UNIT 11 Navigation Paths

Lesson 1

UNIT OBJECTIVES

- Define a navigation path

Unit 11
Lesson 1

Defining Drill Down Navigation Paths

LESSON OVERVIEW

The SAP BusinessObjects reporting tools, specifically Web Intelligence, have drilling capabilities to enable end users to analyze data at predefined levels of details. These predefined navigational paths are created in the business layer of the universe exclusively for this drilling functionality.

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Define a navigation path

Navigation Paths

Navigation path is an object that defines the drill path used in SAP BusinessObjects reporting tools. A drill path is a list of drillable business objects that allows a report analyst to drill down on a dimension.

A navigation path object can be one of two types: Default or Custom.

Table 23: Navigation Paths

Navigation path type	Description
Default	The path is defined by the hierarchical organization of the business objects in the business layer. If the business layer contains an analysis dimension, the navigation paths include the dimensions under each analysis dimension. Otherwise, the navigation paths are the dimensions under each folder.You can view the default navigation path in the **Navigation Paths** tab of the business layer editor. The default path cannot be edited.
Custom	You define the path based on the available dimensions.

A navigation path is an ordered series of related dimension objects that are used for multi-dimensional analysis. For example, a geographical navigation path could group together dimension objects such as Country, Region, and City.

Multi-dimensional analysis is a technique for manipulating data so that it can be viewed from different perspectives and at different levels of detail. In Business Objects end user querying tools, users can analyze data at different levels of detail using a feature called drill mode.

The Natural Navigation Path example shows a navigation path of the dimension objects Country, Region, and City. At the highest level, the user sees a Country. At the next level down, the Country is broken down into more detail: the regions. At the next lower level, the regions are broken down into more detail: the towns. A user can analyze a measure object, such as Sales Revenue, against any of the levels in the navigation path.

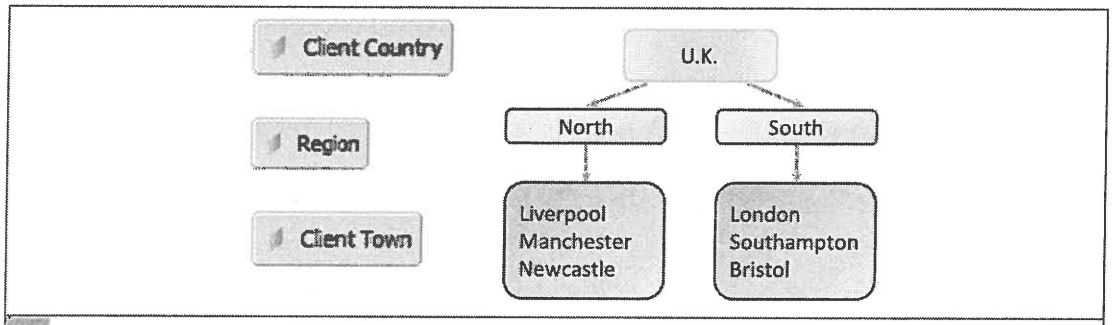


Figure 57: Natural Navigation Path

Navigation Path Creation

Natural Hierarchies

A natural navigation path is one that follows a naturally occurring pattern from the most general at the highest level to the most detailed at the lowest level. Examples of natural hierarchies can be found in the geographical definitions of places and in the measurement of time:

- Country, Region, State, City, District, and Street
- Year, Quarter, Month, Week, and Day

Logical Hierarchies

BusinessObjects hierarchies are not restricted to natural hierarchies. You can construct a navigation path from any related group of dimension objects that create a sensible analysis path. The relationship between the dimension objects in a navigation path normally are one-to-many as you descend through the levels. For example, one country has many towns; one town has many showrooms; one showroom has many franchises.

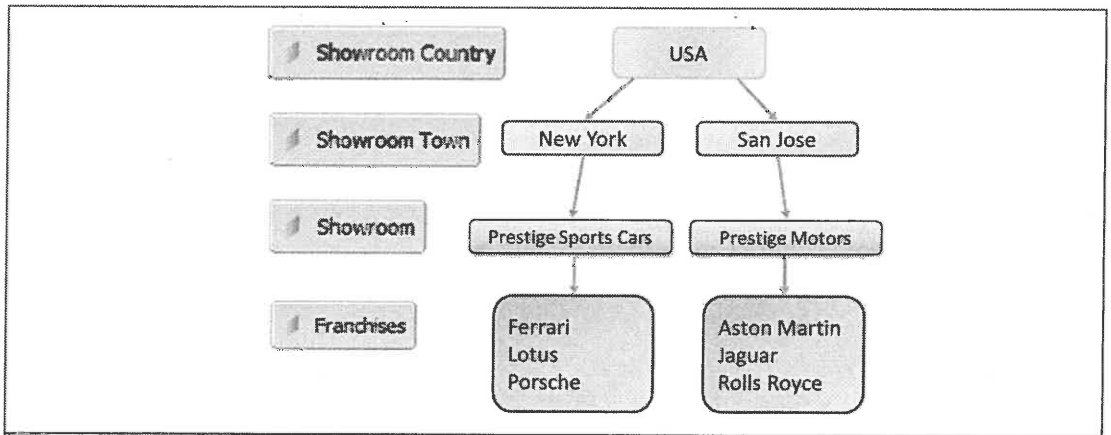


Figure 58: Logical Navigation Paths

Unit 11
Exercise 26

Create a Navigation Path

Business Example

The business users want to use the drilling functionality of Web Intelligence. You must create navigation paths for them to follow.

1. Choose the business layer. In the Client, Car, Showroom, and Financial Period folder, organize the dimension objects in your folder from highest level to lowest level.

> Note:
> Ensure that the *Associate List of Values* is set to the default.

2. In the business layer, choose the *Navigation Paths* pane. Use *Insert Navigation Path* or drag and drop the folders listed in step 1.
3. Save your business layer.

Unit 11
Solution 26

Create a Navigation Path

Business Example

The business users want to use the drilling functionality of Web Intelligence. You must create navigation paths for them to follow.

1. Choose the business layer. In the Client, Car, Showroom, and Financial Period folder, organize the dimension objects in your folder from highest level to lowest level.

> Note:
> Ensure that the *Associate List of Values* is set to the default.

   a) Drag and drop the dimension objects in the following order:

Folder	Dimension Objects
Client	Client Country, Region, Area, Town, Client Name. Everything else is an attribute.
Car	Maker Country, Maker, Category of Car, Model
Showroom	Showroom Country, Showroom Town, Showroom
Financial Period	Financial Year, Financial Quarter, Financial Month

2. In the business layer, choose the *Navigation Paths* pane. Use *Insert Navigation Path* or drag and drop the folders listed in step 1.
   a) Drag and drop each folder listed in step 1 to the *Navigation* pane.
   b) Choose the Car navigation path, and delete the Luxury Model and Model Color dimensions on the right.
   c) From the Car navigation path, delete the Luxury Model and the Model Color dimensions.
3. Save your business layer.
   a) On the tool bar, choose *Save*.

LESSON SUMMARY

You should now be able to:

- Define a navigation path

Learning Assessment

1. A navigation path object can be one of two types. Identify those two types from the list.

   Choose the correct answers.

   - ☐ A Default
   - ☐ B Regular
   - ☐ C Custom
   - ☐ D Irregular

Learning Assessment - Answers

1. A navigation path object can be one of two types. Identify those two types from the list.

   Choose the correct answers.

   - [X] A Default
   - [ ] B Regular
   - [X] C Custom
   - [ ] D Irregular

UNIT 12 Derived Tables

Lesson 1

UNIT OBJECTIVES

- Use derived tables

Unit 12
Lesson 1

Creating Derived Tables

LESSON OVERVIEW
Universes can make use of derived data exclusively for reporting purposes to minimize the need to alter the underlying database. They can also leverage primary and foreign keys that have been defined in the database to improve the efficiency of the querying process.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Use derived tables

Derived Tables

A derived table combines other tables using calculations and functions. You can create objects in the business layer on a derived table the same way that you create objects for a standard table. A derived table consists of a set of SQL statements that you create in the business layer, which you can then use as a logical table to create objects.

Derived tables are similar to views in the database, but because they are defined in the data foundation, they give designers more flexibility. You can see a derived table as a query that can be referenced as a table. The table definition SQL is inserted into an end-user query at run time.

Note:
As it is a query and not a physical table, note that the same performance issues that affect queries can also affect derived tables.

Derived tables can be used to merge data together from different tables, when designers are unable to conform the underlying data source, and want to normalize or renormalize the business layer schema.

Use derived tables in the following situations:

- To create a table with columns from other tables. The column definitions can include complex calculations and functions.
- To create a single table that combines two or more tables (called merged tables).
- To create a table that contains a selection of columns from different tables.

To insert a derived table with all the columns in the original table, right-click the table header in the data foundation view and select *Insert → Derived Table*.

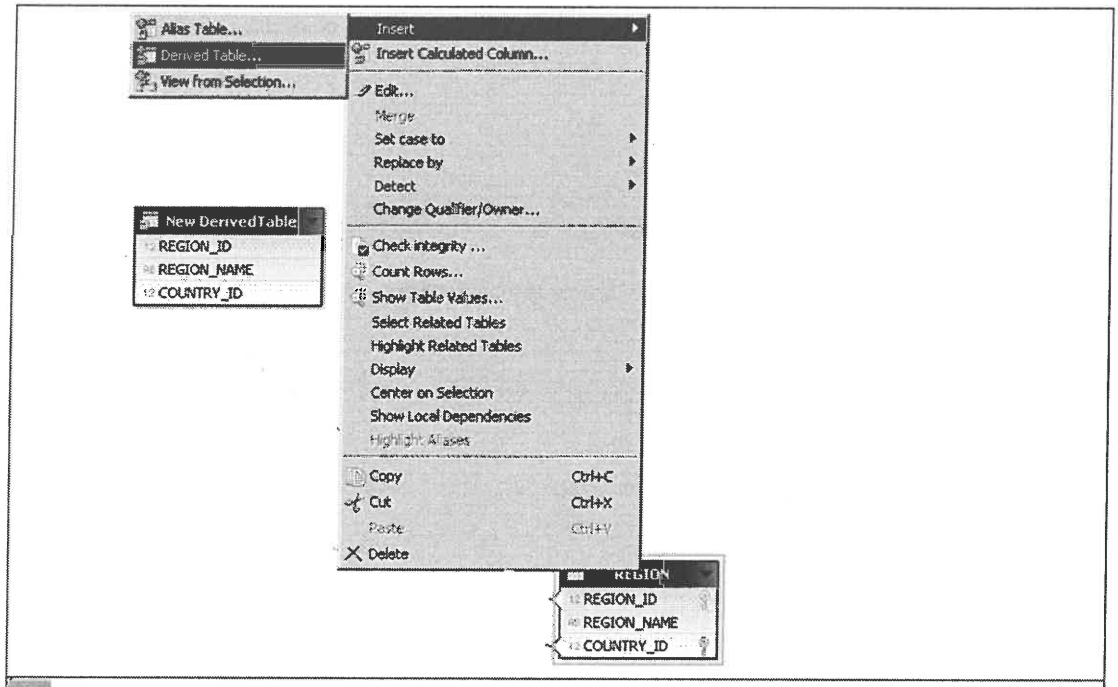


Figure 59: Insert a Derived Table from the Original Table

To insert a derived table and specify the columns, select Insert Derived Table from the Insert menu in the data foundation view.

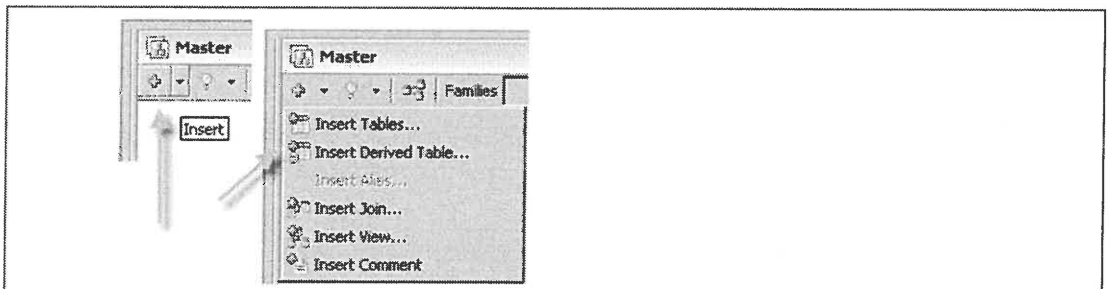


Figure 60: Insert a Derived Table

Enter the table definition in the *Edit Derived Table* dialog box.

Give the derived table a unique name within the data foundation. In a multisource-enabled data foundation, if you want to include database-specific functions in the definition of the derived table, select database-specific syntax. From the tables listed in the Tables and Database Table boxes, drag columns to the Expression box to include in the derived table.

You can also use the SQL Builder, which works like the query panel, to select columns for the derived table. The functions allowed in the expression definition are listed in the Functions box. You can drag functions to the Expression box. To edit a derived table, right-click the table header of the derived table and select Edit.

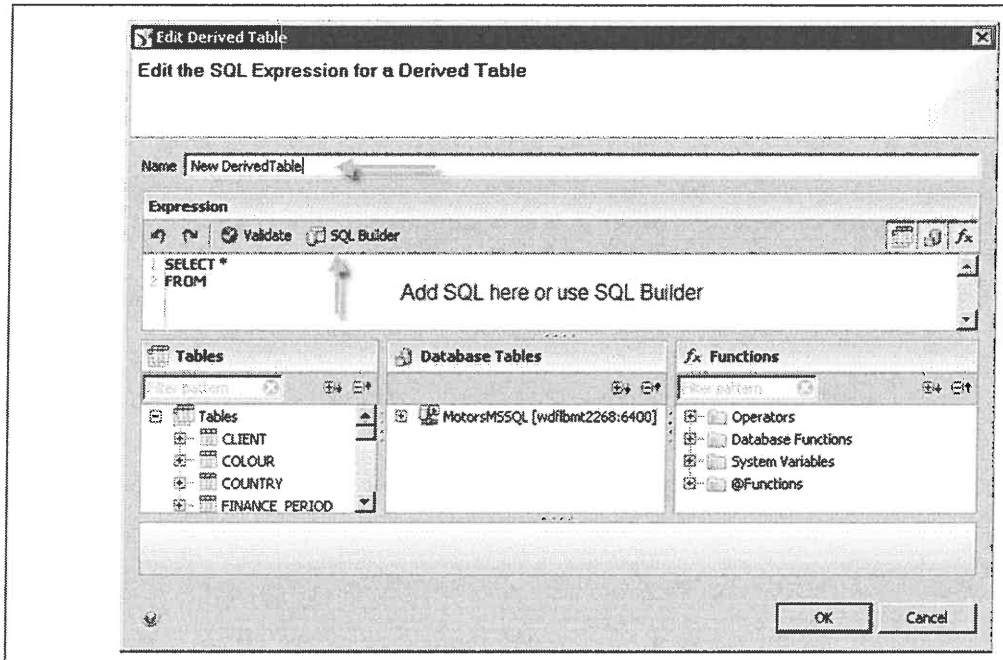


Figure 61: Edit the SQL Expression

Derived Tables by Merging Tables

Merging tables lets you insert a derived table into the data foundation that consists of the combined columns from two or more tables linked by joins. To merge tables, in the data foundation view, select the tables you want to merge in one of the following ways:

- Right-click a table and select Related Tables.
- Click the table headers while holding down the CTRL key. Right-click the selection of tables and then select Merge.

Enter a name for the table that is unique within the data foundation. The merged table is inserted as a derived table. The new table is joined to any tables that the original tables were joined to. The original tables become obsolete. You have the choice of deleting them. If you choose to keep the original tables, the joins linking those tables are deleted; however, the tables stay in the data foundation. To edit a merged table, right-click the table header, and select Edit Derived Table.

Note:
In a multisource-enabled data foundation, a derived table that results from a merge creates expressions using SQL-92 standard syntax. To use database-specific SQL, you edit the derived table and explicitly select database-specific syntax.

Unit 12
Exercise 27

Use Derived Tables

Business Example

Into your data foundation, insert a derived table to show the number of transactions per customer.

1. Using your data foundation, insert a derived table to show the number of transactions per customer.
2. Name the newly derived table DT_Best_Cust.
3. Create the SQL statement so that it looks like this:

```
SELECT CLIENT.CLIENT_ID, COUNT(SALE.SALE_ID) AS
Number_of_Transactions FROM CLIENT, SALE
WHERE CLIENT.CLIENT_ID = SALE.CLIENT_ID GROUP BY CLIENT.CLIENT_ID
```

4. To verify your SQL statement, validate the syntax.
5. Insert a join between the DT_Best_Cust and CLIENT tables with a cardinality of (1:1).
6. Every join (except shortcut joins) must exist in at least one context. Add the new join to the Sales and to the Rentals contexts. The join between the DT_Best_Cust and CLIENT tables is from the Client_ID primary key in the CLIENT table to the Client_ID foreign key in the DT_Best_Cust table. The table schema looks similar to this:

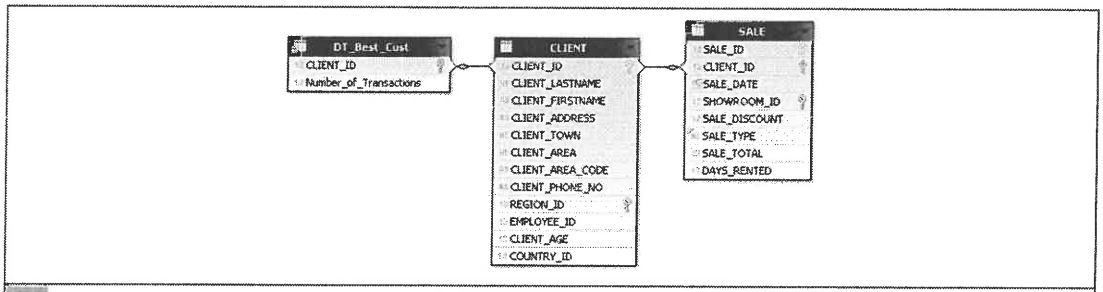


Figure 62: DT_Best_Customer

7. Add the Number of Transactions object to the Client folder. Define the object as a measure object, and ensure to clear the *Associate a List of Values* option. Set the Aggregation function for this measure.
8. In the *Query* pane, create a query with Client Country and Total Transactions. A derived table is created to show the number of transactions per customer.

Unit 12
Solution 27

Use Derived Tables

Business Example

Into your data foundation, insert a derived table to show the number of transactions per customer.

1. Using your data foundation, insert a derived table to show the number of transactions per customer.
   a) Right-click in the *Data Foundation*.
   b) Choose *Insert* → *Derived Table*.
2. Name the newly derived table DT_Best_Cust.
   a) Enter **`DT Best_Cust`**.
3. Create the SQL statement so that it looks like this:

```
SELECT CLIENT.CLIENT_ID, COUNT(SALE.SALE_ID) AS
Number_of_Transactions FROM CLIENT, SALE
WHERE CLIENT.CLIENT_ID = SALE.CLIENT_ID GROUP BY CLIENT.CLIENT_ID
```

   a) In the *Expression* area, enter the SQL statement.
4. To verify your SQL statement, validate the syntax.
   a) Choose *Close*.
   b) Choose *OK*.
5. Insert a join between the DT_Best_Cust and CLIENT tables with a cardinality of (1:1).
   a) Choose *Insert* → *Join* and create the following statement:

```
DT_Best_Cust.CLIENT_ID = CLIENT.CLIENT_ID
```

   b) Set the cardinality to 1:1.
   c) Choose *OK*.
6. Every join (except shortcut joins) must exist in at least one context. Add the new join to the Sales and to the Rentals contexts. The join between the DT_Best_Cust and CLIENT tables is from the Client_ID primary key in the CLIENT table to the Client_ID foreign key in the DT_Best_Cust table. The table schema looks similar to this:

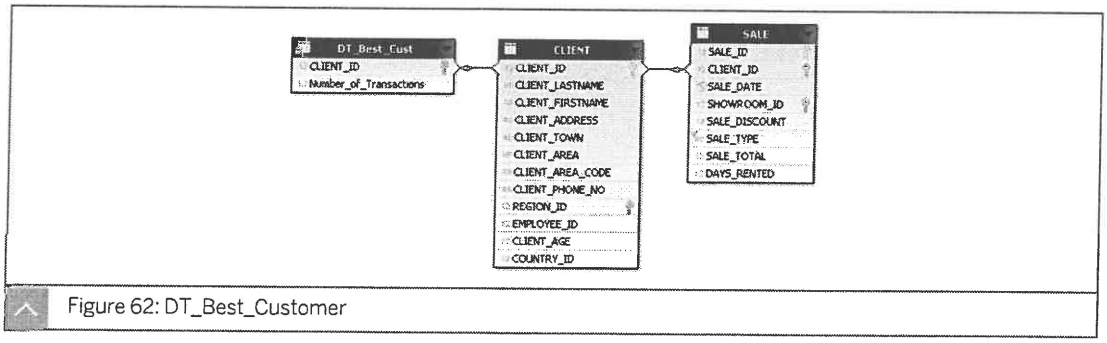


Figure 62: DT_Best_Customer

a) Click the Sale context.

b) To change the State to Include, click the new join until the *State* is Included.

c) Double-click the Rental context.

d) To change the State to Include, click the new join until the *State* is Included.

e) Save the data foundation.

7. Add the Number of Transactions object to the Client folder. Define the object as a measure object, and ensure to clear the *Associate a List of Values* option. Set the Aggregation function for this measure.
   a) Go to the *Business Layer* and drag the Number_of_transactions column over to the Client folder.
   b) Right-click the object and choose *Measure with Aggregate Function Sum*.
   c) Choose the Number of Transactions measure.
   d) On the object properties, choose the *Advanced* tab.
   e) Deselect the *Associate List of Values* option.
8. In the *Query* pane, create a query with Client Country and Total Transactions.
   a) In either the *Query* pane, create a query with Client Country and Total Transactions.

A derived table is created to show the number of transactions per customer.

LESSON SUMMARY

You should now be able to:

- Use derived tables

Learning Assessment

1. Under what situations would you use derived tables?

   Choose the correct answers.

   - [ ] A To create a table that contains a selection of columns from different tables.
   - [ ] B To create a single table that combines two or more tables.
   - [ ] C To create a table that contains no columns and rows
   - [ ] D To create a table with columns from other tables.

Learning Assessment - Answers

1. Under what situations would you use derived tables?

   Choose the correct answers.

   [X] A To create a table that contains a selection of columns from different tables.

   [X] B To create a single table that combines two or more tables.

   [ ] C To create a table that contains no columns and rows

   [X] D To create a table with columns from other tables.

UNIT 13 Key Awareness

Lesson 1

UNIT OBJECTIVES

- Apply key awareness

Unit 13
Lesson 1

Defining Numeric Keys

LESSON OVERVIEW
Key awareness is the ability to take advantage of the database indexes on key columns to speed up data retrieval.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Apply key awareness

Query Performance Improvement with Key Awareness

Key awareness is the ability to take advantage of the indexes on key columns to speed data retrieval.

The objects that you create in the universe are based on database columns that are meaningful to an end user. For example, a Customer object retrieves the field that contains the customer name. In this situation, the customer table has a primary key (an integer) that is not meaningful to the end user, but is important for database performance.

Key Awareness

You use the Information Design Tool to apply primary and foreign key index awareness:

Primary key index awareness

By applying primary key index awareness on an object, Information Design Tool can substitute the object's column value with the associated index value. The generated query uses the key value instead of the actual column value.

