**How MDX queries data**

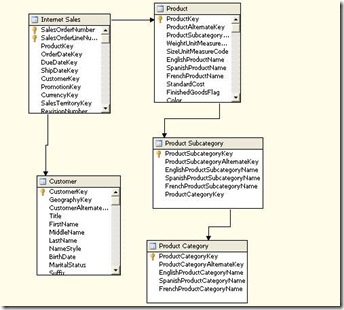
To identify data, MDX uses reference system. This reference system is based on tuples. A tuple can identify a cell uniquely. Like we reference a cell in a spreadsheet by giving the column followed by the row number, we can also reference the tuple. The similarity ends here as there can be multiple dimensions in a cube as against to only rows and columns in a spreadsheet. A tuple can also be referenced by considering a default member. This default member is [All] member for every hierarchy in a cube.

We will start by creating a cube on adventureworksDW database.

1. Create Analysis Services project and name it MDXDemo

2. Add a new data source in it, which directs to Adventure Works DW database namely ‘AdventureWorksDW’

3. Add a new Data Source View (named Adventure Works DW) which consists of tables DimProduct, DimProducSubCategory, DimProducCaetgory, DimCustomer and FactInternetSales. Change friendly names for tables to as shown in figure:

[](http://lh3.ggpht.com/_toA00pxrCRM/SqYZyncekLI/AAAAAAAAAFY/sRMUYSZaeHY/s1600-h/clip_image0024.jpg)

4. Create a cube with default settings but selective measures as Order Quantity, Unit Price, Sales Amount and Internet Sales Count. Give cube name as DemoCube.

5. Go to the Properties of the project (MDXDemo) > select Deployment tab and enter server name. (if you are working with default instance, you can keep the name provided by default i.e. localhost)

6. After successful deployment, open SSMS and connect to Analysis Services to work with MDX queries.

7. Create a new query, connect to MDXDemo database

Enter following MDX query:

SELECT FROM DemoCube

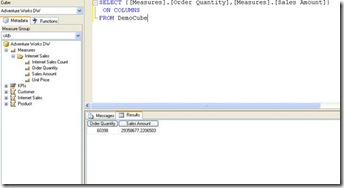
In this case default member for Order Quantity is displayed as 60938

Enter the query as

SELECT [Measures].[Order Quantity] ON 0  
FROM DemoCube

will also give the same result. (0 can also be replaced by COLUMNS) which is the axis.

8. Let us add one more measure as follows:

[](http://lh6.ggpht.com/_toA00pxrCRM/SqYZ194YopI/AAAAAAAAAFg/dipQ4v-XMAI/s1600-h/clip_image0044.jpg)  
  
In this case we see two measures Order Quantity and Sales Amount on Columns

Let us change the default measure. Go to Cube Structure tab in MDXDemo project in BIDS (Business Intelligence Development Studio). Right click on Sales Amount and click Move Up till it becomes first measure. Redeploy the cube and again enter the query

SELECT FROM DemoCube

Now the value shown is different. Let us format this to show only 2 digits after decimal. Enter the FormatString as #,###,##.## and see the difference now.

The default measure can also be changed by MDX as follows

ALTER CUBE DemoCube  
UPDATE DIMENSION  
Measures, DEFAULT\_MEMBER=[Measures].[Order Quantity]

In this case we are again changing the default measure back to Order Quantity

9. Go to DSV (Data Source View), right click on Customers and add a named calculation with name as FullName and expression as FirstName + ‘ ‘ + LastName. Double click on Customers dimensions and specify FullName as NameColumn (so as to display name and not number).

10. Specify similarly EnglishProductName for Product key, EnglishProductCategoryName for product category key and EnglishProductSubcategoryName for product subcategory key in Products dimension.

11. Let us give query for a specific product category Bikes as follows

SELECT [Product].[Product Category Key].[Bikes]  
ON COLUMNS  
FROM DemoCube  
OR  
SELECT [Product].[Product Category Key].&[1]  
ON COLUMNS  
FROM DemoCube

12. Let us add rows now (add another axis)

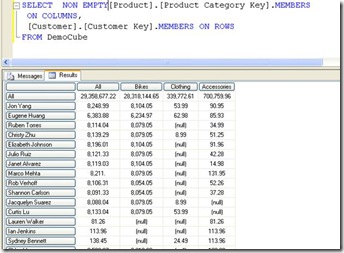
SELECT [Product].[Product Category Key].&[1]  
ON COLUMNS,  
Customer].[Customer Key].MEMBERS ON ROWS  
FROM DemoCube

With this we get Sales Amount for Bikes for all different customers  
  
This can also be achieved by using ordinals as 0 and 1 instead or COLUMNS and ROWS

13.

