

A Brief MDX Tutorial Using Mondrian

Wei Wang

weiw AT cse.unsw.edu.au

School of Computer Science & Engineering
University of New South Wales

[Pentaho].[Mondrian]

Pentaho: Open source business intelligence suite

- Mondrian - Open Source OLAP Server
- JFreeReport - Open Source Reporting
- Kettle - Open Source Data Integration (ETL)
- Pentaho - Comprehensive Open Source BI Suite
- Weka - Open Source Data Mining

Mondrian

- an OLAP server written in Java
- It implements the MDX language, and the XML for Analysis (XMLA) and JOLAP specifications.
- It reads from SQL and other data sources, and aggregates data in a memory cache.

Installation

- Installation (the embedded version):
 - Download Mondrian-embedded version and Tomcat
 - Follow the instructions in `install.html`
- Start tomcat (run the `startup.bat`)
- Browse `http://localhost:8080/mondrian-embedded/`

Motivation

- Every input/output in SQL must be relation

```
SELECT Store.state, SUM(sales)
FROM F, Store, Time
WHERE F.storekey=Store.storekey AND F.timekey=Time.timekey
GROUP BY Store.state
```

```
CA | 111,111
IL | 222,222
```

```
SELECT Store.state, Time.quarter, SUM(sales)
FROM F, Store, Time
WHERE F.storekey=Store.storekey AND F.timekey=Time.timekey
GROUP BY Store.state, Time.quarter
```

```
CA | Q1 | 44,444
CA | Q2 | 11,111
... | ... | ...
IL | Q4 | 88,888
```

- Typical reporting/analytical applications requires **cross-tab**

```
      | Q1 | Q2 | Q3 | Q4 |
CA | ... | ... | ... | ... |
IL | ... | ... | ... | ... |
```

Pivot and Unpivot

- SQL can produce cross tab via the CASE construct

```
SELECT Store.state AS STATE,  
       SUM(CASE WHEN Time.quarter='Q1' THEN sales END) AS Q1  
       ... ..  
       SUM(CASE WHEN Time.quarter='Q4' THEN sales END) AS Q4  
FROM F, Store, Time  
WHERE F.storekey=Store.storekey AND F.timekey=Time.timekey  
GROUP BY Store.state
```

- PIVOT has been proposed to help.
- MDX can support these types of queries more naturally and more efficiently.

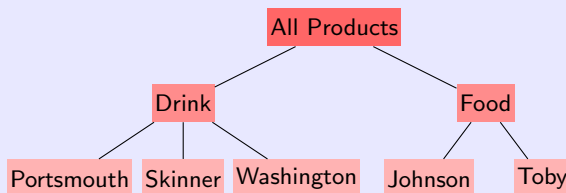
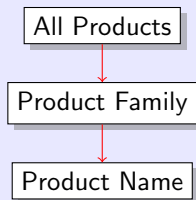
```
SELECT {[Time].[Year].[1997].CHILDREN} ON COLUMNS,  
       {[Store].[Store State].MEMBERS} ON ROWS  
FROM [Sales]  
WHERE ([Measures].[Store Sales])
```

- Who speaks MDX?
 - MS SQL Server 200? Analysis Service (SSAS), Essbase (now owned by Oracle), Mondrian, etc.

Basic Concepts

- **Dimensions** and **Members**
 - A dimension may have several levels
 - Each level has a number of members
- **Axes**
 - Refers to the “dimensions” of a query’s result cube
 - An axis could contain several cube dimensions in combination
- **Measures**
 - Attributes (of numerical values) to be aggregated and analyzed.
 - They collectively forms the Measures dimension.
- Default member: the top-level member (usu. ALL)
- Default measure: the first measure specified in the cube
- One can specify the default members/measures or disallow the ALL level in the schema
- **Tuple**
 - to define a slice of data from a cube
 - ([Product].[Product Family].[Drink], [Store].[USA].[CA])
- **Set**
 - an ordered collection of zero, one or more tuples
 - usually used to define axis and slicer dimensions
 - {[Time].[2007].[Q1], [Time].[2007].[Q2] }

Example



the Product dimension

Members of the Product dimension

Return a sets of members:

