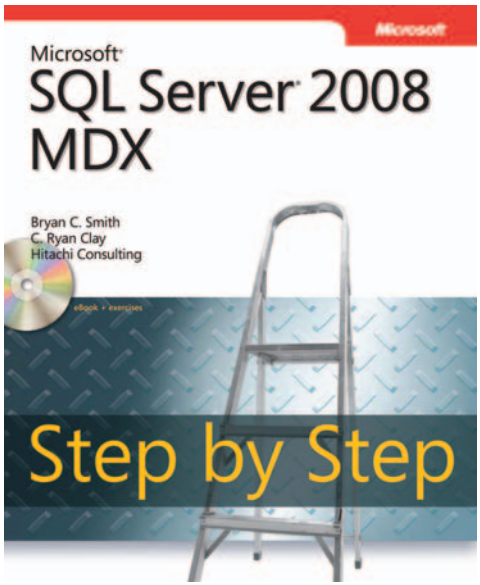


Microsoft® SQL® Server 2008 MDX Step by Step

Bryan C. Smith
C. Ryan Clay
Hitachi Consulting

To learn more about this book, visit Microsoft Learning at
http://www.microsoft.com/learning/en/us/Books/12914.aspx

9780735626188

Microsoft Press

Table of Contents

Chapter 3

Understanding Tuples

After completing this chapter, you will be able to:

- Explain the concept of cube space
- Retrieve data from a cube using tuples
- Reference hierarchy members using a variety of syntax

For the purpose of data access, Analysis Services presents cubes as n-dimensional spaces referred to as cube spaces. Within a *cube space*, data are made accessible through *cells*, each uniquely identified by a *tuple*.

In this chapter, you learn how to assemble tuples to access individual cells. This is foundational to your success with MDX.

N-dimensional Space

To understand the concept of cube space, picture a simple number line. As you may remember from your school days, a number line is a line marked at regular intervals by integer (whole-number) values. Figure 3-1 provides an illustration of such a line.

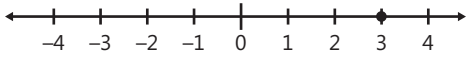


FIGURE 3-1 A number line with a point at (3)

In this illustration, a point resides along the line at the position indicated by the number 3. This number, 3, is the point's *coordinate*. When you wrap the coordinate in parentheses like so

(3)

you have a simple system for expressing the point's position along the line.

Now consider the introduction of another number line perpendicular to the one above. These two lines define a two-dimensional space, as illustrated in Figure 3-2.

Traditionally, the horizontal line in this two-dimensional space is referred to as the x-axis and the vertical line is referred to as the y-axis. Points within this space are identified by their position relative to these two axes. (*Axes* is the plural of axis.)

To express the position of a point, the x-coordinate and y-coordinate of the point is presented in a comma-delimited list. In this list, the x-coordinate precedes the y-coordinate, and the entire list is wrapped in parentheses. This double coordinate system is generically described using the form (x, y).

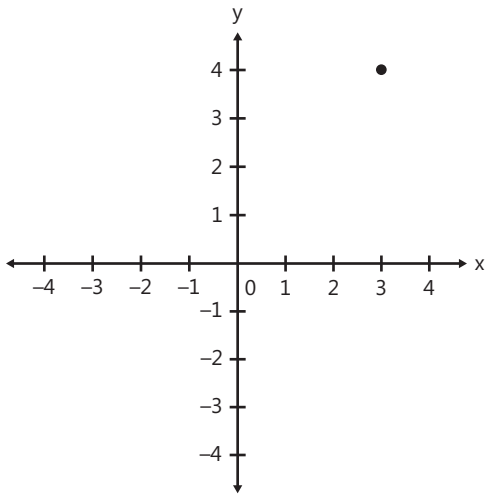


FIGURE 3-2 Two perpendicular number lines with a point at (3, 4)

To illustrate this, consider the point in Figure 3-2. It resides at the intersection of the value 3 along the x-axis and 4 along the y-axis. It is therefore identified using the double coordinate (3, 4).

Taking this one step further, consider the addition of a third line perpendicular to both the x and y axes. Keeping with tradition, the newly introduced third axis is referred to as the z-axis. The space formed by these three axes is illustrated in Figure 3-3. Together with the x and y axes, the z-axis forms a three-dimensional space. Points within this space are represented using a triple-coordinate system, (x, y, z). While challenging to see on paper, a point is presented in Figure 3-3 at the position identified by the triple coordinate (3, 4, 2).

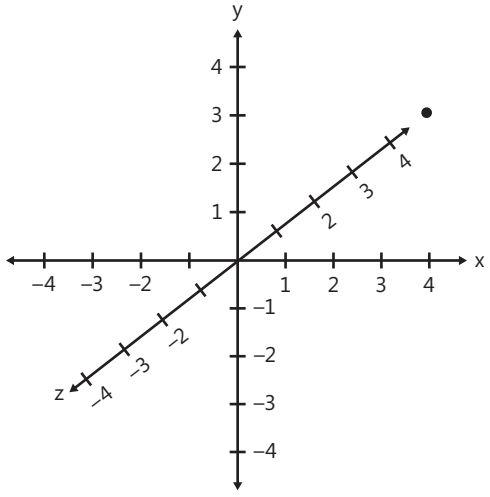


FIGURE 3-3 Three perpendicular number lines with a point at (3, 4, 2)

Now add a fourth axis. The four-dimensional space created can no longer be easily visualized. Still, points within this space can be located using a quadruple-coordinate system.

To describe the form of the quadruple-coordinate system, it's helpful to re-label the axes with the letter *a* and a numerical subscript. Using this approach, the x-axis becomes axis a_1, the y-axis becomes axis a_2, the z-axis becomes axis a_3, and the newly introduced fourth axis becomes axis a_4. Points within this space are then located using a quadruple-coordinate system of the form (a_1, a_2, a_3, a_4).

Adding a fifth axis makes the space even more complex, but points within this space are easily addressed using a quintuple-coordinate system of the form (a_1, a_2, a_3, a_4, a_5). A sixth axis leads to a sextuple-coordinate system (a_1, a_2, a_3, a_4, a_5, a_6); a seventh axis leads to a septuple-coordinate system (a_1, a_2, a_3, a_4, a_5, a_6, a_7); and an eighth axis leads to an octuple-coordinate system (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8).

