Data Warehouse Systems: Design and Implementation

Alejandro VAISMAN
Department of Information Engineering
Instituto Tecnológico de Buenos Aires
avaisman@itba.edu.ar

Esteban ZIMÁNYI
Department of Computer & Decision Engineering (CoDe)
Université Libre de Bruxelles
ezimanyi@ulb.ac.be

Chapter 6: Querying the Data Warehouse

Outline

- Introduction to the MDX Language
- Advanced MDX
- Querying the Northwind Cube in MDX
- Querying the Northwind Cube in SQL
- Comparison of MDX and SQL

Chapter 6: Querying the Data Warehouse

Outline

➥ **Introduction to the MDX Language**

- MDX Basics
- Slicing
- Navigation
- Cross Join
- Subqueries
- Calculated Members
- Relative Navigation

◆ Advanced MDX

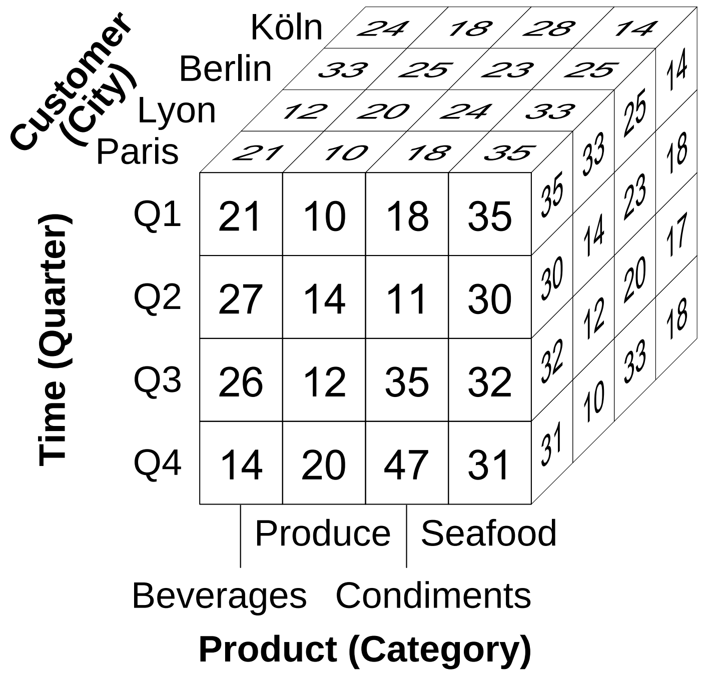
◆ Querying the Northwind Cube in MDX

◆ Querying the Northwind Cube in SQL

◆ Comparison of MDX and SQL

Introduction to the MDX Language

- A simple three-dimensional cube with one measure



- Two fundamental concepts in MDX: **tuples** and **sets**
  - A tuple identifies a single cell in a multidimensional cube
  - A tuple is defined by stating one member from one or several dimensions of the cube
- To identify the cell in the top left corner (value 21) we provide the coordinates of each dimension:
  (Product.Category.Beverages, Time.Quarter.Q1, Customer.City.Paris)

Introduction to the MDX Language

- **Tuples**
  - Several ways to specify a member of a dimension
  - The order of the members is not significant; these two ways are equivalent:
    (Product.Category.Beverages, Time.Quarter.Q1, Customer.City.Paris)
    (Time.Quarter.Q1, Product.Category.Beverages, Customer.City.Paris)
  - Since a tuple points to a single cell, each member in the tuple must belong to a different dimension
  - A tuple does not need to specify a member from every dimension:
    (Customer.City.Paris)
  - The tuple below points to the sales of beverages in Paris
    (Customer.City.Paris, Product.Category.Beverages)
  - If a member for a dimension is not specified, the default member for the dimension is implied (typically the All member)
- **Sets**
  - A collection of tuples defined using the same dimensions
    { (Product.Category.Beverages, Time.Quarter.Q1, Customer.City.Paris),
    (Product.Category.Beverages, Time.Quarter.Q1, Customer.City.Lyon) }
  - This set points to the cell with value 21, and the one behind it with value 12

Introduction to the MDX Language

- **Tuples and hierarchies**
  - Hierarchy in the Customer dimension: Customer → City → State → Country
  - This tuple points to the cell corresponding to total sales of beverages in France in the first quarter:
    - (Customer.Country.France, Product.Category.Beverages, Time.Quarter.Q1)
- **Measures**
  - In MDX measures act like dimensions
  - If there are three measures in our cube: UnitPrice, Discount, and SalesAmount, then:
    - The Measures dimension (exists in every cube), contains three members
  - We can specify the measure we want as in the following tuple:
    (Customer.Country.France, Product.Category.Beverages, Time.Quarter.Q1, Measures.SalesAmount)
  - If a measure is not specified, a default is used.

Basic MDX Queries

- The syntax of a typical MDX query is:

```
SELECT  ⟨ axis specification ⟩
FROM    ⟨ cube ⟩
[ WHERE ⟨ slicer specification ⟩ ]
```

- MDX resembles SQL, but differ in many ways
  - The **axis specification** allows to state the axes of a query as well as the members selected for each of these axis
  - Up to 128 axes in an MDX query: each axis has a number: 0 for the *x*-axis, 1 for the *y*-axis, 2 for the *z*-axis, ...
  - The first axes have predefined names: COLUMNS, ROWS, PAGES, CHAPTERS, and SECTIONS; query axes cannot be skipped, e.g., a query cannot have a ROWS axis without a COLUMNS axis
  - The **slicer specification** on the WHERE clause is optional; If not specified, the query returns the default measure for the cube

Basic MDX Queries

- Simplest form of an axis specification: take the members of the required dimension, including those of the special Measures dimension
- *Display all the measures for customers summarized at the country level*

```
SELECT [Measures].MEMBERS ON COLUMNS,
       [Customer].[Country].MEMBERS ON ROWS
FROM   Sales
```

	Unit Price	Quantity	Discount	Sales Amount	Freight	Sales Count
Austria	€ 84.77	4,644	21.71 %	€ 115,328.31	€ 6,827.10	114
Belgium	€ 64.65	1,242	9.72 %	€ 30,505.06	€ 1,179.53	49
Denmark	€ 70.28	1,156	17.94 %	€ 32,428.94	€ 1,377.75	45
Finland	€ 54.41	848	9.09 %	€ 17,530.05	€ 827.45	51
France	€ 64.51	3,052	11.76 %	€ 77,056.01	€ 3,991.42	172
Germany	€ 79.54	8,670	19.26 %	€ 219,356.08	€ 10,459.01	309
Ireland	(null)	(null)	(null)	(null)	(null)	(null)
...	...	...	...	...	...	...

Slicing

- *Show all measures by year*

```
SELECT Measures.MEMBERS ON COLUMNS,
       [Order Date].Year.MEMBERS ON ROWS
FROM   Sales
```

	Unit Price	Quantity	Discount	Sales Amount	Freight	Sales Count
All	€ 134.14	46,388	27.64 %	€ 1,145,155.86	€ 58,587.49	1,931
1996	€ 99.55	8,775	21.95 %	€ 191,849.87	€ 9,475.00	371
1997	€ 116.63	23,461	25.89 %	€ 570,199.61	€ 29,880.49	982
1998	€ 205.38	14,152	35.74 %	€ 383,106.38	€ 19,232.00	578

- To restrict the result to Belgium, we can write

```
SELECT Measures.MEMBERS ON COLUMNS,
       [Order Date].Year.MEMBERS ON ROWS
FROM   Sales
WHERE  (Customer.Country.Belgium)
```

- The above query only changes the values returned for each cell

Slicing

- Adding multiple members from different hierarchies
- *All measures for all years for customers from Belgium who bought products in the category beverages*

```
SELECT Measures.MEMBERS ON COLUMNS,
       [Order Date].Year.MEMBERS ON ROWS
FROM   Sales
WHERE  (Customer.Country.Belgium, Product.Categories.Beverages)
```

- For multiple members from the same hierarchy we need to include a set
- Aggregated values for Belgium and France in each cell

```
SELECT Measures.MEMBERS ON COLUMNS,
       [Order Date].Year.MEMBERS ON ROWS
FROM   Sales
WHERE  ( { Customer.Country.Belgium, Customer.Country.France },
       Product.Categories.Beverages)
```

- A set in the WHERE clause implicitly aggregates values for all members in the set

Slicing

- Specifying in the WHERE clause the measure to be displayed

```
SELECT [Order Date].Year.MEMBERS ON COLUMNS,
       Customer.Country.MEMBERS ON ROWS
FROM   Sales
WHERE  Measures.[Sales Amount]
```

	All	1996	1997	1998
Austria	€ 115,328.31	€ 24,467.52	€ 55,759.04	€ 35101.7502
Belgium	€ 30,505.06	€ 5,865.10	€ 9,075.48	€ 15,564.48
Denmark	€ 32,428.93	€ 2,952.40	€ 25,192.53	€ 4,284.00
Finland	€ 17,530.05	€ 2,195.760	€ 13,077.29	€ 2,257.00
...	...	...	...	...

