Legal notice

Copyright © 2013 Infor. All rights reserved. The word and design marks set forth herein are trademarks and/or registered trademarks of Infor and/or related affiliates and subsidiaries. All other trademarks listed herein are the property of their respective owners. www.infor.com.
# Table of Contents

- Lesson 1: Introduction Data Modeling ................................................................. 1
- Lesson 2: Set up an OLAP database .................................................................... 6
- Lesson 3: Classification of Dimensions ............................................................... 8
- Lesson 4: Hierarchies .......................................................................................... 12
- Lesson 5: Elements with attributes ..................................................................... 15
- Lesson 6: Introduction Rules ............................................................................... 18
- Lesson 7: Creating a Price Volume Rule .............................................................. 24
- Lesson 8: Creating an Allocation Rule ................................................................. 32
- Lesson 9: Currency Translation .......................................................................... 38
- Lesson 10: Attribute Tables ............................................................................... 50
- Lesson 11: Calculation Rules ............................................................................... 58
- Lesson 12: Differentiating between the Old and New Rules Engine ................. 65
- Lesson 13: Importing Dimensions and Automation ........................................... 70
- Lesson 14: Converting Rules into Physical Values (Advanced Lesson) ............ 79
- Course Summary .................................................................................................. 83
About This Workbook

Welcome to this Infor Education course! We hope you will find this learning experience enjoyable and instructive. This Training Workbook is designed to support the following forms of learning: modeling

- Classroom instructor-led training
- Virtual instructor-led training

This Training Workbook is not intended for self-study or as a product user guide.

Activity Data

You will be asked to complete some practice exercises during this course. Step-by-step instructions are provided in this guide to assist you with completing the exercises. Where necessary, data columns are included for your reference.

Your instructor will provide more information on systems and databases used in class, including server addresses, login IDs and passwords.

Reference Materials

Infor BI reference materials are available from the following locations:

- Infor Sales Portal
- Office Plus Online Help: Data Access>Cubes and Data Areas>Cube Rules
- The OLAP Server Rules Engine document

Symbols Used in this Workbook

- Hands-on exercise ("Exercise")
- For your reference

- Instructor demonstration ("Demo")
- Your notes

- Question

- Answer

@ Copyright 2014 Infor  
All rights reserved.  
Do not distribute and reproduce without authorization.
Course Overview

This course introduces how to model and program BI OLAP databases. The examples used in this course include data modeling exercises and rules that efficiently deal with:

- Price/Volume calculation
- Currency conversion
- Allocations
- Variance calculations based on account type
- Treatment of balance and flow calculations over time
- Lookup tables to assign standard costs

By building business rules using BI OLAP, you will be able to address even the most complex business models with one of these procedures or a combination thereof.

Course Length

- 2 days

Course Goal

- Provide advanced BI OLAP users with a series of examples on how to program BI OLAP databases to handle complex, yet common, business problems by setting up cubes and rules.

Learning Objectives

Upon completion of this course, you will be able to:

- Model dimensions, attributes and cubes the right way.
- Explain cube rules and syntax.
- Create and edit cube rules.
- Look up data via attribute tables.
- Use attributes to determine calculation rules.
- Create macros to import data.
- Create and edit accelerators to optimize performance.

Audience

- Advanced BI OLAP Users
- Business Consultants
- Infor Business Partners

System Requirements

- BI OLAP Demo Databases

Prerequisite Knowledge

- Knowledge of multidimensional databases
<table>
<thead>
<tr>
<th>Lesson</th>
<th>Lesson title</th>
<th>Learning objectives</th>
<th>Estimated time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Overview Day 1 Data Modeling and Rules</td>
<td>Introduction Data Modeling</td>
<td>• Review course expectations.</td>
<td>1 hour</td>
</tr>
<tr>
<td>Lesson 1</td>
<td>Introduction Data Modeling</td>
<td>• Install and configure the demo databases for this training.</td>
<td>1 hour</td>
</tr>
<tr>
<td>Lesson 2</td>
<td>Set up an Olap database</td>
<td>• General overview</td>
<td>1 hour</td>
</tr>
<tr>
<td>Lesson 3</td>
<td>Classification of Dimensions</td>
<td>• Structures, Master Data – How to set up and combine dimensions and attributes</td>
<td>1 hour</td>
</tr>
<tr>
<td>Lesson 4</td>
<td>Hierarchies</td>
<td>• Why and how to build hierarchies (parallel)</td>
<td>1 hour</td>
</tr>
<tr>
<td>Lesson 5</td>
<td>Elements with attributes</td>
<td>• How to set up fixed elements</td>
<td>1 hour</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Flow and stock figures</td>
<td></td>
</tr>
<tr>
<td>Lesson 6</td>
<td>Introduction Rules</td>
<td>• Explain cube rules.</td>
<td>2 hours</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Create and edit cube rules.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Describe general cube rule syntax.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Identify cube rule conventions.</td>
<td></td>
</tr>
<tr>
<td>Course Overview Day 2 Rules</td>
<td></td>
<td>• Review course expectations.</td>
<td>30 minutes</td>
</tr>
<tr>
<td>Lesson 7</td>
<td>Creating a Price Volume Rule</td>
<td>• Create and edit a price volume rule.</td>
<td>1 hour</td>
</tr>
<tr>
<td>Lesson 8</td>
<td>Creating an Allocation Rule</td>
<td>• Create and edit an allocation rule.</td>
<td>30 minutes</td>
</tr>
<tr>
<td>Lesson 9</td>
<td>Currency Translation</td>
<td>• Calculate currency with the exchange rate cube.</td>
<td>1 hour</td>
</tr>
<tr>
<td>Lesson 10</td>
<td>Attribute Tables</td>
<td>• Look up product cost data using attribute tables.</td>
<td>1 hour</td>
</tr>
<tr>
<td>Lesson 11</td>
<td>Calculation Rules</td>
<td>• Use attribute tables and cube rules to determine calculation rules.</td>
<td>30 minutes</td>
</tr>
<tr>
<td>Lesson</td>
<td>Lesson title</td>
<td>Learning objectives</td>
<td>Estimated time</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------------</td>
<td>---------------------------------------------------------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| Lesson 12 | Differentiating between the Old and New Rules Engine   | • Identify the differences between the old rules engine and the new rules engine.  
|           |                                                        | • Optimize cube rule performance.  
|           |                                                        | • Optimize cube rules.                                        | 1 hour         |
| Lesson 13 | Importing Dimensions and Automation                    | • Use macros to import dimensions and data.                   | 1 hour         |
| Lesson 14 | Converting rules into Physical Values (Advanced Lesson)| • Use macros to convert rules into physical values            | 1 hour         |
| Course Summary |                                               | • Debrief course.                                             | 30 minutes     |

**TOTAL ESTIMATED TIME** 16 hours
Lesson 1: Introduction Data Modeling

Estimated Time

- 1 hour

Learning Objectives

After completing this lesson, you will be able to:

- Install and configure the demo databases for this training.

Topics

- Configure the Demo Databases
Configure the Demo Databases

This training uses several demo databases that you must configure properly before you begin. After you add the databases, you will then need to edit the database settings. For example, to adjust the rules cache size. This will ensure performance is not reduced. Your instructor will direct you to select the appropriate database for use during certain exercises.

Exercise 1.1: Configuring the Demo Databases

In this exercise, you will configure the Rules demo databases that you will use during this training.

Exercise Steps

1. Start the BI OLAP application. The BI OLAP Administration window opens.

2. Select Computer Configuration.
3. Select Local Computer.
4. Right-click Local computer.
5. Click the Add Databases button.
6. Select each of the demo databases, as shown in the screen shot below:

7. Click OK to accept changes. The demo databases will now display in the Local computer list.
8. Right-click on the first database name.
10. Edit the repository connections in the database settings by linking to the valid OLAP Permission Management:

- Repository registration
- Project name
- OLAP Permission Management

If you have more than one database, you can globally adjust the settings, with the exception of OLAP Permission Management.

When you are creating or editing rules, you must adjust the rules cache size in the Database Properties (in the BI OLAP Administration/Computer Configuration/Databases folder). By turning on the cache, the performance should be much better. This must be done for each database separately.

For better performance it also makes sense to increase the size of the calculation block depending on the size of the cubes.

**Demo: Set Up Rules Cache**

Your instructor will demonstrate how to set up the rules cache that is required for each database to help avoid performance impacts.

1. Select Database Settings.
2. Go to the Rules Engine section of the window.
3. Set the Calculated Rules Cache* value to 1. Any value greater than 0 enables it, while 0 will disable the cache.

Setting the rules cache can also be done at runtime. To apply the rules cache at runtime, click the corresponding server/database and select the Apply Runtime Settings command.

**Demo: Set Up the Size of calculation block**

Your instructor will demonstrate how to setup increase Size of calculation block corresponding to the biggest data cube.

4. Select Database Settings.
5. Go to the Memory section of the window.
6. Set the Size of the calculation block at least to 1E20.

Setting the size of the calculation block can also be done at runtime. To apply the rules cache at runtime, click the corresponding server/database and select the Apply Runtime Settings command.

Cache Partitioning

The OLAP Server saves calculated values in the cache to retrieve them fast instead of recalculation. Changed values and dependent cells will be invalidated (deleted from the cache) according to the dependencies in the cache partitions. Due to these saved dependencies between cache partitions, it is not necessary to delete the whole cube cache, but only dependent partitions. This is also valid for dependencies between different cubes.

As each of the dimension members in the partitioning dimension gets its own partition, the effort to maintain partitions is expected to rise quickly with more elements. Therefore, it is recommended to use this feature with a small dimension. The maximum number of elements is 256, in case of an inappropriate configuration (the partitioned dimensions contain too many elements) cache partitioning will be disabled and a message will be written into the AleaPr.txt log file.

Example:
To indicate the dimension "YEARS" as to be partitioned, enter this XML information into the Extended Properties dialog box:

```xml
<Alea:Properties xmlns:Alea="http://www.misag.com">
  <Alea:CachePartitionDimension Name="YEARS"/>
</Alea:Properties>
```

Note: For some use cases, more than one cube must be properly partitioned.

Selecting the Demo Database

Each of the lessons in this training refers to different demo databases. At the beginning of each lesson your instructor will ask you to select the appropriate server/database.

Note: For ease of use, ensure the File, Panes, and Database toolbars are displayed, as shown below. If the panes are not displayed, select Tools > Run OLAP add-in in Office Plus.
Exercise 1.2: Selecting a Demo Database
In this exercise, you will select the databases for use during this portion of the course.

Exercise Steps
1. Select BI Office Plus> Log On to start the application (or click the appropriate toolbar icon).
2. Click on the Database Structure icon to display the left pane.
3. Log on to the database as directed by your instructor.
   a. User name: Admin
   b. Password: Not required
4. Create a new database alias, if none exists, by clicking on the New Database Alias symbol and entering or selecting the data noted below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type a database alias.</td>
<td>Use the database name</td>
</tr>
<tr>
<td>Provider</td>
<td>Select a value.</td>
<td>BI OLAP Provider</td>
</tr>
<tr>
<td>Driver</td>
<td>Select a driver.</td>
<td>XMLA</td>
</tr>
<tr>
<td>Server</td>
<td>Select a server name.</td>
<td>Localhost OR server name</td>
</tr>
</tbody>
</table>

5. Click OK to accept the settings.
Lesson 2: Set up an OLAP database

Estimated Time

- 1 hour

Learning Objectives

After completing this lesson, you will be able to:

- Understand the general data modeling of an OLAP database.

Topics

- Main principles for setting up a database
Set up an OLAP database

In this topic, the main principles for setting up an OLAP database will be discussed.

