

**IBM Cognos Framework Manager:
Design Metadata Models (v10.2.2)**
Student Guide Volume 2
Course Code: B5A52

IBM Cognos Framework Manager: Design Metadata Models (v10.2.2)

B5A52

ERC: 1.0

Published January 2015

All files and material for this course, B5A52 IBM Cognos Framework Manager: Design Metadata Models, are IBM copyright property covered by the following copyright notice.

© Copyright IBM Corp. 2003, 2015

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM corp.

IBM, the IBM logo, ibm.com and IBM DB2 are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

P-2

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Contents

SPECIFYING DETERMINANTS	11-1
OBJECTIVES	11-3
FRAMEWORK MANAGER WORKFLOW	11-4
IBM COGNOS DETERMINANTS	11-5
EXAMPLE: TIME DIMENSION.....	11-6
SPECIFYING DETERMINANTS	11-7
DEMO 1: SPECIFY DETERMINANTS ON THE TIME DIMENSION	11-9
EXAMPLE: PRODUCTS DIMENSION.....	11-18
WORKSHOP 1: SPECIFY DETERMINANTS.....	11-19
SUMMARY	11-23
CREATING THE PRESENTATION VIEW	12-1
OBJECTIVES	12-3
FRAMEWORK MANAGER WORKFLOW	12-4
USING A PRESENTATION VIEW	12-5
POPULATING THE PRESENTATION VIEW	12-6
IDENTIFYING CONFORMED DIMENSIONS	12-7
DEMO 1: CREATE THE PRESENTATION VIEW	12-8
CREATE THE PRESENTATION VIEW	12-13
USING THE MODEL DESIGN ACCELERATOR	12-18
DEMO 2: RAPIDLY CREATE A MODEL USING THE MODEL DESIGN ACCELERATOR	12-19
SUMMARY	12-25

WORKING WITH DIFFERENT QUERY SUBJECT TYPES.....	13-1
OBJECTIVES	13-3
DATA SOURCE QUERY SUBJECTS	13-4
SETTING THE SQL TYPE	13-5
SQL TYPE: COGNOS SQL	13-6
SQL TYPE: NATIVE SQL	13-7
SQL TYPE: PASS-THROUGH SQL	13-8
IBM COGNOS QUERY GENERATION ARCHITECTURE.....	13-9
DEMO 1: CONFIGURE SQL TYPE SETTING	13-10
STORED PROCEDURE QUERY SUBJECTS	13-14
USING PROMPT VALUES	13-15
DEMO 3: CREATE AND TEST A DATA QUERY STORED PROCEDURE QUERY SUBJECT.....	13-16
DATA MODIFICATION STORED PROCEDURES	13-22
DEMO 3: CREATE AND TEST A DATA MODIFICATION STORED PROCEDURE QUERY SUBJECT	13-23
SUMMARY	13-27
SETTING SECURITY IN FRAMEWORK MANAGER	14-1
OBJECTIVES	14-3
FRAMEWORK MANAGER WORKFLOW	14-4
OVERVIEW OF IBM COGNOS SECURITY	14-5
APPLYING FRAMEWORK MANAGER SECURITY	14-6
ASSIGNING PERMISSIONS	14-7
ADDING SECURITY IN THE PUBLISH WIZARD	14-8
DEMO 1: SPECIFY PACKAGE ACCESS	14-9
SPECIFYING DATA SECURITY.....	14-15
DEMO 2: SPECIFY DATA SECURITY.....	14-16
SPECIFYING OBJECT SECURITY.....	14-20
OBJECT SECURITY RULES	14-21
OBJECT SECURITY METHODOLOGIES	14-22
DEMO 3: SPECIFY OBJECT SECURITY.....	14-23
CREATING DYNAMIC DATA SECURITY	14-28
DEMO 4: USE A MACRO IN A DATA SECURITY FILTER	14-29
DEMO 5: REMOVE SECURITY	14-34
SUMMARY	14-37

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

CREATING ANALYSIS OBJECTS.....	15-1
OBJECTIVES	15-3
FRAMEWORK MANAGER WORKFLOW	15-4
DIMENSIONALLY MODELED RELATIONAL (DMR) METADATA.....	15-5
REVIEW OLAP DATA STRUCTURES.....	15-6
WHAT DO REPORT AUTHORS SEE?.....	15-7
DECIDE ON A MODELING STYLE	15-8
EXAMINE REGULAR DIMENSIONS	15-9
DETERMINANTS AND REGULAR DIMENSIONS.....	15-10
WORKING WITH MEMBERS	15-11
DEMO 1: CREATE REGULAR DIMENSIONS	15-12
WORKSHOP 1: CREATE A REGULAR DIMENSION	15-18
DEFINING MEASURE DIMENSIONS	15-21
DEFINING SCOPE RELATIONSHIPS.....	15-22
EDITING DMR METADATA	15-23
DEMO 2: CREATE MEASURE DIMENSIONS, SET SCOPE, AND CREATE A PRESENTATION VIEW	15-24
SORTING COMPATIBLE QUERY MODE (CQM) PROJECTS	15-34
DEMO 3: METADATA TREE AND MEMBER SORTING.....	15-35
WHAT IS A MEMBER UNIQUE NAME (MUN)?	15-41
CHANGES THAT IMPACT A MUN	15-42
DEMO 4: IDENTIFY HOW CHANGES TO MUNS IMPACT REPORTS.....	15-43
DIMENSIONAL MODELING CONSIDERATIONS	15-47
SUMMARY	15-48
MANAGING OLAP DATA SOURCES	16-1
OBJECTIVES	16-3
OLAP DATA SOURCES IN IBM COGNOS	16-4
USING OLAP DATA SOURCES	16-5
DEMO 1: IMPORT AND PUBLISH AN OLAP DATA SOURCE	16-6
DEMO 2: IMPORT AND PUBLISH MULTIPLE OLAP DATA SOURCES.....	16-9
SUMMARY	16-13

ADVANCED GENERATED SQL CONCEPTS AND COMPLEX QUERIES.....	17-1
OBJECTIVES	17-3
EXPLORE SQL GENERATION.....	17-4
GOVERNORS THAT AFFECT SQL GENERATION	17-5
USING DERIVED TABLES.....	17-6
IDENTIFYING STITCH QUERY SQL.....	17-8
WHAT IS A COALESCE FUNCTION?	17-9
NON-CONFORMED DIMENSIONS IN GENERATED SQL.....	17-10
WHAT IS AN RSUM(1.. ASC LOCAL) AS SC?.....	17-11
WHY DO I SEE XSUM	17-13
DEMO 1: IDENTIFY STITCH QUERIES IN GENERATED SQL	17-14
WORKSHOP 1: REVERSE ENGINEER A FRAMEWORK MANAGER MODEL FROM GENERATED COGNOS SQL	17-25
COGNOS SQL IN REPORT STUDIO	17-33
EXTENDED VS. RUNNING AGGREGATES	17-35
DEMO 2: EXAMINE GENERATED SQL IN REPORT STUDIO.....	17-36
GENERATED SQL FOR DMR METADATA	17-42
DEMO 3: EXAMINE GENERATED SQL FOR DIMENSIONALLY MODELED RELATIONAL METATDATA	17-43
USING CROSS-PRODUCT JOINS	17-46
CROSS-PRODUCT JOIN SQL	17-47
MULTI-FACT QUERY RESULTS	17-48
CONTIGUOUS RESULT SETS	17-49
TROUBLESHOOTING UNEXPECTED RESULTS	17-51
SUMMARY	17-52

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

USING ADVANCED PARAMETERIZATION TECHNIQUES.....	18-1
OBJECTIVES	18-3
IBM COGNOS SESSION PARAMETERS	18-4
ENVIRONMENT SESSION PARAMETERS	18-5
USING ENVIRONMENT SESSION PARAMETERS.....	18-6
MODEL SESSION PARAMETERS	18-7
FILTERING DATA BY PACKAGE ACCESS	18-8
CUSTOM ENVIRONMENT SESSION PARAMETERS	18-9
DEMO 1: USE CUSTOM ENVIRONMENT SESSION PARAMETERS	18-10
USING SESSION PARAMETERS.....	18-13
DEMO 2: DYNAMICALLY CHANGE THE DATA SOURCE CONNECTION	18-14
USING PROMPT MACROS	18-17
USING THE PROMPT MACRO FUNCTION	18-18
USING PRECEDING TEXT IN A PROMPT	18-19
DEMO 3: CREATE PROMPT MACROS TO FILTER DATA	18-20
SECURITY MACRO FUNCTIONS	18-23
CSVIDENTITYNAME	18-24
CSVIDENTITYNAMELIST.....	18-25
DEMO 4: LEVERAGE A MACRO FUNCTION ASSOCIATED WITH SECURITY ...	18-26
WORKSHOP 1: MAKE THE SECURITY INFORMATION CALCULATION	
DYNAMIC	18-31
SUMMARY	18-33
MODEL MAINTENANCE AND EXTENSIBILITY	19-1
OBJECTIVES	19-3
MANAGING PROJECTS	19-4
DEPLOYING PACKAGES AND CONTENT.....	19-5
ANALYZE PUBLISH IMPACT	19-6
DEMO 1: PERFORM IMPACT ANALYSIS ON A MODIFIED PACKAGE.....	19-7
REMAPPING AN OBJECT TO A NEW SOURCE.....	19-11
DEMO 2: REMAP THE PHYSICAL LAYER	19-12
LINKING MULTIPLE DATA SOURCES.....	19-15
NON-DATABASE DATA SOURCE	19-16
PLAYING BACK PARTS OF AN ACTION LOG	19-17

SYNCHRONIZING PROJECTS	19-18
CHECKING A PROJECT	19-19
DEMO 3: RUN A SCRIPT TO REPLAY ACTIONS	19-20
CREATING A MODEL REPORT	19-23
SUMMARY	19-24
OPTIMIZE AND TUNE FRAMEWORK MANAGER MODELS.....	20-1
OBJECTIVES	20-3
MATERIALIZED VIEWS.....	20-4
ACHIEVING MINIMIZED SQL	20-5
USING DYNAMIC QUERY MODE (DQM).....	20-6
USING DIMENSIONAL METADATA WITH DQM	20-7
MEMBER SORTING IN DQM.....	20-8
CREATING A DQM ENABLED PROJECT	20-9
DEMO 1: CREATE A DYNAMIC QUERY MODE ENABLED PROJECT.....	20-10
USING DQM FEATURES IN A COMPATIBLE MODE PROJECT	20-16
TESTING AND PUBLISHING CQM PROJECTS IN DQM	20-17
MIGRATING AN EXISTING PROJECT TO DQM	20-18
USING GOVERNORS TO SET LIMITS ON QUERY EXECUTION.....	20-19
DYNAMIC QUERY MODE GOVERNORS	20-20
METADATA CACHING	20-21
HOW DO REPORTS USE CACHED METADATA?	20-22
WHEN IS METADATA NOT CACHED?	20-23
SECURITY AWARE CACHING IN DQM	20-24
DEMO 2: IDENTIFY THE USE OF CACHED METADATA BY A QUERY	20-25
QUERY PROCESSING TYPES	20-27
DETERMINE WHERE AGGREGATES ARE CALCULATED	20-28
DEMO 3: EXAMINE ROLLUP PROCESSING AND GENERATED SQL	20-29
USING FILTERS TO IMPROVE PERFORMANCE.....	20-32
APPLYING DESIGN MODE FILTERS	20-33
DEMO 4: APPLY DESIGN MODE FILTERS	20-34
REDUCING DATABASE CONNECTIONS	20-37
INDICATING THE PERFORMANCE IMPACT OF FUNCTIONS	20-38
SUMMARY	20-39

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

WORKING IN A MULTI-MODELER ENVIRONMENT.....	21-1
OBJECTIVES	21-3
CREATING A SEGMENT.....	21-4
SEGMENTING AND LINKING EXAMPLE.....	21-5
DEMO 1: CREATE A SEGMENT.....	21-7
CREATING A LINK	21-12
DEMO 2: CREATE A LINK	21-13
CONSOLIDATING LINKED SEGMENTS.....	21-16
DEMO 3: CONSOLIDATE LINKED SEGMENTS.....	21-17
BRANCHING A MODEL	21-19
DEMO 4: BRANCH A MODEL, MAKE CHANGES AND MERGE RESULTS.....	21-20
SUMMARY	21-23
MANAGING FRAMEWORK MANAGER PACKAGES.....	22-1
OBJECTIVES	22-3
WHAT IS A FRAMEWORK MANAGER PACKAGE?	22-4
CREATE AND MODIFY PACKAGES.....	22-5
LANGUAGES AND FUNCTION SETS.....	22-6
PUBLISHING PACKAGES	22-7
SET MODEL VERSION CONTROL	22-8
DEMO 1: CONTROL MODEL VERSIONS	22-9
NEST PACKAGES	22-16
DEMO 2: NEST PACKAGES	22-17
SUMMARY	22-19
APPENDIX A: ADDITIONAL FRAMEWORK MANAGER MODELING TECHNIQUES	A-1
OBJECTIVES	A-3
LEVERAGING USER-DEFINED FUNCTIONS	A-4
REGULAR AGGREGATE PROPERTY	A-5
ORDER OF OPERATIONS FOR MODEL CALCULATIONS	A-6
DEMO 1: SET ORDER OF OPERATIONS FOR A MODEL CALCULATION	A-7
IBM COGNOS AS A TRANSFORMER DATA SOURCE	A-11
CREATING QUERY SETS	A-12
USING SOURCE CONTROL	A-13
SUMMARY	A-15

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

P-9

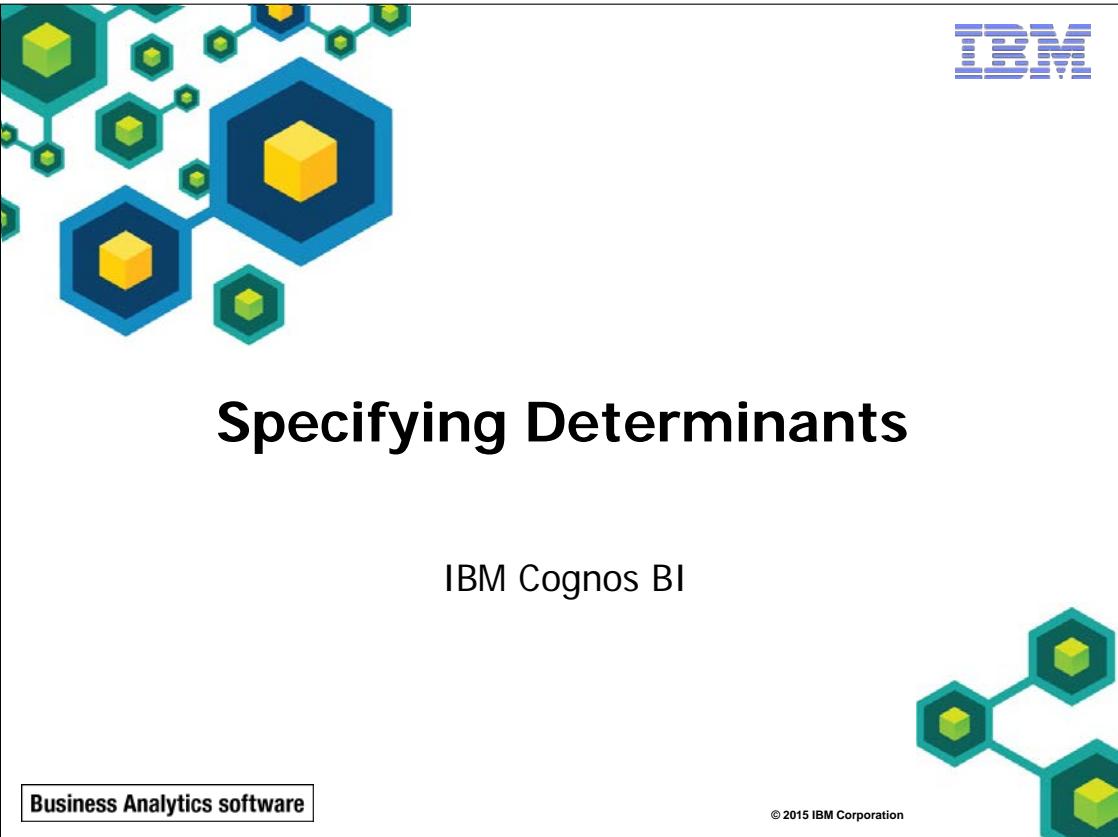
MODELING MULTILINGUAL METADATA.....	B-1
OBJECTIVES	B-3
FRAMEWORK MANAGER WORKFLOW	B-4
MODIFYING LANGUAGE PROPERTIES	B-5
DEMO 1: APPLY MULTILINGUAL QUERY ITEM PROPERTIES.....	B-6
DEMO 2: APPLY MULTILINGUAL TRANSLATION FILES.....	B-10
SUMMARY	B-13

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

P-10

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



The slide features a white background with a decorative header graphic in the top-left corner consisting of blue and green hexagonal nodes connected by thin lines, resembling a molecular or network structure. In the top-right corner is the classic blue IBM logo. Below the header, the main title "Specifying Determinants" is centered in a large, bold, black sans-serif font. Underneath the title, the subtitle "IBM Cognos BI" is displayed in a smaller, regular black font. At the bottom left, a white rectangular box contains the text "Business Analytics software". On the right side, there is a small graphic of three interconnected hexagonal nodes, each containing a yellow cube, and a copyright notice "© 2015 IBM Corporation".

Specifying Determinants

IBM Cognos BI

Business Analytics software

© 2015 IBM Corporation

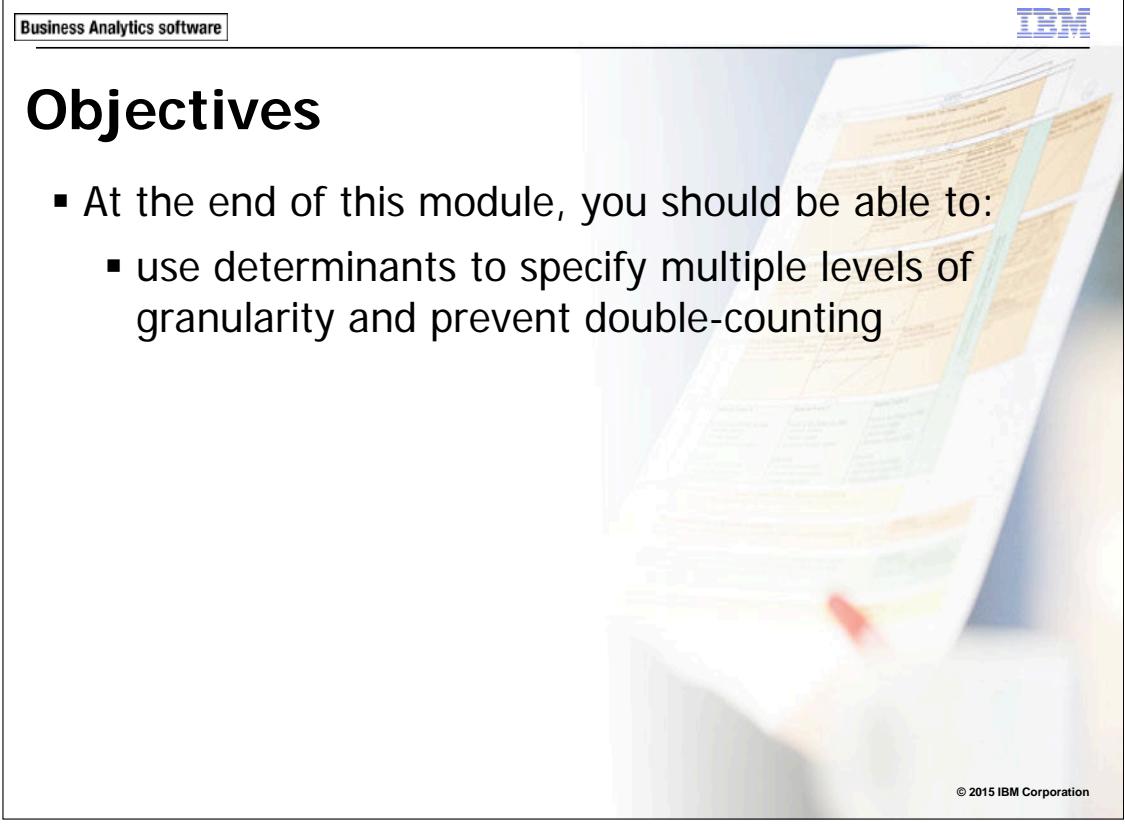
This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Objectives

- At the end of this module, you should be able to:
 - use determinants to specify multiple levels of granularity and prevent double-counting



© 2015 IBM Corporation

Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

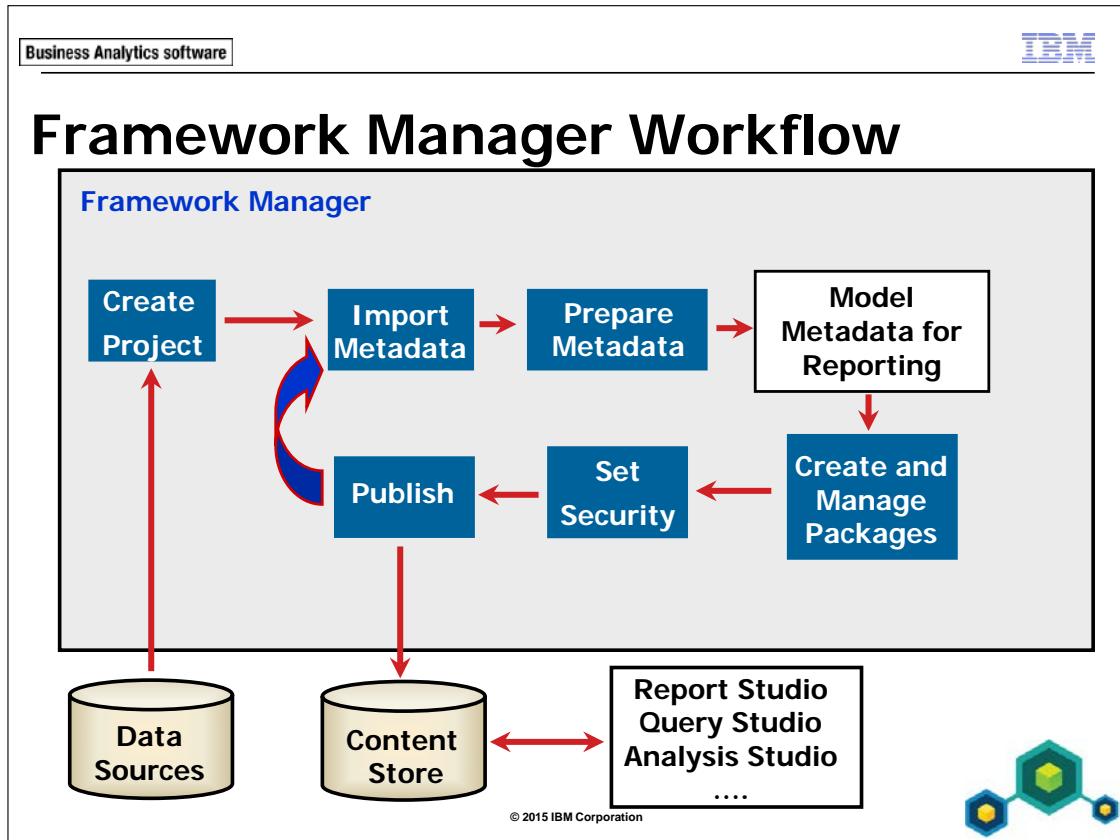
- User ID: admin
- Password: Education1

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

11-3



This module teaches the technique of specifying determinants to prevent double-counting.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

IBM Cognos Determinants

- use determinants to control granularity when aggregating
- required for dimensions connected to facts at levels of granularity that have repeating keys
- determinants are also used for:
 - preventing distinct clauses on unique keys
 - BLOB data types in the query subject

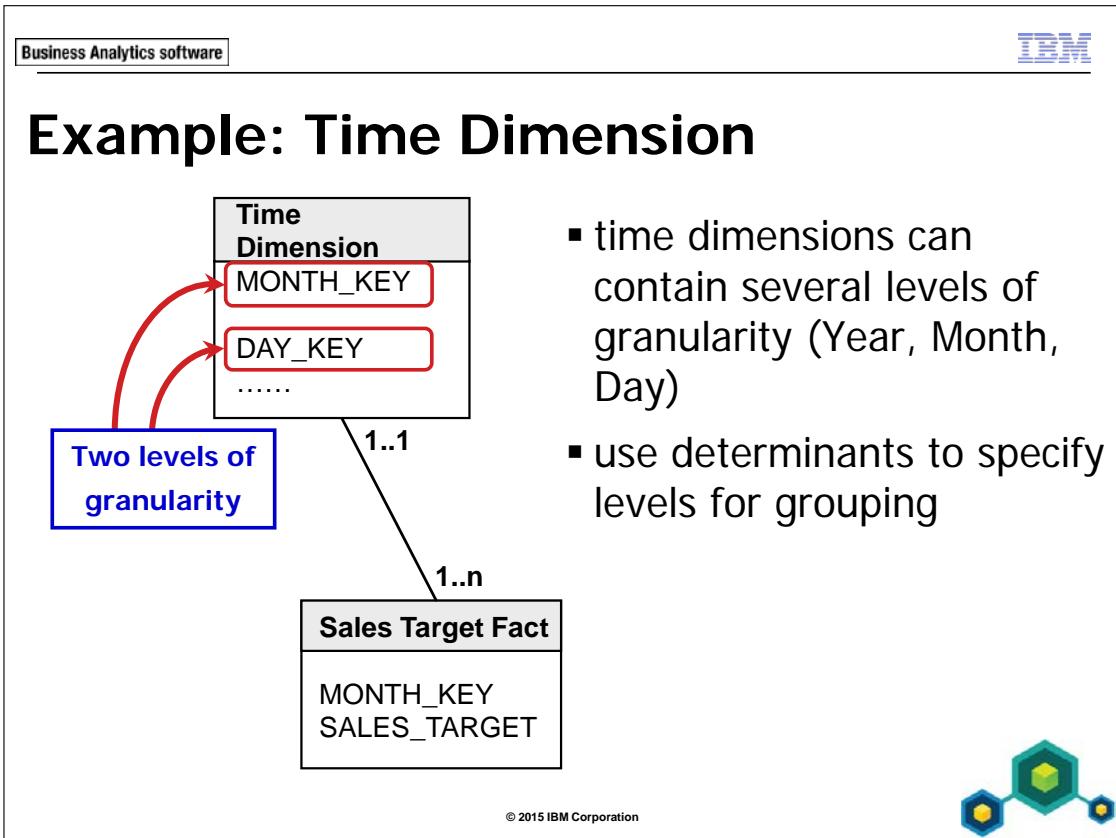
© 2015 IBM Corporation



Typically, determinants are specified to allow IBM Cognos to correctly aggregate facts in queries. Some designers also specify determinants proactively on any query subject with multiple levels of granularity, since you may link a fact at one of those levels of granularity in the future. Determinants are linked to the relationships their keys represent and are only used when the corresponding relationship is used in a query.

Determinants on unique data values (such as surrogate keys or dates), can prevent the generation of the distinct clause in the select statement when summarizing the data. Selecting distinct values unnecessarily can reduce performance. If the values are truly unique, then there is no need to scan for distinct values.

Querying Binary Large Objects (BLOBs) require additional key and index type information. If this information is not present in the data source, you can add it using determinants. To leverage indexes with DMR (especially when filtering on captions) associating your captions with the correct key in a determinant will improve query generation.



In the slide example above, each unique instance of MONTH_KEY repeats once for every day in the month.

If you do not tell IBM Cognos that MONTH_KEY requires grouping at the month level, your queries will double-count SALES_TARGET for every day of the month. Setting a determinant at each level of granularity avoids any potential instances of double-counting.

Specifying Determinants (1 of 2)

- Data Set Example #1

Data

Year Key	Month Key	Month Name	Day Key	Day Name
2012	201201	Jan 12	20120101	Sunday, Jan 1, 2012
2012	201201	Jan 12	20120102	Monday, Jan 2, 2012

Determinant Settings



Name	Key	Attributes	Uniquely Identified	Group By
Year	Year Key	None	No	Yes
Month	Month Key	Month Name	No	Yes
Day	Day Key	Day Name Month Key Month Name Year Key	Yes	No

© 2015 IBM Corporation



In the above data set, you can define two non-unique determinants (based on Year Key and Month Key), and one unique determinant (based on Day Key).

Day Key is the unique key of the table; therefore you can associate all the columns in the table to this key. Because it is a unique key at the lowest level of granularity, you check the Uniquely Identified setting and do not check the Group By setting.

Month Key is not unique, since the same value is used for every day in the month. To query by month, you need to use Select Min, and Group By the Month Key, which is why the Group By box is checked for the determinant setting. Similar logic is applied to the Year determinant.

When a query uses a column from the table above, the IBM Cognos query engine evaluates the determinants one at a time until it finds the column reference (either the key or an attribute of the key) in a determinant. For example, if you query Month Name and Sales Target (Sales Target has a relationship to the Time dimension on the Month Key), IBM Cognos would evaluate the Year determinant and determine that there is no reference to Month Name. It would then move on to the Month determinant, find the Month Name reference, and stop evaluating. It would then generate the appropriate SQL and group on the Month Key.

Business Analytics software

IBM

Specifying Determinants (2 of 2)

- Data Set Example #2

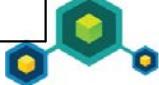
Data

Year Key	Month Key	Month Name	Day Key	Day Name
2012	01	January	20120101	Sunday, Jan 1, 2012
2012	01	January	20120102	Monday, Jan 2, 2012

Determinant Settings

Name	Key	Attributes	Uniquely Identified	Group By
Year	Year Key	None	No	Yes
Month	Year Key, Month Key	Month Name	No	Yes
Day	Day Key	Day Name Month Key Month Name Year Key	Yes	No

© 2015 IBM Corporation



In the event a key requires another key to provide uniqueness to identify the row of data, you need to nest the keys. For example, looking at the Month Key above, you cannot identify which year it belongs to without looking at the Year Key. Therefore, the Month determinant nests the Year Key and the Month Key to generate the proper grouping in the SQL at run time. In this case the grouping would combine Year Key and Month Key.

Demo 1: Specify Determinants on the Time Dimension

Purpose:

Report consumers require a report that compares the monthly sales targets to actual sales. In a previous test you discovered that sales target values are being double-counted. In this demo, you will specify determinants on TIME_DIMENSION to correct this problem.

Components: **Framework Manager, Query Studio**

Project: **GO Operational**

Package: **GO Operational**

Task 1. Review double-counting issue.

1. In **Framework Manager**, open the **GO Operational** project located at **C:\Edcognos\B5A52\CBIFM-Start Files\Module 11\GO Operational**. If prompted, log in as User ID **admin**, and Password **Education1**.
2. In **IBM Cognos Connection**, navigate to **Public Folders > GO Operational**, and then click **Test Time Dimension**.

The results appear as follows:

Year	Month (numeric)	Sales Target
2010	1	1,786,471,100
2010	2	1,898,274,000
2010	3	2,161,992,700
2010	4	1,798,641,000
2010	5	2,093,302,900
2010	6	2,207,847,000
2010	7	2,300,395,300
2010	8	2,203,015,000
2010	9	1,979,466,000
2010	10	2,079,077,000

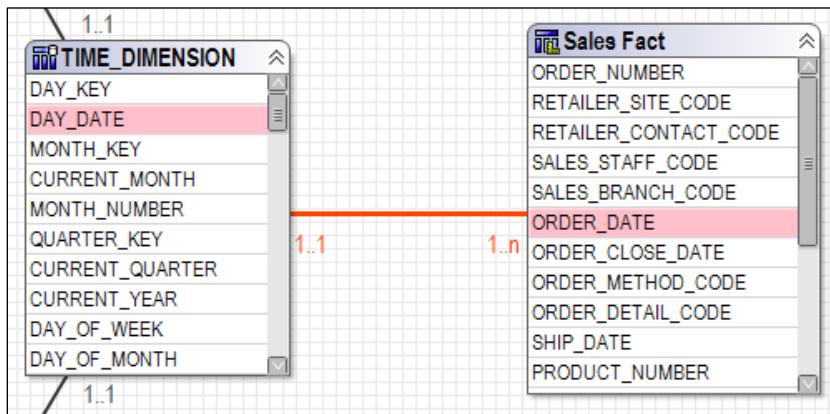
The monthly sales target values are in the billions, which is too high. This is due to the revenue value being counted once for every day in a month, rather than just once per month. You will specify determinants for the underlying TIME_DIMENSION query subject to resolve this issue.

3. Click **Return** , in the top right corner.

Task 2. Examine keys and unique values in TIME_DIMENSION.

1. In **Framework Manager**, in **Foundation Objects View > gosales**, right-click **TIME_DIMENSION**.
2. Click **Launch Context Explorer**, and then click **Show Related Objects**.
3. Click the relationship between **TIME_DIMENSION** and **Sales Fact**.

The results appear as follows:



When you specify determinants, it is good practice to view the relationships to the object and examine the keys used in the relationships. In this case **DAY_DATE**, not **DAY_KEY**, relates **TIME_DIMENSION** to both Sales Fact and Returns fact. This is an important piece of information for your determinant at the day level.

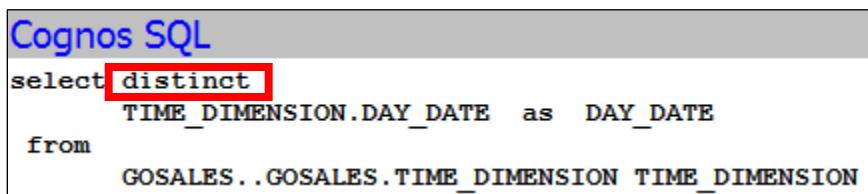
4. Click the relationship between **TIME_DIMENSION** and **SALES_TARGET**.

This relationship is based on **CURRENT_YEAR** and **CURRENT_MONTH**. Before applying determinants on **TIME_DIMENSION** you will also quickly test effects of having no determinants on a unique value in the data. In this case **DAY_DATE** is unique.

5. Close **Context Explorer**, expand **TIME_DIMENSION** and then test the **DAY_DATE** query item with **Auto Sum** enabled.

- Click the **Query Information** tab.

The results appear as follows:



```
Cognos SQL
select distinct
    TIME_DIMENSION.DAY_DATE as DAY_DATE
from
    GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION
```

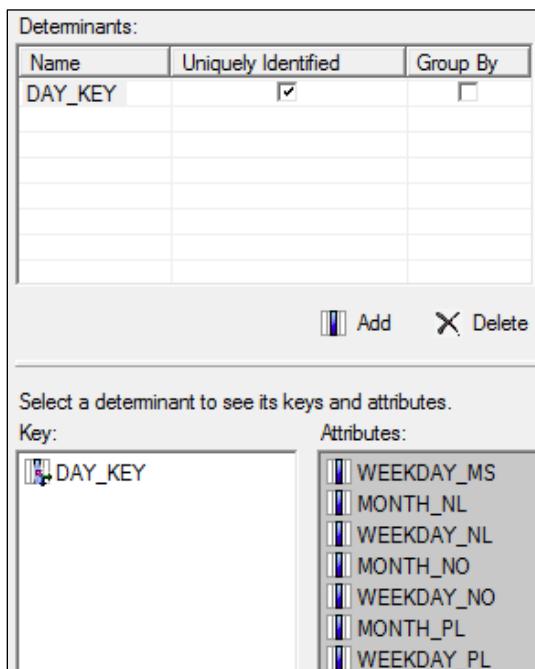
The distinct clause in the select statement is unnecessary, since the DAY_DATE values in this table are all unique. When you specify determinants, the distinct clause will not be applied when querying DAY_DATE.

- Click **Close**.

Task 3. Specify determinants in TIME_DIMENSION.

- In the **Project Viewer**, double-click **TIME_DIMENSION**, and then click the **Determinants** tab.

The results appear as follows:



Determinants:		
Name	Uniquely Identified	Group By
DAY_KEY	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Add Delete

Select a determinant to see its keys and attributes.

Key: Attributes:

- DAY_KEY
- WEEKDAY_MS
- MONTH_NL
- WEEKDAY_NL
- MONTH_NO
- WEEKDAY_NO
- MONTH_PL
- WEEKDAY_PL

During import, the primary key value is used to create a determinant for you. This value is unique and therefore all other values in the table can be associated with it. However, the relationships to Sales Fact and Returns Fact use DAY_DATE rather than DAY_KEY. You will alter this determinant to reflect these relationships.

- In the **Key** pane, select **DAY_KEY**, and then press **Delete**.

3. Under **Available items**, drag **DAY_DATE** to the **Key** pane.
You will now add the remaining determinants.
4. Under the **Determinants** pane, click **Add**.
New Determinant appears below **DAY_KEY** in the Determinants pane:

Determinants:

Name	Uniquely Identified	Group By
DAY_KEY	<input checked="" type="checkbox"/>	<input type="checkbox"/>
New Determinant	<input type="checkbox"/>	<input type="checkbox"/>

Add Delete

5. Right-click **New Determinant**, and then click **Rename**.
6. Type **Year**, and then press **Enter**.
7. Click the **Up Arrow** key on the right to move **Year** above **DAY_KEY**.
8. From the **Available items** pane, drag **CURRENT_YEAR** to the **Key** pane.
9. Select the **Group By** check box beside **Year**.
10. Repeat steps **4** to **9** to create a determinant named **Quarter**, with the **Key** set to **QUARTER_KEY**.
11. With the focus still on **Quarter**, drag **CURRENT_QUARTER** into the **Attributes** box.
12. Repeat steps **4** to **9** to create a determinant named **Month**, with two Keys: **CURRENT_MONTH** and **CURRENT_YEAR**.

These query items are used as the key for this determinant because the relationship to **SALES_TARGET** is based on them. These two fields uniquely identify the relationship to **SALES_TARGET** and therefore will be used to correctly aggregate sales target values.

13. With the focus still on **Month**, use the Ctrl key to select **MONTH_KEY** and all **MONTH_xx** items, and then right-click one of the items, and click **Add as Attributes**.

Typically **MONTH_KEY** would be used as the key for a determinant describing month data values. In this case however, **MONTH_KEY** is not used in any relationships, and therefore can act as an attribute of the Month determinant, so that it is available if it is used in a report. If you use this key in a relationship in the future, you will need to create a new determinant to represent that relationship. Ideally, you would use a month key exclusively and consistently across all facts that report at the month level. You can request these types of keys from the database administrator if they do not exist.

14. Rename the **DAY_KEY** determinant to **Day**.

The results appear as follows:

Determinants:		
Name	Uniquely Identified	Group By
Year	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Quarter	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Month	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Day	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Add Delete

Select a determinant to see its keys and attributes.

Key: DAY_DATE Attributes:

- MONTH_TC
- WEEKDAY_TC
- MONTH_TH
- WEEKDAY_TH

15. Click **OK**, and then save the project.

Task 4. Re-test DAY_DATE.

Determinants and relationships work together. If a query uses Date (attribute of the Day determinant) with Revenue from Sales Fact (which is at the day level), Date will not be grouped, since the determinant is specified as unique. However, if you query Month (attribute of the Month determinant) and Sales Target (which is at the month level), the query will be grouped by CURRENT_YEAR and CURRENT_MONTH, as specified by the Month determinant

- Under **TIME_DIMENSION**, test **DAY_DATE** with **Auto Sum** enabled, and then click the **Query Information** tab.

The results appear as follows:

```
Cognos SQL
select
    TIME_DIMENSION.DAY_DATE  as  DAY_DATE
  from
    GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION
```

There is no longer a distinct clause in your select statement because the DAY_DATE key is identified as unique; therefore the IBM Cognos query engine will not scan the table for distinct values. If you need another relationship later on using the DAY_KEY, you would also specify a determinant on it and set it as unique. You can have more than one unique determinant if they are truly unique. At query time, the relationship being used will determine which unique determinant to use. If more than one relationship to a unique determinant is used, then each related determinant will be used in the query.

- Click **Close**.

Task 5. Publish and test the package.

- Publish the **GO Operational** package.
- In **IBM Cognos Connection**, click **Test Time Dimension** to run the report again.

3. When prompted to update the report, click **OK**.
A section of the results appear as follows:

Year	Month (numeric)	Sales Target
2010	1	57,628,100
2010	2	67,795,500
2010	3	69,741,700
2010	4	59,954,700
2010	5	67,525,900
2010	6	73,594,900

You now have accurate sales targets.

4. Under **Menu**, click **Insert Data**.
5. From the **Consolidation View**, add **Time > Date**, and **Sales Fact > Revenue** to the report.
6. Ctrl-click **Year** and **Month (numeric)**, and then click **Group** .

The results appear as follows:

Year	Month (numeric)	Sales Target	Date	Revenue
2010	1	57,628,100	Jan 12, 2010 12:00:00 AM	\$39,036,796.40
		57,628,100	Jan 13, 2010 12:00:00 AM	\$11,488,458.59
		57,628,100	Jan 14, 2010 12:00:00 AM	\$3,232,160.48
		57,628,100	Jan 15, 2010 12:00:00 AM	\$3,080,441.44
		57,628,100	Jan 16, 2010 12:00:00 AM	\$1,976,896.69
		57,628,100	Jan 19, 2010 12:00:00 AM	\$2,159,831.74
		57,628,100	Jan 20, 2010 12:00:00 AM	\$2,558,694.30
		57,628,100	Jan 21, 2010 12:00:00 AM	\$2,935,024.72
		57,628,100	Jan 22, 2010 12:00:00 AM	\$3,559,670.33
		57,628,100	Jan 23, 2010 12:00:00 AM	\$2,371,064.81
		57,628,100	Jan 24, 2010 12:00:00 AM	\$342,583.15
		1	57,628,100	\$72,741,622.65

Although the sales target values repeat (because they are not at the day level) they are not double counted at the Month grouping level.

7. Select the **Year** column, and then click **Filter**.
8. Set the **From** and **To** boxes to **2010**, and then click **OK**.
9. Select **Month (numeric)** column, and then click **Filter**.
10. Set the **From** and **To** boxes to **1**, and then click **OK**.

11. At the bottom of the window, click **Apply**.

The results appear as follows:

 Year: 2010 AND Month (numeric): 1				
Year	Month (numeric)	Sales Target	Date	Revenue
2010	1	57,628,100	Jan 12, 2010 12:00:00 AM	\$39,036,796.40
		57,628,100	Jan 13, 2010 12:00:00 AM	\$11,488,458.59
		57,628,100	Jan 14, 2010 12:00:00 AM	\$3,232,160.48
		57,628,100	Jan 15, 2010 12:00:00 AM	\$3,080,441.44
		57,628,100	Jan 16, 2010 12:00:00 AM	\$1,976,896.69
		57,628,100	Jan 19, 2010 12:00:00 AM	\$2,159,831.74
		57,628,100	Jan 20, 2010 12:00:00 AM	\$2,558,694.30
		57,628,100	Jan 21, 2010 12:00:00 AM	\$2,935,024.72
		57,628,100	Jan 22, 2010 12:00:00 AM	\$3,559,670.33
		57,628,100	Jan 23, 2010 12:00:00 AM	\$2,371,064.81
		57,628,100	Jan 24, 2010 12:00:00 AM	\$342,583.15

This provides a smaller data set that you can use to conduct your next test. You will now add Product Line from Products to display their results.

12. In the left pane, expand **Products**, and drag **Product Line** onto the report after **Month (numeric)**.

13. In the bottom of the window, click the **Bottom** link.

The results appear as follows:

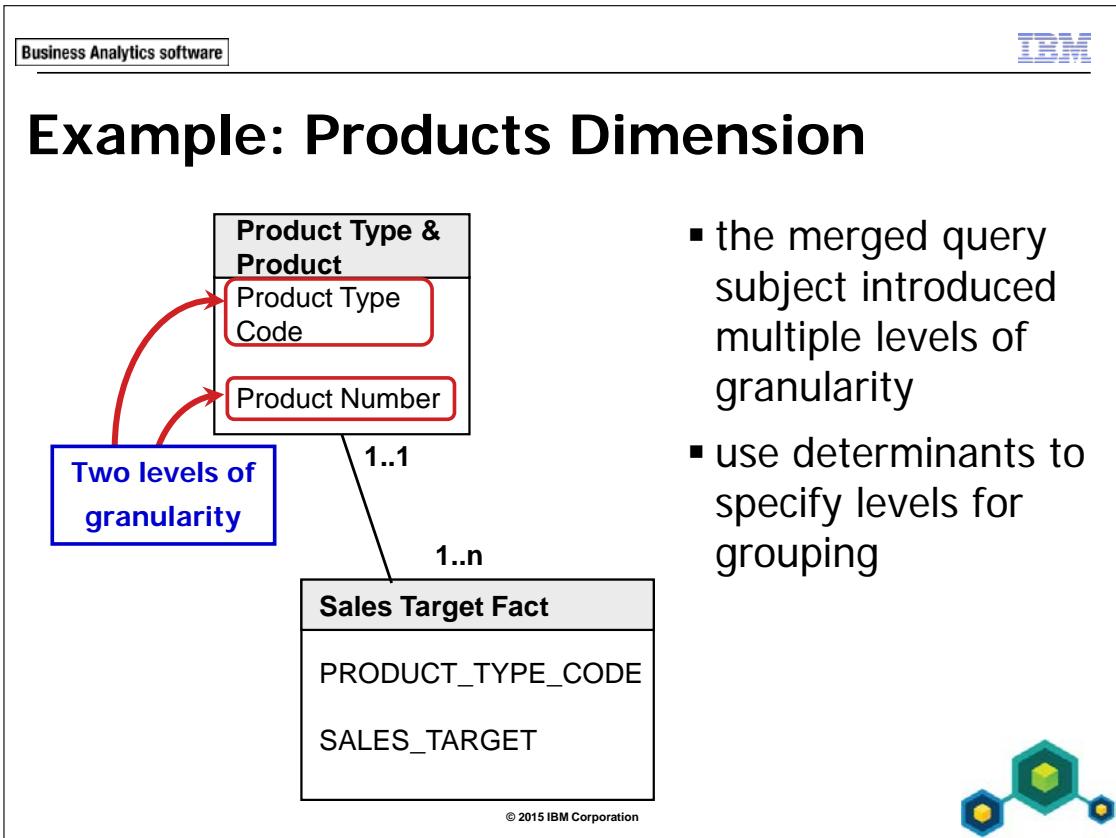
Year	Month (numeric)	Product Line	Sales Target	Date	Revenue	
2010	1	Outdoor Protection	12,396,500	Jan 12, 2010 12:00:00 AM	\$2,263,380.47	
		Outdoor Protection	12,396,500	Jan 13, 2010 12:00:00 AM	\$474,025.75	
		Outdoor Protection	12,396,500	Jan 14, 2010 12:00:00 AM	\$91,322.21	
		Outdoor Protection	12,396,500	Jan 15, 2010 12:00:00 AM	\$62,434.09	
		Outdoor Protection	12,396,500	Jan 16, 2010 12:00:00 AM	\$80,466.92	
		Outdoor Protection	12,396,500	Jan 19, 2010 12:00:00 AM	\$11,686.08	
		Outdoor Protection	12,396,500	Jan 21, 2010 12:00:00 AM	\$22,214.32	
		Outdoor Protection	12,396,500	Jan 22, 2010 12:00:00 AM	\$83,753.03	
		Outdoor Protection	12,396,500	Jan 23, 2010 12:00:00 AM	\$50,249.28	
		Personal Accessories	1,220,801,200	Jan 12, 2010 12:00:00 AM	\$7,414,443.06	
		Personal Accessories	1,220,801,200	Jan 13, 2010 12:00:00 AM	\$3,477,197.59	
		Personal Accessories	1,220,801,200	Jan 14, 2010 12:00:00 AM	\$2,118,932.80	
		Personal Accessories	1,220,801,200	Jan 15, 2010 12:00:00 AM	\$1,858,835.02	
		Personal Accessories	1,220,801,200	Jan 16, 2010 12:00:00 AM	\$1,557,191.77	
		Personal Accessories	1,220,801,200	Jan 19, 2010 12:00:00 AM	\$1,931,953.91	
		Personal Accessories	1,220,801,200	Jan 20, 2010 12:00:00 AM	\$2,558,694.30	
		Personal Accessories	1,220,801,200	Jan 21, 2010 12:00:00 AM	\$2,557,571.92	
		Personal Accessories	1,220,801,200	Jan 22, 2010 12:00:00 AM	\$2,329,018.16	
		Personal Accessories	1,220,801,200	Jan 23, 2010 12:00:00 AM	\$1,869,246.87	
		Personal Accessories	1,220,801,200	Jan 24, 2010 12:00:00 AM	\$342,583.15	
1			1,440,860,600		\$72,741,622.65	
2010			1,440,860,600		\$72,741,622.65	
Summary			1,440,860,600		\$72,741,622.65	

The sales target values are double-counted. You will examine and correct this issue in the next workshop.

14. Save the report as **Products Dimension Test**, and then click **Return**.

Results:

By specifying determinants on TIME_DIMENSION, you were able to correct a double-counting problem that occurred with sales targets.



In the slide example above, the Product Type & Product query subject is a query subject you merged earlier to remove ambiguity from PRODUCT_TYPE. Each unique instance of Product Type Code repeats once for every related product. If you do not specify determinants, sales target values will be double-counted for every product belonging to a particular product type.

Again, you need to provide information to the IBM Cognos query engine that indicates Product Type Code requires grouping when queried against sales targets.

Workshop 1: Specify Determinants

Earlier in the modeling process, you merged query subjects together to resolve ambiguous query subjects. You merged PRODUCT_TYPE with PRODUCT to create Product Type & Product, and RETAILER with RETAILER_SITE to create Retailer & Retailer Site. Both these query subjects now contain multiple levels of granularity, and cause double-counting when querying against sales targets.

You will resolve this reporting issue by specifying determinants, as follows:

- **Product Type & Product**
 - **PRODUCT_NUMBER** is unique
 - group on **PRODUCT_TYPE_CODE** and add appropriate attributes
- **Retailer & Retailer Site**
 - **RETAILER_SITE_CODE** is unique
 - Group on **RETAILER_CODE** and add appropriate attributes
- **Save** and **Close** the project.
- **Test** the Product Type & Product determinants using the **Product Dimension Test** report in **IBM Cognos Connection**.

For more information about where to work and the workshop results, refer to the Tasks and Results section that follows. If you need more information to complete a task, refer to earlier demos for detailed steps.

Workshop 1: Tasks and Results

Task 1. Specify determinants for Product Type & Product.

- In the **Foundation Objects View**, double-click the **Product Type & Product** query subject, and click the **Determinants** tab.
- Add the following new determinant:
 - Name: **Product**
 - Key: **PRODUCT_NUMBER**
 - Uniquely Identified: **checked**
- Add the following new determinant:
 - Name: **Product Type**
 - Key: **PRODUCT_TYPE_CODE**
 - Attributes: **PRODUCT_LINE_CODE**, and all **PRODUCT_TYPE_xx**
 - Group By: **checked**
- Move the **Product Type** determinant to the top.

The results appear as follows:

Determinants:		
Name	Uniquely Identified	Group By
Product Type	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Product	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Add Delete

Select a determinant to see its keys and attributes.

Key: **PRODUCT_TYPE_CODE**

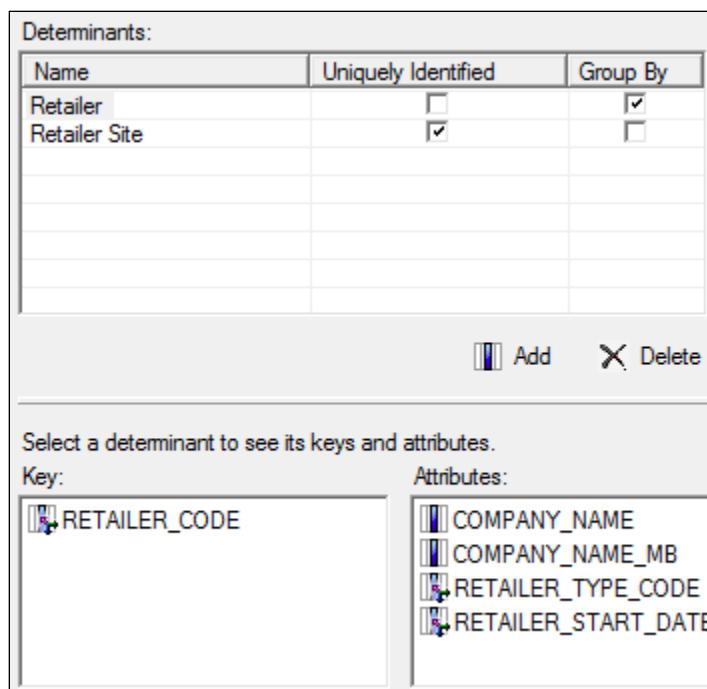
Attributes:

- PRODUCT_LINE_CODE**
- PRODUCT_TYPE_EN**
- PRODUCT_TYPE_FR**
- PRODUCT_TYPE_DE**
- PRODUCT_TYPE_NL**
- PRODUCT_TYPE_U**

- Click **OK**.

Task 2. Specify determinants for Retailer & Retailer Site.

- In **Foundation Objects View**, double-click the **Retailer & Retailer Site** query subject, and click the **Determinants** tab.
 - Add the following determinant:
 - Name: **Retailer Site**
 - Key: **RETAILER_SITE_CODE**
 - Uniquely Identified: **checked**
 - Add the following determinant:
 - Name: **Retailer**
 - Key: **RETAILER_CODE**
 - Attributes: **COMPANY_NAME**, **COMPANY_NAME_MB**, **RETAILER_TYPE_CODE**, and **RETAILER_START_DATE**
 - Group By: **checked**
- The results appear as follows:
- Move the **Retailer** determinant to the top, and then click **OK**.



- Save the project.

Task 3. Test the Products Dimension Test report.

- Publish the **Go Operational** package.
- In **IBM Cognos Connection**, in **Public Folders > GO Operational**, run the **Products Dimension Test** report.

If you are prompted to update the report, click **OK**. If the report is still open from the last demo, it may not work as expected due to caching. Simply close the browser, and then try again.

- Click the **Bottom** link.

The results appear as follows:

 Year: 2010 AND Month (numeric): 1						
Year	Month (numeric)	Product Line	Sales Target	Date	Revenue	
2010	1	Outdoor Protection	2,479,300	Jan 12, 2010 12:00:00 AM	\$2,263,380.47	
		Outdoor Protection	2,479,300	Jan 13, 2010 12:00:00 AM	\$474,025.75	
		Outdoor Protection	2,479,300	Jan 14, 2010 12:00:00 AM	\$91,322.21	
		Outdoor Protection	2,479,300	Jan 15, 2010 12:00:00 AM	\$62,434.09	
		Outdoor Protection	2,479,300	Jan 16, 2010 12:00:00 AM	\$80,466.92	
		Outdoor Protection	2,479,300	Jan 19, 2010 12:00:00 AM	\$11,686.08	
		Outdoor Protection	2,479,300	Jan 21, 2010 12:00:00 AM	\$22,214.32	
		Outdoor Protection	2,479,300	Jan 22, 2010 12:00:00 AM	\$83,753.03	
		Outdoor Protection	2,479,300	Jan 23, 2010 12:00:00 AM	\$50,249.28	
		Personal Accessories	22,218,600	Jan 12, 2010 12:00:00 AM	\$7,414,443.06	
		Personal Accessories	22,218,600	Jan 13, 2010 12:00:00 AM	\$3,477,197.59	
		Personal Accessories	22,218,600	Jan 14, 2010 12:00:00 AM	\$2,118,932.80	
		Personal Accessories	22,218,600	Jan 15, 2010 12:00:00 AM	\$1,858,835.02	
		Personal Accessories	22,218,600	Jan 16, 2010 12:00:00 AM	\$1,557,191.77	
		Personal Accessories	22,218,600	Jan 19, 2010 12:00:00 AM	\$1,931,953.91	
		Personal Accessories	22,218,600	Jan 20, 2010 12:00:00 AM	\$2,558,694.30	
		Personal Accessories	22,218,600	Jan 21, 2010 12:00:00 AM	\$2,557,571.92	
		Personal Accessories	22,218,600	Jan 22, 2010 12:00:00 AM	\$2,329,018.16	
		Personal Accessories	22,218,600	Jan 23, 2010 12:00:00 AM	\$1,869,246.87	
		Personal Accessories	22,218,600	Jan 24, 2010 12:00:00 AM	\$342,583.15	
1			57,628,100		\$72,741,622.65	
2010			57,628,100		\$72,741,622.65	
Summary			57,628,100		\$72,741,622.65	

The report did not double-count Sales Target values.

- Close **IBM Cognos Connection** without saving, and **Close Framework Manager**, saving if prompted.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Summary

- You should now be able to:
 - use determinants to specify multiple levels of granularity and prevent double-counting

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

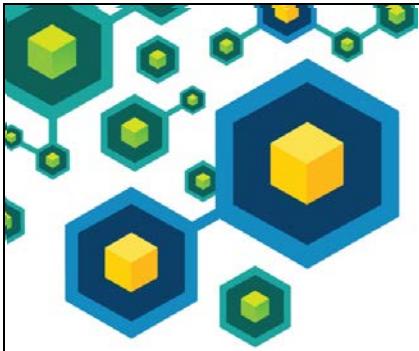
This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

11-23

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



Creating the Presentation View

IBM Cognos BI

Business Analytics software

© 2015 IBM Corporation



This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

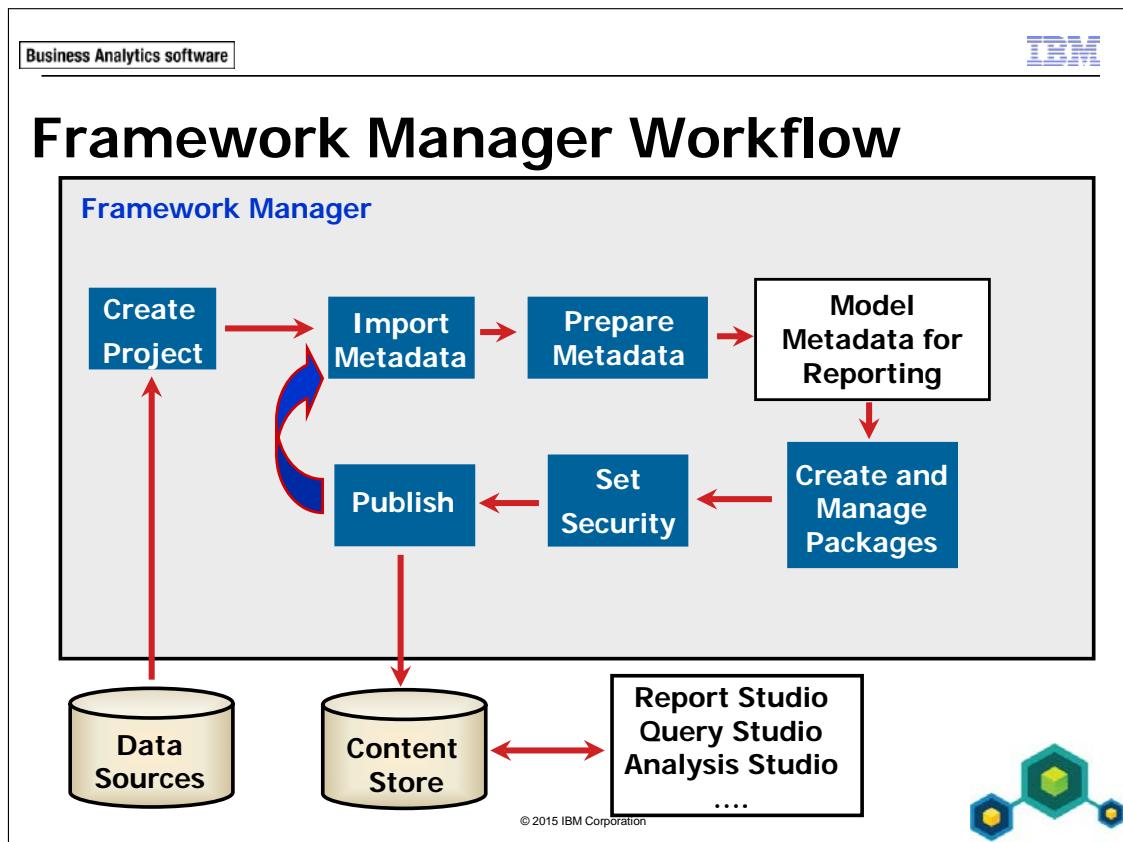
Objectives

- At the end of this module, you should be able to:
 - identify the dimensions associated with a fact table
 - identify conformed vs. non-conformed dimensions
 - create star schema groupings to provide authors with logical groupings of query subjects
 - rapidly create a model using the Model Design Accelerator

© 2015 IBM Corporation

Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

- User ID: admin
- Password: Education1

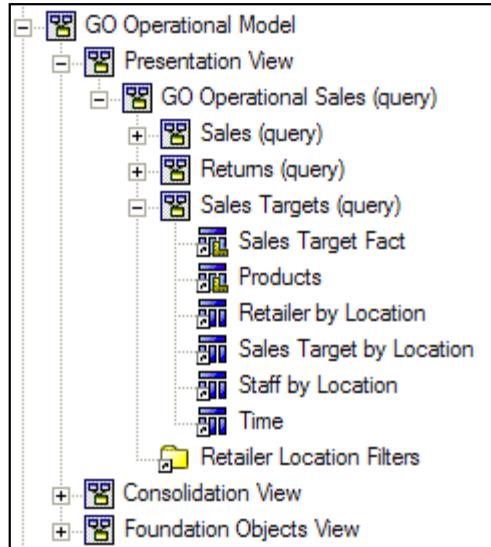


This module teaches how to create a presentation view consisting of logical groupings of query subjects (facts and their related dimensions) that focus on various areas of the business.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Using a Presentation View

- provides a logical and simplified presentation of metadata for report authors
- groups related model objects together



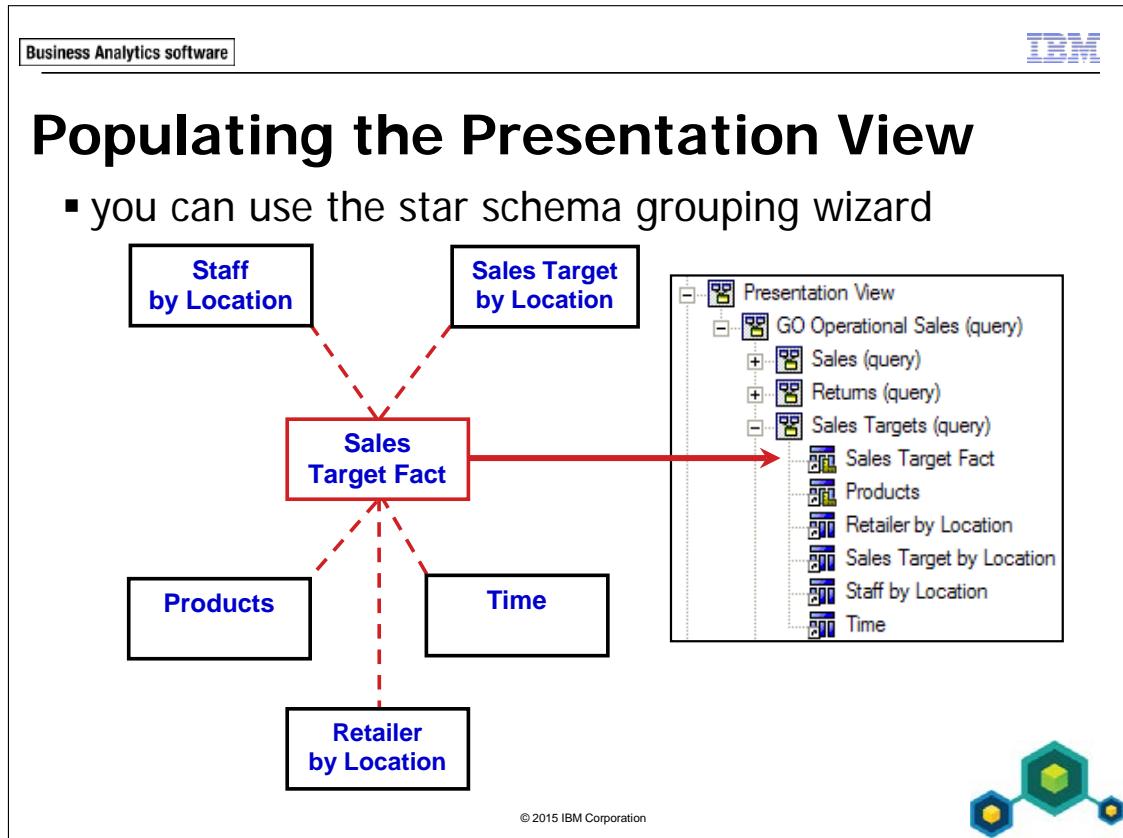
© 2015 IBM Corporation



The Presentation view provides a simplified view for report authors. It groups the metadata in a logical manner, and provides authors with commonly used tools such as filters or calculations.

Generally, the Consolidation View and Foundation Objects View are hidden from report authors. The Presentation View in the slide example consists of shortcuts to Consolidation View model query subjects, arranged in star schema groupings (a fact query subject and all its related dimensions). You can create and publish several packages that are based on the Presentation View, each one providing a different view of metadata for different reporting needs.

You do not have to model and present as a star schema. For example, if your model is designed to satisfy only a certain set of pre-built reports from which authors cannot stray, then you can model your metadata to that specific end. However, if you are modeling to a broader and largely ad hoc audience, then modeling as a star schema is an excellent choice for achieving predictable results.



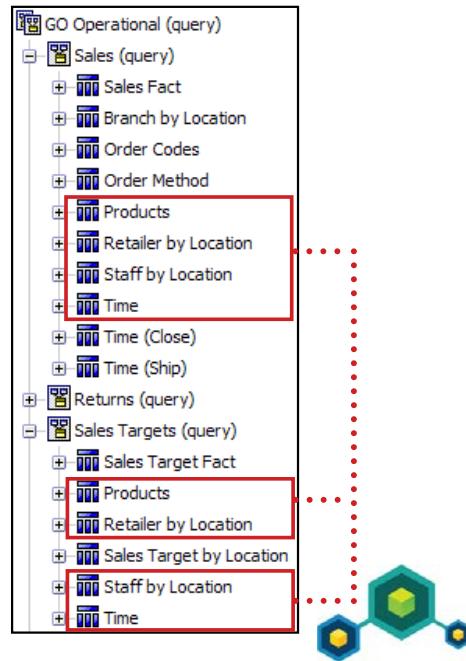
The Create Star Schema Grouping wizard creates logical groupings of central fact tables and their related dimensions. These groupings consist of shortcuts to the underlying objects and are placed in a namespace so that the same dimension names can occur in other star schema groupings. This allows authors to identify conformed dimensions.

As you model, you should document your logical groupings with a dimension map. You can then use the dimension map to quickly create your star schema groupings.

In the slide example, the objects on the left are model query subjects in the Consolidation View, which are based on objects in the Foundation Objects View. The model query subjects are related to each other in the Foundation Objects View (represented by the dashed lines in the diagram above). These objects are then grouped for presentation as shown on the right side of the diagram.

Identifying Conformed Dimensions

- conformed dimensions are based on matching names
- you must use at least one conformed dimension to report across facts to:
 - allow for stitch queries
 - ensure that each fact is correctly aggregated



© 2015 IBM Corporation

The screen capture illustrates how you can query Sales Fact and Sales Target Fact by using one or all of the conformed dimensions in the Presentation View. The dimension shortcuts, in each namespace, have the same name to indicate they are conformed and point back to the same original query subject.

Modelers and authors can quickly identify conformed dimensions in the Presentation View based on naming conventions. If designed correctly, dimensions with the exact same name in different namespaces are shared between the facts.

Dimensions that are not shared between facts (non-conformed) can still be used in multi-fact queries providing at least one conformed dimension is used.

Demo 1: Create the Presentation View

Purpose:

To provide report authors with an intuitive view of the metadata, you will create a presentation view, based on star schema groupings of your relational metadata. You will use a provided dimension map to create these groupings. You will then create a package, publish it, and test it.

Components: Framework Manager

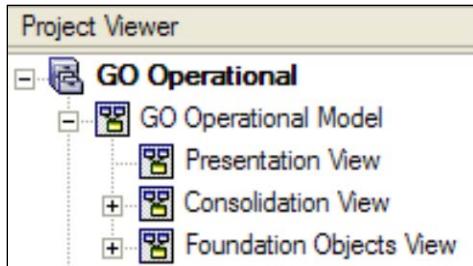
Project: GO Operational

Package: GO Operational (query)

Task 1. Use star schema groupings to populate the Presentation View.

1. In **Framework Manager**, open the **GO Operational** project located at **C:\Edcognos\B5A52\CBIFM-Start Files\Module 12\GO Operational**.
2. In the **Project Viewer** pane, under the **GO Operational Model** namespace, create a new namespace called **Presentation View**, and then drag it above the **Consolidation View** namespace.

The results appear as follows:



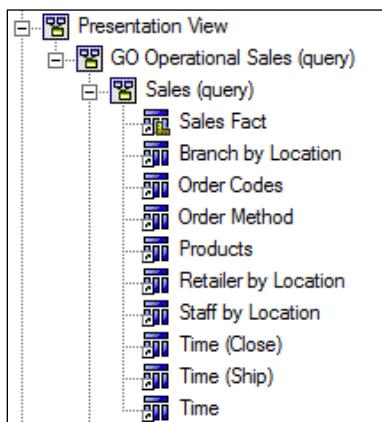
3. In the **Presentation View** namespace, create a new namespace called **GO Operational Sales (query)**.

The (query) suffix indicates that these objects are relational. When we create dimensional objects in a later module, we will use the suffix (analysis) to indicate that the objects are dimensional.

You will populate this new namespace with star schema groupings of your Consolidation View model query subjects.

4. In the **Consolidation View**, select the following query subjects:
 - **Sales Fact**
 - **Branch by Location**
 - **Order Codes**
 - **Order Method**
 - **Products**
 - **Retailer by Location**
 - **Staff by Location**
 - **Time**
 - **Time (Close)**
 - **Time (Ship)**
 5. Right-click one of the selected objects, and then click **Create Star Schema Grouping**.
- The objects that represent tables in a star schema grouping function as shortcuts.
6. Under **Namespace name**, type **Sales (query)**, and then click **OK**.
 7. Drag the new namespace into the **GO Operational Sales (query)** namespace in the **Presentation View**.
 8. Expand **Sales (query)**.

The results appear as follows:

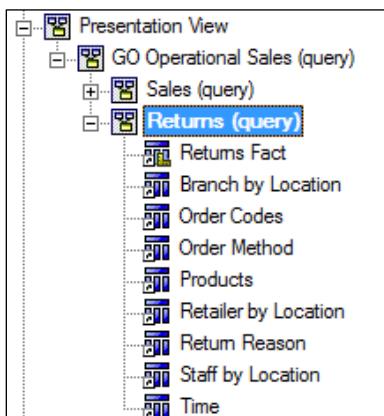


Note that all the model query subjects are shortcuts to the ones in the Consolidation View. You have simply grouped the ones needed for sales queries in one place.

Tip: If the dimensions are not listed alphabetically, you can use the Reorder feature to sort them and then drag Sales Fact to the top for easy access.

9. In the **Consolidation View**, select the following:
 - Returns Fact
 - Branch by Location
 - Order Codes
 - Order Method
 - Products
 - Retailer by Location
 - Return Reason
 - Staff by Location
 - Time
10. Right-click one of the selected objects, and then click **Create Star Schema Grouping**.
11. Under **Namespace name**, type **Returns (query)**, and then click **OK**.
12. Drag **Returns (query)** into the **GO Operational Sales (query)** namespace.

