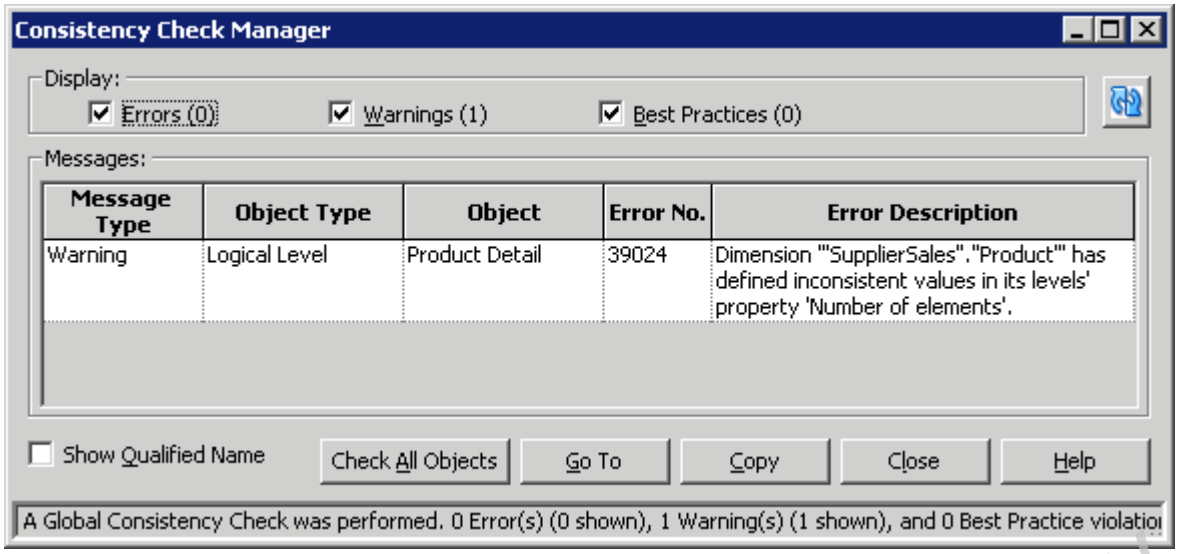
Notice that despite the fact that both aggregate logical table sources have their aggregation content defined for the same logical dimensions, Product and Time, the query uses Fact_D1_ORDER_AGG1. This is because Oracle BI Server determined it was more economical to access Fact_D1_ORDER_AGG1 instead of Fact_D1_ORDER_AGG2 based on the levels. For example, to access Fact_D1_ORDER_AGG2, it calculates 3072 potential rows: 16 months * 192 products. To access Fact_D1_ORDER_AGG1, it calculates 352 potential rows: 16 months * 22 product types. Therefore, Fact_D1_ORDER_AGG1 is the more economical aggregate source.

10. Adjust the number of elements to alter the aggregate fact source selected by Oracle BI Server.
    a. Return to the Administration Tool and open the repository in **online** mode.
    b. Enter **welcome1** as the repository password and user password and click **Open**.
    c. In the Business Model and Mapping layer, expand the **Product** logical dimension.
    d. Double-click the **Type** level.
    e. Click **Check Out**.
    f. Click the **General** tab and change the number of elements at this level from **22** to **220**.
    g. Click **OK** to close the Logical Level dialog box. If you run the same analysis (Month, Dollars), changing the number of elements at this level increases the potential rows for Fact_D1_ORDER_AGG1from 352 to 3520 (16 months * 220 product types), which is

now higher than the potential rows for Fact_D1_ORDER_AGG2, which is 3072. Thus the same query will access Fact_D1_ORDER_AGG2 instead of Fact_D1_ORDER_AGG1. Please notice that you are doing this only to demonstrate a teaching point. This is not a correct modeling technique.

h. Check in the changes.

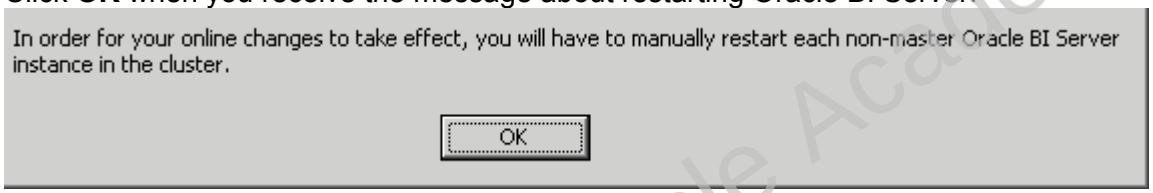
i. Check consistency. For the purpose of this teaching point you can ignore the warning about inconsistent values and click **Close**.



j. Save the repository.

k. Close the repository.

l. Click **OK** when you receive the message about restarting Oracle BI Server.



m. Leave the Administration Tool open.

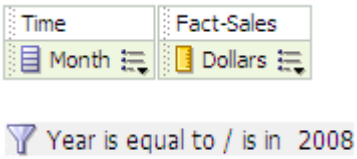
11. Create an analysis to check your work.

a. Return to Oracle BI and sign in as **weblogic/welcome1**.

b. Click **Analysis > SupplierSales**.

c. Click **Reload Server Metadata**.

d. Create the same analysis and filter:



e. Click **Results**.

Month	Dollars
January	$3,595,669
February	$3,945,187
March	$3,975,774
April	$3,907,292
May	$4,061,558
June	$3,994,531
July	$4,062,212
August	$4,242,611
September	$3,810,263
October	$4,596,372
November	$3,655,169
December	$3,997,616

f. Sign out of Oracle BI.

g. Inspect the query log.

```
---------- Sending query to database named orcl (id:
<<11914>>), connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T889.DOLLARS) as c1,
     T862.MONTHNAME as c2,
     T862.MONTHCODE as c3
from
     MONTHS T862 /* Dim_MONTHS */ ,
     D1_ORDER_AGG2 T889 /* Fact_D1_ORDER_AGG2 */
where  ( T862.MONTHCODE = T889.PERKEY and T862.YEAR = 2008 )
group by T862.MONTHCODE, T862.MONTHNAME)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c3 as c3,
     D1.c1 as c4
from
     SAWITH0 D1
order by c3
```

Notice that the query now uses Fact_D1_ORDER_AGG2 instead of Fact_D1_ORDER_AGG1. This is because Oracle BI Server determined it was more economical to access Fact_D1_ORDER_AGG2 instead of Fact_D1_ORDER_AGG1 based on the levels. To access Fact_D1_ORDER_AGG2, it calculates 3072 potential rows: 16 months * 192 products. To access Fact_D1_ORDER_AGG1, it calculates 3520 potential rows: 16 months * 220 product types. Therefore, Fact_D1_ORDER_AGG2 is now the more economical aggregate source. The important point is that setting the number of elements in the logical dimension hierarchies helps Oracle BI Server determine the most economical source to access when there are multiple aggregate sources.

Practice 10-3: Using the Aggregate Persistence Wizard

Goal

To use the Aggregate Persistence Wizard to automate the creation of aggregate tables and their corresponding objects in the repository

Scenario

The traditional process for creating aggregates for Oracle BI Server queries is manual, requiring the writing of DDL and DML scripts to create tables in the databases involved. Additionally, these aggregated tables must be mapped into the repository metadata to be available for queries. This is a time consuming and, possibly, an error prone process. The Aggregate Persistence Wizard allows an administrator to automate the creation of aggregate tables and their corresponding objects in the repository. Recall that your repository contains an aggregate table called D1_ORDER_AGG1, which contains sales fact data aggregated at the month and product type levels, and corresponding dimension aggregate tables for product type and months. In this practice, you use the Aggregate Persistence Wizard to create similar aggregate tables and the corresponding metadata.

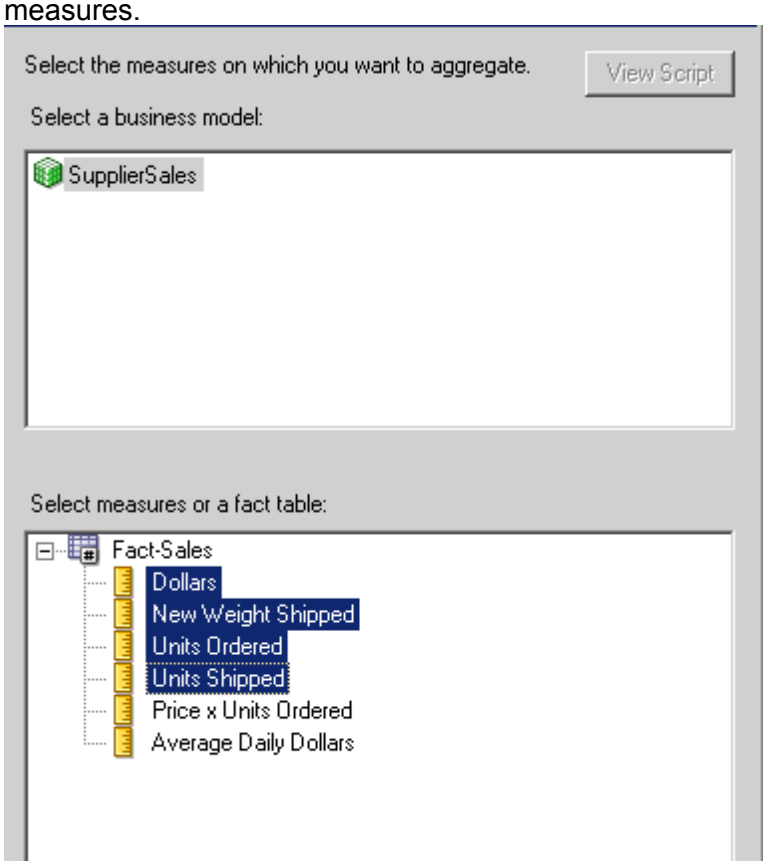
Time

30 minutes

Tasks

1. In this step, you use the Aggregate Persistence Wizard to build a script that is used to create aggregate tables and the corresponding repository metadata.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. Select **Tools > Utilities > Aggregate Persistence** and click **Execute**.
   c. In the Select File Location screen, in the File Name field, enter **CREATE_AGG**. This specifies the file where the output script is saved. This file stores the aggregate specifications and is updated if more aggregates are specified.
   d. Click **Browse**, navigate to **D:\PracticeFiles** and click **OK**.
   e. Click **Next**.
   f. In the top pane, select the **SupplierSales** business model. When there are multiple business models only one business model can be selected. In this example, there is only one.
   g. In the bottom pane expand the **Fact-Sales** fact table. When there are multiple fact tables, only one fact table can be selected.

h. Select the **Dollars**, **Units Shipped**, **Units Ordered**, and **Net Weight Shipped** measures.



i. Click **Next**.

j. Select the following levels. Leave the Use Surrogate Key? field check box deselected.
Time: Month
Product: Type

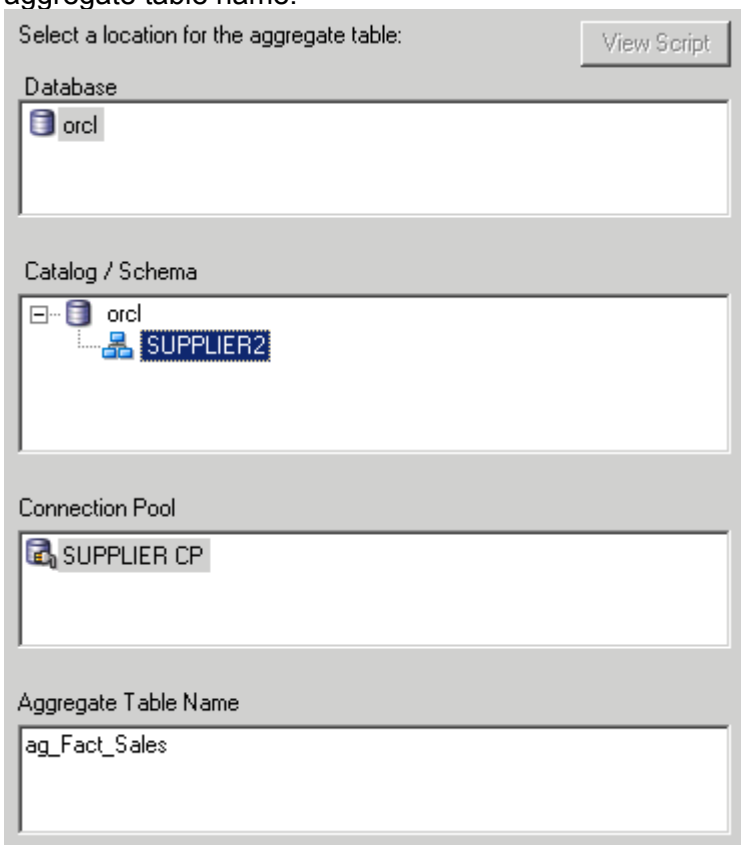
Logical Dimension	Logical Level	Use Surrogate Key?
Time	Month	☐
Product	Type	☐
Employee		☐
Customer		☐

k. Click **Next**.

l. In the top pane, select the **orcl** database.

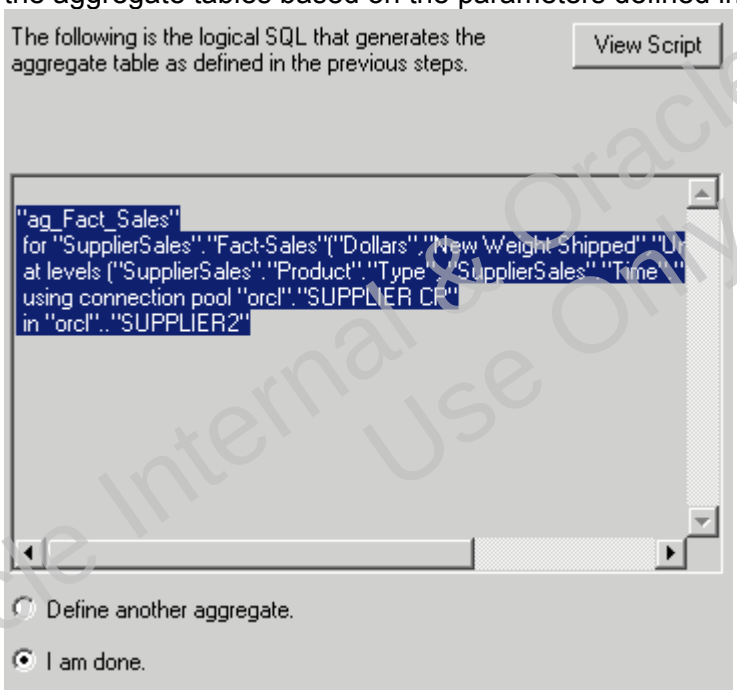
m. In the second pane, expand **orcl** and select the **SUPPLIER2** schema. The **SUPPLIERCP** connection pool is selected by default.

n. In the **Aggregate table name** field, accept the default name **ag_SalesFacts** for the aggregate table name.



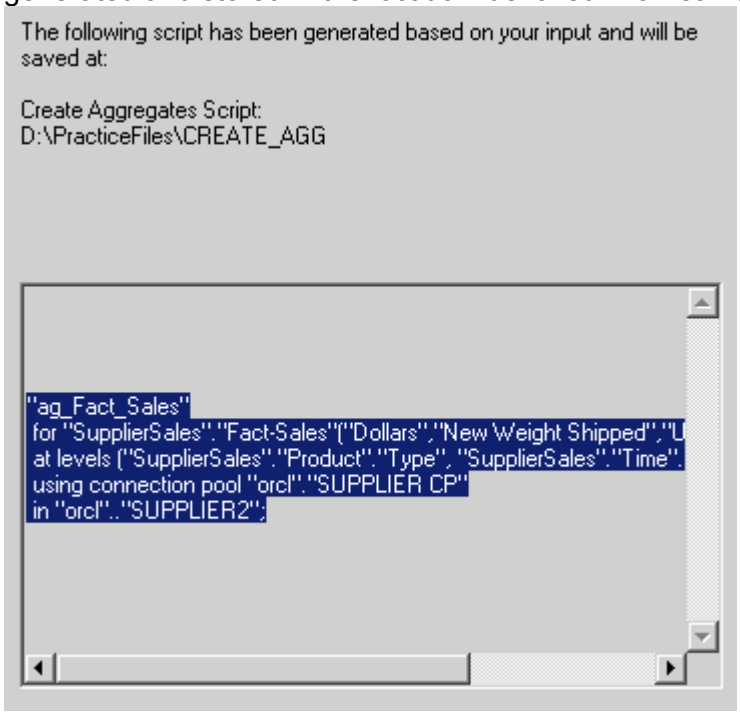
o. Click **Next**.

p. Review the aggregate definition. The screen displays the logical SQL that generates the aggregate tables based on the parameters defined in the previous steps.



q. Select **I am done**.

r. Click **Next**. The Finish Script dialog box appears confirming that the script has been generated and stored in the location identified in an earlier step.



s. Click **Finish**.

t. Save the repository.

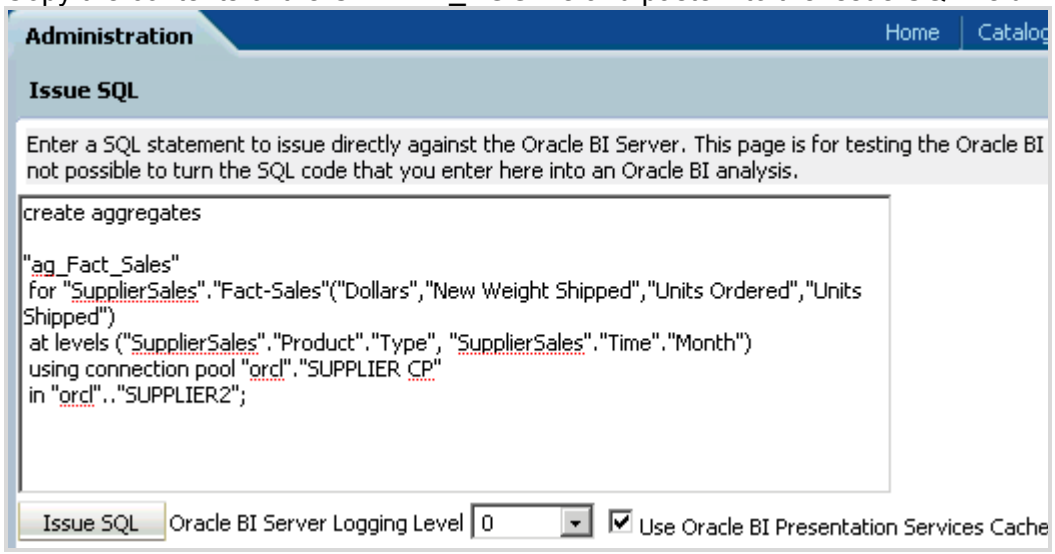
u. Check consistency. Fix any warnings or errors before proceeding.

v. Close the repository. Leave the Administration Tool open.

w. Navigate to **D:\PracticeFiles** and confirm that the **CREATE_AGG** script file was generated.

2. Examine the aggregate parameters in the NQSConfig.ini file.
   a. Navigate to **C:\OracleBIHome\instances\instance1\config\OracleBIServerComponent\coreapplication_obis1**and open **NQSConfig.ini**.
   b. In the Aggregate Persistence section, notice the setting for **AGGREGATE_PREFIX**. This is the prefix that is added automatically to the names of the dimension aggregates when they are generated by the script.
   c. Close **NQSConfig.ini** without making any changes.
3. Use Issue SQL to run the script.
   a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier in this set of practices.
   b. Sign in to Oracle BI as **weblogic** with password **welcome1**.
   c. Click **Administration**.
   d. In the Maintenance and Troubleshooting section, click **Issue SQL**.
   e. Navigate to **D:\PracticeFiles** and open the **CREATE_AGG** file in Notepad.

f. Copy the contents of the CREATE_AGG file and paste into the Issue SQL field.

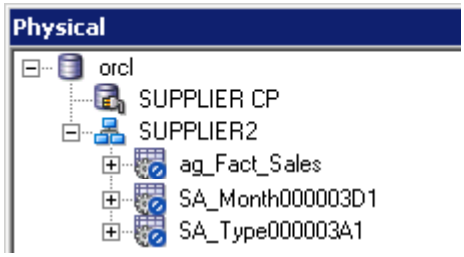


g. Click **Issue SQL** and wait for loading to complete.

4. Verify that the aggregates are created in the Physical layer of the repository.

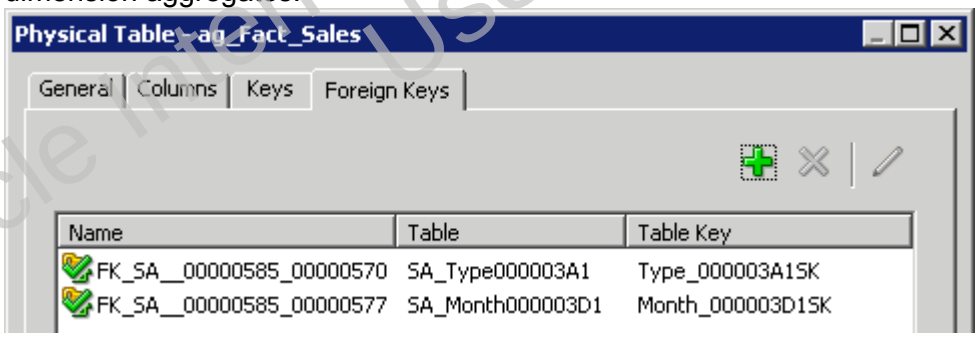
a. Return to the Administration Tool and select **File > Open > Online** to open the ABC repository in online mode. Use **welcome1** as the repository and user name password.

b. In the Physical layer, ensure that the aggregates were created in the SUPPLIER2 schema. There should be one new **ag_Fact_Sales** aggregate and two new dimension aggregates beginning with **SA_** in the Physical layer. Your results should resemble the screenshot.



c. Update row counts for the new aggregates and confirm that you see the following row counts. Check out objects when prompted.
ag_Fact_Sales: 324 rows
SA_Month*: 16 rows
SA_Type*: 21 rows

d. Double-click **ag_Fact_Sales** to open the Physical Table dialog box, click the **Foreign Keys** tab, and verify that joins are created between ag_Fact_Sales and the two new dimension aggregates.



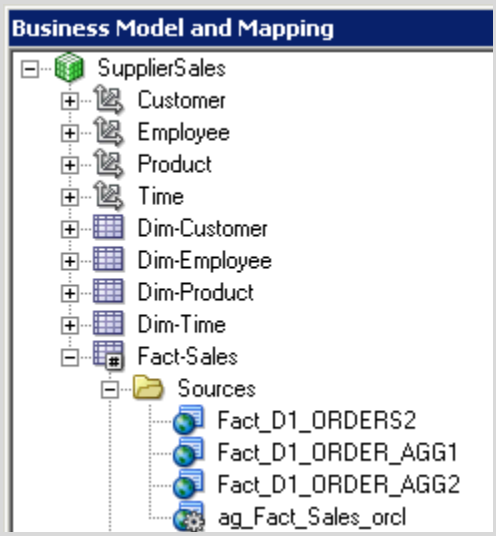
e. Double-click the **foreign keys** to view the join relationships in the Physical Foreign Key dialog box.

f. Click **Cancel** to close the **Physical Foreign Key** dialog box.

g. Click **Cancel** to close the **Physical Table** dialog box.

5. Verify that the aggregates are created in the Business Model and Mapping layer of the repository.

a. In the Business Model and Mapping layer, open the **Sources** folders for the Fact-Sales, Dim-Product, and Dim-Time logical tables and confirm that new logical table sources are created for the aggregates. Fact-Sales > Sources is shown in the following screenshot:



b. Double-click the **ag_Fact_Sales_orcl** logical table source.

c. Click **Check Out**.

d. Click the **General** tab and confirm that the ag_Fact_Sales_orcl logical table source maps to the ag_Fact_Sales physical table.

e. Click the **Column Mapping** tab and confirm that the Dollars, Units Ordered, Units Shipped, and Net Weight Shipped logical columns map to the corresponding physical columns in the ag_Fact_Sales physical table.

Logical Column	Expression		Physical Table
Dollars	Dollars0000027A		ag_Fact_Sales
New Weight Shippe	New_Weight000		ag_Fact_Sales
Units Ordered	Units_Orde00000		ag_Fact_Sales
Units Shipped	Units_Ship00000		ag_Fact_Sales
Price x Units			
Average Daily Dollar			

f. Click the **Content** tab and confirm that the logical levels are set correctly.

Logical Dimension	Logical Level	
Time	Month	
Product	Type	
Customer		
Employee		

g. Click **Cancel** to close the Logical Table Source dialog box.

h. Check in the changes.

i. Check global consistency. Fix any errors or warnings before proceeding.

j. Save the repository.

k. Close the repository.

6. Verify that the aggregates are created in the database.

   a. Select **Start > Programs > Oracle – OraDB11g_home1 > Application Development > Sql*Plus**.

   b. Enter **supplier2** as the username.

   c. Enter **supplier2** as the password.

   d. Run the following SQL to verify that the aggregate tables were created in the database:

```
select table_name from user_tables where table_name like 'SA_%'
or table_name like 'AG_%';
```

   e. Verify that three tables are returned.

```
SQL> select table_name from user_tables where table_name like 'SA_%'
 or table_name like 'AG_%';

TABLE_NAME
------------------------------
AG_FACT_SALES
SA_MONTH000003D1
SA_TYPE000003A1
```

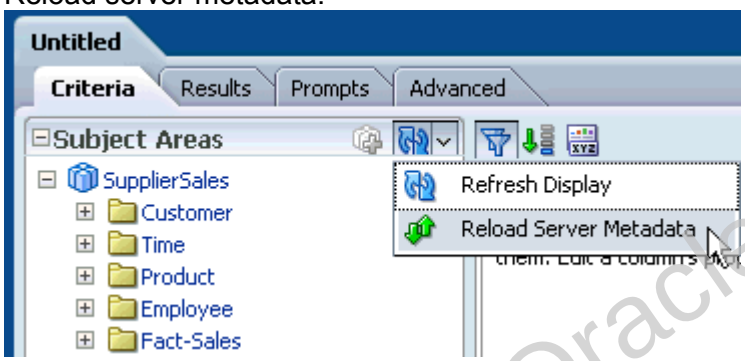
   f. Exit SQL*Plus.

7. Use Analysis Editor to test your work.

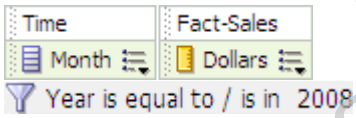
   a. Sign in to Oracle BI as **weblogic** with password **welcome1**.

   b. Return to Analysis Editor and create a new analysis in the SupplierSales subject area.

   c. Reload server metadata.



   d. Create and run the following analysis and filter:



e. Click **Results**

Month	Dollars
January	$7,897,788
February	$7,910,307
March	$8,296,803
April	$6,607,225
May	$4,061,558
June	$3,994,531
July	$4,062,212
August	$4,242,611
September	$3,810,263
October	$4,596,372
November	$3,655,169
December	$3,997,616

f. Sign out of Oracle BI.

g. Inspect the query log and confirm that the query used the **ag_Fact_Sales** aggregate table and the related **SA_Month*** dimension aggregate. Oracle BI Server uses the most economical sources to satisfy the query. Your log should resemble the following screenshot.

```
SAWITH1 AS (select sum(T925.Dollars0000027A) as c1,
    T911.Month000001DD as c2
from
    SA_Month000003D1 T911,
    ag_Fact_Sales T925
where  ( T911.Year000001E3 = 2008 and T911.Month_000003D1SK = T925.Month_000003D1SK )
group by T911.Month000001DD),
```

Practices for Lesson 11: Using Partitions and Fragments

Lesson 11

Practices for Lesson 11

Lesson Overview

In these practices, you will model partitions in an Oracle BI repository.

Practice 11-1: Modeling a Value-Based Partition

Goal

To model a value-based partition

Scenario

ABC wants to store its fact data in two separate partitions, one for recent data and one for historical data. Each partition contains the same columns. Only the data values are different. The historical data partition stores invoice fact data up to and including December 31, 2008. The recent data partition stores invoice fact data after December 31, 2008.

Outcome

In the Physical layer, there are two new physical sources: D1_ORDERS_RECENT and the D1_ORDERS_HISTORICAL. In the Business Model and Mapping layer, logical table sources are modified for the Fact-Sales logical table.

Time

30 minutes

Tasks

1. Import two value-based partitioned sources with fact data into the Physical layer.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. In the Physical layer, expand **orcl**.
   c. Right-click the **SUPPLIER CP** connection pool and select **Import Metadata**.
   d. Accept the defaults in the Select Metadata Types window and click **Next** to open the Select Metadata Objects window.
   e. Expand the **SUPPLIER2** schema.
   f. In the Data source view pane, select the **D1_ORDERS_RECENT** and the **D1_ORDERS_HISTORICAL** tables for import.
   g. Click the **Import selected** button to add the tables to the Repository View pane.
   h. Expand **SUPPLIER2** in the Repository View pane and confirm that the D1_ORDERS_RECENT and the D1_ORDERS_HISTORICAL tables are added.
   i. Click **Finish** to add the tables to the repository.
   j. Expand **SUPPLIER2** in the Physical Layer and confirm that the D1_ORDERS_RECENT and the D1_ORDERS_HISTORICAL tables are added to the repository. D1_ORDERS_HISTORICAL has fact data for all of 2008. D1_ORDERS_RECENT has fact data for the first four months of 2009.
   k. Update row counts for the two new tables to ensure connectivity.
      D1_ORDERS_RECENT: 80223 rows
      D1_ORDERS_HISTORICAL: 271413 rows
   l. Double-click **PERIODKEY** in D1_ORDERS_RECENT to open the Physical column dialog box.
   m. Change the Type to **INT**.
   n. Repeat for **PERIODKEY** in D1_ORDERS_HISTORICAL**.**

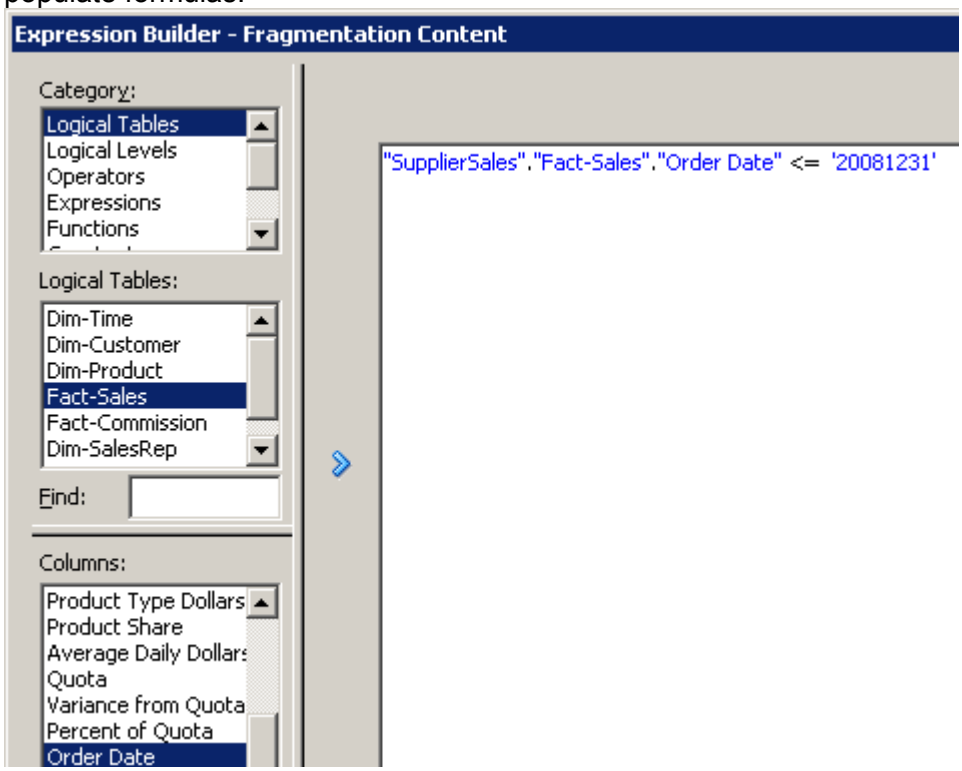
o. Create the following aliases:
Fact_D1_ORDERS_RECENT
Fact_D1_ORDERS_HISTORICAL

p. Use the Physical Diagram to create the following physical joins:

```
Dim_D1_CALENDAR2.YYYYMMDD = Fact_D1_ORDERS_RECENT.PERIODKEY
Dim_D1_PRODUCTS.PRODUCTKEY = Fact_D1_ORDERS_RECENT.PRODKEY
Dim_D1_CUSTOMER2.NEWKEY = Fact_D1_ORDERS_RECENT.CUSTKEY

Dim_D1_CALENDAR2.YYYYMMDD = Fact_D1_ORDERS_HISTORICAL.PERIODKEY
Dim_D1_PRODUCTS.PRODUCTKEY = Fact_D1_ORDERS_HISTORICAL.PRODKEY
Dim_D1_CUSTOMER2.NEWKEY = Fact_D1_ORDERS_HISTORICAL.CUSTKEY
```

2. Create two new logical table sources for the Fact-Sales logical table.
   a. Expand **SupplierSales > Fact-Sales** in the Business Model and Mapping layer.
   b. Expand **Fact_D1_ORDERS_RECENT** in the Physical layer.
   c. Drag the **DOLLARS** column from Fact_D1_ORDERS_RECENT to the **Dollars** column in Fact-Sales. This creates a new logical table source named Fact_D1_ORDERS_RECENT and maps the column.
   d. Repeat the steps for the **DOLLARS** column in **Fact_D1_ORDERS_HISTORICAL**. This creates a new logical table source named Fact_D1_ORDERS_HISTORICAL and maps the column.
3. Create a new logical column.
   a. Right-click **Fact-Sales** and select **New Object > Logical Column**.
   b. Name the column **Order Date**.
   c. Drag the **PERIODKEY** column from **Fact_D1_ORDERS_RECENT** to the **Order Date** column in **Fact-Sales**.
   d. Drag the **PERIODKEY** column from **Fact_D1_ORDERS_HISTORICAL** to the **Order Date** column in **Fact-Sales**.
   e. Drag **Order Date** to the **Fact-Sales** presentation table.
4. Create fragmentation content for the two new logical table sources.
   a. Double-click **Dim_Fact_D1_ORDERS_HISTORICAL** in the Sources folder and click the **General** tab.
   b. Rename the logical table source to **Facts-Historical**.
   c. Click the **Content** tab.
   d. Click the **formula** button next to the Fragmentation content field to open the Fragmentation Content expression builder.
   e. Select **Logical Tables > Fact-Sales**, and then double-click the **Order Date** column. The expression is added to the expression builder.
   f. Click "**<=**" on the toolbar to add it to the formula and then add **'20081231'** to the formula. You are hard-coding this formula so that this logical table source will be used for queries where Order Date (PERIODKEY) is equal to or less than December 31, 2008 (20081231). In the next lesson you learn how to use variables to dynamically

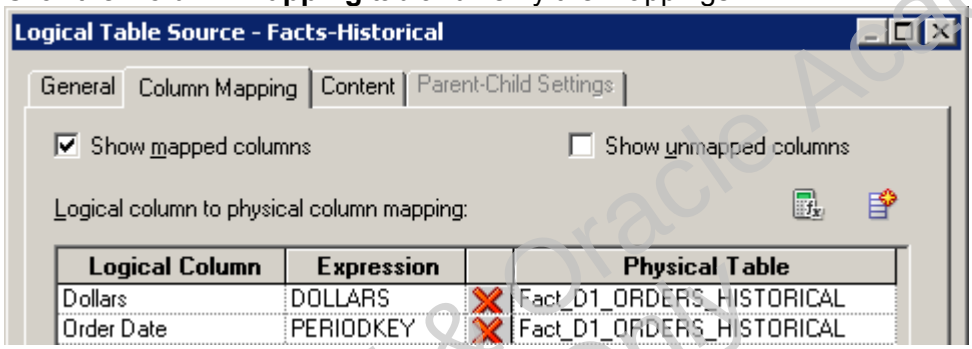
populate formulas.



g. Click **OK** to close the Expression Builder. Notice that the expression is added to the Fragmentation content field.

h. Select **This source should be combined with others at this level**.

i. Click the **Column Mapping** tab and verify the mappings.



j. Click **OK** to close the Logical Table Source dialog box.

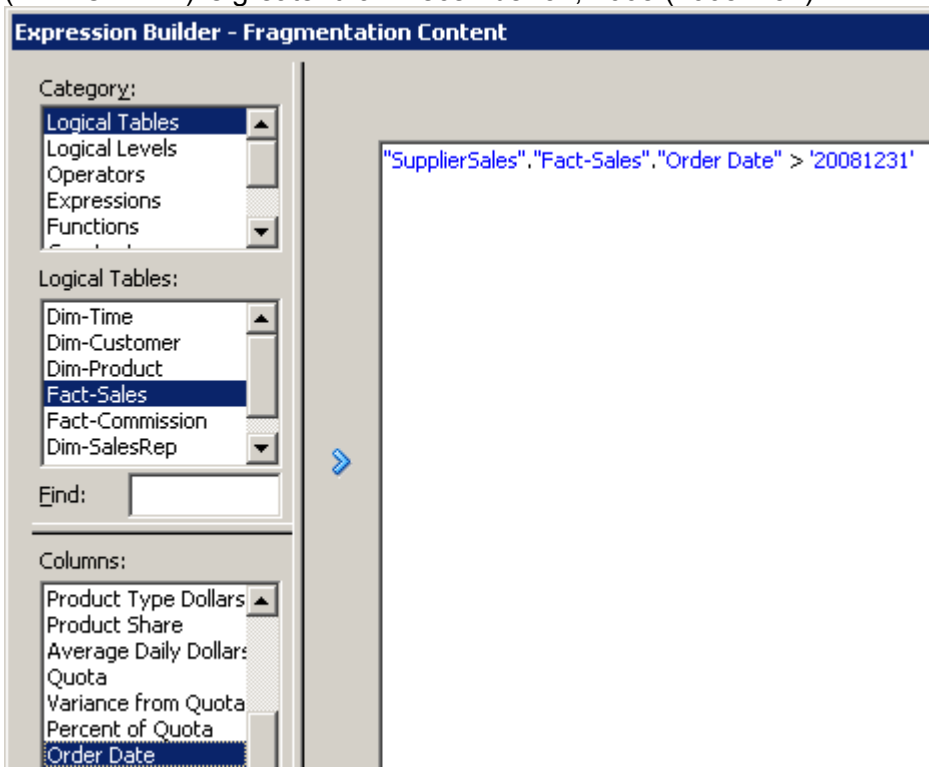
k. Double-click **Fact_D1_ORDERS_RECENT** in the Sources folder and click the **General** tab.

l. Rename the logical table source to **Facts-Recent**.

m. Click the **Content** tab.

n. Open the Fragmentation Content expression builder and create the following expression:
SupplierSales.”Fact-Sales”.”Order Date” > ‘20081231’. You are hard-coding this formula so that this logical table source will be used for queries where Order Date

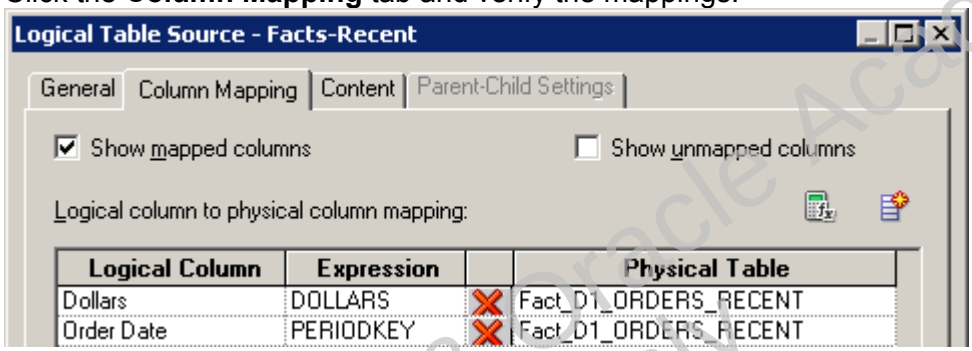
(PERIODKEY) is greater than December 31, 2008 (20081231).



o. Click **OK**.

p. Select **This source should be combined with others at this level**.

q. Click the **Column Mapping** tab and verify the mappings.



r. Click **OK**.

s. Notice that, other than adding the Order Date column to the Fact-Sales presentation table, you did not have to change the Presentation layer. The Order Date column is added for testing purposes.

t. Save the repository.

u. Check consistency. Fix any errors or warnings before you proceed.

v. Close the repository.

5. Create and run analyses to test your work.

a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.

b. Return to Analysis Editor.

c. Create an analysis.

d. Click **Results**.

Order Date	Dollars
20080102	$26,036
20080103	$12,210
20080105	$140,140
20080106	$85,079
20080107	$399,278
20080108	$151,171
20080109	$107,925
20080110	$110,789

Notice that Order Date records are returned for both before and after 20081231. Click the **Display Maximum** button at the bottom of the table to confirm.

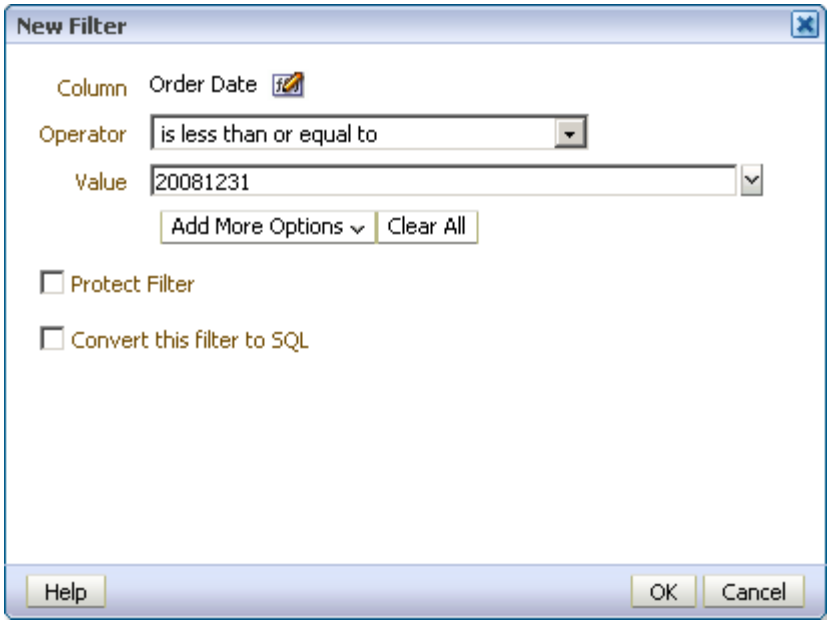
e. Leave Analysis Editor open.

f. Open the query log. Confirm that the query references both partition tables. Both partition tables are used because you did not apply a filter in the query. Notice that a separate select statement is sent to each table and Oracle BI Server performs a UNION ALL to combine the results.

```
---------- Sending query to database named orcl (id:
<<1693>>), connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS ((select T928.PERIODKEY as c2,
     T928.DOLLARS as c3
from
     D1_ORDERS_RECENT T928 /* Fact_D1_ORDERS_RECENT */
union all
select T919.PERIODKEY as c2,
     T919.DOLLARS as c3
from
     D1_ORDERS_HISTORICAL T919 /* Fact_D1_ORDERS_HISTORICAL */
)),
SAWITH1 AS (select sum(D3.c3) as c1,
     D3.c2 as c2
from
     SAWITH0 D3
group by D3.c2)
select distinct 0 as c1,
     D2.c2 as c2,
     D2.c1 as c3
from
     SAWITH1 D2
order by c2
```

Partitioning can create complexity in the query environment. Queries must access multiple tables and then consolidate the results. One of the important benefits of Oracle BI Server is that it can do this type of navigation and consolidation automatically and invisibly to the end user.

g. Return to Analysis Editor and add a filter where **Order Date** is less than or equal to **20081231**.



h. Run the analysis again.

i. Verify that your results display only Order Dates that are less than or equal to 20081231.

Order Date	Dollars
20080102	$26,036
20080103	$12,210
20080105	$140,140
20080106	$85,079
20080107	$399,278
20080108	$151,171
20080109	$107,925
20080110	$110,789

j. Leave Analysis Editor open.

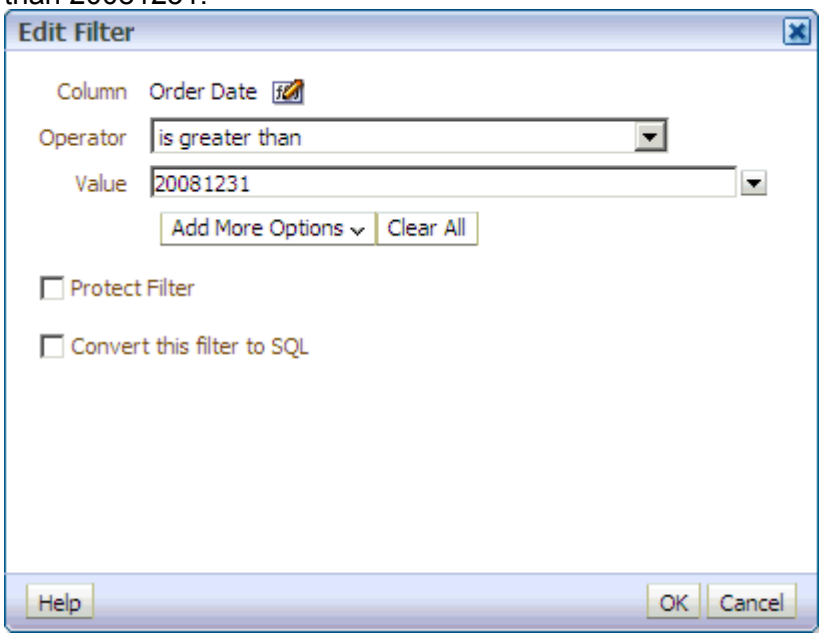
k. Open the query log and confirm that the query accessed only one partition, Fact_D1_ORDERS_HISTORICAL.

```
---------- Sending query to database named orcl (id:
<<2669>>), connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T919.DOLLARS) as c1,
     T919.PERIODKEY as c2
from
     D1_ORDERS_HISTORICAL T919 /* Fact_D1_ORDERS_HISTORICAL */
where  ( T919.PERIODKEY <= 20081231 )
group by T919.PERIODKEY)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c1 as c3
from
     SAWITH0 D1
order by c2 NULLS FIRST
```

l. Close the log file.

6. Test that your results are correct when Order Date is greater than 20081231.

a. Return to the analysis and edit the filter to show records where Order Date is greater than 20081231.



b. Run the analysis.

c. Verify that your result only displays records where Order Date is greater than 20081231.

Order Date	Dollars
20090102	$143,693
20090104	$324,001
20090105	$172,597
20090106	$112,132
20090107	$170,791
20090108	$171,649
20090109	$119,321
20090111	$302,795
20090112	$136,086

d. Sign out of Oracle BI.

e. Open the query log and confirm that the query being issued accesses the correct partition, Fact_D1_ORDERS_RECENT.

```
---------- Sending query to database named orcl (id:
<<2930>>), connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T928.DOLLARS) as c1,
     T928.PERIODKEY as c2
from
     D1_ORDERS_RECENT T928 /* Fact_D1_ORDERS_RECENT */
where  ( 20081231 < T928.PERIODKEY )
group by T928.PERIODKEY)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c1 as c3
from
     SAWITH0 D1
order by c2 NULLS FIRST
```

Practice 11-2: Modeling a Fact-Based Partition

Goal

To incorporate quotas in the business model and create new business measures using a fact-based partition

Scenario

ABC sets sales quotas for its sales organization. These quotas are set at the region level by quarter. In addition, each of the region sales quotas is broken down by product type. Quota numbers are stored in an Excel workbook. The workbook, quota.xls, is stored on your machine. You incorporate the quota numbers in the business model and create business measures to report variance from quota and percent of quota.

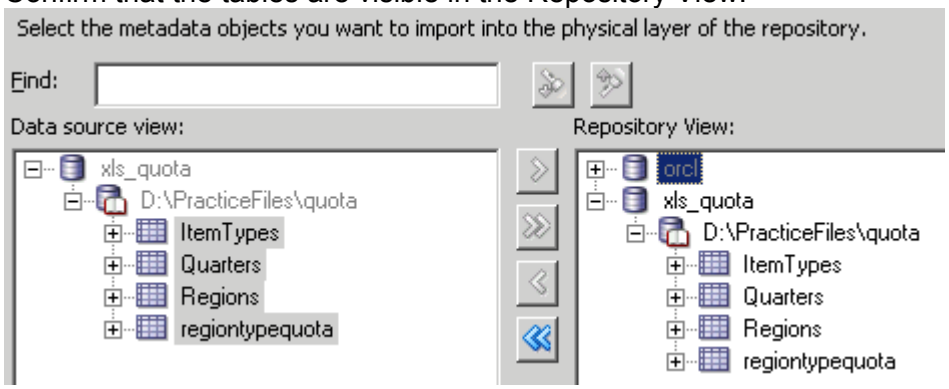
Time

20 minutes

Tasks

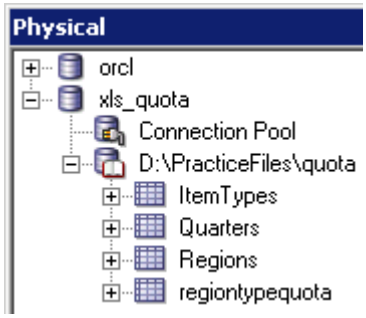
1. Create an ODBC data source for an Excel spreadsheet.
   a. Select **Start > Programs > Administrative Tools > Data Sources (ODBC)** to open the ODBC Data Source Administrator.
   b. Click the **System DSN** tab.
   c. Click **Add**.
   d. Select **Microsoft Excel Driver(*.xls)** and click **Finish**.
   e. Enter **xls_quota** as the data source name.
   f. Click **Select Workbook**.
   g. Navigate to **D:\PracticeFiles** and select the **quota.xls** workbook.
   h. Click **OK**.
   i. Click **OK** to close the ODBC Microsoft Excel Setup dialog box.
   j. Click **OK** to close the ODBC Data Source Administrator.
2. Import the xls_quota data source into the ABC repository.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. Select **File > Import Metadata**.
   c. Verify that Connection Type is set to **ODBC 3.5**.
   d. Select the **xls_quota** data source, and enter **Administrator** as the username. No password is needed.
   e. Click **Next**.
   f. Accept the defaults in the Select Metadata Types window and click **Next**.
   g. Expand **D:\PracticeFiles\quota**.
   h. Select the following tables:
      Item Types
      Quarters
      Regions
      regiontypequota
   i. Click **Import Selected** to add the tables to the Repository View.

j. Confirm that the tables are visible in the Repository View.



k. Click **Finish**.

l. Confirm that the xls_quota data source is added to the Physical layer with the expected tables.



m. Here is a snapshot of the quota data:

YR	Quarter	Region	ItemType	Dollars
2008	1	Central	Baking	170000
2008	1	Central	Beef	140000
2008	1	Central	Beverage	130000
2008	1	Central	Bread	30000
2008	1	Central	Cereal	50000
2008	1	Central	Cheese	390000
2008	1	Central	Condiments	310000
2008	1	Central	Dessert	60000
2008	1	Central	Entree	60000
2008	1	Central	Grains	0
2008	1	Central	Lamb	0
2008	1	Central	Non-food	160000
2008	1	Central	Pasta	10000
2008	1	Central	Pork	110000

n. Because Excel has limited data types, confirm or change the data types to conform to existing data types for the relevant fields. Expand the physical tables and double-click the physical columns to open the properties dialog box.
Itemtype: VARCHAR 20
Quarter: DOUBLE
YR: DOUBLE
Region VARCHAR 16
Dollars: DOUBLE

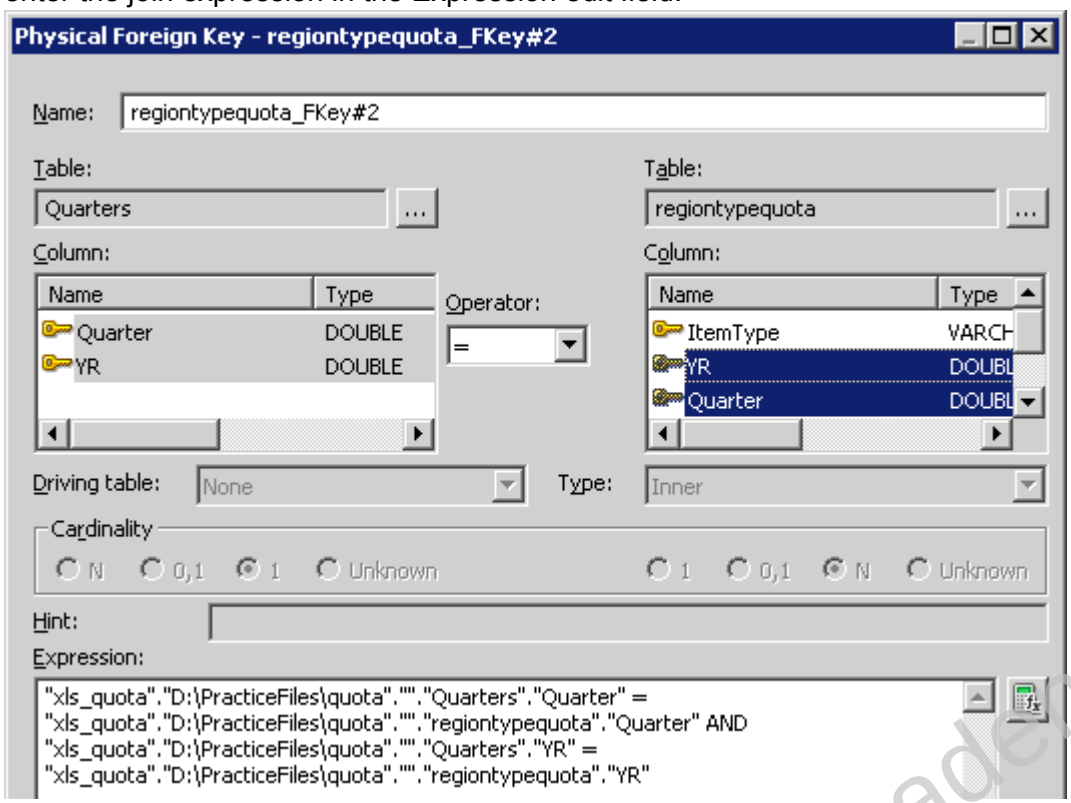
4. Specify joins and keys.

a. Use the Physical Table Diagram to specify the following foreign key joins:

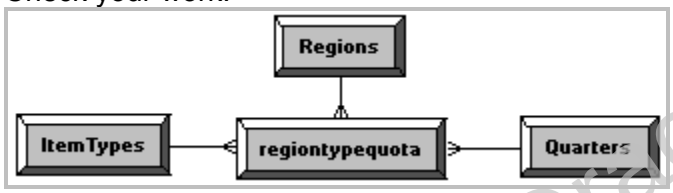
```
ItemTypes.ItemType = regiontypequota.ItemType
Regions.Region = regiontypequota.Region
```

```
Quarters.YR = regiontypequota.YR AND Quarters."Quarter" =
regiontypequota."Quarter"
```

Hint: Press and hold **Ctrl** to create the multicolumn join for the Quarters columns or enter the join expression in the Expression edit field.

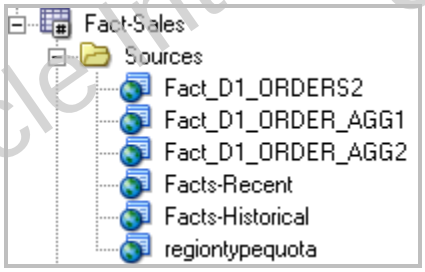


b. Check your work.



c. Close the Physical Diagram.

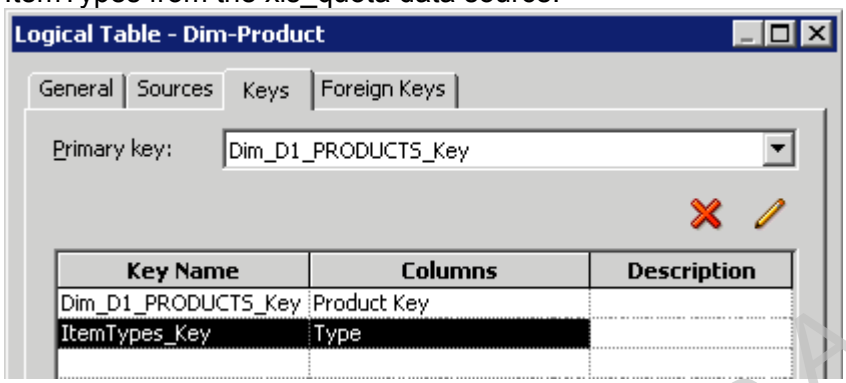
5. Create a logical column and logical table source for the quota data.
   a. Create a new logical column in the Fact-Sales logical table, name it **Quota**, and set the aggregation rule to **SUM**.
   b. Create a new logical table source for Fact-Sales by dragging the **Dollars** physical column from the **regiontypequota** physical table onto the **Quota** logical column that you just created. This automatically creates a new regiontypequota logical table source.



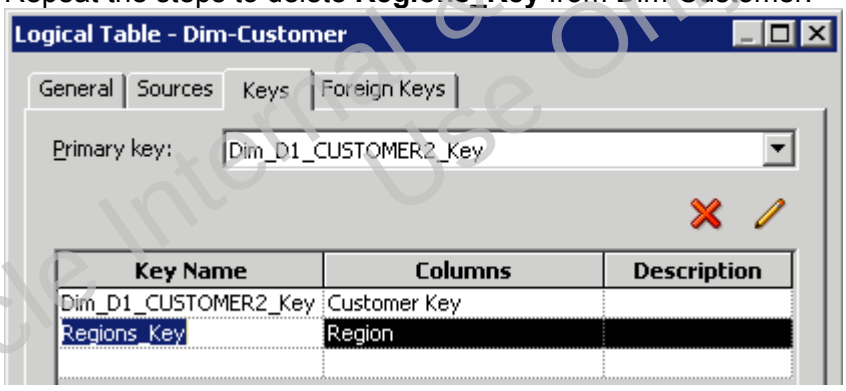
6. Create new logical table sources for the dimension tables.
   a. Drag the **YR** physical column from the **Quarters** physical table onto the **Year** logical column in the **Dim-Time** logical table. This automatically creates a new Quarters logical table source for the Periods logical table.
   b. Repeat the steps for the following columns:

Physical Table.Column		Logical Table.Column
Quarters.Quarter	onto	**Dim-Time.Quarter**
Regions.Region	onto	**Dim-Customer.Region**
ItemTypes.ItemType	onto	**Dim-Product.Type**

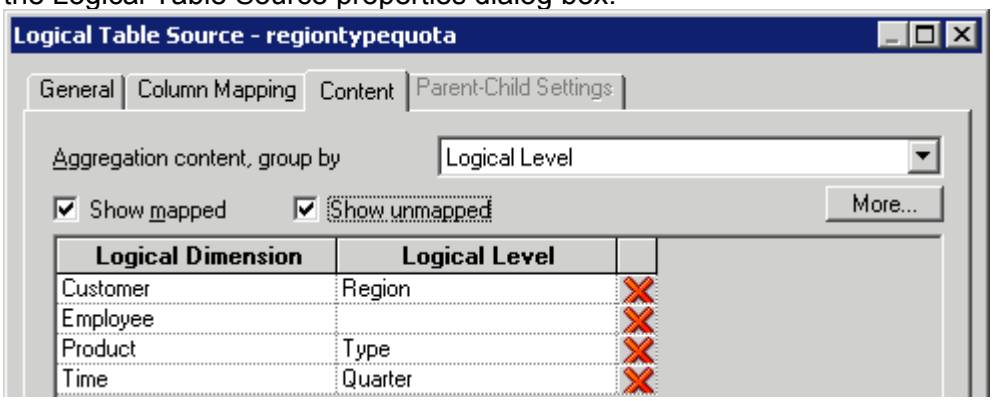
   This automatically creates a Regions logical table source for the Dim-Customer logical table and an ItemTypes logical table source for the Dim-Product logical table.
7. Remove the logical keys created for the new logical table sources. This prevents consistency check errors related to hierarchy levels.
   a. Double-click **Dim-Product** to open the Logical Table dialog box.
   b. Click the **Keys** tab.
   c. Select **ItemTypes_Key**. This is a new logical key that was created when you dragged ItemTypes from the xls_quota data source.



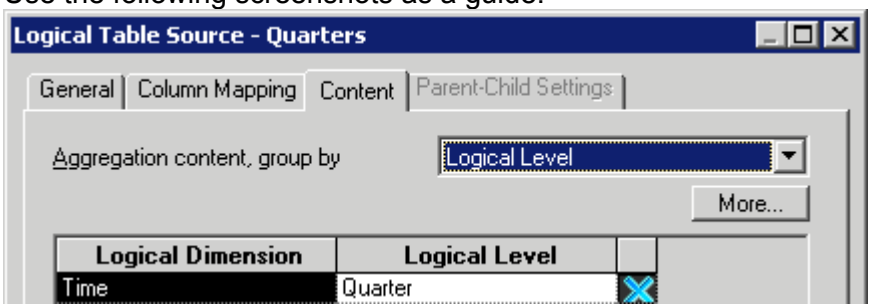
   d. Click **Delete** (the red X) to delete the key. This will prevent consistency check errors when you map a key column to a hierarchy level that is higher than the level with the detail key (Product Key in this model).
   e. Click **Yes** to confirm the deletion.
   f. Click **OK** to close the Logical Table dialog box.
   g. Repeat the steps to delete **Regions_Key** from Dim-Customer.

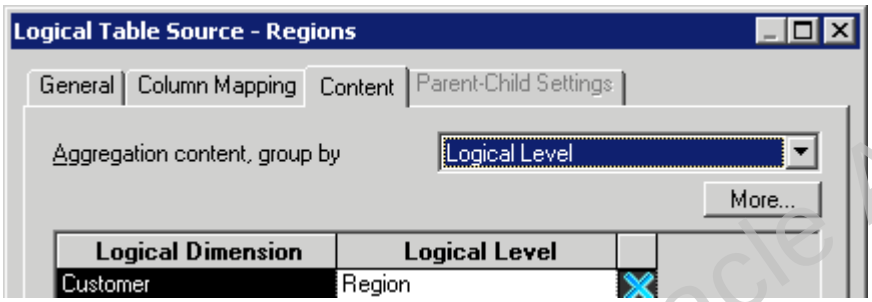


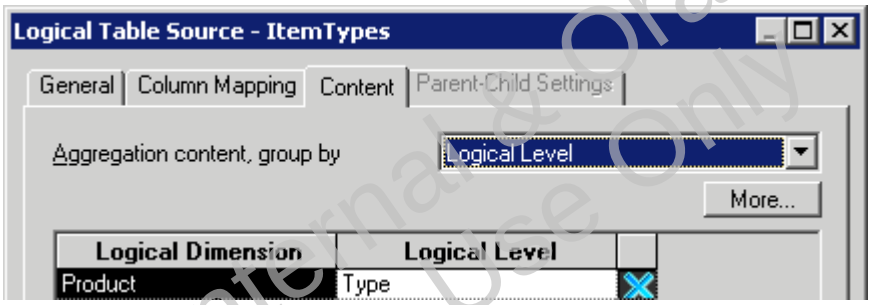
8. Specify the content for the new regiontypequota logical table source on the Content tab of the Logical Table Source properties dialog box.



9. Specify the content level for the Quarters, Regions, and ItemTypes logical table sources. Use the following screenshots as a guide:







Recall that it is best practice to set the aggregation level of these dimension sources. You specify the content of a dimension table source in terms of the hierarchy of that dimension only. The content of a fact table source is specified in terms of the content of all dimensions.

10. Drag the **Quota** logical column into the **Fact-Sales** presentation table.
11. Reset database features to their defaults.

a. In the Physical layer, double-click **xls_quota** to open the Database dialog box.
b. Click the **Features** tab.
c. Click **Reset to defaults**.
d. Click **OK** to close the Database dialog box.

12. Save the repository.
13. Check consistency. Fix any errors or warnings before you proceed.
14. Leave the repository open. You check your work after completing the next practice.