You could go on like this forever, and while imagining spaces such as these is a bit mind-blowing, locating a point within any of them is a simple matter of employing an appropriately sized coordinate system.

Generically, these spaces are referred to as *n-dimensional spaces*. These spaces have *n* number of axes, and points within them are located using coordinate systems of the form (a_1, a_2,. . .,a_n). These coordinate systems are generically referred to as *tuples*.

Note The question of how to properly pronounce the word *tuple* always seems to come up. Some folks pronounce it with a *u* like the one in *cup*. Others pronounce it like with a *u* like the one in *dude*. We aren't really sure which way is right and use both forms ourselves.

Cube Space

In Analysis Services, a cube is presented as an n-dimensional space referred to as a cube space. Each attribute-hierarchy within the dimensions of the cube forms an axis. Along each axis, each member of the associated attribute-hierarchy, including the (All) member, occupies a position. This translation of an attribute-hierarchy to a cube space axis is illustrated in Figure 3-4 for the Product dimension's Category attribute-hierarchy, first described in Chapter 1, "Welcome to MDX."

Measures are also assigned an axis. Although handled differently during cube design, for the purposes of defining a cube space, a cube's measures are simply members of an attribute-hierarchy called Measures, which belongs to the Measures dimension. One thing that differentiates the Measures attribute-hierarchy from other attribute-hierarchies is that it does not (and cannot) have an (All) member.

With each traditional attribute-hierarchy and the measures of a cube translated into axes, the cube space is defined. Points within the cube space can then be referenced using a tuple. Unlike tuples in the n-dimensional spaces formed by number lines, tuples in cube spaces use member references for coordinate values.

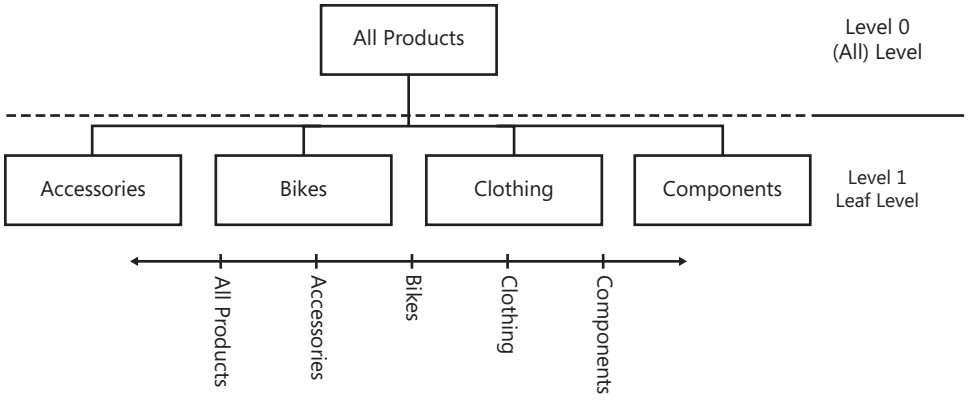


FIGURE 3-4 The representation of the Category attribute-hierarchy as a cube space axis

Basic Member References

You can reference a member within an attribute-hierarchy in a number of ways. The basic member reference identifies the member along with its associated attribute-hierarchy and dimension using the following form:

```
[Dimension].[Hierarchy].[Member].
```

Each of the dimension, attribute-hierarchy, and member object identifiers within the member reference are encapsulated in square brackets. These are separated from each other by periods.

The square brackets around a particular object identifier are optional as long as the object identifier:

1. Is not one of 200+ reserved words identified in SQL Server Books Online
2. Does not start with a character other than a letter or underscore
3. Does not otherwise contain any characters other than letters, numbers, or underscores

Instead of keeping up with all this, you might find it easier to just consistently wrap each identifier in square brackets. This is a standard used throughout this book.

Object names are used as the identifiers for dimensions and attribute-hierarchies. Members are a bit more complex in that they can be identified by either name or key.

A member's name is its user-friendly label. This is what is usually presented in result sets and browsers such as the MDX Query Editor. The following example demonstrates a name-based reference to the member Bikes of the Product dimension's Category attribute-hierarchy:

```
[Product].[Category].[Bikes]
```

Member names suffer one key drawback: They are not guaranteed to be unique within an attribute-hierarchy. This is problematic if more than one member within a hierarchy shares the same name (which is quite common in some dimensional models). Using key-based references resolves this problem.

A member's key is its unique identifier within its associated attribute-hierarchy. Because of its guaranteed uniqueness, a key is the most precise means of identifying a member within an attribute-hierarchy. When identifying a member by key, the identifier is preceded by the ampersand character (&). The previous Bikes reference is demonstrated using its key-based reference:

```
[Product].[Category].&[1]
```

This example illustrates a common issue with key-based references. If you are not aware that the member named Bikes employs a key-value of 1, the key-based reference may be difficult to interpret. This leaves you in the position of using name-based references that may be ambiguous or key-based references that may be difficult to interpret. In this book, we make use of named-based references for interpretability unless a particular concept or ambiguity dictates we use keys. The right choice in your applications depends on the structure of your data.

Accessing Data with Tuples

The MDX Step-by-Step sample database accompanying this book contains a highly simplified cube named Chapter 3 Cube. The cube consists of two dimensions—Product and Date—and a single measure, Reseller Sales Amount. Figure 3-5 presents the structure of this cube.

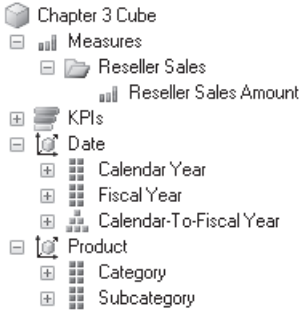


FIGURE 3-5 The structure of the Chapter 3 Cube cube

Within the Product dimension are two attribute-hierarchies, Subcategory and Category. The Date dimension also contains two attribute-hierarchies, Fiscal Year and Calendar Year, which together form the levels of the user-hierarchy Calendar-To-Fiscal Year.