Foreign key index awareness

By applying foreign key index awareness on an object, Information Design Tool can restrict the values returned without the need to join the tables.

When you set up index awareness in Universe Designer, this determines which database columns are primary and foreign keys. It has a dramatic effect on query performance in the following ways:

- The Information Design Tool can take advantage of the indexes on key columns to speed data retrieval.
- The Information Design Tool can generate SQL that filters in the most efficient way.

  This is particularly important in a star schema database. If you build a query that involves filtering on a value in a dimension table, Information Design Tool can apply the filter directly on the fact table by using the dimension table foreign key. This eliminates unnecessary and costly joins to dimension tables.

Note:
Information Design Tool does not ignore duplicates with index awareness. If two customers have the same name, Universe Designer retrieves only one unless it is aware that each customer has a separate primary key.

For example, a query in Web Intelligence Rich Client, with Client Name in the Result Objects pane, and Client Country in the Query Filters pane, restricted on the countries United Kingdom and the United States, produces the following SQL when primary key index has been applied:

```
SELECT
CLIENT.CLIENT_LASTNAME +', '+ CLIENT.CLIENT_FIRSTNAME,
FROM
CLIENT INNER JOIN REGION ON (CLIENT.REGION_ID=REGION.REGION_ID)
INNER JOIN COUNTRY COUNTRY_REGION ON
(COUNTRY_REGION.COUNTRY_ID=REGION.COUNTRY_ID)
WHERE
(
COUNTRY_REGION.COUNTRY_ID In ( 44,1 )
AND (COUNTRY_REGION.COUNTRY_ID=CLIENT.COUNTRY_ID)
)
```

The restricted country name values have been replaced by their respective primary key index value to help increase query performance.

Avoiding Joins in Tables

With foreign key index awareness, you can tell Universe Designer that Country_ID is the primary key of the Country_Region table and that it also appears in the Client table as a foreign key. Using this information, Universe Designer can restrict the countries without joining to the Country_Region table.

Reducing the number of joins in a query can help improve query performance. The SQL from the primary key index awareness example improved performance by restricting on the primary key (indexed) of the Country_Region table.

The example query includes the Client and Country_Region tables, which both can provide the Country_ID column.

The Country_Region table is only needed to satisfy the WHERE clause. It is not needed in the SELECT or GROUP BY clauses.

Using index awareness it is possible to remove the Country_Region table from the query completely, if you specify that BusinessObjects uses the Maker table to retrieve the Country_ID data from.

By defining the foreign key in the Client Country object, the same query returns the following SQL:

```
SELECT
CLIENT.CLIENT_LASTNAME +', '+ CLIENT.CLIENT_FIRSTNAME,
FROM
CLIENT
WHERE
(
CLIENT.COUNTRY_ID In ( 44,1 )
)
```

The Country_Region table is no longer referenced in the SQL. This Country_Region table no longer appears in the FROM clause, in the WHERE clause as a join to the Client table, or as a table used to retrieve the condition values from.

Unit 13
Exercise 28

Set Up Key Awareness

Business Example

Set up key awareness.

1. Using the Client Country object in your Business Layer, enter the following under Keys:

Key Type	SELECT Statement
Primary Key	COUNTRY.COUNTRY_ID
Foreign Key	REGION.COUNTRY_ID
Foreign Key	CLIENT.COUNTRY_ID

2. Add foreign keys from the REGION and CLIENT tables.
3. Create a query in the *Query* drawer with Client Town and Client Name.
4. Apply a query filter, and use the values from list option to restrict the data to a single country, such as the United States.
5. View the SQL.
   Notice that the WHERE clause uses `CLIENT.COUNTRY_ID = 1`.

Unit 13
Solution 28

Set Up Key Awareness

Business Example

Set up key awareness.

1. Using the Client Country object in your Business Layer, enter the following under Keys:

Key Type	SELECT Statement
Primary Key	COUNTRY.COUNTRY_ID
Foreign Key	REGION.COUNTRY_ID
Foreign Key	CLIENT.COUNTRY_ID

   a) Open the business layer and choose the Client Country object and make sure that Client country has a default List of values.

   b) Choose the *Keys* tab, and at the bottom, choose *Detect*.

> Note:
> Detection only takes place if primary and foreign key have been defined in the data foundation beforehand.

   c) Choose *OK*.

   d) Delete any Foreign Keys that do not match the entries in the table. Confirm that the Primary Key is COUNTRY.COUNTRY_ID.

2. Add foreign keys from the REGION and CLIENT tables.

   a) Choose *Add Key*.

   b) Click the area under *Select* and choose *SQL*.

   c) Create the following SQL Expression:

```
REGION.COUNTRY_ID
```

   d) Choose *OK*.

   e) Choose *Add Key*.

   f) Click the area under *Select* and choose *SQL*.

   g) Create the following SQL Expression:

```
CLIENT.COUNTRY_ID
```

   h) Choose *OK*.

3. Create a query in the *Query* drawer with Client Town and Client Name.
   a) In the *Business Layer*, select the *Query* drawer.
   b) Choose *Create Query*.
   c) Drag the Client Town and Client Name objects into the *Refresh Objects* area.
   d) Choose *Results*.
   e) Note the results.
4. Apply a query filter, and use the values from list option to restrict the data to a single country, such as the United States.
   a) Drag the Client Country object into the *Query Filter* section.
   b) Use "Equal to" as the operator.
   c) Choose the drop-down box, use Value(s) from list, and choose USA.
5. View the SQL.
   Notice that the WHERE clause uses `CLIENT.COUNTRY_ID = 1`.
   a) At the top, choose *View Script*.
   b) Notice that the number is used and not the name of the country.

LESSON SUMMARY
You should now be able to:

- Apply key awareness

Unit 13

Learning Assessment

1. Index awareness is the ability to take advantage of the indexes on key columns to filter data retrieval.

   Determine whether this statement is true or false.

   ☐ True

   ☐ False

Learning Assessment - Answers

1. Index awareness is the ability to take advantage of the indexes on key columns to filter data retrieval.

   Determine whether this statement is true or false.

   ☐ True

   ☒ False

UNIT 14 Universe Management with Data Foundation and Business Layer Views

Lesson 1

Lesson 2

UNIT OBJECTIVES

- Manage a universe using the data foundation view
- Manage a universe using the business layer view

Unit 14
Lesson 1

Managing a Universe using the Data Foundation View

LESSON OVERVIEW
One of the primary responsibilities of the universe designer is to ensure that the universe is constructed to produce accurate data results as efficiently as possible. This lesson discusses how you can use Data Foundation Views to help manage your universe more effectively.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Manage a universe using the data foundation view

Data Foundation Views

A custom data foundation view is a subset of the data foundation Master view.

Use custom data foundation views when you are:

- editing a data foundation that contains many tables
- working with a subset of tables

You can define multiple custom views for the data foundation.

Unit 14
Exercise 29

Insert a Custom Data Foundation View

Business Example

To create a subset of the data foundation Master view, insert a custom data foundation view.

1. Insert a custom data foundation view.
2. Add the ABSENCE and EMPLOYEE tables to the view.
3. Join the tables by the EMP_ID column and set the cardinalities.
4. Go back to the Master view and note that the new view is now part of the Master.

Unit 14
Solution 29

Insert a Custom Data Foundation View

Business Example

To create a subset of the data foundation Master view, insert a custom data foundation view.

1. Insert a custom data foundation view.
   a) Open the Data Foundation of the Motors project.
   b) Choose *Insert* → *View* and name the view **`Staff Information`**.
   A new tab appears at the bottom of the view pane. Initially, the view is empty.
2. Add the ABSENCE and EMPLOYEE tables to the view.
   a) Click the Insert button and choose *Insert Tables*.

Hint:
To select and add multiple tables, click the table headers while holding down the CTRL key.

Hint:
Another way to insert a view is to choose one or several tables and then right-click and choose *Insert View from Selection*. The view is inserted and contains the selected tables.

   b) Select the *ABSENCE* and *EMPLOYEE* tables.
   c) Click *OK*.
3. Join the tables by the EMP_ID column and set the cardinalities.
4. Go back to the Master view and note that the new view is now part of the Master.

LESSON SUMMARY

You should now be able to:

- Manage a universe using the data foundation view

Unit 14
Lesson 2

Managing a Universe using the Business Layer View

LESSON OVERVIEW
This lesson discusses how you can use Business Layers Views to help manage your universe more effectively.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Manage a universe using the business layer view

Business Layer Views

You can modify the display of business layer objects by using business layer views to restrict the number of objects displayed in the business layer pane. Use business layer views to group objects that share a business relationship. Business layer views can be selected in the Query Pane. They can be used to define security, as well as to grant or deny the use of business layer objects to certain users or groups.

Unit 14
Exercise 30

Create and Edit a Business Layer View

Business Example

To restrict the number of objects displayed in the business layer pane or to group objects that share a business relationship, create a business layer view.

1. Create a business layer view.
2. Demonstrate that you can now apply the view.
3. Demonstrate that you can open the new view for editing.

Unit 14
Solution 30

Create and Edit a Business Layer View

Business Example

To restrict the number of objects displayed in the business layer pane or to group objects that share a business relationship, create a business layer view.

1. Create a business layer view.
   a) Open the business layer of the Motors project.
   b) From the business layer, at the top of the *Business Layer* pane, choose the *Manage Business Layer Views* icon.
   The *Edit Business Layer View* dialog appears.
   c) Choose New.
   d) For the name of the view, enter `Sales only`.
   e) In the *Objects in View* box, to include objects and folders in the view, choose the check boxes next to them in the business layer. Check all the folders except Rentals and Day Rental Charges (in the Car folder).

   > Hint:
   > To work with only the objects already included in the view, choose *Show selected objects only*.

   f) Choose *OK*.
2. Demonstrate that you can now apply the view.
   a) In the drop-down list at the top of the *Business Layer* pane, verify that the new view appears and that it can be selected.
3. Demonstrate that you can open the new view for editing.
   a) At the top of the Business Layer pane, choose the *Manage Business Layer Views* icon and choose the view you just created.

LESSON SUMMARY

You should now be able to:

- Manage a universe using the business layer view

Learning Assessment

1. A custom data foundation view is a subset of what data foundation view?

   Choose the correct answer.

   - ☐ A Major
   - ☐ B Minor
   - ☐ C Master
   - ☐ D Motor

2. Which of the following statements about Business Layer View are correct?

   Choose the correct answers.

   - ☐ A Business layer views can only be used to define security.
   - ☐ B Business layer views are used to group objects that share a business relationship.
   - ☐ C Business layer views can be used to restrict the number of objects displayed in the business layer pane.
   - ☐ D Business layer views can be selected in the Query Pane.

Learning Assessment - Answers

1. A custom data foundation view is a subset of what data foundation view?

   Choose the correct answer.

   - [ ] A Major
   - [ ] B Minor
   - [x] C Master
   - [ ] D Motor

2. Which of the following statements about Business Layer View are correct?

   Choose the correct answers.

   - [ ] A Business layer views can only be used to define security.
   - [x] B Business layer views are used to group objects that share a business relationship.
   - [x] C Business layer views can be used to restrict the number of objects displayed in the business layer pane.
   - [x] D Business layer views can be selected in the Query Pane.

UNIT 15 Universe Optimization

Lesson 1

UNIT OBJECTIVES

- Use connection configuration parameters
- Use query script parameters
- Explain best practices for universe design

Unit 15
Lesson 1

Optimizing Universes Using Parameters

LESSON OVERVIEW
One of the primary responsibilities of the universe designer is to ensure that the universe is constructed to produce accurate data results as efficiently as possible.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Use connection configuration parameters
- Use query script parameters
- Explain best practices for universe design

Connection Configuration Parameters

When creating or editing a connection, you can set certain parameters to optimize the connection against the database in use. Query time can be shortened by optimizing the connection. The connection configuration page for relational connections contains parameters used to optimize a universe:

Connection Configuration Parameters to Optimize a Universe

- Define the duration of a connection into a pool.
- Define how BusinessObjects products respond when database resources are not available.
- Define the size of the array fetch.

You can set the following configuration parameters for most relational connections:

- Connection pool mode.
- Array fetch size.
- Login time-out.

Connection Pool Mode

The connection pool mode options refer to how long a universe designer or an end-user using the reporting tools remain connected to the underlying data source.

The following are connection pool mode options:

Table 24: Connection Pool Mode Options

Pool Mode Option	Description

Disconnect after each transaction	Use this option if you want your connection to disconnect immediately after the transaction is completed. Users have to reconnect each time they want to access data.
Keep the connection active for	Use this option if you want your connection to stay active for a defined period of time. You enter a value for the number of minutes. This value is the default Pool timeout value.
Keep the connection active during the whole session (local mode only)	Use this option if you want your connection to stay active during the entire session of the product. The connection ends when the user exits the application.
Pool timeout	If you select **Keep the pool active**, this parameter specifies the length of time to keep the connection open.

Array Fetch Size

Specifies the maximum number of rows authorized with each fetch from the database. For example, if you enter 20, and your query returns 100 rows, the connection retrieves the data in 5 fetches of 20 rows each.

To deactivate array fetch, enter an Array Fetch Size of 1. The Data is retrieved row by row.

Note:
Deactivating array fetch size can increase the efficiency of retrieving your data, but slows server performance. The greater the value in the Array Fetch Size, the faster your rows are retrieved. Ensure that you have adequate client system memory.

Array Bind Size

This parameter is not used for universes created using the Information Design Tool.

Login Time-Out

Specifies the number of minutes that must be spent to establish a connection before an error message is displayed.

JDBC Driver Properties (key=value,key=value)

Specifies values for JDBC driver properties when that driver is selected for the connection. You can define the value of more than one property, separated by commas. For example, the following value for JDBC Driver Properties sets the oracle.jdbc.defaultNChar and defaultNChar driver properties: oracle.jdbc.defaultNChar=true,defaultNChar=true

Note:
If a property is defined in the <driver>.sbo file, the value defined in this parameter is used. For more information about SBO files, see the "Data Access Guide."

Owner Name

For DB2 connections, this parameter adds the name of the owner of the table as a prefix on the table name, to match the DB2 convention for naming tables.

Table Suffix

For DB2 connections, this parameter adds a suffix on the table name, to match the DB2 convention for naming tables.

Depending on the middleware driver selected when creating the connection, certain Custom Parameters may be available. These parameters should be modified only by an advanced user, DBA, or SAP BusinessObjects BI administrator.

Query Script Parameters

The purpose of a universe is to allow reporting tools, such as WebIntelligence, to generate a complete SQL statement to pass to a data source in order to return accurate reporting results. Each data foundation has Query Script Parameters that can be changed to ensure the SQL generated by a reporting tool is as efficient as possible.

Each data source has different Query Script Parameter options. To see a description of all the predefined SQL generation parameters and their values, click the help button.

These Query Script Parameters can be changed and new parameters added in the data foundation. In addition, parameters concerning lists of values can be changed in the business layer properties. These changes should be made after you discuss their implications with your DBA, network team, and SAP BusinessObjects administrative team.

At query time, the query server uses the values it finds in the following order:

1. The value in the business layer, if it is set.
2. The value in the data foundation, if it is set.
3. The default value.

Unit 15
Exercise 31

Edit Query Script Parameters

Business Example

To improve the efficiency of the SQL that Web Intelligence generates and improve query performance, you edit query script parameters.

Note:
Do not alter the query script parameters without first consulting others in your organization such as the DBA, network team, and SAP BusinessObjects administrative team.

1. Using the Motors universe, edit the END_SQL parameter to add the following value to the end of each SQL statement generated:

   ```
   /*@variable('BOUSER')*/
   ```

2. Using the Motors universe, edit the JOIN_BY_SQL parameter to activate it and create a query with the Client Name, Sales Revenue, and Rental Revenue objects.

   View the resulting script and note any differences.

3. Using the Motors universe, edit the ANSI92 and FILTER_IN_FROM parameters to activate them.

   Create a query with the Client Name and Client Country objects and put a Query Filter on the Country object, restricting it to one of the valid Country values. View the resulting script and note any differences.

Unit 15
Solution 31

Edit Query Script Parameters

Business Example

To improve the efficiency of the SQL that Web Intelligence generates and improve query performance, you edit query script parameters.

Note:
Do not alter the query script parameters without first consulting others in your organization such as the DBA, network team, and SAP BusinessObjects administrative team.

1. Using the Motors universe, edit the END_SQL parameter to add the following value to the end of each SQL statement generated:

   `/*@variable('BOUSER')*/`

   a) Use the Motors universe and from the Data Foundation, choose the *Data Foundation* drawer.

   b) Choose *Properties* and choose *SQL Parameters*.

   c) In the list of parameters, locate the END_SQL parameter.

   d) In the *Value* column, enter the following value and then press Enter:

   ```
   /*@variable('BOUSER')*/
   ```

   This is a SQL comment that will place the login name of the SAP BusinessObjects user at the end of every SQL statement generated.

   e) To close the *Parameter* dialog, choose *OK*.

   f) Save the business layer and create a query.

   g) View the script and note the comment at the end of the SQL statement.

   When the query is created by another SAP BusinessObjects product that uses this universe, the actual user name appears where the *@variable('BOUSER')* appears now

2. Using the Motors universe, edit the JOIN_BY_SQL parameter to activate it and create a query with the Client Name, Sales Revenue, and Rental Revenue objects.

   View the resulting script and note any differences.

   a) Use the Motors universe and from the Business Layer, choose the business layer name (the pinwheel icon).

   b) From the *Universe Parameters*, choose *Properties* and choose *SQL Parameters*.

c) From the drop–down in the lower left corner, choose JOIN_BY_SQL and to add it to the list of existing parameters, choose *Add*.

d) In the *Value* column, from the drop–down option, change No to Yes.

e) To close the *Parameter* dialog, choose *OK*.

f) Save the business layer and create a query with the Client Name, Sales Revenue, and Rental Revenue objects.

g) View the script and note the single SQL statement (instead of two with the contexts).

3. Using the Motors universe, edit the ANSI92 and FILTER_IN_FROM parameters to activate them.

   Create a query with the Client Name and Client Country objects and put a Query Filter on the Country object, restricting it to one of the valid Country values. View the resulting script and note any differences.

   a) Use the Motors universe and from the Business Layer, choose the business layer name (the pinwheel icon).

   b) From the *Universe Parameters*, choose *Properties* and choose *SQL Parameters*.

   c) From the drop–down in the lower left corner, choose ANSI92 and to add it to the list of existing parameters, choose *Add*.

   d) In the *Value* column, from the drop–down option, change No to Yes.

   e) From the drop–down in the lower left corner, enter FILTER_IN_FROM and to add it to the list of existing parameters, choose *Add*.

   f) In the *Value* column, from the drop–down option, change No to Yes.

   g) To close the *Parameter* dialog, choose *OK*.

   h) Save the business layer and create a query with the Client Name object and put a Query Filter on the Country object, restricting it to one of the valid Country values.

   i) View the script and note that the query filter is now in the FROM clause and there is no WHERE clause.

Universe Design Best Practices

As with any project, always base your universe design and implementation on best practices. The following information outlines a general workflow of best practices to design an effective universe that meets end users' needs.

The following are general best practices:

- Identify reporting requirements.

  Involve users at every step of the universe design and production process, especially in naming objects (this ensures the terminology is correct).

- Identify the data source relevant to the universe you are creating and ensure that there is representative data available.

  Ensure that the data source is stable and not likely to change dramatically during universe development.

- Insert tables into the data foundation one at a time, not in bulk, understanding how each table relates to the overall universe.

- Insert joins and define cardinalities.

  Specify cardinalities manually as opposed to using the **Detect Cardinalities** option in the Information Design Tool. Always define cardinality, even for self-referencing and self-restricting joins, to avoid cardinality warning messages.

- Lay out your data foundation with cardinalities facing the same direction and always arrange tables logically in the structure window. Doing so helps you identify and visualize contexts and resolve loops and SQL Traps.

- Add relevant comments to your data foundation to document universe development changes. Remember that you may not be the only person building and designing this universe.

  Consider using shared projects when working with multiple universe designers.

- Build relevant objects. Keep the business layer business-focused when naming objects and folders.

- Do not normalize. Use Multi-Dimensional Modeling instead.

- Always check integrity.

- Use aggregate awareness, which speeds up query time by using special tables containing pre-calculated data. Do all aggregation in the data source rather than in the reporting document. This helps to expedite the query process.

- Define aggregate SQL functions on business layer measures. Fewer rows get returned from the database thus making reports smaller and the calculation of variables, ranks, and formatting rules quicker.

- Set the connection parameters to disconnect after each transaction to ensure that the SQL statement is terminated cleanly and that the are no treads left open to the database, which are inactive.

Note:
A number of databases are licensed on the number of concurrent connections, and if a connection is left open for a period of time, then end users could experience difficulties in accessing the database.

- Remove unnecessary lists of values, such as on dates and measure objects. Base lists of values on lookup tables.
- Add any other elements to the business layer such as filters and navigation paths.
- Test the universe by building queries in SAP BusinessObjects end-user querying tools such as WebIntelligence or Dashboards.

Best Practices for Folders and Objects

The following are best practices for folders and objects:

- Use the minimum possible objects in the Business Layer.

  500 objects is the recommended maximum. Too many objects may slow down the time required for a user to find objects.
- Create condition objects when possible.
- Always type a description for objects.

  Start the description with the object's full name. This is particularly helpful to users if the object's name is long and cannot be fully displayed in the report/query panel.
- Consider putting the full folder path and object name in the object's description.

  This may be a laborious process but one which is useful if end-users are working with a large universe. The description is shown in the report/query panel when a user points to an object.
- Dimension, measure, and attribute objects should be within their logical folders unless the measure objects are generic to the whole universe, in which case you may choose to create a general Measure folder.
- Format objects and measures within the Business Layer.

  Doing so prevents users having to spend valuable time formatting data from within the reporting tool every time they create a report.

Best Practices for Joins

The following are best practices for joins:

- Avoid building a data foundation with no joins between the tables. There are exceptions, such as using summary tables in the data foundation; however, in those instances ensure that incompatibilities are set using the **Aggregate Navigation** tool to avoid SQL errors.
- Optimize database performance by using shortcut joins.

  Shortcut joins provide an alternative path between two tables. Shortcut joins can improve query performance by using shorter paths and bypassing intermediate tables.

Note:
Placing shortcut joins in an existing Data Foundation may change the results returned by existing queries.

- Avoid outer joins where possible.

  Outer joins may have a negative effect on performance since more rows are returned. Additionally, some databases do not use indexes when outer joins are involved.

Note:
It is recommended that outer joins be placed at the end of the flow of data, otherwise outer join errors may occur. Potentially, this could cause the SQL to try to match on the equality of a NULL value, which it cannot do in some RDBMSs. Using aggregate aware and aliases may help if outer joins are necessary, but cannot be placed at the end of the path. If you do place outer joins in the middle of a table path, the subsequent joins in the path may also have to be made outer joins to avoid errors, which can have a significant negative impact on performance.

LESSON SUMMARY

You should now be able to:

- Use connection configuration parameters
- Use query script parameters
- Explain best practices for universe design

Learning Assessment

1. What do the connection pool mode options refer to?

   Choose the correct answer.

   - [ ] A How BusinessObjects products respond when database resources are not available.
   - [ ] B How long a universe designer or an end-user using the reporting tools remain connected to the underlying data source.
   - [ ] C How specific values add to JDBC driver properties.
   - [ ] D How the middleware driver is selected when creating the connection.