Practice 11-3: Using the Calculation Wizard to Create Derived Measures

Goal

To utilize the Administration Tool Calculation Wizard to create derived measures

Scenario

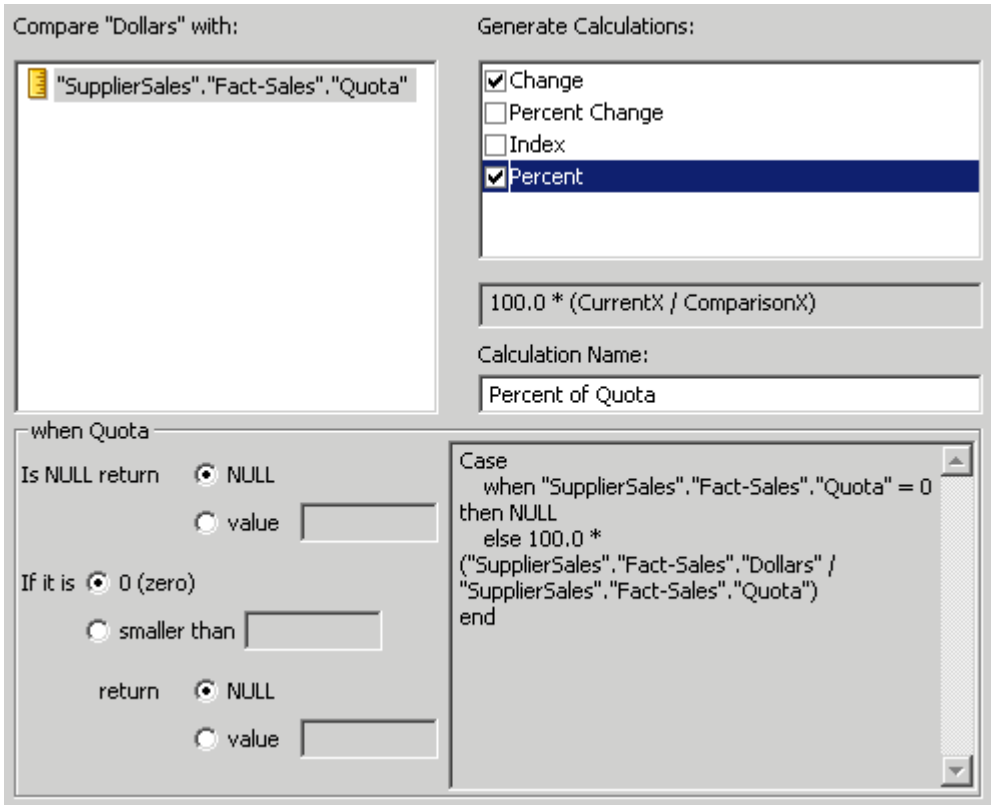
You use the Calculation Wizard to create two derived measures: Variance from Quota and Percent of Quota.

Time

10 minutes

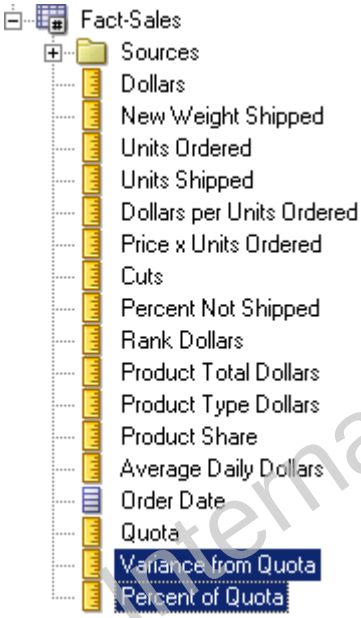
Tasks

1. In this step, you use the Calculation Wizard to create two new measures.
   a. Right-click the **Dollars** logical column and select **Calculation Wizard**. The Calculation Wizard opens.
   b. Click **Next**. The Calculation Wizard – Select Columns dialog box opens.
   c. Click **Fact-Sales** in the Choose columns pane.
   d. In the right pane, select the **Quota** check box.
   e. Click **Next**.
   f. Ensure that the **Change** check box is checked and **Change** is selected in the Generate Calculations section. Enter **Variance from Quota** in the Calculation Name field to change the name of this calculation.
   g. Deselect the **Percent Change** calculation check box.
   h. Select the **Percent** check box, and rename the measure to **Percent of Quota**. The Calculation Wizard writes the correct formula for each measure, handling the cases of NULL and zero values according to the default.



i. Click **Next**. The new calculations that the wizard will create are displayed.

j. Click **Finish** to close the Calculation Wizard. The new measures are added to the business model.



k. Drag the two new measures to the **Fact-Sales** presentation table in the Presentation layer.

l. Save the repository.

m. Check consistency. Fix any errors or warnings before you proceed.

n. Close the repository.

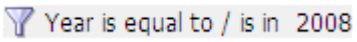
o. Leave the Administration Tool open.

2. Create and run analyses to test your work.

a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.

b. Return to Analysis Editor.

c. Create the following analysis and filter:



d. Check your results. In the following screenshot, the columns have been formatted:

Region	Quarter	Dollars	Quota	Variance from Quota	Percent of Quota
Central	1	$2,465,458	$2,230,000	$235,458	110.56%
	2	$2,407,135	$2,410,000	-$2,865	99.88%
	3	$2,555,321	$2,410,000	$145,321	106.03%
	4	$2,639,330	$2,840,000	-$200,670	92.93%
East	1	$4,588,564	$4,380,000	$208,564	104.76%
	2	$4,775,649	$4,701,000	$74,649	101.59%
	3	$4,897,643	$4,900,000	-$2,357	99.95%
	4	$5,134,833	$5,021,000	$113,833	102.27%
West	1	$4,775,286	$4,380,000	$395,286	109.02%
	2	$5,024,377	$4,730,000	$294,377	106.22%
	3	$4,937,165	$4,970,000	-$32,835	99.34%
	4	$4,827,825	$5,300,000	-$472,175	91.09%

e. Sign out of Oracle BI.

f. Check the query log. The log shows that a single logical query generated two physical queries. Oracle BI Server automatically joined the two result sets on the basis of common values in the logical dimension columns. The join of two results sets from two different sources is always a full outer join. Mapping two sources to the same logical columns (as you did for region, year, quarter, and type) was all that was necessary to

cause this join to happen.

```
---------- Sending query to database named orcl (id:
<<2704>>), connection pool named SUPPLIER CP: [[

select T76.REGION as c1,
     case  when T58.MONTH_IN_YEAR < 4 then 1 when
T58.MONTH_IN_YEAR < 7 then 2 when T58.MONTH_IN_YEAR < 10 then 3
else 4 end  as c2,
     sum(T90.DOLLARS) as c3
from
     D1_CALENDAR2 T58 /* Dim_D1_CALENDAR2 */ ,
     D1_CUSTOMER2 T76 /* Dim_D1_CUSTOMER2 */ ,
     D1_ORDERS2 T90 /* Fact_D1_ORDERS2 */
where  ( T58.YEAR = 2008 and T58.YYYYMMDD = T90.PERIODKEY and
T76.NEWKEY = T90.CUSTKEY )
group by T76.REGION, case  when T58.MONTH_IN_YEAR < 4 then 1
when T58.MONTH_IN_YEAR < 7 then 2 when T58.MONTH_IN_YEAR < 10
then 3 else 4 end
order by c2, c1

]]
[2010-08-17T11:20:19.000+00:00] [oracleBIServerComponent]
[TRACE:2] [USER-18] [] [ecid:
0000Idzlo4P7i4WzLwFS8A1COfkc0003z0] [tid: 4cc] [requestid:
7c75000e] [sessionid: 7c750000] [username: weblogic] ---------
---------- Sending query to database named xls_quota (id:
<<2886>>), connection pool named Connection Pool: [[
select sum(T964."Dollars") as c1,
     T962."Region" as c2,
     T959."Quarter" as c3
from
     "Quarters" T959,
     "Regions" T962,
     "regiontypequota" T964
where  ( T959."YR" = T964."YR" and T959."Quarter" =
T964."Quarter" and T959."YR" = 2008 and T962."Region" =
T964."Region" and T964."YR" = 2008 )
group by T959."Quarter", T962."Region"
```

Practices for Lesson 12: Using Repository Variables

Lesson 12

Practices for Lesson 12

Lesson Overview

In these practices, you will create and use dynamic repository variables with initialization blocks.

Practice 12-1: Creating Dynamic Repository Variables

Goal

To create and use dynamic repository variables with initialization blocks

Scenario

Dynamic repository variables are useful for defining the content of logical table sources. ABC has two sources for information about orders: one source contains recent orders and the other source contains historical data. You must describe the content of these sources on the Content tab of the Logical Table Source dialog box. Without using dynamic repository variables, you would have to describe the content of the source containing recent data with a static expression such as: `SupplierSales."Fact-Sales"."Order Date" > '20081231'`, as you did in the previous practice. This content statement becomes invalid as new data is added to the recent source and older data is moved to the historical source. To accurately reflect any new data, you would have to modify the fragmentation content description manually. You can set up dynamic repository values to do this automatically. To use a dynamic repository variable, you must also set up an initialization block to refresh the variable on a continuing basis.

Outcome

A new initialization block called OrderSplit is created with a variable named EndHistoricalData. In the Business Model and Mapping layer, logical tables' sources are modified for the Fact-Sales logical table.

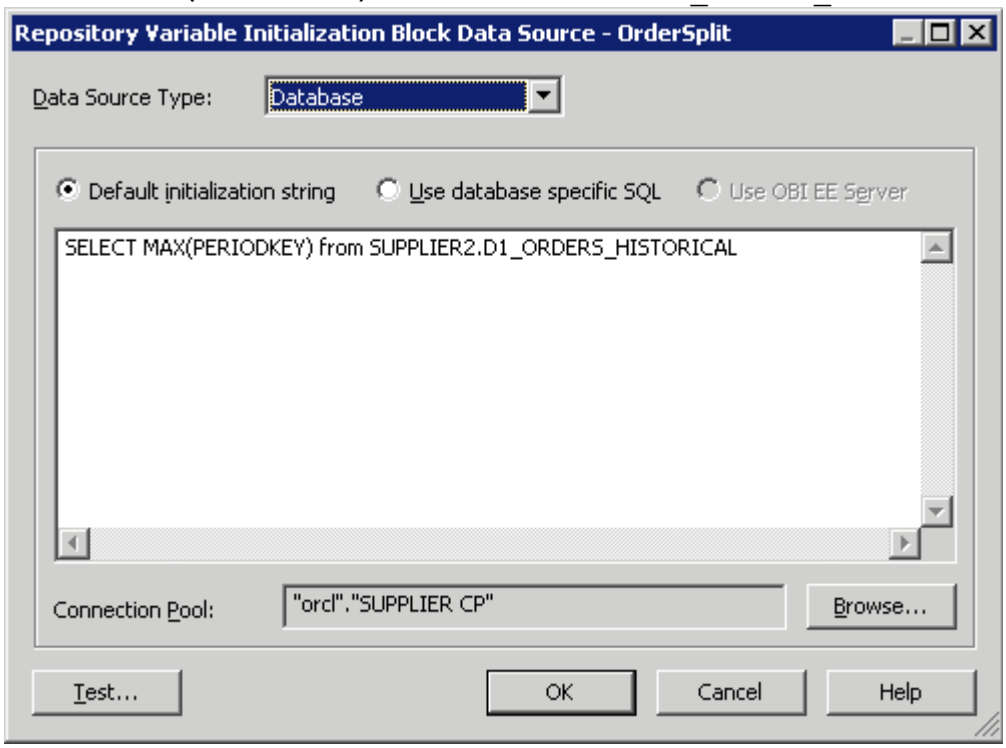
Time

25 minutes

Tasks

1. In this step, you create an initialization block. Initialization blocks contain SQL statements that initialize variables. The variables are repository variables and their values persist until the initialization block resets them the next time it runs. Repository initialization blocks execute whenever Oracle BI Server starts up, and thereafter according to any schedule that is applied.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. Select **Manage > Variables**.
   c. Select **Action > New > Repository> Initialization Block**.
   d. Name the initialization block **OrderSplit**.
   e. Leave the schedule set to the default.
   f. Click **Edit Data Source**.
   g. Click the **Browse** button.
   h. Double-click the **SUPPLIER CP** connection pool object to select it. The connection pool is added to the Repository Variable Init Block Data Source dialog box.
   i. Enter the following SQL in the Default Initialization String field to capture the most recent period key (for example 20081231) from the D1_ORDERS_HISTORICAL table:

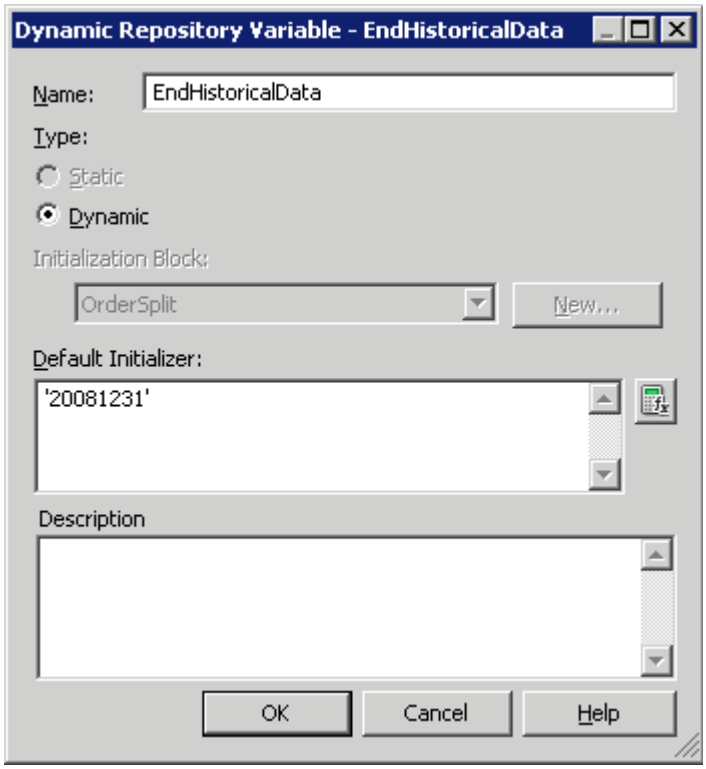
```
SELECT MAX(PERIODKEY) from SUPPLIER2.D1_ORDERS_HISTORICAL
```



j. Click **OK** to close the Repository Variable Initialization Block Data Source dialog box.

2. Create the variable.
   a. Click **Edit Data Target**.
   b. Click the **New** button.
   c. In the Name field, enter **EndHistoricalData**.

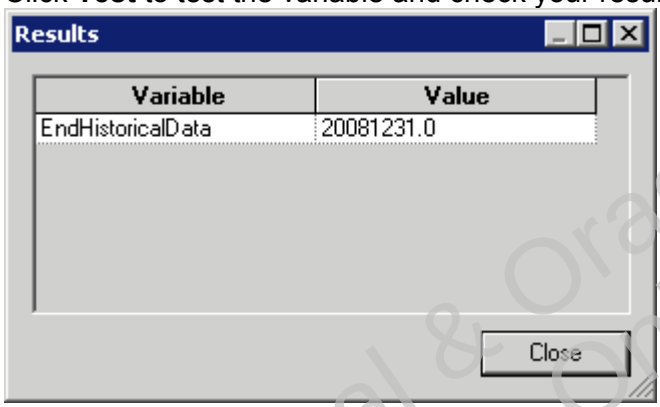
d. Enter **'20081231'** in the Default Initializer field.



e. Click **OK** to close the Dynamic Repository Variable dialog box.

f. Click **OK** to close the Repository Variable Initialization Block Variable Target dialog box.

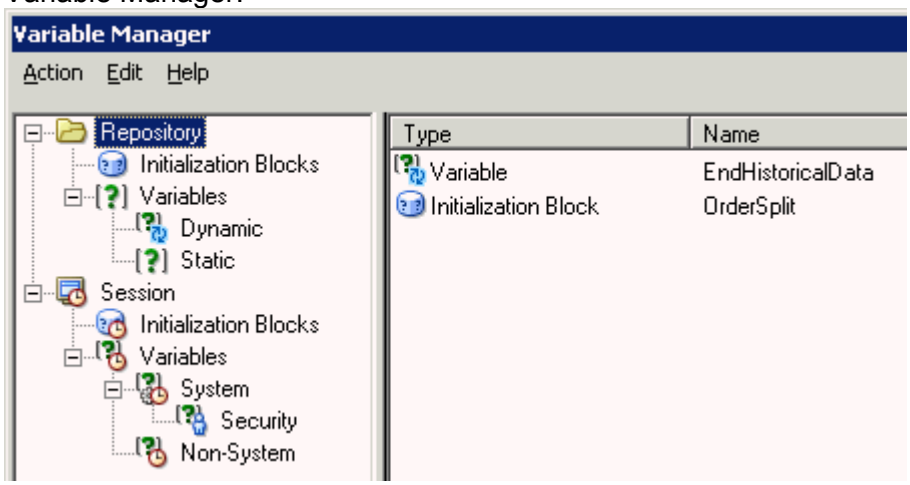
g. Click **Test** to test the variable and check your results:



h. Click **Close**.

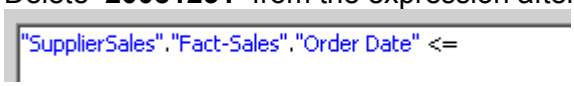
i. Click **OK** to close the Repository Variable Initialization Block dialog box. The OrderSplit initialization block and EndHistoricalData dynamic repository variable are visible in the

Variable Manager.

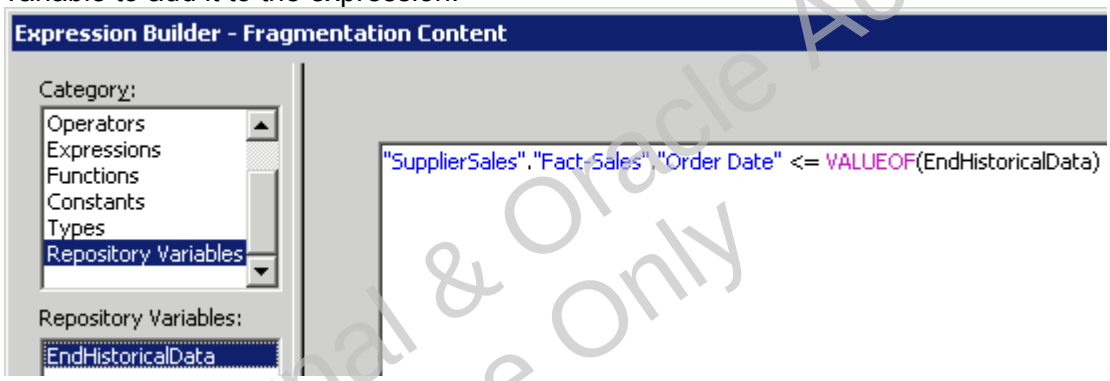


j. Select **Action > Close** to close the Variable Manager.

3. Use the variable to dynamically determine the fragmentation content of the **Facts-Historical** logical table source.
   a. In the **SupplierSales** business model, expand **Fact-Sales > Sources**.
   b. Double-click **Facts-Historical**.
   c. Click the **Content** tab.
   d. Open the Expression Builder by clicking the **Edit Expression** button next to the Fragmentation content field.
   e. Delete **'20081231'** from the expression after the <= operator.

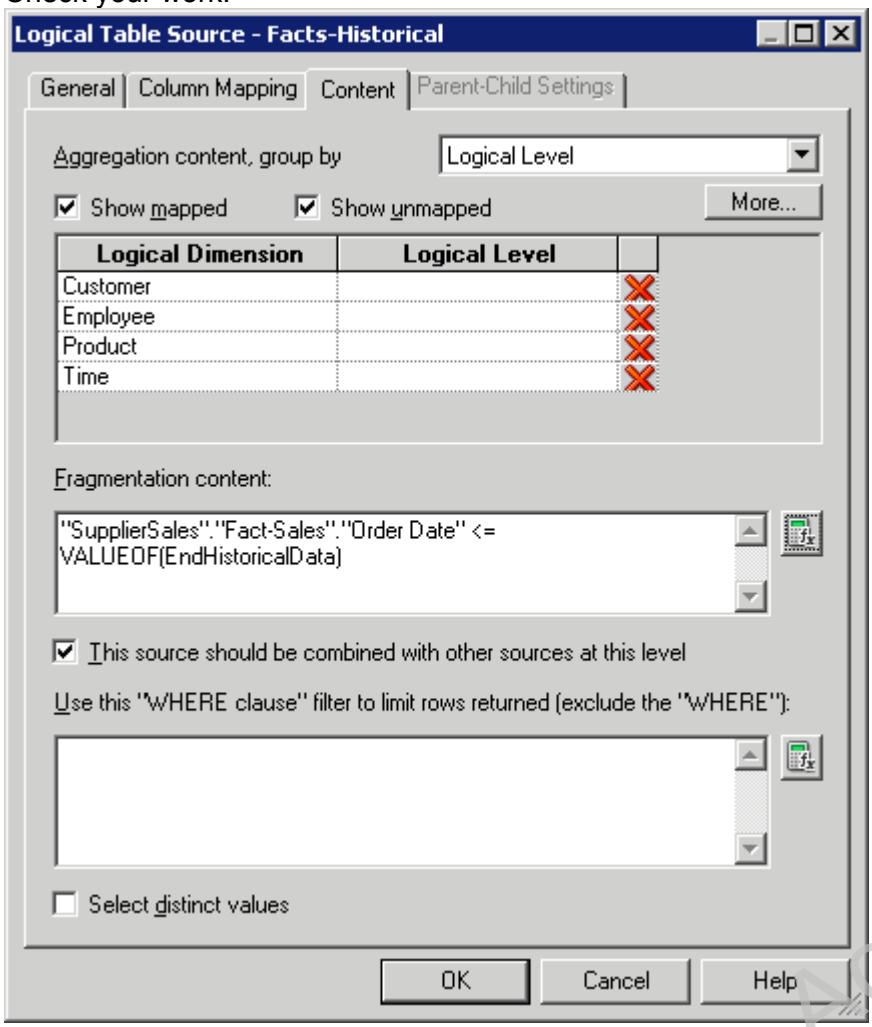


   f. In the left pane, select **Repository Variables** and double-click the **EndHistoricalData** variable to add it to the expression.



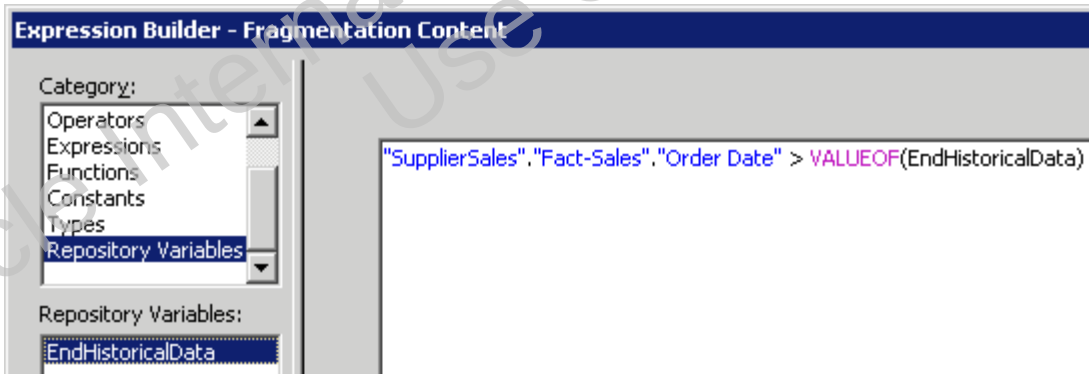
   g. Click **OK** to close Expression Builder.
   h. Ensure that the **This source should be combined with other sources at this level** check box is selected.

i. Check your work.



j. Click **OK** to close the Logical Table Source dialog box.

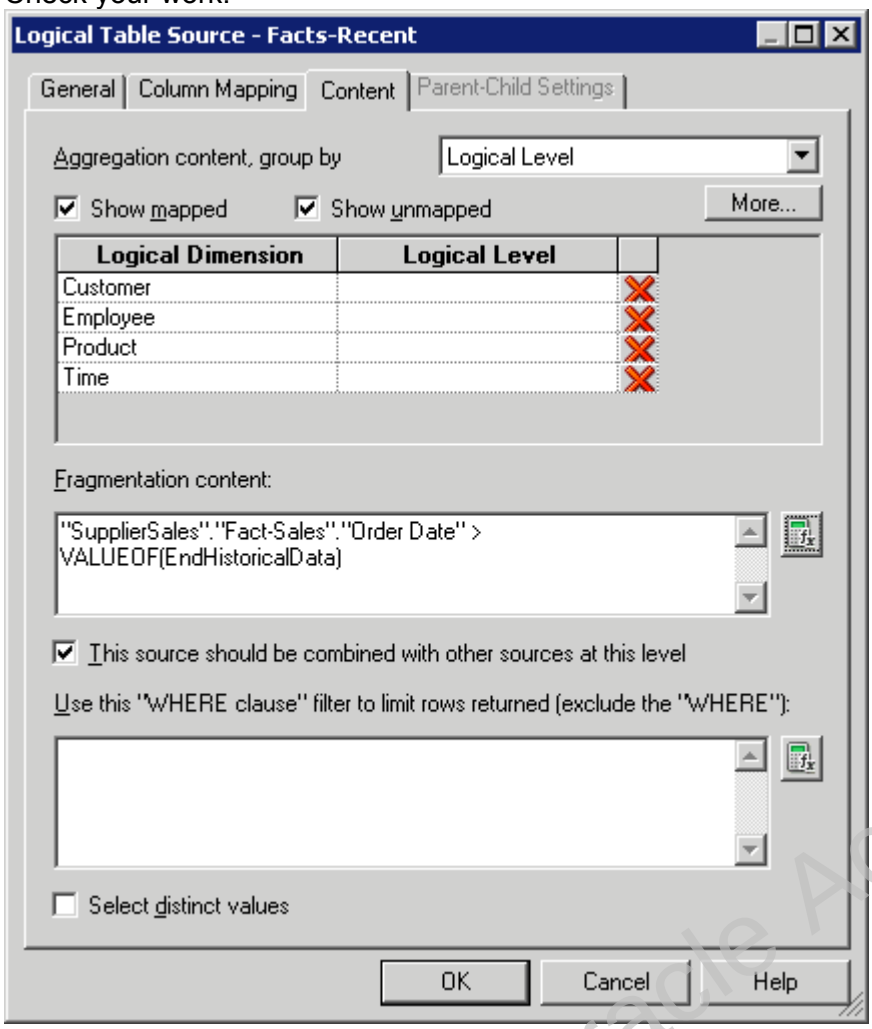
4. Use the variable to dynamically determine the content in the **Facts-Recent** logical table source.
   a. Double-click **Facts-Recent**.
   b. Click the **Content** tab.
   c. Open the Expression Builder.
   d. Modify the existing expression by replacing **20081231** with the **EndHistoricalData** variable:



e. Click **OK** to close Expression Builder.

f. Ensure that the **This source should be combined with other sources at this level** check box is selected.

g. Check your work.



h. Click **OK** to close the Logical Table Source dialog box.

i. Save the repository.

j. Check consistency. Fix errors or warning messages before you proceed.

k. Close the repository.

l. Leave the Administration Tool open.

5. Test your work.

a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.

b. Return to Analysis Editor.

c. Create an analysis:

d. Click **Results**.

Order Date	Dollars
20080102	$26,036
20080103	$12,210
20080105	$140,140
20080106	$85,079
20080107	$399,278
20080108	$151,171
20080109	$107,925
20080110	$110,789

Notice that Order Date records are returned for both before and after 20081231. Click the **Display Maximum** button at the bottom of the table to confirm.

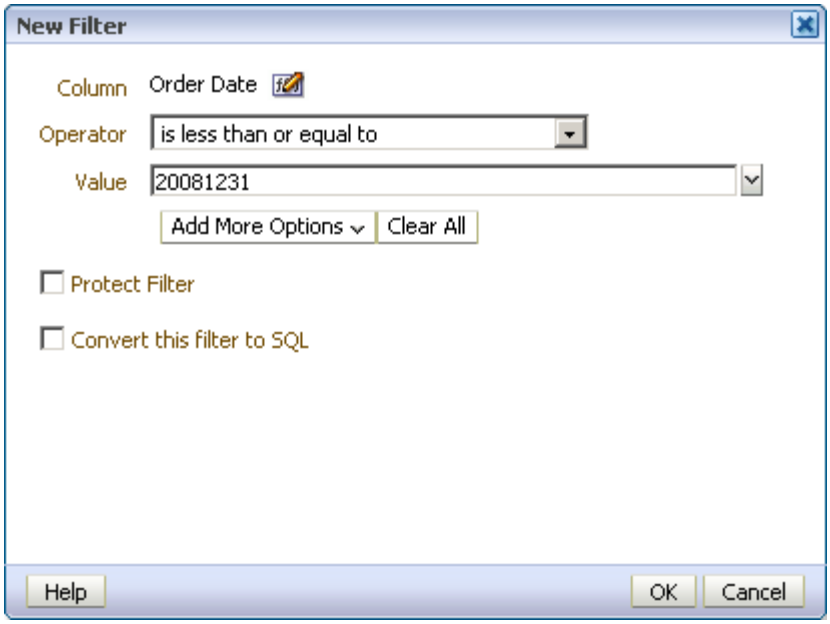
e. Leave Analysis Editor open.

f. Open the query log. Confirm that the query references both partition tables. Both partition tables are used because you did not apply a filter in the query. Notice that a separate `SELECT` statement is sent to each table and Oracle BI Server performs a `UNION ALL` to combine the results.

```
--------- Sending query to database named orcl (id: <<2641>>),
connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS ((select T928.PERIODKEY as c2,
     T928.DOLLARS as c3
from
     D1_ORDERS_RECENT T928 /* Fact_D1_ORDERS_RECENT */
union all
select T919.PERIODKEY as c2,
     T919.DOLLARS as c3
from
     D1_ORDERS_HISTORICAL T919 /* Fact_D1_ORDERS_HISTORICAL */
)),
SAWITH1 AS (select sum(D3.c3) as c1,
     D3.c2 as c2
from
     SAWITH0 D3
group by D3.c2)
select distinct 0 as c1,
     D2.c2 as c2,
     D2.c1 as c3
from
     SAWITH1 D2
order by c2
```

g. Close the log file.

h. Return to Analysis Editor and add a filter where **Order Date** is less than or equal to **20081231**.



i. Run the analysis again.

j. Verify that your results display only Order Dates that are less than or equal to 20081231.

Order Date	Dollars
20080102	$26,036
20080103	$12,210
20080105	$140,140
20080106	$85,079
20080107	$399,278
20080108	$151,171
20080109	$107,925
20080110	$110,789

k. Leave Analysis Editor open.

l. Open the query log and verify that the query accessed only one partition, Fact_D1_ORDERS_HISTORICAL.

```
---------- Sending query to database named orcl (id:
<<2630>>), connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T919.DOLLARS) as c1,
     T919.PERIODKEY as c2
from
     D1_ORDERS_HISTORICAL T919 /* Fact_D1_ORDERS_HISTORICAL */
where  ( T919.PERIODKEY <= 20081231 )
group by T919.PERIODKEY)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c1 as c3
from
     SAWITH0 D1
order by c2 NULLS FIRST
```

m. Close the log file.

n. Return to the analysis and edit the filter to show records where **Order Date** is greater than 20081231.

o. Run the analysis.

p. Verify that your result only displays records where Order Date is greater than 20081231.

Order Date	Dollars
20090102	$143,693
20090104	$324,001
20090105	$172,597
20090106	$112,132
20090107	$170,791
20090108	$171,649
20090109	$119,321
20090111	$302,795
20090112	$136,086

q. Sign out of Oracle BI.

r. Open the query log and confirm that the query being issued accesses the correct partition, Fact_D1_ORDERS_RECENT.

```
---------- Sending query to database named orcl (id:
<<2887>>), connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T928.DOLLARS) as c1,
     T928.PERIODKEY as c2
from
     D1_ORDERS_RECENT T928 /* Fact_D1_ORDERS_RECENT */
where  ( 20081231 < T928.PERIODKEY )
group by T928.PERIODKEY)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c1 as c3
from
     SAWITH0 D1
order by c2 NULLS FIRST
```

Practice 12-2: Using Dynamic Repository Variables as Filters

Goal

To create and use dynamic repository variables as filters

Scenario

Rather than creating hard-coded column filters like Year = 2009 in analyses, ABC wants to use variables that always return the current year, current month, and current day. You create these dynamic repository variables and then use them in column filters in Oracle BI analyses.

Outcome

A new initialization block, Current Periods, and three new dynamic repository variables (CurrentYear, CurrentMonth, and CurrentDay) are created.

Time

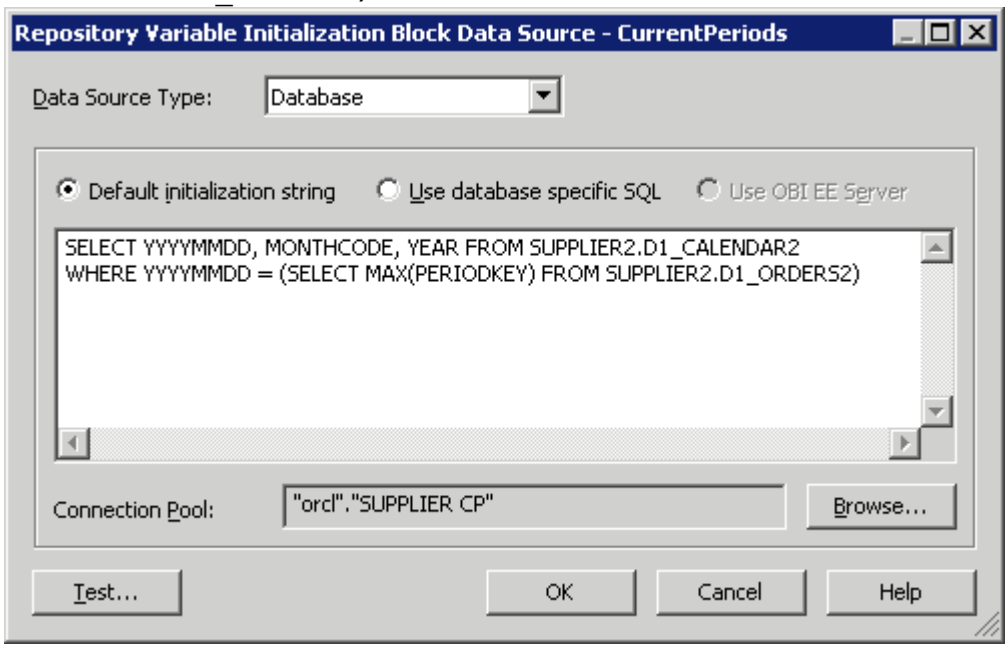
15 minutes

Tasks

1. In this step, you create a new initialization block. This initialization block will initialize three variables named CurrentYear, CurrentMonth, and CurrentDay. The variables will get their values from the initialization block SQL according to the schedule you set. In this example, the system determines the value of the current day by finding the maximum value of the period key (YYYMMDD) in the D1_ORDERS2 table and then determining the month code and year that correspond to that value in the D1_CALENDAR2 table.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. Select **Manage > Variables**.
   c. Select **Action > New > Repository > Initialization Block**.
   d. Name the initialization block **CurrentPeriods**.
   e. Click **Edit Data Source**.
   f. Click the **Browse** button.
   g. Double-click the **SUPPLIER CP** connection pool object to select it.
   h. Enter the following SQL in the block to determine the value of the current day by finding the maximum value of the period key (YYYMMDD):

```
SELECT YYYYMMDD, MONTHCODE, YEAR FROM SUPPLIER2.D1_CALENDAR2
WHERE YYYYMMDD = (SELECT MAX(PERIODKEY) FROM
```

```
SUPPLIER2.D1_ORDERS2)
```

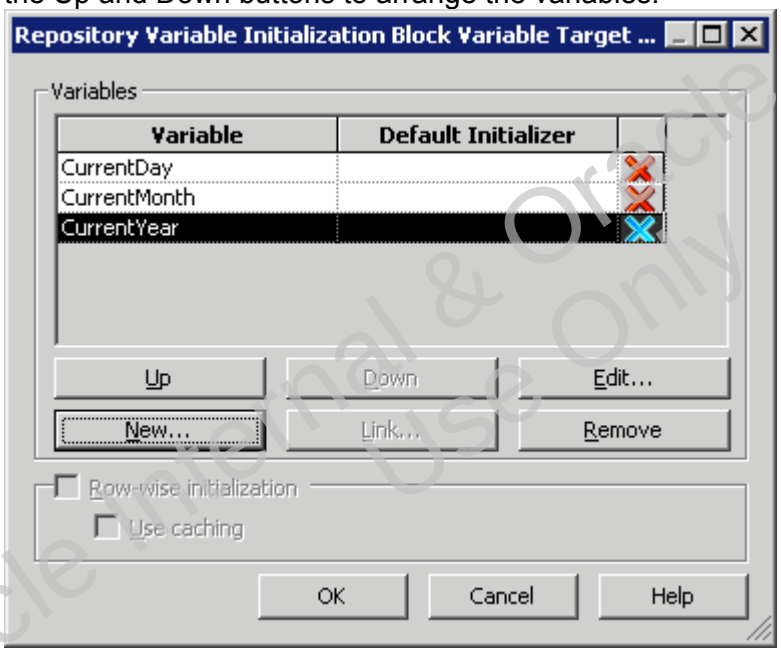


i. Click **OK** to close the Repository Variable Initialization Block Data Source dialog box.

2. Create the variables.

a. Click the **Edit Data Target**.

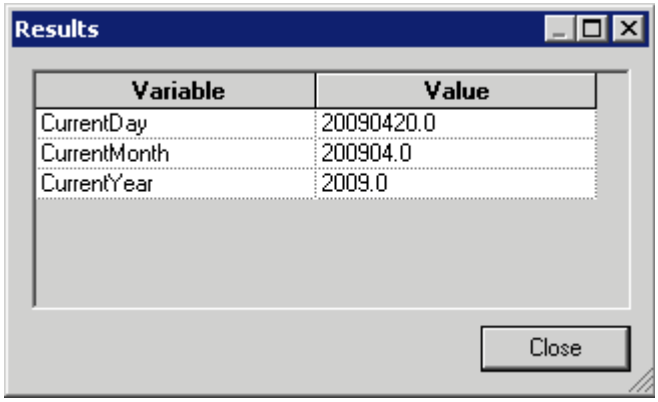
b. Create three new variables: **CurrentDay**, **CurrentMonth**, and **CurrentYear**. The order is important. The value returned from the first column in the initialization block SQL, YYYYMMDD, is assigned to the CurrentDay variable. The value of the second column, MONTHCODE, is assigned to CurrentMonth (the second variable), and the value of the third column, YEAR, is assigned to CurrentYear (the third variable). If necessary, use the Up and Down buttons to arrange the variables.



c. Click **OK** to close the Repository Variable Initialization Block Variable Target dialog box.

d. Leave the default refresh interval set to every hour. This means that the variables will be reinitialized every hour.

e. Click the **Test** button and check the results.

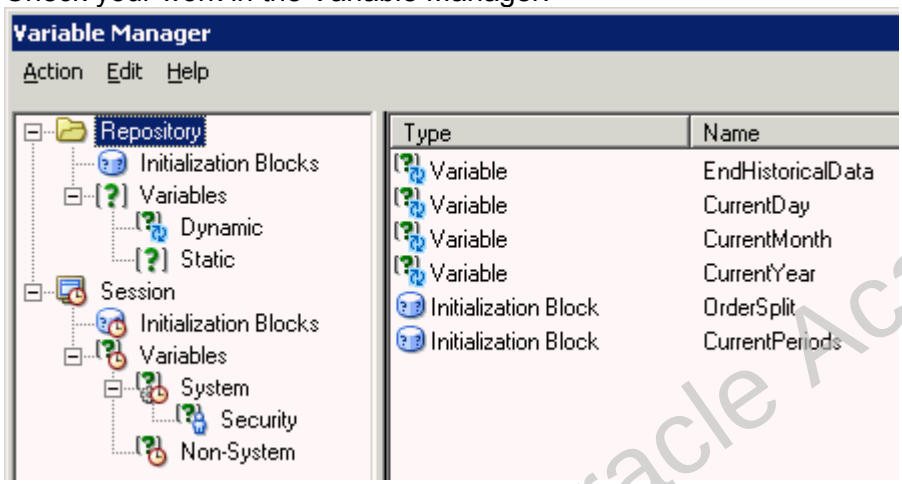


In this example, the results are determined by the data in the SUPPLIER2 database used for this course, which holds data only through April 2009.

f. Click **Close** to close the Results window.

g. Click **OK** to close the Repository Variable Initialization Block dialog box.

h. Check your work in the Variable Manager:



i. Close the Variable Manager.

j. Save the repository.

k. Check consistency. Fix any errors or warnings before proceeding.

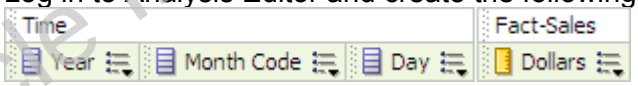
l. Close the repository.

m. Leave the Administration Tool open.

3. Test your work.

a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.

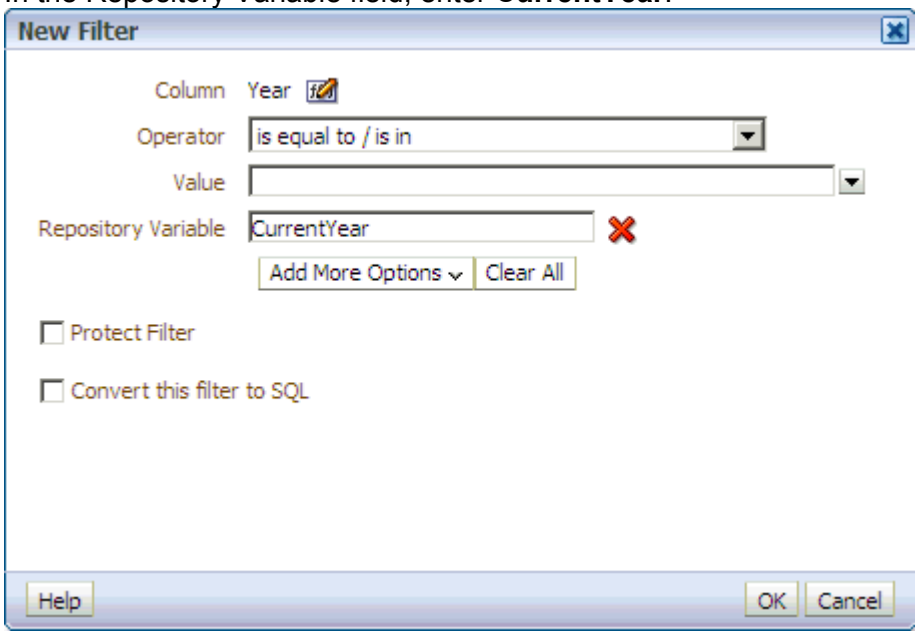
b. Log in to Analysis Editor and create the following analysis:



c. Select **Filter** for the Year column. The New Filter dialog box opens.

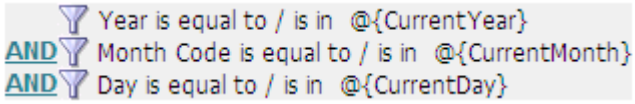
d. Select **Add More Options > Repository Variable**.

e. In the Repository Variable field, enter **CurrentYear**.



f. Click **OK**.

g. Repeat to add the **CurrentMonth** and **CurrentDay** repository variables as filters for the **Month Code** and **Day** columns, respectively.



h. Click **Results** and ensure that only data for the current year, current month, and current day (2009 based on this data set) is returned.

Year	Month Code	Day	Dollars
2009	200904	20090420	$123,540

i. Sign out of Oracle BI.

j. Check the query log. Notice that the logical request filters for the variables. The following screenshot shows only a partial view of the log file.

```
########################################## [[
-------------------- SQL Request:
SET VARIABLE QUERY_SRC_CD='Report';SELECT s_0, s_1, s_2, s_3,
s_4 FROM (SELECT
   0 s_0,
   "SupplierSales"."Time"."Day" s_1,
   "SupplierSales"."Time"."Month Code" s_2,
   "SupplierSales"."Time"."Year" s_3,
   "SupplierSales"."Fact-Sales"."Dollars" s_4
FROM "SupplierSales"
WHERE
(("Time"."Year" =  VALUEOF("CurrentYear")) AND ("Time"."Month
Code" =  VALUEOF("CurrentMonth")) AND ("Time"."Day" =
VALUEOF("CurrentDay")))
) djm ORDER BY 1, 4 ASC NULLS FIRST, 3 ASC NULLS FIRST, 2 ASC
NULLS FIRST
```

Practices for Lesson 13: Modeling Time Series Data

Lesson 13

Practices for Lesson 13

Lesson Overview

In these practices, you will create time series calculation measures using Oracle BI time series functions.

Practice 13-1: Creating Time Series Comparison Measures

Goal

To create time series calculation measures using Oracle BI time series functions

Scenario

You use the Oracle BI time series functions to create new measures to enable users to compare current dollar performance to previous time periods. You then add the new measures to the presentation catalog and test those by using analyses.

Time series functions include AGO, TODATE, and PERIODROLLING. These functions let 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 AGO, TODATE, and PERIODROLLING functions based on the calendar tables in your data warehouse, not on standard SQL date manipulation functions.

Time

25 minutes

Background

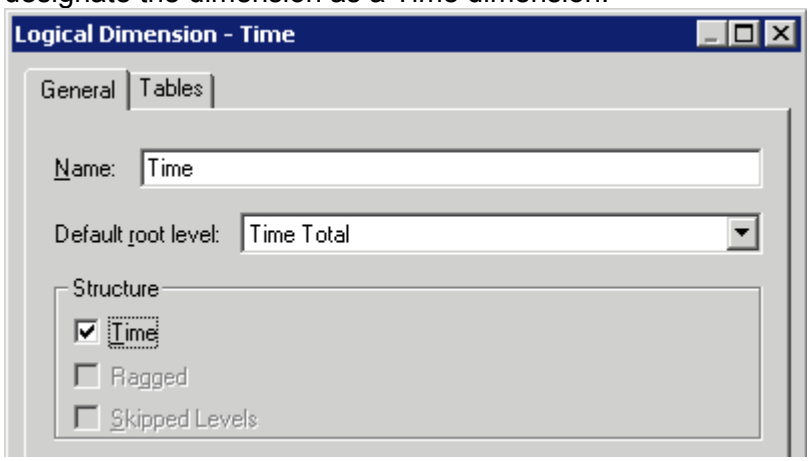
The ability to compare business performance with previous time periods is fundamental to understanding a business. Yet, as Ralph Kimball states, SQL was not designed to make straightforward comparisons over time:

“The most difficult area of data warehousing is the translation of simple business analyses into SQL. SQL was not designed with business reports in mind. SQL was really an interim language designed to allow relational table semantics to be expressed in a convenient and accessible form, and to enable researchers and early developers to proceed with building the first relational systems in the mid-1970s. How else can you explain the fact that there is no direct way in SQL to compare this year to last year?” – Ralph Kimball

Tasks

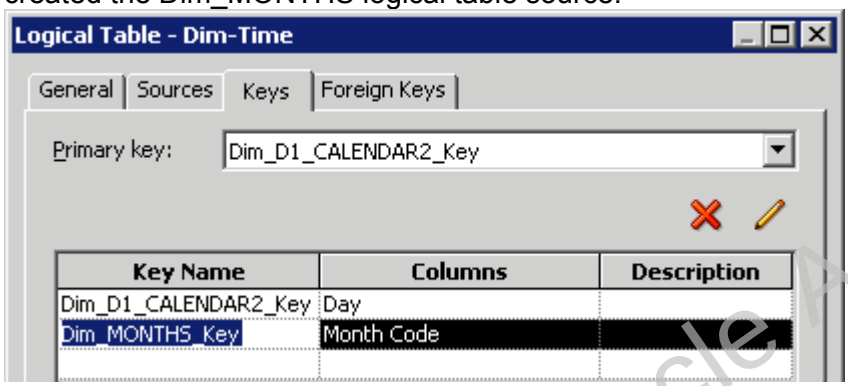
1. In the SupplierSales business model, set the Time logical dimension as a time dimension.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. In the SupplierSales business model, double-click the **Time** logical dimension.
   c. On the General tab, select the **Time** check box. Time series functions operate on time-oriented dimensions. To use these functions on a particular dimension, you must

designate the dimension as a Time dimension.



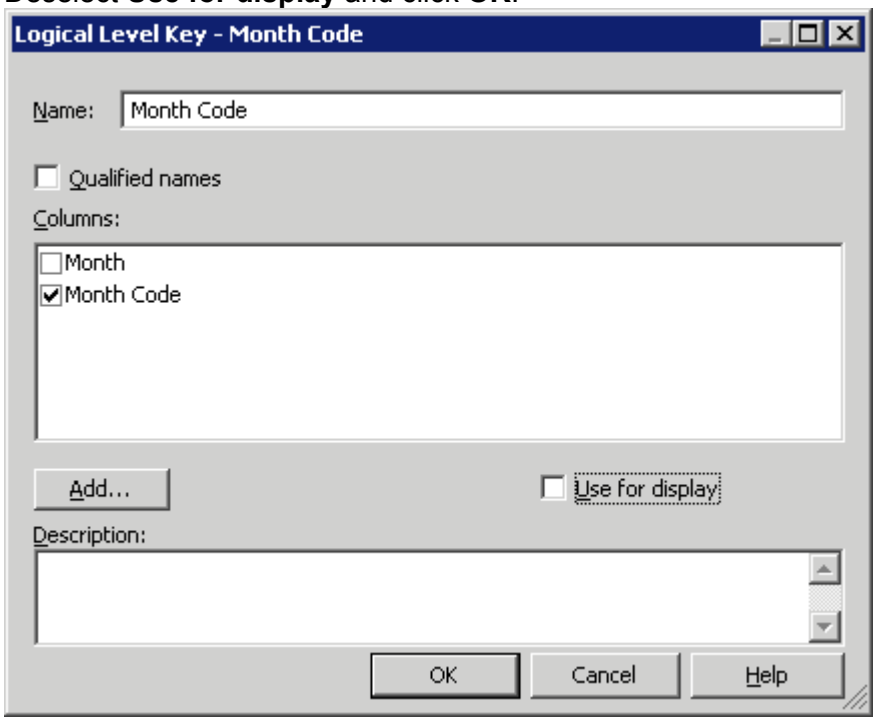
d. Click **OK** to close the dialog box.

2. Remove the logical key created for the Dim_MONTHS logical table source. This prevents consistency check errors related to hierarchy levels.
   a. Double-click **Dim-Time** to open the Logical Table dialog box.
   b. Click the **Keys** tab.
   c. Select **Dim_MONTHS_Key**. This is a new logical key that was created when you created the Dim_MONTHS logical table source.

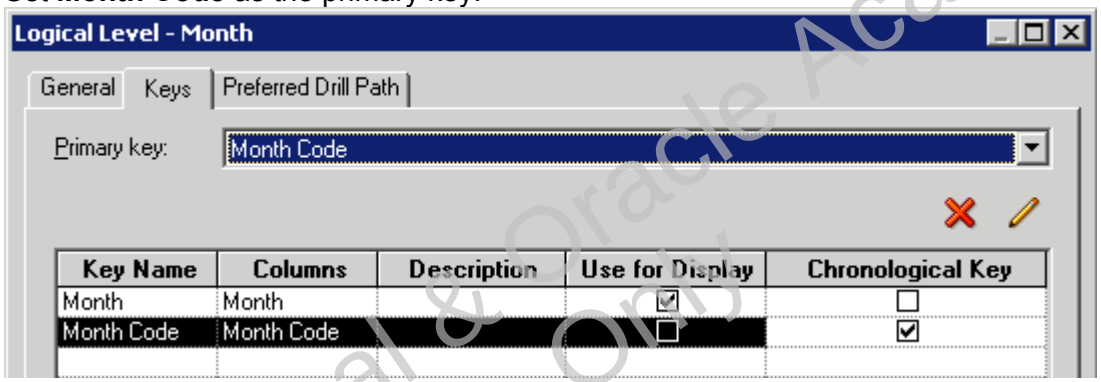


   d. Click **Delete** (the red X) to delete the key. This will prevent consistency check errors when you map a key column to a hierarchy level that is higher than the level with the detail key (Product Key in this model).
   e. Click **Yes** to confirm the deletion.
   f. Click **OK** to close the Logical Table dialog box.
3. Add MonthCode to the Month level in the Time logical dimension.
   a. Expand the **Dim-Time** logical table.
   b. Expand the **Time** logical dimension.
   c. Drag **Month Code** from the **Dim-Time** logical table to the **Month** level in the **Time** logical dimension.
   d. Right-click the **Month Code** level column and select **New Logical Level Key** to open the Logical Level Key dialog box.

e. Deselect **Use for display** and click **OK**.



4. Identify level keys as chronological keys. It is best practice to designate a chronological key for every level of a time dimension hierarchy.
   a. Double-click the **Month** level to open the Logical Level dialog box.
   b. Click the **Keys** tab.
   c. Select the **Chronological Key** check box for **Month Code**.
   d. Set **Month Code** as the primary key.

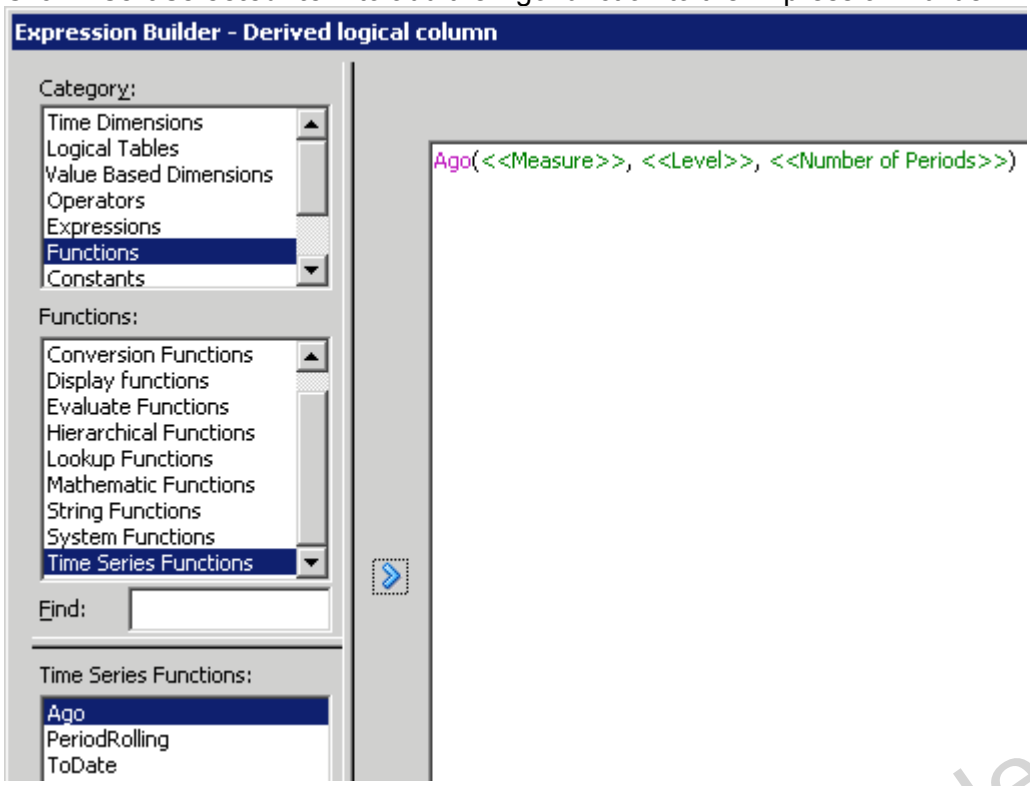


   e. Click **OK** to close the Logical Level dialog box.
   f. Repeat and set chronological keys for the remaining levels:
      Time Detail: Day
      Quarter: Quarter
      Year: Year
5. Create a measure that calculates dollars for the previous month by using the AGO function.
   a. Right-click the **Fact-Sales** logical table and select **New Object > Logical Column**.
   b. On the General tab, name the column **Month Ago Dollars**.
   c. On the **Column Source** tab, select **Derived from existing columns using an expression**.

d. Open the Expression Builder.

e. Select **Functions > Time Series Functions > Ago**.

f. Click **Insert selected item** to add the Ago function to the Expression Builder.



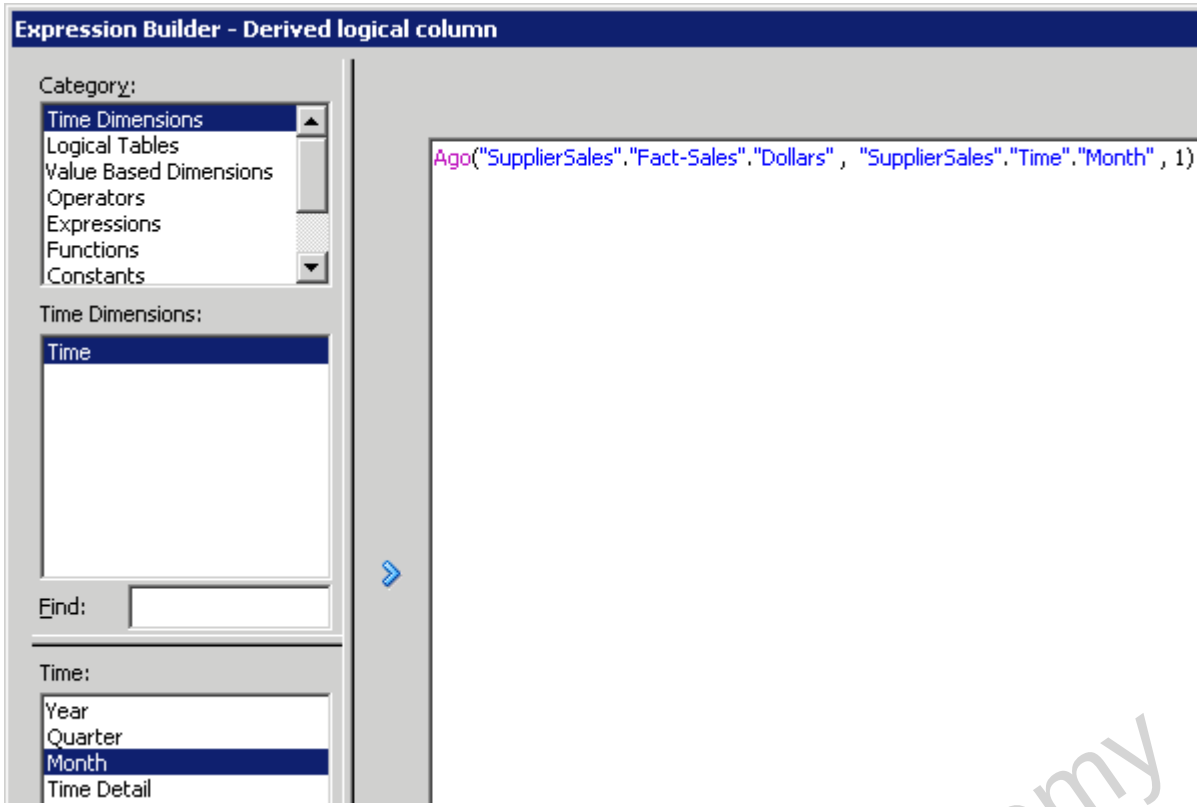
g. Click **<<Measure>>** in the expression.

h. Select **Logical Tables > Fact-Sales** and then double-click **Dollars** to add it to the expression.

i. Click **<<Level>>** in the expression.

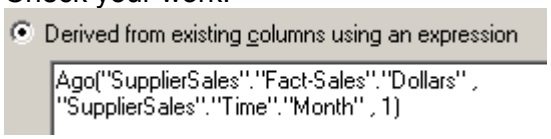
j. Select **Time Dimensions > Time** and then double-click **Month** to add it to the expression.

k. Click **<<Number of Periods>>** and enter **1**. The Ago function calculates the Dollars value one month before the current month.



l. Click **OK** to close the Expression Builder.

m. Check your work.



n. Click **OK** to close the Logical Column dialog box.

o. Drag the **Month Ago Dollars** logical column to the Fact-Sales presentation folder.

6. Create a measure that calculates the difference in dollars between the current month and the previous month.

a. Right-click **Fact-Sales** and select **New Object > Logical Column**.

b. On the General tab, name the new logical column **Change Month Ago Dollars**.

c. On the Column Source tab, select **Derived from existing columns using an expression**.

d. Open the Expression Builder.

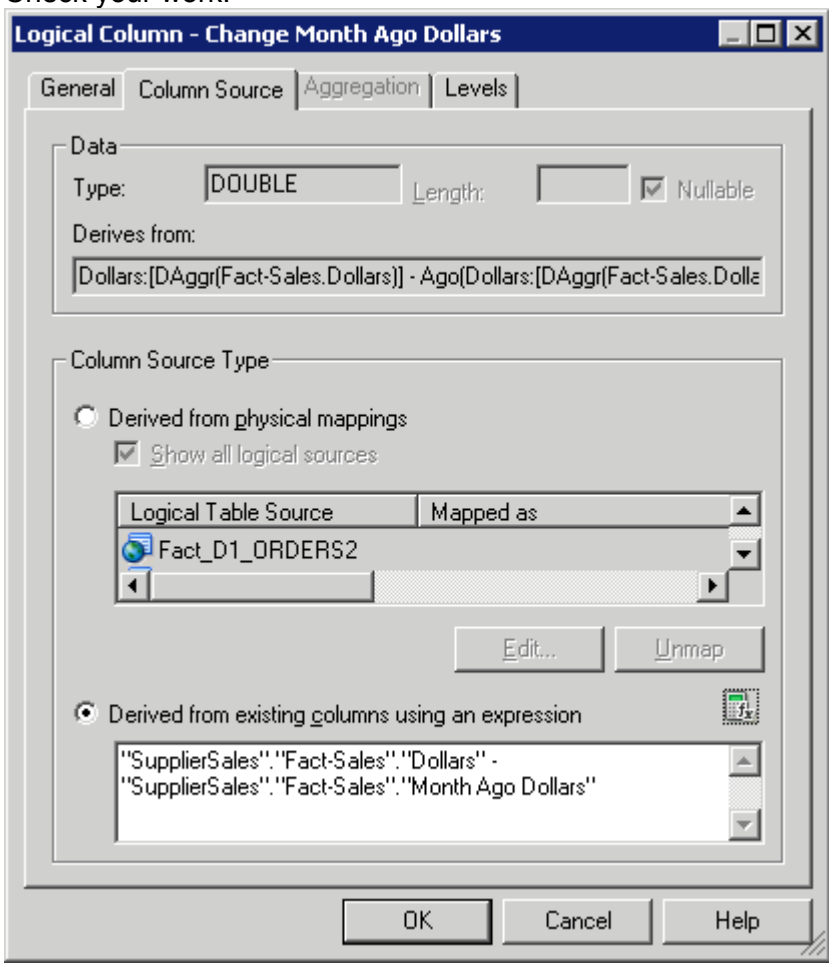
e. Select **Logical Tables > Fact-Sales** and then double-click **Dollars** to add it to the expression.

f. Insert a **minus sign**.

g. Select **Logical Tables > Fact-Sales** and then double-click **Month Ago Dollars**.

h. Click **OK** to close the Expression Builder.

i. Check your work.



j. Click **OK** to close the Logical Column dialog box.

k. Drag the **Change Month Ago Dollars** logical column to the Fact-Sales presentation folder.

7. Create a measure that calculates the percentage change in dollars between the current month and the previous month.

   a. Right-click **Fact-Sales** and select **New Object > Logical Column**.

   b. Name the new logical column **Percent Change Month Ago Dollars**.

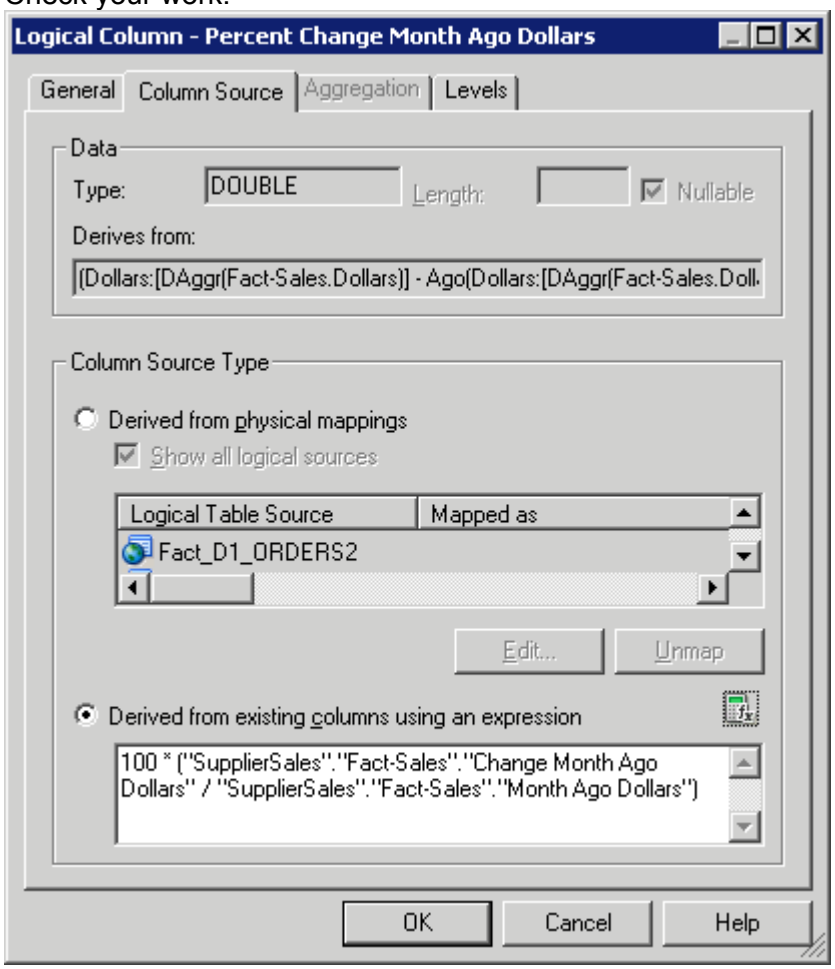
   c. On the Column Source tab, select **Derived from existing columns using an expression**.

   d. Open the Expression Builder and create the following expression:

```
100 * ("SupplierSales"."Fact-Sales"."Change Month Ago Dollars" /
"SupplierSales"."Fact-Sales"."Month Ago Dollars")
```

   e. Click **OK** to close the Expression Builder.

f. Check your work.



g. Click **OK** to close the Logical Column dialog box.

h. Drag the **Percent Change Month Ago Dollars** logical column to the Fact-Sales presentation folder.