The results appear as follows:



The conformed dimensions, such as Products and Staff by Location, allow authors to create queries across sales and returns facts.

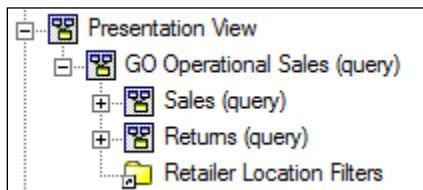
Note: If you encounter any problems with these groupings, check the Foundation Objects View to ensure all the proper relationships are in place and that there are no unresolved reporting traps.

Task 2. Make model filters available to report authors.

You will now make the Retailer Location Filters available in the Presentation View by using a shortcut.

1. In the **Consolidation View**, expand **Model Filters**, right-click **Retailer Location Filters**, point to **Create**, and click **Shortcut**.
2. Drag the shortcut to the **GO Operational Sales (query)** namespace, and then rename it to **Retailer Location Filters**.

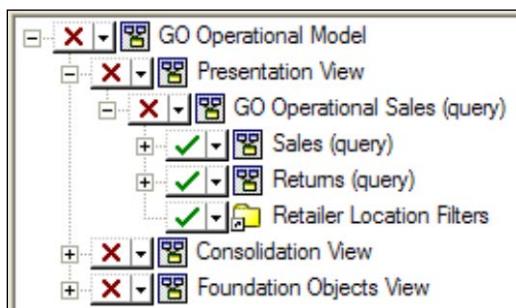
The results appear as follows:



Task 3. Create and publish a Presentation View package.

You will create a new package containing just the metadata that was modeled for report authors.

1. Right-click **Packages**, point to **Create**, and then click **Package**.
2. In the **Name** box, type **GO Operational (query)**, and then click **Next**.
3. Clear **GO Operational Model**, expand **Presentation View > GO Operational Sales (query)**.
4. Select all children of **GO Operational Sales (query)** as shown below:



5. Click **Finish**.

You are prompted to open the Publish Package wizard.

6. Click **Yes**.
7. Clear the **Enable model versioning** check box, click **Next** twice, and then click **Publish**.

Tip: You can open IBM Cognos Connection from this dialog in order to quickly view and test your work.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

8. Click **Yes**, and then click **Finish**.

The Verify Model dialog box appears listing messages that indicate underlying objects will be published with the package but hidden from authors. This is necessary as IBM Cognos will require information from these items to properly generate queries.

9. **Save** the project, and leave it open for the workshop.

Task 4. Test the GO Operational Model package.

1. In **IBM Cognos Connection**, launch **Cognos Workspace Advanced**, and select **GO Operational (query)** package.
2. Click **Create New**, select **List**, and click **OK**.
3. In the **Source** pane, expand **GO Operational (query) > Sales (query)**, and add **Products > Product Line**, and **Sales Fact > Quantity** to the report.
4. Expand **Returns (query)**, and drag **Returns Fact > Return Quantity** to the report.

The results appear as follows:

Product Line	Quantity	Return Quantity
Camping Equipment	27,301,149	304,443
Golf Equipment	5,113,701	53,247
Mountaineering Equipment	9,900,091	106,133
Outdoor Protection	12,014,445	329,568
Personal Accessories	34,907,705	286,900

You have created a report that returns data from two facts (Sales Fact and Returns Fact), based on a conformed dimension (Product Line).

5. Close the report without saving.

Results:

You have created a simplified view of the metadata by using star schema groupings to populate the Presentation View. This view uses namespaces that logically group related dimensions. The namespaces also contain conformed dimensions, which allow the report author to create reports that query across multiple facts.

Workshop 1: Create the Presentation View

Using the Star Schema Grouping wizard, you will use the Presentation View to create a logical grouping for sales target information.

To accomplish this, do the following:

- In the **Consolidation View**, select **Sales Target Fact** and the following related dimensions:
 - **Products**
 - **Retailer by Location**
 - **Sales Target by Location**
 - **Staff by Location**
 - **Time**
- Use the **Star Schema Grouping** wizard to create a new namespace called **Sales Targets (query)** containing the selected items.
- Move the new namespace to the **GO Operational Sales (query)** namespace to the **Presentation View**, below **Returns (query)**.
- Edit the **GO Operational (query)** package to add the new namespace.
- Publish the package, and then open **GO Operational (query)** in **Query Studio**.
- Create a report with the following items in **Sales Target (query)**:
 - **Sales Target by Location > Sales Target Country**
 - **Sales Target Fact > Sales Target**

When you add the items, Query Studio returns an error message is returned.

- In **Framework Manager**, review the objects related to the query in each view to identify the issue.
- Fix the underlying issue, and publish the **GO Operational (query)** package.
- Close the browser to clear the cache memory, and then recreate the report in **Query Studio**.

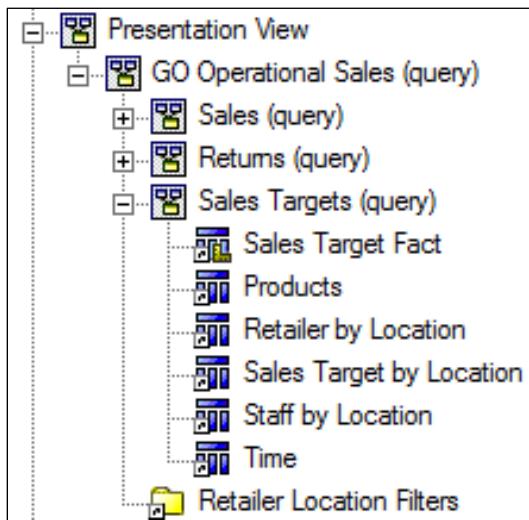
For more information about where to work and the workshop results, refer to the Tasks and Results section that follows. If you need more information to complete a task, refer to earlier demos for detailed steps.

Workshop 1: Tasks and Results

Task 1. Create star schema grouping for sales targets.

- In the **Consolidation View**, select the following:
 - **Sales Target Fact**
 - **Products**
 - **Retailer by Location**
 - **Sales Target by Location**
 - **Staff by Location**
 - **Time**
- Right-click one of the selected items, click **Create Star Schema Grouping**, and name the new namespace **Sales Target (query)**, and then click **OK**.
- Move the new namespace to the **Presentation View > GO Operational (query)**, below **Returns (query)**.

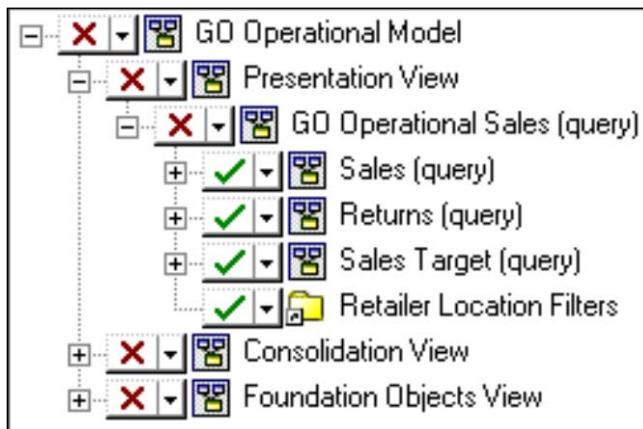
The results appear as follows:



Task 2. Update and publish the GO Operational (query) package.

- In **Packages**, edit the definition for **GO Operational (query)** to add the **GO Operational (query) > Sales Targets (query)** namespace.

The results appear as follows:



- Click **OK**, and then publish the **GO Operational (query)** package.

Task 3. Test Sales Targets (query) objects in Query Studio.

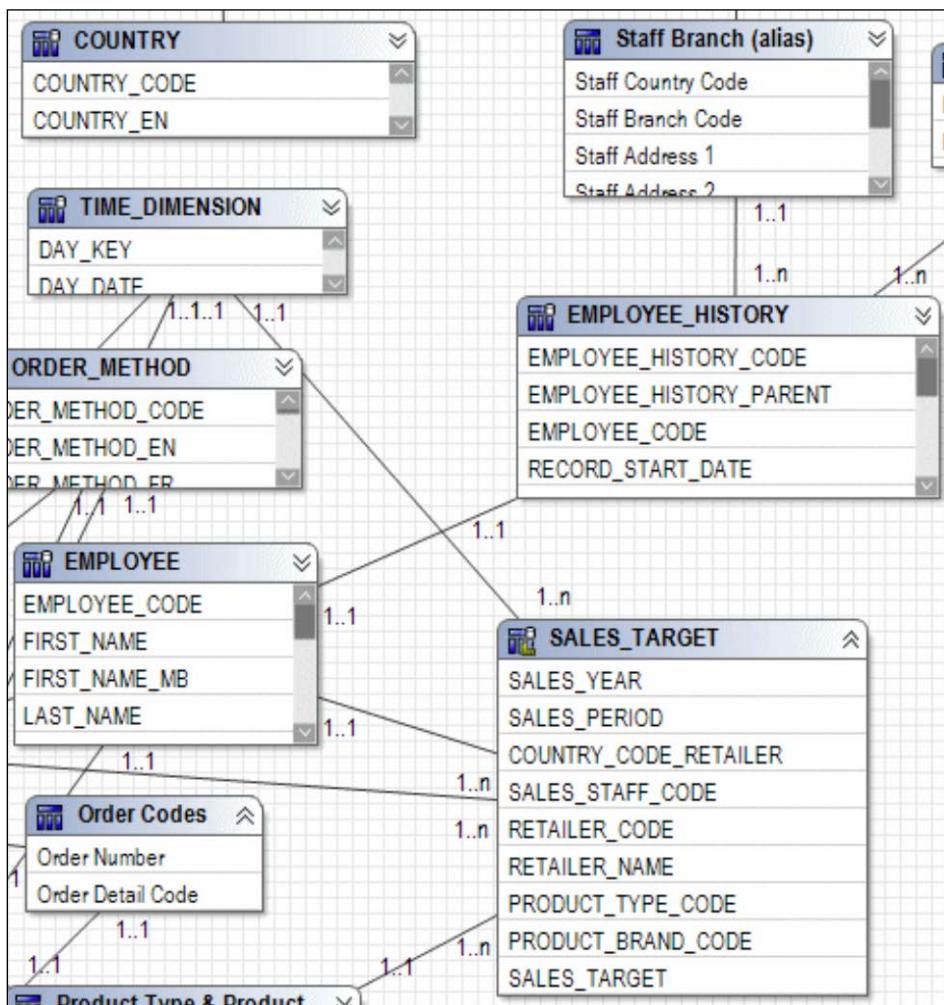
- In **IBM Cognos Connection**, launch **Query Studio**, and select the **GO Operational (query)** package.
- Expand **Sales Target (query)**, and add the following items to the report:
 - **Sales Target by Location > Sales Target Country**
 - **Sales Target Fact > Sales Target**

Query Studio displays an error message:



- In **Framework Manager**, review the relevant objects (Sales Target Country, and Sales Target) in the Diagram pane, working your way backwards from the Consolidation View to the Foundation Objects View.

In the Foundation Objects View, you notice that there is no relationship between COUNTRY and SALES_TARGET as shown below:



This relationship was intentionally deleted from the start model in order to demonstrate how reporting issues can arise from Consolidation View/Presentation View objects.

- In **Framework Manager**, in the **Foundation Objects View > gosales**, create a relationship between **COUNTRY** (COUNTRY_CODE, 1..1) and **SALES_TARGET** (COUNTRY_CODE_RETAILER, 1..n).

- Publish the **GO Operational (query)** package, and then test it again in **Query Studio**.

A section of the results appear as follows:

Sales Target Country	Sales Target
Australia	98,545,000
Austria	128,744,500
Belgium	101,979,100
Brazil	123,728,300
Canada	272,116,900
China	286,772,000
Denmark	55,215,000
Finland	169,332,500
France	257,675,400
Germany	235,055,620

The report returns the expected results.

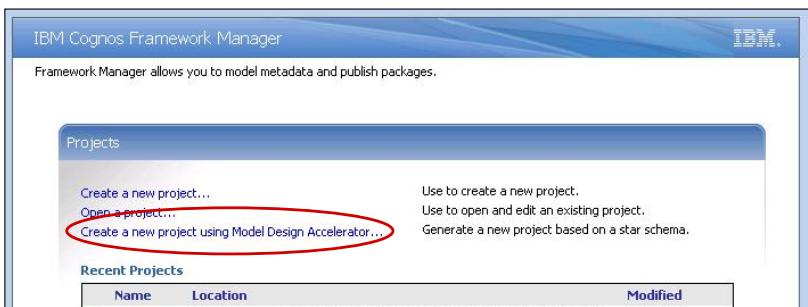
- Close the browser, without saving the report.

Business Analytics software

IBM

Using the Model Design Accelerator

- a graphical utility that guides you through a simplified modeling process



The screenshot shows the 'IBM Cognos Framework Manager' interface. At the top, it says 'Framework Manager allows you to model metadata and publish packages.' Below this is a 'Projects' section with three options: 'Create a new project...', 'Open a project...', and 'Create a new project using Model Design Accelerator...'. The third option is circled in red. To the right of these options is a brief description of each. Below this is a 'Recent Projects' table with columns for 'Name', 'Location', and 'Modified'. At the bottom of the interface, it says '© 2015 IBM Corporation'. To the right of the interface, there is a decorative graphic of interconnected hexagons.

The Model Design Accelerator is a graphical utility designed to guide both novice and experienced modelers through a simplified modeling process. The Model Design Accelerator applies IBM Cognos best practices to quickly produce single star schemas.

Experienced modelers can accelerate the modeling process so that the overall time to build a model is reduced. However, for complex transactional systems, it is better to model manually as that type of modeling requires a great attention to detail.

You can create multiple star schemas using the Model Design Accelerator and link the results together. You can then use Framework Manager to further refine the model.

Demo 2: Rapidly Create a Model using the Model Design Accelerator

Purpose:

A senior manager wants to create reports about returned products to review returns data by product, customer, and return reason. Since this project has a limited scale, you will use the Model Design Accelerator to quickly produce a package for reporting.

A data warehouse has been created and will be used as it is better suited for reporting and ease of modeling.

Component: **Framework Manager**

Project: **GO Returns**

Task 1. Start the Model Design Accelerator and import data.

1. In **Framework Manager**, close any projects that may be open.
2. Click **Create a new project using Model Design Accelerator**.
The New Project dialog opens.
Note: To use the Model Design Accelerator with an existing project, select Tools > Run Model Design Accelerator.
3. In the **Location** box, navigate to **C:\Edcognos\B5A52\Course_Project**, and then click **OK**.
4. In the **Project name** box, type **GO Returns**, and then clear the **Use Dynamic Query Mode** check box, and click **OK**.
The GO Returns folder is created by default and appears by default in the Location box.
5. Ensure that **English** is selected, and then click **OK**.
The Metadata Wizard appears.
6. Select the **great_outdoors_warehouse** data source, and then click **Next**.

7. In the list of objects, expand **GOSALES** > **Tables**, and then select the following tables:
 - **DIST_RETURNED_ITEMS_FACT**
 - **DIST_RETURN_REASON_DIM**
 - **SLS_PRODUCT_DIM**
 - **SLS_PRODUCT_LINE_LOOKUP**
 - **SLS_PRODUCT_LOOKUP**
 - **SLS_PRODUCT_TYPE_LOOKUP**
 - **SLS_RTL_DIM**

8. Click **Continue**.

The IBM Cognos Framework Manager User Guide window opens, displaying information about the Model Design Accelerator. The information in this window explains the steps to create a model using the Model Design Accelerator.

9. Click **Close**.

Task 2. Create a Returns fact table.

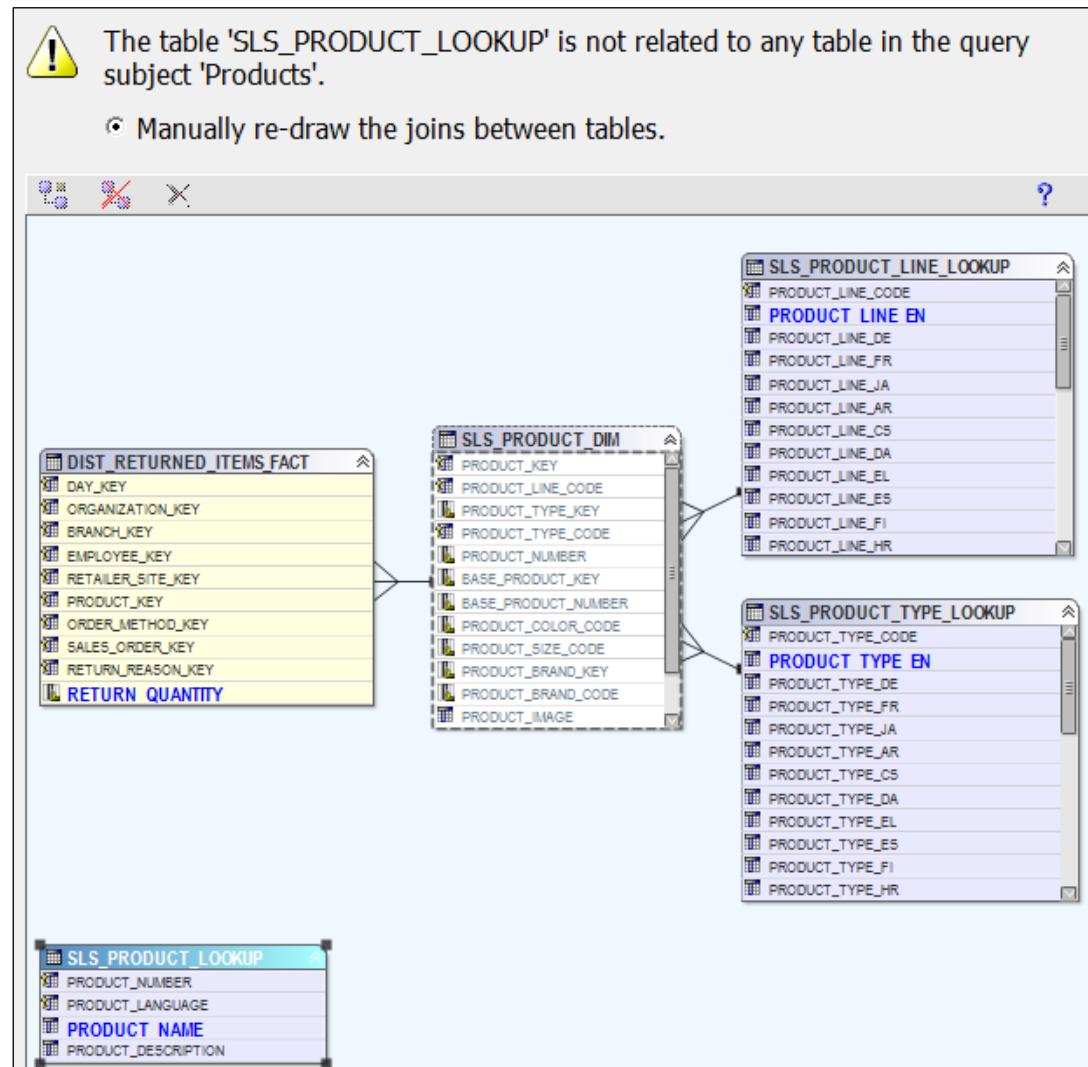
1. In the **Model Accelerator** pane, right-click the **Fact Query Subject** in the center of the pane and click **Rename**.
2. Type **Returns Fact**, and press **Enter**.
3. In the **Explorer Tree** pane, expand the **DIST_RETURNED_ITEMS_FACT** table, and drag the **RETURN_QUANTITY** data item into the **Returns Fact** query subject in the **Model Accelerator** pane.

Task 3. Create a Product dimension table.

1. In the **Model Accelerator** pane, rename **New Query Subject 1** to **Products**.
2. In the **Explorer Tree** pane, expand the **SLS_PRODUCT_LINE_LOOKUP** table, and drag the **PRODUCT_LINE_EN** data item into the **Products** query subject.
3. Expand the **SLS_PRODUCT_TYPE_LOOKUP** table, and drag the **PRODUCT_TYPE_EN** data item into the **Products** query subject.

4. Expand the **SLS_PRODUCT_LOOKUP** table, and drag the **PRODUCT_NAME** data item into the **Products** query subject.

The Relationship Editing Mode for: Products dialog opens.



This indicates that Framework Manager cannot determine the relationship between the **SLS_PRODUCT_LOOKUP** table and the **DIST_RETURNED_ITEMS_FACT** table. You will need to establish the relationship yourself.

- In the **Relationship Editing Mode** window, Ctrl+click the following:
 - SLS_PRODUCT_LOOKUP > PRODUCT_NUMBER**
 - SLS_PRODUCT_DIM > PRODUCT_NUMBER**
- In the top left corner of the dialog, click **Create a Model Relationship between these Columns** .

The Modify the Relationship dialog opens.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

7. From the **Relationship Cardinality** drop-down list, ensure that **One to Many** is selected.

The SLS_PRODUCT_LOOKUP table has an entry for each product for each language. This results in a one-to-many relationship with the PRODUCT table. Once you finish generating the basic model, you will add a filter to filter out all non-English product names, thus creating a one-to-one relationship.

8. Click **OK**, and then click **OK** again.

Task 4. Create a Retailer dimension table.

1. Rename **New Query Subject 2** to **Retailers**.
 2. In the **Explorer Tree** pane, expand the **SLS_RTL_DIM** table, and drag the **RETAILER_TYPE_EN** and **RETAILER_NAME** data items into the **Retailers** query subject.
 3. Double-click the **Retailers** table.
 4. Double-click the link between the **SLS_RTL_DIM** and **DIST_RETURNED_ITEMS_FACT** tables.
- Notice how the link between the tables is based on RETAILER_SITE_KEY. The Model Design Accelerator creates this join for you.
5. Click **Close**, and then close the **Query Subject Diagram** window.

Task 5. Create a Return Reason dimension table.

1. Rename **New Query Subject 3** to **Return Reason**.
2. In the **Explorer tree** pane, expand the **DIST_RETURN_REASON_DIM** table, and drag the **REASON_DESCRIPTION_EN** data item into the **Return Reason** query subject.
3. Right-click **New Query Subject 4**, and then click **Delete**.
4. In the bottom-right, click **Generate Model**, and then click **Yes**.

The Model Design Accelerator creates a model for you based on your selections.

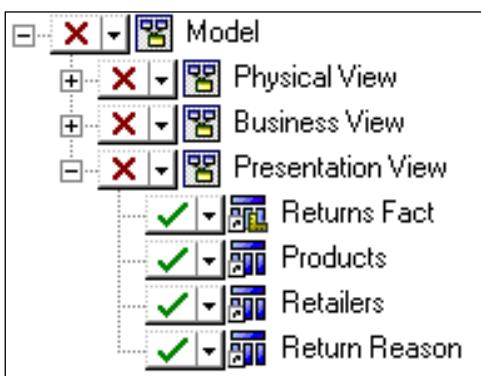
Task 6. Add a filter to the model.

1. In the **Project Viewer**, expand **Model > Physical View > GOSALESDW**.
 2. Double-click the **SLS_PRODUCT_LOOKUP** table.
- The Query Subject Definition window opens.
3. Click the **Filters** tab, and then click **Add**.

4. In the **Name** text box, type **Language Filter**.
5. From the **Available Components** pane, drag the **PRODUCT_LANGUAGE** query item to the **Expression definition** pane.
6. After **PRODUCT_LANGUAGE**] type **='EN'**.
7. Click **OK**, click the **Test** tab, and then click **Test Sample** in the bottom right of the window.
All the values in the PRODUCT_LANGUAGE column should read "EN."
8. Click **OK**.

Task 7. Create a GO Returns package.

1. In the **Project Viewer**, right-click **Packages**, point to **Create** and then click **Package**.
2. In the **Name** text box, type **GO Returns**, and then click **Next**.
3. Clear **Model** namespace, expand the **Presentation View**, and then select all the children, as shown below:



4. Click **Finish**, and click **Yes**, to publish the model.
5. Clear the **Enable model versioning** check box, and then click **Next** twice.
6. Clear the **Verify the package before publishing** check box, and then click **Publish**.
7. Click **Finish** to close the wizard, and then save the project.

Task 8. Test the GO Returns package.

1. In **IBM Cognos Connection**, launch **IBM Cognos Workspace Advanced**.
2. Select the **GO Returns** package, and then create a new **List** report.
3. From the **Source** pane, add the following items to the report:

Query Subject	Query Item
Retailers	RETAILER_NAME
Return Reason	REASON_DESCRIPTION_EN
Products	PRODUCT_LINE_EN
Returns Fact	RETURN_QUANTITY

4. Click the **RETAILER_NAME** column header, and then click **Group / Ungroup** .

A section of the results appear as follows:

RETAILER_NAME	REASON_DESCRIPTION_EN	PRODUCT_LINE_EN	RETURN_QUANTITY
1 for 1 Sports shop	Defective product	Personal Accessories	144
	Incomplete product	Camping Equipment	56
	Unsatisfactory product	Outdoor Protection	1,891
	Wrong product ordered	Personal Accessories	306
	Wrong product shipped	Personal Accessories	265

5. Close the browser, without saving the report, and close **Framework Manager**, saving changes if prompted.

Results:

The senior manager can now review Returns data by Return Reason, Retailer, and Product. The model and package were created in much less time than it would have taken had you not used the Model Design Accelerator.

Summary

- You should now be able to:
 - identify the dimensions associated with a fact table
 - identify conformed vs. non-conformed dimensions
 - create star schema groupings to provide authors with logical groupings of query subjects
 - rapidly create a model using the Model Design Accelerator

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



Working with Different Query Subject Types

IBM Cognos BI



Business Analytics software

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

The slide features a background image of a computer screen displaying the IBM Business Analytics software interface. In the top left corner of the slide area, there is a small white box containing the text "Business Analytics software". In the top right corner, the IBM logo is visible. The main title "Objectives" is displayed prominently in large, bold, black font at the top left of the slide area. Below the title, a bulleted list of objectives is presented.

- You should now be able to:
 - identify the effects of modifying query subjects on generated SQL
 - specify two types of stored procedure query subjects
 - use prompt values to accept user input

© 2015 IBM Corporation

Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

- User ID: admin
- Password: Education1

Business Analytics software

IBM

Data Source Query Subjects

- imported as an all-inclusive Select statement
- modifying a data source query subject results in customized SQL
- data source changes are only reflected if you have *not* modified the query subject

The diagram illustrates the process of modifying a data source query subject. On the left, a simple query subject is shown: "Select * from [GOSL].RETURNED_ITEM". An arrow points to the right, leading to a more complex, customized SQL statement. This statement includes additional columns from related tables (ORDER_HEADER and ORDER_DETAILS) and adds specific WHERE and AND clauses to filter the data.

```

Select
  ORDER_HEADER.RETAILER_SITE_CODE,
  ORDER_HEADER.RETAILER_CONTACT_CODE,
  ORDER_HEADER.SALES_STAFF_CODE,
  ORDER_HEADER.SALES_BRANCH_CODE,
  ORDER_HEADER.ORDER_METHOD_CODE,
  ORDER_DETAILS.PRODUCT_NUMBER,
  RETURNED_ITEM.RETURN_CODE,
  RETURNED_ITEM.RETURN_DATE,
  RETURNED_ITEM.ORDER_DETAIL_CODE,
  RETURNED_ITEM.RETURN_REASON_CODE,
  RETURNED_ITEM.RETURN_QUANTITY
from
  [GOSL].RETURNED_ITEM,
  [GOSL].ORDER_HEADER ORDER_HEADER,
  [GOSL].ORDER_DETAILS ORDER_DETAILS
where
  RETURNED_ITEM.ORDER_DETAIL_CODE =
  ORDER_DETAILS.ORDER_DETAIL_CODE
and
  ORDER_HEADER.ORDER_NUMBER =
  ORDER_DETAILS.ORDER_NUMBER
  
```

© 2015 IBM Corporation

In the above example, the SQL for the imported RETURNED_ITEM data source query subject is a simple, all-inclusive select statement. If you do not alter this SQL and new columns are added to the table, they will automatically be included when you update the query subject or test it. If you modify the SQL as seen on the right side of the example, new columns will need to be added manually in the SQL statement.

Sometimes customized SQL is required for a specific application. You can modify the SQL as required, to generate SQL that meets specific needs. You can also implement parameter driven dynamic SQL. However, you should alter the simple select statements as little as possible to generate the most efficient SQL and simplify model maintenance.

You should try to have only one instance of a SQL statement per table to reduce future maintenance. This is not always possible, but should be implemented as much as possible.

Setting the SQL Type

- you can set the SQL type for data source query subjects
- available SQL Type settings:
 - Cognos
 - Native
 - Pass-Through

© 2015 IBM Corporation



At run time, IBM Cognos generates native SQL that:

- is designed to use the database's optimizers
- is optimized for database vendor and version
- leverages features of databases wherever possible

The SQL Type setting is local to the query subject and impacts how a table-based query is defined and used in query generation. By default, Framework Manager uses Cognos SQL to create and edit query subjects.

For more information about changing the SQL Type, see the product documentation.

SQL Type: Cognos SQL

- adheres to SQL standards
- can contain metadata from multiple data sources
- has fewer database restrictions
- works with all relational and tabular data sources
- is portable

© 2015 IBM Corporation



If you need to port your model from one vendor to another, use Cognos SQL since it works with all relational and tabular data sources. It also allows IBM Cognos to generate the most optimized SQL possible, for example by removing unused elements at query time.

If a database does not support a particular function, using Cognos SQL will allow the function to be performed locally if Limited Local processing is allowed.

SQL Type: Native SQL

- allows SQL that is specific to your database
- may not be portable
- cannot contain metadata from multiple data sources

© 2015 IBM Corporation



When you edit a query subject, you can specify Native SQL. Native SQL is the SQL the data source uses, such as Oracle SQL. Native SQL lets you use keywords that are not available in Cognos SQL. You can copy and paste SQL from another application into Framework Manager for quick replication of application specific requirements and leverage work already done.

When viewing generated Cognos SQL at run-time for a query subject that is set to Native SQL, the native SQL appears as a sub-query contained between {}. IBM Cognos may add statements to the SQL you enter in order to optimize the performance of the query.

SQL Type: Pass-Through SQL

- use Pass-Through SQL when a database vendor does not support a sub-query construct
- IBM Cognos will pass the sub-query directly to the database

© 2015 IBM Corporation



Pass-Through SQL lets you use native SQL without any of the restrictions the data source imposes on sub-queries. There are some databases that do not extend support for all constructs to sub-queries. In these cases, as well as cases where you require constructs that are not supported by our query layer, use Pass-Through SQL.

Use this setting with caution as it may have a negative performance impact. With Cognos SQL and Native SQL, when SQL is generated, IBM Cognos may create wrappers for sub-query constructs, and pass the entire construct (wrapper and sub-query) to the database. Some vendors may not support this. Pass-Through SQL will tell IBM Cognos to send only the sub-query to the database and then process the remaining SQL construct (wrapper) locally.

When viewing Cognos SQL for a query subject that is set to Pass-Through SQL, the native SQL that you typed will appear as a sub-query contained between {{}}.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Business Analytics software

IBM

IBM Cognos Query Generation Architecture

- IBM Cognos user interfaces submit SQL
- the selected SQL type determines how IBM Cognos generates SQL

UI
(Framework Manager/
IBM Cognos BI
Studios)

IBM Cognos Query
Generation
(Cognos SQL,
Native SQL,)

© 2015 IBM Corporation

(Pass-Through SQL)

Cognos SQL is generated by one layer in the query engine and then passed to another for conversion to native SQL and optimization. The query is then passed to the appropriate database.

If you have chosen the Native SQL option, IBM Cognos will send the SQL directly to the optimization layer mentioned above and then on to the appropriate database.

Pass-through SQL will simply send the sub-queries of unsupported sub-query constructs directly to the database.

Demo 1: Configure SQL Type Setting

Purpose:

This demo explores the difference between the SQL Framework Manager uses to create data source query subjects and the SQL that is generated by query subjects during runtime.

You have been asked to provide a query item that returns the current year for use in various reports. To accomplish this, you will use a select construct, and a vendor specific function that requires you to change the SQL Type setting.

Component: **Framework Manager**

Project: **GO Operational**

Task 1. View data source query subject SQL.

1. In **Framework Manager**, open the **GO Operational** project located at **C:\Edcognos\B5A52\CBIFM-Start Files\Module 13\GO Operational**. If prompted, log in as User ID **admin**, and Password **Education1**.
2. In the **Project Viewer** pane, expand **GO Operational Model > Foundation Objects View > gosalles**.
3. Double-click **SALES_TARGET**.

On the SQL tab, notice the simple, all-inclusive select statement shown below:

Select * from [GOSALES].SALES_TARGET

This statement is written in Cognos SQL. It defines the scope of the query subject and generates run-time SQL when authoring a report or testing query subjects/items in Framework Manager.

4. Click the **Test** tab, and then in the bottom right corner, click **Test Sample**. The data is retrieved and displayed in the Test results pane.

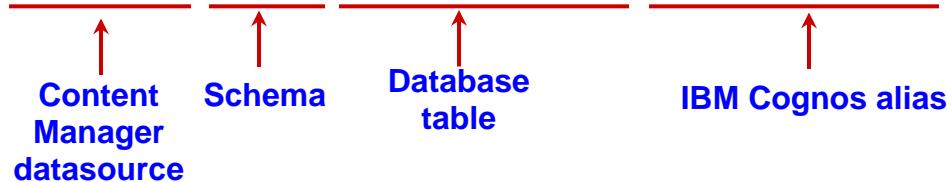
5. Click the **Query Information** tab.

This tab shows the SQL that the query engine used to retrieve the data in the Test results pane. The SQL is presented in both Cognos SQL and Native SQL as follows:

Cognos SQL <pre> select SALES_TARGET.SALES_YEAR as SALES_YEAR, SALES_TARGET.SALES_PERIOD as SALES_PERIOD, SALES_TARGET.COUNTRY_CODE_RETAILER as COUNTRY_CODE_RETAILER, SALES_TARGET.SALES_STAFF_CODE as SALES_STAFF_CODE, SALES_TARGET.RETAILER_CODE as RETAILER_CODE, SALES_TARGET.RETAILER_NAME as RETAILER_NAME, SALES_TARGET.PRODUCT_TYPE_CODE as PRODUCT_TYPE_CODE, SALES_TARGET.PRODUCT_BRAND_CODE as PRODUCT_BRAND_CODE, SALES_TARGET.SALES_TARGET as SALES_TARGET from GOSALES..GOSALES.SALES_TARGET SALES_TARGET </pre>	 Projection list
Native SQL <pre> select "SALES_TARGET"."SALES_YEAR" "SALES_YEAR", "SALES_TARGET"."SALES_PERIOD" "SALES_PERIOD", "SALES_TARGET"."COUNTRY_CODE_RETAILER" "COUNTRY_CODE_RETAILER", "SALES_TARGET"."SALES_STAFF_CODE" "SALES_STAFF_CODE", "SALES_TARGET"."RETAILER_CODE" "RETAILER_CODE", "SALES_TARGET"."RETAILER_NAME" "RETAILER_NAME", "SALES_TARGET"."PRODUCT_TYPE_CODE" "PRODUCT_TYPE_CODE", "SALES_TARGET"."PRODUCT_BRAND_CODE" "PRODUCT_BRAND_CODE", "SALES_TARGET"."SALES_TARGET" "SALES_TARGET" from "GOSALES"."SALES_TARGET" "SALES_TARGET" FOR FETCH ONLY </pre>	

The syntax in the From clause of the Cognos SQL contains the following parts:

GOSALES..GOSALES.SALES_TARGET SALES_TARGET



Cognos SQL is an easy-to-read version, while Native SQL represents the SQL that is sent to the database. In both, the SQL selects each column individually, rather than using Select * from RETURNED_ITEM. This is because the SQL is generated based on the individual query items that make up the query subject. When you test the entire query subject, all query items are included in the query and therefore you see each column in the generated SQL.

6. Click **Cancel**.
7. Under **SALES_TARGET**, right-click the **SALES_TARGET** query item, click **Test**, and then click **Test Sample**.
8. Click the **Query Information** tab.

The results appear as follows:

Cognos SQL
<pre>select SALES_TARGET.SALES_TARGET as SALES_TARGET from GOSALES..GOSALES.SALES_TARGET SALES_TARGET</pre>
Native SQL
<pre>select "SALES_TARGET"."SALES_TARGET" "SALES_TARGET" from "GOALS". "SALES_TARGET" "SALES_TARGET" FOR FETCH ONLY</pre>

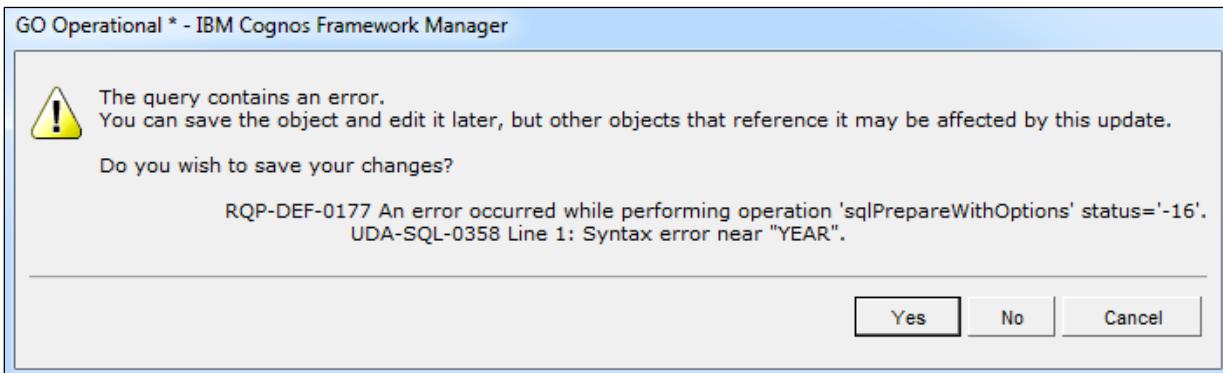
Because you only selected one query item to test, only one column appears in the generated SQL. All unused items in the scope of the query subject have been dropped during optimization.

9. Click **Close**.

Task 2. Use a vendor specific function and configure the SQL Type setting.

1. In the **Foundation Objects View**, right-click the **gosales** namespace, point to **Create**, and then click **Query Subject**.
2. In the **Name** box, type **Current Year**, select **Data Source (Tables and Columns)**, and then click **OK**.
3. Under **Select a data source**, ensure **GOSALES** is selected, clear the **Run database query subject wizard** check box, and then click **Finish**.
4. Change the **SQL** statement to:
Select YEAR(current timestamp) "Current Year" FROM sysibm.sysdummy1

- Click the **Test** tab.



A message indicates that there is a syntax error near "Year". Cognos SQL expects the From clause to specify a table name. You will use Native SQL to leverage this statement.

- Click **OK**, and then click **Options** in the lower right corner.
- Click the **SQL Settings** tab, from the **SQL Type** list, select **Native**, click **OK** to the message, and then click **OK**.
- Click **Test Sample**.

You have successfully retrieved the current year using a vendor-specific function by using Native SQL.

- Click the **Query Information** tab.

The results appear as follows:

Cognos SQL
<pre>select Current_Year."Current Year" as Current_Year from (GOSALES... {Select YEAR(current timestamp) "Current Year" from sysibm.sysdummy1}) Current_Year</pre>
Native SQL
<pre>select "Current_Year"."Current Year" "Current_Year" from (Select YEAR(current timestamp) "Current Year" from sysibm.sysdummy1) "Current_Year" FOR FETCH ONLY</pre>

The native SQL is reflected in the derived table portion of the Cognos SQL between the {} brackets. Derived tables will be discussed in further detail in another module.

- Click **OK**, and then save the project.

Results:

You have used the SQL and Query Information tabs to explore the difference between Cognos SQL, and the Native SQL that is generated at run time. You also changed the SQL Setting to Native, in order to use function that was not recognized by Cognos SQL.

Stored Procedure Query Subjects

- two types of stored procedure query subjects:

Type	Description
Data Query	returns a single result set based on a simple or complex query
Data Modification	leverages a stored procedure in the data source to modify the data source

- both types can accept arguments

© 2015 IBM Corporation



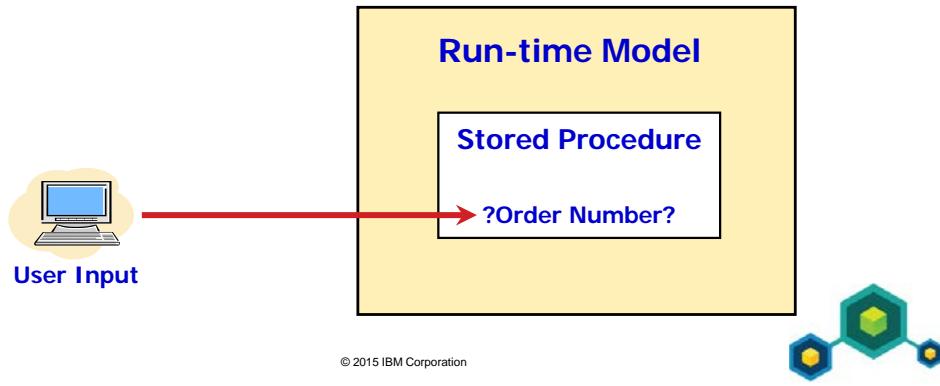
If a stored procedure returns multiple result sets, IBM Cognos only supports the first result set. Framework Manager defines the metadata according to the result set returned by the stored procedure when it is first created. If an existing stored procedure returns a different result set than when it was created, it will cause an error.

You can import a stored procedure into Framework Manager by either creating a query subject, or using the Metadata Import Wizard. If you use the Metadata Import Wizard, the query subject will appear to be broken until you verify its projection list.

Some data source systems allow for multiple stored procedures with the same name; however each accepts a different number and/or type of argument that determines which stored procedure is used. This is known as an overloaded signature. To work with overloaded signatures, create multiple stored procedures with unique names, and then create a separate query subject for each result set.

Using Prompt Values

- use prompt values when user input is required for variables beyond the report author's control
- the syntax for using a prompt as a value is:
 - ?Prompt Name?



In general, it is better to define prompts in the reporting application to make use of the additional prompt features. However, there are some variables that report authors cannot modify such as parameters in a stored procedure. For these, you can use Framework Manager to define prompts.

Prompt values can also be used in:

- parameter maps
- session parameters
- expressions including filters, calculations, and relationships

If a stored procedure with an order number parameter returns rows for a specified order, instead of using a hard-coded order number as the argument for the stored procedure query subject, you can use a prompt, such as ?Order Number?. This will allow the end-user to specify which order they want to retrieve information for.

Demo 2: Create and Test a Data Query Stored Procedure Query Subject

Purpose:

Phone representatives at the Sample Outdoors Company call center need an application to help them quickly retrieve information for specific orders. There is a stored procedure in the GOSALES database that can be used to create this application. To do this, you will create and configure a stored procedure query subject.

Components: Framework Manager, Cognos Workspace Advanced

Project: GO Operational

Package: GO Call Center

Task 1. Create a stored procedure query to retrieve data.

1. In the **Project Viewer**, under **GO Operational Model > Foundation Objects View > gosales**, create a **Folder** called **Stored Procedures**, click **Next**, and then click **Finish**.
2. Right-click the **Stored Procedures** folder, point to **Create**, and then click **Query Subject**.
3. In the **Name** box, type **Find Order Information**, select **Stored Procedure**, and then click **OK**.
4. In the **Select a data source** pane, ensure **GOSALES** is selected, and then click **Next**.
5. In the **Stored Procedures** pane, expand **GOSALES > Procedures**.
6. Click the **FINDORDERINFO** stored procedure, and then click **Finish**.
The Query Subject Definition window for the stored procedure appears.
Note: If the stored procedure is missing, navigate to C:\Edcognos\B5A52\Instructor Files and double-click B5A52.bat.
You will now add a prompt value to allow for user input as opposed to hard coding a value for the ORDERNUMBER argument.

7. Click the **ORDERNUMBER** argument, and then click the **ellipsis** in the **Value** column.
8. In the **Value** pane, type **?Order Number?**.
9. Click **OK**, and then click the **Test** tab.

The Prompt Values dialog box appears. In order to prevent continually being prompted, you will clear the Always prompt for values when testing box.

10. Double-click in the **Value** field, type **100002**, and press **Enter**.
11. Clear the **Always prompt for values when testing**, and then click **OK**.
12. Click **Test Sample** in the bottom right corner.

The results appear as follows:

Test Results				
ORDER_NUMBEF	RETAILER_NAME	PRODUCT_NUMBER	ORDER_DATE	SHIP_DATE
100002	Ar fresco	75110	Jan 12, 2010 12:00:00 AM	Jan 19, 2010 12:00:00 AM
100002	Ar fresco	76110	Jan 12, 2010 12:00:00 AM	Jan 19, 2010 12:00:00 AM
100002	Ar fresco	85110	Jan 12, 2010 12:00:00 AM	Jan 19, 2010 12:00:00 AM
100002	Ar fresco	65110	Jan 12, 2010 12:00:00 AM	Jan 19, 2010 12:00:00 AM
100002	Ar fresco	100110	Jan 12, 2010 12:00:00 AM	Jan 19, 2010 12:00:00 AM

Several records are returned in the Test Results pane with related order information.

13. Click **OK**, and then save the project.

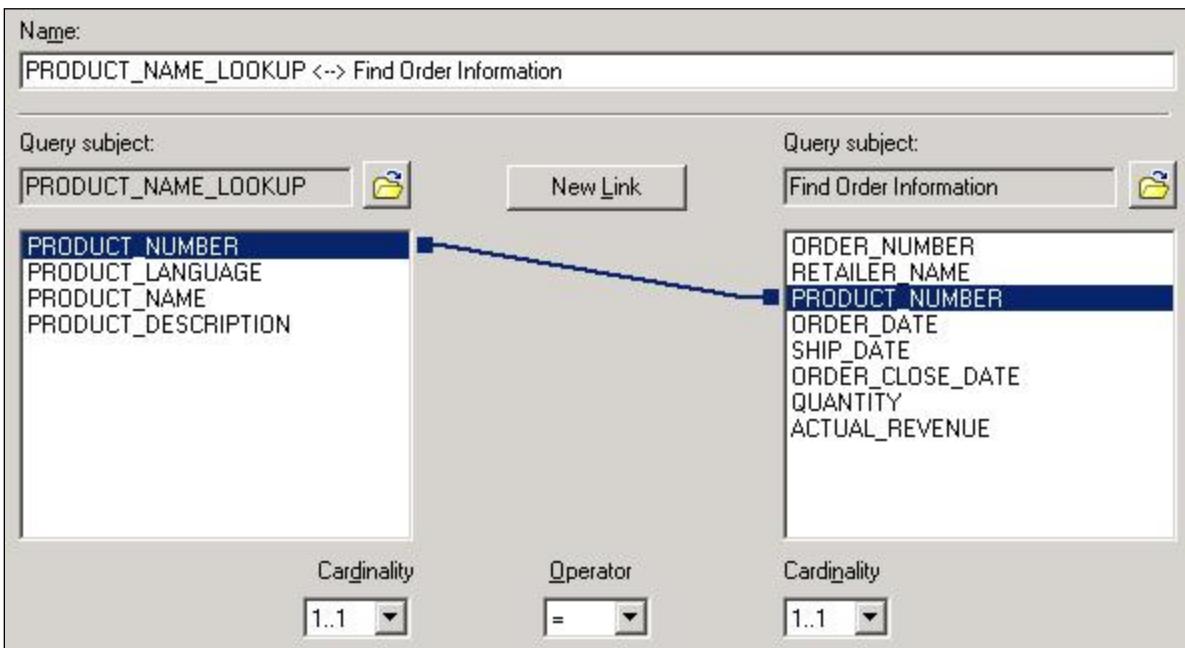
Task 2. Edit query item properties and create a relationship.

The stored procedure returns a product number; however a product name would be easier for report consumer to interpret. To enable users to obtain a product name, you will create a relationship between PRODUCT_NAME_LOOKUP and the Find Order Information stored procedure, based on the PRODUCT_NUMBER attribute.

1. In the **Stored Procedures** folder, expand **Find Order Information**.
2. Ctrl+click **ORDER_NUMBER** and **PRODUCT_NUMBER**, and in the **Properties** pane, change the **Usage** property for both query items to **Identifier**.

In this case, you know that the underlying fields in the database are indexed and can therefore be set as identifiers instead of attributes.

3. In the **Project Viewer** under **Foundation Objects View > gosales**, click **PRODUCT_NAME_LOOKUP**, and then Ctrl+click **Find Order Information**.
4. Right-click either query subject, and then select **Create > Relationship**.
5. Define the relationship from **PRODUCT_NAME_LOOKUP** (PRODUCT_NUMBER, 1..1) to **Find Order Information** (PRODUCT_NUMBER, 1..1).



PRODUCT_NAME_LOOKUP will act as a lookup table for Find Order Information to retrieve product name values.

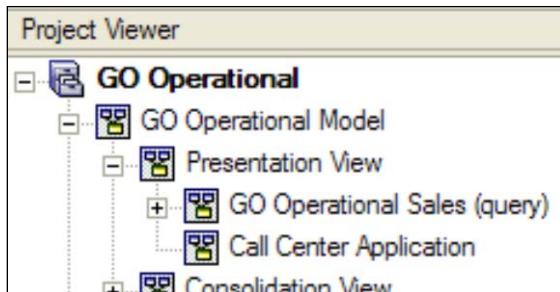
6. Click **OK**, and then save the project.

Task 3. Create a Find Order Information model query subject.

Now that you have a relationship between Find Order Information and PRODUCT_NAME_LOOKUP, you can create a model query subject that retrieves all the required information for your call center application. You will create a business view for this application.

1. In the **Project Viewer**, in the **Presentation View** namespace, create a new namespace called **Call Center Application**.

The results appear as follows:



2. In the **Call Center Application** namespace, create a new model query subject called **Find Order Information**, and then click **OK**.
3. In the **Available Model Objects** pane, expand **Foundation Objects View > gosales > Stored Procedures > Find Order Information**.
4. Add all query items to the **Query Items and Calculations** pane, except for **PRODUCT_NUMBER**.
5. Expand **PRODUCT_NAME_LOOKUP**, and add **PRODUCT_NAME** to the **Query Items and Calculations** pane.
6. Use the up arrow to move **PRODUCT_NAME** under **RETAILER_NAME**.
7. Click the **Test** tab, and then click **Test Sample** in the bottom right corner.

The order information appears with the appropriate product name, as shown below:

Test results			
ORDER_NUMBEF	RETAILER_NAME	PRODUCT_NAME	ORDER_DATE
100002	Ar fresco	Mountain Man Deluxe	Jan 12, 2010 12:00:00 AM
100002	Ar fresco	Edge Extreme	Jan 12, 2010 12:00:00 AM
100002	Ar fresco	Bear Edge	Jan 12, 2010 12:00:00 AM
100002	Ar fresco	Glacier GPS Extreme	Jan 12, 2010 12:00:00 AM
100002	Ar fresco	Insect Bite Relief	Jan 12, 2010 12:00:00 AM

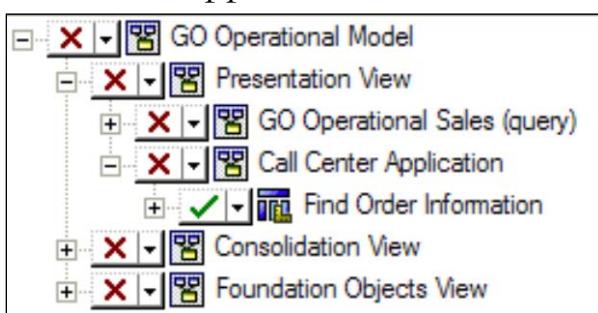
8. Click **OK**.
9. Expand **Find Order Information** and, rename the query items to lower case, with no underscores.

If time is short, you can skip renaming all query items as this will be done for you at the start of the next module. This renaming is optional for the purposes of the demo, but it reinforces the value of making objects in the Presentation View user-friendly for reporting.

Task 4. Create a new package for the call center phone representatives.

1. In **Packages**, create a package called **GO Call Center** that only contains **Presentation View > Call Center Application > Find Order Information**.

The results appear as follows:



2. Click **Finish**.

You are prompted to open the Publish Package wizard.

3. Click **Yes**.
 4. Clear the **Enable model versioning** check box, click **Next** twice.
 5. Click **Publish**, and then click **Finish**.
- The Verify Model dialog appears listing informational messages.
6. Click **Close**, and then **Save** the project.

Task 5. Test the Find Order Information query subject in IBM Cognos Workspace Advanced.

1. In **IBM Cognos Connection**, launch **IBM Cognos Workspace Advanced**, and then select the **GO Call Center** package, and create a **List** report.
2. Drag the **Find Order Information** query subject to the report.
An Order Number prompt appears.
3. In the **Provide a number** box, type **100004**, and then click **OK**.

A list appears displaying all the records for the requested order number, as shown below:

Order Number	Retailer Name	Product Name	Order Date	Ship Date	Order Close Date	Quantity	Actual Revenue
100004	Ao ar livre	Hibernator Lite	Jan 12, 2010 12:00:00 AM	Jan 21, 2010 12:00:00 AM	Jan 21, 2010 12:00:00 AM	354	29,658.12
100004	Ao ar livre	Star Gazer 2	Jan 12, 2010 12:00:00 AM	Jan 21, 2010 12:00:00 AM	Jan 21, 2010 12:00:00 AM	139	75,289.35
100004	Ao ar livre	Star Lite	Jan 12, 2010 12:00:00 AM	Jan 21, 2010 12:00:00 AM	Jan 21, 2010 12:00:00 AM	261	89,841.42
100004	Ao ar livre	TrailChef Deluxe Cook Set	Jan 12, 2010 12:00:00 AM	Jan 21, 2010 12:00:00 AM	Jan 21, 2010 12:00:00 AM	279	33,658.56

4. Close the browser, and leave **Framework Manager** open for the next demo.

Results:

By creating a stored procedure query subject with a prompt value, you were able to provide an application that allows call center representatives to view the information they need about specific orders.

Data Modification Stored Procedures

- update data sources by adding, updating, or deleting records
- if available in a package, report authors can define conditions that trigger stored procedures to execute
- limited use:
 - many organizations do not allow external applications to update source data
 - database administrators limit which tables and columns can be modified

© 2015 IBM Corporation



Data Sources have two properties that control how report authors can use stored procedure query subjects:

- Transaction Access Mode: controls the level of access for each transaction. This value can be set to either Read-Only, or Read-Write.
- Transaction Statement Mode: specifies the action that will occur when the transaction ends. This value can be set to either: Rollback, Commit, or Autocommit.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

13-22

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Demo 3: Create and Test a Data Modification Stored Procedure Query Subject

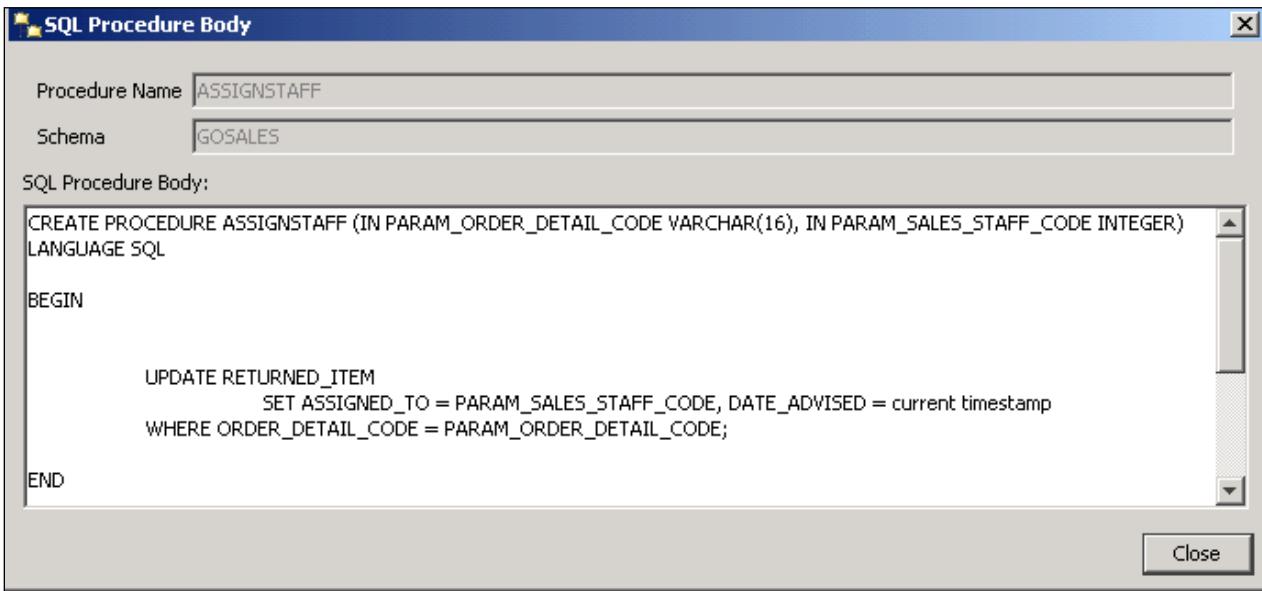
Purpose:

Report Authors want to automate a process that notifies sales reps when one of their orders has been returned. To do this, you will create a stored procedure query subject, based on an existing stored procedure in the GOSALES database, which updates columns in the RETURNED_ITEM table.

Component: **Framework Manager**

Project: **GO Operational**

The Sample Outdoors Company database contains a data modification stored procedure named **ASSIGNSTAFF**, which is defined as follows:



The screenshot shows the 'SQL Procedure Body' dialog box. At the top, there are fields for 'Procedure Name' (set to 'ASSIGNSTAFF') and 'Schema' (set to 'GOSALES'). Below these, the 'SQL Procedure Body:' section contains the following code:

```

CREATE PROCEDURE ASSIGNSTAFF (IN PARAM_ORDER_DETAIL_CODE VARCHAR(16), IN PARAM_SALES_STAFF_CODE INTEGER)
LANGUAGE SQL

BEGIN

    UPDATE RETURNED_ITEM
        SET ASSIGNED_TO = PARAM_SALES_STAFF_CODE, DATE ADVISED = current timestamp
    WHERE ORDER_DETAIL_CODE = PARAM_ORDER_DETAIL_CODE;

END

```

A 'Close' button is visible at the bottom right of the dialog.

This stored procedure updates the RETURNED_ITEM table. It adds the sales staff code to the ASSIGNED_TO column, and then retrieves and adds the system date to the DATE ADVISED column. It also has two arguments that must be supplied: SALES_STAFF_CODE and ORDER_DETAIL_CODE.

Task 1. Test the RETURNED_ITEM query subject.

1. In Foundation Objects View > gosales, expand the **Original Sales & Returns Objects** folder.
2. Test the **RETURNED_ITEM** query subject.
3. Scroll to the right until you see the **ASSIGNED_TO** column.

The results appear as follows

Test results					
	ORDER_DETAIL_CODE	RETURN_REASON_CODE	RETURN_QUANTITY	ASSIGNED_TO	FOLLOW_UP_CODE
	4000059	1	30		-1
	6000325	4	152		-1
	5000097	1	146		-1
	80214477	4	47		-1

Notice that the ASSIGNED_TO values are null. You will import a stored procedure to update columns in this table and test for the ORDER_DETAIL_CODE value of 4000059, the first row in this table.

4. Click **Close**.

Task 2. Import the stored procedure into the model.

1. Right-click the **Stored Procedures** folder, point to **Create**, and then click **Query Subject**.
2. In the **Name** box, type **AssignStaff**, select **Stored Procedure**, and then click **OK**.
3. In the **Select a data source** pane, ensure **GOSALES** is selected, and then click **Next**.
4. In the **GOSALES > Procedures**, click the **ASSIGNSTAFF** stored procedure, and then click **Finish**.

The Query Subject Definition window for the stored procedure appears.

You will now add prompt values to allow for user input, as opposed to hard coding values for the expected arguments.

5. Click the **PARAM_SALES_STAFF_CODE** argument, and then click the ellipsis in the **Value** column.
6. In the **Value** pane, type **?SalesStaffCode?**, and then click **OK**.

7. Click the **PARAM_ORDER_DETAIL_CODE** argument, and then add the following prompt value to the **Value** column:

?OrderDetailCode?

The results appear as follows:

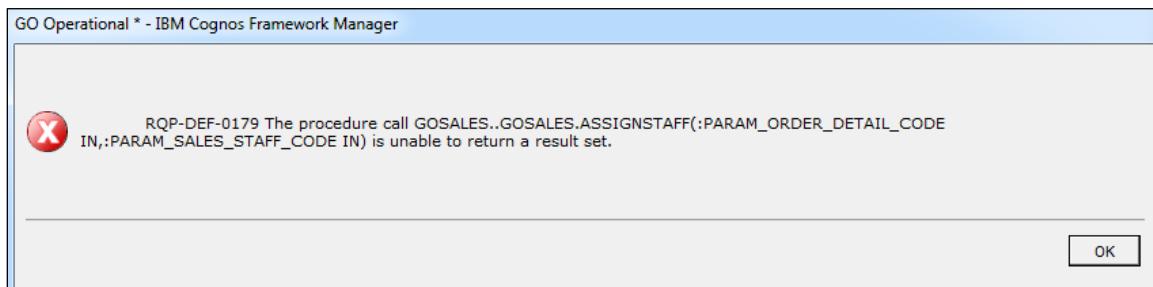
Argument Name	Mode	Type	Format	Value
PARAM_ORDER_DETAIL_CODE	in	characterLength16	Size=34, Precision=16, Scale ?OrderDetailCode?	
PARAM_SALES_STAFF_CODE	in	int32	Size=4, Precision=0, Scale= ?SalesStaffCode?	

8. Click the **Test** tab, and enter the following values:

- Order Detail Code: **4000059**
- Sales Staff Code: **572**

9. Click **OK**.

An error message appears stating the stored procedure is unable to return a result set:



This is because the stored procedure updates a table and does not retrieve rows from a table. You will modify the definition of this stored procedure query subject to act as a data modification stored procedure query subject.

10. Click **OK**, click **Cancel**, and then click the **Definition** tab.

In the Type box, notice that the current setting is Data Query. With this setting, the stored procedure, when tested, should return a result set.

11. Change the **Type** setting to **Data Modification**.

12. Click the **Test** tab, and then click **Test Sample**.

A message appears stating the stored procedure executed successfully.

13. Click **OK** three times, and then **Save** the project.

Task 3. Re-test the RETURNED_ITEM query subject.

1. In the **Original Sales & Returns Objects** folder, test the **RETURNED_ITEM** query subject.
2. Scroll to the right until you see the **ASSIGNED_TO** column.
The results appear as follows:

Test results			
ORDER_DETAIL_CODE	RETURN_REASON_CODE	RETURN_QUANTITY	ASSIGNED_TO
4000059	1	30	572
6000325	4	152	
5000097	1	146	

Notice that the ASSIGNED_TO value in the first row is now 572, the value you provided in the prompt dialog. If you scroll right a little further, you will see that the DATE ADVISED column has a value for the current date and time. The row for ORDER_DETAIL_CODE 4000059 was successfully updated.

3. Click **Close**.
4. In **Data Sources**, click **GOSALES**, and in the **Properties** pane, set **Transaction Access Mode** to **Read-Write**, and **Transaction Statement Mode** to **Commit**.
5. **Save** and **Close** the project.

This stored procedure can now be added to a package and used by report authors to automate a process that notifies sales reps when one of their orders has been returned.

Results:

You have created a data modification stored procedure query subject, based on an existing stored procedure in the GOSALES data source, which updates columns in the RETURNED_ITEM table.

Summary

- You should now be able to:
 - identify the effects of modifying query subjects on generated SQL
 - specify two types of stored procedure query subjects
 - use prompt values to accept user input

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

13-28

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



The advertisement features a white background with a decorative border of teal hexagonal icons containing yellow cubes. In the center, there is a larger, more prominent hexagonal icon with a yellow cube, surrounded by smaller ones. The IBM logo is positioned in the top right corner. Below the graphic, the title "Setting Security in Framework Manager" is displayed in a large, bold, black sans-serif font. Underneath the title, the text "IBM Cognos BI" is written in a smaller, regular black font. At the bottom left, a small rectangular box contains the text "Business Analytics software". At the bottom right, a copyright notice reads "© 2015 IBM Corporation".

Setting Security in Framework Manager

IBM Cognos BI

Business Analytics software

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

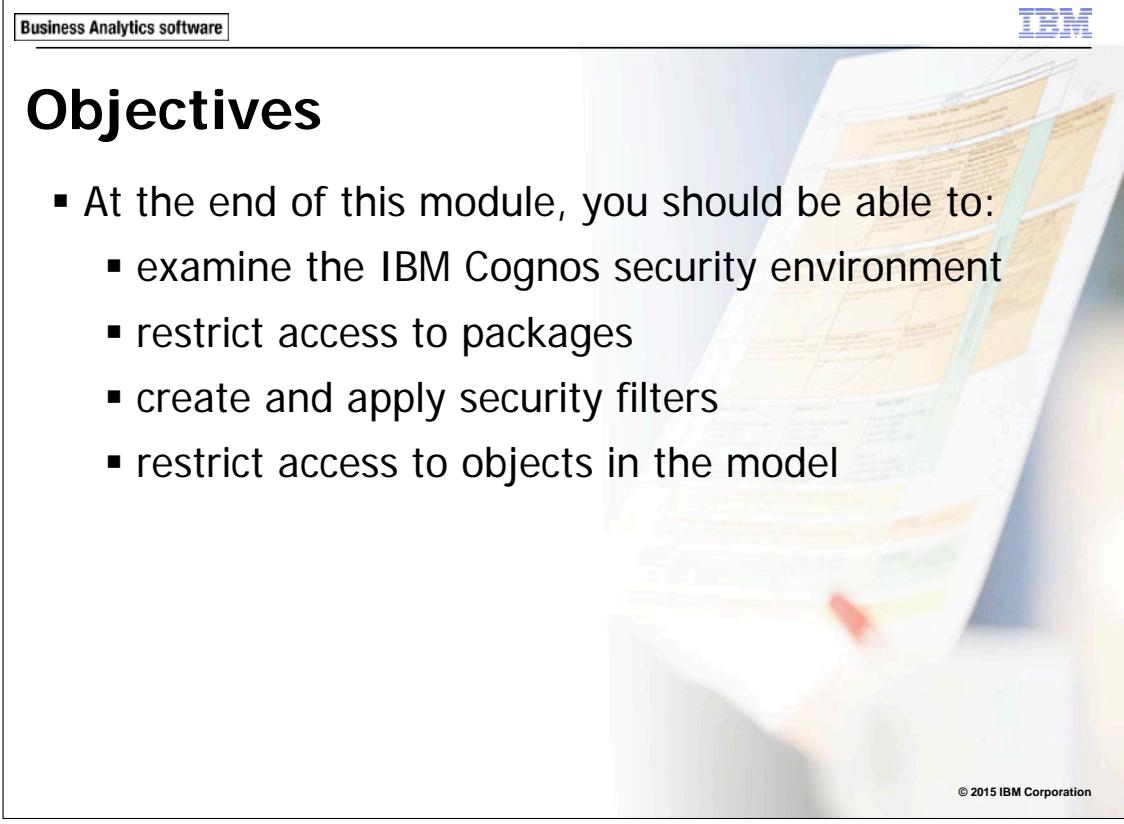
14-2

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Objectives

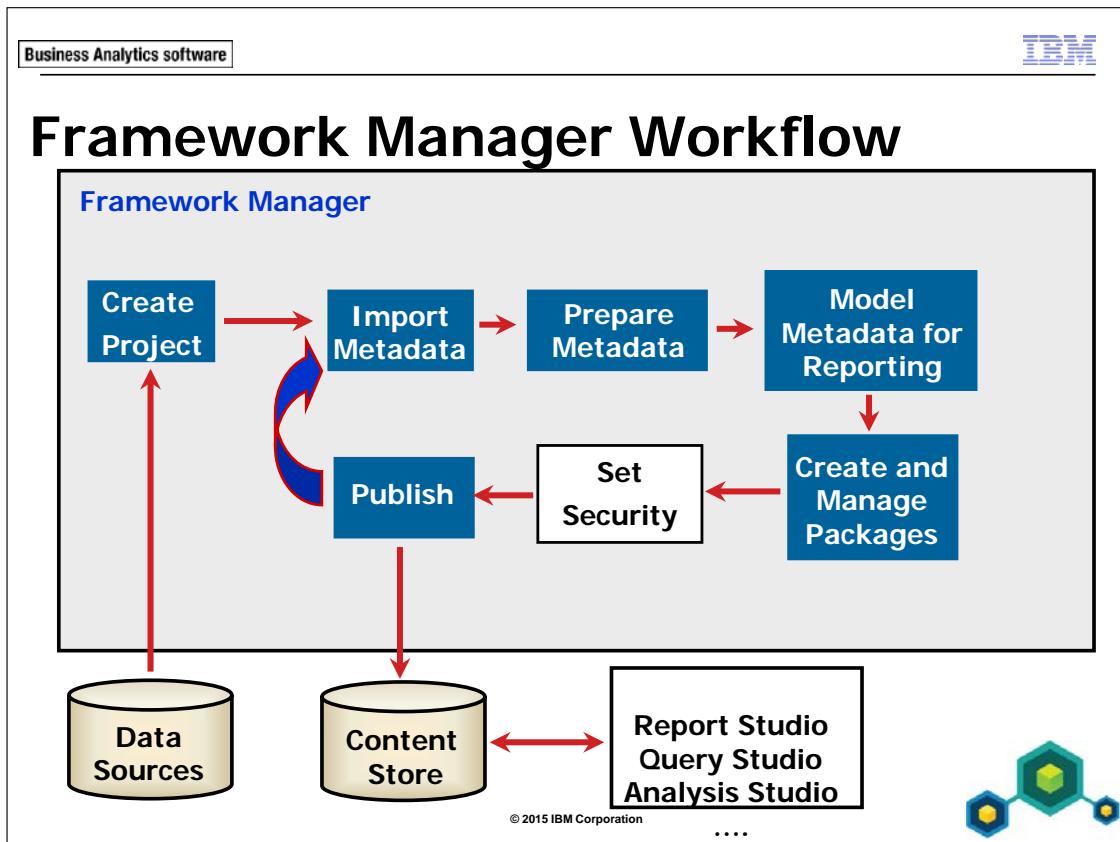
- At the end of this module, you should be able to:
 - examine the IBM Cognos security environment
 - restrict access to packages
 - create and apply security filters
 - restrict access to objects in the model



© 2015 IBM Corporation

Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

- User ID: admin
- Password: Education1



This module provides an overview of how security is defined in the IBM Cognos Business Intelligence environment, and demonstrates how you can apply additional security to a package in Framework Manager.

Administrators use IBM Cognos Administration to define most, if not all, of the security required for their organization. While you can define security for model objects in Framework Manager, this feature is only available the first time you publish a package, and is only intended to be a supplement to the security defined by the Administrator.

Overview of IBM Cognos Security

- there are three possible layers of security:

3rd-Party Authentication	authenticates users and secures objects
Cognos Namespace	uses IBM Cognos groups and roles to secure applications
capabilities and UI profiles	grant or deny users access to IBM Cognos components or features

- security is optional - anonymous access is allowed

© 2015 IBM Corporation



IBM Cognos can leverage third-party authentication providers to authenticate users. You can use the provider to define and maintain users, groups, and roles. You set up and configure authentication providers in IBM Cognos Configuration. Each authentication provider known to IBM Cognos is referred to as a namespace.