- Measures and dimensions in the WHERE clause

```
SELECT [Order Date].Year.MEMBERS ON COLUMNS,
       Customer.Country.MEMBERS ON ROWS
FROM   Sales
WHERE  (Measures.[Sales Amount], Product.Category.[Beverages])
```

Navigation

- The result of the query above contains aggregated values from all the years, including the All column
- To omit the All member we must use the CHILDREN function

  SELECT [Order Date].Year.CHILDREN ON COLUMNS, ...
- However, rows do not include All
- Reason: Customer.Country.MEMBERS is shorthand for Customer.Geography.Country.MEMBERS
  - Thus, it selects the members of Country
  - All is the topmost member of the hierarchy, not a member of the Country level
  - Therefore it does not appear in the result
- Customer has an attribute hierarchy Company Name; thus, using the expression

  Customer.[Company Name].MEMBERS

  The result will contain the All member, plus the names of all the customers

Navigation

- *Sales amount of customers by year and by state, in France and Italy*

```
SELECT [Order Date].Year.MEMBERS ON COLUMNS,
       NON EMPTY { Customer.France.CHILDREN,
       Customer.Italy.CHILDREN } ON ROWS
FROM   Sales
WHERE  Measures.[Sales Amount]
```

	All	1996	1997	1998
Bas-Rhin	€ 18,534.07	€ 9,986.20	€ 7,817.87	€ 730.00
Bouches-du-Rhne	€ 19,373.10	€ 2,675.88	€ 10,809.36	€ 5,887.86
. . .	. . .	. . .	. . .	. . .
Reggio Emilia	€ 6,641.83	€ 80.10	€ 3,000.84	€ 3,560.89
Torino	€ 1,545.70	(null)	€ 249.70	€ 1,296.00

- Note again, All in columns (does not include CHILDREN), not in rows (includes CHILDREN)

Navigation

- To drill-down we need the DESCENDANTS function
- *Sales amount for German cities*

```
SELECT [Order Date].Year.MEMBERS ON COLUMNS,
       NON EMPTY DESCENDANTS(Customer.Germany, Customer.City)
       ON ROWS
FROM   Sales
WHERE  Measures.[Sales Amount]
```

	All	1996	1997	1998
Mannheim	€ 2,381.80	(null)	€ 1,079.80	€ 1,302.00
Stuttgart	€ 8,705.23	€ 2,956.60	€ 4,262.83	€ 1,485.80
Mnchen	€ 26,656.56	€ 9,748.04	€ 11,829.78	€ 5,078.74
...	...	...	...	...

- By default, DESCENDANTS displays only members at the level specified as second attribute
- An optional flag as third argument allows to include or exclude descendants or children before and after the specified level

Navigation: Function Qualifiers

- SELF: default, displays values for the City level above
- BEFORE: displays values from the state level up to the Country level
- SELF_AND_BEFORE: displays values from the City level up to the Country level
- AFTER: displays values from the Customer level, since it is only level after City
- SELF_AND_AFTER: displays values from the City and Customer levels
- BEFORE_AND_AFTER: displays values from the Country level to the Customer level, excluding the former
- SELF_BEFORE_AFTER: displays values from the Country level to the Customer level
- LEAVES: displays values from the City level, since it is the only leaf level between Country and City
- If LEAVES is used without specifying the level, as in

  DESCENDANTS(Customer.Geography.Germany, ,LEAVES)

  the leaf level, i.e., Customer will be displayed

Navigation

- ASCENDANTS returns a set that includes all the ancestors of a member and the member itself
- *Sales amount measure for a particular customer and all its ancestors*

```
SELECT Measures.[Sales Amount] ON COLUMNS,
       ASCENDANTS(Customer.Geography.[Du monde entier]) ON ROWS
FROM   Sales
```

	Sales Amount
Du monde entier	€ 1,548.70
Nantes	€ 4,720.86
Loire-Atlantique	€ 4,720.86
France	€ 77,056.01
Europe	€ 683,523.76
All Customers	€ 1,145,155.86

- To obtain the result for an ancestor at a specified level, the function ANCESTOR can be used:

```
SELECT Measures.[Sales Amount] ON COLUMNS,
       ANCESTOR(Customer.Geography.[Du monde entier],
       Customer.Geography.State) ON ROWS
FROM   Sales
```

Cross Join

- Combines several dimensions in a single axis to display more than 2 axes
- *Sales amount by product category, country and quarter* (customer and time combined in same axis)

```
SELECT Product.Category.MEMBERS ON COLUMNS,
       CROSSJOIN(Customer.Country.MEMBERS,
       [Order Date].Calendar.Quarter.MEMBERS) ON ROWS
FROM   Sales
WHERE  Measures.[Sales Amount]
```

- Can also use ‘*’

```
SELECT Product.Category.MEMBERS ON COLUMNS,
       Customer.Country.MEMBERS *
       [Order Date].Calendar.Quarter.MEMBERS) ON ROWS
FROM   Sales
WHERE  Measures.[Sales Amount]
```

		Beverages	Condiments	Confections	...
Austria	Q3 1996	€ 708.80	€ 884.00	€ 625.50	...
Austria	Q4 1996	€ 12,955.60	€ 703.60	€ 36.00	...
Austria	Q1 1997	(null)	€ 3,097.50	€ 1,505.22	...
Austria	Q2 1997	€ 1,287.50	€ 1,390.95	€ 3,159.00	...
...	...	...	...	...	...

Cross Join

- More than two crossjoins

```
SELECT Product.Category.MEMBERS ON COLUMNS,
       Customer.Country.MEMBERS *
       [Order Date].Calendar.Quarter.MEMBERS *
       Shipper.[Company Name].MEMBERS ON ROWS
FROM   Sales
WHERE  Measures.[Sales Amount]
```

			Beverages	Condiments	Confections	···
Austria	All	All	€ 20,818.30	€ 14,103.42	€ 13,176.91	···
Austria	All	Federal Shipping	€ 11,657.20	€ 1,980.93	€ 6,412.01	···
Austria	All	Speedy Express	€ 7,063.60	€ 5,847.54	€ 868.45	···
Austria	All	United Package	€ 2,097.50	€ 6,274.95	€ 5,896.45	···
Austria	Q3 1996	All	€ 708.80	€ 884.00	€ 625.50	···
Austria	Q3 1996	Federal Shipping	€ 100.80	(null)	€ 625.50	···
Austria	Q3 1996	Speedy Express	€ 608.00	€ 884.00	(null)	···
Austria	Q3 1996	United Package	(null)	(null)	(null)	···
Austria	Q4 1996	All	€ 12,955.60	€ 703.60	€ 36.00	···
···	···	···	···	···	···	···

Subqueries

- WHERE clause applies a slice to the cube
    - Not only applied to select the measure to be displayed, also the dimensions
- *Sales amount of the categories beverages and condiments*
- Classic WHERE slicer

```
SELECT Measures.[Sales Amount] ON COLUMNS,
              [Order Date].Calendar.Quarter.MEMBERS ON ROWS
FROM      Sales
WHERE {Product.Category.Beverages, Product.Category.Condiments}
```

- Alternative with a **subquery**

```
SELECT Measures.[Sales Amount] ON COLUMNS,
              [Order Date].Calendar.Quarter.MEMBERS ON ROWS
FROM      ( SELECT { Product.Category.Beverages,
                         Product.Category.Condiments } ON COLUMNS
                FROM     Sales )
```

- Unlike in SQL, in the outer query we can mention attributes not selected in the subquery
- **Key difference**:
    - If Category hierarchy is in the WHERE clause, cannot appear on any axis
    - Not the case in the subquery approach

Subqueries

- Difference between the two approaches (continued)

```
SELECT Measures.[Sales Amount] ON COLUMNS,
       [Order Date].Calendar.Quarter.MEMBERS *
       Product.Category.MEMBERS ON ROWS
FROM   ( SELECT { Product.Category.Beverages,
                  Product.Category.Condiments } ON COLUMNS
         FROM   Sales )
```

		Sales Amount
Q3 1996	Beverages	€ 8,996.98
Q3 1996	Condiments	€ 4,003.30
Q4 1996	Beverages	€ 32,937.70
Q4 1996	Condiments	€ 10,778.16
...	...	...