2. At query time, the query server uses the values it finds in a specific order. Arrange the following values in the correct order.

   Arrange these steps into the correct sequence.

   - [ ] The value in the data foundation, if it is set.
   - [ ] The default value.
   - [ ] The value in the business layer, if it is set.

3. As with any project, you should always base your universe design and implementation on what?

   Choose the correct answer.

   - [ ] A vague notions
   - [ ] B minimal effort
   - [ ] C some good ideas
   - [ ] D best practices

Learning Assessment - Answers

1. What do the connection pool mode options refer to?

   Choose the correct answer.

   - [ ] A How BusinessObjects products respond when database resources are not available.
   - [X] B How long a universe designer or an end-user using the reporting tools remain connected to the underlying data source.
   - [ ] C How specific values add to JDBC driver properties.
   - [ ] D How the middleware driver is selected when creating the connection.

2. At query time, the query server uses the values it finds in a specific order. Arrange the following values in the correct order.

   Arrange these steps into the correct sequence.

   - [2] The value in the data foundation, if it is set.
   - [3] The default value.
   - [1] The value in the business layer, if it is set.

3. As with any project, you should always base your universe design and implementation on what?

   Choose the correct answer.

   - [ ] A vague notions
   - [ ] B minimal effort
   - [ ] C some good ideas
   - [X] D best practices

UNIT 16 Universe Deployment and Security

UNIT OBJECTIVES

- Document universes
- Deploy a universe
- Secure a published universe

- Define data security profiles
- Define business security profiles
- Assign security profiles to users
- Describe security setting priorities
- Update a published universe

Unit 16
Lesson 1

Deploying a Universe

LESSON OVERVIEW
Once you have created a universe, you are ready to publish it. This lesson describes how to deploy a universe.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Document universes
- Deploy a universe

Resource Documentation

To create documentation for a resource, you can save any resource in a local project as a report in a local file. For larger resources (data foundations and business layers), you can select which metadata elements to include in the report.

Universe Deployment

Publication is the last step in the universe creation process. Using the Publish Universe Wizard, you publish a business layer to either your local file system or a repository. When you publish a business layer, the wizard exports the business layer and the resources it references (local connection, connection shortcuts, and data foundation), and creates a universe that is available to those who use query, reporting and analysis tools.

Only business layers built on local connections can be published locally. This can be a business layer based on a local OLAP connection, or a business layer based on a single-source data foundation with a local connection.

The published universe is saved in the local file system folder that you specify and is available only to users of that local machine.

Repository Publishing

To secure a universe, you must publish it first to a repository on a Central Management Server (CMS). The universe inherits the object-level security and user security rights defined for the CMS. The data and meta data in the universe are secured by defining security profiles in the information design tool Security Editor.

To browse and manage resources once they are published to a repository, use the Repository Resources View.

To publish a universe to a repository, the business layer must reference one or more secured connection shortcuts. All shortcuts must reference connections defined in the repository where the universe is to be published.

Note:
If the business layer references a local connection and you want to publish to a repository, first publish the connection and change the connection reference in the data foundation (relational), or in the business layer (OLAP) to use the connection shortcut.

Recommended Actions Before Publishing a Universe

- Save the business layer and all the resources it references.
- If the business layer references resources that are shared, synchronize the project to ensure that all changes are taken into account in the published universe.
- Check integrity of the business layer and, if applicable, the data foundation. The Publish Universe Wizard gives you the option to run a check integrity before publishing.

Unit 16
Exercise 32

Publish a Universe to a Repository

Business Example

Now that your universe is completed, you need to make it available to your business users. First, however, you need to add security to the universe so that not all users have access to all data the universe provides access to.

> Note:
> To complete the exercise, replace the *xx* with the group number the instructor provided you with at the beginning of the class.

1. Publish the Motors connection to the repository and add it as a shortcut to your project.
2. Change the connection used by the data foundation to the new connection shortcut.
3. Publish your Motors_xx universe to a repository.

Unit 16
Solution 32

Publish a Universe to a Repository

Business Example

Now that your universe is completed, you need to make it available to your business users. First, however, you need to add security to the universe so that not all users have access to all data the universe provides access to.

> Note:
> To complete the exercise, replace the *xx* with the group number the instructor provided you with at the beginning of the class.

1. Publish the Motors connection to the repository and add it as a shortcut to your project.
   a) In your project, right-click the Motors_XX.cnx connection and choose *Publish Connection to a Repository*.
   b) If necessary, log on to the SAP BusinessObjects BI Platform Repository.
   c) Choose *Next*.
   d) On the left side of the pop-up window, choose the Connections folder.
   e) Choose *Finish*.
   f) When prompted to create a connection shortcut in the same local folder, choose *Yes*.
2. Change the connection used by the data foundation to the new connection shortcut.
   a) If it is not already open, open the Motors data foundation.
   b) Choose the Connection drawer.
   c) Choose *Change Connection*.
   d) Choose the Motors_XX.cns connection shortcut.
   e) Choose *Finish*.
3. Publish your Motors_xx universe to a repository.
   a) In the *Local Project* view, choose the Motors business layer.
   b) Right-click the business layer and choose *Publish* → *To a Repository*.
   c) Choose *Next*.
   d) Choose the Universes folder, choose *Finish*, and *Yes* to overwrite.

LESSON SUMMARY
You should now be able to:

- Document universes
- Deploy a universe

Unit 16
Lesson 2

Securing a Published Universe

LESSON OVERVIEW
As well as deploying the universe to use throughout your organization. You also need to secure the universe so that users have appropriate access to the data provided through the universe. This lesson describes how to secure a universe so that business users can utilize it appropriately.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Secure a published universe

Published Universe Security

Universe security starts when the universe is published to a repository on a Central Management Server (CMS). Published universes are stored in the Universes folder and secured connections are stored in the Connections folder.

Two Levels of Security

First level of security

The SAP BusinessObjects administrator grants the right to access specific folders, resources, universes, and connections in the repository to specific users and groups.

Second level of security

Using the information design tool Security Editor, You can restrict the data that is returned in a query using query limits and controls, filters, and row restrictions. You can also grant or deny access to objects and views in the business layer. To create this level of security, you define security profiles for the universe and assign these profiles to users and groups.

Security Profiles

A security profile is a group of security settings that apply to a universe published in the repository. The settings control the data that is displayed and modify the parameters defined in the data foundation and/or business layer. Once the profile is assigned to a user or a group, the settings in the profile determine what objects, data, and connections the user sees when connecting to the universe.

Types of Security Profiles

- Data Security Profiles have security settings defined on objects in the data foundation and on data connections.
- Business Security Profiles have security settings defined on objects in the business layer.

Note:
All Data Security Profile settings apply to relational universes only. Business Security Profiles can apply to both relational and OLAP universes.

Multiple profiles can be defined for each universe. The profiles are saved in the repository.

A user of query and reporting tools who has been granted access to a universe using the Central Management Console (CMC) and who has no security profiles assigned or inherited, can see all the objects in the universe and all the data returned by those objects.

When you assign a profile to the user, the security settings defined in the profile are applied whenever the user runs a query on the universe.

You can assign more than one profile to a user or group. A user might be assigned a profile and also inherit profiles from groups. When more than one profile is assigned to a user, the profiles are aggregated to produce a single group of settings called the net profile.

Aggregation follows priority and restriction levels that you can modify in the Security Editor. You can also see which profiles a user or group inherits, and preview net profiles for a user or group.

Profiles are stored independently from the universe itself: changes in the data foundation or business layer of the universe do not affect the profiles when the universe is republished. Similarly, changes in a profile are independent of assignments, so you do not have to reassign a profile when it is modified. It remains assigned, including any changes.

If you republish a universe, run a check integrity on the universe to flag any discrepancies between the universe and its security profiles.

Profiles created for a universe are deleted when the universe is deleted.

LESSON SUMMARY
You should now be able to:

- Secure a published universe

Unit 16
Lesson 3

Creating Data Security Profiles

LESSON OVERVIEW
Data Security Profiles have security settings defined on objects in the data foundation and on data connections. This lesson describes how to create data security profiles.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Define data security profiles

Data Security Profiles

Data Security Profile is a group of settings that define security on a published universe using objects in the data foundation and the data connections. Data Security Profiles can be defined for relational universes only.

Data Security Profile Connection Settings

Use the Data Security Profile Connections setting to define replacement connections that can override the connections defined in the universe. Once a user is assigned or inherits a profile that contains a replacement connection, when the user runs a query on the universe, the replacement connection is used instead of the connection defined in the universe.

Only secured connections can be defined as replacement connections. Relational connections fall into one of three types, listed below. The replacement connection must be of the same type as the original connection.

- SAP BW relational databases
- SAS relational databases
- Other relational databases

You can select a connection in the Connections folder and subfolders for which you have the "View objects" right granted for the repository where you are defining the security profiles. For multisource universes that rely on multiple connections, you can define a replacement for each connection.

Data Security Profile Controls Settings

The following table lists data security profile controls settings:

Table 25: Data Security Profile Controls Settings

Query Limit	Possible values
Limit size of result set to	• True and a numerical size between 0 and 2147483647 rows• False

Limit execution time to	• True and a numerical size between 0 and 2147483647 minutes• False
Warn if cost estimate exceeds	• True and a numerical size between 0 and 10000 minutes• False

Use the Data Security Profile Controls settings to define replacement query limits to override default limits when retrieving data from the database. Default query limits are set by the universe designer in the business layer. Once a user is assigned or inherits a profile with replacement Controls settings, when the user runs a query, the replacement limits are used instead of the limits defined in the business layer properties.

In the editor for Data Security Profiles, the limits selected and the limit values defined in the business layer are displayed. When you select or deselect a limit, or enter a new value for a limit, the label appearance changes to bold. This bolding shows that the limit is an override and not the default limit defined in the universe.

Table 26: Data Security Profile Controls Settings

Query Option	Possible values
Allow use of subqueries	• True• False
Allow use of union, intersect and minus operators	• True• False
Allow complex operands in Query Panel	• True• False
Multiple SQL statements for each context	• True• False
Multiple SQL statements for each measure	• True• False
Allow Cartesian products	• Warn• Prevent

Data Security Profile SQL Settings

Use the Data Security Profile SQL settings to define replacement query options. The universe designer defines default query options in the business layer and data foundation properties. Once a user is assigned or inherits a profile with SQL settings, when the user uses the query panel, the replacement options are used instead of the query options defined in the universe.

In the editor for Data Security Profiles, the SQL settings selected in the business layer and data foundation are displayed. When you select or deselect an option, the label appearance changes to bold. This bolding shows that the option is an override and not the default defined for the universe.

Data Security Profile Rows Settings

Use Data Security Profile Rows settings to restrict the rows returned in a query. You restrict the rows by defining an SQL WHERE clause for a specified table. Once a user is assigned or inherits a profile with a Rows setting, when the user runs a query on the universe, the defined WHERE clause is added to the SQL generated if the table is referenced in the query.

Note:
A user who has the right to edit the generated SQL in the reporting tool can change the WHERE clause generated by the Rows setting. Remember to manage the user's rights in the reporting tool to prevent the user from modifying the SQL.

You can define the WHERE clause for any standard table in the data foundation. The SQL for the WHERE clause can include:

- @Functions such as @Variable and @Prompt
- For multisource enabled universes, references to other tables in any connection defined for the universe
- For multisource-enabled universes, SAP BusinessObjects SQL functions

The SQL for the WHERE clause cannot include:

- Calculated columns
- Derived tables

Data Security Profile Tables Settings

Use the Data Security Profile Tables setting to define replacement tables. Once a user is assigned or inherits a profile with a Tables setting, when the user runs a query that references the original table, the replacement table is used instead.

Note:
A user who has the right to edit the generated SQL in the reporting tool can change the replacement table name. Remember to manage the user's rights in the reporting tool to prevent the user from modifying the SQL.

You can replace a standard table in the data foundation with a database table in any connection defined for the universe, or another standard table in the data foundation.

Note:
If you want to specify an owner and qualifier for the replacement table, you must enter these in the fields provided, and not as part of the table name.

Unit 16
Exercise 33

Define Data Security Profiles

Business Example

To limit users to only be able to access certain data based on row settings, create a data security profile.

1. To limit data from the Showroom folder to Prestige Motors, create a Data Security Profile.

Unit 16
Solution 33

Define Data Security Profiles

Business Example

To limit users to only be able to access certain data based on row settings, create a data security profile.

1. To limit data from the Showroom folder to Prestige Motors, create a Data Security Profile.
   a) From the menu bar, choose *Window* → *Security Editor*, and enter the CMS credentials your instructor assigned to you.
   b) Navigate in the *Universe/Profiles* tree and choose the Motors universe you have published.
   c) Right-click the Motors universe and from the context menu, choose *Insert Data Security Profile*.
   d) Rename this Data Security Profile to **`Prestige Motors Showroom`**.
   e) To create a row restriction security, choose the Rows tab, and choose *Insert*.
   f) Choose the browse button for the table and choose the Showroom table.
   g) Choose *OK*.
   h) Choose the browse button for the WHERE clause and create the following WHERE clause and choose *OK*: `SHOWROOM.SHOWROOM_NAME = 'Prestige Motors'`
   i) To close the *Define Row Restriction* dialog, choose *OK* and to save the Prestige Motors Showroom profile, choose *OK* again.

LESSON SUMMARY

You should now be able to:

- Define data security profiles

Unit 16
Lesson 4

Creating Business Security Profiles

LESSON OVERVIEW
Security profiles are defined on published universes and stored in the repository. You define security profiles on universes published to a repository using the Security Editor. This lesson describes how to define business security profiles.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Define business security profiles

Business Security Profiles

Business Security Profile is a group of settings that define security on a published universe using objects in the business layer.

In the business layer, designers set the status of objects to Active, Hidden, or Deprecated. When defining profile settings, you have access to all active objects in the business layer. Objects that are hidden or deprecated in the business layer never appear in the query panel or on reports.

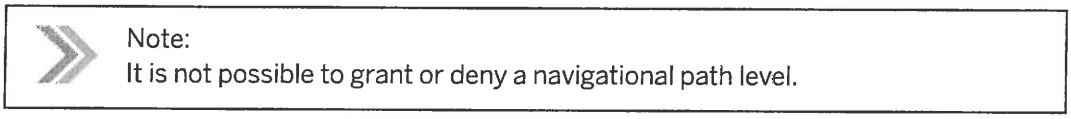
This setting defines the universe views and business layer objects available to the user in the query panel. By default, a user with access to the universe granted in the repository can see all universe objects in the query panel. Once the user is assigned or inherits a profile with a Create Query setting, only the views and objects granted by the setting are displayed and can be selected for a query.

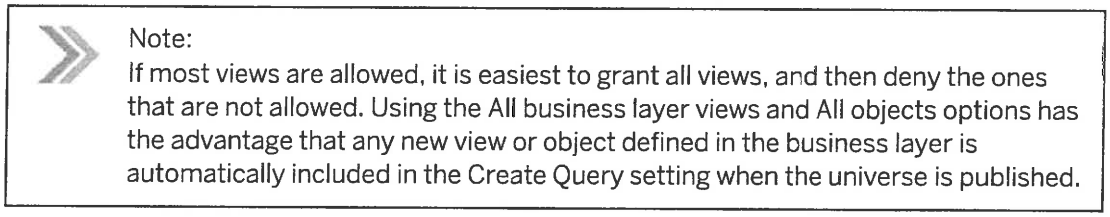
If an object is not granted and not denied explicitly, it is denied by default. Unlike objects that are explicitly denied, objects that are denied by default could be granted by inheritance after aggregating Business Security Profiles to determine the net profile for a user.

Methods to Grant and Deny Objects

- By business layer view: grant or deny all views defined for the universe.
- By object: grant or deny all objects in the business layer.
  - Dimensions
  - Attributes
  - Measures
  - Calculated members
  - Filters
  - Prompts

- Named sets
- Folders: Grants or denies all objects in the folder.
- Analysis dimensions: Grants or denies all objects in the dimension.
- Hierarchies: Grants or denies all objects in the hierarchy.





The objects in a granted view are granted in that view only. If the same object is contained in another view, it is not automatically granted.

Whether or not a user sees a particular object in the query panel is determined after aggregating the Create Query settings in all profiles assigned to the user, and taking into consideration object access level.

Business Security Profile Display Data Settings

Using the *All objects* option has the advantage that any new object defined in the business layer is automatically included in the display data setting when the universe is published.

A user who is denied an object by a Display Data setting might refresh a report containing the denied object. You can specify what the refresh should do in this case by setting the SQL generation parameter AUTO_UPDATE_QUERY in the business layer.

- If this parameter is set to No, then refreshing the report generates an error message.
- If this parameter is set to Yes, then the denied objects are removed from the query and from any filters defined in the business layer. Data for other granted objects is retrieved and displayed to the user in a partial report.

Whether or not a user sees data for a particular object is determined after aggregating the Display Data settings in all profiles assigned to the user, and taking into consideration object access level.

Business Security Profile Filters Settings

Use the Business Security Profile Filters setting to define a filter using objects in the business layer or named member sets. You create and edit filters explicitly for the Business Security Profile using the Security Editor. Filters in the Business Security Profile are not accessible in the business layer. If the Business Security Profile is deleted, the filter or named set is also deleted.

Once the user is assigned or inherits a profile with a Filters setting, the filter is added to the query script (and therefore combined with any filters defined in the business layer) to restrict the data displayed.

Relational Universes

For relational universes, you define filters on dimensions and measures in the business layer. You can define compound filters that are linked by the AND and OR operators. You can also define multiple filters to apply to the query.

When a user runs a query, the filters are always applied to the query and to the returned data. This is different from the Data Security Profile Rows setting, which only applies if a defined table is referenced in the query.

OLAP Universes

For OLAP universes, you define a named set of members. You can include or exclude members for any dimension in the business layer. The excluded members are removed from the query when data is retrieved from the cube.

Note:
The filter does not impact the aggregation of values in the report. Only the display of members is filtered. You can include or exclude members from multiple dimensions.

You can also define multiple named sets to apply to the query.

Unit 16
Exercise 34

Define Business Security Profiles

Business Example

To limit users to only be able to access certain data based on filter settings, create a business security profile.

1. Create a Business Security Profile that hides all Rentals-related objects.

Unit 16
Solution 34

Define Business Security Profiles

Business Example

To limit users to only be able to access certain data based on filter settings, create a business security profile.

1. Create a Business Security Profile that hides all Rentals-related objects.
   a) Navigate in the *Universe/Profiles* tree and choose the Motors universe you have published.
   b) Right-click the Motors universe and from the context menu choose *Insert Business Security Profile*.
   c) Rename this business security profile to **`Sales Only`**.
   d) To define which objects a user can use to retrieve data, choose the *Create Query* tab.
   e) For the Business Layer Views, choose *Insert Granted*.

> Caution:
> There are two *Insert Granted* buttons. Choose the one in the middle, not the one at the bottom.

   f) Choose the *Master View*, and choose *OK*.
   g) For the objects, choose *Insert Denied*.

> Caution:
> There are two *Insert Denied* buttons. Choose the one in the middle, not the one at the bottom.

   h) Choose the Rentals folder as well as any other rental-related objects from other folders such as the Day Rental Charges subfolder of the Car folder, and choose *OK*.
   Use the Ctrl key to choose multiple objects.
   i) To confirm that the Rentals folder and any other rental-related objects you define appear as Denied, review the *Status* column.
   j) In the *Display Data* tab, choose *Insert Granted* and choose "all objects".
   k) To save the Business Security Profile, choose *OK*.

LESSON SUMMARY

You should now be able to:

- Define business security profiles

Unit 16
Lesson 5

Assigning Security Profiles to Users

LESSON OVERVIEW
The Security Editor is also used to assign profiles to users and groups. This lesson describes assigning Security Profiles to Users.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Assign security profiles to users

User Profile Assignment

More than one Data Security Profile or Business Security Profile defined for a universe may be assigned to the same user. Multiple profiles can be directly assigned to a user or group, and inherited from parent groups. When this happens, the security settings in the different profiles are aggregated to result in one effective Data Security Profile, and one effective Business Security Profile, called net profiles. The settings in the net profiles are applied when the user creates a query or views a report.

Methods for Aggregating Security Settings

Priority

Used to aggregate Data Security Profile settings. You can prioritize the Data Security Profiles in the Security Editor.

Restriction

Some Data Security Profile settings and all Business Security Profile settings are aggregated based on restriction level: very restrictive, moderately restrictive, or less restrictive. The restriction level defines which operators to use to aggregate profiles. Different aggregation operators are used depending on whether the profile is inherited or merged.

Inherited and Merged Profiles

- If the user or group is assigned Profile A and belongs to a group that is assigned Profile B, Profile A and Profile B are inherited.
- If the user or group belongs to a group assigned Profile A and another group assigned Profile B, Profile A and Profile B are merged.
- If the user or group is assigned both Profile A and Profile B, Profile A and Profile B are merged.

Restriction Levels and Profile Aggregation

You can change these restriction levels in the Security Editor to affect how the profiles are aggregated.

- The less restrictive level is appropriate when security is designed with roles, each role granting new rights to the user.
- The most restrictive level is appropriate when each profile is used to restrict what the user can see.
- The moderately restrictive level uses the most restrictive level for inherited profiles and the less restrictive level for merged profiles.

Note:
The Data Security Profile Rows setting and Business Security Profile Filters setting both generate a WHERE clause to filter the query. The Rows setting is applied first. The WHERE clause in the Filters setting is then applied to the results of the first query. In effect, the two WHERE clauses are aggregated with the AND operator.

The operations used to aggregate profile settings (for example AND, OR) vary for the different settings.

Hint:
For more information about aggregation for each type of setting, see the Information Design Tool User Guide.

Security Profile Creation and Editing

Use the *Security Editor* to create and edit security profiles, and assign profiles to users and groups.

Note:
You can create security for .unx universes only.

The session name displays on the tab of the Security Editor. If the session name is prefixed by an asterisk (*), it means that you have changed the security profiles or assignments in the Security Editor that have not yet been saved in the repository.

The *Security Editor* can be viewed in two ways: by universe, or by users/groups. Select the tab on the left-hand side of the *Security Editor* to display the view you want to work with.

- The *Universes / Profiles* tab lets you to do tasks by first selecting a universe in the repository.
- The *Users / Groups* tab lets you to do tasks by first selecting a user or group.

Unit 16
Exercise 35

Assign Security Profiles to Users

Business Example

After security profiles have been created, you assign them to users based on the data they need to access for their role.

1. Assign the Data and Business Security Profiles to yourself.

Unit 16
Solution 35

Assign Security Profiles to Users

Business Example

After security profiles have been created, you assign them to users based on the data they need to access for their role.

1. Assign the Data and Business Security Profiles to yourself.
   a) In the *Universes/Profiles* list, choose the newly created data security profile.
   b) In the *Groups/Users* list (on the right), expand the *Administrators* group and choose your username.
   c) To add the user to the *Assigned Users* column, choose the active arrow.
   d) From the *File* menu choose *Save*.

Unit 16
Exercise 36

View Assigned Security Profiles

Business Example

Sometimes you need to review the security profiles a user has been assigned.

1. Test that the restrictions are applied to your user.

Unit 16
Solution 36

View Assigned Security Profiles

Business Example

Sometimes you need to review the security profiles a user has been assigned.

1. Test that the restrictions are applied to your user.
   a) In the *Universes/Profiles* list, right-click the Motors.unx file and choose *Run Query*.
   b) Confirm that you do not see the Rentals or Day Rental Charges folders.
   c) To the *Result Objects* pane, add the Showroom objects from the Showroom folder, and add the Sales Revenue object from the *Sales* → *Sales Figures* subfolder.
   d) In the *Data Preview* pane, choose *Refresh*.
   Note that the results display data only for the Prestige Motors.

