



Diseño Lógico de Almacenes de Datos

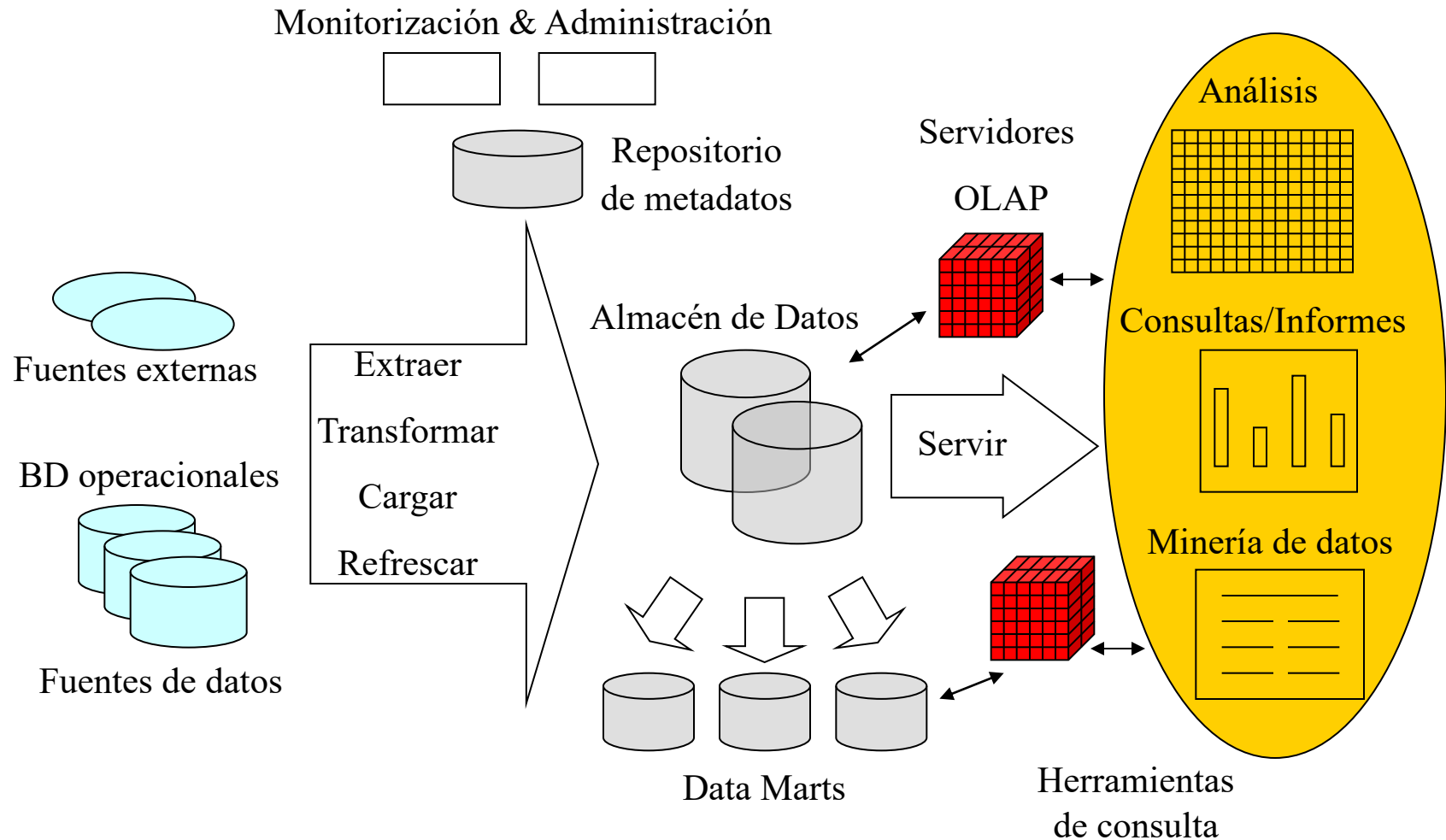
Tema 6. Consultas



Indice

- Operaciones SQL/OLAP
- Consultas MDX

Esquema de una arquitectura de DW



Operaciones SQL/OLAP

- El Cubo de datos en el modelo relacional
- BDR no son la mejor estructura para los datos multidimensionales
- Por ejemplo. Cubo SALES con dimensiones Product y Customer y una medida SalesAmount
- El cubo contiene todas (2^2) las agregaciones posibles. SalesAmount por cliente, por producto y por cliente y producto, junto con los datos no agregados