- `[Product].[Product Family].[Skinner]`
- `[Product].[Product Name].MEMBERS = { Portsmouth, Skinner, Washington, Johnson, Toby }`
- `[Product].[Drink].CHILDREN = { Portsmouth, Skinner, Washington }`
- `[Product].[Drink].[Skinner]:[Washington] = { Skinner, Washington }`
- `DESCENDANTS([Product].[Food], [Product Name]) = { Johnson, Toby }`

- The [Sales] cube

- Measures: [Unit Sales], [Store Cost], [Store Sales], [Sales Count], [Customer Count], [Promotion Sales].

```
SELECT {[Measures].Members} ON COLUMNS  
FROM [Sales]
```

- Calculated measures: [Profit], [Profit last Period], [Gewinn-Wachstum].

```
SELECT {AddCalculatedMembers([Measures].Members)}  
      ON COLUMNS  
FROM [Sales]
```

- Dimensions:

- [Store]: [Store Country] → [Store State] → [Store City] → [Store Name]
- [Time]: [Year] → [Quarter] → [Month]
- ...

Example

Use Mondrian's workbench to open `./demo/FoodMart.xml`.

MDX is not SQL

Basic Syntax

```
-- One of the three ways to write comments
SELECT {set 0} on COLUMNS, /* block comment */
      {set 1} on ROWS // line comment
      ...
      {set n} on AXIS(n)
FROM [cube]
WHERE (tuple) // called "slicer dimension"
```

Note

- No axis or the WHERE statement can share any of the same dimensions
- JPivot cannot display results with > 2 dimensions

Key differences between MDX and SQL

- “Cube in, Cube out” for MDX.
- set notation needs to be used after SELECT.
- FROM clause can name only one cube
- The WHERE clause describes the slicer axis (i.e., all the axes that is not a query axis) and is filtered by its default members.

Example

- A typical query

```
SELECT {[Time].[Year].[1997], [Time].[Year].[1998]} ON COLUMNS,  
       {[Store].[Store Name].MEMBERS} ON ROWS  
FROM [Sales]  
WHERE ([Measures].[Store Sales])
```

- Using the level.MEMBERS and member.CHILDREN functions

```
SELECT {[Time].[Year].[1997].CHILDREN} ON COLUMNS,  
       {[Store].[Store City].MEMBERS} ON ROWS  
FROM [Sales]  
WHERE ([Measures].[Store Sales])
```

Output

	1997	1998
HQ	?	?
Store 6	?	?
Store 7	?	?
...	?	?

	Q1	Q2	Q3	Q4
HQ	?	?	?	?
Store 6	?	?	?	?
Store 7	?	?	?	?
...	?	?	?	?

Slicer Dimension vs. Filter

- Slicer Dimension

➔ *slice on the [Product] dimension*

```
SELECT {[Time].[Year].[1997].CHILDREN} ON COLUMNS,  
       {[Store].[Store City].MEMBERS} ON ROWS  
FROM [Sales]  
WHERE ([Product].[Product Family].[Drink],  
       [Measures].[Store Sales])
```

- FILTER(set, search_condition) function.

➔ *if we are only interested in stores whose 1997 unit sales exceed 1000*

```
SELECT {[Time].[Year].[1997].CHILDREN} ON COLUMNS,  
       FILTER(  
           {[Store].[Store City].MEMBERS},  
           (Measures.[Unit Sales], [Time].[1997]) > 1000  
       ) ON ROWS  
FROM [Sales]  
WHERE ([Measures].[Store Sales])
```

ORDER

- Syntax: ORDER(set, expression, [, ASC | DESC | BASC | BDESC]
- The “B” prefix indicates the hierarchical order can be broken.
 - ➔ *List all measures for each city in decreasing order of their sales count*

```
SELECT Measures.MEMBERS ON COLUMNS,  
    ORDER(  
        [Store].[Store City].MEMBERS,  
        Measures.[Sales Count], BDESC  
    ) ON ROWS  
FROM [Sales]
```

- ➔ *if we are only interested in stores whose name is between “Beverly Hills” and “Spokane”*

```
SELECT Measures.MEMBERS ON COLUMNS,  
    ORDER(  
        {[Store].[Store City].[Beverly Hills]:[Spokane]},  
        [Store].CURRENTMEMBER.Name, BASC  
    ) ON ROWS  
FROM [Sales]
```