- Members of Category hierarchy now are only the beverages and condiments categories, and not the other categories

Subqueries

- Not restricted to one dimension in the subquery

```
SELECT Measures.[Sales Amount] ON COLUMNS,
        [Order Date].Calendar.Quarter.MEMBERS * Product.Category.MEMBERS ON ROWS
FROM    ( SELECT ( { Product.Category.Beverages, Product.Category.Condiments },
                 { [Order Date].Calendar.[Q1 1997], [Order Date].Calendar.[Q2 1997] } ) ON COLUMNS
         FROM Sales )
```

		Sales Amount
Q1 1997	Beverages	€ 33,902.08
Q1 1997	Condiments	€ 9,912.22
Q2 1997	Beverages	€ 21,485.53
Q2 1997	Condiments	€ 10,875.70

- Multiple nesting

```
SELECT Measures.[Sales Amount] ON COLUMNS,
        [Order Date].Calendar.[Quarter].Members ON ROWS
FROM    ( SELECT TOPCOUNT(Customer.Country.MEMBERS, 2,
                Measures.[Sales Amount]) ON COLUMNS
         FROM ( SELECT { Product.Category.Beverages, Product.Category.Condiments } ON COLUMNS
                FROM    Sales ) )
```

- TOPCOUNT sorts a set in descending order w.r.t. the expression given as third parameter returning the specified number of elements with the highest values

Calculated Members and Named Sets

- **Calculated members** define new members in a dimension, or new measures, computed at runtime

  WITH MEMBER Parent.MemberName AS ⟨ expression ⟩

- **Named sets** define new sets

  WITH SET SetName AS ⟨ expression ⟩

- Example: A measure calculating the percentage profit of sales

```
WITH MEMBER Measures.Profit% AS
         (Measures.[Sales Amount] - Measures.[Freight]) /
         (Measures.[Sales Amount]), FORMAT_STRING = '#0.00%'
SELECT { [Sales Amount], Freight, Profit% } ON COLUMNS,
         Customer.Country ON ROWS
FROM     Sales
```

	Sales Amount	Freight	Profit%
Austria	€ 115,328.31	€ 6,827.10	94.08%
Belgium	€ 30,505.06	€ 1,179.53	96.13%
Denmark	€ 32,428.94	€ 1,377.75	95.75%
Finland	€ 17,530.05	€ 827.45	95.28%
...	...	...	...

Static and Dynamic Named Sets

- **Static named set**: Nordic Countries composed of Denmark, Finland, Norway, and Sweden

```
WITH SET [Nordic Countries] AS
        { Customer.Country.Denmark, Customer.Country.Finland,
        Customer.Country.Norway, Customer.Country.Sweden }
SELECT Measures.MEMBERS ON COLUMNS, [Nordic Countries] ON ROWS
FROM    Sales
```

	Unit Price	Quantity	Discount	Sales Amount	Freight	Sales Count
Denmark	€ 70.28	1,156	17.94 %	€ 32,428.94	€ 1,377.75	45
Finland	€ 54.41	848	9.09 %	€ 17,530.05	€ 827.45	51
Norway	€ 97.95	152	0.00 %	€ 5,321.15	€ 257.45	15
Sweden	€ 68.73	2,149	19.57 %	€ 51,292.64	€ 3,032.12	94

- **Dynamic named set**: *Top five selling products*

```
WITH SET TopFiveProducts AS
        TOPCOUNT ( Product.Categories.Product.MEMBERS, 5, Measures.[Sales Amount] )
SELECT { [Unit Price], Quantity, Discount, [Sales Amount] } ON COLUMNS, TopFiveProducts ON ROWS
FROM    Sales
```

	Unit Price	Quantity	Discount	Sales Amount
Cte de Blaye	€ 256.63	623	4.78 %	€ 141,396.74
Raclette Courdavault	€ 53.17	1,369	3.96 %	€ 65,658.45
Thringer Rostbratwurst	€ 115.24	596	6.21 %	€ 63,657.02
Tarte au sucre	€ 46.56	1,068	5.53 %	€ 46,643.97
Camembert Pierrot	€ 34.32	1,498	7.21 %	€ 44,200.68

Relative Navigation

- Methods to traverse a hierarchy: CURRENTMEMBER, PREVMEMBER, NEXTMEMBER, PARENT, FIRSTCHILD, LASTCHILD
- *Sales of a member of the* Geography *hierarchy as a percentage of the sales of its parent*

```
WITH MEMBER Measures.[Percentage Sales] AS
          (Measures.[Sales Amount], Customer.Geography.CURRENTMEMBER) /
          (Measures.[Sales Amount], Customer.Geography.CURRENTMEMBER.PARENT),
          FORMAT_STRING = '#0.00%'
SELECT { Measures.[Sales Amount], Measures.[Percentage Sales] } ON COLUMNS,
          DESCENDANTS(Customer.Europe, Customer.Country, SELF_AND_BEFORE) ON ROWS
FROM    Sales
```

- CURRENTMEMBER returns the current member along a dimension during an iteration; abbreviated: (Measures.[Sales Amount]) / (Measures.[Sales Amount], Customer.Geography.CURRENTMEMBER.PARENT)

	Sales Amount	Percentage Sales
Europe	€ 683,523.76	59.69%
Austria	€ 115,328.31	16.87%
Belgium	€ 30,505.06	4.46%
Denmark	€ 32,428.94	4.74%
Finland	€ 17,530.05	2.56%
...	...	...

Relative Navigation

- Previous query does not work for All (All does not have a parent); a conditional expression is required:

```
WITH MEMBER Measures.[Percentage Sales] AS
        IIF((Measures.[Sales Amount], Customer.Geography.CURRENTMEMBER.PARENT)=0, 1,
        (Measures.[Sales Amount]) / (Measures.[Sales Amount],
        Customer.Geography.CURRENTMEMBER.PARENT))
```

- IIF has three parameters
  (1) A Boolean condition, in this case:
  `(Measures.[Sales Amount],Customer.Geography.CURRENTMEMBER.PARENT)=0`
  (2) The value returned if the condition is true: ‘1’ in this case, since All has no parent, which will be given to the percentage sales
  (3) The value returned if the condition is false
- PREVMEMBER used to show growth over a time period
- *Net sales and incremental change from the* **previous** *time member for all months in 1996*

```
WITH MEMBER Measures.[Net Sales Growth] AS
          (Measures.[Net Sales]) - (Measures.[Net Sales], [Order Date].Calendar.PREVMEMBER),
          FORMAT_STRING = '€ ###,##0.00; € -###,##0.00'
SELECT {Measures.[Net Sales], Measures.[Net Sales Growth]} ON COLUMNS,
          DESCENDANTS([Order Date].Calendar.[1996], [Order Date].Calendar.[Month]) ON ROWS
FROM      Sales
```

Generating Sets

- GENERATE: Iterates through the members of a set, using a second set as template for the resultant set
- *Sales amount by category for all customers in Belgium and France*
- GENERATE function used to avoid enumerating all customers for each country

```
SELECT Product.Category.MEMBERS ON COLUMNS,
       GENERATE({Customer.Belgium, Customer.France},
       DESCENDANTS(Customer.Geography.CURRENTMEMBER,[Company Name])) ON ROWS
FROM   Sales
WHERE  Measures.[Sales Amount]
```

	Beverages	Condiments	Confections	Dairy Products	···
Maison Dewey	€ 108.00	€ 680.00	€ 2,659.38	€ 2,972.00	···
Suprêmes délices	€ 3,108.08	€ 1,675.60	€ 4,820.20	€ 5,688.00	···
Blondesddsl père et fils	€ 3,975.92	(null)	€ 1,939.00	€ 2,872.00	···
Bon app’	€ 877.50	€ 2,662.48	€ 2,313.67	€ 1,912.43	···
La maison d’Asie	€ 1,499.15	€ 525.90	€ 2,085.90	€ 757.76	···
Du monde entier	€ 194.00	(null)	€ 60.00	€ 201.60	···
···	···	···	···	···	···

Chapter 6: Querying the Data Warehouse

Outline

- Introduction to the MDX Language
- **Advanced MDX**
  - Time Series Functions
  - Sorting and Filtering
  - Top and Bottom Analysis
  - Aggregation
- Querying the Northwind Cube in MDX
- Querying the Northwind Cube in SQL
- Comparison of MDX and SQL

Time Series Functions

- PARALLELPERIOD: compares values of a specified member with those of a member in the same relative position in a prior period

```
WITH MEMBER Measures.[Previous Year] AS
        (Measures.[Net Sales], PARALLELPERIOD([Order Date].Calendar.Quarter, 4)),
        FORMAT_STRING = '€ ###,##0.00'
MEMBER Measures.[Net Sales Growth] AS
        Measures.[Net Sales] - Measures.[Previous Year],
FORMAT_STRING = '€ ###,##0.00; € -###,##0.00'
SELECT { [Net Sales], [Previous Year], [Net Sales Growth] } ON COLUMNS,
        [Order Date].Calendar.Quarter ON ROWS
FROM Sales
```