8. Create a measure that calculates a running sum of dollars over the past year on a monthly basis using the TODATE function.

a. Right-click **Fact-Sales** and select **New Object > Logical Column**.

b. Name the new logical column **Year To Date Dollars**.

c. On the Column Source tab, select **Derived from existing columns using an expression**.

d. Open the Expression Builder.

e. Select **Functions > Time Series Functions** and double-click **ToDate** to insert the expression.

f. Click **<<Measure>>**.

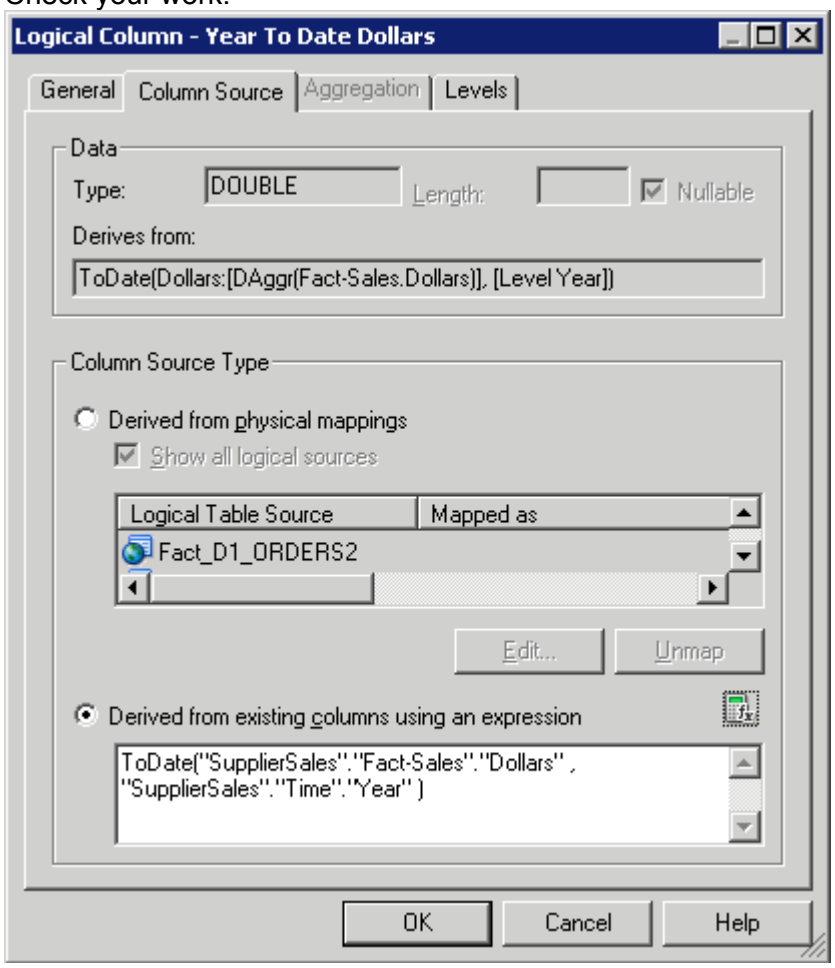
g. Select **Logical Tables > Fact-Sales** and then double-click **Dollars** to add it to the expression.

h. Click **<<Level>>**.

i. Select **Time Dimensions > Time** and then double-click **Year** to add it to the expression.

j. Click **OK** to close the Expression Builder.

k. Check your work.



l. Click **OK** to close the Logical Column dialog box.

m. Drag the **Year To Date Dollars** logical column to the Fact-Sales presentation folder.

9. Create a measure that calculates a three period rolling sum for dollars using the PERIODROLLING function.

a. Right-click **Fact-Sales** and select **New Object > Logical Column**.

b. On the General tab, name the logical column **Dollars 3-Period Rolling Sum**.

c. On the Column Source tab, select **Derived from existing columns using an expression**.

d. Open the Expression Builder.

e. Select **Functions > Time Series Functions** and then double-click **PeriodRolling** to insert the expression.

f. Click **<<Measure>>**.

g. Select **Logical Tables > Fact-Sales** and then double-click **Dollars** to add it to the expression.

h. Click **<<Starting Period Offset>>**.

i. Enter **-2**. This identifies the first period in the rolling aggregation.

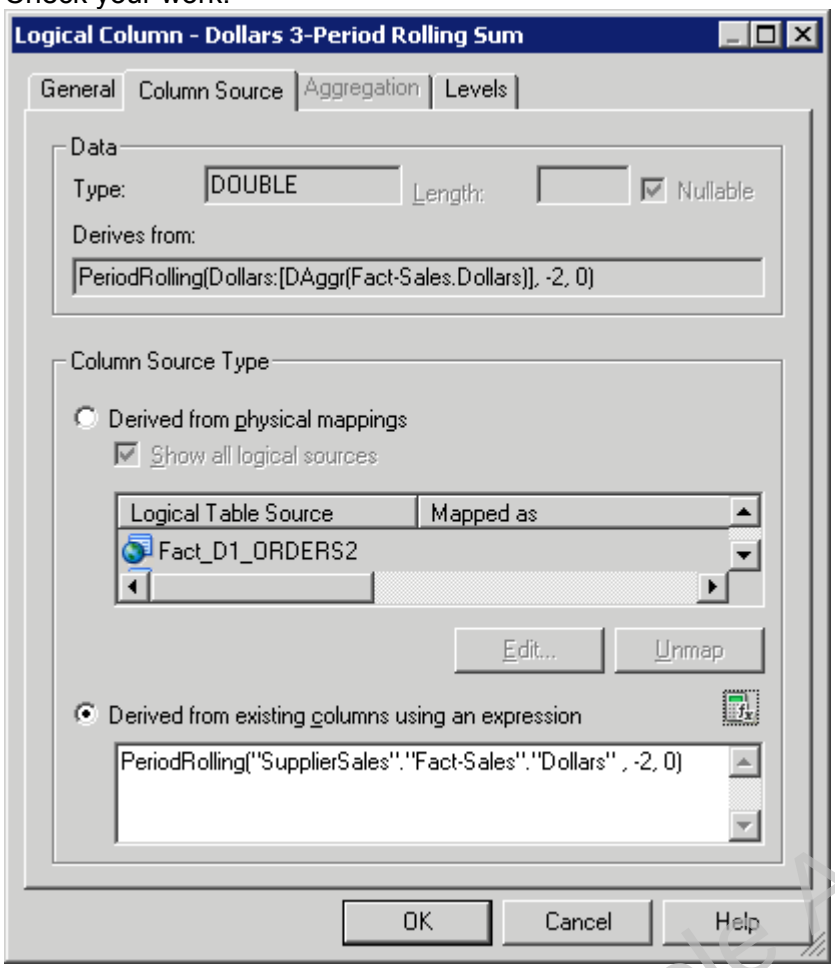
j. Click **<<Ending Period Offset>>**.

k. Enter **0**. This identifies the last period in the rolling aggregation. These integers are the relative number of periods from a displayed period. In this example, if the query grain is

month, the three-month rolling sum starts two months in the past (-2) and includes the current month (0).

l. Click **OK** to close the Expression Builder.

m. Check your work.



n. Click **OK** to close the Logical Column dialog box.

o. Drag the **Dollars 3-Period Rolling Sum** logical column to the Fact-Sales presentation folder.

10. Create a measure that calculates a three-period rolling average for dollars.

a. Right-click **Fact-Sales** and select **New Object > Logical Column**.

b. On the General tab, name the logical column **Dollars 3-Period Rolling Avg**.

c. On the Column Source tab, select **Derived from existing columns using an expression**.

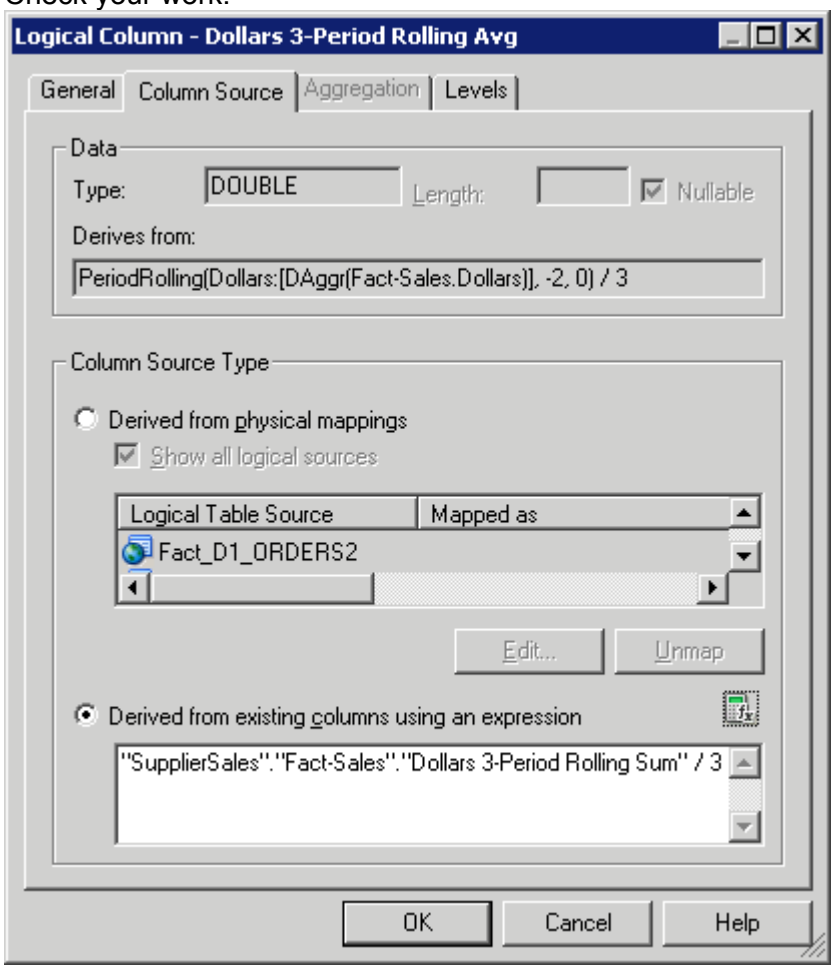
d. Open the Expression Builder.

e. Select **Logical Tables > Fact-Sales** and then double-click **Dollars 3-Period Rolling Sum** to add it to the expression.

f. Divide the column by **3**.

g. Click **OK** to close the Expression Builder.

h. Check your work.



i. Click **OK** to close the Logical Column dialog box.

j. Drag the **Dollars 3-Period Rolling Avg** logical column to the Fact-Sales presentation folder.

k. Save the repository.

l. Check consistency. Fix errors or warnings, if any, before you proceed.

m. Close the repository.

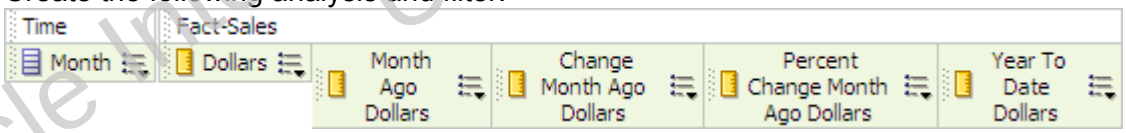
n. Leave the Administration Tool open.

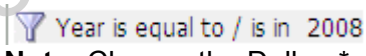
11. Create analyses to check your work for the AGO and TODATE functions.

a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.

b. Sign in to Analysis Editor as **weblogic/welcome1**.

c. Create the following analysis and filter:





Note: Change the Dollars* columns data format to currency and the Percentage column data format to percentage.

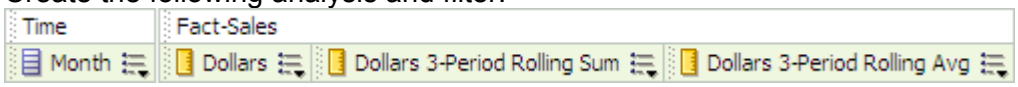
d. Click **Results**.

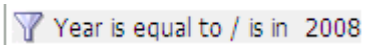
Month	Dollars	Month Ago Dollars	Change Month Ago Dollars	Percent Change Month Ago Dollars	Year To Date Dollars
January	$3,595,669				$3,595,669
February	$3,945,187	$3,595,669	$349,518	9.72%	$7,540,856
March	$3,975,774	$3,945,187	$30,586	0.78%	$11,516,630
April	$3,907,292	$3,975,774	-$68,482	-1.72%	$15,423,922
May	$4,061,558	$3,907,292	$154,266	3.95%	$19,485,480
June	$3,994,531	$4,061,558	-$67,027	-1.65%	$23,480,010
July	$4,062,212	$3,994,531	$67,681	1.69%	$27,542,222
August	$4,242,611	$4,062,212	$180,399	4.44%	$31,784,833
September	$3,810,263	$4,242,611	-$432,348	-10.19%	$35,595,096
October	$4,596,372	$3,810,263	$786,109	20.63%	$40,191,468
November	$3,655,169	$4,596,372	-$941,203	-20.48%	$43,846,637
December	$3,997,616	$3,655,169	$342,448	9.37%	$47,844,253

e. Note that **Month** is sorted automatically based on the MonthCode sort-order column that you set in the repository.

12. Create analyses to check your work for the PERIODROLLING function.

a. Create the following analysis and filter:

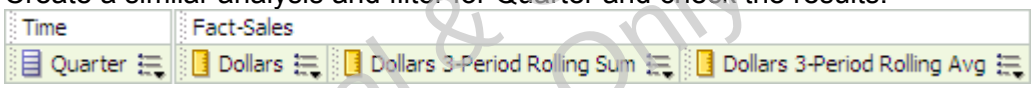


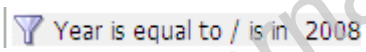


b. Click **Results**:

Month	Dollars	Dollars 3-Period Rolling Sum	Dollars 3-Period Rolling Avg
January	$3,595,669	$3,595,669	$1,198,556
February	$3,945,187	$7,540,856	$2,513,619
March	$3,975,774	$11,516,630	$3,838,877
April	$3,907,292	$11,828,252	$3,942,751
May	$4,061,558	$11,944,623	$3,981,541
June	$3,994,531	$11,963,380	$3,987,793
July	$4,062,212	$12,118,301	$4,039,434
August	$4,242,611	$12,299,354	$4,099,785
September	$3,810,263	$12,115,086	$4,038,362
October	$4,596,372	$12,649,246	$4,216,415
November	$3,655,169	$12,061,804	$4,020,601
December	$3,997,616	$12,249,157	$4,083,052

c. Create a similar analysis and filter for Quarter and check the results.





Quarter	Dollars	Dollars 3-Period Rolling Sum	Dollars 3-Period Rolling Avg
1	$11,516,630	$11,516,630	$3,838,877
2	$11,963,380	$23,480,010	$7,826,670
3	$12,115,086	$35,595,096	$11,865,032
4	$12,249,157	$36,327,623	$12,109,208

The PERIODROLLING measures display results based on the query grain.

d. Sign out of Oracle BI.

Practices for Lesson 14: Modeling Many-to-Many Relationships

Lesson 14

Practices for Lesson 14

Lesson Overview

In these practices, you will model many-to-many relationships in an Oracle BI repository.

Practice 14-1: Modeling a Bridge Table

Goal

To configure a bridge table for commission data

Scenario

An ABC sales representative may participate in many deals that pay commission. Additionally, each deal may include many sales representatives so that each sales representative receives a percentage of the commission. You must model this many-to-many relationship in the repository.

There is a D1_COMMISSION fact table with commissions paid per invoice. D1_COMMISION_BRIDGE is the bridge table used to create a many-to-many relationship between the D1_COMMISSION fact table and the D1_SALESREP dimension table. D1_COMMISION_BRIDGE includes a weight factor to calculate the weighted distribution of commissions among sales teams. You import the tables and model them in the repository.

Time

25 minutes

Tasks

1. Import tables into the repository.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. In the Physical layer, expand **orcl**.
   c. Right-click the **SUPPLIER CP** connection pool and select **Import Metadata**.
   d. Accept the defaults in the Select Metadata Types screen and click **Next** to open the Select Metadata Objects screen.
   e. Scroll to the **SUPPLIER2** schema and expand it.
   f. In the **Data source view** pane select the **D1_COMMISSION**, **D1_COMMISSION_BRIDGE,** and **D1_SALESREPS** tables for import.
   g. Click the **Import selected** button to add the tables to the Repository View pane.
   h. Expand **SUPPLIER2** in the Repository View pane and confirm that tables are added.
   i. Click **Finish** to add the tables to the repository.
   j. In the Physical layer, expand **orcl > SUPPLIER2** and ensure that the tables are visible.
   k. Update row counts on each table to check connectivity. Both commission tables should return **38** rows. D1_SALESREPS should have **34** rows.
   l. View data for **D1_COMMISSION**. Notice that COMM_KEY 1111 has two rows, each with a commission amount of $22,000 for a total commission amount of $44,000. This is an example of a situation where commissions may be paid in installments (for example, 50% of the commission is paid when the item is ordered, and the other 50% is paid when the item is paid for). Notice that CUST_KEY 1118 is associated with these two rows. You use this information to check your work when you run analyses later in this practice.
   m. View data for **D1_COMMISSION_BRIDGE**. Notice that Alan Ziff shares commission with Andrew Taylor for COMM_KEY 1111. Each earns 50% of the commission (0.5 weight factor). Since the total commission for COMM_KEY 1111 is $44,000, each

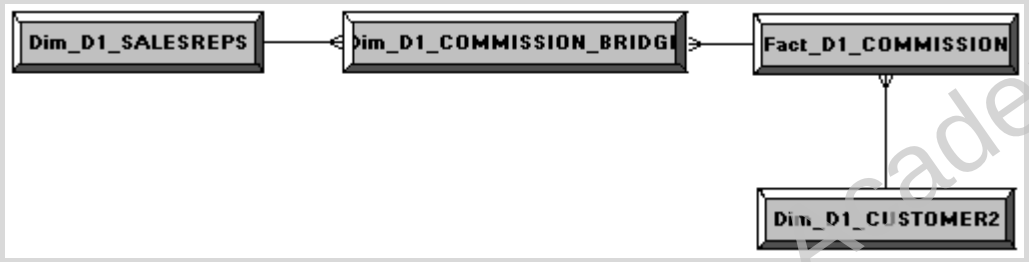
sales representative earns $22,000 for this deal. You use this information to check your work later in this practice.

n. Create the following aliases:
D1_COMMISSION = **Fact_D1_COMMISSION**
D1_COMMISSION_BRIDGE = **Dim_D1_COMMISSION_BRIDGE**
D1_SALESREPS = **Dim_D1_SALESREPS**

2. Create the physical joins.
   a. Select the following tables:
   Dim_D1_COMMISSION_BRIDGE
   Dim_D1_CUSTOMER2
   Dim_D1_SALESREPS
   Fact_D1_COMMISSION
   b. Right-click any one of the highlighted tables and select **Physical Diagram > Selected Object(s) Only**.
   c. Create the following foreign key joins:

```
Dim_D1_CUSTOMER2.NEWKEY = Fact_D1_COMMISSION.CUST_KEY
Dim_D1_SALESREPS.SALESREP = Dim_D1_COMMISSION_BRIDGE.SALESREP
Fact_D1_COMMISSION.COMM_KEY = Dim_D1_COMMISSION_BRIDGE.COMM_KEY
```

   d. Notice the one-to-many relationships between Fact_D1_COMMISSION, Dim_D1_SALESREPS and the Dim_D1_COMMISSION_BRIDGE bridge table.



   e. Close the Physical Diagram window.
3. Create the logical model.
   a. Right-click the **SupplierSales** business model and select **New Object > Logical Table**.
   b. Name the logical table **Fact-Commission** and click **OK**.
   c. From the Physical layer, drag **Fact_D1_COMMISSION.COMM_AMT** to **Fact-Commission** in the BMM layer. This action creates a logical table source named Fact_D1_COMMISSION and a logical column named COMM_AMT.
   d. Rename **COMM_AMT** to **Commission Amount**.
   e. Drag **Dim_D1_SALESREPS** from the Physical layer onto the SupplierSales business model in the Business Model and Mapping layer.
   f. Rename the **Dim_D1_SALESREPS** logical table to **Dim-SalesRep**.
   g. Rename the Dim-SalesRep logical columns:
   DISTRICT to **District**
   REGION to **Region**
   SALESREP to **Sales Rep**
   h. Delete the **SSN** logical column.
   i. Select the **Fact-Commission, Dim-Customer,** and **Dim-SalesRep** logical tables.

j. Right-click any of the highlighted tables and select **Business Model Diagram > Selected Tables Only**.

k. Create logical joins from the **Dim-Customer** and **Dim-SalesRep** logical tables to **Fact-Commission**.

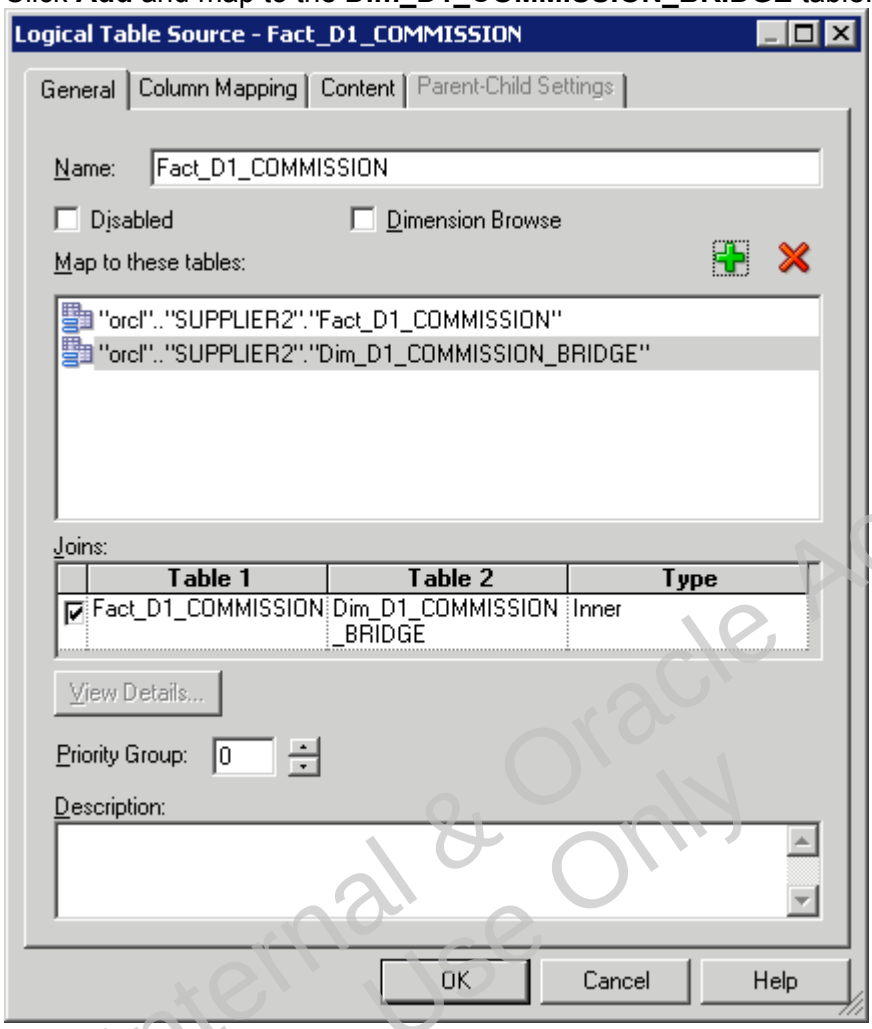
l. Close the **Business Model Diagram**.

4. Map the bridge table to the fact logical table source.

a. Expand **Fact-Commission > Sources** and double-click the **Fact_D1_COMMISSION** logical table source.

b. Click the **General** tab.

c. Click **Add** and map to the **Dim_D1_COMMISSION_BRIDGE** table.



5. Create a calculation measure using commission and weight factor physical columns.

a. Click the **Column Mapping** tab.

b. Select the **Commission Amount** logical column and open Expression Builder.

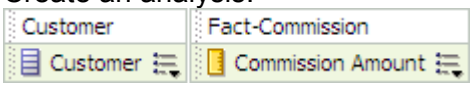
c. Build the following expression:

```
"ORCL"."".."SUPPLIER2"."Fact_D1_COMMISSION"."COMM_AMT" *
"ORCL"."".."SUPPLIER2"."Dim_D1_COMMISSION_BRIDGE"."WEIGHT_FACTOR"
```

d. Close the **Expression Builder**.

e. Close the **Logical Table Source** dialog box.
f. Set the **Commission Amount** aggregation rule to **SUM**.

6. Add objects to the Presentation layer.
   a. Drag the **Dim-SalesRep** and **Fact-Commission** tables onto **SupplierSales** in the Presentation layer.
   b. Rename **Dim-SalesRep** to **Sales Rep.**
   c. Reorder the presentation tables so that Sales Rep is above Fact-Sales.
   d. For testing purposes, drag **Dim-Customer.Customer Key** onto the **Customer** presentation table.
   e. Save the repository.
   f. Check consistency. Fix any errors or warnings before proceeding.
   g. Close the repository.
   h. Leave the Administration Tool open.
7. Create and run an analysis to check your work.
   a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.
   b. Return to Analysis Editor and sign in as **weblogic/welcome1**.
      a. Create an analysis:



      b. Click **Results**. The results show total commissions by customer.

Customer	Commission Amount
Arloi Dee	$50,000
B L T's Cobblefish	$12,800
Big River Grille & Brewing	$9,000
Billy's Hickory-Pit Bar-B-Q	$1,800
Caesar's Frozen Custard	$11,600
Demos' Steak & Spaghetti House	$12,000
Felix	$68,000
Half-Shell Restaurant	$47,800
Heaping Bowl & Brew	$48,600
Il Gargano Italian Trattoria	$56,000
Little Sichuan Restaurant	$49,000
Mayflower Cuisinier	$44,000
Rice Bowl	$38,680
Royal Lunch	$46,000
Tom's Famous Recipe Chicken	$9,600

      c. Double-click **Customer.Customer Key** to add it to the results. Notice that Mayflower Cuisinier is the customer associated with customer key 1118. Recall that earlier in the practice you examined the source data to determine that the total commission associated with customer key 1118 is equal to $44,000.
      d. Double-click **Sales Rep.Sales Rep** to add it to the results. The results show commission earned by each sales representative calculated by the weight factor in the bridge table.
      e. Click the **Display Maximum** button and confirm that Alan Ziff shares commission with Andrew Taylor for Mayflower Cuisinier. Each earned $22,000, 50% of the total commission, which matches the source data that you examined earlier in this

practice.

Mayflower Cuisinier	$22,000	1118	ALAN ZIFF
	$22,000		ANDREW TAYLOR

f. Check the log file and confirm that the Fact_D1_COMMISSION fact table and Dim_D1_COMMISSION_BRIDGE bridge table were both accessed.

```
---------- Sending query to database named orcl (id: <<3691>>),
connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T1051.COMM_AMT * T1057.WEIGHT_FACTOR) as
c1,
     T76.NEWKEY as c2,
     T76.NAME as c3,
     T1061.SALESREP as c4
from
     D1_SALESREPS T1061 /* Dim_D1_SALESREPS */ ,
     D1_CUSTOMER2 T76 /* Dim_D1_CUSTOMER2 */ ,
     D1_COMMISSION T1051 /* Fact_D1_COMMISSION */ ,
     D1_COMMISSION_BRIDGE T1057 /* Dim_D1_COMMISSION_BRIDGE */
where  ( T76.NEWKEY = T1051.CUST_KEY and T1051.COMM_KEY =
T1057.COMM_KEY and T1057.SALESREP = T1061.SALESREP )
group by T76.NAME, T76.NEWKEY, T1061.SALESREP)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c3 as c3,
     D1.c4 as c4,
     D1.c1 as c5
from
     SAWITH0 D1
order by c3, c2 NULLS FIRST, c4
```

g. Close the log file

h. Sign out of Oracle BI.

Practices for Lesson 15: Localizing Oracle BI Metadata

Lesson 15

Practices for Lesson 15

Lesson Overview

In these practices, you will localize Oracle BI repository metadata.

Practice 15-1: Localizing Repository Metadata

Goal

To localize Oracle BI repository metadata

Scenario

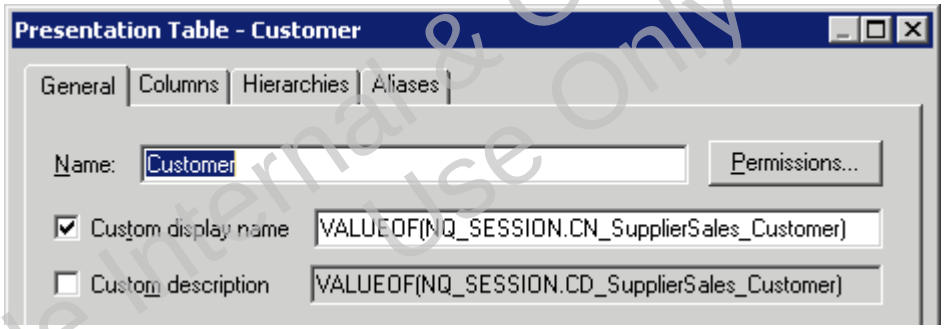
You want to display analysis metadata to users in languages other than English. To do this, you must perform a series of steps. First, you run the Externalize Strings repository utility to export Presentation layer object strings to a file in comma-separated format. You then modify the file to include translated strings. You then load the data from the file into a database table, create initialization blocks to populate session variables, modify localization preferences, and check the results by creating and running analyses.

Time

40 minutes

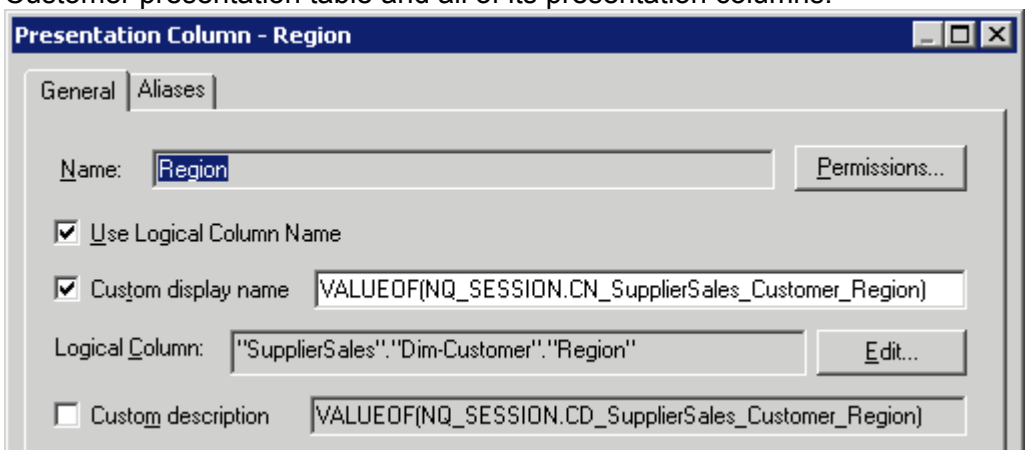
Tasks

1. Run the Externalize String utility to export Presentation layer objects to a spreadsheet. You can use the Externalize Strings utility to localize the names of Presentation layer subject areas, tables, hierarchies, columns, and their descriptions. You can save these text strings to an external file with ANSI, Unicode, and UTF-8 coding options.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. In the Presentation layer, double-click the **Customer** presentation table and notice that "Custom display name" is not checked on the General tab.
   c. Click **Cancel** to close the Presentation Table dialog box.
   d. Right-click the **Customer** presentation table and select **Externalize Display Names** > **Generate Custom Name**. Notice also that there is an option to Externalize Descriptions. Leave that unselected.
   e. Double-click the **Customer** presentation table again. On the **General** tab, notice that "Custom display name" is now checked. The name that appears in analyses will now be provided by the CN_SupplierSales_Customer session variable. Later in this practice you create a table and initialization block to populate this and other Presentation layer session variables.



   f. Click **Cancel** to close the Presentation Table dialog box.
   g. Expand the **Customer** presentation table and double-click one of the presentation columns to open the Presentation Column dialog box. Notice, on the General tab, that "Custom display name" is now selected for the column as well. Choosing one of the two right-click externalization options automatically selects the "Custom display name"

or “Custom description” options in the Properties dialog box for the selected object and all of its child objects. In this case, “Custom display name” is now selected for the Customer presentation table and all of its presentation columns.



h. Close the Presentation Column dialog box.

i. Select **Tools > Utilities**.

j. Select **Externalize Strings** and click **Execute**.

k. Select **SupplierSales** in the left pane.

l. In the right pane, notice that there is a row for each metadata object with a description of the object, the corresponding session variable, and the original name of the object. For the purpose of this exercise, you are running the Externalize Strings utility for only the Customer presentation table columns.

Presentation Table Customer	CN_SupplierSales_Customer	Customer
Presentation Column Region	CN_SupplierSales_Customer_Region	Region
Presentation Column Sales District	CN_SupplierSales_Customer_Sales_District	Sales District
Presentation Column Sales Rep	CN_SupplierSales_Customer_Sales_Rep	Sales Rep
Presentation Column Customer	CN_SupplierSales_Customer_Customer	Customer
Presentation Column Address	CN_SupplierSales_Customer_Address	Address
Presentation Column Phone	CN_SupplierSales_Customer_Phone	Phone
Presentation Column City	CN_SupplierSales_Customer_City	City
Presentation Column State	CN_SupplierSales_Customer_State	State
Presentation Column Zip Code	CN_SupplierSales_Customer_Zip_Code	Zip Code
Presentation Column Route Code	CN_SupplierSales_Customer_Route_Code	Route Code
Presentation Column Customer Key	CN_SupplierSales_Customer_Customer_Key	Customer Key

m. Click **Save**.

n. Name the file **TRANSLATIONS** and click **Save** to save as a .csv file in the Repository folder.

o. Close the **Externalize Strings** dialog box.

p. Leave the repository open.

2. Modify the translation file.

a. Navigate to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository** and double-click **TRANSLATIONS.csv** to open it in Excel. The first column contains the actual repository object names, which have a prefix of their type: Presentation Table, Presentation Column, and so forth. The second column contains the session variables that correspond to each object’s name, with a default prefix of CN_. The third column contains the English translation of the object name is it appears in Analysis Editor. The fourth column contains the mapping to the logical

column in the BMM layer.

Presentation Table Customer	CN_SupplierSales_Customer	Customer
Presentation Hierarchy Customer - Sales Rep	CN_SupplierSales_Customer_Customer_-_Sales_Rep	Customer - Sales Rep
Presentation Level Customer Total	CN_Customer_Customer_-_Sales_Rep_Customer_Total	Customer Total
Presentation Level Sales Rep	CN_Customer_Customer_-_Sales_Rep_Sales_Rep	Sales Rep
Presentation Level Customer Detail	CN_Customer_Customer_-_Sales_Rep_Customer_Detail	Customer Detail
Presentation Hierarchy Customer - Region	CN_SupplierSales_Customer_Customer_-_Region	Customer - Region
Presentation Level Customer Total	CN_Customer_Customer_-_Region_Customer_Total	Customer Total
Presentation Level Region	CN_Customer_Customer_-_Region_Region	Region
Presentation Level District	CN_Customer_Customer_-_Region_District	District
Presentation Level Sales Rep	CN_Customer_Customer_-_Region_Sales_Rep	Sales Rep
Presentation Level Customer Detail	CN_Customer_Customer_-_Region_Customer_Detail	Customer Detail
Presentation Column Region	CN_SupplierSales_Customer_Region	Region
Presentation Column Sales District	CN_SupplierSales_Customer_Sales_District	Sales District
Presentation Column Sales Rep	CN_SupplierSales_Customer_Sales_Rep	Sales Rep
Presentation Column Customer	CN_SupplierSales_Customer_Customer	Customer
Presentation Column Address	CN_SupplierSales_Customer_Address	Address
Presentation Column Phone	CN_SupplierSales_Customer_Phone	Phone
Presentation Column City	CN_SupplierSales_Customer_City	City
Presentation Column State	CN_SupplierSales_Customer_State	State
Presentation Column Zip Code	CN_SupplierSales_Customer_Zip_Code	Zip Code
Presentation Column Route Code	CN_SupplierSales_Customer_Route_Code	Route Code
Presentation Column Customer Key	CN_SupplierSales_Customer_Customer_Key	Customer Key

b. Right-click **row 1** and select **Insert** to insert a header row at the top of the spreadsheet.

c. Enter the following headings for the three columns: **METADATA_OBJECT, MSG_NUM,** and **MSG_TEXT**.

METADATA_OBJECT	MSG_NUM	MSG_TEXT
Presentation Table Customer	CN_SupplierSales_Customer	Customer
Presentation Hierarchy Customer - Sales Rep	CN_SupplierSales_Customer_Customer_-_Sales_Rep	Customer - Sales Rep
Presentation Level Customer Total	CN_Customer_Customer_-_Sales_Rep_Customer_Total	Customer Total
Presentation Level Sales Rep	CN_Customer_Customer_-_Sales_Rep_Sales_Rep	Sales Rep
Presentation Level Customer Detail	CN_Customer_Customer_-_Sales_Rep_Customer_Detail	Customer Detail
Presentation Hierarchy Customer - Region	CN_SupplierSales_Customer_Customer_-_Region	Customer - Region

d. Insert a fourth column after **MSG_TEXT** and enter **LANG_ID** as the column heading.

METADATA_OBJECT	MSG_NUM	MSG_TEXT	LANG_ID
Presentation Table Customer	CN_SupplierSales_Customer	Customer	
Presentation Hierarchy Customer - Sa	CN_SupplierSales_Customer_Customer_-_Sale	Customer - Sales Rep	
Presentation Level Customer Total	CN_Customer_Customer_-_Sales_Rep_Custom	Customer Total	
Presentation Level Sales Rep	CN_Customer_Customer_-_Sales_Rep_Sales_R	Sales Rep	
Presentation Level Customer Detail	CN_Customer_Customer_-_Sales_Rep_Custom	Customer Detail	
Presentation Hierarchy Customer - R	CN_SupplierSales_Customer_Customer_-_Regi	Customer - Region	
Presentation Level Customer Total	CN_Customer_Customer_-_Region_Customer_T	Customer Total	
Presentation Level Region	CN_Customer_Customer_-_Region_Region	Region	
Presentation Level District	CN_Customer_Customer_-_Region_District	District	
Presentation Level Sales Rep	CN_Customer_Customer_-_Region_Sales_Rep	Sales Rep	
Presentation Level Customer Detail	CN_Customer_Customer_-_Region_Customer_I	Customer Detail	
Presentation Column Region	CN_SupplierSales_Customer_Region	Region	
Presentation Column Sales District	CN_SupplierSales_Customer_Sales_District	Sales District	
Presentation Column Sales Rep	CN_SupplierSales_Customer_Sales_Rep	Sales Rep	
Presentation Column Customer	CN_SupplierSales_Customer_Customer	Customer	
Presentation Column Address	CN_SupplierSales_Customer_Address	Address	
Presentation Column Phone	CN_SupplierSales_Customer_Phone	Phone	
Presentation Column City	CN_SupplierSales_Customer_City	City	
Presentation Column State	CN_SupplierSales_Customer_State	State	
Presentation Column Zip Code	CN_SupplierSales_Customer_Zip_Code	Zip Code	
Presentation Column Route Code	CN_SupplierSales_Customer_Route_Code	Route Code	
Presentation Column Customer Key	CN_SupplierSales_Customer_Customer_Key	Customer Key	

e. Delete all of the data rows *except* for the **Presentation Column** rows.

METADATA_OBJECT	MSG_NUM	MSG_TEXT	LANG_ID
Presentation Column Region	CN_SupplierSales_Customer_Region	Region	
Presentation Column Sales District	CN_SupplierSales_Customer_Sales_District	Sales District	
Presentation Column Sales Rep	CN_SupplierSales_Customer_Sales_Rep	Sales Rep	
Presentation Column Customer	CN_SupplierSales_Customer_Customer	Customer	
Presentation Column Address	CN_SupplierSales_Customer_Address	Address	
Presentation Column Phone	CN_SupplierSales_Customer_Phone	Phone	
Presentation Column City	CN_SupplierSales_Customer_City	City	
Presentation Column State	CN_SupplierSales_Customer_State	State	
Presentation Column Zip Code	CN_SupplierSales_Customer_Zip_Code	Zip Code	
Presentation Column Route Code	CN_SupplierSales_Customer_Route_Code	Route Code	

f. Right-click the data row for **Presentation Column Region** and select **Copy**. Be sure to select the entire row, not just a cell.

g. Right-click the row immediately below **Presentation Column Region** and select **Insert Copied Cells** to create a duplicate row.

h. Repeat to create duplicate rows for all of the metadata objects.

METADATA_OBJECT	MSG_NUM	MSG_TEXT	LANG_ID
Presentation Column Region	CN_SupplierSales_Customer_Region	Region	
Presentation Column Region	CN_SupplierSales_Customer_Region	Region	
Presentation Column Sales District	CN_SupplierSales_Customer_Sales_District	Sales District	
Presentation Column Sales District	CN_SupplierSales_Customer_Sales_District	Sales District	
Presentation Column Sales Rep	CN_SupplierSales_Customer_Sales_Rep	Sales Rep	
Presentation Column Sales Rep	CN_SupplierSales_Customer_Sales_Rep	Sales Rep	
Presentation Column Customer	CN_SupplierSales_Customer_Customer	Customer	
Presentation Column Customer	CN_SupplierSales_Customer_Customer	Customer	
Presentation Column Address	CN_SupplierSales_Customer_Address	Address	
Presentation Column Address	CN_SupplierSales_Customer_Address	Address	
Presentation Column Phone	CN_SupplierSales_Customer_Phone	Phone	
Presentation Column Phone	CN_SupplierSales_Customer_Phone	Phone	
Presentation Column City	CN_SupplierSales_Customer_City	City	
Presentation Column City	CN_SupplierSales_Customer_City	City	
Presentation Column State	CN_SupplierSales_Customer_State	State	
Presentation Column State	CN_SupplierSales_Customer_State	State	
Presentation Column Zip Code	CN_SupplierSales_Customer_Zip_Code	Zip Code	
Presentation Column Zip Code	CN_SupplierSales_Customer_Zip_Code	Zip Code	
Presentation Column Route Code	CN_SupplierSales_Customer_Route_Code	Route Code	
Presentation Column Route Code	CN_SupplierSales_Customer_Route_Code	Route Code	

i. Enter **en** in the LANG_ID column for the first **Presentation Column Region** row to identify this row as the English translation.

j. Enter **fr** in the LANG_ID column for the second **Presentation Column Region** row to identify this row as the French translation.

k. Enter **Region** in the MSG_TEXT column for the second Presentation Table Customer row to provide the French translation for Customer.

METADATA_OBJECT	MSG_NUM	MSG_TEXT	LANG_ID
Presentation Column Region	CN_SupplierSales_Customer_Region	Region	en
Presentation Column Region	CN_SupplierSales_Customer_Region	Region	fr

l. Repeat to add French translations for the remaining metadata objects. Use the following screenshot as a guide. The translations do not have to exactly match the screenshot. Just be sure that every cell has a value. For the purpose of this exercise,

you are providing translations only for the Customer presentation table columns.

METADATA_OBJECT	MSG_NUM	MSG_TEXT	LANG_ID
Presentation Column Region	CN_SupplierSales_Customer_Region	Region	en
Presentation Column Region	CN_SupplierSales_Customer_Region	Region	fr
Presentation Column Sales District	CN_SupplierSales_Customer_Sales_District	Sales District	en
Presentation Column Sales District	CN_SupplierSales_Customer_Sales_District	Les ventes de district	fr
Presentation Column Sales Rep	CN_SupplierSales_Customer_Sales_Rep	Sales Rep	en
Presentation Column Sales Rep	CN_SupplierSales_Customer_Sales_Rep	Représentant des ventes	fr
Presentation Column Customer	CN_SupplierSales_Customer_Customer	Customer	en
Presentation Column Customer	CN_SupplierSales_Customer_Customer	Client	fr
Presentation Column Address	CN_SupplierSales_Customer_Address	Address	en
Presentation Column Address	CN_SupplierSales_Customer_Address	Adresse	fr
Presentation Column Phone	CN_SupplierSales_Customer_Phone	Phone	en
Presentation Column Phone	CN_SupplierSales_Customer_Phone	Téléphone	fr
Presentation Column City	CN_SupplierSales_Customer_City	City	en
Presentation Column City	CN_SupplierSales_Customer_City	Ville	fr
Presentation Column State	CN_SupplierSales_Customer_State	State	en
Presentation Column State	CN_SupplierSales_Customer_State	État	fr
Presentation Column Zip Code	CN_SupplierSales_Customer_Zip_Code	Zip Code	en
Presentation Column Zip Code	CN_SupplierSales_Customer_Zip_Code	Code postal	fr
Presentation Column Route Code	CN_SupplierSales_Customer_Route_Code	Route Code	en
Presentation Column Route Code	CN_SupplierSales_Customer_Route_Code	Route Code	fr

m. Delete the **header row** from the spreadsheet.

n. Save the file as **TRANSLATIONS.csv** to **D:\PracticeFiles**.

o. Close the file and exit Excel.

3. In this step, you use SQL Loader to load the .csv file into an existing table. A table named TRANSLATIONS has already been created for you in the SUPPLIER2 schema.

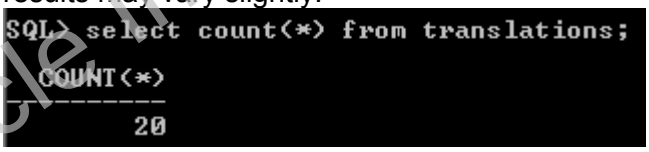
a. Open a command window.

b. Change the directory to **D:\PracticeFiles.**

c. Enter **translations.bat** and press **Enter** to run the batch file. This batch file starts the SQL Loader utility and uses a control file to load the data from the TRANSLATIONS.csv file into the TRANSLATIONS table in the database.

d. Close the command window.

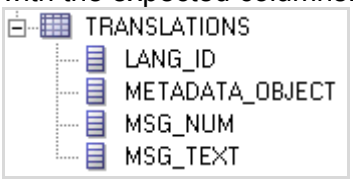
e. Open **SQL*Plus** and confirm that there are 20 rows in the TRANSLATIONS table. Your results may vary slightly.



f. Close SQL*Plus.

4. Import the **TRANSLATIONS** table into the Physical layer.

a. Return to the ABC repository.
b. In the Physical layer, expand **orcl**.
c. Right-click the **SUPPLIER CP** connection pool and select **Import Metadata**.
d. Accept the defaults in the Select Metadata Types window and click **Next** to open the Select Metadata Objects window.
e. Scroll to the **SUPPLIER2** schema and expand it.
f. In the **Data source view** pane select the **TRANSLATIONS** table for import.
g. Click the **Import selected** button to add the table to the Repository View pane.
h. Expand **SUPPLIER2** in the Repository View pane and confirm that the table is added.
i. Click **Finish** to add the tables to the repository.
j. In the Physical layer, expand **orcl > SUPPLIER2** and confirm that the table is visible with the expected columns.



k. Update the row count for the TRANSLATIONS table to verify connectivity. There should be 20 rows. Your results may vary slightly.
l. View data for the TRANSLATIONS table. It should look similar to the following:

LANG_ID	METADATA_OBJECT	MSG_NUM	MSG_TEXT
en	Presentation Column Region	CN_SupplierSales_Customer_Region	Region
fr	Presentation Column Region	CN_SupplierSales_Customer_Region	Region
en	Presentation Column Sales District	CN_SupplierSales_Customer_Sales_District	Sales District
fr	Presentation Column Sales District	CN_SupplierSales_Customer_Sales_District	Les ventes de district
en	Presentation Column Sales Rep	CN_SupplierSales_Customer_Sales_Rep	Sales Rep
fr	Presentation Column Sales Rep	CN_SupplierSales_Customer_Sales_Rep	Représentant des ventes
en	Presentation Column Customer	CN_SupplierSales_Customer_Customer	Customer
fr	Presentation Column Customer	CN_SupplierSales_Customer_Customer	Client
en	Presentation Column Address	CN_SupplierSales_Customer_Address	Address
fr	Presentation Column Address	CN_SupplierSales_Customer_Address	Adresse
en	Presentation Column Phone	CN_SupplierSales_Customer_Phone	Phone
fr	Presentation Column Phone	CN_SupplierSales_Customer_Phone	Téléphone
en	Presentation Column City	CN_SupplierSales_Customer_City	City
fr	Presentation Column City	CN_SupplierSales_Customer_City	Ville
en	Presentation Column State	CN_SupplierSales_Customer_State	State
fr	Presentation Column State	CN_SupplierSales_Customer_State	État
en	Presentation Column Zip Code	CN_SupplierSales_Customer_Zip_Code	Zip Code
fr	Presentation Column Zip Code	CN_SupplierSales_Customer_Zip_Code	Code postal
en	Presentation Column Route Code	CN_SupplierSales_Customer_Route_Code	Route Code
fr	Presentation Column Route Code	CN_SupplierSales_Customer_Route_Code	Route Code

m. Save the repository.
n. Check consistency. Fix any errors or warnings before proceeding.

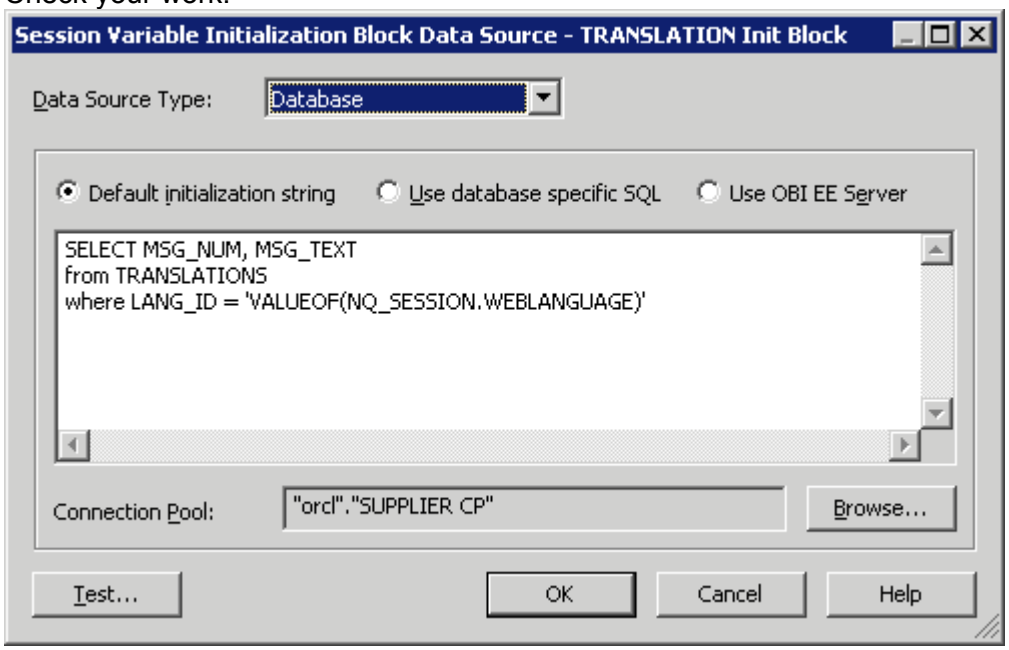
5. Create an initialization block to select the correct translations from the TRANSLATIONS table based on the LANGUAGE variable.
   a. Select **Manage > Variables** to open Variable Manager.
   b. Select **Action > New > Session > Initialization Block**.
   c. Name the initialization block **TRANSLATION Init Block**.
   d. Click **Edit Data Source**.
   e. In the Default Initialization String field, enter the following SELECT statement:
   `SELECT MSG_NUM, MSG_TEXT`

```
from TRANSLATIONS
where LANG_ID = 'VALUEOF(NQ_SESSION.WEBLANGUAGE)'
```

f. Click **Browse** to select the connection pool.

g. Double-click **SUPPLIER CP** to select it.

h. Check your work.

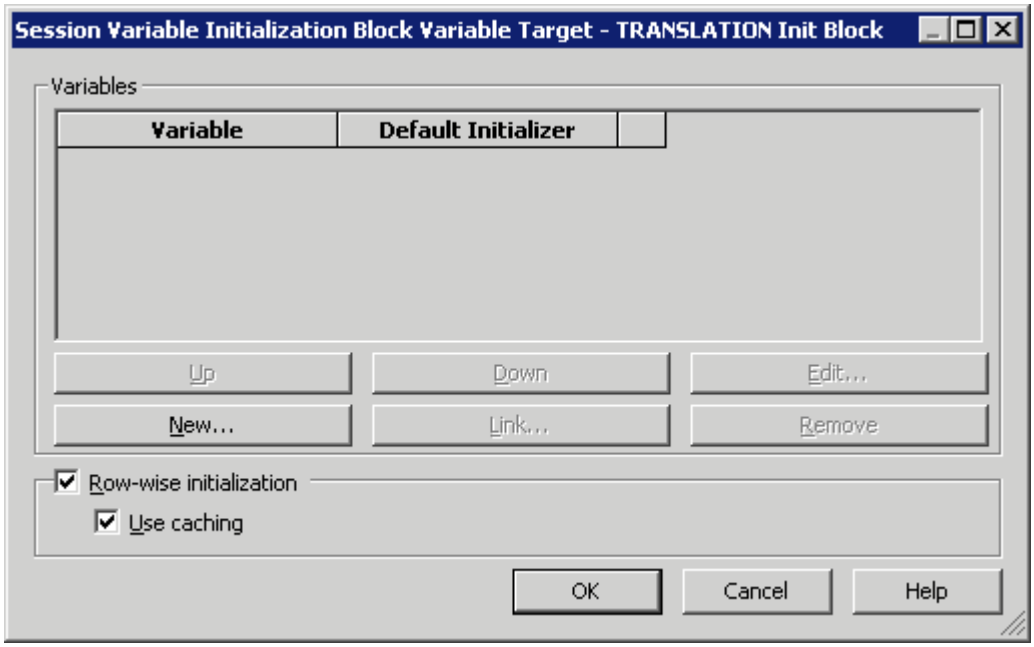


WEBLANGUAGE is a session variable that is initialized automatically, based on the language selected when a user logs in. WEBLANGUAGE is set to the language of the user's browser when a user first logs on to an integrated Oracle BI application. For example, if a user with a browser language set to French signs in to Oracle Business Intelligence for the first time, then the value for WEBLANGUAGE is French, and the metadata is translated to French. As you will see, users can also change their language and locale in My Account in Analysis Editor.

i. Click **OK** to close the Session Variable Initialization Block Data Source dialog box.

j. Click **Edit Data Target**.

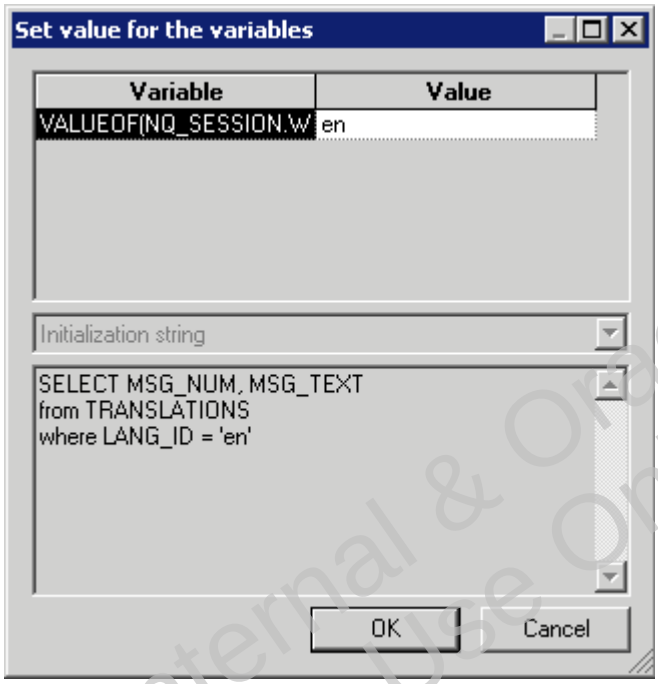
k. Select the **Row-wise initialization** check box to enable it.



l. Click **OK**.

m. Click **Test**.

n. Enter **en** in the Value field and click **OK**.



o. Verify that the rows with English translations are returned.

	1	2
0	CN_SupplierSales_Customer_Region	Region
1	CN_SupplierSales_Customer_Sales_District	Sales District
2	CN_SupplierSales_Customer_Sales_Rep	Sales Rep
3	CN_SupplierSales_Customer_Customer	Customer
4	CN_SupplierSales_Customer_Address	Address
5	CN_SupplierSales_Customer_Phone	Phone
6	CN_SupplierSales_Customer_City	City
7	CN_SupplierSales_Customer_State	State
8	CN_SupplierSales_Customer_Zip_Code	Zip Code
9	CN_SupplierSales_Customer_Route_Code	Route Code

p. Repeat for **fr** and verify that the rows with French translations are returned.

	1	2
0	CN_SupplierSales_Customer_Region	Region
1	CN_SupplierSales_Customer_Sales_District	Les ventes de district
2	CN_SupplierSales_Customer_Sales_Rep	Représentant des ventes
3	CN_SupplierSales_Customer_Customer	Client
4	CN_SupplierSales_Customer_Address	Adresse
5	CN_SupplierSales_Customer_Phone	Téléphone
6	CN_SupplierSales_Customer_City	Ville
7	CN_SupplierSales_Customer_State	État
8	CN_SupplierSales_Customer_Zip_Code	Code postal
9	CN_SupplierSales_Customer_Route_Code	Route Code

q. Click **OK** to close the Session Variable Initialization Block dialog box.

r. Select **Action > Close** to close the Variable Manager.

s. Save the repository.

t. Check consistency. Fix any errors or warnings before proceeding.

u. Close the repository.

v. Leave the Administration Tool open.

6. Sign in to Oracle BI to check your work.

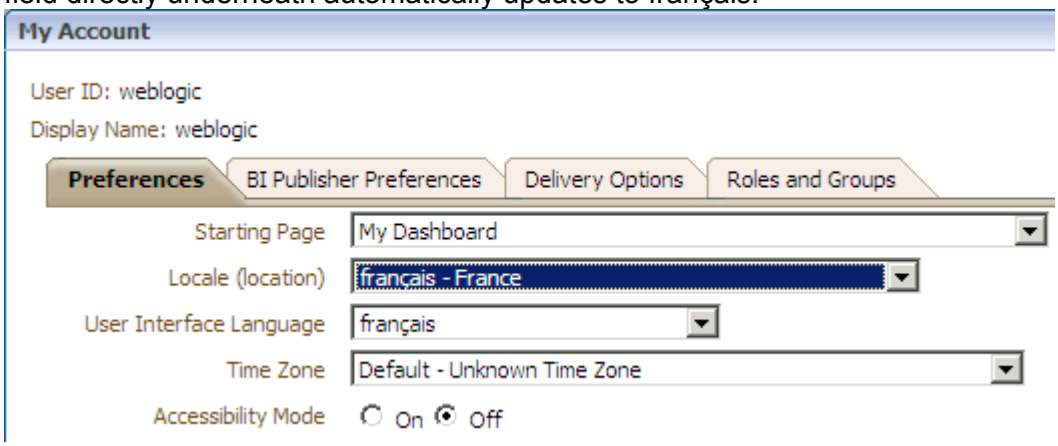
a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.

b. Sign in to Oracle BI as **weblogic/welcome1** with **English** selected.

c. Click **Home > Analysis > SupplierSales** to open Analysis Editor.

d. Click **weblogic > My Account** near the top-right corner of the window.

e. Set **Locale (location)** to **français – France**. Notice that the User Interface Language field directly underneath automatically updates to français.

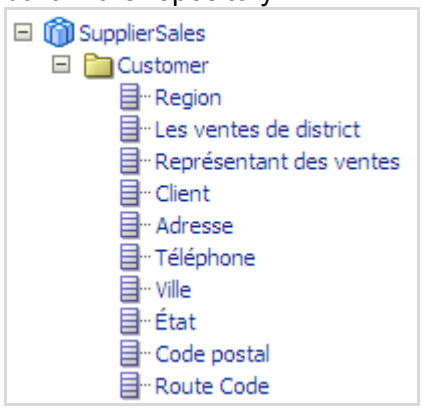


f. Click **OK**.

g. Click the **Refresh** button in the Web Browser.

h. Notice that tabs, links, messages, and so forth have change to French. This is a result of changing the Locale settings in My Account.

i. Notice also that the Customer table columns now display the French translations. This is a result of the translation table that you imported and the initialization block that you built in the repository.



j. Sign out of Oracle BI.

k. Navigate to **D:\bi\instances\instance1\diagnostics\logs\OracleBIServerComponent\coreapplication_obis1** and open the **NQQuery.log** file.

l. Scroll to the bottom of the file and verify that the initialization blocks issued the SQL query.

```
---------- An initialization block named 'TRANSLATION Init
Block', on behalf of a Session Variable, issued the following
SQL query: [[

    SELECT MSG_NUM, MSG_TEXT
from TRANSLATIONS
where LANG_ID = 'VALUEOF(NQ_SESSION.WEBLANGUAGE)'

Returned 10 rows.  Query status: Successful Completion
```

m. Close **NQQuery.log**.

Practices for Lesson 16: Localizing Oracle BI Data

Lesson 16

Practices for Lesson 16

Lesson Overview

In these practices, you will localize Oracle BI data.

Practice 16-1: Localizing Oracle BI Data

Goal

To localize product type data from English to French

Scenario

To localize product type data from English to French, you import the necessary localization tables, create an initialization block and session variable to establish a user-preferred language, create a lookup table in the business model, build an expression for the column being translated, and check results.

