**Introduction to Axis in MDX Query**

MDX queries can have 0, 1, 2 or up to 129 query axes in the SELECT statement. Each axis behaves in exactly the same way, unlike SQL where there are significant differences between how the rows and the columns of a query behave.

Refer to the following table for Axis Numbers reserved and Alias given to them:

|  |
| --- |
|  |

|  |  |
| --- | --- |
| **Axis Number** | **Alias** |
| **0** | Columns |
| **1** | Rows |
| **2** | Pages |
| **3** | Section |
| **4** | Chapter |

Using SQL Server Management Studio (SSMS), we can only browse values on two axis, **Columns**(Axis **0**) and **Rows**(Axis **1**).

Getting Started With MDX

**1.** **Start with Simple MDX Query**

**Syntax:**

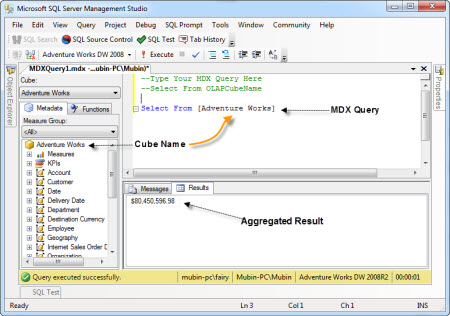
**Select From [Your Cube Name] ;**

Which will give you aggregated result as shown in result pane, MDX is not Case Sensitive except member keys defined within dimension. This query will use default member defined in all the dimensions and use default measure defined by OLAP cube designer.

You can do drag and drop of cube name, dimension members from left pane to query window instead of typing. This query is also known as no axis query.

Hide   Copy Code

Select From [Adventure Works];



2. Dropping Dimensions on Axis

If we will not specify Axis for dimensions and measures, it may lead us to wrong result while design change take place.

**Example:**

Retrieve all customer names on Columns from Adventure Works Cube.

**Syntax**

**Select Dimension.Member on Column From [OLAPCubeName ]  
or  
Select Dimension.Member on 0 From [OLAPCubeName ]**

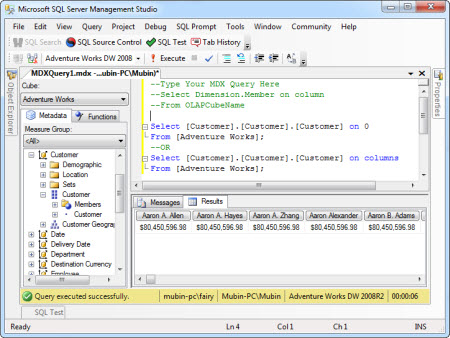
Hide   Copy Code

Select [Customer].[Customer].[Customer] on 0

From [Adventure Works];

Select [Customer].[Customer].[Customer] on columns

From [Adventure Works];



As you can notice one thing here if your dimension is not associated with Measure Group, you can have same values in each result cell against every customer.

But here we are trying to learn how we can bring Customer values on Columns, so we are not focusing on Measures right now. Let us proceed with next.

3. Using Both the Axis (Rows & Columns)

You can select Dimension or Measure on any Axis.

**Example:**

Retrieve Internet Sales Amount As Per Customer. In other words, we can say show the Detail of amount spent by customers during purchase from Internet.

**Syntax**

**Select [Measure] on Columns,  
[Dimension].[Members] on Rows From [Cube Name] ;**

***OR***

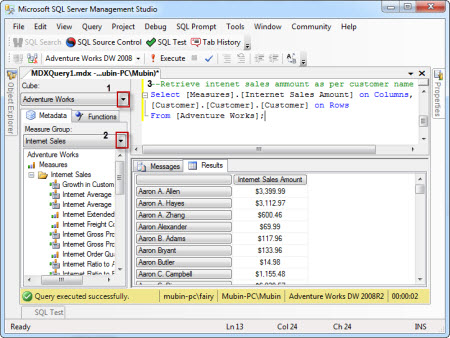
**Select [Measure] on Rows,  
[Dimension].[Members] on Columns From [Cube Name] ;**

Hide   Copy Code

Select [Measures].[Internet Sales Amount] on Columns,

[Customer].[Customer].[Customer] on Rows

From [Adventure Works];



Here, you can see Measure Value (Internet Sales Amount) is properly getting divided as per the customer.

**Note**: You can also do drag & drop of Measures and Dimension members from left vertical Pane marked with number 2 to Query Designer portion number 3.

4. Introduction to .members, and .children in MDX Query

**.Members**

If you will use this with hierarchy level, then it will retrieve all the values below it and also bring agreegation of that in the form of [ALL].