Note The Calendar-To-Fiscal Year user-hierarchy is provided in this cube for no other purpose than to illustrate a few critical concepts while sidestepping a few issues addressed later on. The Calendar-To-Fiscal Year user-hierarchy is not found in the Step-by-Step cube, and such a user-hierarchy combining fiscal year and calendar year attributes is rarely found in the real world. Please consider this hierarchy nothing more than an educational construct.

With four traditional attribute-hierarchies plus the Measures attribute-hierarchy discussed earlier in this chapter, the cube space formed by this cube contains a total of five axes. Points within this cube space are therefore located using a five-part tuple.

For example, the point located at the intersection of the Category member Bikes, the Subcategory member Mountain Bikes, the Calendar Year and Fiscal Year members All Periods, and the Measures member Reseller Sales Amount is identified with the following five-part tuple:

```
(
    [Date].[Calendar Year].[All Periods],
    [Date].[Fiscal Year].[All Periods],
    [Product].[Category].[Bikes],
    [Product].[Subcategory].[Mountain Bikes],
    [Measures].[Measures].[Reseller Sales Amount]
)
```

The use of this tuple to retrieve data is demonstrated in the following exercise.

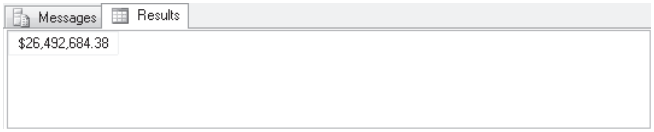
Use a tuple to access a point in a cube space

1. Open the MDX Query Editor to the MDX Step-by-Step database. If you need assistance with this task, refer to Chapter 2, "Using the MDX Query Editor."
2. In the code pane, enter the following query:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Date].[Calendar Year].[All Periods],
    [Date].[Fiscal Year].[All Periods],
    [Product].[Category].[Bikes],
    [Product].[Subcategory].[Mountain Bikes],
    [Measures].[Measures].[Reseller Sales Amount]
    )
```

Note The line breaks and indentions used with this tuple are purely for readability.

3. Execute the query.



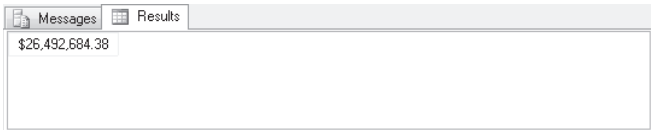
The tuple is employed in the *SELECT* statement to retrieve data from a single point within the cube space formed by the Chapter 3 Cube. Like tuples associated with number lines, this tuple used here consists of a parentheses-enclosed, comma-delimited list of coordinate values. Each of these values consists of a basic member reference identifying a member (by name) and its associated attribute-hierarchy and dimension.

Since an attribute-hierarchy represents an axis in the cube space and a member reference identifies the attribute-hierarchy, the member reference identifies the axis with which it is associated. In other words, member references are self-describing. Therefore, you don't need to rely on the position of a member reference (coordinate value) in the tuple to determine which axis it is associated with. This allows member references to be placed in any order within a tuple without impacting the point identified.

4. Move the *[Product].[Subcategory].[Mountain Bikes]* member reference to the top of the tuple:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Product].[Subcategory].[Mountain Bikes],
    [Date].[Calendar Year].[All Periods],
    [Date].[Fiscal Year].[All Periods],
    [Product].[Category].[Bikes],
    [Measures].[Measures].[Reseller Sales Amount]
    )
```

5. Execute the query and verify the same value as before is returned.



Try moving around other member references within the tuple. Notice that so long as the tuple is properly formed, the same point within the cube space is identified.

Understanding Cells

In the previous exercise, you used a tuple to locate a point within a cube space. On the surface, it appeared that a simple value is recorded at this point, which is what is returned by the *SELECT* statement. The reality is a bit more complex.

Points within cube spaces are occupied by cells. Cells are objects and as such have a number of properties. When cells are accessed, various properties are returned. The default properties returned are *VALUE* and *FORMATTED_VALUE*.

The *VALUE* property contains an aggregated measure value. That value is based on the measure aggregated against all the other attribute-hierarchy members associated with the cell. For example, the *VALUE* property of the cell associated with the previously employed tuple, repeated here for clarity, contains the aggregated value for the Reseller Sales Amount measure limited to the Calendar Year and Fiscal Year attribute-hierarchies' All Periods members, the Category attribute-hierarchy's Bikes member, and the Subcategory attribute-hierarchy's Mountain Bikes member:

```
(
    [Date].[Calendar Year].[All Periods],
    [Date].[Fiscal Year].[All Periods],
    [Product].[Category].[Bikes],
    [Product].[Subcategory].[Mountain Bikes],
    [Measures].[Measures].[Reseller Sales Amount]
)
```

The *FORMATTED_VALUE* property contains the string representation of the *VALUE* property, formatted per instructions associated with the cell at design time. The *FORMATED_VALUE* is what is displayed in the results pane of the MDX Query Editor. A bit more information on assigning formats is provided in Chapter 5, "Working with Expressions."

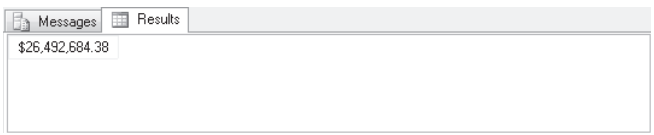
A number of other properties can be returned with a cell. Within a *SELECT* statement, these are accessed using the *CELL PROPERTIES* keyword as demonstrated in the following exercise.