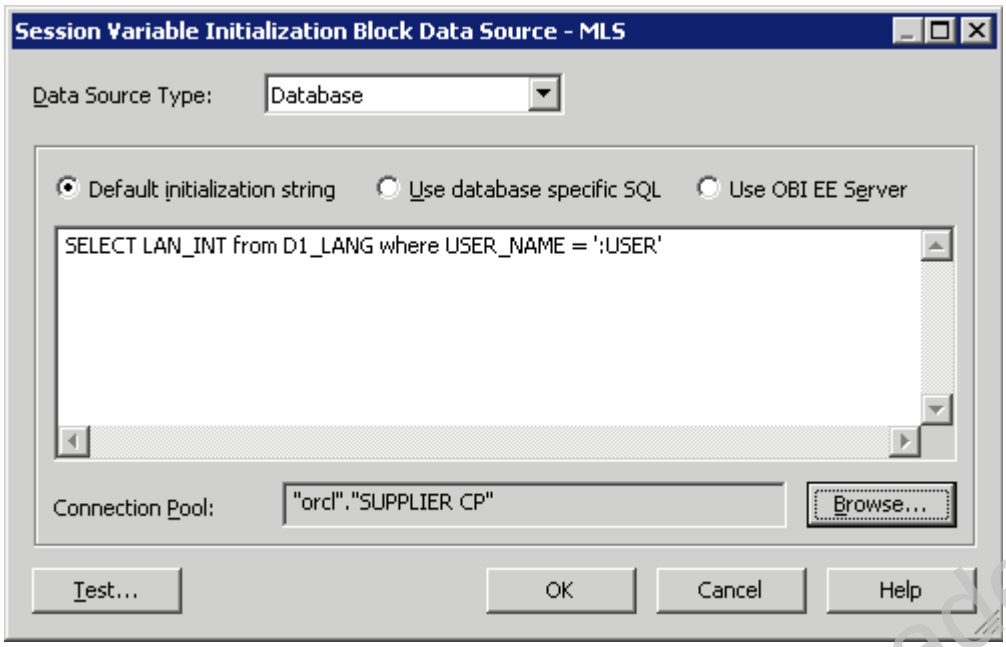
Time

35 minutes

Tasks

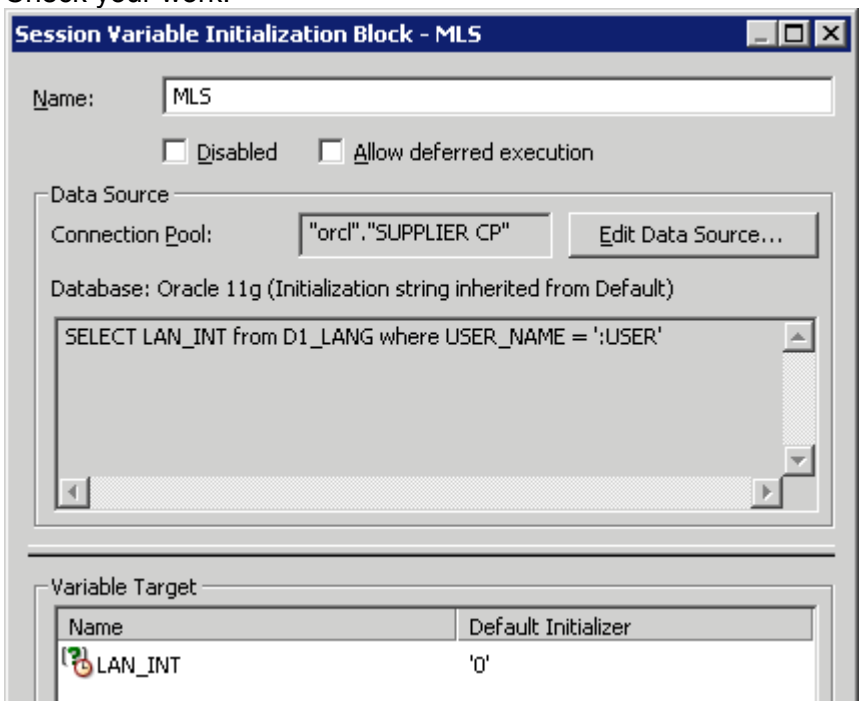
1. Import localization tables into the repository.
    a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
    b. In the Physical layer, expand **orcl**.
    c. Right-click the **SUPPLIER CP** connection pool and select **Import Metadata**.
    d. Accept the defaults in the Select Metadata Types window, and click **Next** to open the Select Metadata Objects screen.
    e. Scroll to the **SUPPLIER2** schema and expand it.
    f. In the **Data source view** pane select the **D1_PRODUCT_TYPE_TRANS** and **D1_LANG** tables for import.
    g. Click the **Import selected** button to add the tables to the Repository View pane.
    h. Expand **SUPPLIER2** in the Repository View pane and confirm that tables are added.
    i. Click **Finish** to add the tables to the repository.
    j. In the Physical layer, expand **orcl > SUPPLIER2** and ensure that the tables are visible.
    k. Update row count for the **D1_PRODUCT_TYPE_TRANS** table. You should see **42** rows.
    l. View data for the **D1_PRODUCT_TYPE_TRANS** table. This translation table contains a foreign key reference (TYPECODE) to records in the D1_PRODUCT_TYPE base table, and contains the values for each translated field in a separate column (ITEMTYPE). LANG_ID identifies the two digit language code for each row. Assuming a completely dense lookup table, the number of rows in the lookup table for a particular language is equal to the number of rows in the base table. In this example, there are rows for German (de) and French (fr) for each product type.
    m. View data for **D1_LANG**. D1_LANG contains one row for each language and a corresponding user. This table defines the language for a user when the user logs in to Oracle BI server. LAN_INT is set to 0 for English and set to 1 for all other languages. In the next step, you create an initialization block with a session variable that will be populated by values in this column.
2. Create an initialization block and session variable.
    a. Select **Manage > Variables** to open Variable Manager.
    b. Select **Action > New > Session > Initialization Block**.

c. Name the initialization block **MLS**.
d. Click **Edit Data Source**.
e. Leave Data Source Type set to **Database**.
f. Select **Default initialization string**.
g. Enter the following SQL in the field:
select LAN_INT from D1_LANG where user_name = ':USER'
h. Click **Browse** next to the Connection Pool field.
i. Double-click **SUPPLIER CP** to select it.



j. Click **OK** to close the Session Variable Initialization Block Data Source dialog box.
k. Click **Edit Data Target**.
l. Create a new variable named **LAN_INT**.
m. Set the default initializer to **'0'**.
n. Click **OK** to close the Session Variable dialog box.
o. Click **OK** to close the Session Variable Initialization Block Variable Target dialog box.

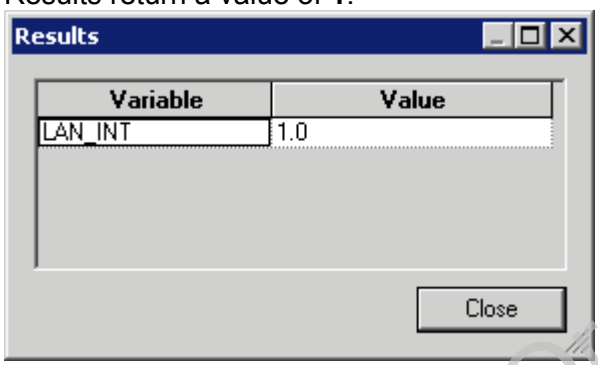
p. Check your work.



q. Click **Test**.

r. Enter **Administrator_f** in the Value field, and click **OK**.

s. Results return a value of **1**.



t. Repeat for **Administrator_d** and verify that a value of **1** is returned.

u. Repeat for **Administrator** and verify that a value of **0** is returned.

v. Click **OK** to close the Session Variable Initialization Block dialog box.

w. Right-click **TRANSLATION Init Block** and select **Disable**.

x. Close Variable Manager.

3. Create a logical lookup table.

a. Create an alias for **D1_PRODUCT_TYPE_TRANS** and name it **Dim_D1_PRODUCT_TYPE_TRANS**.

b. Drag **Dim_D1_PRODUCT_TYPE_TRANS** into the SupplierSales business model in the Business Model and Mapping layer.

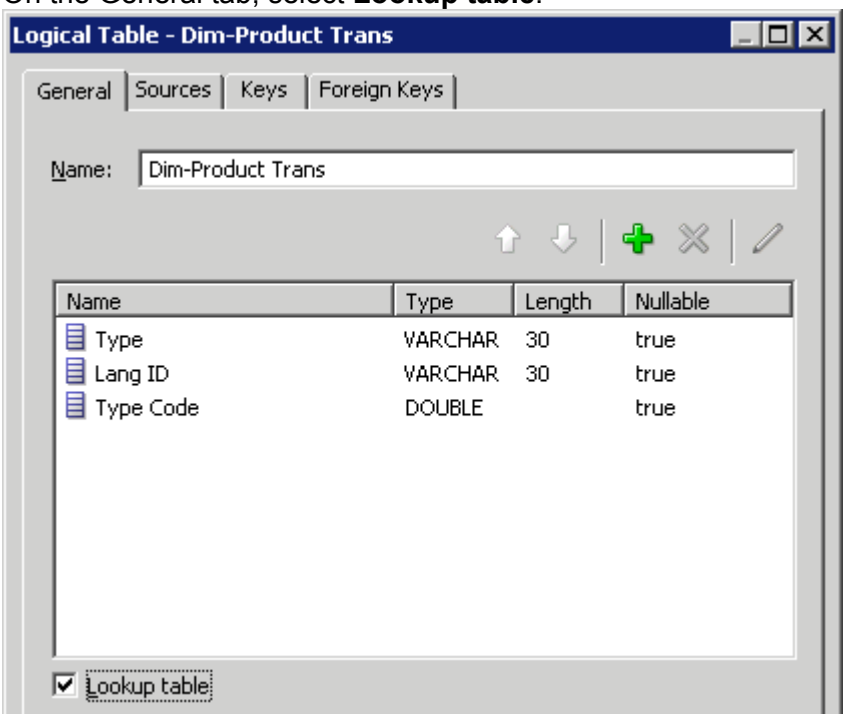
c. Rename the table **Dim-Product_Trans**.

d. Rename the logical columns:
ITEMTYPE to **Type**

LANG_ID to **Lang ID**
TYPECODE to **Type Code**

e. Double-click **Dim-Product_Trans** to open its properties dialog box.

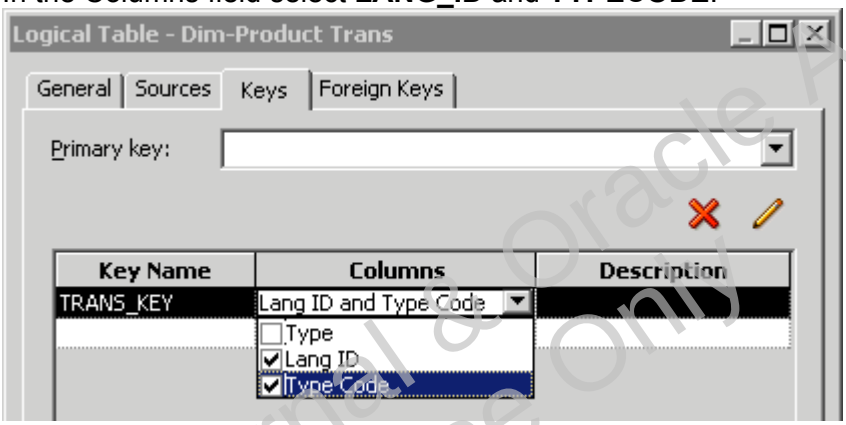
f. On the General tab, select **Lookup table**.



g. Click the **Keys** tab.

h. Name the key **TRANS_KEY**.

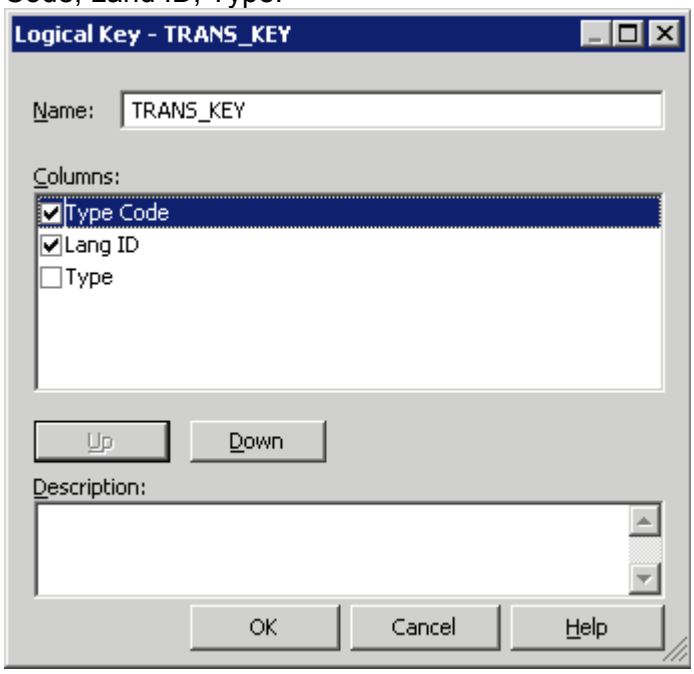
i. In the Columns field select **LANG_ID** and **TYPECODE**.



j. Set **TRANS_KEY** as the primary key.

k. Select **TRANS_KEY** and click the **Edit** button.

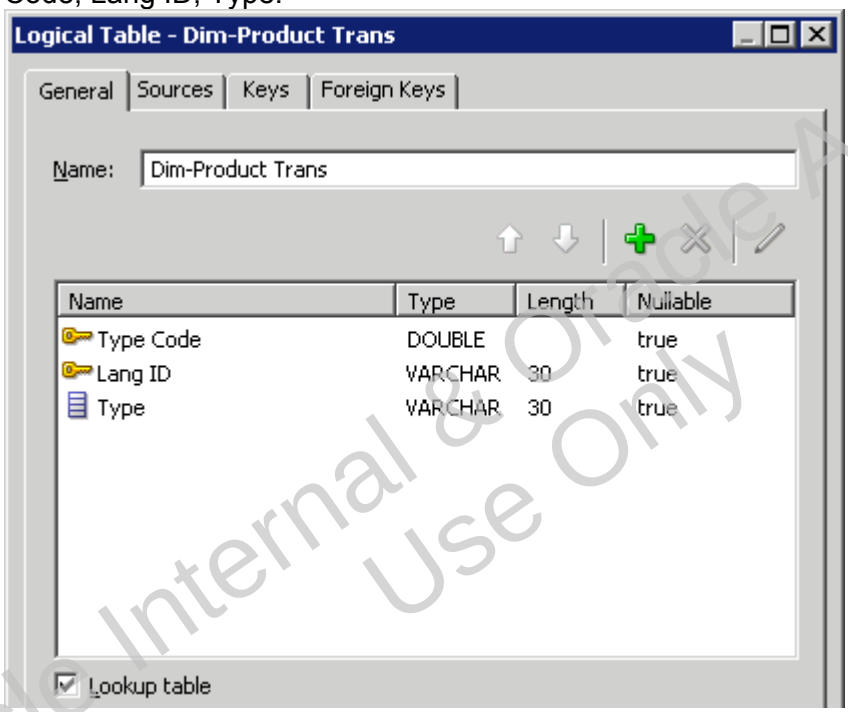
l. Use the **Up** and **Down** buttons to arrange the key columns in the following order: Type Code, Land ID, Type.



m. Click **OK** to close the Logical Key dialog box.

n. Click the **General** tab.

o. Use the **Up** and **Down** arrows to arrange the columns in the following order: Type Code, Lang ID, Type.



A lookup key is different from a logical table key because lookup key columns are order-sensitive. The order in which the columns are specified in the lookup table primary key determines the order of the corresponding arguments in the LOOKUP function. You build the LOOKUP function in the next step.

p. Click **OK** to close the Logical Table dialog box.

4. Create a logical column that uses the INDEXCOL and LOOKUP functions.
   a. Create a new logical column named **Type (Translated)** for the Dim-Product logical table (not the Dim-Product_Trans lookup table).
   b. On the **Column Source** tab, select **Derived from existing columns using an expression**.
   c. Click the **Edit Expression** button to open Expression Builder.
   d. Select **Functions > Conversion Functions** and then double-click **IndexCol** to add it to the expression.
   e. Build the following expression:

```
IndexCol (VALUEOF(NQ_SESSION.”LAN_INT”), “SupplierSales”.”Dim-Product”.”Type”, Lookup (DENSE “SupplierSales”.”Dim-Product_Trans”.”Type”, “SupplierSales”.”Dim-Product”.”Type Code”, VALUEOF(NQ_SESSION.”WEBLANGUAGE”)))
```

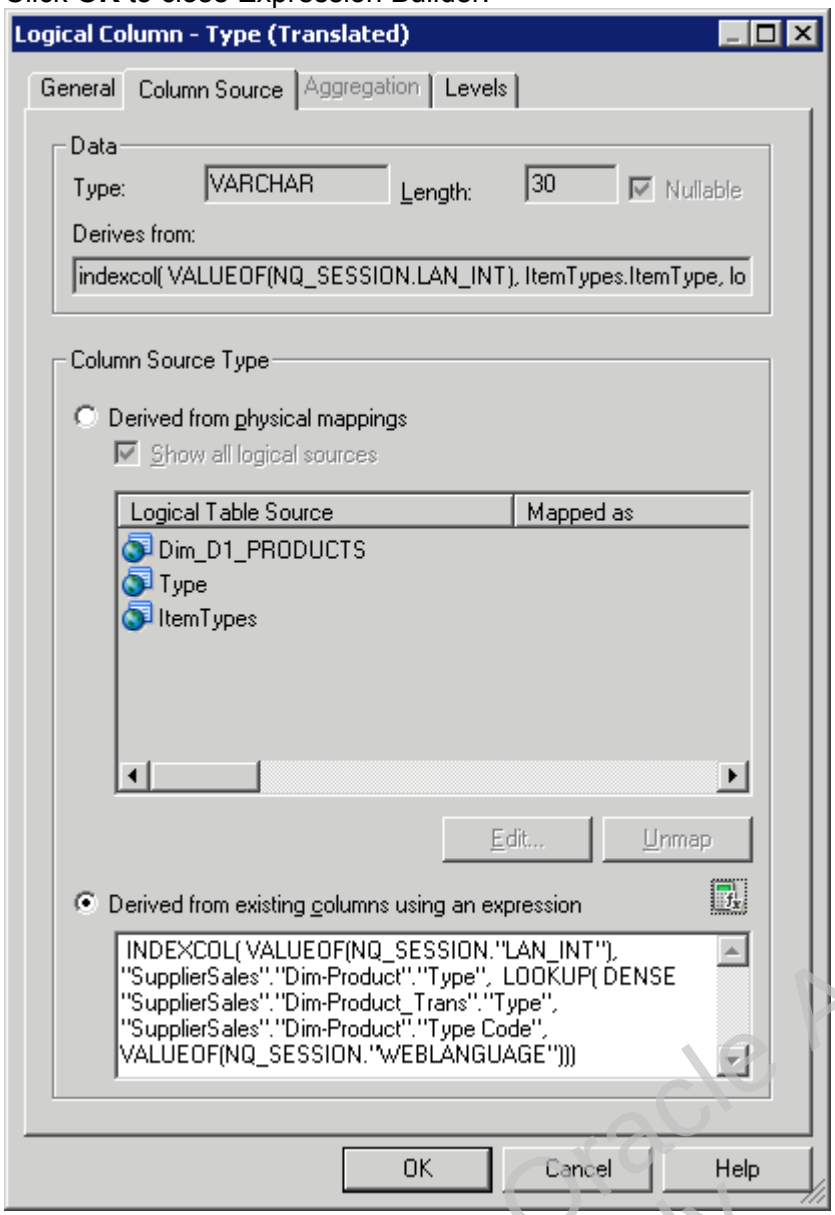
```
INDEXCOL( VALUEOF(NQ_SESSION."LAN_INT"), "SupplierSales"."Dim-Product"."Type",
LOOKUP( DENSE "SupplierSales"."Dim-Product_Trans"."Type", "SupplierSales"."Dim-Product"."Type Code",
VALUEOF(NQ_SESSION."WEBLANGUAGE")))
```

   f. The meaning of the expression is as follows:
   The INDEXCOL function helps to select the proper column. In this example, the expression will return the value of the base column (Dim-Product.Type) only if the user language is the base language (that is, when the value of session variable LAN_INT is 0). If the user language is not the base language (when the value of the session variable LAN_INT is 1), then the expression will return the value from the lookup table of the language that is passed in the WEBLANGUAGE session variable.
   The meaning of the lookup is: Return the translated value of Type from the translation table, Dim-Product_Trans, with the following conditions for the Dim-Product_Trans key columns:

```
Type Code = SupplierSales.Dim-Product.Type Code
Lang ID = VALUEOF(NQ_SESSION."WEBLANGUAGE")
```

   Notice that the expressions must reference the lookup key columns in the same order in which they are defined in the lookup table. In this example, the order is Type Code, Lang_ID.

g. Click **OK** to close Expression Builder.



h. Click **OK** to close the Logical Column dialog box.

i. Drag **Type (Translated)** to the Product presentation table.

j. Save the repository.

k. Check consistency. Fix any errors or warnings before proceeding.

l. Close the repository.

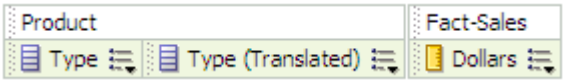
m. Leave the Administration tool open.

5. Create an analysis to test your work.

   a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.

b. Sign in to Oracle BI as **Administrator_f** with password **Administrator_f** and **francais** as the language.

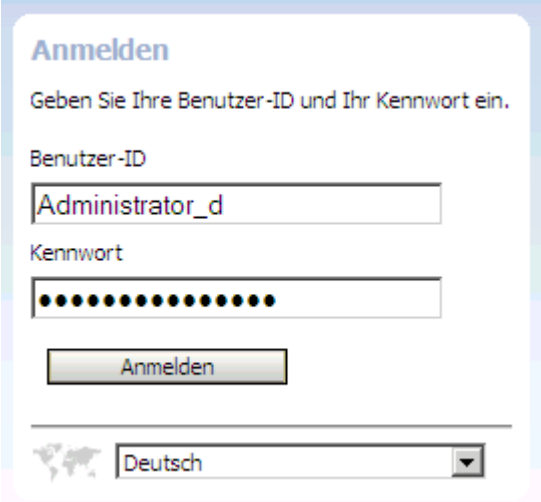
c. Create the following analysis:



d. Click **Resultats**. Verify that you see product types translated to French.

Type	Type (Translated)	Dollars
Baking	Cuire	$4 925 521
Beef	Boeuf	$4 916 016
Beverage	Boisson	$4 398 107
Bread	Pain	$1 578 743
Cereal	Céréale	$1 309 071
Cheese	Fromage	$7 140 616
Condiments	Condiments	$9 105 121
Dessert	Dessert	$2 208 427
Entre	Entre	$1 807 794
Frozen	Gelé	$521
Grains	Grains	$39 419
Lamb	Agneau	$71 553
Non-food	Non-nourriture	$4 547 002
Pasta	Pâtes	$147 409
Pork	Porc	$3 431 672
Poultry	Volaille	$7 202 342
Rice	Riz	$271 889
Seafood	Nourriture de mer	$2 736 436
Snacks	Snacks	$1 482 841
Soup	Soupe	$826 005
Vegetable	Légume	$4 985 949

e. Sign out and sign back in as Administrator_d with language set to Deutsch.



f. Run the same analysis and confirm that product type data is now translated to German.

Type	Type (Translated)	Dollars
Baking	Backen	$4.925.521
Beef	Rindfleisch	$4.916.016
Beverage	Getränk	$4.398.107
Bread	Brot	$1.578.743
Cereal	Getreide	$1.309.071
Cheese	Käse	$7.140.616
Condiments	Würzen	$9.105.121
Dessert	Nachtisch	$2.208.427
Entre	Entre	$1.807.794
Frozen	Eingefroren	$521
Grains	Körner	$39.419
Lamb	Lamm	$71.553
Non-food	Nichtlebensmittel	$4.547.002
Pasta	Teigwaren	$147.409
Pork	Schweinefleisch	$3.431.672
Poultry	Geflügel	$7.202.342
Rice	Reis	$271.889
Seafood	Essbare Meerestiere	$2.736.436
Snacks	Imbisse	$1.482.841
Soup	Suppe	$826.005
Vegetable	Gemüse	$4.985.949

g. Sign out of Oracle BI.

Practices for Lesson 17: Setting an Implicit Fact Column

Lesson 17

Practices for Lesson 17

Lesson Overview

In these practices, you will set an implicit fact column for a subject area.

Practice 17-1: Setting an Implicit Fact Column

Goal

To set an implicit fact column for a subject area

Scenario

ABC tracks product sales facts and product return facts in separate fact tables. ABC wants to ensure that users will get the expected results if they run dimension-only queries with columns from more than one dimension. To ensure that users get the expected results, you set an implicit fact column for the SupplierSales subject area.

Time

20 minutes

Tasks

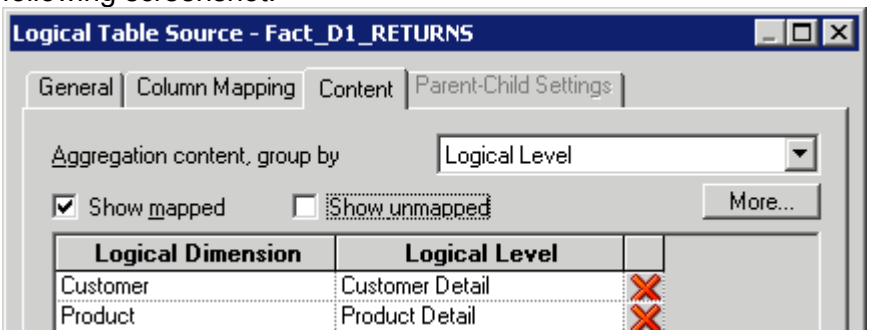
1. Import a new fact table into the Physical layer.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. In the Physical layer, expand **orcl**.
   c. Right-click the **SUPPLIER CP** connection pool and select **Import Metadata**.
   d. Accept the defaults in the Select Metadata Types window, and click **Next** to open the Select Metadata Objects screen.
   e. Scroll to the **SUPPLIER2** schema and expand it.
   f. In the **Data source view** pane select the **D1_RETURNS** table for import.
   g. Click the **Import selected** button to add the table to the Repository View pane.
   h. Expand **SUPPLIER2** in the Repository View pane and confirm that the table is added.
   i. Click **Finish** to add the tables to the repository.
   j. In the Physical layer, expand **orcl > SUPPLIER2** and ensure that the **D1_RETURNS** table is visible.
   k. Update row count for **D1_RETURNS** to verify connectivity. You should see **263** rows.
   l. View data for **D1_RETURNS**. The table contains fact data about product returns.
2. Create an alias and physical joins.
   a. Create an alias for D1_RETURNS and name it **Fact_D1_RETURNS**.
   b. Create the following physical joins:

```
"orcl"."".."SUPPLIER2"."Dim_D1_PRODUCTS"."PRODUCTKEY" =
"orcl"."".."SUPPLIER2"."Fact_D1_RETURNS"."PRODKEY"

"orcl"."".."SUPPLIER2"."Dim_D1_CUSTOMER2"."NEWKEY" =
"orcl"."".."SUPPLIER2"."Fact_D1_RETURNS"."CUSTKEY"
```

3. Modify the business model.
   a. Drag **Fact_D1_RETURNS** to the **SupplierSales** business model.
   b. Rename the **Fact_D1_RETURNS** logical table to **Fact-Returns**.

c. Create the following logical joins:

d. Set the content tab for the **Fact_D1_RETURNS** logical table source as shown in the following screenshot:



e. Set the aggregation rule for the **RETURNS** logical column to **SUM**.

f. Drag **Fact-Returns** to the **SupplierSales** subject area in the Presentation layer. Fact-Returns and Fact-Sales are joined to the same two dimensions, Dim-Product, and Dim-Customer. Fact-Sales is also joined to Dim-Time and Dim-Employee. Both fact tables and all four dimensions are in the same subject area. In the next set of steps, you explore the effect this has on dimension-only queries.

g. Save the repository.

h. Check consistency. Fix any errors or warnings before proceeding.

i. Close the repository.

j. Leave the Administration Tool open.

4. Create an analysis to test your work.

a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.

b. Return to Analysis Editor. Sign in as **weblogic/welcome1**.

c. Create the following analysis to show products purchased by customers. Notice that this is a dimension-only query that includes columns from two dimensions. No fact columns are included in the query.

d. Click **Results**.

Customer	Generic
B L T's Cobblefish	American Cheese Slices
	Refrigerated Dough
	Take-out Containers
Barbecue Lodge Inc	Apple Juice
	Blackened Cajun Seasoning
	Canned Tuna
	Dill Pickles
	Frank's Diet Italian
	Frank's Italian Salad Dressing
	Frank's Strawberry Jam
	Frozen Chicken
	Frozen Crab Legs
	Frozen Peas
	Frozen Ribs
	Frozen Turkey
	Gravy Mix
	Lean Ground Beef Patties

e. Click the **Display maximum** button to view all of the records. You should see **258** records.

f. Sign out of Oracle BI.

g. View the query log.

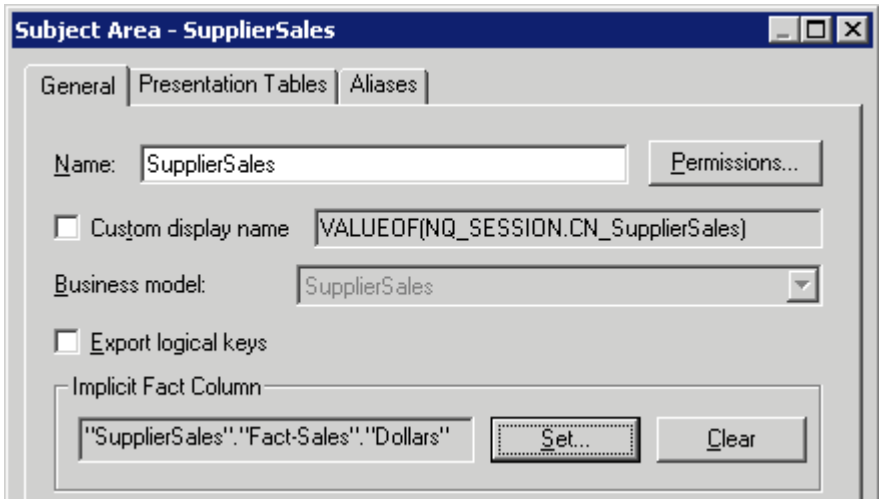
```
---------- Sending query to database named orcl (id: <<2889>>),
connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select distinct T76.NAME as c1,
     T99.GENERICDESCRIPTION as c2
from
     D1_CUSTOMER2 T76 /* Dim_D1_CUSTOMER2 */ ,
     D1_PRODUCTS T99 /* Dim_D1_PRODUCTS */ ,
     D1_RETURNS T1159 /* Fact_D1_RETURNS */
where  ( T76.NEWKEY = T1159.CUSTKEY and T99.PRODUCTKEY =
T1159.PRODKEY ) )
select distinct 0 as c1,
     D1.c1 as c2,
     D1.c2 as c3
from
     SAWITH0 D1
order by c2, c3
```

Notice that the query returned data from the expected dimensions, Customer (Dim_D1_CUSTOMER2) and Product (Dim_D1_PRODUCTS), and that the Fact_D1_RETURNS table was accessed instead of the Fact_D1_ORDERS2 table. In a dimension-only query with columns from multiple dimensions, where common dimensions are joined to multiple fact tables, BI Server must choose a fact table to include in the query. If no implicit fact column is set, Oracle BI Server will choose the most economical fact table. In this case, it is the fact table with the least number of joins. However, this is the “incorrect” data if the user expected to see all products for all customers.

5. Set an implicit fact column for the SupplierSales subject area to point to a measure in the Fact-Sales fact table.

   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.

b. In the Presentation layer, double-click **SupplierSales** to open the Subject Area dialog box.
c. Click the **General** tab.
d. In the Implicit Fact Column section, click **Set**.
e. Expand **Fact-Sales**, select the **Dollars** column and click **OK**. The implicit fact column is set to Fact-Sales.Dollars.



f. Click **OK** to close the Subject Area dialog box.
g. Save the repository.
h. Check consistency. Fix any errors or warnings before proceeding.
i. Close the repository.
j. Leave the Administration Tool open.

6. Create analysis to test your work.
   a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.
   b. Return to Analysis Editor.
   c. Create the same dimension-only request to show products purchased by customers.

d. Click **Results**. This is the “correct” data because it shows all products for all customers.

Customer	Generic
2nd & Goal Sports Cafe	American Cheese Slices
	Apple Dumplings
	Apple Juice
	Apple Sauce
	Baked Beans
	Balsamic Vinegar
	Beef Bouillon
	Biscuit Mix
	Black Pepper
	Blackened Cajun Seasoning
	Breading Mix
	Breakfast Sausage Links
	Brown Sugar
	Butter
	Can Liners
	Canned Beef
	Canned Tuna
	Cheddar Cheese
	Children's Snacks
	Chocolate Fudge Topping
	Chorrizo Sausage
	Coffee
	Cooking Wine
	Corn Flakes
	Crackers

e. Sign out of Oracle BI.

f. View the query log.

```
---------- Sending query to database named orcl (id: <<3058>>),
connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T90.DOLLARS) as c1,
     T76.NAME as c2,
     T99.GENERICDESCRIPTION as c3
from
     D1_CUSTOMER2 T76 /* Dim_D1_CUSTOMER2 */ ,
     D1_ORDERS2 T90 /* Fact_D1_ORDERS2 */ ,
     D1_PRODUCTS T99 /* Dim_D1_PRODUCTS */
where  ( T76.NEWKEY = T90.CUSTKEY and T90.PRODKEY =
T99.PRODUCTKEY )
```

Notice that the query again returns data from the expected dimensions, Customer and Product, but this time the Fact_D1_ORDERS2 table is accessed instead of the Fact_D1_RETURNS table. Setting the implicit fact column forced Oracle BI Server to join the dimensions through this fact table. The column is visible in the query log, but not visible in the analysis results. It is used to specify a default join path between dimension tables when there are several possible alternatives or contexts.

Practices for Lesson 18: Importing Metadata from Multidimensional Data Sources

Lesson 18

Practices for Lesson 18

Lesson Overview

In these practices, you will add to your business model using an Essbase data source.

Practice 18-1 Importing a Multidimensional Data Source into a Repository

Goal

To import an Essbase multidimensional data source into a repository

Scenario

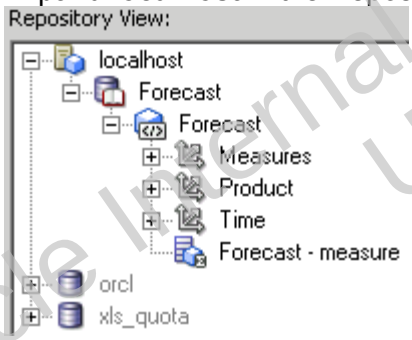
ABC has forecast data stored in an Essbase cube. You import the cube into the repository, create a business model and subject area, and verify the results in Analysis Editor.

Time

15 minutes

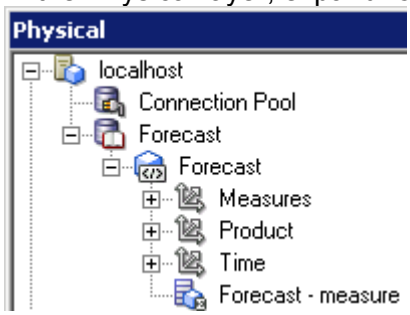
Tasks

1. Start the Essbase Server.
   a. Select **Start > Programs > Oracle EPM System > Essbase > Essbase Server > Start Essbase**. A Start Essbase command window opens.
   b. Wait for processing to complete and for the command window to close.
   c. Right-click the **taskbar** and select **Task Manager** to open the Windows Task Manager.
   d. Click the **Processes** tab.
   e. Verify that an **ESSBASE.exe** process is visible.
   f. Select **File > Exit Task Manager** to close Windows Task Manager.
2. Import an Essbase cube into the repository.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. Select **File > Import Metadata**.
   c. Select the **Essbase** connection type.
   d. Enter **localhost** for Essbase Server.
   e. Enter **admin** as username and **essbase** as password.
   f. Click **Next**.
   g. Expand **localhost** and select **Forecast**. Make sure that **localhost** is *not* selected.
   h. Click the **Import selected** button to add the tables to the Repository View pane.
   i. Expand **localhost** in the Repository View pane and verify that Forecast is added.



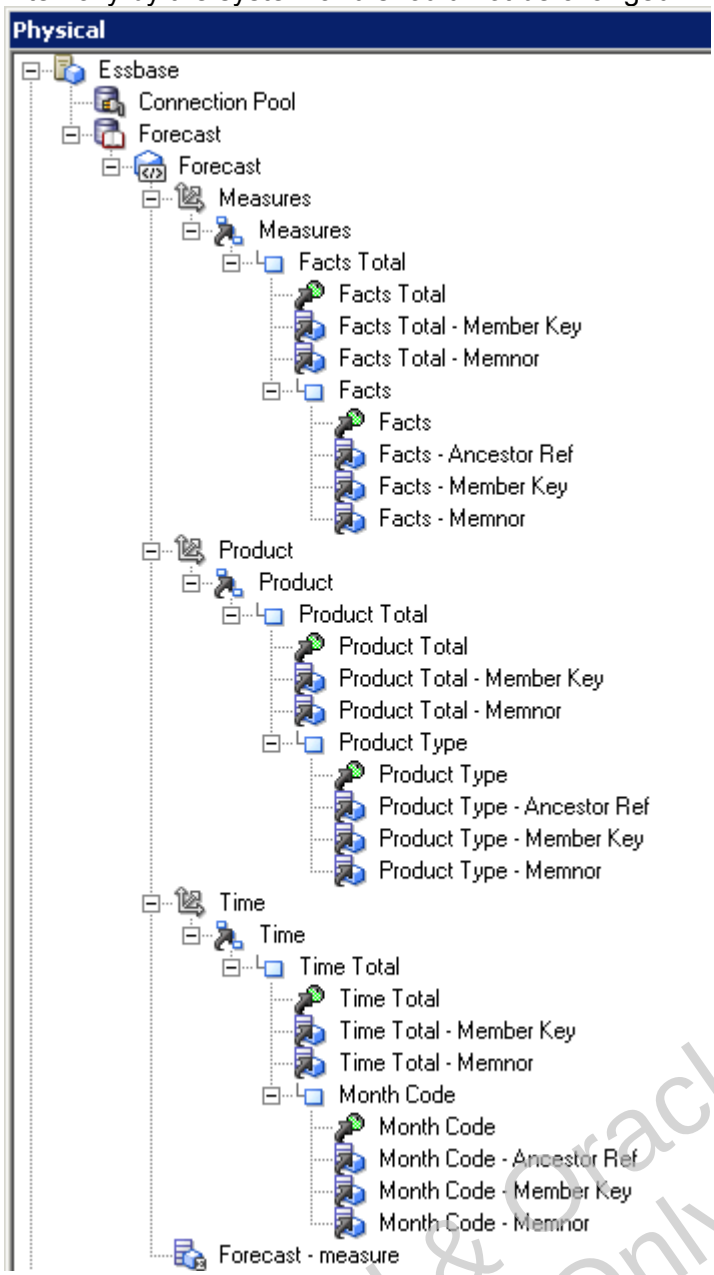
   j. Click **Finish** to add the data source to the repository.

k. In the Physical layer, expand **localhost** and verify that Forecast is visible.



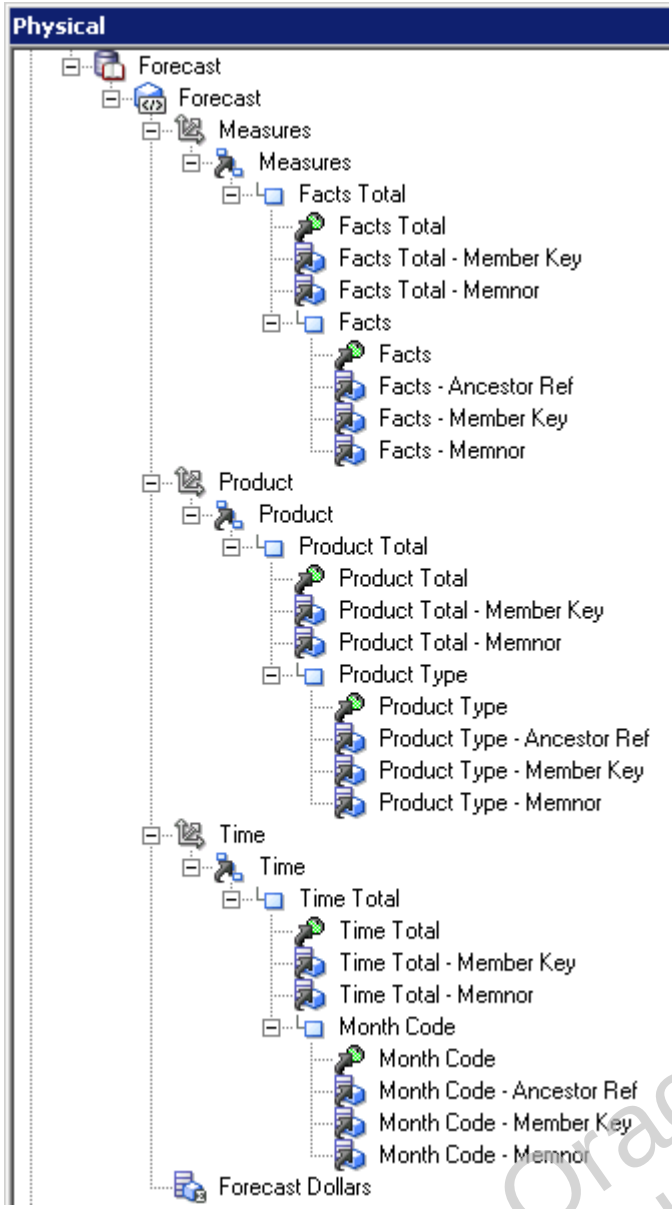
3. Explore the multidimensional schema in the Physical layer.
   a. In the Physical layer, rename **localhost** to **Essbase**.
   b. Right-click **Forecast** and select **Expand All**. When you import metadata from Essbase data sources, the cube metadata is mapped to the Physical layer in a way that supports the Oracle Business Intelligence logical model. Metadata that applies to all members of the dimension, such as aliases, are modeled as dimension properties by default. Level-based properties, such as outline sort/memnor information, are mapped as separate physical cube columns in the dimension. Column types such as Outline Sort, Ancestor Reference, Member Key, Leaf, Root, and Parent Reference are used

internally by the system and should not be changed.



4. Convert the measure dimension to flat measures.
   a. By default, measures are imported as measure hierarchies. In other words, the cube contains a single measure column that represents all the measures. Alternatively, you can choose to flatten the measure hierarchy to view each measure as an individual column. To do this, right-click the **Forecast** cube object and select **Convert measure**

dimension to flat measures. The **Forecast Dollars** measure is now visible.

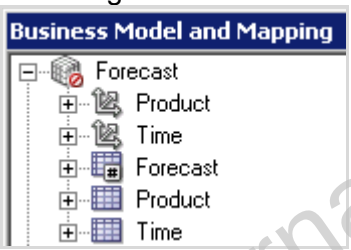


5. Verify the connection.

a. Right-click the **Product Type** level and select **View Members**.

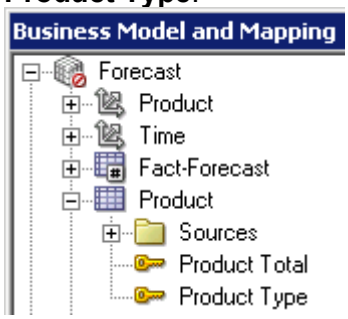
1
Baking
Beef
Beverage
Bread
Cereal
Cheese
Condiments
Dessert
Entre
Frozen
Grains
Lamb
Non-food
Pasta
Pork
Poultry
Rice
Seafood
Snacks
Soup
Vegetable

6. Change the Month Code data types.
   a. Double-click the **Month Code** *column* (not the level).
   b. Change the data type to **DOUBLE** and click **OK**. This is required for a later step in this practice.
7. Create the business model.
   a. Drag the **Forecast** cube from the Physical layer to white space in the Business Model and Mapping layer to create a Forecast business model.
   b. Expand **Forecast** in the Business Model and Mapping layer. Dragging the cube from the Physical layer to the BMM layer automatically creates the business model and all of its objects, including logical dimension hierarchies, logical dimension tables, logical fact tables, logical columns, and logical joins. Notice that Forecast has a hash sign, indicating that it is the fact table in the logical schema.

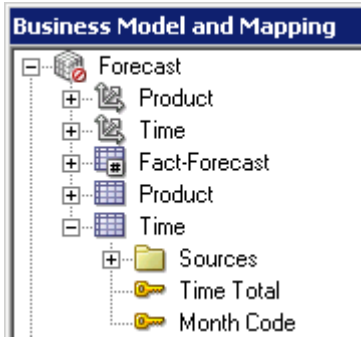


   c. Rename the Forecast logical table **Fact-Forecast**.

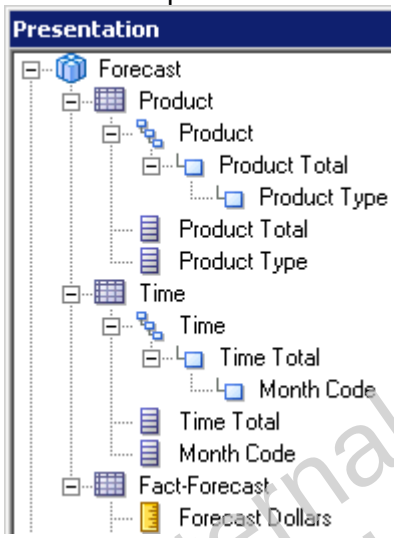
d. Delete all columns from the Product logical table *except* for **Product Total** and **Product Type**.



e. Delete all columns from the Time logical table *except* for **Time Total** and **Month Code**.



8. Create the subject area.
   a. Drag the **Forecast** business model from the Business Model and Mapping layer to the Presentation layer to create the **Forecast** subject area.
   b. Reorder the presentation tables:



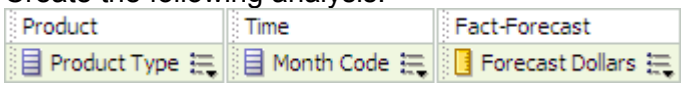
   c. Save the repository.
   d. Check consistency. You should receive a message that the Forecast business model is consistent. Click **Yes** to mark it as available for queries.
   e. Verify that you do not receive any consistency check errors or warnings.
   f. Close the repository.
   g. Leave the Administration Tool open.
9. Check your work.

a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.

b. Return to Analysis Editor.

c. Select the **Forecast** subject area.

d. Create the following analysis:



e. Click **Results**. Forecast dollars data is displayed by product type and month code.

Product Type	Month Code	Forecast Dollars
Baking	200801	$271,234
	200802	$289,782
	200803	$316,107
	200804	$298,547
	200805	$336,265
	200806	$321,441
	200807	$320,904
	200808	$318,544
	200809	$297,708
	200810	$363,307
	200811	$301,131
	200812	$323,062
	200901	$333,398
	200902	$329,676
	200903	$328,551
	200904	$210,930
Beef	200801	$272,431
	200802	$323,016
	200803	$320,182
	200804	$313,324
	200805	$332,965
	200806	$325,336
	200807	$344,206
	200808	$338,470
	200809	$307,258

f. Sign out of Oracle BI.

g. Check the query log. Notice that the query is sent to the Essbase data source.

```
---------- Sending query to database named Essbase (id:
<<2988>>), connection pool named Connection Pool: [[
with
  set [_Product2]  as '[Product].Generations(2).members'
  set [_Time2]  as '[Time].Generations(2).members'
select
  { [Measures]
  } on columns,
  NON EMPTY {crossjoin({[_Product2]},{[_Time2]})} properties
GEN_NUMBER, [Time].[MEMBER_UNIQUE_NAME], [Time].[Memnor],
[Product].[MEMBER_UNIQUE_NAME], [Product].[Memnor] on rows
from [Forecast.Forecast]
```

Practice 18-2: Incorporating Horizontal Federation into a Business Model

Goal

To build a business model that incorporates horizontal federation of Essbase and relational data sources

Scenario

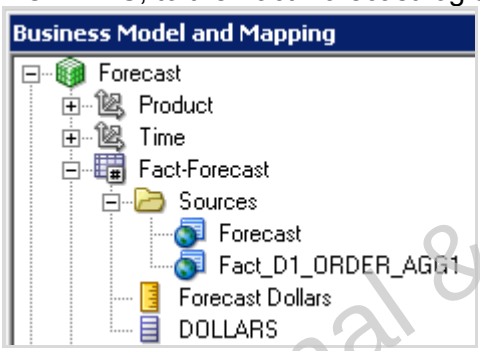
Horizontal federation provides the ability to create reports that can display data from both Essbase and relational data sources. In the previous, practice you imported an Essbase cube and viewed analysis results. In this practice, you add objects from the Supplier Sales relational data source to the Essbase business model so that you can build a report that compares forecast dollars with actual dollars.

Time

15 minutes

Tasks

1. Add a new logical table source and column to the Fact-Forecast logical table to establish a relationship between the Essbase source and the relational source in the business model.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. In the Forecast business model, expand **Fact-Forecast > Sources** and notice that there is currently only one logical table source, **Forecast**, for this logical table.
   c. Drag **SUPPLIER2 > Fact_D1_ORDER_AGG1 > DOLLARS** from the Physical layer to the **Fact-Forecast** logical table (not the logical table source) in the BMM layer. This adds a new logical table source, Fact_D1_ORDER_AGG1, and a new logical column, DOLLARS, to the Fact-Forecast logical table.



   d. Double-click the **Fact_D1_ORDER_AGG1** logical table source to open the Logical Table Source dialog box.
   e. Click the **Column Mapping** tab. Notice that the DOLLARS logical column is mapped to the DOLLARS physical column in the Fact_D1_ORDER_AGG1 physical table in the SUPPLIER2 relational schema.
   f. Click **OK** to close the Logical Table Source dialog box.
   g. Double-click the **Forecast** logical table source to open the Logical Table Source dialog box.

h. If necessary, click the **Column Mapping** tab. Select both the **Show mapped columns** and **Show unmapped columns** options. Notice that the Forecast Dollars logical column is mapped to the Forecast cube in the Essbase data source.

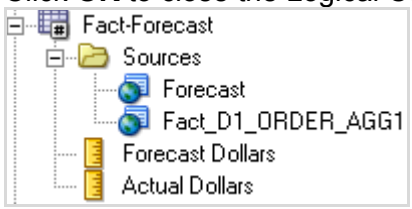
i. Click **OK** to close the Logical Table Source dialog box.

j. Double-click the **DOLLARS** logical column in the BMM layer to open the Logical Column dialog box.

k. On the General tab, rename **DOLLARS** to **Actual Dollars**.

l. Click the **Aggregation** tab and set the Default aggregation rule to **Sum**.

m. Click **OK** to close the Logical Column dialog box.



2. Add a new logical table source and column to the Product logical table to establish the relationship between the Essbase source and the relational source in the business model.

a. In the Forecast business model, rename the **Product** logical table (not the logical dimension) to **Dim-Product**.

b. In the BMM layer, expand **Forecast > Dim-Product > Sources**. Notice that there is currently only one logical table source, Forecast, for this logical table.

c. Drag the **Dim_D1_PRODUCT_TYPE > ITEMTYPE** physical column from the Physical layer to the **Forecast > Dim-Product > Product Type** logical column.

d. Notice that the Dim-Product logical table now has two logical sources: Forecast and Dim_D1_PRODUCT_TYPE.

e. Double-click the **Dim_D1_PRODUCT_TYPE** logical table source to open the Logical Table Source dialog box.

f. Click the **Column Mapping** tab and notice that the Product Type logical column maps to the ITEMTYPE physical column in the Dim_D1_PRODUCT_TYPE physical table in the SUPPLIER2 schema.

g. Close the Logical Table Source dialog box.

h. Open the **Forecast** logical table source and notice that the remaining logical columns are mapped to the Essbase cube columns. Product Type is now mapped to two physical columns in two physical data sources: an Essbase multidimensional data source and an Oracle relational data source.

i. Close the Logical Table Source dialog box.

3. Add a new logical table source and column to the Time logical table to establish the relationship between the Essbase source and the relational source in the business model.

a. Rename the **Time** logical table to **Dim-Time**.

b. In the BMM layer, expand **Forecast > Dim-Time > Sources**. Notice that there is currently only one logical table source, Forecast, for this logical table.

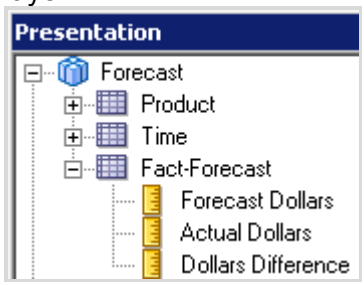
c. Drag the **Dim_MONTHS > MONTHCODE** physical column from the Physical layer to the **Forecast > Dim-Time > Month Code** logical column.

d. Notice that the Dim-Time logical table now has two logical sources: Forecast and Dim_MONTHS. Month Code is now mapped to columns in two physical sources.

e. Double-click the **Dim-Time** logical table to open the Logical Table dialog box.

f. Click the **Keys** tab. Dragging the MONTHCODE column created a duplicate logical key. Redundant logical table keys should be removed.
g. Select **Dim_MONTHS_Key** and click the **Delete** button.
h. Click **OK** to confirm the delete.
i. Click **OK** to close the Logical Table dialog box.

4. Create derived measures.
   a. Right-click the **Fact-Forecast** logical table and select **New Object > Logical Column**.
   b. On the General tab, name the logical column **Dollars Difference**.
   c. Click the **Column Source** tab.
   d. Select **Derived from existing columns using an expression**.
   e. Open **Expression Builder**.
   f. Create the following expression:
      "Forecast"."Fact-Forecast"."Actual Dollars" - "Forecast"."Fact-Forecast"."Forecast Dollars"
   g. Click **OK** to close Expression Builder.
   h. Click **OK** to close the Logical Column dialog box.
5. Modify the presentation layer.
   a. Expand **Forecast > Fact-Forecast** in the Presentation layer.
   b. Drag **Actual Dollars** and **Dollars Difference** to **Fact-Forecast** in the Presentation layer.



   c. Save the repository.
   d. Check consistency. Fix any errors or warnings before proceeding.
   e. Close the repository.
   f. Leave the Administration Tool open.
6. Create an analysis to test your work.
   a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.
   b. Return to Analysis Editor and sign in as **weblogic/welcome1**.
   c. Select the **Forecast** subject area.
   d. Create the following analysis.

Product	Time	Fact-Forecast		
Product Type	Month Code	Forecast Dollars	Actual Dollars	Dollars Difference

e. Click **Results**.

Product Type	Month Code	Forecast Dollars	Actual Dollars	Dollars Difference
Baking	200,801.00	$271,234	$269,183	-$2,051
	200,802.00	$289,782	$287,437	-$2,345
	200,803.00	$316,107	$312,651	-$3,456
	200,804.00	$298,547	$293,180	-$5,367
	200,805.00	$336,265	$331,326	-$4,939
	200,806.00	$321,441	$316,510	-$4,931
	200,807.00	$320,904	$317,913	-$2,991
	200,808.00	$318,544	$314,553	-$3,991
	200,809.00	$297,708	$290,870	-$6,838
	200,810.00	$363,307	$359,976	-$3,331
	200,811.00	$301,131	$297,091	-$4,040
	200,812.00	$323,062	$315,993	-$7,069
	200,901.00	$333,398	$336,281	$2,883
	200,902.00	$329,676	$333,677	$4,001
	200,903.00	$328,551	$334,120	$5,569
	200,904.00	$210,930	$214,758	$3,828
Beef	200,801.00	$272,431	$271,197	-$1,234
	200,802.00	$323,016	$320,671	-$2,345
	200,803.00	$320,182	$316,726	-$3,456
	200,804.00	$313,324	$307,958	-$5,366
	200,805.00	$332,965	$328,027	-$4,938
	200,806.00	$325,336	$320,404	-$4,932
	200,807.00	$344,206	$341,216	-$2,990
	200,808.00	$338,470	$334,479	-$3,992
	200,809.00	$307,258	$300,419	-$6,839

f. Sign out of Oracle BI.

g. Examine the query log. Data for the Forecast Dollars measure is coming from the Essbase source. Data for the Actual Dollars measure is coming from Fact_D1_ORDERS_AGG1 in the SUPPLIER2 relational source. Dollars Difference is calculated across the measures. Data from the conforming dimensions, Product and

Time, is applied across the measures.

```
---------- Sending query to database named orcl (id: <<3244>>),
connection pool named SUPPLIER CP: [[

select T502.ITEMTYPE as c1,
     T862.MONTHCODE as c2,
     sum(T744.DOLLARS) as c3
from
     MONTHS T862 /* Dim_MONTHS */ ,
     D1_PRODUCT_TYPE T502 /* Dim_D1_PRODUCT_TYPE */ ,
     D1_ORDER_AGG1 T744 /* Fact_D1_ORDER_AGG1 */
where  ( T502.TYPECODE = T744.TYPEKEY and T744.PERKEY =
T862.MONTHCODE )
group by T502.ITEMTYPE, T862.MONTHCODE
order by c1, c2

]]
[2010-08-19T23:22:37.000+00:00] [OracleBIServerComponent]
[TRACE:2] [USER-18] [] [ecid:
0000IeCeK5h7i4WzLwFS8A1CQlb^0002Rf] [tid: 29f0] [requestid:
4103000f] [sessionid: 41030000] [username: weblogic] ----------
---------- Sending query to database named Essbase (id:
<<3315>>), connection pool named Connection Pool: [[
with
  set [_Product2]  as '[Product].Generations(2).members'
  set [_Time2]  as '[Time].Generations(2).members'
select
  { [Measures]
  } on columns,
  NON EMPTY {crossjoin({[_Product2]},{[_Time2]})} properties
GEN_NUMBER on rows
from [Forecast.Forecast]
```

Practice 18-3: Incorporating Vertical Federation into a Business Model

Goal

To build a business model that incorporates vertical federation of Essbase and relational data sources

Scenario

Vertical federation provides reports with the ability to drill through aggregate Essbase data into detail relational data. In this practice, you expand the business model to include vertical federation of Essbase and relational data sources.

Time

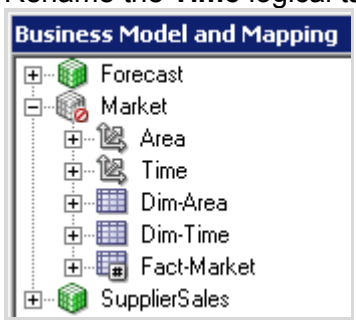
30 minutes

Tasks

1. Import another Essbase cube into the repository.
    a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
    b. Expand **Essbase** in the Physical layer.
    c. Right-click the **connection pool** and select Import Metadata.
    d. Expand **localhost** and select **Market**. Make sure **localhost** is not selected.
    e. Click the **Import selected** button to add the tables to the Repository View pane.
    f. Expand **localhost** in the Repository View pane and verify that Market is added.
    g. Click **Finish** to add the data source to the repository.
        a. In the Physical layer, expand **Essbase** and verify that Market is visible.
        b. Expand the **Market** object in the Physical Layer and examine the structure. The cube aggregates sales data across two dimensions: Area and Time. The Area hierarchy has three levels: Area Total, Region, and District. The Time dimension has two levels: Time Total and Year. There is only one measure: Market Dollars.
        c. Right-click **MarketDB** and select **Convert measure dimension to flat measures**. The Market Dollars measure is now visible.
        d. Double-click the **YR** column to open the Physical Cube Column dialog box.
        e. Change Type to **DOUBLE** and click **OK**.
2. Import a new physical table into the repository.
    a. Expand **orcl**, right-click **SUPPLIER CP**, and select **Import Metadata**.
    b. Import **D1_SALESREPS_AGG** from SUPPLIER2.
    c. In the Physical layer, create an alias for D1_SALESREPS_AGG and name it **Fact_D1_SALESREPS_AGG.**
    d. View data for **Fact_D1_SALESREPS_AGG.** The table aggregates sales data at the region, district, sales rep, and year levels.
3. Create the business model.
    a. Drag the **MarketDB** cube from the Physical layer to the BMM layer to create the MarketDB business model.
    b. Rename the **MarketDB** business model **Market**.
    c. Rename the **MarketDB** logical table **Fact-Market**.

d. Rename the **Area** logical table **Dim-Area**.

e. Rename the **Time** logical table **Dim-Time**.



4. Add a new logical column and logical table source to Fact-Market.

   a. In the Physical layer, expand **SUPPLIER2 > Fact_D1_SALESREPS_AGG.**

   b. In the BMM layer, expand **Fact-Market.**

   c. Drag the **DOLLARS** physical column from **SUPPLIER2 > Fact_D1_SALESREPS_AGG** to the **Market Dollars** logical column. This creates a new logical table source named Fact_D1_SALEREPS_AGG for the Fact-Market logical table.

   d. Double-click the **Market Dollars** logical column to open the Logical Column dialog box.

   e. Click the **Column Source** tab. Notice that the Market Dollars logical column maps to physical columns in two data sources: the Essbase cube and the SUPPLIER2 relational source.

   f. Click the **Aggregation** tab and set the aggregation rule for the logical column to **Sum**.

   g. Click **OK** to close the Logical Column dialog box.

5. Add new logical columns and a logical table source to Dim-Area.

   a. Expand **Dim-Area**.

   b. Delete all columns from Dim-Area except for **Area Total**, **Region**, and **District**.

   c. Drag the **REGION** and **DISTRICT** columns from **Fact_D1_SALESREPS_AGG** to their corresponding logical columns in **Dim-Area**. This creates a new logical table source named Fact_D1_SALESREPS_AGG for the Dim-Area logical table.

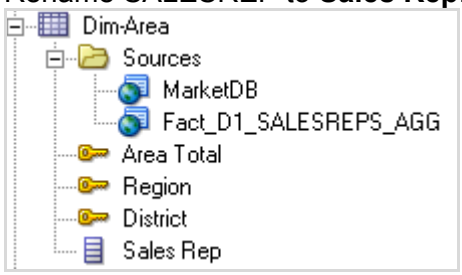
   d. Double-click the **Region** logical column to open the Logical Column dialog box.

   e. Click the **Column Source** tab. Notice that the **Region** logical column maps to physical columns in two data sources: the Essbase cube and the SUPPLIER2 relational source. The same is true for the **District** logical column.

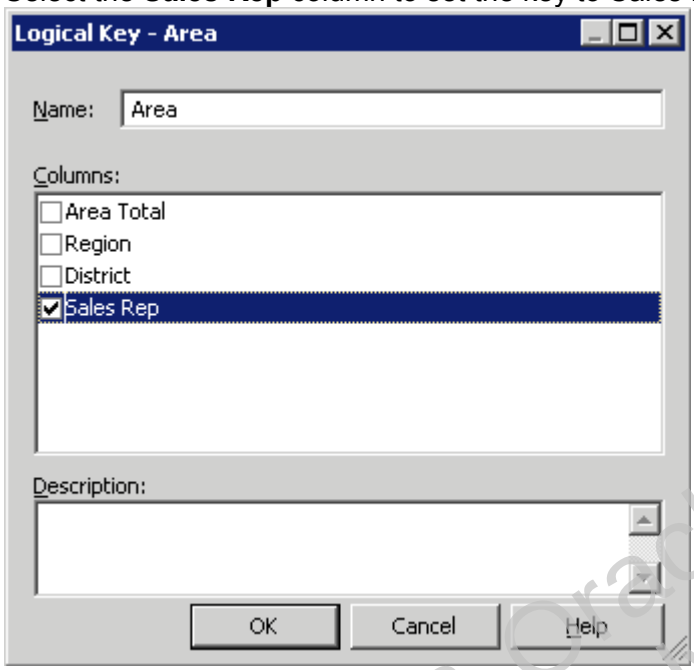
   f. Click **OK** to close the Logical Column dialog box.

   g. Drag the **SALESREP** column from **Fact_D1_SALESREPS_AGG** to the **Dim-Area** logical table. This creates a new SALESREP logical column.

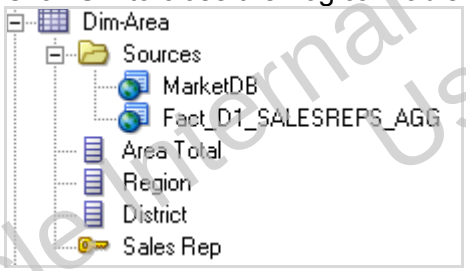
h. Rename SALESREP to **Sales Rep**.



6. Set the key for the Dim-Area logical table.
   a. Double-click the **Dim-Area** logical table to open the Logical Table dialog box.
   b. Click the **Keys** tab.
   c. Select the **Area** key.
   d. Click the **Edit** button.
   e. Deselect **Area Total**, **Region**, and **District**.
   f. Select the **Sales Rep** column to set the key to Sales Rep only.



   g. Click **OK** to close the Logical Key dialog box.
   h. Click **OK** to close the Logical Table dialog box.



7. Add a new logical column and logical table source to Dim-Time.
   a. Expand **Dim-Time**.
   b. Delete all columns from Dim-Time except for **Time Total** and **YR**.

c. Drag the **YEAR** column from **Fact_D1_SALESREPS_AGG** to the YR logical column in **Dim-Time**. This creates a new logical table source named Fact_D1_SALESREPS_AGG for the Dim-Time logical table.

d. Double-click the **YR** logical column to open the Logical Column dialog box.

e. Click the **Column Source** tab. Notice that the YR logical column maps to physical columns in two data sources: the Essbase cube and the SUPPLIER2 relational source.

f. Click **OK** to close the Logical Column dialog box.

8. Set the key for the Dim-Time logical table.

a. Double-click the **Dim-Time** logical table to open the Logical Table dialog box.

b. Click the **Keys** tab.

c. Select the **Time** key and click the Edit button.

d. Deselect **Time Total**. Only **YR** should be selected.

e. Click **OK** to close the Logical Key dialog box.

f. Click **OK** to close the Logical Table dialog box.

9. Modify the Area logical dimension hierarchy.

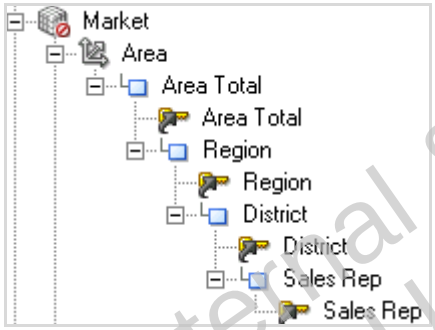
a. Expand the **Area** logical dimension.

b. Right-click the **District** logical level and select **New Object > Child Level** to open the Logical Level dialog box.

c. Name the level **Sales Rep** and click **OK.**

d. Drag the **Sales Rep** logical column from the **Dim-Area** logical table to the **Sales Rep** child level in the Market hierarchy.

e. Right-click the **Sales Rep** column (not the level) in the hierarchy and select **New Logical Level Key** to open the Logical Level Key dialog box. Verify that the **Sales Rep** column is selected and click **OK**.



f. Save the repository.

g. Click **No** when prompted to check global consistency. The important point to notice here is that because you cannot create physical joins between Essbase and relational sources, the relational source is unable to inherit the Area hierarchy from the Essbase source. Therefore, you have to physically define the hierarchy to allow for drilldown. In this case, you must have Region, District, and Sales Rep columns in your relational source, so that you are able to drill all the way through Region and District in the Essbase cube to Sales Rep in the relational source. In this set of steps, you modified

the Area hierarchy to allow for this drilldown through all of the hierarchy levels. This technique enables drill-through from aggregated data in the Essbase cube into detail data in the relational source.