IBM Cognos provides its own namespace called Cognos, which contains groups and roles that define user privileges in the IBM Cognos environment. The Cognos namespace is not used to authenticate; however, you can use it to enhance your security policies, and ensure the portability of your applications. For example, you can add users and groups from a third-party authentication provider to Cognos namespace groups and roles, in order to secure your application. If you port your application to a different environment, you can continue to use the Cognos groups and roles, and add the appropriate users or groups from the new authentication provider.

Security can be omitted entirely, or combined with anonymous access for open access to specific items. Typically, anonymous users have limited, read-only access. The anonymous authentication process does not require a user to provide logon credentials. The anonymous authentication uses a pre-defined account under which all anonymous users are logged in.

Applying Framework Manager Security

- control access for selected users, groups, and roles
- grant or deny access for:
 - packages
 - data
 - objects

© 2015 IBM Corporation



- Package security: controls which IBM Cognos BI users can author or run a report that uses the package. Users without access to the package can view saved report outputs if they have been granted access to those reports. You can give administrative access to packages, for users who are required to republish packages or perform impact analysis based on model changes.
- Data security: restricts the data returned by query subjects, by using a security filter that is applied to the specific users, groups, or roles that require that query subject.
- Object security: restricts access to objects, such as query subjects, query items, and filters.

A user who only has package-level access cannot set properties (including permissions), or edit the entry in IBM Cognos Connection. These access rights are assigned through IBM Cognos Connection by a user who has access to package administration features.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Assigning Permissions

- allow or deny the following permissions:

Read: view entries

Write: delete entries, create entries in a container (i.e. package or folder), and modify report specs/output

Execute: process entries (i.e. run a report)

Set Policy: view and modify security settings for entries

Traverse: view container contents and properties

© 2015 IBM Corporation



Adding Security in the Publish Wizard

- the Add Security page lets you add security to a package the first time you publish it
- add users, groups, or roles to either:

User Access	Read, Write, Execute, and Traverse permissions
Administrator Access	Read, Write, Set Policy, and Traverse permissions.
- after the initial publish, you can modify package permissions in the Permissions dialog box

© 2015 IBM Corporation



You can access the Permissions dialog box for a package through either:

- IBM Cognos Connection, by clicking the Properties icon for the package, and then clicking the Permissions tab.
- In Framework Manager, by selecting the package in the Project Viewer, and then clicking the Action > Package > Edit Package Security menu item.

Demo 1: Specify Package Access

Purpose:

You want to set up security to control who can access the GO Operational (query) package. You also want to ensure that only System Administrators can administer the package. To do this, you will use the Publish wizard to assign the required permissions to existing Cognos BI user groups. You will then test the results in IBM Cognos Connection by logging in as a system administrator, query user, and report author.

Components: **Framework Manager, IBM Cognos Connection**

Project: **GO Operational**

Package: **GO Operational (query)**

Task 1. Provide administrator privileges to the GO Operational (query) package for system administrators.

Because you can only specify security in the Publish wizard on the first publish of a package, you will first remove the package from the Content Store which will reset the publish history. You can also configure admin access for a package in the package properties.

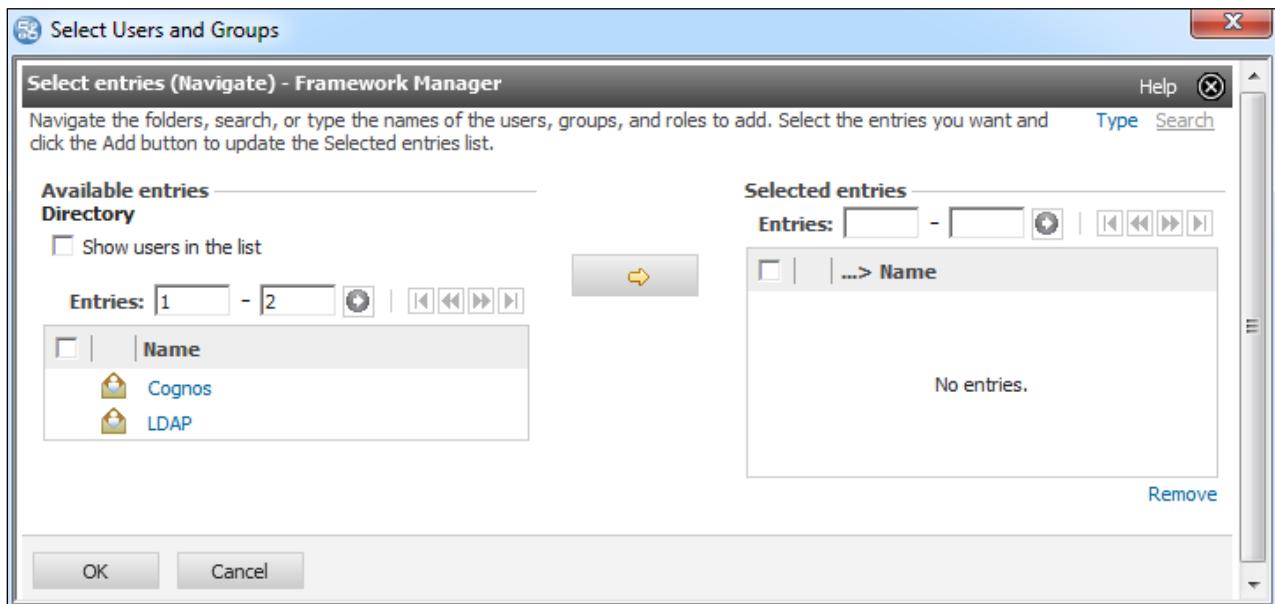
1. In **IBM Cognos Connection**, navigate to **Public Folders**.
2. Click the **GO Operational (query)** checkbox, click the **Delete** icon, and then click **OK**.
3. In **Framework Manager**, open **GO Operational.cpf** located in **C:\Edcognos\B5A52\CBIFM-Start Files\Module 14\GO Operational**. If prompted, log in as User ID **admin**, and Password **Education1**.
4. Expand **Packages**, right-click **GO Operational (query)**, click **Publish Packages**, and then click **Next**.

Next you will grant User access to the package for the Query Users role, and Administrator access to the System Administrators role.

5. In the **User Access** tab, click **Add**.

Note: If the GO Operational (query) package already existed in Cognos Connection this button would not be enabled.

The results appear as follows:



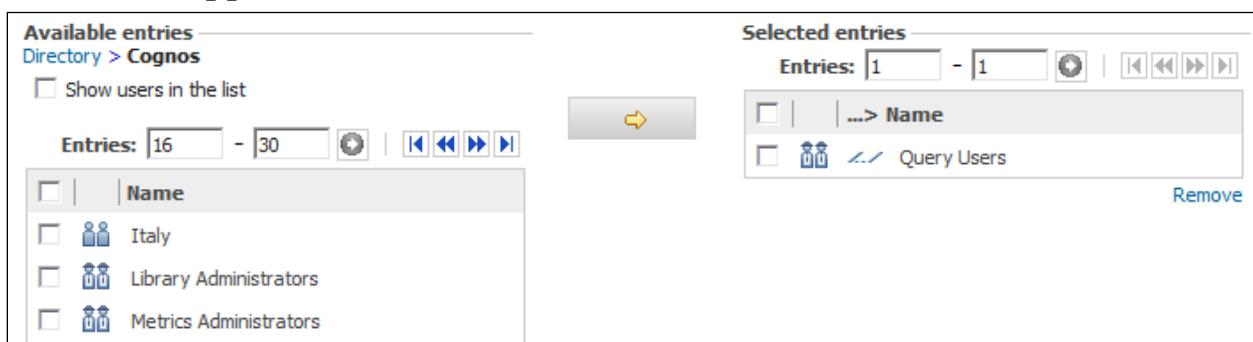
This is the same dialog box that is used to add security in IBM Cognos Connection. Framework Manager is simply providing a portal to it.

There are two sources to identify, security users, groups, and roles.

- Cognos: contains Groups and Roles defined within IBM Cognos BI.
- LDAP: contains Users and Groups defined within the Apache Directory Authentication Server.

6. Click **Cognos**, click **Next Page** , select **Query Users**, and then click **Add** .

The results appear as follows:



7. Click **OK** to close the dialog.

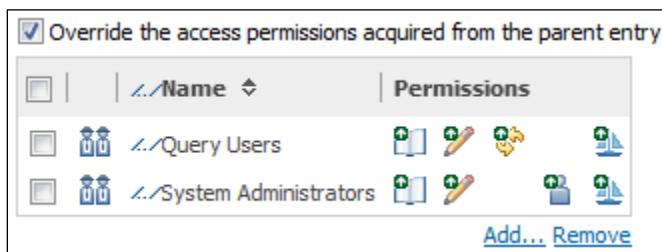
This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

8. Click the **Administrator Access** tab, and then click **Add**.
9. Click **Cognos**, click **Next Page**  twice.
10. Select **System Administrators**, and then click **Add** .
11. Click **OK** to close the dialog.
12. Click **Next**, and then deselect the **Verify the package before publishing** check box.
13. Click **Publish**, and then click **Finish**.

Task 2. Test the package security.

1. In **IBM Cognos Connection**, click **Refresh** .
2. Beside **GO Operational (query)**, click **Set Properties** .
3. Click the **Permissions** tab.

The results appear as follows:



You can use this dialog to add or delete Users, Groups, and Roles, and to assign them permissions. The "Override the access permissions acquired from the parent entry" check box is selected because you added roles, which overrides the default setting of the parent container (Public Folders).

4. Hover the cursor over each of the **Permissions** icons to display the descriptions.

The existing roles have the following permissions:

- Query Users: Read, Write, Execute, and Traverse
- System Administrators: Read, Write, Set Policy, and Traverse

5. Click **Cancel** to close the dialog.
6. On the title bar, beside **Admin Person**, click **Log Off**.

You will now log on as a Query User member to see the effects of package access security.

7. **Log on again as whites** (Sally White) (password = **Education1**).

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

8. Click **IBM Cognos content**, and then beside **GO Operational (query)**, click  **Set Properties**.

Notice that there is no Permissions tab. This is because Sally White is only a member of the Query Users role, and not the System Administrators role. You are able to see the package in Public Folders (and can access it in a report studio), but you cannot view or edit its permissions without the Set Policy permission.

You will now log on as a user from the Authors role.

9. Click **Cancel**, and then, on the title bar beside **Sally White**, click **Log Off**.
10. **Log on again as brettonf** (Frank Bretton) (password = **Education1**).
11. Click **IBM Cognos content**.

Notice the GO Operational (query) package is not visible. This is because Frank Bretton is only a member of the Authors role, which is not assigned access to this package.

Next, you will give the Authors role user access permissions, and then limit the extent of that access.

Task 3. Define package permissions in Framework Manager.

1. In **Framework Manager**, select the **GO Operational (query)** package.
2. From the **Actions** menu, point to **Package**, and then click **Edit Package Settings**.
3. Click the **Permissions** tab.
You have access to the Permissions tab because you are still logged on as admin within Framework Manager.
4. Click **Add**, and then click **Cognos**.
5. Select the **Authors** check box, and click the **Add** button, and then click **OK**.

- Select the **Authors** check box, and select the **Grant** check box beside **Traverse**, and then select the **Deny** check box beside **Read**, **Write**, and **Execute**.

The results appear as follows:

	<input type="checkbox"/> Grant	<input checked="" type="checkbox"/> Deny
Read	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Write	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Execute	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Set Policy	<input type="checkbox"/>	<input type="checkbox"/>
Traverse	<input checked="" type="checkbox"/>	<input type="checkbox"/>

While Read, Write, and Execute permissions have been explicitly denied, Set Policy is only implicitly denied (by not being granted). You will learn more about this distinction later in the module.

- Click **OK**, and then publish the **GO Operational (query)** package, clicking **Yes** to overwrite the existing version.

Task 4. Test access in IBM Cognos Workspace Advanced.

- In **IBM Cognos Connection**, still logged on as **Frank Bretton**, click **Refresh**. The GO Operational (query) package now appears.
- From the **Launch** menu, click **IBM Cognos Workspace Advanced**. Notice that the GO Operational (query) package is no longer an active link.
- Click **Cancel**, and then click **GO Operational (query)**.
- While in the **GO Operational (query)**, from the **Launch** menu, click **IBM Cognos Workspace Advanced**.
- Click **Create New**, and then double-click **List**.

The results appear as follows:

QE-DEF-0157

X The model or package /content/package[@name='GO Operational (query)']/model[@name='model'] does not exist or you are not allowed to use it because of security settings.

Details:

CAF-WRN-2082 An error has occurred. Please contact your administrator. The complete error has been logged by CAF with SecureErrorID:2014-11-05-17:38:57.617-#1

Frank Bretton cannot author reports with this package because he is a member of the Authors role, which you denied Read, Write, and Execute permissions.

- Click **OK**, and then close **IBM Cognos Workspace Advanced**.

Task 5. Edit package permissions and retest access.

1. In **Framework Manager**, select the **GO Operational (query)** package.
2. From the **Actions** menu, point to **Package**, and then click **Edit Package Settings**.
3. Click the **Permissions** tab, select the **Authors** check box, and then select the **Grant** check box for the **Read**, **Write**, and **Execute** permissions.
4. Click **OK**.
5. Publish the **GO Operational (query)** package, clicking **Yes** to overwrite the existing package.
6. In **IBM Cognos Connection**, still logged on as **Frank Bretton**, launch **IBM Cognos Workspace Advanced**.
7. Click **Create new**, ensure **GO Operational (query)** package is selected and then double-click **List**.
IBM Cognos Workspace Advanced opens, and Frank Bretton has access to the contents of the package.
8. Close the report, and then in **Framework Manager**, **Save** the project.

Results:

You used both the Publish Wizard and Permissions dialog to set security for the GO Operational (query) package. You granted different permissions to Query User and Author roles, and tested the impact of those settings in IBM Cognos Connection and IBM Cognos Workspace Advanced.

Specifying Data Security

- to specify data security:
 - add the groups/roles that will access the data (i.e. the Italy group)
 - specify a security filter for a query subject (i.e. Retailer.Region='Italy')
- data security affects:
 - authors when they create a report
 - users when they run a report

© 2015 IBM Corporation



If a user does not belong to any of the groups specified in security filters, they will have unrestricted access to the data, regardless of the restrictions specified by the filter expressions. In some cases, you may want to avoid this scenario and completely restrict access to particular groups or roles. To do this, you can create a filter expression that will always resolve to a false outcome, such as:

[Consolidation View].[Products].[Product Line Code] = 1

where 1 does not exist in the data for Product Line Code. You could then specify the groups or roles to which this filter expression will be applied. The result is that a user who is a member of the defined groups or roles for this filter expression will be denied access to the data, provided they are not members of a group that does have access to the data.

The filter expression can incorporate macros, parameter maps, and session parameters. You can base the security filter on existing security filters, in which case the new security filter inherits the existing filter and all its properties. You can also use existing project filters, or create new filters using the Expression Editor.

Demo 2: Specify Data Security

Purpose:

Sales managers at The Sample Outdoors Company want to ensure that Camping Equipment sales reps only see orders related to the Camping Equipment product line. To do this, you will create and add members to the Sales Managers and Camping Equipment Reps groups. You will then grant these groups access to the GO Operational (query) package. Next, you will apply a security filter to the Products query subject that restricts access to camping equipment data.

Components: Framework Manager, IBM Cognos Workspace Advanced

Project: GO Operational

Package: GO Operational (query)

Task 1. Grant access to the GO Operational (query) package.

1. In **Framework Manager**, select the **GO Operational (query)** package.
2. From the **Actions** menu, point to **Package**, and click **Edit Package Settings**.
3. Click the **Permissions** tab, click **Add**, and then click **Cognos**.
4. Select the **Camping Equipment Reps** check box, and click the **Add** button.
5. Click **Next Page** twice, select the **Sales Manager** check box, and then click the **Add** button.
6. Click **OK**.
7. Select the **Camping Equipment Reps** and **Sales Manager** check boxes.

8. Under **Grant**, select the **Read**, **Write**, **Execute**, and **Traverse** check boxes.
- The results appear as follows:

Name	Permissions	Grant	Deny
...>Authors		<input type="checkbox"/>	<input type="checkbox"/>
...>Camping Equipment Reps		<input checked="" type="checkbox"/>	<input type="checkbox"/>
...>Query Users		<input type="checkbox"/>	<input type="checkbox"/>
...>Sales Manager		<input checked="" type="checkbox"/>	<input type="checkbox"/>
...>System Administrators		<input type="checkbox"/>	<input type="checkbox"/>

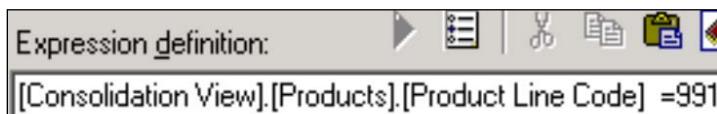
9. Click **OK**.

Task 2. Create a security filter for the Product query subject.

1. In the **Project Viewer** pane, expand **GO Operational Model > Consolidation View**.
 2. Click **Products**, and then from the **Actions** menu, click **Specify Data Security**.
 3. Click **Add Groups**.
 4. Click **Cognos**, select the **Camping Equipment Reps** check box, and then click the **Add** button.
 5. Click **OK**.
- At this point, if you already had another group defined with a filter on camping equipment products, you could click below **Based On** and specify that other group. In this situation you do not, so you will create a new filter.
6. In the first row, click the **Filter** value, and then select **Create/Edit Embedded** from the drop-down list.
 7. In the **Available Components** pane, expand **GO Operational Model > Consolidation View > Products > Codes**.

8. Double-click **Product Line Code**, and in the **Expression definition** pane, at the end of the expression, type **= 991**.

The results appear as follows:



This filter will ensure that members of the Camping Equipment Reps will only see camping equipment products.

9. Click **OK** twice, and then **Save** the project.

Task 3. Publish the package and view the results in IBM Cognos Workspace Advanced.

1. Publish the **GO Operational (query)** package, overwriting the existing package.
2. In **IBM Cognos Connection**, log on as **uragomek** (password = **Education1**).
Uragomek is a member of the Sales Manager group.
3. In the **Welcome** screen, click **Author business reports**, select the **GO Operational (query)** package, and create a new **List** report.
4. In the **Source** pane, expand **Sales (query) > Products**, and then double-click **Product Line** and **Product Type**.

The results appear as follows:

Product Line	Product Type
Personal Accessories	Binoculars
Mountaineering Equipment	Climbing Accessories
Camping Equipment	Cooking Gear
Personal Accessories	Eyewear
Outdoor Protection	First Aid
Golf Equipment	Golf Accessories
Outdoor Protection	Insect Repellents
Golf Equipment	Irons
Personal Accessories	Knives
Camping Equipment	Lanterns

Kazumi Uragome can view data for all product lines. He is a member of the Sales Managers group, so the security filter you applied to Camping Equipment Reps does not apply to this group.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

5. On the toolbar, click **Save** , in the **Name** box, type **Security Filter Test**, and then click **Save**.
6. Close **IBM Cognos Workspace Advanced**, log off, and then log on again as **kunzej** (password = **Education1**).
Jörg Kunze is a member of the Camping Equipment Reps group.
7. Click **IBM Cognos content**, select the **GO Operational (query)** folder, and then click **Security Filter Test**.

The result appears as shown below:

Product Line	Product Type
Camping Equipment	Cooking Gear
Camping Equipment	Lanterns
Camping Equipment	Packs
Camping Equipment	Sleeping Bags
Camping Equipment	Tents

Due to the data security you added to the Product query subject for the Camping Equipment Reps group (of which Jörg Kunze is a member), the report only displays product types from the Camping Equipment product line. Note: if you added Jörg to the Sales Managers group (which can see all lines), his membership in Camping Equipment Reps would not block him from viewing other product lines.

8. **Close** the browser.

Results:

You granted the Sales Manager and Camping Equipment Reps groups access to the GO Operational (query) package. You then applied a security filter to the Products query subject to restrict access to product line data. Finally, you published the package and viewed the results as different users in IBM Cognos Workspace Advanced.

Specifying Object Security

- allow or deny access to objects such as:
 - namespaces
 - folders
 - query subjects
 - query items
 - filters

© 2015 IBM Corporation



When granting access to objects, ensure that the selected users, groups, or roles have access to the package that contains them. By default, the Everyone group is on the access control list of every object.

An example of applying object security is to only allow access to the Sales Targets namespace to the Sales Manager group. Then only members of the Sales Manager group will see the Sales Target namespace in the studios.

When you publish a package that contains secured objects, the visible objects for IBM Cognos users are the intersection of the package definition and the object security applied to those objects.

If you run a report and you do not have access to a query subject or query item referenced in the report, the report will fail. However, you can still view saved report outputs.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

14-20

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Object Security Rules

- if no object security is applied, all objects are accessible
- after applying object security, only objects to which access has been granted can be accessed by the selected users, groups, or roles
- child objects inherit the security of parent objects
- Deny overrides Allow

© 2015 IBM Corporation



When you begin to apply object security (Allow or Deny), all other objects (other than the children of the object you specified security on) are hidden from users until access is explicitly granted. Only the objects to which access has been explicitly granted for the selected users, groups, or roles, are visible.

When you set object security on a parent object, a child object inherits the security of the parent if object security has not already been specified for the child object.

In the case of an access conflict, such as being a member of two groups with conflicting access to an object, denied access to the object overrides granted access to it. For example, to deny access to the Sales Targets query subject for the Sales Reps group, you will have to add the Sales Reps group to the Sales Target query subject's access control list. You must then deny access for the Sales Reps group. As soon as you do this the Sales Reps group will be added to the access control list of every object.

Object Security Methodologies

- two ways to implement object-based security:
 - allow access to all objects, and then restrict access to specific objects for selected users, groups or roles
 - restrict an object, and then allow access as required

© 2015 IBM Corporation



- Allow access to all objects, and then restrict access to specific child objects for selected users, groups, or roles.
You can either explicitly deny access (Deny option) or implicitly deny access by leaving both Allow and Deny options unselected. Deny overrides allow, which ensures that a user does not accidentally gain access to an object through another group or role that has access.
- Restrict an object, and then allow access as required.
Applying initial security to an object that allows specific users, groups or roles access will automatically implement security across the model and make all other objects inaccessible to all users. You can then grant access to specific objects for selected users, groups, or roles as required.

You can remove all object security from the model by double-clicking the Packages folder in Project Viewer , and deleting the Everyone role-based package. This will also remove all other role-based views.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Demo 3: Specify Object Security

Purpose:

In order to run reports with the projected sales targets for each sales rep, managers need access to all data in GO Operational Sales (query), including the Sales Target namespace. However, you want to ensure that sales reps cannot access projected sales targets for every sales rep. To support these requirements, you will create a Sales Reps group and grant it access to the GO Operational (query) package. You will then assign the Sales Manager and Sales Reps groups access to the appropriate objects.

Components: IBM Cognos Connection, Framework Manager, IBM Cognos Workspace Advanced

Project: GO Operational

Package: GO Operational (query)

Task 1. Create a new Sales Reps group and add a member.

1. In **IBM Cognos Connection**, log on as **admin** (password = **Education1**).
2. Under **Administration**, click **Administer IBM Cognos content**, and then click the **Security** tab.
3. Click the **Cognos** namespace, and then click **New Group** .
4. In **Name**, type **Sales Reps**, and click **Next**.
5. Click **Add**, and then click **LDAP**.
6. Click **Show users in the list**, and click **People**.
7. In the upper-right, click the **Search** link, type **Bart Scott**, and then click the **Search** button.
8. Select the **Bart Scott** check box, and then click the **Add** button.
9. In the search field, type **Daniel Turpin**, and click the **Search** button.
10. Check the box next to **Daniel Turpin**, and then click the **Add** button.
11. Click **OK**, and then click **Finish**.

Task 2. Grant access to the GO Operational (query) package.

1. In **Framework Manager**, in the **Project Viewer** pane, select the **GO Operational (query)** package.
2. From the **Actions** menu, point to **Package**, and then click **Edit Package Settings**.
3. Click the **Permissions** tab, click **Add**, and then click **Cognos**.
4. Click **Next Page** twice, select the **Sales Reps** check box, and then click the **Add** button.
5. Click **OK**, select the **Sales Reps** check box, and then under **Grant**, select the **Read**, **Write**, **Execute**, and **Traverse** check boxes.

The results appear as follows:



6. Click **OK**.

Task 3. Specify object security on the GO Operational Sales (query) namespace for the Sales Managers and Sales Reps groups.

1. In **Project Viewer**, under **Presentation View**, click the **GO Operational Sales (query)** namespace.
2. From the **Actions** menu, click **Specify Object Security**, and then click **Add**.
3. Click **Cognos**, click **Next Page** twice, and then select the **Sales Manager** and **Sales Reps** check boxes.
4. Click the **Add** button, and then click **OK**.
5. Under **Allow**, select the **Sales Manager** and **Sales Reps** check boxes, and then click **OK** twice.

The security on the GO Operational Sales (query) namespace is inherited by all its children. You will now restrict access to the Sales Targets (query) namespace.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Task 4. Specify object security on the Sales Targets (query) namespace for the Sales Reps group.

1. In the **Presentation View**, expand the **GO Operational Sales (query)** namespace, and click the **Sales Targets (query)** namespace.

2. From the **Actions** menu, click **Specify Object Security**.

The Specify Object Security box appears. Notice that the Sales target namespace has inherited the security that was applied to its parent object, the GO Operational Sales (query) namespace.

3. Beside **Sales Reps**, select the **Deny** check box, and then click **OK**.

A message appears, indicating that the security for the Sales Targets (query) namespace will override the settings already specified for the GO Operational Sales (query) namespace.

Rather than select deny, you could also just have deselected the Allow setting for Sales Reps, which would have implicitly denied access, but the method you have implemented ensures members of this group will not have access regardless of any other group they belong to.

4. Click **OK**, and then save the project.

Task 5. View the results in IBM Cognos Workspace Advanced.

1. Publish the **GO Operational (query)** package, overwriting the existing package.

2. In **IBM Cognos Connection**, log off, and then log on again as **uragomek** (password = **Education1**).

3. Under **My Actions**, click **Author business reports**, and then click **GO Operational (query)**.

As a member of the Sales Manager group, the current user has access to all namespaces in the package, including the Sales Targets (query) namespace.

4. Click **Create new**, and then double-click **List**.

5. In the right pane, expand all of the namespaces.

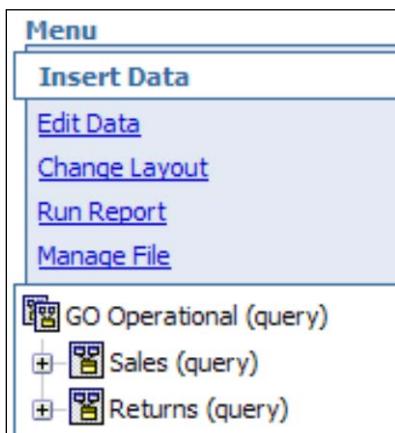
Notice that they are empty, and there are no query subjects available. You will investigate this issue later in Framework Manager.

6. Close **IBM Cognos Workspace Advanced**.

7. Click **Log off**, and then **Log on again** as **turpind** (Daniel Turpin) (password = **Education1**).

8. Click **Query my data**, and then click **GO Operational (query)**.

The results appear as follows:



As a member of the Sales Reps group, the current user does not have access to the Sales Target (query) namespace. This group has been explicitly denied access to this object.

9. Close the browser.

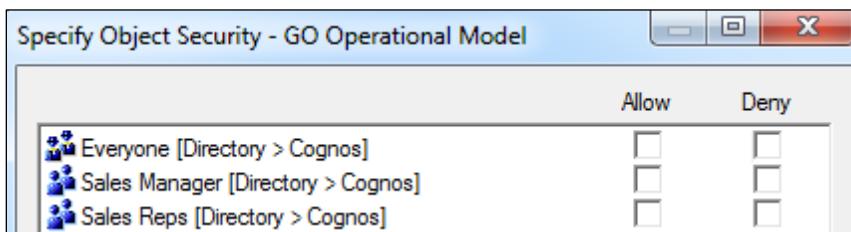
Task 6. Grant access to objects pointed to by shortcuts.

1. In **Framework Manager**, expand **Presentation View > GO Operational Sales (query) > Sales (query)**.

When you granted access to the GO Operational Sales (query) namespace, the security was inherited at all levels below it. However, those levels only contain shortcuts to the query subjects, and security on shortcuts is not inherited by underlying objects. To ensure that users can access underlying objects, and then restrict objects as required, you will grant access to everyone for model objects that do not have object security configured.

2. Click the **GO Operational Model** namespace (the root namespace), and then from the **Actions** menu, click **Specify Object Security**.

The results appear as follows:

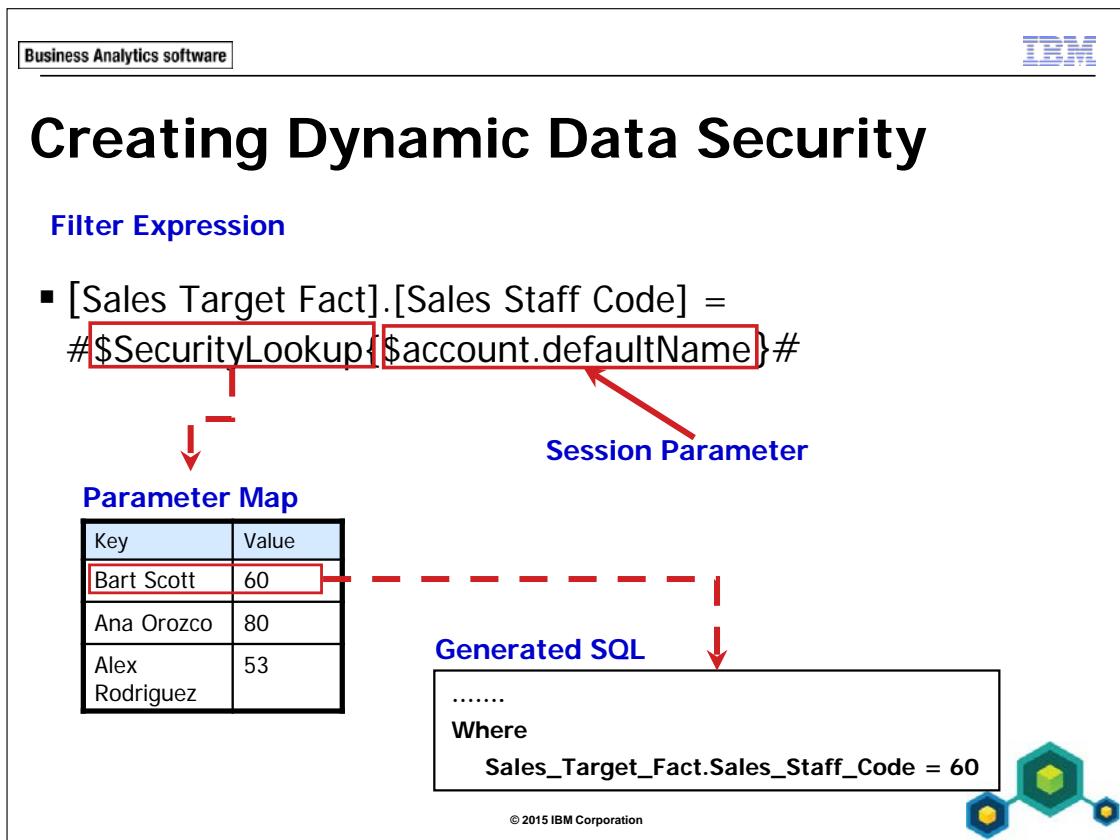


Notice that no one is allowed access. By allowing access to everyone, all objects will inherit this setting with the exception of the objects you already configured.

3. Under **Allow**, select the **Everyone** check box, and then click **OK**.
You can use a different group to open up access to users, such as the All Authenticated Users group.
4. Save the project, and then publish the **GO Operational (query)** package.
5. In **IBM Cognos Connection**, log on as **turpind** (password = **Education1**).
6. Launch **IBM Cognos Workspace Advanced**, and select the **GO Operational (query)** package for a **List** report.
Daniel Turpin cannot see the Sales Targets (query) namespace because he is part of the Sales Reps group, which has been denied access to the namespace. Notice that the Retailer Location Filters folder is now visible. This is also a shortcut in the GO Operational Sales (query) namespace. Now that the underlying object is available to everyone, it is visible.
7. Expand **Sales (query) > Sales Fact**.
You now see the query subjects and query items.
8. Close the browser.

Results:

You created a Sales Reps group and granted it, and the Sales Managers group, access to the GO Operational Sales (query) namespace. You then denied Sales Reps access to the underlying Sales Targets (query) namespace. After publishing the GO Operational (query) package and viewing the results in Query Studio, you identified an issue with your security implementation. You then opened up security on all remaining objects in the project at the root namespace to fix the problem.



The above slide illustrates an advanced technique for applying security. In this example, we use a macro to look up the current user in a parameter map and provide an appropriate value for the Where clause of the generated SQL. If Bart Scott was logged on, his account.defaultName would be sent to the parameter map and be substituted for a value of 60. Then the Where clause in the generated SQL would equate to:

Where Sales_fact_employee_secured.STAFF_KEY = 60

You can use environment session parameters in a macro to create a dynamic data security filter when you want the security to only apply to specific groups. You could also use the macro in the slide example above directly in the query subject, but then the security would be applied to everyone using the query subject rather than specific groups or people. For example, you want members of the Sales Managers group to have access to all sales targets and not be restricted. Conversely, you want to restrict members of the Sales Reps group to only have access to their own data and not other sales reps. Data security filters can accomplish this.

Demo 4: Use a Macro in a Data Security Filter

Purpose:

You have been asked to publish a package in which members of the Sales Manager group can view sales targets for all employees, but members of the Sales Reps group can only see their own sales targets. To do this, you will create a macro that references an environment session parameter in order to implement data security for a particular group on a query subject.

Components: Framework Manager, IBM Cognos Workspace Advanced

Project: GO Operational

Package: GO Operational (query)

Task 1. Change the security from the last demo.

In order to allow Sales Reps members access to their own sales targets data, you will need to grant Sales Reps members access to the Sales Targets (query) namespace.

1. In **Framework Manager**, under **Presentation View > GO Operational Sales (query)**, click the **Sales Targets (query)** namespace.
2. From the **Actions** menu, click **Specify Object Security**.
3. Select **Allow** for **Sales Reps**, and then click **OK**.

Task 2. Create a parameter map.

1. In the **Project Viewer** pane, right-click **Parameter Maps**, point to **Create**, and then click **Parameter Map**.
2. In the **Name** box, type **SecurityLookup**, select **Base the parameter map on existing Query Items**, and then click **Next**.
3. Expand **Consolidation View > Staff by Location**.
4. Click **Staff Full Name**, and then click **Set as Key**.
5. Expand the **Codes** folder, select **Sales Staff Code**, and then click **Set as Value**.

Staff Full Name acts as the unique key for the parameter map and Sales Staff Code acts as the substitution value that will be used to apply security in a data security filter.

By basing this parameter map on existing query items in the data source, the parameter map will always reflect the latest data updates. As new employees join the company, they will automatically be reflected in the parameter map.

To optimize this approach, you could filter the query subject used to feed the parameter map to make it more efficient. In this case, first make a copy of the query subject, filter it dynamically on Staff Full Name = account.defaultName, and then base the parameter map on the new query subject. It will only return one record when the parameter map is called, and prevents scanning the whole table. Since this is a relatively small table, you will simply use the existing query subject.

6. Click **Next**.

The results appear as follows:

Default value:	
Key	Value
Denis Pagé	10004
Élizabeth Michel	10005
Émile Clemont	10006
Étienne Jauvin	10007
Elsbeth Wiesinger	10012
Else Mörike	10013
Frank Fuhlroth	10014
Gunter Erler	10015
Björn Winkler	10016
Fritz Hirsch	10017
Jörg Kunze	10018

You can provide a default value that will be used if no match is found in the parameter map. In this demo, you do not want to return any data if there is no match, so you will leave the field blank.

7. Click **Finish**.

Task 3. Apply data security to a query subject.

You will now secure the original SALES_TARGET query subject in the Foundation Objects View so that the security is applied to all objects that reference it.

1. Under **Foundation Objects View > gosales**, click **SALES_TARGET**.
2. From the **Actions** menu, click **Specify Data Security**.
3. Click **Add Groups**, click **Cognos**, and then click **Next Page** twice.
4. Click the **Sales Reps** checkbox, and then click the **Add** icon.
5. Click **OK**.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

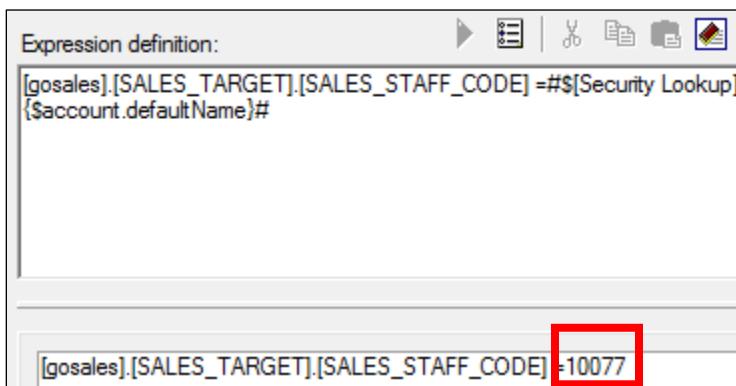
6. In the first row, click the **Filter** value, and select **Create/Edit Embedded** from the drop-down list.
7. Under **Available Components**, expand **Foundation Objects View > gosales > SALES_TARGET**.
8. Double-click **SALES_STAFF_CODE** to add it to the expression definition.
9. At the end of the expression, type **=**.
10. Under **Available Components**, click the **Parameters** tab, expand **Parameter Maps**, and then double-click **SecurityLookup** to add it to the expression.
11. In the **Expression definition**, place the cursor between **{}** of the **#\$SecurityLookup{}#** expression.
12. Under **Available Components**, collapse **SecurityLookup**, expand **Session Parameters**, and then double-click **account.defaultName** to add it to the expression.

The red line in the expression definition indicates that there is an error with the expression. This is due to the fact that you are currently logged in as admin, which does not exist in the data, and therefore cannot resolve the filter.

You will now override the **account.defaultName** value in order to test and validate this filter.

13. Click **Options** , and then click **Set**.
14. In the **Override Value** field for **account.defaultName**, type **Daniel Turpin**.
15. Click **OK** twice.

The results appear as follows:



Notice that the red line is gone and that the expression now resolves to a value.

16. Click **OK** twice, and then save the project.

Task 4. Publish and test the package in IBM Cognos Workspace Advanced.

1. Publish the **GO Operational (query)** package, overwriting the existing package.
2. In **IBM Cognos Connection**, ensure you are still logged on as **turpind** (password = **Education1**).
Remember, Daniel Turpin is a member of the Sales Reps group.
3. Launch **IBM Cognos Workspace Advanced**, and select the **GO Operational (query)** package.
4. Click **Create New**, and then double-click **List**.
5. In the **Source** pane, expand **Sales Targets (query) > Staff by Location**.
6. Add **Staff Full Name** to the report.
All names are returned since the filter is on the sales target data.
7. Expand **Sales Target Fact**, and then add **Sales Target** to the report.
The results appear as follows:

Staff Full Name	Sales Target
Daniel Turpin	30,808,000

The report is now limited to sales target for Daniel Turpin.

8. Save the report as **Macro Security Filter Test**.
9. Click **Return** , click **Log Off**, and then **Log on again** as **scottb** (password=**Education1**).
Bart Scott is also a member of the Sales Reps group.
10. Click **IBM Cognos content**, click **GO Operational (query)**, and then click **Macro Security Filter Test**.

The results appear as follows:

Staff Full Name	Sales Target
Bart Scott	38,209,900

Data is now limited to Bart Scott. You will now log on as a member of the Sales Manager group to test their access.

11. Close the report, click **Log off**, and then **Log on again** as **uragomek** (password=**Education1**).

12. Click IBM Cognos content, click **GO Operational (query)**, and then click **Macro Security Filter Test**.

The results appear as follows:

Staff Full Name	Sales Target
Aaltje Hansen	22,118,800
Abram Ruiz	44,933,900
Adda Heijman	21,862,800
Adriaantje Haanraads	24,468,300
Agatha Reyes	21,327,200
Agnelo Chavez	15,469,300
Agnes Ramos	25,934,800
Aidan Chaplin	10,978,300
Aiko Watanabe	39,354,200
Aila Forssell	18,515,400
Aimi Tanaka	14,553,010
Akemi Takahashi	43,558,700

The report returns all employees and their sales targets.

13. Close the browser.

Results:

You published a package in which members of the Sales Manager group can view sales targets for all employees, but members of the Sales Reps group can only see their own sales targets. You created a parameter map, and a macro that references an environment session parameter to implement data security for the Sales Reps group on the SALES_TARGET query subject.

Demo 5: Remove Security

Purpose:

In order to remove the impact of the security you applied in this module on future modules, you will remove the security applied throughout this module.

Components: **Framework Manager**

Project: **GO Operational**

Package: **GO Operational (query)**

Task 1. Remove security from the package.

1. In **Framework Manager**, in the **Packages** folder, select the **GO Operational (query)** package, and then go to **Actions > Package > Edit Package Settings**.
2. Click the **Permissions** tab, and then clear the **Override the access permissions acquired from the parent entry** check box.

A window opens explaining that this will cause the parent's policies to be acquired.

3. Click **OK.**

The results appear as follows:

<input type="checkbox"/> Override the access permissions acquired from the parent entry		
<input type="checkbox"/>	Name	Permissions
	//Analysis Users	
	//Authors	
	//Consumers	
	//Controller Administrators	
	//Controller Users	
	//Data Manager Authors	
	//Express Authors	
	//Metrics Administrators	
	//Metrics Authors	
	//Metrics Users	
	//Planning Contributor Users	
	//Planning Rights Administrators	
	//PowerPlay Administrators	
	//PowerPlay Users	
	//Query Users	
	//Readers	
	//Report Administrators	

This package is now available to all the roles that have access to the other packages found in the Public Folders area of IBM Cognos Connection.

4. Click **OK.**

Task 2. Remove data security.

1. In the **Project Viewer**, under **Consolidation View**, select **Products**.
2. From the **Actions** menu, click **Specify Data Security**.
3. With the **Camping Equipment Reps** group selected, click **Delete Group**, and then click **OK**.
4. Repeat steps 1 to 3 to delete the group from **Foundation Objects View > gosales > SALES_TARGET**.

Task 3. Remove object security.

1. In the middle pane, click **Explorer**.
2. In the **Project Viewer**, double-click the **Packages** folder to give it focus in the **Explorer**.

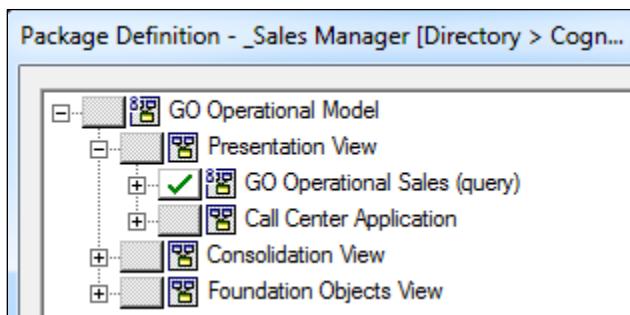
The results appear as follows:



Notice the security-based packages. These packages contain the security information you specified earlier.

3. Double-click **_Sales Manager** to view the definition.

The results appear as follows:



Here you can see the objects this group has access to.

4. Click **Cancel**.

You will now remove all object security by deleting the **_Everyone** package.

5. Select the **_Everyone** package, and then click **Delete**.

A message appears stating that this role is required for security implementation and that removing it will remove all object security from the model.

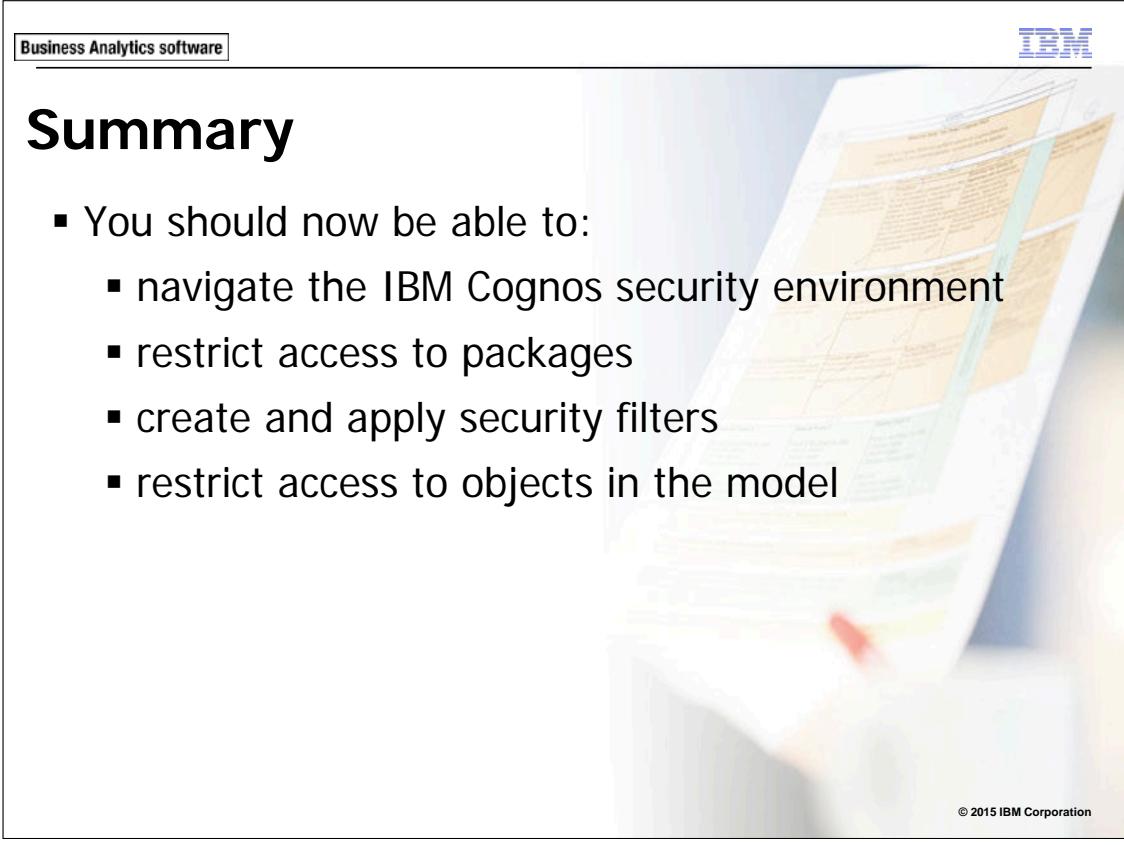
6. Click **OK**, and then **Save** and **Close** the project.

Results:

You removed the user and admin access restrictions you applied earlier by inheriting the GO Operational (query) parent's permissions setting. You also removed data security filters that you applied, and then removed all object security from the project by deleting the Everyone role security.

Summary

- You should now be able to:
 - navigate the IBM Cognos security environment
 - restrict access to packages
 - create and apply security filters
 - restrict access to objects in the model

A blurred background image shows a person's hands interacting with a large touchscreen display. The screen displays a complex interface with various charts, graphs, and data tables, suggesting a business intelligence or analytics application.

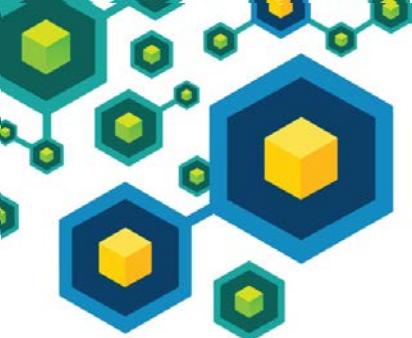
© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

14-38

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



Creating Analysis Objects

IBM Cognos BI

Business Analytics software



© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

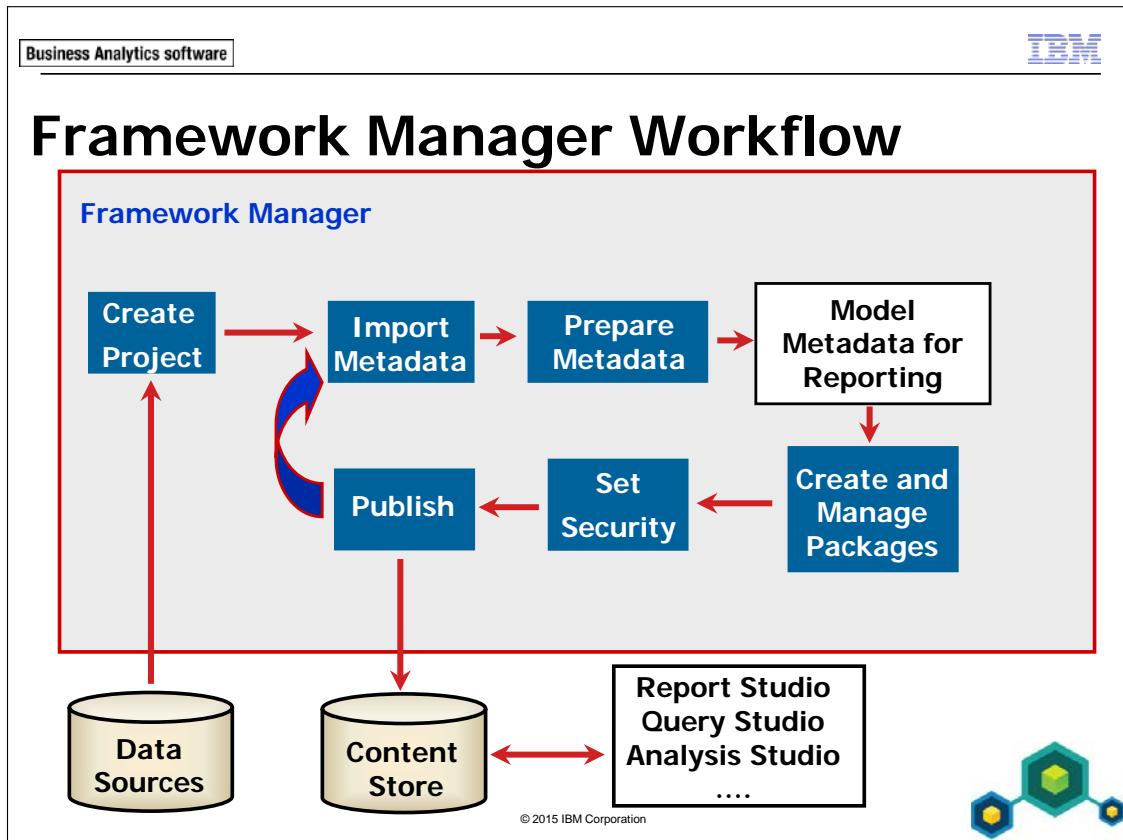
Objectives

- At the end of this module, you should be able to:
 - apply dimensional information to relational metadata to enable OLAP-style queries
 - sort members for presentation and predictability
 - define members and member unique names
 - identify changes that impact a MUN

© 2015 IBM Corporation

Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

- User ID: admin
- Password: Education1



This module deals with creating analysis objects that allow authors to perform OLAP-style queries.

Dimensionally Modeled Relational (DMR) Metadata

- dimensional metadata that enables OLAP-style queries on relational data sources
- defined by using:
 - Regular Dimensions
 - Measure Dimensions
 - Scope Relationships
- Dynamic Query Mode optimizes DMR performance

© 2015 IBM Corporation

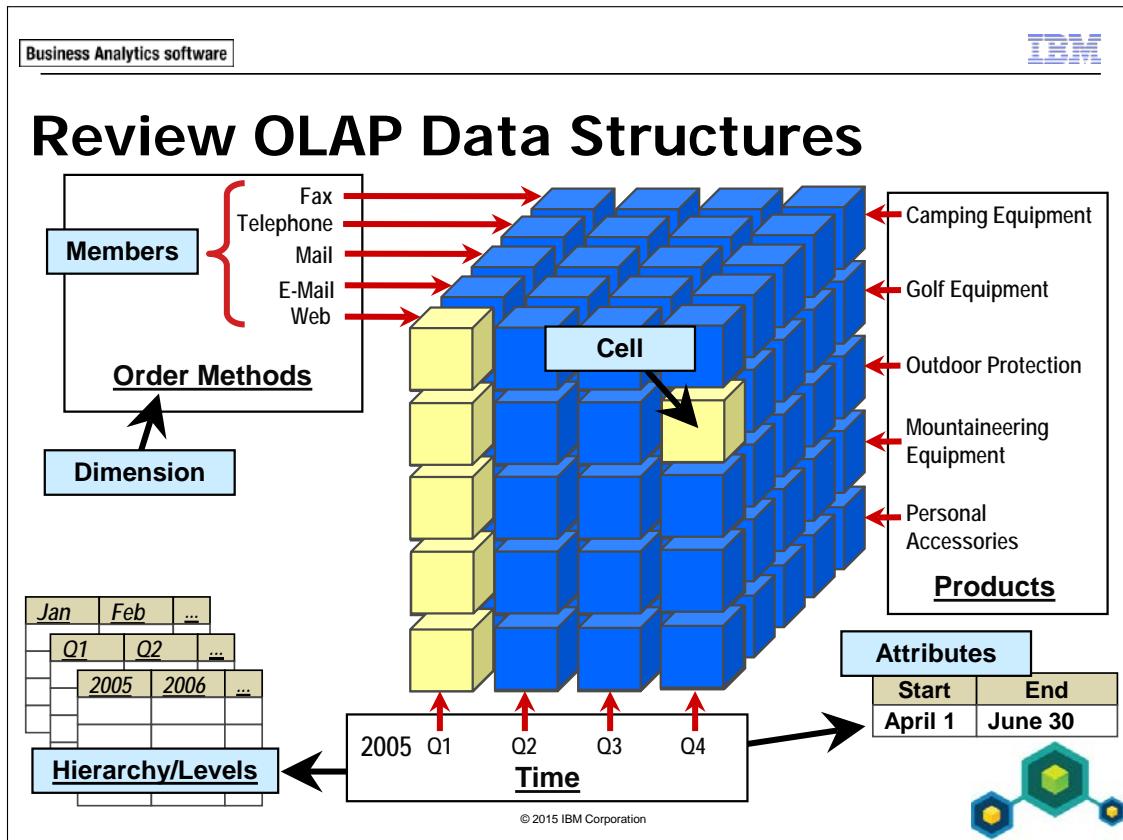


DMR metadata enables OLAP-style queries that include drill through functionality, as well as the ability to use dimensional functions, such as `parent([member])`.

You can provide dimensional information to any metadata that is in star schema format, such as adding hierarchy information to dimensions, and defining scope relationships for measures.

DMR metadata works best with projects that are set up to use Dynamic Query Mode (DQM). When you use DQM with DMR metadata you get:

- an increased ability to perform complex aggregation, and consistent results from OLAP functions
- a default sort order, which removes the requirement to manually specify sort options
- improved query execution performance



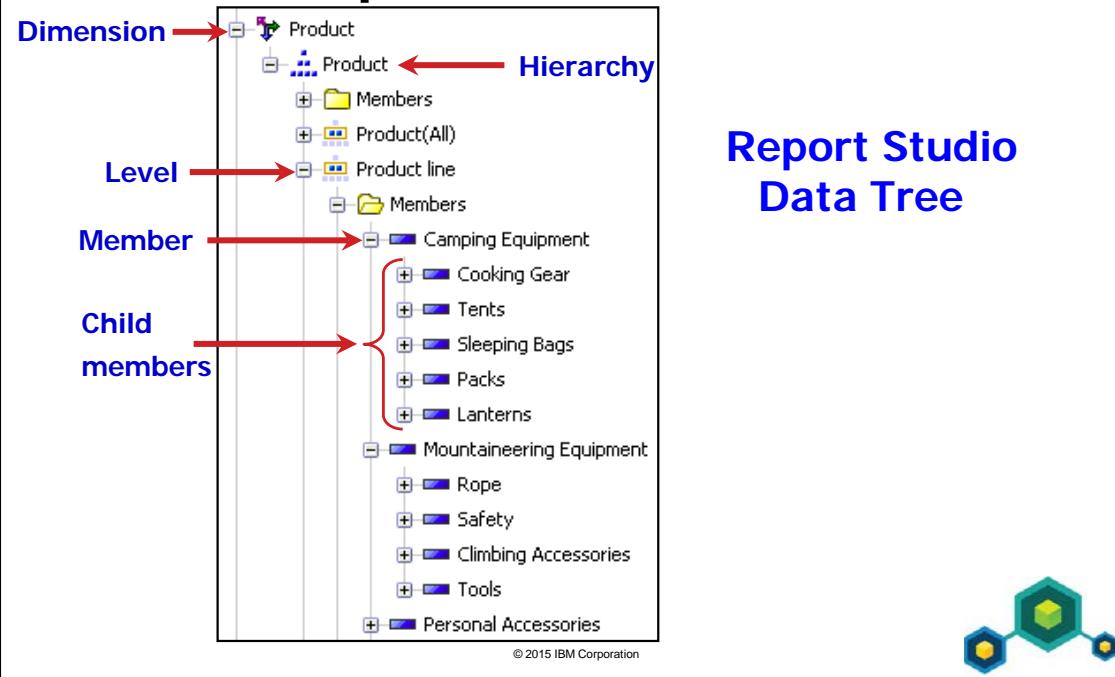
Providing dimensional information in your Framework Manager models allows IBM Cognos to create multi-dimensional OLAP (MOLAP) structures, such as the one above, at run time.

To optimize a model for large data sets, you can create a Dynamic cube with the IBM Cognos Dynamic Cube Designer, which offers Relational over OLAP style models. Note: Modeling with the Dynamic Cube Designer is outside the scope of this course.

IBM Cognos dynamically generates the following elements at run-time:

- dimensions: contain members, which may be structured into hierarchies and levels
- hierarchies: provide context to the level structures they contain
- levels: provide structure for the members of a hierarchy
- members: data entities that provide context to cell values
- attributes: provide additional information for members
- cells: are intersection points containing values (measures) for various members from different dimensions (also referred to as tuples)

What Do Report Authors See?



Just as with OLAP data sources, report authors are presented with multi-dimensional metadata in the studios when you apply dimensional information to your model. They also see members in member aware studios (Analysis Studio and Report Studio).

With relational models, report authors cannot see the underlying data while they are creating a report unless they create a filter (i.e. Product Type = Tents). With a DMR model, they can drag the data item (i.e. Tents) into the report to see the data.

Decide on a Modeling Style

- Use standard relational modeling to:
 - enable basic relational ad hoc querying and reporting
- Use DMR metadata when you want to:
 - enable analysis on relational data within Analysis Studio
 - enable drill up and down functionality in reports and ad hoc queries
 - use member-specific functions in an authoring tool

© 2015 IBM Corporation



Cognos Workspace Advanced, Report Studio, Query Studio, and Event Studio can access all types of packages (Relational, OLAP, or DMR). However, while Cognos Workspace Advanced, Report Studio, and Event Studio are member aware (allow you to work directly with members), Query Studio is not.

Analysis Studio deals only with dimensional packages. If the package is not dimensional, then it will not be available when opening this studio.

Examine Regular Dimensions

- Consists of one or more user-defined hierarchies
- Each hierarchy consists of
 - levels
 - keys
 - captions
 - attributes

The screenshot shows the 'Hierarchies' panel in the IBM Business Analytics software. The 'Products' hierarchy is selected. Below the hierarchy tree, there is a table with the following data:

Name	Role	Source
Product Name	_memberCaption	Products.Product Name
Product Number	_businessKey	Products.Codes.Product Number
Product Description	_memberDescription	Products.Product Description
Product Image		Products.Product Image
Introduction Date		Products.Introduction Date

At the bottom left, there is a copyright notice: © 2015 IBM Corporation. On the right side, there is a decorative graphic of interconnected hexagons.

Hierarchies consist of levels, keys, captions, and attributes. Level information is used to roll up measures accurately when performing queries or analyses. Regular dimensions require that each level have key and captions specified, and that captions be of the type string. These items are used to generate members in the studio data trees (where applicable) and retrieve the members at run time.

To indicate that the keys in the levels above the current level are not necessary to identify the members in a level, select the Unique Level check box.

Determinants and Regular Dimensions

Determinants:

- specified for query subjects
- required for dimensions with granularity levels with repeating keys
- required for blobs
- do not provide OLAP functionality

Regular Dimensions:

- include hierarchies, which allow OLAP-style analysis against relational data
- define levels for aggregation rollup
- required for Analysis Studio

© 2015 IBM Corporation

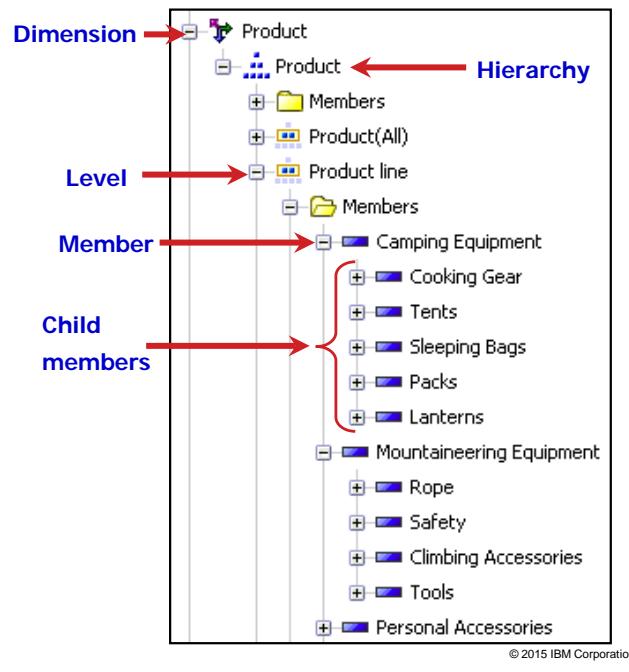


Determinants enable IBM Cognos to aggregate facts correctly by preventing double-counting, but do not provide OLAP functionality or the ability to use the metadata in Analysis Studio. When you create a regular dimension, you specify hierarchy levels, but do not define granularity. Therefore, Framework Manager requires both the dimensional information and determinants to generate the proper SQL.

If you convert a data source query subject to a regular dimension, Framework Manager will use joins and defined levels to properly interpret granularity. However, because the original data source query subject is replaced by a dimensional object, you should only use this feature if the underlying data structure is a perfect star schema and the final model is only intended for OLAP-style analysis.

Note: when you use the Merge in New Regular Dimension feature on a query subject with determinants, Framework Manager creates a regular dimension whose hierarchy is based on the determinants, and the original query subject is retained.

Working with Members



- members are located in levels of an OLAP or DMR structure

Members used as data items for a report

Cooking Gear	Rope	Safety
<Cooking Gear>	<Rope>	<Safety>
<Cooking Gear>	<Rope>	<Safety>
<Cooking Gear>	<Rope>	<Safety>



All studios let authors create reports using levels, which return all members of that level. If the studio in which you create reports is "member-aware", members can be used independently as data items. The metadata items (member attributes) from the multidimensional model can also be used for report creation.

Each member must have a query item that is assigned the role of member key (`_businessKey`) and a query item that is assigned the role of caption (`_memberCaption`). The member key is used to identify a particular member in a multidimensional structure and can be used as a value in drill though and master-detail operations. The member caption is the name that is displayed for the member. Members may also have attributes such as alternate member names or other descriptive information.

Demo 1: Create Regular Dimensions

Purpose:

Business Analysts require OLAP-style queries that allow them to analyze Sales and Sales Target measures against Products and Time. To support this requirement, you will create a dimensional view and use DMR techniques to create the required regular dimensions.

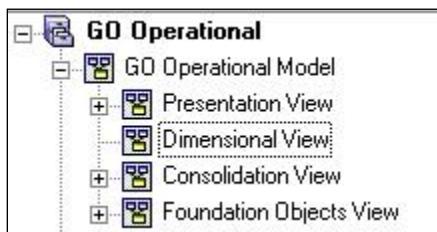
Component: **Framework Manager**
 Project: **GO Operational**

Task 1. Create the top level of a regular dimension for products.

You will organize dimensional objects in the Dimensional View namespace. Once the regular and measure dimensions are complete, you can use star schema groupings to populate the Presentation View.

1. In **Framework Manager**, open the **GO Operational** project located at **C:\Edcognos\B5A52\CBIFM-Start Files\Module 15\GO Operational**. If necessary, log in as User ID **admin**, and Password **Education1**.
2. In **GO Operational Model**, create a namespace called **Dimensional View**.
3. Drag the new namespace above the **Consolidation View** namespace.

The results appear as follows:



4. Right-click **Dimensional View**, point to **Create**, and then click **Regular Dimension**.
5. In the **Available items** pane, expand **Consolidation View > Products**.
6. Drag **Product Line** into the **Hierarchies** pane, right-click the top **Product Line** hierarchy, and then click **Rename**.
7. Type **Products**, and then press **Enter**.
8. Rename **Product Line(All)** to **Product (All)**.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

9. In the **Hierarchies** pane, click the **Product Line** level.
10. In the **Available items** pane, expand the **Codes** query item folder, and then drag **Product Line Code** to the bottom-right pane ("Select a level...") below **Product Line**.
You are prompted to select a role.
11. Select **_businessKey**.

Task 2. Create remaining levels for the Products regular dimension.

1. In the **Available items** pane, under **Consolidation View > Products**, drag **Product Type** to the **Hierarchies** pane, below **Product Line**.
2. Drag **Product Type Code** to the bottom right pane, select **_businessKey** as the role, and then click the **Unique Level** check box above.
Selecting Unique Level indicates that this level (Product Type Code) is unique, and does not rely on the parent object (Product Line Code) for uniqueness.
While it is also true that every Product Line Code is unique, you do not need to set the Unique Level check box for the Product Line level because there is no parent level above it.
3. In **Available items**, drag:
 - **Product Name** to the **Hierarchies** pane (below **Product Type**), and rename it **Product**.
 - **Codes > Product Number** to the bottom-right panel, and set it as the **_businessKey**.
 - **Product Description** to the bottom-right pane and set it as **_memberDescription**.
4. Shift-click **Product Image** and **Discontinued Date**, and drag the three selected items to the bottom-right pane, with **No Role**.
The Product level is the lowest level of the hierarchy and is also represented by a unique key (Product Number). You will specify this level as unique as well.

5. Select the **Unique Level** check box.

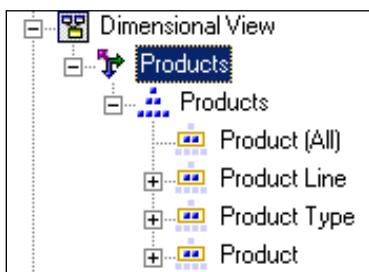
The results appear as follows:

The screenshot shows the Hierarchy Editor interface. At the top, there is a tree view labeled "Hierarchies:" containing the following levels: Products, Product (All), Product Line, Product Type, and Product. Below this is a toolbar with icons for "Add Hierarchy", "Add Level", "Delete", and "Clear All". A checked checkbox labeled "Unique Level" is present. A note below the toolbar says "Select a level in the hierarchy control to see the query items." Below this, a table displays query items with columns for Name, Role, and Source. The table contains the following data:

Name	Role	Source
Product Name	_memberCaption	Products.Product
Product Number	_businessKey	Products.Codes.Pi...
Product Description	_memberDescription	Products.Product
Product Image		Products.Product
Introduction Date		Products.Introduct...
Discontinued Date		Products.Discontin...

6. In the bottom-right pane, rename **Product Name** to **Product Caption**.
 7. Click **OK**.
 8. In the **Project Viewer**, rename the new dimension to **Products**, and expand the dimension and hierarchy.

The results appear as follows:



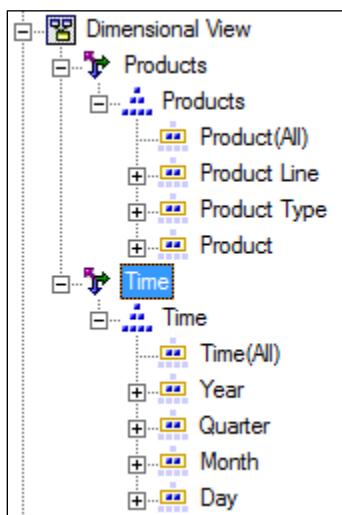
The Dimensional View contains the Products dimension, which contains the Products hierarchies and four levels.

Task 3. Create a regular dimension from the time dimension.

1. Create a **Time** regular dimension as follows:

- In **Dimensional View**, create a new **Regular Dimension**.
- From **Consolidation View > Time**, drag **Year**, **Quarter Key**, **Month Key**, and **Day Key** into the **Hierarchies** pane.
- Name the levels as follows: **Time**, **Time(All)**, **Year**, **Quarter**, **Month**, **Day**.
- Individually select the **Quarter**, **Month**, and **Day** hierarchies, and select the **Unique Level** check box.
- Ensure that for the **Year**, **Quarter**, **Month**, and **Day** hierarchies, each level's **Role** is set to **_businessKey**
- Select the **Month** hierarchy, and drag **Month (numeric)** to the bottom-right pane, selecting **No Role**.
- Click **OK**, and then rename the new dimension **Time**.

The results appear as follows:



You are creating the Time dimension manually, so that all hierarchies come from the Consolidation View. However, you could also create it from the TIME_DIMENSION data query subject by right-clicking it, and selecting Merge Into Regular Dimension. This would allow you to take advantage of the determinants specified in the query subject, which are used to automatically populate the hierarchy information.

2. Right-click the **Time** dimension, click **Verify Selected Objects**, and then click **Verify Model**.

There are four error messages and one warning message. The error messages indicate that four of the levels do not have captions specified. To attempt to resolve one of these errors, you will set the Year level as both the business key and the caption. You can safely ignore the warning message, which indicates that you have not yet validated the new Time dimension.

3. Click **Close**, and then double-click **Time** to re-open the **Dimension Definition** dialog box.
4. In the **Hierarchies** pane, click **Year**, and then, in the bottom-right pane, click the **ellipsis (...)** in the **Role** column.
5. Select **_memberCaption**, click **Close**, and then click **OK**.
6. Right-click **Time**, click **Verify Selected Objects**, and then **Verify Model**.

There are still four errors. The error related to the Year level, however, has changed. It indicates that you cannot assign the **_memberCaption** role to a data type that is not "string". You will fix this in the next task.

7. Click **Close**, and then **Save** the project.

Task 4. Create string calculations for member captions.

1. In **Dimensional View**, double-click **Time** to re-open the **Dimension Definition** dialog box.
 2. In the **Hierarchies** pane, click **Year**, and in the **Role** dialog box, clear the **_memberCaption** check box, and click **Close**.
- Note: **_businessKey** should still be selected.
3. With the **Year** hierarchy selected, click **Add** in the bottom-right corner.
 4. In the **Name** box, type **Year Caption**, and then, in the **Expression definition** pane, type **cast(**.
 5. In **Available Components**, expand **Consolidation View > Time**.
 6. Double-click **Year** to add it to the **Expression definition** pane, and then type the following: **, VARCHAR(4))**.