LESSON SUMMARY

You should now be able to:

- Assign security profiles to users

Unit 16
Lesson 6

Identifying the Priority of Security Settings

LESSON OVERVIEW
Universe designers identify the priority on which security settings set on user profiles are processed. This lesson discusses security aggregation.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Describe security setting priorities

Security Aggregation

Aggregation of Connection Settings

If more than one Data Security Profile for a universe is assigned to or inherited by the same user, the connection defined in the Data Security Profile with the highest priority is used. If the universe has multiple connections, aggregation of the Connections setting is done for each connection independently.

Table 27: Aggregation of Controls Settings

Restriction Level	Aggregation Rule
Very Restrictive	The limit is active only if it is selected in all merged and inherited profiles. The value used is the minimum value for the limit among all the merged and inherited profiles.
Moderately Restrictive	The limit is active only if it is selected in all inherited profiles and selected in at least one merged profile. First, the minimum value is determined for the limit comparing the inherited profiles. This value is compared to the values among the merged profiles. The value used is the maximum among these values.
Less Restrictive	The limit is active if it is selected in any merged or inherited profile. The value used is the maximum value for the limit among all the merged and inherited profiles.
Priority (default)	The activation and value of the limit in the Data Security Profile with the highest priority is used.

If more than one Data Security Profile for a universe is assigned to or inherited by the same user, the rules in the preceding table are used to aggregate the Controls settings. The rules are applied to each query limit to determine the value to be used when the user runs a query or report.

Table 28: Aggregation of SQL Settings

Restriction Level	Aggregation Rule
Very Restrictive	The option is active only if it is selected in all merged and inherited profiles. For Cartesian products, Prevent is used only if the value is Prevent in all merged and inherited profiles.
Moderately Restrictive	The option is active if it is selected in all inherited profiles and selected in at least one assigned profile. For Cartesian products, Prevent is used if the value is Prevent in all inherited profiles and Prevent in at least one merged profile.
Less Restrictive	The option is active if it is selected in any merged or inherited profile. For Cartesian products, Warn is used if the value is Warn in any merged or inherited profile.
Priority (default)	The activation and value of the option in the Data Security Profile with the highest priority is used.

If more than one Data Security Profile for a universe is assigned to or inherited by the same user, the rules listed above are used to aggregate the SQL settings. The rules are applied to each query option to determine the value to be used when the user creates a query.

Aggregation of Rows Settings

If more than one Data Security Profile for a universe is assigned to or inherited by the same user, the following rules are used to aggregate the Rows settings and determine the WHERE clause to be used when the user runs a query or report.

Table 29: Aggregation of Rows Settings

Restriction Level	Aggregation Rule
Very Restrictive (default)	The WHERE clauses in all profiles that apply to the same table are combined with the AND operator.
Moderately Restrictive	Inherited WHERE clauses are aggregated using the AND operator. Merged WHERE clauses are aggregated using the OR operator.
Less Restrictive	The WHERE clauses in all profiles that apply to the same table are combined with the OR operator.

First, the WHERE clauses for each table are aggregated according to the restriction level. After aggregation according to restriction level, the WHERE clauses for each table are aggregated together with the AND operator to produce the final WHERE clause that is applied to the query.

> Note:
> For a definition of inherited and merged profiles, see the related topic on security profile aggregation.

Aggregation of Tables Settings

If more than one Data Security Profile for a universe is assigned to or inherited by the same user, the replacement table defined in the Data Security Profile with the highest priority is used. If settings are defined for multiple tables, the aggregation is done for each table independently.

Aggregation of Create Query Settings

If more than one Business Security Profile for a universe is assigned to or inherited by the same user, the Create Query settings are aggregated. Object access levels, if they are defined, are applied to determine whether or not a user sees a particular object in the query panel.

First, the list of views that the user can select in the query panel is determined by aggregating the profiles according to the restriction level:

Table 30: Aggregation of Create Query Settings - Views

Restriction Level	Aggregation Rule
Very Restrictive (default)	The user can select the view in the query panel only if it is granted in all inherited and merged profiles.
Moderately Restrictive	The user can select the view in the query panel only if it is granted in all inherited profiles and granted in at least one merged profile.
Less Restrictive	The user can select the view in the query panel if it is granted in any inherited or merged profile.

Once a view is selected in the query panel, an object appears if it is included in the view, and if it is not explicitly denied after aggregating the profiles according to restriction level.

Table 31: Aggregation of Create Query Settings - Object

Restriction Level	Aggregation Rule
Very Restrictive (default)	The object is denied if it is explicitly denied in any inherited or merged profile.
Moderately Restrictive	The object is denied if it is explicitly denied in any inherited profile and denied in all merged profiles.
Less Restrictive	The object is denied only if it is explicitly denied in all inherited and merged profiles.

After aggregation, the denied objects are not displayed even if they belong to a granted view. If a folder is denied, then all sub-folders and objects in the folder are denied.

Finally, the access level granted to the user in the Central Management Console determines which of the objects granted by the net Business Security Profile are available in the query panel. The user sees only the objects with an access level lower than or equal to this authorized access level. You assign access levels to objects in the business layer editor.

Aggregation of Display Data Settings

If more than one Business Security Profile for a universe is assigned to or inherited by the same user, the Display Data settings are aggregated. Object access levels, if they are defined, are applied to determine whether or not a user sees the data for an object in the business layer.

First, the list of objects that the user can see data for is determined by aggregating the profiles according to restriction level.

Table 32: Aggregation of Display Data Settings

Restriction Level	Aggregation Rule
Very Restrictive (default)	The data appears only if it is granted in all inherited and merged profiles.
Moderately Restrictive	The data appears only if the object is granted in all inherited profiles and granted in at least one merged profile.
Less Restrictive	The data appears if the object is granted in any inherited or merged profile.

If a folder is denied, then the data for all objects in the folder and its sub folders is denied.

Finally, the access level granted to the user in the Central Management Console determines which of the objects granted by the net Business Security Profile the user sees data for. The user sees data only for the objects with an access level lower than or equal to his authorized access level. You assign access levels to objects in the business layer editor.

Aggregation of Filter Settings

If more than one Business Security Profile for a universe is assigned to or inherited by the same user, the following rules are used to aggregate the filters settings and determine the filter to be added to the query script when the user runs a query or report.

For relational universes, the filters are aggregated according to the restriction level. The resulting filter is added to the WHERE clause applied to the query.

Table 33: Aggregation of Filters Settings - Relational

Restriction Level	Aggregation Rule
Very Restrictive (default)	The filters in all profiles are combined using the AND operator.
Moderately Restrictive	Inherited filters are aggregated using the AND operator. Merged filers are aggregated using the OR operator.
Less Restrictive	The filters in all profiles are combined using the OR operator.

For OLAP universes, the named sets are aggregated according to the restriction level.

Table 34: Aggregation of Filters Settings - OLAP

Restriction Level	Aggregation Rule
Very Restrictive (default)	The user sees a member only if it is included in every named set defined in all profiles.
Moderately Restrictive	The user sees a member if it is included in every named set defined in the inherited profiles, and included in at least one named set defined in the merged profiles.

Less Restrictive	The user sees a member if it is included in any named set defined in any profile.

Example: Restrictive (AND), Permissive (OR)

If R(X) is the function that returns the value set for the parameter by the security profile assigned to the user or group X, and AND represents the restrictive operator, whereas OR is the permissive operator, then, the aggregated security is:

- R(U) AND R(G_1) AND R(G_2) if you have selected very restrictive (AND).
- R(U) AND R(G_1) OR R(G_2) if you have selected moderately restrictive (ANDOR).
- R(U) OR R(G_1) OR R(G_2) if you have selected very restrictive (OR).

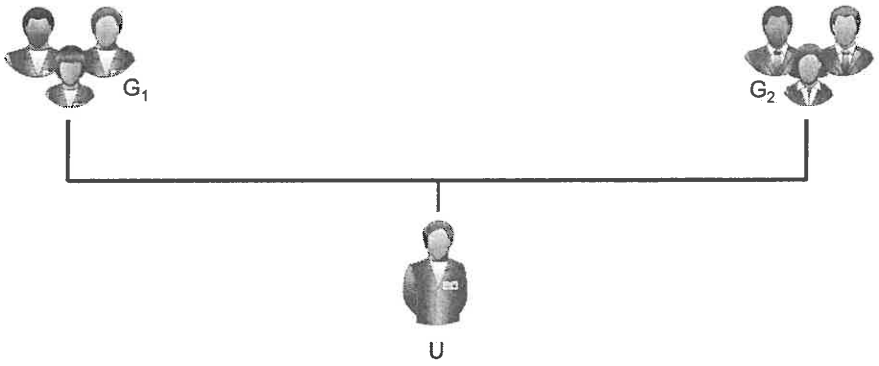


Figure 63: Inheritance Example: Restrictive (AND), Permissive (OR)

Example: Moderately Restrictive (ANDOR)

If you have selected moderately restrictive (ANDOR), then the resulting security is aggregated into the following:

- R(U) AND [R(G_1) AND (R(G_{11}) OR (R(G_{12}))) OR R(G_2) AND R(G_{21}))]

Very restrictive (AND) (respectively, less restrictive (OR)) aggregation is computed by using the AND (respectively, OR) operator only.

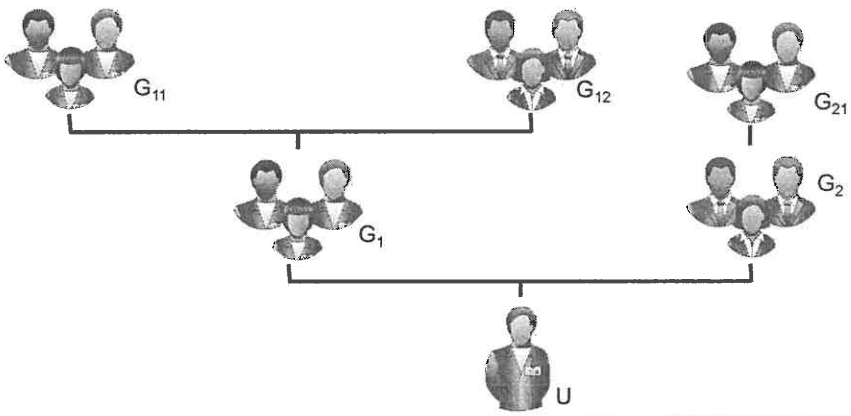


Figure 64: Inheritance Example: Moderately Restrictive (ANDOR)

LESSON SUMMARY

You should now be able to:

- Describe security setting priorities

Unit 16
Lesson 7

Updating a Published Universe

LESSON OVERVIEW
You use the Universal Retrieval Wizard to retrieve a published universe. This lesson describes how to update a published universe.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Update a published universe

Universe Retrieval Wizard

To retrieve a published universe, you must have a project in the Local Projects View where the business layer and referenced resources are to be saved.

1. Start the Retrieve a Published Universe Wizard:
   - To retrieve a published universe from a repository, select the project in the Local Projects View, right-click the project, and select *Retrieve Universe → from a Repository*. You can also retrieve a universe from the repository in the Repository Resources View. Right-click the universe and select *Retrieve Universe*.

Note:
By default, the resources are retrieved into the local project and are secured locally by requiring you to enter the CMS authentication when opening a retrieved data foundation or business layer. To remove the local security requirement, select the Save for all users check box when selecting the universe in the repository.

   - To retrieve a published universe from a local folder, select the project in the Local Projects View, right-click the project, and select *Retrieve Universe → from Local Folder*.
2. Follow the instructions on the wizard pages. For more information about what to do on a particular page, click the help icon. When the wizard finishes, the business layer and its dependent resources (connections, connection shortcuts, data foundation) are created in the local project and are ready to be edited.

Resource Dependencies

Making changes to a resource, such as deleting it from a local project, moving it to another local project, or updating it, may impact other resources that depend on it. You are warned of the impact before you delete or move a resource.

To help you understand the impact of changes and plan your work, commands exist that show the dependencies between resources and their objects.

Table 35: Show Dependencies Commands

Command	Dependency
Show Local Dependencies	Shows the dependencies between resources in a local project.
	Shows the local resources that depend on a particular table, column, or business layer object in the data foundation and business layer editors.
Show Repository Dependencies	Lists the universes published in a particular repository that are referenced by the selected data foundation or business layer.

LESSON SUMMARY

You should now be able to:

- Update a published universe

Learning Assessment

1. You can save any resource in a local project as a report in a local file to create documentation for that resource.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

2. What are the two types of security profiles?

   Choose the correct answers.

   - [ ] A Universe Security Profile
   - [ ] B Aggregate Security Profile
   - [ ] C Data Security Profile
   - [ ] D Business Security Profiles

3. Once a user is assigned or inherits a profile that contains a replacement connection, when the user runs a query on the universe, the replacement connection is used instead of the connection defined in the universe.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

4. What profile is described as a group of settings that define security on a published universe using objects in the business layer?

   Choose the correct answer.

   - [ ] A Business Security Profile
   - [ ] B Display Data Profile
   - [ ] C Folder Display Profile
   - [ ] D Master View Profile

5. What do you use to create and edit security profiles, and assign profiles to users and groups?

   Choose the correct answer.

   - ☐ A Information Editor
   - ☐ B Security Editor
   - ☐ C Profile Editor
   - ☐ D Query Editor

6. If more than one Data Security Profile for a universe is assigned to or inherited by the same user, the connection defined in the Data Security Profile with the lowest priority is used.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

7. To retrieve a published universe, you must have a project in the Local Projects View where the business layer and referenced resources are to be saved.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

Learning Assessment - Answers

1. You can save any resource in a local project as a report in a local file to create documentation for that resource.

   Determine whether this statement is true or false.

   [X] True

   [ ] False

2. What are the two types of security profiles?

   Choose the correct answers.

   [ ] A Universe Security Profile

   [ ] B Aggregate Security Profile

   [X] C Data Security Profile

   [X] D Business Security Profiles

3. Once a user is assigned or inherits a profile that contains a replacement connection, when the user runs a query on the universe, the replacement connection is used instead of the connection defined in the universe.

   Determine whether this statement is true or false.

   [X] True

   [ ] False

4. What profile is described as a group of settings that define security on a published universe using objects in the business layer?

   Choose the correct answer.

   - [X] A Business Security Profile
   - [ ] B Display Data Profile
   - [ ] C Folder Display Profile
   - [ ] D Master View Profile

5. What do you use to create and edit security profiles, and assign profiles to users and groups?

   Choose the correct answer.

   - [ ] A Information Editor
   - [X] B Security Editor
   - [ ] C Profile Editor
   - [ ] D Query Editor

6. If more than one Data Security Profile for a universe is assigned to or inherited by the same user, the connection defined in the Data Security Profile with the lowest priority is used.

   Determine whether this statement is true or false.

   - [ ] True
   - [X] False

7. To retrieve a published universe, you must have a project in the Local Projects View where the business layer and referenced resources are to be saved.

   Determine whether this statement is true or false.

   - [X] True
   - [ ] False

UNIT 17 SQL Clause Processing Problems

UNIT OBJECTIVES

- Analyze SQL traps
- Identify chasm traps
- Resolve chasm traps
- Identify fan traps
- Resolve fan traps

Unit 17
Lesson 1

Determining How the Order of SQL Clauses Affects Data Returned

LESSON OVERVIEW
When working with relational data sources at the SQL level it is common to encounter SQL traps: chasm and fan. SQL traps are caused by the way in which the parts of a select statement are combined when referencing a relational database. They can cause queries to return inaccurate results. The Information Design Tool provides ways of resolving these traps. This lesson describes how to analyze SQL traps.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Analyze SQL traps

SQL Traps

SQL traps are an inherent problem in SQL These traps are caused by the order in which the elements of the *SELECT* statement are processed.

Types of SQL Traps

Chasm trap

A chasm trap is a type of join path between three tables when two many-to-one joins converge on a single table, and there is no context in place that separates the converging join paths.

Fan trap

A fan trap occurs when a one-to-many join links a table, which is in turn linked by another one-to-many join.

In SQL, a *SELECT* statement processes the *SELECT, FROM, and WHERE* clauses first (with the exception of any aggregates). This process creates a product of all the tables in the FROM clause based on the joins and restrictions specified in the WHERE clause. This can be thought of as a virtual table. Problems can occur if an aggregate is applied. This results in the wrong output being generated. This is problematic because SQL does not produce an error message, it just projects the results.

Unlike loops that return fewer rows than expected, chasm traps and fan traps return too many rows.

LESSON SUMMARY
You should now be able to:

- Analyze SQL traps

Unit 17
Lesson 2

Detecting Chasm Traps

LESSON OVERVIEW
Working with relational data sources at the SQL level, you might encounter chasm traps. This lesson explains how to detect chasm traps in the universe structure.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Identify chasm traps

Chasm Trap Detection

Unlike loops, chasm traps are not detected automatically by the Information Design Tool. However, you can detect them in one of the following ways:

- Analyze the one-to-many (1-N) join paths in your schema to detect chasm traps graphically.
- Choose *Tools* → *Detect Contexts* or choose the *Detect Contexts* button to automatically detect and propose candidate contexts in your schema.

  Detect Contexts examines the many-to-one (N–1) joins in the schema and proposes contexts to separate the queries run on the table. This is the most effective way to ensure that your schema does not have a chasm trap.
- Add additional dimension or detail objects to display more information in the report. If there is a chasm trap, aggregated values are multiplied, alerting you to the problem.

  You can use Detect Contexts to detect and propose candidate contexts, and then examine the table where any two contexts diverge. The point where two contexts intersect is the source of a chasm trap.

  Any two tables that have multiple rows converging to a single row in the table with the "one" relationship can potentially cause a chasm trap.

In a chasm trap, there is a many-to-one-to-many relationship between three tables n the universe structure. For example, in this diagram there is no loop, but the flow around the three tables is many–to–one–to–many.

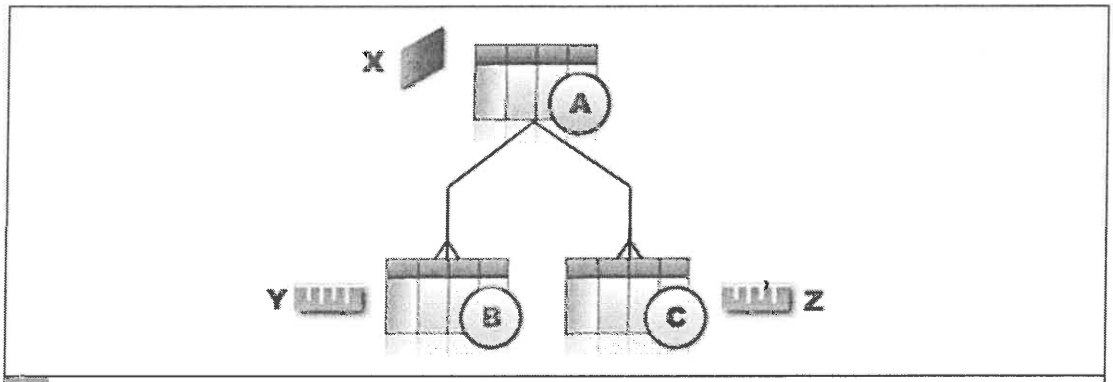


Figure 65: Many-to-One-to-many Join Flow

Note:
A chasm trap is not dependent on the object types. The query could be made up of only dimensions, only details, or only measures, or any combination of the three types with the "many" tables for a chasm to occur.

When a query that uses objects Y and Z is run, the inferred SQL includes tables B, A, and C that have a "many-to-one-to-many" relationship respectively. The chasm trap causes a query to return every possible combination of rows for one measure with every possible combination of rows for the other measure. This results in the values for each object being multiplied by the other. The effect is similar to a Cartesian product, but is known as a chasm trap.

The chasm trap is resolved by executing separate *SELECT* statements for object Y and object Z.

LESSON SUMMARY
You should now be able to:

- Identify chasm traps

Unit 17
Lesson 3

Resolving Chasm Traps

LESSON OVERVIEW
Upon detecteding chasm traps in the universe structure, this lesson explains how to resolve any chasm traps that may be present.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Resolve chasm traps

Chasm Trap Resolution

To resolve a chasm trap, you make two separate queries and then combine the results. Depending on the type of objects defined for the fact tables and the type of end user environment, you can use the following methods to resolve a chasm trap:

- Ensure that the Query Options in the Business Layer generate separate SQL statements for each measure.

  This method is not recommended as it only works with measures and results in certain inefficiencies in processing. It does not generate separate queries for dimension or detail objects.

- Create a context for each fact table.

  This solution works in all cases and does not result in inefficiencies.

Multiple SQL Statements Per Measure Method

If you have only measure objects defined for both fact tables, then you can use the Query Option setting of the Business Layer Multiple SQL statements for each measure. This forces the generation of separate SQL queries for each measure that is used in the query.

With the option Multiple SQL statements for each measure selected, Universe Designer now makes separate SQL SELECT statements for each measure object in the query.

The results in the report are now correct, as the query has automatically generated two SQL statements.

Using this option resolves the chasm trap problem. However, there are drawbacks to using this method to resolve chasm traps.

Multiple SQL Statements Per Measure Method Drawbacks

The Results Can Be Confusing

The Query Option specifies: "Multiple SQL statements for each measure". One drawback is that it does not run separate SELECT statements if the query contains only dimension objects.

The resulting report contains a single block with the results displayed as a Cartesian product.

Client Name	Rental date	Sale Date
Brent, Paul	4/22/03	4/10/03
Brent, Paul	4/22/03	1/10/04
Brent, Paul	9/10/04	4/10/03
Brent, Paul	9/10/04	1/10/04

The Report contains a single block with the results display as a Cartesian product – unclear for Users.

Figure 66: Cartesian Product-Like Results

It is not that there is anything inaccurate about the dates, but the multiple occurrences are confusing to users.

The Query is Inefficient

Another drawback is that any query including multiple measures infers a separate SELECT statement for each measure, whether it is required or not. To find a complete solution to chasm traps, you use contexts.

Contexts Method

You can define a context for each table at the many end of the joins. In our classroom example, you could define a context from Client to Sale and from Client to Rental.

When you run a query that includes objects from both contexts, this creates two SELECT statements that are synchronized at runtime in SAP BusinessObjects end user query tools to prevent the creation of a Cartesian product.

Creating contexts always solves a chasm trap in a universe. When you have a "many–to–one–to–many" situation, always use a context.

Unit 17
Exercise 37

Use Contexts to Resolve a Chasm Trap

Business Example

You need to report on client sales and rentals. Due to the structure of the database this will create a chasm trap, which will need to be resolved to return accurate results.

Note:
To complete the exercises, replace the *xx* with the group number the instructor provided you with at the beginning of the class.

1. Using the information in the table, create a new project and data foundation called Chasm_xx, where "xx"" stands for your user number. Use the Motors connection.