Access cell properties

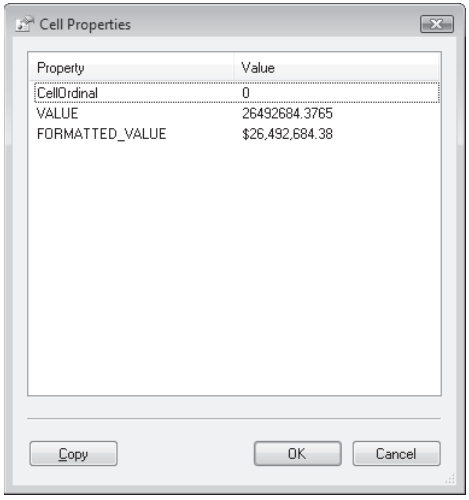
1. If you have not already done so, open the MDX Query Editor to the MDX Step-by-Step database.
2. In the code pane, re-enter the last query from the previous exercise:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Product].[Subcategory].[Mountain Bikes],
    [Date].[Calendar Year].[All Periods],
    [Date].[Fiscal Year].[All Periods],
    [Product].[Category].[Bikes],
    [Measures].[Measures].[Reseller Sales Amount]
    )
```

3. Execute the query to retrieve the results.



4. Double-click the cell returned in the Results pane to open the Cell Properties dialog box.



The default properties *VALUE* and *FORMATTED_VALUE* are returned with the cell. The *CELL_ORDINAL* property, displayed as CellOrdinal, is also returned to indicate the position of the returned cell in the query's cell set. Cell sets are discussed in Chapter 4, "Working with Sets."

You can retrieve additional properties by including the *CELL PROPERTIES* keyword in your query. If you use the *CELL PROPERTIES* keyword, the *VALUE* and *FORMATTED_VALUE* properties are not returned unless explicitly requested. (The *CELL_ORDINAL* property is always returned as it is a property of the retrieved data.)

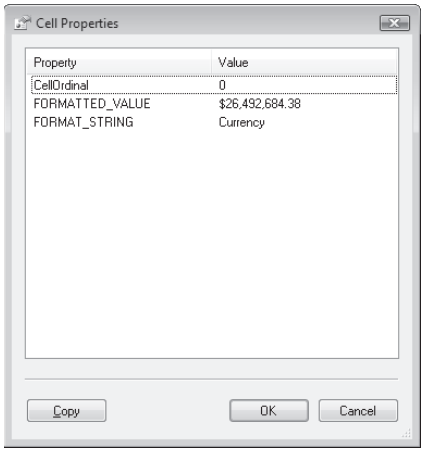
5. Click the OK button in the Cell Properties dialog box to close it.

6. Modify the query to request the *FORMATTED_VALUE* and *FORMAT_STRING* cell properties, purposely omitting the *VALUE* and *CELL_ORDINAL* properties:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Product].[Subcategory].[Mountain Bikes],
    [Date].[Calendar Year].[All Periods],
    [Date].[Fiscal Year].[All Periods],
    [Product].[Category].[Bikes],
    [Measures].[Measures].[Reseller Sales Amount]
    )
CELL PROPERTIES FORMATTED_VALUE, FORMAT_STRING
```

7. Execute the query.

8. Double-click the returned cell to open the Cell Properties dialog box.



Notice that *VALUE* is omitted from the list of cell properties, but the *CELL_ORDINAL* property is returned with the cell.

9. Review the property values and then click OK to close the dialog box.

The complete list of available cell properties and their descriptions is provided in Table 3-1. Additional information on each property is available through SQL Server Books Online.

TABLE 3-1 Available cell properties

Cell Property	Description
ACTION_TYPE	A bitmask indicating the type of action(s) associated with the cell.
BACK_COLOR	A bitmask indicating the background color to use when displaying the *VALUE* or *FORMATTED_VALUE* property of the cell.
CELL_ORDINAL	The ordinal number of the cell in the cell set.
FONT_FLAGS	A bitmask indicating whether the cell's font should be presented using italic, bold, underline, or strikeout detailing.
FONT_NAME	The name of the font to use when displaying the *VALUE* or *FORMATTED_VALUE* property of the cell.
FONT_SIZE	The font size to use when displaying the *VALUE* or *FORMATTED_VALUE* property of the cell.
FORE_COLOR	A bitmask indicating the foreground color to use when displaying the *VALUE* or *FORMATTED_VALUE* property of the cell.
FORMAT	This is the same as the *FORMAT_STRING* property.
FORMAT_STRING	The format string used to create the value of *FORMATTED_VALUE* property of the cell.
FORMATTED_VALUE	The character string representation of the *VALUE* property formatted per the *FORMAT_STRING* value.
LANGUAGE	The locale against which the *FORMAT_STRING* will be applied.
UPDATEABLE	A value indicating whether the cell can be updated.
VALUE	The unformatted value of the cell.

Working with Partial Tuples

The cube used in this chapter has a very simple structure. With only five attribute-hierarchies (including Measures), points within this cube are identifiable using a five-part tuple. Imagine a more typical cube with tens or even hundreds of attributes. Having to specify a member reference for each attribute-hierarchy within the cube to complete a tuple would simply be overwhelming.

Thankfully, Analysis Services allows you to submit partial tuples. Within a partial tuple one or more member references are omitted. Because a complete tuple is required to locate a point in the cube space, Analysis Services takes responsibility for filling in the missing references. This is done by applying the following rules for each missing attribute-hierarchy member reference:

1. If the member reference is omitted, use the attribute's default member.
2. If the member reference is omitted and no default member is specified, use the attribute's (All) member.
3. If the member reference is omitted, no default member is specified, and the (All) member does not exist, use the attribute's first member.

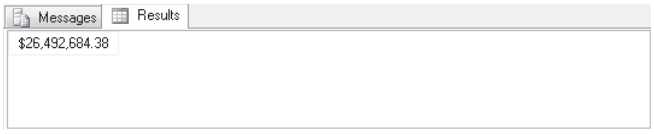
In the following exercise, you put these rules to work.

Access cells in a cube using partial tuples

1. If you have not already done so, open the MDX Query Editor to the MDX Step-by-Step database.
2. In the code pane, enter the following query specifying a complete tuple:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Date].[Calendar Year].[All Periods],
    [Date].[Fiscal Year].[All Periods],
    [Product].[Category].[Bikes],
    [Product].[Subcategory].[Mountain Bikes],
    [Measures].[Measures].[Reseller Sales Amount]
    )
```

3. Execute the query and note the result.

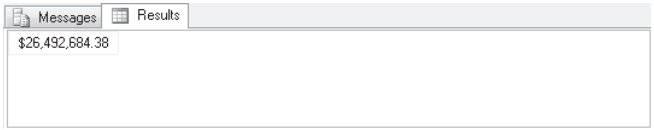


4. Now, specify a partial tuple by removing the Measures member reference:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Date].[Calendar Year].[All Periods],
    [Date].[Fiscal Year].[All Periods],
    [Product].[Category].[Bikes],
    [Product].[Subcategory].[Mountain Bikes]
    )
```

Note Be certain to remove the comma following the Mountain Bikes member reference.

