MICROSOFT
TRAINING
AND CERTIFICATION

Microsoft Official Curriculum

Module 3: Using MDX Expressions

Contents

Microsoft

Instructor Notes

Presentation:
100 Minutes

Labs:
50 Minutes

In this module, students will learn how to create very simple multidimensional expressions (MDX). These are expressions that use constant values or expressions that display the names of members. Students will also learn how to work with members in a dimension and how to use MDX member functions to navigate from one member to another.

After completing this module, students will be able to:

- Create MDX expressions in calculated members and manipulate the expressions by using numeric and string constants.
- Display information about a member—particularly the name of the member and the name of the level of the member.
- Display information about the parent or other ancestor of a member.
- Display the member property associated with a member, and use the value in an arithmetic expression.
- Create and use simple and complex conditional expressions.

Materials and Preparation

This section provides you with the required materials and preparation tasks that are needed to teach this module.

Required Materials

To teach this module, you need the following materials:

- Microsoft® PowerPoint® file 2093A_03.ppt

Preparation Tasks

To prepare for this module, you should:

- Read all the materials for this module.
- Read the instructor notes and margin notes.
- Practice the lecture and group activities.
- Complete the labs.
- Review the Trainer Preparation materials on the Trainer Preparation compact disc.

Instructor Setup for Group Activities and Labs

This section provides setup instructions that are required to prepare the instructor computer or classroom configuration for group activities and labs.

All group activities and labs use the same database setup, which requires restoring a database archive.

► To prepare for group activities and labs

In this procedure, you restore the **Market** database, which is a .cab file type.

1. Start Analysis Manager.
2. In the left pane, expand the Analysis Services folder.
3. Expand the **Server** icon and verify that the **Market** database does not exist.
4. Right-click the **Server** icon, and then click **Restore Database**.
5. Navigate to the **C:\Moc\2093A\Batches** folder.
6. Select **Market.cab**, click **Open**, and then click **Restore**.

If the **Market** database already exists from a previous group exercise or lab, and cubes in the database contain extraneous information, you can return the **Market** database and its cubes to a beginning position by either:

- Deleting any calculated members that were created in a specific cube and then saving the cube.

  - or -

- Repeating the previous restore database procedure.

Other Activities

Difficult Questions

Below are difficult questions that students may ask you during the delivery of this module and answers to the questions. These materials delve into subjects that are within the scope of the module but are not specifically addressed in the content of the student notes.

1. Are calculated members stored in the cube or aggregated?

   The values of a calculated member are not stored or aggregated. The definition of the calculated member is executed each time a client application establishes a connection to the cube that requests the calculated member's values.

2. Do calculated members execute for all of the members of the cube?

   No. A calculated member is executed only when the value is actually displayed on the browser grid. Unless a cell is displayed in the browser, its formula is never calculated.

3. Is the **Empty** value (Null) a string or a number?

   The Empty value is treated as a number—typically as the number zero.

4. What is the difference between the four **Name** functions—dimension, hierarchy, level, and name?

   There is only one Name function. It will return the name of whatever object it is used with. The four entries in the Functions list simply put different tokens in front of the function name.

5. How do I find the available Microsoft Excel or Microsoft Visual Basic® for Applications (VBA) functions?

   Run Books Online for Microsoft Analysis Services, and click the Index page. Type function libraries and choose Excel or VBA. All functions that do *not* have an asterisk are available for use from MDX.

6. To use an Excel function from MDX, does Excel have to be installed on the client or on the server?

   Under default conditions, an external library (including Excel or VBA) must be available *both* on the client on the server. That is because the expression may be evaluated either by PivotTable Services (on the client) or the Analysis server. It is possible to create a connection in a way that forces expressions to be evaluated exclusively on the client or on the server as opposed to the default, which is that the expression could be evaluated in either place, and an external library must be available on both the client and the server.

7. Why does the **CurrentMember** function follow the dimension name in COM object format, whereas the **Ancestor** function uses arguments?

   Some MDX functions use property-style syntax, whereas others use argument-list syntax. Only functions that require no arguments can use the property-style syntax. The choice between the two styles often appears arbitrary, and you simply look at the syntax example to see which to use.

8. Is there a way for an MDX expression to find out what member properties are available for a given level?

   No. When using MDX, you must know the name of the member property and the level at which it applies. You cannot refer to a property by number, and you cannot request the count of the properties. You can find this information out programmatically by using Microsoft ActiveX® Data Objects (Multidimensional) (ADO MD), but not by using an MDX expression.

9. When would you use a string value from an MDX expression in a practical situation?

   One example is to show the name of the best-selling product for any given state.

10. Can an MDX expression return a date?

    Typically, when you want to display a date, you are displaying the name of a member on a Time dimension, which is a string. If you want to display a true date or time value, it is really a formatted number.

Module Strategy

Each major section of this module begins with a group activity followed by a review lecture and then labs. The following guidelines are for delivering materials in the context of group activities:

- Using group activities to introduce new content

  You often introduce new concepts or functionality while delivering the procedures in a group activity as a live demonstration. For example, you may present a new MDX function by showing first its construction and then its result set as an actual calculated member formula or within a query statement.

  Use the topic slides that follow the group activity as a review of the content, for example, the syntax of a specific function.

- Interaction with students

  A group activity flows best when you deliver it as a shared exploration. Ask students such questions as: "What would happen if we…?" "Why did this happen?" "Was that what you expected?" Encourage students to ask you questions about the results being tested.

- Students follow along

  In some cases, you may want to encourage students to follow your live demonstration on their own computers. This practice works best for simpler group activities or for a group activity that is not replicated by a later lab.

  It is not a problem if a student does not follow your demonstration, or if a student starts following and then stops before the group activity is completed. There is no file or structure dependency between group activities or between a group activity and a later lab.

- Lab replication of group activity

  The exercises in the labs closely follow the group activity procedures but do not define each step or show the code answer. Encourage students to write and test the MDX expressions on their own, referring back to the group activity procedures for clarification. Students can also refer to answer files that are available for each procedure in the exercises.

  Labs are generally more challenging when students have not followed the instructor on their own computers during the group activity. However, many students benefit from the two hands-on experiences of following the group activity and then completing the labs.

- Answer files for group activities

  Where applicable, answer files are provided for each procedure in a group activity. If necessary to facilitate your demonstration, copy and paste the correct expression from the answer file into the Calculated Member Builder.

Use the following additional strategies to present this module:

- Using MDX Expressions

  Introduce the Calculated Member Builder and the concept of an MDX expression as a formula that returns a string or numeric value. Present materials as simply as possible—similar to the classic "Hello, World" programming examples.

  Emphasize the fact that the MDX expression in the calculated member is calculated anew for each cell in which it displays—even if it is a constant value. Point out that entering a constant as the value of a calculated member is similar to entering a constant value in a spreadsheet cell.

  Compare with a spreadsheet formula, in which it is possible to manually change some of the formulas in a column, to MDX, in which all the copies of a formula are identical according to the MDX formula definition.

- Displaying Member Information

  Use the example of the **Name** function, which must explicitly use the **CurrentMember** function, as a method to return a member's name. Point out how the Level.Name part of the section shows the versatility of the **Name** function and is useful in advanced calculations.

  Reinforce the differences among a dimension, a level, and a member because students are often confused by these concepts and how they relate to each other. Minimize the importance of a hierarchy.

- Displaying Family Tree Relatives

  Focus on the **Parent** and **Ancestor** functions, which are extremely useful for advanced calculations. Point out the possibility of using **Ancestor** to determine security information.

- Working with Member Properties

  Point out how a member property can be easily viewed and manipulated without accessing the actual data values in a cube. Clearly explain that the #ERR values are normal and to be expected. Keep the focus on the level where the member property does exist.

  Show the students how to use Books Online to find the complete list of VBA and Excel functions.

- Using Conditional Expressions

  Keep the focus on constructing simple conditional expressions. The **IIF** function that follows is easy for students to understand after they understand basic conditional expressions.

Overview

Topic Objective
To provide an overview of the module topics and objectives.

Lead-in
In this module, you will learn how to create calculating members by using MDX expressions plus how to view metadata and work with conditionals.

- **Using MDX Expressions**
- **Displaying Member Information**
- **Displaying Family Tree Relatives**
- **Working with Member Properties**
- **Using Conditional Expressions**

In this module, you will learn how to create very simple multidimensional expressions (MDX). These are expressions that use constant values or expressions that display the names of members. You will also learn how to work with members in a dimension, and how to use MDX member functions to navigate from one member to another.

After completing this module, you will be able to:

- Create MDX expressions in calculated members and manipulate the expressions by using numeric and string constants.
- Display information about a member—particularly the name of the member and the name of the level of the member.
- Display information about the parent or other ancestor of a member.
- Display the member property associated with a member, and use the value in an arithmetic expression.
- Create and use simple and complex conditional expressions.

Using MDX Expressions

Topic Objective
To provide an overview of the topics in this section.

Lead-in
MDX expressions are formulas that have different types of values and use traditional arithmetic and string text operators.

- **Group Activity: Creating Expressions with Constants**
- **MDX Expressions**
- **Expression Operators**

In this section, you learn about the basic elements of MDX expressions, including how to use them to create calculated members and how to construct MDX expressions by using arithmetic operators and string text.

Delivery Tip
Briefly explain the bullets on this slide to provide context for the upcoming group activity in which the items are functionally demonstrated.

This group activity introduces concepts and mechanics for creating MDX expressions by using constants. Topics that follow the group activity include:

- MDX Expressions

  This topic describes how MDX expressions are used to create calculated members. Calculated members add the calculating ability of a spreadsheet to an online analytical processing (OLAP) database.

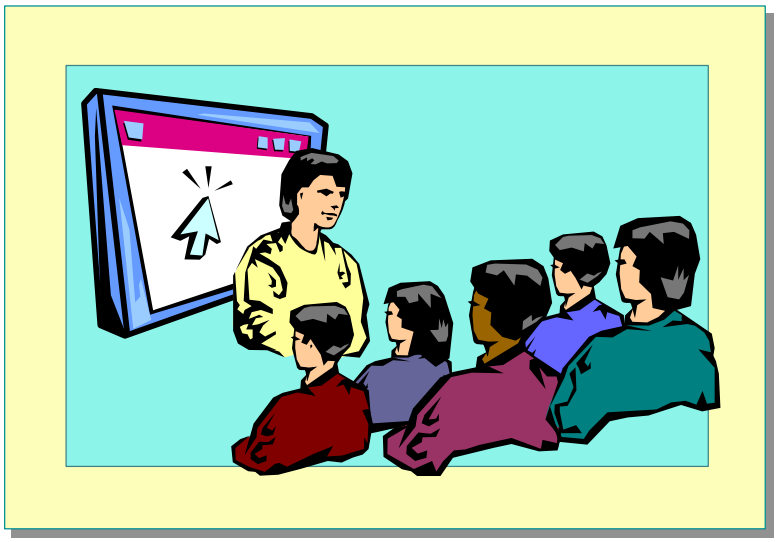
- Expression Operators

  This topic describes how expression operators are used in MDX expressions to return a value that can result from an arithmetic or string text manipulation.

Group Activity: Creating Expressions with Constants

Topic Objective
To learn expressions by using constants and formulas with constants.

Lead-in
In this group activity, you will learn how to create simple calculated members that return constants.



In this group activity, you may follow along on your own computer with your instructor or observe the demonstration.

You will learn how to use constants to create MDX expressions in a calculated member. Constant expressions display the same value in all the cells. You will also learn how to incorporate numeric and string text operators in expressions.

Calculated members are created in the **Basic Sales** virtual cube. This cube has only one measure—**Sales Dollars**—and three dimensions. Creating a calculated member in a virtual cube is the same as creating one in a regular cube.

The examples may seem easy, but they allow you to focus on the main concepts of this group activity, including:

- How to enter expressions in the Calculated Member Builder.
- How the formula is automatically applied to all necessary cells in the grid.
- How an expression always returns a single value—a string, a number, or an empty cell—each time it is calculated.

One important use of MDX expressions is to create calculated members. Calculated members add the calculating ability of a spreadsheet to an OLAP database.

Delivery Tip
ConstantNumberD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

► To create a calculated member with a constant numeric value

In this procedure, you will assign a numeric value constant value to a calculated member. This demonstrates the first of three data types that can be returned by an expression. The three data types are:

- Numeric value
- String value
- Empty value

1. In the **Market** database, expand the **Cubes** folder, right-click the **Basic Sales** virtual cube, and then click **Edit**.

   In the **Virtual Cube Editor**, the right pane always displays a preview.

2. Click the **Insert Calculated Member** button.
   - In the **Calculated Member Builder** dialog box, type **Constant** in the **Member name** box.
   - In the **Value expression** box, type **500**
   - To create the member, click **OK**.

	MeasuresLevel	
+ Category	Sales Dollars	Constant
All Product	$76,741.28	500
+ Bread	$30,600.42	500
+ Dairy	$32,533.02	500
+ Meat	$13,607.84	500

   In the preview pane—also called the browser—notice that the **Constant** column is filled with the value 500. If you think of the grid as a spreadsheet, you can think of a calculated member as a value or formula that automatically fills all the cells of the column.

3. To display the **Subcategory** level on the Rows axis, double-click the **Category** button.

		MeasuresLevel	
- Category	+ Subcategory	Sales Dollars	Constant
All Product	All Product Total	$76,741.28	500
- Bread	Bread Total	$30,600.42	500
	+ Bagels	$3,552.28	500
	+ Muffins	$13,081.28	500
	+ Sliced Bread	$13,966.86	500
- Dairy	Dairy Total	$32,533.02	500
	+ Cheese	$17,709.11	500
	+ Milk	$7,259.42	500
	+ Sour Cream	$2,746.58	500
	+ Yogurt	$4,817.91	500
- Meat	Meat Total	$13,607.84	500
	+ Deli Meats	$4,337.12	500
	+ Fresh Chicken	$1,143.56	500
	+ Frozen Chicken	$3,214.98	500
	+ Hamburger	$2,175.56	500
	+ Hot Dogs	$2,736.62	500

Notice that the value from the calculated member automatically expands to fill the needed cells. This differs from a spreadsheet, where a user must copy values into new rows and delete them from old rows.

4. To remove the **Subcategory** level, double-click the **Category** button.

► To create a calculated member with a string value

In this procedure, you will learn how you can change the expression for a calculated member simply by changing the **Value** property in the Properties pane. This procedure also demonstrates how an MDX expression can be a string as well as a number.

A string value is the second data type that an expression can return.

1. In the **Basic Sales** cube, in the Calculated Members folder, select **Constant**.
2. In the **Properties** box on the Basic tab, select the **Value** property, type **"Hello"** and then press ENTER.