10. Set the logical levels for the Dim-Area logical table.
    a. Expand **Dim-Area > Sources**.
    b. Double-click the **Fact_D1_SALESREPS_AGG** logical table source to open the Logical Table Source dialog box.
    c. Click the **Column Mapping** tab. Make sure that both mapped columns and unmapped columns are visible and notice that all logical columns except Area Total map to physical columns in the **Fact_D1_SALESREPS_AGG** relational table.

Logical Column	Expression		Physical Table
Area Total			
Region	REGION		Fact_D1_SALESREPS_AGG
District	DISTRICT		Fact_D1_SALESREPS_AGG
Sales Rep	SALESREP		Fact_D1_SALESREPS_AGG

    d. Click the **Content** tab and set the Area dimension logical level to **Sales Rep**.
    e. Click **OK** to close the Logical Table Source dialog box.
    f. Double-click the **MarketDB** logical table source.
    g. Click the **Column Mapping** tab and notice that all the logical columns except Sales Rep map to physical columns in the Essbase cube.

Logical Column	Expression		Physical Table
Area Total	Area Total		MarketDB
Region	Region		MarketDB
District	District		MarketDB
Sales Rep			

    h. Click the **Content** tab and verify that the Area dimension logical level is set to **District**.
    i. Click **OK** to close the Logical Table Source dialog box.
11. Set the logical levels for the Dim-Time logical table.
    a. Expand **Dim-Time > Sources**.
    b. Double-click the **Fact_D1_SALESREPS_AGG** logical table source to open the Logical Table Source dialog box.
    c. Click the **Content** tab and set the Time dimension logical level to **YR**.
    d. Click **OK** to close the Logical Table Source dialog box.
    e. Double-click the **MarketDB** logical table source.
    f. Click the **Content** tab and verify that the Time dimension logical level is set to **YR**.
    g. Click **OK** to close the Logical Table Source dialog box.
12. Set the logical levels for the Fact-Market logical table.
    a. Expand **Fact-Market > Sources**.
    b. Double-click the **MarketDB** logical table source.
    c. Click the **Content** tab.
    d. Verify that the logical level for the Area dimension is set to **District**.
    e. Verify that the logical level for the Time dimension is set to **YR**.
    f. Click **OK** to close the Logical Table Source dialog box.
    g. Double-click the **Fact_D1_SALESREPS_AGG** logical table source to open the Logical Table Source dialog box.

h. Set the logical level for the Area dimension to **Sales Rep**.

i. Set the logical level for the Time dimension to **YR**.

j. Click **OK** to close the Logical Table Source dialog box. The lowest level in the cube is District. Setting different levels for the logical table sources tells the Oracle BI engine that for any queries that include the District level and above, use the Essbase cube. For any queries below the District level (Sales Rep in this example) use the relational source. You verify this when you run an analysis later in this practice.

13. Create the presentation catalog.

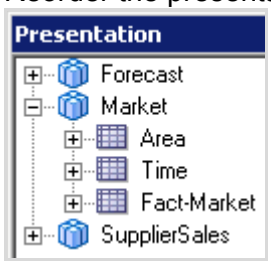
a. Drag the **Market** business model to the Presentation layer to create the Market subject area.

b. Expand the **Market** subject area.

c. Rename Dim-Area **Area**.

d. Rename Dim-Time **Time**.

e. Reorder the presentation tables:



f. Save the repository.

g. Check consistency. You should receive the following message: “Business model ‘Market’ is consistent. Do you want to mark it available for queries?”.

h. Click **Yes**. Fix and errors or warnings before proceeding.

i. Close the repository.

j. Leave the Administration Tool open.

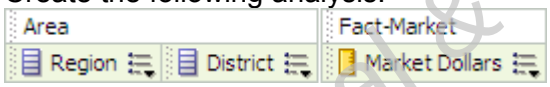
14. Create an analysis to test your work.

a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.

b. Return to Analysis Editor and sign in as **weblogic/welcome1**.

c. Select the **Market** subject area.

d. Create the following analysis:



e. Click **Results**.

Region	District	Market Dollars
Central	Gulf	$814,648
	LowerMidWest	$4,654,545
	MidWest	$2,354,705
	Texas	$3,984,062
	UpperMidWest	$1,132,842
East	Florida	$1,625,588
	MidAtlantic	$3,775,583
	UpperSouth	$6,244,606
	Yankee	$13,344,845
West	California	$16,127,207
	Desert	$6,911,760
	Northwest	$2,162,063

f. Leave the analysis results open and check the query log. As expected, the data is retrieved from the Essbase cube.

```
--------- Sending query to database named Essbase (id:
<<3132>>), connection pool named Connection Pool: [[
With
  set [_Area3]  as 'Generate([Area].Generations(2).members,
Descendants([Area].currentmember, [Area].Generations(3),
leaves))'
select
  { [Time]
  } on columns,
  NON EMPTY {{[_Area3]}} properties ANCESTOR_NAMES, GEN_NUMBER
on rows
from [Market.MarketDB]
```

g. Return to **Analysis Editor** to the open analysis.

h. Drill down on any **District** to view **Sales Rep** data.

Region	District	Sales Rep	Market Dollars
East	Yankee	ANN JOHNSON	$2,382,251
		BETTY NEWER	$4,727,671
		STEVEN SMITH	$6,121,802
		TRACIE BELL	$113,121

i. Sign out of Oracle BI.

j. Check the query log. Verify that as expected the data is retrieved from the relational source.

```
--------- Sending query to database named orcl (id: <<3544>>),
connection pool named SUPPLIER CP: [[
WITH
SAWITH0 AS (select sum(T1519.DOLLARS) as c1,
     T1519.DISTRICT as c2,
     T1519.REGION as c3,
     T1519.SALESREP as c4
from
     D1_SALESREPS_AGG T1519 /* Fact_D1_SALESREPS_AGG */
where  ( T1519.DISTRICT = 'Yankee' and T1519.REGION = 'East' )
group by T1519.DISTRICT, T1519.REGION, T1519.SALESREP)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c3 as c3,
     D1.c4 as c4,
     D1.c1 as c5
from
     SAWITH0 D1
order by c3, c2, c4
```

This report shows vertical federation between Essbase and a relational source. It

demonstrates that you can drill from aggregated data in Essbase cubes into detail data in relational sources.

Practice 18-4: Adding Essbase Measures to a Relational Model

Goal

To add an Essbase measure to an existing fact table with a relational source

Scenario

You add a forecast measure from an Essbase cube to the Fact-Sales table in the SupplierSales business model.

Time

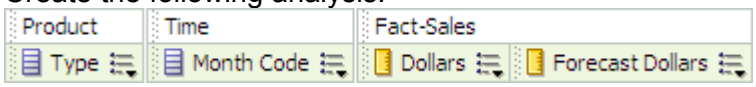
15 minutes

Tasks

1. Add a new logical table source to the Dim-Product logical table.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. In the Physical layer, right-click **Essbase** and select **Expand All**.
   c. In the Business Model and Mapping layer, expand **SupplierSales > Dim-Product**.
   d. Drag the **Product Type** column from Forecast in the Physical layer to the **Type** column in Dim-Product. This creates a new logical table source named **Forecast** for the Dim-Product logical table.
2. Add a new logical table source to the Dim-Time logical table.
   a. In the Business Model and Mapping layer, expand **SupplierSales > Dim-Time**.
   b. Drag the **Month Code** column from Forecast in the Physical layer to the **Month Code** column in Dim-Time. This creates a new logical table source named **Forecast** for the Dim-Time logical table.
3. Add a new logical column and logical table source to the Fact-Sales logical table.
   a. In the Business Model and Mapping layer, expand **SupplierSales > Fact-Sales**.
   b. Drag the **Forecast Dollars** column from Forecast in the Physical layer to the **Fact-Sales** logical table. This creates a new logical column named **Forecast Dollars** and a new logical table source named **Forecast** for the Fact-Sales logical table.
   c. Double-click the **Forecast Dollars** logical column.
   d. Click the **Aggregation** tab and set the aggregation rule to **SUM**.
   e. Click **OK**.
   f. Drag **Forecast Dollars** to **SupplierSales > Fact-Sales** in the Presentation layer.
4. Set the logical levels for the new logical table sources.
   a. Expand **Dim-Product > Sources**.
   b. Double-click the **Forecast** logical table source.
   c. Click the **Content** tab.
   d. Set the logical level to **Type** for the Product dimension.
   e. Click **OK**.
   f. Expand **Dim-Time > Sources**.
   g. Double-click the **Forecast** logical table source.
   h. Click the **Content** tab.
   i. Set the logical level to **Month** for the Time dimension.

j. Click **OK**.
k. Expand **Fact-Sales > Sources**.
l. Double-click the **Forecast** logical table source.
m. Click the **Content** tab.
n. Set the logical level to **Type** for the Product dimension.
o. Set the logical level to **Month** for the Time dimension.
p. Click **OK**. The Forecast Dollars measure now can be displayed in reports where the aggregation level is product type and above, or month code and above.
q. Save the repository.
r. Check consistency. Fix any errors or warnings before proceeding.
s. Close the repository.
t. Leave the Administration Tool open.

5. Create an analysis to check your work.
   a. Use Fusion Middleware Control Enterprise Manager to upload the ABC repository and restart Oracle BI components. If you need help, refer to steps from earlier practices.
   b. Return to Analysis Editor.
   c. Select the **SupplierSales** subject area.
   d. Create the following analysis:



   e. Check Results.

Type	Month Code	Dollars	Forecast Dollars
Baking	200,801.00	$269,183	$271,234
	200,802.00	$287,437	$289,782
	200,803.00	$312,651	$316,107
	200,804.00	$293,180	$298,547
	200,805.00	$331,326	$336,265
	200,806.00	$316,510	$321,441
	200,807.00	$317,913	$320,904
	200,808.00	$314,553	$318,544
	200,809.00	$290,870	$297,708
	200,810.00	$359,976	$363,307
	200,811.00	$297,091	$301,131
	200,812.00	$315,993	$323,062
	200,901.00	$336,281	$333,398
	200,902.00	$333,677	$329,676
	200,903.00	$334,120	$328,551
	200,904.00	$214,758	$210,930
Beef	200,801.00	$271,197	$272,431
	200,802.00	$320,671	$323,016
	200,803.00	$316,726	$320,182
	200,804.00	$307,958	$313,324
	200,805.00	$328,027	$332,965
	200,806.00	$320,404	$325,336
	200,807.00	$341,216	$344,206
	200,808.00	$334,479	$338,470
	200,809.00	$300,419	$307,258

   f. Sign out of Oracle BI.

g. Check the query log. As expected, data is retrieved from both the Essbase source and the relational source.

```
----------- Sending query to database named orcl (id:
<<4655>>), connection pool named SUPPLIER CP: [[

select T502.ITEMTYPE as c1,
     T862.MONTHCODE as c2,
     sum(T744.DOLLARS) as c3
from
     MONTHS T862 /* Dim_MONTHS */ ,
     D1_PRODUCT_TYPE T502 /* Dim_D1_PRODUCT_TYPE */ ,
     D1_ORDER_AGG1 T744 /* Fact_D1_ORDER_AGG1 */
where  ( T502.TYPECODE = T744.TYPEKEY and T744.PERKEY =
T862.MONTHCODE )
group by T502.ITEMTYPE, T862.MONTHCODE
order by c1, c2

]]
[2010-08-20T11:21:37.000+00:00] [OracleBIServerComponent]
[TRACE:2] [USER-18] [] [ecid:
0000IeFEsQn7i4WzLwFS8A1CQlb^0002n2] [tid: 2730] [requestid:
4a52000e] [sessionid: 4a520000] [username: weblogic] --------
----------- Sending query to database named Essbase (id:
<<4724>>), connection pool named Connection Pool: [[
with
  set [_Product2]  as '[Product].Generations(2).members'
  set [_Time2]  as '[Time].Generations(2).members'
select
  { [Measures]
  } on columns,
  NON EMPTY {crossjoin({[_Product2]},{[_Time2]})} properties
GEN_NUMBER on rows
from [Forecast.Forecast]
```

Practices for Lesson 19: Security

Lesson 19

Practices for Lesson 19

Lesson Overview

In these practices, you will learn how to implement Oracle BI security.

Lab 19-1: Exploring Default Security Settings

Goal

To explore the default security settings for Oracle Business Intelligence

Scenario

During installation, three Oracle Business Intelligence security controls are preconfigured with initial (*default*) values to form the default security model. The security controls include:

- An embedded directory server functioning as an *identity store* designed to hold all user and group definitions, required to control authentication
- A file-based *policy store* designed to hold the application role and permission grant mappings to users and groups, required to control authorization
- A file-based *credential store* designed to hold all user and system credentials, required to control authentication or authorization

Before you implement data access security in the Oracle BI repository, you explore these default security settings.

Time

10 minutes

Tasks

1. Log in to the WebLogic Server Administration Console. This console is used to manage users and groups for the embedded LDAP server that serves as an out-of-the-box identity store.
   a. Open a new tab on the browser and enter the following URL: **http://localhost:7001/console**.
   b. Log in as **weblogic** with password **welcome1**. During installation, you are prompted for a username and password to use as an Oracle BI Administrator. In this training environment, the Administrator user created during installation was weblogic. This is an arbitrary choice and there is nothing special about the name. This user has administrative privileges across the Oracle BI stack. This includes the OBI repository, the OBI presentation catalog, BI Publisher, RTD, Essbase, the identify store in WebLogic, and the Policy Store in Fusion Middleware Enterprise Manager.
2. Explore default settings for providers in the WebLogic security realm.
   a. On the left side of the console, under Domain Structure, notice there is a single WebLogic domain named bifoundation_domain into which all the BI applications are deployed.
   b. Click **Security Realms**.
   c. Notice that there is a single default security realm named myrealm. The OBI installer installs a single domain with a single security realm in it. A security realm is a container for the mechanisms that are used to protect WebLogic resources. This includes users, groups, security roles, security policies, and security providers. Whereas multiple security realms can be defined for the BI domain, only one can be active, meaning designated as the default realm, at any given time.
   d. Click **myrealm** to view the default settings.
   e. Click the **Providers** tab.

f. Notice that there is a default WebLogic Authentication Provider. An authentication provider establishes the identity of users and system processes, transmits identity information, and serves as a repository for identity information from which components can retrieve it. Oracle Business Intelligence is configured to use the directory server embedded in Oracle WebLogic Server as the default security provider. Alternate security providers can be used if desired and managed in the Oracle WebLogic Administration Console, but the WebLogic authentication provider is used by default.

g. Notice that there is a default WebLogic Identity Assertion Provider. WebLogic Identity Assertion Provider is used primarily for Single Sign On and is not covered in this training.

3. Explore default settings for users.

a. Click the **Users and Groups** tab.

b. Click the **Users** subtab. The default identity store is pre-seeded with user names specific to Oracle Business Intelligence. These default user names are provided as a convenience so that you can begin using the Oracle Business Intelligence software immediately after installation but you are not required to maintain the default names in your deployment.

c. Notice the weblogic user. This is the administration user created during the installation process. A single administrative user is shared by Oracle Business Intelligence and Oracle WebLogic Server. This username is created during installation, can be any desired name, and therefore does not need to be “Administrator.” The password is likewise provided during installation. In the default security configuration, an administrative user is a member of the BIAdministrators group and has all rights granted to the Oracle Business Intelligence Administrator user in earlier releases, with the exception of impersonation. An administrative user cannot impersonate other users. An administrative user is also a member of the Oracle WebLogic Server default Administrators group, which enables this user to perform all its administration tasks, including the ability to manage Oracle WebLogic Server’s embedded directory server and policy store.

d. Notice the OracleSystemUser user. Oracle Business Intelligence system components establish a connection to each other as OracleSystemUser instead of as the Administrator, the latter being the practice in earlier releases. Using a trusted system account, such as OracleSystemUser, to secure communication between Oracle BI components enables you to change the password of your deployment’s system administrator account without affecting communication between these components. The name of this user is the default, and it can be changed or a different user can be created for the purpose of inter-process communication. This is a highly privileged user whose credentials should be protected from non-administrative users.

e. Notice the other users such as Administrator_d and Administrator_f. These users are provided as part of the training environment and are not default Oracle BI users.

4. Explore default settings for groups.

a. Click the **Groups** subtab. Groups are logical ordered sets of users. Creating groups of users who have similar system resource access needs enables easier security management. Managing a group is more efficient than managing a large number of users individually. Oracle recommends that you organize your users into groups for easier maintenance. As you will see later in this practice, groups are then mapped to application roles in order to grant rights.

b. Notice the three default groups specific to Oracle BI: BIAdministrators, BIAuthors, and BIConsumers. These default groups are provided as a convenience so you can begin

using the Oracle Business Intelligence software immediately after installation, but you are not required to maintain the default names in your deployment. Members of the BIAdministrators group have permissions equivalent to those of the Administrator user of earlier releases. Members of the BIAuthors group have the permissions necessary to create content for others to consume. Members of the BIConsumers group have the permissions necessary to consume content created by others. Groups are nested in a hierarchy. Members of the BIAdministrators group are by default members of both other groups. Members of BIAuthors are members of BIConsumers.

c. Click the **Users** subtab.

d. Click the **weblogic** user.

e. Click the **Groups** tab.

f. Notice that the oracle11 user is a member of the two administrator groups: Administrators and BIAdministrators. Administrators is the WebLogic administrators group, which gives rights to administer WebLogic and FMW enterprise manager. Membership in both administration groups gives this user a single unified administration account for the entire product stack.

5. Explore the default settings for application roles. An application role defines a set of permissions granted to a user or group.

a. Return to the browser tab where Fusion Middleware Control Enterprise Manager is open. If it is not open, enter **http://localhost:7001/em** and log in as **weblogic/welcome1**.

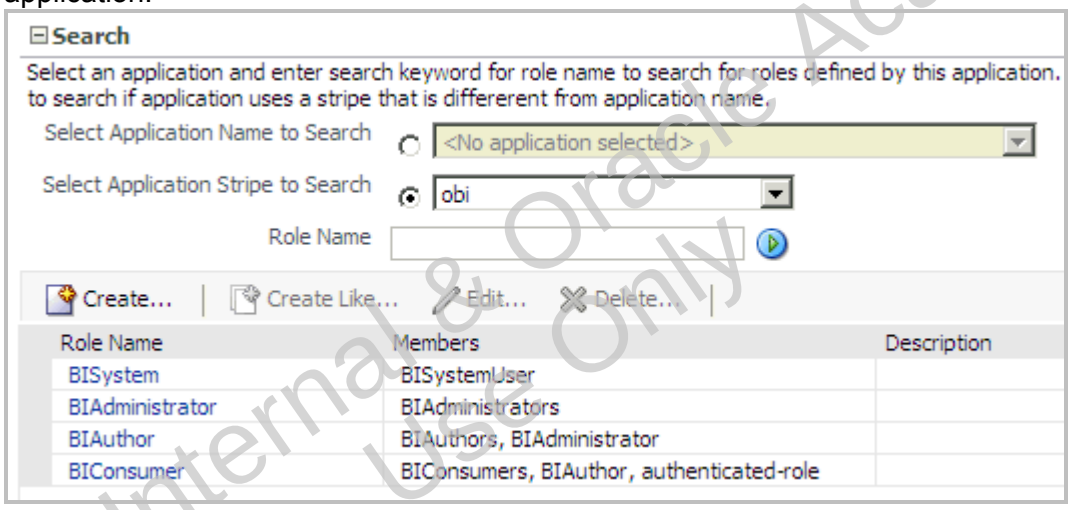
b. In the left pane, expand **WebLogic Domain**.

c. Right-click **bifoundation_domain** and select **Security > Application Roles**.

d. Select **Application Stripe to Search**.

e. Select **obi** in the drop-down list.

f. Click the **arrow** next to **Role Name** to see a list of application roles defined by this application.



g. Notice the following default application roles:

BIAdministrator: Grants administrative permissions necessary to configure and manage the Oracle Business Intelligence installation. Any member of the BIAdministrator *group* is explicitly granted this role and implicitly granted the BIAuthor and BIConsumer roles

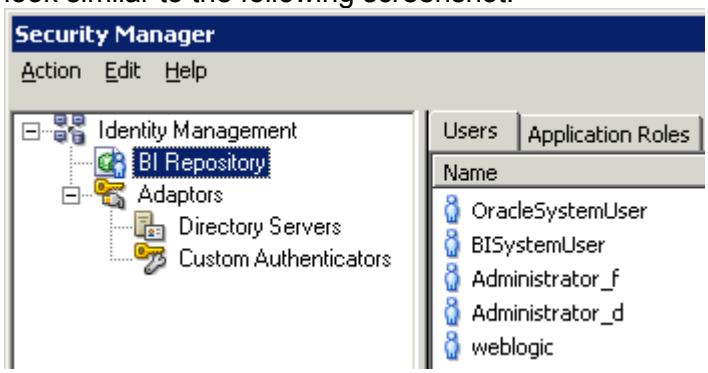
BIAuthor: Grants permissions necessary to create and edit content for others to

consume. Any member of the BIAuthor *group* is explicitly granted this role and implicitly granted the BIConsumer role.
BIConsumer: Grants permissions necessary to consume content created by others. Any member of the BIAuthor *group* is explicitly granted this role.
BISystem: Grants the permissions necessary to impersonate other users. This role is required by Oracle Business Intelligence system components for inter-component communication.

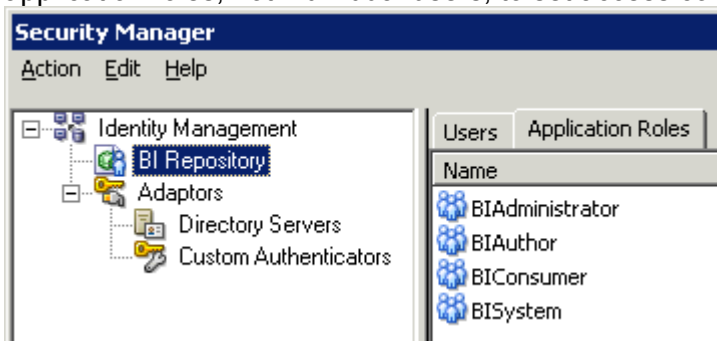
The default logical roles are mapped to groups in the default WebLogic LDAP. The groups are listed in the Members column. If you moved to a different LDAP server, rather than the default WebLogic LDAP server, you could map these roles to groups in the new LDAP server. Application roles are in the policy store, whereas groups are in the identity store.

6. Explore default settings for application policies.
   a. Right-click **bifoundation_domain** and select **Security > Application Policies**.
   b. Select the **obi** application stripe.
   c. Click the **arrow** next to **Permission** to see a list of permissions for each application role. The default file-based policy store is pre-seeded with the Oracle BI-specific permissions. All Oracle Business Intelligence permissions are provided and you cannot create additional permissions. These permissions are granted by the default application roles in the default security configuration. The out-of-the-box application role hierarchy and permission grants can be changed as needed. Also notice that these permissions are not the same as those used to define access to BI objects (metadata, dashboards, reports, etc.). Policy store permissions are only used to define what BI functionality the assigned roles can access.
   d. Notice, for example, that the BIAdministrator role has been granted the permission to administer repositories in BI Server, BI Publisher Server, BI Scheduler, and so forth.
   e. Notice that the BISystem user has been specifically granted two permissions that it requires for communicating with BI Server. These are powerful permissions, so they are mapped directly to the user rather than a role, so that any users mapped to other application roles do not accidently get these permissions.
7. Explore default security settings in the repository.
   a. Open the ABC repository in online mode with **welcome1** as the repository and user password.
   b. Because weblogic is a member of the BIAdministrators group, which is a member of the BIAdministrator Role with the permissions assigned to that role, weblogic can log in to the Administration Tool.
   c. Select **Manage > Identity** to open Security Manager.
   d. In the left pane, expand **Identity Management** and select **BI Repository**.
   e. Click the **Users** tab and notice that you can see the same set of users as those listed in the WebLogic Server Administration Console. The key point is that users are no longer in the repository. They are in the WebLogic LDAP, or whatever identify store your system is configured with. You must add a new user in the identity store, not in the repository, as you will see later in this set of practices. Security Manager should

look similar to the following screenshot:



f. Click the **Application Roles** tab to view all application roles in the policy store. As you will see later in this set of practices, you can now use the application roles to set access control permissions for repository objects. The recommended practice is to use application roles, not individual users, to set access control permissions.



g. Close the Security Manager without making any changes.

h. Close the repository.

i. Leave the Administration Tool open.

Practice 19-2: Creating Users and Groups

Goal

To create users and groups in the WebLogic identity store

Scenario

Groups are logical ordered sets of users. Managing a group is more efficient than managing a large number of users individually. Oracle recommends that you first organize all Oracle Business Intelligence users into groups that make sense for your organization's goals and map application roles to the groups in order to convey system privileges. The out-of-the-box identity store provided for managing users and groups is Oracle WebLogic Server's embedded directory server. You use WebLogic Administration Server Console to create users and groups.

Time

10 minutes

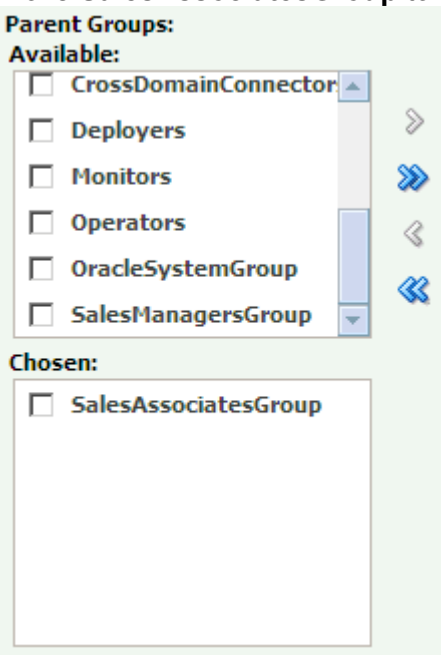
Tasks

1. Create new groups.
   a. Return to the WebLogic Server Administration Console, which should still be open in a browser tab. If not, enter **http://localhost:7001/console** and log in as **weblogic/welcome1**.
   b. Click **Security Realms**.
   c. Click **myrealm**.
   d. Click **Users and Groups**.
   e. Click the **Groups** subtab.
   f. Click the **New** button and create the following three groups:

Name	Description	Provider
SalesManagersGroup	**Sales Managers Group**	**DefaultAuthenticator**
SalesSupervisorsGroup	**Sales Supervisors Group**	**DefaultAuthenticator**
SalesAssociatesGroup	**Sales Associates Group**	**DefaultAuthenticator**

   g. Verify that the groups appear in the groups table. It may be necessary to click **Next** at the bottom of the page.

2. Create a group hierarchy.
   a. Click **SalesSupervisorsGroup**.
   b. Click the **Membership** tab.
   c. In the Available list, select **SalesAssociatesGroup**.
   d. Move **SalesAssociatesGroup** to the **Chosen** list.

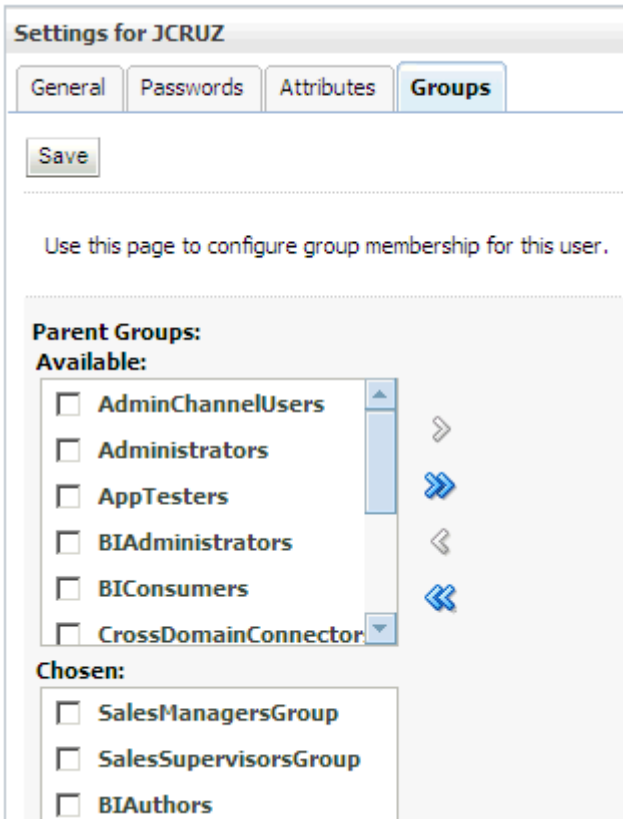


   e. Click **Save**. This means that any privileges or permissions assigned to the Sales Associates group will be inherited by the Sales Supervisors group.
   f. Use the browser Back button to return to the Groups page.
   g. Click **SalesManagersGroup**.
   h. Click the **Membership** tab.
   i. In the Available list, select **SalesAssociatesGroup**.
   j. Move **SalesAssociatesGroup** to the **Chosen** list.
   k. Click **Save**. This means that any privileges or permissions assigned to the Sales Associates group will be inherited by the Sales Managers group.
3. Create new users.
   a. Click the **Users** tab.
   b. Click the **New** button and create the following users and passwords:

Name	Description	Provider	Password
JCRUZ	**Jose Cruz**	**DefaultAuthenticator**	**JoseCruz1**
AZIFF	**Alan Ziff**	**DefaultAuthenticator**	**AlanZiff1**

4. Assign users to groups.
   a. Click **AZIFF**.
   b. Click the **Groups** tab.
   c. Move **BIAuthors** and **SalesAssociatesGroup** from the Available list to the Chosen list.
   d. Click **Save**.

e. Repeat to add JCRUZ to **BIAuthors, SalesSupervisorsGroup,** and **Sales ManagersGroup**.



Practice 19-3: Creating Application Roles

Goal

To create application roles in the policy store

Scenario

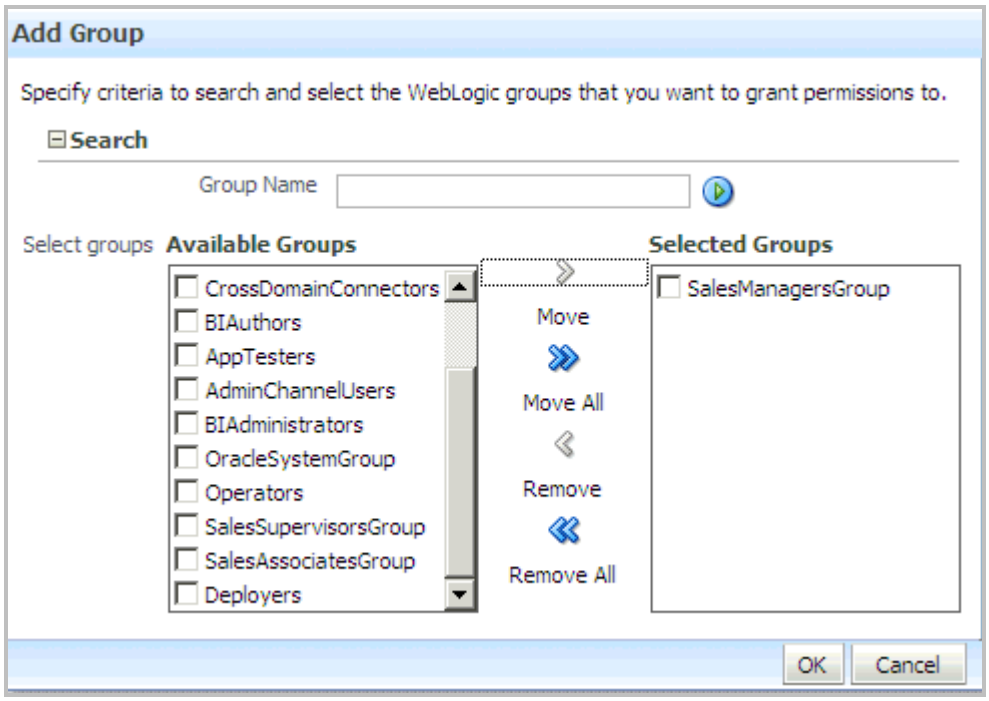
An application role conveys its permission grants to the users, groups, and application roles mapped to that role. Being mapped to an application role established membership in the role. Binding the permission grants to the application role streamlines the process of granting system privileges. Once the application role and permission grant definitions are established, you control system rights by managing membership in each role. Oracle recommends that you map groups to application roles and not individual users. Once mapped, all members of the group are granted the same rights. Controlling membership in a group reduces the complexity of tracking access rights for multiple individual users. You use Enterprise Manager Fusion Middleware Control to create application roles.

Time

20 minutes

Tasks

1. Create new application roles.
    a. Return to Fusion Middleware Control Enterprise Manager, which should still be open in a browser tab. If not, enter **http://localhost:7001/em** and log in as **weblogic** with password **welcome1**.
    b. Expand **WebLogic Domain** on the left.
    c. Right-click **bifoundation_domain** and select **Security > Application Roles**.
    d. Select **obi** in the **Select Application Stripe to Search** drop-down list.
    e. Click the **arrow** next to Role Name field. This shows all the BI application roles in the policy store.
    f. Click **Create**.
    g. In the Role Name field, enter **SalesManagersRole**.
    h. In the Display Name field, enter **Sales Managers Role**.
    i. Click **Add Group** to open the Add Group dialog box.
    j. Click the **blue arrow** next to the Group Name field to see a list of available groups from the identity store.
    k. In the Available Groups list, select **SalesManagersGroup** and move it to the Selected Groups list.

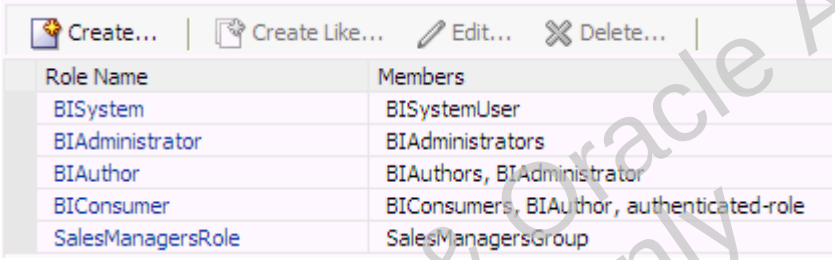


This means that any user who is a member of the selected group is assigned to this application role and receives any privileges or permissions assigned to the application role. It is possible to add individual users to a role, but best practice is to add groups, not individual users, to roles. In this example, Jose Cruz is a member of the Sales Managers group and is therefore assigned to the SalesManagersRole application role.

l. Click **OK**.

m. Click **OK** again to return to the Applications Role list.

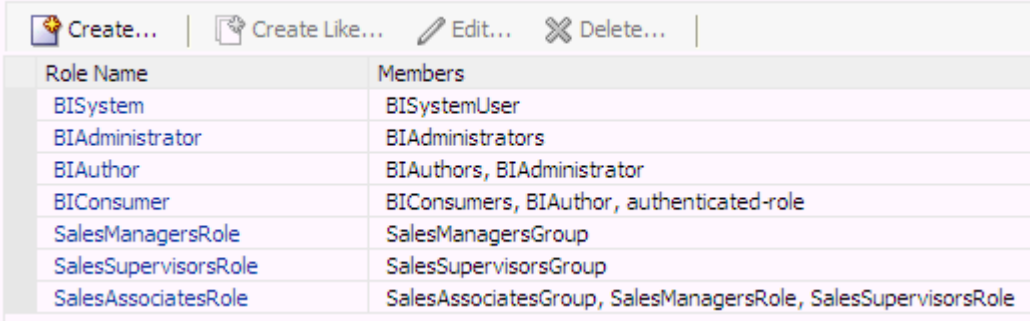
n. Verify that SalesManagersRole appears in the list with SalesManagersGroup listed as a member.



o. Repeat the steps to create the additional application roles and corresponding members listed in the following table. Please notice that you are adding a group and two application roles as members of the SalesAssociatesRole application role.

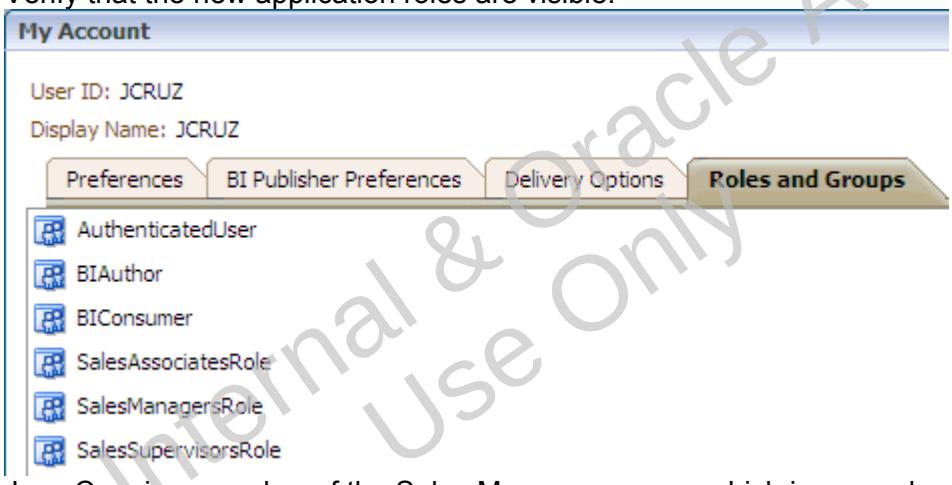
Application Role	Members
SalesSupervisorsRole	**SalesSupervisorsGroup**
SalesAssociatesRole	**SalesAssociatesGroup** **SalesSupervisorsRole** **SalesManagersRole**

p. Check your work. Your application roles should look similar to the following screenshot:



As with the groups that you created in the WebLogic identity store, you now have an application role hierarchy in the policy store. This means that any privileges or permissions assigned to the SalesAssociatesRole application role will be inherited by the SalesSupervisorsRole role and the SalesManagersRole role. What is the difference between users and groups created in the identity store in the WebLogic LDAP Server and application roles created in the policy store in Enterprise Manager? In the WebLogic LDAP server you have users and groups. An application role is a logical role that can be used within the application to secure content in a way that is independent of any particular LDAP server and the users and groups within that LDAP server. Security rules are built using application roles. If the underlying LDAP environment changes, the security rules persist. In a different LDAP environment, where group or user names might be different, you could remap the application roles to different groups or users and the BI security structure, built with application roles, would not be affected.

2. Verify that the new users and application roles are now visible in Oracle BI.
   a. Sign in to Oracle BI as **JCRUZ** with password **JoseCruz1**.
   b. In the upper-right corner, select **JCRUZ > My Account**.
   c. Click the **Roles and Groups** tab.
   d. Verify that the new application roles are visible.



Jose Cruz is a member of the Sales Managers group, which is a member of the Sales Managers application role, and a member of the Sales Supervisors group, which is a member of the SalesSupervisorsRole application role. Because both of these roles are members of the Sales Associate role, he is also a member of that role. By default, all BI users are also members of the out-of-the-box group, Authenticated Users, and the out-of-the-box application role, BIConsumer. Recall that you also added Jose Cruz to the BIAuthor application role.

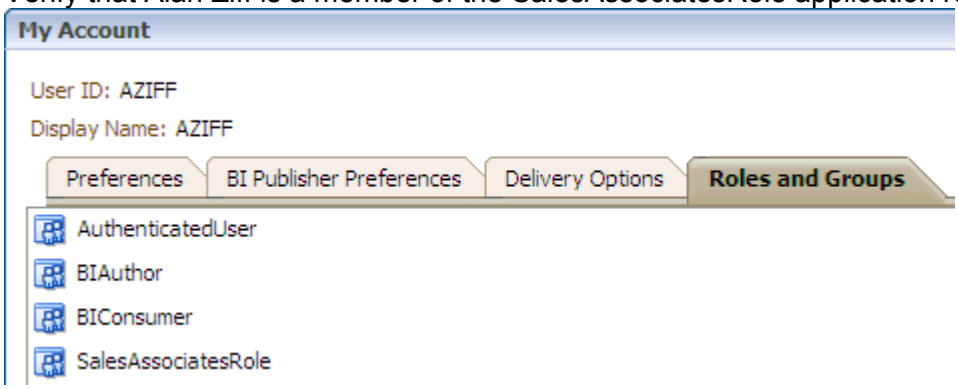
e. Click **OK** to close the My Account dialog box.

f. Sign out and sign back in as **AZIFF** with password **AlanZiff1**.

g. Select **AZIFF > My Account**.

h. Click **Roles and Groups**.

i. Verify that Alan Ziff is a member of the SalesAssociatesRole application role.



Alan Ziff is a member of the Sales Associates group, which is a member of the SalesAssociatesRole application role. Because the SalesAssociatesRole application role is the highest role in the hierarchy, he is not a member of the other two application roles in the hierarchy.

j. Click **OK** to close the My Account dialog box.

k. Sign out of Oracle BI. In sum, application roles serve a variety of purposes in both development and production environments. In a development environment, developers can be granted one or more of the roles. One approach is to build roles that will eventually be used in production, and then map developers to those roles for administering, building, and testing the development environment. As you will see in the next practice, you also use the logical application roles to secure access to repository objects and data. Therefore, application roles can be used to control access to both objects and functionality in the product. The value of using application roles comes from the fact that you can move the system that you have built between environments without having to rewire all the security. For example, you would not have to change security settings in your presentation catalog or repository. You can just remap your application roles to the target environment.

Practice 19-4: Setting Up Object Permissions

Goal

To set up object permissions in the repository

Scenario

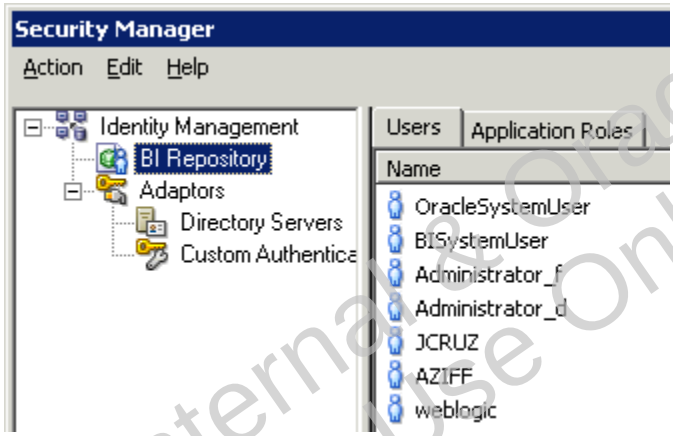
You can set up object permissions in your repository to control access to Presentation layer and Business Model and Mapping layer objects. You set object permissions using the Administration Tool. There are two approaches to setting object permissions: you can set permissions for particular users or application roles in the Security Manager, or you can set permissions for individual objects in the Presentation layer. In this practice you use both approaches. Setting up object permissions for individual users or application roles is useful when you want to define permissions for a large set of objects at one time. It is a best practice to set up object permissions for particular application roles rather than for individual users.

Time

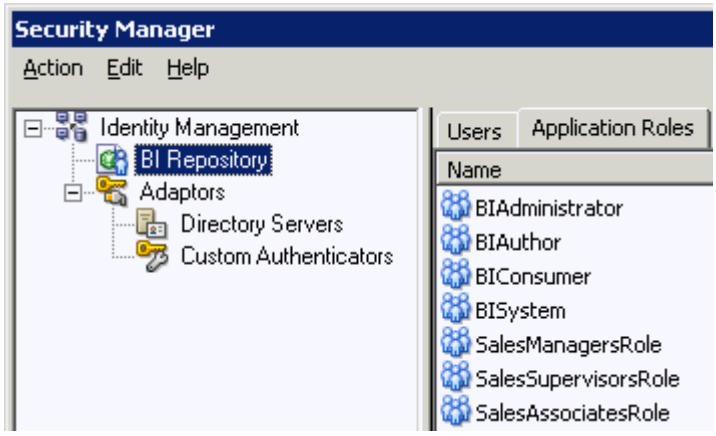
30 minutes

Tasks

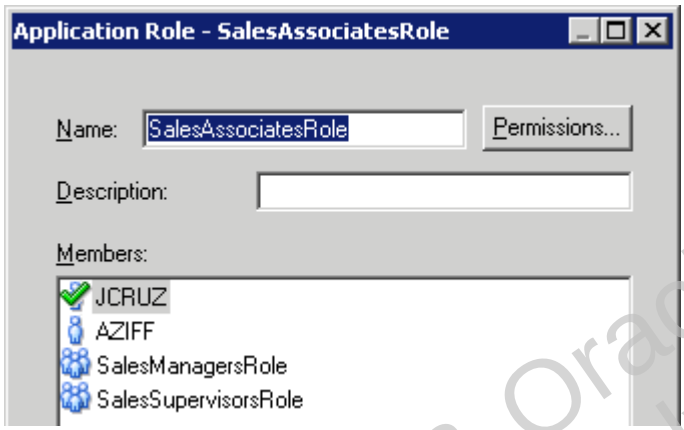
1. Verify that users and application roles are now visible in the repository.
   a. Return to Fusion Middleware Control Enterprise Manager and restart Oracle BI components.
   b. Open the repository in online mode, using **welcome1** as both the repository and user password.
   c. Select **Manage > Identity**.
   d. If necessary, expand Identity Management in the left pane and select **BI Repository**.
   e. Click the **Users** tab and verify that the users you created in the identity store are now visible.



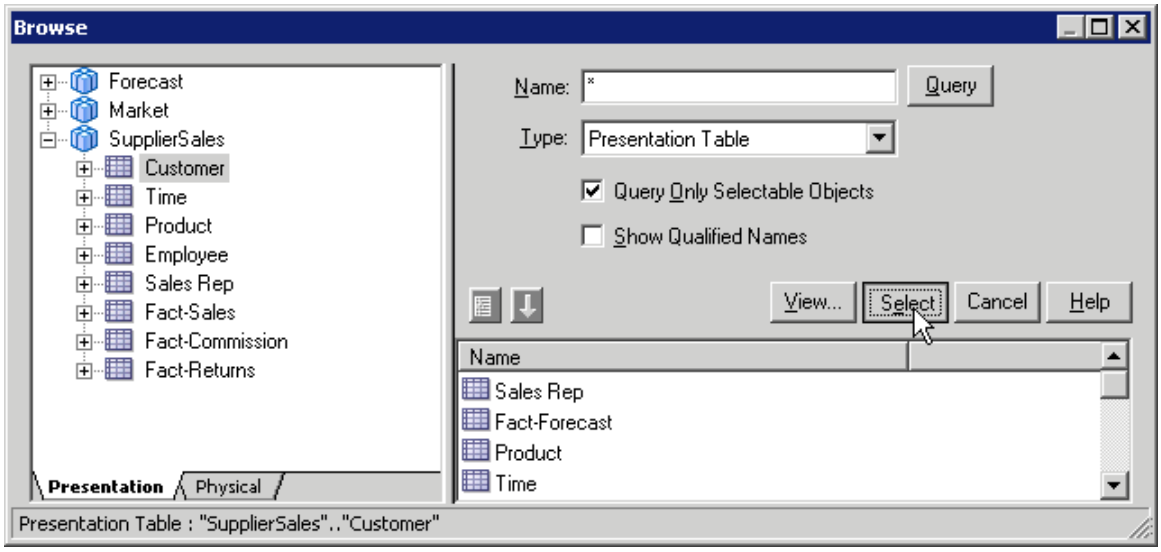
f. Click the **Application Roles** tab and verify that the application roles that you created are now visible.



2. Set permissions for a repository object. It is strongly recommended that you perform data access security tasks in the Administration Tool in online mode. If you must apply data access security in offline mode, be aware that users and application roles do not appear in the Administration Tool in offline mode unless you have first modified them in the Administration Tool in online mode.
   a. Double-click the **SalesAssociatesRole** application role.
   b. Click **Check Out**.
   c. Notice that the application roles and users assigned to this role are visible. Users who are members of these roles are therefore indirect members of this role as well.



   d. Click **Permissions** to open the User/Application Role Permissions dialog box.
   e. Click the **Object Permissions** tab.
   f. Click the **Name** field to open the Browse dialog box.
   g. In the left pane, verify that Presentation is selected at the bottom of the pane.
   h. Expand **SupplierSales**.
   i. Select the **Customer** presentation table in the left pane, and then click **Select** in the right pane.



The Customer presentation table is added to the Name field in the User/Application Role Permissions dialog box.

j. Click **No Access**.

Type	Name	Read	Read/Write	No Access
	"SupplierSales".."Customer"	○	○	◉
	Click here to add an object & set permission	○	○	○

k. Click **OK** to close the User/Application Role Permissions dialog box.

l. Click **Check Out**.

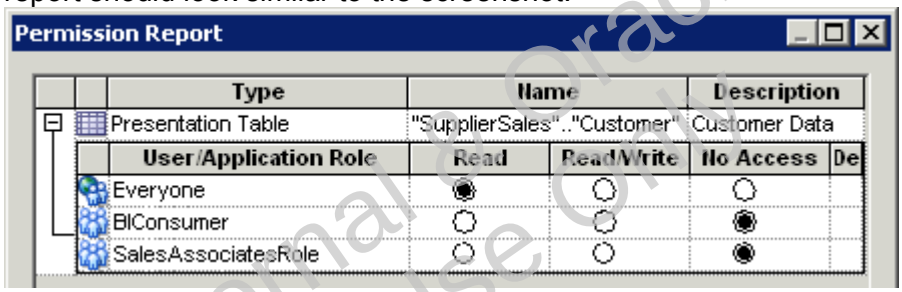
m. Click **OK** in the Application Role dialog box.

n. Repeat the steps and set the permissions for the **Customer** presentation table to **No Access** for the **BIConsumer** role.

o. Close the **Security Manager**.

p. In the Presentation layer, expand **SupplierSales**.

q. Right-click the **Customer** presentation table and select **Permission Report**. Your report should look similar to the screenshot:



r. Click **Cancel**.

s. Check in the changes.

t. Check consistency. Fix any errors or warnings before proceeding.

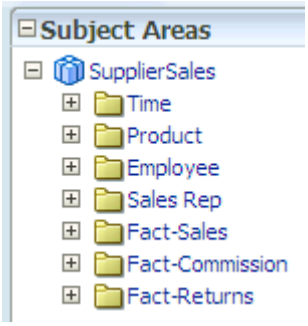
u. Save the repository.

v. Close the repository.

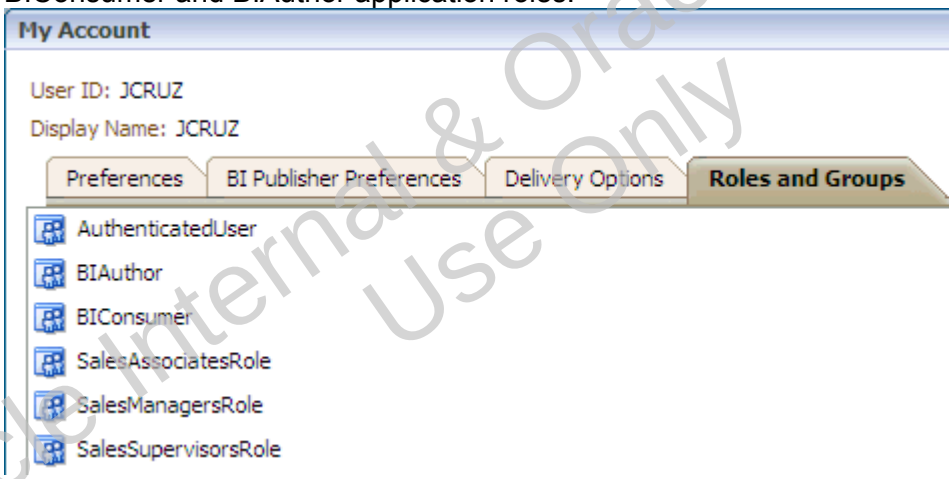
w. Click **OK** when you see the “In order for your online changes to take effect…” message.

x. Leave the Administration Tool open.

3. Check you work in Analysis Editor.
   a. Sign in to Oracle BI as **AZIFF** with password **AlanZiff1**. There is no need to restart the Oracle BI components.
   b. Select **AZIFF > My Account**.
   c. Click **Roles** and **Groups**.
   d. Notice that **Alan Ziff** is a member of the following application roles:
      BIAuthor
      BIConsumer
      SalesAssociatesRole
      Recall that you restricted access to the Customer presentation table for these three application roles. Given that, do you expect that Alan Ziff will have access to the Customer presentation table? Continue with the next step to verify your answer.
   e. Click **OK** to close the My Account dialog box.
   f. Click **New > Analysis**.
   g. Select the **SupplierSales** subject area.
   h. Notice that the **Customer** table is not visible.



   i. Sign out of Oracle BI and log back in as **JCRUZ** with password **JoseCruz1**.
   j. Select **JCRUZ > My Account**.
   k. Click **Roles** and **Groups**.
   l. Notice that **Jose Cruz** is a member of the **SalesAssociatesRole** role as well as the **SalesManagersRole** and **SalesSupervisorsRole** application roles, in addition to the BIConsumer and BIAuthor application roles.



   Recall that you restricted access to the Customer presentation table for the SalesAssociatesRole application role, but not for the SalesManagersRole or SalesSupervisorsRole. Given that, will Jose Cruz have access to the Customer presentation table? Continue with the next step to verify your answer.

m. Click **OK** to close the My Account dialog box.

n. Select **New > Analysis**.

o. Select the **SupplierSales** subject area.

p. Notice that the **Customer** table is not visible. Explanation: Jose Cruz is a member of the SalesManagersRole and SalesSupervisorsRole application roles, which are members of the SalesAssociatesRole application role. When you restricted access for the SalesAssociatesRole role the permissions were inherited by the other two roles. In the next set of steps you give the SalesSupervisorsRole role explicit access to the Customer table.

q. Sign out of Oracle BI.

4. Set permissions to give members of the SalesSupervisorsRole role explicit access to the Customer presentation table.

a. Return to the Administration Tool and open the **ABC** repository in online mode.

b. Open **Security Manager**.

c. Click the **Application Roles** tab.

d. Double-click **SalesSupervisorsRole** to open the Application Role dialog box.

e. Click **Check Out**.

f. Click **Permissions**.

g. Click the **Object Permissions** tab.

h. Click the **Name** field to open the Browse dialog box.

i. Expand **SupplierSales** in the left pane.

j. Double-click the **Customer** presentation table to add it to the User/Application Role Permissions dialog box.

k. Verify that the permission is set to **Read** (this is the default).

Type	Name	Read	Read/Write	No Access
	"SupplierSales".."Customer"	◉	○	○
	Click here to add an object & set permission	○	○	○

l. Click **OK** to close the User/Application Role Permissions dialog box.

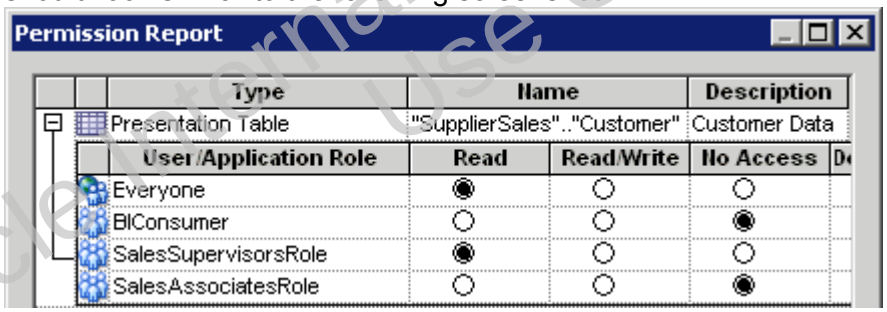
m. Click Check Out.

n. Click **OK** to close the Application Role dialog box.

o. Close **Security Manager**.

p. In the Presentation layer, expand **SupplierSales**.

q. Right-click **Customer** and select **Permission Report**. Verify that **SalesSupervisorsRole** has explicit Read access for the Customer table. Your report should look similar to the following screenshot:

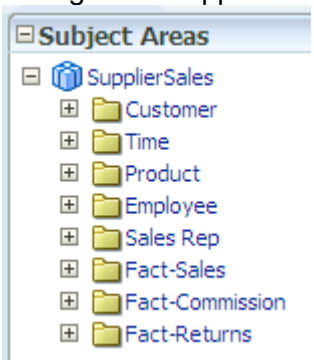


r. Click **Cancel**.

s. Check in the changes.

t. Check consistency. Fix any errors or warnings before proceeding.
u. Save the repository.
v. Close the repository.
w. Click OK.
x. Leave the Administration Tool open.

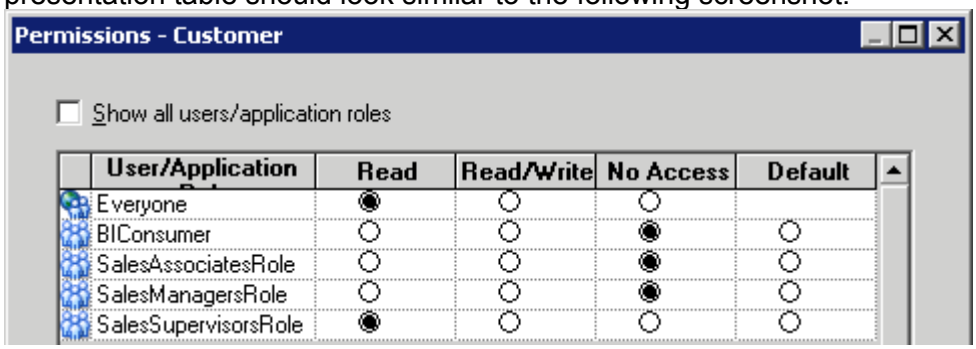
5. Check you work in Analysis Editor.
   a. Sign in to Oracle BI as **weblogic** with password **welcome1**.
   b. Click **New > Analysis**.
   c. Select **Reload Server Metadata**. Only users with administrative privileges can reload server metadata.
   d. Sign out and then sign back in as **JCRUZ** with password **JoseCruz1**.
   e. Click New > **Analysis**.
   f. Select the **SupplierSales** subject area.
   g. Verify that the **Customer** table is visible. In this example, Jose Cruz is a member of both the SalesAssociatesRole role and the SalesSupervisorsRole role. You restricted access to the Customer presentation table for the SalesAssociatesRole application role, but gave explicit Read access for the SalesSupervisorsRole role. Permissions granted explicitly to an application role take precedence over any permissions granted through other application roles.



   h. Sign out of Oracle BI.

6. Set conflicting permissions for application roles at the same level in a hierarchy.
   a. Return to the Administration Tool and open the **ABC** repository in online mode.
   b. Double-click the **Customer** presentation table to open the Presentation Table dialog box.
   c. Click **Check Out**.
   d. On the **General** tab, click **Permissions** to open the Permissions dialog box.
   e. Select **Show all users/application roles**.
   f. Select **No Access** for the **SalesManagersRole** application role.

g. Deselect **Show all users/application roles**. Permissions for the Customer presentation table should look similar to the following screenshot:

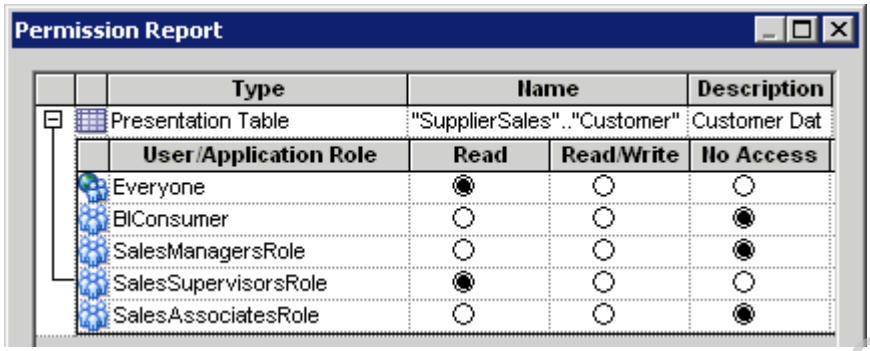


h. Click **OK** to close the Permissions dialog box.

i. Click **Check Out**.

j. Click **OK** to close the Presentation Table dialog box.

k. Right-click **Customer** and select **Permission Report**. Verify that **SalesManagersRole** has No Access for the Customer table. Your report should look similar to the following screenshot:



Jose Cruz now has conflicting permissions across two application roles. The SalesSupervisorsRole role gives him read access to the Customer table. The Sales Managers role gives him no access to the Customer table. Given that, will Jose Cruz have access to the Customer presentation table? Continue with the next steps to verify your answer.

l. Click **Cancel**.

m. Check in the changes.

n. Check consistency. Fix any errors or warnings before proceeding.

o. Save the repository.

p. Close the repository.

q. Click OK.

r. Leave the Administration Tool open.

7. Check you work in Analysis Editor.

a. Sign in to Oracle BI as **weblogic** with password **welcome1**.

b. Click **Analysis**.

c. Select the **SupplierSales** subject area

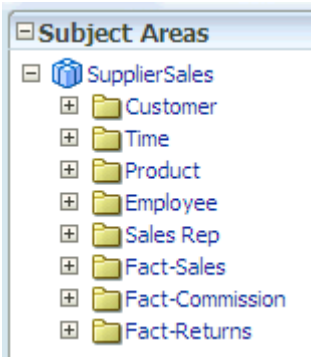
d. Select **Reload Server Metadata**. Only users with administrative privileges can reload server metadata.

e. Sign out and then sign back in as **JCRUZ** with password **JoseCruz1**.

f. Click **Analysis**.

g. Select the **SupplierSales** subject area.

h. Verify that the **Customer** table is still visible for Jose Cruz. If multiple application roles act on a user or application role at the same level, but with conflicting security attributes, the user or application role is granted the least restrictive security attribute. In this case, the least restrictive security attribute is read access for the Customer table.



i. Sign out of Oracle BI.

Practice 19-5: Setting Row-Level Security (Data Filters)

Goal

To set row-level security (data filters) in the repository

Scenario

Data filters provide a way to enforce row-level security rules in the repository. Data filters are set up in the repository by using the Administration Tool and are applied for a particular user or application role.

Data filters can be set for objects in both the Business Model and Mapping layer and the Presentation layer. Applying a filter on a logical object will affect all Presentation layer objects that use the object. If you set a filter on a Presentation layer object, it is applied in addition to any filters that might be set on the underlying logical objects. It is a best practice to set up data filters for particular application roles rather than for individual users.

In this practice you set a filter on the Customer presentation table so that customer data is visible only for those records where Jose Cruz or his direct reports are the sales representatives.

Time

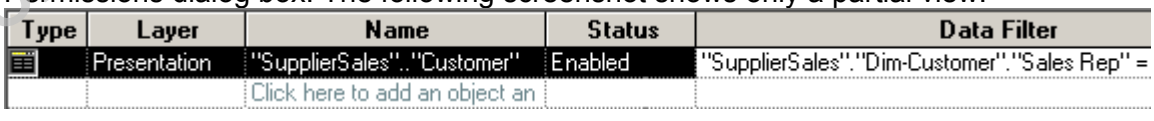
15 minutes

Tasks

1. Set a data filter for the Customer presentation table.
   a. Return to the Administration Tool and open the **ABC** repository in online mode.
   b. Open **Security Manager**.
   c. Click the **Application Roles** tab.
   d. Double-click **SalesSupervisorsRole** to open the Application Role dialog box.
   e. Click **Check Out**.
   f. Click **Permissions** to open the User/Application Role Permissions dialog box.
   g. Click the **Data Filters** tab.
   h. Click the **Name** field to open the Browse dialog box.
   i. Select the **Customer** presentation table to return to the User/Application Role Permissions dialog box.
   j. Click the **Data Filter** field.
   k. Click the **Edit Expression icon** to open the Expression Builder.
   l. Build the following expression. This sets a filter on the Customer presentation table so that customer data is visible only for those records where Jose Cruz or his direct reports are the sales representatives:

```
"SupplierSales"."Dim-Customer"."Sales Rep" = 'JOSE CRUZ' OR
"SupplierSales"."Dim-Customer"."Sales Rep" = 'ALAN ZIFF' OR
"SupplierSales"."Dim-Customer"."Sales Rep" = 'BETTY NEWER'
```

   m. Click **OK** to close Expression Builder to return to the User/Application Role Permissions dialog box. The following screenshot shows only a partial view.



   n. Click **OK** to close the User/Application Role Permissions dialog box.

o. Click **OK** to close the Application Role dialog box.
p. Close **Security Manager**.
q. Check in the changes.
r. Save the repository.
s. Close the repository.
t. Click OK.
u. Leave the Administration Tool open.

2. Check you work in Analysis Editor.
   a. Sign in to Oracle BI as **JCRUZ** with password **JoseCruz1**.
   b. Select the **SupplierSales** subject area and create the following analysis:

   c. Click **Results**.
   d. Verify that customer data is visible only for those records where Jose Cruz or his direct reports are the sales representatives.

Sales Rep	Customer	Dollars
ALAN ZIFF	Chang's Mongolian Grill	$331,217
	Globus Office	$212,182
	Half-Shell Restaurant	$880,096
	Times On Bay	$104,436
BETTY NEWER	Arloi Dee	$569,507
	Dentico's Italian Villa	$1,728
	Felix	$1,022,360
	Johnny's	$3,153,640
	Satterwhite Restaurant & Ctrng	$9,533
JOSE CRUZ	Aibonitos Restaurant	$185,964
	Baseline Cafe	$447,187
	Mayflower Cuisinier	$399,553
	Peter's Pub	$14,939

   e. Sign out of Oracle BI.