Key topics are:
- One big cubes vs. several small cubes
- Data to be stored in an OLAP database (master data, transactional data – only data which should be consolidated for analysis)
- Splitting master data in different dimensions
- Target is high filling level of a cube – which data are needed?
- Key dimensions like time or measure with different hierarchies

Dimensions, Elements and Cubes are the building blocks of Infor BI OLAP multidimensional databases. A **Dimension** is made up of a group of similar items, or simply a list of related members. These members are called **Elements**. A dimension might be a list of items that make up columns or rows in a report. For example, the region dimension may contain the elements Germany, France and United Kingdom. The account dimension might be made up of Sales, Units, Variable Cost and Gross Margin. The product dimension could simply be a list of all products. Dimensions contain no information; they simply serve as indices to the information contained in cubes. Relationships between dimension elements are defined in plain English: Central Europe = Germany + France etc. or Southern Europe = Spain + Italy. Cubes are the components of the Infor BI OLAP database that actually contain data. A cube is simply a collection of dimensions assembled to create a matrix. Examine the following diagram of a three-dimensional cube that contains the dimensions region, account and month. By specifying the elements for a particular value we are able to locate or identify that value, and by doing so, we can retrieve the value from a database or store the value in a database.
Lesson 3: Classification of Dimensions

Estimated Time

- 1 hour

Learning Objectives

After completing this lesson, you will be able to:
- Structure dimensions in an OLAP database and combine them to different cubes.

Topics

- Structuring dimensions of an OLAP database.
Classification of Dimensions

In this topic, the structuring and classification of dimensions will be discussed.

Each data cube for historical data contains on a regular basis dimensions for time, measure and scenarios. On top of that depending on the purpose or using of the cube dimensions for organizational aspects (like units or cost center), market structures (customers, sales offices, regions) or structures for adding values (item, product group) can also be part of a data cube.
Exercise 3.1: Set up only one dimension with attribute Zip Code or two dimensions (customer and Zip Code separately)?

In this exercise you will choose setting up dimensions for customers in relation to Zip Codes.

<table>
<thead>
<tr>
<th>Customer1</th>
<th>Customer2</th>
<th>Customer3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zip Code6</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Zip Code7</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Zip Code8</td>
<td></td>
<td>✗</td>
</tr>
</tbody>
</table>
Exercise Steps

1. Assumption: Customer to Zip Code relation is a 1 to 1 relation. How many dimensions need to be set up?
2. Assumption: One customer has two Zip Codes. How many dimensions need to be set up?
3. Assumption: Two customers have the same Zip Code. How many dimensions need to be set up?
Lesson 4: Hierarchies

Estimated Time
- 1 hour

Learning Objectives
After completing this lesson, you will be able to:
- Understand the general data modeling of a dimension with different hierarchies OLAP database.

Topics
- Hierarchies of dimensions
Hierarchies

In this topic, the main principles for setting up hierarchies in dimensions will be discussed.

Key topics are:
- Hierarchies vs. attributes
- Hierarchies for different analysis of consolidated data
- Combine different aspects like products and regions in only 1 dimension with region as consolidated level and products as base level.
- Assign a base element to different consolidated elements/hierarchies
- Data are only stored on base elements level

![Diagram of Product structure/hierarchy]

- = Base Elements (N)
- = Consolidated Elements (C)
Hierarchy 1

Hierarchy 2

@ = Base Elements (N)  • = Consolidated Elements (C)

14 Lesson 4: Hierarchies
© 2013 Infor Education
Lesson 5: Elements with attributes

Estimated Time

- 1 hour

Learning Objectives

After completing this lesson, you will be able to:

- Understand the general data modeling of elements with attributes.

Topics

- Data modeling of elements and attribute tables.
Elements with attributes

**Elements**
- N-Elements are (numeric) base elements
- C-Elements are consolidated (calculated) elements

**Attributes**
- Store additional information about elements, for example:
  - Product name in different languages
  - Account descriptions
  - Market launch date
- Up to 3 attribute tables per dimension
- Up to 49 attribute fields per attribute table
- e.g.: Dimension „Product“ → Element „Product 1“ → Attribute: Product manager „Mayer“

- If you don’t need to summarize this information you can create an attribute
- Attributes are detailed information of elements like description in different languages, currency or company information
- Attributes are not dynamic. Don’t use attributes which are aligned to other information like price of a unit.
- By changing attributes history data are changed as well. Attributes haven’t got any time relation.
Lesson 6: Introduction Rules

Estimated Time

- 2 hours

Learning Objectives

After completing this lesson, you will be able to:

- Explain cube rules.
- Create and edit cube rules.
- Describe general cube rule syntax.
- Identify cube rule conventions.

Topics

- Create and Edit Cube Rules
Create and Edit Cube Rules

The BI OLAP cube rules are flexible and powerful, but their syntax is similar to spreadsheet formulas. However, because they are written for a cube, rather than for a flat table, they must be able to address multidimensional slices. This requires a syntax that takes elements of multiple dimensions into account. For example, a cube rule may state that:

\['Price', 'Brazil'\] = \['Sales', 'Brazil'\]/\['Units', 'Brazil'\]

This rule applies to the element Price from the Measures dimension and Brazil from the Region dimension. It takes the elements from two different dimensions into account.

Cube rules may also refer to values in cubes other than the one for which the rule is being written.

**General Cube Rule Syntax**

Cube rules take on the general syntax, as shown in the example below:

\((Target)=fx(Source)\)

In the example shown above, the target is a function of source. The target is an area within the cube for which the rule is written. The source is either an area within the cube for which the rule is written, or an area within another cube. This is very similar to a spreadsheet formula, but with a more sophisticated source and target referencing.

To define an area in the cube rules syntax, you must specify an element, group of elements, or a subset for each dimension that you want to limit. If you want to take all the elements from a dimension into account, do not specify any elements from that dimension.

**Overview of Cube Rule Conventions**

This section includes several conventions that you should consider when writing cube rules.

**Arguments**

Listed below are the arguments you can use for a cube rule:

- **Areas**
  - Areas must be enclosed in square brackets: ['Price'], ['Sales', 'Brazil'].
- **Numeric constants**
- **The standard arithmetical operands** + - * /
- **Parentheses ( )**
- **A number of mathematical functions**
- **Logical functions** (like IF statements)
- **Other functions** to read information about elements, attribute tables, or cell values.

The rules compiler ignores spaces and carriage returns. You can insert spaces and carriage returns as necessary to make a rule easier to read and understand.
Arithmetical Operands

Arithmetical operands are evaluated with standard algebraic priority. For example, multiplication and division are evaluated first followed by addition and subtraction. The order of evaluation can be forced using parentheses.

For example, the result of following calculation is:

\[ 3 + 2 \times 5 = 13 \]

whereas,

\[ (3+2) \times 5 = 25 \]

IF Statements

The logical IF statement takes on the format noted below:

IF(Conditional statement, expression 1, expression 2)

The conditional statement is evaluated. If it is TRUE, the results of expression 1 will be returned. If it is FALSE, the results of expression 2 will be returned.

This is similar to the "if" condition formula in an Excel spreadsheet.

Frequently used Operators

The table below list frequently used operators in conditional statements.

<table>
<thead>
<tr>
<th>Operand</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not equal to</td>
</tr>
<tr>
<td>+</td>
<td>Plus (or Add)</td>
</tr>
<tr>
<td>-</td>
<td>Minus (or Subtract)</td>
</tr>
<tr>
<td>/</td>
<td>Divided by</td>
</tr>
<tr>
<td>*</td>
<td>Multiplied by</td>
</tr>
</tbody>
</table>

BI OLAP returns zero as the result of a division by an empty cell.
Strings

Strings may be used as result values of a cell for cube rules. They are enclosed in single/double quotes, as shown below:

\[ \text{IF( } [\text{Sales}] > 1000, 'OK', 'Not so good') \]

When calculating values for cells, BI OLAP first calculates cube rules, then dimension rules. The last calculations performed are those within the dimension definition, such as consolidations which include element weights.

New or edited cube rules are only saved when you save the entire database, the single cube or when you click Save in the Cubes and Data Areas dialog box. This allows you to experiment with the rules.

BI OLAP distinguishes between internal references and external references. An internal reference is an area within the cube for which you are writing the rule. An external reference is an area within a cube other than the one for which you are writing the rule.

Exercise 6.1: Start Creating a Cube Rule

In this exercise, you will begin the process of creating a cube rule using the Price data model.

Exercise Steps

1. Click the Cubes button in the OLAP Administration toolbar (or click Database > Cubes in the Office Plus menu) to open the Cubes and Data Areas dialog box.
2. Select the cube for which you want to create a cube rule.
3. Click the Edit Rule button. The Cube Rules dialog box displays.
4. Click the New Rule button (second from left) in the Cube Rules dialog box toolbar. The Edit Cube Rule dialog opens.
**Edit Cube Rule Dialog Box**

In the Edit Cube Rule dialog box you can write or edit cube rules. It is divided into three areas: a target, a source and a description. Type the text into either of these areas. You may insert spaces and line breaks as necessary in a rule.

The Edit Cube Rule dialog box also provides you with a number of utilities for inserting the components of a cube rule, including dimension and element names, cube areas and functions. When you have entered the rule, click OK. The rules editor will run a syntax check and apply the rule to the cube, if there are no syntax errors.

![](edit_cube_rule_dialog_box.png)

*Edit Cube Rule Dialog Box*
Exercise 6.2: Finish Creating a Cube Rule

In this exercise, you will finish creating the cube rule you started in the prior exercise.

Exercise Steps

1. In the Edit Cube Rule dialog box, enter the information noted below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>Type a target location in the cube.</td>
<td>[REGION:’Germany’ MEASURES:’Sales’]</td>
</tr>
<tr>
<td>Rule</td>
<td>Type the rule.</td>
<td>[MEASURES:’Units’][MEASURES:’Average price’]</td>
</tr>
<tr>
<td>Description</td>
<td>Type a description for the rule.</td>
<td>Calculate the revenue for Germany</td>
</tr>
</tbody>
</table>

2. Select the option Basic cells in the affected cells area.

3. Click OK.

Avoid writing the rule as follows: [REGION:’Germany’, MEASURES:’Units’][REGION:’Germany’, MEASURES:’Average price’].

The region should be not part of the rules section because it is already defined in the target area and will reduce performance.
Lesson 7: Creating a Price Volume Rule

Estimated Time
- 1 hour

Learning Objectives
After completing this lesson, you will be able to:
- Create and edit a Price Volume rule.

Topics
- Creating and Editing a Price Volume Rule
Creating and Editing a Price Volume Rule

In this topic, a specific business scenario will be used with the Price data model to illustrate how rules relate to hierarchical dimensions.

Scenario

"I receive unit sales data as well as unit price and unit cost data. My manager wants me to report sales, cost and gross margin dollars. Is there a way I can calculate this data in my BI OLAP cube?"

Rules That Apply Differently to Basis and Consolidated Elements

In this section we will create three rules that calculate gross margin based on sales volume, prices, and costs.

Before you write cube rules, it is useful, and sometimes required, to divide the problem into smaller pieces. Decide which data is affected, which data is needed, and what the calculation must be. In this example, all dollar values (other than unit price and cost) are stored and reported in thousands. Each region has different costs and prices associated with a product. The unit information also changes over time.

Considerations:

- You have the TOTSALES cube, where all the information (including unit prices and costs) is stored.
- You must calculate Sales and Variable Costs, but show the results in thousands, not in whole dollars.
- The basic calculation for Sales is:
  \[ \text{Sales} = \frac{\text{Units} \times \text{Average Price}}{1000} \]
- The basic calculation for Variable Costs is:
  \[ \text{Variable Costs} = \frac{\text{Units} \times \text{Unit Cost}}{1000} \]
- The Sales and Variable Costs must be calculated for each region and product combination and summed to get region or product totals (unit prices cannot be summed prior to multiplying by unit totals).
- These calculations involve several measures (Units, Sales, Unit Cost, etc.) and several mathematical operations.
Exercise 7.1: Create a Rule to Calculate the ['Sales'] Measure

In this exercise, you will create a rule for ['Sales'] by multiplying units and price.