Delivery Tip
ConstantTextD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

	MeasuresLevel	
+ Category	Sales Dollars	Constant
All Product	$76,741.28	Hello
+ Bread	$30,600.42	Hello
+ Dairy	$32,533.02	Hello
+ Meat	$13,607.84	Hello

3. In the browser, notice that the values in the **Constant** column change to the word Hello.

Note You can use a text string as a constant value in a calculated member if you enclose the string in quotation marks.

► To create a calculated member with an empty value

In this procedure, you will learn how to create an empty cell. Technically, an empty cell is a numeric value. It is treated as zero in most arithmetic expressions but is ignored in some contexts.

Delivery Tip
ConstantNullD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

An empty value is the third data type that an expression can return.

1. Select the **Value** property of the **Constant** member, type **Null,** and then press ENTER.
2. In the browser, notice that the cells in the **Constant** column appear to be empty.

	MeasuresLevel	
+ Category	Sales Dollars	Constant
All Product	$76,741.28	
+ Bread	$30,600.42	
+ Dairy	$32,533.02	
+ Meat	$13,607.84	

You cannot leave the **Value** property empty. If you want to create an empty cell, you must use the keyword **Null**.

Tip When you first create a new calculated member, assign **Null** as its value. You can then see the empty cells in the browser, which helps you visualize the context for the MDX expression.

► To rename a calculated member

In this procedure, you will learn how to rename an existing calculated member.

1. Select the **Constant** calculated member.
2. In the Properties pane, select the **Name** property.
3. Type **Expression** as the new name, and then press ENTER.

	MeasuresLevel	
+ Category	Sales Dollars	Expression
All Product	$76,741.28	
+ Bread	$30,600.42	
+ Dairy	$32,533.02	
+ Meat	$13,607.84	

Notice that the name at the top of the calculated column changes to **Expression**.

► To create a simple numeric formula

Delivery Tip
ConstantNumericD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

In this procedure, you will learn that the value calculated by an MDX expression can come from arithmetic or string text manipulation.

1. In the **Expression** member, click the **Value** property, type **50+17** and then press ENTER.
2. In the browser, notice that the result of the expression—the number 67—appears in all the cells of the **Expression** column.

	MeasuresLevel	
+ Category	Sales Dollars	Expression
All Product	$76,741.28	67
+ Bread	$30,600.42	67
+ Dairy	$32,533.02	67
+ Meat	$13,607.84	67

Again, the grid is similar to a spreadsheet, where the value or formula that you type is automatically copied into all of the relevant cells—that is, it is as if there were several different copies of the formula, each one calculating a value independently of the others.

In an MDX expression, you can use any standard arithmetic operator:

- Plus sign (+) for addition
- Minus sign (-) for subtraction
- Asterisk (*) for multiplication
- Slash mark (/) for division
- Caret (^) for raising to a power (exponent)

MDX does not do type conversions. In a numeric expression, both operands must be numbers. In a string expression, both operands must be strings.

► To create a simple string formula

Delivery Tip
ConstantStringD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

In this procedure, you will learn that the value of an MDX expression can also be a string created by using a plus sign (+) to join together two text strings.

1. Click the **Value** property of the **Expression** member, type **"Hello" + ", " + "World"** and then press ENTER.

	MeasuresLevel	
+ Category	Sales Dollars	Expression
All Product	$76,741.28	Hello, World
+ Bread	$30,600.42	Hello, World
+ Dairy	$32,533.02	Hello, World
+ Meat	$13,607.84	Hello, World

Delivery Tip
The font in the Calculated Member Builder is small. You can copy the expression from the Value box, paste it into Microsoft Notepad, and then enlarge the font in Notepad so that students can more easily read the text.

2. In the browser, notice that the result of the expression—Hello, World—appears in all of the cells of the **Expression** column.

   If you are familiar with Microsoft® Excel or Microsoft Visual Basic® string concatenation, be aware that in an MDX expression you must use a plus sign (+). This is different from an Excel formula, which uses an ampersand (&) to combine text strings. In Visual Basic, you can use either a plus sign or an ampersand.

   Unlike Excel or Visual Basic, an MDX expression does not automatically convert a number to a text string, so you cannot use an expression, such as "The value is " + 45. You can combine a text string that looks like a number, such as "The value is " + "45".

   An MDX expression is like a spreadsheet formula that automatically fills all the appropriate cells. The formula is calculated in each cell, but you do not have to worry about copying the formula to new cells as the row and column headings change.

► To delete the Expression calculated member

- In the **Basic Sales** cube, right-click the **Expression** calculated member, select **Delete**, click **OK**, and then click **Yes** to save the cube.

MDX Expressions

Topic Objective
To highlight the types of data values that can be used in an expression.

Lead-in
The most important use for an MDX expression is as the value of a calculated member.

- **Are Used to Create Calculated Members**
- **Function Like Spreadsheet Formulas**
  - Calculated for each cell of browser grid
- **Are Entered by Using the Calculated Member Builder**
- **Are Edited by Using Calculated Member Builder or Value Property**
- **Are Renamed by Using Name Property**
- **Expression Can Return:**
  - Number
  - Text String
  - Empty Value

One of the most important uses of MDX expression is to create calculated members. Calculated members add the calculating ability of a spreadsheet to an online analytical processing (OLAP) database.

Delivery Tip
Because students have already been exposed to the constant expressions list in the previous group activity, treat this topic as a review—that is, do not spend a lot of time.

When using MDX expressions, consider the following facts and guidelines:

- An MDX expression functions like a spreadsheet formula that is calculated for each relevant cell of the browser grid. If you think of the browser grid as a spreadsheet, you can think of a calculated member as a value or formula that automatically fills all the cells of the calculated member. This differs from a spreadsheet, where a user must copy values into new rows and delete them from old rows.
- To create a new calculated member, you enter an MDX expression in the **Value expression** box of the Calculated Member Builder.
- To edit an existing calculated member, you can either use the Calculated Member Builder or change the expression directly in the **Value** property for the calculated member.
- To rename an existing calculated member, simply type a new value as the **Name** property for the member.
- An MDX expression can return only the following types of values:
  - A numeric value.
  - A string value. You must enclose the string in double quotation marks.
  - An empty value. The constant **Null** creates an empty cell. An empty cell behaves like zero when used in an arithmetic expression, but is ignored when calculating averages. If you want to create an empty cell, you must use the keyword **Null**.

Tip When you first create a new calculated member, assign **Null** as its value. You can then see the empty cells in the browser, which helps you think of an expression in concrete terms.

Expression Operators

Topic Objective
To explain the elements of expression operators.

Lead-in
Think of MDX expressions as simple formulas.
In the same way that you can create formulas in a worksheet cell, you can create simple formulas by using an MDX expression.

- **Arithmetic or String Used in Creating MDX Expressions**
- **Arithmetic Operators Combine Numeric Values**
  - Add: +
  - Subtract: –
  - Multiply: *
  - Divide: /
  - Raise to a power: ^
- **String Operator Combines String Values by Using +**
- **Cannot Combine Numeric and String Values**

An MDX expression is like a spreadsheet formula that automatically fills all of the appropriate cells by using arithmetic operators or string text. The formula is calculated in each cell, but unlike a spreadsheet formula, you do not have to copy the expression operators to new cells as the row and column headings change.

An MDX expression returns a value that can result from an arithmetic or string text manipulation.

Example 1

If you want to create a numeric expression by multiplying the value 10 by the value 8, you use the following MDX formula:

```
10 * 8
```

This result is the number 80.

Example 2

If you want to create the string expression “Hello, World” from three separate strings, you use the following MDX formula:

```
“Hello” + “, ”+ “World”
```

This results in the string “Hello, World”.

Tip You can include spaces between tokens in an MDX expression. Spaces around punctuation make an expression much easier to read. You cannot include extra spaces in the name of a member or other cube object, nor can you include extra spaces within quotation marks.

Consider the following facts and guidelines when using MDX expression operators:

- In an MDX expression, you can use any standard arithmetic operator:
  - Plus sign (+) for addition
  - Minus sign (-) for subtraction
  - Asterisk (*) for multiplication
  - Slash mark (/) for division
  - Caret (^) for raising to a power (exponent)
- The value of an MDX expression can also be a string created by using a plus sign that joins together two text strings. This is different from an Excel formula, which uses an ampersand (&) to combine text strings. In Visual Basic, you can use either a plus sign or an ampersand.
- MDX does not do type conversions—that is, it does not combine numeric and string values. In a numeric expression, both operands must be numbers. In a string expression, both operands must be strings.

  This differs from Excel and Visual Basic. You cannot use an expression, such as “The value is ” + 45. Rather, you would need to use an expression where both parts are strings, such as “The value is ” + “45”.

Lab A: Using Expressions from Constants

Topic Objective
To introduce the lab.

Lead-in
In this lab, you will create simple calculated members by using constants and operators.

Explain the lab objectives.

Objectives

After completing this lab, you will be able to:

- Use the Calculated Member Builder.
- Create simple expressions by using constants.

Prerequisites

Before working on this lab, you must have successfully completed modules 1 and 2 in course 2093A, *Implementing Business Logic with MDX in Microsoft SQL Server™ 2000*.

Estimated time to complete this lab: 10 minutes

Exercise 1
Creating a Constant Measure

Delivery Tip
The procedures in this exercise essentially replicate the previous group activities, but without the answers.

In this exercise, you will create a calculated member on the **Measures** dimension of the **Basic Sales** cube, assigning it various expressions that use only constant values.

As you complete each procedure, compare the result shown in the procedure screen shot to the result on your own computer. If there is a difference, recheck your entry and refer back to the related group activity procedures as necessary.

If you still cannot reconcile your result, then refer to the answer file, which is located in:

C:\MOC\2093A\Labfiles\L03\Answers

You can copy and paste expressions from this file into the **Value expression** box for any given procedure.

Before beginning the exercises in this lab, verify that the following objects exist in Analysis Manager:

- The **Market** database
- The **Basic Sales** virtual cube

If these objects do not exist in Analysis Manager, you must perform the steps in the following procedure before you continue with this lab.

► To restore the Market database

1. In the left pane of Analysis Manager, expand the Microsoft Analysis Services folder.
2. Right-click the **Server** icon, and then click **Restore Database**.
3. Navigate to the C:\Moc\2093A\Batches folder.
4. Select **Market.cab**, click **Open**, and then click **Restore**.

► To clear calculated members from the Basic Sales cube

If the **Basic Sales** cube contains calculated members from a preceding group activity or lab, delete them all before starting this lab.

1. Right-click the **Basic Sales** cube, and then click **Edit**.
2. Expand the Calculated Members folder, and then select the bottom calculated member.
3. Press DELETE, and when prompted, click **Yes**.

   This automatically selects the new bottom calculated member.
4. Repeat step 3 until all calculated members are removed.
5. Save and close the **Basic Sales** cube.

► To create and test a member with a constant numeric value

In this procedure, you will create a calculated member in the **Basic Sales** virtual cube by entering a constant numeric value. ConstantNumber.txt in the Answer folder is the completed MDX expression used in this procedure.

1. In the **Market** database, expand the Cubes folder, right-click the **Basic Sales** virtual cube, and then click **Edit**.
2. Click the **Insert Calculated Member** button.
   - In the **Calculated Member** dialog box, type **Constant Value** as the name of the new member.
   - In the **Value** property box, type **150**
   - To create the member, click **OK**.
3. In the preview data pane—also called the browser—notice the new column labeled **Constant Value**. Verify that the browser content is similar to the following table.

	MeasuresLevel	
+ Category	Sales Dollars	Constant Value
All Product	$76,741.28	150
+ Bread	$30,600.42	150
+ Dairy	$32,533.02	150
+ Meat	$13,607.84	150

► To create and test a member with a constant string value

In this procedure, you will modify an existing calculated member to display a constant string value. ConstantText.txt in the Answer folder is the completed MDX expression used in this procedure.

1. In the **Value** property, change the expression for the **Constant Value** member to display "Goodbye".

   What MDX expression did you use?

   "Goodbye"

   _____

2. Press ENTER, and then verify that the browser content is similar to the following table.

	MeasuresLevel	
+ Category	Sales Dollars	Constant Value
All Product	$76,741.28	Goodbye
+ Bread	$30,600.42	Goodbye
+ Dairy	$32,533.02	Goodbye
+ Meat	$13,607.84	Goodbye

► To create and test a member with a constant empty value

In this procedure, you will modify an existing calculated member to display empty cells. ConstantNull.txt in the Answer folder is the completed MDX expression used in this procedure.

1. Change the expression for the **Constant Value** member to display empty cells.

   What MDX expression did you use?

   Null

2. Press ENTER, and then verify that the browser content is similar to the following table.

	MeasuresLevel	
+ Category	Sales Dollars	Constant Value
All Product	$76,741.28	
+ Bread	$30,600.42	
+ Dairy	$32,533.02	
+ Meat	$13,607.84	

► To rename a member and assign it an expression that uses numeric constants

In this procedure, you will assign an expression that uses constant numeric values to a calculated member and give the calculated member a new name. ConstantNumeric.txt in the Answer folder is the completed MDX expression used in this procedure.

1. Change the name of the **Constant Value** member to **Constant Expression**, and then assign it an expression of 4 plus 5.

   What MDX expression did you use?

   4+5

2. Press ENTER, and then verify that the browser content is similar to the following table.

	MeasuresLevel	
+ Category	Sales Dollars	Constant Expression
All Product	$76,741.28	9
+ Bread	$30,600.42	9
+ Dairy	$32,533.02	9
+ Meat	$13,607.84	9

► To create and test a member with an expression that uses string constants

In this procedure, you will modify an existing calculated member to display the result of an expression that uses constant string values. ConstantString.txt in the Answer folder is the completed MDX expression used in this procedure.

1. Change the expression for the Constant Expression calculated member to produce the string "Goodbye, World" by combining three strings: one for each word and one for the comma and space that separate the words.

   What MDX expression did you use?

   ```
   "Goodbye" + ", " + "World"
   ```

   _____

2. Press ENTER, and then verify that the browser content is similar to the following table.

	MeasuresLevel	
+ Category	Sales Dollars	Constant Expression
All Product	$76,741.28	Goodbye, World
+ Bread	$30,600.42	Goodbye, World
+ Dairy	$32,533.02	Goodbye, World
+ Meat	$13,607.84	Goodbye, World

► To delete the Constant Expression calculated member

1. In the **Virtual Cube Editor**, right-click the **Constant Expression** member.
2. On the shortcut menu, click **Delete** and then click **Yes**.

Displaying Member Information

Topic Objective
To provide an overview of the topics that this section covers.

Lead-in
In this section, you will learn how to use MDX functions from the **Functions** list and metadata items from the **Data** list.

- **Group Activity: Displaying Member Information**
- **Using MDX Functions**
- **CurrentMember Function**
- **Name Function**
- **Level Function**

Delivery Tip
Briefly explain the bullets on this slide to provide context for the upcoming group activity where the items are functionally demonstrated.