Practice 19-6: Setting Query Limits and Timing Restrictions

Goal

To manage the query environment by setting query limits in the repository

Scenario

You can manage the query environment by setting query limits (governors) in the repository for users or application roles. You want to prevent queries from consuming too many resources by limiting how long a query can run and how many rows a query can retrieve. You also want to regulate when individual users can query databases to prevent users from querying when system resources are tied up with batch reporting, table updates, or other production tasks. You set the maximum rows of any query to five rows, the maximum time of any query to 1 minute, and restrict access to a database on Sunday from 12:00 AM to 7:00 AM.

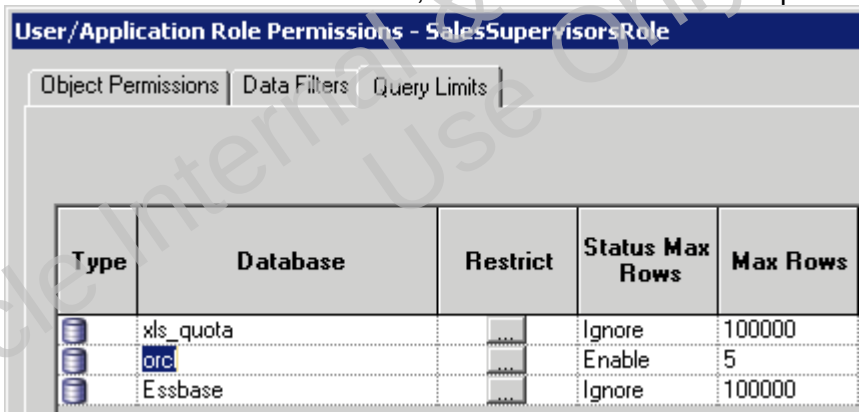
It is a best practice to set query limits for particular application roles rather than for individual users.

Time

20 minutes

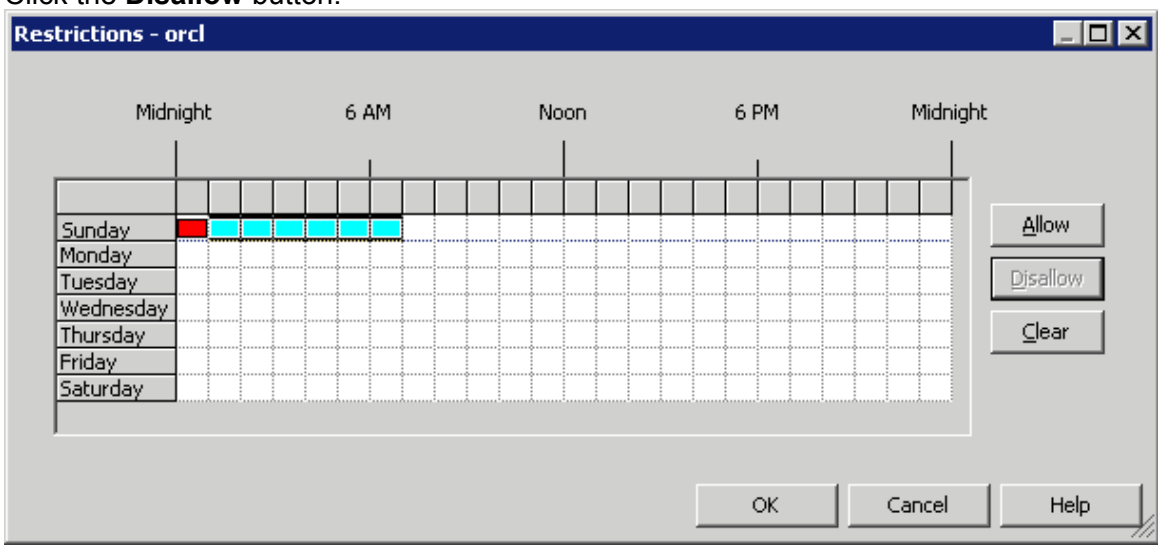
Tasks

1. Disallow queries that may consume too many system resources by setting query limits for the SalesSupervisorsRole application role.
   a. Return to FMW Enterprise Manager and restart Oracle BI components.
   b. Return to the Administration Tool and open the **ABC** repository in online mode.
   c. Open **Security Manager**.
   d. Click the **Application Roles** tab.
   e. Double-click **SalesSupervisorsRole** to open the Application Role dialog box.
   f. Click **Check Out**.
   g. Click **Permissions**.
   h. Click the **Query Limits** tab.
   i. Locate the **orcl** database and change its **Max Rows** value to **5**. This specifies the maximum number of rows that each query can retrieve from the orcl database for members of this **SalesSupervisorsRole** application role.
   j. In the Status Max Rows column, select **Enable** from the drop-down list.



2. Restrict the time period for which users can access specified repository resources.

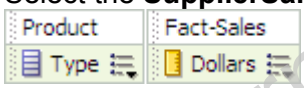
a. Click the ellipsis button ... in the Restrict column of the orcl database.
b. Highlight the blocks from Sunday at midnight to 7 AM.
Hint: Click the first box and drag to the last block.
c. Click the **Disallow** button.



If a time period is not highlighted, the access rights remain unchanged. If access is allowed or disallowed explicitly to one or more groups, the user is granted the least restrictive access for the time periods that are defined.

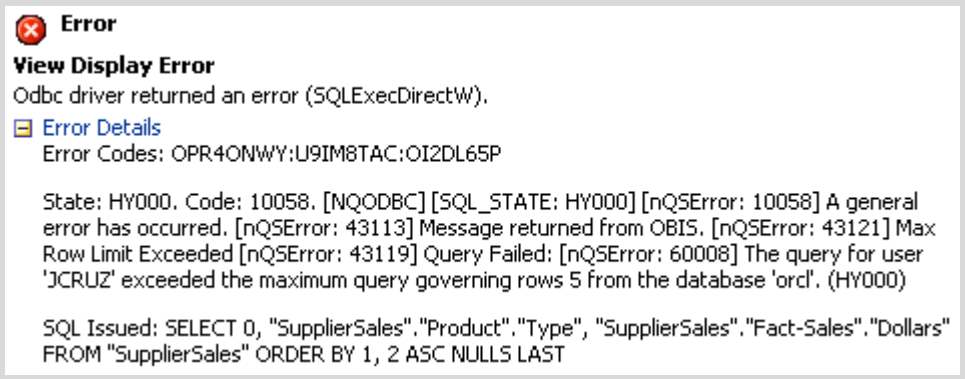
d. Click **OK** to close the Restrictions dialog box.
e. Click **OK** to close the User/Application Role Permissions dialog box.
f. Click **OK** to close the Application Role dialog box.
g. Close Security Manager.
h. Check in the changes
i. Save the repository.
j. Close the repository.
k. Click OK.
l. Leave the Administration Tool open.

3. Check your work in Analysis Editor.
a. Sign in to Oracle BI as **JCRUZ** with password **JoseCruz1**.
b. Select the **SupplierSales** subject area and create the following analysis:



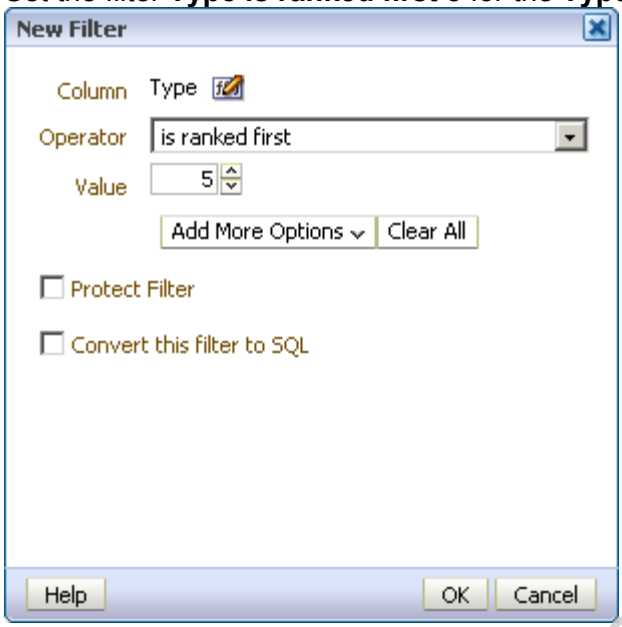
c. Click **Results**.

d. Expand **Error Details** to view message. The error message states that the maximum row limit is exceeded.



e. Click the **Criteria** tab.

f. Set the filter **Type is ranked first 5** for the **Type** column and click **Results**.



g. Verify that results are now returned.

Type	Dollars
Baking	$4,925,521
Beef	$4,916,016
Beverage	$4,398,107
Bread	$1,578,743
Cereal	$1,309,071

Because of the filter, the query returned only five rows, which does not exceed the maximum query limit.

h. Sign out of Oracle BI.

4. Deactivate the restrictions you set.
   a. Return to FMW Enterprise Manager and restart Oracle BI components.
   b. Return to the Administration Tool and open **ABC** repository in online mode.
   c. Open the **Security Manager**.
   d. Click the **Application Roles** tab.
   e. Double-click the **SalesSupervisorsRole** application role.

f. Click **Check Out**.
g. Click the **Permissions** button.
h. Click the **Query Limits** tab.
i. In the orcl row, set **Max Rows** to **100000** and select **Ignore** for **Status Max Rows**.
j. Click **OK**.
k. Close all dialog boxes and the Security Manager.
l. Expand **SupplierSales** subject area in the Presentation layer.
m. Double-click the **Customer** presentation table.
n. Click **Check Out**.
o. Click **Permissions**.
p. Select **Show all users/application roles**.
q. Change all permissions to **default**.
r. Click **OK**.
s. Click **Check Out**.
t. Click **OK** to close the Presentation Table dialog box.
u. Check in the changes.
v. Check consistency.
w. Save the repository.
x. Close the repository.
y. Leave the Administration Tool open.

Practices for Lesson 20: Cache Management

Lesson 20

Practices for Lesson 20

Lesson Overview

In these practices, you will enable query caching and inspect cache entries by using Cache Manager in the Administration Tool.

Practice 20-1: Enabling Query Caching

Goal

To enable query caching and inspect cache entries using Cache Manager

Scenario

You use Fusion Middleware Control Enterprise Manager to enable query caching and then use Cache Manager in the Administration Tool to inspect cache entries and analyze cache hits and non-hits.

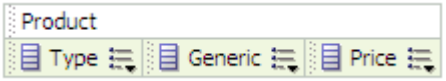
Time

25 minutes

Tasks

1. Set logging levels for users to allow you to track queries in query logs.
   a. Open the **ABC** repository in online mode with username **weblogic** and password **welcome1**.
   b. Select **Manage > Identity** to open Security Manager.
   c. In the left pane, expand **Identity Management > BI Repository**.
   d. In the right pane, double-click **JCRUZ** to open the User properties dialog box.
   e. Click **Check Out**.
   f. Set the logging level to **2** and click **OK**.
   g. Click **Check Out**.
   h. Repeat to set the logging level to 2 for **AZIFF**.
   i. Close Security Manager.
   j. Check in the changes.
   k. Save the repository.
   l. Close the repository.
2. Enable query caching.
   a. Return to Fusion Middleware Control Enterprise Manager and log in as **weblogic** with password **welcome1**.
   b. In the left pane, expand **Business Intelligence**.
   c. Click **coreapplication**.
   d. Click **Capacity Management**.
   e. Click the **Performance** tab.
   f. Click **Lock and Edit Configuration**.
   g. Select **Cache Enabled**.
   h. Click **Apply** to activate the changes.
   i. Click **Activate Changes**.
   j. Restart **Oracle BI components** to apply recent changes.
3. Create and run a request.
   a. Log in to Oracle BI as **JCRUZ** with password **JoseCruz1**.

b. Create the following analysis in the SupplierSales subject area:



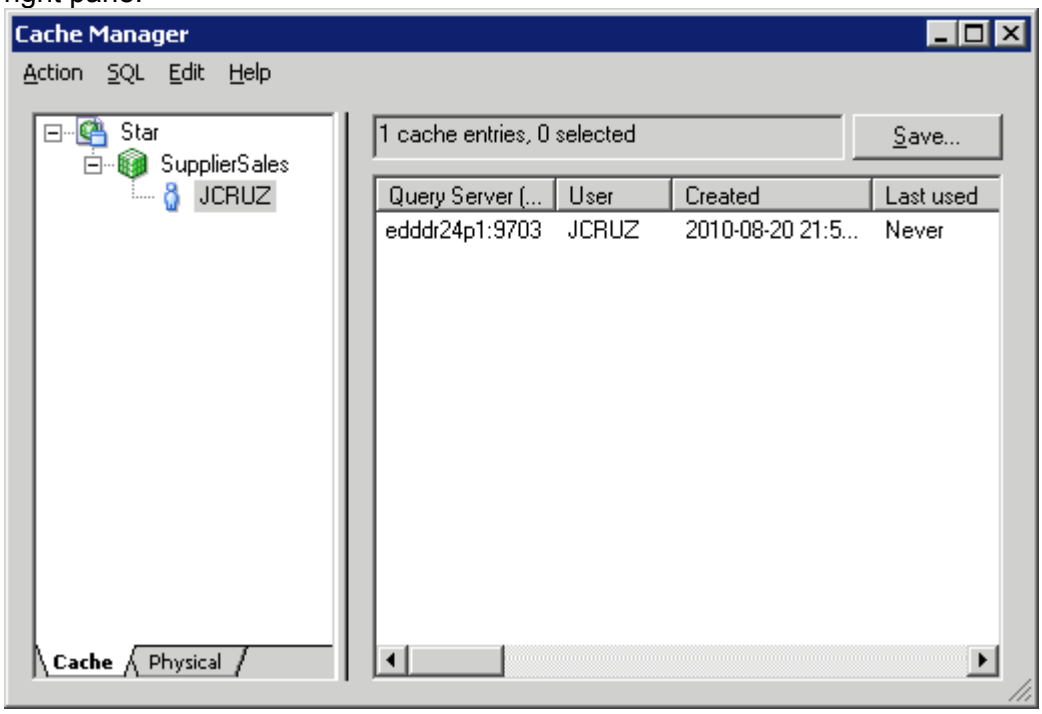
c. Click **Results**.

Type	Generic	Price
Baking	Beef Bouillon	$17.23
	Biscuit Mix	$22.37
	Breading Mix	$22.24
	Brown Sugar	$38.01
	Egg Substitute	$16.84
	Iodized Salt	$8.00
	Pancake Mix	$18.60
	Powdered Sugar	$15.39
	Stuffing Mix	$16.66
	Vanilla Extract	$10.60
	White Flour	$9.18
Beef	Breakfast Sausage Links	$30.13
	Canned Beef	$26.89
	Chorrizo Sausage	$23.98
	Deli-Style Pastrami	$38.44
	Frozen Sausage	$19.82
	Frozen Steaks	$75.97
	Frozen Veal Cutlets	$15.33
	Hamburger Patties	$144.58
	Lean Ground Beef Patties	$19.63
	Wieners	$16.39
Beverage	Coconut Cream Drink Mix	$79.10
	Orange Juice	$16.05
Bread	English Muffins	$14.34
	Frozen Bread Dough	$33.31

d. Sign out of Oracle BI.

4. Open the Cache Manager and verify that the query is listed as a cache entry.
   a. Open the **ABC** repository in online mode and select **Manage > Cache**.
   b. Expand **Star > SupplierSales**. By selecting the appropriate leaf of the tree in the left pane, you can limit the cache entries that appear in the right pane. The Cache Manager allows you to view cache entries by repository, subject area, and user.

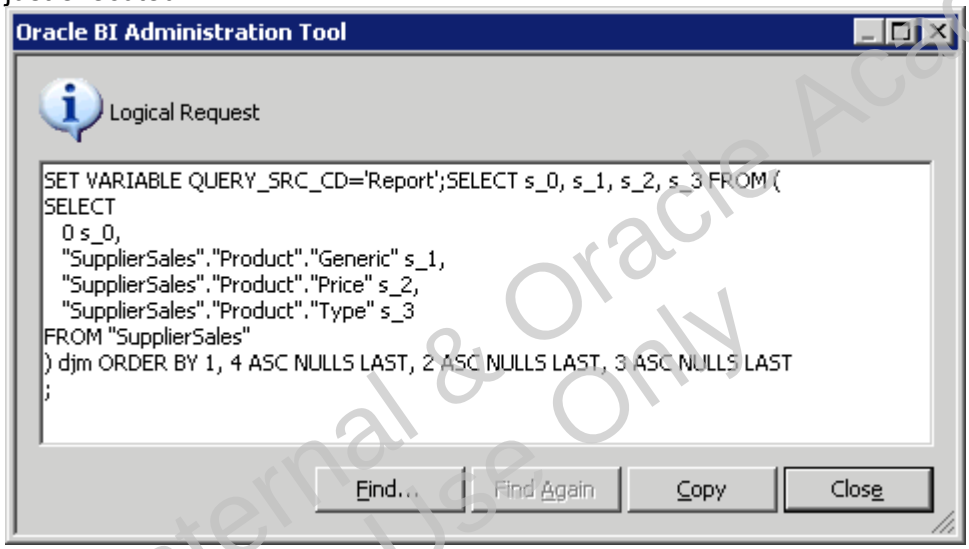
c. Click **JCRUZ** in the left pane. All cache entries associated with this user appear in the right pane.



d. Select the cache entry (there should only be one).

e. Select **SQL > Show**.

f. By inspecting the SQL you can verify that this is the cache entry for the query that you just executed.



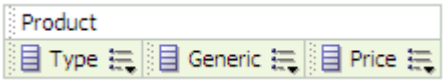
g. Click **Close** to close the SQL.

h. With this entry still selected, scroll to the right to view the columns. Notice that values for the Created and Last used columns are not the same. The Created column shows the date and time that the cache entry was created. Write down this date and time. The Last used column shows that the cache entry has never been used. This indicates that the results for this query were returned directly from the database and not derived from an existing query.

i. Leave the Cache Manager open.

j. Leave the ABC repository open in online mode.

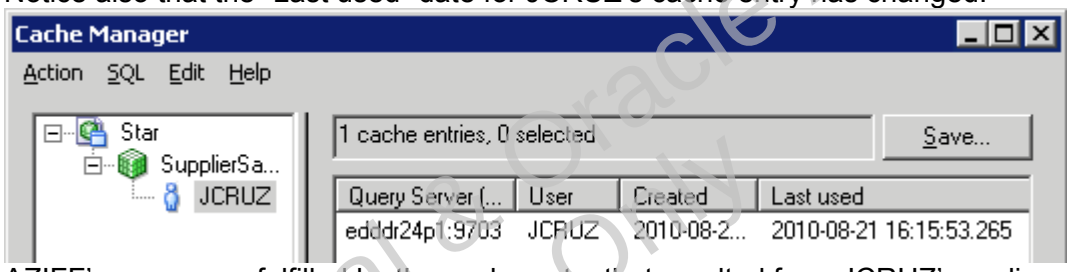
5. Inspect the cache file and ensure that the results of the query were stored as a file in the cache directory. The modified time of the file should coincide with the time that you originally created the request (the time you recorded above).
   a. Navigate to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\cache**. The file in this directory is the cache file that resulted from your query.
   b. Verify that the modified time of this file is the same as the time that you created the request.
6. Create and run the same analysis logged in as a different user.
   a. Sign in to Oracle BI as **AZIFF** with password **AlanZiff1**.
   b. Create the same analysis in the SupplierSales subject area that you created earlier and inspect the results.



   c. Click **Results**.
   d. Open the query log.
   e. Verify that you see a cache entry for the query initiated by AZIFF.

```
7bef0006] [sessionid: 7bef0000] [username: AZIFF] ------------
-------- The logical query hits the plan cache [[
```

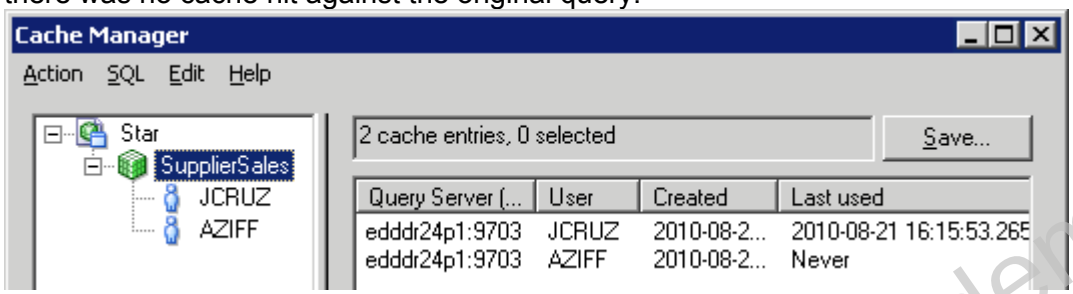
7. In response to AZIFF's request, determine whether a new cache entry was made in the Cache Manager and filed in the Cache directory. If a new entry was made in the Cache Manager and filed in the directory, then an existing cache was not used to satisfy his request.
   a. Return to the **ABC** repository open in online mode
   b. Select **Action > Refresh** in the **Cache Manager**.
   c. Notice that no cache entry is listed with AZIFF as the user.
   d. Notice also that the "Last used" date for JCRUZ's cache entry has changed.



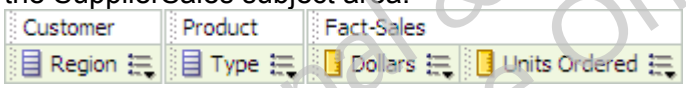
   AZIFF's query was fulfilled by the cache entry that resulted from JCRUZ's earlier query. Because AZIFF's request was identical to the cache, the server used the cache (cache hit) instead of processing against the database. The last used date is updated to reflect this.
   e. Leave Cache Manager open.
   f. Leave the repository open in online mode.
   g. Navigate to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\cache**.
   h. Notice that no new cache file is created as a result of AZIFF's request. This is because AZIFF's request was satisfied by an existing cache entry.

8. Create and run a new analysis to illustrate a non-cache hit using a dimension-only query.
   a. Return to Analysis Editor where you should still be logged in as **AZIFF** with password **AlanZiff1**.
   b. Create and run the following new analysis in the SupplierSales subject area:

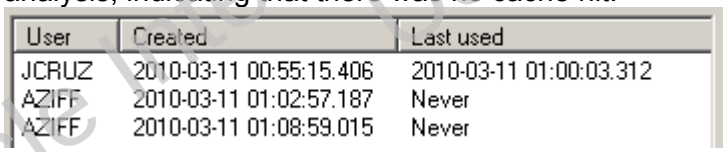
   Notice that this is a dimension-only query, meaning that no fact or measure is included in the query. Notice also that this query contains a subset of the columns in the previous query.
   c. Leave Oracle BI open.
9. Determine whether a new cache entry was made in the Cache Manager.
   a. Return to the **ABC** repository open in online mode.
   b. Select **Action > Refresh** in the **Cache Manager**.
   c. Select **Star > SupplierSales** in the left pane.
   d. Notice the new cache entry listed for AZIFF. The presence of a new entry indicates that there was no cache hit against the original query.



   Explanation: If a query is dimension only, meaning that no fact or measure is included in the query, only an exact match of the projection columns of the cached query will hit the cache. This behavior prevents false positives when there are multiple logical sources for a dimension table. In this example, only two columns, Type and Generic, matched the columns of the cached query, which included three columns: Type, Generic, and Price.
   e. Leave Cache Manager open.
   f. Leave the ABC repository open in online mode.
10. Create and run a new analysis.
   a. Return to Analysis Editor and create and run the following new analysis as AZIFF in the SupplierSales subject area:

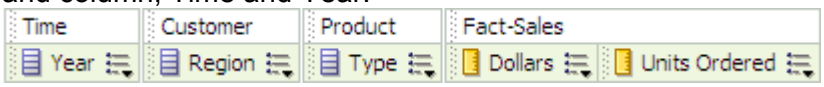


   b. Refresh the Cache Manager and verify that a new cache entry is created for this analysis, indicating that there was no cache hit.



   c. Return to Analysis Editor and create and run the following new analysis. Notice that this analysis contains all the columns from the previous query, plus one additional table

and column, Time and Year.

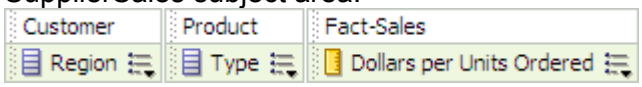


d. Return to Cache Manager and select **Action > Refresh**. Notice that a new cache entry is listed for AZIFF, indicating there was no cache hit against the previous query.

User	Created	Last used
JCRUZ	2010-03-11 00:55:15.406	2010-03-11 01:00:03.312
AZIFF	2010-03-11 01:02:57.187	Never
AZIFF	2010-03-11 01:08:59.015	Never
AZIFF	2010-03-11 01:12:27.921	Never

Explanation: The set of logical tables must match. To qualify as a cache hit, all incoming queries must have the same set of logical tables as the cache entry. In this example the Time logical table did not exist in the cached query.

e. Return to Analysis Editor and create and run the following new analysis in the SupplierSales subject area:

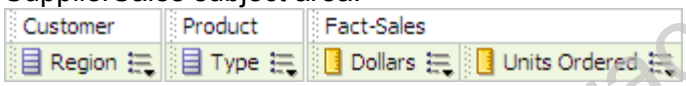


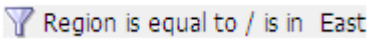
f. Refresh Cache Manager and notice that the “Last used” column is updated for AZIFF’s second query, indicating a cache hit.

User	Created	Last used
JCRUZ	2010-03-11 00:55:15.406	2010-03-11 01:00:03.312
AZIFF	2010-03-11 01:02:57.187	Never
AZIFF	2010-03-11 01:08:59.015	2010-03-11 01:16:01.171
AZIFF	2010-03-11 01:12:27.921	Never

Explanation: 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. In this example, Dollars per Units Ordered (dollars/units ordered) can be computed from Dollars and Units Ordered in the cached query that was hit.

g. Return to Analysis Editor and create and run the following new analysis and filter in the SupplierSales subject area:



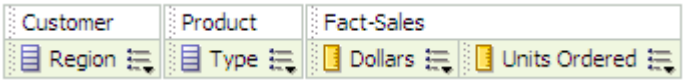


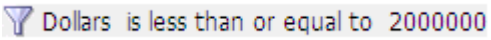
h. Refresh **Cache Manager** and watch the Last used column for the second AZIFF query. The column is updated, indicating a cache hit.

User	Created	Last used
JCRUZ	2010-03-11 00:55:15.406	2010-03-11 01:00:03.312
AZIFF	2010-03-11 01:02:57.187	Never
AZIFF	2010-03-11 01:08:59.015	2010-03-11 01:25:27.250
AZIFF	2010-03-11 01:12:27.921	Never

Explanation: For the query to qualify as a cache hit, the WHERE clause constraints must be either equivalent to the cached results, or a subset of the cached results. In this example, a query requesting fewer elements of an IN list cached query qualify for a cache hit. The WHERE clause in the cached query is `WHERE Region in ('East', 'West', 'Central')`. The WHERE clause in the query that hit the cache is a subset of the cached results: `WHERE Region in ('East')`.

i. Return to Analysis Editor and create and run the following new analysis and filter in the Supplier Sales subject area where Dollars is less than or equal to $2 million:



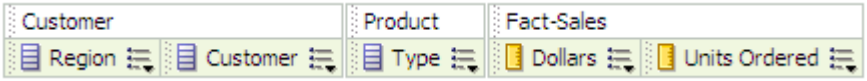


j. Refresh Cache Manager and watch the "Last used" column for the second AZIFF query. The column is updated, indicating a cache hit.

User	Created	Last used
JCRUZ	2010-03-11 00:55:15.406	2010-03-11 01:00:03.312
AZIFF	2010-03-11 01:02:57.187	Never
AZIFF	2010-03-11 01:08:59.015	2010-03-11 01:44:45.140
AZIFF	2010-03-11 01:12:27.921	Never

Explanation: Again, the `WHERE` clause constraints are a subset of the cached results. In this example, the `WHERE` clause contains a logical subset of a literal comparison.

k. Return to Analysis Editor and create and run the following analysis, which adds a new column, Customer, to the previous query:



l. Refresh Cache Manager. Notice that a new cache entry is created.

User	Created	Last used
JCRUZ	2010-03-11 00:55:15.406	2010-03-11 01:00:03.312
AZIFF	2010-03-11 01:02:57.187	Never
AZIFF	2010-03-11 01:08:59.015	2010-03-11 01:44:45.140
AZIFF	2010-03-11 01:12:27.921	Never
AZIFF	2010-03-11 01:57:58.078	Never

Explanation: 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. In this example, the Customer column did not exist in the cached query.

m. Sign out of Oracle BI.

n. Leave Cache Manager and the repository open for the next practice.

o. Navigate to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\cache** and verify there are now five files, one for each cache entry.

Practice 20-2: Modifying Cache Parameters

Goal

To use Fusion Middleware Control and the NQSConfig.ini file to modify cache parameters

Scenario

You use the Cache Manager to inspect the cache parameters. Then you modify the number of rows per cache, as well as the number of cache entries allowed. In addition to modifying cache parameters, you make certain tables noncacheable.

Outcome

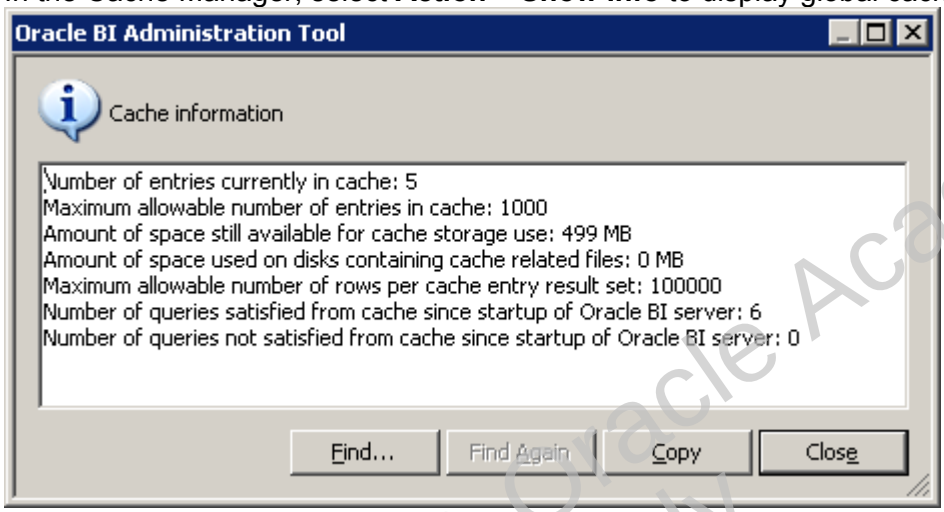
A decrease in the number of rows per cache, as well as a decrease in the number of cache entries allowed

Time

20 minutes

Tasks

1. Display global cache information in the Cache Manager.
   a. In the Cache Manager, select **Action > Show Info** to display global cache information.



   b. **Number of entries currently in cache** displays the current number of entries in your global cache. These entries may relate to multiple repositories.
   c. **Maximum allowable number of entries in cache** displays the maximum number of entries that can be in the cache.
   d. **Amount of space still available for cache storage use** displays the amount of space, in megabytes, still available for cache storage.
   e. **Amount of space used on disks containing cache related files** displays the total amount of space, in megabytes, used on the disk containing cache-related files (not just space used for the cache-related files). In this case, the amount of used space is less than 1 MB.
   f. **Maximum allowable number of rows per cache entry result set** displays the maximum number of rows allowed for each cache entry's result set. This number is set in the MAX_ROWS_PER_CACHE_ENTRY parameter in the NQSConfig.INI file.

g. **Number of queries satisfied from cache since startup of Oracle BI Server** displays the number of cache hits since the last time the Oracle BI Server was started.

h. **Number of queries not satisfied from cache since startup of Oracle BI Server** displays the number of cache misses, since the last time the Oracle BI Server was started.

i. Click **Close**.

j. Close the Cache Manager.

k. Close the repository.

2. Use Fusion Middleware Control Enterprise Manager to set query cache parameters. You can use Fusion Middleware Control to set the maximum number of cache entries in the query cache, as well as the maximum size for a single cache entry.

   a. Return to Fusion Middleware Control, which should still be open. If it is not open, enter the **http://localhost:7001/em** and log in to Fusion Middleware Control as **weblogic** with password **welcome1**.

   b. In the left pane, expand **Business Intelligence**.

   c. Click **coreapplication**.

   d. Click **Capacity Management**.

   e. Click the **Performance** tab.

   f. Click **Lock and Edit Configuration**.

   g. Change **Maximum cache entries** from 1000 to **10000**.

   h. **Apply** and **activate** your changes.

   i. Leave Enterprise Manager open, but do not restart to apply recent changes at this time.

3. Use NQSConfig.ini to view and manually edit additional query cache parameters.

   a. Navigate to **D:\bi\instances\instance1\config\OracleBIServerComponent\coreapplication_obis1**.

   b. Open **NQSConfig.ini**.

   c. Navigate to the **CACHE** section.

   d. Make the following modifications:

   e. Modify the MAX_ROWS_PER_CACHE_ENTRY parameter as follows:
   ```
   MAX_ROWS_PER_CACHE_ENTRY = 50000
   ```
   This parameter controls the maximum number of rows for any cache entry. 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. 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.

   f. Notice the DATA_STORAGE_PATH parameter. This parameter specifies one or more directories for query cache storage, and the maximum size for each storage directory. These directories are used to store the cached query results and are accessed when a cache hit occurs.

   g. Notice that the MAX_CACHE_ENTRIES parameter is changed to 10000. Changes made to cache configuration in FMW Enterprise Manager are written to this file.

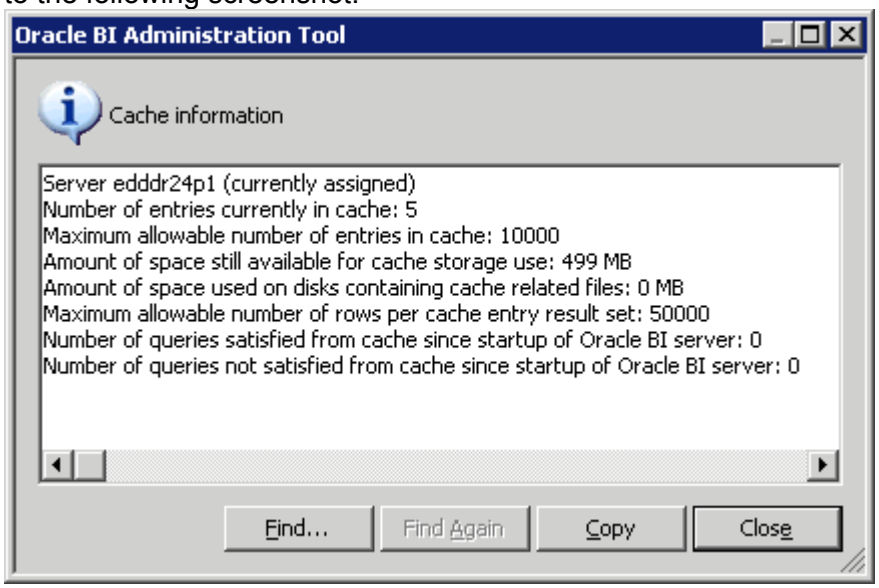
   h. Return to FMW Enterprise Manager and restart Oracle BI components.

4. Validate your changes in the Administration Tool.

   a. Return to the Administration Tool and open the **ABC** repository in online mode.

b. Select **Manage > Cache**.

c. Select **Action > Show Info** and ensure that changes are applied. Maximum allowable number of entries in cache should now equal **10000**. Maximum allowable number of rows per cache entry result should now equal **50000**. Your results should look similar to the following screenshot:



d. Click **Close**.

e. Close the Cache Manager.

5. Make some tables noncacheable, that is, if a request is made against them, there are no cache entries.

a. In the Physical layer, double-click the **orcl.SUPPLIER2.D1_CUSTOMER2** table.

b. Click **Check Out**.

c. Click the **General** tab.

d. Deselect the **Cacheable** check box.

e. Click **OK**.

f. Repeat this process for the **D1_ORDERS2** table.

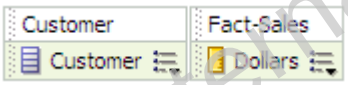
g. Check in the changes and save the repository. You do not need to check consistency.

h. Leave the repository open.

6. Test your work.

a. Log in to Oracle BI as **JCRUZ** with password **JoseCruz1**.

b. Create the following analysis in the SupplierSales subject area:



c. Click **Results**.

d. Leave Analysis Editor open.

e. Return to the repository open in online mode in the Administration Tool and open the Cache Manager.

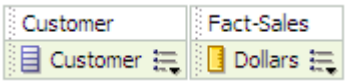
f. Verify that you do not see a cache entry for the analysis you just executed.

g. Close the Cache Manager.

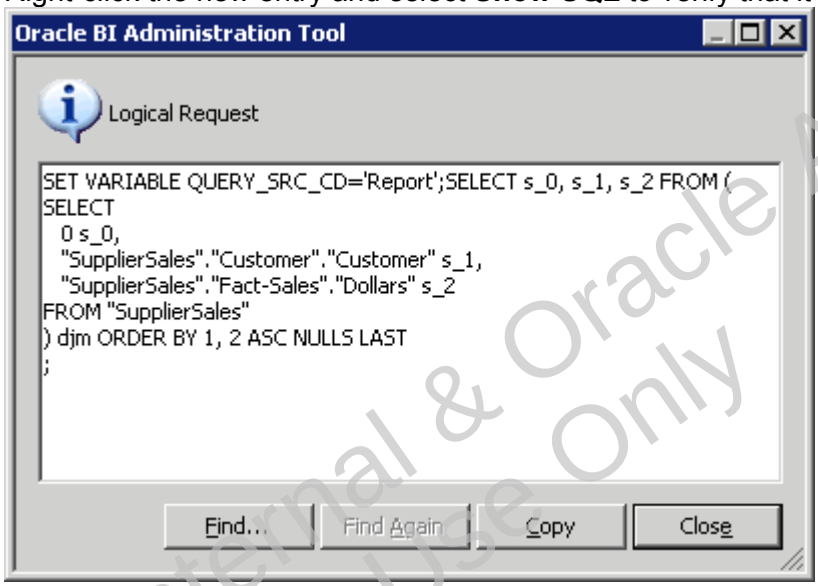
7. Make the tables cacheable.

a. In the Physical layer, double-click the **D1_CUSTOMER2** table.
b. Click **Check Out**.
c. Select the **General** tab.
d. Select the **Cacheable** check box.
e. Click **OK**.
f. Repeat this process for the **D1_ORDERS2** table.
g. Check in the changes and save the repository. You do not need to check for consistency.
h. Leave the repository open.

8. Test your work.
a. Return to Analysis Editor.
b. Sign out and sign back in as **weblogic** with password **welcome1**.
c. Click **Administration**.
d. Under Maintenance and Troubleshooting click **Reload Files and Metadata**.
e. Create and run a new analysis using the same columns from the SupplierSales subject area:



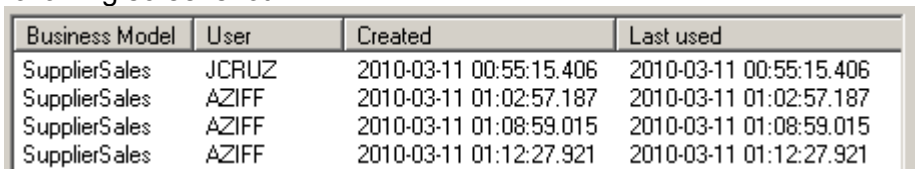
f. Return to the repository open in online mode in the Administration Tool and open the Cache Manager.
g. Verify that there is a new cache entry for the analysis executed by weblogic.
h. Right-click the new entry and select **Show SQL** to verify that it is the expected query:



i. Click **Close**.

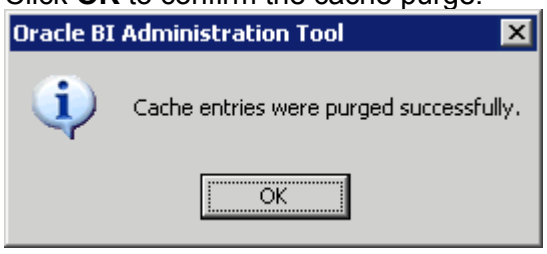
9. Alter how the Cache Manager displays information.
a. Select **Edit > Options**.
b. Deselect **Creation elapsed time**.
c. Use the **Up** button to move **Business Model** to the top of the options list.
d. Click **OK**.

e. Verify the changes in the Cache Manager. Your results should look similar to the following screenshot:

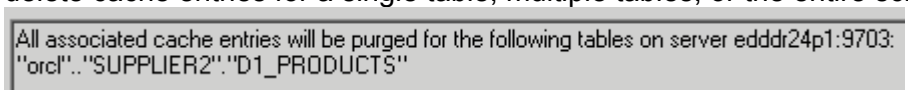


Business Model	User	Created	Last used
SupplierSales	JCRUZ	2010-03-11 00:55:15.406	2010-03-11 00:55:15.406
SupplierSales	AZIFF	2010-03-11 01:02:57.187	2010-03-11 01:02:57.187
SupplierSales	AZIFF	2010-03-11 01:08:59.015	2010-03-11 01:08:59.015
SupplierSales	AZIFF	2010-03-11 01:12:27.921	2010-03-11 01:12:27.921

10. Purge cache entries.
    a. Right-click the cache entry for **JCRUZ** and select **Purge**, or select **Edit > Purge**. In this pane, it is possible to purge a single cache entry, multiple entries, or all entries.
    b. Click **OK** to confirm the cache purge.



    c. Click the **Physical** tab at the bottom of the left pane.
    d. In the left pane, expand **orcl > Supplier2**.
    e. Select the **D1_PRODUCTS** table. A message appears on the right stating that all associated cache entries will be purged for this table. In this pane, it is possible to delete cache entries for a single table, multiple tables, or the entire schema.



    f. Select **Edit > Purge**.
    g. Click **OK** to confirm the cache purge.
    h. Click the **Cache** tab at the bottom of the left pane.
    i. Verify that the **AZIFF** cache entries are deleted.
    j. Close the Cache Manager.
    k. Leave the repository open in online mode.

Practice 20-3: Seeding the Cache

Goal

To create an analysis and an agent to seed the Oracle BI Server cache

Scenario

You have identified requests that are used frequently by sales representatives. To improve performance, you want to seed the cache with this data. In this practice, you create and save a query to populate the cache, and then create an agent to seed the cache. During this process, you use a programmatic ODBC call to purge the cache.

Time

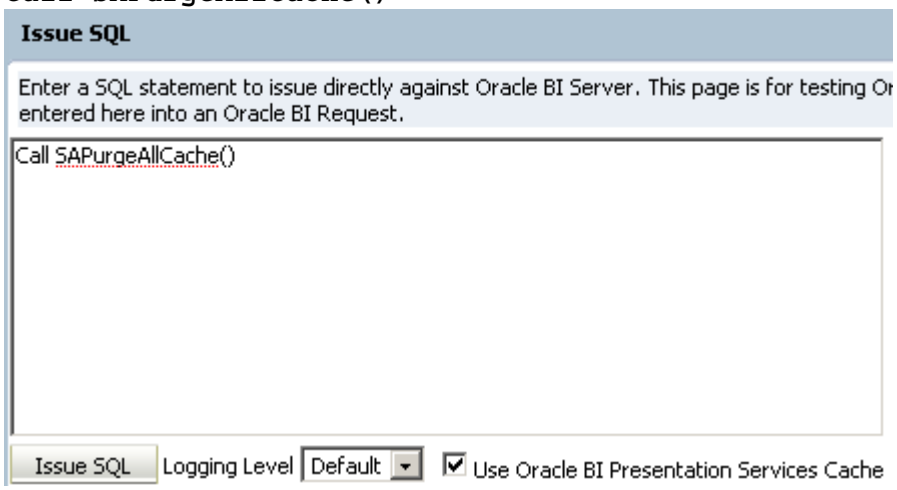
15 minutes

Tasks

1. Create a query to seed the cache.
   a. Return to Analysis Editor where you should still be signed in as **weblogic** with password **welcome1**.
   b. Create the following new analysis in the SupplierSales subject area:

   c. Save the request as **Cache Seed** in **My Folders**.
2. Use an ODBC procedure to purge the cache before seeding the cache. 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, the entire 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 may need to purge the cache entries associated with a specific database. **Note:** Only Oracle BI Administrators have the right to purge the cache. Therefore, scripts that call these ODBC-extension functions must run under an Oracle BI Administrator login ID.
   a. Click **Administration** on the top of the screen.
   b. Under Maintenance and Troubleshooting, click **Issue SQL** to issue SQL directly to the Oracle BI Server.

c. Enter the following command:
```
Call SAPurgeAllCache()
```



d. Click **Issue SQL**.

e. Verify that the operation succeeded.

RESULT_CODE	RESULT_MESSAGE
integer	varchar
1	[59118] Operation SAPurgeAllCache succeeded!

f. Click **Home** to navigate to the Oracle BIEE Home page.

g. Return to the ABC repository, which should still be open in online mode.

h. Open the **Cache Manager** and verify that there are no cache entries.

i. Leave the Cache Manager open.

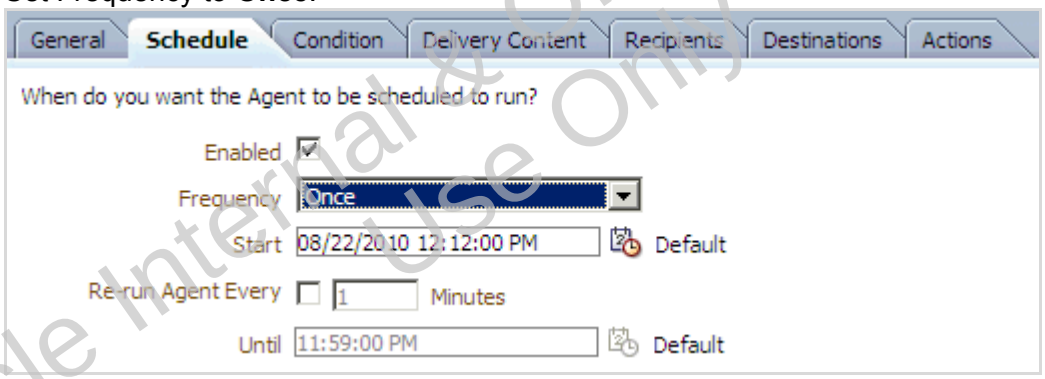
3. Create and schedule an Agent to seed the cache with the saved query. It is common to schedule an Agent to run immediately after a daily load to reseed the cache. For example, the data warehouse is loaded at midnight and the cache is purged during or after the load, then the cache is re-seeded by an Agent. In this example, you run the Agent immediately.

a. Return to the **Oracle BIEE Home** page.

b. Under Actionable Intelligence, click **Agent**.

c. Click the **Schedule** tab.

d. Set Frequency to **Once**.



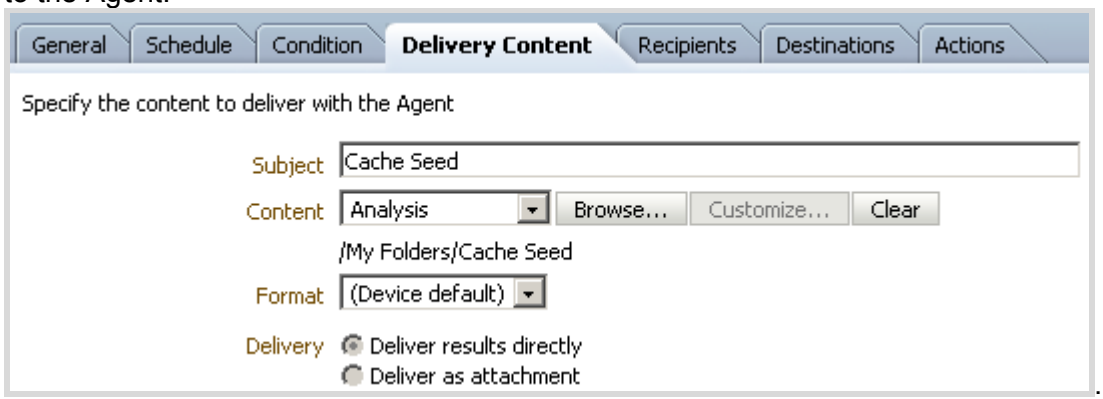
e. Click the **Delivery Content** tab.

f. Enter **Cache Seed** in the Subject field.

g. Verify that Content is set to **Analysis**.

h. Click **Browse**.

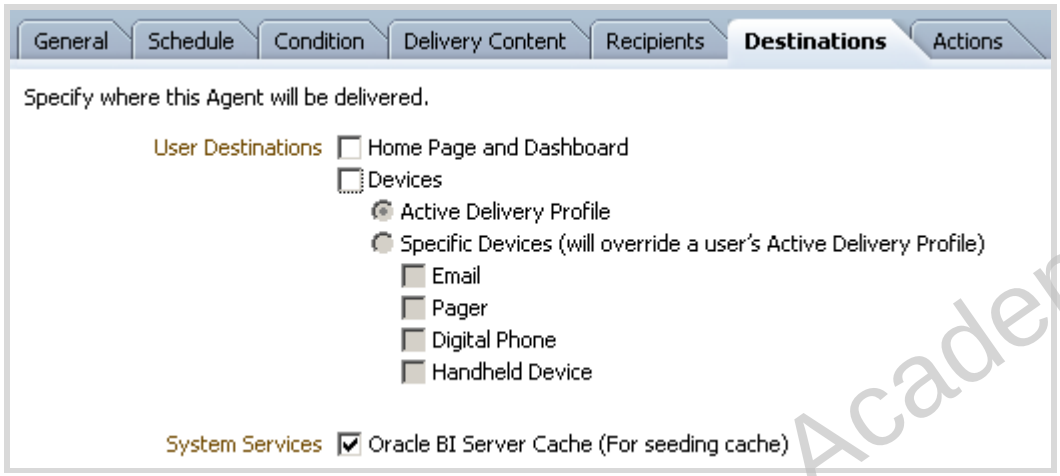
i. Select the **Cache Seed** request under My Folders and click **OK**. The request is added to the Agent.



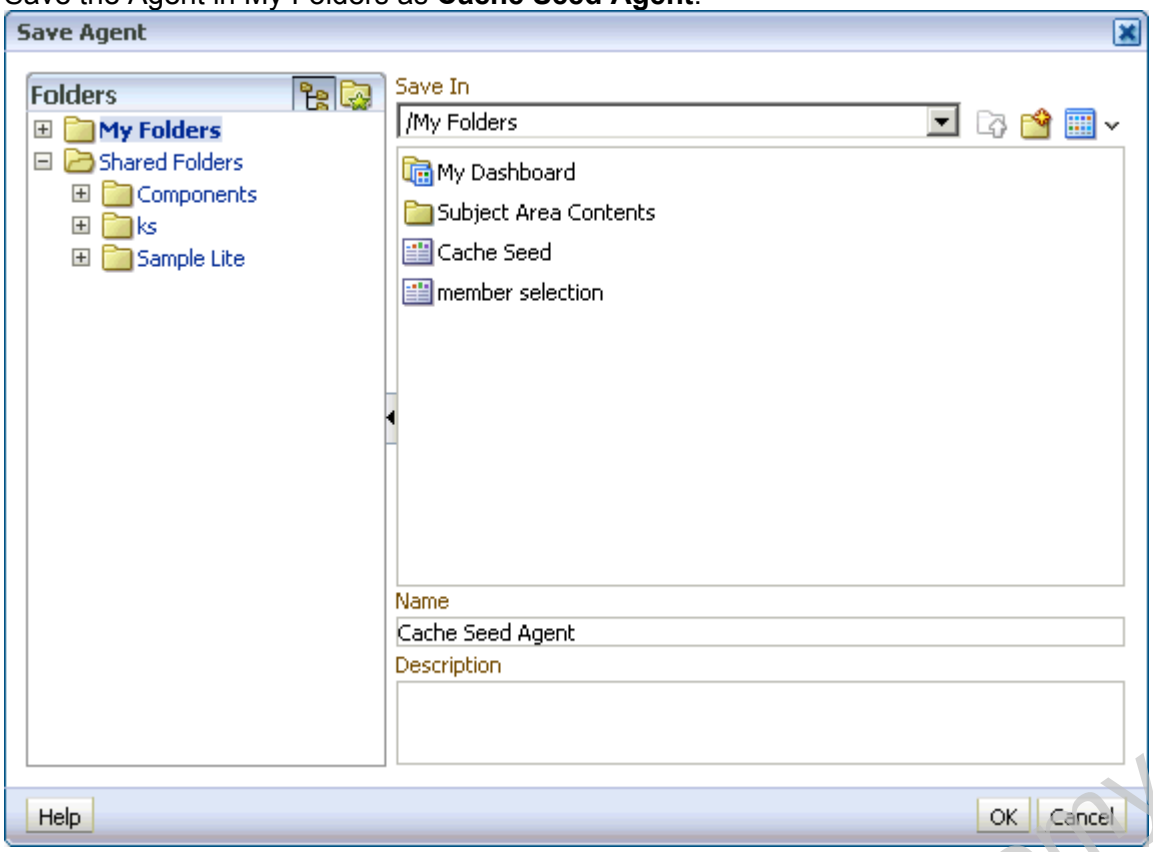
j. Click the **Destinations** tab.

k. Select the **Oracle BI Server Cache** check box.

l. Deselect all other check boxes.



m. Save the Agent in My Folders as **Cache Seed Agent**.



When you save the Agent it is run immediately by the Oracle BI Scheduler.

4. Check your work.
   a. Return to the Administration Tool. The ABC repository should still be open in online mode.
   b. In the Cache Manager, select **Action > Refresh**.
   c. Ensure that there is an entry in the cache for the weblogic user. Right-click the entry and select **Show SQL** to verify that it is the expected query. You have successfully seeded the cache using an Agent. The only difference between cache seeding agents and other agents is that they clear the previous cache automatically and do not appear on the dashboard as Alerts. Notice that cache seeding agents only purge exact match queries, so stale data may still exist. Your caching strategy should always include cache purging, because agent queries do not address ad hoc queries or drills.
   d. Click **Close** to close the SQL window.
   e. Close **Cache Manager**.
   f. Close the repository.
   g. Click **OK**.
   h. Leave the Administration Tool open.

Practices for Lesson 21: Managing Usage Tracking

Lesson 21

Practices for Lesson 21

Lesson Overview

In these practices, you will set up and administer usage tracking.

Practice 21-1 Setting Up Usage Tracking

Goal

To set up and administer usage tracking

Scenario

The Oracle BI Server supports the accumulation of usage tracking statistics that can be used in a variety of ways, such as for 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. ABC wants to monitor the queries generated by users to help identify performance improvement areas. You use the recommended usage tracking approach, which is to track statistics by loading them directly into a database table rather than a log file. You use a provided script to create the necessary database table, modify the `NQSConfig.ini` file to support usage tracking, and test results.

Time

30 minutes

Tasks

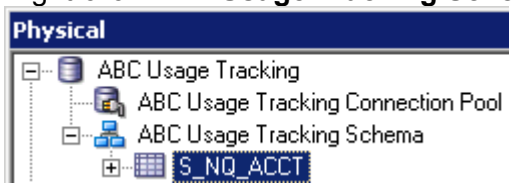
1. Use a provided script to create the `S_NQ_ACCT` usage-tracking table. This table stores the usage tracking data when queries are run against Oracle BI Server.
   a. Double-click the **SQL*Plus icon** on your desktop to open Oracle SQL*Plus.
   b. Log in to the **orcl** host as **supplier2** with password **supplier2**.
   c. At the SQL prompt, enter:
      `start D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\schema\SAACCT.Oracle.sql;`
      Hint: Open Windows Explorer to the location, and copy and paste the location and file name from Windows Explorer to SQL*Plus.
   d. Press **Enter**. Verify that the table and three indexes are created.

e. At the SQL prompt, enter **DESC S_NQ_ACCT;** and press **Enter** to view the table structure.

```
SQL> desc S_NQ_ACCT;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------

 USER_NAME                                          VARCHAR2(128)
 REPOSITORY_NAME                                    VARCHAR2(128)
 SUBJECT_AREA_NAME                                  VARCHAR2(128)
 NODE_ID                                            VARCHAR2(15)
 START_TS                                           DATE
 START_DT                                           DATE
 START_HOUR_MIN                                     CHAR(5)
 END_TS                                             DATE
 END_DT                                             DATE
 END_HOUR_MIN                                       CHAR(5)
 QUERY_TEXT                                         VARCHAR2(1024)
 QUERY_BLOB                                         CLOB
 QUERY_KEY                                          VARCHAR2(128)
 SUCCESS_FLG                                        NUMBER(10)
 ROW_COUNT                                          NUMBER(10)
 TOTAL_TIME_SEC                                     NUMBER(10)
 COMPILE_TIME_SEC                                   NUMBER(10)
 NUM_DB_QUERY                                       NUMBER(10)
 CUM_DB_TIME_SEC                                    NUMBER(10)
 CUM_NUM_DB_ROW                                     NUMBER(10)
 CACHE_IND_FLG                             NOT NULL CHAR(1)
 QUERY_SRC_CD                                       VARCHAR2(30)
 SAW_SRC_PATH                                       VARCHAR2(250)
 SAW_DASHBOARD                                      VARCHAR2(150)
 SAW_DASHBOARD_PG                                   VARCHAR2(150)
 PRESENTATION_NAME                                  VARCHAR2(128)
 ERROR_TEXT                                         VARCHAR2(250)
 IMPERSONATOR_USER_NAME                             VARCHAR2(128)
 NUM_CACHE_INSERTED                                 NUMBER(10)
 NUM_CACHE_HITS                                     NUMBER(10)
```

2. Create a usage tracking database object.
   a. Open the ABC repository in offline mode with password **welcome1**.
   b. Import **S_NQ_ACCT** into the Physical layer.
   c. Right-click inside the Physical layer white space and select **New Database** to open the Database properties dialog box.
   d. Click the **General** tab and name the database **ABC Usage Tracking**.
   e. In the Database drop-down list, select **Oracle 11g**.
   f. Click the **Connection Pools** tab.
   g. Click **Add** to open the Connection Pool dialog box.
   h. Name the connection pool **ABC Usage Tracking Connection Pool**.
   i. Enter **orcl** for data source name.
   j. Enter **supplier2** for username and password.
   k. Click **OK**.
   l. Enter **supplier2** to confirm the password, and click **OK**.
   m. Click **OK** to close the Database properties dialog box.
   n. Right-click the **ABC Usage Tracking** database object and select **New Object > Physical Schema**.
   o. Name the physical schema **ABC Usage Tracking Schema**.
   p. Click **OK**.
3. Copy the S_NQ_ACCT table to the usage-tracking database object.

a. In the Physical layer, expand **orcl > SUPPLIER2**.
b. Right-click **S_NQ_ACCT** and select **Copy**.
c. Right-click **ABC Usage Tracking Schema** and select **Paste**.



4. Create a usage tracking business model.
   a. Right-click in the Business Model and Mapping layer white space and select **New Business Model**.
   b. Name the business model **ABC Usage Tracking** and click **OK**.
   c. Right-click **ABC Usage Tracking** and select **New Object > Logical Table**.
   d. Name the table **Measures** and click **OK**.
   e. Repeat and add three more tables, **Time**, **Topic**, and **User** to the ABC Usage Tracking business model.
   f. Drag the following physical columns from ABC Usage Tracking Schema > S_NQ_ACCT to the **Measures** logical table in the ABC Usage Tracking business model and rename:

Physical Column	Rename
QUERY_TEXT	Query Count
ROW_COUNT	Row Count
TOTAL_TIME_SEC	Total Time Seconds

   g. Apply the following aggregation rules:

Logical Column	Aggregation Rule
Query Count	Count
Row Count	Sum
Total Time Seconds	Sum

   h. Drag the following physical columns from ABC Usage Tracking Schema > S_NQ_ACCT to the **Time** logical table in the ABC Usage Tracking business model and rename:

Physical Column	Rename
START_DT	Start Date
START_HOUR_MIN	Start Hour Minute
END_HOUR_MIN	End Hour Minute

   i. Set **Start Date** as the logical key for the Time logical table.
   j. Drag the following physical columns from ABC Usage Tracking > ABC Usage Tracking Schema > S_NQ_ACCT to the **Topic** logical table in the ABC Usage Tracking business model and rename:

Physical Column	Rename

QUERY_TEXT	Logical SQL
REPOSITORY_NAME	Repository
SUBJECT_AREA_NAME	Subject Area

k. Set **Logical SQL** as the logical key for the **Topic** logical table.

l. Drag the **USER_NAME** physical column from ABC Usage Tracking Schema > S_NQ_ACCT to the **User** logical table in the ABC Usage Tracking business model and rename to **User Name**.

m. Set **User Name** as the logical key for the **User** logical table.

n. Use the Business Model Diagram to create logical joins from **Time**, **Topic**, and **User** to **Measures**.

o. Drag the **ABC Usage Tracking** business model to the Presentation layer to create the Presentation layer objects.

p. Save the repository.

q. Check consistency. You should get a “Business model “ABC Usage Tracking” is consistent. Do you want to mark it as available for queries?” message.

r. Click **Yes**.

s. Click **OK** when you get the consistency check message.

t. Close the repository.

u. Leave the Administration Tool open.

5. Modify the NQSConfig.ini file to support usage tracking.

a. Navigate to **D:\bi\instances\instance1\config\OracleBIServerComponent\coreapplication_obis1**

b. Before making changes to NQSConfig.ini, make a copy of the file and paste it in the same directory.

c. Open **NQSConfig.ini** using Notepad.

d. Scroll to the **Usage Tracking** section.

e. Set **ENABLE = YES;**
This enables usage tracking. Include the semicolon at the end of the line.

f. Scroll past the “Parameters used for writing data to a flat file” section and ensure that DIRECT_INSERT = YES. This parameter determines whether the query statistics are inserted directly into a database table, or are written to a file for subsequent loading. You set this parameter to YES to enable direct insertion. Direct insertion is the recommended method for setting up usage tracking.

g. Scroll to the “Parameters used for inserting data into a table” section and locate the PHYSICAL_TABLE_NAME parameter. To insert query statistic information into a table, you must provide the fully qualified physical table name of 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 (.). This fully qualified physical table name must match a table name in the physical layer of the loaded repository.

h. Notice that two format options are provided for the PHYSICAL_TABLE_NAME parameter. One has three components and one has four components. In this example, the fully qualified path to the usage tracking physical table consists of three components: a database component (ABC Usage Tracking), a schema component (ABC Usage Tracking Schema), and a physical table (S_NQ_ACCT). Therefore, set

the PHYSICAL_TABLE_NAME parameter as follows:
`PHYSICAL_TABLE_NAME = "ABC Usage Tracking"."ABC Usage Tracking Schema"."S_NQ_ACCT";`

i. Set the CONNECTION_POOL variable to
`CONNECTION_POOL = "ABC Usage Tracking"."ABC Usage Tracking Connection Pool";`
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. In order 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. In this example, supplier2 has write access.

j. Leave the default settings for BUFFER_SIZE, BUFFER_TIME_LIMIT_SECONDS, NUM_INSERT_THREADS, and MAX_INSERTS_PER_TRANSACTION. Your file should look like the following screenshot:

```
[USAGE_TRACKING]

ENABLE = YES;

#===============================================================================
# Parameters used for writing data to a flat file (i.e. DIRECT_INSERT = NO).
#
# Note that the directory should be relative to the instance directory.
# In general, we prefer directo insert to flat files.  If you are working in
# a cluster, it is strongly recommended you use direct insert.  If there is
# only one Oracle BI Server instance, then you may use flat file data.
# The directory is then assumed relative to the process instance.  For
# example, "UTData" is resolved to
# "$(ORACLE_INSTANCE)/bifoundation/OracleBIServerComponent/<instance_name>/UTData
STORAGE_DIRECTORY = "<directory path>";
CHECKPOINT_INTERVAL_MINUTES = 5;
FILE_ROLLOVER_INTERVAL_MINUTES = 30;
CODE_PAGE = "ANSI";  # ANSI, UTF8, 1252, etc.
#
#===============================================================================

DIRECT_INSERT = YES;

#===============================================================================
#  Parameters used for inserting data into a table (i.e. DIRECT_INSERT = YES).
#
PHYSICAL_TABLE_NAME = "ABC Usage Tracking"."ABC Usage Tracking Schema"."S_NQ_ACCT";
CONNECTION_POOL = "ABC Usage Tracking"."ABC Usage Tracking Connection Pool";
BUFFER_SIZE = 250 MB;
BUFFER_TIME_LIMIT_SECONDS = 5;
NUM_INSERT_THREADS = 5;
MAX_INSERTS_PER_TRANSACTION = 1;
#
```

k. Save and close the NQSConfig.ini file.