The results appear as follows:

cast([Consolidation View].[Time].[Year] , VARCHAR(4))

7. Click the **Test Sample** button to verify the results, click **OK**, and then set the **Role for Year Caption** to **_memberCaption**.
8. Repeat steps **3** to **7** to create member-caption levels for the Quarter, Month, and Day hierarchies, based on the following:

Query Item	Expression
Quarter Caption	<code>cast([Consolidation View].[Time].[Year] ,VARCHAR(4)) ' Q' cast([Consolidation View].[Time].[Quarter], VARCHAR (2))</code>
Month Caption	<code>[Consolidation View].[Time].[Month]</code>
Day Caption	<code>cast([Consolidation View].[Time].[Date], VARCHAR (10))</code>

Tip: remember to select the appropriate hierarchy (i.e. Quarter) before you create the query item for its member caption.

Instead of creating a calculation for Month Caption, you could have directly added the Month query item back in Step 1, and then renamed it to Month Caption. Either method is fine.

9. Click **OK**.
10. Right-click the **Time** dimension, and click **Verify Selected Objects**, and then click **Verify Model**.
There are no errors. You can safely ignore the warning message, which is generated because you are publishing a query item that references another query item in a different namespace.
11. Click **Close**, and **Save** the project.

Results:

You created a Dimensional View, and then added regular dimensions that have the hierarchies and levels that are required to support the OLAP-style queries that your business analysts require.

Workshop 1: Create a Regular Dimension

Business Analysts need to analyze staff, as well as products and time, against sales and sales target measures In order to support this requirement; you will add another dimension to the new Dimensional View.

To accomplish this, you will:

- In **Dimensional View**, create a Regular Dimension.
- Based on the objects in **Consolidation View > Staff by Location**, create a dimension called **Staff by Location** with the following levels and attributes:
 - **Staff by Location**
 - **Staff by Location (All)**
 - **Staff Region**
 - **Staff Region Code** (business key), **Staff Region** (member caption, renamed to Staff Region Caption)
 - **Staff Country** (unique level)
 - **Staff Country Code** (business key), **Staff Country** (member caption, renamed to Staff Country Caption)
 - **Staff City** (unique level)
 - **Staff Branch Code** (business key), **Staff City** (member caption, renamed to Staff City Caption), **Staff Address 1** (no role), **Staff Address 2** (no role), **Staff Prov/State** (no role), **Staff Postal Zone** (no role)
 - **Staff Name** (unique level)
 - **Sales Staff Code** (business key), **Staff Full Name** (member caption, renamed to Staff Name Caption), **First Name** (no role), **Last Name** (no role), **Work Phone** (no role), **Extension** (no role), **Fax** (no role), **Email** (no role), **Manager** (no role), **Position** (no role)

For more information about where to work and the workshop results, refer to the Tasks and Results section that follows. If you need more information to complete a task, refer to earlier demos for detailed steps.

Workshop 1: Tasks and Results

Task 1. Create a regular dimension named Staff by Location.

- In **Dimensional View**, create a **Regular Dimension**.
- In **Available items**, expand **Consolidation View > Staff by Location**, and drag the following items to the **Hierarchies** pane
 - **Staff Region**
 - **Staff Country**
 - **Staff City**
 - **Staff Full Name**
- Rename the hierarchy to **Staff by Location**, and rename the top level **Staff by Location (All)**.
- Rename the **Staff Full Name** level to **Staff Name**.
- Set the **Staff Country**, **Staff City**, and **Staff Name** levels as unique.

The results appear as follows:

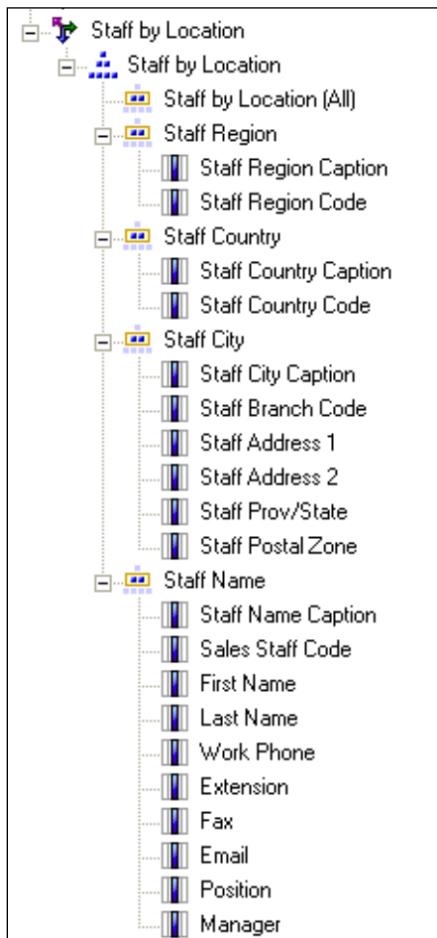
Hierarchies:
Staff by Location
Staff by Location (All)
Staff Region
Staff Country
Staff City
Staff Name

Task 2. Define the query items and attributes for each level.

- **Staff Region** level:
 - ensure **Staff Region** is the member caption, and then rename it to **Staff Region Caption**
 - add **Staff Region Code**, as the business key
- **Staff Country** level:
 - ensure **Staff Country** is the member caption, and then rename it to **Staff Country Caption**
 - add **Staff Country Code** as the business key

- **Staff City** level:
 - ensure **Staff City** is the member caption, and then rename it to **Staff City Caption**
 - add **Staff Branch Code** as the business key
 - add **Staff Address 1**, **Staff Address 2**, **Staff Prov/State**, **Staff Postal Zone**, (no role for each)
- **Staff Name** level:
 - ensure **Staff Full Name** is the member caption, and then rename it to **Staff Name Caption**
 - add **Sales Staff Code** as the business key
 - add **First Name**, **Last Name**, **Work Phone**, **Extension**, **Fax**, **Email**, **Position**, **Manager** (no role for each)
- Click **OK**, rename the new dimension to **Staff by Location**.

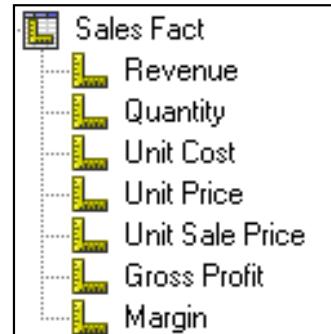
The results appear as follows:



This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Defining Measure Dimensions

- a logical collection of facts that enables OLAP-style analytical querying
- related to regular dimensions within scope
- can be created from:
 - a single table in a database
 - multiple tables across multiple databases



© 2015 IBM Corporation

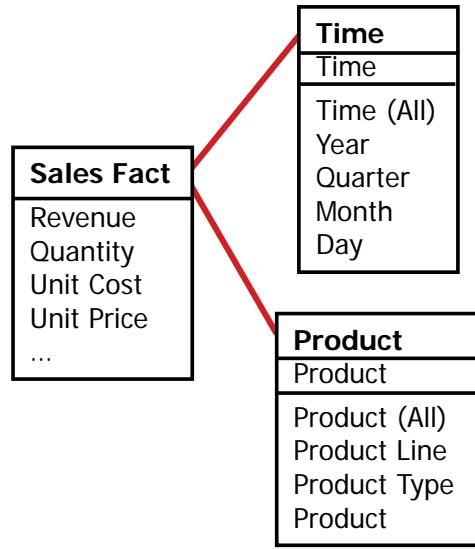


Measure Dimensions are related to Regular Dimensions through scope relationships that define at what levels a measure is in scope. However, underlying join relationships are required to generate the SQL that is sent to the data source.

With respect to joins, even with Scope relationships in place, physical relationships between query subjects will always be required and cardinality is still used to determine if a query subject is a fact or a dimension in context to the query.

Defining Scope Relationships

- scope relationships exist between measure dimensions and regular dimensions
- they define the dimensions, hierarchies and levels that are in scope for measures
- you can create, modify or delete scope relationships



© 2015 IBM Corporation

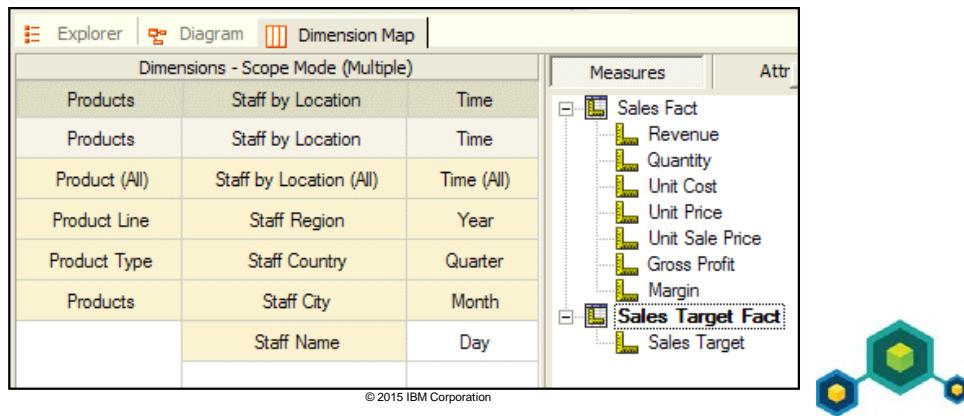
A scope relationship is automatically created between a regular dimension and a measure dimension whose underlying query subjects have a valid JOIN relationship defined. Scope relationships are required between measures and their related dimensions to achieve predictable rollups. Scope relationships define which regular dimensions are included by default in star schema groupings.

Scope relationships are not the same as join relationships. They do not impact the WHERE clause of the generated SQL, but rather which levels in a dimension are available for reporting for a particular measure.

Shortcuts cannot be created for scope relationships. Nor can scope relationships be created for shortcuts. When shortcuts to dimensions are used, the scope will be derived from the scope of the original objects.

Editing DMR Metadata

- use the Dimension Map to create and modify:
 - regular or measure dimensions
 - hierarchies or levels
 - scope relationships



The Dimension Map view displays all regular and measure dimensions contained in a namespace and can be directly modified in this view. This slide example shows how the Sales Target Fact measure dimension is in scope for all levels of the Staff by Location dimension and for all but the lowest level of the Products dimension.

If a measure is not in scope for a particular level of a regular dimension, you will see blank values in Analysis Studio, or repeating values (based on the values found at the parent level) in Query Studio or Report Viewer, but the values are not double counted.

You can edit scope relationships for either measure dimensions or individual measures within a measure dimension (in cases where the fact table has multiple levels of granularity).

Demo 2: Create Measure Dimensions, Set Scope, and Create a Presentation View

Purpose:

To continue the development of the DMR view required by authors and business analysts, you will create two measure dimensions, one for Sales Facts, and one for Sales Target Facts. You will specify the scope of the measures, and then populate a new Presentation view before testing the DMR portion of the model.

Components: Framework Manager, Analysis Studio

Project: GO Operational

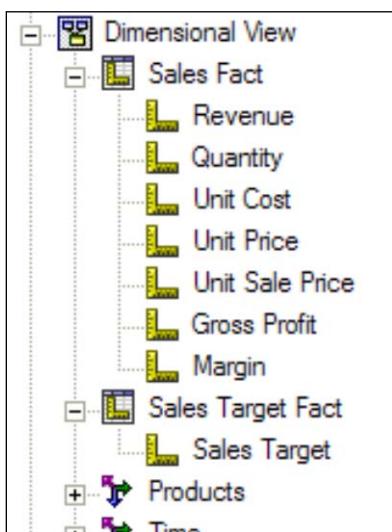
Package: GO Operational (analysis)

Task 1. Create Sales Fact and Sales Target Fact measure dimensions.

1. Right-click **Dimensional View**, point to **Create**, and then click **Measure Dimension**.
2. Expand **Consolidation View > Sales Fact**.
3. Click **Revenue**, Shift+click **Margin**, and drag the selected measures to the **Measures** pane, and then click **OK**.
4. Rename the new dimension **Sales Fact**, and then move it above **Products**.
5. In **Dimensional View**, create a new measure dimension, expand **Consolidation View > Sales Target Fact**, and drag **Sales Target** to the measures pane.
6. Click **OK**, rename the new dimension **Sales Target Fact**, and then move it below the **Sales Fact** measure dimension.

7. Expand the **Sales Fact** and **Sales Target Fact** measure dimensions.

The results appear as follows:



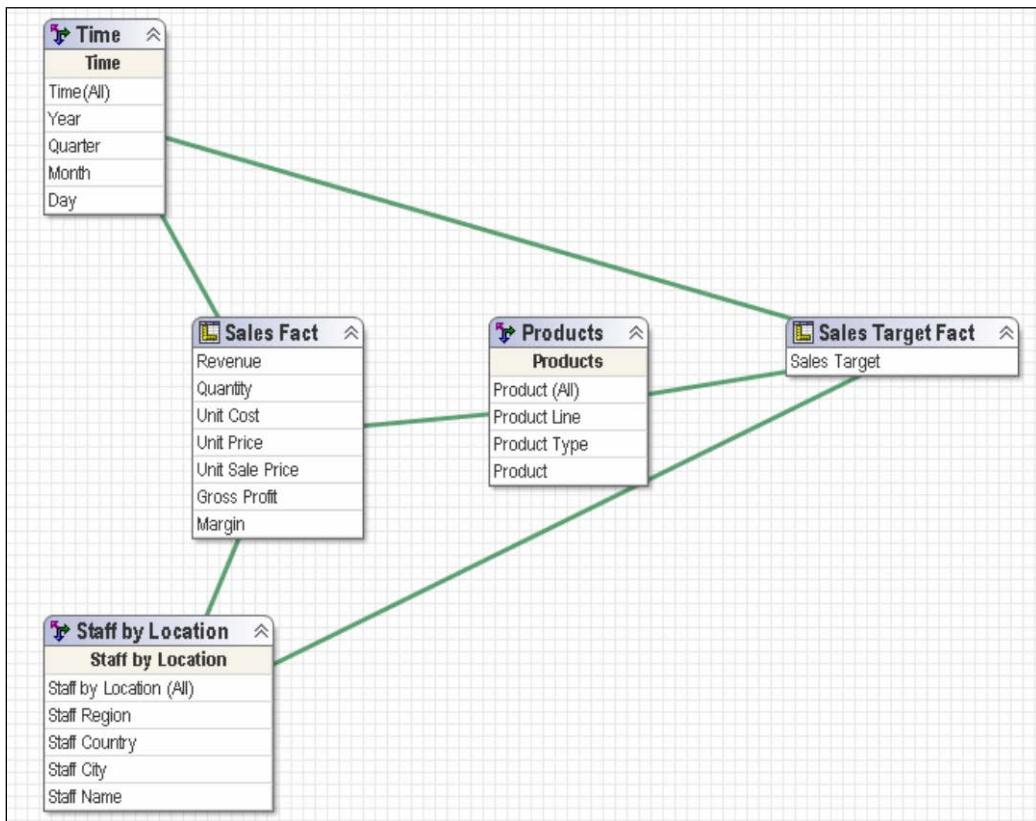
8. **Save** the project.

Task 2. Set scope relationships for the measure dimensions.

1. Double-click the **Dimensional View** namespace to give it focus in the middle pane, and then click the **Diagram** tab
2. From the **Diagram** menu, click **Diagram settings**, select the **Scope Relationships** check box, and then click **OK**.
3. Click **Auto Layout** , and then, beside **Layout Style**, select **Star**.

4. Set the **Sibling Distance** to 30, click **Apply**, and then click **Close**.

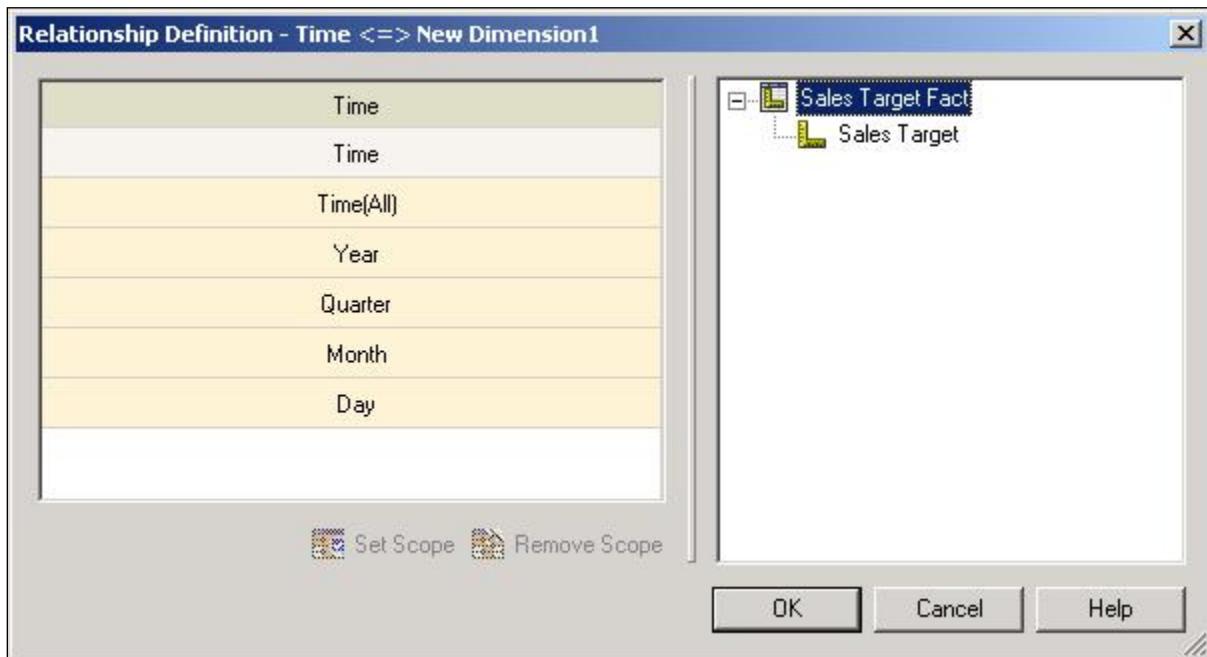
The results appear as follows:



The facts are joined to their related dimensions through scope relationships. These relationships were automatically generated based on the underlying relationships in the Foundation Objects View.

5. Double-click the scope relationship between **Time** and **Sales Target Fact** and then, in the right pane, click **Sales Target Fact**.

The results appear as follows:



All levels in the Time dimension are highlighted, which indicates that they are in scope. This is not the case for sales targets since they are at the month level. You can set the scope in this dialog, or in the Dimension Map pane. You will quickly set scope for this measure here, but then cancel the changes and then set scope in the Dimension Map in order to learn both methods.

6. Click the **Month** level, and then click **Set Scope**.

The results appear as follows:

The Day level is no longer highlighted and is now out of scope for the Sales Target measure.

7. Click **Cancel**, and then in the middle pane, click the **Dimension Map** tab.

The results appear as follows:

Dimensions - Scope Mode (no selection)		
Products	Time	Staff by Location
Products	Time	Staff by Location
Product (All)	Time (All)	Staff by Location (All)
Product Line	Year	Staff Region
Product Type	Quarter	Staff Country
Product	Month	Staff City
	Day	Staff Name

The three dimensions appear in the left pane and the measure dimensions appear in the right pane. Here you can create, edit, and delete dimensions and set scope.

8. In the **Measures** pane, click **Sales Fact**.

The results appear as follows:

Dimensions - Scope Mode (multiple)		
Products	Time	Staff by Location
Products	Time	Staff by Location
Product (All)	Time (All)	Staff by Location (All)
Product Line	Year	Staff Region
Product Type	Quarter	Staff Country
Product	Month	Staff City
	Day	Staff Name

Measures Attributes
Sales Fact
Sales Target Fact

All regular dimensions are highlighted, indicating that all measures in Sales are in scope.

9. Click **Sales Target Fact**.

Again, the measure is in scope for all dimensions. You will set the scope for Sales Target Fact to be at the Month level for Time and Product Type level for Products.

10. In the **Time** dimension, click **Month**, and then on the toolbar, click



11. In the **Products** dimension, click **Product Type**, and then click **Set Scope**.

The results appear as follows:

Dimensions - Scope Mode (Multiple)		
Products	Time	Staff by Location
Products	Time	Staff By Location
Product (All)	Time (All)	Staff By Location (All)
Product Line	Year	Staff Region
Product Type	Quarter	Staff Country
Product	Month	Staff City
	Day	Staff Name

Measures Attributes
Sales Fact
Sales Target Fact

The Day level for the Time dimension and the Products level for the Products dimension are no longer highlighted and are out of scope for Sales Target Fact. You will now explore the Attributes tab.

12. Click the **Attributes** tab.

The results appear as follows:

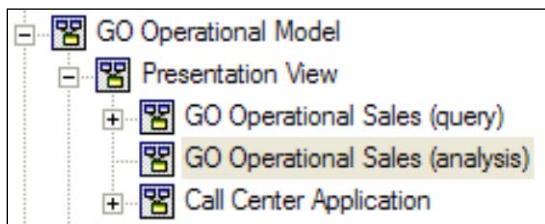
Measures		Attributes
Name	Role	
Product Type	_memberCaption	...
Product Type Code	_businessKey	...

Here you see the business key, member caption and any other attributes for the selected level in the dimension map. You can edit these items here if you wish. Each time you select a different level in any of the dimensions, their attributes will be displayed here.

Task 3. Create a DMR Presentation view using star schema groupings.

1. In the **Presentation View**, create a namespace called **GO Operational Sales (analysis)**, and then drag below **GO Operational Sales (query)**.

The results appear as follows:



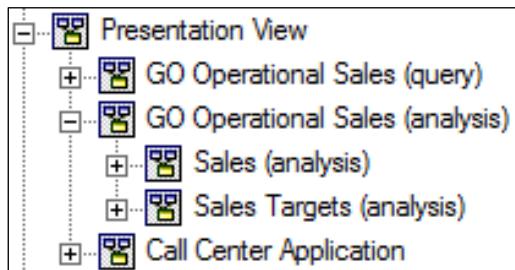
2. In the **Dimensional View**, right-click **Sales Fact**, and then click **Create Star Schema Grouping**.

Unlike when you create star-schema groupings for relational metadata in the Consolidation View, here you do not need to select all the desired dimensions along with the fact. This is because Framework Manager uses the scope relationships to identify related dimensions. The Consolidation View objects have no relationships; therefore you needed to select all required objects. If there had been relationships, the Star Schema Grouping wizard would use them to detect related objects for the grouping.

3. In **Namespace name**, type **Sales (analysis)**, and then click **OK**.

4. Drag **Sales (analysis)** into the **GO Operational Sales (analysis)** namespace.
5. Repeat steps **2** to **3** with **Sales Target Fact** to create **Sales Targets (analysis)**, and then drag **Sales Targets (analysis)** into the **GO Operational Sales (analysis)** namespace.

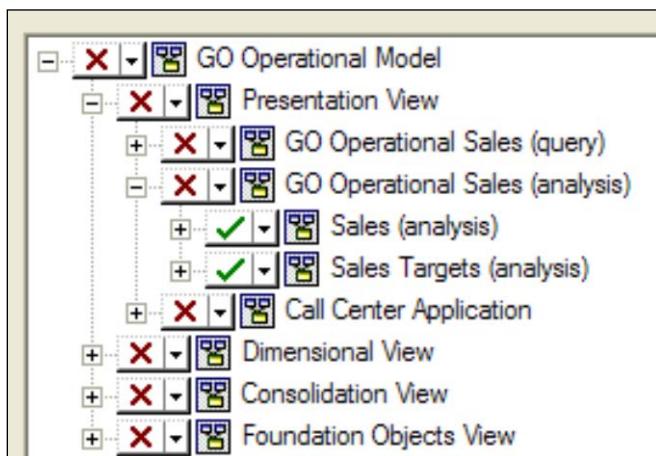
The results appear as follows:



Task 4. Create a GO Operational (analysis) package.

1. Right-click **Packages**, point to **Create**, and then click **Package**.
2. In the **Name** box, type **GO Operational (analysis)**, and then click **Next**.
3. Clear the **GO Operational Model** check box, and then expand **Presentation View > GO Operational Sales (analysis)**.
4. Select **Sales (analysis)** and **Sales Targets (analysis)**.

The results appear as follows:



5. Click **Finish**, and click **Yes** to open the Publish Package wizard.
 6. Clear the **Enable model versioning** check box, click **Next** twice, click **Publish**, and then click **Finish**.
- The Verify Model dialog box appears listing informational messages.
7. Click **Close**, and then **Save** the project.

Task 5. Test the new package in Analysis Studio.

1. In **IBM Cognos Connection**, log on as **admin** (Password: **Education1**), launch **Analysis Studio**, and then select the **GO Operational (analysis)** package.
2. Click **OK** to create a **Blank Analysis** report.
3. In the **Insertable Objects** pane, expand **Sales (analysis)**, and then drag **Products** to the **Columns** drop zone in the analysis work area.
4. Drag **Time** to the **Rows** drop zone in the analysis work area.
5. Expand the **Sales Fact** measure dimension, and then drag **Revenue** to the **Measure** drop zone in the analysis work area.

A section of the results appear as follows:

Revenue	Camping Equipment	Mountaineering Equipment	Personal Accesso
2010	\$332,986,338.06		\$3
2011	\$402,757,573.17	\$107,099,659.94	\$4
2012	\$500,382,422.83	\$161,039,823.26	\$5
2013	\$352,910,329.97	\$141,520,649.70	\$4
Time(All)	\$1,589,036,664.03	\$409,660,132.90	\$1,88

6. Click the intersection of **Camping Equipment** and **2013** once to give **\$352,910,329.97** focus, and then click it again to drill down on both the row and the column at the same time.
7. Drag **Staff by Location** to just below the **Lanterns** heading.

Tip: when you hover over the correct area, a black line appears between the headers and first row of data.

A section of the results appear as follows:

Revenue	Lanterns				
	Northern Europe	Southern Europe	Central Europe	Asia Pacific	Americas
2013 Q3	\$423,481.10	\$476,173.75	\$927,158.38	\$1,148,554.36	\$1,218,981
2013 Q1	\$1,194,758.23	\$1,344,938.05	\$2,612,282.59	\$3,241,950.93	\$3,436,174
2013 Q2	\$1,212,045.67	\$1,362,717.92	\$2,649,479.61	\$3,304,103.35	\$3,481,675
2013	\$2,830,285.00	\$3,183,829.72	\$6,188,920.58	\$7,694,608.64	\$8,136,831

You can now further analyze sales in relation to location.

- Under **Lanterns**, click the cell where **Central Europe** and **2013 Q2** intersect twice.

The results appear as follows:

Revenue	Lanterns				
	Switzerland	Belgium	Germany	France	United Kingdom
May	\$103,788.14	\$108,971.53	\$199,138.07	\$233,202.91	\$230,380.16
April	\$106,732.05	\$104,301.67	\$225,212.78	\$249,805.42	\$232,948.94
June	\$111,810.79	\$106,272.62	\$214,289.23	\$199,949.19	\$222,676.11
2013 Q2	\$322,330.98	\$319,545.82	\$638,640.08	\$682,957.52	\$686,005.21

- In the left column, click **June**.
- In the **Insertable Objects** pane, expand **Sales Targets (analysis) > Sales Target Fact**.
- Drag **Sales Target** to the measures section of the analysis to replace the existing measure.

The results appear as follows:

Sales Target	Cooking Gear			
	France	Germany	Switzerland	United Kingdom
2013-06-01				
2013-06-02				
2013-06-03				
2013-06-04				
2013-06-05				

The measure values are blank because the Sales Target measure is not in scope at the Day level.

- In the bottom-left corner, click **June** to drill up.
Values appear because you are at the Month level, which is in scope.
- Leave **Analysis Studio** open for the next demo.

Results:

You have successfully created and tested a dimensionally modeled relational package by adding measure dimensions and specifying scope.

Sorting Compatible Query Mode (CQM) Projects

- define the order that DMR members are sorted in:
 - metadata trees (i.e. Report Studio)
 - report results
- sort for OLAP compatibility when using member-relative functions, such as:
`parallelPeriod([Sales (analysis)].[Time].[Time].[Year],1,[2007-01-09])`
- Dynamic Query Mode (DQM) projects are automatically sorted

© 2015 IBM Corporation



In a Compatible Query Mode (CQM) project, you can define the sort order of DMR members to control how they appear in the metadata tree (Report Studio), and the report results.

Member order is vital when using member-relative functions. In these cases, you can also choose to sort the members for OLAP compatibility to ensure the members are always returned in the same order based on your sort criteria. For example, in the `parallelPeriod` example above, which compares sales for a particular date with sales on the same day in the previous year, date values must be in chronological order.

You do not need to specify sorting options for Dynamic Query Mode (DQM) projects, as the members are automatically sorted in ascending order by the member caption. If there are duplicate captions, then those are sorted by business key.

Each data source may use a different naming convention for the items used to describe the member key and member caption. The end result is the same, however. A unique identifier is used for the member key and a user friendly name can be used for the member caption.

Demo 3: Metadata Tree and Member Sorting

Purpose:

You want to apply Framework Manager sorting options in order to ensure more intuitive and predictable report authoring environments. You will use these options to ensure that members in metadata trees and reports are sorted in alphabetical order. You will also configure sorting to be compatible with OLAP style queries.

Component: Framework Manager, Analysis Studio, Report Studio

Package: GO Operational (analysis)

Task 1. Examine member sorting in Analysis Studio.

1. In Analysis Studio, in the **GO Operational (analysis)** package, expand **Sales (analysis) > Staff by Location > Central Europe**.

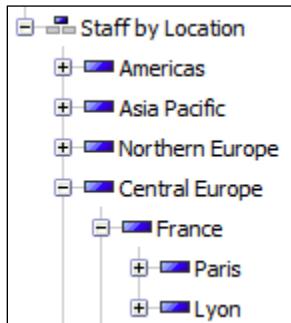
The results appear as follows:



Notice that the members under Central Europe are not sorted alphabetically.

2. Expand **France**.

The values below France are also not sorted alphabetically.



3. Examine the data in the analysis.

Sales Target	Lanterns					
	Belgium	Switzerland	France	Germany	United Kingdom	Central Europe
May	92,400	97,000	201,400	188,000	207,600	786,400
June	95,900	104,000	187,300	199,300	205,400	791,900
April	163,300	158,000	382,700	325,500	263,900	1,293,400
2013 Q2	351,600	359,000	771,400	712,800	676,900	2,871,700

The members for Central Europe are also not sorted alphabetically (from left to right) in the report. You can change this default behavior in Framework Manager by configuring member sorting for the regular dimension.

4. Save the analysis as **Member Sorting Test**, and then close **Analysis Studio**.
The Analysis will be used later in this Demo.

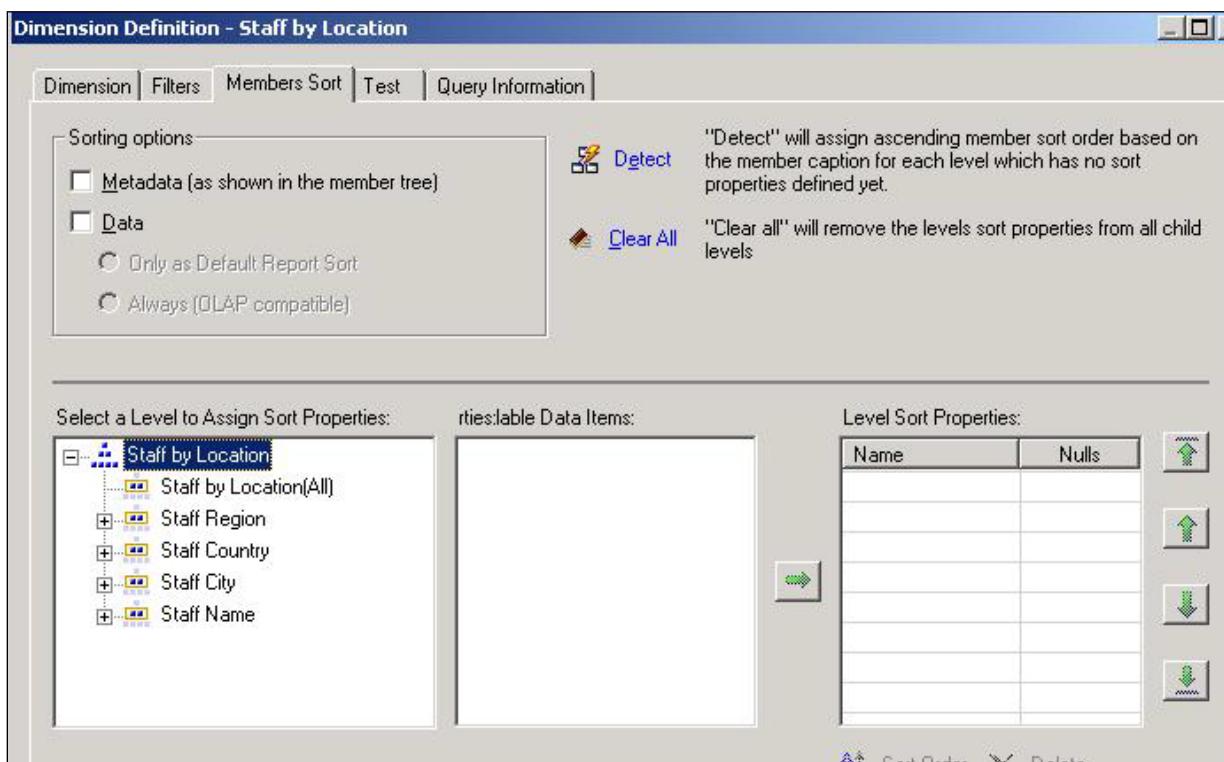
Task 2. Sort members in Framework Manager.

You will resolve two issues using the member sorting feature. One is to ensure that members appear sorted in the data trees in applicable studios and the other is to ensure that the time dimension is compatible with dimensional functions.

1. In **Framework Manager**, in the **GO Operational Model > Dimensional View** namespace, double-click **Staff by Location**.

2. Click the **Members Sort** tab.

The results appear as follows:



The checkboxes control the following sorting behavior:

- **Metadata:** sorts members of the level in the metadata tree.
- **Data - Only as Default Report Sort:** sorts members in the report according to the sort information specified on the levels.
- **Data - Always (OLAP compatible):** provides member-relative functions with a sorted structure of the members that can be navigated with consistency. The members of the level will also be sorted in the metadata tree and reports.

You will now choose to sort the members of this regular dimension both in the data tree and in the reports.

3. Under **Sorting options**, select both the **Metadata** and **Data** check boxes, and then click **Detect**.

A message appears indicate member sort properties were added to four levels.

4. Click **OK**, and then click **Staff Region**.

The results appear as follows:

Select a Level to Assign Sort Properties:		Available Data Items:		Level Sort Properties:					
<ul style="list-style-type: none"> ... Staff by Location <ul style="list-style-type: none"> ... Staff by Location (All) Staff Region ... Staff Country ... Staff City ... Staff Name 		<ul style="list-style-type: none"> Staff Region Caption Staff Region Code 		<table border="1"> <thead> <tr> <th>Name</th> <th>Nulls</th> </tr> </thead> <tbody> <tr> <td>Staff Region Caption</td> <td>Last</td> </tr> </tbody> </table>		Name	Nulls	Staff Region Caption	Last
Name	Nulls								
Staff Region Caption	Last								
				<div style="text-align: right;"> ↑ ↑ ↓ ↓ </div>					

Here you can see the criteria that are used to sort the members for each level. In this case Staff Region Caption (the item used as the member caption) is used to sort the members. All the levels are currently using the member caption as the sort criteria. If you require the sorting to be applied on another item such as a key or some other attribute, you can edit the settings. You can also use multiple items in the sorting criteria. For example, you can sort first by name and then by code. If there are two identical names, then the code will be used to determine which name is displayed first.

You also have the option on where to display null values in your results. This is configured under the Nulls column. Selecting First places the null values at the beginning, and Last places the null values at the bottom. Unspecified uses the setting defined in the data source.

5. Click **OK**.

You will now ensure that the Time dimension members are compatible with dimensional functions.

6. Double-click the **Time** dimension, and then click the **Members Sort** tab.
7. Under **Sorting options**, select both the **Metadata** and **Data** check boxes, and then select **Always (OLAP compatible)**.
8. Click **Detect**, click **OK**, and then click **OK** again.
9. **Save** the project.

Task 3. Re-test member sorting in the studios.

1. Publish the **GO Operational (analysis)** package, ensuring that you deselect the **Verify the package before publishing** option.
2. In **IBM Cognos Connection**, click **GO Operational (analysis)**, and then click **Member Sorting Test**.

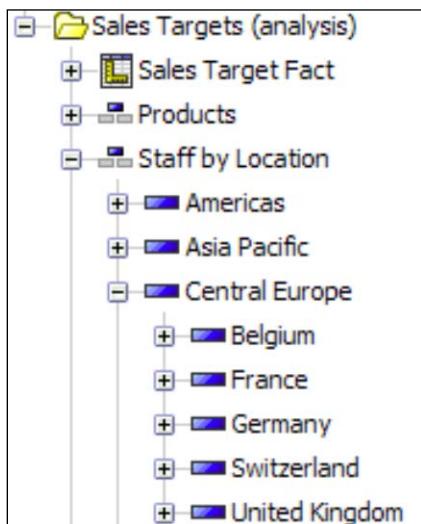
The results appear as follows:

Sales Target	Cooking Gear					Central Europe
	Belgium	France	Germany	Switzerland	United Kingdom	
April	379,200	494,800	467,000	271,300	461,500	
May	187,800	459,000	389,000	213,400	432,800	
June	167,300	380,600	425,900	213,500	424,400	
2013 Q2	734,300	1,334,400	1,281,900	698,200	1,318,700	

The members in the analysis for Central Europe are now alphabetically sorted, from left to right.

3. In **Insertable Objects**, expand **Sales Targets (analysis) > Staff by Location > Central Europe**.

The results appear as follows:



The members are sorted here as well. If you continue to expand the members in the data tree for this dimension, you will see all levels are sorted.

Note: In the Report Studio data tree, sorting is applied to the members of each level. For example, at the Product Type Level, members are sorted alphabetically for Camping Equipment, and then sorted alphabetically for Golf Equipment.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

15-39

4. Close IBM Cognos Analysis Studio.
5. Launch Report Studio, and click **Create New**.
6. With the **GO Operational (analysis)** package selected, click **Crosstab**, and then click **OK**.
7. In the **Source** pane, expand **Sales (analysis)**, and add the following:
 - Columns: **Products > Products > Product Line**
 - Rows: **Time > Time > Day**
 - Measure: **Sales Fact > Revenue**
8. **Run** the report.

The results appear as follows:

Revenue	Camping Equipment	Personal Accessories	Outdoor Protection	Golf Equipment
2010-01-01				
2010-01-02				
2010-01-03				
2010-01-04				
2010-01-05				
2010-01-06				
2010-01-07				
2010-01-08				
2010-01-09				
2010-01-10				
2010-01-11				
2010-01-12	\$20,217,372.98	\$7,414,443.06	\$2,263,380.47	\$9,141,599.8
2010-01-13	\$5,000,710.60	\$3,477,197.59	\$474,025.75	\$2,536,524.6
2010-01-14	\$633,110.20	\$2,118,932.80	\$91,322.21	\$388,795.2

The dimensional function now returns the correct date from the previous year, because you configured this regular dimension to sort members for OLAP compatibility. This may require more processing, but the results are predictable. If performance becomes an issue for large dimension tables using this technique, you may consider sorting at the table level.

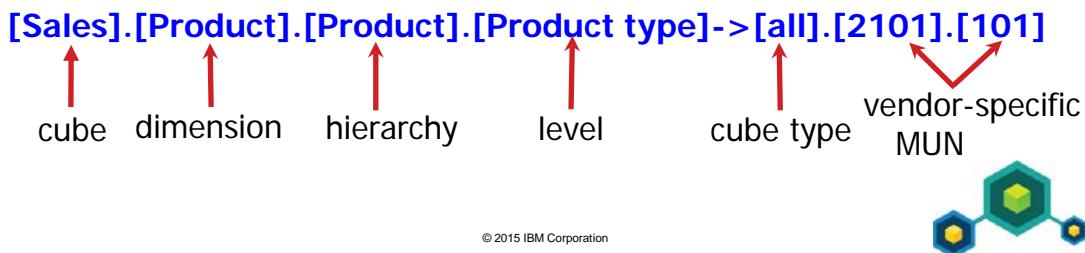
9. Close the report and then close Report Studio.

Results:

By using the member sorting options in Framework Manager, you were able to create a more intuitive and predictable experience in the studios for authors.

What is a Member Unique Name (MUN)?

- ensures that multidimensional members are unique
- referenced in an expression when a member is used:
 - in a report
 - in a filter or calculation
 - for drill-through
- example:



Member Unique Names (MUNs) ensure that members are unique within the multidimensional structure.

When modeling DMR metadata in Framework Manager, you do not manually create the MUN for each member. Rather, you specify the member key and member caption, and the member key is used in the MUN when it is generated at run time.

Changes that Impact a MUN

- MUNs can change when:
 - hierarchies or levels change
 - member keys change
 - category codes in PowerCubes
 - Member Key Column in MSAS cubes
 - _memberKey role in DMR models
 - members no longer exist in the data source
 - a production environment has more members than in the test environment

© 2015 IBM Corporation



You should avoid changing the objects in a Package after it has been published into the production environment. When MUNs change, they impact the reports that directly reference the members to which they point. Those MUNs must then be identified and fixed in the report.

In drill-through scenarios, once a broken MUN reference is fixed, there is potential for the report to pass the wrong parameter to the target report. This can occur when the member key changes. This is why it is not recommended to change member keys. It is critical that business keys are conformed across the business ensuring that they do not change and that there is no need to change them.

Demo 4: Identify How Changes to MUNs Impact Reports

Purpose:

In this demo, you will explore how package changes can impact reports. To do so, you will use a DMR source to create a report that uses members as data items, and then identify a MUN for a member in the report. You will then make a change to the model that impacts the MUN, re-publish the package, and re-run the report. You will identify how the report is impacted, and subsequently how the MUN is impacted. You will then fix the report.

Components: **Framework Manager, Report Studio**

Project: **great_outdoors_warehouse**

Package: **GO Data Warehouse (analysis)**

For this demo you will use the great_outdoors_warehouse model, which is based on a reporting database (star schema) for the Sample Outdoors Company.

Task 1. Publish the GO Data Warehouse (analysis) package.

1. In **Framework Manager**, close any projects that may be open, and then open the **great_outdoors_warehouse** project located at
C:\Edcognos\B5A52\CBIFM-Start Files\Module 15\great_outdoors_warehouse.
2. Publish the **GO Data Warehouse (analysis)** package.
If prompted, enter month and year values (i.e. May, 2010).

Task 2. Create a report using members.

1. In **IBM Cognos Connection**, launch **Report Studio**, and click **Create New**.
2. In **Package**, select **GO Data Warehouse (analysis)**, and click **Crosstab**, and then click **OK**.
3. In the **Source** pane, expand **Sales and Marketing (analysis) > Sales target > Sales target fact**, and then drag the **Sales target** measure to the **Measures** drop zone in the report.
4. Expand **Employee (by position) > Employee (by position-department) > Position-department (level 1) > Members > Executive > Operations > Sales > Level 3 Sales Representative**.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

15-43

5. Click **Aiko Watanabe**, and Shift+click **Alessandra Torta**, and then drag the selected items to the **Rows** drop zone.
6. Drag the **Employee (by position-department)(All)** level, to the **Rows** drop zone under the existing rows.
Tip: When the cursor is in the correct location, a black line will appear.
7. Under **Sales and Marketing (analysis)**, expand **Sales > Time dimension > Time dimension**, and then drag the **Year** level to the **Columns** drop zone.
The results appear as follows:

Sales target	<#Year#>	<#Year#>
<#Set - Employee (by position)#>	<#1234#>	<#1234#>
<#Employee (by position-department)(All)#>	<#1234#>	<#1234#>

8. On the toolbar, click **Run Report** .

The results appear as follows:

Sales target	2010	2011	2012	2013
Aiko Watanabe	12,260,000	9,870,300	12,648,200	4,575,700
Akemi Yamada	5,309,700	8,681,600	12,219,000	8,031,660
Alessandra Torta	7,408,000	7,996,500	8,136,100	7,529,800
Employee (by position-department)(All)	812,885,300	1,036,923,300	1,332,553,100	1,023,006,840

The report contains the values of the member items that you added during design.

9. Close **IBM Cognos Viewer**.
10. In the **Source** pane, right-click the **Aiko Watanabe** member, and then click **Properties**.
Notice the Member Unique Name property:
[Sales target].[Employee (by position)].[Employee (by position-department)].[Employee]->[all].[100].[220].[390].[43639].[4116]
At the very end of the MUN, the _businessKey role value used is 4116. This value is based on the Employee key in the data source.
11. Click **Close**.
12. Save the report in **GO Data Warehouse (analysis)** as **MUN Test**, and then close **Report Studio**.

Task 3. Change the _businessKey role for a level within a dimension.

1. In **Framework Manager**, in the **Project Viewer**, expand **go_data_warehouse > Dimensional view**.
2. Double-click **Employee (by position-department)**.
The Dimension Definition dialog opens.
3. In the **Hierarchies** pane, click the **Employee** level.
In the bottom pane, the **_businessKey** role is set on Employee key. The values represented by this item are those that appear as the member key for the MUN of a member as shown earlier.
4. In the bottom pane, under the **Role** column, click the **ellipsis** beside **Employee code**.
5. Check the **_businessKey** box, and then click **OK** to dismiss the warning message.
6. Click **Close**, and then click **OK**.
7. **Save** the project in **C:\Edcognos\B5A52\Course_Project\Great Outdoors Warehouse**.
8. **Publish** the **GO Data Warehouse (analysis)** package, overwriting the existing package.

Task 4. Examine the impact of a model change on the MUN.

1. In **IBM Cognos Connection**, in **Public Folders**, click the **GO Data Warehouse (analysis)** folder, and then click the **MUN Test** report.

The results appear as follows:

Sales target	2010	2011	2012	2013
Employee (by position-department)(All)	812,885,300	1,036,923,300	1,332,553,100	1,023,006,840

Data appears to be missing. The report does not contain the values of the member items that you added during design. It contains only the values for the metadata item you added (Employee (by position-department)(All)). The report is running against the most recent version of the package.

2. At the top-right corner of the page, click **Open with Report Studio** , click **OK**.

3. In the **Insertable Objects** pane, expand **Sales and Marketing (analysis) > Sales target > Employee (by position) > Employee (by position-department) > Position-department (level 1) > Members > Executive > Operations > Sales > Level 3 Sales Representative.**
4. Right-click **Aiko Wantanabe**, and then click **Properties**.

At the end of the Member Unique Name, the _businessKey role value is now 10572. This value is based on the Employee code in the data source. This MUN is now different based on the change you made to the model. The current employee members in the report layout are associated with MUNs that no longer exist and therefore are not returned in the report.

To correct this you must replace the existing members in the report layout with the current members from the Insertable Objects pane.

5. Click **Close**, delete the members in the report layout, and then add the same members back into the report layout from the **Source** pane.
6. **Run** the report.

The results appear as follows:

Sales target	2010	2011	2012	2013
Aiko Watanabe	12,260,000	9,870,300	12,648,200	4,575,700
Akemi Yamada	5,309,700	8,681,600	12,219,000	8,031,660
Alessandra Torta	7,408,000	7,996,500	8,136,100	7,529,800
Employee (by position-department)(All)	812,885,300	1,036,923,300	1,332,553,100	1,023,006,840

The rows are now returned appropriately.

Note: If the report contains calculations based on members whose MUNs change, you need to delete the members referenced in the calculation before adding the new members to the report.

7. **Close** all browser windows, and then close Framework manager, saving the project if prompted.

Results:

You have examined how a report can be impacted when you make a model change that impacts the MUN for a member in a report. This reinforces the importance of using the appropriate business key from the beginning so that reports are not broken by modifications that are made after you move to a production environment.

Dimensional Modeling Considerations

- Dimensional objects provide an additional layer of metadata that enables OLAP behaviors
- It may be necessary to employ a form of database vendor materialization to improve performance
- Build mandatory filters into your model to ensure that end users do not accidentally retrieve excessively large data sets

© 2015 IBM Corporation



The rules regarding data volumes that apply to building cubes also apply to a DMR source. The key difference is that with filtering strategies, you can perform analysis against larger volumes in a relational source that is practical to do with most OLAP sources.

Vendor Examples for materialization are:

- Oracle - Materialized Views
- SQL Server - Indexed Views
- DB2 - Cube Views

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Summary

- You should be able to:
 - apply dimensional information to relational metadata to enable OLAP-style queries
 - sort members for presentation and predictability
 - define members and member unique names
 - identify changes that impact a MUN

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

15-48

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



Managing OLAP Data Sources

IBM Cognos BI

Business Analytics software

© 2015 IBM Corporation



This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

16-2

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Objectives

- At the end of this module, you should be able to:
 - connect to an OLAP data source (cube) in a Framework Manager project
 - publish an OLAP model
 - publish a model with multiple OLAP data sources
 - publish a model with an OLAP data source and a relational data source

© 2015 IBM Corporation

Business Analytics software

IBM

OLAP Data Sources in IBM Cognos

- IBM Cognos Business Intelligence provides full support for the analysis of PowerCubes and other OLAP data sources.

Revenue	2011	2012	2013	Years
Camping Equipment	402,757,573.17	500,382,422.83	352,910,329.97	1,589,036,664.03
Golf Equipment	168,006,427.07	230,110,270.55	174,740,819.29	726,411,367.89
Mountaineering Equipment	107,099,659.94	161,039,823.26	141,520,649.70	409,660,132.90
Outdoor Protection	25,008,574.08	10,349,175.84	4,471,025.26	75,994,296.25
Personal Accessories	456,323,355.90	594,009,408.42	443,693,449.85	1,885,673,307.78
Products	1,159,195,590.16	1,495,891,100.90	1,117,336,274.07	4,686,775,768.85

© 2015 IBM Corporation



Other OLAP sources include:

- IBM Cognos Planning Analyst Models
- Microsoft Analysis Services
- IBM DB2 OLAP
- Hyperion Essbase

SAP BW is also an OLAP data source, but unlike other cube sources, you can lightly model SAP BW metadata after it is imported.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Using OLAP Data Sources

- use Framework Manager or IBM Cognos Connection to connect to an OLAP data source
- cubes are published directly to the portal, without any modeling required
- use Framework Manager to:
 - create multi-cube packages
 - combine cubes and relational metadata in one package

© 2015 IBM Corporation



You can use Framework Manager or IBM Cognos Connection to create data source connections to various cube data sources. When you import a cube into Framework Manager, no modeling is required since the modeling has already been done in an OLAP modeling tool such as IBM Cognos Transformer. The package is then published directly to the portal, making it available to report authors.

Not only can you publish IBM Cognos PowerCubes from Framework Manager, you can also publish them directly in IBM Cognos Connection or from IBM Cognos Transformer.

Creating reports that link multiple cube sources, or a cube source and a relational source, may require the caption function on the OLAP source(s) to retrieve a string value that can be used to match a string value in the other source. This is required when MUNs don't match or the value extracted from the MUN does not match the value in the relational source. MUNs are discussed in more detail later in this module.

Demo 1: Import and Publish an OLAP Data Source

Purpose:

Authors would like to be able to report from data stored in multiple OLAP sources. To support this, you will create a Framework Manager project and import a PowerCube.

Components: **Framework Manager, Analysis Studio**

Project: **OLAP Model**

Package: **GO Cube**

Task 1. Create a new project and import a cube.

1. In **Framework Manager**, click **Create a new project**, and then name the new project **OLAP Model**.
2. Clear the **Use Dynamic Query Mode** check box.
3. Set the **Location** to **C:\Edcognos\B5A52\Course_Project\OLAP Model**.
4. Click **OK**.
5. In the **Select Languages** dialog box, ensure that **English** is selected, and then click **OK**.
6. Ensure that **Data Sources** is selected, and click **Next**.
7. Click **New**, and then click **Next**.
8. In the **Name** box, type **GO Cube**, and then click **Next**.
9. Under **Type**, select **IBM Cognos PowerCube**, and then click **Next**.

The connectivity information you supply for a cube data source connection will depend on the type of cube source you are accessing. For example, PowerCubes require that you supply the filename and path for the cube.

To specify the location, you will use Windows Explorer to obtain the path and file name since the cube is local. For a network cube, use a UNC path.

10. Open **Windows Explorer**, and navigate to **C:\Program Files (x86)\IBM\cognos\c10\webcontent\samples\datasources\cubes\PowerCubes\EN**.
11. Copy the path from the **Address** box to the **Windows location** box in the **New data source Wizard**, and then type **\sales_and_marketing.mdc**.
12. Scroll down, under **Testing**, click **Test the connection**, and then click **Test**.
A message appears indicating that the test was successful.

13. Click **Close**, and then click **Close** again.
14. Click **Finish**, and then click **Close**.
15. In the list of data sources, select **GO Cube**, click **Next** and then click **Finish**.
You are now prompted to create a package for this project. You will use the name GO Cube for your package.
16. Click **Finish**.
17. Click **No**.

Task 2. Examine objects and publish a package.

1. In the **Project Viewer** pane, expand **Model**.

You can see that the GO Cube model does not include any additional information. Everything needed to publish it is stored internally in the data source. You could have created your connection to the cube and your package entirely in IBM Cognos Connection, but you are doing it here so that you can add additional cubes to the model in the next demo.

2. Under **Packages**, right-click **GO Cube**, and then click **Publish Packages**.
3. Clear the **Enable model versioning** check box, click **Next** twice, click **Publish**, and then click **Finish**.
4. Save your project.

Task 3. Test the GO Cube package.

1. Log into **IBM Cognos Connection**, and then launch **Analysis Studio** selecting the **GO Cube** package.
2. Select **Default Analysis**, and then click **OK**.

The results appear as follows:

Revenue	Camping Equipment	Mountaineering Equipment	Personal Accessories	Outdoor Protection	Golf Equipment
2010	332,986,338.06		391,647,093.61	36,165,521.07	
2011	402,757,573.17	107,099,659.94	456,323,355.90	25,008,574.08	
2012	500,382,422.83	161,039,823.26	594,009,408.42	10,349,175.84	
2013	352,910,329.97	141,520,649.70	443,693,449.85	4,471,025.26	
Time	1,589,036,664.03	409,660,132.90	1,885,673,307.78	75,994,296.25	

You have successfully published an IBM Cognos PowerCube, which can be used to analyze data and write reports in IBM Cognos.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

3. Close **Analysis Studio** without saving the analysis.
4. Leave **IBM Cognos Connection** and **Framework Manager** open for the next demo.

Results:

You created a new Framework Manager project, connected to a PowerCube, and created a data source connection and package. You then published the package and tested it in Analysis Studio.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Demo 2: Import and Publish Multiple OLAP Data Sources

Purpose:

Authors would like to access financial data stored in a Microsoft Analysis Services cube. They would also like to be able to link data from this cube to the original Sample Outdoors Company PowerCube. You will add a second cube to your model and then create a test report.

Components: **Framework Manager, Report Studio**

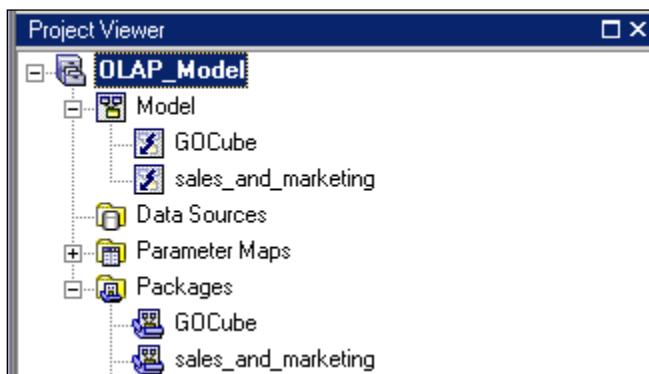
Project: **OLAP Model**

Package: **GO Cube**

Task 1. Add the sales_and_marketing cube to the model.

1. In **Framework Manager**, in the **Project Viewer** pane, right-click **Model**, and then click **Run Metadata Wizard**.
2. Ensure that **Data Sources** is selected, and then click **Next**.
3. Click **sales_and_marketing**, and then click **Next**.
4. Click **Finish** to complete the wizard.
5. Click **Finish** to create the **sales_and_marketing** package.
6. Click **No** when asked if you want to open the Publish Package Wizard.

The results appear as follows:



Task 2. Add the sales_and_marketing package to the GO Cube package and publish.

1. In the **Project Viewer**, under **Packages**, double-click **GO Cube**.
2. Select the **sales_and_marketing** checkbox to select it, and then click **OK**.
3. Publish the **GO Cube** package, overwriting the existing package.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Task 3. Create a report with the GO Cube package.

1. In **IBM Cognos Connection**, launch **Report Studio**, and select the **GO Cube** package.

2. Create a new **List** report.

3. In the **Source** pane, under **GO Cube**, expand **GO Cube** and **sales_and_marketing**.

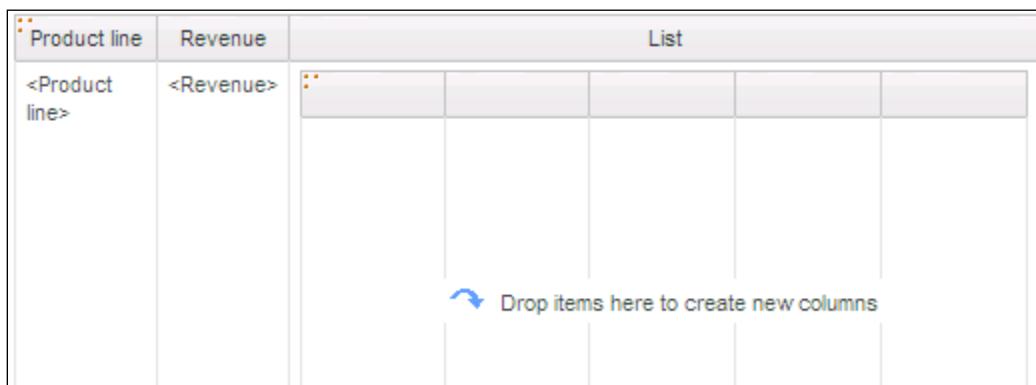
The dimensions for each cube appear. You will now create a report that displays data from both cubes.

4. Expand **GO Cube > Products > Products**, and then drag the **Product line** level to the report.

5. Expand **GO Cube > Measures**, and then double-click **Revenue**.

6. In the **Source** pane, click the **Toolbox**  tab, and then drag a **List** object beside **Revenue**, and click **OK**.

The results appear as follows:



A screenshot of the Report Studio interface showing a 'List' object. The object has two columns: 'Product line' and 'Revenue'. A tooltip 'Drop items here to create new columns' is visible at the bottom of the list area.

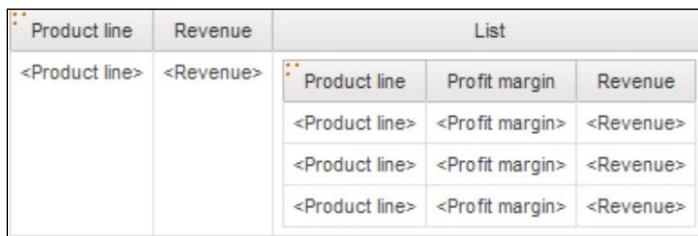
The list is added as a new column.

7. Click the **Source** tab, expand **sales_and_marketing**.

8. Expand **Products > Products**, and drag the **Product line** into the **List** object.

9. Expand **sales_and_marketing > Measures**, and then drag **Profit margin** and **Revenue** beside the **Product line** column under **List**.

The results appear as follows:



A screenshot of the Report Studio interface showing the completed 'List' object. It now includes three columns: 'Product line', 'Profit margin', and 'Revenue'. The 'Revenue' column is aligned to the right. The 'Profit margin' and 'Revenue' columns are aligned to the left.

10. On the toolbar, click **Run Report** .

The results appear similar to the following:

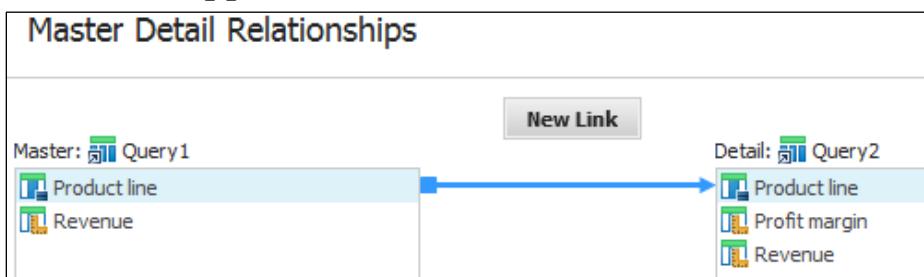
Product line	Revenue	List		
		Product line	Profit margin	Revenue
Camping Equipment	1,589,036,664.03	Camping Equipment	36.9%	1,589,036,664.03
		Mountaineering Equipment	39.9%	409,660,132.90
		Personal Accessories	41.2%	1,885,673,307.78
		Outdoor Protection	60.5%	75,994,296.25
		Golf Equipment	48.5%	726,411,367.89
Golf Equipment	726,411,367.89	Product line	Profit margin	Revenue
		Camping Equipment	36.9%	1,589,036,664.03
		Mountaineering Equipment	39.9%	409,660,132.90
		Personal Accessories	41.2%	1,885,673,307.78
		Outdoor Protection	60.5%	75,994,296.25
		Golf Equipment	48.5%	726,411,367.89
Mountaineering Equipment				

The report opens in Cognos Viewer. Profit Margin and Revenue is repeated for each Product line. Next, you will link the two list reports on Product line.

Task 4. Create a master-detail link.

1. Close **IBM Cognos Viewer**.
2. Under **List**, click **Product line**, and then, from the **Data** menu, click **Master Detail Relationships**.
3. Click **New Link**.
4. Select **Product line** in both columns.

The results appear as follows:



A link appears between the first column of each query (between Product line and Product line). This is the link that you want to create, so you do not have to modify it.

5. Click **OK**, and then **Run** the report.
- The results appear similar to the following:

Product line	Revenue	List		
		Product line	Profit margin	Revenue
Camping Equipment	1,589,036,664.03	Camping Equipment	36.9%	1,589,036,664.03
Golf Equipment	726,411,367.89	Golf Equipment	48.5%	726,411,367.89
Mountaineering Equipment	409,660,132.90	Mountaineering Equipment	39.9%	409,660,132.90

The margin and revenue data for each Product line appears in the proper rows. You have left the Product line column in the report for verification, but you would remove it from the report layout for the production report.

6. Close **IBM Cognos Viewer**, and then close **Report Studio** without saving the report.
7. Close **IBM Cognos Connection**, and **Framework Manager**, without saving the changes.

Results:

You added an additional cube to your existing model and package. You then published the package and created a report that linked data from the new cube to the original cube.

Summary

- At the end of this module, you should be able to:
 - connect to an OLAP data source (cube) in a Framework Manager project
 - publish an OLAP model
 - publish a model with multiple OLAP data sources
 - publish a model with an OLAP data source and a relational data source

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

16-13

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



Advanced Generated SQL Concepts and Complex Queries

IBM Cognos BI

Business Analytics software

© 2015 IBM Corporation



This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

17-2

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Objectives

- At the end of this module, you should be able to:
 - governors that affect SQL generation
 - stitch query SQL
 - conformed and non-conformed dimensions in generated SQL
 - multi-fact/multi-grain stitch query SQL
 - variances in Report Studio generated SQL
 - dimensionally modeled relational SQL generation
 - cross join SQL
 - various results sets for multi-fact queries

© 2015 IBM Corporation

Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

- User ID: admin
- Password: Education1

Business Analytics software

Explore SQL Generation

```

select
    coalesce(D2.Year1,D3.Year1) as Year1,
    D2.Revenue as Revenue,
    D3.Sales_Target as Sales_Target
from
    (select
        Time_Dimension.Year1 as Year1,
        XSUM(Sales_Fact.Revenue for
            Time_Dimension.Year1) as Revenue
    from
        .....
    where
        (Time_Dimension.Day_Date =
        Sales_Fact.Order_Date)
    group by
        Time_Dimension.Year1
    ) D2
    full outer join
    (select
        Time_Dimension.Year1 as Year1,
        XSUM(Sales_Target.Sales_Target
            for Time_Dimension.Year1) as
            Sales_Target
    from
        .....
    where
        ((Time_Dimension.Year1 =
        Sales_Target.Sales_Year) and
        (Time_Dimension.Month1 =
        Sales_Target.Sales_Period))
    group by
        Time_Dimension.Year1
    ) D3
    on (D2.Year1 = D3.Year1)

```

© 2015 IBM Corporation

Confusion is a common reaction to the SQL illustrated in the slide example. Frequently asked questions are:

- What is the coalesce function?
- Why do I see the same columns being selected in two different derived tables?
- Why do I see a full outer join?
- What does the XSUM function do?

These questions, and others, will be answered throughout this module as you explore complex Cognos generated SQL.

The SQL in the slide example is Cognos SQL generated, based on the selection of various query items, and with auto aggregation enabled. Most samples shown in this module will use SQL with auto aggregation values, since it is the default setting in the studios.

Governors That Affect SQL Generation

- Outer Joins (Allow, Deny)
- Cross-Product Joins (Allow, Deny)
- Shortcut Processing (Automatic, Explicit)
- SQL Join Syntax (Explicit, Implicit)
- Grouping of Measure Attributes (Enable, Disable)
- SQL Generation for Level Attributes (Group by, Minimum)
- SQL Generation for Determinant Attributes (Group by, Minimum)
- SQL Parameter Syntax (Marker, Literal)
- Use WITH clause when generating SQL (Yes, No)

© 2015 IBM Corporation



Several project governors can limit queries and affect the SQL generated at run time. This module examines only some of the governors in this slide in detail, since some are used in rare cases. Please refer to the product documentation for details on each governor setting.

The SQL Join Syntax governor controls how SQL is generated for inner joins. Selecting Explicit will generate INNER JOIN syntax, and selecting Implicit will use WHERE syntax. This setting you choose depends on your own personal preference.

The Use WITH clause when generating SQL governor lets you choose to use the WITH clause with Cognos SQL if your data source supports it.

The With clause governor toggles Common Table Expression syntax. We currently only support the Non-Recursive form of common table expressions. The With clause is used to avoid scanning the same table several times if the query against it is required more than once in a larger query.

Business Analytics software

IBM

Using Derived Tables (1 of 2)

- derived tables:
 - are aliased sub-selects
 - enable developers to see values being returned without viewing complex SQL
 - are generated in both Cognos SQL and Native SQL

Cognos SQL

```
select
  .....
  from
  (select
    .....
  ) D2
```

Derived Table **Derived Table Alias**

© 2015 IBM Corporation

A derived table retrieves a record set that fulfills the requirements of the parent query. Although the use of derived tables can create queries that are very long and verbose, the advantage is that they articulate the work being done by the query in blocks that can be linked back to the database. Not only are derived tables essential for complex queries that require layers of calculations and filters, but they also make the queries easier to debug. Each block of native SQL in a derived table query can be executed independently in the native interface of a database vendor, and are therefore more easily diagnosed when the behavior is not as expected.

The outer blocks of SQL are derived from the inner blocks of SQL.

In Oracle shops, derived tables are known as "in-line views".

Using Derived Tables (2 of 2)

```

select
    coalesce(D2.Year1,D3.Year1) as Year1,
    D2.Revenue as Revenue,
    D3.Sales_Target as Sales_Target
from
    (select
        Time_Dimension.Year1 as Year1,
        XSUM(Sales_Fact.Revenue for
            Time_Dimension.Year1) as Revenue
    from
        .....
    where
        (Time_Dimension.Day_Date =
            Sales_Fact.Order_Date)
    group by
        Time_Dimension.Year1
    )D2
    full outer join
    (select
        Time_Dimension.Year1 as Year1,
        XSUM(Sales_Target.Sales_Target for
            Time_Dimension.Year1) as Sales_Target
    from
        .....
    where
        ((Time_Dimension.Year1 =
            Sales_Target.Sales_Year) and
        (Time_Dimension.Month1 =
            Sales_Target.Sales_Period))
    group by
        Time_Dimension.Year1
    )D3
on (D2.Year1 = D3.Year1)

```

© 2015 IBM Corporation



Derived tables use alias names that make it easy to identify the query from which the projected items come. In the slide example, there are two derived tables, D2 and D3, which achieve the final projections list. The derived table alias names are also used in the join statement.

Derived tables are instrumental in stitch queries. So we will review them first before examining stitch query SQL.

Each derived table returns values that are used to achieve the final projection list (the first select statement) as well as values to achieve the final join statement. Notice the alias names in the first select statement (top left corner) and in the final join statement (bottom right corner).

Business Analytics software

IBM

Identifying Stitch Query SQL

```

select
  coalesce(D2.Year1,D3.Year1) as Year1,
  D2.Revenue as Revenue,
  D3.Sales_Target as Sales_Target
from
  (select
    Time_Dimension.Year1 as Year1,
    XSUM(Sales_Fact.Revenue for
      Time_Dimension.Year1) as Revenue
   from
    .....
   where
    (Time_Dimension.Day_Date =
     Sales_Fact.Order_Date)
   group by
    Time_Dimension.Year1
  ) D2
  full outer join
  (select
    Time_Dimension.Year1 as Year1,
    XSUM(Sales_Target.Sales_Target
      for Time_Dimension.Year1) as
      Sales_Target
   from
    .....
   where
    ((Time_Dimension.Year1 =
      Sales_Target.Sales_Year) and
     (Time_Dimension.Month1 =
      Sales_Target.Sales_Period))
   group by
    Time_Dimension.Year1
  ) D3
  on (D2.Year1 = D3.Year1)

```

© 2015 IBM Corporation

Stitch queries are used to achieve predictable results for multi-fact queries.

There are three essential components of a stitch query:

- coalesce function
- full outer join
- multiple queries that query some of the same information

In the slide, the SQL represents a multi-fact query based on one conformed dimension (Time Dimension) and two facts (Sales_Fact.Revenue, and Sales_Target.Sales_Target). If the query requires local processing, IBM Cognos includes a local relational engine that is able to process local stitch queries efficiently.

In the slide, Year1 is queried more than once. Repeated information comes from conformed dimensions used in multi-fact queries. This is done to generate result sets that can be merge-sorted together in the full outer join.

For each additional fact table that is included in a query, there will be another full outer join. If you added another fact to the query, there would be another full outer join and another derived table for the new fact introduced into the query.

Business Analytics software

IBM

What is a Coalesce Function?

```
select
    coalesce(D2.DAY_DATE,D3.DAY_DATE) as DAY_DATE,
    coalesce(D2.ORDER_METHOD,D3.ORDER_METHOD) as ORDER_METHOD,
```

Sales Fact			Returned Items Fact		
DAY_DATE	ORDER_METHOD	SALE_TOTAL	DAY_DATE	ORDER_METHOD	RETURN_QUANTITY
01/01/2011	E-mail	\$10	01/01/2011	E-mail	2
01/02/2011	Telephone	\$25	01/02/2011	Telephone	4
01/03/2011	Web	\$40	01/10/2011	Fax	15
01/04/2011	E-mail	\$20	01/11/2011	Sales visit	1

Report Set

DAY_DATE	ORDER_METHOD	SALE_TOTAL	RETURN_QUANTITY
01/01/2011	E-mail	\$10	2
01/02/2011	Telephone	\$25	4
01/03/2011	Web	\$40	
01/04/2011	E-mail	\$20	
01/10/2011	Fax		15
01/11/2011	Sales visit		1



A coalesce function:

- merges query items that exist on multiple sides of the query
- indicates that a query item is part of a conformed dimension

Anything included in the report as a column, filter, or prompt can be treated as a conformed dimension if it is common to the fact items in the query.

Coalesce functions:

- return the first non-NULL value from a series of expressions. It returns NULL if the series contains only NULL.
- create the results set, including nulls, but not necessarily by the shared dimension key. For example, if two different products had the same product name, they would end up as a single result on this report.