	Net Sales	Previous Year	Net Sales Growth
Q3 1996	€ 67,531.59	(null)	€ 67,531.59
Q4 1996	€ 114,843.27	(null)	€ 114,843.27
Q1 1997	€ 125,174.40	(null)	€ 125,174.40
Q2 1997	€ 121,518.78	(null)	€ 121,518.78
Q3 1997	€ 133,636.32	€ 67,531.59	€ 66,104.73
Q4 1997	€ 159,989.61	€ 114,843.27	€ 45,146.34
Q1 1998	€ 259,322.36	€ 125,174.40	€ 134,147.95
Q2 1998	€ 104,552.03	€ 121,518.78	€ -16,966.75

Time Series Functions

- OPENINGPERIOD and CLOSINGPERIOD return the first or last sibling among the descendants of a member at a specified level
- *Difference between sales quantity of a month and that of the opening month of the quarter*

```
WITH MEMBER Measures.[Quantity Difference] AS
        (Measures.[Quantity]) - (Measures.[Quantity],
        OPENINGPERIOD([Order Date].Calendar.Month,
        [Order Date].Calendar.CURRENTMEMBER.PARENT))
SELECT {Measures.[Quantity], Measures.[Quantity Difference]} ON COLUMNS,
        [Order Date].Calendar.[Month] ON ROWS
FROM Sales
```

- To compute the calculated member Quantity Difference, the opening period at the month level is taken for the quarter to which the month corresponds
- If CLOSINGPERIOD is used, query returns sales based on the **final month** of the specified season

	Quantity	Quantity Difference
July 1996	1,425	(null)
August 1996	1,221	-204
September 1996	882	-543
October 1996	1,602	(null)
November 1996	1,649	47
December 1996	1,996	394
...	...	...

Time Series Functions

- PERIODSTODATE returns a set of periods (members) from a specified level starting with the first period and ending with a specified member
- *Compute a set of all the months up to and including the month of June for the year 1997*

```
PERIODSTODATE([Order Date].Calendar.Year, [Order Date].Calendar.[June 1997])
```

- *Sum of sales amount for Italy and Greece*
  - For this, in addition to PERIODSTODATE we need to use the SUM function

```
SUM({Customer.Country.Italy, Customer.Country.Greece}, Measures.[Sales Amount])
```

- We can also display the sum of the current time member over the year level

```
SUM(PERIODSTODATE([Order Date].Calendar.Year,
        [Order Date].Calendar.CURRENTMEMBER), Measures.[Sales Amount])
```

Time Series Functions

- **YTD**, **QTD**: Year-to-date and quarter-to-date

```
WITH MEMBER Measures.YTDSales AS SUM(PERIODSTODATE([Order Date].Calendar.Year,
            [Order Date].Calendar.CURRENTMEMBER), Measures.[Sales Amount])
        MEMBER Measures.QTDSales AS SUM(PERIODSTODATE([Order Date].Calendar.Quarter,
            [Order Date].Calendar.CURRENTMEMBER), Measures.[Sales Amount])
SELECT { [Sales Amount], YTDSales, QTDSales } ON COLUMNS,
            [Order Date].Calendar.Month.MEMBERS ON ROWS
FROM    Sales
```

	Sales Amount	YTDSales	QTDSales
July 1996	€ 27,246.10	€ 27,246.10	€ 27,246.10
August 1996	€ 23,104.98	€ 50,351.07	€ 50,351.07
September 1996	€ 20,582.40	€ 70,933.47	€ 70,933.47
October 1996	€ 33,991.56	€ 104,925.04	€ 33,991.56
November 1996	€ 44,365.42	€ 149,290.46	€ 78,356.98
December 1996	€ 42,559.41	€ 191,849.87	€ 120,916.40
January 1997	€ 57,187.26	€ 57,187.26	€ 57,187.26
February 1997	€ 36,275.14	€ 93,462.39	€ 93,462.39
...	...	...	...

- YTDSales and QTDSales for February 1997: Sum of Sales Amount of January and February 1997
- YTDSales for December 1996: Sum of Sales Amount from July 1996 to December 1996 (no sales prior to July 1996)
- QTDSales for December 1996: Sum of Sales Amount from October 1996 to December 1996

Moving Average

- The LAG function, combined with the Range operator ':' help us to write moving averages in MDX
- **Range** returns a set of members made of two given members and all the members in between
- *Three-month moving average of the number of orders*

```
WITH MEMBER Measures.MovAvg3Months AS
          AVG([Order Date].Calendar.CURRENTMEMBER.LAG(2):
          [Order Date].Calendar.CURRENTMEMBER, Measures.[Order No]),
          FORMAT_STRING = '###,##0.00'
SELECT { Measures.[Order No], MovAvg3Months } ON COLUMNS,
          [Order Date].Calendar.Month.MEMBERS ON ROWS
FROM      Sales
WHERE  (Measures.MovAvg3Months)
```

- The LAG(2) function obtains the month that is two months before to the current one
- Range returns the set containing the three months over which the average is computed

	Order No	MovAvg3Months
July 1996	21	21.00
August 1996	25	23.00
September 1996	21	22.33
October 1996	25	23.67
November 1996	25	23.67
December 1996	29	26.33
...	...	...

Filtering

- Allows to reduce the number of axis members that are displayed
- *Sales amount in 1997 by city and product category, only for cities whose sales amount exceed €20,000*

```
SELECT Product.Category.MEMBERS ON COLUMNS,
       FILTER(Customer.City.MEMBERS, (Measures.[Sales Amount],
       [Order Date].Calendar.[1997])>25000) ON ROWS
FROM   Sales
WHERE  (Measures.[Net Sales Growth], [Order Date].Calendar.[1997])
```

	Beverages	Condiments	Confections	Dairy Products	···
Graz	€ -2,370.58	€ 6,114.67	€ 8,581.51	€ 7,171.01	···
Cunewalde	€ 6,966.40	€ 2,610.51	€ 8,821.85	€ 7,144.74	···
London	€ 2,088.23	€ 683.88	€ 1,942.56	€ 83.13	···
Montral	€ 9,142.78	€ 2,359.90	€ 213.93	€ 3,609.16	···
Boise	€ 1,871.10	€ 94.84	€ 4,411.46	€ 6,522.61	···

Filtering

◆ *Customers who in 1997 had profit margins below the state average*

```
WITH MEMBER Measures.[Profit%] AS
          (Measures.[Sales Amount] - Measures.[Freight]) /
          (Measures.[Sales Amount]), FORMAT_STRING = '#0.00%'
MEMBER Measures.[Profit%City] AS
          (Measures.[Profit%], Customer.Geography.CURRENTMEMBER.PARENT),
          FORMAT_STRING = '#0.00%'
SELECT { Measures.[Sales Amount], Measures.[Freight], Measures.[Net Sales],
          Measures.[Profit%], Measures.[Profit%City] } ON COLUMNS,
          FILTER(NONEMPTY(Customer.Customer.MEMBERS),
          (Measures.[Profit%]) < (Measures.[Profit%City])) ON ROWS
FROM     Sales
WHERE  [Order Date].Calendar.[1997]
```

	Sales Amount	Freight	Net Sales	Profit%	Profit%City
France restauration	€ 920.10	€ 30.34	€ 889.76	96.70%	97.40%
Princesa Isabel Vinhos	€ 1,409.20	€ 86.85	€ 1,322.35	93.84%	95.93%
Around the Horn	€ 6,406.90	€ 305.59	€ 6,101.31	95.23%	95.58%
North/South	€ 604.00	€ 33.46	€ 570.54	94.46%	95.58%
Seven Seas Imports	€ 9,021.24	€ 425.03	€ 8,596.21	95.29%	95.58%
...	...	...	...	...	...

◆ Profit% computes the profit percentage of the current member, and Profit%City applies Profit% to the parent of the current member, that is, the profit of the state to which the city belongs

Sorting

- All the members in a dimension have a hierarchical order, e.g.:

```
SELECT Measures.MEMBERS ON COLUMNS,
          Customer.Geography.Country.MEMBERS ON ROWS
FROM    Sales
```

	Unit Price	Quantity	Discount	Sales Amount	Freight	Sales Count
Austria	€ 84.77	4,644	21.71 %	€ 115,328.31	€ 6,827.10	114
Belgium	€ 64.65	1,242	9.72 %	€ 30,505.06	€ 1,179.53	49
Denmark	€ 70.28	1,156	17.94 %	€ 32,428.94	€ 1,377.75	45
Finland	€ 54.41	848	9.09 %	€ 17,530.05	€ 827.45	51
France	€ 64.51	3,052	11.76 %	€ 77,056.01	€ 3,991.42	172
...	...	...	...	...	...	...