Exercise Steps

1. Open the Office Plus application.
2. Click the Cubes button on the BI File and Database toolbar (or, select Cubes and Data Areas from the BI Office Plus menu). The Cubes and Data Areas dialog box opens.
3. Select the TOTSALES cube.
4. Click the Edit Rules button.
5. Click the New Rule button. The Edit Cube Rules dialog box opens, where you will write the new rule.
6. Click once in the Target section to place the cursor there.
7. Click the Internal Reference button. The Cell Reference dialog box opens, where you can define an area within the TOTSALES cube.
8. Click the Select One Element button next to the Measures dimension in the Cell Reference dialog box. The dimension browser for the Measures dimension opens.
9. Click the element Gross Margin to display its children.
10. Select Sales from the displayed elements.
11. Click OK.
12. Verify that the Cell Reference dialog box looks like the example shown below:

| Measures | | | | | | | | | | Select One Element |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| Preview: | | | | | | | | | |
| | | | | | | | | | |

13. Click OK.
14. Verify that the Edit Cube Rule dialog box reflects the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[MEASURES: 'Sales']</td>
</tr>
</tbody>
</table>
15. Click once within the **Rule** section to place the cursor there.

16. Click the **Internal Reference** button. The Cell Reference dialog box opens.

17. Click the **Select One Element** button next to the Measures dimension in the Cell Reference dialog box. The dimension browser for the Measures dimension opens.

18. Select **Units** from the displayed elements.

19. Click **OK**.

20. Verify that the Cell Reference dialog box looks like the example shown below:

21. Click **OK**. The Edit Cube Rule dialog box opens.

22. Click the **Multiply** operator button in the Edit Cube Rule dialog box. Or, you can position the cursor in the **Rule** section of the dialog (after "[Units]") and type an asterisk (*).

23. Click the **Internal Reference** button.

24. Click the **Select One Element** button next to the Measures dimension in the Cell Reference dialog box. The dimension browser for the Measures dimension opens.

25. Select **Avg. Price** from the displayed elements.

26. Click **OK** to confirm.

27. Verify that the Cell Reference dialog box looks like the example shown below:

28. Click **OK**. The Edit Cube Rule dialog box opens.

29. Click the **Divide** operator button (or type "/" at the end of the rule in the **Rule** section).

30. Type (or use the numeric buttons to enter) 1,000 after the "/".

31. Verify that the Edit Cube Rule dialog box reflects the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[MEASURES:'Sales']</td>
</tr>
</tbody>
</table>
### Lesson 7: Creating a Price Volume Rule

#### Field Description

| Rule | [MEASURES:’Units’]*[MEASURES:’Avg. Price’]/1000 |

32. Click **OK** to accept the rule. The Cube Rules dialog box opens showing that a rule has been written that affects the area [‘Sales’] in the TOTSALES cube:

33. Click **OK** to save the rule and close the Cube Rules dialog box. **Note:** If you click **Cancel**, the rule will not be saved yet and will be removed.

34. Verify the rule has been saved by browsing the TOTSALES cube.

### Editing Cube Rules for Base Elements

In this continuing scenario, when you want the rule to apply only to base level elements, you must first prevent the Units * Price calculation from applying to North America and other consolidated elements.

To designate the rule to apply only to base level elements, you must select the Basic cells option in the affected cells area of the Edit Cube Rule dialog box.

**Exercise 7.2: Edit the [‘Sales’] Rule**

In this exercise, you will edit the Price Volume Rule you created in Exercise 2.1.

**Exercise Steps**

1. Open the Cube Rules dialog box for the TOTSALES Cube.
2. Select **Cubes and Database Areas** on the BI OLAP Office Plus menu. The Cubes and Data Areas dialog box opens.
3. Select the **TOTSALES** cube.
4. Click the **Edit Rules** button in the Cubes and Data Areas dialog box. The Cube Rules dialog box opens showing the rules for the TOTSALES cube.
5. Double-click the [‘Sales’] rule. The Edit Cube Rules dialog box for that rule opens.
6. Select the option **Basic cells** in the Affected cells area.
7. Click **OK** to accept the rule.
8. Click **OK** to close the Cube Rules dialog box.
9. Click **OK** to close the Cubes and Data Areas dialog box.
10. Click the **Recalculate** button (or press F9) to recalculate the view.

© 2013 Infor Education
11. Examine the data to confirm that the Sales data for North America and the other consolidations are now correct.

Defining Additional Rules for Consolidated Elements

In this continuing scenario, when you want the rule to apply only on consolidated elements, you cannot apply a rule for Avg. Price (at the base level) using the following formula:

\[
\text{Average Price} = \frac{\text{Sales}}{\text{Units}} \times 1000
\]

As the average price and unit data is input on the base level and sales are calculated you must define an additional rule for the Avg. Price at the consolidated level. Otherwise, the price would be aggregated on the consolidated cells which make no sense. You want to see weighted average price on these cells.

To designate the rule to apply only to consolidated level elements, you must select the Consolidated cells option in the affected cells area of the Edit Cube Rule dialog box.

Exercise 7.3: Create an ['Average Price'] Rule

In this exercise, you will create the ['Average Price'] rule using the following formula:

\[
\text{Average Price} = \frac{\text{Sales}}{\text{Units}} \times 1000
\]

Exercise Steps

1. Leave your current Area view of the data open.
2. Create the following Average Price rule:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[MEASURES:'Avg.Price']</td>
</tr>
<tr>
<td>Rule</td>
<td>[MEASURES:'Sales']/[MEASURES:'Units']*1000</td>
</tr>
<tr>
<td>Affected Cells</td>
<td>Select the <strong>Consolidated cells</strong> option.</td>
</tr>
</tbody>
</table>

3. Close all open dialog boxes to save the rules to the cube.
4. Click the **Recalculate** button (or press F9) to recalculate your data view.
Exercise 7.4: Create a ['Variable Costs'] Rule

In this exercise, you will create the ['Variable Costs'] rule using the following formula:
Variable Costs = Units * Unit Cost / 1000

Exercise Steps

1. Create the following Variable Costs rule:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>['MEASURES:’Variable Costs’]</td>
</tr>
<tr>
<td>Rule</td>
<td>[MEASURES:’Units’][MEASURES:’Unit Cost’]/1000</td>
</tr>
<tr>
<td>Affected Cells</td>
<td>Select the Basic cells option.</td>
</tr>
</tbody>
</table>

2. Verify the results by referring to the screen shot that follows:

<table>
<thead>
<tr>
<th>Totsales</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTVSBUD Actual</td>
<td></td>
</tr>
<tr>
<td>MONTHS January</td>
<td></td>
</tr>
<tr>
<td>PRODUCT ProView VGA 12</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REGION</th>
<th>...</th>
<th>MEASURES</th>
<th>Units</th>
<th>Unit Cost</th>
<th>Variable Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>World</td>
<td></td>
<td></td>
<td>16.645</td>
<td>15.881</td>
<td>8.510</td>
</tr>
<tr>
<td>North/South America</td>
<td></td>
<td></td>
<td>6.150</td>
<td>3.585</td>
<td>3.149</td>
</tr>
<tr>
<td>Europe</td>
<td></td>
<td></td>
<td>6.877</td>
<td>7.157</td>
<td>3.408</td>
</tr>
<tr>
<td>Middle Region</td>
<td></td>
<td></td>
<td>1.488</td>
<td>1.013</td>
<td>752</td>
</tr>
<tr>
<td>Germany</td>
<td></td>
<td></td>
<td>897</td>
<td>500</td>
<td>449</td>
</tr>
<tr>
<td>France</td>
<td></td>
<td></td>
<td>591</td>
<td>513</td>
<td>303</td>
</tr>
<tr>
<td>North Region</td>
<td></td>
<td></td>
<td>947</td>
<td>1.533</td>
<td>484</td>
</tr>
<tr>
<td>Denmark</td>
<td></td>
<td></td>
<td>313</td>
<td>511</td>
<td>160</td>
</tr>
<tr>
<td>Norway</td>
<td></td>
<td></td>
<td>304</td>
<td>510</td>
<td>155</td>
</tr>
<tr>
<td>Sweden</td>
<td></td>
<td></td>
<td>330</td>
<td>512</td>
<td>169</td>
</tr>
<tr>
<td>Benelux Region</td>
<td></td>
<td></td>
<td>1.122</td>
<td>1.539</td>
<td>575</td>
</tr>
<tr>
<td>South Region</td>
<td></td>
<td></td>
<td>2.191</td>
<td>2.050</td>
<td>1.122</td>
</tr>
<tr>
<td>West Region</td>
<td></td>
<td></td>
<td>929</td>
<td>1.023</td>
<td>475</td>
</tr>
<tr>
<td>Other Countries</td>
<td></td>
<td></td>
<td>3.818</td>
<td>5.138</td>
<td>1.953</td>
</tr>
<tr>
<td>European Union</td>
<td></td>
<td></td>
<td>6.287</td>
<td>6.647</td>
<td>3.209</td>
</tr>
</tbody>
</table>
Exercise 7.5: Create a ['Unit Cost'] Rule

In this exercise, you will create the ['Unit Cost'] rule.

Exercise Steps

1. Create the following Unit Cost rule:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>['MEASURES:'Unit Cost']</td>
</tr>
<tr>
<td>Rule</td>
<td>[MEASURES:'Variable Cost']/[MEASURES:'Units']*1000</td>
</tr>
<tr>
<td>Affected Cells</td>
<td>Select the <strong>Consolidated cells</strong> option.</td>
</tr>
</tbody>
</table>

2. Verify the results by referring to the screen shot that follows:

Other Uses for this Method

You can use this method for interest rates and interest income. You can calculate budgeted interest income by supplying the budgeted interest rate and average balance. For Actual data, you will most likely have all this information available to you, so the rules would apply only to Budget data. However, you can calculate an average interest rate for all consolidated values, both Actual and Budget, as shown below:

Average Rate = (Interest Income / Average Balance) /12
Lesson 8: Creating an Allocation Rule

Estimated Time
- 30 minutes

Learning Objectives
After completing this lesson, you will be able to:
- Create and edit an Allocation rule.

Topics
- Creating and Editing an Allocation Rule
Creating and Editing an Allocation Rule

In this topic, a specific business scenario will be used with the Alloc1 data model to discuss rule-related information.

Scenario

“I receive most of my income statement data by product and region. I get Manufacturing Overhead and Marketing/Admin Cost totals for each month. How can I allocate these totals to each region based on product sales data for that region so I can include these costs on my region/product income statements?”

It is very common for data from various sources to have different detail levels. In this example, overhead cost totals are kept by month only, while monthly sales are associated with a product and the region that sold it. We must calculate the portion of the cost that applies to each region/product combination based on the portion of sales. The cost data must then be fed into the BI OLAP cube where the rest of the income statement data resides.

Rules can be used to put data from multiple cubes into a single mathematical formula that can be applied to the entire cube of a part thereof. As such, creating cube rules to create a cube that holds the overhead cost totals and allocates the portion of the cost systematically alleviates the need for cumbersome manual processes that require a calculator or an Excel worksheet.

Considerations:

- The LOCAL/ALLOC1 server contains the TOTSALES and OVERHEAD cubes.
  - The TOTSALES cube is the main data cube. Sales and Cost data and income statements are stored here.
  - The OVERHEAD cube contains Total Manufacturing Overhead and Marketing and Admin. Costs for the company.
- Manufacturing Overhead and Marketing and Admin. must be fully allocated to the field based on Sales data.
- You must determine the correct percentage of Sales within the TOTSALES cube and multiply it with the corresponding overhead costs in the OVERHEAD cube. The result, Allocated Cost, must be stored in the TOTSALES cube, using the basic calculation shown below:

\[
\text{Allocated Cost} = \text{Total Cost} \times \left( \frac{\text{Sales}}{\text{Total World Sales}} \right)
\]

Below is the rule that you would use to allocate Manufacturing Overhead in the OVERHEAD cube to the Allocated Manuf. Ovhd. measure in the TOTSALES cube for every product, every region, every month and every year:

\[
\text{Allocated Manuf. Ovhd.} = \text{DB('OVERHEAD',!years,!actvsbud,!months,'Manufacturing Overhead')} \times \left( \frac{\text{Sales}}{\text{World',Total','Sales'}} \right)
\]
Exercise 8.1: Create a Rule for the TOTSALES Cube

In this exercise, you will create the rules for the TOTSALES cube.