6. Create and run analyses to load the S_NQ_ACCT table.
   a. Use FMW Enterprise Manger to load the ABC repository and restart the Oracle BI components.
   b. Sign in to Oracle BI as **AZIFF** with password **AlanZiff1**.
   c. Create and run the following analysis in the **SupplierSales** subject area:

   d. Click the **Display maximum** button and verify that **181** records are returned.
   e. Sign out and sign back in as **JCRUZ** with password **JoseCruz1**.

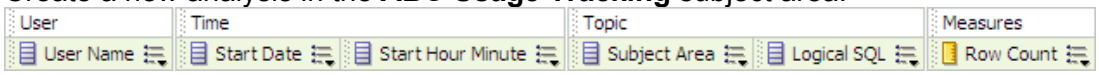
f. Create and run the following analysis in the **SupplierSales** subject area:

g. Click the **Display maximum** button and verify that **400** records are returned.

7. Create and run analyses to check usage tracking.

a. Sign out of Oracle BI and log back in as **weblogic** with password **welcome1**.

b. Create a new analysis in the **ABC Usage Tracking** subject area:



c. Click **Results**. Your results should look similar to the following screenshot.

User Name	Start Date	Start Hour Minute	Subject Area	Logical SQL	Row Count
AZIFF	8/24/2010 12:00:00 AM	18:06	SupplierSales	SELECT s_0, s_1, s_2, s_3 FROM (SELECT 0 s_0, "SupplierSales"."Product"."Specific" s_1, "SupplierSales"."Fact-Sales"."Dollars" s_2, "SupplierSales"."Fact-Sales"."Units Ordered" s_3 FROM "SupplierSales") djm ORDER BY 1, 2 ASC NULLS LAST	181
JCRUZ	8/24/2010 12:00:00 AM	18:07	SupplierSales	SELECT s_0, s_1, s_2 FROM (SELECT 0 s_0, "SupplierSales"."Time"."Date" s_1, "SupplierSales"."Fact-Sales"."Dollars" s_2 FROM "SupplierSales") djm ORDER BY 1, 2 ASC NULLS LAST	400

d. Check the query log and verify that the **OBI Usage Tracking** database and **S_NQ_ACCT** table are accessed in the query.

```
-------------- Sending query to database named ABC Usage Tracking (id:
<<4800>>), connection pool named ABC Usage Tracking Connection Pool:
[[
WITH
SAWITH0 AS (select sum(T1780.ROW_COUNT) as c1,
     T1780.START_DT as c2,
     T1780.START_HOUR_MIN as c3,
     T1780.QUERY_TEXT as c4,
     T1780.SUBJECT_AREA_NAME as c5,
     T1780.USER_NAME as c6
from
     S_NQ_ACCT T1780
group by T1780.QUERY_TEXT, T1780.START_DT, T1780.START_HOUR_MIN,
T1780.SUBJECT_AREA_NAME, T1780.USER_NAME)
select distinct 0 as c1,
     D1.c2 as c2,
     D1.c3 as c3,
     D1.c4 as c4,
     D1.c5 as c5,
     D1.c6 as c6,
     D1.c1 as c7
from
     SAWITH0 D1
order by c6, c2, c3, c5, c4
```

e. Sign out of Oracle BI.

Practices for Lesson 22: Setting Up and Using the Multiuser Development Environment

Lesson 22

Practices for Lesson 22

Lesson Overview

In these practices, you will set up an Oracle BI multiuser development environment to support two developers.

Practice 22-1: Setting Up a Multiuser Development Environment

Goal

To set up a multiuser development environment to support two developers

Scenario

In Oracle Business Intelligence, multiuser development facilitates the development of application metadata in enterprise-scale deployments. Application metadata is stored in a centralized metadata repository (RPD) file. The Administration Tool is used to work with these repositories. In this practice, you set up and the Oracle BI multiuser development environment, including defining projects and setting up the multiuser development directory.

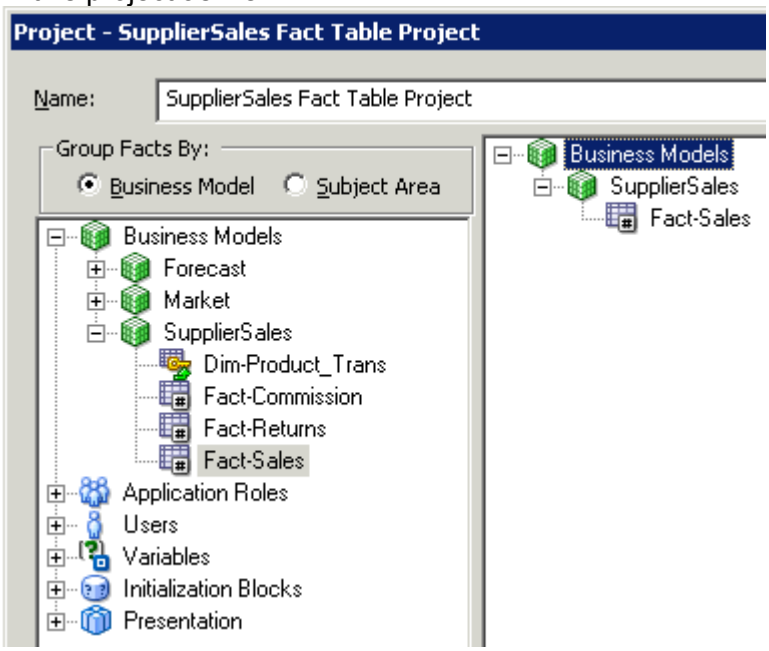
Time

15 minutes

Tasks

1. Verify that only one user at a time can modify a repository before you set up the multiuser development environment.
   a. Return to the Administration Tool and open the **ABC** repository in offline mode.
   b. Select **Start > Programs > Oracle Business Intelligence > Administration** to open a second instance of the Oracle BI Administration Tool.
   c. Select **File > Open > Offline**.
   d. Select the **ABC** repository.
   e. Click **Open**. Notice that you can open the file only as read-only. This is because multiuser development has not been set up, so only one user can edit the repository at a time.
   f. Click **No**.
   g. Select **File > Exit** to close this second instance of the Oracle BI Administration Tool.
2. In this step, you create projects in the master repository. The primary reason to create projects is to support multiuser development. During the development process, you can split up the work (metadata) between different teams within your company by extracting the metadata into projects so that each project group can access a different part of the metadata. Typically you create projects to contain all the subject areas in the repository, broken down in a reasonable manner. In this practice, you create only two projects.
   a. Return to the other instance of the Administration Tool with the **ABC** repository open in offline mode.
   b. Select **Manage > Projects**. The Project Manager window is displayed.
   c. Select **Action > New Project**.
   d. Enter **SupplierSales Fact Table Project** as the name.
   e. Expand **Business Models** in the left pane.
   f. Expand **SupplierSales**.
   g. Select **Fact-Sales.**
   h. Click **Add**. Notice that a **Business Models** folder appears in the right pane.
   i. Expand **Business Models** in the right pane. Notice that the **SupplierSales** business model is automatically included in the project.

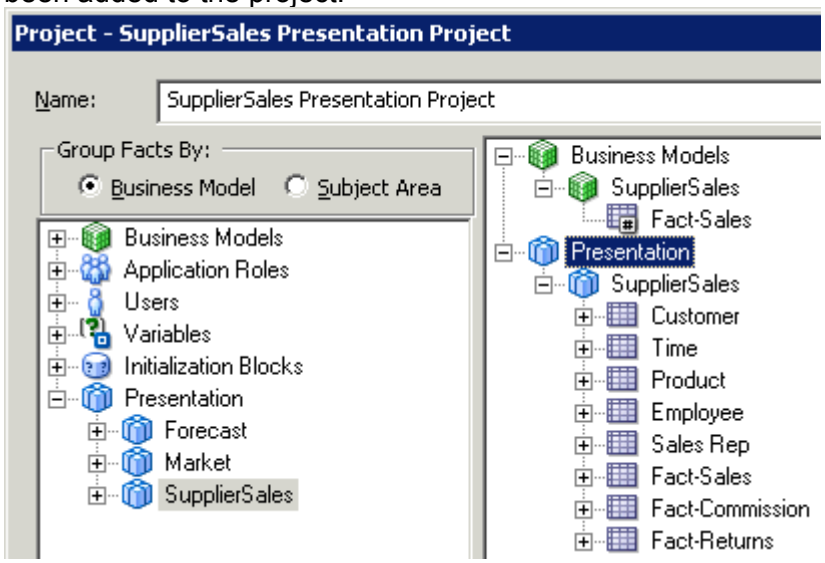
j. Expand **SupplierSales** in the right pane and notice that **Fact-Sales** has been included in this project as well.



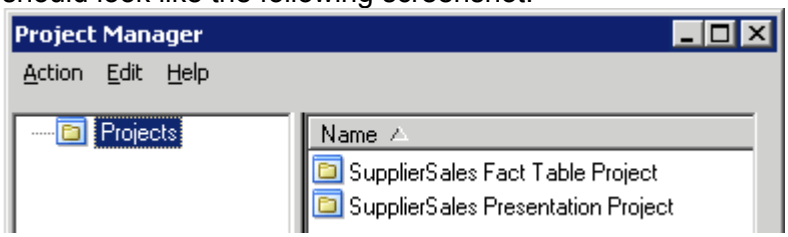
k. Click **OK** to save the project and close the Project window.

3. Create a second project that contains the SupplierSales subject area. Notice that this project also contains the Fact-Sales logical fact table. You are adding a second project to demonstrate that it is possible for projects to overlap, and to demonstrate how to select from multiple projects during project check out.
   a. Select **Action > New Project**.
   b. Enter **SupplierSales Presentation Project** as the name.
   c. Expand **Business Models** in the left pane.
   d. Expand **SupplierSales** in the left pane.
   e. Double-click **Fact-Sales** to add it to the right pane.
   f. Notice that Business Models > SupplierSales > Fact-Sales is added to the right pane.
   g. Expand **Presentation** in the left pane.
   h. Double-click **SupplierSales** to add it to the right pane. Notice that a **Presentation** folder appears in the right pane.
   i. Expand **Presentation** in the right pane. Notice that the **SupplierSales** subject area has been added to the project.

j. Expand **SupplierSales** in the right pane and notice that the presentation tables have been added to the project.



k. Click **OK** to save the project and close the Project window. Your Project Manager should look like the following screenshot:



l. Select **Action > Close** to close the Project Manager.

m. Save your changes to the repository. It is not necessary to perform a global consistency check.

n. Close the repository.

o. Leave the Administration Tool open.

4. Copy your master repository to a shared directory. This allows all the developers on the development team to access the repository.

a. Open **Windows Explorer**.

b. Create a new folder on the D drive, and name it **RPD**.

c. Right-click **RPD** and select **Sharing and Security**.

d. Select **Share this folder**.

e. Click **Permissions**.

f. Allow **Full Control for Everyone**.

g. Click **Apply**.

h. Click **OK** to close the “Permissions for RPD” dialog box.

i. Click **OK** to close the PRD Properties dialog box.

j. Copy **ABC.rpd** from **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository** to **D:\RPD.**

k. In the D:\RPD folder, rename the repository **ABCMaster.rpd**.

5. Set the multiuser directory in the Oracle BI Administration Tool to point to the shared repository that you just created.
   a. In the Oracle BI Administration Tool, select **Tools > Options**.
   b. Click the **Multiuser** tab.
   c. Click **Browse** next to the multiuser development directory field.
   d. Browse to select **D:\RPD**.
   e. In the Full Name field, enter **Administrator**. Before checking out projects, each developer must set up their Administration Tool 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 multiuser development directory. When setting up the pointer, the developer can also complete this 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. Administrator is used as the developer name because it is not possible to reproduce multiple users on multiple machines in this practice.
   f. Click **OK** to close the Options dialog box.
   g. Leave the Administration Tool open. You are ready to perform multiuser development.

Practice 22-2: Using a Multiuser Development Environment

Goal

To check out, modify, and check in projects in a multiuser development environment

Scenario

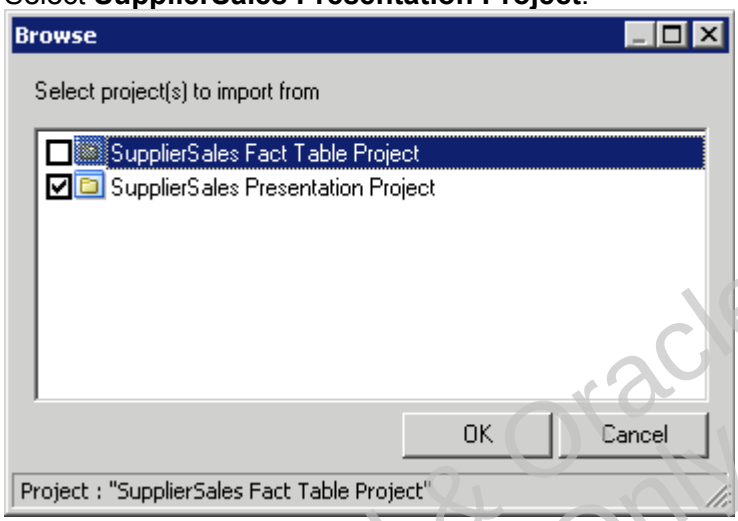
Two developers, JCRUZ and AZIFF, work in the Oracle BI multiuser development environment and modify the same project simultaneously, including checking out and checking in projects, and merging metadata.

Time

25 minutes

Tasks

1. Connect to the shared repository and check out the SupplierSales Presentation Project as developer JCRUZ.
   a. In the Oracle BI Administration Tool, select **File > Multi-User > Checkout**. The **Extract from ABCMaster.rpd** dialog box opens.
   b. Enter **welcome1** as the repository password and click **OK**.
   c. Notice that the two projects that you created are listed in the Browse dialog box.
   d. Select **SupplierSales Presentation Project**.



   e. Click **OK**. The “Create new subset repository” dialog box opens.
   f. In the “File name” field, change the file name to **JCRUZ.rpd**. The file name is arbitrary. Notice that this repository file is being saved to the default repository directory and not the shared repository directory you created earlier. Also, notice that a copy of the shared master repository, ABCMaster.rpd, has been copied to the default repository directory.
   g. Click **Save**. The JCRUZ repository is displayed with the subset of data from the SupplierSales Presentation Project. Notice that it only contains the SupplierSales subject area and business model. The other ABC subject areas and business models are not shown.
   h. Leave the JCRUZ repository open.
2. Review the files that were created or modified by the checkout process.

a. Using Windows Explorer, navigate to D:\RPD. Notice that there are two new files: ABCMaster.000 and ABCMaster.mhl.

b. Navigate to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**. Notice the three new files: **originalJCRUZ.rpd**, **JCRUZ.rpd**, and **JCRUZ.rpd.log**. The JCRUZ.rpd file is a subset of the repository. JCRUZ.rpd contains the SupplierSales Presentation Project metadata. JCRUZ.rpd is the file that you modify. The originalJCRUZ.rpd is the original repository file, which you can use to track your changes or revert to the original. The JCRUZ.rpd.log file is your local log file.

3. As a second developer, AZIFF, check out the same project by using a second instance of the Oracle BI Administration Tool. This demonstrates that multiple users can work with the same repository and the same project simultaneously.

   a. Select **Start > Programs > Oracle Business Intelligence > Administration** to open a new instance of the Administration Tool.

   b. Select **File > Multiuser > Checkout**.

   c. The **Extract from ABCMaster.rpd** dialog box opens.

   d. Enter **welcome1** as the password and click **OK**. The Browse dialog box opens. Notice that both projects are still shown. There is no indication that another developer, JCRUZ**,** has the SupplierSales Presentation Project open. This is an intentional feature of the product: multiple developers can work on a single project, and changes are merged during the check-in process.

   e. Select **SupplierSales Presentation Project**.

   f. Click **OK** to open the **Create new subset repository** window.

   g. Enter **AZIFF.rpd** as the file name.

   h. Click **Save**. The AZIFF repository is displayed with the subset of data from the SupplierSales Presentation Project. Notice that it also contains only the SupplierSales subject area and business model. Two developers are simultaneously working on the SupplierSales Presentation Project.

   i. Navigate to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository** and notice that the same set of AZIFF files has been added: **originalAZIFF.rpd**, **AZIFF.rpd**, and **AZIFF.rpd.log**. Typically, each developer would see his or her own set of files on his or her own development machine. Both sets of files for JCRUZ and AZIFF are located in the same directory because there is only one directory available in this training environment.

4. To learn about the check-in process, modify the project as JCRUZ.

   a. Return to the Administration Tool that has **JCRUZ.rpd** open. The repository name is displayed on the title bar of the application.

   b. In the Presentation layer, expand **SupplierSales.**

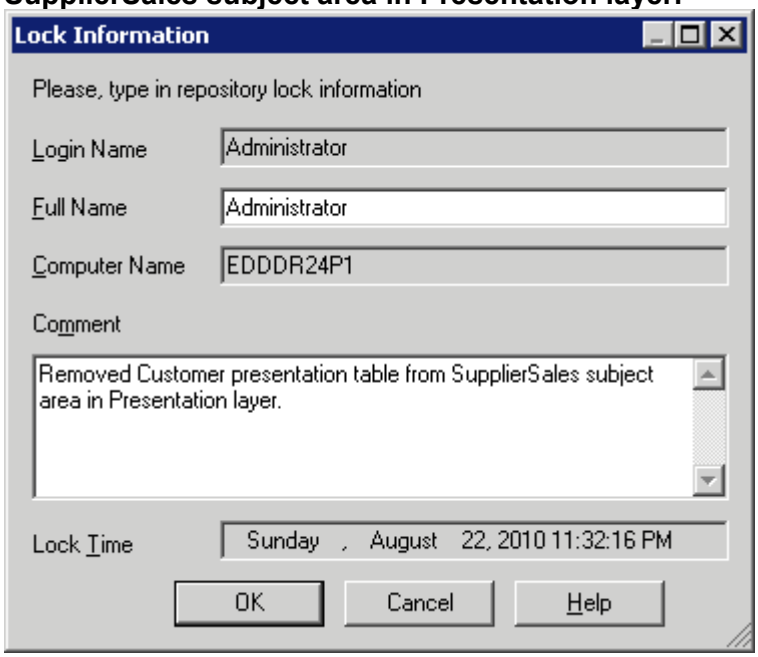
   c. Right-click the **Customer** presentation table and select **Delete**. This is an obvious change that will be easy to track.

   d. Click **Yes** to confirm the deletion.

   e. Save the repository. Do not perform a global consistency check.

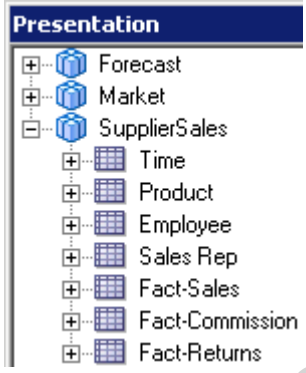
   f. Select **File > Multiuser > Merge Local Changes**. This merges your changes to the shared repository. A Lock Information window is displayed. The shared repository will be locked until check-in is complete.

g. In the comment section, enter **Removed Customer presentation table from SupplierSales subject area in Presentation layer.**



h. Accept the other default values and click **OK**. The Merge Repository Wizard flashes and the ABCMaster repository opens. If there had been any conflicts, the Merge Repository Wizard would have opened and displayed the Define Merge Strategy screen. You would then make merge decisions about whether to include or exclude objects by choosing Current or Modified from the Decision list. You learn more about this process later in this practice.

i. Expand **SupplierSales** and verify that **Customer** is no longer included in the SupplierSales subject area.



5. Inspect the merge log file.

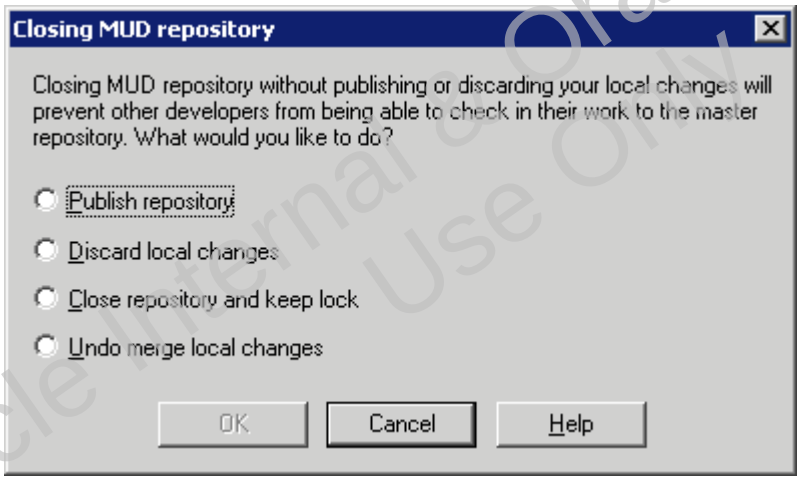
a. Navigate to **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**. Notice that there is a new file named **ABCMaster.merge_log.csv**. This is a comma-separated values file listing the changes made to the repository in the merge.

b. Double-click **ABCMaster.merge_log.csv** to open it in Microsoft Excel and check the changes.

Modified	Application Role	"BIAdministrator"	Default decision
Modified	Application Role	"BIAuthor"	Default decision
Modified	Application Role	"BIConsumer"	Default decision
Deleted	Presentation Level	"SupplierSales".."Customer"."Customer - Sales Rep"."Customer Total"	Default decision
Deleted	Presentation Level	"SupplierSales".."Customer"."Customer - Sales Rep"."Sales Rep"	Default decision
Deleted	Presentation Level	"SupplierSales".."Customer"."Customer - Sales Rep"."Customer Detail"	Default decision
Deleted	Presentation Level	"SupplierSales".."Customer"."Customer - Region"."Customer Total"	Default decision
Deleted	Presentation Level	"SupplierSales".."Customer"."Customer - Region"."Region"	Default decision
Deleted	Presentation Column	"SupplierSales".."Customer"."Address"	Default decision
Deleted	Presentation Level	"SupplierSales".."Customer"."Customer - Region"."District"	Default decision
Deleted	Presentation Column	"SupplierSales".."Customer"."City"	Default decision
Deleted	Presentation Level	"SupplierSales".."Customer"."Customer - Region"."Sales Rep"	Default decision
Deleted	Presentation Column	"SupplierSales".."Customer"."Sales District"	Default decision
Deleted	Presentation Column	"SupplierSales".."Customer"."Customer"	Default decision
Deleted	Presentation Table	"SupplierSales".."Customer"	Default decision
Deleted	Presentation Column	"SupplierSales".."Customer"."Phone"	Default decision
Deleted	Presentation Column	"SupplierSales".."Customer"."Customer Key"	Default decision
Deleted	Presentation Column	"SupplierSales".."Customer"."Region"	Default decision
Deleted	Presentation Column	"SupplierSales".."Customer"."Route Code"	Default decision
Deleted	Presentation Column	"SupplierSales".."Customer"."Sales Rep"	Default decision
Deleted	Presentation Column	"SupplierSales".."Customer"."State"	Default decision
Deleted	Presentation Column	"SupplierSales".."Customer"."Zip Code"	Default decision
Deleted	Presentation Hierarchy	"SupplierSales".."Customer"."Customer - Sales Rep"	Default decision
Deleted	Presentation Hierarchy	"SupplierSales".."Customer"."Customer - Region"	Default decision
Deleted	Presentation Level	"SupplierSales".."Customer"."Customer - Region"."Customer Detail"	Default decision
Children reordered	Subject Area	"SupplierSales"	Tables

c. Close **ABCMaster.merge_log.csv** without saving any changes and close Microsoft Excel.

6. Commit changes to the master repository.
   a. Return to the Administration Tool with **ABCMaster.rpd** open.
   b. Save the modified **ABCMaster** repository. Do not check global consistency.
   c. Select **File > Close** to close the repository. A warning appears that indicates that you are closing a multiuser development (MUD) repository without publishing or discarding your local changes, so the lock on the repository has not been released. At this point, you have the option to publish the repository (copy the local copy of the shared repository to the server), discard the local changes, or close the repository and keep the lock.



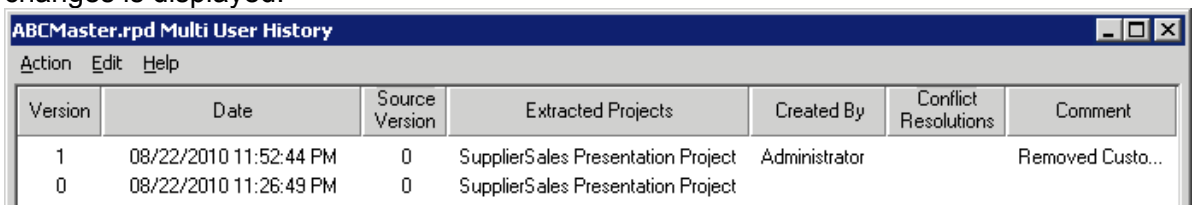
   d. Click **Cancel**.

e. Select **File > Multiuser > Publish to Network**. The local copy of the master repository is merged with the master repository in the shared folder and then closed and deleted.

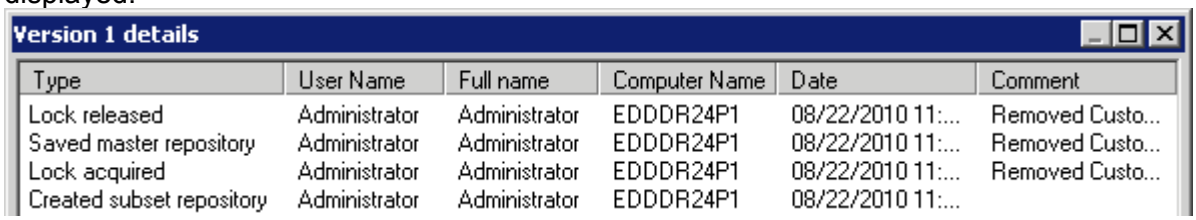
f. Select **File > Multiuser > History**. The **Open Offline Version 1 of ABCMaster.rpd** dialog box opens.

g. Enter **welcome1** as the password and click **OK**.

h. The **Version 1 of ABCMaster** repository is opened and the history of multiuser changes is displayed.



i. Right-click the entry and select **View > Details**. The event details for the project are displayed:



j. Close the **details** window.

k. Select **Action > Close**. The History window and **Version 1 of SharedABC** are closed.

l. In the Administration Tool, select **File > Open > Offline**.

m. Navigate to **D:\RPD** and select **ABCMaster.rpd** to open the shared master repository.

n. Click **Open**.

o. Click **Yes** to acknowledge that it can only be opened as read-only.

p. Enter **welcome1** as the password and click **OK**. The **ABCMaster** repository opens.

q. Expand the **SupplierSales** subject area and ensure that the Customer presentation table no longer appears.

r. Close the repository.

s. Select **File > Exit** to close this instance of the Administration Tool.

7. Return to the instance of the Administration Tool that has AZIFF.rpd open, and verify that AZIFF sees the changes, even though he checked out his project before those changes were applied.

   a. Return to the remaining instance of the Administration Tool with the AZIFF repository open.

   b. Expand the **SupplierSales** subject area and ensure that the **Customer** presentation table still appears. You see the table because AZIFF checked out this project before JCRUZ deleted the Customer table.

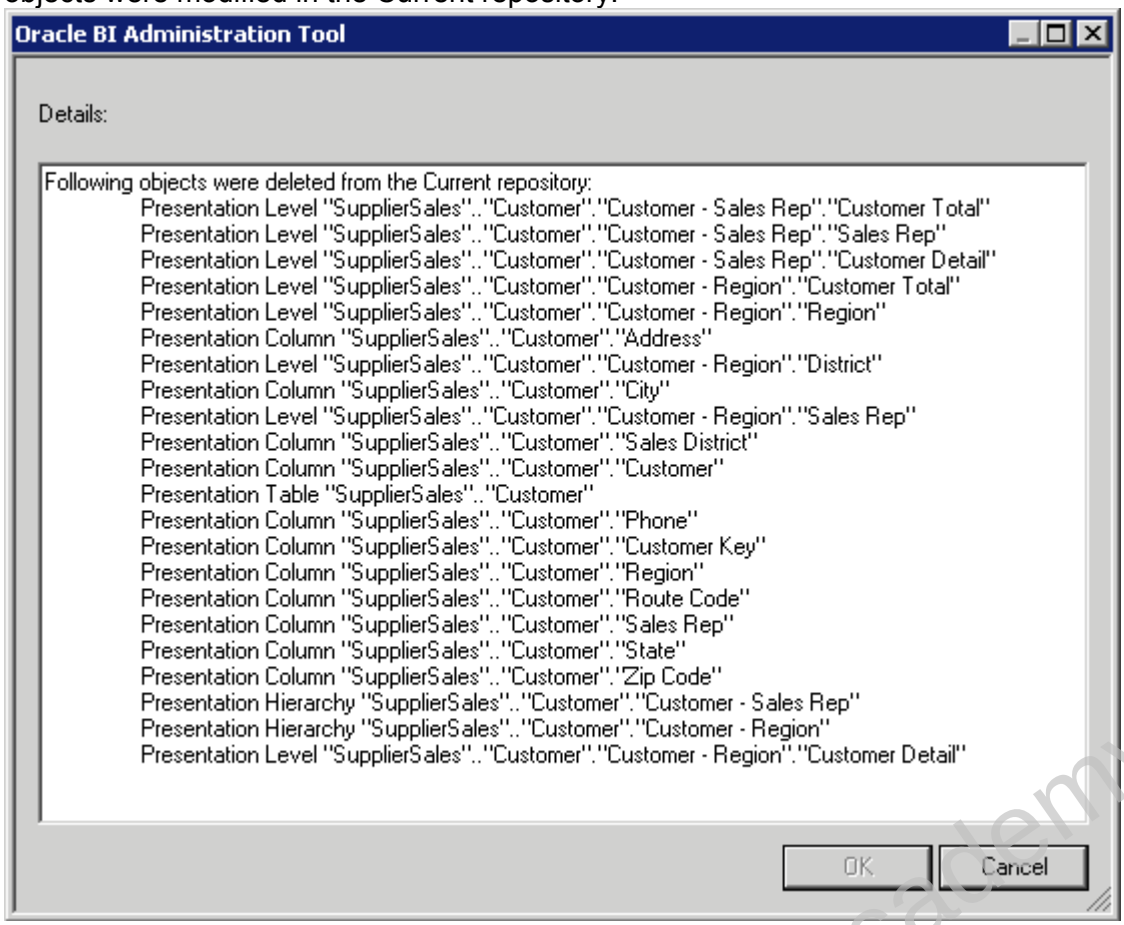
   c. Select **File > Multiuser > Merge Local Changes**.

   d. Click **OK** to accept the default lock information. The Merge Repository Wizard opens.

   e. Notice that the **Customer** presentation table is listed in the Conflicts section.

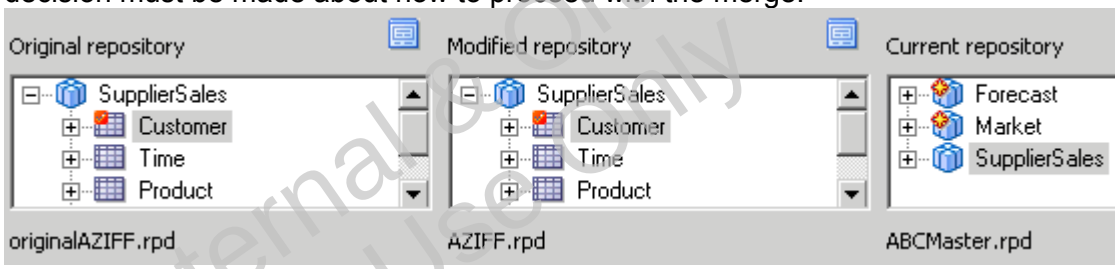
   f. Select the **Customer** listing to highlight it.

g. Click the **ellipsis button** next to the Differences field to view the details of which objects were modified in the Current repository.



h. Click **Cancel** to close the details.

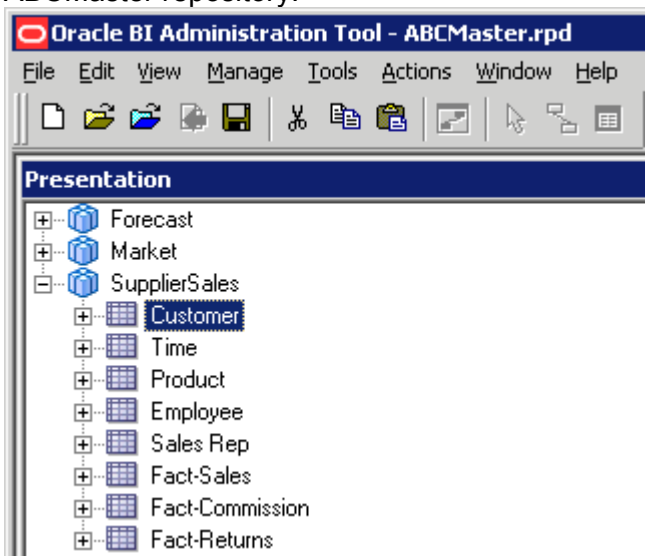
i. Notice that the bottom panes display the original AZIFF repository, the modified AZIFF repository, and the current ABCMaster repository. The differences between the projects are displayed. In this case, the original and modified repositories have a Customer presentation table that is not in the current shared repository. At this point, a decision must be made about how to proceed with the merge.



j. In the Decision column for the highlighted listing, choose **Modified (A)** from the drop-down list (it may be necessary to scroll to the right to see the Decision column). This chooses the modified repository over the current repository, thereby rejecting the changes of the other developer and restoring the repository to the state that it was in before JCRUZ deleted Customer.

k. Click **Finish**.

l. Verify that **Customer** now appears in the SupplierSales subject area of the local ABCMaster repository.



m. Select **File > Multiuser > Publish to Network** to publish the repository.

n. Click **No** when prompted to check global consistency. The local master repository closes and is merged with the shared master repository.

o. Open **ABCMaster.rpd** from D:\RPD in offline mode.

p. Click **Yes** to open as read only.

q. Enter **welcome1** as the password.

r. Expand the **SupplierSales** subject area and confirm that the changes were applied to the master repository and that Customer now appears in the SupplierSales subject area.

s. Close the repository.

t. Leave the Administration Tool open.

Practices for Lesson 23: Configuring Write Back in Analyses

Lesson 23

Practices for Lesson 23

Lesson Overview

In these practices, you will set up and configure write back for Oracle BI dashboards and analyses.

Practice 23-1: Configuring Write Back

Goal

To set up and configure write back for Oracle BI dashboards and analyses

Scenario

Users of a dashboard page or an analysis have the ability to modify the data that they see in a table view or in a pivot table view. This ability is often referred to as "write back." Configuring write back involves steps that span Oracle BI components. This includes setting up write back in the repository, granting write back permissions, and enabling columns for write back in analyses.

Time

20 minutes

Tasks

1. Import a new table into the ABC repository.
   a. Return to the Administration Tool, which should still be open, and open the **ABC** repository in offline mode with repository password **welcome1**.
   b. Import **D1_FORECAST** from the **SUPPLIER2** schema. If you need help, refer to steps from earlier practices.
   c. In the Physical layer, double-click **D1_FORECAST** to open the Physical Table dialog box.
   d. On the General tab, deselect **Cacheable**. This ensures that data written back to the database is not a cached value.
   e. Create the following physical join:
      ```
      "orcl"."".."SUPPLIER2"."D1_PRODUCT_TYPE"."TYPECODE" =
      "orcl"."".."SUPPLIER2"."D1_FORECAST"."TYPECODE"
      ```
2. Enable write back in the connection pool.
   a. In the Physical layer, double-click the **SUPPLIER CP** connection pool to open the Connection Pool dialog box.
   b. Click the **Write Back** tab.
   c. Because this is an Oracle data source, you can accept the defaults.
   d. Explanation of fields:

Property	Description
Prefix	When the Oracle BI Server creates a temporary table, the prefix is the first two characters in the temporary table name. The default value is TT.
Owner	The table owner name used to qualify a temporary table name in a SQL statement (for example, 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.
Database name	The database where the temporary table will be created. This property applies only to IBM OS/390 because IBM OS/390 requires the 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.
Tablespace Name	The tablespace where the temporary table will be created. This property applies to OS/390 only, because OS/390 requires the 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.
Buffer size (KB)	Used for limiting the number of bytes each time data is inserted in a database table. For optimum performance, consider setting this parameter to 128.
Transaction boundary	Controls the batch size for an insert in a database table. For optimum performance, consider setting this parameter to 1000.
Unicode Database Type	Select this option when working with columns of an explicit Unicode data type, such as NCHAR, in a Unicode database. This makes sure that the binding is correct and that data will be inserted correctly. Different database vendors provide different character data types and different levels of Unicode support.

e. Click **Cancel**.

3. Create a new business model in the Business Model and Mapping layer.
   a. Right-click the **white space** in the Business Model and Mapping layer and select **New Business Model**.
   b. Name the business model **Write Back** and click **OK**. Please notice that this is for training purposes only. It is not necessary to create a separate business model to enable write back.
   c. Drag **D1_PRODUCT_TYPE** and **D1_FORECAST** simultaneously to the Write Back business model. Because you drag the tables simultaneously, there is no need to create a logical join. The join relationship is inherited from the Physical layer.
   d. In the D1_FORECAST logical table, set aggregation to **SUM** for **DOLLARS**.
4. Enable write back for logical columns.
   a. In the BMM layer, expand **D1_FORECAST**.
   b. Double-click **DOLLARS**.
   c. On the General tab, select **Writeable**.
   d. Click **OK**.
   e. Drag the **Write Back** business model to the Presentation layer.
5. Set write back permissions in the Presentation layer.
   a. In the Presentation layer, expand **Write Back > D1_FORECAST**.
   b. Double-click **DOLLARS**.
   c. Click **Permissions**.
   d. Select **Show all users/application roles**.
   e. Select **Read/Write** for **BIAdministrator** and **weblogic**.
   f. Click **OK**.
   g. Click **OK** to close the Presentation Column dialog box.
   h. Save the repository.

i. Check consistency. Click **Yes** when prompted with the message: **Business model "Write Back" is consistent. Do you want to mark it as available for queries?** If you do not get this message, fix any errors or warnings before proceeding.

j. Close the repository.

k. Leave the Administration Tool open.

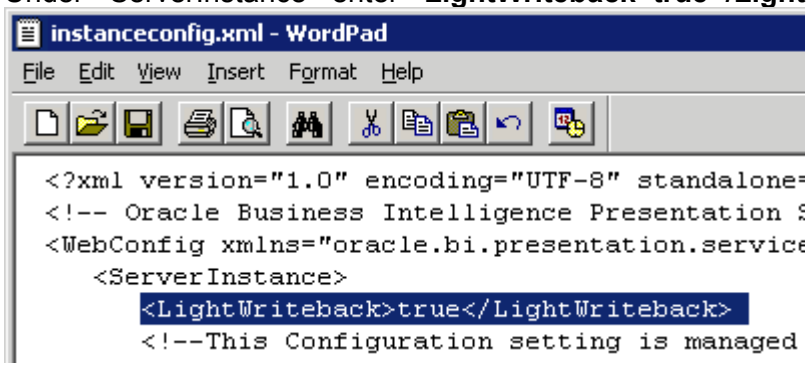
6. Set write back in instanceconfig.xml.

a. Navigate to **D:\bi\instances\instance1\config\OracleBIPresentationServicesComponent\coreapplication_obips1**.

b. Before making any changes to instanceconfig.xml, make a backup copy in the same directory.

c. Right-click **instanceconfig.xml** and select **Open With > WordPad**.

d. Under <ServerInstance> enter **<LightWriteback>true</LightWriteback>**.



e. Save and close **instanceconfig.xml**.

7. Explore the write back template file.

a. Navigate to **D:\PracticeFiles**.

b. Double-click **WriteBackTemplate.xml** to open it. This file has been created and provided as part of this training.

c. The write back template file can have any name of your choosing, because the system reads all XML files in the CustomMessages folder.

```
<?xml version="1.0" encoding="utf-8" ?>
- <WebMessageTables xmlns:sawm="com.siebel.analytics.web/message/v1">
  - <WebMessageTable lang="en-us" system="WriteBack" table="Messages">
    - <WebMessage name="SetForecast">
      - <XML>
        - <writeBack connectionPool="SUPPLIER CP">
            <insert>INSERT INTO D1_FORECAST (TYPECODE,
              MONTHCODE, DOLLARS)VALUES (@1,@3,@4)</insert>
            <update>UPDATE D1_FORECAST SET DOLLARS=@4
              WHERE TYPECODE=@1 AND
              MONTHCODE=@3</update>
          </writeBack>
        </XML>
      </WebMessage>
    </WebMessageTable>
  </WebMessageTables>
```

d. Notice the WebMessage name **SetForecast**. To ensure that write back works correctly, the WebMessage element of the file must include the name of the SQL

template that you will specify when you create the write back table (later in this practice). You can have multiple WebMessage elements in one file, with each element specifying one SQL template.

e. To meet security requirements, you must specify the connection pool (SUPPLIER CP, in this example).

f. You must also include SQL commands to insert and update records. These SQL commands reference the values passed in the write back schema to generate the SQL statements to modify the database table. Values can be referenced either by position (such as @1, @3) or by column ID (@{c0}, @{c2}). Column positions start numbering with 1, whereas column IDs start with c0. If a parameter's data type is not an integer or real number, add single quotation marks around it.

g. You must include both an <insert> and an <update> element in the template. If you do not want to include SQL commands within the elements, then you must insert a blank space between the opening and closing tags. The insert tag is only necessary if there are null values in the write back physical column. This example uses both elements. Oracle BI Server will choose between update and insert, depending on whether the column is null. In this example, the DOLLARS column in the D1_FORECAST table will be modified by the write back function.

h. Close the **WriteBackTemplate** file.

i. Copy the file.

j. Navigate to **D:\bi\Oracle_BI1\bifoundation\web\msgdb**.

k. Create a new folder named **customMessages**.

l. Paste the **WriteBackTemplate.xml** file into **customMessages**.

8. Grant the appropriate write back privileges

a. Use FMW Enterprise Manager to load the ABC repository and restart Oracle BI components.

b. Sign in to Oracle BI as **weblogic** with password **welcome1**.

c. Click **Administration**.

d. Under **Security**, click **Manage Privileges**.

e. Scroll to the bottom and locate the **Write Back to Database** privilege. By default it is denied for AuthenticatedUser.

f. Click **Denied: AuthenticatedUser** to open the "Privilege: Write Back to Database" dialog box.

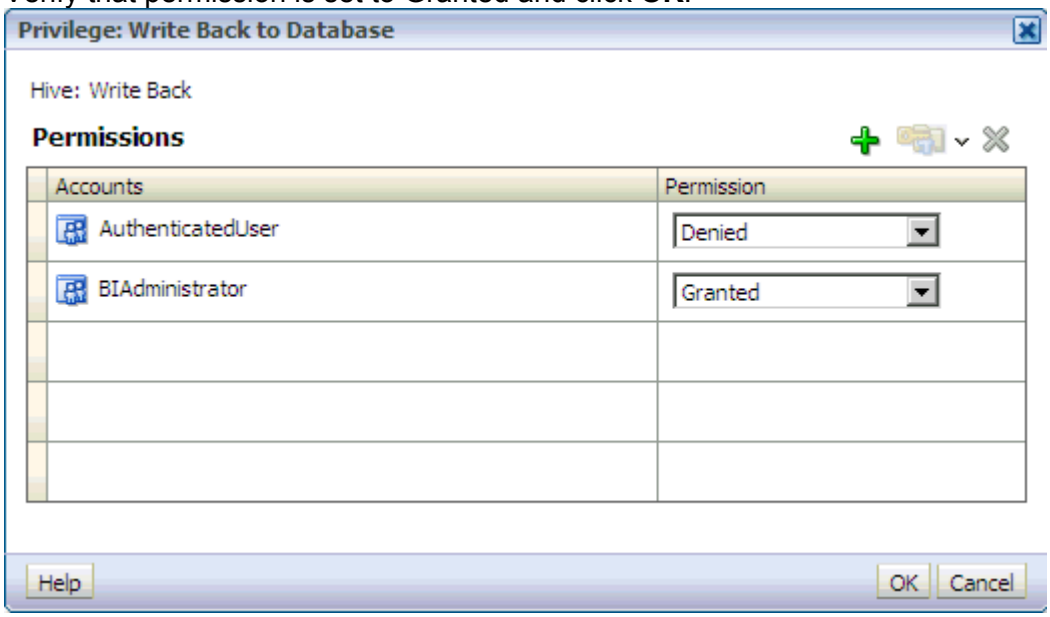
g. Click the **green plus sign** (add users/roles).

h. In the List field select **Application Roles**.

i. Click **Search** to display a list of application roles.

j. Select **BIAdministrator** and move it to Selected Members.

k. Verify that permission is set to Granted and click **OK**.



l. Click **OK** in the "Privilege: Write Back to Database" dialog box.

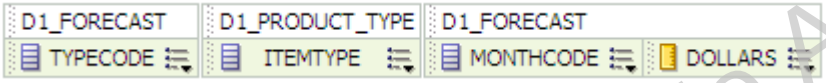
m. Notice that **BIAdministrator** is now listed for the "Write Back to Database" privilege.

9. Create an analysis and enable write back for a column in the analysis.

a. Click **New > Analysis**.

b. Select the **Write Back** subject area.

c. Create the following analysis. Drag columns to make sure that they are in the correct order.



d. Select **Column Properties** for the DOLLARS column.

e. Click the **Write Back** tab.

f. Select **Enable Write Back**.

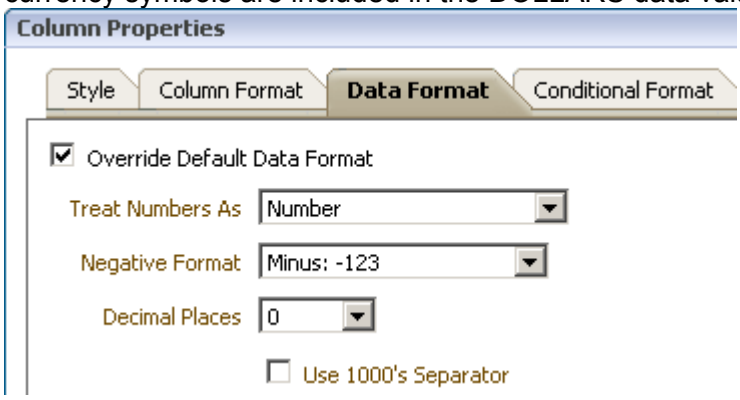
g. Accept the default Text Field Width.

h. Click the **Data Format** tab.

i. Select **Override Default Data Format**.

j. Make sure that **Use 1000's Separator** and **Decimal Places** are unchecked. Data will not correctly write back to the database if commas or other text values such as

currency symbols are included in the DOLLARS data value.



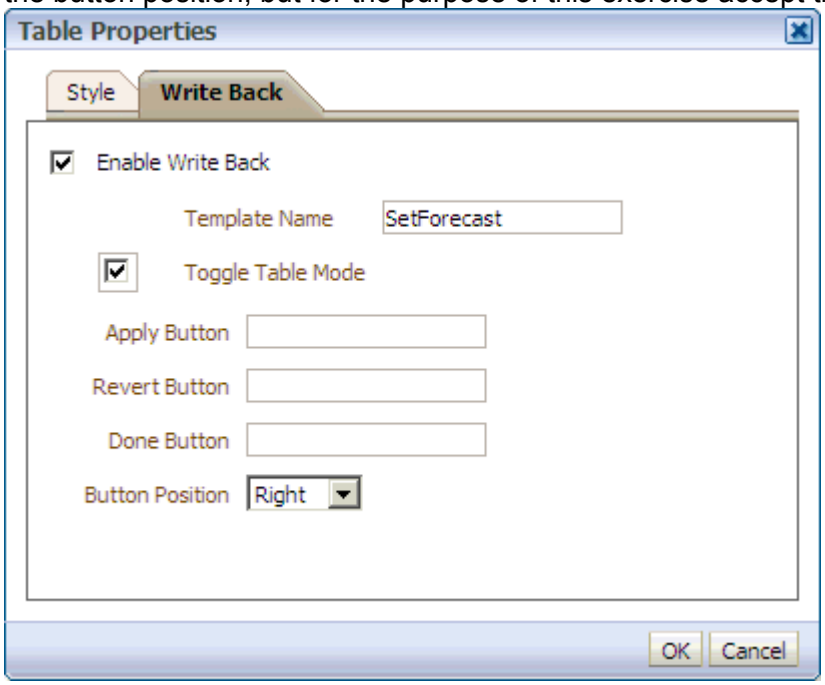
k. Click **OK** to close the Column Properties dialog box.
l. Select **Column Properties** for **MONTHCODE**.
m. Click the **Data Format** tab.
n. Select **Override Default Data Format**.
o. Make sure that **Use 1000's Separator** and **Decimal Places** are unchecked.
p. Click **OK** to close the Column Properties dialog box.

10. Enable write back in the table view.
    a. Click **Results**
    b. Click the **Edit View** button (pencil) for the Table view.
    c. Click the **Table View Properties** button.

TYPECODE	ITEMTYPE	MONTHCODE	DOLLARS
100	Baking	200801	271234
		200802	289782
		200803	316107
		200804	298547
		200805	336265
		200806	321441
		200807	320904

    d. Click the **WriteBack** tab.
    e. Select **Enable Write Back**.
    f. Verify that **Toggle Table Mode** is selected.
    g. In the Template Name field, enter **SetForecast**. Recall that this is Web message name in the template.

h. At this point, you could modify the names of the Apply, Revert, and Done buttons, or the button position, but for the purpose of this exercise accept the defaults.



i. Click **OK** to close the Table Properties dialog box.

j. Click **Done** to exit Compound Layout edit mode.

k. Scroll to the bottom and click the **Update** button to make the DOLLARS field editable.

TYPECODE	ITEMTYPE	MONTHCODE	DOLLARS
100	Baking	200801	271234
		200802	289782
		200803	316107
		200804	298547
		200805	336265

l. Modify DOLLARS data in a few rows.

m. Click the **Apply** button and verify that the records are updated. If you click the Revert button before clicking Apply, the data will return to the original values.

n. Click **Done** to return to the original table view where the DOLLARS column is not editable.

o. Sign out of Oracle BI.

Practices for Lesson 24: Performing a Patch Merge

Lesson 24

Practices for Lesson 24

Lesson Overview

In these practices, you will perform a repository patch merge in a development-to-production scenario.

Practice 24-1: Performing a Patch Merge

Goal

To perform a repository patch merge in a development-to-production scenario.

Scenario

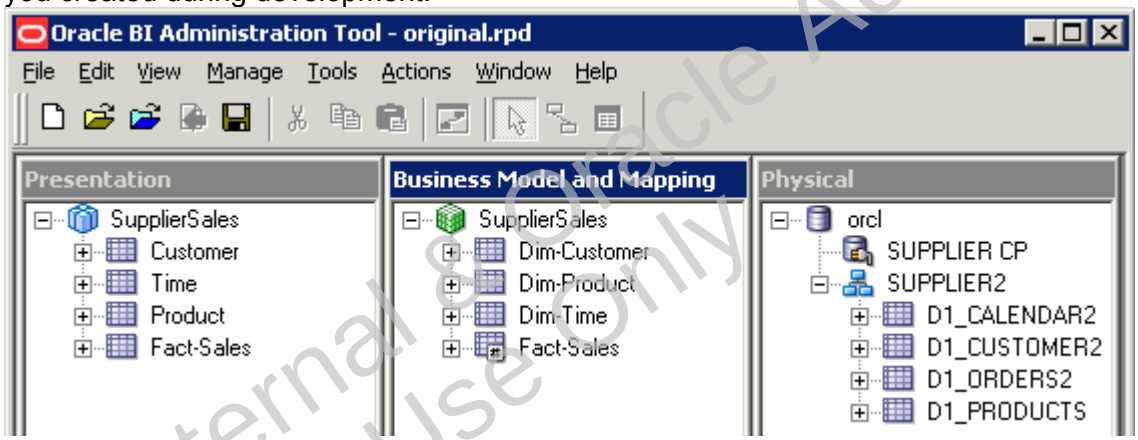
In a patch merge, you create a patch that contains the differences between the current repository file and the original repository file, and then you apply the patch file to the modified repository file. In a development-to-production scenario, you have an original parent file, a current file that contains the latest development changes, and a modified file that is the deployed copy of the original file. To generate a patch, you open the current file and select the original file, and then create the patch. To apply the patch, you open the modified file and select the original file, and then apply the patch.

Time

15 minutes

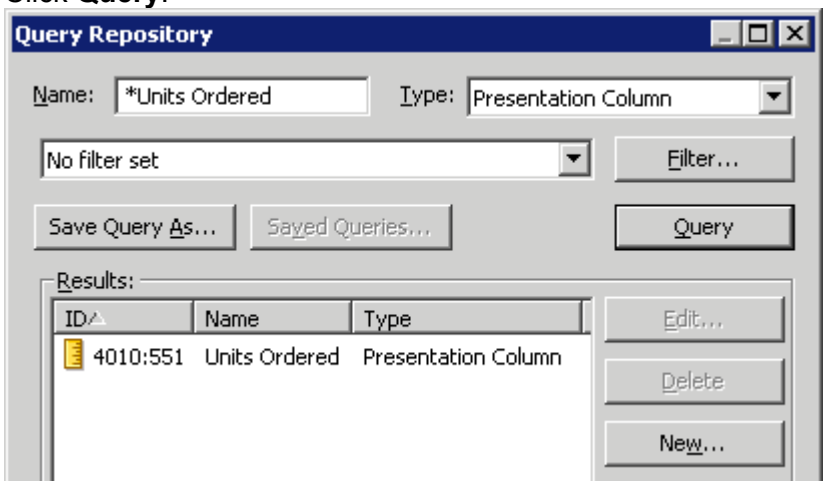
Task

1. Before creating and applying a patch merge, explore the repositories that you will use in this practice.
   a. Navigate to **D:\PracticeFiles**.
   b. Copy the **original**, **modified**, and **current** repository files and paste them into **D:\bi\instances\instance1\bifoundation\OracleBIServerComponent\coreapplication_obis1\repository**.
   Overwrite the existing original.rpd when prompted.
   c. In the Administration Tool, open **original.rpd** in offline mode with password **welcome1**. For the purpose of this training, assume this is the original repository that you created during development.



   d. Update row count to check connectivity.
   e. Select **Tools > Query Repository** to open the Query Repository utility.
   f. In the Name field, enter **Units Ordered** after the asterisk.
   g. Change the Type to **Presentation Column**.

h. Click **Query**.

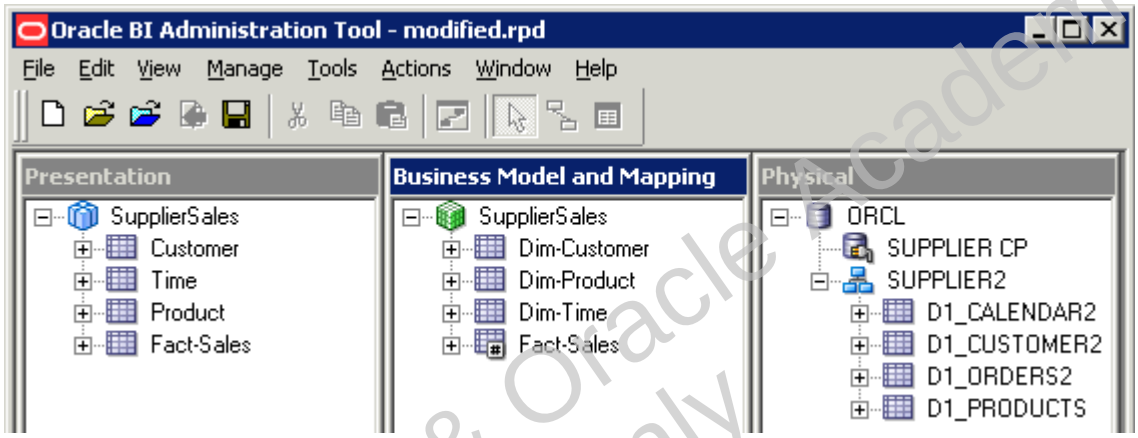


i. Notice the **ID** for the object (4010:551, in this example). The Administration Tool tracks the history of each repository object by using the upgrade ID of the object. The upgrade ID is a unique identifier for each object.

j. Close the **Query Repository** utility.

k. Close **original.rpd** without making any changes.

l. Open **modified.rpd** in offline mode with **welcome1** as the password. For the purpose of this training, assume this is the production repository. Before applying the patch in the steps that follow, the contents of the **original** and **modified** repositories are the same.



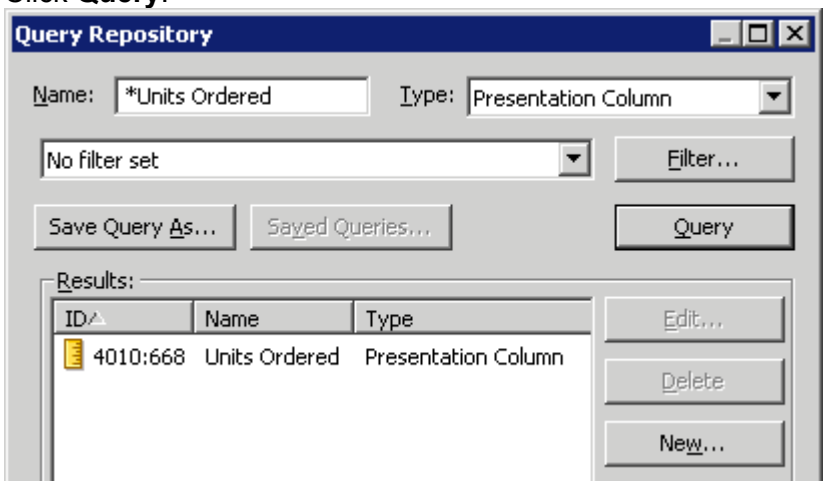
m. Update row counts to verify connectivity.

n. Select **Tools > Query Repository** to open the Query Repository utility.

o. In the Name field, enter **Units Ordered** after the asterisk.

p. Change the Type to **Presentation Column**.

q. Click **Query**.



r. Notice the **ID** for the object (4010:668, in this example). Although the Units Ordered presentation column object exists in both repositories, the object ID is different. This is because Units Ordered was deleted and then recreated in the modified repository. The significance of this becomes apparent in the steps that follow.

s. Close the **Query Repository** utility.

t. Leave the **modified** repository open.

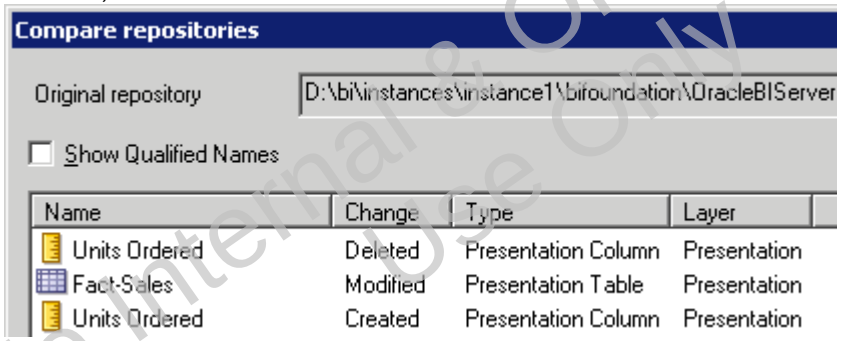
2. Use the Equalize Objects option in the Compare Repositories dialog box to compare and equalize the original and modified repositories. Objects may need to be equalized because the Administration Tool tracks the history of each repository object by using the upgrade ID of the object. Sometimes, the upgrade ID can change because of user actions or during merge. When this occurs, and a subsequent comparison is done, the Administration Tool treats the new upgrade ID as a new object, and the missing original upgrade ID as a deleted object.

a. Select **File > Compare**.

b. Select the **original** repository and click **Open**.

c. Enter **welcome1** as the password and click **OK** to open the Compare Repositories dialog box.

d. Notice that the tool compares the two repositories and marks objects as modified, deleted, or created.



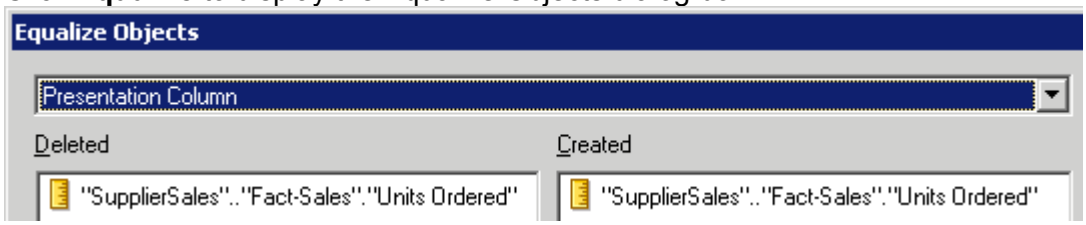
– **Created** means the object was created in the current repository and does not exist in the original repository.

– **Deleted** means the object exists in the original repository but has been deleted from the current repository.

- **Modified** means the object exists in the original repository but has been modified in the current repository.

e. In this example, notice that the **Units Ordered** presentation column is marked as both deleted and created. This is because the object was deleted and then re-created in the modified repository. The Fact-Sales presentation table is marked as modified because the Units Ordered presentation column is in this table.

f. Click **Equalize** to display the Equalize Objects dialog box.

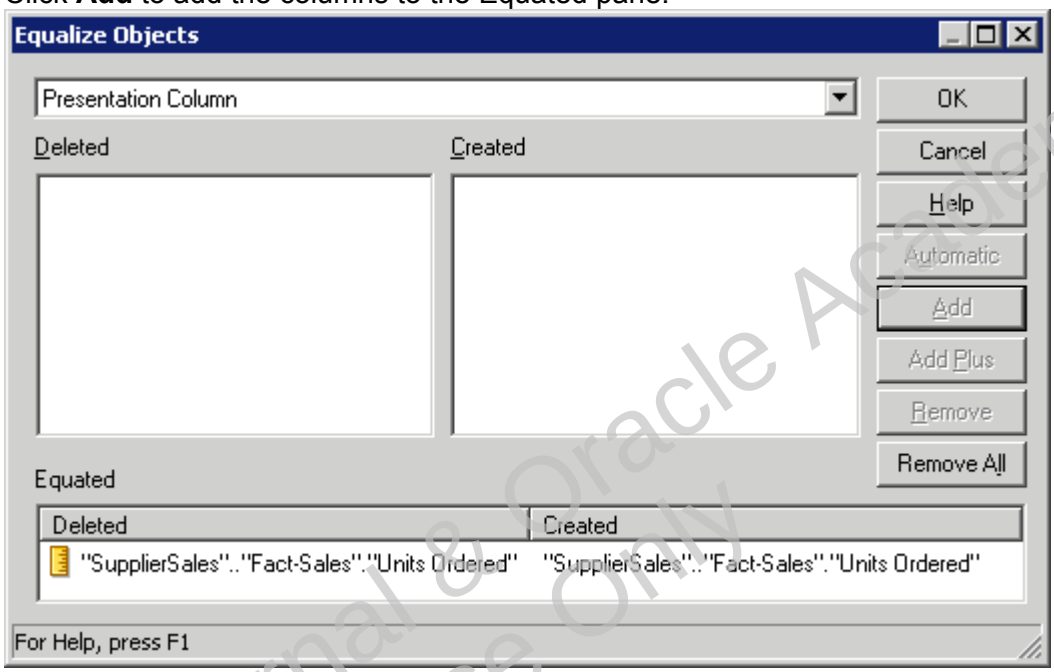


g. Notice that only **Presentation Column** is available in the drop down list. This is because, in this example, only presentation columns have been deleted. If more object types had been deleted, they would be displayed in this list (Logical Table, Logical Column, and so forth).

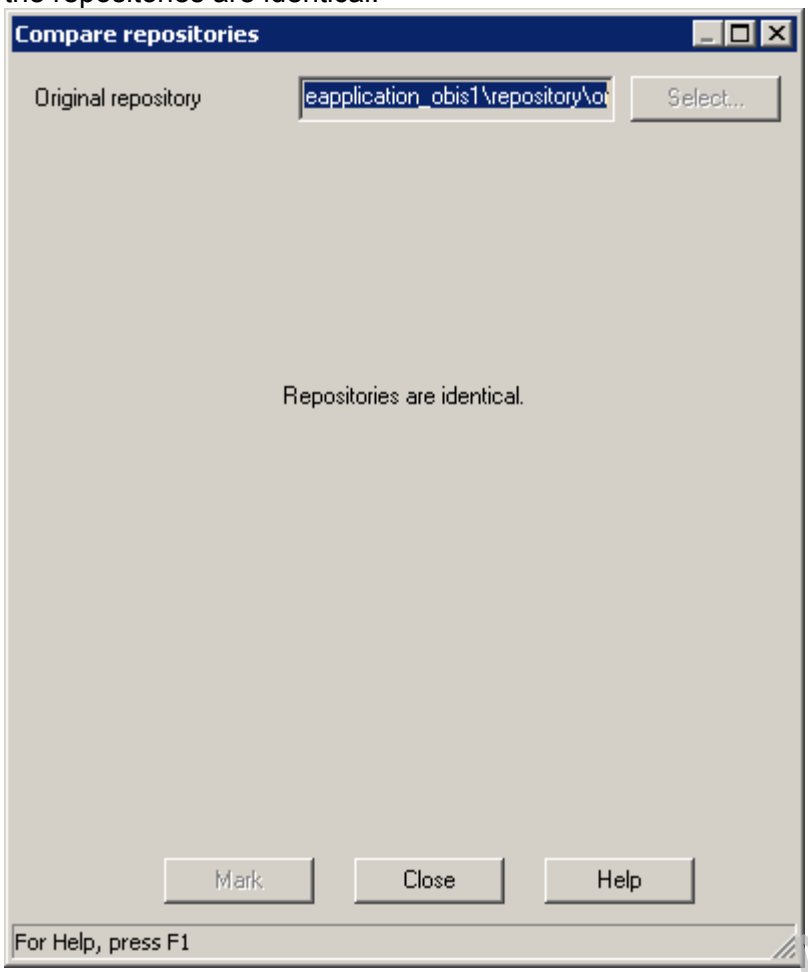
h. In the Deleted pane, select **"SupplierSales"."Fact-Sales"."Units Ordered"**.

i. In the Created pane, select **"SupplierSales"."Fact-Sales"."Units Ordered"**.

j. Click **Add** to add the columns to the Equated pane.



k. Click **OK** to close the Equalize Objects dialog box. You should receive a message that the repositories are identical.



l. Click **Close** to close the Compare repositories dialog box.

m. Save the repository.

n. Check consistency. You should receive the following message: “Consistency check didn’t find and errors, warnings, or best practice violations.”