- Countries are displayed according to the order of the hierarchy: first the European countries, then the North American countries, etc., i.e., according to the ordering of the parent level of country (Area)
- To sort countries by their name, we can use the ORDER function:

  ORDER(Set, Expression [, ASC | DESC | BASC | BDESC])

Sorting

- Sorting the set of countries in the previous query skipping the hierarchy

```
SELECT Measures.MEMBERS ON COLUMNS,
       ORDER(Customer.Geography.Country.MEMBERS,
       Customer.Geography.CURRENTMEMBER.Name,BASC) ON ROWS
FROM   Sales
```

- Name returns the name of a level, dimension, member, or hierarchy
- The answer displays countries in alphabetical order
- Sorting the query result based on the sales amount (a measure)

```
SELECT Measures.MEMBERS ON COLUMNS,
       ORDER(Customer.Geography.Country.MEMBERS,
       Measures.[Sales Amount],BDESC) ON ROWS
FROM   Sales
```

Sorting

- Ordering on multiple criteria: difficult to express in MDX
- *Analyze sales amount by area and category, sort the result first by area name and then by category name*. For this we need to use the GENERATE function as follows

```
SELECT Measures.[Sales Amount] ON COLUMNS,
       NON EMPTY GENERATE(ORDER( Customer.Geography.Area.ALLMEMBERS,
       Customer.Geography.CURRENTMEMBER.NAME, BASC ),
       ORDER( { Customer.Geography.CURRENTMEMBER } *
       Product.Categories.Category.ALLMEMBERS,
       Product.Categories.CURRENTMEMBER.NAME, BASC ) ) ON ROWS
FROM   Sales
```

- The first argument of GENERATE sorts areas in ascending order; the second argument cross joins the current area with the categories sorted in ascending order of their name

		Sales Amount
Europe	Beverages	€ 120,361.83
Europe	Condiments	€ 60,517.12
Europe	Confections	€ 95,690.12
Europe	Dairy Products	€ 137,315.75
Europe	Grains/Cereals	€ 48,781.57
...	...	...

Top and Bottom Analysis

- HEAD and TAIL functions return the first (last) members in the set based on a number
- *Top three best-selling store cities*

```
SELECT Measures.MEMBERS ON COLUMNS,
       HEAD(ORDER(Customer.Geography.City.MEMBERS,
       Measures.[Sales Amount],BDESC),3) ON ROWS
FROM   Sales
```

	Unit Price	Quantity	Discount	Sales Amount	Freight	Sales Count
Cunewalde	€ 101.46	3,616	21.40 %	€ 103,597.43	€ 4,999.77	77
Boise	€ 90.90	4,809	32.41 %	€ 102,253.85	€ 6,570.58	113
Graz	€ 88.00	4,045	23.57 %	€ 93,349.45	€ 5,725.79	92

- Alternatively, TOPCOUNT can be used

```
SELECT Measures.MEMBERS ON COLUMNS,
       TOPCOUNT(Customer.Geography.City.MEMBERS,5,
       Measures.[Sales Amount]) ON ROWS
FROM   Sales
```

Top and Bottom Analysis

- *Top three cities, based on sales count, and how much all the other cities combined have sold*

```
WITH SET SetTop3Cities AS TOPCOUNT(
              Customer.Geography.City.MEMBERS, 3, [Sales Amount])
MEMBER Customer.Geography.[Top 3 Cities] AS
              AGGREGATE(SetTop3Cities)
MEMBER Customer.Geography.[Other Cities] AS
              (Customer.[All]) - (Customer.[Top 3 Cities])
SELECT Measures.MEMBERS ON COLUMNS,
              { SetTop3Cities, [Top 3 Cities], [Other Cities],
              Customer.[All] } ON ROWS
FROM     Sales
```

	Unit Price	Quantity	Discount	Sales Amount	Freight	Sales Count
Cunewalde	€ 101.46	3,616	21.40 %	€ 103,597.43	€ 4,999.77	77
Boise	€ 90.90	4,809	32.41 %	€ 102,253.85	€ 6,570.58	113
Graz	€ 88.00	4,045	23.57 %	€ 93,349.45	€ 5,725.79	92
Top 3 Cities	€ 95.46	12,470	26.69 %	€ 299,200.73	€ 17,296.14	282
Other Cities	€ 38.68	33,918	0.95 %	€ 845,955.13	€ 41,291.35	1,649
All Customers	€ 134.14	46,388	27.64 %	€ 1,145,155.86	€ 58,587.49	1,931

- AGGREGATE aggregates each measure using the default operator specified for each measure
- For measures Unit Price and Discount the average, for the other measures sum

Top and Bottom Analysis

- Other functions for top filter processing: TOPPERCENT and TOPSUM return the top elements whose cumulative total is at least a specified percentage or a specified value, respectively
- *Cities whose sales count accounts for thirty percent of all the sales*

```
SELECT Measures.[Sales Amount] ON COLUMNS,
{ TOPPERCENT(Customer.Geography.City.MEMBERS, 30,
Measures.[Sales Amount]),Customer.Geography.[All] } ON ROWS
FROM Sales
```

	Sales Amount
Cunewalde	€ 103,597.43
Boise	€ 102,253.85
Graz	€ 93,349.45
London	€ 51,169.01
Albuquerque	€ 49,290.08
All Customers	€ 1,145,155.86

- Note: The sum of the sales of the cities in the answer amounts to 34% of the total sales amount
- An analogous series of BOTTOM functions, returning the bottom items in a list
- In the previous query, can use BOTTOMSUM to obtain the bottom cities whose cumulative sales amount is less than € 10,000.

Aggregation Functions

- MDX provides many aggregation functions: SUM, AVG, MEDIAN, MAX, MIN, VAR, and STDDEV
- *Total, maximum, minimum, and average sales amount for a one-month period in 1997*

```
WITH MEMBER Measures.[Maximum Sales] AS
        MAX(DESCENDANTS([Order Date].Calendar.Year.[1997], [Order Date].Calendar.Month),
        Measures.[Sales Amount])
MEMBER Measures.[Minimum Sales] AS
        MIN(DESCENDANTS([Order Date].Calendar.Year.[1997], [Order Date].Calendar.Month),
        Measures.[Sales Amount])
MEMBER Measures.[Average Sales] AS
        AVG(DESCENDANTS([Order Date].Calendar.Year.[1997], [Order Date].Calendar.Month),
        Measures.[Sales Amount])
SELECT { [Sales Amount], [Maximum Sales], [Minimum Sales], [Average Sales] } ON COLUMNS,
        Product.Categories.Category.MEMBERS ON ROWS
FROM    Sales
```

	Sales Amount	Maximum Sales	Minimum Sales	Average Sales
Beverages	€ 237,203.91	€ 21,817.76	€ 2,109.84	€ 7,652.65
Condiments	€ 91,528.81	€ 5,629.70	€ 1,252.33	€ 3,842.09
Confections	€ 162,443.91	€ 11,538.61	€ 2,174.89	€ 6,798.83
Dairy Products	€ 221,157.31	€ 12,992.48	€ 5,584.84	€ 9,119.26
Grains/Cereals	€ 80,870.58	€ 6,012.65	€ 1,891.00	€ 4,193.64
Meat/Poultry	€ 139,428.18	€ 14,110.16	€ 1,029.00	€ 6,217.45
Produce	€ 90,216.14	€ 12,157.90	€ 1,650.00	€ 4,429.52
Seafood	€ 122,307.02	€ 8,448.86	€ 1,587.11	€ 5,263.19

Aggregation Functions

- *Maximum sales by category as well as the month in which the maximum sales*

```
WITH MEMBER Measures.[Maximum Sales] AS
        MAX(DESCENDANTS([Order Date].Calendar.Year.[1997],
        [Order Date].Calendar.Month), Measures.[Sales Amount])
MEMBER Measures.[Maximum Period] AS
        TOPCOUNT(DESCENDANTS([Order Date].Calendar.Year.[1997],
        [Order Date].Calendar.Month), 1,
        Measures.[Sales Amount]).ITEM(0).NAME
SELECT { [Maximum Sales], [Maximum Period] } ON COLUMNS,
        Product.Categories.Category.MEMBERS ON ROWS
FROM    Sales
```

- TOPCOUNT obtains the tuple corresponding to the maximum sales amount; then, ITEM retrieves the first member from the specified tuple, and finally, NAME obtains the name of this member