**Syntax**

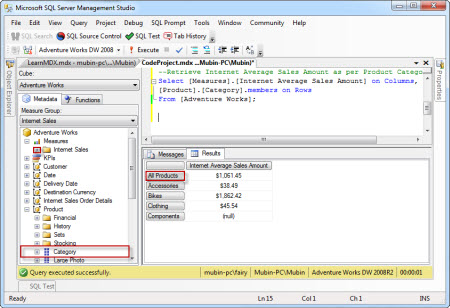
**Select [Dimension].[Hierarchy].members on Columns from CubeName  
or   
Select [Dimension].[Hierarchy].[Level].members on Columns from CubeName**

Hide   Copy Code

Select [Measures].[Internet Average Sales Amount] on Columns,

[Product].[Category].members on Rows

From [Adventure Works];



**.Children**

When we want to retrieve all members values under particular level of a dimension at that time we use **.children**,This will exclude aggregation values [ALL] in your result set.

**Syntax**

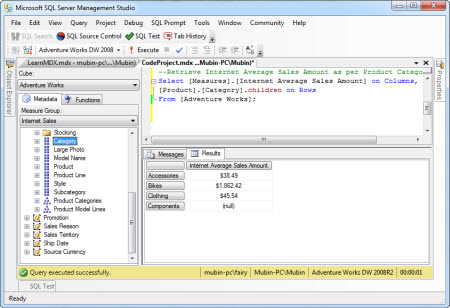
**Select [Dimension].[Hierarchy].[Level].children on Columns from CubeName**

Hide   Copy Code

Select [Measures].[Internet Average Sales Amount] on Columns,

[Product].[Category].children on Rows

From [Adventure Works];



5. Introduction to Tuple and Set

**Tuple**:

When we need to place more than one members of a dimension or hierarchy of that dimension on a axis at that time tuple comes into the picture, tuple is enclosed within curly bracket { }, for single tuple bracket is optional.

We can say Tuple is used to identify particular location in the cube using your dimension members. Tuple will define slice of your cube. Tuple can contain one or more members, but it cannot have members from the same dimension.

This is example of tuples from same date dimension members. Combination of more than one tuple will make a set.

**Example:**

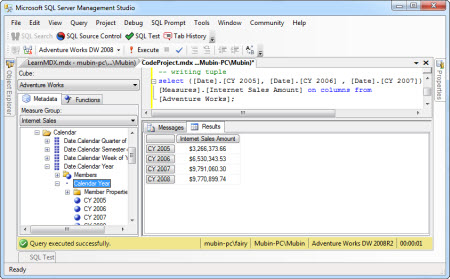
View Internet Sales amount detail between year 2005 to 2007 using tuples.

Hide   Copy Code

select {[Date].[CY 2005], [Date].[CY 2006] , [Date].[CY 2007]} on rows,

[Measures].[Internet Sales Amount] on columns from

[Adventure Works];



**Set:**

A set is an ordered collection of zero, one or more tuples. A set is most commonly used to define axis and slicer dimensions in an MDX query.

Combination of tuple or tuples will give you set , When You want to include range at that time you can use **:**instead of separating tuple members by comma if they are belonging to same dimension member.

**Or**

**Syntax**

**{[Date].[CY 2008] : [Date].[CY 2005]} or {[Date].[CY 2005], [Date].[CY 2006] , [Date].[CY 2007]}   
or  
( [Date].[Calendar Year].members , [Product].[Product].members )**

**Example:**

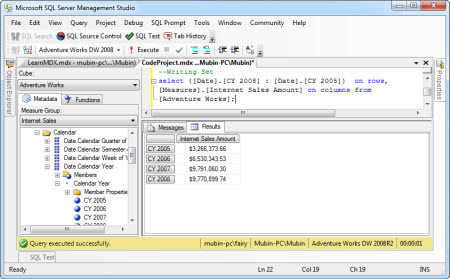
View Internet Sales amount detail between year 2005 to 2008

Hide   Copy Code

select {[Date].[CY 2008] : [Date].[CY 2005]} on rows,

[Measures].[Internet Sales Amount] on columns from

[Adventure Works];



To use combination of tuples from various dimensions, we have to use Cross Join that we will learn soon.

6. Using CROSS JOIN

Cross Join Function returns cross product of one or more sets.

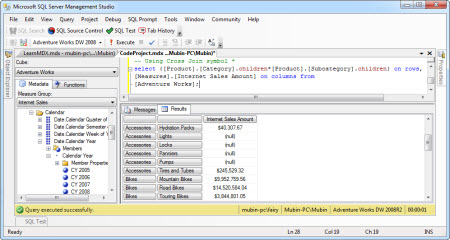
Whenever we need to combine more than one member from same or different dimension at that time we can use cross join. **\*** sign can be use to implement cross join between dimension members.

Hide   Copy Code

select {[Product].[Category].children **\*** [Product].[Subcategory].children} on rows,

[Measures].[Internet Sales Amount] on columns from

[Adventure Works];



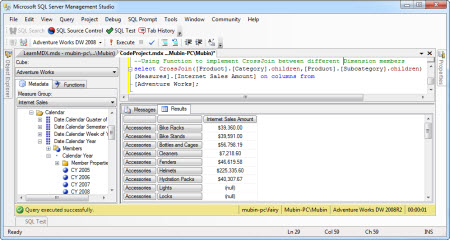
We can also use Cross Join Function to implement cross join between different dimension members, but result will stay same if you use**\*** or CrossJoin Function.