Exercise Steps

1. Click the Cubes button on the BI File and Database toolbar (or, select Cubes and Data Areas from the BI Office Plus menu). The Cubes and Data Areas dialog box opens.
2. Select the TOTSALES cube.
3. Click the Edit Rules button on the toolbar. The Cube Rules dialog box displays. Because there are no rules for this cube yet, none are shown.
4. Click the New Rule button. The Edit Cube Rules dialog box opens.
5. Click once in the Target section to place the cursor there.
6. Click the Internal Reference button.
7. Click the Select One Element button next to the Measures dimension.
8. Select the Allocated Manuf. Ovhd. element.
9. Click OK.
10. Verify the Cell Reference dialog box looks like the example shown below:

   ![Cell Reference Dialog Box](image)

   11. Click OK to accept the selection. The rule target Allocated Costs = is created.
12. Click once in the Rule section to place the cursor there.

   ![External Reference Dialog Box](image)

   14. Click OK to accept the selection. The rule target Allocated Costs = is created.
15. Verify the Cell Reference dialog box looks like the example shown below:

   ![Cell Reference Dialog Box](image)

   16. Click OK to accept the selection. The rule target Allocated Costs = is created.
17. Because the rule is for the TOTSALES cube, you must refer to the OVERHEAD cube to retrieve the amount to be allocated.

   ![External Reference Dialog Box](image)

   18. Click OK to accept the selection. The rule target Allocated Costs = is created.
19. Select the OVERHEAD cube, which is the cube to which you want to make an external reference.
20. Click OK. The Cell Reference dialog box opens.
21. Click Select One Element next to the ovhdmeas dimension.
22. Select Manufacturing Overhead.

   ![Cell Reference Dialog Box](image)

   23. Click OK to accept the selection. The rule target Allocated Costs = is created.
19. Click **OK**.
20. Click **OK** to close the Cell Reference dialog box.
21. Verify that the Edit Cube Rules dialog box reflects the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[MEASURES:’Allocated Manuf. Ovhd.’]</td>
</tr>
<tr>
<td>Rule</td>
<td>DB(’OVERHEAD’,!YEARS,!ACTVSBUD,!MONTHS,’Manufacturing Overhead’)</td>
</tr>
</tbody>
</table>

22. Click once to place your cursor at the end of the existing rule in the Rule field.
23. Click the Multiply button, or type an asterisk (*).
24. Click the Internal Reference button.
25. Click the Select One Element button next to the Measures dimension.
26. Select the **Sales** element.
27. Click **OK**.
28. Click **OK** to close the Cell Reference dialog box.
29. Verify that the Edit Cube Rules dialog box reflects the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[MEASURES:’Allocated Manuf. Ovhd.’]</td>
</tr>
<tr>
<td>Rule</td>
<td>DB(’OVERHEAD’,!YEARS,!ACTVSBUD,!MONTHS,’Manufacturing Overhead’)*[MEASURES:’Sales’]</td>
</tr>
</tbody>
</table>

30. Click **Divide** or type / at the end of the rule.
31. Click the Internal Reference button.
32. Click the Select One Element button next to the Measures dimension.
33. Select the **Sales** element and click **OK**.
34. Click the Select One Element button next to the Region dimension.
35. Select the **World** element and click **OK**.
36. Click the Select One Element button next to the Product dimension.
37. Select the **Total** element and click **OK**.
38. Click **OK** to accept the cell references.
39. Click the Check Syntax button to be sure that the rule is written correctly.
40. Verify the Edit Cube Rules dialog box displays the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[MEASURES:’Allocated Manuf. Ovhd.’]</td>
</tr>
</tbody>
</table>
Field | Description
--- | ---
Rule | DB('OVERHEAD',!YEARS,!ACTVSBUD,!MONTHS,'Manufacturing Overhead')*[MEASURES:;'Sales']/*[REGIONS:;'World',PRODUCTS:;'Total',MEASURES:;'Sales']

41. Correct any problems, if necessary.
42. Click **OK** to accept the rule. The Cube Rules dialog box reopens.
43. Click **OK** to close the **Cube Rules** dialog box.
44. Close the **Cubes and Data Areas** dialog box.
45. Open the **CHKALLOC.XLS** worksheet in the same directory as the ALLOC1 database.
46. View the worksheet to determine how this rule works.

**Editing the Rule to Apply Only to Basis Level Cells**
In a continuation of the scenario used throughout this lesson, you must now correct a problem in the worksheet from the prior exercise. You will need to force this rule to apply only to basis level cells. Any cells having a consolidated element as a key or identifier must not calculate their data with the rule, rather they must add up the subordinate cells.

**Exercise 8.2: Editing the ['Allocated Manuf. Ovhd.'] Rule**
In this exercise, you will edit the ['Allocated Manuf. Ovhd.'] rule.

**Exercise Steps**
1. Click the **Cubes** button on the BI File and Database toolbar (or, select **Cubes and Data Areas** from the BI Office Plus menu). The Cubes and Data Areas dialog box opens.
2. Select the **TOTSALES** cube.
3. Click the **Edit Rules** button. The Cube Rules dialog box displays.
4. Double-click the rule for ['Allocated Manuf. Ovhd.']. The Edit Cube Rules dialog box will display the created rule.
5. Select the **Basic Cells** option.
6. Click **OK** to close the Edit Cube Rules dialog box and accept the rule.
7. Close the Cube Rules dialog box.
8. Close the Cubes and Data Areas dialog box.
9. Examine the CHKALLOC.XLS worksheet you opened in the prior exercise.
10. Press **F9** to recalculate the worksheet.
11. Examine the numbers displayed in the **1st Quarter** fields.
Exercise 8.3: Create a Rule for ['Allocated Mktg. and Admin. ']

In this exercise, you will create a rule for ['Allocated Mktg. and Admin. '] using the same steps from the previous two exercises.

Exercise Steps

1. Create and edit a rule for ['Allocated Mktg. and Admin. '] using the rule criteria provided below.

   Follow the same steps in Exercise 3.1 and Exercise 3.2.

   DB('OVERHEAD',!years,!actvsbud,!months,'Marketing and Admin.' ) *
   (['Sales'] ['World' ,'Total' ,'Sales' ])

2. Verify the results display like the following:

   ![Cube Rules Table]

<table>
<thead>
<tr>
<th>Rules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules of the 'TOTSALES' cube</td>
<td></td>
</tr>
<tr>
<td>[MEASURES:'Allocated Manuf. Ovhd']</td>
<td></td>
</tr>
<tr>
<td>[MEASURES:'Allocated Mktg. and Admin. ']</td>
<td></td>
</tr>
</tbody>
</table>
Lesson 9: Currency Translation

Estimated Time

- 1 hour

Learning Objectives

After completing this lesson, you will be able to:

- Calculate currency with the Exchange Rate cube.

Topics

- Calculating Currency
Calculating Currency

In this topic, a specific business scenario will be used to discuss rule-related information.

Scenario

“My company has offices throughout the world. Each country submits data in its local currency. I need to consolidate the data and report in my home currency as well as in each country’s local currency. I do not have the time to manually translate all data.”

Calculating Home Currency and Local Currency with the Exchange Rate Cube

Considerations:

- Each office reports in its own currency.
- You must report in local currency and in home currency.
- You must multiply or divide each submitted value by some factor to arrive at the home office currency. This factor changes regularly.
- Certain items (units, headcount, etc.) are not subject to translation.
- Data from offices that report in different currencies cannot be summed in the local currency.

In this example, assume that each office uses a different currency. Each office will enter a unique rate each month for conversion purposes. The same rates are used for Actual and Budget data.

- Use the TOTSALES cube from the Currency data model to store your main data (income statement, etc.). In this cube, you need a place to display values in your home currency, such as USD or EUR, as well as a place to keep the local currency data submitted by the foreign offices. This is the currency dimension.
- Use the EXCHRATE cube to store the exchange rates.

Submitted data must be translated to home currency based on the rate for the current month. Local currency data must be divided by the proper rate in the EXCHRATE cube and then calculated as home currency data in the main cube. You will use a cube rule because the calculation involves a second cube and division by a variable rate. The rule must be created in the internal cube TOTSALES, since the result will be displayed as noted below:

Home Office = Local Currency / Exchange Rate

Unit, headcount and statistical data (anything not currency-related) must be excluded from this translation. In these cases, the following applies:

Home Office Currency = Local Currency
Exercise 9.1: Create a Rule for the TOTSALES Cube

In this exercise, you will create the currency conversion rules for calculating home currency in the TOTSALES cube from the Currency data model.

Exercise Steps

1. Click the Cubes button on the BI File and Database toolbar (or, select Cubes and Data Areas from the BI Office Plus menu). The Cubes and Data Areas dialog box opens.
2. Select the TOTSALES cube.
3. Click the Edit Rules button on the toolbar. The Cube Rules dialog box displays.
4. Click the New Rule button. The Edit Cube Rules dialog box opens.
5. Click once in the Target section to place the cursor there.
6. Click the Internal Reference button.
7. Click the Select One Element button next to the Currency dimension.

While this rule is for Home Office Currency, it applies to all elements of the other dimensions.

9. Click OK.
10. Click OK to accept the selection. The Edit cube Rule dialog opens.
11. Click once in the Rule section to place the cursor there.
12. Click the Internal Reference button.
13. Click the Select One Element button next to the Currency dimension.
14. Select the Local element.
15. Click OK to return to the Cell Reference dialog box.
16. Click OK to accept the selection.
17. Click the Divide operator button (or type / at the end of the rule).
18. Click the External Reference button.
19. Click the Cube button.
20. Select the EXCHRATE cube.
21. Click OK.
22. Click OK to accept the selection. Note: This rule applies to all years, months and regions in both the TOTSALES and the EXCHRATE cube.
23. Click OK to accept the rule.
24. Click the Save button to save your changes to the TOTSALES cube.
25. View the EXCHRATE cube.
26. Verify the results by referring to the screen shot that follows:

27. View the rule results in the SALES cube.

28. Verify the results by referring to the screen shot that follows:

29. Click the **Recalculate View** button (or press **F9**) to see the rule results.
Editing the TOTSALES Rule for [Home Office] to Apply to Basis Elements Only

In this continuing scenario, you want the rule to apply only to base-level elements; i.e. you must first prevent the currency translation calculation from applying to North America and other consolidated elements. At the consolidated level, Home Office Currency should be the sum of the translated non-consolidated values.

To define a rule as applicable only to base-level elements, enable the ‘Basic Elements’ in the affected cells section of rules editor.

Exercise 9.2: Edit the TOTSALES Rule for [Home Office]

In this exercise, you will edit the rule you created in the previous exercise.

Exercise Steps

1. Edit the TOTSALES rule for [Home Office] by enabling the ‘Basic Elements’ in the affected cells section of rules editor.
2. Click OK to accept the rule.
3. Click OK to exit the Edit Cube Rules dialog box.
4. Close the Cubes and Data Areas dialog box.
5. View the TOTSALES cube again.

Create a New Rule to Address Units and Local Currency at Consolidated Levels

In this continuing scenario, Unit data and Local Currency at the consolidated levels must be addressed. You will first correct data for Units because Units are not converted, but still should be shown in Home Currency. You will then correct the data for Local Currency.

The calculation should be: Home Office Currency = Local Currency

The order of the rules governs the order in which they are evaluated. If there are two rules that apply to the same area of the cube (in this case, Home Office) the first one in the list of rules is evaluated and that calculation is performed. Any others are ignored.
Exercise 9.3: Create a New TOTSALES Rule for [Home Office]

In this exercise, you will create another TOTSALES rule for [Home Office].

Exercise Steps

1. Click the Cubes button on the BI File and Database toolbar (or, select Cubes and Data Areas from the BI Office Plus menu). The Cubes and Data Areas dialog box opens.