A data cube with two dimensions

| | c1 | c2 | c3 | TotalBy Product |
|---------------------|-----|-----|-----|--------------------|
| p1 | 100 | 105 | 100 | 305 |
| p2 | 70 | 60 | 40 | 170 |
| p3 | 30 | 40 | 50 | 120 |
| TotalBy Customer | 200 | 205 | 190 | 595 |

A relational fact table representing the same data

| ProductKey | CustomerKey | SalesAmount |
|------------|-------------|-------------|
| p1 | c1 | 100 |
| p1 | c2 | 105 |
| p1 | c3 | 100 |
| p2 | c1 | 70 |
| p2 | c2 | 60 |
| p2 | c3 | 40 |
| p3 | c1 | 30 |
| p3 | c2 | 40 |
| p3 | c3 | 50 |



Operaciones SQL/OLAP

- En la tabla de hechos SALES, para calcular todas las posibles agregaciones sobre Product y Customer hay que recorrer la relación completa

```
SELECT ProductKey, CustomerKey, SalesAmount
```

```
FROM Sales
```

```
UNION
```

```
SELECT ProductKey, NULL, SUM(SalesAmount)
```

```
FROM Sales
```

```
GROUP BY ProductKey
```

```
UNION
```

```
SELECT NULL, CustomerKey, SUM(SalesAmount)
```

```
FROM Sales
```

```
GROUP BY CustomerKey
```

```
UNION
```

```
SELECT NULL, NULL, SUM(SalesAmount)
```

```
FROM Sales
```

Data cube

| ProductKey | CustomerKey | SalesAmount |
|------------|-------------|-------------|
| p1 | c1 | 100 |
| p2 | c1 | 70 |
| p3 | c1 | 30 |
| NULL | c1 | 200 |
| p1 | c2 | 105 |
| p2 | c2 | 60 |
| p3 | c2 | 40 |
| NULL | c2 | 205 |
| p1 | c3 | 100 |
| p2 | c3 | 40 |
| p3 | c3 | 50 |
| NULL | c3 | 190 |
| p1 | NULL | 305 |
| p2 | NULL | 170 |
| p3 | NULL | 120 |
| NULL | NULL | 595 |



Operaciones SQL/OLAP

- Con n dimensiones, tenemos 2^n GROUP BY -> no muy eficiente
- SQL/OLAP extiende GROUP BY con los operadores ROLLUP y CUBE
- ROLLUP calcula subtotales agrupados en el orden dado por una lista de atributos
- CUBE computa todos los totales de dicha lista

```
SELECT ProductKey, CustomerKey, SUM(SalesAmount)
FROM Sales
GROUP BY ROLLUP(ProductKey, CustomerKey)
```

GROUP BY ROLLUP

| ProductKey | CustomerKey | SalesAmount |
|------------|-------------|-------------|
| p1 | c1 | 100 |
| p1 | c2 | 105 |
| p1 | c3 | 100 |
| p1 | NULL | 305 |
| p2 | c1 | 70 |
| p2 | c2 | 60 |
| p2 | c3 | 40 |
| p2 | NULL | 170 |
| p3 | c1 | 30 |
| p3 | c2 | 40 |
| p3 | c3 | 50 |
| p3 | NULL | 120 |
| NULL | NULL | 595 |



Operaciones SQL/OLAP

```
SELECT ProductKey, CustomerKey, SUM(SalesAmount)
FROM Sales
GROUP BY CUBE (ProductKey, CustomerKey)
```

GROUP BY CUBE

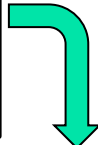
| ProductKey | CustomerKey | SalesAmount |
|------------|-------------|-------------|
| p1 | c1 | 100 |
| p2 | c1 | 70 |
| p3 | c1 | 30 |
| NULL | c1 | 200 |
| p1 | c2 | 105 |
| p2 | c2 | 60 |
| p3 | c2 | 40 |
| NULL | c2 | 205 |
| p1 | c3 | 100 |
| p2 | c3 | 40 |
| p3 | c3 | 50 |
| NULL | c3 | 190 |
| NULL | NULL | 595 |
| p1 | NULL | 305 |
| p2 | NULL | 170 |
| p3 | NULL | 120 |