Field	Value
User Name	Train-xx
Password	Assigned password
Authentication	Enterprise

2. Add the following tables to the Data Foundation:
   - CLIENT
   - SALE
   - RENTAL (as an alias of the SALE table)
3. Create the following joins and set the cardinalities:

Join	Cardinality
CLIENT.CLIENT_ID=SALE.CLIENT_ID	1,n
CLIENT.CLIENT_ID=RENTAL.CLIENT_ID	1,n
SALE.SALE_TYPE='S'	1,1
RENTAL.SALE_TYPE='R'	1,1

4. Create a Business Layer called Chasm_xx.
5. Deselect the *Multiple SQL statements for each measure* option.
6. Create the following two folders:

- Chasm Objects
- Measures

7. Add the following objects with the following syntax.
   For the Chasm objects class:

Object Name	Syntax
Client Name	CLIENT.CLIENT_LASTNAME +', '+
	CLIENT.CLIENT.FIRSTNAME
Sale Date	SALE.SALE_DATE
Rental Date	RENTAL.SALE_DATE

   For the Measures class:

Object Name	Syntax
Sales Revenue	SUM(SALE.SALE_TOTAL)
Rental Revenue	SUM(RENTAL.SALE_TOTAL)

Hint:
To build the syntax, use the *SQL Assistant*.

8. Perform an integrity check on the following objects:
   - Tables
   - Joins
   - Business Layer
9. Save all changes.
10. To test the universe, create a query in the business layer and restrict the query results to Paul Brent only.
11. So that both Rental Revenue and Sales Revenue are part of the query in addition to Client Name, in the same query on the business layer, add Sales Revenue.
    Refresh the query and note the results.
12. In the Information Design Tool on the data foundation, add the following two contexts to resolve the chasm trap:
    - Sales
    - Rental
13. Save the changes to the data foundation.
14. Recreate the last query from Step 11.
15. Answer these questions:

What is the Sales Revenue?

What is the Rental Revenue?

Unit 17
Solution 37

Use Contexts to Resolve a Chasm Trap

Business Example

You need to report on client sales and rentals. Due to the structure of the database this will create a chasm trap, which will need to be resolved to return accurate results.

Note:
To complete the exercises, replace the *xx* with the group number the instructor provided you with at the beginning of the class.

1. Using the information in the table, create a new project and data foundation called Chasm_xx, where "xx"" stands for your user number. Use the Motors connection.

Field	Value
User Name	Train-xx
Password	Assigned password
Authentication	Enterprise

a) *Choose Start → Program → SAP BusinessObjects → SAP BusinessObjects BI platform 4.0 → SAP BusinessObjects BI platform Client Tools → Information Design Tool.*

b) To create a project called Chasm_xx, choose *File → New → Project*

c) Choose *Finish*.

d) To add a repository session, under the Repository Resources pane, choose + and choose *Insert Session.*

e) Log into the system using the information provided.

f) Browse to the connection folder.

g) Find the existing connection Motors.

h) Right-click the Motors connection and choose *Create Relational Connection Shortcut*.

i) From the pop-up box, choose Chasm_xx.

j) Choose *OK*.

k) To proceed, choose *Close*.
Motors.cns appears under the local project.

l) To create a data foundation with the name Chasm_xx, choose *New* → *Data Foundation*.

m) Choose *Next* → *Single Source* → *Next* → *Select Motors shortcut* → *Finish*.

2. Add the following tables to the Data Foundation:

- CLIENT
- SALE
- RENTAL (as an alias of the SALE table)

a) Choose the *Data Foundation* tab.

b) Choose the + and choose *Insert Tables*.

c) Browse the tables under dbo.

d) Under connections, double-click the CLIENT and SALE tables.

e) Right-click the Sales table that is in the data foundation and from the context menu choose *Insert* → *Alias Table*.

f) Double-click Alias_of_Sales and change the proposed name to **RENTAL**.

g) Choose *OK*.

3. Create the following joins and set the cardinalities:

Join	Cardinality
CLIENT.CLIENT_ID=SALE.CLIENT_ID	1,n
CLIENT.CLIENT_ID=RENTAL.CLIENT_ID	1,n
SALE.SALE_TYPE='S'	1,1
RENTAL.SALE_TYPE='R'	1,1

a) Drag the join between the columns of the tables.

b) To display the properties, double-click the join.

c) To set the cardinality for the self-joins, choose the cardinality drop-down.

d) Choose *Insert* → *Insert Join*.

e) Set the *Expression* and cardinality.

f) Choose Validate.

g) To save the data foundation, choose *Save*.

4. Create a Business Layer called Chasm_xx.

a) Right-click the project.

b) Choose *New* → *Business Layer*.

c) Choose *Relational Data Source* and choose *Next*.

d) Name the business layer **Chasm_xx**, and choose *Next*.

e) To choose the data foundation, choose "..."

f) Choose Chasm_xx.dfx and choose *OK*.

g) Deselect the option *Automatically create classes and objects*.

h) Choose *Finish*.

5. Deselect the *Multiple SQL statements for each measure* option.
   a) Choose the top level of the business layer.
   b) Choose the *Query Options* tab.
   c) Deselect the option *Multiple SQL statements for each measure*.
6. Create the following two folders:
   - Chasm Objects
   - Measures

   a) Choose the top level of the business layer.
   b) From the *Insert Object* drop down, choose *Folder*.
   c) Change the name to **`Chasm Objects`**.
   d) Choose the top level of the business layer.
   e) From the *Insert Object* drop down, choose *Folder*.
   f) Change the name to **`Measures`**.
7. Add the following objects with the following syntax.
   For the Chasm objects class:

Object Name	Syntax
Client Name	CLIENT.CLIENT_LASTNAME +', '+
	CLIENT.CLIENT.FIRSTNAME
Sale Date	SALE.SALE_DATE
Rental Date	RENTAL.SALE_DATE

For the Measures class:

Object Name	Syntax
Sales Revenue	SUM(SALE.SALE_TOTAL)
Rental Revenue	SUM(RENTAL.SALE_TOTAL)

Hint:
To build the syntax, use the *SQL Assistant*.

a) Choose the folder Chasm Objects.

b) From the *Insert Object* drop down, choose *Dimension or Measure*.

c) Change the names as indicated in the table in step 7.

Hint:
For Sale Date and Rental Date, you have to set *Data Type* to Date Time.

d) Choose *SQL Assistant*.

e) Enter the SELECT statement as indicated in the table in step 7.

f) Choose *Validate*.

g) Choose *OK*.

8. Perform an integrity check on the following objects:
   - Tables
   - Joins
   - Business Layer

   a) Right-click the top level of the business layer.

   b) Choose *Check Integrity*.

   c) Choose the following:
      - Tables
      - Joins
      - Business Layer

   d) Choose *Check Integrity*.

   e) Choose *OK*.

9. Save all changes.
   a) Choose *File* → *Save All*.

10. To test the universe, create a query in the business layer and restrict the query results to Paul Brent only.
    a) From the Business Layer choose *Queries*.

    b) Choose *Insert Query*.

    c) Create a query with the Client Name and Sales Revenue objects.

    d) By dragging Client Name to the *Query Filter* area and completing the query filter, apply the following query filter on Client Name for Brent, Paul: Client Name In list Brent, Paul.

    e) Choose *Refresh* and note the results.
    The results should be Sales Revenue = 315,964.50.

f) From the *Result Objects* area, remove Sales Revenue.

g) Add Rental Revenue and refresh the query.

h) Note the results, which should be Rental Revenue = 1,100.

11. So that both Rental Revenue and Sales Revenue are part of the query in addition to Client Name, in the same query on the business layer, add Sales Revenue.

    Refresh the query and note the results.

    a) Add the Sales Revenue object back to the *Result Objects* area.

    b) Choose *Refresh*.

    c) Note that both revenue amounts have doubled.

12. In the Information Design Tool on the data foundation, add the following two contexts to resolve the chasm trap:

    - Sales
    - Rental

    a) Under *Aliases and Contexts* choose *Detect Contexts*.

    b) Choose both contexts and choose *OK*.

13. Save the changes to the data foundation.

    a) Choose *Save*.

14. Recreate the last query from Step 11.

    a) From the business layer, choose *Query*.

    b) Choose the Client Name, Sales Revenue, and Rental Revenue objects.

    c) Drag the Client Name object down into the *Query Filter* area.

    d) Complete the query filter as follows:
    Client Name In list Brent, Paul

    e) Choose *Refresh*.

15. Answer these questions:

    What is the Sales Revenue?

    The Sales Revenue = 315,964.50.

    What is the Rental Revenue?

    The Rental Revenue = 1,100.00.

LESSON SUMMARY

You should now be able to:

- Resolve chasm traps

Unit 17
Lesson 4

Identifying Fan Traps

LESSON OVERVIEW
Working with relational data sources at the SQL level, you can encounter fan traps. This lesson explains how fan traps can occur in the universe structure.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Identify fan traps

Fan Trap Identification

Fan traps occur when there is a "one–to–many" join to a table that "fans out" into another "one–to–many" join to another table.

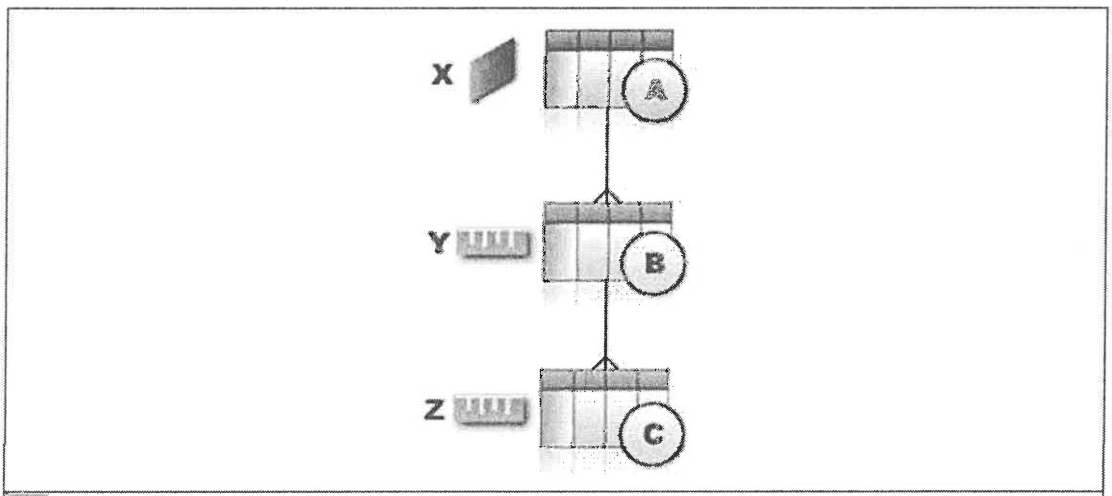


Figure 67: One-to-many fanning to another one-to-many join flow

This is a common structure and does not tend to result in a fan trap. You only get incorrect results from the fan trap when the query includes a measure object on the middle table ('B') of the table path and an object (of any kind) from the subsequent table ('C'). The trap only occurs where (due to the database design) a column in table B holds data values, which are already a sum of those values held at table C. The results are noticeably wrong.

When a query is run using objects Y and Z, the inferred SQL includes tables B and C, which have a 'one-to-many' relationship. This results in a value for the Y object being multiplied by the number of values of the Z object related to that Y object value. Like the chasm trap, the effect is similar to a Cartesian product.

Like the chasm trap, the fan trap can be resolved by executing a separate SELECT statement for object Y and object Z.

You cannot automatically detect fan traps. You need to visually examine the direction of the cardinalities displayed in the table schema.

If you have two tables that are referenced by measure objects and are joined in a series of "many-to-one" joins, then you may have a potential fan trap.

LESSON SUMMARY

You should now be able to:

- Identify fan traps

Unit 17
Lesson 5

Resolving Fan Traps

LESSON OVERVIEW
You resolve fan traps to ensure that valid data is returned in SQL queries. This lesson explains how to resolve fan traps in the universe structure.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Resolve fan traps

Fan Trap Resolution

The possible ways to solve a fan trap problem are:

- Alter the SQL parameters for the universe
- Use a combination of aliases and contexts
- Avoid the fan trap scenario

Alter the SQL Parameters for the Universe

This method is not recommended as it only works for measure objects and may result in inefficiencies in processing the query. This resolution works the same for chasm and fan traps.

Use a Combination of Aliases and Contexts

There are two possible situations that require different solutions.

If You Have	Then
• Three tables in a path containing the initial aggregation, joining it back to the one-to-many relationship • A dimension coming from the first table and measures coming from the two subsequent tables	• Create an alias for the table (on the many end of the join) Use the Detect Contexts tool to detect and propose a context for the alias table and a context for the original table • This is the most effective way to solve the fan trap problem because it works with measure and dimension objects and does not cause inefficiencies
• Two tables in a one-to-many relationship	• Create an alias for the table containing the initial aggregation, joining it back to the original table and then use the *Detect Contexts* tool to detect and propose a

If You Have	Then
	context for the alias table and a context for the original table

Both methods solve the fan trap problem because they work with both measure and dimension objects and do not cause inefficiencies.

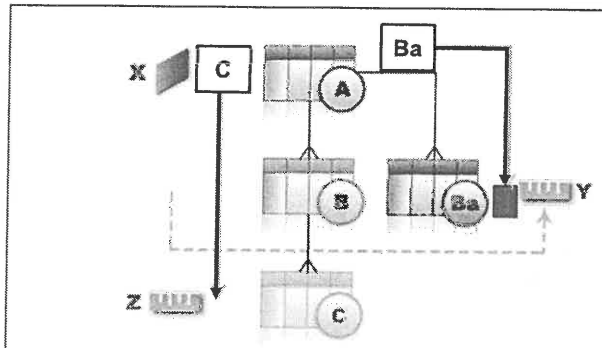
Note:
However, to be more efficient still, using the two–table scenario, you could also use the @aggregate_aware function.

Avoid the Fan Trap Scenario

You can avoid the scenario in the first place by relating all measure objects in the universe to the same table in the universe structure. Avoid placing a measure on anything other than the last table in a table path, which is the table with the "many" cardinality attached to it.

Aliases and Context Resolution Method

You create an alias table for the table producing the aggregation and then detect and implement contexts to separate the query. This procedure is demonstrated in the diagram below:



1. Create an Alias of Table B.
2. Create a join between the Alias Ba and Table A and set cardinality.
3. Set Contexts C and Ba.
4. Change the SELECT clause of object Y so that it refers to the alias Ba rather than Table B.

Figure 68: Fan Trap Solution

As with resolving a chasm trap problem, two contexts must be created. In this example, a context for Sale, and a context for Sale_Model must be defined. This allows for the results to be merged into a single microcube to produce the correct results.

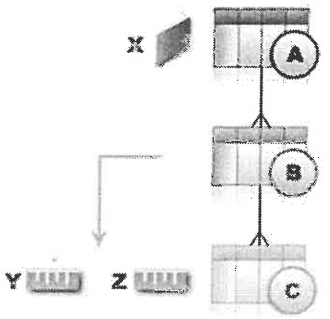
In addition, the SELECT clause of the Sales Revenue object must be edited, so that it refers to the alias table rather than the original Sale table.

If you make a query that includes a dimension object on the lower table in the "one-many-many" path, you do not get the fan trap, even when that dimension object contains the same value for all rows related to the measure value. The fact that the measure and dimension objects are in separate contexts forces two separate SELECT statements, thus avoiding the problem.

Fan Traps Avoidance

In certain situations, it is possible to avoid the fan trap completely, as shown in the diagram below.

To avoid the trap, the database column in table B to which the Y measure object relates must represent a pre-aggregation of more detailed data in table C. If this is the case, you can change the code of the Y measure object so that it refers to table C. Therefore, there is no longer a "one-to-many" relationship incurred.



1. This is only possible where the measure on Table B is a preaggregation of more detailed data which exists in Table C.
2. Code measure object Y so that it refers to the more detailed data in Table C.

Note: This will be less efficient but at least it is accurate.

Figure 69: Avoiding a Fan Trap

This method is used to avoid the fan trap in the Motors universe, when the Sales Revenue and Number of Cars Sold measure objects are included in the same query.

In the Motors universe, the Sales Revenue measure is not based on the total figure in the Sale table, but on a number of columns from the Sale, Sale_Model and Model tables, which are held in the database at the same level of granularity as the number of cars sold. Therefore, no fan trap exists and the correct result is obtained.

Unit 17
Exercise 38

Resolve Fan Traps

Business Example

You need to report on car sales revenue and the number of cars sold at the client and order level. Due to the database structure, this will result in a fan trap, which will need to be resolved to return accurate results.

Note:
In the data provided for you to complete the exercises, replace the xx with the group number the instructor provided you with at the beginning of the class.

1. Using the Motors connection, create a new project and data foundation called Fan_xx.
2. Add the following tables to the Data Foundation:
   - SALE
   - SALE_MODEL
3. Create the following joins and set the cardinalities as indicated:
   - CLIENT.CLIENT_ID=SALE.CLIENT_ID 1,n
   - SALE.SALE_ID=SALE_MODEL.SALE_ID 1,n
   - SALE.SALE_TYPE='S' 1,1
4. Create a business layer called Fan_xx.
5. Deselect the "Multiple SQL statements for each measure" option.
6. Create the following two folders:
   - Fan Objects
   - Measures
7. Add the following objects with the following syntax:
   For the Fan Objects class:

Object Name	Syntax
Client Name	`CLIENT.CLIENT_LASTNAME +', '+`
	`CLIENT.CLIENT_FIRSTNAME`

Object Name	Syntax
Model ID	`SALE_MODEL.MODEL_ID` Note: For Model ID you have to set Data Type to Number.

•

For the Measures class:

Object Name	Syntax
Sales Revenue	`SUM(SALE.SALE_TOTAL)`
Number of Cars Sold	`SUM(SALE_MODE.SALE_QTY)`

8. Perform an integrity check on the following objects:
   - Tables
   - Joins
   - Business Layer
9. Save all the changes.
10. To test the universe, in the business layer, create a query. Restrict the query results to Randall, Sean. Note the results.

    What is the Sales Revenue?

    _____

    _____

    _____
11. In the same query on the Business Layer, add the Model ID object to the *Result Objects* area. Refresh the query and note the results.

    What is the measure value?

    _____

    _____

    _____
12. On the *Data Foundation* tab of the Information Design Tool, create an alias of the Sales table.
13. Using the data in the following table, join the new table to the Client table and add the appropriate self join to the alias.
    - CLIENT.CLIENT_ID=Alias_of_SALE.CLIENT_ID 1,n
    - Alias_of_SALE.SALE_TYPE ='S' 1,1

14. On the *Data Foundation* tab in the Information Design Tool, add the following two contexts:
    - Sale Model
    - Alias_of_SALE
15. Save the changes to the data foundation.
16. Modify the definition of the Sales Revenue object so that it is defined on the new alias table.
17. Save the changes to the business layer.
18. Recreate the last query from Step 10.

    What is the Sales Revenue?

    _____

    _____

    _____

Note:
Choose the *Flow* above the results to see the:

- Revenue: 57, 091.50
- Models purchased: 1034 and 1081

Unit 17
Solution 38

Resolve Fan Traps

Business Example

You need to report on car sales revenue and the number of cars sold at the client and order level. Due to the database structure, this will result in a fan trap, which will need to be resolved to return accurate results.

Note:
In the data provided for you to complete the exercises, replace the xx with the group number the instructor provided you with at the beginning of the class.

1. Using the Motors connection, create a new project and data foundation called Fan_xx.
   a) Choose *New* → *Project* and enter **`Fan_xx`** as the project name.
   b) Choose *Finish*.
   c) Add a repository session.
   d) Browse the connection folder.
   e) Find the existing connection "Motors".
   f) Right-click *Motors connection* → *Create Relational Connection Shortcut*.
   g) From the pop-up box, choose Fan_xx.
   h) Choose *Finish*.
   i) To proceed, choose *Close*.
   Motors.cns appears under the local project.
   j) Choose *New* → *Data Foundation* and enter the name **`Fan_xx`** and choose *Next*.
   k) Choose *Single Source* and then *Next*.
   l) Choose *Select Motors shortcut* and then *Finish*.
2. Add the following tables to the Data Foundation:
   - SALE
   - SALE_MODEL

   a) Choose the *Data Foundation* tab.
   b) Choose + and choose *Insert Tables*.
   c) Browse the tables under dbo.

d) Under connections, double-click the CLIENT, SALE and SALE_MODEL tables.

e) Choose *Finish*.

3. Create the following joins and set the cardinalities as indicated:

   - CLIENT.CLIENT_ID=SALE.CLIENT_ID 1,n
   - SALE.SALE_ID=SALE_MODEL.SALE_ID 1,n
   - SALE.SALE_TYPE='S' 1,1

   a) Using the information in the first two rows of the table above, drag the join between the columns of the tables.

   b) To display the properties, double-click the join.

   c) To set the cardinality as indicated, choose the cardinality drop-down.

   d) For the self-join (third join listed above), choose *Insert* → *Insert Join*.

   e) Set the expression and cardinality.

   f) Choose *Validate*.

   g) Save your changes.

4. Create a business layer called Fan_xx.

   a) Choose the project.

   b) Choose *New* → *Business Layer*.

   c) Choose *Relational Data Source* and then *Next*.

   d) Name the business layer **`Fan_xx`** and choose *Next*.

   e) To choose the data foundation, choose ... (the ellipsis).

   f) Choose Fan_xx.dfx and choose *OK*

   g) Deselect the *Automatically create classes and objects* option.

   h) Choose *Finish*.

5. Deselect the "Multiple SQL statements for each measure" option.

   a) Choose the top level of the business layer.

   b) Choose the *Query Options* tab.

   c) Deselect the Multiple SQL statements for each measure option.

6. Create the following two folders:

   - Fan Objects
   - Measures

   a) Choose the top level of the business layer.

   b) From the *Insert Object* drop down, choose *Folder*.

   c) Change the name to **`Fan Objects`**.

d) Choose the top level of the business layer.

e) From the *Insert Object* drop down, choose *Folder*.

f) Change the name to **Measures**.

7. Add the following objects with the following syntax:

For the Fan Objects class:

Object Name	Syntax
Client Name	CLIENT.CLIENT_LASTNAME +', '+
	CLIENT.CLIENT_FIRSTNAME
Model ID	SALE_MODEL.MODEL_ID  Note: For Model ID you have to set Data Type to Number.

For the Measures class:

Object Name	Syntax
Sales Revenue	SUM(SALE.SALE_TOTAL)
Number of Cars Sold	SUM(SALE_MODE.SALE_QTY)

a) Choose the Fan Objects folder.

b) From the *Insert Object* drop down, choose *Dimension*.

c) Change the names to Client Name.

d) Choose *SQL Assistant*.

e) Enter the SELECT statement as indicated in the table in step 7.

f) Choose *Validate*.

g) Choose *OK*.

h) Repeat steps a through g for the second object for the Fan Objects class as listed in the table in step 7.

i) Repeat steps a through g for the objects in the Measures class as listed in the second table in step 7.

8. Perform an integrity check on the following objects:

- Tables
- Joins
- Business Layer

a) Right-click the top level of the business layer and choose *Check Integrity*.

b) Choose the following: objects:

- Tables
- Joins
- Business Layer

c) Choose *Check Integrity*.

d) Choose *OK*.

9. Save all the changes.
   a) Choose *File* → *Save All*.

10. To test the universe, in the business layer, create a query. Restrict the query results to Randall, Sean. Note the results.

    What is the Sales Revenue?

    Sales Revenue = 57,091.50

    a) From the business layer, choose *Query*.

    b) Choose *New*.

    c) Create a query with the Client Name and Sales Revenue objects.

    d) By dragging Client Name to the *Query Filter* area and completing the query filter, apply a query filter on Client Name for Randall, Sean: Client Name In list Randall, Sean.

    e) Choose *Refresh* and note the results.

11. In the same query on the Business Layer, add the Model ID object to the *Result Objects* area. Refresh the query and note the results.

    What is the measure value?