2. Select the TOTSALES cube.

3. Click the New Rule button.

4. Create the Home Office rule displayed in the screen shot below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[CURRENCY:'Home Office',MEASURES:'Units']</td>
</tr>
<tr>
<td>Rule</td>
<td>[CURRENCY:'Local',MEASURES:'Units']</td>
</tr>
<tr>
<td>Affected Cells</td>
<td>Select the All cells option.</td>
</tr>
</tbody>
</table>

For best performance, always select the All cells setting when you have only simple rules, such as A=B. This instructs the rules engine to calculate all data at one time, so the OLAP server doesn’t need to complete the aggregation after.

5. Verify that the Cube Rules dialog box displays as shown in the following example:

6. Click and drag the ['Home Office',MEASURES:'Units'] rule above the general rule for ['Home Office'].

© 2013 Infor Education
Creating Rules to Apply Only to Consolidated Cells

In this continuing scenario, you have created rules that deal with the majority of the data values. You need to ensure that users are viewing accurate Local Currency values at some of the consolidated levels.

The measure Unit in the data model may be different depending on the Product. It could be volume, tons, hours etc. Therefore, it doesn’t make sense to summarize the data for units above different product levels.

Because the rule will only apply to consolidated values, the Consolidated cells option must be selected in the Affected cells section.

**Exercise 9.4: Create the ['Germany', 'Local Currency'] Rule**

In this exercise, you will create a rule and ensure it applies only at consolidated levels.

**Exercise Steps**

1. Create the following rule for the ['Germany', 'Local Currency']. **Note:** The function #NA should be used.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[CURRENCY:'Local',REGION:'Germany']</td>
</tr>
<tr>
<td>Rule</td>
<td>#NA</td>
</tr>
</tbody>
</table>

2. Verify the results by referring to the screen shot that follows:
The order of the regional rules and the ['Home Office', 'Units'] rule as they relate to each other is irrelevant because these rules are mutually exclusive. However, the general ['Home Office'] rule must appear after the ['Home Office', 'Units'] rule because these rules cover overlapping areas. The ['Home Office', 'Units'] rule must take precedence over the general ['Home Office'] rule, wherever it applies.

Using Rules Function STET
This function states: for a specified area ignore all rules and apply the properties that the area would have had if no rules were written. This function will be used for exceptions, such as for data that should not be affected by global rules.

Exercise 9.5: Edit the ['Germany', 'Local Currency'] Rule
In this exercise, you will edit the rule with the STET rules function.

Exercise Steps
1. Edit the ['Germany', 'Local Currency'] rule as indicated by the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[CURRENCY:'Local',REGION:'Germany']</td>
</tr>
<tr>
<td>Rule</td>
<td>IF(!MEASURES@='Units',STET,'#NA')</td>
</tr>
</tbody>
</table>

2. Display the results.

The number of rules to define business logics has nothing to do with performance of cube. Sometimes it is better to split one rule into several rules. If-Statements can decrease performance because the requests cannot be sending to the server at once.

3. Optimize the rule by splitting the rule into 2 rules:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[CURRENCY:'Local',REGION:'Germany', MEASURES:'Units']</td>
</tr>
<tr>
<td>Rule</td>
<td>STET</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[CURRENCY:'Local',REGION:'Germany']</td>
</tr>
<tr>
<td>Rule</td>
<td>#NA</td>
</tr>
</tbody>
</table>

4. Display the results.
Exercise 9.6: Create Multiple Currency Rules

In this exercise, you will create currency rules for multiple regions. Refer to the previous exercises in this lesson as necessary.

Exercise Steps

1. Create currency rules for the following regions:
   - European Union
   - North/South America
   - North America
   - South America
   - Europe
   - Other Countries
   - Middle Region
   - North Region
   - Benelux Region
   - South Region
   - West Region

2. Verify the results by referring to the screen shot that follows:

   ![Cube Rules](image)

   The number of rules to define business logics has nothing to do with performance of cube. In this case it is better for maintaining just to create one rule for all selected regions. If you choose single or multiple elements or subsets in the target area this would not affect the performance in this case.

3. Create a subset ‘C-Elements’ in the dimension ‘REGION’.

4. Just write one rule which is using this subset. Delete all other rules which have only a single region as selected element in the target area.
Adding a Rule to the EXCHRATE Cube

While this rule is not required for the proper function of other rules, it is a good business practice. This rule will help prevent erroneous consolidated values. Because this rule will affect all consolidated elements, no single elements will be selected. To show that all elements are affected, square brackets will be placed in the Target.

Exercise 9.7: Adding a Rule to the EXCHRATE Cube

In this exercise, you will add a rule to the EXCHRATE cube.

Exercise Steps

1. Select the EXCHRATE cube from within the Cubes and Data Areas dialog box.
2. Click the Edit Rules button.
3. Click the New Rules button.
4. Click the Internal Reference button.
5. Click the OK button to accept the selection.
6. Set all consolidated elements to #NA, similar to what you did in the TOTSALES cube.
7. Click OK to accept the rule.
8. Verify that the Cube Rules dialog box looks like the example shown below:

   ![Cube Rules Dialog Box Example]

9. Close the Cube Rules dialog box.
10. Verify the results by referring to the screen shot that follows:

   ![Screen Shot Example]

11. Save the cubes.
Other Uses for this Method

You can use this method for unit cost information. You can keep unit cost data in a cube similar to the EXCHRATE cube. This cube would include the product dimension in lieu of the region dimension (or in addition to it, if different regions have different prices for the same items).

Other Considerations

Your company may have multiple offices that report in the same currency. You can eliminate duplicate exchange rate entries by using an attribute table (for the office/region dimension) to look up the currency used and only entering rates by currency, instead of by office. Refer to Lesson 5: Attribute Tables for more information.

If your company uses different exchange rates for balance sheet and income statement data, you may need to add an additional dimension to the EXCHRATE cube that distinguishes between the two rates. For example, this new dimension may have elements called Average Rate and Month-end Rate. The rules would use the appropriate rate, based on the account type from an attribute table for the account dimension.

You may want to add an additional month’s dimension to the EXCHRATE rate cube to account for changes over time.

You may want to add the actvsbud dimension to account for the use of different rates for actual and budget data.
Operative vs. Currency Variance

Given the power and ability to perform translations as above, you may want to enhance your model to allow for the analysis of operative vs. currency variance. Distinguishing between operative and currency variance means distinguishing between the variance caused by operations (e.g. we sold fewer units than expected) and the variance that occurs due to currency projections (e.g. the value of our local currency increased in value over what we had expected, inflating our sales numbers when we reported them in the home office currency).

To do the above analysis, the exchange rate cube must contain a fourth dimension, actvsbud. The basic currency rule is essentially the same:

\[
\text{['Home Office Currency']} = \text{['Local Currency']} / \text{DB('EXCHRATE', !actvsbud, !months, !years, !region)}
\]

You would then need to add three elements to your actvsbud dimension, as follows:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Element Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual at budget rates</td>
<td>Basis level element</td>
</tr>
<tr>
<td>Variance from Operations</td>
<td>Consolidated element defined as: &quot;Actual at budget rates&quot; - &quot;Budget&quot;</td>
</tr>
<tr>
<td>Variance from Exchange Rate</td>
<td>Consolidated element defined as: &quot;Actual&quot; - &quot;Actual at budget rates&quot;</td>
</tr>
</tbody>
</table>

The final step is to write the rule defining Actual at budget rates, as follows:

\[
\text{['Home Office Currency', 'Actual at Budget-Rate']} = \text{['Local Currency', 'Actual']}/\text{DB ('EXCHRATE', 'Budget', !months, !years, !region)}
\]

Because this rule is an exception to the main rule, it must be above the main rule in the list of rules.
Lesson 10: Attribute Tables

Estimated Time

- 1 hour

Learning Objectives

After completing this lesson, you will be able to:

- Look up product cost data using attribute tables.

Topics

- Look up product cost data using attribute tables
Looking Up Product Cost Data with Attribute Tables

In this topic, a specific business scenario will be used to discuss rule-related information using the Lookup data model.

Scenario

"I need to calculate some product cost data based on standard costs. Each of my product groups has different standard cost values. I have about 1000 products broken into about 15 groups. In other systems that I have used in the past, I had to create a lookup table to do something like this. Can I do this with BI OLAP?"

In this scenario, each product group has a different Standard Cost Rate. Rather than repeating the cost for each of the products in the group in the main data cube, you can create an attribute table. A maximum of three attribute tables may be defined, each of them may contain up to 49 attributes. These attributes may be account types, regional managers, etc. In this case, the attribute to be defined is the Standard Cost Type that should be applied to each product. The rate can then be referenced from the Standard Cost Rate cube and used in any calculation rules.

Considerations:

- TOTSALES is the main data cube in the Lookup data model; unit and cost data is stored here.
- You must calculate Variable Costs data based on Unit Sales and Standard Cost Rates.
- Standard Cost Rates vary by product and year.
- Standard Cost Rates are stored in the RATETAB cube.
- Units are stored in whole numbers. Standard Cost Rates are in whole dollars. All of the Revenue and Cost data is stored in thousands. Therefore, you must divide the result of the calculation by 1000.
- Because each individual product has a different rate, the calculation must be done at the lowest, or basis level, and then consolidated.

The calculation for Variable Cost is:

\[
\text{Variable Costs} = \text{Units} \times \text{Standard Cost Rate}
\]

The following table shows an example of the Standard Cost Rates to be applied. The RATETAB Cube contains these rates.

<table>
<thead>
<tr>
<th>Standard Cost Type</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>STDCST01</td>
<td>553</td>
<td>574</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>STDCST02</td>
<td>580</td>
<td>534</td>
<td>612</td>
<td>612</td>
</tr>
<tr>
<td>STDCST03</td>
<td>0</td>
<td>2,211</td>
<td>2,600</td>
<td>2,989</td>
</tr>
<tr>
<td>STDCST04</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4,515</td>
</tr>
<tr>
<td>STDCST05</td>
<td>1,161</td>
<td>1,278</td>
<td>1,247</td>
<td>1,387</td>
</tr>
</tbody>
</table>
Below is an example of some of the products and their Standard Cost Types:

<table>
<thead>
<tr>
<th>Product</th>
<th>Standard Cost Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProView VGA 12</td>
<td>STDCST01</td>
</tr>
<tr>
<td>ProView VGA 14</td>
<td>STDCST08</td>
</tr>
<tr>
<td>ProView VGA 15</td>
<td>STDCST01</td>
</tr>
</tbody>
</table>

To associate the product with a Standard Cost, an attribute table must be created for the Product dimension. Information in this table will be saved with the dimension.

**Exercise 10.1: View the Attribute Table Structure for the Product Dimension**

In this exercise, you will view the structure of the attribute table for the product dimension.

**Exercise Steps**

1. Click the **Dimensions** button in the Database toolbar. The Dimensions and Subsets dialog box opens.
2. Select the **Product** dimension.
3. Click the Attribute Tables button to view the information about the table.
4. Click OK to close the Attribute Tables dialog box.
5. Double-click the Product dimension in the Dimensions and Subsets dialog box to select it.
6. Click the Attribute **Tables** button (in the upper left-hand corner of the main window in the Product dimension dialog box).

![Edit Dimension 'PRODUCT'](image)

7. Select **Attribute Table 1**.
8. Expand the dimension list to see the Standard Cost Types associated with the products.
9. Click **Cancel** to close the Product dimension dialog box.
10. Click **Close** to close the Dimensions and Subsets dialog box.

**Exercise 10.2: Building the [‘Variable Costs’] Rule**

In this exercise, you will create the [‘Variable Costs’] Rule.

**Exercise Steps**

1. Click the **Cubes** button on the BI File and Database toolbar (or, select **Cubes and Data Areas** from the BI Office Plus menu). The Cubes and Data Areas dialog box opens.
2. Select the **TOTSALES** cube.
3. Click the **Edit Rules** button on the toolbar. The Cube Rules dialog box displays. Because there are no rules for this cube yet, none are shown.
4. Click the **New Rule** button. The Edit Cube Rules dialog box opens.
5. Click once in the **Target** section of the dialog box to place the cursor there.
6. Click the **Internal Reference** button.
7. Click the **Select One Element** button next to the Measures dimension.
8. Select **Variable Costs**.
9. Click **OK**.
10. Click **OK** to accept the selection.
11. Click once in the **Rule** section of the dialog box to place the cursor there.
12. Click the **Internal Reference** button.
13. Click the **Select One Element** button next to the Measures dimension.
14. Select **Units**.
15. Click **OK**.
16. Click **OK** to accept the selection.