	Maximum Sales	Maximum Period
Beverages	€ 21,817.76	January 1997
Condiments	€ 5,629.70	December 1997
Confections	€ 11,538.61	April 1997
Dairy Products	€ 12,992.48	November 1997
Grains/Cereals	€ 6,012.65	June 1997
Meat/Poultry	€ 14,110.16	October 1997
Produce	€ 12,157.90	December 1997
Seafood	€ 8,448.86	September 1997

Aggregation Functions

◆ *Maximum sales by category* **and country**, *as well as the month when they occurred*

```
WITH MEMBER Measures.[Maximum Sales] AS
         MAX(DESCENDANTS([Order Date].Calendar.Year.[1997], [Order Date].Calendar.[Month]),
         Measures.[Sales Amount])
MEMBER Measures.[Maximum Period] AS
         TOPCOUNT(DESCENDANTS([Order Date].Calendar.Year.[1997],
         [Order Date].Calendar.[Month]), 1, Measures.[Sales Amount]).ITEM(0).NAME
SELECT { [Maximum Sales], [Maximum Period] } ON COLUMNS,
         Product.Categories.Category.MEMBERS * Customer.Geography.Country.MEMBERS ON ROWS
FROM     Sales
```

		Maximum Sales	Maximum Period
Beverages	Austria	€ 2,149.40	December 1997
Beverages	Belgium	€ 514.08	March 1997
Beverages	Denmark	€ 10,540.00	January 1997
Beverages	Finland	€ 288.00	February 1997
Beverages	France	€ 915.75	December 1997
Beverages	Germany	€ 8,010.00	May 1997
...	...	...	...

Aggregation Functions

- COUNT counts the number of tuples in a set
- Two options: include or exclude empty cells
- Number of customers that purchased a particular product category
- Done by counting the number of tuples obtained by joining the sales amount and customer names
- Must exclude empty cells to count only customers with sales in the corresponding product category

```
WITH MEMBER Measures.[Customer Count] AS
        COUNT({Measures.[Sales Amount] *
        [Customer].[Company Name].MEMBERS}, EXCLUDEEMPTY)
SELECT {Measures.[Sales Amount], [Customer Count]} ON COLUMNS,
        Product.Category.MEMBERS ON ROWS
FROM    Sales
```

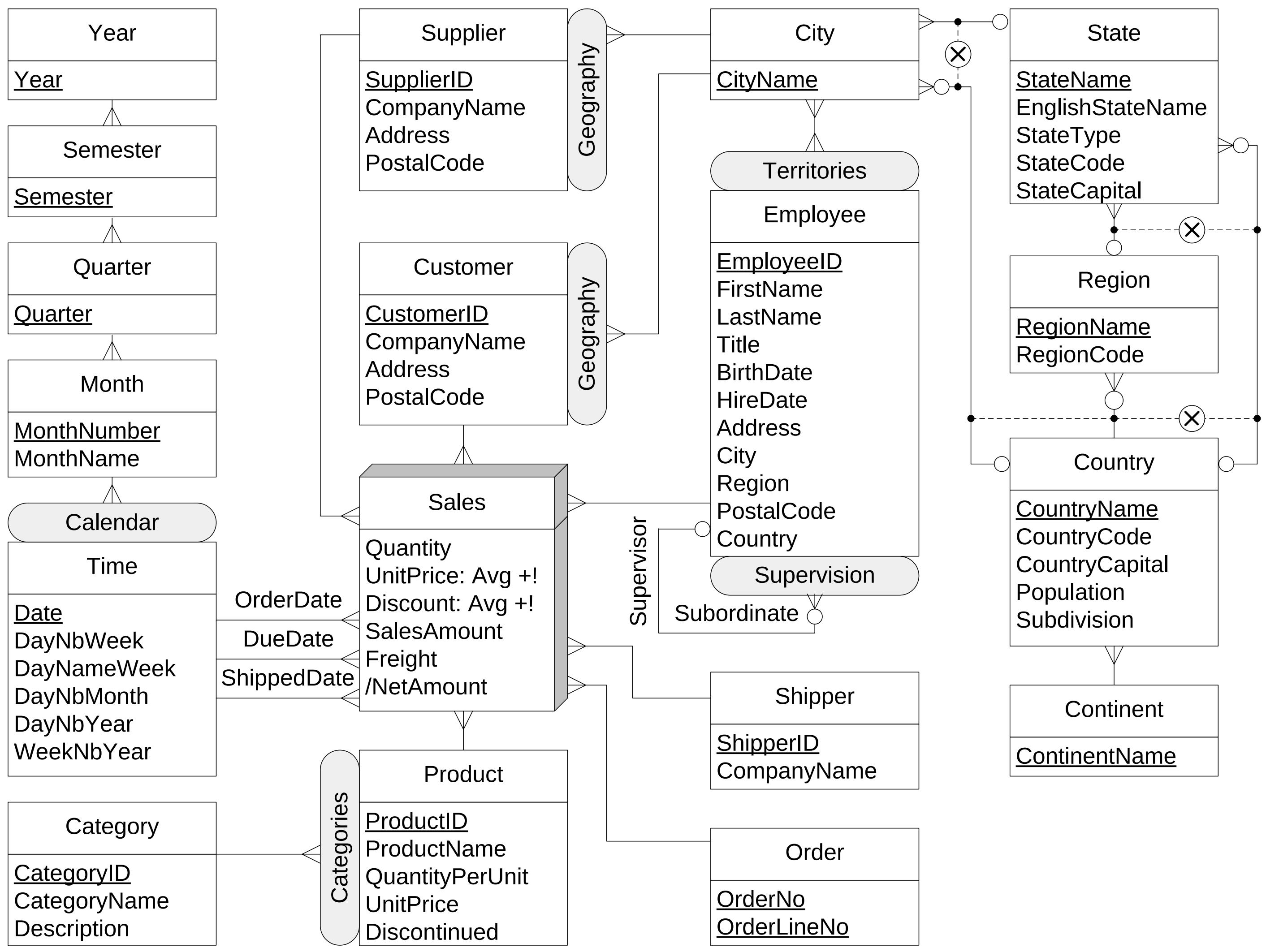
	Sales Amount	Customer Count
Beverages	€ 237,203.91	82
Condiments	€ 91,528.81	65
Confections	€ 162,443.91	79
Dairy Products	€ 221,157.31	80
...	...	...

Chapter 6: Querying the Data Warehouse

Outline

◆ Introduction to the MDX Language
◆ Advanced MDX
➥ **Querying the Northwind Cube in MDX**
◆ Querying the Northwind Cube in SQL
◆ Comparison of MDX and SQL

The Northwind Data Cube: Conceptual Schema



Querying the Northwind Cube in MDX

- **Query 6.1**: *Total sales amount per customer, year, and product category*

```
SELECT [Order Date].Year.CHILDREN ON COLUMNS,
       NON EMPTY Customer.[Company Name].CHILDREN *
       Product.[Category Name].CHILDREN ON ROWS
FROM   Sales
WHERE  Measures.[Sales Amount]
```

- Use CHILDREN instead of MEMBERS to prevent displaying the All members of the three dimensions
- NON EMPTY used to avoid displaying customers that never ordered articles from a particular category

		1996	1997	1998
Alfreds Futterkiste	Beverages	(null)	€ 553.50	(null)
Alfreds Futterkiste	Condiments	(null)	€ 938.00	€ 400.80
Alfreds Futterkiste	Dairy Products	(null)	(null)	€ 1,255.00
Alfreds Futterkiste	Produce	(null)	€ 513.00	€ 91.20
Alfreds Futterkiste	Seafood	(null)	€ 18.00	€ 503.50
...	...	...	...	...

Querying the Northwind Cube in MDX

- **Query 6.3**: *Monthly sales by customer state compared to those of the previous year*

```
WITH MEMBER Measures.[Previous Year] AS
        (Measures.[Sales Amount], PARALLELPERIOD([Order Date].Calendar.Month,12)),
        FORMAT_STRING = '€ ###,##0.00'
SELECT { Measures.[Sales Amount], [Previous Year] } ON COLUMNS,
        NON EMPTY ORDER(Customer.Geography.State.MEMBERS,
        Customer.Geography.CURRENTMEMBER.NAME, BASC) * [Order Date].Calendar.Month.MEMBERS ON R
FROM Sales
```

- A cross join of the Customer and Order Date dimensions displays the states and months rows
- ORDER sorts the states of the customers in alphabetical order irrespective of the Geography hierarchy
- Calculated measure Previous Year computes the sales amount of the same month of the previous year for the current state and month using the PARALLELPERIOD function

		Sales Amount	Previous Year
Alaska	September 1996	€ 3,741.30	
Alaska	October 1996	€ 934.50	
Alaska	February 1997	€ 1,755.00	
Alaska	July 1997	€ 565.50	
Alaska	September 1997	€ 1,261.88	€ 3,741.30
Alaska	October 1997	€ 1,893.00	€ 934.50
Alaska	January 1998	€ 3,638.89	
...	...	...	...