In this section, you learn how to display metadata information such as member names, level names, and dimension names. You do not display actual values from the cube.

The group activity introduces how to use the **Functions** list and specific functions for displaying member information. Several topics follow this group activity:

- Using MDX Functions

  This topic describes how to access, learn about, and use MDX functions by using the **Functions** list in the Calculated Member Builder.

  Functions in MDX are the foundation for building expressions. In the Calculated Member Builder, you combine functions with member, level, and dimension objects to create expressions that define a calculated member.

- CurrentMember Function

  This topic describes how to display the names of current members by using the **CurrentMember** function. The **CurrentMember** function returns the current member.

- Name Function

  This topic describes how to display a member name by using the **Name** function. The **Name** function returns the name of a metadata object as a constant.

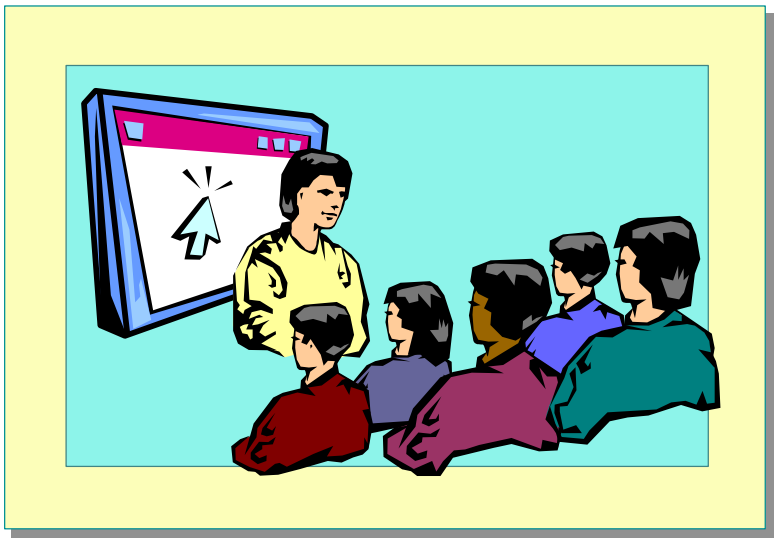
- Level Function

  This topic describes how to display the level name of a current member by using the **Level** function. This function returns the level of the current member.

Group Activity: Displaying Member Information

Topic Objective
To learn how to display member information in a report.

Lead-in
In this group activity, you will learn how to use MDX to display metadata. Metadata describes naming information.



In this group activity, you may follow along on your own computer with your instructor or observe the demonstration.

You will learn how to use MDX functions to display metadata information. Metadata is the naming information that describes the specific dimensions, levels, and members in a cube.

You will also learn how to use the **Functions** list in the Calculated Member Builder. In that list, you will find the **Name** function in the String group and the **CurrentMember** function in the Member group.

You will not retrieve specific numeric values from the cube. Retrieving numeric values is addressed extensively in later modules.

► To display the name of a specific member

Delivery Tip
MemberNameD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

In this procedure, you will learn how to display the name of a single constant member. The same name appears in every cell created by the calculated member.

1. In the **Basic Sales** virtual cube, create a new calculated member. Type **Name** as the name of the member.
2. In the **Functions** list, expand the **String** folder, and then double-click **Name – Member** function.

   All MDX functions are listed in the **Functions** list of the Calculated Member Builder and appear in groups based on what the function returns. The **Name** function returns a text string; therefore, the **Name** function appears in the String group.

   Notice that the **Name** function now appears in the **Value expression** box, and notice the token «Member» that precedes the function name.

Tip When you insert a function from the **Functions** list, you see the syntax for the function with placeholder tokens enclosed in chevrons (« and »). Tokens are easy to replace: simply click anywhere in the token to select it, and then double-click an appropriate item from the **Data** or **Functions** list.

3. Click in the «**Member**» token, and then, in the **Data** list, expand **Product** dimension and **Category** level.

   Member objects have a lollipop-shaped icon.

4. Double-click the **Bread** member to change the expression to:

   ```
   [Product].[Bread].Name.
   ```

5. To return to the browser, click **OK**.

	MeasuresLevel	
+ Category	Sales Dollars	Name
All Product	$76,741.28	Bread
+ Bread	$30,600.42	Bread
+ Dairy	$32,533.02	Bread
+ Meat	$13,607.84	Bread

Notice the word "Bread" in all rows of the **Name** column. The expression displays the name of the specified member, which does not change from member to member.

► To display the name of the current member

Delivery Tip
CurrentNameD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

In this procedure, you will learn how to display the name for each specific product—that is, to display the name of the current member—because displaying a constant member name for all cells is not useful.

1. Edit the **Name** member and then clear the current contents of the **Value expression** box.
2. In the **Functions** list, expand the String folder, and then double-click the **Name – Member** function.
3. Click in the «**Member**» token, in the **Functions** list expand the Members folder, and then double-click **CurrentMember**.

   Rather than enter an explicit member, you want a member that changes from cell to cell. To do this, use the **CurrentMember** function that returns the current member of a dimension.

Point out that the calculated member changes when the **Product** dimension changes, but not when other dimensions change.

4. Click anywhere in the «**Dimension**» token, click in the **Data** list, and then double-click the **Product** dimension to change the expression to:

   ```
   [Product].CurrentMember.Name
   ```

   The **CurrentMember** function requires a dimension as its argument.

5. To return to the browser, click **OK**.

	MeasuresLevel	
+ Category	Sales Dollars	Name
All Product	$76,741.28	All Product
+ Bread	$30,600.42	Bread
+ Dairy	$32,533.02	Dairy
+ Meat	$13,607.84	Meat

Notice the correct name of the product in each row of the **Name** column. The result of the function changes from cell to cell, based on the current member.

6. Expand the **Product** dimension levels and notice that the calculated value of the current member expands as needed.
7. To view a different dimension on the Rows axis, drag the **State** dimension button to the **Category** button.

   The value in the **Name** column changes to **All Product** for all the rows. This occurs because the **All Product** member is selected in the filter list above the grid. The value in the filter list is the current member for that dimension.
8. In the **Product** filter list, click **Bread**.

   The value in the **Name** column changes to **Bread** because **Bread** is now the current member of the **Product** dimension.
9. To put the **Product** dimension back on the Rows axis, drag the **Product** dimension button to the **Country** button.

Note Each cell in the grid has a current member for each dimension. If a dimension is represented on the Rows or Columns axis, the current member is the member that appears on the current row or column. If a dimension appears in the filter area, the current member is the member that appears in the filter box.

► To display the name of the current member's level

Delivery Tip
LevelNameD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

In this procedure, you will learn how to display the name of the level for the current member. A member belongs to a specific level in a dimension.

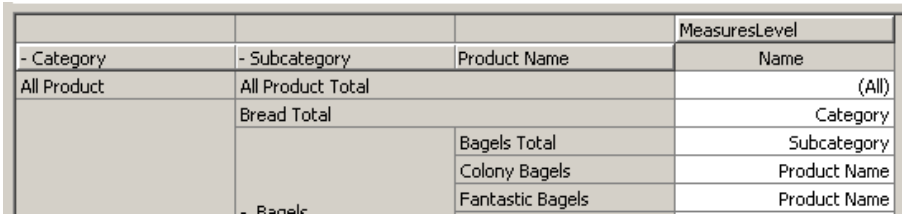
1. Click the **Value** property, click the **ellipsis** button, and then:
   - Clear the contents of the **Value** box, and then in the **String** group, double-click the **Name – Level** function.
   - Click the «**Level**» token, and then in the **Level** group, double-click the **Level** function.
   - Click the «**Member**» token, and then in the **Member** group, double-click the **CurrentMember** function.
   - Click the «**Dimension**» token, and then in the **Data** list, double-click the **Product** dimension.

   When you have finished, the expression will look like the following:

```
Product.CurrentMember.Level.Name
```

2. To return to the browser, click **OK**, and then to display all the levels of the **Product** dimension, double-click the **Category** button and the **Subcategory** button.
3. If necessary, scroll to view the **Name** column.

   This displays a changing level name, depending on the current member of the **Product** dimension. The top few rows of the grid appear similar to the following table.



			MeasuresLevel
- Category	- Subcategory	Product Name	Name
All Product	All Product Total		(All)
	Bread Total		Category
		Bagels Total	Subcategory
		Colony Bagels	Product Name
		Fantastic Bagels	Product Name

   Notice how the level name changes for each member.

► To omit the CurrentMember function for a dimension

Delivery Tip
LevelNameShortD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

In this procedure, you will learn what happens when you omit the CurrentMember function in an expression.

1. Right-click the **Name** calculated member, and then click **Edit**.

   Delete the **CurrentMember** function and the period that follows it from the expression. The result is

   ```
   [Product].Level.Name
   ```

2. To return to the browser, click **OK**.

   The display does not change. The **Level** function works only with a member, not with a dimension. The **CurrentMember** function, however, is the default function for a dimension. If you include a dimension name in a place where a member is required—such as before the **Level** function—you implicitly get the current member of that dimension.

Note The **CurrentMember** function is extremely common in MDX expressions, but you often do not see it because it can often be omitted.

Delivery Tip
DimensionNameD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

► To omit the CurrentMember function for a dimension

In this procedure, you will learn what happens when you omit the **CurrentMember** function before the Name function.

1. Right-click the **Name** calculated member, and then click **Edit**.
2. Delete the **Level** function and the period that follows it from the expression. The result is

```
[Product].Name
```

3. To return to the browser, click **OK**, and then scroll as needed to view the **Name** member.

			MeasuresLevel
- Category	- Subcategory	Product Name	Name
All Product	All Product Total		Product
	Bread Total		Product
		Bagels Total	Product
		Colony Bagels	Product

All the cells of the **Name** calculated member display **Product**—the name of the dimension. The **CurrentMember** function can be omitted after the dimension name only in a context where a member is required. The **Name** function can follow a dimension as well as a member, so omitting the **CurrentMember** function displays the name of the dimension.

► To delete the Name calculated member

- In the **Virtual Cube Editor**, delete the **Name** calculated member.

Using MDX Functions

Topic Objective
To review how to use the **Functions** list in the Calculated Member Builder.

Lead-in
The **Functions** list in the Calculated Member Builder gives you access to many different functions that return values.

- **Function List Groupings**
  - String functions
  - Member functions
  - Numeric functions
- **Functions List Features**
  - View syntax
  - Double-click to add to Value expression box
  - Appears with tokens

Functions in MDX are the foundation for building expressions. In the Calculated Member Builder, you combine functions with member, level, and dimension objects to create expressions that define a calculated member.

Delivery Tip
Because students have already been exposed to the **Functions** list in the previous group activity, treat this topic as a review—that is, do not spend a lot of time.

Functions List Groupings

The **Functions** list in the Calculated Member Builder is a useful tool for finding and learning about functions. The **Functions** list generally groups functions by the type of value the function returns. For example:

- If the function returns a string, it appears in the String group.
- If the function returns a member, it appears in the Member group. An example of a member function is the **CurrentMember** function.
- If the function returns a number, it appears in the Numeric group.

Functions List Features

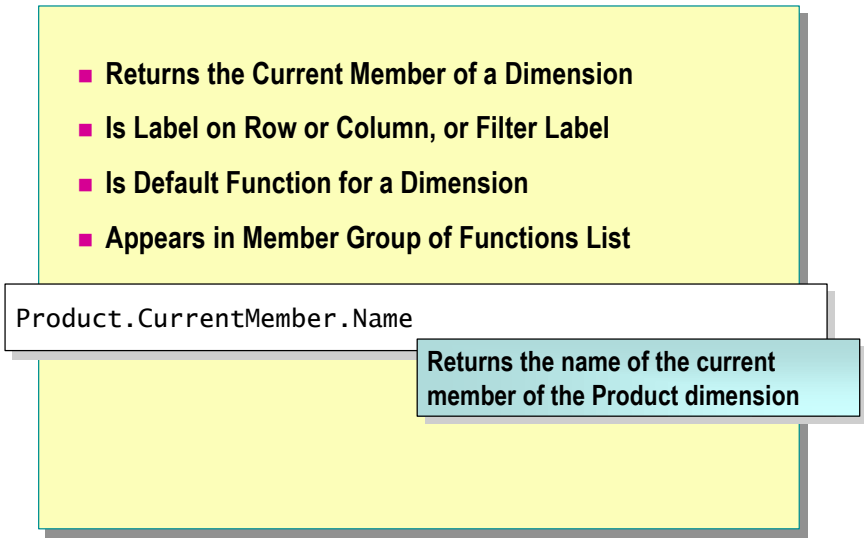
The following are the principal features of the **Functions** list:

- If you select a function name in the **Functions** list, you can view the syntax for the function at the bottom of the Calculated Member Builder.
- If you double-click the function name in the **Functions** list, it appears at the current location of the cursor in the **Value expression** box. You can also select a function and click the **Insert** button to insert it in the expression.
- After you have inserted a function, it appears with tokens as placeholders for each necessary object such as a dimension, a level, or a member.

CurrentMember Function

Topic Objective
To review the **CurrentMember** function.

Lead-in
If you want an expression to return a different value for each member, use the **CurrentMember** function.



An MDX expression that returns a constant value is not typically useful. In order for an expression to change from row to row, you must link the expression to the current member of a dimension. The **CurrentMember** function allows you to do that.

Syntax

«*Dimension*».CurrentMember

Example

The following example returns the name of the current member of the **Product** dimension.

```
Product.CurrentMember.Name
```

Delivery Tip
Because students have already been exposed to the **CurrentMember** function in the previous group activity, use this topic as review—that is, do not spend a lot of time.

When using the **CurrentMember** function, consider the following facts and guidelines:

- Each cell in a browser grid has a current member for each dimension. If a dimension appears on the Rows or Columns axis, the current member is the member that appears on the current row or column. If a dimension appears in the filter area, the current member is the member that appears in the filter box.
- The **CurrentMember** function is the default function for a dimension—that is, if you use a dimension name in a context where a member is required, MDX will attempt to insert the **CurrentMember** function to convert the dimension to a member.

  For example, in the expression `Product.Level.Name`, the word Product is a dimension, but the **Level** function must be preceded by a member, so MDX interprets the expression as if it were `Product.CurrentMember.Level.Name`.

  If, however, a dimension is valid in the context, MDX does not insert the **CurrentMember** function.

For example, in the expression `Product.Name`, the word Product is a dimension, but because you can precede the **Name** function with a dimension, the expression returns the name of the dimension, not the name of the current member of the dimension.

- Because it returns a member object, the **CurrentMember** function appears in the Member group of the **Functions** list.

Related Functions

MDX includes the following other functions that display metadata similar to the **CurrentMember** function:

Delivery Tip
These related functions are included for reference only. Do not spend any time explaining them.