17. Click the **Multiply** operator button, or type an asterisk (*) the end of the rule.

18. Click the **External Reference** button to refer to the RATETAB Cube.

19. Click the **Cube** button in the Cell Reference dialog box.

20. Select the **RATETAB** cube.

21. Click **OK**.

22. Click **OK** to select the reference.

23. Verify that the Edit Cube Rule dialog box reflects the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[MEASURES:’Variable Costs’]</td>
</tr>
<tr>
<td>Rule</td>
<td>[MEASURES:’Units’]*DB(‘RATETAB’,!YEARS,!Std cst)</td>
</tr>
</tbody>
</table>

24. Click the **Check Syntax** button. You will receive an error message because BI OLAP currently has no way of matching up the elements in the std cst dimension with anything in the TOTSALES cube. You must edit the rule to include the lookup to the attribute table.

25. Highlight the "!Std cst" portion in the Rule section of the Edit Cube Rule dialog box.

26. Select the **GETATTR()** function button to add the function to the rule. This function returns the content of a specified field in an attribute table.

27. Edit the GETATTR function within the Rule section as shown below:

   \[
   \text{GETATTR(‘Product’,!product,1,’Ratetype’)}
   \]

   **Description:**
   
   ‘Product’ is the name of the dimension,

   !product is the element,

   ‘Ratetype’ is the field name of the attribute.

28. Verify that the Edit Cube Rule dialog box reflects the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[MEASURES:’Variable Costs’]</td>
</tr>
<tr>
<td>Rule</td>
<td>[MEASURES:’Units’]*DB(‘RATETAB’,!YEARS,GETATTR(‘Product’,!product,1,’Ratetype’))</td>
</tr>
</tbody>
</table>

29. Click the **Check Syntax** button to confirm the syntax.
Exercise 10.3: Editing the ['Variable Costs'] Rule

In this continuation of the prior exercise, you will edit the ['Variable Costs'] rule so that it is only performed on basis level elements and it will divide by 1000 to convert the Variable Cost data to thousands.

Exercise Steps

1. Select the Basic cells option in the Affected cells area to ensure this rule applies only to basis level elements.
2. Type /1000 at the end of the rule in the Rule section, as shown in the screen shot below:

   ![Rule Screen Shot](image)

   ```
   'MEASURES:Units']/DB('RATETAB',YEARS,GETATTR('Product', product, 'Ratetype'))/1000)
   ```

3. Click OK to accept the rule.
4. Click OK to close the Cube Rules dialog box.
5. View the TOTSALES cube to verify the rule.
6. Verify the results by referring to the screen shot for data model Lookup, cube Totsales and year 2013 that follows:

   ![Data Model Lookup](image)

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>ACTYSBUD Actual</th>
<th>MEASURES</th>
<th>Sales</th>
<th>Variable Costs</th>
<th>Gross Margin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>11,059</td>
<td>21,229</td>
<td>0</td>
<td>10,726</td>
<td></td>
</tr>
<tr>
<td>Total Monitors</td>
<td>3,364</td>
<td>3,361</td>
<td>0</td>
<td>1,501</td>
<td></td>
</tr>
<tr>
<td>ProView VGA 12</td>
<td>3,364</td>
<td>3,361</td>
<td>1,660</td>
<td>1,501</td>
<td></td>
</tr>
<tr>
<td>ProView VGA 14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ProView VGA 15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ProView SVG A 14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ProView SVG A 15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ProView SVG A 17+</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Ultraview SVG A 19+</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Total Servers</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Total Desktops</td>
<td>5,009</td>
<td>11,642</td>
<td>0</td>
<td>4,898</td>
<td></td>
</tr>
<tr>
<td>Total Laptops</td>
<td>1,885</td>
<td>6,226</td>
<td>0</td>
<td>4,328</td>
<td></td>
</tr>
</tbody>
</table>
Exercise 10.4: Optimize Rules – Instead of using Getattr-function use subsets

In this continuation of the prior exercise, you will use subsets to replace the Getattr-function. This is suggested for bigger data models referring to performance. Simple rules using subsets are always faster than rules with functions.

1. Created different subsets in the Product dimension for the attribute Ratetype e.g. one subset for all product elements with attributes Stdcst01, one subset for all elements with attribute Stdcst02 etc. By using a data modeling tool this can be automated.

2. Change the rule and created different rules instead like:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[PRODUCT:{{Products.Stdcst01}}, MEASURES:‘Variable Costs’]</td>
</tr>
<tr>
<td>Rule</td>
<td>[MEASURES:‘Units’]*DB(‘RATETAB’,!YEARS,’Stdcst01’)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[PRODUCT:{{Products.Stdcst02}}, MEASURES:‘Variable Costs’]</td>
</tr>
<tr>
<td>Rule</td>
<td>[MEASURES:‘Units’]*DB(‘RATETAB’,!YEARS,’Stdcst02’)</td>
</tr>
</tbody>
</table>

3. Check the results. They should be the same but the execution is faster.

Testing for Invalid Cost Types Using the DE.INDEX Function

In order to help ensure that BI OLAP knows how to deal with invalid cost types, it may be beneficial to test that the cost type exists before attempting to do the calculation. The easiest way to do this is to use the DE.INDEX function. This function returns the index number within a specified dimension for a specified element. If an element does not exist, a zero is returned. Therefore, once the rate type for a particular product has been determined by looking it up in the attribute table, its existence in the stdcst dimension can be established by determining its index value. If the index value is zero, then the calculation can be bypassed.

The syntax for the DE.INDEX function is:

DE.INDEX(Dimension,Element)

In this case, the element to be tested is the Standard Cost Type in the stdcst dimension. Also, the GET.ATTR function has to be used within the DE.INDEX function. And, all of this has to be used within the IF function so that only IF the INDEX of the ATTRIBUTE is not equal to zero, then do the calculation. Otherwise, return "#NA".
Exercise 10.5: Testing for Invalid Cost Types

In this continuation of the prior exercise, you will use the DEINDEX function to test for an invalid cost type.

Exercise Steps

1. Edit the ['Variable Costs'] rule as follows:

   B: IF(DE.INDEX('stdcst', GETATTR('product', !product, '1', 'ratetype')) = 0, #NA, ['Units'] * B('RATETAB', !years, GETATTR('Product', !product, '1', 'ratetype')) / 1000)

2. Verify that the Edit Cube Rule dialog box looks like the example shown below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>[MEASURES: 'Variable Costs']</td>
</tr>
<tr>
<td>Rule</td>
<td>IF( DE.INDEX('stdcst', GETATTR('product', !product, '1', 'ratetype')) = 0, #NA, [MEASURES: 'Units'] * DB('RATETAB', !YEARS, GETATTR('Product', !product, '1', 'Ratetype')) / 1000 )</td>
</tr>
</tbody>
</table>

Other Uses for this Method

You can use this method for currency translation. If many locations use the same currency, use an attribute table to lookup the proper currency code to apply from your exchange rate table. Use currency code (rather than location) as one of the dimensions in the exchange rate table.
Lesson 11: Calculation Rules

Estimated Time

- 30 minutes

Learning Objectives

After completing this lesson, you will be able to:

- Use attribute tables and cube rules to determine calculation rules.

Topics

- Using Attributes to Determine Calculation Rules
Using Attributes to Determine Calculation Rules

In this topic, a specific business scenario will be used to discuss rule-related information.

Scenario

“I need to store balance sheet data along with my income statement data. How can I do this and get correct quarterly values for all of my data? How can I show favorable/unfavorable Budget to Actual variances easily?”

BI OLAP consolidates all data in the cube, if a dimension specifies it to do so. First, you will create a Variance rule. Then, set up an attribute table for the Measures dimension and define a single attribute, Measure Type, for each of the elements in this dimension. There will be four measure types: Asset, Liability, Revenue and Cost. Then, set up rules for the months and the actvsbud dimensions that will use the measure type to determine how a particular measure should be treated when calculating quarterly and yearly values, and how it should be treated when calculating Variance.

Considerations:

- Use the IF function to test the Measure Types (i.e. Revenue or Asset). The results (true or false) will determine the calculation for Variance.
  The syntax is:
  
  \[
  \text{IF(\text{TestExpression}, \text{TrueValue}, \text{FalseValue})}
  \]
  
  The calculation is:
  
  \[
  \text{If (Measure Attribute = Revenue OR Measure Attribute = Asset, then (Variance = Actual - Budget), Else (Variance = Budget - Actual))}
  \]

- Use the GETATTR function to retrieve the measure attribute.
  The syntax is:
  
  \[
  \text{GETATTR(DimName, ElemName, ATabID, FieldName)}
  \]
Exercise 11.1: Creating the ['Variance'] Rule

In this exercise, you will create the ['Variance'] rule using the Busnrule data model.

Exercise Steps

1. Click the Cubes button on the BI File and Database toolbar (or, select Cubes and Data Areas from the BI Office Plus menu). The Cubes and Data Areas dialog box opens.
2. Select the TOTSALES cube.
3. Click the Edit Rules button.
4. Click the New Rule button. The Edit Cube Rules dialog box opens.
5. Click once in the Target section of the dialog box to place the cursor there.
6. Click the Internal Reference button.
7. Click the Select One Element button. This rule will calculate variances, so the Target is Variance (an element in the actvsbud dimension).
8. Select Variance.
9. Click OK.
10. Click once in the Rule section of the dialog box to place the cursor there.
11. Select the IF() function in the Rule section to begin creating the rule.
12. Select the GETATTR() function.
13. Click the green checkmark button to add the function to the rule.
14. Edit the GETATTR function within the Rule section to read as follows:
   GETATTR('Measures',!measures,1,'Type')
15. Append the existing rule as shown in the screenshot below:

   ```
   Rule:
   IF(GETATTR('Measures',!measures,1,'Type')='Revenue'
   ```
16. Click the Internal Reference button to begin entering the calculation for the TRUE statement.
17. Click the Select One Element button next to the actvsbud dimension.
18. Select Actual.
19. Click OK.
20. Click OK again to accept the selection.
21. Click the Minus operator button, or type a minus sign ( - ).
22. Click the Internal Reference button.
23. Click the Select One Element button next to the actvsbud dimension.
24. Select Budget.
25. Click OK.
26. Click OK to accept the selection.
27. Click the Comma operator button, or type a comma (, ) to signify the end of the true statement.
28. Click the Internal Reference button to begin entering the calculation for the FALSE statement.
29. Click the Select One Element button next to the actvsbud dimension.
30. Select Budget.
31. Click OK.
32. Click OK to accept the selection.
33. Click the Minus operator button, or type a minus sign (-).
34. Click the Internal Reference button.
35. Click the Select One Element button next to the actvsbud dimension.
36. Select Actual.
37. Click OK.
38. Click OK to accept the selection.
39. Type ) at the end of the rule to complete the IF statement, as shown in the screen below:

![Edit Cube Rule window]

Due to performance reasons do not use OR-function like this:

![Edit Cube Rule window with OR-function]

40. Click the Check Syntax button.

If you receive a message that reads "Rules Compiler – Syntax error," then check the location where the cursor is pointed and make any necessary corrections.

41. Click OK after your rule is correct.
42. Browse the TOTSALES Cube to see that the rule is working properly.
43. Verify the results by referring to the following screen for BUSNRULE data model, cube TOTSALES and year 2013:

<table>
<thead>
<tr>
<th>MONTHS</th>
<th>Year</th>
<th>Region</th>
<th>World</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT</td>
<td>Total</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MEASURES</th>
<th>Actual</th>
<th>Budget</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Units</td>
<td>810,938</td>
<td>812,200</td>
<td>-1,262</td>
</tr>
<tr>
<td>Gross Margin</td>
<td>818,364</td>
<td>821,728</td>
<td>-3,364</td>
</tr>
<tr>
<td>Sales</td>
<td>1,553,180</td>
<td>1,557,540</td>
<td>-4,360</td>
</tr>
<tr>
<td>Variable Costs</td>
<td>734,816</td>
<td>735,812</td>
<td>996</td>
</tr>
<tr>
<td>Receivables</td>
<td>768,530</td>
<td>629,460</td>
<td>139,070</td>
</tr>
<tr>
<td>Payables</td>
<td>58,748</td>
<td>48,221</td>
<td>-10,526</td>
</tr>
</tbody>
</table>

Creating the ['1st Quarter'] Rule

As a continuation of the scenario, you will now create a rule to update to the Quarter calculations. First, you must test the Measure attribute and do a calculation based on the measure type. The STET function is used for this purpose; it states that nothing should be done and to leave the data as is, or skip it.