Business Analytics software

IBM

Non-Conformed Dimensions in Generated SQL

```

select
    coalesce(D2.DAY_DATE,D3.DAY_DATE) as DAY_DATE,
    coalesce(D2.ORDER_METHOD,D3.ORDER_METHOD) as ORDER_METHOD,
D3.REASON_DESCRIPTION as REASON_DESCRIPTION, ← Non-conformed dimension
    D2.SALE_TOTAL as SALE_TOTAL,
    D3.RETURN_QUANTITY as RETURN_QUANTITY
from
    (select
        Time_Dimension.DAY_DATE as DAY_DATE,
        Order_Method_Dimension.ORDER_METHOD_EN as ORDER_METHOD,
        Return_Reason_Dimension.REASON_DESCRIPTION REASON_DESCRIPTION,
        .....
        group by
            Time_Dimension.DAY_DATE,
            Order_Method_Dimension.ORDER_METHOD_EN,
            Return_Reason_Dimension.REASON_DESCRIPTION ← Found only in query of related fact
    ) D3
full outer join
    (select
        .....
    ) D2
on ((D2.DAY_DATE = D3.DAY_DATE) and (D2.ORDER_METHOD = D3.ORDER_METHOD))

```

© 2015 IBM Corporation



Non-conformed dimensions will not use a coalesce function, and only show up in the derived table of the fact to which they are related.

If you are using what you expect to be a conformed dimension in a multi-fact query and no coalesce function is generated, you should investigate your model to ensure that no query path has been missed, or that the IBM Cognos query engine is not identifying the dimension as a fact based on cardinality.

REASON_DESCRIPTION comes from the Return Reason Dimension which is not conformed between the Sales Fact and Returns Fact tables. Because it is not conformed, it will only be a part of the derived table query that is related to the Returned Items Fact table. By looking at the SQL, we can quickly determine a non-conformed dimension by the absence of the coalesce function

What is an RSUM(1 .. asc local) as sc? (1 of 2)

- IBM Cognos is generating a stitch column that will be used to stitch queries together locally
- found in multi-fact queries when automatic summarization is not enabled, there is no common level of granularity, and either:
 - at least one conformed dimension is present

`RSUM(1 for Order_Method_Dimension.ORDER_METHOD_KEY order by Order_Method_Dimension.ORDER_METHOD_KEY asc local) as sc1`

- no conformed dimensions are present

`RSUM(1 order by Sales_Target_Fact.SALES_TARGET asc local) as sc`

© 2015 IBM Corporation



If automatic aggregation is not enabled and at least one conformed dimension is present, you will see the same stitch column generated in both derived tables of a stitch query. This stitch column takes a common key(s) from the conformed dimension(s) between the two queries and sorts it ascending locally on the IBM Cognos server. These columns and others then merge the two result sets. The fact values in the fact columns will be related to the conformed dimension but not necessarily to each other.

If automatic aggregation is not enabled and there are no conformed dimensions present, IBM Cognos will attempt to generate a stitch column by selecting a column from each query and using it to create unique values that will merge the queries. There are no definite relationships between the facts.

In either case RSUM(1....asc local) as sc is cause for investigation to ensure the correct results are returned.

A running total (RSUM) is used to create unique instances of the stitch key. The sc stands for stitch column.

Business Analytics software

IBM

What is an RSUM(1 asc local) as sc? (2 of 2)

- the following syntax indicates that a conformed dimension is not included in a multi-fact query

Stitch column from Query 1 of stitch query

```

    RSUM(1 order by ORDER_DETAILS.ORDER_DETAIL_CODE asc local) as sc
    .....
    full outer join
    .....
    RSUM(1 order by PRODUCT_FORECAST.PRODUCT_NUMBER asc local) as sc
    .....
    on (Query 1.sc = Query 2.sc)
  
```

Stitch column from Query 2 of stitch query

© 2015 IBM Corporation

Each query uses a different column to generate the stitch column that in most cases, returns unrelated results.

This functionality may not work, depending on the nature of the data. For predictable results, ensure that there is a conformed dimension between the two facts and a common level of granularity.

If you do see this type of SQL being generated, you should investigate and rework your query so that it appears as expected.

Why Do I See XSUM?

- Cognos SQL uses windowed aggregates to improve readability
- Cognos SQL:
 - **XSUM**(Sales_Fact.SALE_TOTAL for Time_Dimension.MONTH_KEY) as SALE_TOTAL
- Native SQL:
 - **sum**("Sales_Fact"."SALE_TOTAL") AS "SALE_TOTAL"

© 2015 IBM Corporation



XSUM in Cognos SQL indicates a windowed aggregate, in which you can see what value is being aggregated and to what level(s).

The X in XSUM stands for extended, which indicates that the overall total for each row of a particular grouping will be calculated and retrieved. Extended vs. Running aggregates (XSUM vs. RSUM) are discussed later in this module.

For database vendors who support SQL-OLAP aggregates such as Oracle and DB2, you can also quickly identify to what levels facts are aggregated. For example Oracle uses sum and over syntax as shown below:

```
select
  "T0"."C0" "ORDERDATE", "T0"."C1" "ACTUALREVENUE",
  sum("T0"."C1") over (order by "T0"."C0" asc rows unbounded preceding)
  "ACTUALREVENUE1"
from
```

Demo 1: Identify Stitch Queries in Generated SQL

Purpose:

When running multi-fact queries, you must identify the components of the generated SQL. Understanding the patterns of correctly generated stitch queries will let you effectively troubleshoot improperly constructed queries. To that end, you will test various multi-fact query scenarios and explore the generated SQL.

Component: **Framework Manager**

Project: **GO Operational**

Task 1. Test fact queries individually.

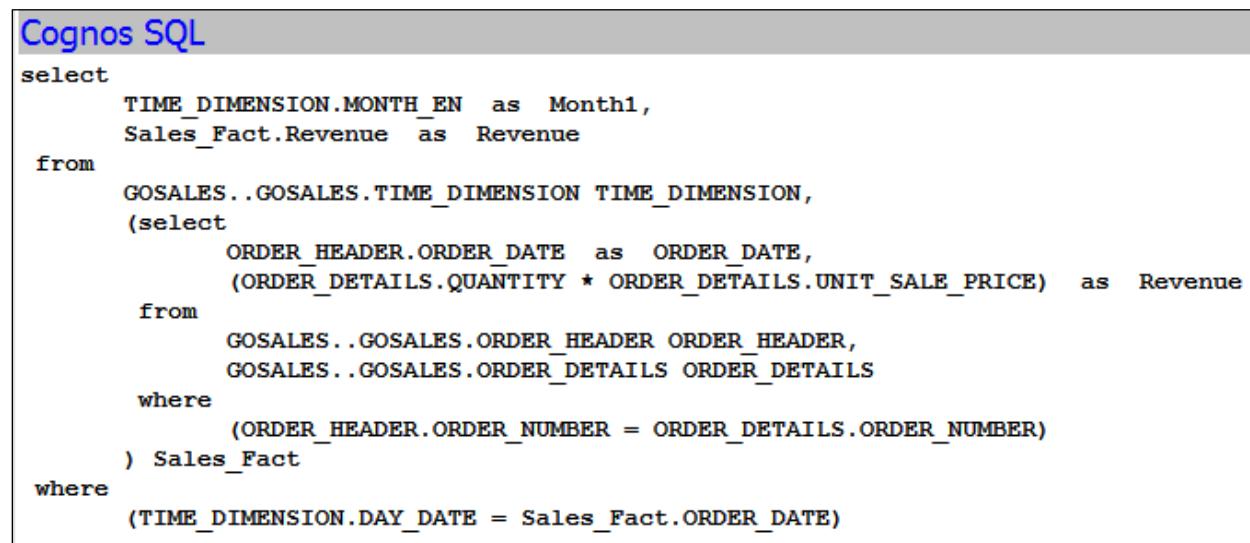
1. In **Framework Manager**, open the **GO Operational** project located at **C:\Edcognos\B5A52\CBIFM-Start Files\Module 17\GO Operational**. If prompted, log in as User ID **admin**, and Password **Education1**.
2. In the **Project Viewer**, expand **GO Operational Model > Consolidation View**.
3. **Test** the following items with Auto Sum deselected:

Query Subject	Query Item
Time	Month
Sales Fact	Revenue

The Test Results dialog appears. You will view the generated SQL before applying Auto Sum.

4. Click the **Query Information** tab.

The results appear as follows:



```

Cognos SQL

select
    TIME_DIMENSION.MONTH_EN as Month1,
    Sales_Fact.Revenue as Revenue
from
    GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
    (select
        ORDER_HEADER.ORDER_DATE as ORDER_DATE,
        (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
    from
        GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
        GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
    where
        (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
    ) Sales_Fact
where
    (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)

```

This is a basic query where the TIME_DIMENSION dimension and Sales Fact query subjects are joined. A derived table is generated in the Cognos SQL for Sales Fact since the Revenue column you selected in the query is based on a calculation. Again, Cognos SQL is more verbose. The native SQL does not require a derived table for this basic calculation based on columns from the same table.

Note: Month1 appears in the generated SQL as opposed to Month because month is a reserved word. Therefore, a 1 is appended to avoid any conflicts.

5. Click the **Test** tab, select the **Auto Sum** check box, and then click **Test Sample**.

6. Click the **Query Information** tab.

The results appear as follows:

```
Cognos SQL
select
    TIME_DIMENSION.MONTH_EN as Month1,
    XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN) as Revenue
from
    GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
    (select
        ORDER_HEADER.ORDER_DATE as ORDER_DATE,
        (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
    from
        GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
        GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
    where
        (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
    ) Sales_Fact
where
    (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
group by
    TIME_DIMENSION.MONTH_EN

Native SQL
select "TIME_DIMENSION"."MONTH_EN" "Month1" , sum("coguda11"."QUANTITY" * "coguda11"."UNIT_SALE_PRICE")
"Revenue" from "GOSALES"."TIME_DIMENSION" "TIME_DIMENSION" , "GOSALES"."ORDER_HEADER" "coguda10",
"GOSALES"."ORDER_DETAILS" "coguda11" where "coguda10"."ORDER_NUMBER" = "coguda11"."ORDER_NUMBER" and
"TIME_DIMENSION"."DAY_DATE" = "coguda10"."ORDER_DATE" group by "TIME_DIMENSION"."MONTH_EN" FOR FETCH ONLY
```

XSUM has been applied in the Cognos SQL to aggregate Revenue to the month level. The sum function is used in the native SQL.

7. Click **Close**.

8. In the **Project Viewer**, test the following items together from the **Consolidation View**:

Query Subject	Query Item
Time	Month
Sales Target Fact	Sales Target

- Click the **Query Information** tab.

The results appear as follows:

```
Cognos SQL
select
    TIME_DIMENSION.MONTH_EN as Month1,
    SALES_TARGET.SALES_TARGET as Sales_Target
from
    (select
        TIME_DIMENSION.CURRENT_YEAR as CURRENT_YEAR,
        TIME_DIMENSION.CURRENT_MONTH as CURRENT_MONTH,
        XMIN(TIME_DIMENSION.MONTH_EN for TIME_DIMENSION.CURRENT_YEAR,TIME_DIMENSION.CURRENT_MONTH ) a
    from
        GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION
    group by
        TIME_DIMENSION.CURRENT_YEAR,
        TIME_DIMENSION.CURRENT_MONTH
    ) TIME_DIMENSION,
    GOSALES..GOSALES.SALES_TARGET SALES_TARGET
where
    ((TIME_DIMENSION.CURRENT_YEAR = SALES_TARGET.SALES_YEAR) and (TIME_DIMENSION.CURRENT_MONTH = SALES_TAR
```

Because Sales Target rolls up to the month level (instead of day level), and you have specified determinants on the TIME_DIMENSION, an XMIN function and a group by clause is generated in the Cognos SQL. The XMIN function in the Cognos SQL (min in the native SQL) ensures that only one month value is returned for each month. The determinant for the month level specifies a multi-part key, which is why the derived table for TIME_DIMENSION uses a group by clause on CURRENT_YEAR and CURRENT_MONTH. This prevents double-counting for Sales Target, because the values are not aggregated for every day in the month, but rather at the grouped month level.

Without the determinants, the generated SQL would not include the XMIN function or a grouping on the keys. You will test this in the next few steps.

- Click **Close**, expand Foundation Objects View > **gosales**, and then double-click **TIME_DIMENSION**.

The X in XMIN will be covered in more detail later in this module. XMIN is used in order to prevent 'dirty' data from slipping through. For example, the data may include Aug and August. By using min, only one of these values is selected, which results in a cleaner view of the data. If this is not desired, you can use the SQL Generation for Determinant Attributes governor to change the behavior.

- Click the **Determinants** tab, delete all the determinants except for the **Day** determinant, and then click **OK**.

12. Test the following query items in the **Consolidation View**:

Query Subject	Query Item
Time	Month
Sales Target Fact	Sales Target

Remember, the items in the Consolidation View are based on items in the Foundation Objects View. You are testing the objects that authors will use.

13. Click the **Query Information** tab.

The results appear as follows:

```
Cognos SQL
select
    TIME_DIMENSION.MONTH_EN as Month1,
    SALES_TARGET.SALES_TARGET as Sales_Target
from
    GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
    GOSALES..GOSALES.SALES_TARGET SALES_TARGET
where
    ((TIME_DIMENSION.CURRENT_YEAR = SALES_TARGET.SALES_YEAR) and (TIME_DIMENSION.CURRENT_MONTH = SALES_TARGET.SALES_PERIOD))

Native SQL
select "TIME_DIMENSION"."MONTH_EN" "Month1" , "SALES_TARGET"."SALES_TARGET" "Sales_Target" from "GOSALES"."TIME_DIMENSION"
"TIME_DIMENSION", "GOSALES"."SALES_TARGET" "SALES_TARGET" where "TIME_DIMENSION"."CURRENT_YEAR" = "SALES_TARGET"."SALES_YEAR" and
"TIME_DIMENSION"."CURRENT_MONTH" = "SALES_TARGET"."SALES_PERIOD" FOR FETCH ONLY
```

The XMIN function and group by clause is no longer present. Now, if you aggregated Sales Target, each value would be double-counted, once for every day in the month for which it is associated.

You can verify the effect of double-counting by retesting with Auto Sum enabled. Take note of the totals and then compare them to the values seen in the following steps when determinants are returned.

14. Click **Close**, and then from the **Edit** menu, click **Undo Edit Definition**.

15. Retest the same query items with **Auto Sum** enabled, and then click the **Query Information** tab.

The results appear as shown below:

Cognos SQL
<pre> select TIME_DIMENSION.MONTH_EN as Month1, XSUM(SALES_TARGET.SALES_TARGET for TIME_DIMENSION.MONTH_EN) as Sales_Target from (select TIME_DIMENSION.CURRENT_YEAR as CURRENT_YEAR, TIME_DIMENSION.CURRENT_MONTH as CURRENT_MONTH, XMIN(TIME_DIMENSION.MONTH_EN for TIME_DIMENSION.CURRENT_YEAR, TIME_DIMENSION.CURRENT_MONTH) as MONTH_EN from GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION group by TIME_DIMENSION.CURRENT_YEAR, TIME_DIMENSION.CURRENT_MONTH) TIME_DIMENSION, GOSALES..GOSALES.SALES_TARGET SALES_TARGET where ((TIME_DIMENSION.CURRENT_YEAR = SALES_TARGET.SALES_YEAR) and (TIME_DIMENSION.CURRENT_MONTH = SALES_TARGET.SALES_PERIOD)) group by TIME_DIMENSION.MONTH_EN </pre>
Native SQL
<pre> select "TIME_DIMENSION"."MONTH_EN" "Month1", sum("SALES_TARGET"."SALES_TARGET") "Sales_Target" from (select "TIME_DIMENSION"."CURRENT_YEAR" "CURRENT_YEAR", "TIME_DIMENSION"."CURRENT_MONTH" "CURRENT_MONTH", min ("TIME_DIMENSION"."MONTH_EN") "MONTH_EN" from "GOSALES"."TIME_DIMENSION" "TIME_DIMENSION" group by "TIME_DIMENSION"."CURRENT_YEAR", "TIME_DIMENSION"."CURRENT_MONTH") "TIME_DIMENSION", "GOSALES"."SALES_TARGET" "SALES_TARGET" where "TIME_DIMENSION"."CURRENT_YEAR" = "SALES_TARGET"."SALES_YEAR" and "TIME_DIMENSION"."CURRENT_MONTH" = "SALES_TARGET"."SALES_PERIOD" group by "TIME_DIMENSION"."MONTH_EN" FOR FETCH ONLY </pre>

XSUM has been applied in the Cognos SQL to aggregate Sales Target to the month level. The sum function is used in the native SQL. The sales target values are correct now that determinants have been re-applied.

16. Click **Close**.

Task 2. Test multi-fact and multi-grain queries.

Now that you have reviewed how the generated SQL appears for each of the fact queries individually, you will test them together. This task also illustrates multiple levels of granularity, since Revenue rolls up to the day level, and Sales Target rolls up to the month level.

1. In the **Project Viewer**, in the **Consolidation View**, test the following items together:

Query Subject	Query Item
Time	Month
Sales Fact	Revenue
Sales Target Fact	Sales Target

Due to the large data set, this query may take some time.

2. Click the **Query Information** tab.

The results appear as follows:

```
Cognos SQL
select
    coalesce(D2.Month2,D3.Month2) as Month1,
    D2.Revenue as Revenue,
    D3.Sales_Target as Sales_Target
from
    (select
        TIME_DIMENSION.DAY_DATE as sc,
        TIME_DIMENSION.MONTH_EN as Month2,
        Sales_Fact.Revenue as Revenue,
        RSUM(1 for TIME_DIMENSION.DAY_DATE order by TIME_DIMENSION.DAY_DATE asc local) as sc4
    from
        GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
        (select
            ORDER_HEADER.ORDER_DATE as ORDER_DATE,
            (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
        from
            GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
            GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
        where
            (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
        ) Sales_Fact
    where
        (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
    order by
        sc asc
    ) D2
    full outer join
    (select
        TIME_DIMENSION.DAY_DATE as sc,
        TIME_DIMENSION.MONTH_EN as Month2,
        SALES_TARGET.SALES_TARGET as Sales_Target,
        RSUM(1 for TIME_DIMENSION.DAY_DATE order by TIME_DIMENSION.DAY_DATE asc local) as sc4
    from
        GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
        GOSALES..GOSALES.SALES_TARGET SALES_TARGET
    where
        ((TIME_DIMENSION.CURRENT_YEAR = SALES_TARGET.SALES_YEAR) and (TIME_DIMENSION.CURRENT_MONTH = SALES_TARGET.SALES_PERIOD))
    order by
        sc asc
    ) D3
    on ((D2.sc = D3.sc) and (D2.sc4 = D3.sc4))
```

The generated Cognos SQL contains a coalesce function, because Time is a conformed dimension between Sales Fact and Sales Target Fact. Auto Sum is not enabled, so there are two facts without a common level of granularity between them. Because Revenue rolls up to the day level, and Sales Target rolls up to the month level, the RSUM(1....asc local) as sc syntax appears in both derived tables (D2 and D3), which are joined together by a full outer join. The full outer join uses two stitch columns, sc, and sc4, to merge the two record sets together.

The native SQL does not contain a full outer join. It is actually two separate queries sent to the database, since you are asking for local processing in our RSUM functions. Therefore, the merging is done locally. The merged results for Revenue and Sales Target will be related to Month, but not necessarily to each other.

3. Click the **Test** tab, select **Auto Sum**, and then click **Test Sample**.

4. Click the **Query Information** tab.

The results appear as shown below:

```
Cognos SQL
select
    coalesce(D2.Month1,D3.Month1) as Month1,
    D2.Revenue as Revenue,
    D3.Sales_Target as Sales_Target
from
    (select
        TIME_DIMENSION.MONTH_EN as Month1,
        XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN ) as Revenue
    from
        GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
        (select
            ORDER_HEADER.ORDER_DATE as ORDER_DATE,
            (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
        from
            GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
            GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
        where
            (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
        ) Sales_Fact
    where
        (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
    group by
        TIME_DIMENSION.MONTH_EN
    ) D2
full outer join
(select
    TIME_DIMENSION.MONTH_EN as Month1,
    XSUM(SALES_TARGET.SALES_TARGET for TIME_DIMENSION.MONTH_EN ) as Sales_Target
from
    (select
        TIME_DIMENSION.CURRENT_YEAR as CURRENT_YEAR,
        TIME_DIMENSION.CURRENT_MONTH as CURRENT_MONTH,
        XMIN(TIME_DIMENSION.MONTH_EN for TIME_DIMENSION.CURRENT_YEAR,TIME_DIMENSION.CURRENT_MONTH ) as MONTH_EN
    from
        GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION
    group by
        TIME_DIMENSION.CURRENT_YEAR,
        TIME_DIMENSION.CURRENT_MONTH
    ) TIME_DIMENSION,
    GOSALES..GOSALES.SALES_TARGET SALES_TARGET
    where
        ((TIME_DIMENSION.CURRENT_YEAR = SALES_TARGET.SALES_YEAR) and (TIME_DIMENSION.CURRENT_MONTH = SALES_TARGET.SALES_PERIOD))
    group by
        TIME_DIMENSION.MONTH_EN
) D3
on (D2.Month1 = D3.Month1)
```

The D2 and D3 derived tables represent the same generated SQL you saw when you tested the fact queries individually. However, they are now a part of a larger query that will merge them. There is also a common level of granularity between the queries since both are now aggregated to and grouped by the Month level. In the Native SQL, IBM Cognos is now sending a single query and requesting that the database process the full outer join.

5. Click **Close**.

Task 3. Identify improperly formed multi-fact queries.

1. In the **Project Viewer**, in the **Consolidation View**, test the following items together:

Query Subject	Query Item
Order Method	Order Method
Sales Fact	Revenue
Sales Target Fact	Sales Target

2. Click the **Query Information** tab.

The results appear as follows:

```
Cognos SQL
select
    D2.Order_Method as Order_Method,
    D2.Revenue as Revenue,
    D3.Sales_Target as Sales_Target
from
    (select
        ORDER_METHOD.ORDER_METHOD_EN as Order_Method,
        Sales_Fact.Revenue as Revenue,
        RSUM(1 order by ORDER_METHOD.ORDER_METHOD_EN asc local) as sc
    from
        GOSALES..GOSALES.ORDER_METHOD ORDER_METHOD,
        (select
            ORDER_HEADER.ORDER_METHOD_CODE as ORDER_METHOD_CODE,
            (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
        from
            GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
            GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
        where
            (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
        ) Sales_Fact
    where
        (ORDER_METHOD.ORDER_METHOD_CODE = Sales_Fact.ORDER_METHOD_CODE)
    order by
        Order_Method asc
    ) D2
    full outer join
    (select
        SALES_TARGET.SALES_TARGET as Sales_Target,
        RSUM(1 order by SALES_TARGET.SALES_TARGET asc local) as sc
    from
        GOSALES..GOSALES.SALES_TARGET SALES_TARGET
    order by
        Sales_Target asc
    ) D3
    on (D2.sc = D3.sc)
```

You can immediately tell that this query is suspect by the absence of the coalesce function. When you author multi-fact queries, you must include at least one conformed dimension. Other indicators are the generated stitch columns highlighted above. Each one uses a different column to generate a key that will merge the queries.

3. Click the **Test** tab, select **Auto Sum**, and then click **Test Sample**.

The results appear as follows:

Order Method	Revenue	Sales Target
E-mail	179843044.16	4205368540
Fax	70073542.01	4205368540
Mail	46091338.97	4205368540
Sales visit	310194834	4205368540
Special	27351320.25	4205368540
Telephone	340985781.06	4205368540
Web	3712235908.4	4205368540

Sales Target repeats the overall total for each row.

4. Click the **Query Information** tab.

The results appear as follows:

```
Cognos SQL
select
    D2.Order_Method as Order_Method,
    D2.Revenue as Revenue,
    D3.Sales_Target as Sales_Target
from
    (select
        ORDER_METHOD.ORDER_METHOD_EN as Order_Method,
        XSUM(Sales_Fact.Revenue for ORDER_METHOD.ORDER_METHOD_EN ) as Revenue
    from
        GOSALES..GOSALES.ORDER_METHOD ORDER_METHOD,
        (select
            ORDER_HEADER.ORDER_METHOD_CODE as ORDER_METHOD_CODE,
            (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
        from
            GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
            GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
        where
            (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
        ) Sales_Fact
    where
        (ORDER_METHOD.ORDER_METHOD_CODE = Sales_Fact.ORDER_METHOD_CODE)
    group by
        ORDER_METHOD.ORDER_METHOD_EN
    ) D2,
    (select distinct
        XSUM(SALES_TARGET.SALES_TARGET ) as Sales_Target
    from
        GOSALES..GOSALES.SALES_TARGET SALES_TARGET
    ) D3
```

The query can still be identified as an improper attempt to query multiple facts. There is no coalesce function, and you are sending two separate queries, one for each fact. The first derived table, which queries the Revenue fact, appears to be correct. But the second derived table, which queries the Sales Target fact, is simply requesting a distinct overall total. This value will be repeated for each row returned by the first derived table.

5. Click **Close**, and leave Framework Manager open.

Results:

By testing various fact and multi-fact queries, you identified traits and patterns in the generated SQL, which indicate correctly modeled query subjects. You also examined and identified patterns for improperly formed multi-fact queries.

Workshop 1: Reverse Engineer a Framework Manager Model from Generated Cognos SQL

You have just been handed some generated SQL from a report author who is concerned about the SQL, which uses explicit join syntax (INNER JOIN) rather than implicit join syntax (WHERE clauses). You will review the SQL and try to reverse-engineer the model, based on the query subjects used to author the report. During this process, note any unexpected SQL, and explain it.

Write notes in the margin explaining the various portions of the SQL. Use the blank space provided to draw the report that this SQL created, and a free-hand diagram of the portion of the model representing the queries used in the report. Be sure to include the cardinalities in the diagram.

```

select
    coalesce(D2.MONTH1,D3.MONTH1) as MONTH1,
    coalesce(D2.ORDER_METHOD,D3.ORDER_METHOD) as
        ORDER_METHOD,
    D3.REASON_DESCRIPTION as REASON_DESCRIPTION,
    D2.QUANTITY as QUANTITY,
    D3.RETURN_QUANTITY as RETURN_QUANTITY

from

(select

    Time_Dimension.MONTH1 as MONTH1,
    Order_Method_Dimension.ORDER_METHOD as
        ORDER_METHOD,
    Return_Reason_Dimension.REASON_DESCRIPTION as
        REASON_DESCRIPTION,
    XSUM(Returned_Items_Fact.RETURN_QUANTITY for
    Time_Dimension.MONTH1,Order_Method_Dimension.ORDER_M
    ETHOD,Return_Reason_Dimension.REASON_DESCRIPTION )
    as RETURN_QUANTITY

from

(select

    Time_Dimension.DAY_KEY as DAY_KEY,
    Time_Dimension.MONTH_EN as MONTH1
from

    go_data_warehouse.GOSLDW.dbo.TIME_DIMENSION
    Time_Dimension) Time_Dimension

join

```

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

17-25

```

go_data_warehouse.GOSLDW.dbo.RETURNED_ITEMS_FACT
    Returned_Items_Fact

on (Time_Dimension.DAY_KEY =
    Returned_Items_Fact.DAY_KEY)

join

(select

    Order_Method_Dimension.ORDER_METHOD_KEY
        as ORDER_METHOD_KEY,
    Order_Method_Dimension.ORDER_METHOD_EN
        as ORDER_METHOD

from

go_data_warehouse.GOSLDW.dbo.ORDER_METHOD_DIMENSION
Order_Method_Dimension) Order_Method_Dimension

on (Order_Method_Dimension.ORDER_METHOD_KEY =
    Returned_Items_Fact.ORDER_METHOD_KEY)

join

(select

    Return_Reason_Dimension.RETURN_REASON_KEY  as
    RETURN_REASON_KEY,
    Return_Reason_Dimension.REASON_DESCRIPTION_EN
        as REASON_DESCRIPTION

from

go_data_warehouse.GOSLDW.dbo.RETURN_REASON_DIMENSION
Return_Reason_Dimension) Return_Reason_Dimension

on (Return_Reason_Dimension.RETURN_REASON_KEY
    = Returned_Items_Fact.RETURN_REASON_KEY)

group by
    Time_Dimension.MONTH1,
    Order_Method_Dimension.ORDER_METHOD,
    Return_Reason_Dimension.REASON_DESCRIPTION

) D3

full outer join

(select

    Time_Dimension.MONTH1  as MONTH1,
    Order_Method_Dimension.ORDER_METHOD  as
        ORDER_METHOD,
    XSUM(Sales_Fact.QUANTITY  for
    Time_Dimension.MONTH1,Order_Method_Dimension.ORDER_M
    ETHOD )  as QUANTITY

```

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

```

from

(select

    Time_Dimension.DAY_KEY as DAY_KEY,
    Time_Dimension.MONTH_EN as MONTH1

from

    go_data_warehouse.GOSLDW.dbo.TIME_DIMENSION
    Time_Dimension) Time_Dimension

join

go_data_warehouse.GOSLDW.dbo.SALES_FACT
    Sales_Fact

on (Time_Dimension.DAY_KEY =
    Sales_Fact.ORDER_DAY_KEY)

join

(select

    Order_Method_Dimension.ORDER_METHOD_KEY
        as ORDER_METHOD_KEY,
    Order_Method_Dimension.ORDER_METHOD_EN
        as ORDER_METHOD

from

    go_data_warehouse.GOSLDW.dbo.ORDER_METHOD_DIMENSION
    Order_Method_Dimension) Order_Method_Dimension

on (Order_Method_Dimension.ORDER_METHOD_KEY =
    Sales_Fact.ORDER_METHOD_KEY)

group by
    Time_Dimension.MONTH1,
    Order_Method_Dimension.ORDER_METHOD

) D2

on ((D3.MONTH1 = D2.MONTH1) and (D3.ORDER_METHOD =
D2.ORDER_METHOD))

```

Workshop 1: Solution

High-Level Representation of the Overall Query

Select.....from

(Select.....from.....join.....on) D3

Query 1

Full outer join

(Select.....from.....join.....on) D2

Query 2

on ((D3.StitchKey1 = D2.StitchKey1) and
(D3.StitchKey2 = D2.StitchKey2))

```
select
```

```
coalesce(D2.MONTH1,D3.MONTH1) as MONTH1,
coalesce(D2.ORDER_METHOD,D3.ORDER_METHOD) as ORDER_METHOD,
D3.REASON_DESCRIPTION as REASON_DESCRIPTION,
D2.QUANTITY as QUANTITY,
D3.RETURN_QUANTITY as RETURN_QUANTITY
```

Initial select statement indicates the final projection list.

from

```
(select
```

```
Time_Dimension.MONTH1 as MONTH1,
Order_Method_Dimension.ORDER_METHOD as ORDER_METHOD,
Return_Reason_Dimension.REASON_DESCRIPTION as REASON_DESCRIPTION,
XSUM(Returned_Items_Fact.RETURN_QUANTITY for
Time_Dimension.MONTH1,Order_Method_Dimension.ORDER_METHOD,Return_Reason_Dimension.REASON_DESCRIPTION ) as RETURN_QUANTITY
```

The coalesce functions indicate which columns come from conformed dimensions.

REASON_DESCRIPTION is taken from a non-conformed dimension. This can be determined by the absence of a coalesce function.

These are the two facts used in the report.

This is the first query in the stitch query. Its projection list requests the conformed dimensions, one non-conformed dimension, and one of the facts, in this case RETURN_QUANTITY.

from

```
(select
```

```
Time_Dimension.DAY_KEY as DAY_KEY,
Time_Dimension.MONTH_EN as MONTH1
from
go_data_warehouse.GOSLDW.dbo.TIME_DIMENSION
Time_Dimension) Time_Dimension
```

The XSUM indicates that aggregation is occurring for specific groupings.

```
join
go_data_warehouse.GOSLDW.dbo.RETURNED_ITEMS_FACT
Returned_Items_Fact
on (Time_Dimension.DAY_KEY =
Returned_Items_Fact.DAY_KEY)
```

This derived table retrieves rows from the Time Dimension which are then joined to values from Returned Items Fact on the DAY_KEY. The final rollup for the report will be done at the month level. The same derived table is also found in the second derived table of this stitch query because Time Dimension is a conformed dimension.

join

```
(select
```

```
Order_Method_Dimension.ORDER_METHOD_KEY as ORDER_METHOD_KEY,
Order_Method_Dimension.ORDER_METHOD_EN as ORDER_METHOD
```

This derived table retrieves rows from the Order Method Dimension which are then joined to values from Returned Items Fact on the ORDER_METHOD_KEY. The final rollup for the report will also be based on this key. The same derived table is also found in the second derived table of this stitch query because Order Method Dimension is a conformed dimension.

from

```
go_data_warehouse.GOSLDW.dbo.ORDER_METHOD_DIMENSION
Order_Method_Dimension) Order_Method_Dimension
```

```
on (Order_Method_Dimension.ORDER_METHOD_KEY =
Returned_Items_Fact.ORDER_METHOD_KEY)
```

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

join

```
(select
    Return_Reason_Dimension.RETURN_REASON_KEY as RETURN_REASON_KEY,
    Return_Reason_Dimension.REASON_DESCRIPTION_EN as REASON_DESCRIPTION
    from
```

```
go_data_warehouse.GOSLDW.dbo.RETURN_REASON_DIMENSION
Return_Reason_Dimension) Return_Reason_Dimension
```

```
on (Return_Reason_Dimension.RETURN_REASON_KEY
= Returned_Items_Fact.RETURN_REASON_KEY)
```

```
group by
    Time_Dimension.MONTH1,
    Order_Method_Dimension.ORDER_METHOD,
    Return_Reason_Dimension.REASON_DESCRIPTION
```

) D3

full outer join

(select

```
Time_Dimension.MONTH1 as MONTH1,
Order_Method_Dimension.ORDER_METHOD as ORDER_METHOD,
XSUM(Sales_Fact.QUANTITY for
Time_Dimension.MONTH1, Order_Method_Dimension.ORDER_METHOD) as QUANTITY
```

from

(select

```
Time_Dimension.DAY_KEY as DAY_KEY,
Time_Dimension.MONTH_EN as MONTH1
```

from

```
go_data_warehouse.GOSLDW.dbo.TIME_DIMENSION
Time_Dimension) Time_Dimension
```

join

```
go_data_warehouse.GOSLDW.dbo.SALES_FACT
Sales_Fact
```

```
on (Time_Dimension.DAY_KEY =
Sales_Fact.ORDER_DAY_KEY)
```

join

This derived table retrieves rows from the Return Reason Dimension and joins the values with facts from Returned Items Fact on the RETURN_REASON_KEY. The final roll up for RETURN_QUANTITY will also be based on this key as seen in the group by clause. However, Return Reason Dimension is not a conformed dimension and therefore, the same derived table will not be found in the second derived table of this stitch query and it will have no impact on the roll up for the QUANTITY.

The group by clause, in this case, determines how the RETURN_QUANTITY values will be summed.

D3 is the alias name for the first derived table in this stitch query.

The full outer join pertains to the coalesce function which merges the two queries together.

This is the second query in the stitch query. Its projection list requests the conformed dimensions and the other fact, in this case, QUANTITY.

The XSUM indicates that aggregation is occurring for specific groupings.

This derived table retrieves rows from the Time Dimension which are then joined to values from Sales Fact on the DAY_KEY. The final roll up for the report will be done at the month level. The same derived table is also found in the first derived table of this stitch query because Time Dimension is a conformed dimension.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

```
(select
    Order_Method_Dimension.ORDER_METHOD_KEY
    as ORDER_METHOD_KEY,
    Order_Method_Dimension.ORDER_METHOD_EN
    as ORDER_METHOD
from
    go_data_warehouse.GOSLDW.dbo.ORDER_METHOD_DIMENSION
    Order_Method_Dimension) Order_Method_Dimension
```

on (Order_Method_Dimension.ORDER_METHOD_KEY =
Sales_Fact.ORDER_METHOD_KEY)

group by
Time_Dimension.MONTH1,
Order_Method_Dimension.ORDER_METHOD

) D2

on ((D3.MONTH1 = D2.MONTH1) and (D3.ORDER_METHOD =
D2.ORDER_METHOD))

This derived table retrieves rows from the Order Method Dimension which are then joined to values from Sales Fact on the ORDER_METHOD_KEY. The final roll up for the report will also be based on this key. The same derived table is found in the first derived table of this stitch query because Order Method Dimension is a conformed dimension.

The group by clause, in this case, determines how the QUANTITY values will be summed. Notice that REASON_DESCRIPTION is not found in this group by clause. That is because it is not a conformed dimension.

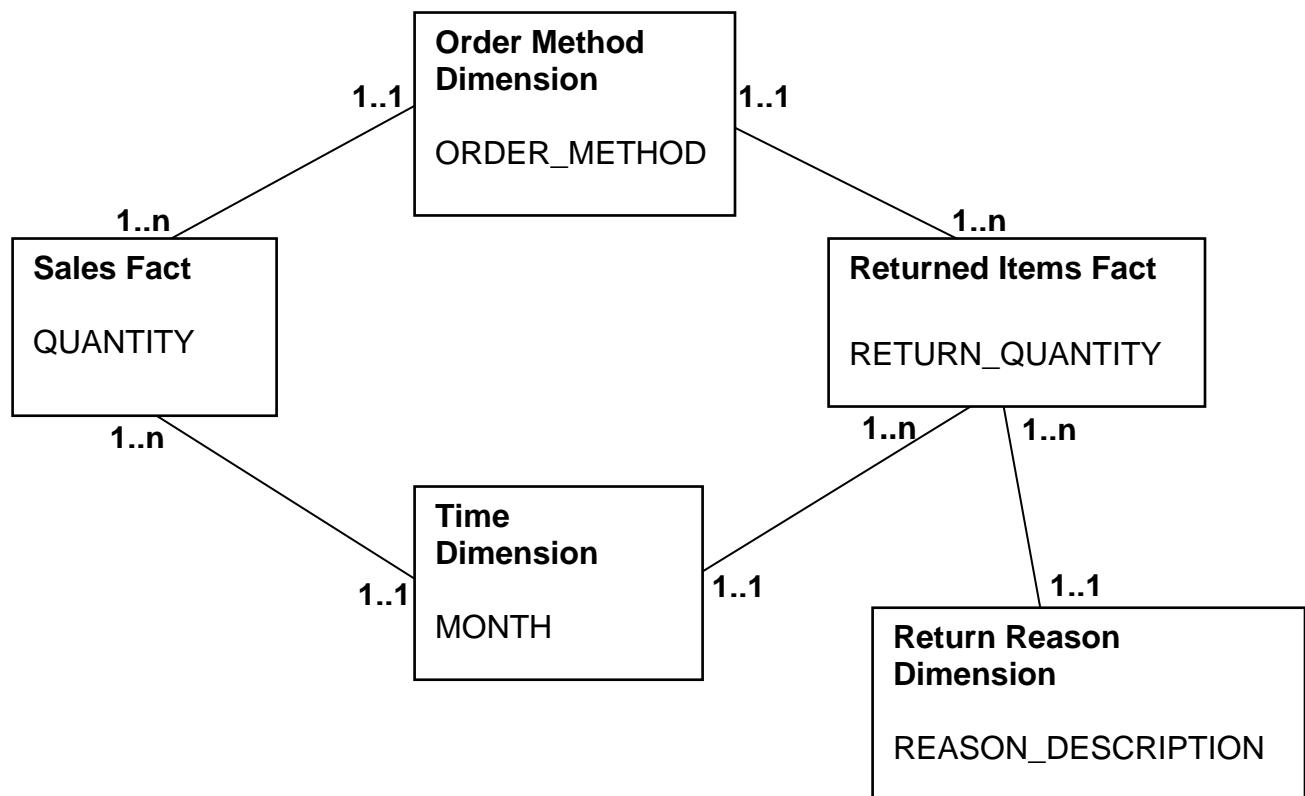
D2 is the alias name for the second derived table in this stitch query.

This portion indicates the join criteria between the two queries of the stitch query. In this case they are joined on columns from the conformed dimensions.

Report Representation Based on Generated SQL

MONTH	ORDER_METHOD	REASON_DESCRIPTION	QUANTITY	RETURN_QUANTITY
MONTH	ORDER METHOD	REASON_DESCRIPTION	QUANTITY	RETURN_QUANTITY
MONTH	ORDER METHOD	REASON_DESCRIPTION	QUANTITY	RETURN_QUANTITY
MONTH	ORDER METHOD	REASON_DESCRIPTION	QUANTITY	RETURN_QUANTITY

Model Diagram Based on Query Subjects Used to Generate SQL



This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Cognos SQL in Report Studio (1 of 2)

- by default IBM Cognos summarizes data

```

select
    TIME_DIMENSION.MONTH_EN as Month1,
    XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN ) as Revenue,
    XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN ) as
    Total_Revenue_
from
    .....
where
    (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
group by
    TIME_DIMENSION.MONTH_EN

```

Automatic Summarization

Automatic Grouping

© 2015 IBM Corporation



Month1 is automatically grouped to roll Revenue and Total Revenue up to each individual instance of a month. This is done at the tabular level of the query. Extra aggregation may be done at the report layout level if summary footers are involved.

If Auto Group & Summarize is turned off in Report Studio, the XSUM does not appear at the tabular level, but will appear at the report level for any summaries in the report layout.

The SQL in the slide is from the Generated SQL/MDX property of the query object, which displays the basic tabular query without any header/footer information. This is similar to the SQL Framework Manager generates when Auto Sum is enabled.

When viewing the generated SQL at the tabular level in Report Studio you are not necessarily looking at the SQL that will be submitted to the database. If the page contains fewer columns than are defined in the query object, the SQL may remove columns from the query at runtime that are not needed. For example, a list may show columns A and B but the tabular query has columns A, B, and C. Since column C is not required in the layout it may be dropped at runtime. You can see this by looking at the generated SQL from the Tools menu.

Business Analytics software

IBM

Cognos SQL in Report Studio (2 of 2)

- why do I see two XSUMs?

```

select
    TIME_DIMENSION.MONTH_EN as Month1,
    XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN ) as Revenue,
    XSUM(XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN ) at
TIME_DIMENSION.MONTH_EN ) as Total_Revenue_
from
    .....
where
    (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
group by
    TIME_DIMENSION.MONTH_EN

```

Footer Summarization



© 2015 IBM Corporation

When summary footers are created in the report, they are represented by another XSUM as seen in the slide example. Total_Revenue_ is aggregated at the tabular level (Auto Group & Summarize is turned on) as seen by the inner XSUM, and then summarized for a summary footer for a particular grouping in the report layout as seen by the outer XSUM.

Typically, you will see a nested XSUM for footers because auto summarization is enabled in the report. If it were not enabled, there would be no nested XSUM.

If rollup processing is done locally on the IBM Cognos servers, you will see RSUM instead of XSUM as seen below:

```
RSUM(XSUM(Sales_Fact.SALE_TOTAL for Time_Dimension.MONTH1 ) at
Time_Dimension.MONTH1 order by Time_Dimension.MONTH1 asc local)
as SALE_TOTAL1
```

The SQL in the slide example is taken from Report Studio's Generated SQL/MDX selection under the Tools menu. This tool allows you to view the SQL including summary requests for footers

Extended vs. Running Aggregates

- Extended aggregate (XSUM, XAVG, XMIN) operations retrieve overall totals.
 - found in generated SQL for queries run in batch mode (PDF, CVS, XML and so on)

```
XSUM(XSUM(Sales_Fact.Revenue for  
TIME_DIMENSION.MONTH_EN ) at  
TIME_DIMENSION.MONTH_EN ) as Total_Revenue_
```

- Running aggregate (RSUM, RAVG, RMIN) operations calculate totals as they are needed.
 - found in generated SQL for queries run in interactive mode (HTML)

```
RSUM(XSUM(Sales_Fact.Revenue for  
TIME_DIMENSION.MONTH_EN ) at  
TIME_DIMENSION.MONTH_EN ) as Total_Revenue_
```

© 2015 IBM Corporation



If the Rollup Processing property in Framework Manager or Report Studio is set to default, IBM Cognos will generate the appropriate SQL based on the report output type. Depending on the selected setting, you can force extended aggregation for interactive reports. For Query Studio, the Framework Manager setting will be used.

In the slide, X represents extended aggregates, while R represents running aggregates. These functions may be calculated locally if we request the processing to be local to improve performance, or if the database vendor does not support the requested construct, such as SQL-OLAP constructs.

Running aggregates are useful when you need to display the first page of an interactive report quickly. For these types of reports, a user may not look at all pages of the report. Therefore, where possible a query should avoid computing information until the last possible moment if possible. However, there are exceptions. For example, totals in headers, or comparing detail rows to group-totals may prevent the use of a running aggregate.

Demo 2: Examine Generated SQL in Report Studio

Purpose:

As a modeler, you need to understand SQL patterns that may be generated in Report Studio to help troubleshoot problems encountered by authors, or to simply answer their questions. By testing the model in Report Studio and viewing the generated SQL for different aggregation scenarios, you can identify these patterns and learn how to identify what a report is requesting.

Components: Framework Manager, Report Studio

Project: GO Operational

Package: GO Operational (query)

Task 1. Author a simple report in Report Studio and view the generated SQL.

1. In the **Project Viewer**, expand **Packages**, and then publish the **GO Operational (query)** package.
2. Log into **IBM Cognos Connection**, and then launch **Report Studio**.
3. Select the **GO Operational (query)** package, click **Create New**, and then double-click **List**.
4. In the **Source** pane, expand **Sales (query)**, and then add the following items to the report:

Query Subject	Query Item
Time	Month
Sales Fact	Revenue

5. To the left of the report, point to the **Query Explorer**  button, and then click **Query 1**.
6. In the **Properties** pane, click the box beside **Generated SQL**, and then click the **ellipsis**.

A warning message appears indicating that you will be viewing a tabular representation of the query. Viewing the generated SQL by this method lets you see the SQL specific to this query. Header and footer summaries are not reflected in this SQL because it is specific to the report.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

- Click **OK**, and then from the list, select **IBM Cognos SQL**.

The results appear as follows:

Generated SQL	Help
<p>IBM Cognos SQL</p> <pre> select TIME_DIMENSION.MONTH_EN as Month1, XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN) as Revenue from GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION, (select ORDER_HEADER.ORDER_DATE as ORDER_DATE, (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue from GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER, GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS where (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)) Sales_Fact where (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE) group by TIME_DIMENSION.MONTH_EN </pre>	

Because the Auto Group & Summarize property is set to Yes (by default), there is an XSUM for Revenue. To see detailed rows of data, set this property to No, which tells IBM Cognos not to apply the sum function.

- Click **Close**.

Task 2. Create a footer and view the generated SQL.

- Click the **Page Explorer**  button, and then click **Page 1**.
- In the report, click the **Revenue** column header, on the toolbar, click  **Summarize**, and then click **Total**.

The report appears as shown below:

Month	Revenue
<Month>	<Revenue>
<Month>	<Revenue>
<Month>	<Revenue>
Overall - Total	<Total(Revenue)>

- Click the **Query Explorer** button, and then click **Query 1**.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

4. In the **Properties** pane, view the **Generated SQL**.
5. Click **OK** to close the warning, and then select **IBM Cognos SQL**.

The results appear as follows:

```

IBM Cognos SQL
select
    TIME_DIMENSION.MONTH_EN as Month1,
    XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN) as Revenue,
    XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN) as Total_Revenue_
from
    GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
    (select
        ORDER_HEADER.ORDER_DATE as ORDER_DATE,
        (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
    from
        GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
        GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
    where
        (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
    ) Sales_Fact
where
    (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
group by
    TIME_DIMENSION.MONTH_EN
  
```

The SQL is similar to that in the previous task, except that the tabular SQL is requesting a second summed Revenue as Total Revenue. This does not mean that two identical requests for Revenue will be sent to the database at run time. This just represents the data items that make up the query at the tabular level. The derived table only requests Revenue once. If you look at the native SQL, you will also see only one request is made for revenue aliased as C1.

6. Click **Close**.
7. From the **Tools** menu, click **Show Generated SQL/MDX**.

Viewing the generated SQL by this method allows you to see the complete SQL statement including footer summary requests.

8. From the list, select **IBM Cognos SQL**.

The results appear as follows:

```
IBM Cognos SQL
select
    TIME_DIMENSION.MONTH_EN as Month1,
    XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN) as Revenue,
    XSUM(XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN) at
        TIME_DIMENSION.MONTH_EN) as Total_Revenue
from
    GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
    (select
        ORDER_HEADER.ORDER_DATE as ORDER_DATE,
        (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
    from
        GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
        GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
    where
        (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
    ) Sales_Fact
where
    (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
group by
    TIME_DIMENSION.MONTH_EN
```

The additional XSUM for the Revenue column populates the report footer. If you change the Rollup processing property to Local, you will see an RSUM instead of an XSUM. This is illustrated in the Optimize and Tune Framework Manager Models module.

If you turn off Auto Group & Summarize, you will not see nested XSUMs, since you would then be aggregating detailed rows for the footer and not summarized values. You will test this in the next steps.

9. Select Native SQL.

Footer summary information is not requested, which indicates that the footer summary information will be processed locally on the IBM Cognos servers. This environment uses a DB2 data source. The request for the footer summary is processed locally because DB2 does not support SQL-OLAP. If the data source were an Oracle database, you would see SQL-OLAP syntax requesting the footer value as shown below:

select

```
"T0"."C0" "MONTH1",      "T0"."C1" "Revenue",
sum("T0"."C1") over ()  "Total_Revenue_"
```

from

```
(.....) "C1"
```

where

```
.....
```

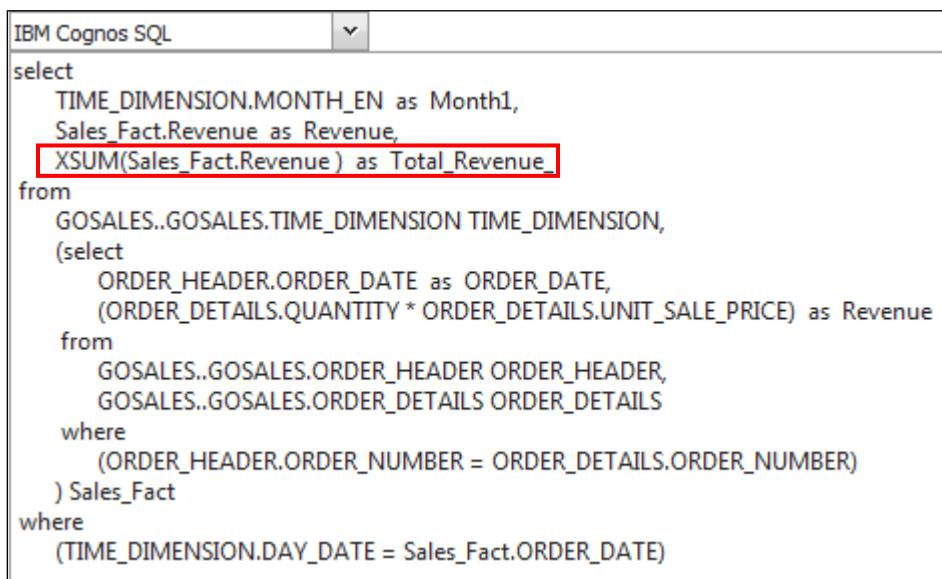
group by

```
.....
```

10. Click Close.

11. In the **Properties** pane, click the cell beside **Auto Group & Summarize**, and set it to **No**.
12. From the **Tools** menu, click **Show Generated SQL/MDX**, and then select **IBM Cognos SQL**.

The results appear as follows:



The screenshot shows the 'IBM Cognos SQL' editor window. The code is as follows:

```
IBM Cognos SQL
select
    TIME_DIMENSION.MONTH_EN as Month1,
    Sales_Fact.Revenue as Revenue,
    XSUM(Sales_Fact.Revenue) as Total_Revenue_
from
    GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
    (select
        ORDER_HEADER.ORDER_DATE as ORDER_DATE,
        (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
    from
        GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
        GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
    where
        (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
    ) Sales_Fact
where
    (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
```

The line `XSUM(Sales_Fact.Revenue) as Total_Revenue_` is highlighted with a red rectangle.

The footer value (Total_Revenue_) is now based on the aggregation of detailed rows, rather than summarized values. In the Native SQL, Revenue is now requested twice from the database: once as detail rows aliased as C1, and again as a summarized row using the sum function aliased as C0. This type of aggregation is supported by the database and is therefore conducted at the database level.

13. Click **Close**, and then close **Report Studio** without saving the report.
14. Leave **IBM Cognos Connection** and **Framework Manager** open for the next demo.

Results:

By testing the model in Report Studio and viewing the generated SQL for different aggregation scenarios, you have identified patterns in the SQL that you can use to understand what a report is requesting.

Generated SQL for DMR Metadata

- regular dimensions can return un-requested columns in Framework Manager
- this occurs when you test levels that have attributes
- it does not occur in the IBM Cognos studios

© 2015 IBM Corporation

When testing regular dimensions in Framework Manager, the generated SQL can include columns that you did not select. This occurs when you test levels that have attributes specified. This does not happen in the IBM Cognos studios, where only the items required for OLAP-style querying are requested.

In Framework Manager, you are always testing all objects in the parent container. For example, if you test a query subject you are in fact requesting all query items in that query subject. When you test a level in a regular dimension, you are requesting the business key, member caption and any attributes.

When you are in the studios and add a level to the report, IBM Cognos will only request the business key and the member caption and parent keys unless you specifically ask for other attributes of the level.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

17-42

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Demo 3: Examine Generated SQL for Dimensionally Modeled Relational Metadata

Purpose:

When testing DMR levels in either Framework Manager or one of the IBM Cognos studios, you should be aware of the SQL generation behavior in either scenario so that you can verify the modeling techniques.

Components: Framework Manager, Report Studio

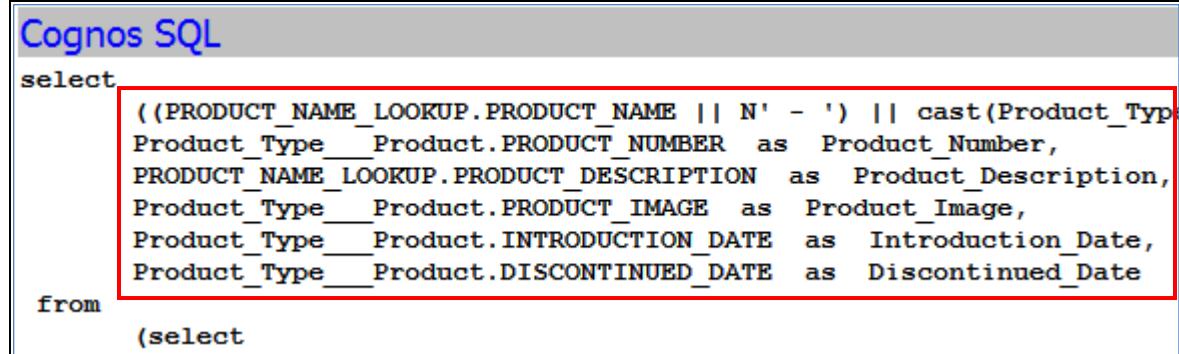
Project: GO Operational

Package: GO Operational (analysis)

Task 1. Test a level in the Products regular dimension in Framework Manager.

1. In **Framework Manager**, in the **Project Viewer**, expand **Dimensional View > Products > Products**.
2. Test the **Product** level (below Product Type), and then click the **Query Information** tab.

The results appear as follows:



```

Cognos SQL

select
    ( (PRODUCT_NAME_LOOKUP.PRODUCT_NAME || N' - ') || cast(Product_Type_
Product_Type__Product.PRODUCT_NUMBER as Product_Number,
PRODUCT_NAME_LOOKUP.PRODUCT_DESCRIPTION as Product_Description,
Product_Type__Product.PRODUCT_IMAGE as Product_Image,
Product_Type__Product.INTRODUCTION_DATE as Introduction_Date,
Product_Type__Product.DISCONTINUED_DATE as Discontinued_Date
from
    (select

```

The projection list is requesting columns other than the business key (PRODUCT_NUMBER) and member caption (PRODUCT_NAME + PRODUCT_NUMBER). PRODUCT_IMAGE, INTRODUCTION_DATE, and DISCONTINUED_DATE are attributes of the Product level, and PRODUCT_DESCRIPTION is the member description.

3. Click **Close**, and then double-click the **Products** regular dimension to open the **Dimension Definition** dialog.

- In the **Hierarchies** pane, click the **Product** level.

The results appear as follows:

The screenshot shows the 'Hierarchies' pane in IBM Cognos Framework Manager. At the top, there is a list of hierarchy levels: Products, Product (All), Product Line, Product Type, and Product. The 'Product' level is currently selected, indicated by a blue highlight. Below this list are several control buttons: 'Add Hierarchy' (with a plus sign icon), 'Add Level' (with a folder icon), 'Delete' (with a minus sign icon), and 'Clear All' (with a clear icon). A checkbox labeled 'Unique Level' is checked. A note below the controls says 'Select a level in the hierarchy control to see the query items.' Below this note is a table containing six rows of data, each representing a query item for the selected level:

Name	Role	Source
Product Caption	_memberCaption	Products.Product ...
Product Number	_businessKey	Products.Codes.P ...
Product Description	_memberDescripti ...	Products.Product ...
Product Image		Products.Product ...
Introduction Date		Products.Introduc ...
Discontinued Date		Products.Disconti ...

This displays the member description and other attributes that were returned when you tested the Product level. In Framework Manager, these columns are returned to show the contents of the level. These columns would not necessarily be returned in the studios unless requested. Only the member caption and business key are returned with the level's parent keys, which you will see in the next step.

- Click **Cancel**, and then publish the **GO Operational (analysis)** package.

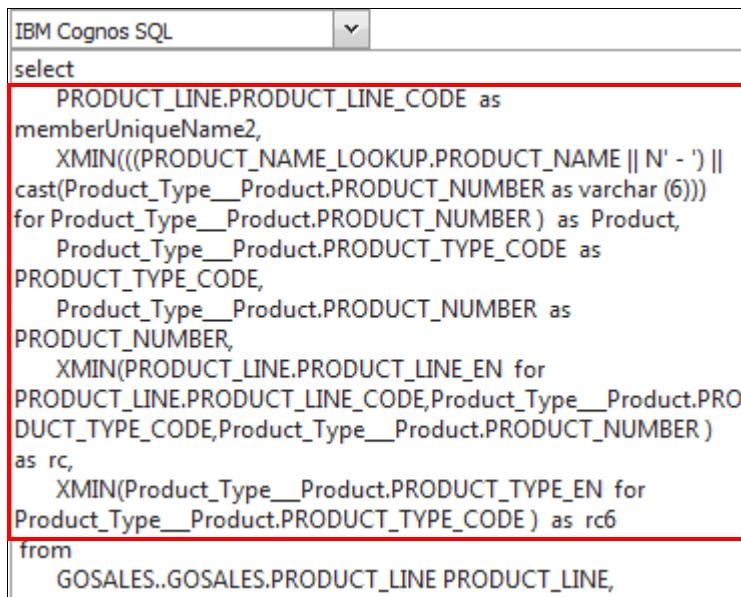
Task 2. Test a level in the Products regular dimension in Framework Manager.

- In **IBM Cognos Connection**, launch **Report Studio**, and select the **GO Operational (analysis)** package.
- Create a new **List** report.
- In the **Source** pane, expand **Sales (analysis) > Products > Products**, and then drag the **Product** level onto the report.
- From the **Tools** menu, click **Show Generated SQL/MDX**.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

5. From the list, click **IBM Cognos SQL**.

The results appear as shown below:



```

IBM Cognos SQL
select
    PRODUCT_LINE.PRODUCT_LINE_CODE as
memberUniqueName2,
    XMIN(((PRODUCT_NAME_LOOKUP.PRODUCT_NAME || N' - ') ||
cast(Product_Type__Product.PRODUCT_NUMBER as varchar (6)))
for Product_Type__Product.PRODUCT_NUMBER) as Product,
    Product_Type__Product.PRODUCT_TYPE_CODE as
PRODUCT_TYPE_CODE,
    Product_Type__Product.PRODUCT_NUMBER as
PRODUCT_NUMBER,
    XMIN(PRODUCT_LINE.PRODUCT_LINE_EN for
PRODUCT_LINE.PRODUCT_LINE_CODE,Product_Type__Product.PRO
DUCT_TYPE_CODE,Product_Type__Product.PRODUCT_NUMBER)
as rc,
    XMIN(Product_Type__Product.PRODUCT_TYPE_EN for
Product_Type__Product.PRODUCT_TYPE_CODE) as rc6
from
    GOSALES..GOSALES.PRODUCT_LINE PRODUCT_LINE,

```

The projection list only requests the member caption (PRODUCT_NAME + PRODUCT_NUMBER), the business key (PRODUCT_NUMBER), and parent business key and member caption columns. The parent information is returned to support drill operations such as drill-up, drill-down, and drill-through.

6. Click **Close**, and then close all browser windows without saving.
7. Close the project without saving, and then close **Framework Manager**, saving if prompted.

Results:

By examining the SQL generation for DMR levels in both Framework Manager and Report Studio, you can see that extraneous columns will not be returned in the studios unless authors request them. Only columns required to support OLAP-style querying will be returned

Business Analytics software

IBM

Using Cross-Product Joins

- combines data by matching each row in one table to each row in another table
- not the same as a full outer join

The diagram shows three tables: Manager, Product, and a Report Set. The Manager table has rows for Philippe and Beatrice. The Product table has rows for 1, 2, and 3. Red arrows point from both Manager and Product to the Report Set table, which contains all possible combinations of Sales Rep and Product.

Sales Rep	Product
Philippe	1
Philippe	2
Philippe	3
Beatrice	1
Beatrice	2
Beatrice	3

© 2015 IBM Corporation

Cross-product joins are also known as a Cartesian product. A cross-product join occurs when there is no relationship between two query subjects. The results and aggregation totals for these types of queries are typically meaningless. By default, Framework Manager does not allow cross joins because they can be resource intensive.

Sometimes this type of join produces meaningless results, but under the right circumstances, it can be very useful. For example, you could use this to combine each product with a pricing table so you can analyze each product at each price. In one table, you have the current price of each product. In a second table, you have a list of increase values such as .05 percent, .06 percent, .10 percent, and so on. In this case, a cross join combines each product price with each increase value.

Cross-Product Join SQL

- selects the requested columns from their respective tables
- no joins exist

```
select  
    Manager.Manager as Manager,  
    Product.Product as Product  
from  
    datasource_name.database_name.schema.Manager Manager  
    datasource_name.database_name.schema.Product Product
```

© 2015 IBM Corporation



Business Analytics software

IBM

Multi-Fact Query Results

- a contiguous result set uses only conformed dimensions

MONTH	PRODUCT_NAME	QUANTITY	EXPECTED_VOLUME
April 2010	Aloe Relief	1,410	1,690
April 2010	Bear Edge	574	529
April 2010	Bear Survival Edge	758	954

- in a correlated list, non-conformed dimensions are present

repeating values

MONTH	PRODUCT_NAME	ORDER_METHOD	QUANTITY	EXPECTED_VOLUME
April 2010	Aloe Relief	Telephone	286	1,690
April 2010	Aloe Relief	Web	854	1,690
April 2010	Aloe Relief	E-mail	270	1,690
April 2010	Bear Edge	Telephone	224	529
April 2010	Bear Edge	Web	60	529

© 2015 IBM Corporation



Interpreting a multi-fact query is not as easy as authoring one. It is important to understand the impact of conformed and non-conformed dimensions on a multi-fact query, and the level of granularity and additive nature of the data. A contiguous result set means the results of each fact query can be mapped to each other with 0..1 to 1..0 precision. A correlated list refers to a looser coupling of the data in which non-conformed dimensions have been introduced.

In the slide, a non-conformed dimension (Order Method Dimension) introduces a difference in granularity. In this case the result is more closely related to a master-detail report. There are dimensions that both queries have in common, and a dimension that is not common to both. This additional level creates a 0..n relationship between the fact queries, resulting in 0..n records in query 2 for any record in query 1.

ORDER_METHOD is related to QUANTITY and therefore further breaks down QUANTITY values into the various order methods used to sell products. Because EXPECTED_VOLUME is not related to ORDER_METHOD, the overall EXPECTED_VOLUME totals for each instance of PRODUCT_NAME are simply repeated across each new ORDER_METHOD instance, but not double counted.

Contiguous Result Sets (1 of 2)

- multi-fact queries with only conformed dimensions can have different levels of granularity

Common Grain from Conformed Dimensions

MONTH	PRODUCT_NAME	QUANTITY	EXPECTED_VOLUME
April 2010	Aloe Relief	1,410	1,690
April 2010	Bear Edge	574	529
April 2010	Bear Survival Edge	758	954

repeating values



Different Levels of Granularity from Conformed Dimension

MONTH	DAY_DATE	PRODUCT_NAME	QUANTITY	EXPECTED_VOLUME
April 2010	Apr 25, 2010	Aloe Relief	286	1,690
April 2010	Apr 27, 2010	Aloe Relief	854	1,690
April 2010	Apr 28, 2010	Aloe Relief	270	1,690
April 2010	Apr 2, 2010	Bear Edge	224	529
April 2010	Apr 4, 2010	Bear Edge	60	529
April 2010	Apr 7, 2010	Bear Edge	166	529

© 2015 IBM Corporation



The top report in the slide example illustrates the same contiguous result set shown in the previous slide. We have only included an element from the Time Dimension at the month level. There is a common grain between the two fact tables because QUANTITY is automatically rolled up to the month level and EXPECTED_VOLUME is a value that begins at the month level.

If we include the DAY_DATE in our report (as seen in the bottom report example), we still have a contiguous result set, but with another level of granularity.

Provided determinants have been specified on the Time Dimension, Cognos will still stitch and aggregate the facts correctly, but you will see repeating values on the higher level of granularity. Although it has the appearance of a correlated result set, it is not. The facts are stitched together by only conformed dimensions (Time Dimension and Product Dimension).

Contiguous Result Sets (2 of 2)

- adding a lower granularity level than the common level between facts causes repeating values for the fact at the higher level
- determinants can prevent double-counting

Multi-Fact/Multi-Grain Report

MONTH	DAY_DATE	PRODUCT_NAME	QUANTITY	EXPECTED_VOLUME
April 2010	Apr 25, 2010	Aloe Relief	286	1,690
April 2010	Apr 27, 2010	Aloe Relief	854	1,690
April 2010	Apr 28, 2010	Aloe Relief	270	1,690
April 2010	Apr 2, 2010	Bear Edge	224	529
April 2010	Apr 4, 2010	Bear Edge	60	529
April 2010	Apr 7, 2010	Bear Edge	166	529
Summary			2,215,354	2,166,005

© 2015 IBM Corporation



With determinants specified, IBM Cognos will not double-count the values at the higher level of granularity.

As seen in the slide example, EXPECTED_VOLUME is not double counted because the determinants specified on the Time Dimension dictate that those values should be aggregated to, and grouped by, the month key.

Troubleshooting Unexpected Results

- When the data results are unexpected, check the SQL:
 - What kind of joins do you see?
 - Do they correspond with the relationships you have in Framework Manager?
 - Do you see a stitch query that you do not expect?
 - Do you see a stitch query without all required conformed dimensions?
 - Do you see a stitch query with no conformed dimensions?
- Do you have determinants specified for multiple levels of granularity in a single query subject?
- Are you using the correct SQL options?
- Check the native SQL to verify if it is expected.

© 2015 IBM Corporation



This slide is a summary of the content learned in this module, and can be used as a checklist when encountering unexpected results or SQL.

Summary

- At the end of this module, you should be able to:
 - governors that affect SQL generation
 - stitch query SQL
 - conformed and non-conformed dimensions in generated SQL
 - multi-fact/multi-grain stitch query SQL
 - variances in Report Studio generated SQL
 - dimensionally modeled relational SQL generation
 - cross join SQL
 - various results sets for multi-fact queries

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

17-52

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



Using Advanced Parameterization Techniques

IBM Cognos BI

Business Analytics software



© 2015 IBM Corporation

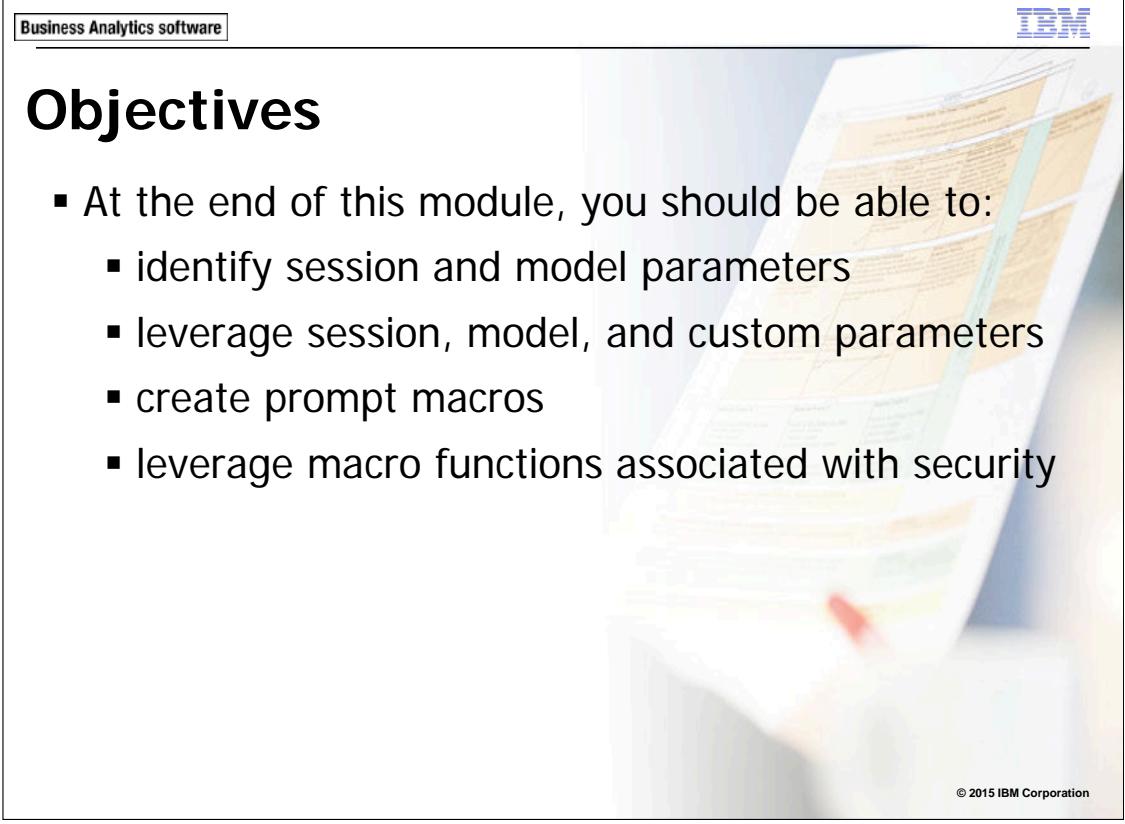
This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Objectives

- At the end of this module, you should be able to:
 - identify session and model parameters
 - leverage session, model, and custom parameters
 - create prompt macros
 - leverage macro functions associated with security



© 2015 IBM Corporation

Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

- User ID: admin
- Password: Education1

Business Analytics software

IBM

IBM Cognos Session Parameters

- a session parameter is a variable associated with a IBM Cognos session
- there are two types of session parameters:
 - environment
 - model

Parameter	Value	Override Value
account.defaultName	Admin Person	
account.personalInfo.email	admin@grtd123.com	
account.personalInfo.givenName	Admin	
account.personalInfo.surname	Person	
account.personalInfo.userName	admin	
current_timestamp	2008-10-08 11:04:25....	
machine	TP-KAMALA	
runLocale	en	

© 2015 IBM Corporation



Each session parameter must have a name and a default value to ensure the session parameter resolves to a value at run time. You cannot have more than one session parameter with the same name. You can set override values only for testing inside the model.