    Sales Revenue = 114,183.00

    a) Double-click the Model ID object.

    b) Choose *Refresh*.

    c) Note the results.

12. On the *Data Foundation* tab of the Information Design Tool, create an alias of the Sales table.
    a) Right-click the Sales table and choose *Insert* → *Alias Table*.

    b) Choose *OK*.

13. Using the data in the following table, join the new table to the Client table and add the appropriate self join to the alias.

- CLIENT.CLIENT_ID=Alias_of_SALE.CLIENT_ID 1,n
- Alias_of_SALE.SALE_TYPE ='S' 1,1

a) Use drag and drop to create the following join:
```
CLIENT.CLIENT_ID=Alias_of_SALE.CLIENT_ID
```

b) Double-click the join.

c) Set the cardinality to 1,n and choose *OK*.

d) Choose *Insert* → *Join*.

e) Add the following expression:
```
Alias_of_SALE.SALE_TYPE ='S'
```

f) Choose *Validate*.

g) Choose *Close*.

h) Set the cardinality to 1,1.

i) Choose *OK*.

14. On the *Data Foundation* tab in the Information Design Tool, add the following two contexts:
    - Sale Model
    - Alias_of_SALE

    a) Under Aliases and Contexts choose *Detect Contexts*.

    b) Choose both contexts.

    c) Choose *OK*.

15. Save the changes to the data foundation.
    a) Choose *Save*.

16. Modify the definition of the Sales Revenue object so that it is defined on the new alias table.
    a) Choose the business layer.

    b) Choose the Sales Revenue object.

    c) Edit the SELECT statement to include the following phrase:
    ```
    SUM(Alias_of_SALE.SALE_TOTAL)
    ```

    d) Choose *OK*.

17. Save the changes to the business layer.
    a) Choose *Save*.

18. Recreate the last query from Step 10.

    What is the Sales Revenue?

    Sales Revenue = 57,091.50

Note:
Choose the *Flow* above the results to see the:

- Revenue: 57, 091.50
- Models purchased: 1034 and 1081

a) From the *Business Layer*, choose *Query*.

b) Choose the Client Name, Sales Revenue, and Model ID objects.

c) Drag the Client Name object down into the *Query Filter* area.

d) Complete the *Query Filter* as follows:
Client Name In list Randall, Sean

e) Choose *Refresh* and note the results.

LESSON SUMMARY

You should now be able to:

- Resolve fan traps

Learning Assessment

1. Which of the following traps are problems inherent in SQL?

   Choose the correct answers.

   - [ ] A Mouse traps
   - [ ] B Fan traps
   - [ ] C Wind traps
   - [ ] D Dirt traps
   - [ ] E Chasm Traps

2. Loops and chasm traps are detected automatically by the Information Design Tool.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

3. What can be used to avoid chasm traps?

   Choose the correct answer.

   - [ ] A Aggregates
   - [ ] B Builds
   - [ ] C Contexts
   - [ ] D Demonstrations

4. Fan traps occur when there is a "one–to–many" join to a table that "fans out" into another "one–to–many" join to another table.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

5. What are the recommended ways to solve a fan trap problem?

Choose the correct answers.

- [ ] A Avoid the fan trap scenario
- [ ] B Alter the SQL parameters for the universe
- [ ] C Edit the client list and name
- [ ] D Use a combination of aliases and contexts

Learning Assessment - Answers

1. Which of the following traps are problems inherent in SQL?

   Choose the correct answers.

   - [ ] A Mouse traps
   - [x] B Fan traps
   - [ ] C Wind traps
   - [ ] D Dirt traps
   - [x] E Chasm Traps

2. Loops and chasm traps are detected automatically by the Information Design Tool.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

3. What can be used to avoid chasm traps?

   Choose the correct answer.

   - [ ] A Aggregates
   - [ ] B Builds
   - [x] C Contexts
   - [ ] D Demonstrations

4. Fan traps occur when there is a "one–to–many" join to a table that "fans out" into another "one–to–many" join to another table.

   Determine whether this statement is true or false.

   - [x] True
   - [ ] False

5. What are the recommended ways to solve a fan trap problem?

   Choose the correct answers.

   - [X] A Avoid the fan trap scenario
   - [X] B Alter the SQL parameters for the universe
   - [ ] C Edit the client list and name
   - [X] D Use a combination of aliases and contexts

UNIT 18 Outer Join Problem Resolution

Lesson 1

UNIT OBJECTIVES

- Resolve an ambiguous outer join

Unit 18
Lesson 1

Resolving an Ambiguous Outer Join Using @AggregateAware

LESSON OVERVIEW
This lesson explains how to use @AggregateAware to resolve issues surrounding outer joins when used in the Data Foundation of a universe.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Resolve an ambiguous outer join

Ambiguous Outer Joins

The Information Design Tool supports outer joins on the data foundation. Depending on the relational database management system (RDBMS), however, outer joins can create issues with WebIntelligence queries.

For example, consider the Motors universe. The REGION table is joined to the CLIENT table, which is joined to the SALES table. With some RDBMSs, the CLIENT table is joined to the SALE table, so when a query uses data from all three tables, the query either may not run or may return incorrect results if a Region does not have a Client, because there can be no Sale.

@Aggregate_Aware Function

The aggregate awareness functionality within the Information Design Tool allows the universe designer to take advantage of pre-aggregated data that exists within a data structure. However, the function can be viewed simply as a method for specifying preferential coding based on the other objects contained in a query. When viewed from this perspective, it becomes apparent that aggregate awareness can be used to resolve other issues than those it is originally intended to resolve.

For instance, aggregate awareness can be used to resolve a particular fan trap that, when resolved using the classic fan trap solution outlined in a previous unit, results in efficient SQL being generated.

It can also be used to resolve issues surrounding outer joins when used in the Data Foundation of a universe.

Note:
There are no issues with the SQL Server that we are using in this class. In Oracle, no null values are retrieved; in some other databases, an ambiguous outer join message appears.

If there are issues with the RDBMS used, there are two ways to resolve them:

1. Do not use outer joins in your data foundation. Obviously, this solution is not practical if the end-users' reporting needs require blank rows to be returned.
2. Create an alias of the table where the outer join is required, and use @Aggregate_Aware in the relevant object(s) to identify which join path to take depending on the objects used in the query.

Unit 18
Exercise 39

Resolve an Ambiguous Outer Joins

Business Example

You need to provide outer join information to your end users, but because of the relational database you are using, an outer join in the middle of a path of joins on your data foundation will produce an error in the end users' queries. You will use @Aggregate_Aware to resolve this issue.

> Note:
> In the data provided for you to complete the exercises. replace the value xx with the group number the instructor provided you with at the beginning of the course.

1. Create a new project and data foundation called AOJ_xx. User the Motors connection.
2. Add the following tables to the Data Foundation:
   - REGION
   - CLIENT
   - SALE
3. Create the following joins and set the cardinalities:

Join	Cardinality
REGION.REGION_ID=CLIENT.REGION_ID	1,n
CLIENT.CLIENT_ID=SALE.CLIENT_ID	1,n

> Note:
> Because we are using a SQL Server database for this course, there is no ambiguous outer join, but we will proceed as if there would be a problem.

4. Create an alias of the CLIENT table.
5. Join the new table to the REGION table and specify the join as an outer join on the REGION table.

Join	Cardinality
REGION.REGION_ID= Alias_of_CLIENT.REGION_ID	1,n

6. Create a business layer called AOJ_xx.
7. Create a folder called AOJ Objects.
8. Add the following objects with the following syntax.
   For the AOJ Objects class:

Object Name	Object Type	Syntax
Region Name	Dimension	REGION.REGION_NAME
Client Last Name	Dimension	@Aggre-gate_Aware(Alias_of_CLI-ENT.CLIENTLAST- NAME, CLIENT.CLIENTLAST-NAME)
Sales Revenue	Measure	sum(SALE.SALE_TOTAL)

9. Make the Sales Revenue object incompatible with the Alias_of_CLIENT table.
10. Perform an integrity check on the following objects:
    - Tables
    - Joins
    - Business Layer
11. Save all changes.
12. To test the universe, create a 1uery in the business layer.
13. In the same query on the Business Layer, add the Sales Revenue object to the *Result Objects* area.

Unit 18
Solution 39

Resolve an Ambiguous Outer Joins

Business Example

You need to provide outer join information to your end users, but because of the relational database you are using, an outer join in the middle of a path of joins on your data foundation will produce an error in the end users' queries. You will use @Aggregate_Aware to resolve this issue.

Note:
In the data provided for you to complete the exercises. replace the value xx with the group number the instructor provided you with at the beginning of the course.

1. Create a new project and data foundation called AOJ_xx. User the Motors connection.
   a) Choose *New* → *Project*, naming the project **AOJ_xx**.
   b) Choose *Finish*.
   c) Add a repository session.
   d) Browse the connection folder.
   e) Find the existing connection Motors.
   f) Right-click Motors connection and from the context menu, choose *Create Relational Connection Shortcut*.
   g) From the pop-up box, chooose AOJ_xx.
   h) Click *OK*.
   i) To proceed, choose *Close*.
   "Motors.cns" will appear under the local project.
   j) Choose *New* → *Data Foundation*, naming it **AOJ_xx** and choose *Next*.
   k) Choose *Single Source* and then *Next*.
   l) Choose Motors.cns and then *Finish*.
2. Add the following tables to the Data Foundation:
   - REGION
   - CLIENT
   - SALE

   a) Choose the *Data Foundation* tab.

b) Choose on the + and choose *Insert Tables*.

c) Browse the tables under dbo.

d) Under connections, double-click the REGION, CLIENT and SALE tables.

e) Choose *Finish*.

3. Create the following joins and set the cardinalities:

Join	Cardinality
REGION.REGION_ID=CLIENT.REGION_ID	1,n
CLIENT.CLIENT_ID=SALE.CLIENT_ID	1,n

Note:
Because we are using a SQL Server database for this course, there is no ambiguous outer join, but we will proceed as if there would be a problem.

a) Drag the join between the columns of the tables.

b) Double-click the join to display the properties.

c) Click the cardinality drop-down to set the cardinality

d) For the self-join, choose *Insert* → *Insert Join* and set the *Expression* and cardinality.

e) Choose *Validate*.

4. Create an alias of the CLIENT table.

a) Right-click the CLIENT table.

b) Choose *Insert* → *Alias Table*.

c) Choose *OK*.

5. Join the new table to the REGION table and specify the join as an outer join on the REGION table.

Join	Cardinality
REGION.REGION_ID= Alias_of_CLIENT.RE-GION_ID	1,n

a) Use drag and drop to create the join in the table in step 5.

b) Double-click the join.

c) Set the cardinality to 1,n.

d) Under the REGION table, check the *Outer Join* box.

e) Choose *OK*.

f) Choose *Save*.

6. Create a business layer called AOJ_xx.

a) Choose the project.

b) Choose *New* → *Business Layer*.

c) Choose *Relational Data Source* and then choose *Next*.

d) Name the business layer **AOJ_xx** and choose *Next*.

e) To select the data foundation, choose "...".

f) Choose AOJ_xx and choose *OK*.

g) Deselect *Automatically create classes and objects*,

h) Click *Finish*.

7. Create a folder called AOJ Objects.
   a) Choose the top level of the Business Layer.
   b) From the *Insert Object* drop down, choose *Select Folder*.
   c) Change name to **AOJ Objects**.
8. Add the following objects with the following syntax.
   For the AOJ Objects class:

Object Name	Object Type	Syntax
Region Name	Dimension	REGION.REGION_NAME
Client Last Name	Dimension	@Aggre-gate_Aware(Alias_of_CLI-ENT.CLIENTLAST- NAME, CLIENT.CLIENTLAST-NAME)
Sales Revenue	Measure	sum(SALE.SALE_TOTAL)

   a) Select the class/folder Fan Objects.
   b) From the *Insert Object* drop down, choose *Dimension* or *Measure*.
   c) Using the data in the table in step 8, change the *Name* of the object.
   d) Choose SQL Assistant.
   e) Using the data in the table in step 8, enter the SELECT statement.

> Note:
> Depending on the objects used in the query, the @Aggregate_Aware makes the business layer aware that there is an alternative for obtaining the data.

   f) Choose *Validate*.
   g) Choose *OK*.
9. Make the Sales Revenue object incompatible with the Alias_of_CLIENT table.

a) Choose *Actions* → *Set Aggregate Navigation*.

b) On the left of the pop-up, highlight the Alias_of_CLIENT table.

c) On the right of the pop-up, open the class and place a check mark next to the Sales Revenue object.

d) Choose *OK*.

10. Perform an integrity check on the following objects:

- Tables
- Joins
- Business Layer

a) Right-click the top level of business layer.

b) Choose *Check Integrity*.

c) Select the following:

- Tables
- Joins
- Business Layer

d) Choose *Check Integrity*.

e) Choose *OK*.

11. Save all changes.

a) Choose *File* → *Save All*.

12. To test the universe, create a 1uery in the business layer.

a) From the Business Layer choose *Query*.

b) Choose *New*.

c) Create a query with the Region Name and Client Last Name objects.

d) View the SQL script and notice which table is used to pull the Client Last Name data: Alias_of_CLIENT

13. In the same query on the Business Layer, add the Sales Revenue object to the *Result Objects* area.

a) Double-click the Sales Revenue object.

b) View the SQL script and notice which table is used to pull the Client Last Name data: CLIENT

LESSON SUMMARY

You should now be able to:

- Resolve an ambiguous outer join

Unit 18

Learning Assessment

1. Which function within the Information Design Tool was intended to allow the universe designer to take advantage of pre-aggregated data that exists within a data structure

   Choose the correct answer.

   - ☐ A @ExecuteData
   - ☐ B @DataAggregate
   - ☐ C @ExecuteAware
   - ☐ D @Aggregate_Aware

Learning Assessment - Answers

1. Which function within the Information Design Tool was intended to allow the universe designer to take advantage of pre-aggregated data that exists within a data structure

   Choose the correct answer.

   - [ ] A @ExecuteData
   - [ ] B @DataAggregate
   - [ ] C @ExecuteAware
   - [x] D @Aggregate_Aware

UNIT 19 Universe Creation from Different Data Sources

UNIT OBJECTIVES

- Identify data sources for universes
- Create an OLAP universe
- Create a multisource universe
- Use calculated columns
- Use federated tables

Unit 19
Lesson 1

Identifying the Different Data Sources

LESSON OVERVIEW
The universe access list continues to grow as the variety of data sources evolve in customer environments. This accessibility creates even more robust reporting capabilities as universes can be created for many data sources.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Identify data sources for universes

Data Sources for Universes

Most organizations have several data sources that serve a variety of different purposes. Many of those data sources can be used for end user reporting. In addition to a relational data source, the Information Design Tool allows you to create a universe from a number of different data sources.

Creating Universes from Different Data Sources

- OLAP cubes
- XML metadata files
- Stored procedures
- A combination of these data sources

We will see how to create a universe on the most common data sources at this time: an OLAP cube and multiple data sources.

LESSON SUMMARY
You should now be able to:

- Identify data sources for universes

Unit 19
Lesson 2

Creating an OLAP Universe

LESSON OVERVIEW

An OLAP universe is a SAP BusinessObjects universe that has been generated from an OLAP cube or query. The Business Layer is created automatically from a selected OLAP Connection. Once the universe has been created it can be exported to the Central Management System (CMS) as any other universe, and is then available to Web Intelligence users to run queries and create reports.

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Create an OLAP universe

OLAP Universes

On Line Analytical Processing, or OLAP, allows end users to extract concise answers to business questions such as: "Show me the sales generated during a specified period, for a specified product in a specified location less than a specified amount".

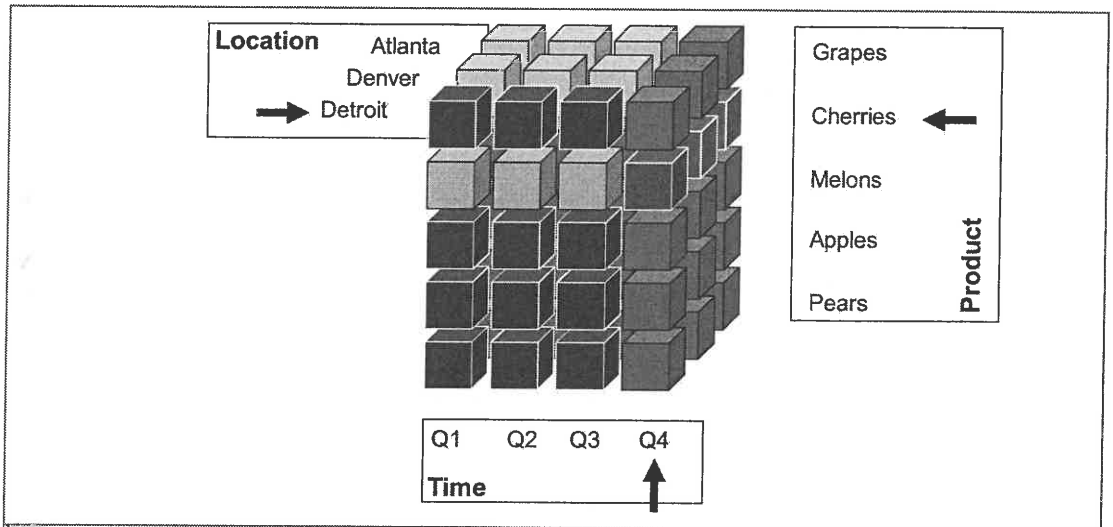


Figure 70: Sample OLAP Cube

Using OLAP, a user can quickly narrow or expand the scope of a business question. For example, in the scenario illustrated above, the sales for a single product, a group of products, or many products can be displayed. In a traditional relational reporting system, each question would require a separate relational query to summarize historical data. If the volume of transactions is high, it takes a long time to produce the desired results. Comparable queries in an OLAP source can deliver answers with a better response time.

The power of an OLAP source is its ability to quickly and dynamically summarize data using a variety of dimensions. In addition, an OLAP source provides a powerful calculation engine that supports on demand analysis to extend the value of the information beyond the data physically stored in the OLAP database.

Dimensions

An OLAP source is constructed from multiple components.

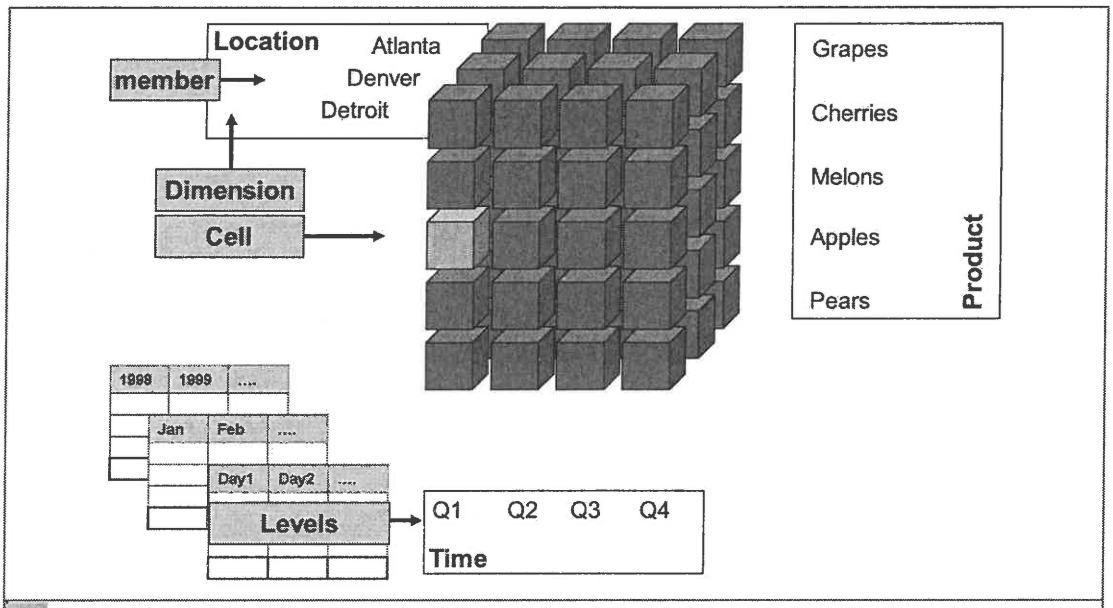


Figure 71: OLAP Dimensions

In the figure OLAP Dimensions, the dimensions are Location, Product, and Time. Dimensions define the context of the numerical data under analysis. For example, knowing that there are $1 million in sales is not meaningful until you put this amount in context with one or more dimensions. Is this sales amount for one month or for one product, or is it the annual sales for all products in a single city?

A dimension is comprised of representative values, or members. In the figure, Atlanta is a member of the Location dimension and cherries is a member of the Product dimension. When you query an OLAP source, you select one or more members of a dimension or you can choose to summarize all members of a dimension. For example, you can select a particular location or you can summarize all locations. When the design of a dimension allows this type of summarization, a special type of member known as the "All Member" is available to provide a summarized value for the members of that dimension.

Members can be organized into multiple "levels". Time is a dimension that can contain many levels; for example Year, Quarter, and Month. Depending on analytical requirements, this dimension can include Week, Day, and even Hour and Minutes. Levels organize a dimension into a hierarchy with the most summarized values at the top and the most detailed values at the bottom. In the example above, Q1, Q2, Q3, and Q4 are all members of the Quarter level of the Time dimension, while Jan and Feb are members of the Month level of the same dimension. When you query an OLAP source, levels allow you to drill up or drill down a dimension to view summary or detailed data.

Measures

Dimensions form the sides, or axes, of a cube. The cube cells represent the intersection of each dimension member with members of all other dimensions. Contained within each cell is

the "Measure", which is the numerical data being analyzed and summarized. A cube can contain one or more measures.

OLAP Universe Generation

The Business Layer of an OLAP universe is created automatically from an OLAP Connection.

Note:
OLAP universes support a single cube in the universe.

OLAP structures are mapped directly to folders, measures, dimensions, attributes, and filters in the Business Layer. A unique aspect of an OLAP universe is that there is no Data Foundation.

You can use the Information Design Tool to create OLAP universes from the following OLAP data sources:

- Microsoft Analysis Services 2005
- Microsoft Analysis Services 2008
- Microsoft Analysis Services 2012
- Hyperion Essbase 11.x

Once the Business Layer has been created, it can be exported to the central management server (CMS) as any other universe. The universe is then available to end users to run queries and create reports.

Unit 19
Exercise 40

Create an OLAP Universe

Business Example

The end users want to be able to report against data held in an OLAP data source.

> Note:
> To complete the exercises, replace the value *xx* with the group number the instructor provided you with at the beginning of the course.

1. Create a project and a connection to the required OLAP data source using the following details:

Authentication Mode Used	Specified Name and Password
Server	http://wdflbmt2065:1080/olap/msmdpump.dll
User name	train-olap
Password	Welcome
Language	English (United States)

2. Create a Business Layer called My OLAP Business Layer_xx.
3. Save all changes.
4. Create a *Query* in the business layer to test the universe.

Unit 19
Solution 40

Create an OLAP Universe

Business Example

The end users want to be able to report against data held in an OLAP data source.

> Note:
> To complete the exercises, replace the value *xx* with the group number the instructor provided you with at the beginning of the course.