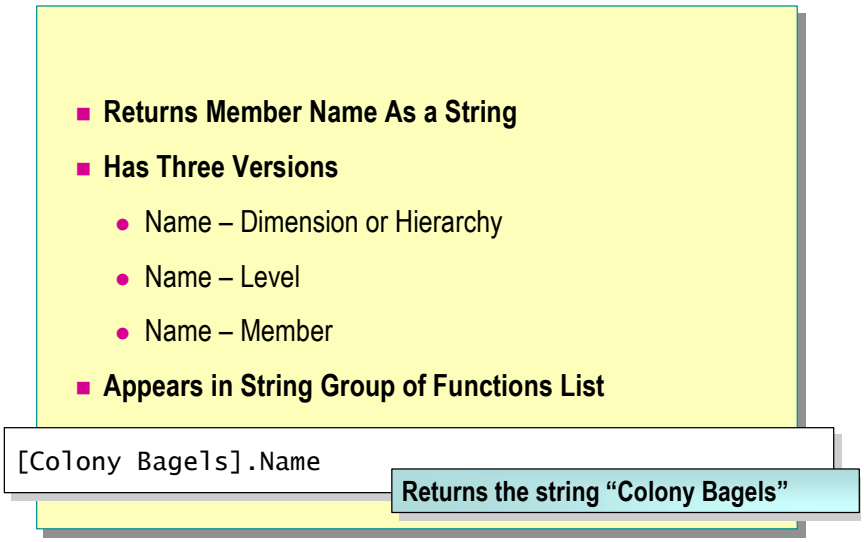
- The **DefaultMember** function returns the default member of a dimension.
- The **DataMember** function returns the system-generated data member associated with a non-leaf member of a parent-child dimension.

You can find more information about these functions by searching on the function name in Microsoft SQL Server Books Online.

Name Function

Topic Objective
To review the **Name** function.

Lead-in
MDX expressions return metadata—specifically, the name of a dimension, level, or member. **Name** is the most elemental function.



An MDX expression returns either a number or a text string. **Name** is a string function that returns the name of a dimension, a level, or a member depending on the object that precedes it.

Syntax

«*Member*».Name

«*Level*».Name

«*Dimension*».Name

Example 1

The following example returns the name of the current member of the **Product** dimension.

```
Product.CurrentMember.Name
```

Example 2

The following example returns the name of the level for the current member of the **Product** dimension.

```
Product.CurrentMember.Level.Name
```

By using the **CurrentMember** function in conjunction with the **Name** function, the calculated member generates a different value for each dimension member.

Delivery Tip
Because students have already seen the **Name** function in the previous group activity, use this topic as a review—that is, do not spend a lot of time.

When you use the **Name** function, consider the following facts and guidelines:

- There are three versions of the **Name** function:
  - **Name – Dimension**
  - **Name – Level**
  - **Name – Member**

  Which one of these three you use depends on the kind of metadata object you want to display. All three functions produce the same results. That is, if you select the **Name – Dimension** function and then put a Level object in front of it, you will automatically get the **Name – Level** version of the function. The three entries in the **Functions** list simply give you helpful tokens for building the rest of the expression.

  Note There is also a **Hierarchy** version of the **Name** function. Pragmatically, you can simply treat the **Hierarchy** version the same as the **Dimension** version. If a dimension has multiple hierarchies, you must treat each hierarchy as if it were a distinct dimension.

- Because name is a text string, the **Name** function and its variations appear in the String group of the **Functions** list.

Related Functions

MDX includes the following other functions that display metadata similar to the **Name** function:

Delivery Tip
These related functions are included for reference only. Do not spend any time explaining them.

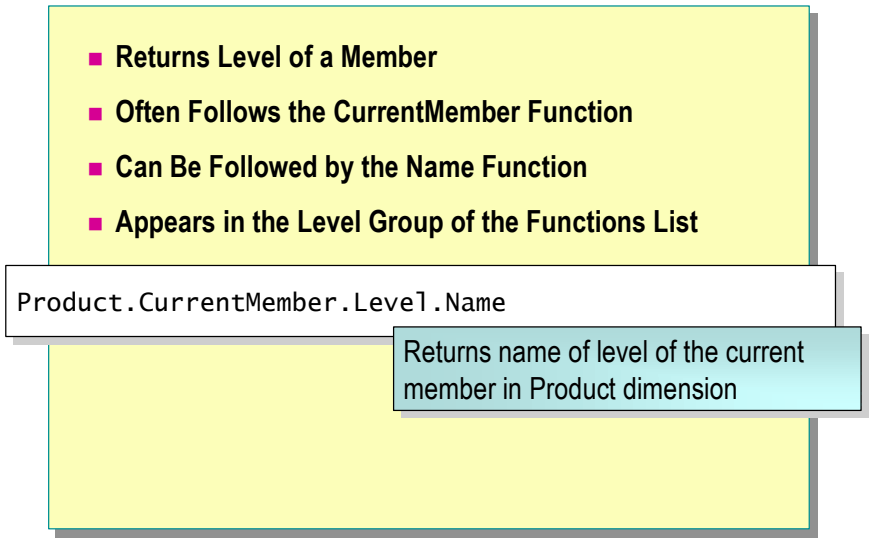
- The **UniqueName** function can follow a member, a level, a dimension, or a hierarchy. It returns a string that includes the entire hierarchy for the dimension, level, or member.
- The **Ordinal** function can follow only a level object. It returns the position of a level in the hierarchy, with zero for the All level.
- The **UserName** function appears all by itself, without following any other object. It returns the name of the current user, including both the domain name and the user identification. The name of the current user is independent of the current cube, so the **UserName** function stands completely alone—as if it were a string constant.

You can find more information about these functions by searching on the function name in Microsoft SQL Server Books Online.

Level Function

Topic Objective
To review the **Level** function.

Lead-in
You use the **Level** and **Ordinal** functions to return level information about a current member.



You can return the level for a member by using the **Level** function. After you have a level object, you can display that level's name.

Syntax

«*Dimension*».CurrentMember.Level

Example

To return the name of the level for the current member of the **Product** dimension, use the following expression:

```
Product.CurrentMember.Level.Name
```

Delivery Tip
Because students have already been exposed to the **Level** function in the previous group activity, treat this topic as a review—that is, do not spend a lot of time.

When you use the **Level** function, consider the following facts and guidelines:

- The **Level** function can follow only a member object. It often follows the **CurrentMember** function. You cannot display the level of a dimension. Using the **Level** function after a dimension name returns the level of the current member of that dimension.
- Several MDX functions use a level argument as an object, but in most cases, you specify an explicit level, rather than using the **Level** function. One common use is to combine the **Level** function with the **Name** function to find the name of the level of the current member of a dimension.
- Because it returns a level object, the **Level** function appears in the Level group of the **Functions** list.

Related Functions

MDX includes the following other functions that return metadata objects similar to the **Level** function:

Delivery Tip
These related functions are included for reference only. Do not spend any time explaining them.

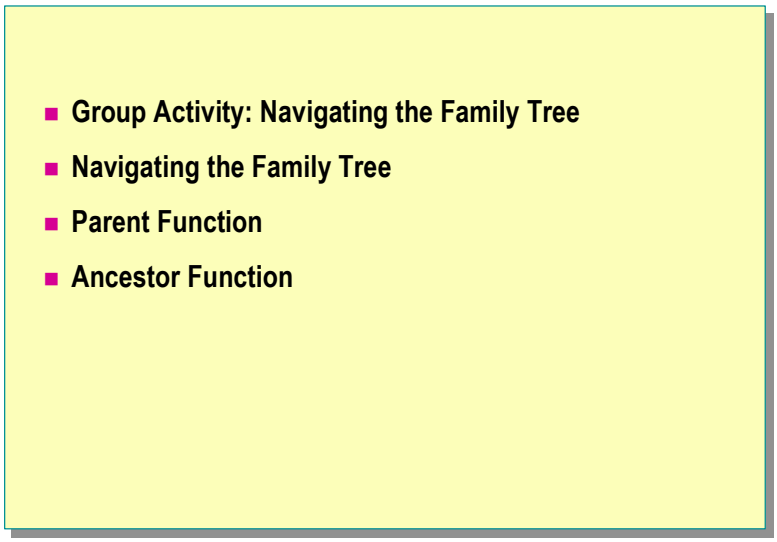
- The **Dimension** function returns a dimension object for a member, level, or hierarchy.
- The **Hierarchy** function returns a hierarchy object for a member or level.

You can find more information about these functions by searching on the function name in Microsoft SQL Server Books Online.

Displaying Family Tree Relatives

Topic Objective
To provide an overview of the materials in this section.

Lead-in
In this section, you will learn how to return the parent or ancestor of any specified member by using the **Parent** and **Ancestor** functions.



Delivery Tip
Briefly explain the bullets on this slide to provide context for the upcoming group activity in which the items are functionally demonstrated.

In this section, you will learn how to use the **Parent** function to find the parent of the current member in a dimension and how to use the **Ancestor** function to find the ancestor of the current member.

The group activity introduces how to navigate the family tree, again by using the **Name** function to display member names. Three topics follow this group activity:

- Navigating the Family Tree

  This topic describes the concepts of parent and ancestor in the context of a dimension hierarchy.

- Parent Function

  This topic describes how to navigate up one level in the family tree by using the **Parent** function. The **Parent** function returns the member that is the parent of a specified member.

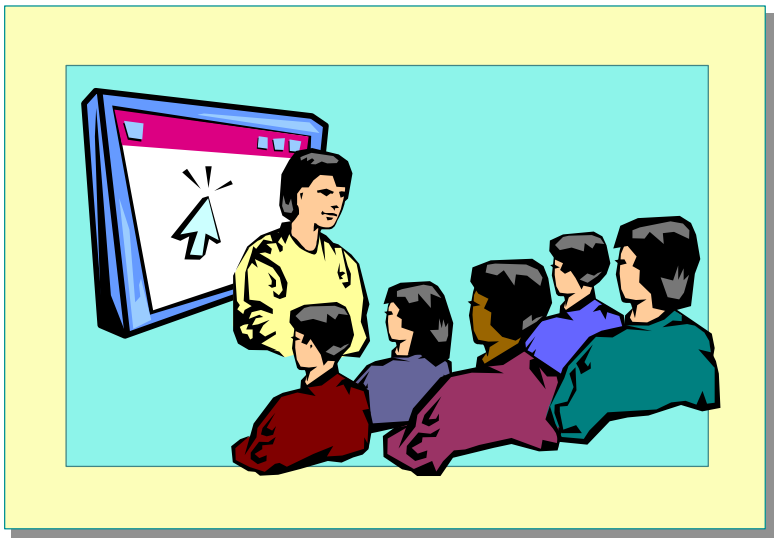
- Ancestor Function

  This topic describes how to navigate up an arbitrary number of levels in the family tree by using the **Ancestor** function. The **Ancestor** function returns the ancestor of a member either at a specified level or at a specified distance away in the hierarchy.

Group Activity: Navigating the Family Tree

Topic Objective
To learn how to navigate from one member to another in the family tree.

Lead-in
In this group activity, you will learn how to return the parent or ancestor of any specified member.



In this group activity, you may follow along on your own computer with your instructor or observe the demonstration

You will learn how to use MDX functions to navigate the family tree of a dimension hierarchy. Specifically, you will learn how to use the **Parent** function and the **Ancestor** function, both of which return members and appear in the Member group of the **Functions** list.

► To display the name of a parent

Delivery Tip
FamilyParentD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

In this procedure, you will learn how you can use the **Parent** function to move from the current member of a dimension to the one parent member.

1. In the **Basic Sales** virtual cube, create a new calculated member on the **Measures** dimension. Type **Family** as the name of the member.
2. In the **Functions** list, expand the String folder, and then double-click the **Name – Member** function.
3. Click in the «**Member**» token, in the **Functions** list, expand the Member folder, and then scroll down to the **Parent** function.
4. In the **Functions** list, double-click **Parent**, and then look at the description at the bottom of the dialog box.

   You must precede the **Parent** function with a member. The **CurrentMember** function returns a member, so you can use it to replace the «Member» token.
5. Click in the «**Member**» token, and then double-click **CurrentMember** in the **Functions** list.
6. Click in the «**Dimension**» token, and then double-click **Product** dimension in the **Data** list. The result is

```
[Product].CurrentMember.Parent.Name
```

7. To accept the new expression, click **OK**.
8. Double-click the **Subcategory** level, and then browse the **Family** column to see how it always displays the appropriate parent of the current product.

		MeasuresLevel	
- Category	+ Subcategory	Sales Dollars	Family
All Product	All Product Total	$76,741.28	
- Bread	Bread Total	$30,600.42	All Product
	+ Bagels	$3,552.28	Bread
	+ Muffins	$13,081.28	Bread
	+ Sliced Bread	$13,966.86	Bread
- Dairy	Dairy Total	$32,533.02	All Product
	+ Cheese	$17,709.11	Dairy
	+ Milk	$7,259.42	Dairy
	+ Sour Cream	$2,746.58	Dairy
	+ Yogurt	$4,817.91	Dairy
- Meat	Meat Total	$13,607.84	All Product
	+ Deli Meats	$4,337.12	Meat
	+ Fresh Chicken	$1,143.56	Meat
	+ Frozen Chicken	$3,214.98	Meat
	+ Hamburger	$2,175.56	Meat
	+ Hot Dogs	$2,736.62	Meat

Notice that the name of the parent for the **All Product** member is blank. The **All Product** member is at the top of the hierarchy and does not have a parent. Attempting to display the name of a nonexistent member does not return an error, but simply an empty cell.

► To display the name of an ancestor at a specific level

Delivery Tip
FamilyAncestorLevelD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

In this procedure, you will learn how you can use the **Ancestor** function to move from the current member of a dimension to a member at a specific higher level.

1. Click the **Value** property of the **Family** member, click the **ellipsis** button, and then clear the contents of the **Value expression** box.
2. In the **Functions** list, expand the String folder, and then double-click the **Name – Member** function.
3. Click in the «**Member**» token, and then in the **Functions** list, expand the Member folder.

   Notice that two versions of the **Ancestor** function appear in the Members group.
4. Double-click **Ancestor – Level** in the **Functions** list, and then look at the description at the bottom of the dialog box.

   The level version of the **Ancestor** function takes two arguments. The first argument must be a member. The **CurrentMember** function returns a member so that you can use it to replace the «Member» token.
5. Click in the «**Member**» token, and then in the **Functions** list, double-click **CurrentMember**.
6. Click in the «**Dimension**» token, and then in the **Data** list, double-click **Product**.

   Notice the «**Level**» token. The second argument for the function is a level. This function returns the ancestor of a member at the specified level.