Hide   Copy Code

select **CrossJoin**([Product].[Category].children,[Product].[Subcategory].children) on rows,

[Measures].[Internet Sales Amount] on columns from

[Adventure Works];



7. Using Non Empty or NonEmpty

To element Null values from the result set, we can use NonEmpty() or Non Empty. Right now, I am not discussing difference between Non Empty and NonEmpty function.

NonEmpty function evaluated first so it will remove rows if there was no data in first measure. Let us take a look at the below example how we can remove null values from result set.

Non Empty or NonEmpty() function can be used on any Axis.

**Example**

Let us take a look on Cross join applied in below example, here you can see how we are retrieving multiple measures by placing them between curly bracket{ } .

You can see null values in the result set while using following MDX Query.

Hide   Copy Code

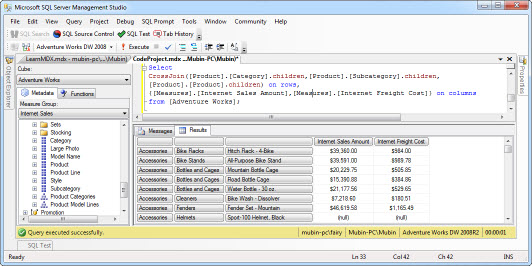
Select

**CrossJoin**([Product].[Category].children,[Product].[Subcategory].children,

[Product].[Product].children) on rows,

{[Measures].[Internet Sales Amount],[Measures].[Internet Freight Cost]} on columns

from [Adventure Works];



Now to eliminate these Null Values from Result Set using Non Empty.

Hide   Copy Code

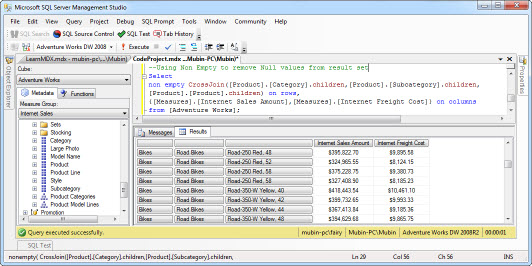
Select

**non empty** CrossJoin([Product].[Category].children,[Product].[Subcategory].children,

[Product].[Product].children) on rows,

{[Measures].[Internet Sales Amount],[Measures].[Internet Freight Cost]} on columns

from [Adventure Works];



8. Apply Slicing using Where Clause

To Slice Data from cube we can use **Where**clause, it is similar to “where” clause we have in T-SQL.

**Example**

I want to see detail of Internet Sales Amount for each product in the Year 2007.

Hide   Copy Code

select [Measures].[Internet Sales Amount] on columns,

[Product].[Product].[Product].members on rows

from [Adventure Works]

where [Date].[Calendar Year].[CY 2007];

**Example**

If I want to see detail of Internet Sales Amount for each product in the Year 2007 and 2009.

Hide   Copy Code

select [Measures].[Internet Sales Amount] on columns,

[Product].[Product].[Product].members on rows

from [Adventure Works]

where {[Date].[Calendar Year].[CY 2007],[Date].[Calendar Year].[CY 2009]};

9. Apply Filtering on data using Filter function

Filter function will also be used to apply filtering on members available in specified set as per the specified Boolean condition.

**Syntax:**

**Filter( <Set>, Boolean Condition)**

Hide   Copy Code

**select [Measures].[Internet Sales Amount] on columns,**

**filter([Product].[Product].[Product].members ,**

**[Measures].[Internet Sales Amount]>5000**

**)**

**on rows**

**from [Adventure Works]**

**where {[Date].[Calendar Year].[CY 2007]};**

**Example:** If I want to retrieve only those products whose names begin with “A” and Internet sales amount <5000.

Hide   Copy Code

**select [Measures].[Internet Sales Amount] on columns,**

**filter([Product].[Product].[Product].members ,**

**([Measures].[Internet Sales Amount]<19000 and \_**

**left([Product].[Product].currentmember.name,1)="A")**

**)**

**on rows**

**from [Adventure Works]**

**where {[Date].[Calendar Year].[CY 2007]};**

**10**. Apply Sorting on your Data using **Order**Function

To sort your data you can use order function, using this function you can override default order specified in the cube design.

**Syntax**

**Order(<set>, Context, Asc | Desc|Bsc|Bdesc)**

**Example:**

Retrieve all the products in descending order of their Internet sales amount of year 2007

Hide   Copy Code

select [Measures].[Internet Sales Amount] on columns,

order([Product].[Product].[Product].members ,[Measures].[Internet Sales Amount],desc)

on rows

from [Adventure Works]

where {[Date].[Calendar Year].[CY 2007]}

Hope you have enjoyed this article. In this beginner article, I have tried to give Initial start up to technical newbies working in Microsoft BI and want to initiate in Learning Custom MDX.

We will learn about different MDX Functions and their usage in my next article on Advance Custom MDX.

I have included many sample queries with appropriate comments , Please download for further practise.

If you find this article helpful, then please do not forget to vote for me.