1. Create a project and a connection to the required OLAP data source using the following details:

Authentication Mode Used	Specified Name and Password
Server	http://wdflbmt2065:1080/olap/msmdpump.dll
User name	train-olap
Password	Welcome
Language	English (United States)

a) Choose *File* → *New* → *Project* and name the Project `My OLAP Resources_xx`.

b) Choose Finish.

c) To create an OLAP connection, right-click on your project and choose *New* → *OLAP Connection*.

d) For the *Resource Name*, enter `My OLAP Connection_xx` and choose *Next*.

e) In the hierarchical list, expand the Microsoft Analysis Services 2008 branch and choose Xmla, and choose *Next*.

f) Enter the details in the table in step 1 and choose *Next*.

> Note:
> Allow a few seconds for the IDT to display all the folders and cubes.

g) Expand the Adventure Works DW folder and choose the Adventure Works cube and choose *Finish*.

h) In the *Local Project* view, expand My OLAP Resources.

i) Right-click the connection and choose *Publish Connection to a Repository*.

j) Enter the credentials for the repository and choose Next.

k) Choose the folder to which the connection is to be published.
(In this case, choose the root folder.)

l) To publish the connection, choose *Finish*.

m) To create a shortcut in the project, choose *Yes*.

n) Choose *Close*.

2. Create a Business Layer called My OLAP Business Layer_xx.
   a) Right-click the project and choose *New* → *Business Layer*.
   b) Choose *OLAP Data Source* and then *Next*.
   c) Name the business layer **`My OLAP Business Layer_xx`** and choose *Next*.
   d) To choose the OLAP connection, choose ...
   e) Choose *My OLAP Connection_xx* and choose *OK*.
   f) Select the *Detect measure aggregation function* and *Create attribute from unique* checkboxes.
   g) Choose *Next*.
   Note that you can choose which dimensions and measures from the cube you want to be part of your business layer. For this exercise, do not change any default check marks.
   h) Choose *Finish*.

Note:
At this point, as a best practice, you should clean up the Business Layer by creating folders for the dimensions, attributes, and measures, just as you would in a business layer that uses a data foundation based on a relational connection. According to the end-user requirements, the dimensions, attributes, and measures that are automatically generated should be moved, deleted, rearranged, and hidden. In this exercise, we will not take the time to perform this task, but remember that it is a best practice.

3. Save all changes.
   a) Select *File* → *Save All*.
4. Create a *Query* in the business layer to test the universe.
   a) From the Business Layer, choose *Query*.
   b) Choose *New*.
   c) Create a query with the Sales Order Number object from the Sales Summary Order Details dimension and the Sales Amount objects from the Sales Summary folder.
   d) Choose *Refresh* and note the results.

OLAP Universe Modification

Once the Business Layer has been generated automatically, designers can make the following types of modifications:

- Hide, duplicate, and rename folders and objects (dimension, detail, and measure).
- Insert new folders and objects (dimension, detail, and measure).
- Edit an object's format.
- Edit an object's data type (character and number only).
- Use universe functions (@variable, @select, @prompt).
- Create a cascading list of values.
- Define a delegate search for a list of values, allowing users to limit loading of the list of values at query runtime.
- Define measures with database delegated projection function (delegated measures).
- Create calculated measures.
- Run the Check Integrity or Parse tools at any time.
- Parse dimension, detail, and measure object MDX syntaxes.
- Create predefined conditions/filters.
- Define optional prompts.

The following features are not supported for OLAP universes:

- You cannot set row-level security authorizations in an OLAP universe.
- You cannot edit a list of values in an OLAP universe.

Calculated Measures

Calculated measures allow universe designers to create their own measures manually. To create calculated measures, designers create an MDX expression using the MDX assistant. Information Design Tool functions are allowed in calculated measure expressions, for example:

- @select
- @prompt
- @variable

The check integrity validates the XML syntax and any of the @Functions.

The Information Design Tool also supports constants in the expression such as: "10" and "ABC".

LESSON SUMMARY

You should now be able to:

- Create an OLAP universe

Unit 19
Lesson 3

Creating a Multi-Source Universe

LESSON OVERVIEW
A Multi-source universe is one that utilizes more than one data foundation for the business layer. This lesson explains how to create a multi-source universe.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Create a multisource universe
- Use calculated columns
- Use federated tables

Multisource Universe

A multisource universe is simply one single data foundation and business layer of a universe that utilizes more than one data connection. This capability allows universe designers to create a single universe that accesses data in multiple data sources, including relational data sources, SAS, and SAP BW.

Creating and maintaining a multisource universe is the same as creating a single-source dimensional universe. For a multisource universe, you simply use more than one connection for your data foundation. You can add connections when you create the data foundation, and anytime later. However, multisource-enabled data foundations support only secured connections, and universes based on this type of data foundation can only be published to a repository.

Multi-Source Enabled Data Foundations

Multisource-enabled data foundations support most relational connections that a single-source data foundation uses. In addition, multisource-enabled data foundations support the following relational connections that are not supported in single-source data foundations:

- SAP BW connections
- SAS connections

SQL-92 standard syntax is the default for any calculated columns, derived tables, and join expressions used in the data foundation. In addition, the SAP BusinessObjects SQL functions are available.

You can use database-specific SQL syntax in a multisource-enabled data foundation by defining a database-specific derived table or calculated column. Database-specific SQL syntax allows functions or operators that are offered by a specific database and not by standard SQL-92 (for example, Oracle analytical functions).

Situations Requiring Multi-Source Enabled Data Foundations

Multisource-enabled data foundations are required in the following situations:

- You want to insert tables and joins from more than one relational data source.
- You want to use SAP BW or SAS connections.
- You want to use SQL-92 standard syntax and SAP BusinessObjects SQL functions.

Multi-Source Enabled Data Foundations with an SAP BW Connection

When you add an SAP BW connection to a multisource-enabled data foundation, tables and joins are automatically inserted. The automatic insertion does the following:

SAP BW Connections: Tables and Joins Insertion

- Based on the InfoProvider specified in the SAP BW connection parameters, inserts the fact table and its related master data and text tables.
- Creates families for each type of table:

  - Prefixes the fact table name by the letter I (InfoCube) and assigns it to family "InfoProvider Fact Table".

  - Prefixes master data tables by the letter D (Dimension) and assigns them to family "Dimension Table".

  - Prefixes text tables by the letter T (Text) and assigns them to family "Text Table".
- Creates alias tables for all dimension and text tables.
- Detects and inserts table keys.
- Detects and inserts joins.
- Creates input columns in tables when needed, to handle time-dependent data. Creates a parameter in the data foundation called key date. By default, at query time, the key date parameter is not prompted. It is automatically assigned the current date

SQL Expressions in Multi-Source Enabled Data Foundations

SQL expressions that define joins, calculated columns, and derived tables in a multisource-enabled data foundation use SQL-92 ANSI standard syntax. In SQL-92 expressions, you can include SAP BusinessObjects SQL functions and @functions. Which @functions you can include depends on the type of expression.

To use functions or operators that are offered by the database and not by SQL-92 (for example, Oracle analytical functions), you define database-specific calculated columns and derived tables. A check box in the SQL Expression Editor allows you to use database-specific SQL. Database-specific calculated columns and derived tables support the SQL syntax of the associated connection.

Rules Applying the Database-Specific SQL Expressions

- You can reference only standard tables and database-specific derived tables in a single connection
- You cannot reference tables in SAS or SAP BW connections
- You can include @functions with certain restrictions.

Hint:
For more details about the @functions, see the "Information Design Tool User's Guide".

Multiple Connection Data Foundations

To add multiple connections to a data foundation, you must select the Multisource-Enabled connection type when you create the data foundation. You can then select multiple connections for the data foundation. You can also add connections to an existing multisource-enabled data foundation.

Connections must be secured, and therefore available in a repository. The connections are represented by a connection shortcut in the local project. The connections in a multisource-enabled data foundation have the following additional properties:

Multisource-Enabled Connection Properties

- A short name used to identify the connection in the data foundation and to modify the table name in SQL expressions. You specify the short name when adding the connection. This name must be unique within the data foundation and is limited to 40 characters. If you change the short name for the connection, the SQL expressions are automatically updated with the new name.
- A color for the connection. This color is used in the table header in data foundation views. You select the color when adding the connection. You can change the color for a connection at any time. This color-coding helps to track what tables are from which data source.
- A catalog used to identify the connection to the query server. A default catalog name is registered automatically with the query server the first time the connection is added to any multisource-enabled data foundation.
- For SAP BW connections, properties related to the automatic insertion of tables and joins.

In a multisource-enabled data foundation, the table name as it appears in SQL expressions has the format: `<@catalog(short name)."database_qualifier.database_owner".table_name>`.

A multisource join can be created between tables from different connections. You can use the Detect Joins command to detect joins between tables referenced in different connections, or explicitly define them with the Insert Join command.

Unit 19
Exercise 41

Create a Multisource Universe

Business Example

Your end users need to report on information about the number of customers and the number of employees in various regions. The information on the customers is stored in a SQL server database, but the information on the employees is stored in an Access database. You need to create a single universe that allows the end users to access the information in both data sources.

In the following exercise, you will use connections that are created on a SQL Server database and an Access database. In the next step, you will create a data foundation based on those two connections, and finally you will create a business layer on top of this data foundation.

Note:
In the data provided for you to complete the exercises. replace the value xx with the group number the instructor provided you with at the beginning of the course.

1. Create a project called CrossOver_xx and add shortcuts to two connections: Motors and estaff.
2. Disable automatic detections.
3. Based on the two connections created previously, create a new data foundation and add the tables REGION and CLIENT in the Motors.dbo owner and add the table Stores and Employee in the PUBLIC owner.
4. Join the following inserted tables using the specified cardinalities:
   - CLIENT.REGION_ID=REGION.REGION_ID (n,1)
   - REGION.REGION_NAME=Stores.Region (1,1)
   - Stores.Store Code=Employee.Store Code (1,n)
5. Create the multisource business layer.
6. Create the following folders:
   - Customer
   - Staff
7. Add the objects that describe the Customers.
8. Change the Client ID to a measure that counts the client ids and renames the object **`Number of Clients`**.

Hint:
To perform this task, choose *SQL Assistant*.

9. Add the objects that describe the Staff.
10. Change the Personnel Number to a measure that counts the personnel numbers and rename the object Number of Employees.
11. Save the business layer.
12. Test the results.

Unit 19
Solution 41

Create a Multisource Universe

Business Example

Your end users need to report on information about the number of customers and the number of employees in various regions. The information on the customers is stored in a SQL server database, but the information on the employees is stored in an Access database. You need to create a single universe that allows the end users to access the information in both data sources.

In the following exercise, you will use connections that are created on a SQL Server database and an Access database. In the next step, you will create a data foundation based on those two connections, and finally you will create a business layer on top of this data foundation.

Note:
In the data provided for you to complete the exercises. replace the value xx with the group number the instructor provided you with at the beginning of the course.

1. Create a project called CrossOver_xx and add shortcuts to two connections: Motors and estaff.
   a) Choose *New* → *Project* using CrossOver_xx as the project name.
   b) Choose *Finish*.
   c) Add a Repository session using the following login credentials:
      - System: wdflbmt2065
      - User name: train-##
      - Password train-##
      - Authentication Mode Enterprise
   d) Browse the connections folder and find the connection named Motors.
   e) Right-click the connection and choose *Create Relational Connection Shortcut*.
   f) From the pop-up, choose the CrossOver_xx project.
   g) When prompted to create a shortcut, choose *OK* and choose *Close*.
   h) Browse the connections folder and find the connection named staff.
   i) Right-click the connection and choose *Create Relational Connection Shortcut*.
   j) From the pop-up, choose the CrossOver_xx project.
   k) When prompted to create a shortcut, choose *OK* and *Close*.

2. Disable automatic detections.
   a) On the main menu bar, choose *Window* → *Preferences*.
   b) In the next window, choose *Information Design Tool* → *Data Foundation Editor* → *Detections*.

   > Note:
   > Make sure that both *Detect Keys* and *Detect Joins* are unchecked.

   c) Choose *OK* and close the *Preferences* dialog.
3. Based on the two connections created previously, create a new data foundation and add the tables REGION and CLIENT in the Motors.dbo owner and add the table Stores and Employee in the PUBLIC owner.
   a) Right-click the CrossOver_xx project and choose *New* → *Data Foundation*, naming the data foundation **`CrossOver_xx`**.
   b) Choose *Next*.
   c) Choose *Multi-source Enabled* and choose *Next*.
      You are prompted to select a BI 4.0 server session.
   d) Since you are already connected, choose *Next*.
   e) Next to each connection in your project, check the boxes and choose *Next*.
   f) Choose *Next* for the color option for each connection.
   g) Choose *Finish*.
   h) If not already open, to open the editor, in the *Local Projects* view, double-click on your data foundation.
   i) Right-click anywhere in the *Master* area and choose *Insert* → *Tables*.
   j) Expand the connection Motors and the Motors.dbo owner.
   k) Choose the REGION and CLIENT tables.
   l) Expand the connection staff and the PUBLIC owner.
   m) Place a check mark the Stores and Employee tables.
   n) Choose *Finish*.
4. Join the following inserted tables using the specified cardinalities:
   - CLIENT.REGION_ID=REGION.REGION_ID (n,1)
   - REGION.REGION_NAME=Stores.Region (1,1)
   - Stores.Store Code=Employee.Store Code (1,n)

   a) In the Master area, choose the + and choose *Insert Join*.
   b) Add the joins listed previously.

Note:
The join between the REGION and the Stores table is not logically correct as there are different values for the regions in each table. But, there are a few values that are the same across both databases, and you will look at those values.

5. Create the multisource business layer.
   a) Right-click your project and choose New ?Insert ? Business Layer ? Relational Data Source and choose Next.
   b) Name it **`CrossOver_xx`**.
   c) Choose *Next*.
   d) Choose the Data Foundation you created previously (CrossOver_xx.dfx).

Note:
Make sure you UNCHECK the box Automatically create classes and objects for all connections.

   e) Choose *Finish*.
6. Create the following folders:
   - Customer
   - Staff
   a) Choose the top level of the business layer.
   b) From the *Insert Object* drop down, choose *Folder*.
   c) Change the name to **`Customer`**.
   d) Repeat steps a-c for the other folder.
7. Add the objects that describe the Customers.
   a) Drag the following columns into the Customer folder you just created:
      - REGION.REGION_NAME
      - CLIENT.CLIENT_ID
8. Change the Client ID to a measure that counts the client ids and renames the object **`Number of Clients`**.

Hint:
To perform this task, choose *SQL Assistant*.

   a) Double-click the Client Id object.
   b) Change the SELECT statement to count(CLIENT.CLIENT_ID).

Be sure to validate the SQL.

c) Rename the object **`Number of Customers`** and qualify it as a measure by right-clicking the object and choosing *Turn into Measure with Aggregation Function* → *Sum*.

9. Add the objects that describe the Staff.
   a) Drag the following columns into the Staff folder you just created:
      - Employee.Personnel Number
10. Change the Personnel Number to a measure that counts the personnel numbers and rename the object Number of Employees.
    a) Double-click the Personnel Number object.
    b) Change the SELECT statement to `count(Employee.Personnel Number)` and validate the SQL.

Hint:
To perform this task, use the *SQL Assistant*.

    c) Rename the object Number of Employees and qualify it as a measure by right-clicking the object and choosing *Turn into measure(s)*.
11. Save the business layer.
    a) Save the business layer.
12. Test the results.
    a) On the bottom left of the *Business Layer* editor, choose the *Queries* panel.
    b) Choose *Insert Query*.
    c) In the query panel, drag the objects Region Name, Number of Clients and Number of Employees.
    d) Refresh the query and note the results for the East, South, and West regions.

Calculated Columns

Calculated columns are virtual columns of a table on the data foundation defined with custom SQL.

Calculated columns provide multiple advantages:

Advantages of Calculated Columns

- No derived tables

  Avoid the use of derived tables when not absolutely necessary.

- Shareability

  SQL expressions defined in tables of the data foundation can be shared in several business layers.

- Person focus

  Writing SQL is more of an IT skill than a user skill; business users could create business layers based on data foundations built by IT without the need of SQL knowledge.

Calculated columns can use only columns from the selected tables, along with native database SQL functions and built-in functions such as @Prompt or @Variable.

Unit 19
Exercise 42

Create Calculated Columns

Business Example

You know that many different business layers will need to provide a Client Full Name object. Instead of creating the same object in many business layers, you create a calculated column on the data foundation that concatenates two columns. As a result, any business layer using your data foundation can use the calculated column for an object definition.

1. In the Motors data foundation, add a calculated column to the CLIENT table that concatenates the CLIENT_LASTNAME and CLIENT_FIRSTNAME columns.
2. Save the data foundation.
3. In the Motors business layer, create and test an object in the Client folder called Client Full Name.

Unit 19
Solution 42

Create Calculated Columns

Business Example

You know that many different business layers will need to provide a Client Full Name object. Instead of creating the same object in many business layers, you create a calculated column on the data foundation that concatenates two columns. As a result, any business layer using your data foundation can use the calculated column for an object definition.

1. In the Motors data foundation, add a calculated column to the CLIENT table that concatenates the CLIENT_LASTNAME and CLIENT_FIRSTNAME columns.
   a) In the Motors data foundation, right-click the CLIENT table and from the context menu that appears, choose *Insert Calculated Column*.
   b) Name the column **`Client_Full_Name`**.
   c) In the SELECT editor, expand the CLIENT table and double-click CLIENT_LASTNAME.
   d) In the *Select statement*, type a + and then ' '+.

   > Note:
   > There is a space between the two single quotes.

   e) From the CLIENT table, double-click CLIENT_FIRSTNAME.
   f) Validate the SQL statement and make any corrections necessary.
   g) Choose *OK*.
   The calculated column appears as the last column in the CLIENT table.
2. Save the data foundation.
   a) Choose *Save*.
3. In the Motors business layer, create and test an object in the Client folder called Client Full Name.
   a) In the Motors business layer, drag the Client_Full_Name column from the CLIENT table into the Client folder.
   b) From the *Query* drawer, create a query using the Client Full Name and Number of Cars Sold objects.
   c) View the SQL script.
   d) Refresh the query and view the results.

Federated Tables

Federated Tables let you define complex relationships between multiple source tables in a simple and graphical manner. They enable you to satisfy user requests that may be too complex to define or too expensive to maintain with only derived tables.

Some of those requests may include:

Complex User Requests

- Partitioned Data
- Data Quality
- Optimized Data Synchronization
- Top-Down Models

Table 36: User Requests - Federated Tables

User Request	Description
Partitioned Data	A federated table can provide a single view on top of all partitions of data and can be set up to access only the necessary sources, depending on what is actually asked for at query runtime.
Data Quality	A federated table can transform the format of data from multiple sources to ensure consistency in the values that are returned from a query
Optimized Data Synchronization	A federated table can be used to synchronize data from multiple sources, defining that data should primarily be pulled from one source but use the other sources only when necessary.
Top-Down Models	Federated tables can be defined in a virtual model of the data but not connected to any data source. When data sources are added over time, the federated tables can then be connected as necessary, and the model begins to take shape. Additions of new sources do not modify the virtual model, and the newly added data is immediately available in the SAP BusinessObjects client tools and in existing reports.

Unit 19
Exercise 43

Use Federated Tables

Business Example

You need to create a multi-source universe using 2 different databases but combine date from each database into a single table on the data foundation. This new table will then be usable by any business layer that uses the data foundation.

Note:
To complete the exercise, replace the value *xx* with the group number the instructor provided you with at the beginning of the course.

Create Relational Connections to an SQL Server Database
Create a relational connection to an SQLserver database.

1. Create a new project called U##_ Federated_Data:
2. Create a relational connection shortcut to the Motors in your local project.

Field	Value
System	wdflbmt2065
Username	train-xx
Password	train-xx
Authentication	Enterprise

3. Create a relational connection shortcut to the efashion-webi in your local project.

Create a Federated Data Foundation
To create a federated data foundation:

1. Create a multisource data foundation with the resource name U##_Federated_DF and use the connection shortcuts Motors.cns and efashio-webi.cns in the data foundation.

Add Federated Tables and Data Flows
To add federated tables and data flows:

1. From MODEL_COLOURS choose motors and from Article_Color_Lookup choose efashion.webi and create a federated table based on the tables.
2. Customize the federated MODEL_COLOURS table by adding three fields named Color, Model_Name and Model_Price.

Column Name	Data Type
Color	VARCHAR

Column Name	Data Type
Model_Name	VARCHAR
Model_Price	DOUBLE

3. Add the MODEL and COLOUR tables and join them to the MODEL_COLOURS table.
4. Add the mapping to the federated table.

Hint:
To create the mapping for you automatically, you can also use *Automap Columns* on the toolbar.

Create an Additional Federated Table to Merge MODEL_COLOURS and Article_Color_Lookup Tables

To create an additional federated table to merge MODEL_COLOURS and Article_Color_Lookup tables:

1. Add a Federated table to merge MODEL_COLOURS and Article_Color_Lookup tables with the following information:

Column Name	Data Type
ID	VARCHAR
Article_Name	VARCHAR
Color	VARCHAR
Price	DOUBLE

2. Assign mapping to the U##_Motors_eFashion table.
3. View the values in the U##_Motors_eFashion table.

Unit 19
Solution 43

Use Federated Tables

Business Example

You need to create a multi-source universe using 2 different databases but combine date from each database into a single table on the data foundation. This new table will then be usable by any business layer that uses the data foundation.

> Note:
> To complete the exercise, replace the value *xx* with the group number the instructor provided you with at the beginning of the course.

Create Relational Connections to an SQL Server Database

Create a relational connection to an SQLserver database.

1. Create a new project called U##_ Federated_Data:
   a) Choose *File* → *New* → *Project*.
   b) Enter `U##_Federated_Data` and choose *Finish*.
2. Create a relational connection shortcut to the Motors in your local project.

Field	Value
System	wdflbmt2065
Username	train-xx
Password	train-xx
Authentication	Enterprise

   a) If you do not already have a session to the BI platform under *Repository Resources*, choose *Insert Session* and open the session with the information provided.
   b) Navigate to the Connections folder.
   c) Right-click your motors connection and choose *Create Relational Connection Shortcut*.
   d) In the *Local Project* window, choose the U##_Federated_Data project.
   e) Choose *OK*.
3. Create a relational connection shortcut to the efashion-webi in your local project.
   a) Under the same session, navigate to the RKT Connections folder.
   b) Right-click efashion-webi and choose *Create Relational Connection Shortcut*.
   c) In the *Local Project* window, select the U##_Federated_Data project.

d) Choose *OK*.
The efashion-webi.cns now appears under your local project.

Create a Federated Data Foundation
To create a federated data foundation:

1. Create a multisource data foundation with the resource name U##_Federated_DF and use the connection shortcuts Motors.cns and efashio-webi.cns in the data foundation.
   a) Choose the project U##_ Federated_Data and choose *File* → *New* → *Data Foundation*.
   b) As the *Resource Name*, enter **`U##_Federated_DF`** and choose *Next*.
   c) Choose *Multisource-Enabled* and choose *Next*.
   d) If there is no active CMS session, use the following data to log on:
      System: **`wdflbmt2065`**
      User name: **`train-xx`**
      Password: **`train-xx`**
      Authentication Mode: **`Enterprise`**
   e) Choose *Next*.
   f) Next to the selections Motors.cns and efashion-webi.cns, choose the checkboxes and choose *Next*.
   g) Leave *Short Names* or *Colors* for *Table Headers* as is.
   h) Choose *Finish*.