Operaciones SQL/OLAP

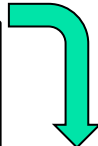
- Con n dimensiones, tenemos 2ⁿ GROUP BY -> no muy eficiente
- SQL/OLAP extiende GROUP BY con los operadores ROLLUP y CUBE
- ROLLUP calcula subtotales agrupados en el orden dado por una lista de atributos
- CUBE computa todos los totales de dicha lista
- Son abreviaturas del operador GROUPING SETS

```
SELECT ProductKey, CustomerKey, SUM(SalesAmount)
FROM Sales
GROUP BY ROLLUP(ProductKey, CustomerKey)
```



```
SELECT ProductKey, CustomerKey, SUM(SalesAmount)
FROM Sales
GROUP BY GROUPING SETS((ProductKey, CustomerKey), (ProductKey), ())
```

```
SELECT ProductKey, CustomerKey, SUM(SalesAmount)
FROM Sales
GROUP BY CUBE(ProductKey, CustomerKey)
```



```
SELECT ProductKey, CustomerKey, SUM(SalesAmount)
FROM Sales
GROUP BY GROUPING SETS((ProductKey, CustomerKey), (ProductKey), (CustomerKey), ())
```


Operaciones SQL/OLAP

- OJO! El uso de ROLLAP o CUBE en las consultas sólo será necesario si necesitamos todas las agregaciones posibles (2^n)
- En otro caso, utilizamos SQL normal para quedarnos con la agregación que nos interese

GROUP BY CUBE

| ProductKey | CustomerKey | SalesAmount |
|------------|-------------|-------------|
| p1 | c1 | 100 |
| p2 | c1 | 70 |
| p3 | c1 | 30 |
| NULL | c1 | 200 |
| p1 | c2 | 105 |
| p2 | c2 | 60 |
| p3 | c2 | 40 |
| NULL | c2 | 205 |
| p1 | c3 | 100 |
| p2 | c3 | 40 |
| p3 | c3 | 50 |
| NULL | c3 | 190 |
| NULL | NULL | 595 |
| p1 | NULL | 305 |
| p2 | NULL | 170 |
| p3 | NULL | 120 |

```
SELECT SUM(SalesAmount)
FROM Sales
```

```
SELECT ProductKey, SUM(SalesAmount)
FROM Sales
GROUP BY ProductKey
```



Operaciones SQL/OLAP: Window Partitioning

- Permite Comparar datos detallados con valores agregados
- Ejemplo: relevancia de cada cliente con respect a las ventas del producto

```
SELECT ProductKey, CustomerKey, SalesAmount,  
MAX(SalesAmount) OVER (PARTITION BY ProductKey) AS MaxAmount  
FROM Sales
```

- Las primeras 3 columnas se obtienen de la table Sales
- La cuarta:
 - Por cada fila se define una ventana llamada **partición** que contiene todas las filas del mismo producto
- SalesAmount se agrega sobre esta ventana usando la función MAX

| ProductKey | CustomerKey | SalesAmount | MaxAmount |
|------------|-------------|-------------|-----------|
| p1 | c1 | 100 | 105 |
| p1 | c2 | 105 | 105 |
| p1 | c3 | 100 | 105 |
| p2 | c1 | 70 | 70 |
| p2 | c2 | 60 | 70 |
| p2 | c3 | 40 | 70 |
| p3 | c1 | 30 | 50 |
| p3 | c2 | 40 | 50 |
| p3 | c3 | 50 | 50 |



Operaciones SQL/OLAP: Window Ordering

- Permite ordenar los datos dentro de una partición
- Útil para calcular rankings, con las funciones ROW_NUMBER y RANK
- Ejemplo: Cual es la posición en las ventas de cada cliente

```
SELECT ProductKey, CustomerKey, SalesAmount, ROW_NUMBER() OVER  
(PARTITION BY CustomerKey ORDER BY SalesAmount DESC) AS RowNo FROM Sales
```

- La primera fila se evalúa abriendo una ventana con todas las filas del cliente c1, ordenado por las ventas
- El producto p1 es el más demandado por el cliente c1

| Product Key | Customer Key | Sales Amount | RowNo |
|-------------|--------------|--------------|-------|
| p1 | c1 | 100 | 1 |
| p2 | c1 | 70 | 2 |
| p3 | c1 | 30 | 3 |
| p1 | c2 | 105 | 1 |
| p2 | c2 | 60 | 2 |
| p3 | c2 | 40 | 3 |
| p1 | c3 | 100 | 1 |
| p3 | c3 | 50 | 2 |
| p2 | c3 | 40 | 3 |