The modeler defines model session parameters, while environment session parameters are defined in the user's environment, such as user ID and locale setting. The amount of environment session parameters IBM Cognos identifies depends on the environment. For example, if a user is on an NT domain, there will be fewer session parameters available than if they were using LDAP as an authentication provider.

A session lasts from the time a user logs on until the time they log off or time out. If you log on anonymously, you will see only runLocale and account.defaultName(Anonymous).

Environment Session Parameters

- environment session parameters are predefined and stored in the content store database

Parameter	Value	Override Value
account.defaultName	Admin Person	
account.personalInfo.email	admin@grtd123.com	
account.personalInfo.givenName	Admin	
account.personalInfo.surname	Person	
account.personalInfo.userName	admin	
current_timestamp	2008-10-13 12:17:27.014-05:00	
machine	TP-KAMALA	
runLocale	en	

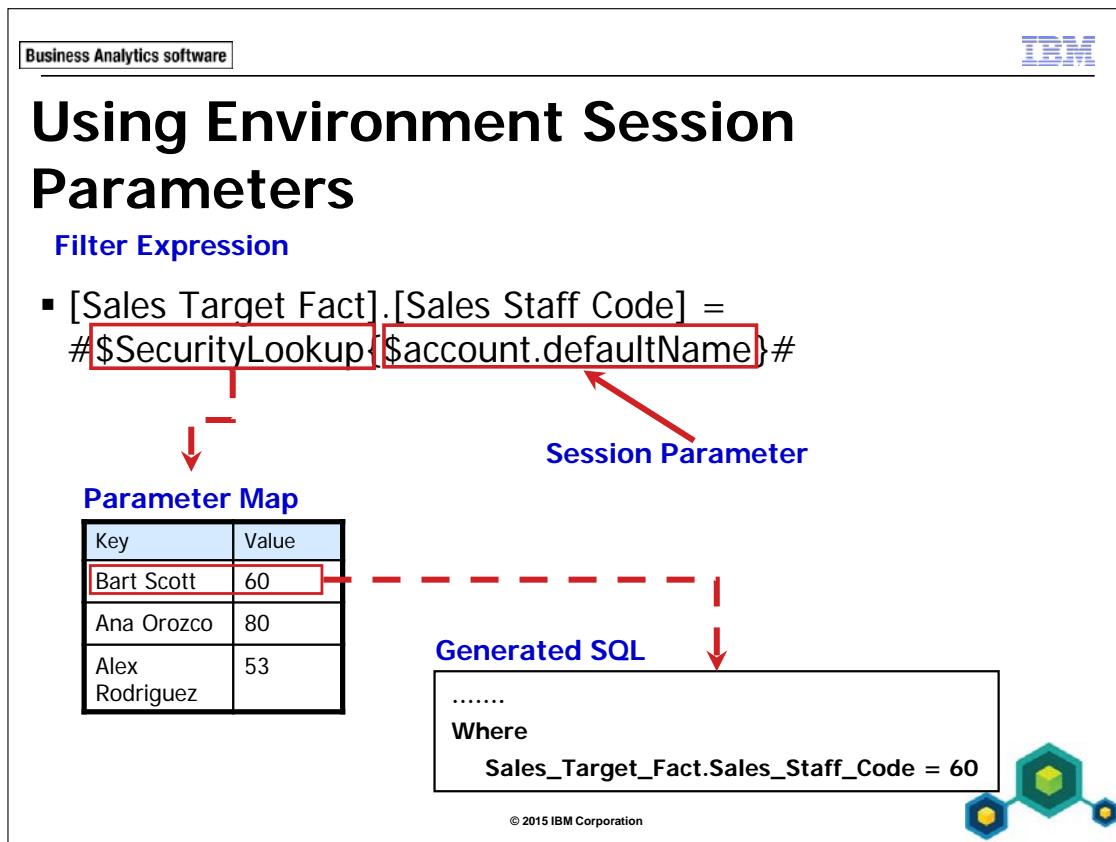
© 2015 IBM Corporation



In the slide example, we see the session parameters available for an LDAP authentication provider. If the LDAP provider supports custom parameters, we can expose them in Framework Manager.

By default, the following session parameters appear in Framework Manager:

- account.defaultName
- account.personalInfo.userName
- runLocale



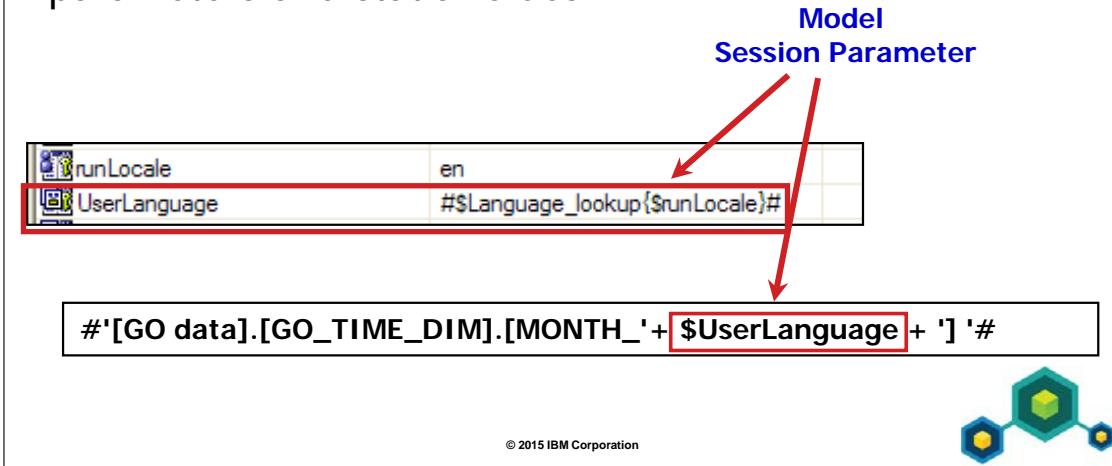
In the slide example, we use a macro to look up the current user in a parameter map to provide an appropriate value for the Where clause of the generate SQL. If Bart Scott was logged on, then the Where clause in the generated SQL would equate to:

Where Sales_fact__employee_secured_.STAFF_KEY = 60

You can use environment session parameters in filters, SQL statements, property settings, and other model objects to create a model with dynamic values.

Model Session Parameters

- Created in Framework Manager and published with every package
- Can include the use of existing environment session parameters and static values



Model session parameters can be useful when you are trying to centralize the maintenance on a macro that is used throughout a model.

For example, a model may have many instances of multilingual data that the modeler must build a macro for. Creating a model session parameter that uses this macro can simplify the SQL for the multilingual query subjects, as seen in the slide example. The modeler simply inserts the UserLanguage session parameter. UserLanguage will be substituted for the value returned by the #Language_lookup{\$runLocale}# macro.

If the macro, or the parameter map it uses, needs to be changed, it can be done in one location, without affecting the rest of the model.

Business Analytics software

IBM

Filtering Data by Package Access

- create model session parameters with static values
- values are published with the package

Static Value Representing a Country Code

The diagram illustrates the relationship between a model session parameter and a filter expression. A red box highlights the 'PackageCountryFilter' parameter in the session parameters table, with a red arrow pointing from it to the filter expression below. The filter expression is enclosed in a red box and contains the code: '[gosales].[SALES_TARGET].[COUNTRY_CODE_RELAYER] = #\$\$PackageCountryFilter#'. The value '6001' is also highlighted in red within the filter expression.

Filter Expression

[gosales].[SALES_TARGET].[COUNTRY_CODE_RELAYER] = #\$\$PackageCountryFilter#

© 2015 IBM Corporation

By using a model session parameter, you can centrally control a value throughout your model. In this case the value is static and controls which country for which a user can see values.

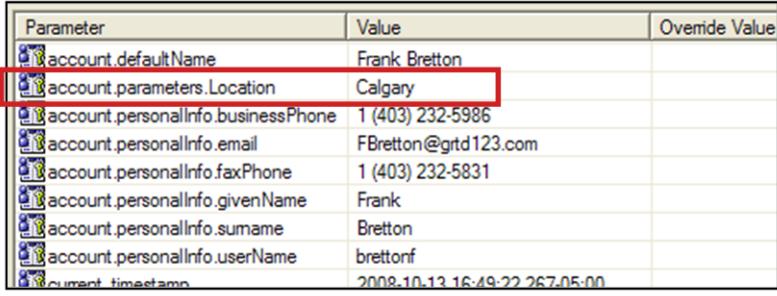
The filter expression in the slide example generates a Where clause in which Country code equates to 6001, which is the value specified in the model session parameter.

Business Analytics software

IBM

Custom Environment Session Parameters

- require a customizable LDAP security provider
- are exposed to IBM Cognos through IBM Cognos Configuration
- appear as session parameters in Framework Manager



Parameter	Value	Override Value
account.defaultName	Frank Bretton	
account.parameters.Location	Calgary	
account.personalInfo.businessPhone	1 (403) 232-5986	
account.personalInfo.email	FBretton@grtd123.com	
account.personalInfo.faxPhone	1 (403) 232-5831	
account.personalInfo.givenName	Frank	
account.personalInfo.surname	Bretton	
account.personalInfo.userName	brettonf	
document_timestamp	2009-10-13 16:49:22.267-05:00	

© 2015 IBM Corporation

Microsoft Active Directory and Sun Java System Server are both supported with respect to custom environment session parameters.

Attributes available from an LDAP security provider are exposed to IBM Cognos as custom properties through IBM Cognos Configuration.

You can use these custom properties in macros as you would any of the other session parameters. You can also use the custom properties inside command blocks that configure Oracle sessions and connections. These command blocks can be used with Oracle lightweight connections and virtual private databases.

Demo 1: Use Custom Environment Session Parameters

Purpose:

Report authors need to filter reports based on location. To support this, you will use a custom property that the Administrator created in the Sample Outdoors Company LDAP authentication provider. You will make this custom property available as a session parameter in Framework Manager, and then use that parameter in a model filter.

Components: **IBM Cognos Configuration, Framework Manager, IBM Cognos Workspace Advanced**

Project: **GO Operational**

Package: **GO Operational (query)**

Task 1. View the customized Location session parameter in Framework Manager.

1. In **Framework Manager**, open the **GO Operational** project located at **C:\Edcognos\B5A52\CBIFM-Start Files\Module 18\GO Operational**.
2. Log on as **brettonf** (password=**Education1**).

Tip: If you are not prompted to log in, click the **Project** menu, click **Logoff**, and then click **Log on again**.

Frank Bretton is a report author that works out of the Calgary office.

3. From the **Project** menu, click **Session Parameters**.

The results appear as follows:

Parameter	Value
account.defaultName	Frank Bretton
account.parameters.Location	Calgary
account.personalInfo.businessPhone	1 (403) 232-5986
account.personalInfo.email	FBretton@grtd123.com
account.personalInfo.givenName	Frank
account.personalInfo.surname	Bretton
account.personalInfo.userName	brettonf
current_timestamp	2015-01-14 10:44:10.986-05:00
machine	VCLASSBASE
model	GO

The current user (Frank Bretton) has an account.parameters.Location parameter set to Calgary, which identifies the city in which he works.

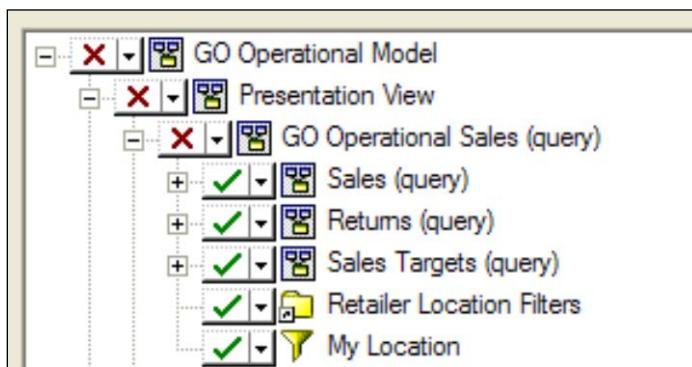
4. Click **Cancel**.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Task 2. Create a model filter that uses the custom parameter.

1. In the **Presentation View**, right-click the **GO Operational Sales (query)** namespace, point to **Create**, click **Filter**.
2. In the **Name** field, type **My Location**, and then in the **Expression definition** pane, enter the following:
[Consolidation View].[Staff by Location].[Staff City] =
 3. Under **Available Components**, click the **Parameters** tab, expand **Macro Functions**, and then double-click **sq** to add it to the expression definition.
 Tip: You can view syntax assistance for macro functions by selecting them and viewing the Tips pane. There you can see explanations of the functions as well as examples.
4. Expand **Session Parameters**, and then double-click **account.parameters.Location** to add it to the expression definition.
 The expression appears as shown below:
[Consolidation View].[Staff by Location].[Staff City] =
#sq(\$account.parameters.Location)#[
5. Click **OK**.
6. In **Packages**, double-click **GO Operational (query)**, and then select **My Location** to add it to the package.

The results appear as follows:



7. Click **OK**, and then **Publish** the **GO Operational (query)** package without verifying the package.
8. **Save** the project.

Task 3. Test model filter in IBM Cognos Workspace Advanced.

1. In **IBM Cognos Connection**, log on as **brettonf** (password = **Education1**).
2. Launch **IBM Cognos Workspace Advanced**, and select the **GO Operational (query)** package.
3. Click **Create New**, and then double-click **List**.
4. In the **Source** pane, expand **Sales (query)**, and then add the following items to the report:

Query Subject	Query Item
Staff by Location	Staff City
Sales Fact	Revenue

All cities and their revenue appear.

5. From the **Source** pane, drag the **My Location** filter onto any column in the report.
6. Click **OK**.

The results appear as follows:

Staff City	Revenue
Calgary	\$111,146,739.19

Because you are logged in as Frank Bretton, the report is filtered by his city, Calgary, using the custom environment session parameter.

7. Close **IBM Cognos Workspace Advanced**, without saving the report, leaving Framework Manager and IBM Cognos Connection open for the next demo.

Results:

By leveraging a custom property in the Sample Outdoors Company LDAP authentication provider, you allow report authors to create reports whose data can be filtered by the user's location.

Using Session Parameters

- session parameters can be used in:
 - object properties
 - SQL
 - filters and calculations
 - stored procedure arguments
 - model session parameter value

© 2015 IBM Corporation



Session parameters can be used in macros to dynamically change many model elements, including:

- object properties such as Content manager datasource, Catalog, and Schema properties in data source connections
- SQL in a data source query subject
- filter and calculation syntax

Although you can add macros to the SQL of a data source query subject, it is recommended to do this in the model query subject layer as specified in the modeling recommendations.

Demo 2: Dynamically Change the Data Source Connection

Purpose:

The Sample Outdoors Company has different databases with the same structure but different data that serve different regions of the business. You need to support the dynamic selection of a database, based on the current user. You will do this by using a macro and session parameter in order to change the Content Manager Datasource connection name at run time.

Components: Framework Manager, IBM Cognos Connection

Project: GO Operational

Task 1. Create a data source lookup parameter map.

1. In Framework Manager, right-click **Parameter Maps**, and then click **Create > Parameter Map**.
2. In the **Name** box, type **DataSourceLookup**, and click **Next**.
3. In **Default value**, type **GOSALES**.
4. Under **Key**, in the first row, type **brettonf**, and then under **Value** type **GOSALES**.
5. Under **Key**, in the second row, type **scottb**, and then under **Value**, type **GOSALES2**.

The results appear as follows:

Key	Value
brettonf	GOSALES
scottb	GOSALES2

6. Click **Finish**.

Task 2. Add a macro to the data source properties.

1. In the **Project Viewer**, expand **Data Sources**, and click **GOSALES**.
 2. In the **Properties** pane, click the **Content Manager Data Source** field, and then click the **ellipsis**.
- If the Properties pane is not open, click **View > Properties**.
3. In **Value**, delete the **GOSALES** text, and then click the **Insert Macro** button.
 4. In **Available components**, expand **Parameter Maps**, and then double-click **DataSourceLookup** to add it to the definition.

5. In Available components, expand **Session Parameters**, and then drag **account.personalInfo.userName** inside the brackets of the **DataSourceLookup** parameter map.

The results appear as follows:

```
#$DataSourceLookup{$account.personalInfo.userName}#
```

6. Click **OK**.

The Value pane displays the new macro.

7. Click **OK**.

Task 3. Test the data source property macro.

You are currently logged on to Framework Manager as Frank Bretton. You will test this account first.

1. Expand **Consolidation View > Return Reason**, and **Test the Return Reason Description** query item.
2. Click the **Query Information** tab.

The results appear as follows:

```
Cognos SQL
select
    RETURN_REASON.REASON_DESCRIPTION_EN as Return_Reason_Description
from
    GOSALES . GOSALES.RETURN_REASON RETURN_REASON
```

Notice the data source connection name. You will now log on as a different user to dynamically use another data source connection.

3. Click **Close**.
4. Click **Project > Log off**, and then log on again with **scottb** (password = **Education1**).
5. Test **Return Reason Description** again.

The results appear as follows:

```
Cognos SQL
select
    RETURN_REASON.REASON_DESCRIPTION_EN as Return_Reason_Description
from
    GOSALES2 . GOSALES.RETURN_REASON RETURN_REASON
```

Notice the data source connection name has changed. For this technique to work, the underlying database structures must be mirror images with the exception of the data they contain.

Note: If you get an error message, launch IBM Cognos Administration, and verify that the GOSALES2 data source connection is defined as follows:

- Name: GOSALES2
- Type: IBM DB2 (deselect Configure JDBC connection)
- Database name: GS_DB
- User ID: GOSALES
- Password: Education1

6. Click **Close**.

7. From the **Project** menu, click **Log off**, and then **Log on again**, to log in as **admin** (password = **Education1**).
8. Test the **Return Reason Description** query item again, and then click the **Query Information** tab.

The query uses the default parameter map value (GOSALES), since the admin user is not a value in the parameter map.

9. **Save** the project and leave it open for the next demo.

Note: You can also set up a dynamic data source connection when you configure your data source connections. For example, you can create one Data Source, and then create multiple connections for that data source. Each connection points to a different version of the database, which contains region specific data. You would then apply security on each of the connections. When a user accesses the data source, they will automatically use the connection to which they have access. If a user has access to more than one connection, they will be prompted at run time to select the connection they want to use. To support multiple data source connections in your Framework Manager project, you must clear the data source catalog and schema properties. This information is obtained from the connection information at run time.

Results:

Using a macro and a session parameter, you were able to dynamically change your data source connection based on the current user.

Using Prompt Macros

- prompt macros allow you to:
 - provide prompt names
 - specify data types
 - provide default values
 - use Prompt Info properties
 - append text
 - make filters and prompts optional
 - request single or multiple values

© 2015 IBM Corporation



The token data type is used for passing SQL syntax.

You can reference query items allowing the use of Prompt Info properties setting.

You can append text at the beginning or end of the prompt.

Specify a default value to make the prompt optional. If no default value is specified the prompt is mandatory.

You can use either the prompt function for single values, or the promptmany function for multiple values.

Business Analytics software

IBM

Using the Prompt Macro Function

```
#prompt('Select language of preference', 'token', 'EN')#
```

The diagram illustrates the components of the prompt macro:

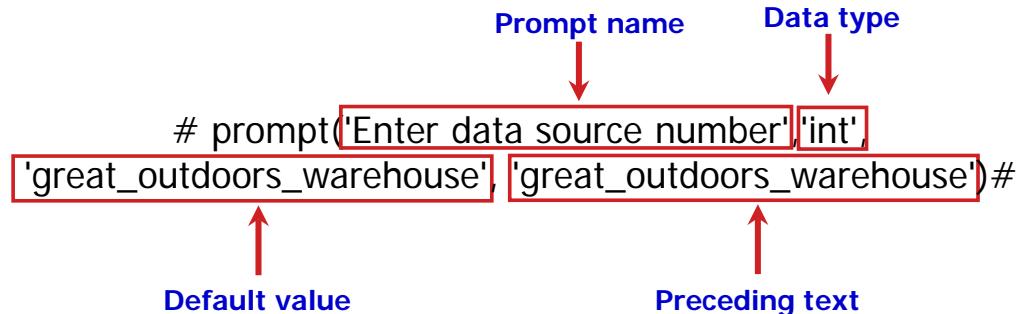
- Prompt name**: Points to the string 'Select language of preference'.
- Default value**: Points to the string 'EN'.
- Data type**: Points to the string 'token'.

© 2015 IBM Corporation

The prompt macro above allows the end user to enter a single value, in this case a token value, to dynamically affect the SQL generated for selecting a language based column.

For example, select rows from the MONTH_FR column if the FR parameter is supplied by the user. If no value is supplied, the default EN value will be used, which will return rows from the MONTH_EN column.

Using Preceding Text in a Prompt



© 2015 IBM Corporation



This prompt macro uses the prompt function with preceding text to allow end users to select the data source to which they would like to connect.

The prompt macro uses preceding text to specify the data source name which is concatenated to a value supplied by the user. If the user supplies a value of 2, it will result in a data source name of GOSALES2.

Demo 3: Create Prompt Macros to Filter Data

Purpose:

Report users want to choose which data source they will connect to, rather than connecting to a data source based on their user ID. You will use a prompt function in the data source properties to support this requirement. As well, Report users want to be prompted for the reason why items are returned, and be able to supply one or more values for the Return Reason code.

Component: **Framework Manager**

Project: **GO Operational**

Task 1. Apply a prompt macro in the data source properties.

1. Expand **Data Sources**, and select the **GOSALES** data source.
2. In the **Properties** pane, for the **Content Manager Data Source** property, change the macro to the following:

#prompt('Enter data source number', 'int', 'GOSALES', 'GOSALES')#

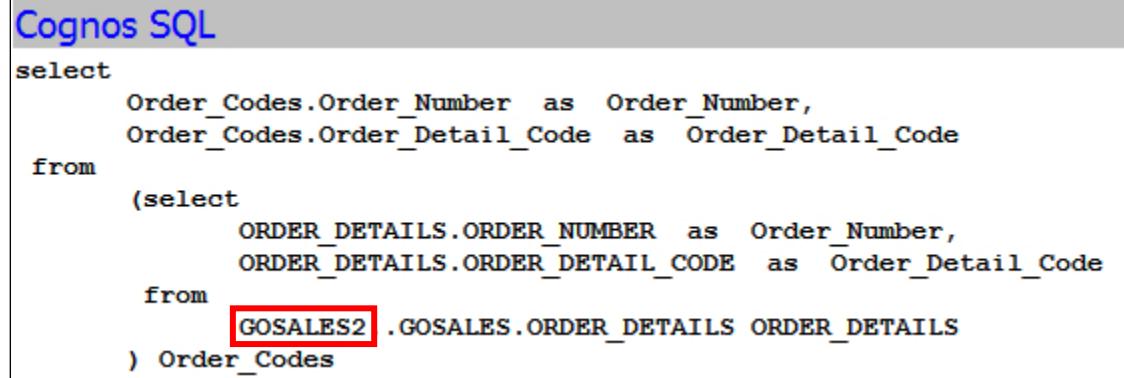
This prompt syntax requests a number value from the user and appends it to the preceding text specified, in this case, GOSALES. If no value is specified, then the default value of GOSALES will be used.

3. In the **Consolidation View**, test the **Order Codes** query subject, setting the prompt value to **2**, and ensuring that **Always prompt for values when testing** is selected.

Note: If you are not prompted as expected, in the Test Results window, click Options. In the Model Prompts Manager, you can modify the prompt as instructed.

- Click **OK**, and then click the **Query Information** tab.

The results appear as follows:



```

Cognos SQL

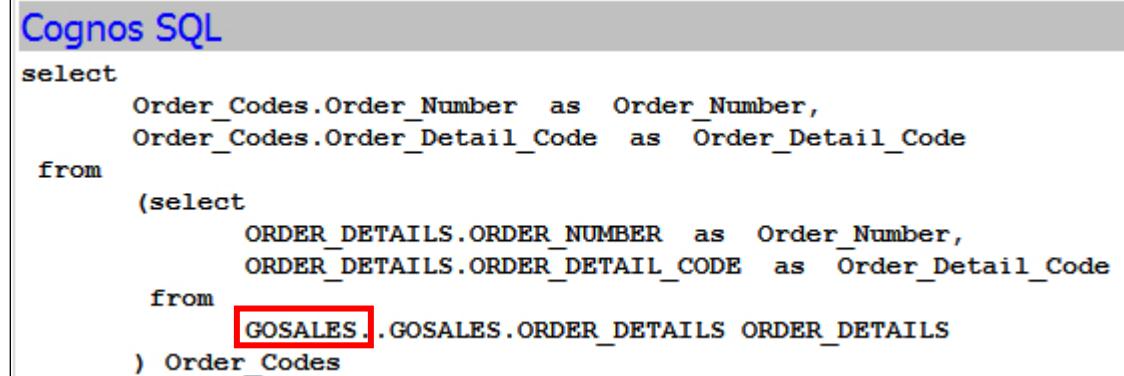
select
    Order_Codes.Order_Number as Order_Number,
    Order_Codes.Order_Detail_Code as Order_Detail_Code
from
    (select
        ORDER_DETAILS.ORDER_NUMBER as Order_Number,
        ORDER_DETAILS.ORDER_DETAIL_CODE as Order_Detail_Code
    from
        GOSALES2..GOSALES.ORDER_DETAILS ORDER_DETAILS
) Order_Codes

```

Notice the data source name is GOSALES2. The value of 2 provided in the prompt was appended to the preceding text GOSALES. This query has gone against the GOSALES2 data source.

- Click the **Test** tab, and click **Test Sample**, leaving the prompt value blank, if prompted.
- Click the **Query Information** tab.

The results appear as follows:



```

Cognos SQL

select
    Order_Codes.Order_Number as Order_Number,
    Order_Codes.Order_Detail_Code as Order_Detail_Code
from
    (select
        ORDER_DETAILS.ORDER_NUMBER as Order_Number,
        ORDER_DETAILS.ORDER_DETAIL_CODE as Order_Detail_Code
    from
        GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
) Order_Codes

```

Notice the data source name is now GOSALES. The default value of GOSALES is used. This query has gone against the GOSALES data source. If you provided a value of anything other than 2 at this point it would return an error since there is no data source configured other than GOSALES and GOSALES2.

- Click **Close**.

You can achieve the same effect in IBM Cognos Connection by giving a user access to more than one connection associated with a Data Source defined. If a user has permissions to access more than one connection for a data source, they will be prompted at run time to select a connection.

Task 2. Apply a filter to the Return Reason Dimension query subject.

1. In **Consolidation View**, double-click the **Return Reason** model query subject to open the **Query Subject Definition** dialog box.
2. Click the **Filters** tab, and **Add** a filter named **Prompt Many for Return Reason** with the following expression:
[Consolidation View].[Return Reason].[Return Reason Code] in
(#promptmany('Enter Return Reason Codes(s)', 'integer')#)
3. Click **OK**, click the **Test** tab, and then click **Test Sample**.
 You are prompted to enter values.
4. Click the **Value** field for **Enter Return Reason Code(s)**.
 A multi-value prompt dialog box appears:

5. In the **Provide a value** box, type **1**, and then click the right arrow  to add the value to the **Choices** list.
6. Repeat to add the values **3** and **5**.
7. Click **OK**, click **OK** again.

The results appear as follows:

Return Reason Description	Return Reason Code
Defective product	1
Wrong product ordered	3
Unsatisfactory product	5

Only the values you typed are returned. This is a mandatory prompt since no default values have been provided. If you do not provide a value, you cannot proceed beyond the prompt.

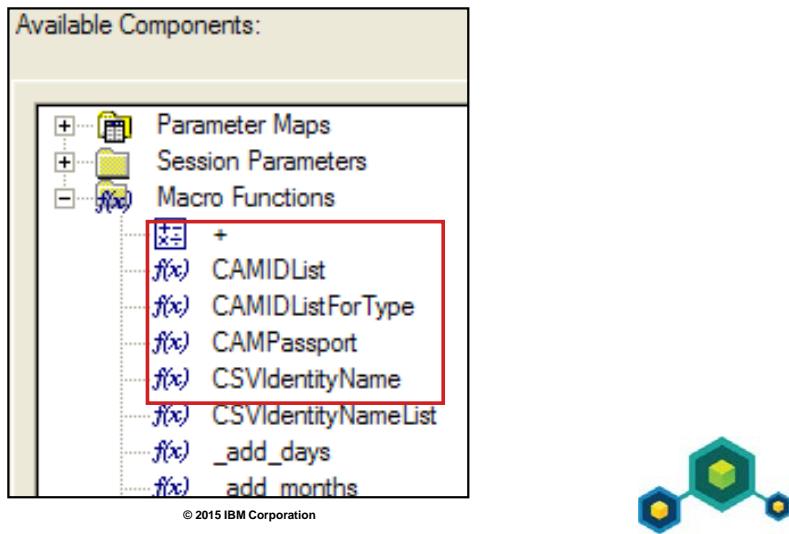
8. Click **OK**, and then save the project.

Results:

You created prompt macros that allow report users to select the data source they connect to, and to enter one or more Return Reason codes.

Security Macro Functions

- there are five macro functions associated with security

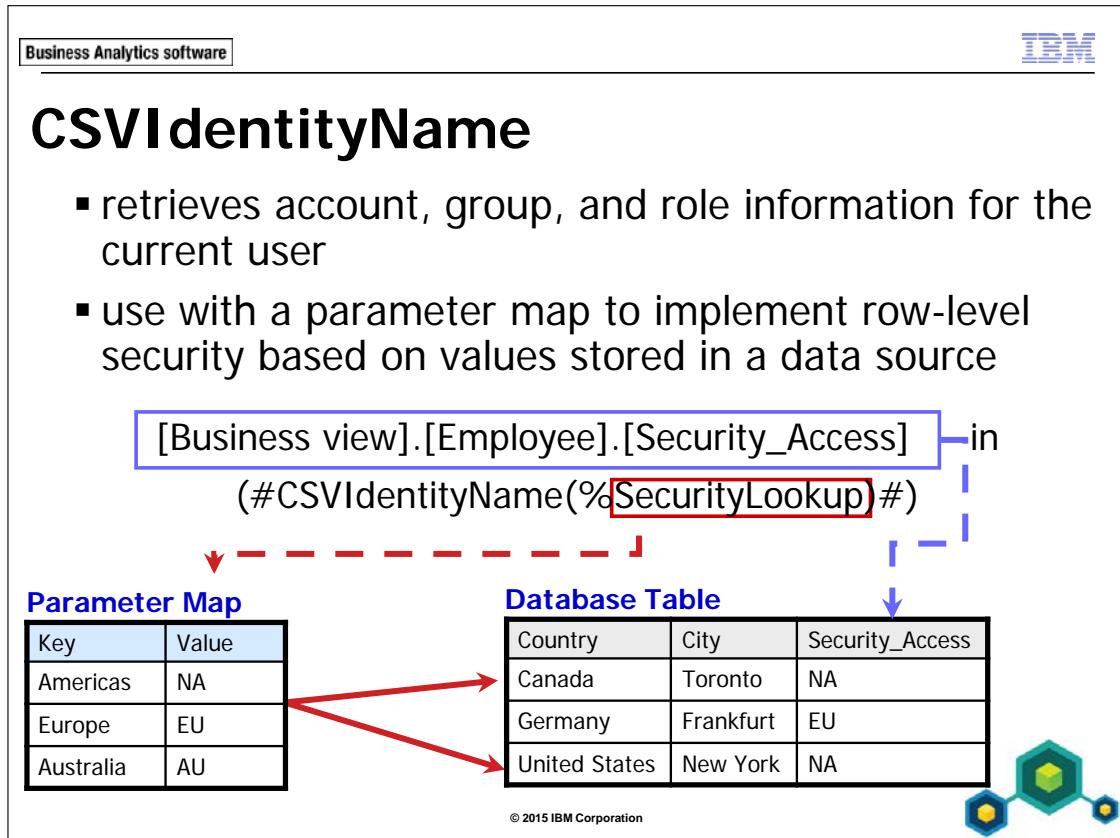


The macro functions above retrieve security related information from a user's session and can be used in a macro to filter data or display user information in a calculation.

CAMIDList, CSVIdentityName, and CSVIdentityNameList are used to retrieve account, group and role information. This module will focus on CSVIdentityName and CSVIdentityNameList.

CAMIDList and CSVIdentityNameList, both return information related to the authentication for the current user(s); however, CAMIDList returns complex information, such as namespace and user ID,s which requires parsing to obtain the information you wish to retrieve for the reporting environment.

CAMPassport returns the current user's CAM passport. This can be useful when implementing single sign-on from a report to a custom IBM Cognos SDK application. CAMIDListForType returns an array of the user's identities, based on the identity type (account, group, or role).



You can use the CSVIdentityName macro function as a key in a parameter map.

In the slide example, the parameter map is called SecurityLookup. If the user belongs to the security group or role by the name of Americas, then they will have access to data from Canada and the United States. If they also belonged to Europe and Australia, then they would see data from these locations as well.

Each value in the comma separated value list for the current user is passed to the parameter map for evaluation and if a match is found for the key, it is substituted with the corresponding value. If the user did belong to the Americas, Europe, and Australia groups, then the resulting filter would look like this:

`[Business view].[Employee].[Security_Access] in ('NA','EU','AU').`

CSVIdentityNameList

- returns a separator delimited list of the account, group, and role information for the current user

[Business view].[Retailer site].[City] in
(#CSVIdentityNameList()#)

Database Table

Country	City
Canada	Toronto
Germany	Frankfurt
France	Lyon

Resulting Filter

[Business view].[Retailer site].[City] in
('Everyone', 'Authors', 'Lyon')



© 2015 IBM Corporation

You can use the CSVIdentityNameList macro function to filter on data that has the same name as the account, group or role information returned in the delimited list.

In the slide example, the city column will be filtered on Lyon.

Demo 4: Leverage a Macro Function Associated with Security

Purpose:

Report authors need to be able to filter report results based on retailers located in the users city. The IT department has set up an LDAP authentication provider with groups based on city names and has added the appropriate users to each group. With this knowledge, you can leverage the group names from the authentication provider in your filter through the CSVIdentityNameList macro function.
Report consumers also need to obtain their personal IBM Cognos security information. To accomplish this, you will create a calculation that will return this information for them.

Components: Framework Manager, IBM Cognos Workspace Advanced

Project: GO Operational

Package: GO Operational (query)

Task 1. Create a filter using the CSVIdentityNameList macro function.

1. In the **Presentation View** namespace, right-click the **GO Operational Sales (query)** namespace, and then create a new filter called **Retailers Near Me**.
2. In the **Available Components** pane, expand **Consolidation View > Retailer by Location**, and then add **Retailer City** to the expression definition.
3. In the **Expression definition** pane, after **[Consolidation View].[Retailer by Location].[Retailer City]**, type **in (**.
4. Under **Available Components**, click the **Parameters** tab, and then expand **Macro Functions**.
5. Double-click **CSVIdentityNameList** to add it to the expression definition, and then type **)** to finish the expression.

The results appear as follows:

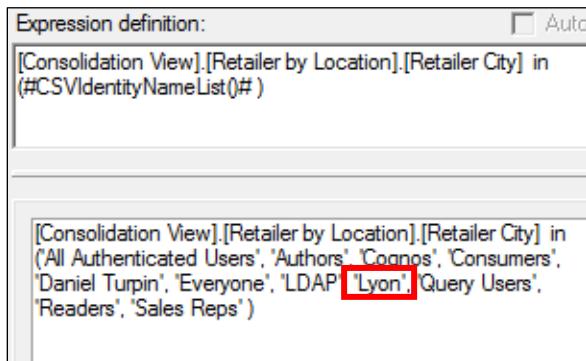
[Consolidation View].[Retailer by Location].[Retailer City] in (#CSVIdentityNameList()#)

Notice the comma-separated list that appears in the Tips pane. This list is alphabetical and indicates the user's name and all the groups and roles to which they belong. The test user you will use is Daniel Turpin who belongs to a group called Lyon named after the city where he works.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

6. Click **OK**.
7. Click **Project > Log off**, and then log on again as **turpind** (password **Education1**).
8. Double-click the **Retailers Near Me** filter.

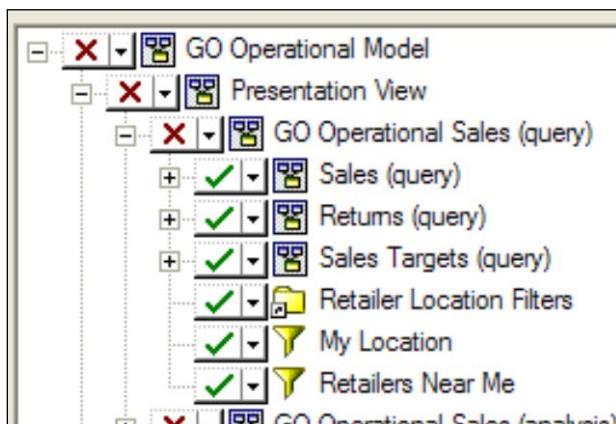
The results appear as follows:



Notice the group called Lyon.

9. Click **OK**.
10. **Log off**, and then **log on again** as admin (password = **Education1**).
11. In **Packages**, double-click **GO Operational (query)**, and then select **Retailers Near Me**.

The results appear as follows:



12. Click **OK**, and **Publish** the **GO Operational (query)** package, deselecting **Verify the package before publishing**.
13. **Save** the project.

Task 2. Test the Retailers Near Me filter.

1. In **IBM Cognos Connection**, log on as **turpind** (password = **Education1**).
2. Launch **IBM Cognos Workspace Advanced**, select the **GO Operational (query)** package, and create a **List** report.
3. Expand **Sales (query)**, and then add the following items to the report:

Query Subject	Query Item
Retailer by Location	Retailer City Retailer Name
Sales Fact	Revenue

4. Click the **Retailer City** column, and then click **Group / Ungroup**.
5. Drag the **Retailers Near Me** filter onto the report, and click **OK** to dismiss the message.

The results appear as follows:

Retailer City	Retailer Name	Revenue
Lyon	Air marin	\$8,248,477.50
	Amis de montagne	\$3,689,130.62
	Amisport	\$5,395,281.79
	Camping Sauvage	\$4,704,758.74
	Conception française	\$17,175,057.69
	Cordages Discount	\$3,119,130.38
	Galerie Sport	\$8,154,265.75
	Golf Plaza	\$5,745,840.93
	Golf Plus	\$4,270,750.10
	Jeunesse active	\$4,962,985.15

The report is now filtered on Lyon since that is the group to which Daniel Turpin belongs.

6. Close **IBM Cognos Workspace Advanced**, without saving the query.

Task 3. Create a calculation to retrieve a user's IBM Cognos security information.

1. In the **Presentation View**, right-click the **GO Operational Sales (query)** namespace, point to **Create**, and then click **Calculation**.
2. In the **Name** box, type **My Security Info**.

You will use the sq (single quote) function, the csv (comma separated values) function, and the CAMIDListForType function to return the user's roles. The sq function places the entire results in single quotes making it a string. The csv function takes the results of the CAMIDListForType function and separates the values with a comma. The CAMIDListForType retrieves specific security information for the user based on a parameter. You can request the user's roles, groups, or account.

3. Use the **Parameters** tab to create the following expression:

#sq(csv(CAMIDListForType('role')))#

In this expression, role has been hard coded into the CAMIDListForType function to return the user's roles.

4. Click the **Test Sample**  button.

The results are similar to the following:

Test Results
My Security Info
'CAMID("::System Administrators")','CAMID("::Consumers")','CAMID("::Readers")'

You are logged on as the admin user, which belongs to the System Administrators, Consumers, and Readers roles. You can change the role value in the expression to Group or Account to retrieve different security information.

5. Click **OK**, and then **save** the project.

Task 4. Test the security information calculation in IBM Cognos Workspace Advanced.

1. In **Packages**, double-click **GO Operational (query)**, and select the **My Security Info** calculation.
2. Click **OK**, and then **Publish** the package without verifying the package.
3. In **IBM Cognos Connection**, launch **IBM Cognos Workspace Advanced**.

4. Select the **GO Operational (query)** package, and create a **List** report.
Note: At this point, you are still logged on as turpind in IBM Cognos Connection.
5. Drag the **My Security Info** calculation to the work area.

The results appear as follows:

My Security Info
'CAMID(":Authors")','CAMID(":Query Users")','CAMID(":Consumers")','CAMID(":Readers")'

This user belongs to the Authors, Query Users, Consumers, and Readers role.

6. Close **IBM Cognos Workspace Advanced** without saving the query.

Results:

Using the CSVIdentityNameList macro function, you created a model filter that lets authors filter reports based on the retailers in the same city as the report consumer. You also created a calculation to return users security information.

Workshop 1: Make the Security Information Calculation Dynamic

Component: **Framework Manager, IBM Cognos Workspace Advanced**

Project: **GO Operational**

Package: **GO Operational (query)**

Currently the **My Security Info** calculation is hard coded to display the roles of the current user. Users would like the flexibility to show different information, such as their account info, or groups they belong to.

To support this, you will:

- Add a prompt macro to the calculation expression.

Ensure that the prompt name indicates the type of values to be entered into the prompt (role, group, account), and that you specify the correct data type (the values submitted by the user will be used in a SQL query to the Content Store database).

Tip: You will need to use the Token data type in your prompt macro.

- Edit the **My Security Info** calculation, test it in **Framework Manager**, and then test in **Cognos Workspace Advanced** as **turpind** (password = **Education1**).
- **Save** and **Close** the project.

For more information about where to work and the workshop results, refer to the Tasks and Results section that follows. If you need more information to complete a task, refer to earlier demos for detailed steps.

Workshop 1: Tasks and Results

Task 1. Add a prompt macro function to the My Security Info calculation.

- In **Presentation View > GO Operational Sales (query)**, edit the definition for the **My Security Info** calculation as follows:

```
#sq(csv(CAMIDListForType(prompt ('Specify info type (role, group, account)', 'token'))))#
```

- Test** the calculation three times, using a different prompt value each time (**role**, **group**, **account**).

You will see the following prompt when testing the calculation in Framework Manager:

Enter prompt values:		
Name	Data Type	Value
* Specify info type (role, group, account)	String (Abc)	group

- Save** the project, and then **Publish** the **GO Operational (query)** package without verifying the package.

Task 2. Test the My Security Info Calculation in IBM Cognos Workspace Advanced.

- In **IBM Cognos Connection**, logged in as **turpind** (password=**Education1**), launch **IBM Cognos Workspace Advanced**, and select the **GO Operational (query)** package for a **List** report.
- Drag the **My Security Info** calculation onto the report, and type **group** in the Prompt value.

The results appear as follows:

My Security Info
'CAMID("LDAP_ID:g:1a2527d4-1901-4a51-9a70-ea6ae394336e")','CAMID("::Everyone")','CAMID("::All Authenticated Users")','CAMID("::Sales Reps")'

- Close** all browser windows without saving the report and Framework Manager, saving the project if prompted.

Summary

- You should now be able to:
 - identify session and model parameters
 - leverage session, model, and custom parameters
 - create prompt macros
 - leverage macro functions associated with security

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



Model Maintenance and Extensibility

IBM Cognos BI

Business Analytics software



© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

19-2

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

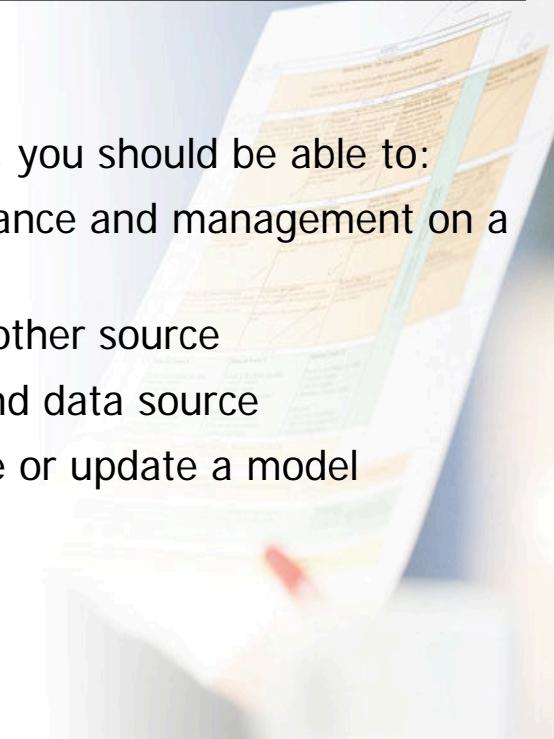
Business Analytics software

IBM

Objectives

- At the end of this module, you should be able to:
 - perform basic maintenance and management on a model
 - remap metadata to another source
 - import and link a second data source
 - run scripts to automate or update a model
 - create a model report

© 2015 IBM Corporation



Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

- User ID: admin
- Password: Education1

Managing Projects

- Copying, Moving, Renaming, and Deleting projects:
 - are available from the File>Manage Projects menu
 - allow you to organize projects in a meaningful way
 - affect the file folder and all of the files it contains (except Rename)
 - require closing the project first (except Copy)

© 2015 IBM Corporation



The .cpf file acts as a pointer to the files that comprise the project.

Rename Project renames the .cpf file but no other files in the folder. If the folder has the same name, it is also renamed. Of course these same actions can be performed manually on the system files in Windows Explorer.

If the project has segments, Copy, Move, and Delete work on all segments and the main project. However deleting a project segment does not delete the main project or other segments. Segments are discussed in more detail in another module.

Business Analytics software

IBM

Deploying Packages and Content

- IBM Cognos Configuration: switch URI from one server to another
- IBM Cognos Connection: export selected content

© 2015 IBM Corporation

Recommended

A simple way to deploy a package to a different server is to change the Dispatcher URI setting (in the Environment>Group properties) in IBM Cognos Configuration. However, this only allows you to deploy a package to a different environment, not content (reports, analysis, and so on) based on that package. For example, reports created in the development environment will not be deployed to the production environment using this method.

To deploy both packages and their related content, an administrator can create an Export Deployment Specification, which can be imported to other IBM Cognos servers (such as multiple production sites). You can also create an export deployment of the entire Content Store. This method is easier because you do not have to reconfigure the IBM Cognos environment each time you want to publish a package to a different server. It is also safer, as it prevents accidentally publishing a package to the wrong server by forgetting to change the Dispatcher URI.

Analyze Publish Impact

- Framework Manager feature that analyzes the effects of model changes on a published package
- identifies:
 - changed model objects
 - other model objects affected by the changes
 - reports that use the changed objects

© 2015 IBM Corporation



For each object identified as a changed model item, an icon identifies whether the change may or will break any related reports. You can then identify specific reports affected by the changes, and even directly run those reports from the Analyze Change Impact dialog box.

With this information, you can notify report authors that a change was made to the model that affects their reports. They can then fix their reports so that consumers are not affected.

Something to be aware of is that changed cardinality is not detected and identified by this tool since changed cardinality does not break reports. It may, however, change the amount of data returned (outer join vs. inner join), and therefore should be discussed with authors.

Demo 1: Perform Impact Analysis on a Modified Package

Purpose:

A number of reports have been created using the GO Operational model. You need to make some changes to the model in Framework Manager and re-publish the package. Before publishing, you want to analyze what effect this change will have on existing reports. To accomplish this you will perform an impact analysis on the GO Operational package.

Component: Framework Manager

Project: GO Operational

Task 1. Modify a model object.

1. In Framework Manager, open the **GO Operational** project located at **C:\Edcognos\B5A52\CBIFM-Start File\Module 19\GO Operational**. If prompted, log in as User ID **admin**, and Password **Education1**.
2. Publish the **GO Operational (query)** package.
3. In the Project Viewer pane, expand **Consolidation View**, and rename **Time** to **Time (Sale)**.

Task 2. Analyze the impact of changing the model.

1. Under **Packages**, click the **GO Operational (query)** package.
2. Right-click the **GO Operational (query)** package, and then click **Analyze Publish Impact**.

The Analyze Publish Impact dialog box appears.

3. In the **Changed Model Items** section, click **Time (Sale)**.

The results appear as follows:

Publish Impact for Package: GO Operational (query)

The list below shows the package changes relevant to the reporting environment. A change in a package may have an effect on reports that reference it.

Changed Model Items:

Object Name	Change	Actions
Time	Modified	[Action Buttons]
Time	Modified	[Action Buttons]
Time	Modified	[Action Buttons]
Time (Sale)	Modified	[Action Buttons]

Find Report Dependencies

Change details for:

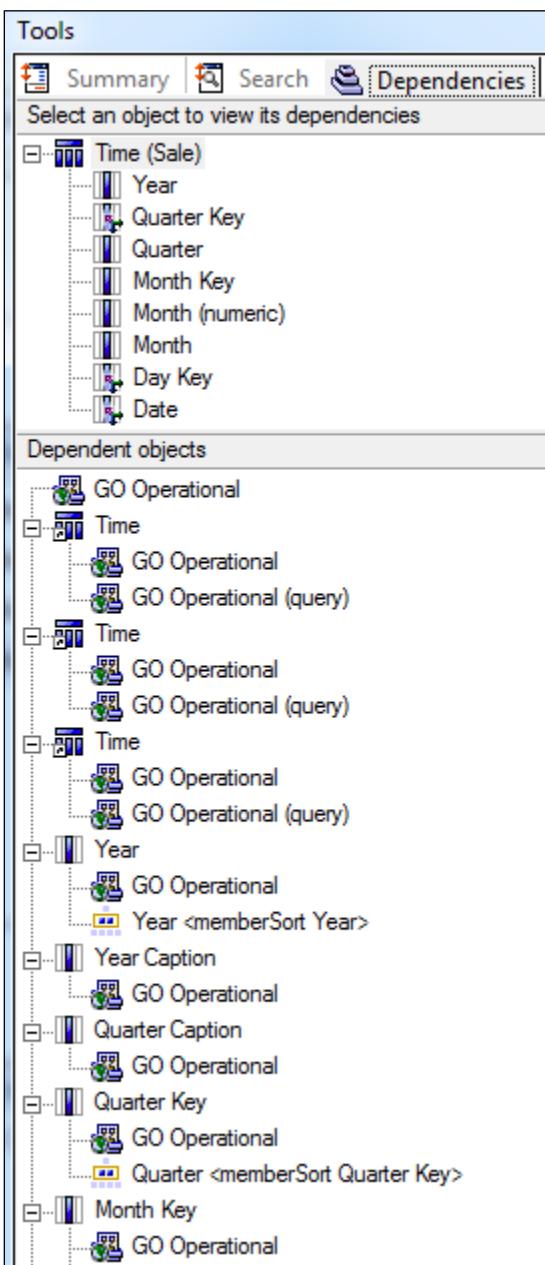
List of the relevant model item properties that have changed.

Property	Old Value	New Value
name[en]	Time	Time (Sale)

Notice the red X icon, which indicates that the change you made will break related reports. The "Change details for" section identifies the nature of the change, in this case the renaming of Time to Time (Sale).

Under Actions, you can show the object dependencies, find the object in the Project Viewer, or open the objects definition.

4. To the right of **Time (Sale)**, under **Actions**, click **Show Dependencies** .
- The results appear as follows:



In the Tools pane, the Dependencies tab identifies the modified query subject and its contents. Below it, the Dependent Objects pane list identifies all objects affected by the modified query subject, including shortcuts, model query subjects and query items.

5. In the **Analyze Publish Impact** window, to the left of **Time (Sale)**, select the check box.

6. Above **Actions**, click the **Find Report Dependencies** link.
You can choose the scope of your dependencies search.
7. Click **Search**.
The results appear as follows:

Model Object ID	Report Name	Query Name	Path
[Consolidation View].[Time].[Year]	Product Dimension Test	--	Public Folders > GO Operational
[Consolidation View].[Time].[Month (numeric)]	Product Dimension Test	--	Public Folders > GO Operational
[Consolidation View].[Time].[Date]	Product Dimension Test	--	Public Folders > GO Operational
[Consolidation View].[Time].[Year]	Test Time Dimension	--	Public Folders > GO Operational
[Consolidation View].[Time].[Month (numeric)]	Test Time Dimension	--	Public Folders > GO Operational

A list of the reports that are affected by your change is returned. Because you have not saved many reports, this change has little effect. You can identify the report the object appears in and how it is being used. Of course, in a production system this could identify hundreds of reports, indicating that it might not be practical to make this change now. You could enable model versioning to allow authors time to repair their reports. Model Versioning is discussed in another module.

You can click on any of the report names in the list to view and edit the report in IBM Cognos.

8. Click **Close** twice, and then **Close** the **Tools** pane.
9. In **Consolidation View**, rename **Time (Sale)** back to **Time**.
10. **Save** the project.

Results:

You made a change to the GO Operational model in Framework Manager. Before you re-published, you analyzed the package to determine what effect this change to the model would have on existing reports.

Remapping An Object to a New Source

- useful when underlying objects have changed
- remaps objects to new items
- typically used to remap middle layer(s) to new or changed data source items
- remaps automatically (based on name matching or by object reference) or manually (drag and drop)

© 2015 IBM Corporation



The remapping feature can be very useful in remapping your consolidation view (middle layer) to new or changed data source objects. For example, you may switch database vendors or database structures (move from an operational structure to a star schema structure).

Remapping is also useful if you decide to change how you modeled an underlying object or objects. For example, if your Product dimension in the Consolidation View references two underlying query subjects (Product and Product Type) and you decide to merge those underlying query subjects, you can use the remap tool to point the Product dimension query subject to the newly merged query subject in the lower layer.

Demo 2: Remap the Physical Layer

Purpose:

You have a new data source, **great_outdoors_warehouse**, which will replace the **gosales** data source under Foundation Objects View. Your goal is to remap all Consolidation View model query subjects to data source query subjects in the new data source. You will start this by remapping the Consolidation View's Time query subject to point to the new **GO_TIME_DIM**.

Component: Framework Manager

Project: GO Operational Maintenance

Task 1. Import the new metadata.

You will create another version of your model to perform the remapping technique. This will allow you to maintain a backup copy of your original project.

1. In **Windows Explorer**, navigate to **C:\Edcognos\B5A52\Course_Project**, and create a folder called **GO Operational Maintenance**.
2. In **Framework Manager**, click **File > Save As** and save your current project as **GO Operational Maintenance** in **C:\Edcognos\B5A52\Course_Project\GO Operational Maintenance**.
3. In the **Foundation Objects View** namespace, create a namespace called **GO Data Warehouse**.
4. Right-click **GO Data Warehouse**, and then click **Run Metadata Wizard**.
5. With **Data Sources** selected, click **Next**.
6. Select **great_outdoors_warehouse**, and then click **Next**.
7. Expand **GOSALESDW > Tables**, and select the **GO_TIME_DIM** table, and then click **Next**.
8. Click **Import**, and then click **Finish**.
9. Expand **GO Data Warehouse > GO_TIME_DIM**.
10. Select all the query items displayed as **Facts** , and change their **Usage** properties to **Attribute**.

Task 2. Remap Time to GO_TIME_DIM.

Before you remap an object, it is a good idea to find all object dependencies for the object that will be replaced with a new one. In this case, you are replacing the TIME_DIMENSION query subject from the gosales namespace. You will use the Dependencies feature to identify which objects will require remapping after you replace this query subject.

1. In the **Foundation Objects View** > **gosales** namespace, right-click **TIME_DIMENSION**, and then click **Show Object Dependencies**.

In the Tools pane, on the Dependencies tab, TIME_DIMENSION and its children appear in the top pane, and dependent objects appear in the bottom pane. Selecting items in the top pane filters the items in the bottom pane. The items in the bottom pane include dependent packages, security, relationships, query items, and determinants. If you scroll down in the Dependent objects list, you can select individual query items. Each one you select receives focus in the Project Viewer tree. This way you can identify which objects will require remapping.

2. In the **Tools** window, in the **Dependent objects** pane, select the first **Day Key**.

In the Project Viewer tree Consolidation, View>Time>Day Key is also selected, so you now know this object requires remapping to the new GO_TIME_DIM query subject. Other objects also require remapping, but for the purposes of this demo, you will only remap one object.

3. In **Consolidation View**, right-click **Time**, and then click **Remap to New Source**.

The right pane identifies the original source for all query items.

4. Under **Available Model Objects**, expand **Foundation Objects View** > **GO Data Warehouse** > **GO_TIME_DIM**.

Comparing left and right panes, it appears that the new query subject uses the same query item names as the original query items. You can drag each query item from the left to the right pane individually, or you can use the matching criteria feature. Note that 'Use matching criteria options' is selected.

5. Click **Options**.

Note that the default for matching criteria is to Remap To by name.

6. Click **Cancel**, and then drag **GO_TIME_DIM** under **Original Source**.

Name	Remap To	Original Source
Year	[GO Data Warehouse].[GO_TIME_DIM].[CURRENT_YEAR]	[gosales].[TIME_DIM]
Quarter Key	[GO Data Warehouse].[GO_TIME_DIM].[QUARTER_KEY]	[gosales].[TIME_DIM]
Quarter	[GO Data Warehouse].[GO_TIME_DIM].[CURRENT_QUARTER]	[gosales].[TIME_DIM]
Month Key	[GO Data Warehouse].[GO_TIME_DIM].[MONTH_KEY]	[gosales].[TIME_DIM]
Month (numeric)	[GO Data Warehouse].[GO_TIME_DIM].[CURRENT_MONTH]	[gosales].[TIME_DIM]
Month		[gosales].[Month]
Day Key	[GO Data Warehouse].[GO_TIME_DIM].[DAY_KEY]	[gosales].[TIME_DIM]
Date	[GO Data Warehouse].[GO_TIME_DIM].[DAY_DATE]	[gosales].[TIME_DIM]

A match was found for all query items, except for one calculation. This must be handled manually, since a match was not found because of different names.

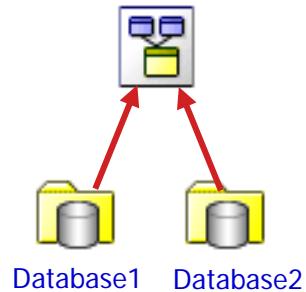
7. Clear the **Use matching criteria options** check box, and drag **MONTH_EN** from the left pane to the **Month** query item in the right pane.
8. Click **OK**.
9. Save the project.

Results:

You have remapped your Time model query subject to a new data source query subject, and are ready to continue the remapping from gosales to GO Data Warehouse.

Linking Multiple Data Sources

- you can import metadata from multiple data sources
- to link the data sources, create relationships between them on common query items



© 2015 IBM Corporation



When linking different types of data sources, you must create relationships on common data to obtain expected results. For example, if you imported an Inventory fact table from another data source that contained a Product Key (matching in values and data type to the Product Key in your existing Product dimension) you can use that key to create a relationship to the Product dimension. By doing so, you can create queries that retrieve data from separate data sources with related results.

Non-Database Data Sources

- In addition to using databases as data sources for your models, several file types can be set up as data sources as well, including:
 - Microsoft Excel spreadsheets
 - Microsoft Word tables
 - XML files
 - Comma separated value (CSV) files

© 2015 IBM Corporation



To use individual files as data sources, you can import them using the My Data Set feature in IBM Cognos BI. Administrators can use IBM Cognos Administration to perform additional management tasks on data sets.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

19-16

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Playing Back Parts of an Action Log

- modeling actions are written to an action log file
- you can save specific recorded actions to play back later
- this is useful for:
 - repeating identical sequences within a model
 - creating a starting point for multiple similar models
 - moving a model change from test environment to production
 - auditing or troubleshooting
 - bulk upgrades in batch mode

© 2015 IBM Corporation



Each sequence of actions that you perform in Framework Manager is considered a transaction. Each transaction is recorded in the action log file. You can view the history of transactions, and play back individual transactions or a combination of transactions in a log file. You can also save transactions to a separate log file (script).

The action log file is an XML file that is stored in the project folder. When troubleshooting a model with IBM Cognos Customer Support, the technical analyst will likely request the log files to be sent in.

You can also create scripts that support bulk upgrades of Framework Manager models. To do this, use the BmtScriptPlayer, which is a console application that you can use to create simple scripts that will process Framework Manager log files in batches.

After the script in a log file has run successfully, a backup of the original product is created in the parent directory of the project. If the outcome of running the script is undesirable, you can use the backup to restore the project to its original state.

You must disable or clear any commands that will conflict with the contents of the model. You can then run the script again. Or, you can use the Synchronize command, which begins with an empty model.

Synchronizing Projects

- use log files to synchronize a project with changes in the metadata source
- all actions performed in the model are replayed. The result is:
 - a new project is created
 - metadata is re-imported to capture any changes
 - the entire modeling process is repeated using updated metadata

© 2015 IBM Corporation



You can choose to accept the new changes and create a new project, or return to the original project. If you accept the changes, the original project is deleted. The Synchronize dialog includes a check box to back up the original project before synchronizing. We recommend that you select this check box.

After you save and close a project, changes are added to the log file. Because every action that you made in your project is re-run, synchronization may take a long time. Synchronization can be run only on the master project or a stand-alone segment. You cannot synchronize linked projects or segments in the master project.

You may encounter errors when running script files if an object that is referenced by a transaction no longer exists or if you renamed objects. If a script stops because an object no longer exists, you should retarget the missing object to another object. All remaining script transactions use the new object. If the script stops for any other reason, you should modify the temporary project to correct the problem.

Checking a Project

- high-level: verify selected objects
 - identifies invalid relationships, references, and object definitions
 - offers automated or manual repairs
- low-level: run model advisor
 - identifies design issues around relationships, determinants, and other model factors
 - offers direct links to online help for resolving issues

© 2015 IBM Corporation



These two tools have already been used and examined earlier in the course and are options available by right-clicking any objects in your model. They can be invoked at any time in the modeling process.

Demo 3: Run a Script to Replay Actions

Purpose:

You need to add a Business view folder and query subject in the test environment model. To quickly apply these changes to the corresponding model in the production environment, you will create a script based on the actions performed on the project in the test environment, and then run that script on the project in the production environment.

Component: Framework Manager

Project: GO Operational Maintenance

Task 1. Model a business view for report authors in the test environment.

1. In the root **GO Operational Model** namespace, create a folder called **Business View**.
2. Right-click the **Business View** folder, and click **Create > Query Subject**.
3. In the **Name** box, type **Retailers**, and then click **OK**.
4. In the **Available Model Objects** pane, expand **Foundation Objects View > gosales**.
5. Add the following items to the **Query Items and Calculations** pane:

Query Subject	Query Item
RETAILER_TYPE	TYPE_NAME_EN
Retailer & Retailer Site	RTL_ADDRESS1 RTL_ADDRESS2 RTL_CITY RTL_COUNTRY_CODE

6. Click **OK**.

The Retailers query subject now appears within the Business view folder.

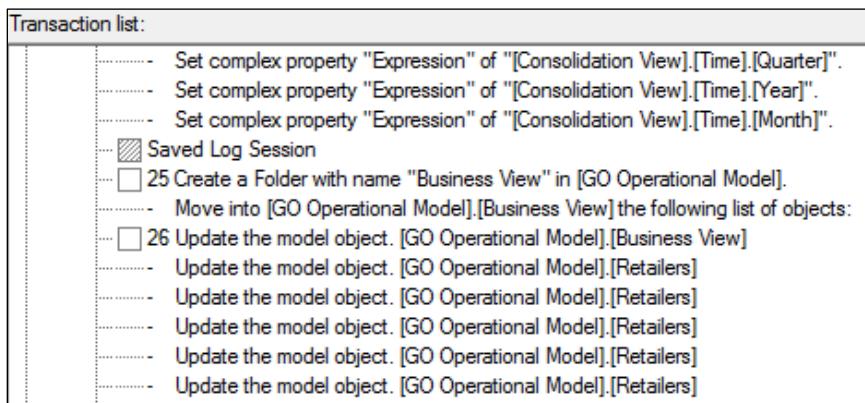
7. **Save** and **Close** the project.

This updates the log file that has been generated for this session. If you wish, you can examine the timestamp for this projects log.xml file in Windows Explorer.

Task 2. Create a script for the creation and population of the Business View folder.

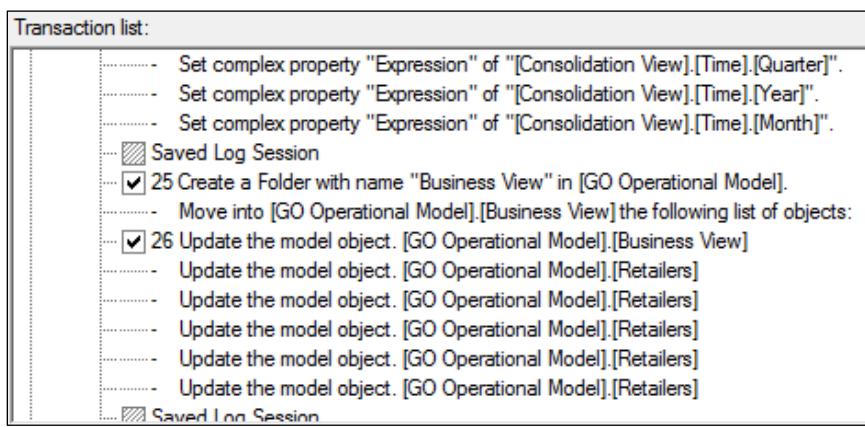
1. Under **Recent Projects**, open the **GO Operational Maintenance** project that you just saved.
2. From the **Project** menu, click **View Transaction History**.
3. Under **Transaction list**, expand **C:\Edcognos\B5A52\Course_Project\GO Operational Maintenance\log.xml** entry (first entry in the list).
4. Scroll to the bottom of the entry, and then expand the last child entry.

The results appear similar to the following:



5. Select the check boxes beside '**Create a folder with name Business View**' and '**Update the model object**'.

The results appear as similar to the following (preceding numbers may be different):



6. Click **Save as Script**.
7. In the **File name** box, type **Create Business View**, and then click **Save**.
8. Click **Close**.

- In Project Viewer, expand the **GO Operational Model** namespace, select the **Business View** folder, and then click the **Delete** button on the toolbar.

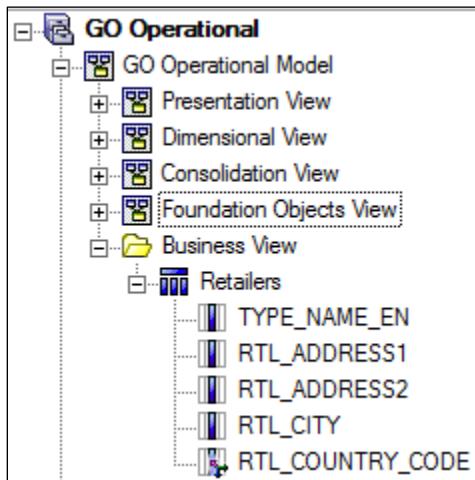
The Business View folder, and the Retailer query subject that it contains, no longer appear in the Project Viewer.

Task 3. Run the script in the production environment to recreate the Business view folder.

For the purpose of this demo, you will now assume that you are in the production environment.

- From the **Project** menu, click **Run Script**, and then double-click **Create Business View.xml**.
- Ensure that all check boxes are selected, and then click **Run**.
A status message quickly appears and then a transaction message appears in the Transaction details pane.
- Click **Accept**, and then expand **Business View > Retailers**.

The results appear as follows:



You successfully used a script to recreate the Business View folder, and the Retailers query subject.

- Save and Close the project.

Results:

You made changes to your project in a test environment, and created a script to record those actions. You then ran that script on your project in the production environment. The result is that all actions taken on your project in the test environment were applied, and are now reflected in the production environment.

Creating a Model Report

The screenshot shows a model report for a relationship named "PRODUCT_TYPE <--> SALES_TARGET". The report includes the following details:

PRODUCT_TYPE <--> SALES_TARGET	
Status	valid
Name ()	PRODUCT_TYPE <--> SALES_TARGET
Expression	[gosales].[PRODUCT_TYPE].[PRODUCT_TYPE_CODE] [PRODUCT_TYPE_CODE]
Left	Refobj [gosales].[PRODUCT_TYPE] Mincard one Maxcard one
Right	Refobj [gosales].[SALES_TARGET] Mincard one Maxcard many

© 2015 IBM Corporation

Select the entire model or any object within it, and then click Tools > Model Report. The details of every model object are presented. They can be saved in HTML or XML format for documentation purposes or for debugging.

To make searching easier, we recommend that you make a separate Model Report for each high-level namespace and/or folder in your project.

The above example shows the details of a relationship within the Foundation Objects View.

Business Analytics software

IBM

Summary

- You should now be able to:
 - perform basic maintenance and management on a model
 - remap metadata to another source
 - import and link a second data source
 - run scripts to automate or update a model
 - create a model report

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

19-24

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

The advertisement features a white background with a decorative border composed of teal hexagonal icons containing yellow cubes. In the top right corner, the blue IBM logo is displayed. The main title, "Optimize and Tune Framework Manager Models", is centered in large, bold, black font. Below it, the text "IBM Cognos BI" is displayed in a smaller, regular black font. At the bottom left, a small rectangular box contains the text "Business Analytics software". At the bottom right, there is a small copyright notice: "© 2015 IBM Corporation".

Optimize and Tune Framework Manager Models

IBM Cognos BI

Business Analytics software

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Objectives

- At the end of this module, you should be able to:
 - identify how minimized SQL affects model performance
 - use governors to set limits on query execution
 - identify the impact of rollup processing on aggregation
 - apply design mode filters
 - limit the number of data source connections
 - use the quality of service indicator

© 2015 IBM Corporation

Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

- User ID: admin
- Password: Education1

Business Analytics software

IBM

Materialized Views

- IBM Cognos queries typically aggregate data
- materialized views that contain aggregated values can significantly improve performance
- database optimization can capitalize on materialized views

Base Sales Fact Table

measures at the day level

Materialized View for Sales Facts

measures aggregated to the month level

© 2015 IBM Corporation

Materialization is a benefit for relational models. However it is highly recommended with dimensional modeling. Modeling relational data dimensionally adds another layer in your model which enables OLAP behaviors in reports. To provide the best performance, materialization in the database should be in place.

Common database vendor implementations include:

- Microsoft SQL Server - view index
- Oracle - materialized view
- IBM DB2 UDB - materialized query tables
- NCR Teradata - aggregate join indexes
- Informix RedBrick - VISTA

Achieving Minimized SQL

- minimized SQL may not occur when:
 - relationships are added to model query subjects
 - data source query subject SQL is altered
 - calculations, filters, or determinants are added
- consider using SQL minimization as you model, not towards the end of development

© 2015 IBM Corporation



Minimized SQL is more likely to take advantage of database optimization, compared to more complex SQL. However, there may be times when losing SQL minimization is necessary, such as when you need to: add a relationship to model query subjects with overriding relationships, to change the SQL of data source queries, or to add filters, calculations, or determinants.

When you add a relationship to a model query subject that comprises two or more underlying query subjects that have relationships to each other, the underlying joins will be honored. This is considered As View behavior. Also, if the SQL for data source query subjects is altered, there is potential for minimized SQL to be lost. For example, hard coding a join in the SQL will force that join in each query using items from that query subject.