Querying the Northwind Cube in MDX

- **Query 6.7**: *Countries that account for top 50% of sales amount*

```
SELECT Measures.[Sales Amount] ON COLUMNS,
       { Customer.Geography.[All],
       TOPPERCENT([Customer].Geography.Country.MEMBERS,50,
       Measures.[Sales Amount]) } ON ROWS
FROM Sales
```

- TOPPERCENT selects the countries whose cumulative total is equal to the specified percentage

	Sales Amount
All Customers	€ 1,145,155.86
United States	€ 238,490.40
Germany	€ 219,356.08
Austria	€ 115,328.31

Querying the Northwind Cube in MDX

- **Query 6.10**: *Monthly year-to-date sales for each product category.*

```
WITH MEMBER Measures.YTDSales AS
        SUM(PERIODSTODATE([Order Date].Calendar.[Year],
        [Order Date].Calendar.CURRENTMEMBER),
        Measures.[Sales Amount]), FORMAT_STRING = '###,##0.00'
SELECT DESCENDANTS([Order Date].[1996], [Order Date].[Month])
        ON COLUMNS, Product.[Category].MEMBERS ON ROWS
FROM    Sales
WHERE   (Measures.YTDSales)
```

- PERIODSTODATE selects all months of the current year up to the current month
- Then, SUM obtains the year-to-date aggregate value of the measure Sales Amount

	July 1996	August 1996	September 1996	October 1996	···
Beverages	€ 3,182.50	€ 6,577.38	€ 8,996.98	€ 15,700.82	···
Condiments	€ 1,753.40	€ 3,141.70	€ 4,003.30	€ 8,127.62	···
Confections	€ 5,775.15	€ 10,781.92	€ 16,527.92	€ 20,056.52	···
Dairy Products	€ 6,838.34	€ 11,600.04	€ 14,416.04	€ 21,353.59	···
Grains/Cereals	€ 1,158.86	€ 1,429.46	€ 2,159.06	€ 4,530.02	···
Meat/Poultry	€ 2,268.72	€ 5,764.38	€ 10,055.38	€ 13,706.68	···
Produce	€ 3,868.80	€ 4,673.12	€ 5,837.92	€ 6,700.92	···
Seafood	€ 2,400.33	€ 6,383.07	€ 8,936.87	€ 14,748.87	···

Querying the Northwind Data Warehouse in MDX

- **Query 6.15**: *For each employee, total sales amount, number of cities, and number of states to which she is assigned*

```
WITH MEMBER NoCities AS Measures.[Territories Count]
MEMBER NoStates AS
          DISTINCTCOUNT(Employee.[Full Name].CURRENTMEMBER * City.Geography.State.MEMBERS)
SELECT { Measures.[Sales Amount], Measures.NoCities, Measures.NoStates }
          ON COLUMNS, Employee.[Full Name].CHILDREN ON ROWS
FROM      Sales
```

- Many-to-many relationship between employees and cities exploited through bridge table Territories
- The Territories Count measure is automatically added when the cube is created
- We rename this measure as NoCities
- DISTINCTCOUNT computes the number of states for the employee

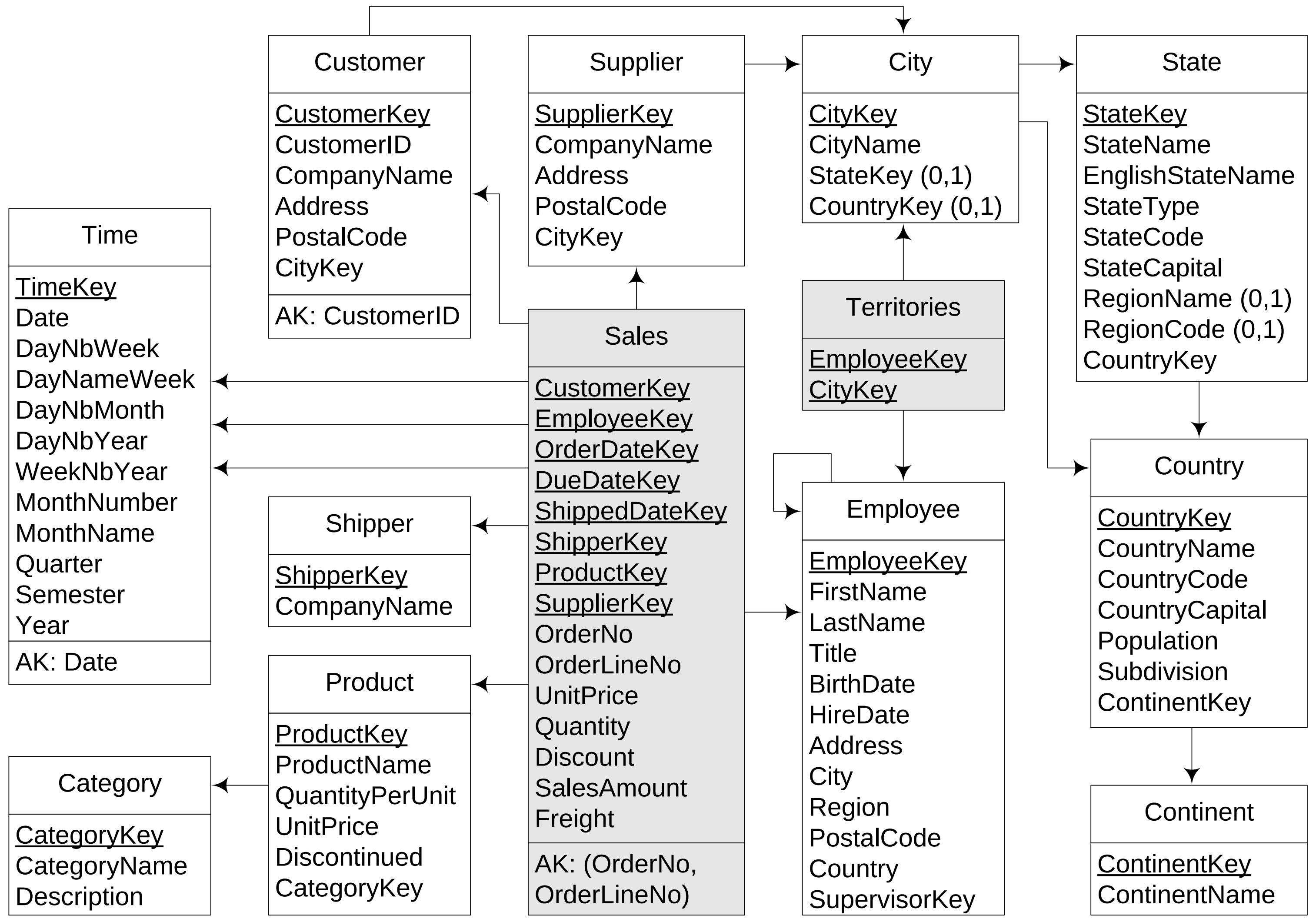
FullName	Sales Amount	NoCities	NoStates
Andrew Fuller	$152,164.80	6	3
Anne Dodsworth	$69,046.17	7	5
Janet Leverling	$186,197.80	4	2
Laura Callahan	$122,528.86	4	3
...	...	...	...

Chapter 6: Querying the Data Warehouse

Outline

- Introduction to the MDX Language
- Advanced MDX
- Querying the Northwind Cube in MDX
- ➡ **Querying the Northwind Cube in SQL**
- Comparison of MDX and SQL

Schema of the Northwind Data Warehouse



Querying the Northwind Data Warehouse in SQL

- **Query 6.1**: *Total sales amount per customer, year, and product category*

```
SELECT    C.CompanyName, T.Year, A.CategoryName,
          FORMAT(SUM(SalesAmount),'€ ###,##0.00') AS SalesAmount
FROM      Sales S, Customer C, Time T, Product P, Category A
WHERE     S.CustomerKey = C.CustomerKey AND
          S.OrderDateKey = T.TimeKey AND
          S.ProductKey = P.ProductKey AND
          P.CategoryKey = A.CategoryKey
GROUP BY C.CompanyName, T.Year, A.CategoryName
```

- We join the fact tables with the involved dimension tables, and aggregate the results by company, year, and category
- The FORMAT function is used to format the aggregated measure

CompanyName	Year	Category	SalesAmount
Alfreds Futterkiste	1997	Beverages	€ 553.50
Alfreds Futterkiste	1997	Condiments	€ 938.00
Alfreds Futterkiste	1997	Produce	€ 513.00
Alfreds Futterkiste	1997	Seafood	€ 18.00
Alfreds Futterkiste	1998	Condiments	€ 400.80
...	...	...	...