5. Execute the query and compare the result to that of the previous query.



With the Measures member removed, a partial tuple is submitted to Analysis Services. Analysis Services supplies the missing Measures reference by first checking for a default member. The default member of the Measures attribute-hierarchy is Reseller Sales Amount. That member is applied and the tuple is complete. The process by which the tuple is completed is illustrated in Figure 3-6. Because the completed tuple is the same tuple specified in the first query of this exercise, the same cell is accessed.

Position	Partial Tuple	Rule 1: Default Member	Rule 2: (All) Member	Rule 3: First Member	Completed Tuple
Date. Calendar Year	All Periods	→	→	→	All Periods
Date. Fiscal Year	All Periods	→	→	→	All Periods
Product. Category	Bikes	→	→	→	Bikes
Product. Subcategory	Mountain Bikes	→	→	→	Mountain Bikes
Measures. Measures	*(omitted)*	Reseller Sales Amount	→	→	Reseller Sales Amount

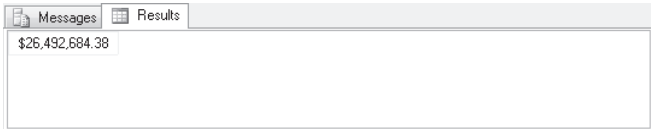
FIGURE 3-6 The process for completing the tuple with a missing Measures member

Note The default member of the Measures attribute-hierarchy is defined at design time when a default measure is assigned to the cube. In this cube, a default measure of Reseller Sales Amount has been assigned. Had this not been explicitly assigned, the third rule would have completed the tuple with Reseller Sales Amount, the first (and only) measure in the cube.

6. Alter the query by removing the two member references associated with the Date dimension:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Product].[Category].[Bikes],
    [Product].[Subcategory].[Mountain Bikes]
    )
```

7. Execute the query and compare the result to that of the previous query.



With this query, Analysis Services supplies the Measures member reference by applying the first rule. For the Date dimension's Calendar Year and Fiscal Year attribute-hierarchies, a default member is not defined so the first rule does not address these omitted references. However, an (All) member, All Periods, is defined for these attribute-hierarchies, so the second rule fills in the blanks. The process by which this partial tuple is completed is illustrated in Figure 3-7. As before, the completed tuple is the same as the tuple in the first query of this exercise so that the same cell as before is accessed.

Position	Partial Tuple	Rule 1: Default Member	Rule 2: (All) Member	Rule 3: First Member	Completed Tuple
Date. Calendar Year	*(omitted)*	*(not available)*	All Periods →		All Periods
Date. Fiscal Year	*(omitted)*	*(not available)*	All Periods →		All Periods
Product. Category	Bikes →				Bikes
Product. Subcategory	Mountain Bikes →				Mountain Bikes
Measures. Measures	*(omitted)*	Reseller Sales Amount →			Reseller Sales Amount

FIGURE 3-7 The process for completing the tuple with missing Measures, Calendar Year, and Fiscal Year members

Now that you understand partial tuples, it should be clear what the basic query introduced in Chapter 2 returns. This query, *SELECT FROM [Step-by-Step]*, returns the cell associated with

the partial tuple within which no member references are supplied. Analysis Services completes each member reference using the three preceding rules and accesses the identified cell.

More Member References

Members in user-hierarchies may also be referenced using the form, *[Dimension].[Hierarchy].[Member],* introduced earlier in this chapter. For example, the calendar year 2003 member of the Calendar-To-Fiscal Year user-hierarchy can be identified as follows:

```
[Date].[Calendar-To-Fiscal Year].[CY 2003]
```

However, because user-hierarchies are assembled from multiple attribute-hierarchies, the member identifier has greater opportunity to be non-unique. This is true not only when member names are employed but also with member keys. To illustrate this, consider the following member reference. Does it reference calendar year 2003 or fiscal year 2003?

```
[Date].[Calendar-To-Fiscal Year].&[2003]
```

This reference is ambiguous. Both the calendar year 2003 and fiscal year 2003 members use the number 2003 as their key. Referencing the member using the form *[Dimension].[Hierarchy].[Level].[Member]* resolves this ambiguity:

```
[Date].[Calendar-To-Fiscal Year].[Calendar Year].&[2003]
```

This new form works with both member keys and member names and is ideal when the member identifier is unique within a specified level but not necessarily unique across the levels of the hierarchy.

Unfortunately, in some situations this new form of member reference is still ambiguous. Consider the Fiscal Year members in Figure 3-8. In particular, pay attention to the two FY 2003 members.

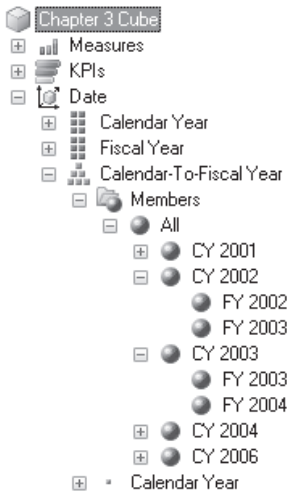


FIGURE 3-8 The relationship between the FY 2003 members and CY 2002 and CY 2003 members

There is one FY 2003 member in the Fiscal Year attribute-hierarchy representing the period July 1, 2002, to June 30, 2003. Since the fiscal year 2003 straddles calendar years 2002 and 2003, two FY 2003 members (one under CY 2002 and the other under CY 2003) are found in the user-hierarchy. Within the user-hierarchy, the FY 2003 member is presented as two distinct members.

In this situation, the only way to differentiate between the two is to identify the Fiscal Year member in relation to its Calendar Year parent. Here are member references identifying these two distinct user-hierarchy members:

```
[Date].[Calendar-To-Fiscal Year].[Calendar Year].[CY 2002].[FY 2003]
[Date].[Calendar-To-Fiscal Year].[Calendar Year].[CY 2003].[FY 2003]
```

Building Tuples with User-Hierarchies

The exercises presented thus far have built tuples exclusively using references to members in attribute-hierarchies. You can also use user-hierarchies to assemble tuples. When a user-hierarchy member reference is employed, Analysis Services translates that reference into one or more attribute-hierarchy member references to assemble a resolvable tuple.

Understanding User-Hierarchy Translation

To translate a user-hierarchy member reference into one or more attribute-hierarchy references, Analysis Services first locates the specified member within the user-hierarchy. With this member located, that member and each member in the levels above it forming the member's lineage in the user-hierarchy is then known. As each level in a user-hierarchy is derived from an attribute-hierarchy, an attribute-hierarchy reference for the specified member and each member in its lineage is then generated. The lone exception to this is the user-hierarchy's (All) member, which does not map to any member in an attribute-hierarchy and is therefore simply ignored in the translation process.

The following exercise demonstrates the process of translating user-hierarchy member references to attribute-hierarchy references.

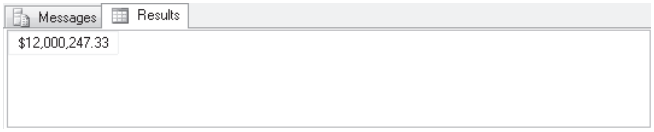
Access cells with tuples containing user-hierarchies

1. If you have not already done so, open the MDX Query Editor to the MDX Step-by-Step database.
2. In the code pane, enter the following query:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Date].[Calendar-To-Fiscal Year].[Calendar Year].[CY 2003].[FY 2003]
    )
```

Note When a tuple is specified using a single member reference, the tuple's parentheses can be omitted. Parentheses are applied to the tuple in the preceding query for the purpose of consistency.

3. Execute the query and note the result.



To resolve this tuple, Analysis Services first locates the FY 2003 member in the Fiscal Year level associated with the CY 2003 member of the Calendar Year level of the Calendar-To-Fiscal Year user-hierarchy. Analysis Services then determines the lineage of this member, which you already know given the explicit structure of the member reference. Each member in the lineage is then translated into an attribute-hierarchy reference and the tuple is completed as illustrated in Figure 3-9.

Position	User-Hierarchy Translation	Partial Tuple	Rule 1: Default Member	Rule 2: (All) Member	Rule 3: First Member	Completed Tuple
Date. Calendar-To-Fiscal Year	Calendar Year. CY 2003. FY 2003					
Date. Calendar Year	CY 2003				→	CY 2003
Date. Fiscal Year	FY 2003				→	FY 2003
Product. Category		*(omitted)*	*(not available)*	All Products	→	All Products
Product. Subcategory		*(omitted)*	*(not available)*	All Products	→	All Products
Measures. Measures		*(omitted)*	Reseller Sales Amount	→		Reseller Sales Amount

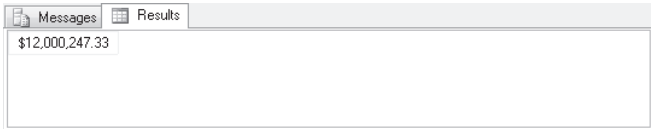
FIGURE 3-9 The process for completing the tuple specifying the FY 2003 member associated with CY 2003 in the Calendar-To-Fiscal Year user-hierarchy

To verify this, you can submit the translated (partial) tuple to see that the same cell is returned.

4. Modify the query to reflect the translated (partial) tuple:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
   [Date].[Calendar Year].[CY 2003],
   [Date].[Fiscal Year].[FY 2003]
   )
```

5. Execute the query and compare the result to that in step 3.

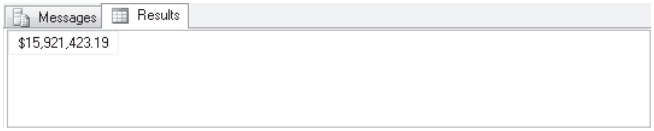


When the lineage for FY 2003 is not specified in the user-hierarchy member reference, the reference becomes ambiguous, as described in the previous sidebar "More Member References". Analysis Services retrieves the first FY 2003 member within the Fiscal Year level of the user-hierarchy it encounters. It then proceeds with the translation process, as previously described.

6. Modify the query to use an ambiguous reference to the FY 2003 member of the Calendar-To-Fiscal Year user-hierarchy:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Date].[Calendar-To-Fiscal Year].[Fiscal Year].[FY 2003]
    )
```

7. Execute the query and note the result.

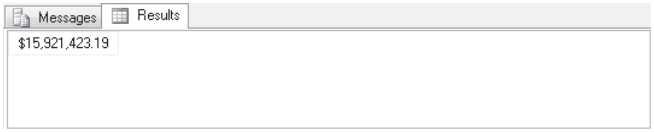


By simply removing the parent member identifier, a different cell is accessed. Analysis Services searches the Fiscal Year level for a member named FY 2003 and the first FY 2003 member encountered just so happens to be the member associated with the CY 2002 member of the Calendar Year level. You can verify this by explicitly requesting this cell and comparing its value to that of the previous query.

8. Modify the query to reflect the translated tuple:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Date].[Calendar Year].[CY 2002],
    [Date].[Fiscal Year].[FY 2003]
    )
```

9. Execute the query and compare its results to those of the previous query.

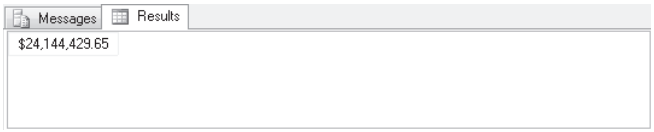


These steps demonstrate the process by which a reference to a leaf-level member in a user-hierarchy is translated into attribute-hierarchy references. You would expect this process to work the same for references to non-leaf members, and it does. When a reference to a non-leaf member in a user-hierarchy is made, the member is identified along with its ancestors, just as before. Descendant members, those related to the specified member in lower levels of the hierarchy are simply ignored for the purposes of translation.

10. Modify the query, specifying a member from the Calendar Year level of the user-hierarchy:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Date].[Calendar-To-Fiscal Year].[Calendar Year].[CY 2002]
    )
```

11. Execute the query.



The CY 2002 member is located within the Calendar Year level of the Calendar-To-Fiscal Year user-hierarchy. This is a non-leaf level. As before, the specified member, CY 2002, is located. That member and the members in its lineage, of which there are none (of any relevance), are translated into attribute-hierarchy references. No Fiscal Year attribute-hierarchy member reference is created, as illustrated in Figure 3-10.