Using Dynamic Query Mode (DQM)

- an enhanced query execution mode that optimizes query planning, execution, and results
- users get the right information quickly without compromising security
- recommended mode for creating new IBM Cognos BI applications in a supported environment
- non-DQM projects can be tested, published, or migrated to dynamic query mode

© 2015 IBM Corporation



Dynamic query mode is a Java-based query mode which offers:

- Query optimizations that address query complexity, data volumes and timeliness expectations, with improved query execution techniques.
- Significant improvement for complex OLAP queries through intelligent combination of local and remote processing and better MDX generation.
- Support for relational databases through JDBC connectivity.
- OLAP functionality for relational data sources when using a dimensionally modeled relational (DMR) package.
- Security-aware caching.

By default, all packages created in Framework Manager version 10.1 and earlier are created in compatible query mode (CQM). If your data sources are supported, you can take advantage of dynamic query mode capabilities by either creating a new DQM project, or converting an existing project to DQM.

For an up-to-date list of supported environments, see
<http://www.ibm.com/support/docview.wss?uid=swg27019126>.

Using Dimensional Metadata with DQM

- when used with dimensional metadata, DQM offers:
 - enhanced member sorting capabilities
 - the ability to perform complex aggregate computations
 - greatly improved query performance
 - optimized SQL statements that create an in-memory cube for OLAP over Relational (OOR) reporting

© 2015 IBM Corporation



Dynamic query mode supports a consistent OLAP-style experience across data sources and studios.

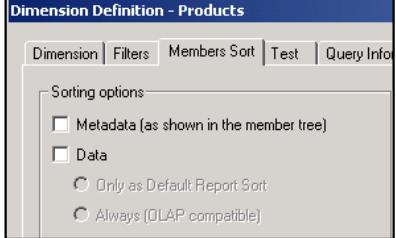
When you publish a DMR package from a DQM-enabled project, it is available to report authors as an OLAP over Relational (OOR) package. The first time you run a report that is based on an OOR package, the query engine creates a virtual cube. Subsequent queries can then re-use the objects that already exist in the cube, and continue to build the in-memory cube by adding newly queried objects.

Note: To avoid the performance impact of creating the cube for the initial query, you can pre-run a report during off hours to populate the cache for the day.

Business Analytics software

Member Sorting in DQM

- sorting options for dimensional metadata are not required in dynamic query mode
- regardless of these settings, order is natural (sort on parent), or as the designer specified
- child members are ordered within their parent members



Region	Retailer country
Americas	Brazil
Americas	Canada
Americas	Mexico
Americas	United States
Asia Pacific	Australia
Asia Pacific	China
Asia Pacific	Japan
Asia Pacific	Korea
Asia Pacific	Singapore
Central Europe	Belgium
Central Europe	France

© 2015 IBM Corporation



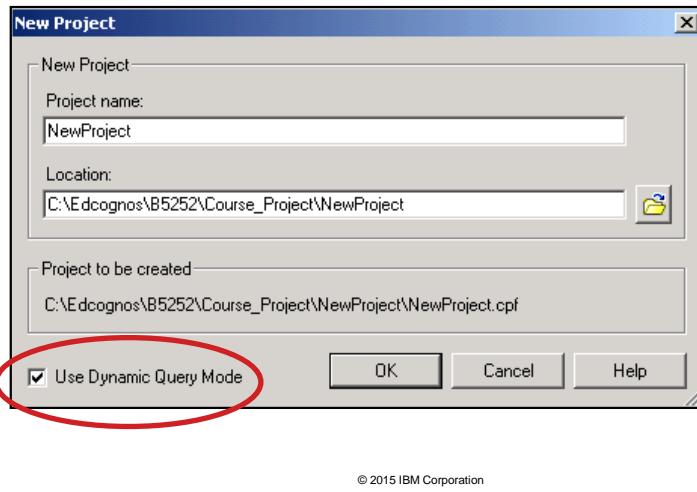
Member order is vital for OLAP analysis to ensure consistent results, especially when using member relative functions, such as NextMember, and ClosingPeriod. In CQM, you can declare sort orders by level in Regular Dimensions; however if no sort order is specified, SQL returns the members in whatever order it retrieves them from the relational tables. Since SQL does not order by default, results can be unpredictable.

With OLAP over Relational, DQM specifies a natural order to all result sets. Child members are only ordered within their parent members, even if all members of a level are reported. If no ordering is specified in the model, the members are sorted in ascending order, by member caption. If there are duplicate captions, they are sorted by business key.

To order all level members by just the captions in the level, the sort order must be specified in the report design.

Creating a DQM Enabled Project

- you can enable dynamic query mode when you create a new project



You can specify that a project will use DQM mode when you create a new package. After you do this, the default mode for testing and publishing objects in that project will be DQM.

Note that the data source connection for DQM-enabled projects must have a connection to both the content store and a JDBC driver. This information is defined in Cognos Administration when creating or editing a data source connection. Data source connections can only be defined by users with administrative access in the Cognos BI environment.

Demo 1: Create a Dynamic Query Mode Enabled Project

Purpose:

You want to create a new project with dimensional metadata, and decide to use dynamic query mode in order to optimize the performance. To do this, you will create a dynamic query mode enabled project, define a JDBC connection to the data source, and publish a package in DQM.

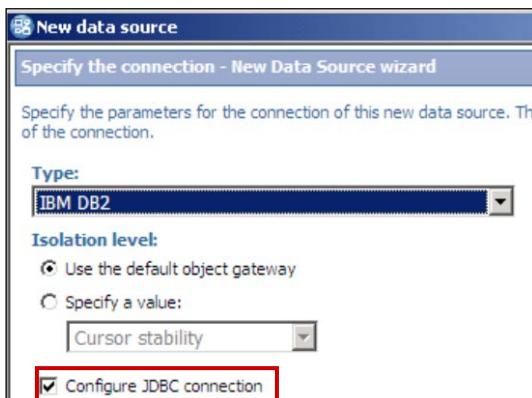
Task 1. Create a DQM-enabled project and data source connection.

The following instructions use the Sample Outdoors Warehouse IBM DB2 database called GS_DB to allow you to create a data source connection to a relational data source and JDBC driver.

1. In **Framework Manager**, click **Create a new project**.
2. In **Project name** box, type **GO Warehouse (DQM)**.
By default, the Use Dynamic Query Mode check box is already selected, which is what you want.
3. In **Location**, navigate to **C:\Edcognos\B5A52\Course_Project**, and then click **OK**.
If necessary, login with **admin/Education1**.
4. In the **Select Language** dialog box, ensure that **English** is selected, and then click **OK**.
5. Ensure **Data Sources** is selected, click **Next**.
6. Click **New** to create a new data source connection.
7. In the **New Data Source** wizard, click **Next**.
8. In the **Name** box, type **GOSALES DW**, and then click **Next**.

- Under **Type**, click **IBM DB2**, and ensure that the **Configure JDBC connection** check box is selected.

You want to configure the JDBC connection so that information can be provided to connect through the JDBC driver which is required for Dynamic Query Mode.



- Click **Next**.

In the next steps, the information provided is based on how the IBM DB2 clients on the Framework Manager machine and the IBM Cognos BI servers were configured and how security is implemented for IBM DB2. Connection information and sign on information should be provided by the database administrator.

- In the **DB2 database name** box, type **GS_DB**.
- Under **Signons**, click the **Password** check box to select it, and in the **User ID** box, type **GOSALESdw**, and then in the **Password** and **Confirm password** boxes, type **Education1**.
- Click **Test the connection**, and then click **Test**.

Notice that the Query Mode is Compatible. This only indicates that Dynamic Query Mode has not been configured for this data source at this point.

...> Name	Type / Query Mode	Status	Message
...> http://vclassbase:9315/p2pd	IBM DB2 / Compatible	Succeeded	

- Click **Close**, click **Close** again, and then click **Next**.
You will now configure the JDBC connection.
- In the **Server name** box, type **VCLASSBASE** (the name of the server hosting the database), in the **Port number** box, type **50000** (the port number of the database), and then in the **Database name** box, type **GS_DB**.

16. Click **Test the connection**, and then click **Test**.

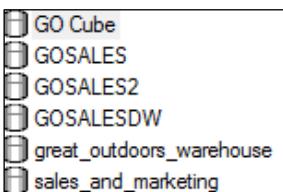
Notice that the Query Mode is Dynamic.

...> Name	Type / Query Mode	Status
...> http://vclassbase:9315/p2pd	IBM DB2 (JDBC) / Dynamic	Succeeded

17. Click **Close**, and then click **Close** again.

18. Click **Finish**, and then click **Close**.

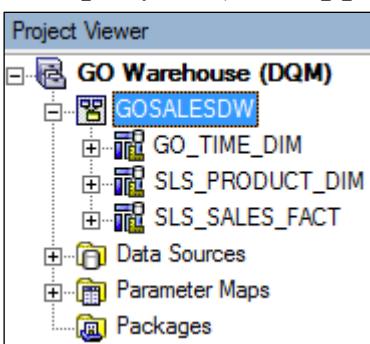
The new data source that appears in the list is configured to query using either query mode.



Task 2. Import metadata and test query subjects.

1. Click **GOSALESDW**, and then click **Next**.
2. Expand **GOSALESDW > Tables**, and select:
 - **GO_TIME_DIM**
 - **SLS_PRODUCT_DIM**
 - **SLS_SALES_FACT**
3. Click **Next**, click **Import**, and then click **Finish**.
4. In the **Project Viewer**, expand **GOSALESDW**.

The query subjects appear as child objects as shown below.



5. Double-click **GO_TIME_DIM** to open its definition, and then click the **Test** tab.

If the Query Mode property of the project is set to Dynamic when testing a query subject, the test query will run in Dynamic Query Mode. If the Query Mode property is set to Compatible, however, there is an option to use the Dynamic Query Mode on the Test tab in the lower left corner, provided that the query subject is for a data source supported by the Dynamic Query Mode. In this case, the property for this project was set to Dynamic Query Mode, and therefore you do not see this check box option.

6. Click **Test Sample**.

Framework Manager sends the test query through the IBM Cognos gateway to one of the IBM Cognos BI servers, which queries the reporting database. The data retrieved by the test query appears in the Test results pane.

A section of the results appears as follows:

Test results						
DAY_KEY	DAY_DATE	MONTH_KEY	CURRENT_MONTH	MONTH_NUMBEF	QUARTER_KEY	CURR
20100000		201000	0	0	20100	0
20100101	Jan 1, 2010 12:00:00 AM	201001	1	1	20101	1
20100102	Jan 2, 2010 12:00:00 AM	201001	1	1	20101	1
20100103	Jan 3, 2010 12:00:00 AM	201001	1	1	20101	1
20100104	Jan 4, 2010 12:00:00 AM	201001	1	1	20101	1
20100105	Jan 5, 2010 12:00:00 AM	201001	1	1	20101	1
20100106	Jan 6, 2010 12:00:00 AM	201001	1	1	20101	1
20100107	Jan 7, 2010 12:00:00 AM	201001	1	1	20101	1
20100108	Jan 8, 2010 12:00:00 AM	201001	1	1	20101	1
20100109	Jan 9, 2010 12:00:00 AM	201001	1	1	20101	1

7. Click **OK** to close the **Query Subject Definition** window.

You should test all your model objects against the Dynamic Query Mode to ensure that the generated SQL is as expected for your requirements. If you are building a DMR model, this includes foundation objects such as Data Source and Model Query Subjects as well as Regular and Measure Dimensions.

After you have built the model, you can create and publish a DQM package.

Task 3. Create and publish a package that uses the dynamic query mode, and then verify the package properties.

1. In the **Project Viewer**, right-click **Packages**, point to **Create**, and then click **Package**.
 2. In the **Name** box, type **GOSALES DW**, click **Next**, and then click **Finish**.
 3. Click **Yes**, deselect the **Enable model versioning** check box, and then click **Next** twice.
- Note: If the Query Mode property of the project was set to Compatible, and the package contained supported DQM data sources, the wizard dialog would have a Use Dynamic Query Mode check box option. You have already defined the property of this project as Dynamic Query Mode, so this option does not appear.

4. Click **Publish**, and then click **Finish**.

The package is now available in IBM Cognos BI, and will use the Dynamic Query Mode for reports written against this package. In IBM Cognos Connection, the query mode used by the package can be verified in the package properties.

5. **Save** and **Close** the project.

Task 4. Create and run a simple report based on a DQM source.

1. Log into **IBM Cognos** using **admin/Education1**, and launch **IBM Cognos Connection**.
 2. Beside the **GOSALES DW** package, click **Set properties** . Notice that the Query Mode is Dynamic.
- Query Mode:**
Dynamic
3. Click **OK**.
 4. Click the **GOSALES DW** package, launch **Report Studio**, and create a **List** report.

- From the **Source** pane, populate the list with the following items:

- GO_TIME_DIM > DAY_DATE**
- SLS_PRODUCT_DIM > PRODUCT_KEY**
- SLS_SALES_FACT > QUANTITY**

You could use any items from the package for your report; the specific content of the items selected above is not important to this demo.

- Click **Run**.

The results appear as follows:

DAY_DATE	PRODUCT_KEY	QUANTITY
Jan 12, 2010 12:00:00 AM	30001	51,522
Jan 12, 2010 12:00:00 AM	30002	24,182
Jan 12, 2010 12:00:00 AM	30003	11,265
Jan 12, 2010 12:00:00 AM	30004	17,567
Jan 12, 2010 12:00:00 AM	30005	9,412
Jan 12, 2010 12:00:00 AM	30006	6,414
Jan 12, 2010 12:00:00 AM	30007	6,302
Jan 12, 2010 12:00:00 AM	30008	3,671
Jan 12, 2010 12:00:00 AM	30009	17,725
Jan 12, 2010 12:00:00 AM	30010	17,506

- Close **IBM Cognos Viewer**.
- Save the report as **DQM Report**, and then close **IBM Cognos Report Studio**.

Results:

By creating and publishing a dynamic query mode enabled package, you were able to provide dimensional metadata to report authors that will generate consistent results with optimized performance.

Using DQM Features in a Compatible Mode Project

- in a compatible query mode project, you can use dynamic query mode to:
 - test query subjects or query items
 - publish a package
- you can migrate a project to DQM if it is no longer required in compatible query mode

© 2015 IBM Corporation



If your Framework Manager project uses compatible query mode, you can still test queries and publish packages using dynamic query mode, as long as your data sources are supported by DQM. You can even change the query mode of the entire project, so that all testing and publishing is automatically performed in dynamic query mode.

Keep in mind that publishing a package in dynamic query mode, or changing the query mode of a project, can negatively impact any reports that use the package or project. This is because dynamic query mode enforces best practice design rules which the existing reports may not already adhere to. As well, when you change a project to DQM, you (or the administrator) will need to update the data source connection to include the JDBC connection information.

When changing a project or package to dynamic query mode, it is important to work closely with the report authors who use those packages. This will help you to ensure that the benefit of changing the query mode outweighs the effort to ensure that reports function as required.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

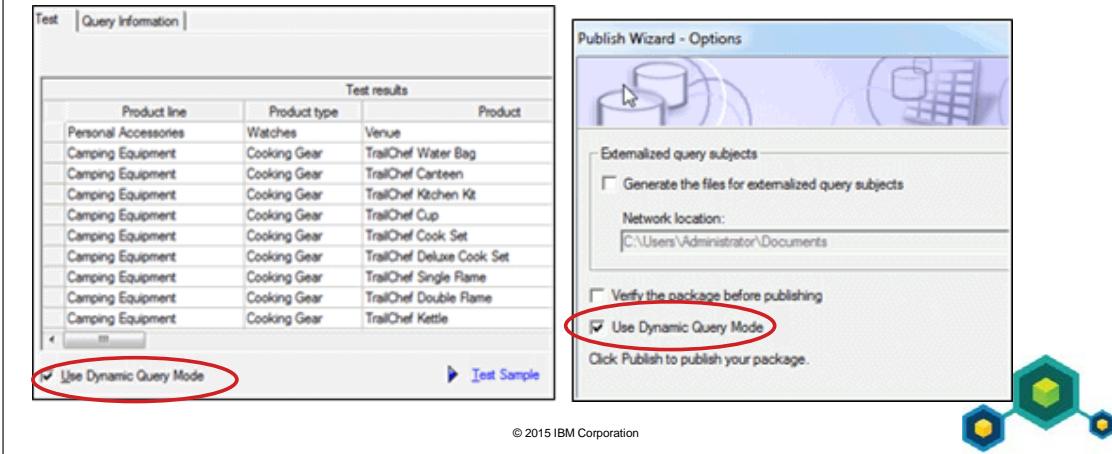
20-16

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Testing and Publishing CQM Projects in DQM

- test query subjects and query items in DQM
- publish CQM packages in DQM, so that all reports assigned to that package will run in dynamic query mode



Whether you are using a compatible or dynamic query mode project, you can test objects and publish packages in dynamic query mode. However, to ensure consistency and accuracy of the metadata, we recommend that you only perform DQM actions on DQM projects.

Before you use DQM to publish an existing package, you should be sure that this will not negatively impact any of the reports that consume this package.

When you create a new project using dynamic query mode, the Use Dynamic Query Mode check box does not appear, because that is the default mode. However, if you change the project's query mode to CQM, the check box will appear.

Business Analytics software

IBM

Migrating an Existing Project to DQM

- change a CQM project to DQM by setting the project's Query Mode property to Dynamic

Name	go_sales
Languages	<Click to edit.>
Design Language	en
Use Design Locale for References	false
Query Mode	Compatible Dynamic

The 'Dynamic' option is highlighted with a red oval.

- before you change the query mode:
 - ensure that all data sources are supported by DQM
 - work with report authors to evaluate the impact on existing reports
 - ensure that the benefits are worth the effort to resolve any impact to existing reports

© 2015 IBM Corporation



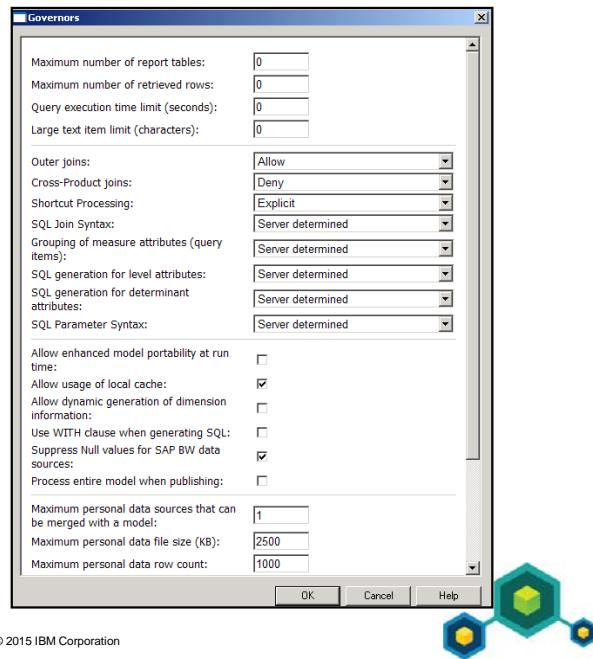
Not all existing applications will benefit from migrating to dynamic query mode. The report author can use Lifecycle Manager to determine the visual and performance impact of migrations without permanently altering package.

When you convert an existing compatible query mode project to dynamic query mode, you will need to identify and fix any issues that arise from the enhanced rules that DQM enforces. For example, if you attempt to publish a package in DQM that has a many-to-many cardinality, you will get an error message.

When connecting to a relational data source, you (or your administrator) must add JDBC connection parameters to the existing data source connection, since dynamic queries run on JDBC.

Using Governors to Set Limits on Query Execution

- reduce system resource requirements and improve performance
- set at the project level, in the Project > Edit Governors menu



SQL is automatically generated whenever you test a query subject, create a query subject based on other objects, or run a report in a studio. By setting governors, you can improve the performance of SQL generation. For example, you can:

- restrict the number of report tables or rows returned
- set time limits for query executions
- deny runtime activities that are resource-intensive, such as outer joins
- allow reports to run against cached data, so that the service does not have to re-query the database

You must specify project governors before you publish packages to ensure the metadata for each package uses the specified limits. All packages that are subsequently published use the new settings.

Business Analytics software

IBM

Dynamic Query Mode Governors

- there are four DQM-specific governors

(DQM) Adjust SQL generation for Exact Numeric Division	Cast to Double
(DQM) Cache is sensitive to Connection Command Blocks:	<input checked="" type="checkbox"/>
(DQM) Cache is sensitive to DB info:	DB + Connection + Signon
(DQM) Cache is sensitive to Model Security:	Automatic

- configure security-sensitive settings, which affect the amount of cached information that is shared between users

© 2015 IBM Corporation



The governors for the dynamic query mode control division calculations and the security of cached data to ensure that only authorized users view cached data. Keys which are unique to users control access to the cache. You can configure these keys with the dynamic query mode cache governors.

- (DQM) Adjust SQL generation for exact numeric division: Controls how calculations with divisions are adjusted to ensure that the division results contain information that is significant for the reports.
- (DQM) Cache is sensitive to connection command blocks: Specifies whether the cache key includes the expanded value of the connection command blocks.
- (DQM) Cache is sensitive to DB info: Controls the sensitivity of the cache associated with a package that is shared by users of the connection. It also specifies what database information is used to restrict sharing in that cache (DB, DB + Connection, or DB + Connection + Signon).
- (DQM) Cache is sensitive to model security: Controls the security that is used to access the cache (Automatic, User, UserClass, None).

Metadata Caching

- Framework Manager metadata:
 - describes objects imported from your data source and any model objects created from them
 - is cached in the RTM file
- the RTM file is:
 - where the server retrieves package metadata from
 - created when a user first accesses a package in any studio
 - stored in <C10_install_location>\data\cqe\RTModels

© 2015 IBM Corporation



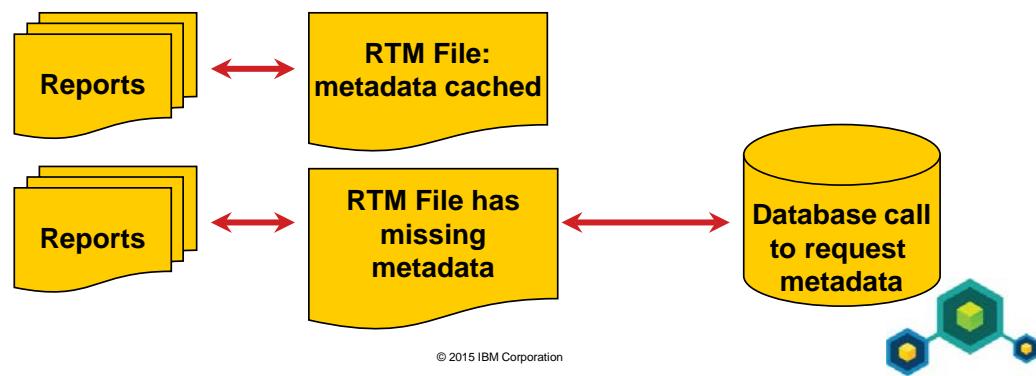
When you run a report for the first time, the query request is sent to the database, and a result set is returned. The query and result are stored in the cache for each user for their current session. When you run the report again, the cached query and result can be used without having to query the database again.

To take advantage of the improved query performance, enable the Allow Usage of Local Cache governor. For users that require the latest data, consider publishing a separate package with this governor disabled.

You should try to only include the metadata that is required in the packages that you publish. The larger that an RTM file is, the longer it will take to open the package in a studio. File size can be especially significant for packages that contain cube data sources.

How do Reports Use Cached Metadata?

- reports require metadata and descriptions of that metadata to fulfill queries
- uncached metadata results in additional database calls



After the data source connection has the metadata information, it is cached in memory for the duration of the connection only.

By default, queries will always use cached RTM file metadata. However, if metadata is not cached, the query service is forced to fetch the metadata from the database. Query performance may become an issue if there are numerous call backs for metadata.

When is Metadata Not Cached?

- metadata is not cached when you:
 - enable the Allow Enhanced Model Portability governor
 - modify a data source query subject by:
 - editing the SQL statement
 - embedding filter or calculation objects
 - adding macros, parameter maps, or prompts

© 2015 IBM Corporation



You can disable metadata caching for an entire project by enabling the Allow Enhanced Model Portability governor, which forces the server to get metadata from the database. Only use this setting if the model is being used in multiple environments where the columns in the database could potentially change in data type.

Metadata is also not cached when you modify a data source query subject. For example, when you add a language macro and parameter map reference to the SQL expression of the PRODUCT_LINE query subject, the query engine retrieves the values from the appropriate PRODUCT_LINE column in the data source based on the user's locale setting. In this case, metadata for the query subject cannot be cached in the model because the definition of the PRODUCT_LINE column is dynamic in nature. From one locale to the next you will receive different columns or expressions with potentially different data types. The end result is that a call must be made to the data source when the query is executed to determine the properties of the relevant column used by PRODUCT_LINE.

Security-aware Caching in DQM

- Members and data are cached as queries are executed.
- Data is retrieved from cache if it exists and if security profiles match.
- Lets you quickly interact with information, such as: sorting, filtering, and formatting.
- The performance benefits of the cache is most noticeable when executing:
 - similar reports with small modifications
 - repeated analysis within the same cube
 - repetitive master-detail requests for large reports

© 2015 IBM Corporation



Dynamic query mode provides a greater degree of secure, smart caching, which offers significant performance improvements for most queries and workloads.

The local processing approach to dimensional reporting is broken down into two simple steps: metadata fetch and data fetch. When a report is executed, the query service retrieves all members requested (metadata), either by level and/or unique member inclusion, and then uses the retrieved members to construct the MDX used for data retrieval (facts). As these calls are performed, for both metadata and data requests, each result is cached.

When connected to secured metadata sources, the caching logic available in Dynamic Query Mode can determine the access level of each user as they query the data source. Then, if any subsequent queries are made in the same context (i.e. security profile, metadata, and data), the cached results will be used.

Demo 2: Identify the Use of Cached Metadata by a Query

Purpose:

In order to understand how modifying data source query subjects can impact query performance, you will identify when the queries generated from our data source query subjects are using cached metadata, and when they are not.

Component: **Framework Manager**

Project: **great_outdoors_warehouse**

Task 1. View a metadata call back to the database in Framework Manager.

You will first add a calculation to a data source query subjects and then view the effects of that calculation.

1. In **Framework Manager**, open the **great_outdoors_warehouse** project located at: **C:\Edcognos\B5A52\CBIFM-Start Files\Module 20\great_outdoors_warehouse**.
2. In the **Project Viewer**, expand **go_data_warehouse > Database view > Sales and marketing data**, and then double-click **SLS_SALES_FACT**.
3. Click the **Calculations** tab, and then click **Add**.
4. In **Name**, type **Planned Revenue**.
5. In **Available Components**, expand the **Database view**, and create the following expression:

[Sales and marketing data].[SLS_SALES_FACT].[QUANTITY] * [Sales and marketing data].[SLS_SALES_FACT].[UNIT_PRICE]

Now this data source query subject includes an embedded calculation.

6. Click **OK** twice.
7. In **SLS_SALES_FACT**, test **Planned Revenue**.

8. Click the **Query Information** tab, and then click the **Response** tab.

The xml response from the IBM Cognos server for this query appears. Notice the text shown below:

RQP-DEF-0543 Metadata will be retrieved from the database, because a simple DB query subject definition with matching metadata does not exist.

This indicates that the data source query subject has been altered and is no longer considered a simple query subject. This occurred because the query subject contains a calculation which results in a dynamic query, which cannot be resolved using the metadata currently cached in the model.

This test can also be run in Report Studio by going to the Tools menu, Validate Options, and selecting Information in the Validation level list. Now when you validate the report, you will be able to see this message if it is generated.

9. Click **Close**, delete the **Planned Revenue** calculation from the **SLS_SALES_FACT** query subject.
 10. Test the **SLS_SALES_FACT**, click the **Query Information** tab, and then click the **Response** tab.
- The metadata call back message no longer appears since cached metadata is used.
11. Save the project.

Results:

You identified when the queries generated from our data source query subjects use cached metadata in the model, and when they do not. This was achieved by viewing the query response information.

Query Processing Types

- control where the query is processed to improve performance
- there are two types of query processing:
 - Limited Local: the database server does as much of the SQL processing and execution as possible
 - Database Only: the database server does all the SQL processing and execution. An error appears if any reports or report sections require local SQL processing.

© 2015 IBM Corporation



For relational metadata, you can specify whether SQL processing is performed by the database server or processed locally. In Framework Manager, Query Processing is a property of a data source object. However, these properties can be overridden for individual reports in Report Studio.

Although the database server can usually run the SQL and run reports much faster, local processing is sometimes necessary. For example, choose limited local processing if you want to create cross-database joins or if you want report authors to use unsupported SQL99 functions. Some complex queries require limited local processing, such as a query that must generate an At clause to avoid double-counting. In this case, the query automatically uses limited local processing even if the package was published with database only processing.

Determine where Aggregates are Calculated

- calculating aggregate values can be process-intensive
- to improve performance, specify whether aggregates are calculated at the database or application level
- there are four values for rollup processing:
 - Unspecified
 - Extended
 - Database
 - Local

© 2015 IBM Corporation



In Framework Manager, Rollup Processing is a property of a data source object, which determines where the aggregation of report summary values is calculated, based on the output type of the report. The default Rollup Processing property setting is Unspecified in Framework Manager (named Default in Report Studio); however you can also set the value to Extended, Database, or Local:

- Extended - instructs the query engine to always use extended forms of aggregates. Remember: Extended aggregate (XSUM, XAVG, XMIN) operations are pushed to the database.
- Database - instructs the query engine to use running aggregates where possible. The Database setting pushes the running aggregation to the database using derived tables if the database vendor supports this feature. If not, the running aggregation is performed locally.
- Local - instructs the query engine to use running aggregates where possible. Local will force the query engine to compute the running aggregates at the IBM Cognos application tier versus at the database tier. This can offload some of the computation load from the RDBMS server, which can be beneficial when there is more compute capacity at the application engine tier than the data tier.

Demo 3: Examine Rollup Processing and Generated SQL

Purpose:

A report author wants IBM Cognos to perform certain aggregations locally rather than at the database level. You will show the report author how to accomplish this in Report Studio, and identify what to expect in the generated SQL.

Components: Report Studio, Framework Manager

Project: great_outdoors_warehouse

Package: GO Data Warehouse (query)

Task 1. Author a report with summary values.

1. In **IBM Cognos Connection**, navigate to **Public Folders > Samples > Models**, and then click **GO Data Warehouse (query)**.
2. Launch **Report Studio**, click **Create New**, and then double-click **List**.
3. In the **Source** pane, expand the **Sales and Marketing (query) > Sales (query) > Retailers**, and then add **Retailer name** to the list.
4. Expand **Sales (query) > Sales Fact**, and add **Quantity** to the list.
5. Click the **Quantity** column, on the toolbar click **Summarize** , and then click **Total**.

The results appear as follows:

Retailer name	Quantity
<Retailer name>	<Quantity>
<Retailer name>	<Quantity>
<Retailer name>	<Quantity>
Overall - Total	<Total(Quantity)>

Task 2. View the Generated SQL and set the appropriate option.

1. From the **Tools** menu, click the **Show Generated SQL/MDX**.
2. In the drop-down list, select **IBM Cognos SQL**.

The SQL appears as shown below:

```
IBM Cognos SQL
select
    Retailer_site.Retailer_name as Retailer_name,
    XSUM(SLS_SALES_FACT.QUANTITY for
Retailer_site.Retailer_name) as Quantity,
    XSUM(XSUM(SLS_SALES_FACT.QUANTITY for
Retailer_site.Retailer_name) at Retailer_site.Retailer_name) as
Total_Quantity_
```

Notice that Total_Quantity is totaled for Retailer_site.Retailer_name. This represents the summary total at the bottom of the report. This aggregated total is displayed using extended aggregates based on the presence of the XSUM(XSUM) function. This SQL was generated based on the default setting. With this setting, you always see SQL generated based on batch mode.

3. Click **Close**.
4. On the **Explorer** bar, point to **Query Explorer** , and then click **Query 1**.
5. In the **Properties** pane, under **Query Hints**, click the **Rollup Processing** row, and then in the list, click **Local**.
6. On the **Explorer** bar, point to **Page Explorer** , and then click **Page 1**.
7. Click **Tools > Show Generated SQL/MDX**, and view the generated **IBM Cognos SQL**.

The SQL appears as shown below:

```
IBM Cognos SQL
select
    Retailer_site.Retailer_name as Retailer_name,
    XSUM(SLS_SALES_FACT.QUANTITY for
Retailer_site.Retailer_name) as Quantity,
    RSUM(XSUM(SLS_SALES_FACT.QUANTITY for
Retailer_site.Retailer_name) at Retailer_site.Retailer_name order by
Retailer_site.Retailer_name asc local) as Total_Quantity_
```

The aggregated total is now displayed as using running aggregates based on the presence of the RSUM function. This setting ensures local processing of the summary totals no matter what the output format of the report is.

8. Click **Close**.
9. In **Query Explorer**, select **Query 1** and change the **Rollup Processing** property to **Extended**.
10. In **Page Explorer**, select **Page 1**, and click **Tools > Show Generated SQL/MDX** to view the generated **IBM Cognos SQL**.
The SQL appears the same as in the first example because you are explicitly asking for an extended aggregate.
11. In **Query Explorer**, select **Query 1**, and change the **Rollup Processing** property to **Database**.
12. In **Page Explorer**, select **Page 1**, and click **Tools > Show Generated SQL/MDX** to view the generated **IBM Cognos SQL**.
Again, the RSUM function is present. The Database setting instructs the IBM Cognos query engine to use running aggregates where possible. The Database setting pushes the running aggregation to the database using derived tables if the database vendor supports this feature. If not, the running aggregation is performed locally.
13. Close **Report Studio** without saving.

Task 3. Examine the Rollup Processing property in Framework Manager.

1. In **Framework Manager**, expand **Data Sources**, and then click **go_data_warehouse**.
2. In the **Properties** pane, click the **Rollup Processing** property value, and then click the drop-down list to view the available settings.
When the corresponding property in Report Studio is set to Default, the model's value defines the default behavior for every report that is based on the data source. However, as you have seen in this demo, you can still override the model setting for an individual report in Report Studio.
3. Leave **IBM Cognos Connection**, and **Framework Manager** open for the next demo.

Results:

You demonstrated how to force local processing of summary totals by setting the Rollup Processing property in Report Studio. You also examined how to set the rollup processing value for a project.

Using Filters to Improve Performance

- reducing the number of retrieved rows improves performance
- you can do this by using:
 - embedded filters
 - embedded filters with prompts (allow users to focus their queries)
 - stand-alone filters
 - security filters

© 2015 IBM Corporation



You can build mandatory filters into the model to ensure that consumers do not retrieve excessively large data sets when running reports. This is useful when the users do not always need to see all of the data.

All filter types (listed above) limit the data set retrieved, resulting in a decreased query processing load.

You should consider filtering data sets when modeling relational data dimensionally. This will help offset any decrease in performance from having an additional layer of metadata in your model.

Prompt Info properties can be set on query items to improve performance as well (as discussed in an earlier module). Essentially you want to ensure that an indexed field is used to filter a query while at the same time providing user friendly values for selection. For example, set the Product Line query item's Filter Item Reference property to use Product Line Code. This way IBM Cognos generated prompts will use the Product Line Code in the Where clause of the query while allowing users to select values represented by Product Line.

Applying Design Mode Filters

- use design mode filters to improve performance when:
 - testing query subjects in Framework Manager
 - designing reports in Report Studio and Query Studio

© 2015 IBM Corporation



Apply design mode filters in query subjects to limit the amount of data that report authors and modelers retrieve when testing and designing. By limiting data retrieval, design time results appear more quickly.

Design mode is one of three options available for a filter's Usage property in Framework Manager. The Usage property is accessible from the Filter tab of the Query Subject Definition dialog box.

After design mode is set as the usage for a filter, it must then be applied. In Framework Manager it is applied for testing purposes using the Options dialog box, which is available from the Test tab of the Query Subject Definition dialog box.

In Query Studio, it is applied for design purposes using the Preview with Limited Data option, which is available from the Run Report menu. In Report Studio, it is applied using the Run Options dialog box, which is available from the Run menu. Select the data mode you would like to run in.

Demo 4: Apply Design Mode Filters

Purpose:

You want to control how long it takes for queries to run when report authors are designing reports in Query Studio and Report Studio. To do this you will begin by adding a design mode filter to one of the query subjects in your model.

Components: Framework Manager, Query Studio

Project: great_outdoors_warehouse

Package: GO Data Warehouse (query)

Task 1. Create a design mode filter.

1. In **Framework Manager**, in the **Project Viewer**, expand **go_data_warehouse > Business view**.
2. Double-click **Sales fact**, and then click the **Filters** tab.
3. Click **Add**, and name the filter **Limited Products Design Mode Filter**.
4. In the **Available Components** pane, expand **Database view**, and then create the following expression:

[Sales and marketing data].[SLS_SALES_FACT].[PRODUCT_KEY] in (30001, 30002, 30003, 30004, 30005)

5. Click **OK**.

Notice that the current Usage for the filter is set to Always.

6. Under the **Usage** column, click the **ellipsis**.
7. In the list, click **Design Mode Only**, and then click the **Test** tab.
8. Click the **Options** link in the bottom-right corner, and under **Design mode filter**, select **Apply all relevant design mode filters when testing**, and then click **OK**.

9. Click **Test Sample**.

A section of the results appears as follows:

Test results (Design Mode filter applied)						
Quantity	Unit cost	Unit price	Unit sale price	Gross margin	Revenue	Gross pro
1172	6.62	12.53	8.77	0.2452	10278.44	2519.8
591	34.97	54.93	52.18	0.3298	30838.38	10171.11
2649	2.9	6.59	6.13	0.5269	16238.37	8556.27
2094	2.9	6.59	6.19	0.5315	12961.86	6889.26
991	0.85	3.66	3.55	0.7606	3518.05	2675.7
467	34.97	54.93	52.73	0.3368	24624.91	8293.92
934	6.62	12.53	8.77	0.2452	8191.18	2008.1
715	15.93	23.8	21.42	0.2563	15315.3	3925.35
663	34.97	54.93	52.18	0.3298	34595.34	11410.23
1035	6.62	12.53	8.77	0.2452	9076.95	2225.25

Notice that the test results have the design mode filter applied. The values retrieved are restricted to the product key specified in the filter.

10. Click **Options**, and clear the **Apply all relevant design mode filters when testing**.
11. Click **OK**, and then click **OK** again.

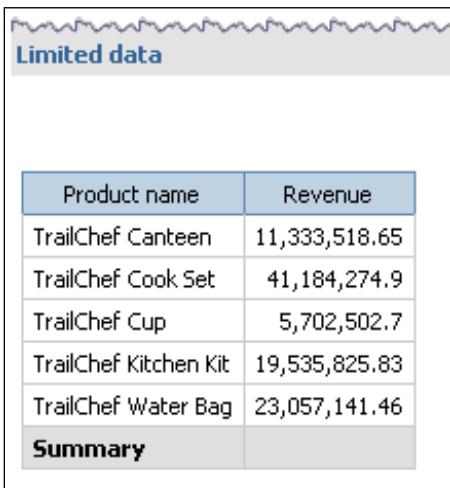
Task 2. Test the design mode filter in Query Studio.

1. Publish the **GO Data Warehouse (query)** package entering **December** and **2010** when prompted for **MONTH** and **YEAR**.
2. Save and Close the project.
3. In **IBM Cognos Connection**, in **Public Folders**, select the **GO Data Warehouse (query)** package, and launch **Query Studio**.
4. Expand **Sales and Marketing (query) > Sales (query)**, and then add the following items to the report:

Query Subject	Query Item
Product	Product name
Sales fact	Revenue

A list report displaying pages of all products and their revenue appears.

5. Under **Menu**, click **Run Report**, and then click **Preview with Limited Data**.
The report appears as shown below:



The screenshot shows a report titled "Limited data". Below the title is a table with two columns: "Product name" and "Revenue". The table contains five rows of data and a summary row at the bottom.

Product name	Revenue
TrailChef Canteen	11,333,518.65
TrailChef Cook Set	41,184,274.9
TrailChef Cup	5,702,502.7
TrailChef Kitchen Kit	19,535,825.83
TrailChef Water Bag	23,057,141.46
Summary	

The design mode filter is applied and the Sales fact Revenue values are restricted to only the products specified in the filter.

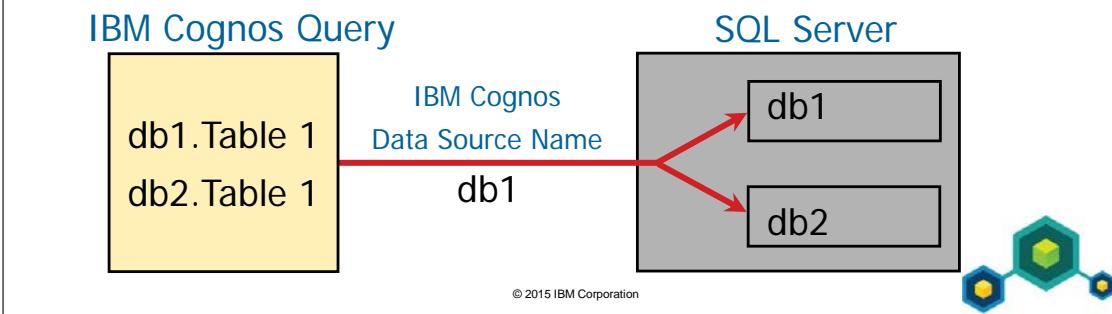
6. Close all browser windows without saving the query.

Results:

You added a design mode filter to one of the query subjects in your model. Doing this lets you control how long it takes for queries to run when a report author is designing a report that uses this query subject.

Reducing Database Connections

- use the same IBM Cognos data source connection name:
 - for all data source connections to the same database server instance
 - to reduce database connections
 - to prevent local processing



Using the same data source name for multiple data source connections to the same database instance allows for fewer database connections at run time and allows the database server to perform the joins between the databases rather than the IBM Cognos servers.

This technique can be implemented in the data source properties in Framework Manager.

In the slide example, the Cognos query is pulling from two tables, each from a separate database in the same SQL Server instance. These two tables have a join defined between them in Framework Manager. Using the same Content Manager Datasource name in the data source properties allows Cognos to make 1 database connection rather than two. It also sends only one Select statement with join criteria to be performed by the database rather than two select statements, which would require local processing for the join condition specified in Framework Manager.

Business Analytics software

IBM

Indicating the Performance Impact of Functions

- provide visual clues about the performance of functions
- can be set at the project or package level

Quality of Service Indicators

	Not Available
	Limited Availability
	Poor Performance
	Unconstrained

© 2015 IBM Corporation

Through a Framework Manager model, report authors can write reports that query any combination of data source types, but not all data sources support functions the same way. The quality of service indicator provides report authors with a visual clue about the performance of individual functions when used in conjunction with the data sources accessed by the model:

- **Not Available** - the function is not available for any data sources
- **Limited Availability** - the function is not available for some data sources in the package
- **Poor Performance** - the function is available for all data sources but may have poor performance in some data sources
- **Unconstrained** - the function is available for all data sources

This lets report authors avoid using functions that could result in long running queries or queries that fail. You can also provide descriptive text about a function.

Summary

- You should now be able to:
 - identify how minimized SQL affects model performance
 - use governors to set limits on query execution
 - identify the impact of rollup processing on aggregation
 - apply design mode filters
 - limit the number of data source connections
 - use the quality of service indicator

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



Working in a Multi-Modeler Environment

IBM Cognos BI

Business Analytics software



© 2015 IBM Corporation

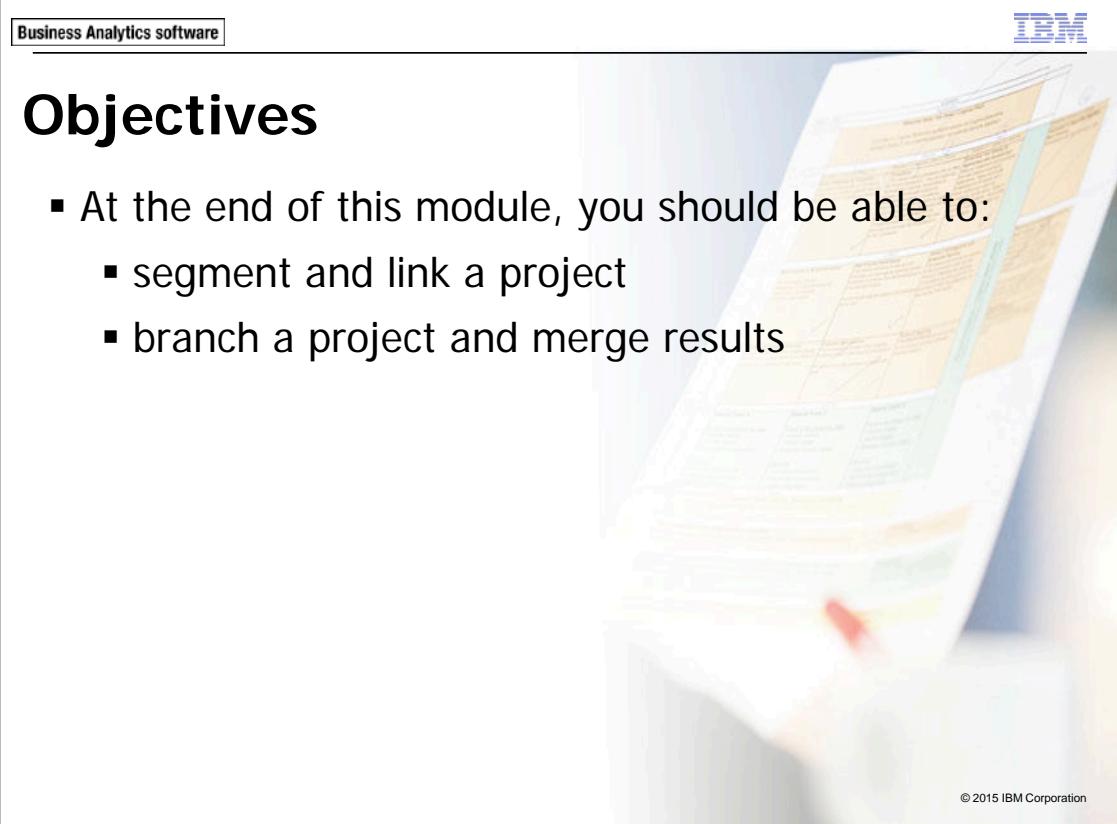
This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Objectives

- At the end of this module, you should be able to:
 - segment and link a project
 - branch a project and merge results



© 2015 IBM Corporation

Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

- User ID: admin
- Password: Education1

Creating a Segment

- creates a new project with its own project files
- use segments to:
 - distribute a project according to business rules and organizational requirements
 - share and re-use project information with other projects

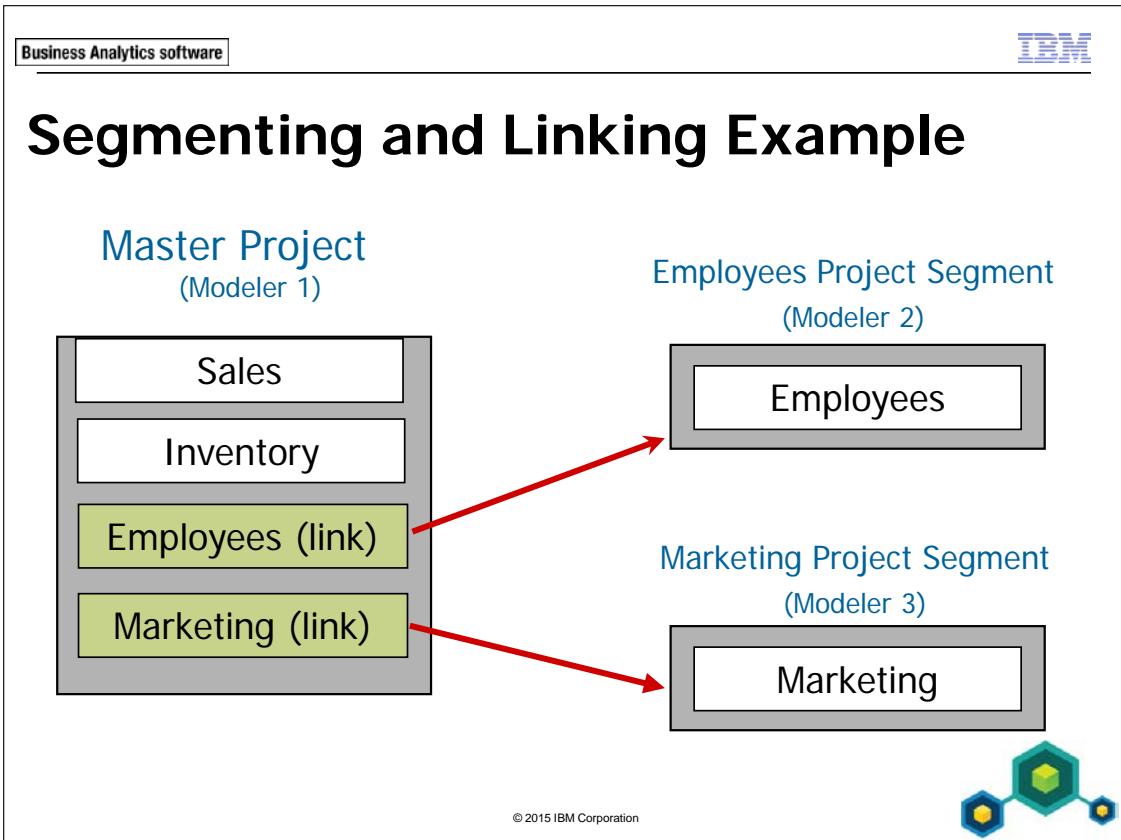
© 2015 IBM Corporation



When you create a segment, you create a new project in a new folder, complete with its own associated project files. The new project is linked by a shortcut in the main project from which it was created. The master project has access to the entire model, including the segments.

You can only segment a project either at the folder or namespace level. You can also link the segments to other projects that contain the same information to maintain consistency and reuse information.

Avoid making changes in both the segment and the main project. If the segment is open when the main project is updated, the potential exists for updates to be lost as a result of overwriting saved changes. We recommend that access to the main project be limited and that you avoid updating segments from the main project. Update the segment from the segmented project.



Modeler 1 can create segments for the Employees and Marketing metadata. These segments are new projects created by Framework Manager and become links in the master project.

Modeler 2 will work on the Employees metadata, to model for predictable results, while modeler 3 will work on the Marketing metadata. The changes they make will be reflected in the master project.

Communication is required between the modelers to ensure that no one overwrites each other's changes. For example, modeler 1 should not make any changes to the Employees or Marketing metadata without talking to the respective segment owner otherwise modeling conflicts might occur and changes may be lost.

The Employees and Marketing segments can also be linked into other projects that may require this type of information.

Regarding source control:

For segmented projects, the segments are simply project directories stored under the parent project directory. There are two ways to work with a segmented project.

1. The segments can be individually opened as stand-alone projects, in which case repository handling is the same as for any other project.
2. Or segments can be opened as part of the main project. In this case you have to 'Check Out' the projects for each segment you intend to modify, which are located, as sub-directories under the main project.

Note: The repository should maintain the same hierarchy as the project directory.

If you do require the Model Synchronization feature, you must 'Check In' any new log files that are created. When you are ready to synchronize, you will need to get a copy of **all** the project log files. It is not necessary to 'Check Out' the log files, as they are never updated after they are first created.

For more information on source control, see Appendix A.

Demo 1: Create a Segment

Purpose:

You want to distribute the project workload between you and another modeler. You want the other modeler to work on employee related data, while you work on other business areas.

Component: **Framework Manager**

Project: **New Project**

Task 1. Create a project.

1. In **Framework Manager**, click **Create a new project**.
 2. In the **Project name** box, type **great_outdoors_warehouse_MASTER**.
 3. In the **Location** box, navigate to **C:\Edcognos\B5A52\Course_Project**, and click **OK**.
- If prompted, log in using admin/Education1.
4. On the **Select a Language** dialog, ensure **English** is selected, and then click **OK**.
 5. Ensure **Data Sources** is selected, and then click **Next**.
 6. Click **GOSALES DW**, and then click **Next**.
 7. Expand **GOSALES DW > Tables**, and then select all tables prefixed with **EMP** and **SLS**.

Tip: You can select the first **EMP_** table, scroll down to the last **EMP_** table, and then Shift+click the last **EMP_** table to select them all. Then repeat for the **SLS_** tables.

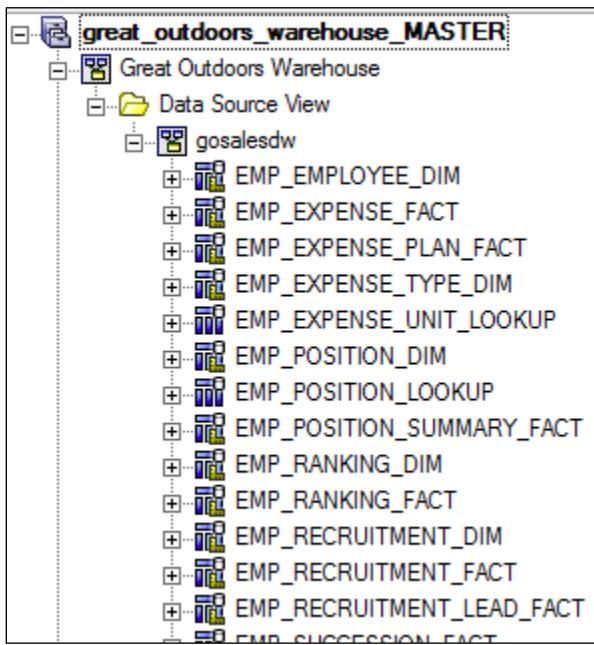
8. Click **Next**, click **Import**, and then **Finish**.

You will rename the root namespace and organize the imported objects.

9. Rename the **GOSALES DW** namespace to **Great Outdoor Warehouse**.
10. In the **Great Outdoor Warehouse** namespace, create a folder called **Data Source View**.
11. In the **Data Source View** folder, create a namespace named **gosalesdw**.

12. In the **Great Outdoor Warehouse** namespace, Shift-click to select all query subjects, and drag them into the **gosalesdw** namespace.

The results appear as follows:



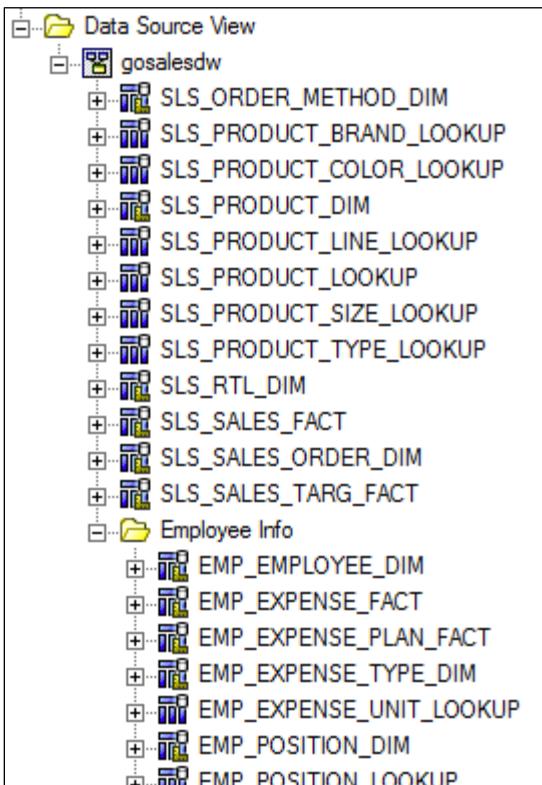
13. Save the project.

Task 2. Create a folder containing objects you would like to segment.

1. In the **gosalesdw** namespace, Shift-click to select all query subjects prefixed with **EMP**.
2. Right-click one of the selected query subjects, point to **New Parent**, and then click **Folder**.
The employee data source query subjects are now contained in the new folder
3. Rename **New Folder** to **Employee Info**.

4. Expand the **Employee Info** folder to view the objects inside.

The results appear as shown below:



5. Save the project.

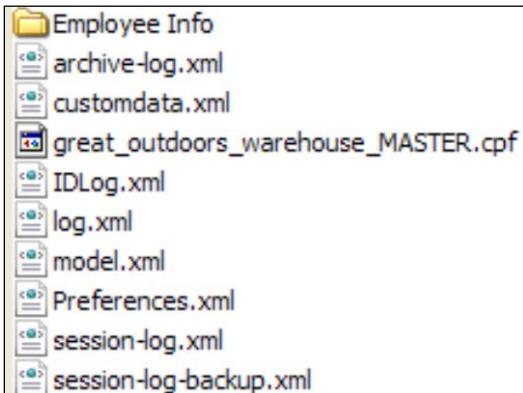
Task 3. Create a segment and view project files.

1. Right-click the **Employee Info** folder, and click **Create Segment**.
2. Accept the defaults and click **OK**.

The Employee Info folder is now represented by a link icon .

3. In **Windows Explorer**, navigate to **C:\Edcognos\B5A52\Course_Project\great_outdoors_warehouse_MASTER**.

The results appear as follows:



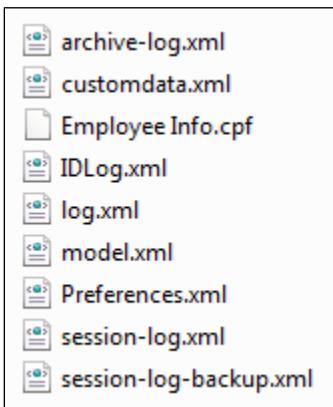
This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

4. Open the **Employee Info** folder.

The results appear as follows:



The folder contains a new set of Framework Manager project files. Another modeler can work on this separate project and have the changes reflected in the main project.

Task 4. Make changes in the segmented project.

1. In **Framework Manager**, close the **great_outdoors_warehouse_MASTER** project, and click **Yes** to save the changes.
2. Open the **Employee Info** project, and in the middle pane, click **Diagram**.
3. In the **Project Viewer**, expand **gosalesdw > Employee Info**.
4. Right-click **EMP_TERMINATION_LOOKUP**, and then click **Locate in Diagram**.
5. In the **Diagram** pane, double-click the relationship between **EMP_TERMINATION_LOOKUP** and **EMP_EMPLOYEE_DIM**, change the cardinality of **EMP_EMPLOYEE_DIM** to **1..1**, and then click **OK**.
6. In the **Project Viewer**, expand **EMP_EMPLOYEE_DIM**, and click **TERMINATION_CODE**.
7. In the **Properties** pane, change the **Usage** property to **Attribute**.
8. **Save** and **Close** the project.

Task 5. View changes in main project.

1. Open the **great_outdoors_warehouse_MASTER** project.
2. In the **Project Viewer**, expand **Great Outdoors Warehouse > Data Source View > gosalesdw > Employee Info > EMP_EMPLOYEE_DIM**, and select **TERMINATION_CODE**.
Notice that the Usage property is now set as an Attribute.
3. Right-click **EMP_TERMINATION_LOOKUP**, and click **Launch Context Explorer**, and then click the **Show Related Objects** button if necessary.
Notice the relationship cardinality to **EMP_EMPLOYEE_DIM** is 1..1, which reflects the change you made in the previous task.
4. Close the **Context Explorer**, and then close the project.

Results:

You created a segment to allow another modeler to work on only part of the project. Changes made to the segment were reflected in the main project.

Creating a Link

- you can only create links to:
 - folders
 - namespaces
 - projects
- linking to an existing project lets you:
 - organize work across large projects
 - maintain consistency
 - re-use information

© 2015 IBM Corporation



A link is a shortcut to an existing project, folder, or namespace. You must create the project, folder or namespace before you can link to it.

Use a link when you have modeled metadata in an existing project and want it available in another project. Use a segment when you want to distribute work to other modelers from a main project or share portions of a project with other projects. Essentially, once you have created a link or segment, they behave the same. They are a shortcut in one project to another project.

Links and segments can be difficult to manage. Use them appropriately. For example if a master project consists of multiple links or segments, consider creating a separate model for each.

Demo 2: Create a Link

Purpose:

You will create a new project that will model the marketing metadata of the Sample Outdoors Company. This project can also be leveraged in the great_outdoors_warehouse_MASTER project, so you will create a link to it.

Component: **Framework Manager**

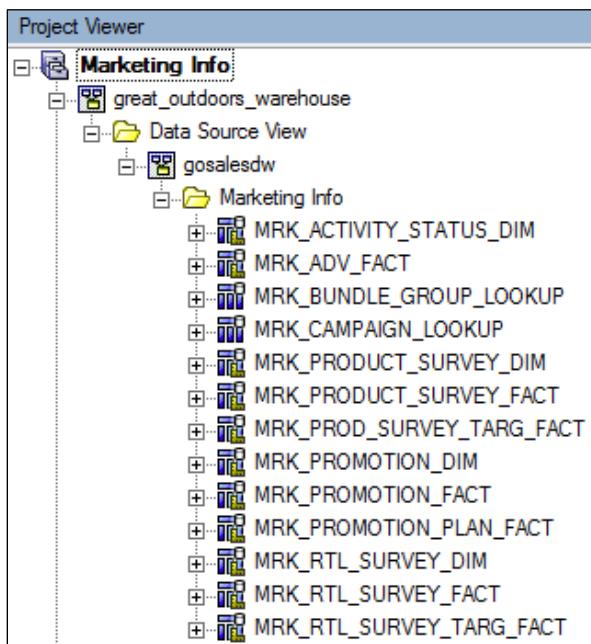
Project: **New Project**

Task 1. Create a New Project for marketing information.

1. In **Framework Manager**, click **Create a new project**.
2. In **Project name**, type **Marketing Info**.
3. In **Location**, navigate to the **Course_Project** folder, and then click **OK**.
If a message displays regarding the structure of the project, click **OK**.
4. On the **Select a Language** dialog, ensure **English** is selected, and then click **OK**.
5. Ensure **Data Sources** is selected, and then click **Next**.
6. Click **GOSALES DW**, and then click **Next**.
7. Expand **GOSALES DW > Tables**, and then select all tables prefixed with **MRK**.
You need to create the same namespace parent in this project as in the main great_outdoors_warehouse_MASTER project to link in the marketing metadata.
8. Click **Next**, click **Import**, and then click **Finish**.
9. Rename the **GOSALES DW** namespace to **great_outdoors_warehouse**.
10. In the **great_outdoors_warehouse** namespace, create a folder named **Data Source View**.
11. In the **Data Source View** folder, create a namespace named **gosalesdw**.
12. In the **gosalesdw** namespace, create a folder called **Marketing Info**.

13. In the **great_outdoors_warehouse** namespace, Shift+click to select all of query subjects, and then drag them into the **Marketing Info** folder.

The results appear as shown below:



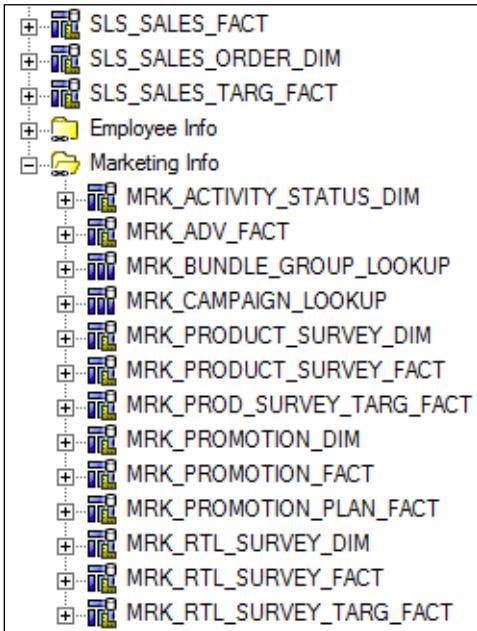
14. **Save** and **Close** the project.

Task 2. Create a link in the main project.

1. Open the **great_outdoors_warehouse_MASTER** project.
2. Expand **Great Outdoors Warehouse > Data Source View**, and right-click the **gosalesdw** namespace, and then click **Link Segment**.
If a message appears regarding the structure of the project, click Yes.
3. Browse to **C:\Edcognos\B5A52\Course_Project\Marketing Info**, select the **Marketing Info.cpf** file, and then click **Open**.
4. Click **OK** to the message.
5. Expand **great_outdoors_warehouse > Data Source View > gosalesdw**, and then click **Marketing Info**.
6. Click **Add**, and then click **OK**.

7. Expand gosalesdw and Marketing Info.

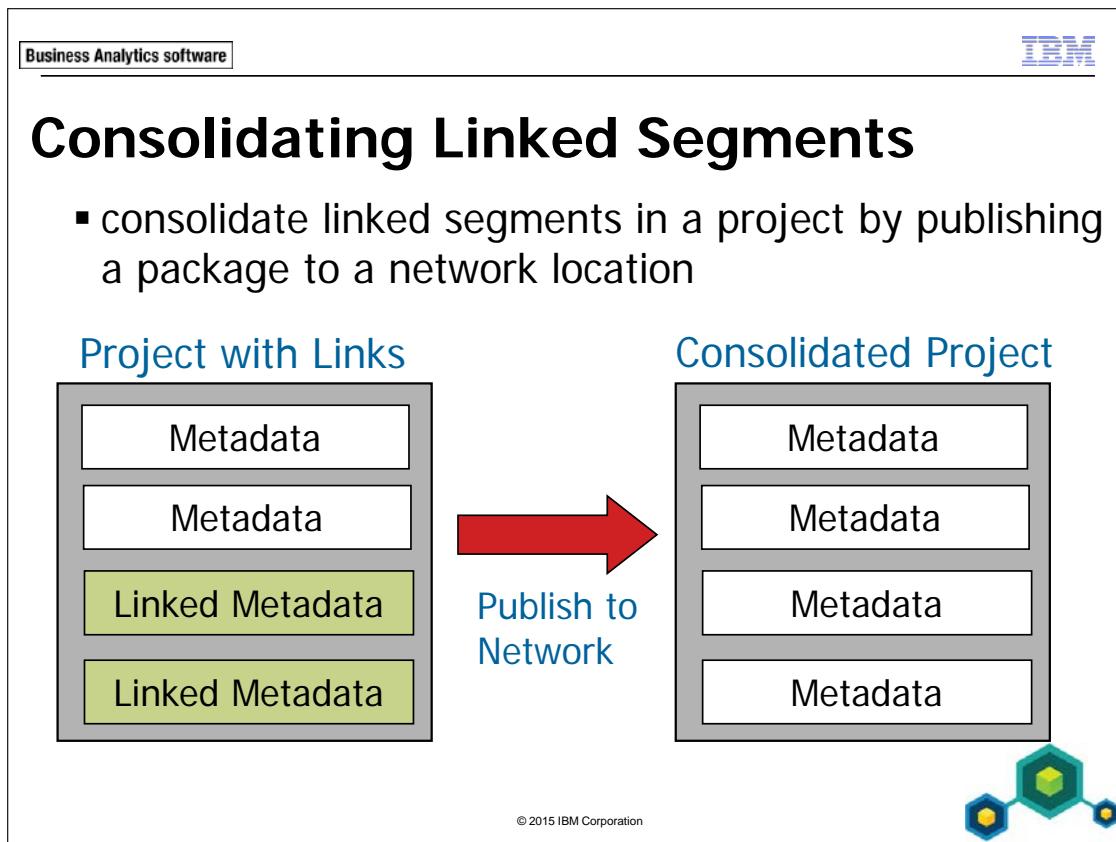
The Marketing Info project is now linked in the great_outdoors_warehouse_MASTER project as shown below:



8. Save the project, and leave it open for the next demo.

Results:

You successfully linked the Marketing Info project in the great_outdoors_warehouse_MASTER project.



The new package published to the network will include all the metadata specified by the modeler without any linked segments.

This process is useful when a multi-modeler environment is no longer required and you wish to consolidate your metadata into one project. This process is also useful if you would like assistance troubleshooting your model from support or other modelers. Rather than sending the master project and all the linked projects, you can send one consolidated model.

Demo 3: Consolidate Linked Segments

Purpose:

You would like to consolidate your multi-modeler environment by creating a single project that contains all the metadata without links to other projects.

Component: **Framework Manager**

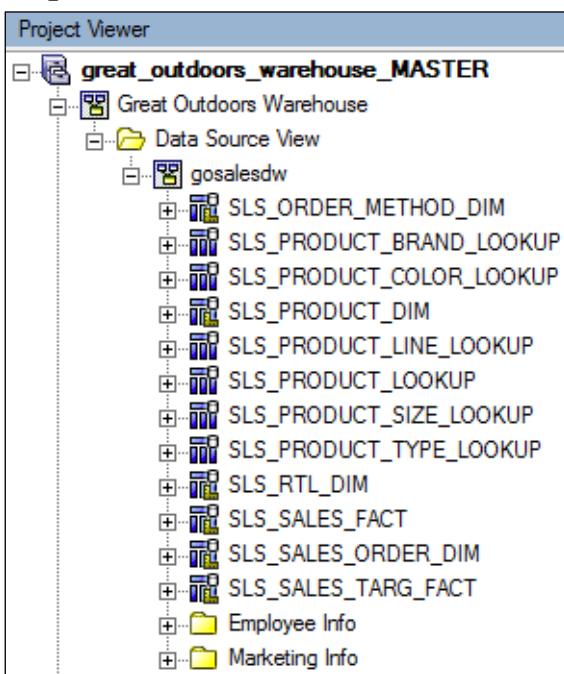
Project: **great_outdoors_warehouse_MASTER**

Task 1. Publish a package to the network.

1. In the **great_outdoors_warehouse_MASTER** project, create a new package named **great_outdoors_warehouse_FINAL**.
2. Click **Next**, and then click **Finish**.
3. Click **Yes** to use the **Publish Package Wizard**.
4. On **Publish Wizard - Select Location Type**, select the **Location on the network** radio button, and click the **Network location** folder, and then browse to **C:\Edcognos\B5A52\Course_Project**.
5. Create a new folder called **great_outdoors_warehouse_FINAL**, double-click it, and then click **OK**.
6. Click **Next** twice, and then click **Publish**.
7. Click **Finish**, and then **Save** and **Close** the project.

Task 2. Examine the consolidated project.

1. Click **Open a project**, navigate to **C:\Edcognos\B5A52\Course_Project\great_outdoors_warehouse_FINAL**, and open the **great_outdoors_warehouse_FINAL** project.
2. Expand **Great Outdoors Warehouse > Data Source View > gosalesdw**.



Notice that the Employee Info and Marketing Info folders are no longer links. All metadata is now contained within this one project. Also notice that the project maintains the original project name, **great_outdoors_warehouse_MASTER**. However the .cpf name is identified in the title bar, **great_outdoors_warehouse_FINAL**. To avoid confusion, it is recommended to rename the project to match the .cpf file.

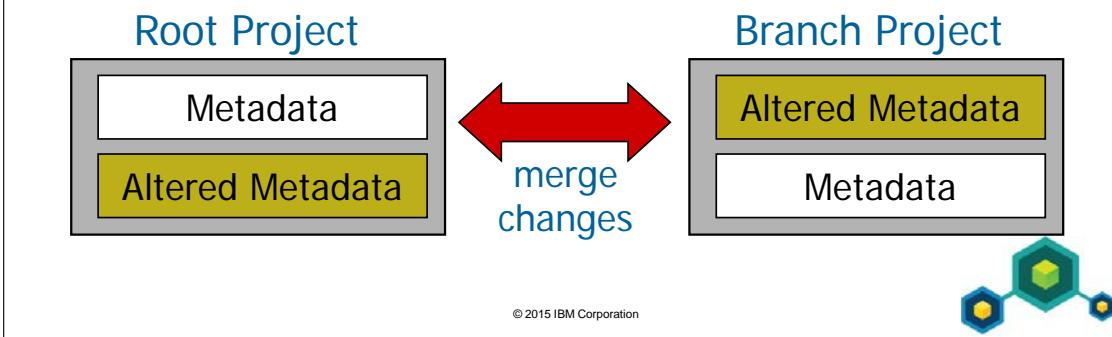
3. Rename the root namespace to **great_outdoors_warehouse_FINAL**, and then **Save** the project.