7. Click in the «**Level**» token, and then expand the **Product** dimension and double-click the **Category** level. The result is

   ```
   Ancestor([Product].CurrentMember,[Product].[Category]).Name
   ```

8. To accept the new expression, click **OK**.
9. Double-click the **Subcategory** button to view the **Product Name** level, and then scroll as needed to view the **Family** column.
10. Browse the **Family** column to see how it always displays the appropriate category ancestor of the current product.

			MeasuresLevel
- Category	- Subcategory	Product Name	Family
- Dairy	- Yogurt	Gorilla Blueberry Yogurt	Dairy
		Gorilla Strawberry Yogurt	Dairy
- Meat	Meat Total		Meat
	- Deli Meats	Deli Meats Total	Meat
		American Corned Beef	Meat

Delivery Tip
FamilyAncestorDistD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

► To display the name of an ancestor at a relative level

In this procedure, you will learn how to use the **Ancestor** function to move from the current member of a dimension to a member at a relative higher level.

The parent of a member is just a special kind of ancestor—the ancestor that is only one level away. The second variation of the **Ancestor** function allows you to specify the distance from the original member to the desired member. For a parent, the distance is 1. For a grandparent, the distance is 2. For a great-grandparent, the distance is 3.

1. Edit the **Family** member and change the **Value expression** to

   ```
   Ancestor([Product].CurrentMember,2).Name
   ```

2. To display the name of the grandparent for each member, click **OK**, and then scroll as needed to view the **Family** column and the beginning rows of the **Dairy** category.

			MeasuresLevel
- Category	- Subcategory	Product Name	Family
- Bread	- Sliced Bread	Sphinx White Bread	Bread
- Dairy	Dairy Total		
	- Cheese	Cheese Total	All Product
		Booker Cheese Spread	Dairy
		Booker Havarti Cheese	Dairy

   Notice that the names at the **Product Name** level show the category for that product. Notice that the name for the Category level member is blank.

   Category does not have a grandparent, so the **Ancestor** function simply returns an empty cell.

   By using the **Ancestor** function—coupled with the **CurrentMember** function—you can display the ancestor of the current member, either at a relative position in the hierarchy or at a specific level of the hierarchy.

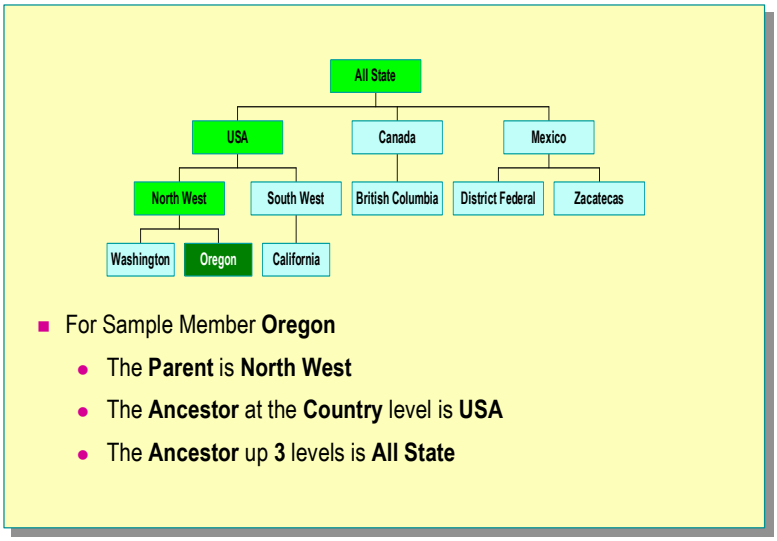
► To delete the Family calculated member

- In the **Virtual Cube Editor**, delete the **Family** calculated member.

Navigating the Family Tree

Topic Objective
To review the concept of navigating from one member to another.

Lead-in
You can use functions that return other members based on the current member.



The **CurrentMember** function returns the member that is on the current row, column, or filter. After you have that member, you can use other functions that return a member that is related to the current member. Some of these functions navigate the family tree.

For example, assume that the current member of the **State** dimension is **Oregon**, and notice the following:

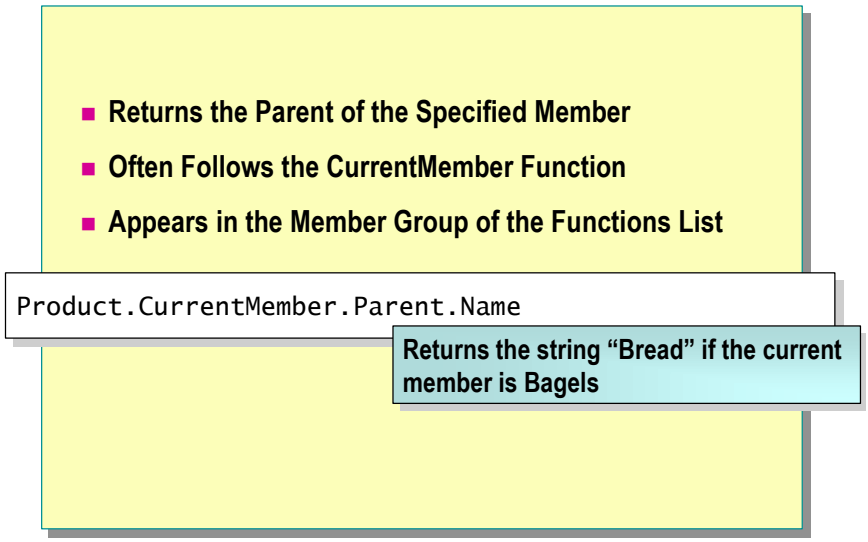
- The **Parent** of the current member is **North West**.
- The **Ancestor** of the current member at the **Country** level is **USA**.
- The **Ancestor** that is **three** levels distant from the current member is **All State**.

These three relationships are captured by the **Parent** function and by the two versions of the **Ancestor** function.

Parent Function

Topic Objective
To review the **Parent** function.

Lead-in
You can use the **Parent** function to return the parent of a specified member.



You can use the **Parent** function to return the parent of a specified member.

Syntax

«*Member*».Parent

Example

The following example returns the string "Bread" if the current member of the **Product** dimension is **Bagels**.

```
Product.CurrentMember.Parent.Name
```

Delivery Tip
Because students have already been exposed to the **Parent** function in the previous group activity, treat this topic as a review—that is, do not spend a lot of time.

When you use the **Parent** function, consider the following facts and guidelines:

- The **Parent** function can follow only a member object. It often follows the **CurrentMember** function. You cannot display the parent of a dimension. Using the **Parent** function after a dimension name returns the parent of the current member of that dimension.
- The **Parent** function appears in the Member group of the **Functions** list.

Related Functions

MDX includes the following other functions that return family tree members similar to the **Parent** function:

Delivery Tip
These related functions are included for reference only. Do not spend any time explaining them.

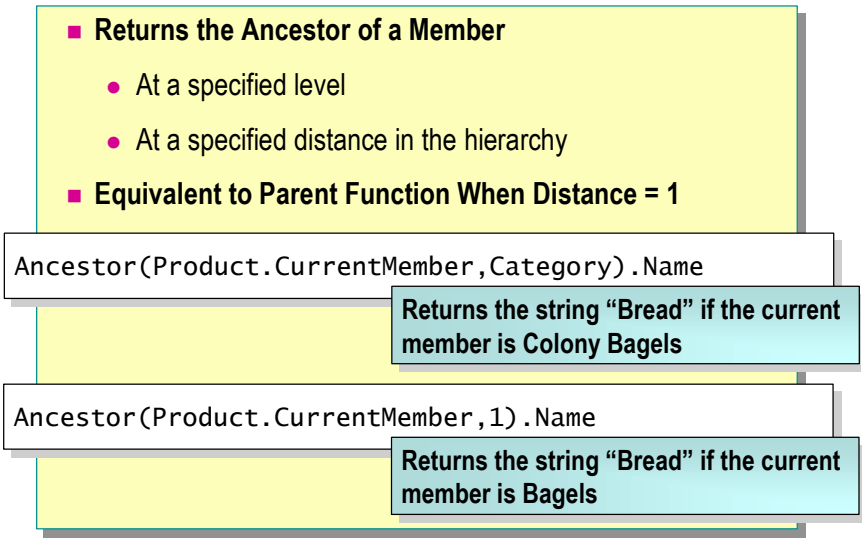
- The **FirstChild** function returns the first child of a specified member.
- The **LastChild** function returns the last child of a specified member.
- The **FirstSibling** function returns the first child of the parent of a specified member.
- The **LastSibling** function returns the last child of the parent of a specified member.

You can find more information about these functions by searching on the function name in Microsoft SQL Server Books Online.

Ancestor Function

Topic Objective
To review the **Ancestor** function.

Lead-in
The **Ancestor** function is a generalized version of the **Parent** function.



Delivery Tip
Because students have already been exposed to the **Ancestor** function in the previous group activity, treat this topic as a review—that is, do not spend a lot of time.

The **Parent** function works with only one level. A generalized version that can work at any level is the **Ancestor** function. By using the **Ancestor** function—coupled with the **CurrentMember** function—you can display the ancestor of the current member, either at a relative position in the hierarchy, or at a specific level of the hierarchy.

The **Ancestor** function has two types:

- Ancestor at a specified level
- Ancestor at a specified distance in the hierarchy

Ancestor at a Specified Level

You can use the **Ancestor** function to display the ancestor of the current member at a specific level of the hierarchy. This use of the **Ancestor** function uses a member as the first argument and a level as the second argument.

Syntax

Ancestor(*«Member»*, *«Level»*)

Example 1

To display the name of the **Category** level ancestor of the current member of the **Product** dimension, use the following expression:

```
Ancestor(Product.CurrentMember,Category).Name
```

Tip Putting the insertion point after one item in a pair of punctuation marks—such as parentheses, brackets, or quotation marks—turns both items bold. Omitting one of a pair of punctuation marks turns the remaining item red. These signals help you properly match punctuation that must come in pairs.

Ancestor at a Specified Distance

A second variation of the **Ancestor** function allows you to specify the distance from the original member to the desired member. For a parent, the distance is 1.

Syntax

Ancestor(*«Member»*, *«Distance»*)

Example 2

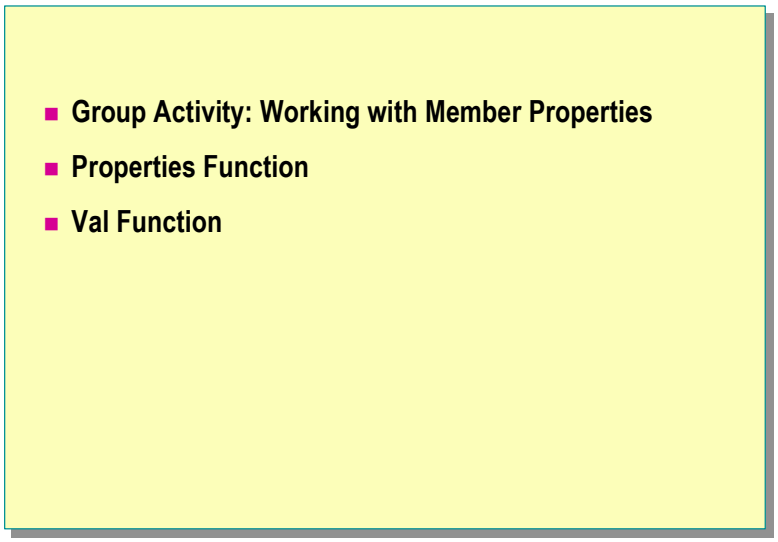
To display the name of the parent of the current member of the **Product** dimension, use the following expression:

```
Ancestor(Product.CurrentMember, 1).Name
```

◆ Working with Member Properties

Topic Objective
To provide an overview of the materials in this section.

Lead-in
In this section, you will learn how to work with member properties.



Delivery Tip
Briefly explain the bullets on this diamond slide to provide context for the upcoming group activity in which the items are functionally demonstrated.

This section introduces how to work with member properties associated with specific members.

Two topics follow the group activity:

- Properties Function

  This topic describes how to retrieve the value from a member property by using the **Properties** function. The **Properties** function returns a string containing the value of the specified member property.

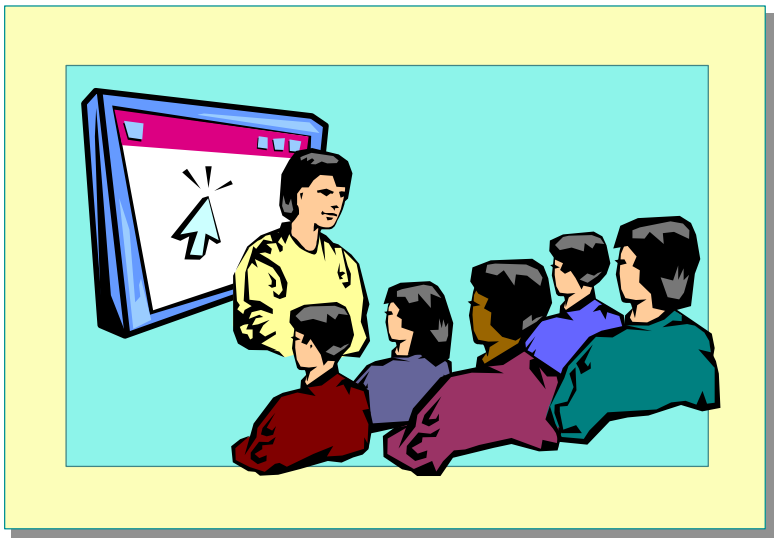
- Val Function

  This topic describes how to convert a string to a number by using the **Val** function. The **Val** function is a Microsoft Visual Basic for Applications (VBA) function that is available from MDX.

Group Activity: Working with Member Properties

Topic Objective
To learn how to display and use member properties of various members.

Lead-in
In this group activity, you will learn how to retrieve the value from a member property.



In this group activity, you may follow along on your own computer with your instructor or observe the demonstration.

You will learn how to use the **Properties** function to retrieve values stored as member properties. Member properties are additional information available for each member of a level. An administrator must use the Analysis Manager to create a member property before you can use MDX to retrieve its value.

You will also learn how to use the **Val** function from the VBA library. The **Val** function allows you to convert a string to a number.

► To display the category manager for a subcategory

Delivery Tip
MemberPropD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

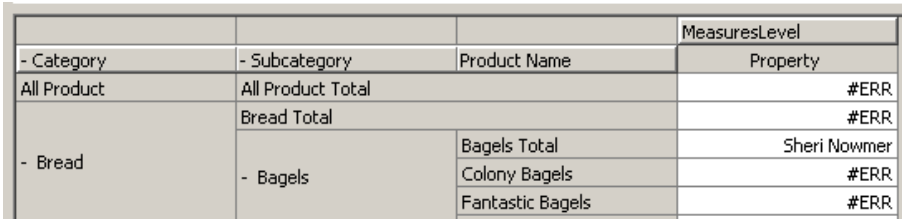
In this procedure, you will learn how to display the member property of a member. The **Product** dimension has a **Category Manager** member property at the subcategory level. You can display the value of this member property for each member.