HEAD and TOPCOUNT

➔ *show me the profit of top-5 cities in terms of sales count*

```
SELECT Measures.[Profit] ON COLUMNS,  
       HEAD( ORDER( {[Store].[Store City].MEMBERS},  
                   Measures.[Sales Count], BDESC  
                   ), 5  
       ) ON ROWS  
FROM [Sales]
```

	[Measures].[Profit]
[Store].[All Stores].[USA].[OR].[Salem]	\$52,394.72
[Store].[All Stores].[USA].[WA].[Tacoma]	\$44,884.68
[Store].[All Stores].[USA].[OR].[Portland]	\$33,109.85
[Store].[All Stores].[USA].[CA].[Los Angeles]	\$32,773.74
[Store].[All Stores].[USA].[CA].[San Diego]	\$32,717.61

CROSSJOIN

- Q: all the results so far are 2-dimensional, what about 3D?
- CROSS JOIN(set1, set2)

```
SELECT [Time].[1997].CHILDREN ON COLUMNS,  
       CROSSJOIN( [Store].[Store State].MEMBERS,  
                  [Product].[Product Family].MEMBERS  
                ) ON ROWS  
FROM [Sales]  
WHERE (Measures.[Profit])
```

- The query axis (ROWS) is the combination of 2 cube dimensions.

NON EMPTY

- Problem with the previous query: many members in the ROW axis is empty, hence many empty rows.
- It needs a simple **filtering** — removing empty members from the axis.

```
SELECT [Time].[1997].CHILDREN ON COLUMNS,  
       NON EMPTY(  
         CROSSJOIN( [Store].[Store State].MEMBERS,  
                   [Product].[Product Family].MEMBERS  
                 )  
       ) ON ROWS  
FROM [Sales]  
WHERE (Measures.[Profit])
```

[Advanced MDX].[Calculate Member]

- **Calculated Members**

```
WITH MEMBER parent.name AS 'expression'
```

- only visible to the query.

```
WITH MEMBER [Time].[1997].[H1] AS  
    '[Time].[1997].[Q1] + [Time].[1997].[Q2]'  
    MEMBER [Time].[1997].[H2] AS  
    '[Time].[1997].[Q3] + [Time].[1997].[Q4]'  
SELECT {[Time].[1997].[H1], [Time].[1997].[H2]} ON COLUMNS  
    [Store].[Store Name].MEMBERS ON ROWS  
FROM [Sales]  
WHERE (Measures.[Profit])
```

[Advanced MDX].[Calculated Members]

- Define and use new measures

```
WITH MEMBER Measures.ProfitPercent AS
    '(Measures.[Store Sales] - Measures.[Store Cost]) /
    (Measures.[Store Cost])'
SELECT [Time].[1997].CHILDREN ON COLUMNS,
    [Store].[Store Name].MEMBERS ON ROWS
FROM [Sales]
WHERE (Measures.[ProfitPercent])
```

➡ *SOLVE_ORDER is important. For those who are interested, see <http://msdn2.microsoft.com/en-us/library/ms145539.aspx>*

Architecture of OLAP Servers

Layer	Description
Client	presentation tools (e.g., pivot table), reporting tools, XMLA client applications
OLAP Server	parse MDX queries and perform query processing and optimization
Storage	Relational or Multidimensional Data Warehouse

- Mondrian
 - JPivot, JRubik
 - Query rewriting, Caching (roll-up, chunk-based), Materialized view (aggregate table).
 - Relational DBMS as the storage engine (i.e., transform MDX queries into SQL queries)

For those interested to read more:

➡ Check out tables in *MondrianFoodMart.mdb*, *FoodMart.xml*, *Mondrian 2.2.2 Technical Guide.pdf*

➡ Check out LucidDB <http://www.luciddb.org/features.html>, which is specifically built for DW and OLAP

References

- MDX Reference @ MSDN:
<http://msdn2.microsoft.com/en-us/library/ms145506.aspx>
- The Baker's Dozen: 13 Tips for Querying OLAP Databases with MDX:
<http://www.devx.com/codemag/Article/37460/1954?pf=true>
- Tutorial: Introduction to Multidimensional Expression (MDX).
http://www.fing.edu.uy/inco/grupos/csi/esp/Cursos/cursos_act/2005/DAP_SistDW/Material/2-SDW-Laboratorio1-2005.pdf
- MDX resources: <http://www.mosha.com/msolap/mdx.htm>