Operaciones SQL/OLAP: Window Framing

- Define el tamaño de la partición
- Utilizado para calcular funciones estadísticas sobre series temporales
- Ejemplo: Promedio variable de ventas por product

```
SELECT ProductKey, Year, Month, SalesAmount, AVG(SalesAmount) OVER  
(PARTITION BY ProductKey ORDER BY Year, Month ROWS 2 PRECEDING) AS MovAvg  
FROM Sales
```

- Por cada fila, se abre una ventana con las filas pertinentes al product actual
- Entonces, ordena la ventana por Año y Mes y calcula la media sobre la fila anterior y las 2 precedents, si existen.

¿Funciona en nuestro almacén
NorthwindDW?

¿Qué tengo que hacer para
corregirlo?

¿Detectas algún problema más?

| Product Key | Year | Month | Sales Amount | MovAvg |
|-------------|------|-------|--------------|--------|
| p1 | 2011 | 10 | 100 | 100 |
| p1 | 2011 | 11 | 105 | 102.5 |
| p1 | 2011 | 12 | 100 | 101.67 |
| p2 | 2011 | 12 | 60 | 60 |
| p2 | 2012 | 1 | 40 | 50 |
| p2 | 2012 | 2 | 70 | 56.67 |
| p3 | 2012 | 1 | 30 | 30 |
| p3 | 2012 | 2 | 50 | 40 |
| p3 | 2012 | 3 | 40 | 40 |

Operaciones SQL/OLAP: Window Framing

- Ejemplo: Suma de ventas por product Year-to-date

```
SELECT ProductKey, Year, Month, SalesAmount, SUM(SalesAmount) OVER (PARTITION  
BY ProductKey, Year ORDER BY Month ROWS UNBOUNDED PRECEDING) AS YTD  
FROM Sales
```

- Por cada fila, se abre una ventana con las filas de cada product y año ordenado por mes
- SUM se aplica a todas las filas anteriores a la actual ROWS UNBOUNDED PRECEDING

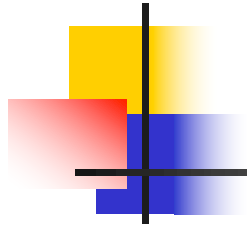
| Product Key | Year | Month | Sales Amount | YTD |
|-------------|------|-------|--------------|-----|
| p1 | 2011 | 10 | 100 | 100 |
| p1 | 2011 | 11 | 105 | 205 |
| p1 | 2011 | 12 | 100 | 305 |
| p2 | 2011 | 12 | 60 | 60 |
| p2 | 2012 | 1 | 40 | 40 |
| p2 | 2012 | 2 | 70 | 110 |
| p3 | 2012 | 1 | 30 | 30 |
| p3 | 2012 | 2 | 50 | 80 |
| p3 | 2012 | 3 | 40 | 120 |

Ejercicio: Corregir la consulta anterior para que muestre una tupla por combinación de ProductKey, Year, Month



Ejercicio

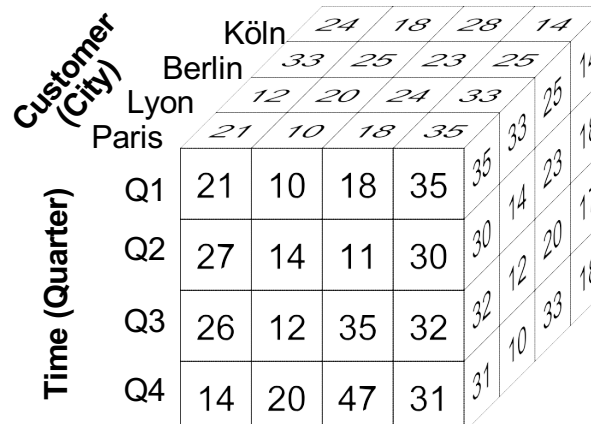
- Realizar las **cinco** consultas en SQL/OLAP propuestas en el CV sobre el almacén de datos NorthwindDW