1. In the **Basic Sales** virtual cube, create a new calculated member, and then type **Property** as the name of the member.
2. In the String group of the **Functions** list, double-click the **Properties** function, replace the «Member» token with the current member of the **Product** dimension and replace the «String Expression» token with **"Category Manager"**, and then click **OK** The final expression should be the following:

   ```
   [Product].CurrentMember.Properties("Category Manager").
   ```

   The name of the member property must be enclosed in quotation marks. If the member property name does not exist, the expression will return an error.

3. Scroll to view the **Property** member, expand all the levels, and then view the category manager name. Notice that the expression returns an error for all levels other than the subcategory level.



			MeasuresLevel
- Category	- Subcategory	Product Name	Property
All Product	All Product Total		#ERR
- Bread	Bread Total		#ERR
	- Bagels	Bagels Total	Sheri Nowmer
		Colony Bagels	#ERR
		Fantastic Bagels	#ERR

Because the category manager member property exists only at the subcategory level, you see an error at all other levels.

► To display the category manager for any subcategory or product

Delivery Tip
MemberPropAncD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

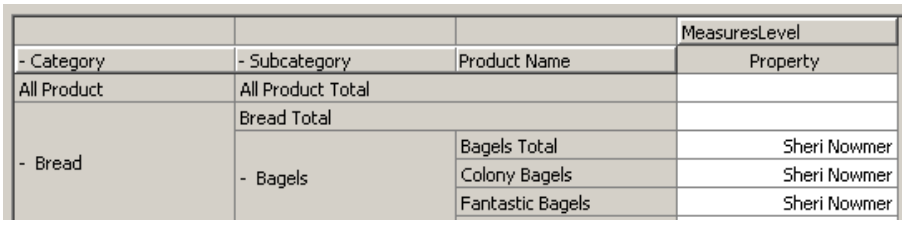
You want to be able to view the appropriate category manager for any product, not just for subcategories. In order to do that, you must navigate from the current member up to the appropriate member at the subcategory level, where the member property exists. The **Ancestor** function can return the appropriate level for the member property.

1. Right-click the **Property** calculated member and click **Edit**.
2. Change the expression to

```
Ancestor(Product,SubCategory).Properties("Category Manager")
```

Note If you use only the dimension name for the member argument of the Ancestor function, the **CurrentMember** function is assumed.

3. Click **OK**, scroll to view the **Property** member, and then expand the levels to show the **Product Name** level and see the manager name appear for all the members in a subcategory.



			MeasuresLevel
- Category	- Subcategory	Product Name	Property
All Product	All Product Total		
- Bread	Bread Total		
	- Bagels	Bagels Total	Sheri Nowmer
		Colony Bagels	Sheri Nowmer
		Fantastic Bagels	Sheri Nowmer

Caution A member property is always returned as a string, even if the member property stores a number. If you want to use the member property in a numeric expression, you must use an external function to convert the string to a number.

► To see an error when using a member property in an arithmetic expression

Delivery Tip
MemberPropErrorD.txt in C:\MOC\2093A\Demo\D03\ Answers contains the completed MDX expression for this procedure.

In this procedure, you will see an error generated when a member property is used in an arithmetic expression. The **Product** dimension has a **Price** member property at the **Product Name** level. This is the list price for each product. It is probably higher than the average price the product actually sells at.

Retrieve the **Price** member property and try to use it in a numeric calculation.

1. Right-click the **Property** calculated member, and then click **Edit**.
2. Change the expression to

   ```
   [Product].CurrentMember.Properties("Price") * 2
   ```

3. To see the error message, click **OK**, and then close the message box.

   The error message indicates that the expression is not valid. This is because you cannot multiply a string by a number. You must convert the **Price** member property value to a number before multiplying the values.

► To use a numeric member property in an arithmetic expression

Delivery Tip
MemberPropVal.txt in C:\MOC\2093A\Demo\D03\ Answers contains the completed MDX expression for this procedure.

In this procedure, you will use the **Val** function to use a numeric property in an arithmetic expression.

MDX does not have an internal function for converting one data type to another, but in an MDX expression you can use almost any function from the VBA or Excel libraries. VBA has a **Val** function that converts a string to a number.

1. Change the expression of the **Property** calculated member to

   ```
   Val(Product.CurrentMember.Properties(“Price”))*2
   ```

2. Click **OK**.

   This version of the expression does not produce an error message box because you can multiply two numbers.

3. In the browser, navigate down to the product level, and then view the prices. The values are double the value stored in the dimension.

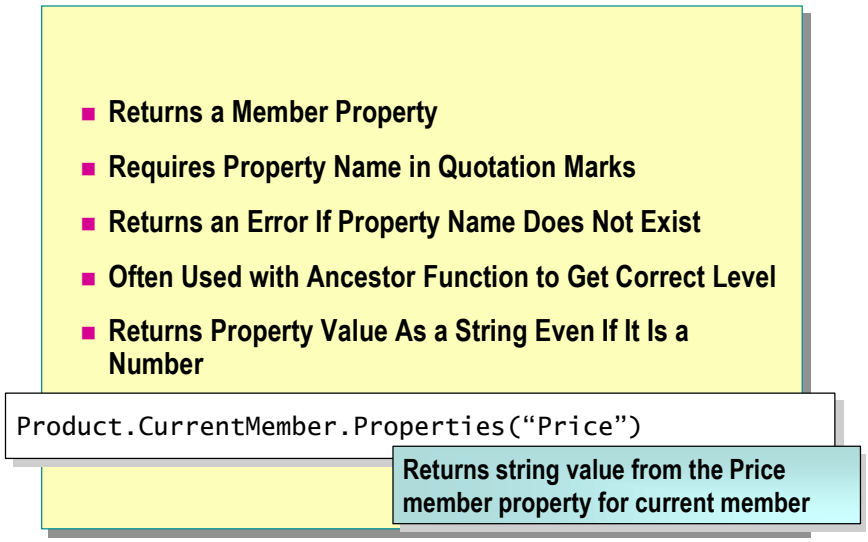
			MeasuresLevel
- Category	- Subcategory	Product Name	Property
All Product	All Product Total		#ERR
- Bread	Bread Total		#ERR
	- Bagels	Bagels Total	#ERR
		Colony Bagels	2.44
		Fantastic Bagels	7.38
		Great Bagels	3.72
		Modell Bagels	6.8
		Sphinx Bagels	3.7
		Muffins Total	#ERR
		Colony Blueberry Muffins	5.3

Note When creating an application that may be used in international locales, you should use the **CDbl** function rather than the **Val** function. The **Val** function always interprets a period as a decimal point, even though some countries use a comma for a decimal point. The **CDbl** function takes into consideration locale settings from the operating system.

Properties Function

Topic Objective
To show how to display the value of a member property by using MDX.

Lead-in
In addition to the name of a member, you can show the text string of a member's member property.



The level of a dimension can have one or more member properties associated with it. Each member can have a unique value for the member property. The **Properties** function allows you to retrieve a member property value for a member.

Syntax

«*Member*».Properties(«*String Expression*»)

Example 1

The following example returns the string value of the **Price** member property for the current member of the **Product** dimension.

```
Product.CurrentMember.Properties("Price")
```

Example 2

The following example returns the **Category Manager** member property for the subcategory of the current member of the **Product** dimension.

```
Ancestor(Product, Subcategory).Properties("Category Manager")
```

Delivery Tip
Because students have already been exposed to the **Properties** function in the previous group activity, treat this topic as a review—that is, do not spend a lot of time.

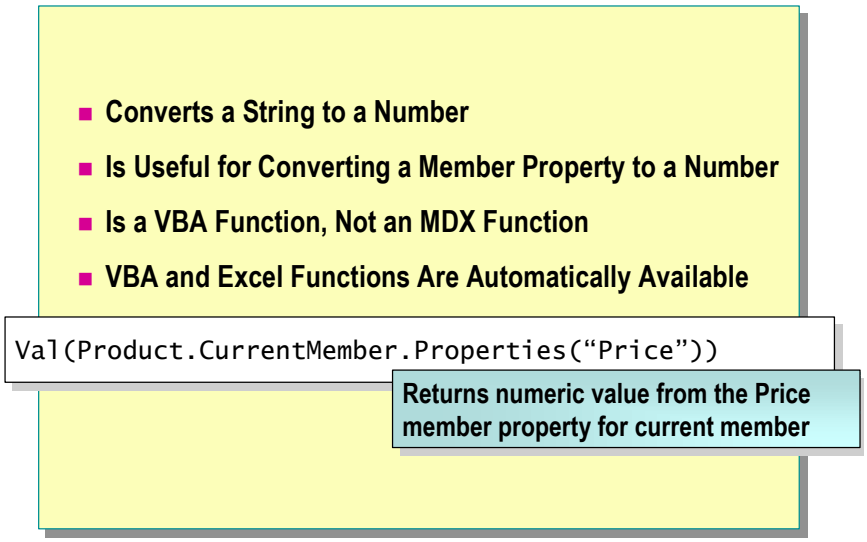
When you use the **Properties** function, consider the following facts and guidelines:

- You must enclose the name of the property in quotation marks because it is the argument to the function.
- If the property name does not exist for a member, the function returns an error value. Spaces in the name are significant, but capitalization is ignored.
- The **Ancestor** function is a useful way to navigate from the current member of a dimension to the level that contains a specified member property.
- The **Properties** function always returns a string value, even if the member property appears to be a number.
- Because the **Properties** function returns a string value, it appears in the String group of the **Functions** list.

Val Function

Topic Objective
To review the **Val** function.

Lead-in
MDX does not have an internal function for converting a string to a number



A member property is always returned as a string, even if the member property stores a number. If you want to use the member property in a numeric expression, you must convert it from a string to a number. MDX does not have an internal function for converting a string to a number.

The **Val** function, which is part of the Microsoft Visual Basic for Applications (VBA) library, converts a text string to a number.

Syntax

Val («*String Expression*»)

Example

The following example returns the number two times the **Price** member property of the current member of the **Product** dimension.

```
Val([Product].CurrentMember. Properties("Price")) * 2
```

Delivery Tip
Because students have already seen the **Name** function in the previous group activity, use this topic as a review—that is, do not spend a lot of time.

When you use the **Val** function, consider the following facts and guidelines:

- The function is useful for converting the value of a member property before including it in an arithmetic expression.
- The **Val** function is not an MDX function. It is part of the VBA library.
- Most VBA and Excel functions are automatically available to MDX expressions. You can search for **function libraries** in Microsoft SQL Server Books Online to see which specific functions are excluded.
- Because the **Val** function is not an MDX function, it does not appear in the **Functions** list in the Calculated Member Builder.

Related Functions

Delivery Tip
These related functions are included for reference only. Do not spend any time explaining them.

VBA includes the following other functions that you may want to use in place of the **Val** function:

- The **CDbl** function converts a string to a number but takes into consideration the regional settings of the operating system.
- The **CLng** function converts a string to a number, rounding to the nearest integer.

You can find other Excel and VBA functions that you can use from MDX by searching for **function libraries** in Microsoft SQL Server Books Online.

Lab B: Displaying Cube Metadata

Topic Objective
To introduce the lab.

Lead-in
In this lab, you will create a calculated member that displays metadata information from the cube.

Explain the lab objectives.

Objectives

After completing this lab, you will be able to:

- Create expressions by using functions that display current member, level, and dimension information.

Prerequisites

Before working on this lab, you must have successfully completed modules 1 and 2 in course 2093A, *Implementing Business Logic with MDX in Microsoft SQL Server 2000*.

Estimated time to complete this lab: 20 minutes

Exercise 1
Displaying Member Information

Delivery Tip
The procedures in this exercise essentially replicate the previous group activities, but without the answers.

Students having difficulty with the procedures should first refer back to the group activity procedures and then go to the answer file for guidance.

In this exercise, you will create and then make modifications to a calculated member in the **Basic Sales** virtual cube on the **Measures** dimension, including assigning the calculated member various expressions that display metadata information.

As you complete each procedure, compare the result as shown in the procedure screen shot to the result on your own computer. If there is a difference, recheck your entry and refer back to the related group activity procedures as necessary.

If you still cannot reconcile your result, then refer to the answer file, which is located in:

C:\MOC\2093A\Labfiles\L03\Answers

You can copy and paste expressions from this file into the **Value expression** box for any given procedure.

► To display the name of the current member

In this procedure, you will create a calculated member in the **Basic Sales** virtual cube by using the **CurrentMember** function. NameMember.txt in the Answer folder is the completed MDX expression used in this procedure.

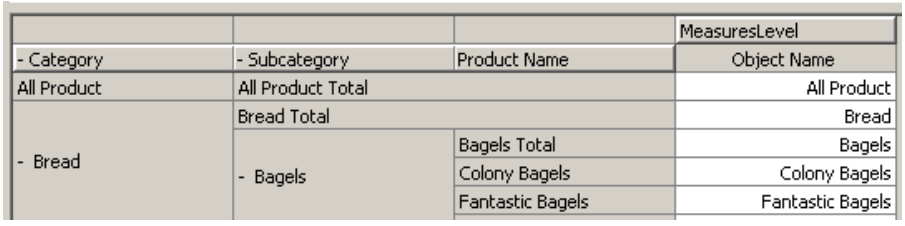
1. If necessary, delete any calculated members from the **Basic Sales** cube, and then create a new calculated member of the **Measures** dimension named **Object Name**.
2. In the **Value expression** box, create an expression to display the name of the current member of the **Product** dimension.

   What MDX expression did you use?

   ```
   Product.CurrentMember.Name
   ```

   _____

3. Browse the cube, scrolling to view the **Object Name** member, and navigating down to all levels of the **Product** dimension, and then, with the **Product Name** level visible, verify that the browser content is similar to the following table.



			MeasuresLevel
- Category	- Subcategory	Product Name	Object Name
All Product	All Product Total		All Product
- Bread	Bread Total		Bread
	- Bagels	Bagels Total	Bagels
		Colony Bagels	Colony Bagels
		Fantastic Bagels	Fantastic Bagels

► To display the name of the current member's level

In this procedure, you will modify a calculated member by using the **Level** function. NameLevel.txt in the Answer folder is the completed MDX expression used in this procedure.

1. Modify the **Object Name** calculated member to display the name of the level of the current member of the **Product** dimension.

   What MDX expression did you use?

   ```
   Product.CurrentMember.Level.Name
   ```

   or

   ```
   Product.Level.Name
   ```

   _____

2. Browse the cube drilling down to all levels of the **Product** dimension, and then, with the **Product Name** level visible, verify that the browser content is similar to the following table.