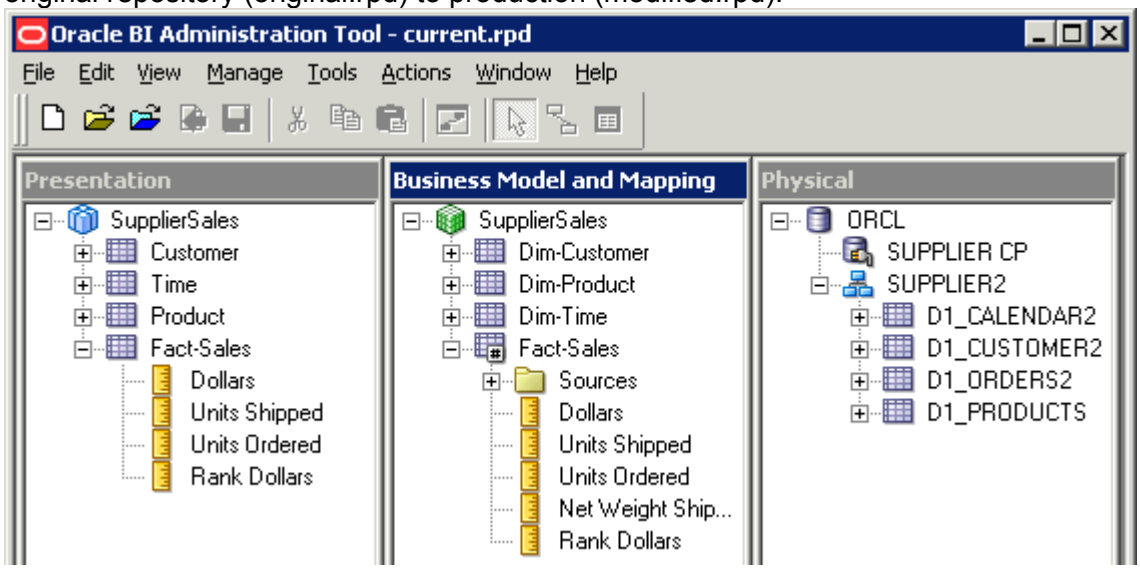
o. Click **OK**.

p. Close the **modified** repository.

3. Generate a repository patch.

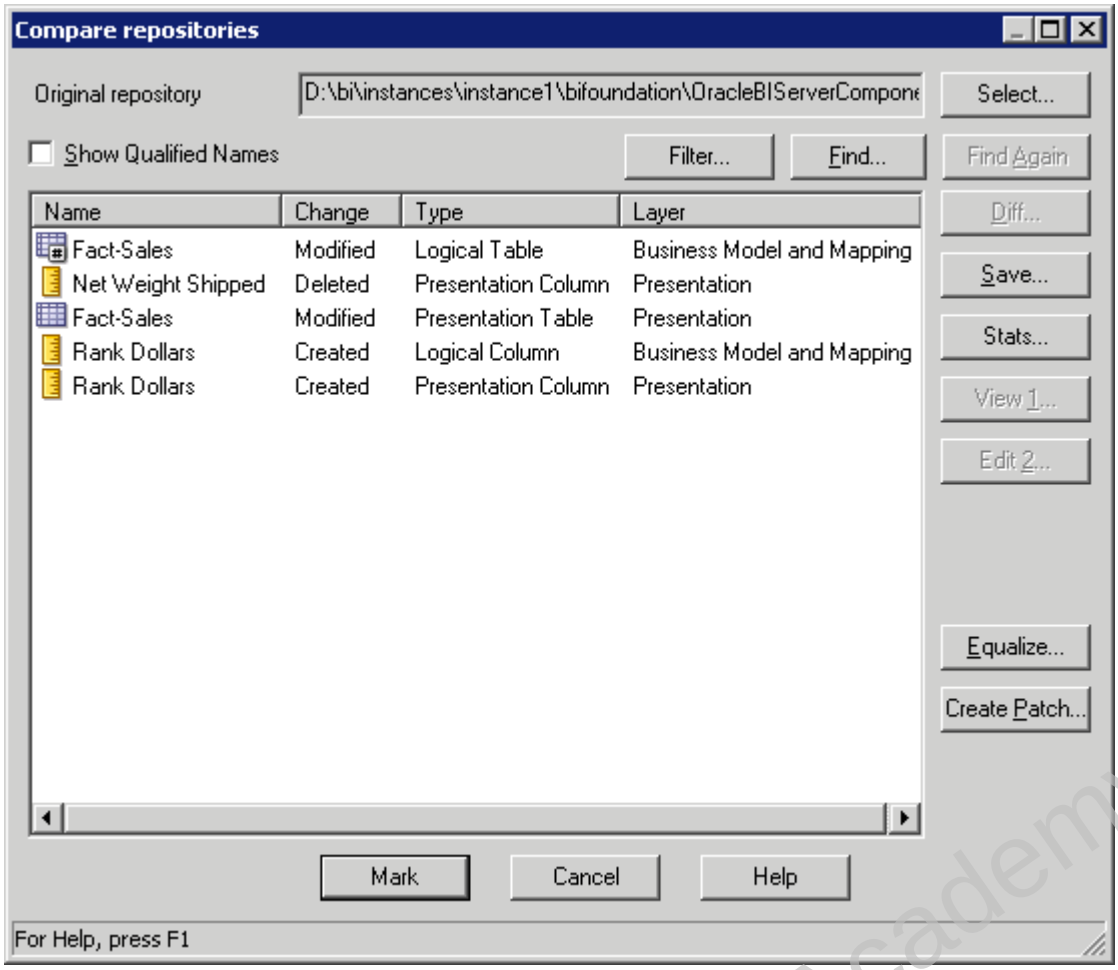
a. In the Administration Tool, open **current.rpd** in offline mode with **welcome1** as the password. This is the repository that contains the changes that you want to put in the patch. Assume that this is a repository that you have updated after rolling out the

original repository (original.rpd) to production (modified.rpd).



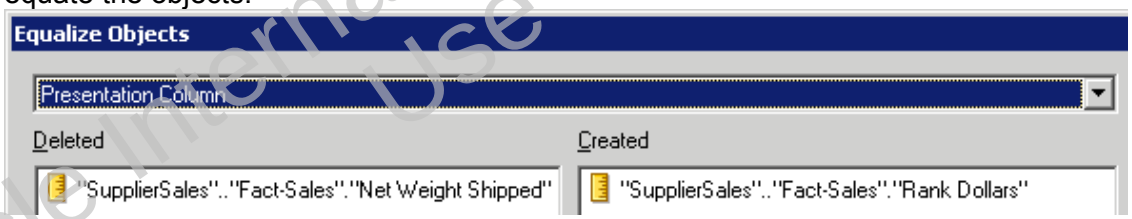
b. Update row counts to check connectivity.
c. Select **File > Compare**.
d. Select the **original** repository and click **Open**.
e. Enter **welcome1** as the password and click **OK**. The Compare Repositories dialog box appears, compares the two repositories, and marks objects as modified, deleted, or

created.



f. In this example, the changes are:
   - A **Rank Dollars** logical column has been created and added to the Fact-Sales logical table in the Business Model and Mapping layer.
   - A **Rank Dollars** presentation column has been created and added to the Fact-Sales presentation table in the Presentation layer.
   - The **Net Weight Shipped** presentation column has been deleted from the Fact-Sales presentation table in the Presentation layer.

g. Click **Equalize** to open the Equalize Objects dialog box. Notice that the objects listed in the Deleted and Created columns are not the same, and, therefore, there is no need to equate the objects.



h. Click **Cancel** to close the Equalize Objects dialog box.

i. In the Compare repositories dialog box, review the changes between the repositories, and then click **Create Patch**.

j. In the Create Patch dialog box, enter **my_patch** as the name for the XML patch file and click **Save**. By default, the file is saved to the repository directory. The Compare Repositories dialog box closes automatically.

k. Close the **current** repository.

4. Apply the repository patch.

a. In the Administration Tool, open the **modified** Oracle BI repository in offline mode with **welcome1** as the password. This is the repository on which you want to apply the patch.

b. Select **File > Merge** to open the Merge Repository Wizard.

c. For **Merge Type**, select **Patch Repository Merge**.

d. Click **Select** next to **Original Master Repository**. Browse to select the **original** repository, and then click **Open**. Notice that the original repository cannot be the same as the modified (currently open) repository.

e. Enter **welcome1** as the password.

f. Click **Select** next to **Patch File**. Browse to select the **my_patch.xml** patch file, and then click **Open**.

g. Click **Select** next to **Save Merged Repository as.**

h. Enter **modified_patched** as the file name and click **Save**.

i. Click **Next**. The Merge Repository Wizard opens.

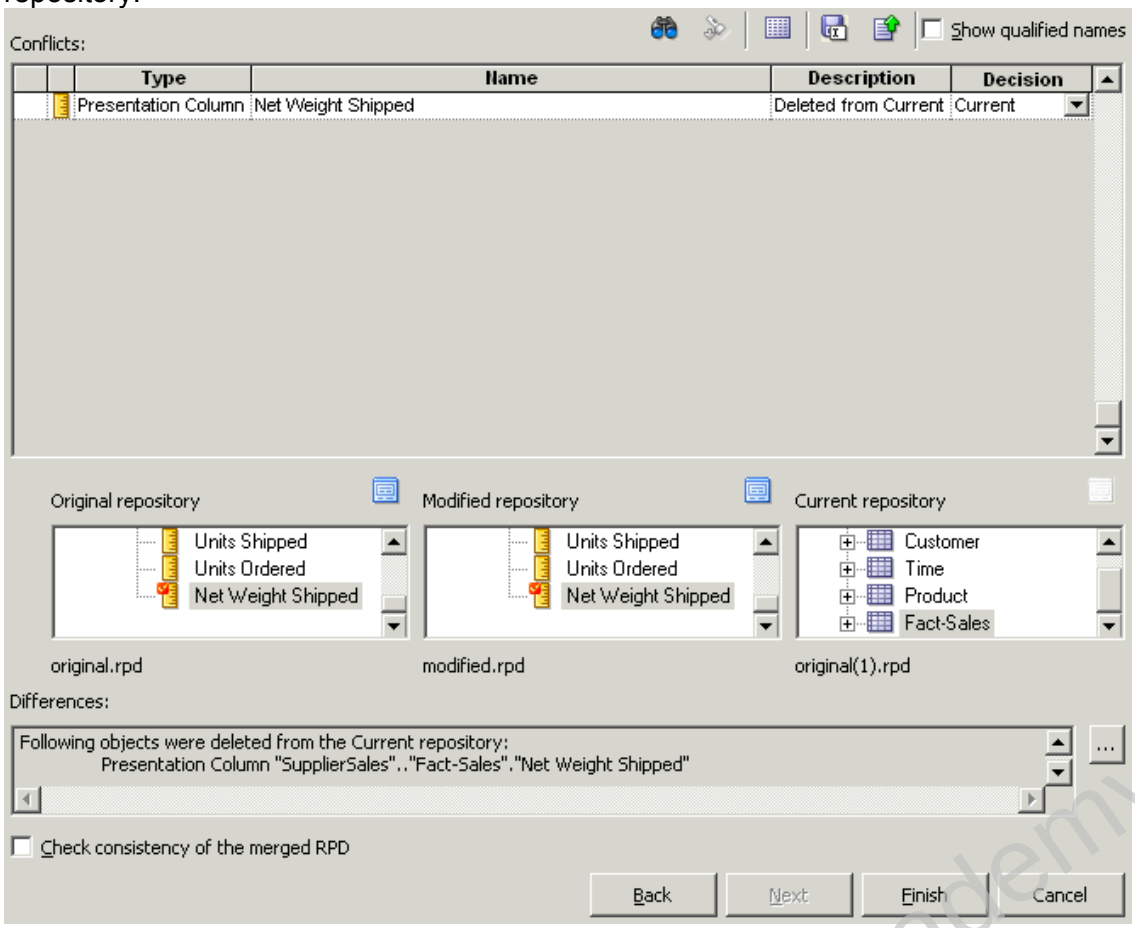
j. Under Conflicts, notice that the description of the conflict for **Net Weight Shipped** is **Deleted from Current**.

k. Click the **Decision** field for New Weight Shipped. Notice there are two choices:

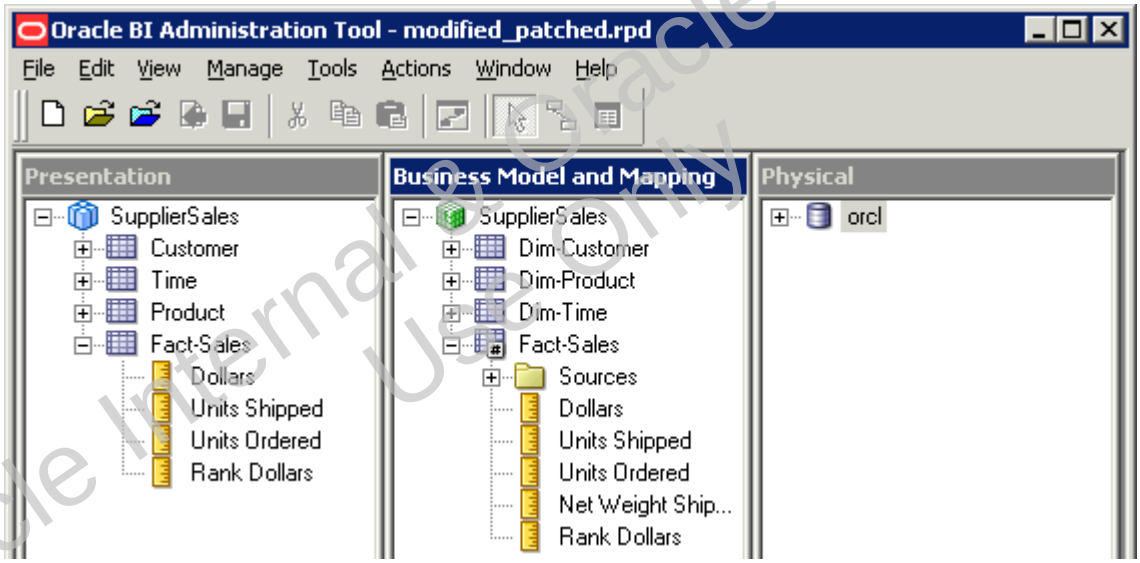
- Choosing **Current** would keep the repository as it is without adding the object to the merged repository.
- Choosing **Modified (A)** would add the object into the merged repository.

l. Select **Current**. In this example, **Net Weight Shipped** was deliberately deleted from the **current** repository and, therefore, you do not want it added to the merged

repository.



m. Click **Finish**. The **modified_patched** repository opens in offline mode. Verify that the expected changes have been applied. A Rank Dollars logical column and presentation column have been created, and the Net Weight Shipped presentation column has been deleted.



n. Close the repository. Leave the Administration Tool open.

Practices for Lesson 25: Configuring Logical Columns for Multicurrency Support

Lesson 25

Practices for Lesson 25

Lesson Overview

In these practices, you will define user-preferred currency options using a static mapping.

Practice 25-1: Defining User-Preferred Currency Options

Goal

To define user-preferred currency options by using a static mapping

Scenario

Oracle Business Intelligence users can select the currency in which they prefer to view currency columns in analyses and dashboards. They select the currency in the Currency box in the My Account dialog box, Preferences tab. You define the currency options that are to be displayed in the Currency box in the userpref_currencies.xml file.

Time

20 minutes

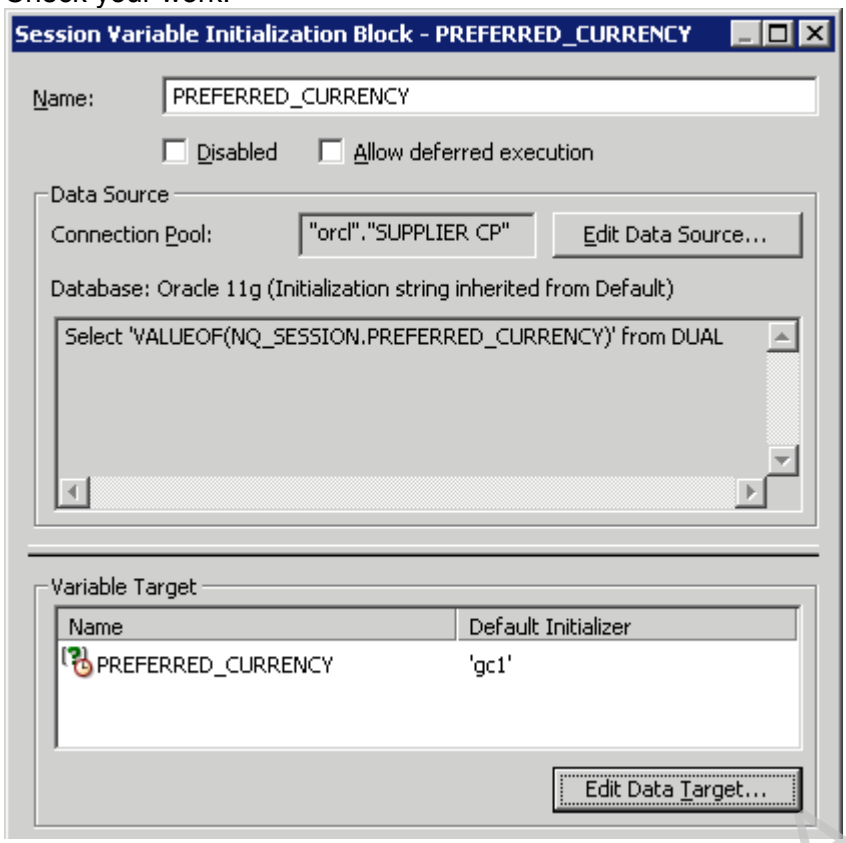
Task

1. Modify the userpref_currencies.xml file.
   a. Navigate to **D:\bi\instances\instance1\config\OracleBIPresentationServicesComponent\coreapplication_obips1**
   b. Before modifying **userpref_currencies.xml**, make a copy of it in the same directory.
   c. Right-click **userpref_currencies.xml** and select **Open With > WordPad**.
   d. Remove the first <`UserCurrencyPreferences`> tag and the <!-- comment marker.
   e. Remove the **-->** comment marker and the last </`UserCurrencyPreferences`> tag. Your file should look similar to the following screenshot:

```
<UserCurrencyPreferences currencyTagMappingType="static">
  <UserCurrencyPreference sessionVarValue="gc1" displayText="Global Currency 1" currencyTag="int:USD" />
  <UserCurrencyPreference sessionVarValue="gc2" displayText="Global Currency 2" currencyTag="int:euro-1" />
  <UserCurrencyPreference sessionVarValue="gc3" displayText="Global Currency 3" currencyTag="loc:ja-JP" />
  <UserCurrencyPreference sessionVarValue="orgc" displayText="Org Currency" currencyTag="loc:en-BZ" />
  <UserCurrencyPreference sessionVarValue="lcl" displayTag="int:DEM" currencyTag="int:DEM" />
</UserCurrencyPreferences>
```

   f. Save and close the **userpref_currencies.xml** file.
2. Create a session variable named PREFERRED_CURRENCY.
   a. Open the **ABC** repository in offline mode with password **welcome1**.
   b. Select **Manage > Variables**.
   c. Select **Action > New > Session > Initialization Block**.
   d. Name the initialization block **PREFERRED_CURRENCY**.
   e. Click **Edit Data Source**.
   f. Select **Default initialization string**.
   g. Enter **Select 'VALUEOF(NQ_SESSION.PREFERRED_CURRENCY)' from Dual** in the field.
   h. Click **Browse**.
   i. Select the **SUPPLIER CP** connection pool.
   j. Click **OK** to close the Session Variable Initialization Block Data Source dialog box.
   k. Click **Edit Data Target**.
   l. Click **New**.
   m. Enter **PREFERRED_CURRENCY** in the Name field.

n. Select **Enable any user to set the value**.
o. Enter **'gc1'** as the default initializer.
p. Click **OK** to close the Session Variable dialog box.
q. Click **OK** to close the Session Variable Initialization Block Variable Target dialog box. Check your work.



r. Click **OK** to close the Session Variable Initialization Block dialog box.
s. Close the Variable Manger.

3. Create logical columns with currency conversions. Please notice that, in this set of steps, you are hard-coding random currency conversion values. This is for the purpose of this training only. Typically, currency conversion would be calculated as part of your data warehouse extract, transform, and load (ETL) process.
   a. In the Business Model and Mapping layer, expand **SupplierSales > Fact-Sales**.
   b. Right-click Fact-Sales and select **New Object > Logical Column**.
   c. On the General tab, name the column **EuroCurrency**.
   d. Click the **Column source** tab.
   e. Select **Derived from existing columns using an expression**.
   f. Click the **Edit Expression** button to open Expression Builder.
   g. Create the following formula:
      ```
      "SupplierSales"."Fact-Sales"."Dollars" * .75
      ```
   h. Click **OK** to close Expression Builder.
   i. Click **OK** to close the Logical Column dialog box.
   j. Repeat the steps to create three more logical columns with calculated currency conversions:

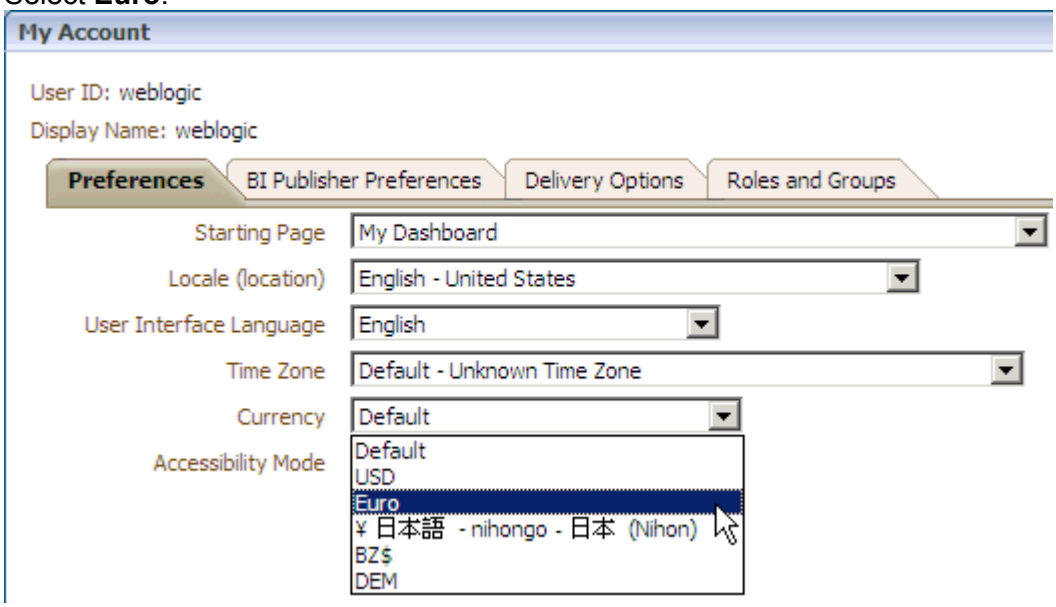
Logical Column	Formula
JapCurrency	`"SupplierSales"."Fact-Sales"."Dollars" * .92`
DEMCurrency	`"SupplierSales"."Fact-Sales"."Dollars" * 1.46`
USDCurrency	`"SupplierSales"."Fact-Sales"."Dollars"`

4. Create a logical column to use the appropriate conversion factor using the PREFERRED_CURRENCY session variable.
   a. Right-click **Fact-Sales** and select **New Object > Logical Column**.
   b. On the General tab, name the column **Preferred Currency**.
   c. Click the **Column source** tab.
   d. Select **Derived from existing columns using an expression**.
   e. Click the **Edit Expression** button to open Expression Builder.
   f. Create the following formula. You can copy the formula from **PREFERRED_CURRENCY.txt** located in **D:\PracticeFiles**.

```
IndexCol( CASE  VALUEOF(NQ_SESSION."PREFERRED_CURRENCY")
WHEN 'gc1' THEN 0
WHEN 'gc2' THEN 1
WHEN 'gc3' THEN 2
WHEN 'lc1' THEN 3
ELSE 4 END,
"SupplierSales"."Fact-Sales"."USDCurrency",
"SupplierSales"."Fact-Sales"."EuroCurrency",
"SupplierSales"."Fact-Sales"."JapCurrency",
"SupplierSales"."Fact-Sales"."DEMCurrency",
"SupplierSales"."Fact-Sales"."USDCurrency")
```

   g. Close **Expression Builder**.
   h. Close the **Logical Column** dialog box.
   i. Drag **Preferred Currency** to the **Fact-Sales** presentation table in the **SupplierSales** subject area.
   j. Save the repository.
   k. Check consistency. Fix any errors or warnings before proceeding.
   l. Close the repository.
5. Check your work.
   a. Use FMW Enterprise Manager to upload the ABC repository and restart Oracle BI components.
   b. Sign in to Oracle BI as **weblogic** with password **welcome1**.
   c. Select **weblogic > My Account > Preferences**.
   d. Confirm that you can see the Currency drop-down list with the preferred currencies listed.

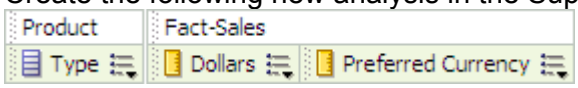
e. Select **Euro**.



f. Click **OK** to close the My Account page.

6. Format the currency column.

a. Create the following new analysis in the SupplierSales subject area:

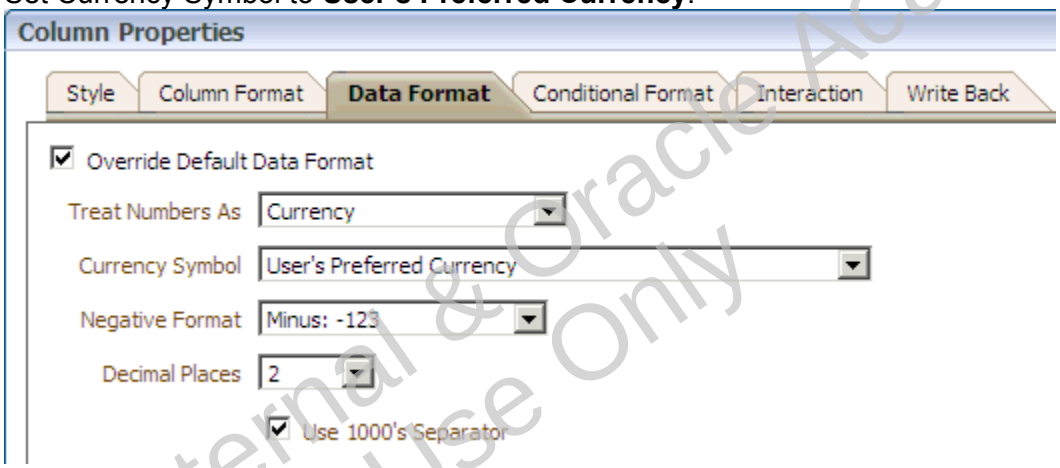


b. For the Preferred Currency column, select **Column Properties > Data Format**.

c. Select **Override Default Data Format**.

d. Set Treat Numbers As to **Currency**.

e. Set Currency Symbol to **User's Preferred Currency**.



f. Click **OK** to close Column Properties.

g. Click **Results**.

h. Confirm that the Preferred Currency column displays the preferred currency selected on the My Account page (Euro, in this example).

Type	Dollars	Preferred Currency
Baking	$4,925,521	€ 3,694,140.60
Beef	$4,916,016	€ 3,687,012.15
Beverage	$4,398,107	€ 3,298,580.61
Bread	$1,578,743	€ 1,184,057.59
Cereal	$1,309,071	€ 981,803.56
Cheese	$7,140,616	€ 5,355,462.07
Condiments	$9,105,121	€ 6,828,841.04
Dessert	$2,208,427	€ 1,656,320.10
Entre	$1,807,794	€ 1,355,845.58
Frozen	$521	€ 390.96
Grains	$39,419	€ 29,564.32
Lamb	$71,553	€ 53,664.63
Non-food	$4,547,002	€ 3,410,251.18
Pasta	$147,409	€ 110,556.99

Optional Challenge Practice: Using Partitions and Fragments

Lesson 26

Optional/Challenge Practice: Using Partitions and Fragments

Lesson Overview

In these practices, you will create a business model for fragmented inventory data.

Optional/Challenge Practice: Modeling Fragmented Inventory Data

Goal

To create a business model for fragmented inventory data

Scenario

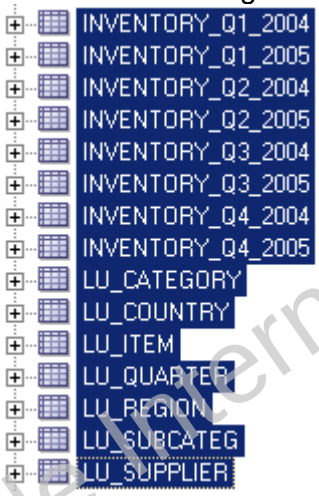
ABC's inventory data is fragmented. That is, it is split into multiple tables. Data values determine which table contains which data. In this example, eight quarters of inventory data are stored in eight separate tables. You must create a business model for this fragmented inventory data. In addition to modeling fragmented data, this practice gives you the opportunity to apply some of the modeling techniques that you have learned in this course up to this point.

Time

50 minutes

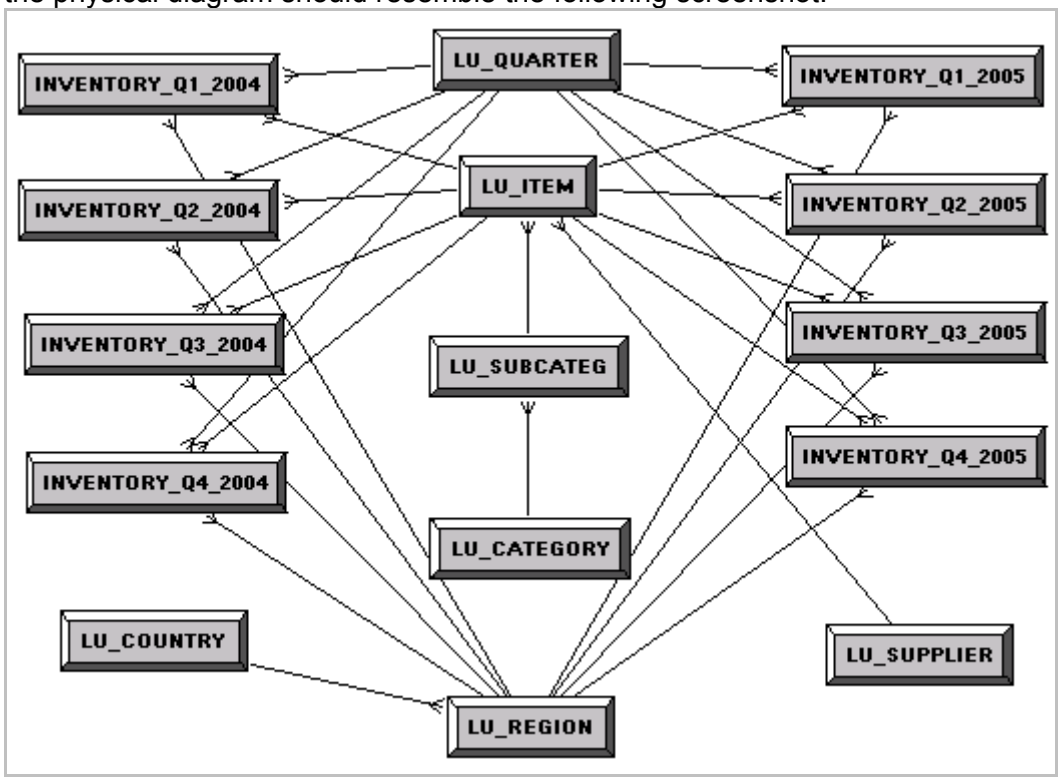
Tasks

1. Create a new repository and import tables.
   a. In the Administration Tool, select **File > New Repository**.
   b. Name the repository **Inventory**.
   c. Select **Yes** for **Import Metadata**.
   d. Enter **welcome1** as the password and to confirm the password.
   e. Click **Next**.
   f. Select the **OCI 10g/11g** connection type.
   g. Enter **orcl** as the Data Source Name.
   h. Enter **supplier2** as the username and password, and click **Next**.
   i. In the Select Metadata Types window, ensure that **Tables**, **Keys**, and **Foreign Keys** are selected and click **Next**.
   j. Select the following tables for import in the SUPPLIER2 schema:



   k. Click **Finish**.
   l. Verify that the new tables are added to the Physical layer.
   m. Select all the newly imported tables and click the **Physical Diagram icon** on the toolbar. Rearrange the tables and use the zoom feature as necessary to make the tables visible in the diagram. Assuming that you imported both keys and foreign keys,

the physical diagram should resemble the following screenshot:



If you did not import the foreign keys, you must manually configure the joins in the physical diagram or delete the tables and reimport with foreign keys selected.

n. Close the Physical Diagram.

o. Update row counts to check connectivity.

p. If desired, right-click the new Inventory tables and select **View Data** to get an idea of what data is contained in these tables.

2. Create a new business model named **Inventory**.

3. Create the fact table.

   a. Create a new logical table named **Fact-Inventory**. This fact table will have four facts: Beginning Inventory Dollars, Ending Inventory Dollars, Beginning Inventory Quantity, and Ending Inventory Quantity. These facts will map to physical columns in the eight Inventory physical tables.

   b. In the Physical layer, expand **INVENTORY_Q1_2004**.

   c. Drag the following four columns simultaneously to the **Fact-Inventory** logical table:
   BOH_DLL
   BOH_QTY
   EOH_DLL
   EOH_QTY
   Notice that **INVENTORY_Q1_2004** is added as a logical table source to the Fact-Inventory table.

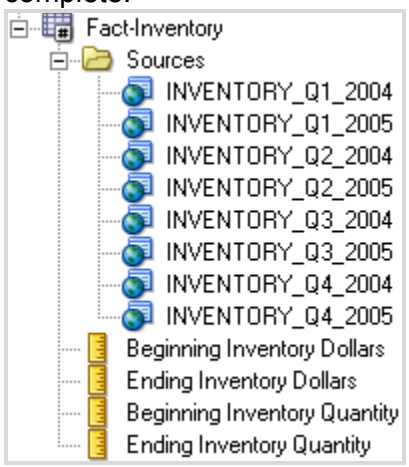
   d. Expand the **INVENTORY_Q1_2005** table and drag the same four columns simultaneously to the Fact-Inventory logical table. Because the column names are identical, you can drag all four columns simultaneously. Notice that **INVENTORY_Q1_2005** is added as a source to the Fact-Inventory table.

e. Repeat this process for the remaining six INVENTORY tables. All these sources are at the same level of aggregation and they all contain mappings to the same logical facts. The only difference in the tables is the time period of the data.

f. Rename the logical fact columns:

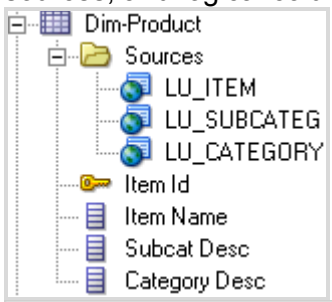
Old Name	New Name
BOH_DLL	Beginning Inventory Dollars
EOH_DLL	Ending Inventory Dollars
BOH_QTY	Beginning Inventory Quantity
EOH_QTY	Ending Inventory Quantity

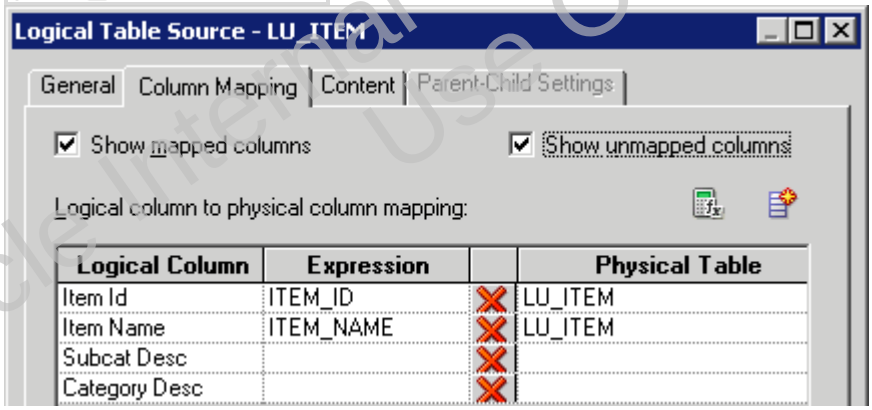
g. The Fact-Inventory table should look like the following screenshot when the process is complete:

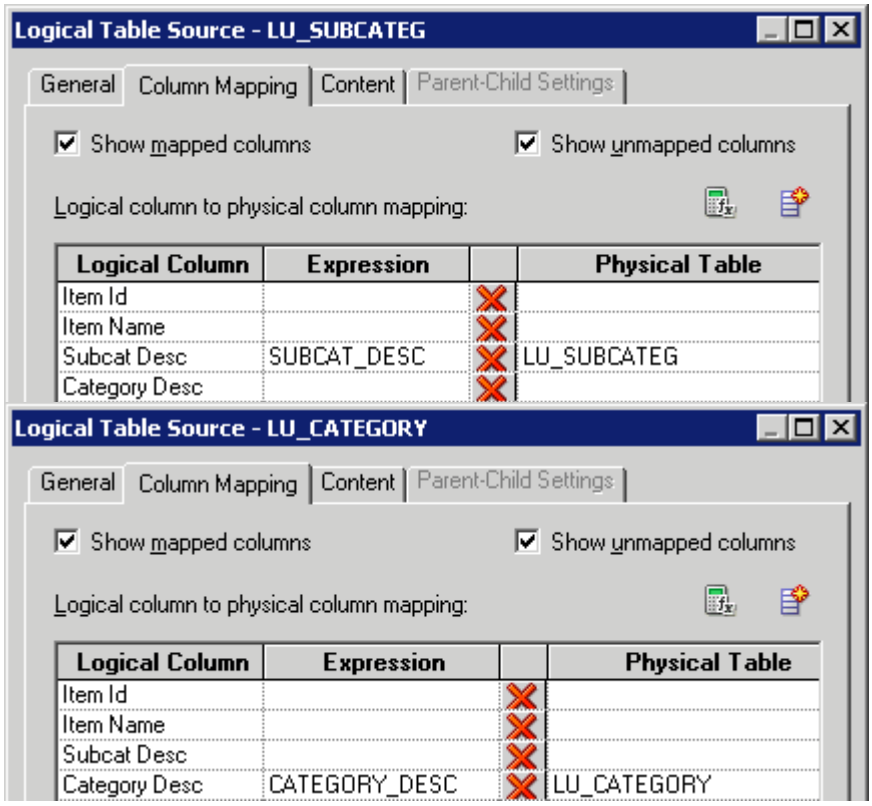


4. Create the dimension tables for the Inventory business model.

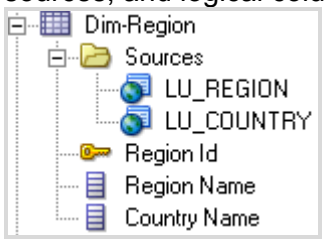
a. Create a **Dim-Product** logical table with the following logical columns, logical table sources, and logical column mappings:

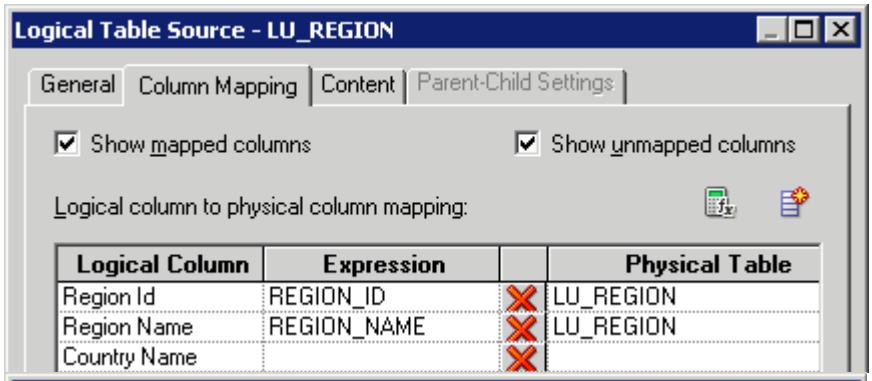


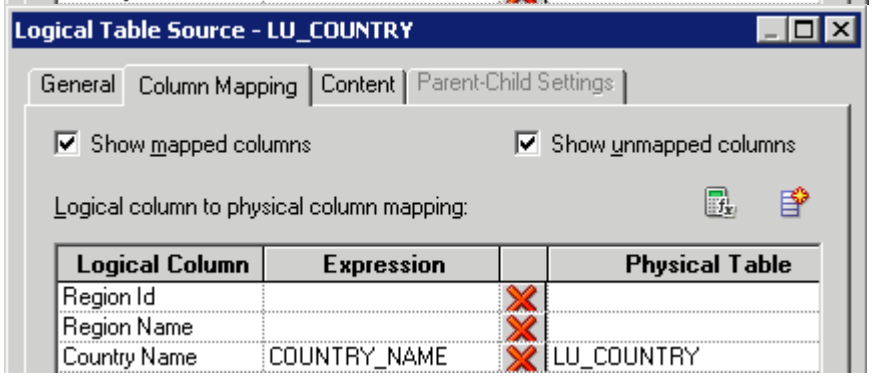




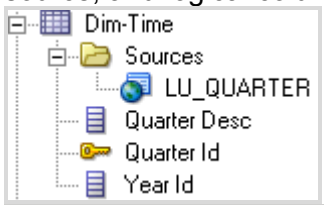
b. Create a **Dim-Region** logical table with the following logical columns, logical table sources, and logical column mappings:

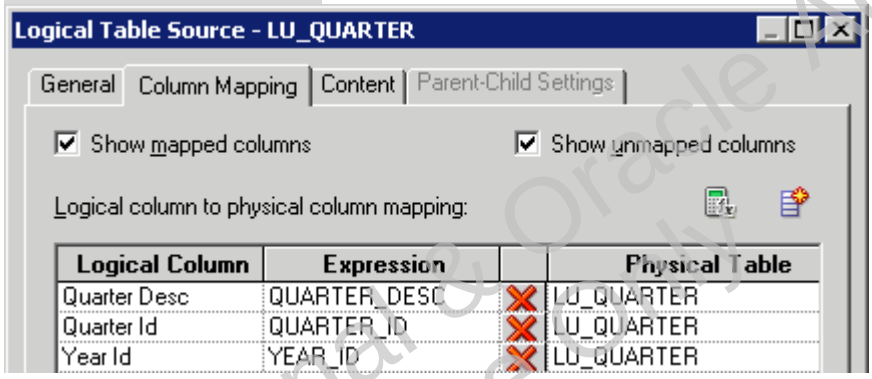




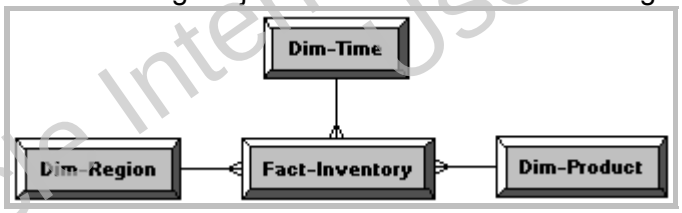


c. Create a **Dim-Time** logical table with the following logical columns, logical table source, and logical column mappings:





d. Create the logical joins in the business model diagram:



e. Drag the **Inventory** business model to the Presentation layer.

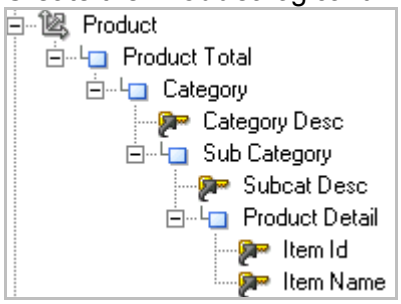
f. Save the repository.

g. Check consistency.

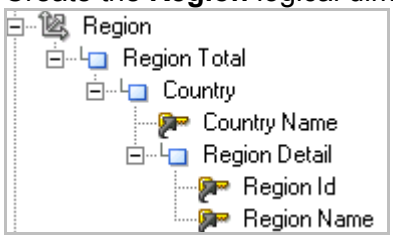
h. Click **Yes** to make the Inventory business model available for queries. Fix any errors or warnings before you proceed.

5. Create three level-based logical dimensions in the Inventory business model. You need the logical dimensions because the inventory measures that you create later in this practice are semi-additive. That is, they have dimension-specific aggregation rules. You must build the logical dimensions in order to define these rules.

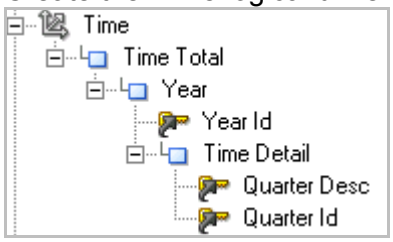
   a. Create the **Product** logical dimension:



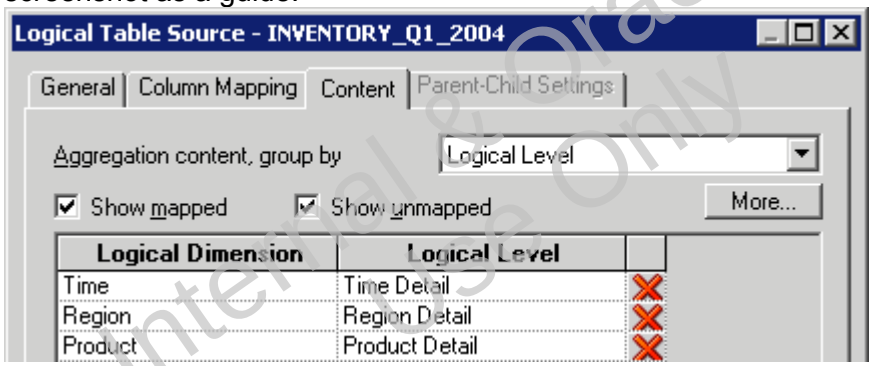
   b. Create the **Region** logical dimension:



   c. Create the **Time** logical dimension:



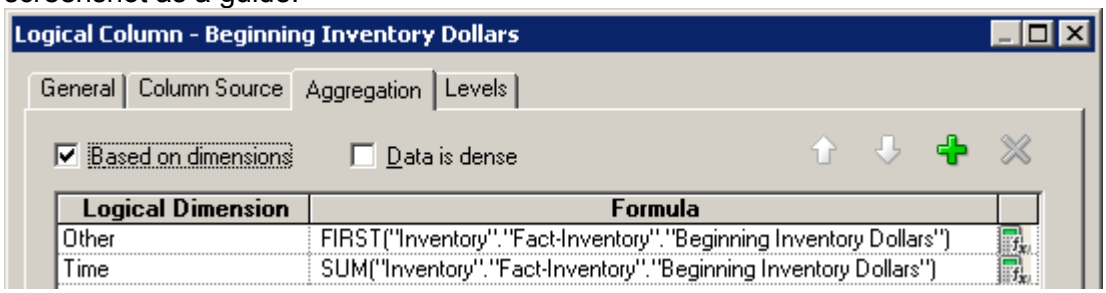
   d. Set the aggregation content for all of the fact logical table sources. Use the following screenshot as a guide:



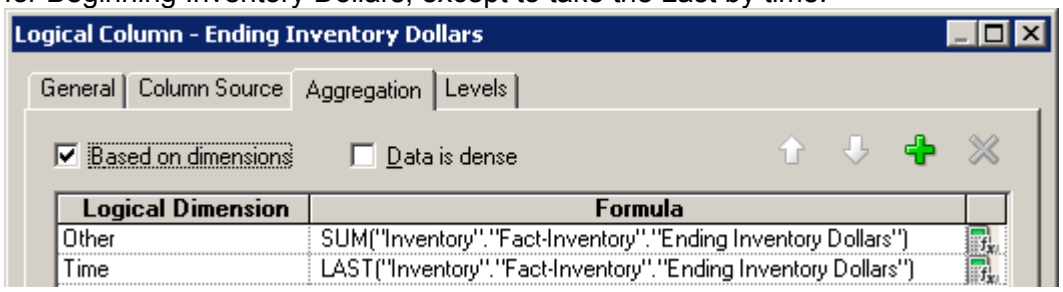
6. Add dimension-specific aggregation rules to the measures.

   a. Create aggregation rules for **Beginning Inventory Dollars**. The rules are to sum beginning inventory dollars over product and region, and then take the first of these by time to get beginning inventory dollars. Double-click the logical column, click the **Aggregation** tab, and check **Based on dimensions** to begin. Use the following

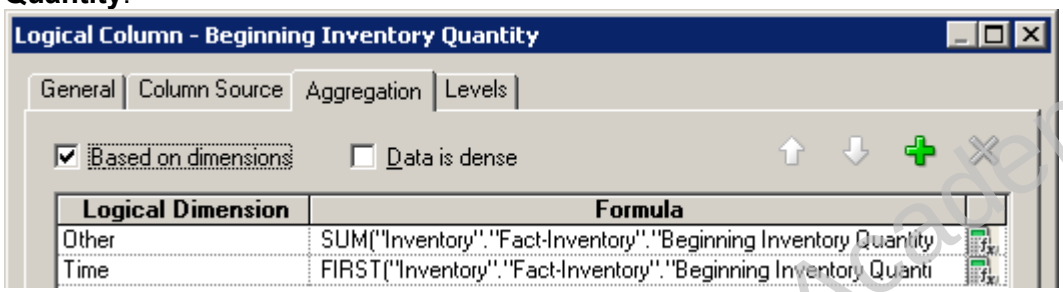
screenshot as a guide.

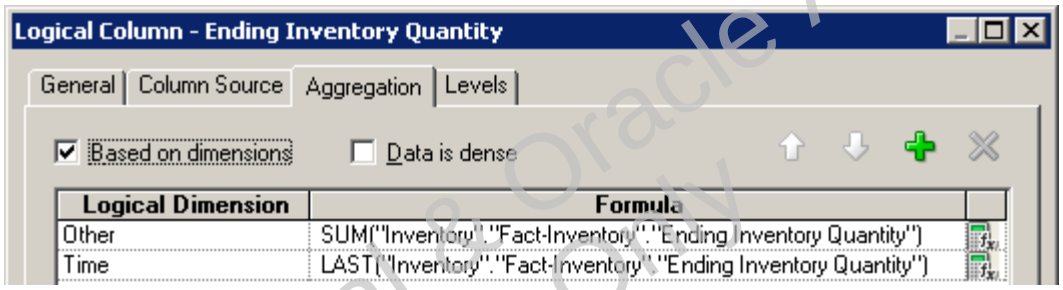


b. Create aggregation rules for **Ending Inventory Dollars**. The rule is the same as that for Beginning Inventory Dollars, except to take the Last by time.



c. Repeat the process for **Beginning Inventory Quantity** and **Ending Inventory Quantity**.





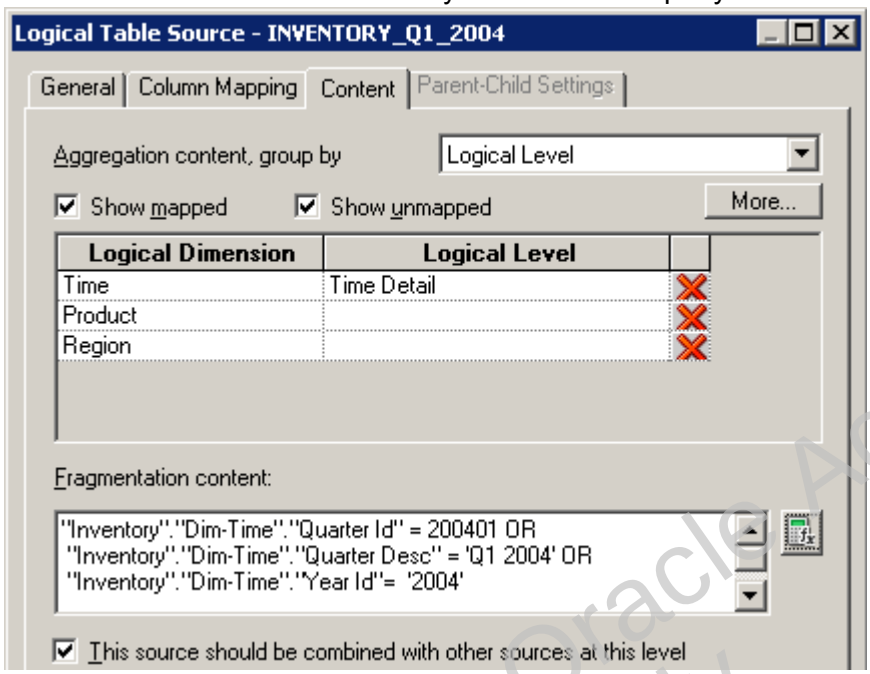
d. Save the repository without checking consistency.

7. Define the fragmentation content for the Fact-Inventory logical table sources.
   a. Right-click the **LU_QUARTER** table in the Physical layer and select **View Data**. You use this information to build the fragmentation content for the logical table sources in

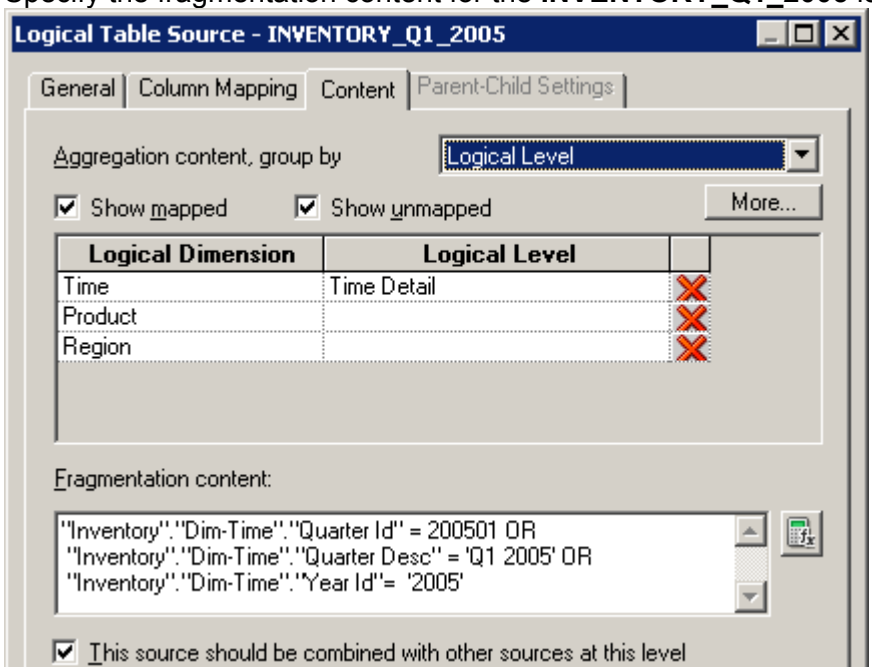
the steps that follow.

QUARTER_DESC	QUARTER_ID	YEAR_ID
Q1 2004	200401	2004
Q2 2004	200402	2004
Q3 2004	200403	2004
Q4 2004	200404	2004
Q1 2005	200501	2005
Q2 2005	200502	2005
Q3 2005	200503	2005
Q4 2005	200504	2005

b. Specify the fragmentation content for the **INVENTORY_Q1_2004** logical table source. Make sure that **This source should be combined with other sources at this level** is selected. This lets Oracle BI Server know that to obtain a complete set of information at this level, the sources must be combined (union). If deselected, the server would infer that the sources contained redundant information. Also, notice that the fragmentation content is specified in terms of the logical columns from the dimension table. That is, it uses the values that a user is likely to include in a query filter.

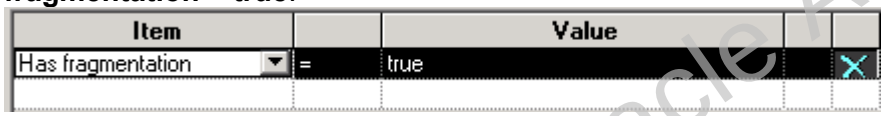


c. Specify the fragmentation content for the **INVENTORY_Q1_2005** logical table source.



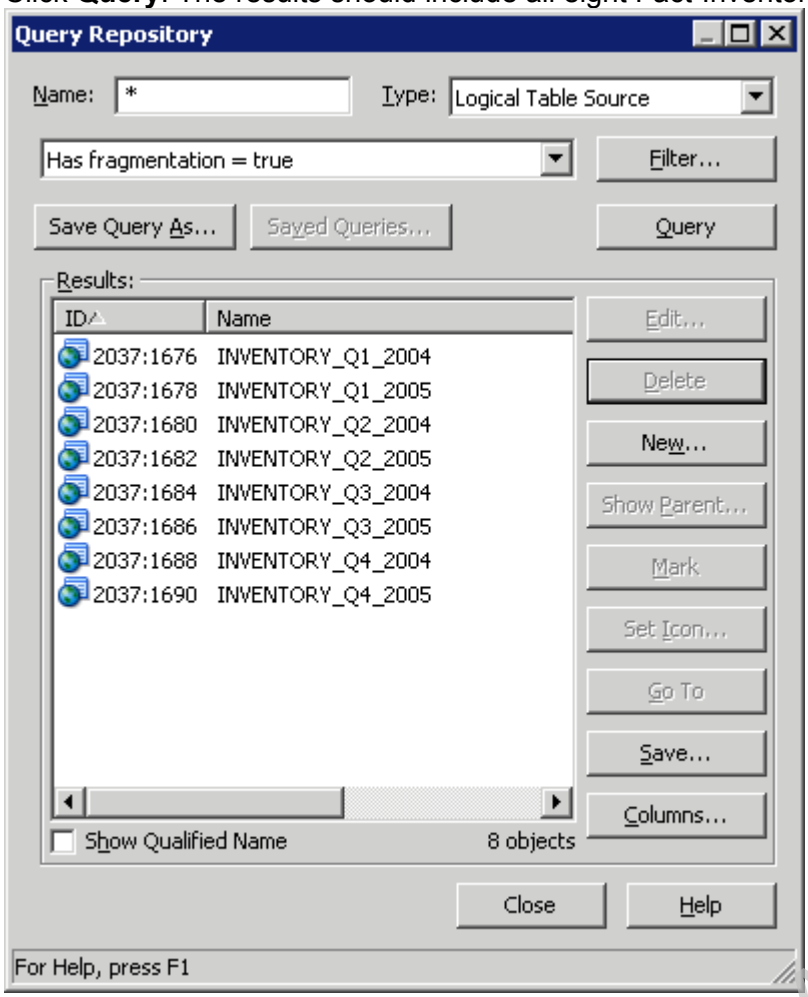
d. Use these two examples as models and specify fragmentation content for the remaining six logical table sources. **Hint:** Copy and paste the fragmentation content formula, and then modify accordingly.

8. Use the Query Repository Utility to check fragmentation content.
   a. Select **Tools > Query Repository**.
   b. In the Type field, select **Logical Table Source**.
   c. Click the **Filter** button.
   d. In the Item field, select **Has fragmentation**. Ensure that the filter condition is **Has fragmentation = true**.



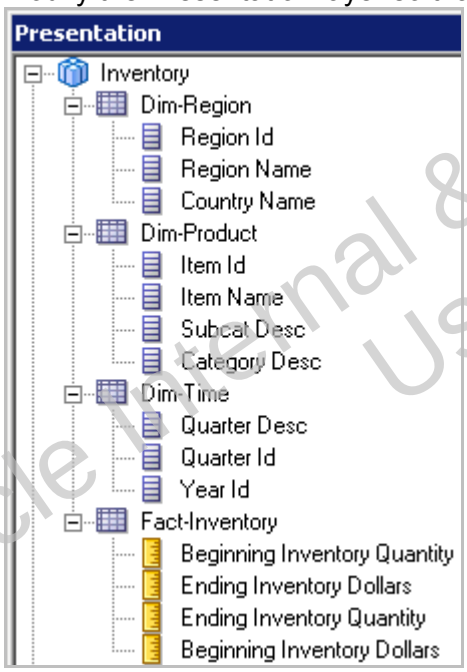
   e. Click **OK**.

f. Click **Query**. The results should include all eight Fact-Inventory logical table sources.



g. Close the Query Repository utility.

h. Modify the Presentation layer so that it resembles the following screenshot:



i. Save the repository.

j. Check consistency. Fix any error or warning messages before you proceed.

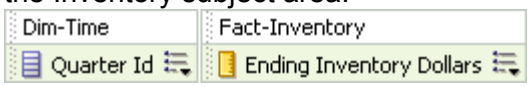
k. Close the repository.

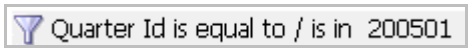
9. Test your work.

a. Use FMW Control Enterprise Manager to load the **Inventory** repository and restart the **Oracle BI components**.

b. Open the **Inventory** repository in online mode and set the logging level to **2** for the **weblogic** user.

c. Sign in to Oracle BI as **weblogic/welcome1** and create the following query and filter in the Inventory subject area:





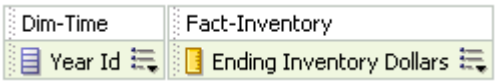
d. Click **Results**.

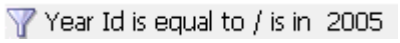
Quarter Id	Ending Inventory Dollars
200501	$439,871

e. Check the query log. The query log should show that the SQL included only a single logical table source (fragment): INVENTORY_Q1_2005. This is because only this logical table source specified fragmentation content that included QUARTER_ID = 200501.

```
select T1607.QUARTER_ID as c1,
    sum(T1536.EOH_DLL) as c2
from
    LU_QUARTER T1607,
    INVENTORY_Q1_2005 T1536
where  ( T1536.QUARTER_ID = T1607.QUARTER_ID and T1536.QUARTER_ID = 200501
group by T1607.QUARTER_ID
order by c1
```

f. Create another query and filter:





g. Click **Results**.

Year Id	Ending Inventory Dollars
2005	$468,396

h. Examine the query log. Notice that there are four query blocks now, one for each logical table source with specified fragmentation content that included YEAR_ID = 2005. Also, notice that the query combines the results (union all), orders the results by quarter within year, and selects the last row for each year. In this case, it is only one

year, 2005. The following screenshot shows only a portion of the query log SQL:

```
select distinct D1.c2 as c1,
     LAST_VALUE(D1.c3 IGNORE NULLS) OVER (PARTITION BY
from
     (select D3.c2 as c2,
               sum(D3.c3) as c3,
               D3.c4 as c4
          from
               ((select T1302.YEAR_ID as c2,
                         T1231.EOH_DLL as c3,
                         T1302.QUARTER_ID as c4
                    from
                         LU_QUARTER T1302,
                         INVENTORY_Q1_2005 T1231
                    where  ( T1231.QUARTER_ID = T1302.
                    union all
                    select T1302.YEAR_ID as c2,
                         T1247.EOH_DLL as c3,
                         T1302.QUARTER_ID as c4
                    from
                         LU_QUARTER T1302,
                         INVENTORY_Q2_2005 T1247
                    where  ( T1247.QUARTER_ID = T1302.
                    union all
                    select T1302.YEAR_ID as c2,
                         T1263.EOH_DLL as c3,
                         T1302.QUARTER_ID as c4
                    from
                         LU_QUARTER T1302,
                         INVENTORY_Q3_2005 T1263
                    where  ( T1263.QUARTER_ID = T1302.
                    union all
                    select T1302.YEAR_ID as c2,
                         T1279.EOH_DLL as c3,
                         T1302.QUARTER_ID as c4
                    from
                         LU_QUARTER T1302,
                         INVENTORY_Q4_2005 T1279
                    where  ( T1279.QUARTER_ID = T1302.
               ) D3
          group by D3.c2, D3.c4
     ) D1
order by c1
```