Using the OR-function in syntax is only suggested for small data areas. If you have bigger cubes, it is not very efficient and you should redesign the data model.

Exercise 11.2: Creating the ['1st Quarter'] Rule

In this exercise, you will create the ['1st Quarter'] rule.

Exercise Steps

1. Create the ['1st Quarter'] rule using the data shown below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>['1st Quarter']</td>
</tr>
<tr>
<td>Rule</td>
<td>IF(GETATTR('MEASURES',!measures, 1,'Type')@='Asset' OR GETATTR('MEASURES',!measures, 1,'Type')@='Liability','March',STET)</td>
</tr>
</tbody>
</table>

2. Click the Check Syntax button to verify the syntax.
3. Create four rules using the data indicated for each rule. You will create a rule for each of other three quarters; and for the Year, to set the value to December.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>['2nd Quarter']</td>
</tr>
<tr>
<td>Rule</td>
<td>IF(GETATTR('Measures',!Measures,1,'Type')@ = 'Asset' OR GETATTR('Measures',!Measures,1,'Type')@ = 'Liability', ['June'],STET)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>['3rd Quarter']</td>
</tr>
<tr>
<td>Rule</td>
<td>IF(GETATTR('Measures',!Measures,1,'Type')@ = 'Asset' OR GETATTR('Measures',!Measures,1,'Type')@ = 'Liability', ['September'],STET)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>['4th Quarter']</td>
</tr>
<tr>
<td>Rule</td>
<td>IF(GETATTR('Measures',!Measures,1,'Type')@ = 'Asset' OR GETATTR('Measures',!Measures,1,'Type')@ = 'Liability', ['December'],STET)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>['Year']</td>
</tr>
<tr>
<td>Rule</td>
<td>IF(GETATTR('Measures',!Measures,1,'Type')@ = 'Asset' OR GETATTR('Measures',!Measures,1,'Type')@ = 'Liability', ['December'],STET)</td>
</tr>
</tbody>
</table>

4. Verify the results by referring to the screen shot that follows:

5. Verify the rule by browsing the TOTSALES cube.
6. Verify the results by referring to the screen shot for data model BUSNRULE, cube TOTSALES and year 2013 that follows:

Other Uses for this Method

The attributes of this cube could also be employed to generate cash-flow analysis statements. The typical problem with cash-flow analysis is the presentation of income statement items at their current period balance, while balance sheet items are presented as they change over a period. The methodology for the attribute table and rules created to calculate variance could be employed to generate cash-flow analysis statements as well.

Other Considerations

Revenue and Liability measures naturally have credit balances within most General Ledger systems. If you choose not to show the data in this manner within the BI OLAP cubes, you will need to reverse the signs on these measures. You can use the measure attribute table to determine whether a value must be reversed. Then, create an import sheet that contains this type of “if, then” logic to determine the value to be sent to your cube.

Sample “if, then” statement shown below:

If (Measure Attribute = Revenue OR Measure Attribute = Liability, then Value * -1, else Value)
Lesson 12: Differentiating between the Old and New Rules Engine

Estimated Time

- 1 hour

Learning Objectives

After completing this lesson, you will be able to:

- Identify the differences between the old rules engine and the new rules engine.
- Optimize cube rule performance.
- Optimize cube rules.

Topics

- Understanding the New Rules Engine
- Optimizing Cube Rule Performance
- Optimizing Cube Rules
Understanding the New Rules Engine

This training workbook is primarily written for the new rules engines, which works without accelerators. The justification for the changes to the OLAP Rules Engine several years ago was that rules accelerators are difficult to set and for that reason they are sometimes set incorrectly. The goal was to eliminate the need for accelerators. This is important, because some people think that the goal was to make the OLAP Server faster.

The great advantage of the new rules engine is ease of use. The new rules engine was simply added to the server in addition to the existing engine. When a cube is loaded the server checks to see if the accelerators are turned on. If they are, which is normally the case in older cubes, then the older engine is loaded. However, if the server detects that there are no accelerators in the cube then the new Rules Engine is loaded and applied. The advantage is that in most cases the new cube will perform as fast as or faster than it would have on the old system with accelerators. This means that creating new cubes will require much less technical skill on the part of the consultant.

To understand how the new, improved Rules Engine works, it is important to have a basic understanding of what the accelerators do, and why they made the rules faster in the past. The key issue is that the old Rules Engine lost a lot of time checking if cells were empty. Since in general cubes tend to be fairly empty, this can take a lot of time in real situations. What the accelerators do is check to see if the target cells of an accelerated rule need to be calculated.

When the cube is loaded, the server looks at the accelerators and marks the target cells of the accelerated rules that have non-empty source cells. Whenever a value is written back to the cube, the server checks to see if the cell is the source of an accelerated rule. If it is, then the corresponding target cells are marked. When the rules are calculated only the target cells that are marked are calculated. This saves a lot of time checking to see if the target cells are empty.

The performance penalty in the old rules engine comes from the fact that the fetches are made individually during the calculation process. In other words, the calculation is started, the Rules Engine interprets the rule, sees that data from a source cell is required, fetches the data, and restarts the calculation where it left off. When it discovers a second source cell is required, it restops the process, and does another fetch operation.

The new Rules Engine has a completely different method of dealing with this situation. Instead of fetching each data cell individually as the need arises, the new Rules Engine calculates the entire range of cells that the rule requires to execute. This data is block fetched in as few operations as possible. The question of whether a given cell is empty is simply ignored. The fetch requests all the cells, whether empty or not. Only the non-empty cells return data, and only this data is processed.

In a sense you can compare the fetching behavior of the new Rules Engine with the behavior of the Alea Excel formula DBGetC. DBGet get individual data cells from the server. Unlike DBGet, DBGetC collects all the database accesses and sends them to the server in as few individual accesses as possible. Since the turnaround time for individual accesses is high, this is a time saver, even though, from the user point of view, the same amount of data is transferred.
Optimizing Cube Rule Performance with Accelerators

In some rare cases using cube rules can reduce performance in a cube. Calling high level consolidations may render it unacceptably slow. However, you can optimize the performance of your rules by implementing accelerators. When correctly implemented, accelerators usually restore the cube’s performance to approximately the level prior to the cube rules’ implementation.

BI OLAP can perform on-the-fly consolidations of vast, multidimensional matrices rapidly because it can ignore empty cells and only focus on non-zero cells when calculating. However, the BI OLAP calculation algorithm relies on cells being physically occupied with data before it uses them in a calculation.

Essentially, an accelerator says: place a flag in the target area every time the corresponding cell in the source area is non-zero.

If a rule has a reference to an external cube, you must specify the name of the external cube in front of the source area and the name of the main cube in front of the target area. To write the associated accelerator, write the area in the external cube as if writing the area of the rules cube, but prefix the area with the external cube name.

The Edit Accelerator dialog box contains two buttons for displaying dialog boxes that facilitate the insertion of write areas for both the main cube and external cubes. These buttons (Internal Reference and External Reference) are the same buttons found in the Edit Cube Rule dialog box.

Exercise 12.1: Create an Accelerator for the Rule

In this exercise, you will create an accelerator to optimize the performance of the cube rule you created in the prior exercises. This exercise uses the Price data model.

Exercise Steps

1. Click the Cubes button in the BI File and Database toolbar (or select Database>Cubes and Data Areas on the BI Office Plus menu) to open the Cubes and Data Areas dialog box.
2. Select the cube for which you want to create an accelerator.
3. Click the Edit Rule button. The Cube Rules dialog box displays.
4. Click the Accelerator button. The Edit Accelerator dialog box opens.
5. Enter the information noted below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>Type a target location in the cube.</td>
<td>TotSales[REGIONS:'Germany',MEASURES:'Units']</td>
</tr>
</tbody>
</table>
6. Click **OK** when you finish writing your accelerator. The Cube Rules dialog box opens.
7. Select the **Use accelerators** check box on the toolbar to apply the accelerator.
8. Click the **Save** button to save the cube.

---

**Infor BI Office Plus Online Help**
Data Access>Cubes and Data Areas>Cube Rules>Rules Functions
Optimizing the Cube Rules

In this continuation of the prior topic’s scenario about old rules engine with accelerators, you are writing an accelerator for the ['Home Office'] rule in the Currency data model. In this scenario, if Local Currency is non-zero for a given product, country, measure and time period, then Home Office will be non-zero. Therefore, if there is a value in a Local Currency cell, place a flag in the corresponding Home Office cell. In BI OLAP terms ['Local Currency'] is the source area and ['Home Office'] is the target area.

Exercise 12.2: Create Accelerators for the Currency Rules

In this exercise, you will optimize the currency rules you created in prior exercises in the Currency data model.

Exercise Steps

1. Open the Cube Rules dialog box for the TOTSALES Cube.
2. Click the New Accelerator button. The Edit Accelerator dialog box opens.
3. Click once in the Source section to place the cursor there.
4. Click the Internal Reference button. The Cell Reference dialog box opens.
5. Select the Local element from the Currency dimension.
6. Click OK. The Edit Accelerator dialog box opens.
7. Click once in the Target section to place the cursor there.
8. Click the Internal Reference button.
9. Select the Home Office element from the Currency dimension.
10. Click OK.
11. Click OK to accept the new accelerator (there might be a delay as the flags are set in the cube).
12. Select the Use accelerators checkbox on the toolbar to apply the accelerator.
Lesson 13: Importing Dimensions and Automation

Estimated Time
- 1 hour

Learning Objectives
After completing this lesson, you will be able to:
- Use macros to import dimensions and data.

Topics
- Using macros to import dimensions and data.
Using Macros to Import Dimensions and Data

In this topic, a specific business scenario will be used to discuss rule-related information.

**Scenario**

*My company is constantly adding new products. This is great for business, but it means that I must manually update my product dimension each month prior to running my data import process. Is there any way to automate this? It would be nice if I could automate the entire update (dimensions, attributes, and data) in one process!*

Because business is always changing, BI OLAP dimensions must be updated regularly to reflect the changes. Sometimes you will need to create an entirely new dimension, other times you will need to just add a few new items to an existing one. BI OLAP functions help automate this process.

**Considerations:**

- You need a data file to work with. This file should include all of the information you will need to build your dimension.
- You must create a processing worksheet that will be used to import the file and create the dimension. This worksheet will include BI OLAP functions designed specifically for this purpose. These functions will allow you to either recreate or append to the dimension when the file is processed, depending on your needs.
- You can then use Excel macros to completely customize and automate the process. There are BI OLAP macro functions that you can use along with Excel macro functions to accomplish any task you may need to perform.

**Demo: Processing the IMPOVR.XLS Worksheet and Viewing Formulas**

Your instructor will demonstrate how to process the IMPOVR.XLS worksheet while viewing important formulas.

---

The parameter for Clear is only read once, when the first DE.ADD formula is evaluated. Be consistent throughout your formulas to ensure you get the expected results.
Lesson 13: Importing Dimensions and Automation

Exercise 13.1: Importing the PROD Dimension and Overwriting the Existing Dimension

In this exercise, you will import and overwrite dimensions.

Exercise Steps

1. Process the PROD.TXT file using the IMPOVR.XLS workbook.
2. Check that all records have processed correctly. Just as in importing data, any records that did not process would be directed to the log file (ERROR.LOG).
3. View the dimension after you process the file. All of the elements, with proper hierarchies, should be present.
4. Verify the results by referring to the screen shot that follows:

![Screen Shot of Dimension]

5. Open the IMPAPP.XLS workbook.
6. Process the first record of the PROD2.TXT data file. Only elements that do not already exist in the dimension will be added.
7. Close the IMPOVR.XLS workbook.