Querying the Northwind Data Warehouse in SQL

- **Query 6.3**: *Monthly sales by customer state compared to those of the previous year*

```
CREATE FUNCTION MonthYear (@Month INT, @Year INT) RETURNS CHAR(14) AS
BEGIN
        DECLARE @Date CHAR(10);
        SET @Date = CAST(@Year AS CHAR(4)) + '-' + CAST(@Month AS CHAR(2)) + '-' + '01';
        RETURN(Datename(month,@Date) + ' ' + CAST(@Year AS CHAR(4)));
END
WITH MonthlySalesState AS (
        SELECT  S.StateName, T.MonthNumber, T.Year, SUM(SalesAmount) AS SalesAmount
        FROM    Sales F, Customer C, City Y, State S, Time T
        WHERE   F.CustomerKey = C.CustomerKey AND C.CityKey = Y.CityKey AND
                Y.StateKey = S.StateKey AND F.OrderDateKey = T.TimeKey
        GROUP BY S.StateName, T.Year, T.MonthNumber )
SELECT  M1.StateName, dbo.MonthYear(M1.MonthNumber,M1.Year) AS Month,
        FORMAT(M1.SalesAmount,'€ ###,##0.00'),
        FORMAT(M2.SalesAmount,'€ ###,##0.00') AS PreviousYear
FROM    MonthlySalesState M1 LEFT OUTER JOIN MonthlySalesState M2 ON
        M1.StateName = M2.StateName AND M1.MonthNumber = M2.MonthNumber AND
        M1.Year-1 = M2.Year
ORDER BY M1.StateName, Month
```

- MonthYear concatenates a month and a year for a more user-friendly display
- In the WITH clause we define a common table expression to compute the monthly sales by state
- The temporary table is joined twice to obtain the sales of a month and the previous month, and a left outer join is used to display a null value in case there are no sales for the previous month

Querying the Northwind Data Warehouse in SQL

- **Query 6.7**: *Countries that account for top 50% of sales amount.*

```
WITH SalesCountry AS (
            SELECT    CountryName, SUM(SalesAmount) AS SalesAmount
            FROM      Sales S, Customer C, City Y, State T, Country O
            WHERE     S.CustomerKey = C.CustomerKey AND C.CityKey = Y.CityKey AND
                      Y.StateKey = T.StateKey AND T.CountryKey = O.CountryKey
            GROUP BY CountryName ),
        CumSalesCountry AS (
            SELECT    S.*, SUM(SalesAmount) OVER (ORDER BY SalesAmount DESC
                      ROWS UNBOUNDED PRECEDING) AS CumSalesAmount
            FROM      SalesCountry S )
SELECT 'All Customers' AS CountryName, FORMAT(SUM(SalesAmount), '€ ###,##0.00') AS SalesAmount
FROM    SalesCountry
UNION
SELECT CountryName, FORMAT(SalesAmount, '€ ###,##0.00') AS SalesAmount
FROM    CumSalesCountry WHERE CumSalesAmount <=
        (SELECT MIN(CumSalesAmount) FROM CumSalesCountry
        WHERE CumSalesAmount >= (SELECT 0.5 * SUM(SalesAmount) FROM SalesCountry) ) )
```

- SalesCountry aggregates the sales amount by country
- For each row in SalesCountry, in CumSalesCountry we define a window with rows sorted in decreasing value of sales amount, and sum the current row and all the preceding rows in the window
- Finally, in the main query we select the countries which satisfy the query condition

Querying the Northwind Data Warehouse in SQL

- **Query 6.10**: *Monthly year-to-date sales for each product category*

```
WITH SalesByCategoryMonth AS (
            SELECT      CategoryName, Year, MonthNumber, MonthName,
                        SUM(SalesAmount) AS SalesAmount
            FROM        Sales S, Product P, Category C, Time T
            WHERE       S.OrderDateKey = T.TimeKey AND S.ProductKey = P.ProductKey AND
                        P.CategoryKey = C.CategoryKey
            GROUP BY CategoryName, Year, MonthNumber, MonthName )
SELECT      CategoryName, MonthName + ' ' + CAST(Year AS CHAR(4)) AS Month,
            FORMAT(SUM(SalesAmount) OVER (PARTITION BY CategoryName, Year
            ORDER BY MonthNumber ROWS UNBOUNDED PRECEDING), '€ ###,##0.00')
            AS YTDSalesAmount
FROM        SalesByCategoryMonth
ORDER BY CategoryName, Year, MonthNumber
```

- In the temporary table we aggregate the sales amount by category and month
- In the main query, for each row in the temporary table, we define a window containing all the rows with the same category and year, sort the rows in the window by month, and compute the sum of the current row and all the preceding rows

CategoryName	MonthName	YTDSales
Beverages	July 1996	€ 3,182.50
Beverages	August 1996	€ 6,577.38
Beverages	September 1996	€ 8,996.98
Beverages	October 1996	€ 15,700.82
...	...	...

Querying the Northwind Data Warehouse in SQL

- **Query 6.15**: *For each employee, total sales amount, number of cities, and number of states to which she is assigned*

```
SELECT FirstName + ' ' + LastName AS FullName,
          FORMAT(SUM(SalesAmount) / COUNT(DISTINCT CityName), '$###,##0.00') AS TotalSales,
          COUNT(DISTINCT CityName) AS NoCities, COUNT(DISTINCT StateName) AS NoStates
FROM     Sales F, Employee E, Territories T, City C, State S
WHERE  F.EmployeeKey = E.EmployeeKey AND E.EmployeeKey = T.EmployeeKey AND
          T.CityKey = C.CityKey AND C.StateKey = S.StateKey
GROUP BY FirstName + ' ' + LastName
ORDER BY FirstName + ' ' + LastName
```

- Territories captures the many-to-many relationship between employees and cities
- The query makes the join of the five tables and then groups the result by employee.
- In the SELECT clause we sum the SalesAmount measure and divide it by the number of distinct CityName assigned to an employee in the Territories table
- This solves the double-counting problem

Outline

- Introduction to the MDX Language
- Advanced MDX
- Querying the Northwind Cube in MDX
- Querying the Northwind Cube in SQL
- **Comparison of MDX and SQL**

Comparison of MDX and SQL

- Main difference between SQL and MDX: Ability of MDX to reference multiple dimensions
- Although possible to use SQL to query cubes, MDX's commands are designed specifically for multi-dimensional data
- SQL refers to only two dimensions, columns and rows
- OLAP tools: Cannot display a result set with more than two dimensions.
- SQL's SELECT clause defines the column layout for a query
- However, MDX's SELECT clause is used to define several axis dimensions.
- SQL's WHERE clause used to *filter* the data returned by a query
- MDX's WHERE clause used to provide a *slice* of the data returned by a query

Comparison Between MDX and SQL

- **Query 6.1**
  - In SQL, joins must be explicitly indicated in the query whereas they are implicit in MDX
  - In SQL, inner join removes empty combinations; in MDX, NON EMPTY must be specified
  - Outer joins are needed in SQL if we want to show empty combinations.
- **Query 6.3**
  - Comparison of measures of the current period with respect to those of a previous period
  - In MDX this can be achieved with calculated members using the WITH MEMBER clause In SQL we define a temporary table in the WITH clause in which the aggregations needed for the roll-up operation are performed for each period
  - An outer join is needed in the main query for obtaining the measure of the current period together with that of a previous period
- **Query 6.15**
  - An example of manipulating many-to-many dimensions
  - The SQL version needs to deal with the double-counting problem while aggregating the measure

Comparison Between MDX and SQL

- Summary of the advantages and disadvantages of both languages

MDX	SQL
Advantages ◆ Data modeling: definition of dimensions, hierarchies, measure groups, from various data sources ◆ Simple navigation within time dimension and hierarchies ◆ Relatively simple expressions for often used business requests ◆ Fast, due to existence of aggregations	**Advantages** ◆ Large user base ◆ Easy-to-understand semantics of queries ◆ Results are easy to visualize: scalars or 2D tables ◆ Various ways of relating tables: joins, derived tables, correlated queries, common table expressions, etc.
Disadvantages ◆ Extra effort for designing a cube and setting up aggregations ◆ Steep learning curve: manipulating an n-dimensional space ◆ Hard-to-grasp concepts: current context, execution phases, etc. ◆ Some operations are difficult to express, such as ordering on multiple criteria	**Disadvantages** ◆ Tables must be joined explicitly inside a query ◆ Sometimes not intuitive and complex syntax for expressing analytical queries ◆ No concept of row ordering and hierarchies: navigation dimensions may be complex ◆ Not so performant for the types of queries used in data analysis