Add Federated Tables and Data Flows
To add federated tables and data flows:

1. From MODEL_COLOURS choose motors and from Article_Color_Lookup choose efashion.webi and create a federated table based on the tables.
   a) Choose the *Federation Layer* drawer and then choose the *Connections* drawer.
   b) From the Motors connection, choose MODEL_COLOURS and bring it to the *Federation Data Flow* window.
      An additional black MODEL_COLOURS is automatically created. This is the corresponding federated table.
   c) From the EFASHION-WEBI connection, choose Article_Color_Lookup and bring it to the *Federation Data Flow* window.
      An additional black Article_Color_Lookup table is automatically created. This is the corresponding federated table.
2. Customize the federated MODEL_COLOURS table by adding three fields named Color, Model_Name and Model_Price.

Column Name	Data Type
Color	VARCHAR
Model_Name	VARCHAR

Column Name	Data Type
Model_Price	DOUBLE

a) Right-click the black MODEL_COLOURS table and choose *Edit*.

b) To add the three column names provided and their respective data types, use the + button.

c) Choose *OK*.

3. Add the MODEL and COLOUR tables and join them to the MODEL_COLOURS table.
   a) Chooaw the black MODEL_COLOURS table.

   b) Under the MODEL_COLOURS properties pane, chooaw *Add* and choose *Insert Input Table*.

   c) Expand *MOTORS* → *Motors.dbo* and select the COLOUR table, and choose *OK*.

   d) Under the MODEL_COLOURS properties pane, choose *Add* and choose *Insert Input Table*.

   e) Expand *MOTORS* → *Motors.dbo* and chose the MODEL table, and choose *OK*.

   f) Click *Add* → *Add Join*.

   g) Change Table1 to MODEL_COLOURS and Table 2 to MODEL.

   h) Under the MODEL_COLOURS table, choose MODEL_ID and under the MODEL table, choose MODEL_ID, and choose *OK*.

   i) Choose *Add* → *Add Join*.

   j) Change Table1 to MODEL_COLOURS and Table 2 to COLOUR.

   k) Under the MODEL_COLOURS table, choose COLOR_ID and under the COLOUR table, choose COLOUR_ID. o) Click OK.

4. Add the mapping to the federated table.

Hint:
To create the mapping for you automatically, you can also use *Automap Columns* on the toolbar.

a) Draw a line from COLOUR.COLOUR to Color.

b) Draw a line from MODEL.MODEL_NAME to the column name Model_Name.

c) Draw a line from MODEL.MODEL_PRICE to the column name Model_Price.

Create an Additional Federated Table to Merge MODEL_COLOURS and Article_Color_Lookup Tables

To create an additional federated table to merge MODEL_COLOURS and Article_Color_Lookup tables:

1. Add a Federated table to merge MODEL_COLOURS and Article_Color_Lookup tables with the following information:

Column Name	Data Type
ID	VARCHAR
Article_Name	VARCHAR
Color	VARCHAR
Price	DOUBLE

a) Choose *Add Federated Tables* and name it **U##_Motors_eFashion**

b) Use the + button to add column names and types provided.

c) Click *OK*.

2. Assign mapping to the U##_Motors_eFashion table.
   a) Right-click the *Default Mapping* tab and choose *Edit Mapping*.
   b) Enter the name **Motors**.
   c) Under the *Motor* area, right-click on the white area and choose *Insert Input Tables*.
   d) Choose *Federation Layer* → *MODEL_COLOURS*.
   e) Draw the following links from the MODEL_COLOURS to the U##_Motors_efashion table:
      - MODEL_COLOURS.MODEL_ID to ID
      - MODEL_COLOURS.Color to Color
      - MODEL_COLOURS.MODEL_Name to Article_Name
      - MODEL_COLOURS.MODEL_Price to Price
   f) Choose *Add Mapping*.
   g) Enter the name **eFashion** and choose *OK*.
   h) Under the *eFashion* area, right-click on the white area and choose *Insert Input Tables*.
   i) Choose *Federation Layer* → *Article_Color_Lookup* and choose *OK*.
   j) Draw the following links from the Article_Color_Lookup to the U##_Motors_efashion table:
      - Article_Color_Lookup.Article_id to ID
      - Article_Color_Lookup.Article_label to Article_Name
      - Article_Color_Lookup.Color_label to Color
      - Article_Color_Lookup.Sale_price to Price
3. View the values in the U##_Motors_eFashion table.
   a) Right-click U##_Motors_eFashion and choose *Show TableValues*.
   The mapping values display in a window.

LESSON SUMMARY

You should now be able to:

- Create a multisource universe
- Use calculated columns
- Use federated tables

Learning Assessment

1. The Information Design Tool allows you to create a universe from a number of different data sources. Identify some of those data sources from the list.

   Choose the correct answers.

   - [ ] A XML metadata files
   - [ ] B ICE Cubes
   - [ ] C OLAP Cubes
   - [ ] D Stored Megadata files

2. Creating and maintaining a multisource universe is the same as creating a single-source dimensional universe. For a multisource universe, you simply use more than one connection for your data foundation.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

3. Identify the advantages of calculated columns from the list.

   Choose the correct answers.

   - [ ] A Refresh tables
   - [ ] B No derived tables
   - [ ] C Shareability
   - [ ] D Person focus

4. Federated Tables let you define complex relationships between multiple source tables in a simple and graphical manner.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

Unit 19

Learning Assessment - Answers

1. The Information Design Tool allows you to create a universe from a number of different data sources. Identify some of those data sources from the list.

   Choose the correct answers.

   - [X] A XML metadata files
   - [ ] B ICE Cubes
   - [X] C OLAP Cubes
   - [ ] D Stored Megadata files

2. Creating and maintaining a multisource universe is the same as creating a single-source dimensional universe. For a multisource universe, you simply use more than one connection for your data foundation.

   Determine whether this statement is true or false.

   - [X] True
   - [ ] False

3. Identify the advantages of calculated columns from the list.

   Choose the correct answers.

   - [ ] A Refresh tables
   - [X] B No derived tables
   - [X] C Shareability
   - [X] D Person focus

4. Federated Tables let you define complex relationships between multiple source tables in a simple and graphical manner.

   Determine whether this statement is true or false.

   [X] True

   [ ] False

UNIT 20 Shared Projects

UNIT OBJECTIVES

- Explain the role of shared projects
- Describe the purpose of project synchronization
- Update shared projects

Unit 20
Lesson 1

Using Shared Projects

LESSON OVERVIEW
You may choose to share the universe design responsibilities among a team of universe designers.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Explain the role of shared projects

Shared Projects

About Shared Projects

A shared project is a project in the SAP BusinessObjects Enterprise repository whose resources are available to other designers. You create a shared project in a repository from an existing local project in the Local Projects View. Use the Project Sychronization View to work on shared resources.

Project Synchronization View

With the Project Synchronization View, you can:

- Synchronize the project to copy resources between the local and shared projects
- Lock and unlock resources in the shared project to inform other designers when you are working on them
- Synchronize a shared project created by another designer. This creates a local project associated with the shared project so you can start working on the shared resources

LESSON SUMMARY
You should now be able to:

- Explain the role of shared projects

Unit 20
Lesson 2

Manipulating Other Designers' Resources

LESSON OVERVIEW

If you choose to share the universe design responsibilities among a team of universe designers, you will need to synchronize your project as well as update shared projects. This lesson discusses how you can create a use a shared project to make a local project's resources available to other designers in your organization.

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Describe the purpose of project synchronization
- Update shared projects

Project Synchronization

Synchronize a project when you want to:

- Update local resources with changes stored in the shared project.
- Save in the shared project the changes you made to local resources.
- Revert local resources to the copy stored in the shared project.
- Create a local copy of a shared project.

About Project Synchronization

Only shared projects can be synchronized.

Synchronizing a project starts with comparing the resources in a project in the Local Projects View with an associated shared project on the repository server. Synchronization detects added resources, deleted resources, and differences between the resources. Based on the differences detected, you can update the local and shared resources. Use the Project Synchronization View to synchronize a project. The view displays synchronization information in two panes:

- Shared Project pane lists the resources in the shared project on the server.

  A lock icon appears next to the resource if it is locked. Additional information about the resources on the server is displayed, including the user who last modified the resource and on what date, and the user who locked the resource and on what date.

- Synchronization Status pane lists the status of each resource.

  The status is determined by comparing the resources in the local and shared projects.

Synchronization Statuses

The following list itemizes the different sychronization statuses:

Status of Synchronization

The different synchronization statuses and what they mean are listed in the following table.

Table 37: Synchronized Statuses and Descriptions

Status	Description
Added Locally	The resource was added in the local project, but not in the shared project.
Changed Locally	The resource was changed in the local project, but not in the shared project since the last synchronization.
Deleted Locally	The resource was deleted in the local project but still exists in the shared project.
Added on Server	The resource is not in the local project but exists in the shared project.
Changed on Server	The resource was changed in the shared project, but not in the local project since the last synchronization.
Deleted on Server	The resource exists in the local project, but was deleted in the shared project.
Conflicting	Any of the following situations creates a conflicting status: • The resource was changed in both the local and shared projects with different changes since the last synchronization. • A resource with the same name was added both in the local and shared projects since the last synchronization. • The resource was changed in the local project, but deleted from the shared project. • The resource was changed in the shared project, but deleted from the local project.
Synchronized	The resources are identical.

Commands to Synchronize Resources

The following three commands allow you to synchronize resources. When you select the resources to be synchronized, you can select individual resources or folders. The Synchronize Resources Comands table summarizes the possible synchronization actions

- Get Changes from Server
- Save Changes on Server
- Revert Changes

Table 38: Synchronize Resources Commands

Command	Synchronization Action
Get Changes from Server	For the selected resources: • If the status is Added on Server, the resource is added to the local project. • If the status is Changed on Server, the resource is updated in the local project. • If the status is Deleted on Server, the resource is deleted from the local project. • If the status is Conflicting, the resource on the server (whether it is changed, added, or deleted) is copied to the local project, regardless of the change made in the local project. For all other statuses, no action is taken. Note: You may want to review the changes made on the server before updating them in the local project.
Save Changes on Server	For the selected resources: • If the status is Added Locally, the resource is added to the shared project on the server. • If the status is Changed Locally, the resource is updated in the shared project on the server. • If the status is Deleted Locally, the resource is deleted from the shared project on the server. • If the status is Conflicting, the resource in the local project (whether it is changed, added, or deleted) is copied to the shared

	project, regardless of the change made in the shared project.  Note: If a resource is locked by another user, an error message displays and the changes and deletions are not made on the server. For all other statuses, no action is taken.
Revert Changes	For the selected resources, the local project is updated with the shared project on the server, regardless of the status.  Note: Revert Changes updates the local project in the same way as Get Changes from Server. The exception is that if a resource is created in the local project and has not yet been saved on the server, Revert Changes deletes the new local resource, whereas Get Changes from Server preserves the new local resource.

Unit 20
Exercise 44

Synchronize a Project

Business Example

Your team shares some projects so different people can make updated as required.

> Note:
> To complete the exercises, replace the value xx with the group number the instructor provided you with at the beginning of the course.

1. Participant A creates a local project and a local connection for the project. Participant A then makes that project a shared project and synchronizes it to the local project.

Unit 20
Solution 44

Synchronize a Project

Business Example

Your team shares some projects so different people can make updated as required.

> Note:
> To complete the exercises, replace the value xx with the group number the instructor provided you with at the beginning of the course.

1. Participant A creates a local project and a local connection for the project. Participant A then makes that project a shared project and synchronizes it to the local project.
   a) Create a project named Shared Project Uxx and Uxx.
   b) In the project, create a local connection using the Motors data source.
   c) Right-click the project and choose *New Shared Project*.
   d) Enter the repository credentials, if needed.
   The *Project Synchronization* window opens with a project in the *Synchronization Status* pane and the *Shared Project* pane.

   > Note:
   > The *Synchronization Status* pane contains the components of the project, in this case only the connection, that other designers can create and share. The Added Locally status means that the project has not been synchronized with the shared project that is now on the server.

   e) Expand the project in the *Synchronization Status* pane and choose the connection (*.cnx).
   f) Choose the *Save Changes on Server*.
   The chosen item is copied to the shared project, and the status changes to Synchronized.

Shared Project Updating

Updating a shared project occurs when a user, different from the Shared Project designer, makes changes to the project. To ensure the original person can see the changes, the project must be synchronized. If the project is not synchronized, only the user who made the changes knows that there are revisions.

Unit 20
Exercise 45

Review and Merge Changes to a Shared Project

Business Example

Review and manually merge changes.

1. To add a data foundation to the project and synchronize the change with the project on the server, Participant B chooses *Project Synchronization*, or chooses *Window* → *Project Synchronization*.

Unit 20
Solution 45

Review and Merge Changes to a Shared Project

Business Example

Review and manually merge changes.

1. To add a data foundation to the project and synchronize the change with the project on the server, Participant B chooses *Project Synchronization*, or chooses *Window → Project Synchronization*.
   a) Choose *Project Synchronization*.
   b) To see the shared project, open a session.
   c) From the *Project* drop-down menu, choose the project.
   d) In the *Synchronization Status* pane, highlight the connection and choose *Get Changes from Server*.
   Notice that the shared project now appears in your local projects pane.
   e) Create a single-source data foundation with the COUNTY and REGION tables joined through the COUNTRY_ID columns.
   f) Save the data foundation.
   g) In the *Project Synchronization* window, choose *Refresh*.
   The data foundation appears with a status of Added Locally.
   h) Choose the data foundation and choose *Save Changes on Server*.
   The data foundation is copied into the *Shared Project* pane.

LESSON SUMMARY

You should now be able to:

- Describe the purpose of project synchronization
- Update shared projects

Unit 20

Learning Assessment

1. Which view to you use when working on shared resources?

   Choose the correct answer.

   - [ ] A Remote Synchronization View
   - [ ] B Project Synchronization View
   - [ ] C Distinct SSynchronization View
   - [ ] D Designer Synchronization View

2. You synchronize a project when you want to perform certain tasks. Identify some of those tasks from the list.

   Choose the correct answers.

   - [ ] A Create a local copy of a shared project
   - [ ] B Create numerous copies for the repository server
   - [ ] C Update local resources with changes stored in the shared project
   - [ ] D Delete resources stored in the Remote Projects View

3. Updating a shared project occurs when the one Project designer makes changes to the project.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

Learning Assessment - Answers

1. Which view to you use when working on shared resources?

   Choose the correct answer.

   - [ ] A Remote Synchronization View
   - [x] B Project Synchronization View
   - [ ] C Distinct SSynchronization View
   - [ ] D Designer Synchronization View

2. You synchronize a project when you want to perform certain tasks. Identify some of those tasks from the list.

   Choose the correct answers.

   - [x] A Create a local copy of a shared project
   - [ ] B Create numerous copies for the repository server
   - [x] C Update local resources with changes stored in the shared project
   - [ ] D Delete resources stored in the Remote Projects View

3. Updating a shared project occurs when the one Project designer makes changes to the project.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

UNIT 21 Universe Conversion

UNIT OBJECTIVES

- Convert published .unv universes

Unit 21
Lesson 1

Converting Existing .unv Universes

LESSON OVERVIEW
As a universe designer, you can leverage your existing universes. This lesson explains how to convert existing .unv universes.

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Convert published .unv universes

Types of Universes

Universe Types

A .unv universe refers to a universe created with any SAP Business Objects XI 3 universe design tool, for example, Universe Designer (now called the Universe Design Tool). A .unx universe is a universe published by the Information Design Tool.

Universe Conversion

Key Points about Converting Universes

When converting a .unv universe, that .unv universe is preserved. Documents in SAP BusinessObjects query and reporting tools based on the universe are still linked to the .unv universe. This gives you the opportunity to check and test the converted universe before changing the documents that depend on it.

Some features of the .unv universes are implemented differently in the .unx universe. Once you have converted a universe, you can edit the universe resources in a local project in the Information Design Tool to check and correct inconsistencies and to take advantage of new universe features.

After converting the universe, it is recommended that you refresh the structure of the data foundation and run a check integrity on the universe.

Hint:
Consult the *Information Design Tool User Guide* for a description of the supported features and how they are implemented in .unx universes, as well as for tips on resolving check integrity errors on converted universes.

To convert a .unv universe in a repository, the .unv universe to be converted must be stored in a repository compatible with the Information Design Tool. If the .unv universe was created with a design tool version earlier than SAP BusinessObjects BI 4.0, first you must upgrade the universe using the Upgrade Management Tool.

Hint:
For more information about upgrading universes, see the *SAP BusinessObjects Upgrade Guide*.

If you want to retrieve the converted .unx universe into a local project to work on it, you must first have a local project folder in the *Local Projects View*.

LESSON SUMMARY
You should now be able to:

- Convert published .unv universes

Unit 21

Learning Assessment

1. A .unv universe is a universe published by the Information Design Tool.

   Determine whether this statement is true or false.

   ☐ True

   ☐ False

Learning Assessment - Answers

1. A .unv universe is a universe published by the Information Design Tool.

   Determine whether this statement is true or false.

   ☐ True

   ☒ False

UNIT 22 Translation

Lesson 1

UNIT OBJECTIVES

- Deploy a translated universe

Unit 22
Lesson 1

Deploying Universes in Different Languages

LESSON OVERVIEW

One of the key features of SAP BusinessObjects BI 4.0 is the ability to produce multilingual metadata and reports from the same universe. This feature enables the end user to have a one-step multilingual reporting solution that is locale sensitive, supported by a single metadata universe model and provides full Unicode support. Reports then can be built once from the same universe and run and generated in several languages based on user preferences. The Information Design Tool user interface can also be displayed in different languages. This unit describes the multilingual universe features. In this lesson, you will use the Translation Management Tool to translate universe components into different languages.

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Deploy a translated universe

Translation Management

Translation Management Tool

The Translation Management Tool is a SAP BusinessObjects application that allows universe designers to translate the Data Foundation and the Business Layer of their universes into different languages. This translation ensures that the end users can read the Business Layer in their preferred language.

The Translation Management Tool is used for translating not only data foundations and business layers of universes but also end user reports and dashboards.

This application is useful for worldwide BusinessObjects deployments. Instead of duplicating universes and documents for each required language, you can use the Translation Management Tool to translate the metadata only once.

When a report designer creates a document and retrieves data using the universe that is translated, all translatable content is stored in several different languages in the single document.

The language chosen to display the document in the end user's application is dependent on the language defined in the user's preferred viewing locale setting in the BI Launchpad or in the Internationalization settings in the end user application.

If the Preferred Viewing Locale or Internationalization settings are set to a language that does not exist in the document, then the data is displayed using the default language. This default language value is defined when the universe data is translated.

Note:
The Translation Management Tool does **not** allow you to translate the data retrieved by the query. A multilingual database is required to display the report data in multiple languages.

Translation Management Tool allows you to translate the following text strings:

- Folder and object names and descriptions
- Business Layer name and descriptions
- Data Foundation names and descriptions
- Prompted query filter text
- Context names

Unit 22
Exercise 46

Translate a Universe

Business Example

You work for an international organization and need to deploy your universes worldwide. You want to avoid creating individual universes for each language and decide to use the Translation Management Tool to translate your universes.

> Note:
> To complete the exercises, replace the value *xx* with the group number the instructor provided you with at the beginning of the course.

1. Create a project called Translate_xx.
2. Retrieve the Motors universe from the repository.
3. Using the same logon credentials used to open a repository session in the Information Design Tool, launch the Translation Management Tool.
4. To translate from a local folder, import the strings.
5. As the language to translate to, choose *German (Germany)*.
6. Make the translations visible when used by other tools.

> Note:
> BE SURE TO PERFORM THIS STEP. Otherwise, your translation will not be visible when you test it later.

7. Using the information in the following table, translate the Client Name, Client Country, and Client Town objects.

Folder Object	German
Client (folder)	Kunde
Client Name	Name des Kunden
Client Country	Land
Client Town	Stadt

8. Export the translated strings.
9. To test the translations, create a query in the business layer of the universe.

Unit 22
Solution 46

Translate a Universe

Business Example

You work for an international organization and need to deploy your universes worldwide. You want to avoid creating individual universes for each language and decide to use the Translation Management Tool to translate your universes.

Note:
To complete the exercises, replace the value *xx* with the group number the instructor provided you with at the beginning of the course.

1. Create a project called Translate_xx.
   a) Choose *New* → *Project* and name the project **`Translate_xx`**.
   b) Choose *Finish*.
2. Retrieve the Motors universe from the repository.
   a) Right-click the project name and choose *Retrieve a Published Universe* → *From a Repository*.
   b) If necessary, using the following information, log on to the repository:
      - System: wdflbmt2065
      - User name: train-##
      - Password: train-##
      - Authentication Mode: Enterprise
   c) Navigate to *Universes* → *BOID* → *Solutions* → *Unit 5* → *Contexts* and choose the motors.unx.
   d) Choose *Finish*.
3. Using the same logon credentials used to open a repository session in the Information Design Tool, launch the Translation Management Tool.
   a) To launch the Translation Management Tool, choose *Start* → *All Programs* → *SAP Business Intelligence* → *SAP BusinessObjects BI Platform 4 Client Tools* → *Translation Management Tool*.
4. To translate from a local folder, import the strings.
   a) Choose *File* → *Import strings to translate from* and choose *Local* folder.

b) Choose *Browse to* → *businessobjects* → *bimodeler_14* → *workspace* → *Translate_xx* → *Retrieval (with today's date)* and choose the business layer motors.blx.

c) Choose *Open*.

5. As the language to translate to, choose *German (Germany)*.
   a) In the lower left portion of the application, from the *Language Management* window, double-click German (Germany).
6. Make the translations visible when used by other tools.

> Note:
> BE SURE TO PERFORM THIS STEP. Otherwise, your translation will not be visible when you test it later.

   a) Under the *Visible* column for *German*, check the box.
7. Using the information in the following table, translate the Client Name, Client Country, and Client Town objects.

Folder Object	German
Client (folder)	Kunde
Client Name	Name des Kunden
Client Country	Land
Client Town	Stadt

   a) In the upper left portion of the application, expand the Outline node and then the Client folder.
   b) Continue expanding the Client Name, Client Country, and Client town objects.
   c) In the *German* column, click on the italicized blue text for the Name field.
   d) In the upper-right portion of the tool, in the *Text Editor* window, in the *Translation* field, enter the translated text and then choose *Apply*.
   e) Save the translations.
8. Export the translated strings.
   a) Choose *File* → *Export translated strings*.
   The `Success` message appears.
   b) Choose *OK*.
9. To test the translations, create a query in the business layer of the universe.
   a) Go back to the Information Design Tool and for the *Preferred Viewing Locale*, change the *Languages preferences* to German (Germany) by choosing *Window* → *Preferences*.
   b) Choose *Languages* and In the *Preferred Viewing Locale* selection list, choose Germany (Germany).

c) Choose *OK*.

d) To refresh the Translate_xx project, right-click the project name and choose *Refresh*.

e) If it is not already open, open the business layer as if you are creating a query.

f) Open the Client folder and observe the translated objects.

LESSON SUMMARY

You should now be able to:

- Deploy a translated universe

Learning Assessment

1. What is the name of the SAP BusinessObjects application that allows universe designers to translate the Data Foundation and the Business Layer of their universes into different languages?

   Choose the correct answer.

   - ☐ A Translation Language Office
   - ☐ B Translate Project Application
   - ☐ C Translation Management Tool
   - ☐ D Project Transation Office

Unit 22

Learning Assessment - Answers

1. What is the name of the SAP BusinessObjects application that allows universe designers to translate the Data Foundation and the Business Layer of their universes into different languages?

   Choose the correct answer.

   - [ ] A Translation Language Office
   - [ ] B Translate Project Application
   - [x] C Translation Management Tool
   - [ ] D Project Transation Office