Whenever you import and process another text file, ensure that the current processing sheet is the only one open; all open sheets are recalculated with each record being read.
Exercise 13.2: Importing the PROD Dimension and Appending the Existing Dimension

In this exercise, you will import and append the dimension.

Exercise Steps

1. Process the PROD2.TXT file using the IMAPP.XLS workbook.
2. Verify that all records have processed correctly. Just as in importing data, any records that did not process would be directed to the log file (ERROR.LOG).
3. View the dimension after you process the file. You should see that new elements have been added, along with their hierarchies.
4. Verify the results by referring to the screen shot that follows:

5. Save the dimension to make sure all of the changes are made permanent.

You can also use the append method to create a new dimension. If a dimension already exists with the same name you are specifying, the existing dimension will be appended rather than a new one created.

Automating the Task with Macros

To automate the task of creating a dimension with macros, you must do the following within the code:

- Set the path and name of the processing workbook.
- Set the name of the sheet within the workbook that will be used for processing.
- Set the name of the data file.
- Set the name of the sheet within the workbook that will be used for file processing.
- Set the error file name and path.
- Ensure that the data file is not empty by testing the length of the file.
  - If file has no data (length is zero), do nothing. Otherwise, process the file.
**Lesson 13: Importing Dimensions and Automation**

- Specify the file to be processed, the name of the error log, the type of delimiter used and how to process the file.
- If an error occurs during process, do something (i.e. display a message box alerting the user to the error).
- If an error log is created, a message box will be displayed.
- When done, close the processing sheet.

The syntax of the macro function is as follows:

\[
\text{TablesImportFromFile} (sFileImport, vntDelimiter, vntLogFile, vntStep)
\]

- \(sFileImport\) = the name of the data file
- \(vntDelimiter\) = the character used as a field delimiter within the data file
- \(vntLogFile\) = the name of the error file
- \(vntStep\) = whether the macro should pause after each record is processed or continue processing the file until the end of file is encountered (TRUE = pause; FALSE = continue)

Using the previous dimension overwrite example:

- \(sFileImport\) = PROD.TXT
- \(vntDelimiter\) = Tab (we will see how to set this in just a moment)
- \(vntLogFile\) = ERRTXT.TXT
- \(vntStep\) = FALSE (we want the entire file to be processed without interruption)

**Note:** The \(vntDelimiter\) is specified by either enclosing the character in quotes (";" or ","), or by using the Visual Basic CHR function with the ANSI code for the character (numbers from 0 to 255). Our file is tab-delimited.

Before using any BI OLAP VBA functions, you must first make a ‘Reference’ to the BI OLAP VBA functions file. This must be done in each file you would like to use in these functions. The reference is saved with the file.

**Exercise 13.3: Writing the Import PROD Dimension Macro and Clearing the Existing Dimension**

In this exercise, you will write a macro and clear a dimension.

**Exercise Steps**

1. Open the VBAIMP.XLS workbook.
2. Insert a new Visual Basic module.

In Microsoft Excel 2007, 2010 and 2013 Ribbon, it is hard to find out the Visual Basic Editor command. That’s because it is moved into Developer tab, which is not in Ribbon:

- Firstly of all you need to add Developer Tab into Microsoft Excel 2007/2010/2013 Ribbon.
- Click the **Developer** tab;
- Go to the **Code** group;
3. Then you will view the **Visual Basic** button. Click it, and you will enter the Visual Basic Editor window.

4. Select **References** from the Tools menu. You will find a list box containing all the available libraries. You will see two entries for 'Alea — Multi Dimensional Server'.

5. Select the file that displays MDSEX32.DLL in the **Location** path in the lower part of the dialog box.

6. Select the checkbox for the file.

7. Click **OK**.

8. Set the variables.

9. Check the length of the PROD.TXT file. If the length of the file is zero, proceed to Step 9. If the length of the file is not zero:
   a. Open the IMPOVR.XLS workbook.
   b. Activate the Overwrite (processing) sheet.
   c. Process the file.

10. Set error messages: if an error occurs, display the message; if no error occurs, closet the spreadsheet without saving it.

11. Run the IMPORT1 macro to create a new prod dimension.

12. Examine the dimension to check that it was created properly.

   The macro is provided in its entirety on the following page.
Macro for Exercise 13.3

Sub IMPORT1()
    sPath = ThisWorkbook.Path & "\"
    sFile = "IMPOVR.XLS"
    sSheet = "Overwrite"
    sData = "PROD.TXT"
    sLog = "ERRTXT.TXT"

    lLength = FileLen(sPath & sData)
    If lLength <> 0 Then
        Workbooks.Open sPath & sFile Sheets(sSheet).Activate
        vntX = TablesImportFromFile(sPath & sData, Chr(9), sPath & sLog, False)
        If IsError(vntX) Then
            MsgBox MdsError(MdsGetLastError)
        End If
        ActiveWorkbook.Close savechanges:=False
    End If

    If FileLen(sLog) > 0 Then
        MsgBox "An error log was created, please check the file."
    End If

End Sub
Exercise 13.4: Writing the Import PROD Dimension Macro and Appending the Existing Dimension

In this exercise, you will write a macro and append a dimension.

Exercise Steps

1. Refer to the steps in the prior exercise, with the following exceptions:
   a. Name the macro IMPORT2
   b. Set sFile = "IMPAPP.XLS"
   c. Set sSheet = "Append"
   d. Set sData = "PROD2.TXT"

2. Set the parameter for Clear to FALSE in the TablesImportFromFile function.

3. Verify your macro against the Macro for Exercise 7.4 provided on the following page.

Other Uses for this Method

Use DB.SET and AT.SET functions to update data in cubes and attribute tables. Add these functions to your process sheets and use macros in a similar manner described in this lesson and you can automate the entire update process.
Macro for Exercise 13.4

Sub IMPORT2()

    sPath = ThisWorkbook.Path & "\" sFile = "IMPAPP.XLS"
    sSheet = "Append"
    sData = "PROD2.TXT"
    sLog = "ERRTXT.TXT"
    lLength = FileLen(sPath & sData)
    If lLength <> 0
        Then
            Workbooks.Open sPath & sFile
            Sheets(sSheet).Activate
            vntX = TablesImportFromFile(sPath & sData, Chr(9), sPath & sLog, False)
            If IsError(vntX) Then
                MsgBox MdsError(MdsGetLastError)
            End If
            ActiveWorkbook.Closesavechanges:=False
        End If
    If FileLen(sLog) > 0 Then
        MsgBox "An error log was created, please check the file."
    End If

End Sub
Lesson 14: Converting Rules into Physical Values (Advanced Lesson)

Estimated Time
- 1 hour

Learning Objectives
After completing this lesson, you will be able to:
- Use macros to calculate rules and store the result as physical values into the cube using a batch process.

Topics
- Using macros to convert rules into physical values.
Exercise 14.1: Writing Macro to convert rules into physical values using exported rule from OLAP cube in XML-format and Appending the Existing Dimension

In this exercise, you will write a macro to convert a rule into physical values.

Sub Write_CalcValues()
' Purpose: Write rules calculated values as fixed base values to cube - using DataAreaCalculate
Dim sServer As String
Dim sUser As String
Dim sPwd As String
Dim sCube As String
Dim mdsret As Variant
Dim asDataArea(0 To 5, 0 To 6) As String
Dim sNextLine As String
Dim sXmlRule As String
sServer = "local/tutor"
sUser = "admin"
sPwd = ""
sCube = "TGUV"
' Call an external file:
Open "D:\AleaRoot\VBA\tguv.rules.xml" For Input As #1 ' Open File with include XML-Statement
' Rules-Export file can be used, but time stamp has to be removed
Do While Not EOF(1) ' check to fileend
Line Input #1, sNextLine ' read next line
sXmlRule = sXmlRule & sNextLine
Loop
Close #1 ' Close File with include XML-Statement

Debug.Print sXmlRule
' Define DataArea (column 2 defines the dimension - order of dimensions as in cube, 0 based)
' dimension Jahre
asDataArea(0, 0) = "*"
' dimension Datenart
asDataArea(0, 1) = "*"
' dimension Regionen
asDataArea(0, 2) = "*"
' dimension Produkte
asDataArea(0, 3) = "*"
' dimension Monate
asDataArea(0, 4) = "*"
' dimension Wertart
asDataArea(0, 5) = "*"

' connection to DB
mdsret = ServerConnect(sServer, sUser, sPwd)
' create DataArea aufbauen
mdsret = DataareaDefine(sServer, sCube, "", asDataArea, 0, 0, 0, False, True)
If IsError(mdsret) Then Debug.Print (MdsError(MdsGetLastError))
Exercise 14.2: Writing Macro to convert rules into physical values using fixed setup rule used in script

In this exercise, you will write a macro to convert a rule into physical values.

Sub Write_CalcValues()
'Purpose: Write rules calculated values as fixed base values to cube - using DataAreaCalculate
Dim sServer As String
Dim sUser As String
Dim sPwd As String
Dim sCube As String
Dim mdsret As Variant
Dim asDataArea(0 To 5, 0 To 6) As String
Dim sNextLine As String
Dim sXmlRule As String
sServer = "local/tutor"
sUser = "admin"
sPwd = ""
sCube = "TGUV"
sXmlRule = sXmlRule & "<?xml version=""1.0"" encoding=""utf-8"">""""<Alea:Rules xmlns:Alea="http://www.misag.com"" Accelerated="true" User="mkueppers"
Time="Jul/04/2011 12:05:13,858""""<Alea:RuleGroup Enabled="true""""<Alea:Rule Enabled="true" Cells="Basic" Type=Cube"
RuleID="0X00082000001"">
  <Alea:Target>[DWAEHRUNG:'KW1_REWE']</Alea:Target>
  <Alea:Formula><![DWAEHUNG:'LW']]/DB('TWAEHR','!DZEIT,'!DZWEIT,!'HW',!'EURD')*DB('TCOMP_CURR','!DZWEIT,'!DGESELLSCHAFT,!
'FAKTOR')</Alea:Formula>
sXmlRule = sXmlRule & """"<Alea:Rule>
  <Alea:Source>TGUV[DWAEHUNG:'LW']</Alea:Source>
  <Alea:Destination>TGUV[DWAEHUNG:'KW1_REWE']</Alea:Destination>
</Alea:Rule>
Debug.Print sXmlRule
'Define DataArea (column 2 defines the dimension - order of dimensions as in cube, 0 based)
'dimension Jahre
asDataArea(0, 0) = "*"
End Sub
'dimension Datenart
asDataArea(0, 1) = "*"
'dimension Regionen
asDataArea(0, 2) = "*"
'dimension Produkte
asDataArea(0, 3) = "*"
'dimension Monate
asDataArea(0, 4) = "*"
'dimension Wertart
asDataArea(0, 5) = "*"
'connection to DB
mdsret = ServerConnect(sServer, sUser, sPwd)
' create DataArea aufbauen
mdsret = DataareaDefine(sServer, sCube, "", asDataArea, 0, 0, 0, False, True)
If IsError(mdsret) Then Debug.Print (MdsError(MdsGetLastError))
mdsret = DataareaCalculate(sXmlRule, 1)
If IsError(mdsret) Then Debug.Print (MdsError(MdsGetLastError))
'delete the dataarea
mdsret = DataareaDestroy
If IsError(mdsret) Then Debug.Print (MdsError(MdsGetLastError))
mdsret = ServerDisconnect(sServer)
End Sub
Course Summary

Estimated Time

- 30 minutes

Course Objectives

Now that you have completed this course, you should be able to:

- Explain cube rules and syntax.
- Create and edit cube rules.
- Create and edit accelerators to optimize performance.
- Look up data via attribute tables.
- Use attributes to determine calculation rules.
- Create macros to import data.

Topics

- Course Review
Course Review

What do cube rules enable you to do?

_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________