SELECT NON EMPTY [Product].[Product Category Key].MEMBERS  
ON COLUMNS,  
[Customer].[Customer Key].MEMBERS ON ROWS  
FROM DemoCube

to get all categories (we added NON EMPTY to get rid of null values)  
The result is as follows:

[](http://lh3.ggpht.com/_toA00pxrCRM/SqYZ5thybVI/AAAAAAAAAFo/SvR3S-KggYM/s1600-h/clip_image0064.jpg)

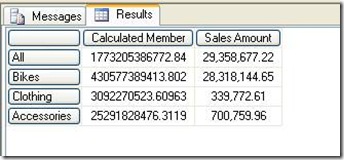
14. Try following query to see Order Quantity for various categories

SELECT [Measures].[Order Quantity]  
ON COLUMNS,  
[Product].[Product Category Key].MEMBERS ON ROWS  
FROM DemoCube

15. Let us see how to add a calculated member

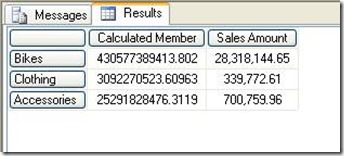
WITH MEMBER Measures.[Calculated Member] AS  
([Measures].[Order Quantity] \* [Measures].[Unit Price])  
SELECT NON EMPTY {Measures.[Calculated Member],  
[Measures].[Sales Amount]} ON COLUMNS,  
NON EMPTY [Product].[Product Category Key].MEMBERS on ROWS  
FROM DemoCube

First we calculate the member and use it with select to view the result

[](http://lh3.ggpht.com/_toA00pxrCRM/SqYZ8sfGKNI/AAAAAAAAAFw/E2qyxLSWUhk/s1600-h/clip_image0084.jpg)

16. If we replace Members after Produt Category Key with CHILDREN we get the result as follows

SELECT NON EMPTY {Measures.[Calculated Member],  
[Measures].[Sales Amount]} ON COLUMNS,  
NON EMPTY [Product].[Product Category Key].CHILDREN on ROWS  
FROM DemoCube

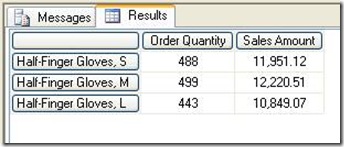
[](http://lh4.ggpht.com/_toA00pxrCRM/SqYZ_ghzIHI/AAAAAAAAAF4/8ybvyUnF7mc/s1600-h/clip_image0104.jpg)

We get only the (non empty) children for categories, all categories are not displayed.

17. Let us see how to display results in order  
  
For this we will first create of products as follows : Within category, Product sub category and within it product name.  
  
Select Product Dimension, drag and drop Product Category key on hierarchies, drag sub category within it and product key within it, name this hierarchy as Product-Category. Deploy the changes to analysis server.  
  
Let us give query without order

select NON EMPTY  
{[Measures].[Order Quantity],[Measures].[Sales Amount]} ON 0,  
NON EMPTY DESCENDANTS  
([Product].[Product-Category].[Product Subcategory Key].[gloves],  
,after)  
on 1 from democube

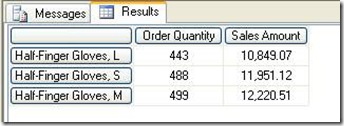
The result looks as follows

[](http://lh5.ggpht.com/_toA00pxrCRM/SqYaCUrJNrI/AAAAAAAAAGA/TltDxe02ghY/s1600-h/clip_image0124.jpg)

With descendants function we can use SELF\_AND\_AFTER clause instead of AFTER clause to get the current descendants member including the specified member  
Now let us give query with Order function:

select NON EMPTY  
{[Measures].[Order Quantity],[Measures].[Sales Amount]} ON 0,  
NON EMPTY ORDER(  
DESCENDANTS  
([Product].[Product-Category].[Product Subcategory Key].[gloves],  
,after),  
[Measures].[Sales Amount]  
,asc) on 1  
from democube

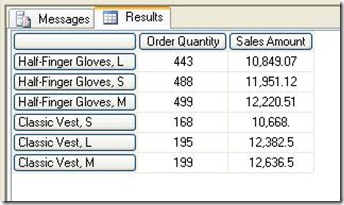
The result will be orderd on Sales Amout as follows

[](http://lh4.ggpht.com/_toA00pxrCRM/SqYaFL114BI/AAAAAAAAAGI/dXbeM5CaKko/s1600-h/clip_image0144.jpg)

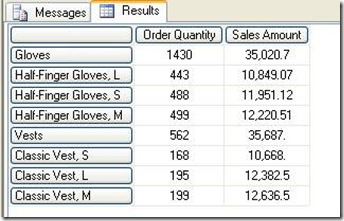
18. Let us add one more sub category and increase a bit of complexity

select NON EMPTY  
{[Measures].[Order Quantity],[Measures].[Sales Amount]}  
ON 0,  
NON EMPTY ORDER(  
{DESCENDANTS  
([Product].[Product-Category].[Product Subcategory Key].[gloves],  
,after),  
DESCENDANTS  
([Product].[Product-Category].[Product Subcategory Key].[vests],  
,after)},  
[Measures].[Sales Amount]  
,asc) on 1  
from democube

In this case we see the result ordered within a particular sub category, sales amount wise.

[](http://lh4.ggpht.com/_toA00pxrCRM/SqYaIJ15IQI/AAAAAAAAAGQ/maV3G8Xvtis/s1600-h/clip_image0164.jpg)

The result of SELF\_AND\_AFTER clause is as follows:

[](http://lh6.ggpht.com/_toA00pxrCRM/SqYaPiohd_I/AAAAAAAAAGY/oUxtTseksa4/s1600-h/clip_image0184.jpg)  
  
In this we see the ‘gloves’ and’ vests’ also included in the result because of the changed clause in descendants function.

In next article we will talk about some advanced functions from MDX.