Results:

By publishing the entire great_outdoors_warehouse_MASTER project to the network, you consolidated all linked metadata into one project.

Branching a Model

- enables parallel modeling
 - creates a copy of the project for simultaneous development on any portion of the model
- unlimited branches
- merge changes (bi-directional)



The Branch/Merge feature essentially executes the scripts from one copy of a project onto another copy. Unlike segmenting and linking, which allow modelers to work on individual segments of a model independently, branching and merging allow multiple modelers to work on the same model in its entirety at the same time. To do this, the project owner makes a copy of the root project, called a branch. A team member can modify the branch as required, independently of the root project.

Branches can be merged back into the root project or from the root project into the branch as required. Conflicts between the root project and a branch are resolved during the merge process. However, only the root (master) version of the model can be the subject of Source Control Repositories (for example, VSS and CVS). Modelers should be in constant communication to ensure their modeling actions will not affect other branches or the root project.

Project branching is compatible with segmenting and linking.

Demo 4: Branch a Model, Make Changes and Merge Results

Purpose:

You would like to allow another modeler to work on your model at the same time as you. Use the project branching feature in Framework Manager to allow a second modeler to work on a branch of your project, and then merge the changes with your root project.

Component: Framework Manager

Project: great_outdoors_warehouse_FINAL

Task 1. Create a Branch of the root model as Modeler 1.

This task is completed as modeler 1.

1. In the **great_outdoors_warehouse_FINAL** project, click the **Project** menu, and then click **Branch to**.
2. Change the branch project name to **great_outdoors_warehouse_FINAL Branch 1**, and then click **OK**.
3. **Save** and then **Close** the **great_outdoors_warehouse_FINAL** project.

Task 2. Modify the branch model as Modeler 2.

This task is completed as modeler 2.

1. Open the **great_outdoors_warehouse_FINAL Branch 1** project.
2. Expand **Great Outdoors Warehouse > Data Source View > gosalesdw > SLS_PRODUCT_DIM**.
3. Change the **Usage** property for the following items from **Fact** to **Attribute**:
 - PRODUCT_TYPE_KEY
 - PRODUCT_NUMBER
 - BASE_PRODUCT_KEY
 - BASE_PRODUCT_NUMBER
 - PRODUCT_BRAND_KEY
4. Right-click **SLS_RTL_DIM**, and click **Launch Context Explorer**, and then if necessary, click the **Show Related Objects** button.
5. Double-click the relationship between **SLS_RTL_DIM** and **SLS_SALES_FACT**.

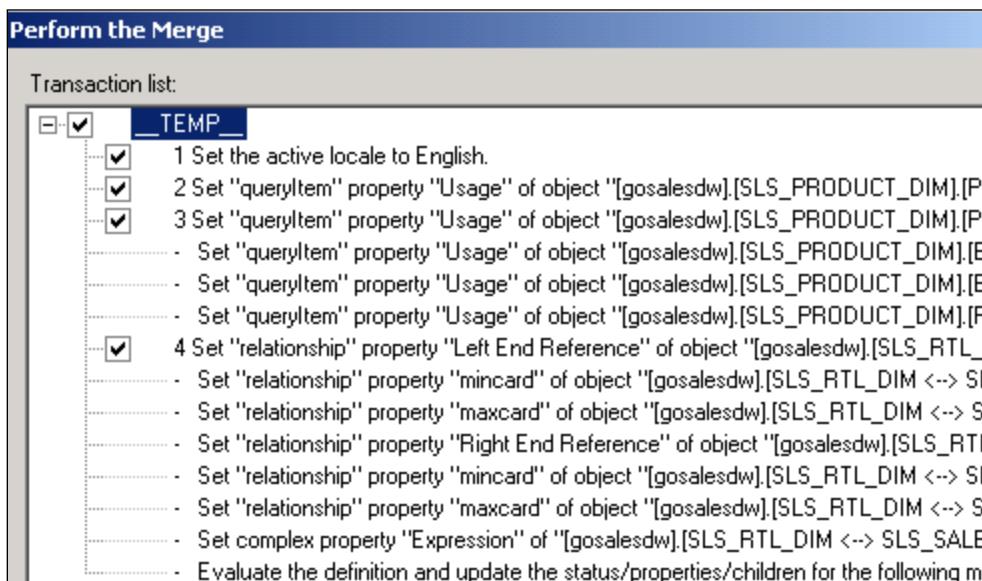
6. Change the cardinality for **SLS_SALES_FACT** to **0..n**.
7. Click **OK**, and then close the **Context Explorer**.
8. **Save** and **Close** the project.

Task 3. Merge the branch into the root project as Modeler 1.

This task is performed as modeler 1.

1. Open the **great_outdoors_warehouse_FINAL** model.
2. From the **Project** menu, select **Merge from**.
3. Navigate to **C:\Edcognos\B5A52\Course_Project\great_outdoors_warehouse_FINAL Branch 1**, click **great_outdoors_warehouse_FINAL Branch 1.cpf**, and then click **Open**.

The results appear as follows:



You are presented with a transaction list. You have the option to clear check boxes for transactions that you do not want updated in the root model. In your case you will accept all the changes. You can also choose to step through each transaction and then pause, or simply run all transactions.

4. Click **Run**.

All transactions are successfully merged into the root project. A backup project is created on the file system in the same location as the root project with a data timestamp. This allows you to revert back to the previous version if required.

After the branch project is merged, it will still exist on the file system. If it is no longer required, it is recommended that you delete the branch project after it is merged into the root project. New branches can be created as required.

5. Click **Accept**.
6. In the **great_outdoors_warehouse_FINAL** project, verify that the branch project changes appear.
7. **Save** and **Close** the project.

Results:

Using the project branching feature in Framework Manager, you were able to apply changes in a branched project and then merge those changes into the root project.

Summary

- You should now be able to:
 - segment and link a project
 - branch a project and merge results

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

21-23

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



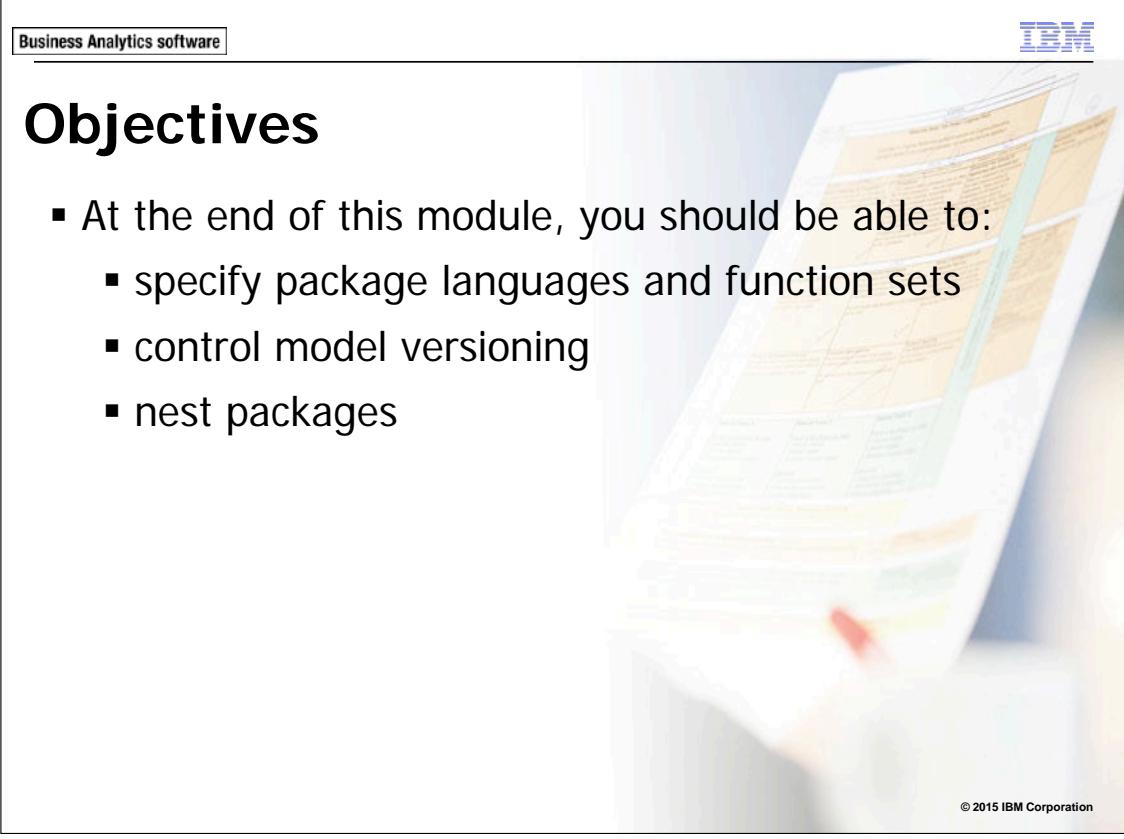
This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Objectives

- At the end of this module, you should be able to:
 - specify package languages and function sets
 - control model versioning
 - nest packages



© 2015 IBM Corporation

Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

- User ID: admin
- Password: Education1

Business Analytics software



What is a Framework Manager Package?

- A subset of the project metadata to be published to IBM Cognos Connection
- You can:
 - create different packages to meet different requirements
 - apply security to packages to restrict access
 - include other packages in a package (nesting)

© 2015 IBM Corporation



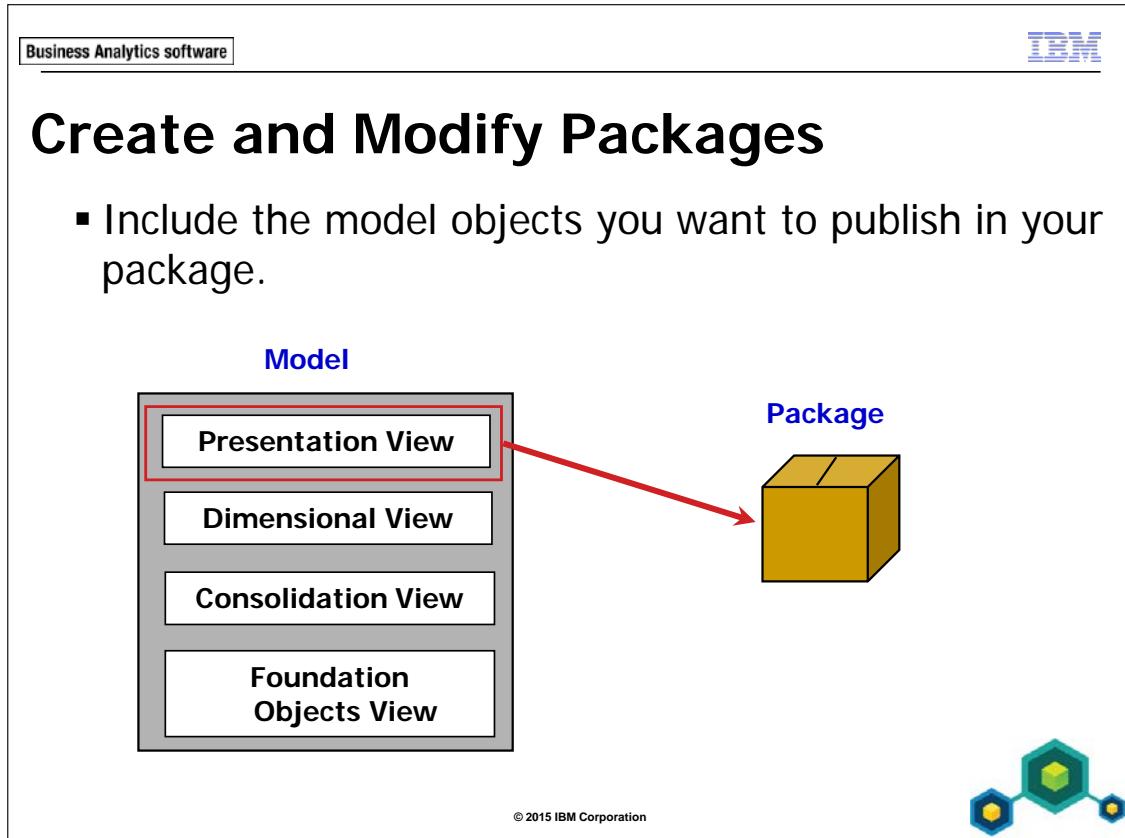
Each package can contain a different set of folders, filters, query subjects, and query items. You can customize its contents to satisfy different reporting requirements and to set up a logical presentation of the metadata. Each report can only contain information from a single package.

Packages can be referenced by other packages. This is known as nesting, which can save development and maintenance time. Nesting is discussed later in this module.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Create and Modify Packages

- Include the model objects you want to publish in your package.



At any time, you may go back and edit the definition of your package by adding or removing objects. You must exercise caution when doing this as you may break reports based on a previous version of the model.

When you create a package, you can choose which model objects will be included, excluded, or hidden, based on the requirements of reports authors. These three options are described as follows:

- **Select** - the object and its child objects can be used in reports by report authors.
- **Hide** - the object and its child objects cannot be used by report authors, but objects available to report authors that reference the hidden objects can. With this option, you will not receive informational messages when publishing your package stating that underlying objects will be published because other objects in the package reference them.
- **Unselect** - the object and its child objects are not published. It cannot be used for reports and cannot be selected by report authors. If other objects in the package reference the unselected object, the object will be included and hidden in the package and you will receive informational messages stating this.

Languages and Function Sets

- Specify which languages are to be included in a package.
 - Each language added will add additional metadata that has been translated.
- Specify which function sets to include in a package.
 - Add vendor specific function sets to match the data sources used in a package.

© 2015 IBM Corporation



In the case where you have modeled for a multilingual audience, you can specify the languages to be published with a package. You must add languages to the project before you can add them to a package, and must translate the metadata for the model objects.

If you have multiple data sources in your model that are heterogeneous, publish your package with the appropriate function sets so that report authors can leverage them while authoring reports.

You can create several packages based on the same model, that all have different languages. For example, you may have one package designed for the Mexican sales offices. This package may include Spanish and English; whereas the Canadian sales offices package would include French and English.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

Publish Packages

- When you publish a package, you can choose to:
 - publish to the IBM Cognos content store or to a network location
 - externalize query subjects
 - verify the package

© 2015 IBM Corporation



When you publish a package to the IBM Cognos server, you make it available to report authors with the appropriate security rights. Publishing to the network lets you back up or share all or a portion of your model with other metadata modelers. To avoid potential problems, select the verify package option in the Publish Wizard to check for errors.

You can choose to externalize query subjects to make metadata and data available from the underlying data source available for use in other applications.

Business Analytics software

IBM

Set Model Version Control

- Select the number of versions of the model you want to retain on the IBM Cognos server.

IBM Cognos Connection

GO Operational Folder/Package

First Publish Second Publish Third Publish

Report

© 2015 IBM Corporation

Model versioning lets you maintain multiple versions of the model in IBM Cognos. This lets you change the model without immediately affecting existing reports. Existing reports can still use the original version of the model, while authors test and repair any damaged reports.

For example, if the number of model versions to retain is set to 2 and you create a report based on the first publish of the package, that report will continue to use the first version of the model, even after the second publish. However, on the third publish, the report will use the latest version. Any reports created or modified by report authors will use the most recent version of the model.

If you disable model versioning, there will only ever be one version of the model on the IBM Cognos server. When you modify a report and a more recent model exists than the one the report was created against, the report author is notified that the report must be updated to the most recent model. To do so, the report author must save the report. The version of the model used by the report is saved in the report specification.

Demo 1: Control Model Versions

Purpose:

The Sales and Marketing (conformed) package is only required in a few regions. Therefore, you will reduce the number of languages published with the project to those required for the regions. You will also reduce the number of function sets published with the package, to one required for the database vendor they are using.

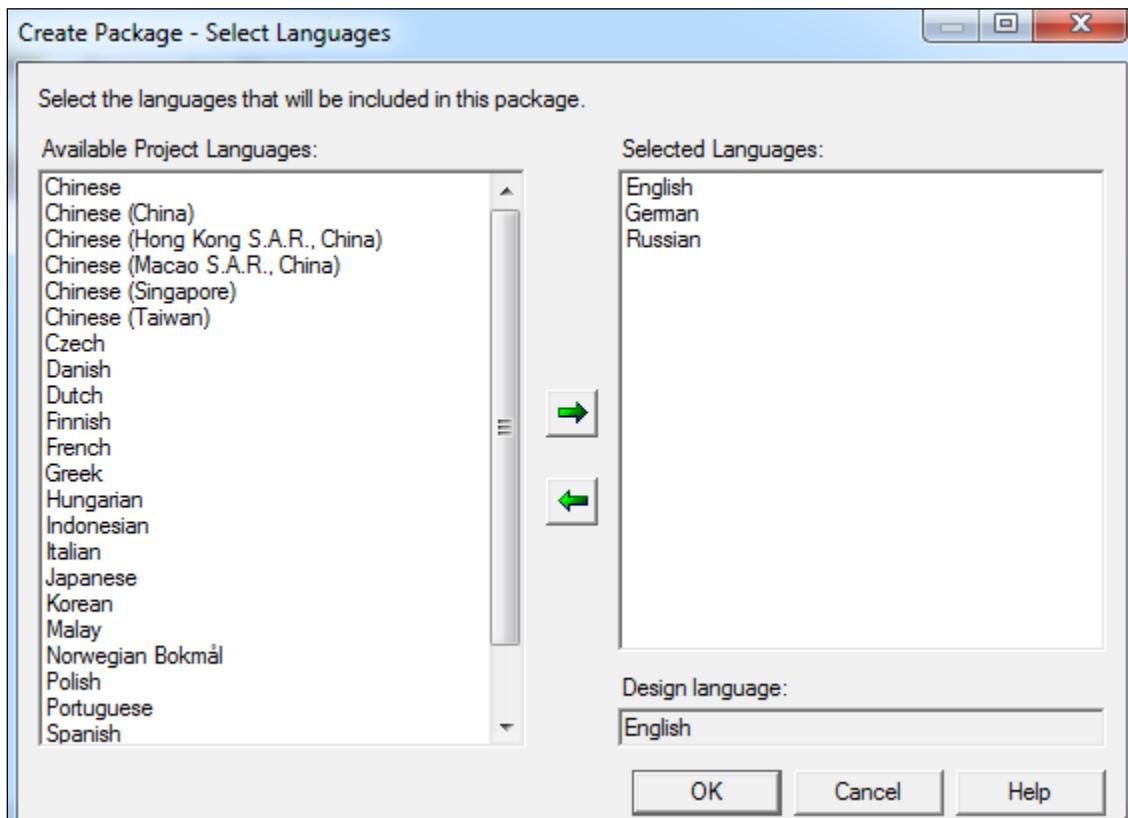
You will also enable model versioning so that authored reports can continue to run existing reports, even when the model has changed. This will allow authors to test and fix reports. You will also enable model versioning and test the results.

Components: Framework Manager, IBM Cognos Workspace Advanced
Project: great_outdoors_warehouse
Package: Sales and Marketing (conformed)

Task 1. Specify package languages and functions.

1. In Framework Manager, open the great_outdoors_warehouse project located at C:\Edcognos\B5A52\CBIFM-Start Files\Module 22\great_outdoors_warehouse.
If prompted, log in using admin/Education1.
2. In the Project Viewer pane, expand the Packages folder, and then click Sales and Marketing (conformed).
This package is used primarily in Germany, Canada and Russia for drill through from dimensional data sources. You will limit the languages published with this package for the appropriate regions.
3. From the menu, click Actions > Package> Specify Package Languages.

4. In the **Selected Languages** pane, remove all languages except **English**, **German**, and **Russian** using the left pointing green arrow button. The results appear as follows:



Note: By default, all languages available in a project will be added to a package when you create it. If languages are added to the project after you create the package, you must add them to the package in order to include them.

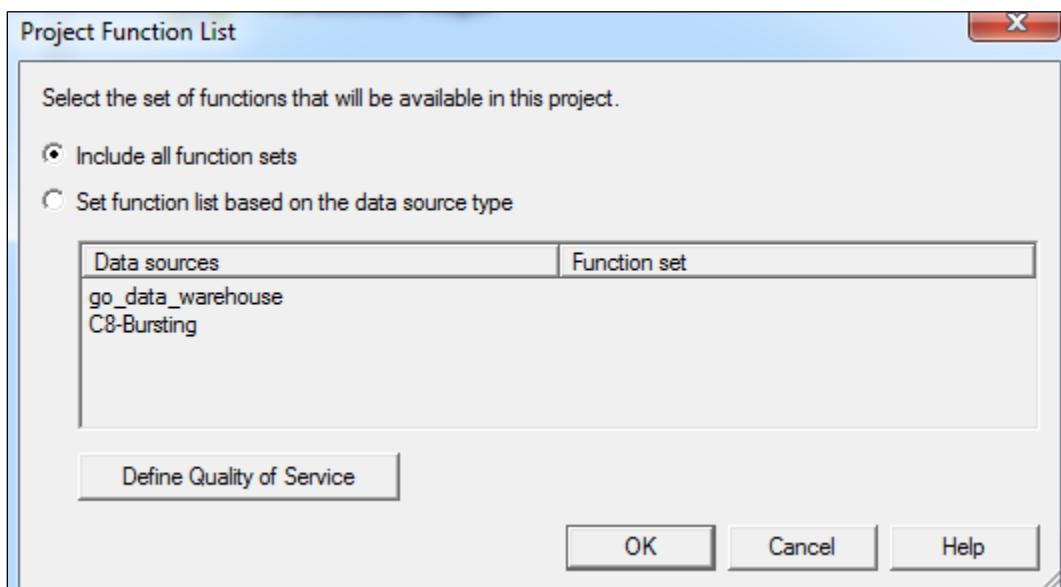
You can specify languages for several packages at a time. To do this, click the Packages folder, and click Actions > Package > Specify Package Languages. From the Packages sub menu, you can also click Explore Packages to view the contents of each package and any object security specified.

5. Click **OK**.

These regions use a DB2 vendor, so you will limit this package to the DB2 function set. First you will examine where you can define which function set is associated with a data source.

- From the **Project** menu, click **Project Function List**.

The results appear as follows:

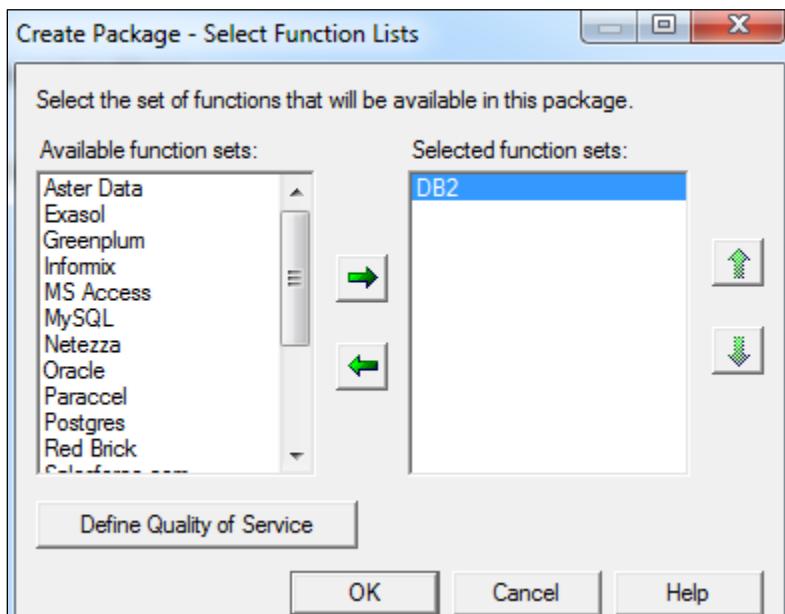


The Project Function List dialog controls the functions sets that will appear when you create a new package. You can choose to include all function sets, or define a function set for each data source in the project. If you choose the second option, only defined function sets will be selected for your package, based on the data sources referenced in the package.

- Click **Set function list based on the data source type**.
 - On the **go_data_warehouse** row, click in the **Function set** column.
A list of database-specific function sets appears.
 - Click **DB2**, and then click **OK**.
- When this package was initially created, the setting was to include all function sets. You will configure this package appropriate to your data source.
- From the menu, click **Actions > Package > Specify Package Function List**.

11. Remove all function sets except **DB2**.

The results appear as follows:



12. Click **OK**.

The next time you publish the Sales and Marketing (conformed) package, it will be reduced in size since it will contain less multilingual metadata and fewer function sets.

Task 2. Enable model versioning.

To ensure that authors have time to fix their reports when changes are made to the model, you will enable model versioning.

1. Right-click the **Sales and Marketing (conformed)** package, and then click **Publish Packages**.
2. Under **Select publish location**, select the **Enable model versioning** check box, and then in the **Number of model versions to retain** box, type **2**.
3. Select the **Delete all previous model versions** check box.
4. Click **Next**, twice.

Deleting all previous versions ensures that you are working with a fresh version of the package. You can use this to force reports and analyses to use the latest version of the model.

5. Click **Publish**.

If you are prompted, specify a Month and Year value (i.e. May, 2010).

6. Click **Finish**.

Task 3. Create a report based on the Sales and Marketing (conformed) package.

1. Open **IBM Cognos Connection**, and launch **IBM Cognos Workspace Advanced**.
2. Select the **Sales and Marketing (conformed)** package, and create a new **List** report.
3. In the **Source** menu, expand **Sales and Marketing (conformed) > Products > Products**, and then drag **Product** (the last level) to the report.
Ensure that you select the Product level, rather than the Products level. You will rename the Product level name later, in order to make it more meaningful.
4. From **Measures > Sales fact**, add **Revenue** and **Planned revenue** to the report.

A section of the results appear as follows:

Product	Revenue	Planned revenue
TrailChef Water Bag	23,057,141.46	28,395,176.52
TrailChef Canteen	11,333,518.65	12,561,922.23
TrailChef Kitchen Kit	19,535,825.83	20,626,722.2
TrailChef Cup	5,702,502.7	6,632,370.18
TrailChef Cook Set	41,184,274.9	44,700,935.4

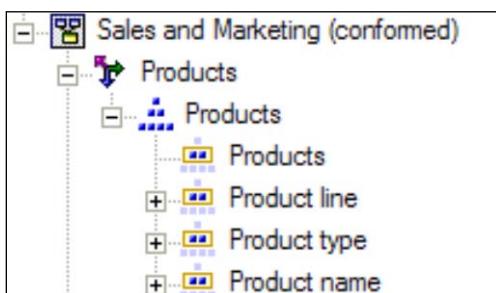
5. On the toolbar, click **Save** , in the **Name** box, type **Model Versioning Test**, and then click **Save**.
6. **Close** the report, but leave **IBM Cognos Connection** open.

Task 4. Modify the project and re-publish the Sales and Marketing (conformed) package.

1. In **Framework Manager**, expand **go_data_warehouse > Sales and Marketing (conformed) > Products > Products**.

2. Rename the **Product** level to **Product name**.

The results appear as follows:



3. **Save** the project.
4. Right-click the **Sales and Marketing (conformed)** package, and select **Publish Packages**.
5. Use the same options as before, without selecting the **Delete previous model versions** check box.
6. Click **Next** twice, and then click **Publish**.

A message states that a previous version of the model already exists, and prompts you to add an additional version.

7. Click **Yes**, and then click **Finish**.

You now have two versions of the Sales and Marketing (conformed) package on the IBM Cognos server.

Task 5. Run the saved report.

1. In **IBM Cognos Connection**, click **Sales and Marketing (conformed)**. You might have to click Refresh to see the report link.
2. Beside **Model Versioning Test**, click **Run with options - Model Versioning Test** .
3. Click **Run**.
This report has run against the original version of the package, which is different from the package you just published.
4. Click **Return**.

Task 6. Re-publish the package and test the saved report.

1. In **Framework Manager**, publish the **Sales and Marketing (conformed)** package using the default options in the Publish Wizard.
A message box appears stating that two model versions already exist, and that publishing will overwrite one of the versions.
2. Click **Yes**, and then click **Finish**.
3. In **IBM Cognos Connection**, click **Sales and Marketing (conformed)**, and then click the **Model Versioning Test** report.

An error message appears:



4. Click **Details**.

The message indicates that

QE-DEF-0359 The query contains a reference to at least one object '[Sales and Marketing (conformed)].[Products].[Products].[Product]' that does not exist.

This is because the model version that the report is based on was removed when you published the package the third time.

5. Click **OK**.
6. Leave **IBM Cognos Connection** and **Framework Manager** open for the next demo.

Results:

You reduced the number of languages and function sets published with the Sales and Marketing (conformed) package, and used model versioning to allow reports to run against different versions of the model.

Business Analytics software

IBM

Nest Packages

- When you create nested packages, you create a master package that is based on other packages.

Framework Manager Model Packages

```

graph TD
    MP[Master Package] --- NA[North America]
    MP --- M[Mexico]
    MP --- US[United States]
    MP --- C[Canada]
  
```

© 2015 IBM Corporation

Use nested packages to reuse model information. Nested packages save time, are easier to maintain, and let you publish only the master package to make all referenced packages available to report authors.

You create three separate packages named Canada, Mexico, and the United States. Each package contains the project objects and security appropriate for that package. You can create one master North America package and include the packages Canada, Mexico, and the United States. When you need to publish the package for report authors, you publish only the North America package.

When users from any of the three groups log on to IBM Cognos Connection and begin to author a report, they will only see the package objects that apply to them in the metadata tree, which was defined in the security settings.

Demo 2: Nest Packages

Purpose:

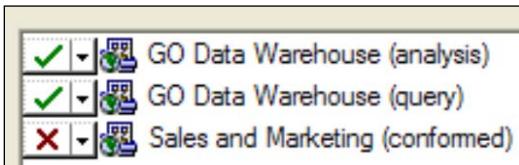
One of the quality assurance teams that tests your packages wants to combine the GO Data Warehouse (query) and GO Data Warehouse (analysis) packages into one package for testing and comparison purposes. You will use a nested package to fulfill this request.

Components: Framework Manager, IBM Cognos Workspace Advanced
 Project: great_outdoors_warehouse
 Package: **GO Data Warehouse (query and analysis)**

Task 1. Create a nested package.

1. In **Framework Manager**, right-click **Packages**, and click **Create > Package**.
2. In **Name**, type **GO Data Warehouse (query and analysis)**, and click **Next**.
3. In the **Define Objects** dialog box, select **Using existing packages**.
4. Select the **GO Data Warehouse (analysis)** and the **GO Data Warehouse (query)** packages.

The results appear as follows:



5. Click **Next**, and then click **Finish**.
6. Click **Yes**, to open the Publish Wizard.
7. Deselect **Enable model versioning**, and click **Next** twice.
8. Clear the **Verify the package before publishing** check box, click **Publish**, and then click **Finish**.
9. **Save** the project.

Task 2. View the package in IBM Cognos Workspace Advanced.

1. In **IBM Cognos Connection**, launch **IBM Cognos Workspace Advanced**, and click **Create New**.
2. Click the ellipsis next to **Package**, select **Public Folders > GO Data Warehouse (query and analysis)**, and click **OK**.
3. Double-click **List**.

The results appear as follows:



Quality assurance staff now has access to both the query and analysis views of the metadata, and can quickly perform tests on either view without having to switch between multiple packages.

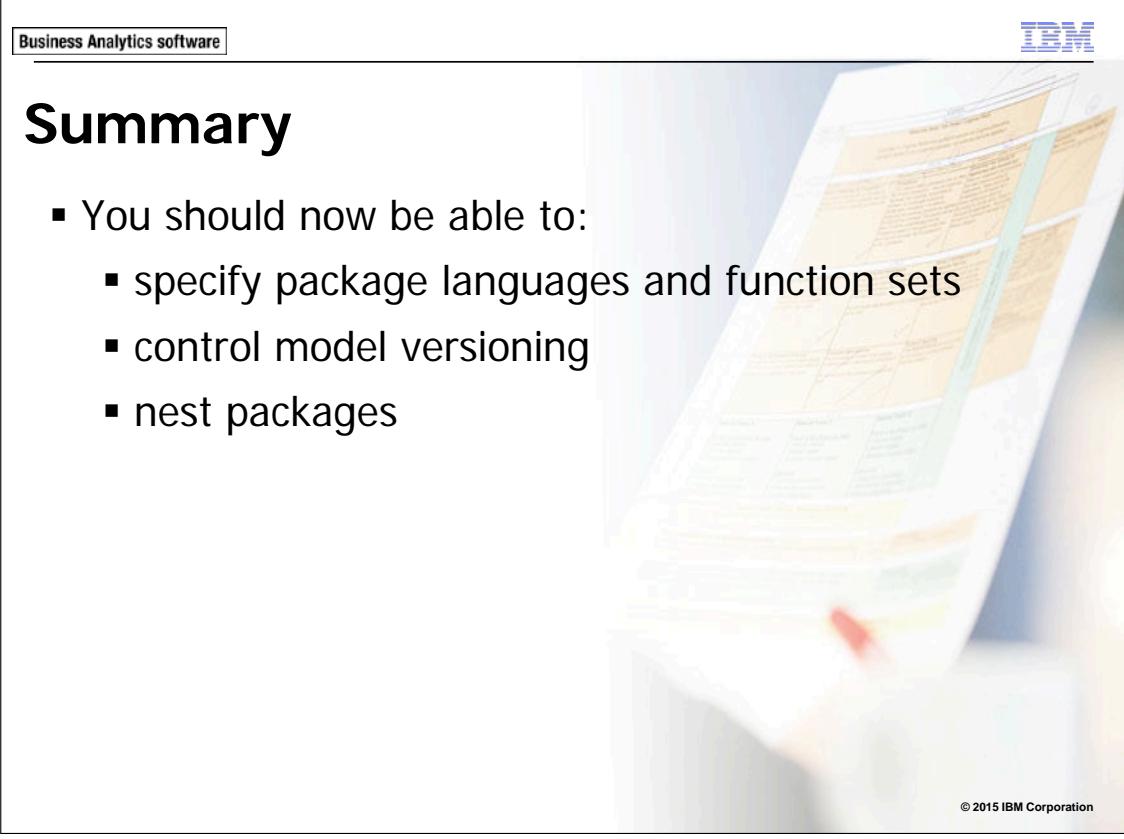
4. **Close** all browser windows, without saving changes.
5. **Close** Framework Manager, saving the project if prompted.

Results:

By nesting existing packages, you created a package that re-used two different presentation views of the metadata in the project for use in comparative testing by the quality assurance team.

Summary

- You should now be able to:
 - specify package languages and function sets
 - control model versioning
 - nest packages



© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

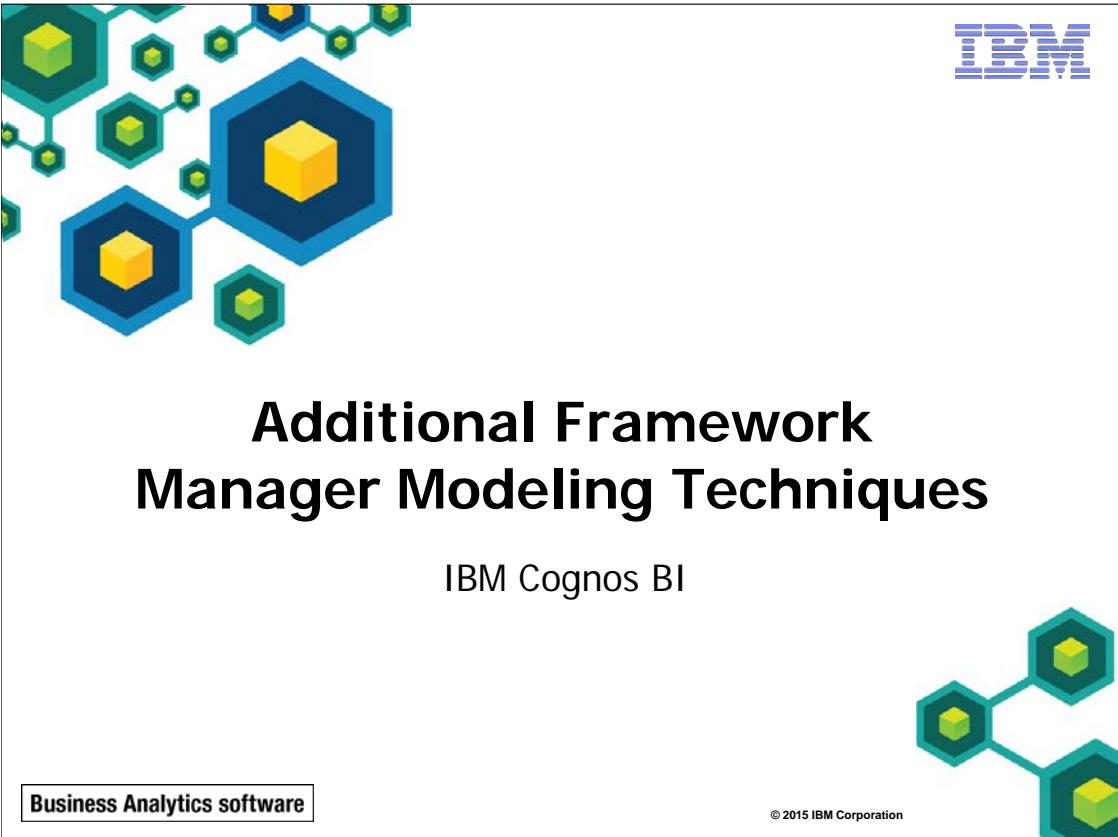
This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

22-19

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



The advertisement features a white background with a decorative border composed of teal hexagonal icons containing yellow cubes. In the top right corner, the IBM logo is displayed in its signature blue font. The main title, "Additional Framework Manager Modeling Techniques", is centered in a large, bold, black sans-serif font. Below the title, the text "IBM Cognos BI" is centered in a smaller, regular black font. At the bottom left, a small rectangular box contains the text "Business Analytics software". At the bottom right, a copyright notice reads "© 2015 IBM Corporation".

Additional Framework Manager Modeling Techniques

IBM Cognos BI

Business Analytics software

© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

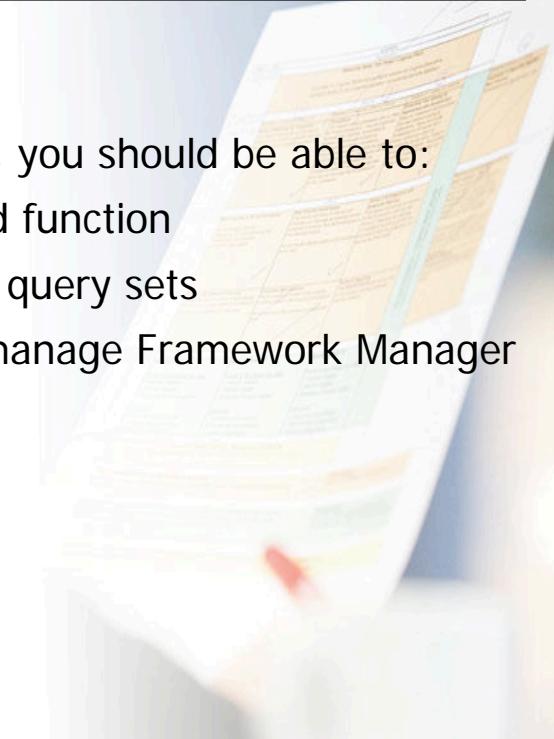
Business Analytics software

IBM

Objectives

- At the end of this module, you should be able to:
 - leverage a user defined function
 - identify the purpose of query sets
 - use source control to manage Framework Manager files

© 2015 IBM Corporation



Unless otherwise specified in demo or workshop steps, you will always log on to IBM Cognos in the Local LDAP namespace using the following credentials:

- User ID: admin
- Password: Education1

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Business Analytics software

IBM

Leveraging User-Defined Functions

- User-defined functions:
 - return a single value per request (per row)
 - can be imported or referenced in the SQL of the query subject definition

Planned Revenue Calculation

`Calc_Planned_Revenue([gosales].[ORDER_DETAILS].[QUANTITY],
[gosales].[ORDER_DETAILS].[UNIT_PRICE])`

User Defined Function Reference



© 2015 IBM Corporation

When you reference a user-defined function in a query subject, it will be represented as a query item and will return a value for each row of data as calculated by the function.

If you wished to reference the user-defined function in the slide example in the SQL of the query subject rather than importing it into the project, the syntax would appear as shown below:

```
select
  ORDER_DETAILS.ORDER_DETAIL_CODE,
  .....
  dbo.Calc_Planned_Revenue(ORDER_DETAILS.QUANTITY, ORDER_DETAILS.UNIT_PRICE) as "Planned Revenue"
from [GOSALES].ORDER_DETAILS
```

Regular Aggregate Property

- the Regular Aggregate property has the following settings:
 - automatic
 - average
 - calculated (applies only to standalone calculations)
 - count
 - maximum
 - minimum
 - sum

© 2015 IBM Corporation



The above are options for relational data source query items with their Usage property set to Fact. The setting made in the model will become the default behavior in the reporting studios.

When IBM Cognos generates SQL, by default, minimum may be applied to aggregated items with the Regular Aggregate property set to unsupported. When the Regular Aggregate property is set to Automatic, if the data type of the query item is numeric, the query item is summarized. If the data type is non-numeric, the query item is grouped. If the query item refers to only one query item in the model, then the aggregation defined for that model item is used.

Business Analytics software

IBM

Order of Operations for Model Calculations

- when a model calculation ($A * B$) is set to Calculated (where A and B are set to sum) the calculation is applied to the summary values

A	B	$A * B$
5	10	50
10	5	50
15	15	225

Summary Row 

Regular Aggregate Property Set to Calculated 

$\rightarrow \text{sum}(A) * \text{sum}(B)$

- when the same calculation ($A * B$) is set to Sum, the calculation is applied, and then the results are summarized

$A * B$
50
50
100

Regular Aggregate Property Set to Sum 

$\rightarrow \text{sum}(A * B)$

© 2015 IBM Corporation 

The Calculated setting in the Regular Aggregate property controls the order of operations in standalone model calculations.

In the slide example, the first $A * B$ column is a result of the summaries of A and B being multiplied ($15 * 15 = 225$). In other words A and B were summarized first and then calculated. This is a less common approach but valid in certain scenarios.

The $A * B$ column is the result of A and B being calculated first and then summarized ($50 + 50 = 100$). This is the more common approach to reporting and is the default behavior in IBM Cognos unless you override it with the calculated setting.

Demo 1: Set Order of Operations for a Model Calculation

Purpose:

Report authors would like to see margin values in their reports. Currently a margin query item calculation is available to them, but it is performing the margin calculation first and then aggregating the values, which is returning undesired results. They would like to see the underlying values aggregated first and then perform the margin calculation.

To accomplish this, you will create a stand-alone margin calculation, since this type of operation is only supported for stand-alone calculations, and set the Regular Aggregate property accordingly.

Components: Framework Manager, Query Studio

Project: GO Operational

Package: GO Operational (query)

Task 1. Test existing Margin query item.

1. Launch Framework Manager, and then open the **GO Operational** project located at **C:\Edcognos\B5A52\CBIFM-Start Files\Appendix A\GO Operational**.

If prompted, log in using admin/Education1.

2. In the **GO Operational Model**, expand **Foundation Objects View > gosales > Sales Fact**.

3. Select and **Test** the following items:

- **Revenue**
- **Gross Profit**
- **Margin**

The results appear as follows:

Revenue	Gross Profit	Margin
8624.64	4625.92	0.53636093796379
9411.6	4840.12	0.51427174975562
18032.22	5072.22	0.28128649717007
0000.0	0000.0	0.0001111111111111

These non-aggregated values. The Margin calculation is Gross Profit divided by Revenue. The margin values are correct. You will now test the items with Auto Sum enabled.

4. Select **Auto Sum**, and then click **Test Sample**.

The results appear as follows:

Revenue	Gross Profit	Margin
4686775768.85	1924834994.68	194030.904167937

Notice the margin value is not what is expected. All the detail records are being calculated first and then aggregated. The value shown in the Margin column is the sum of all the margin values seen in the un-aggregated result set in the previous test. The expected result would be the Gross Profit value above divided by the Revenue value above for a margin of 0.41.

5. Click **Close**.

Task 2. Create a stand-alone Margin calculation and set its Regular Aggregate property.

1. Double-click **Sales Fact**, and in the **Query Items and Calculations** pane, right-click **Margin**, and then click **Convert to Stand-alone Calculation**.

The Margin query item will no longer be required in the Sales Fact query subject. Setting the order of operations to aggregate first and then calculate is only valid for stand-alone calculations. The calculation cannot be restricted by the scope of a query subject container.

2. Delete the **Margin** query item.

A message appears indicating that the Margin query item in the Sales Fact query subject in the Consolidation View will be invalidated. You will also delete that query item in the next few steps.

3. Click **OK** twice.

The stand-alone Margin calculation appears at the bottom of the gosales namespace.

4. In the **Consolidation View**, expand **Sales Fact**, and then delete the **Margin** query item.

A message appears saying that the Margin measure in the Sales Fact regular dimension in the Dimensional View namespace will be invalidated. You will delete this object in the next few steps.

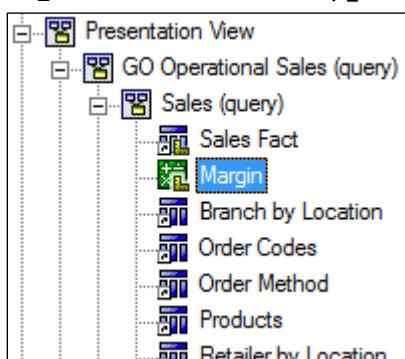
5. Click **OK**.

6. Expand **Dimensional View > Sales Fact**, and then delete the **Margin** query item.

Note: If you want to make the Margin calculation available for your analysis package, simply make the stand-alone calculation available in that package.

Task 3. Change the Regular Aggregate property for the Margin calculation and test in Query Studio.

1. In the **Foundation Objects View**, at the bottom of the **gosales** namespace, select the stand-alone **Margin** calculation, and then in the **Properties** pane, change the **Regular Aggregate** property to **Calculated**.
2. Move the stand-alone **Margin** calculation to the **Presentation View > GO Operational Sales (query) > Sales (query)** namespace, as shown below:



3. Publish the **GO Operational (query)** package, deselecting **Verify the package before publishing**, and clicking **Yes** to overwrite the existing package.
4. In **IBM Cognos Connection**, in **Public Folders**, select the **GO Operational (query)** package, and then launch **Query Studio**.

5. In the **Insert Data** menu, expand **Sales (query)**, and add the following items to the report:

- **Sales Fact > Revenue**
- **Sales Fact > Gross Profit**
- **Margin** (stand-alone calculation)

The results appear as follows:

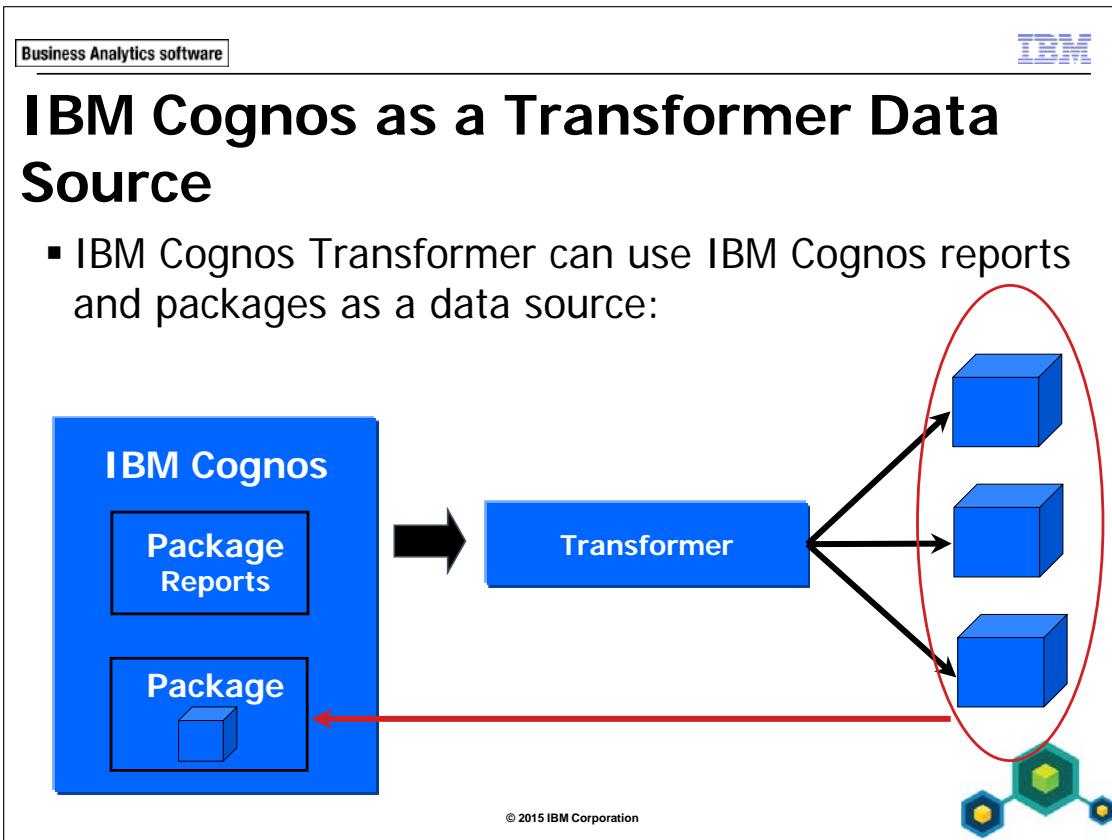
Revenue	Gross Profit	Margin
\$4,686,775,768.85	\$1,924,834,994.68	41%

Notice the aggregated Margin value is now correct. The detail values for Gross Profit and Revenue are aggregated first and then calculated. This value has been formatted as a percentage in Framework Manager. If it had not been formatted, it would have appeared as 0.41069491898314.

6. **Close** the browser without saving the report.
 7. **Save** the project and **Close** Framework Manager.

Results:

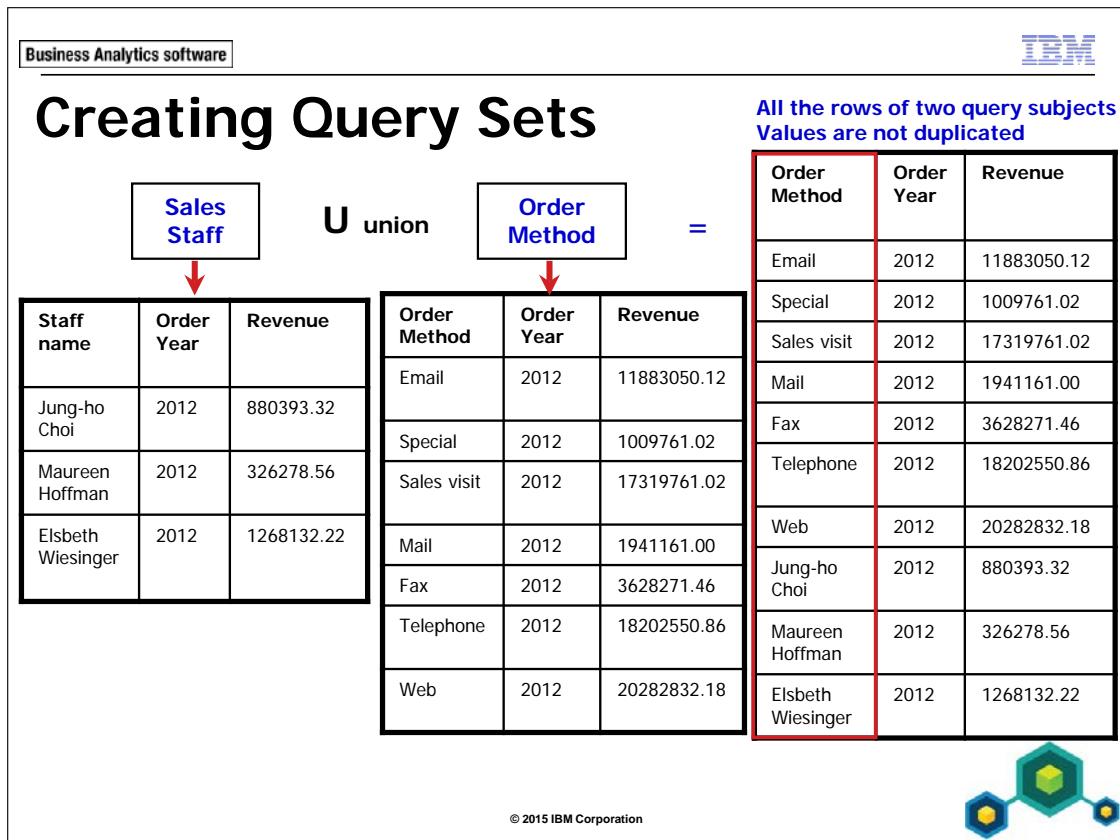
You created a stand-alone margin calculation and set the Regular Aggregate property to Calculated to perform an aggregation before your calculation.



Transformer is an IBM Cognos OLAP modeling component used to model and build PowerCubes. You can make metadata available for use in Transformer. To do this, you must have items that act as category codes (business keys) and captions. The business key values should be unique across all levels in the dimension. For fact data, you need to make items available in your fact query subjects that Transformer will use as measures and as the keys that allow the measures to be related to the dimensions. For example, if your measures will relate to the Product dimension, you will need to make the Product Key available in your fact query subject.

When the appropriate metadata is available, Transformer modelers can access the individual dimension information and measure information (each accessed as a separate data source) through IBM Cognos reports, or directly from metadata in the package.

After a Transformer modeler has created a PowerCube, it can be published as a package in IBM Cognos for use as a multidimensional data source. This data will be conformed to the relational data since both are based on the same source data.



You can define a query set to merge, compare, or equate similar data from different sources. Query sets are useful when modeling data from disparate systems, or when the desired result is not directly available through a query of the data source.

A query set can be defined using the following operations:

- **Union** - include all the rows of two query subjects. For example, your company recently acquired another company and you need a complete list of all customers.
- **Intersect** - include only the rows that are shared between the query subjects. For example, you want to find out which staff members are also managers.
- **Except** - include only the rows that are different between the query subjects. For example, you want to highlight the differences between where your products were sold this year and ten years ago.

Each query subject must have the same number of columns, and columns must be in the same order and have the same (or similar) data types. The data types do not need to be exactly the same if those in the second result set can be automatically converted by the data source to data types compatible with those in the first result set.

Using Source Control

- use a source control system of your choice on Framework Manager project files to:
 - manage and backup versions of your project
 - prevent change conflict between users by checking files in and out
 - ensure modelers are always working on the latest version

© 2015 IBM Corporation



You can manage a Framework Manager project with an external source control system. Please refer to the documentation on how to put the Framework Manager project files into an external repository.

Framework Manager Projects consist of three essential files, plus a log file directory. These files are:

File Name	Optional/Required	Description
[Project name].cpf	Required	This project file tracks segments and links.
model.xml	Required	This is the model data.
customdata.xml	Required	This saves user interface information such as diagram layout.
preferences.xml	Optional	This file is not used.
repository.xml		This file must not be present if you are handling your own repository. If this file is in your project, you must delete it.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

	Name	Size	Type	Date Modified	Att
[+]	godatawarehouse		File Folder	5/9/2005 4:37 PM	
	logs		File Folder	5/9/2005 4:37 PM	
[+]	gosales_goretailers		File Folder	10/4/2004 2:57 PM	A
[+]	junk		File Folder	5/9/2005 3:59 PM	A
[+]	LineageDiagram		File Folder	5/9/2005 3:59 PM	A
	logs	90 KB	XML Document	6/7/2004 11:38 AM	A
	customdata.xml	2 KB	BMT Project File	5/9/2005 3:59 PM	A
	go_data_warehouse.cpf	2,014 KB	XML Document	5/9/2005 3:59 PM	A
	model.xml	1 KB	XML Document	6/7/2004 11:38 AM	A
	Preferences.xml				

Log files are found in the "logs" subdirectory. A new log file is created each for each Framework Manager session. These logs are required only if you intend to use the Model Synchronization feature. External repository handling is simpler if you ignore the log files.

	Name	Size	Type	Date Modified	Att
[+]	godatawarehouse		File Folder	10/5/2004 11:01 AM	A
	logs		File Folder	10/5/2004 11:16 AM	A
[+]	gosales_goretailers		File Folder	5/9/2005 3:59 PM	A
[+]	junk		File Folder	5/9/2005 4:28 PM	A
[+]	LineageDiagram		File Folder	5/9/2005 4:37 PM	A
[+]	link		File Folder	5/9/2005 4:47 PM	A
	go_data_warehouse-20041005105202-log.xml	18 KB	XML Document	10/5/2004 11:01 AM	A
	go_data_warehouse-20041005111445-log.xml	2 KB	XML Document	10/5/2004 11:16 AM	A
	go_data_warehouse-20050509155417-log.xml	1 KB	XML Document	5/9/2005 3:59 PM	A
	go_data_warehouse-20050509162817-log.xml	1 KB	XML Document	5/9/2005 4:28 PM	A
	go_data_warehouse-20050509163719-log.xml	1 KB	XML Document	5/9/2005 4:37 PM	A
	go_data_warehouse-20050509164739-log.xml	1 KB	XML Document	5/9/2005 4:47 PM	A

After your Framework Manager project is created, make sure it is closed, and then, 'Check In' the project file.cpf, model.xml, and customdata.xml into your repository of choice.

Before opening the Framework Manager project to begin modeling again, you will be required to 'Check Out' those three files out from your repository. If you fail to do so (and your repository system marks checked in files as read-only), Framework Manager will open in read-only mode. The text "[Read Only]" will appear on the application title. If this happens, close the project, and check out the files. If you have done a lot of work in a read-only project, then you may save the project, and you must then ensure those files get checked in properly.

When your Framework Manager session is complete, close the project, and then 'Check In' these three files to your repository.

Summary

- You should now be able to:
 - leverage a user defined function
 - identify the purpose of query sets
 - use source control to manage Framework Manager files



© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

A-15

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



Modeling Multilingual Metadata

IBM Cognos BI

Business Analytics software



© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

Business Analytics software

IBM

Objectives

- At the end of this module, you should be able to:
 - customize metadata for a multilingual audience

© 2015 IBM Corporation

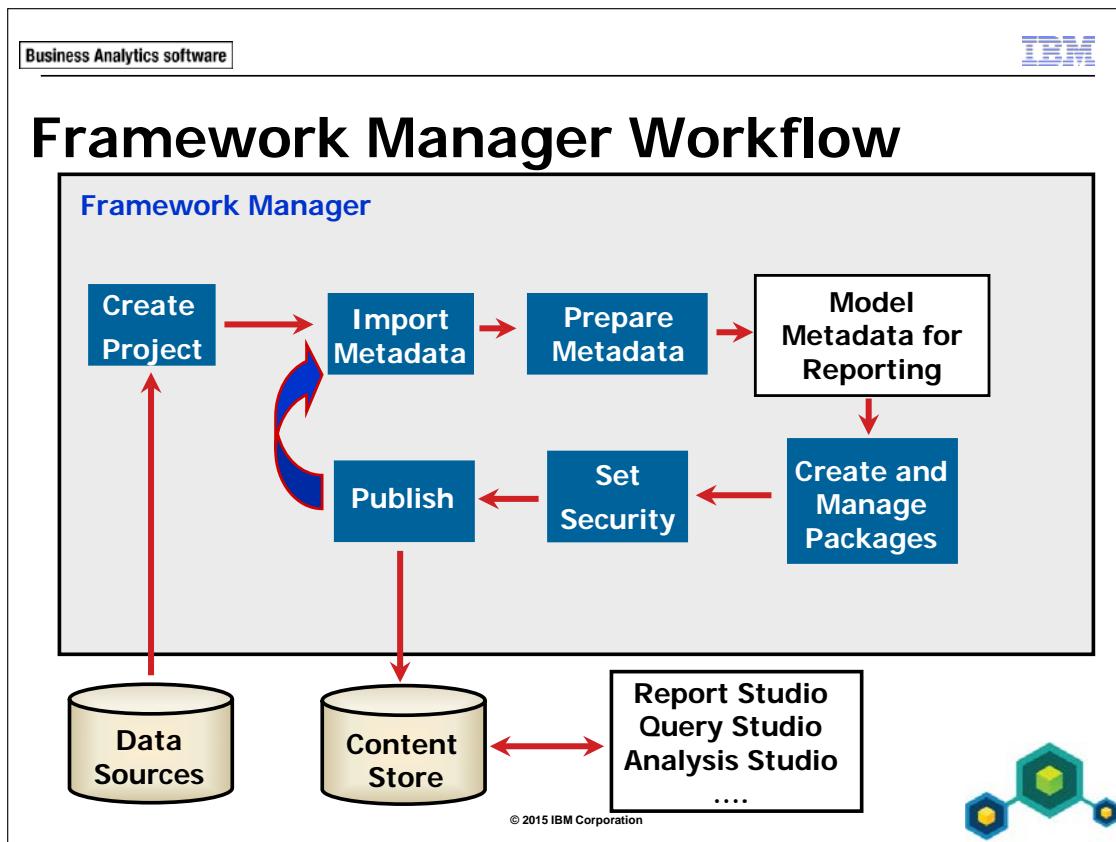
Unless specified otherwise in demo or workshop steps, you will always log on to IBM Cognos in either the Local NT namespace or Local LDAP namespace (either is fine) using the following credentials:

- User ID: admin
- Password: Education1

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.



This module describes modifying language properties to ensure that metadata is available in all languages required for authors.

Modifying Language Properties

- each model object has three Language properties:
 - Name
 - Description
 - Screen Tip
- language properties can be modified:
 - manually in the project
 - by exporting/importing a translation file

© 2015 IBM Corporation



You can translate the multilingual text properties associated with metadata; names, descriptions, and screen tips. Only the metadata is translated. You can then publish your package to make the translated metadata available to authors. To do this:

- define the languages to be included in the project
- modify the Language properties manually or export the multilingual text properties to a translation file, which translators use to input the correct text for each language. Once translation is complete, import the file to update the Language properties with the translated strings.
- specify the languages you want to include in your package (they must be available at the project level first) and then publish the metadata.

Optionally, you can publish the package, set the Content language to French (as done in the previous demo), and show the Pays de Succursale in Query Studio.

Demo 1: Apply Multilingual Query Item Properties

Purpose:

You want to ensure that users working in languages other than English can easily use the GO Operational model to create reports. You will add French support to the project, translate a query item name and description, publish the package with both French and English metadata, and test the package in Report Studio.

Component: **Framework Manager**
 Project: **GO Operational**

Task 1. Add languages to the project.

1. In **Framework Manager**, open the **GO Operational** project located at **C:\Edcognos\B5A52\CBIFM-Start Files\Appendix B\GO Operational**.
 If prompted, log in using admin/Education1.
 This project has had many additional languages removed from the model in order to perform this demonstration.
2. From the **Project** menu, point to **Languages**, and then click **Define Languages**.
3. In the **Available languages** box, click **French**, click **Add** , and then click **OK**.
 A warning message appears indicating that the languages will be added to every text property of every object. Optionally, you can add a locale you will not use in order to preserve the original data source table and column names if you plan to modify them in the English and French Locales
4. Click **OK**.

Task 2. Add a French name and description.

1. Expand **GO Operational Model > Consolidation View > Products**, and then click the **Product Type** query item.
2. At the top of the **Properties** pane, click the **Language** tab.

The results appear as follows:

	Name		Description		Screen Tip	
	English	French	English	French	English	French
Product Type	Product Type	(fr) Product T...				

Notice that for the Name, Description and Screen Tip properties, you now see one entry for each language. However, all entries are in English initially, the project's design language.

The description and screen tip are blank for the French columns. You cannot add a French value for either the description or screen tip until you supply a value for the primary language (English).

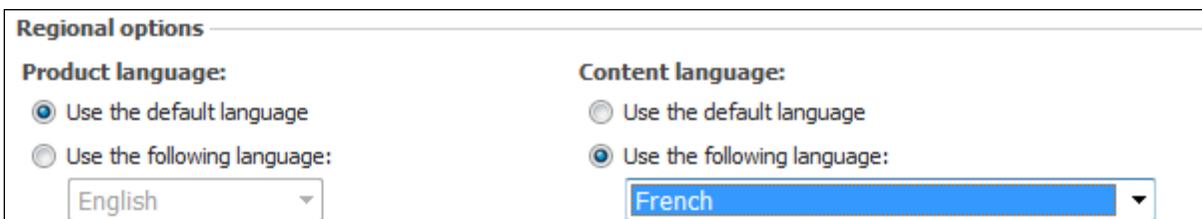
3. In the **Description** row, under **English**, type **Product Type identifies a set of related products**, and then press **Enter**.
You can now add a French value, because you have supplied a value for the primary language.
4. In the **Description** row under **French**, delete the existing text, type **Type de produit**, and then press **Enter**.
If you want to type a full translation, you can enter Le Type de Produit identifie un ensemble de produits liés.
5. In the **Name** row under **French**, type **Type de produit**, and then press **Enter**.
You can also change the active language in the Tools pane to quickly edit query subject names for the appropriate language.
6. **Save** the project.

Task 3. Publish the model in both English and French, and create a report.

1. In **Packages**, select the **GO Operational** package, and from the **Actions** menu, click **Package > Specify Package Languages**.
2. Under **Available Project Languages**, select **French**, click **Add**, and then click **OK**.
3. Publish the **GO Operational** package, deselecting **Verify the package when publishing**, and entering **group** when prompted for a value.
Tip: In the Properties pane for packages, you can also translate the package name and any descriptions or tool tips required by end users.
4. In **IBM Cognos Connection**, launch **Report Studio**, and select the **GO Operational** package.
5. Click **Create New**, and then double-click **List**.
6. In the **Source** pane, expand **Presentation View > GO Operational Sales (query) > Sales (query) > Products**.
7. Right-click **Product Type**, and then click **Properties**.
In Description Properties value contains the text that you specified earlier.
8. Click **Close**, and then close **Report Studio** without saving.

Task 4. Create a report to verify the French query item name and description.

1. In **IBM Cognos Connection**, click **My Area Options**  in the top right corner, and then click **My Preferences**.
2. Under **Regional options**, change the **Content language** setting to **Use the following language**.
3. In the drop-down list, select **French**.



If you change the Product Language to French the UI will be displayed in French.

4. Click **OK**.
5. Launch **Report Studio**, and select the **(fr) GO Operational** package.
6. Create a new **List** report.

Report Studio opens with the metadata from the (fr) GO Operational model appearing in the Sources pane. Notice that all items are prefixed with (fr). This indicates that you are now in the French version of the model but you have not yet translated all of your metadata titles and properties.

7. Expand **(fr) Presentation View > (fr) GO Operational Sales (query) > (fr) Sales (query) > (fr) Products**.

Notice that Type de produit stands out among the (fr)-prefixed query item names.

8. Right-click **Type de produit**, and then click **Properties**.

The Description property value (Type de produit) matches the text that you specified in Framework Manager.

9. Click **Close**, and then close **Report Studio**.
10. In **IBM Cognos Connection**, click **My Area Options** in the top right corner, and then click **My Preferences**.
11. Under **Regional options** and **Content language**, click **Use the default language**, and then click **OK**.

Results:

You added French support to the project, translated a query item name and description, published the package in both French and English, and tested the package in Report Studio.

Demo 2: Apply Multilingual Translation Files

Purpose:

You want to ensure that users working in languages other than English can easily use the GO Operational model to create reports. To this end, you will enhance the model by exporting a translation file and modifying it so that it contains French and English strings. You will then import this file back into the model and view the results.

Component: **Framework Manager**

Project: **GO Operational**

Task 1. Export metadata to a CSV file for translation.

1. In **Framework Manager**, from the **Project** menu, point to **Languages**, and then click **Export Translation File**.

This approach is an alternative to editing the metadata in Framework Manager, as we did in the previous demo. When you export language strings to a TXT or CSV file, you can translate them in a single edit session. You may even send this file to a translation firm for professional translation. The TXT or CSV file only contains the strings that exist. If a given item has no description or screen tip, then no entry will be created during export, and therefore no translation can be imported. For this reason, you should add descriptions and tool tips to the model objects in the primary language before exporting

2. Under the **Project Languages** pane, Ctrl+click **French** so that both **English** and **French** are selected, and then click **Add** to move them to the **Languages to be exported** pane.
3. Next to the **Export languages to this file** box, click **Browse** .
4. In the left pane, click **Desktop**, and in the **File name** box, type **GO_Application_LOC**.
5. In **Save as** type, select **CSV (comma delimited) (*.csv)**.

6. Click **Save**, and then click **OK**.

A message appears, indicating that the language strings were successfully exported.

7. Click **OK**.

8. On the **Windows Desktop**, double-click **GO_Application_LOC.csv** to open it in **Microsoft Excel**, and then expand the first two columns.

Notice that each column represents a given language; in this case, English and French. These are based on the language selections you made when you exported the model languages in previous steps.

9. In the second column of row **18**, change the **French** value of **(fr) Branch City** to **Ville de Succursale**.
10. In the second column of row **21**, change the **French** value of **(fr) Branch Country** to **Pays de Succursale**.
11. Save the file, click **Yes** to the warning message, and close **Excel**.

Task 2. Import a CSV file that contains translated strings.

1. In **Framework Manager**, from the **Project** menu, point to **Languages**, and then click **Import Translation File**.
2. In the **Project Languages** pane, click **French**, and then below **Translate into**, click **Add**  to add it to the **Translate into** pane.
3. Next to the **Import translation table from this file** box, click **Browse**.
4. In the **Files of type** box, click **CSV (comma delimited) (*.csv)**, and then, if necessary, click **Desktop**, and then double-click **GO_Application_LOC.csv**
5. Click **OK**.

A message appears indicating that the import was successful. All objects that reference the translated metadata are updated, such as items in the Foundation Objects View, the Consolidation View and the Dimensional View.

6. Click **OK**.
7. In the **Consolidation View**, expand **Branch by Location**, and then click **Branch Country**.

- In the **Properties** pane, click the **Language** tab.

The results appear as follows:

Properties					
	Properties	Language			
	Branch Country	<table border="1"> <thead> <tr> <th>Name</th> </tr> </thead> <tbody> <tr> <td>English</td> </tr> <tr> <td>French</td> </tr> </tbody> </table>	Name	English	French
Name					
English					
French					
	Branch Country	Pays de Succursale			

Under Name, the French value reflects the change that you made to the translation file.

- Click **Branch City**.

The results appear as follows:

Properties					
	Properties	Language			
	Branch City	<table border="1"> <thead> <tr> <th>Name</th> </tr> </thead> <tbody> <tr> <td>English</td> </tr> <tr> <td>French</td> </tr> </tbody> </table>	Name	English	French
Name					
English					
French					
	Branch City	Ville de Succursale			

Again, in the Properties pane, notice that the French value for the Name property of the query item is in French.

- Save and Close Framework Manager.

Results:

You enhanced the GO Operational model by exporting the model languages to a translation file. You modified this file so that it contained French and English strings. You imported the file back into the model and viewed the results. Users can now begin to work with the GO Operational model in languages other than English.

Summary

- You should now be able to:
 - customize metadata for a multilingual audience



© 2015 IBM Corporation

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

B-13

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

© 2003, 2015, IBM Corporation

This guide contains proprietary information which is protected by copyright. No part of this document may be photocopied, reproduced, or translated into another language without a legal license agreement from IBM Corporation.

This material is meant for IBM Academic Initiative use only. NOT FOR RESALE

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.