			MeasuresLevel
- Category	- Subcategory	Product Name	Object Name
All Product	All Product Total		(All)
- Bread	Bread Total		Category
	- Bagels	Bagels Total	Subcategory
		Colony Bagels	Product Name
		Fantastic Bagels	Product Name

► To delete the Object Name calculated member

- In the **Virtual Cube Editor**, delete the **Object Name** calculated member.

Exercise 2
Displaying Family Tree Relatives

Delivery Tip
The procedures in this exercise essentially replicate the previous group activities, but without the answers.

Students having difficulty with the procedures should first refer back to the group activity procedures and then go to the answer file for guidance.

In this exercise, you will create and modify a calculated member on the **Measures** dimension, assigning it various expressions that display parent and ancestor relationships.

In this exercise, you will create a calculated member on the **Measures** dimension of the **Basic Sales** cube, assigning it various expressions that use only constant values.

As you complete each procedure, compare the result as shown in the procedure screen shot to the result on your own computer. If there is a difference, recheck your entry and refer back to the related group activity procedures as necessary.

If you still cannot reconcile your result, then refer to the answer file, which is located in:

C:\MOC\2093A\Labfiles\L03\Answers

You can copy and paste expressions from this file into the **Value expression** box for any given procedure.

► To display the name of the current member's parent

In this procedure, you will create a calculated member in the **Basic Sales** virtual cube by using the **Parent** function. TreeParent.txt in the Answer folder is the completed MDX expression used in this procedure.

1. In the **Basic Sales** virtual cube, on the **Measures** dimension, create a new calculated member called **Tree Name**.
2. In the **Value expression** box, create an expression to display the name of parent of the current member of the **Product** dimension.

   What MDX expression did you use?

   ```
   Product.CurrentMember.Parent.Name
   ```

   or

   ```
   Product.Parent.Name
   ```

---

3. Browse the cube, drilling to all levels of the **Product** dimension. With the **Product Name** level visible, verify that the browser content is similar to the following.

			MeasuresLevel
- Category	- Subcategory	Product Name	Tree Name
All Product	All Product Total		
- Bread	Bread Total		All Product
	- Bagels	Bagels Total	Bread
		Colony Bagels	Bagels
		Fantastic Bagels	Bagels

► To display the name of the current member's category

In this procedure, you will modify a calculated member by using the **Ancestor** function with a level. TreeAncestorLevel.txt in the Answer folder is the completed MDX expression used in this procedure.

1. Change the **Tree Name** member to show the category name for each category, subcategory, and product.

   What MDX expression did you use?

   ```
   Ancestor(Product.CurrentMember,Category).Name
   ```

   or

   ```
   Ancestor(Product,Category).Name
   ```

   _____

2. Browse the cube, drilling to all levels of the **Product** dimension, and then, with the first few products of the **Dairy** category visible, verify that the browser content is similar to the following table.

			MeasuresLevel
- Category	- Subcategory	Product Name	Tree Name
- Bread	- Sliced Bread	Sphinx Wheat Bread	Bread
		Sphinx White Bread	Bread
- Dairy	Dairy Total		Dairy
	- Cheese	Cheese Total	Dairy
		Booker Cheese Spread	Dairy

► To display the name of the current member's grandparent

In this procedure, you will modify a calculated member by using the **Ancestor** function with a distance. TreeAncestorDistance.txt in the Answer folder is the completed MDX expression used in this procedure.

1. Change the **Tree Name** member to show the name of the grandparent product for the current member of the **Product** dimension.

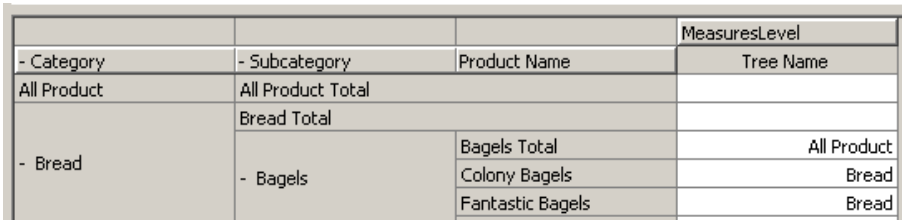
   ```
   Ancestor(Product.CurrentMember,2).Name
   ```

   or

   ```
   Ancestor(Product,2).Name
   ```

   _____

2. Browse the cube, drilling to all levels of the **Product** dimension, and then, with the **Product Name** level visible, verify that the browser content is similar to the following table.



			MeasuresLevel
- Category	- Subcategory	Product Name	Tree Name
All Product	All Product Total		
- Bread	Bread Total		
	- Bagels	Bagels Total	All Product
		Colony Bagels	Bread
		Fantastic Bagels	Bread

► **To delete the Tree Name calculated member**

- In the **Virtual Cube Editor**, delete the **Tree Name** calculated member.

Exercise 3

Working with Member Properties

Delivery Tip
The procedures in this exercise essentially replicate the previous group activities, but without the answers.

Students having difficulty with the procedures should first refer back to the group activity procedures and then go to the answer file for guidance.

In this exercise, you will create and then make modifications to a calculated member in the **Basic Sales** virtual cube on the **Measures** dimension, including assigning the calculated member various expressions that display metadata information from a cube.

As you complete each procedure, compare the result as shown in the procedure screen shot to the result on your own computer. If there is a difference, recheck your entry and refer back to the related group activity procedures as necessary.

If you still cannot reconcile your result, then refer to the answer file, which is located in:

C:\MOC\2093A\Mod3\LabMetadata.

You can copy and paste expressions from this file into the **Value expression** box for any given procedure.

► To display the category manager for a subcategory

In this procedure, you will create a calculated member by using the **Properties** function. Property1.txt in the Answer folder is the completed MDX expression used in this procedure.

1. In the **Basic Sales** virtual cube called **Member Property**, create a new calculated member of the **Measures** dimension.
2. In the **Value expression** box, create an expression to show the name of the category manager for the current member of the **Product** dimension.

   What MDX expression did you use?

   ```
   Product.CurrentMember.Properties("Category Manager")
   ```

   _____

3. Browse the cube, drilling to all levels of the **Product** dimension, and then, with the **Product Name** level visible, verify that the browser content is similar to the following table—that is, you should see an error at any level other than **Subcategory**.

			MeasuresLevel
- Category	- Subcategory	Product Name	Member Property
All Product	All Product Total		#ERR
- Bread	Bread Total		#ERR
	- Bagels	Bagels Total	Sheri Nowmer
		Colony Bagels	#ERR
		Fantastic Bagels	#ERR

► To display a category manager for product or subcategory

In this procedure, you will modify a calculated member by using the **Properties** function in conjunction with the **Ancestor** function. Property2.txt in the Answer folder is the completed MDX expression used in this procedure.

1. Edit the **Member Property** member to show the category manager responsible for each subcategory and product.

   What MDX expression did you use?

   ```
   Ancestor(Product.CurrentMember,SubCategory).
   Properties("Category Manager ").
   ```

   _____

   _____

2. Browse the cube, drilling to all levels of the **Product** dimension, and then, with the first few rows of the **Dairy** category visible, verify that the browser content is similar to the following table.

			MeasuresLevel
- Category	- Subcategory	Product Name	Member Property
- Bread	- Sliced Bread	Sphinx Wheat Bread	Sheri Nowmer
		Sphinx White Bread	Sheri Nowmer
- Dairy	Dairy Total		
	- Cheese	Cheese Total	Derrick Whelply
		Booker Cheese Spread	Derrick Whelply

► To use a price calculated member in a calculation

In this procedure, you will modify a calculated member by using the **Properties** function in conjunction with the **Val** function. PropertyValue.txt in the Answer folder is the completed MDX expression used in this procedure.

1. Edit the **Member Property** member to display the **Price** member property multiplied by 5.

   What MDX expression did you use?

   ```
   Val(Product.CurrentMember.Properties("Price"))*5
   ```

   _____

2. Browse the cube, drilling to all levels of the **Product** dimension, and Then, with the **Product Name** level visible, verify that the browser content is similar to the following table.

			MeasuresLevel
- Category	- Subcategory	Product Name	Member Property
All Product	All Product Total		#ERR
- Bread	Bread Total		#ERR
	- Bagels	Bagels Total	#ERR
		Colony Bagels	6.1
		Fantastic Bagels	18.45

► To delete the Member Property calculated member

- In the **Virtual Cube Editor**, delete the **Member Property** calculated member.

◆ Using Conditional Expressions

Topic Objective
To introduce the topics in the section.

Lead-in
MDX allows you to make decisions by using conditional expressions.

- **Group Activity: Using Conditional Expressions**
- **Conditional Expressions**
- **Multiple Conditions**
- **IIF Function**

Delivery Tip
Explain the bullets on this diamond slide to provide context for the upcoming group activity.

Each item in the bullets will be functionally introduced in the group activity and then reviewed in more complete syntax and other detail in the slide topics that follow.

This section introduces how to create MDX statements that use conditional expressions. Conditional logic is fundamental to sophisticated analysis and model building.

The following topics follow the group activity:

- Conditional Expressions

  This topic describes how to create a conditional expression by using comparison operators.
- Multiple Conditions

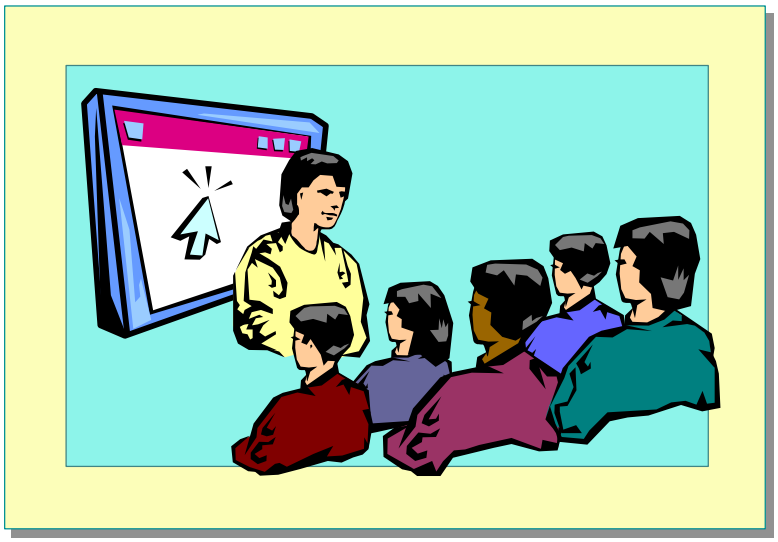
  This topic describes how to combine multiple conditions in a single condition by using conditional operators.
- IIF Function

  This topic describes how to use the **IIF** function to make one expression return different values based on the value of a conditional expression.

Group Activity: Using Conditional Expressions

Topic Objective
To demonstrate how to show names and member properties of various members.

Lead-in
In this demonstration, you will learn how to display the name of dimensions, levels, and members.



In this group activity, you may follow along on your own computer with your instructor or observe the demonstration

You will learn how to use basic conditional expressions. You start by first constructing a simple conditional expression, and then you learn how to use the **IIF** function. In the final step, you create an MDX expression that incorporates multiple conditions.

► To create a basic conditional expression

In this procedure, you learn how to display the result of a simple conditional expression. This allows you to see the 1 or 0 value returned by the expression.

1. In the **Basic Sales** virtual cube, create a new calculated member. Type **Conditional** as the name of the member.

Delivery Tip
ConditionTestD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

2. In the **Value expression** box, construct the following expression:

```
Ancestor([Product].CurrentMember,
[Product].[Category]).Name = "Bread"
```

3. Click **OK**.
4. To show all three levels, in the browser, double-click the **Category** and **Subcategory** buttons, and then scroll to view the **Conditional** member and the first rows of the **Dairy** category. Notice the 1 and 0 values in the **Conditional** column.

			MeasuresLevel
- Category	- Subcategory	Product Name	Conditional
- Bread	- Sliced Bread	Sphinx Wheat Bread	1
		Sphinx White Bread	1
- Dairy	Dairy Total		0
	- Cheese	Cheese Total	0
		Booker Cheese Spread	0

The following are several comparison operators you can use to create a conditional expression:

- Equal to (=)
- Not equal to (<>)
- Greater than (>)
- Less than (<)
- Greater than or equal to (>=)
- Less than or equal to (<=)

► To use a conditional expression by using the IIF function

Delivery Tip
ConditionIFD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

In this procedure, you learn how to use the **IIF** function, which returns different values based on a conditional expression.

Suppose that you wanted to designate all of the products that you could bring to a bread-tasting social. You can modify the **Conditional** member to show which products to bring.

1. Edit the **Conditional** member and then clear the current contents of the **Value expression** box.
2. In the **Functions** list, expand the **String** folder.
3. Select the **IIF – String** function. Notice the description of the function at the bottom of the dialog box.
4. In the **Value** box, type the following expression:

```
IIF(Ancestor([Product].CurrentMember,
[Product].Category).Name = "Bread" , "Bring" , "Don't
Bring" )
```

5. Click **OK**.
6. In the browser, look for the text strings. Notice that they say "Bring" only for the products in the **Bread** category.

			MeasuresLevel
- Category	- Subcategory	Product Name	Conditional
All Product	All Product Total		Don't Bring
- Bread	Bread Total		Bring
	- Bagels	Bagels Total	Bring
		Colony Bagels	Bring
		Fantastic Bagels	Bring

Delivery Tip
ConditionMultipleD.txt in C:\MOC\2093A\Demo\D03\Answers contains the completed MDX expression for this procedure.

► To create a test with multiple conditions

In this procedure, you will learn that a conditional test can be composed of more than one sub test.

Suppose that your bread party changes to a bread and cheese party. You could expand the conditional test to accommodate this multiple condition.

1. Edit the **Conditional** member and then clear the current contents of the **Value expression** box.
2. In **Value** box, construct the following expression:

```
IIF(Ancestor([Product].CurrentMember, [Product].Category
).Name = "Bread" OR Ancestor([Product].CurrentMember,
[Product].Subcategory ).Name = "Cheese" , "Bring" , "Don't
Bring" )
```

3. Click **OK**.
4. In the browser, scroll down to the first few rows of the **Dairy** category and look for the string values in the **Conditional** column. Notice that they now say "Bring" not only for the **Bread** category products but also for products in the **Cheese** subcategory.