You can verify this by submitting the translated tuple and comparing its results to that of the prior query.

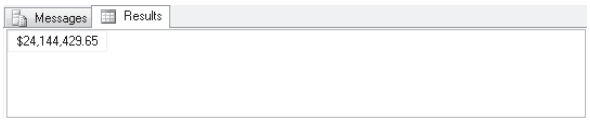
12. Modify the query to reflect the translated tuple:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Date].[Calendar Year].[CY 2002]
    )
```

Position	User-Hierarchy Translation	Partial Tuple	Rule 1: Default Member	Rule 2: (All) Member	Rule 3: First Member	Completed Tuple
Date. Calendar-To-Fiscal Year	Calendar Year. CY 2002					
Date. Calendar Year	CY 2002	→	→	→	→	CY 2002
Date. Fiscal Year		*(omitted)*	*(not available)*	All Periods	→	All Periods
Product. Category		*(omitted)*	*(not available)*	All Products	→	All Products
Product. Subcategory		*(omitted)*	*(not available)*	All Products	→	All Products
Measures. Measures		*(omitted)*	Reseller Sales Amount	→	→	Reseller Sales Amount

FIGURE 3-10 The process for completing the partial tuple specifying the member CY 2002 within the Calendar-To-Fiscal Year user-hierarchy

13. Execute the query and compare the result to those in step 11.



Avoiding Reference Conflicts

As has been mentioned, user-hierarchies are assembled from attribute-hierarchies. The translation process described in this chapter deconstructs a user-hierarchy member reference into its associated attribute-hierarchy member references. But, what if a tuple already contains a reference to one of the attribute-hierarchies from which the user-hierarchy is derived? This creates an opportunity for the translation to generate conflicting attribute-hierarchy references.

Access cells with tuples containing overlapping references

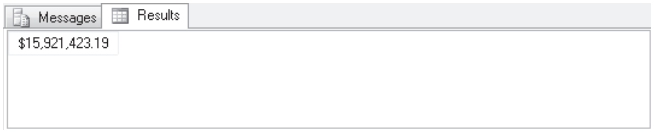
1. If you have not already done so, open the MDX Query Editor to the MDX Step-by-Step database.

2. In the code pane, enter the following query to employ references to both the Calendar-To-Fiscal Year user-hierarchy and Fiscal Year attribute-hierarchy:

```
SELECT
FROM [Chapter 3 Cube]
```

```
WHERE (
    [Date].[Calendar-To-Fiscal Year].[Calendar Year].[CY 2002],
    [Date].[Fiscal Year].[FY 2003]
    )
```

3. Execute the query.



The process of translation and tuple completion is illustrated in Figure 3-11.

Position	User-Hierarchy Translation	Partial Tuple	Rule 1: Default Member	Rule 2: (All) Member	Rule 3: First Member	Completed Tuple
Date. Calendar-To-Fiscal Year	Calendar Year. CY 2002					
Date. Calendar Year	CY 2002	→	→	→	→	CY 2002
Date. Fiscal Year		FY 2003	→	→	→	FY 2003
Product. Category		*(omitted)*	*(not available)*	All Products	→	All Products
Product. Subcategory		*(omitted)*	*(not available)*	All Products	→	All Products
Measures. Measures		*(omitted)*	Reseller Sales Amount	→	→	Reseller Sales Amount

FIGURE 3-11 The process for completing the partial tuple specifying the member CY 2002 within the Calendar-To-Fiscal Year user-hierarchy and FY 2003 within the Fiscal Year attribute-hierarchy

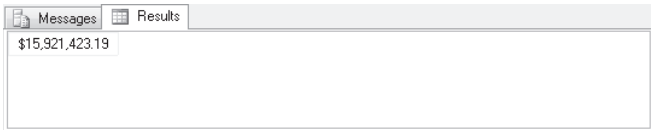
Although the tuple is syntactically valid, the combination of references to an attribute-hierarchy and a user-hierarchy based on that same attribute-hierarchy creates an opportunity for overlapping references following translation. In the previous query, this was avoided. The same is not true in the next query.

4. Modify the query to create an overlapping reference to FY 2003:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Date].[Calendar-To-Fiscal Year].[Calendar Year].[CY 2002].[FY 2003],
    [Date].[Fiscal Year].[FY 2003]
    )
```

5. Execute the query.

The translation process is illustrated in Figure 3-12.



Position	User-Hierarchy Translation	Partial Tuple	Rule 1: Default Member	Rule 2: (All) Member	Rule 3: First Member	Completed Tuple
Date. Calendar-To-Fiscal Year	Calendar Year. CY 2002. FY 2003					
Date. Calendar Year	CY 2002	→				CY 2002
Date. Fiscal Year	FY 2003	→ FY 2003	→			FY 2003
Product. Category		*(omitted)*	*(not available)*	All Products	→	All Products
Product. Subcategory		*(omitted)*	*(not available)*	All Products	→	All Products
Measures. Measures		*(omitted)*	Reseller Sales Amount	→		Reseller Sales Amount

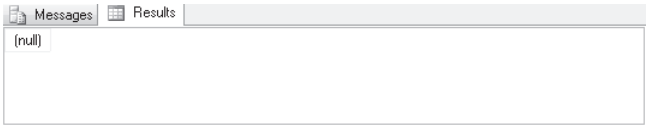
FIGURE 3-12 The process for completing the partial tuple specifying overlapping references to the FY 2003 member

Here, the user-hierarchy member reference is translated to Calendar Year and Fiscal Year attribute-hierarchy references. The tuple already employs a Fiscal Year attribute-hierarchy reference creating overlap. The overlap has a happy ending since the two Fiscal Year attribute-hierarchy member references are identical. Had this not been the case, the overlap would have created a conflict, resulting in an invalid tuple.

6. Modify the query to create an overlapping reference with conflicting Fiscal Year members:

```
SELECT
FROM [Chapter 3 Cube]
WHERE (
    [Date].[Calendar-To-Fiscal Year].[Calendar Year].[CY 2002].[FY 2003],
    [Date].[Fiscal Year].[FY 2002]
    )
```

7. Execute the query.



In this query, the user-hierarchy member reference is translated into Calendar Year and Fiscal Year attribute-hierarchy references. As shown in Figure 3-13, the FY 2003 member reference created through this process conflicts with the FY 2002 attribute-hierarchy member reference. The conflict in member references results in an invalid reference to the Fiscal Year attribute-hierarchy, which results in an empty cell being returned.

Position	User-Hierarchy Translation	Partial Tuple	Rule 1: Default Member	Rule 2: (All) Member	Rule 3: First Member	Completed Tuple
Date. Calendar-To-Fiscal Year	Calendar Year. CY 2002. FY 2003					
Date. Calendar Year	CY 2002					CY 2002
Date. Fiscal Year	FY 2003	FY 2002				*(invalid)*
Product. Category			*(omitted)*	*(not available)*	All Products	All Products
Product. Subcategory			*(omitted)*	*(not available)*	All Products	All Products
Measures. Measures			*(omitted)*	Reseller Sales Amount		Reseller Sales Amount

FIGURE 3-13 The process for completing the partial tuple specifying conflicting overlapping members from the Calendar-To-Fiscal Year user-hierarchy and Fiscal Year attribute-hierarchy

For the reason demonstrated here, it is recommended you consider the possibility of overlap when employing references to user-hierarchies in combination with references to the attribute-hierarchies from which they are derived.

Note Analysis Services enforces a rule that a hierarchy can be referenced no more than once in a given tuple. The process of translation as demonstrated in the last two queries can result in redundant (overlapping) member references, which violates this rule without triggering an error. When working with combinations of attribute and user-hierarchies from a given dimension, be certain to understand which attribute-hierarchies are ultimately being referenced, and employ member references in a way that minimizes the potential for overlapping member references.

Member Reference Shortcuts

The last two sidebars introduced you to three forms of member reference. These forms provide greater and greater degrees of precision to address various forms of ambiguity.

However, not all member references are ambiguous. Many members are unique, whether by name or key, across all hierarchies in a dimension. Still others are unique across all hierarchies in all dimensions. In these situations, omitting the dimension or hierarchy identifier in a member reference still allows the specified member to be found without ambiguity.

Although not encouraged, Analysis Services allows you to take these shortcuts in member reference syntax. These shortcuts can include the omission of dimensions and hierarchy identifiers, allowing tuples to be expressed using a more compact format. For example, the first tuple presented in this chapter can be expressed using the shortened form:

```
(
    [Calendar Year].[All Periods],
    [Fiscal Year].[All Periods],
    [Bikes],
    [Subcategory].[Mountain Bikes],
    [Reseller Sales Amount]
)
```

Although this makes the tuple more compact (and therefore reduces the amount of typing you must do), consider some important pitfalls. First, the shortened syntax is less immediately interpretable and may be harder to support in the long run. Second, unless directed to a specific object, Analysis Services searches the various objects within the cube for matches; this results in noticeable performance overhead. Finally, and most important, Analysis Services discontinues its search as soon as a match is found. If you misjudge the ambiguity of the reference, the result of the query may not be what is expected. For this reason, we encourage you to always employ reasonably precise references supplying at a minimum the dimension and hierarchy identifiers along with the member's key or name.

Having said that, there is one shortcut we employ throughout the remainder of this book. Apart from the previous examples in this chapter, you rarely see a measure identified using its fully qualified form. Instead, measures are almost always identified using the simplified form: *[Measures].[Member]*. Although we refer to the Reseller Sales Amount measure as *[Measures].[Measures].[Reseller Sales Amount]* earlier in this chapter to demonstrate a point about measures as members, we now refer to this measure as *[Measures].[Reseller Sales Amount]* (and all other measures with the same form).

Chapter 3 Quick Reference

To	Do this
Reference a member by name	Write the member reference in the form *[Dimension].[Hierarchy].[Member Name]*. For example: `[Product].[Category].[Bikes]`
Reference a member by key	Write the member reference in the form *[Dimension].[Hierarchy].&[Member Key]*. For example: `[Product].[Category].&[1]`

Reference a member by name within a level of a user-hierarchy	Write the member reference in the form *[Dimension].[Hierarchy].[Level].[Member Name]*. For example:
	`[Date].[Calendar-To-Fiscal Year].[Calendar Year].[CY 2003]`
	In some instances this member reference is ambiguous. To avoid ambiguity, you may use a member reference that includes lineage information, such as this:
	`[Date].[Calendar-To-Fiscal Year].[Calendar Year].[CY 2003].[FY 2003]`

Reference a cell using a tuple	Write a parentheses-enclosed, comma-delimited list of member references. For example:
	`(`
	`  [Date].[Calendar Year].[All Periods],`
	`  [Product].[Category].[Bikes],`
	`  [Product].[Subcategory].[Mountain Bikes]`
	`)`
	Keep in mind user-hierarchy member references will be translated into attribute-hierarchy member references and any missing attribute-hierarchy member references will be supplied by Analysis Services.

Retrieve cell properties as part of the query result set	Include the *CELL PROPERTIES* keyword in the MDX *SELECT* statement, indicating the desired cell properties. For example:
	`SELECT`
	`FROM [Chapter 3 Cube]`
	`WHERE (`
	`  [Product].[Subcategory].[Mountain Bikes],`
	`  [Date].[Calendar Year].[All Periods],`
	`  [Date].[Fiscal Year].[All Periods],`
	`  [Product].[Category].[Bikes],`
	`  [Measures].[Measures].[Reseller Sales Amount]`
	`  )`
	`CELL PROPERTIES FORMATTED_VALUE, FORMAT_STRING`
	Otherwise, do not specify the *CELL PROPERTIES* keyword to return the default properties *VALUE* and *FORMATTED_VALUE*.