			MeasuresLevel
- Category	- Subcategory	Product Name	Conditional
- Bread	- Sliced Bread	Sphinx Wheat Bread	Bring
		Sphinx White Bread	Bring
- Dairy	Dairy Total		Don't Bring
	- Cheese	Cheese Total	Bring
		Booker Cheese Spread	Bring

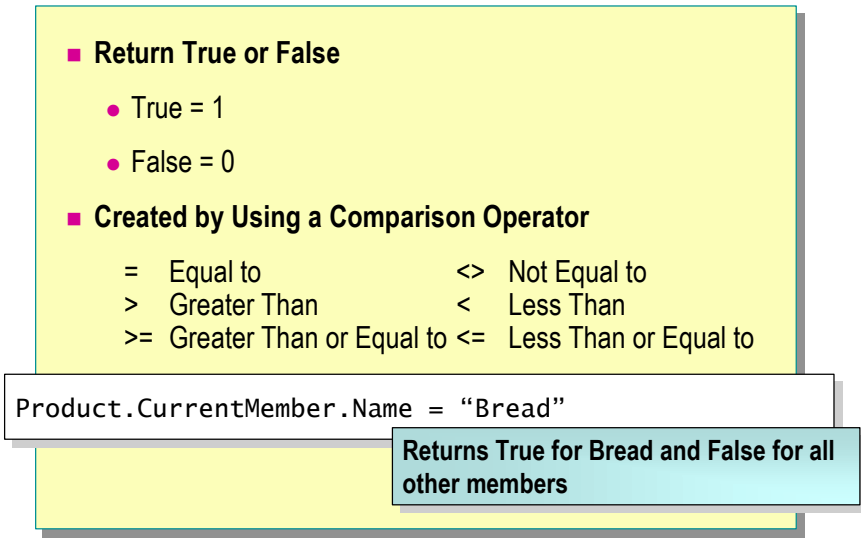
There are several keywords that you can use to combine conditional expressions, for example:

- **OR** returns True if either of two conditional tests returns True.
- **AND** returns True only if both values are True.
- **XOR** returns True if one and only one value is True.
- **NOT** reverses the True and False results of the test.

Conditional Expressions

Topic Objective
To review conditional expressions.

Lead-in
MDX allows you to make comparison tests in an expression.



MDX allows you to make comparison tests in an expression. For example, you can test whether a product belongs in a specific category by testing the name of the product's ancestor. Consider some examples.

Example 1

The following example returns True (1) only if the current member of the **Product** dimension is the **Bread** category.

```
Product.CurrentMember.Name = "Bread"
```

Example 2

The following example returns True (1) only if the current member of the **Product** dimension is a descendant of the **Bread** category.

```
Ancestor(Product.CurrentMember,
  Product.Category).Name = "Bread"
```

Delivery Tip
Because students have already been exposed to the conditional expressions in the previous group activity, use this topic as review—that is, do not spend a lot of time.

When creating conditional expressions, consider the following facts and guidelines:

- The value returned by a conditional expression is either True or False. A True value is represented by the number 1. A False value is represented by the number 0.

Note The values of True as 1 and False as 0 are similar to the usage in Excel, but differ from Visual Basic, in which True is –1 and False is 0.

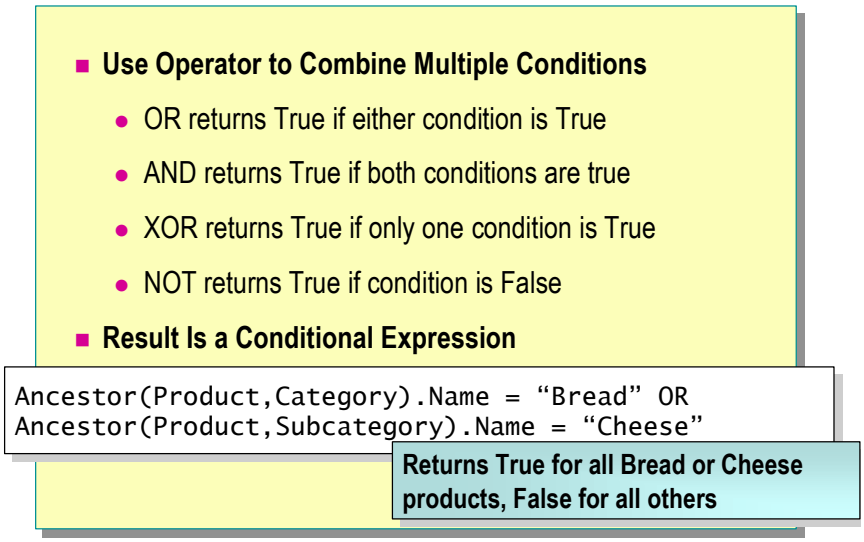
- You create a conditional expression by using a comparison operator. MDX uses the following same comparison operators as Visual Basic and Excel:
  - Equal to (=)
  - Not equal to (<>)
  - Greater than (>),
  - Less than (<)
  - Greater than or equal to (>=),
  - Less than or equal to (<=).

Note Conditional expressions can be used to create advanced security rules in a cube.

Multiple Conditions

Topic Objective
To review combining conditional expressions in one.

Lead-in
Some decisions require more than one condition.



A conditional test can be composed of more than one sub test. The conditional expressions are combined with a logical operator.

Example

The following example displays the text "Bring" for any product that is either in the **Bread** category or in the **Cheese** subcategory.

```
IIF(Ancestor([Product].CurrentMember,
   [Product].Category ).Name = "Bread" OR
Ancestor([Product].CurrentMember,
   [Product].Subcategory ).Name = "Cheese" ,
"Bring" , "Don't Bring" ).
```

Delivery Tip
Because students have already seen multiple conditions in the previous group activity, use this topic as a review—that is, do not spend a lot of time.

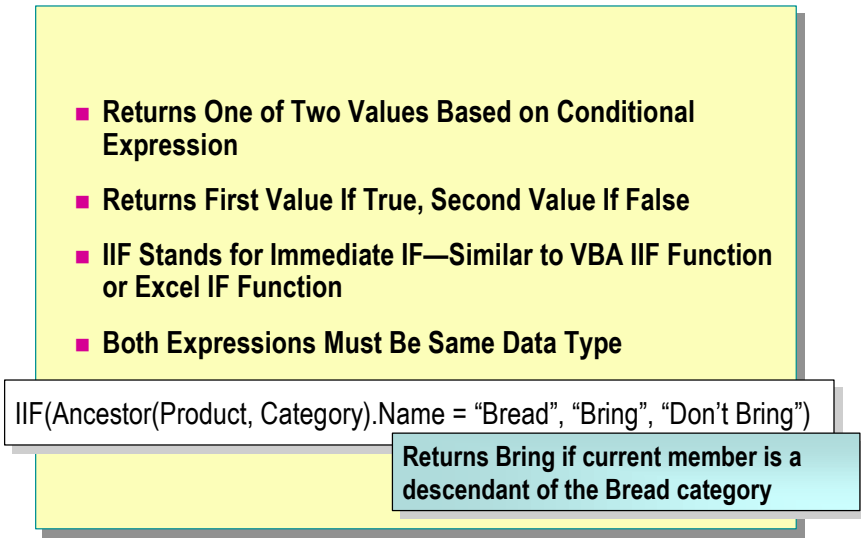
When constructing expressions with multiple conditions, consider the following facts and guidelines:

- In an MDX expression, you can use standard conditional operators, for example:
  - **OR** returns True if either of two conditional tests returns True.
  - **AND** returns True only if both values are True.
  - **XOR** returns True if one and only one value is True.
  - **NOT** reverses the True and False results of the test.
- The result of combining multiple conditional expressions by using a logical operator is a new conditional expression.
- You can nest conditional expressions, using parentheses as needed to clarify the order in which the expressions should be combined.

IIF Function

Topic Objective
To review the **IIF** function.

Lead-in
You often need to make a decision in an expression. The **IIF** function allows you to do that.



Sometimes you need an MDX expression that calculates one value under one condition and a different value under another condition. The **IIF** function allows you to make a decision based on a conditional test. The MDX **IIF** function is similar to the **IF** function in Excel or the **IIF** function in Visual Basic.

Syntax

IIF(«*Conditional Expression*»,«*String Expression*»,«*String Expression*»)

IIF(«*Conditional Expression*»,«*Numeric Expression*»,«*Numeric Expression*»)

Example

The following example displays the text "Bring" or "Don't Bring", depending on the category of the current product.

```
IIF(Ancestor([Product].CurrentMember,
[Product].Category ).Name = "Bread" ,
"Bring" , "Don't Bring" )
```

Delivery Tip
Because students have already seen the **IIF** function in the previous group activity, use this topic as a review—that is, do not spend a lot of time.

When you use the **IIF** function, consider the following facts and guidelines:

- The **IIF** function takes three arguments. The first argument is the conditional expression. The final two arguments are the result values. If the conditional expression returns True, the function returns the first result value. If the conditional expression returns False, the function returns the second result value.
- The name of the function—**IIF**—is an abbreviation for Immediate IF. This function is similar to the Visual Basic **IIF** function and to the Excel **IF** function.
- There are two versions of the **IIF** function. One returns a string; the other returns a number. Both result values must be of the same data type.
  - If the first result value is a string, the second result value must also be a string.
  - Likewise, if the first result value is a number, the second result value must also be a number.

  This is different from the **IF** function in Excel and the **IIF** function in VBA, where the arguments can have different data types.
- The **IIF** function that returns a string value appears in the String group of the **Functions** list. The **IIF** function that returns a number appears in the Numeric group of the **Functions** list.

Note It is not necessary to create expressions with the form **IIF(«condition»,1,0)**. If you want to use the result of a conditional expression as a number in an arithmetic expression, simply use the 1 and 0 values returned by the condition.

Lab C: Using Conditional Expressions

Topic Objective
To introduce the lab.

Lead-in
In this lab, you will create conditional expressions.

Explain the lab objectives.

Objectives

After completing this lab, you will be able to:

- Create conditional expressions using various forms, including the **IIF** function.

Prerequisites

Before working on this lab, you must have successfully completed modules 1 through 2 in course 2093A, *Implementing Business Logic with MDX in Microsoft SQL Server 2000*.

Estimated time to complete this lab: 20 minutes

Exercise 1
Using Conditional Expressions

Delivery Tip
The procedures in this exercise essentially replicate the previous group activities, but without the answers.

Students having difficulty with the procedures should first refer back to the group activity procedures and then go to the answer file for guidance.

In this exercise, you will create and then make modifications to a calculated member in the **Basic Sales** virtual cube on the **Measures** dimension, including assigning the calculated member various expressions that display metadata information from a cube.

As you complete each procedure, compare the result as shown in the procedure screen shot to the result on your own computer. If there is a difference, recheck your entry and refer back to the related group activity procedures as necessary.

If you still cannot reconcile your result, then refer to the answer files, which are located in:

C:\MOC\2093A\Labfiles\L03\Answers

You can copy and paste expressions from this file into the **Value expression** box for any given procedure.

► To test the category of the current member

In this procedure, you will create a calculated member in the **Basic Sales** virtual cube by using a conditional expression along with the **Ancestor** function. The answer file **ConditionAncestor.txt** contains the completed MDX expression used in this procedure.

1. If necessary, delete any calculated members from the **Basic Sales** cube, and then create a new calculated member of the **Measures** dimension named **Test Condition.**
2. Give **Test Condition** an expression that will display 1 for all members that are in the Bread category, and 0 for all other members.

   What MDX expression did you use?

   ```
   Ancestor(Product.CurrentMember,Category).Name = "Bread"
   ```

   _____

3. Browse the cube, navigating to all levels of the **Product** dimension, and then, with the first rows of the **Dairy** category visible, verify that the browser content is similar to the following table.

			MeasuresLevel
- Category	- Subcategory	Product Name	Test Condition
- Bread	- Sliced Bread	Sphinx Rye Bread	1
		Sphinx Wheat Bread	1
		Sphinx White Bread	1
- Dairy	Dairy Total		0
	- Cheese	Cheese Total	0

► To test the category and subcategory of the current member

In this procedure, you will modify a calculated member in the **Basic Sales** virtual cube by using a logical operator to combine two conditional expressions. The answer file **ConditionTwo.txt** contains the completed MDX expression used in this procedure.

1. Edit the **Test Condition** member so that the calculated member displays 1 for members only in either the **Bread** category or the **Milk** subcategory.

   What MDX expression did you use?

   ```
   Ancestor(Product.CurrentMember,Category).Name = "Bread" OR
   Ancestor(Product.CurrentMember,Subcategory).Name = "Milk"
   ```

   _____

   _____

2. Browse the cube, displaying only the **Category** and **Subcategory** levels of the **Product** dimension, and then, with the first few rows of the **Dairy** category visible, verify that the browser content is similar to the following table.

		MeasuresLevel	
- Category	+ Subcategory	Sales Dollars	Test Condition
- Bread	+ Muffins	$13,081.28	1
	+ Sliced Bread	$13,966.86	1
- Dairy	Dairy Total	$32,533.02	0
	+ Cheese	$17,709.11	0
	+ Milk	$7,259.42	1
	+ Sour Cream	$2,746.58	0

► To display different values based on a conditional expression

In this procedure, you will modify a calculated member in the **Basic Sales** virtual cube by using the **IIF** function to display different string values based on a conditional expression. The answer file **ConditionIIF.txt** contains the completed MDX expression used in this procedure.

1. Edit the **Test Condition** member so that the calculated member displays **is bread** for all members that are in the Bread category, and **is not bread** for all other members.

   What MDX expression did you use?

   ```
   IIF(Ancestor(Product.CurrentMember,Category).Name =
   "Bread", "is bread", "is not bread")
   ```

   _____

   _____

2. Browse the cube, displaying only the **Category** and **Subcategory** levels of the **Product** dimension, and then verify that the browser content is similar to the following table.

		MeasuresLevel	
- Category	+ Subcategory	Sales Dollars	Test Condition
All Product	All Product Total	$76,741.28	is not bread
- Bread	Bread Total	$30,600.42	is bread
	+ Bagels	$3,552.28	is bread
	+ Muffins	$13,081.28	is bread
	+ Sliced Bread	$13,966.86	is bread
- Dairy	Dairy Total	$32,533.02	is not bread

► To delete the Test Condition calculated member

- In the **Virtual Cube Editor**, delete the **Test Condition** calculated member.

Review

Topic Objective
To reinforce module objectives by reviewing key points.

Lead-in
The review questions cover some of the key concepts taught in the module.

- Using MDX Expressions
- Displaying Member Information
- Displaying Family Tree Relatives
- Working with Member Properties
- Using Conditional Expressions

1. You want to create an expression that displays empty cells. What value would you put in the expression?

   Null.

2. You want to concatenate two strings in an MDX expression. What operator would you use?

   The plus sign (+).

3. You want to create an expression that displays the name of each member along a dimension. In addition to the **Name** function, what other function must you include?

   The CurrentMember function.

4. If you want to display the name of the grandparent of the current member, what function would you use, and what would you have to specify in that function?

   The Ancestor function, with 2 as the numeric distance argument.

5. You want to use a member property in an arithmetic expression. What must you do?

   Use an external function such as Val to convert the member property to a number.

6. You want to display a number if a condition is true and a string if the condition is false. How would you do this?

   You cannot. The IIF function requires both the Value If True argument and the Value If False argument to have the same data type. You could convert the number to a string.