Consultas **MDX**

Introducción al lenguaje MDX

- ◆ Un cubo de tres dimensiones simple con una medida



| Time (Quarter) | Customer (City) | Product (Category) | | | |
|----------------|-----------------|--------------------|---------|------------|---------|
| | | Beverages | Produce | Condiments | Seafood |
| Q1 | Köln | 24 | 18 | 28 | 14 |
| | Berlin | 33 | 25 | 23 | 25 |
| | Lyon | 12 | 20 | 24 | 33 |
| | Paris | 21 | 10 | 18 | 35 |
| Q2 | Köln | 27 | 14 | 11 | 30 |
| | Berlin | 26 | 12 | 35 | 32 |
| | Lyon | 14 | 20 | 47 | 31 |
| | Paris | 21 | 10 | 18 | 35 |
| Q3 | Köln | 24 | 18 | 28 | 14 |
| | Berlin | 33 | 25 | 23 | 25 |
| | Lyon | 12 | 20 | 24 | 33 |
| | Paris | 21 | 10 | 18 | 35 |
| Q4 | Köln | 24 | 18 | 28 | 14 |
| | Berlin | 33 | 25 | 23 | 25 |
| | Lyon | 12 | 20 | 24 | 33 |
| | Paris | 21 | 10 | 18 | 35 |

- ◆ Dos conceptos fundamentales en MDX: **tuplas y conjuntos**
 - Una tupla identifica una celda simple en un cubo multidimensional
 - Una tupla es definida mediante la declaración de un miembro de una o varias dimensiones del cubo
- ◆ Para identificar la celda en la esquina superior izquierda (valor 21) se proporcionan los valores por cada dimensión:
(Product.Category.Beverages, Time.Quarter.Q1, Customer.City.Paris)



Introducción al lenguaje MDX

Tuplas

- Varias formas de especificar un miembro de una dimensión
- El orden de los miembros no es significativo; estas dos formas son equivalentes: (Product.Category.Beverages, Time.Quarter.Q1, Customer.City.Paris) (Time.Quarter.Q1, Product.Category.Beverages, Customer.City.Paris)
- Debido a que una tupla apunta a una celda simple, cada miembro en una tupla debe pertenecer a una dimensión diferente
- Una tupla no necesita especificar un miembro para cada dimensión: (Customer.City.Paris)
- El ejemplo siguiente indica las ventas de bebidas en Paris (Customer.City.Paris, Product.Category.Beverages)
- Si un miembro para una dimensión no se especifica, se utiliza el miembro por defecto para esa dimensión (típicamente All)

◆ Conjuntos

- Una colección de tuplas definidas usando las mismas dimensiones { (Product.Category.Beverages, Time.Quarter.Q1, Customer.City.Paris), (Product.Category.Beverages, Time.Quarter.Q1, Customer.City.Lyon) }
- Este conjunto agrupa la celda con valor 21 y la que está detrás con valor 12



Introducción al lenguaje MDX

◆ Tuplas y jerarquías

- Jerarquía en la dimensión **Customer**: **Customer** → **City** → **State** → **Country**
- Esta tupla señala a la celda correspondiente al total de ventas de bebidas en Francia durante el primer trimestre:
 - * (**Customer.Country.France**, **Product.Category.Beverages**, **Time.Quarter.Q1**)

◆ Medidas

- En MDX las medidas actúan como dimensiones
- Si hay tres medidas en nuestro cubo: **UnitPrice**, **Discount**, y **SalesAmount**, entonces:
 - * La dimensión **Measures** (existe en cada cubo), contiene tres miembros
- Se puede especificar la medida que queramos como en la siguiente tupla: (**Customer.Country.France**, **Product.Category.Beverages**, **Time.Quarter.Q1**, **Measures.SalesAmount**)
- Si una medida no se especifica se toma la de por defecto.



Consultas Básicas de MDX

◆ La sintaxis de una consulta típica de MDX es:

```
SELECT ( axis specification )  
FROM   ( cube )  
[ WHERE ( slicer specification ) ]
```

◆ MDX se parece a SQL, pero difiere en muchas partes:

- La **especificación de los ejes** permite indicar los ejes de una consulta además de los miembros seleccionados para cada uno de estos ejes
- Es posible contar con hasta 128 ejes en una consulta MDX: cada eje tiene un número: 0 para el eje x, 1 para el eje y, 2 para el eje z, ...
- Los primeros ejes tienen nombres predefinidos: **COLUMNS**, **ROWS**, **PAGES**, **CHAPTERS**, y **SECTIONS**; los ejes en una consulta no pueden saltarse: por ejemplo, una consulta no puede tener un eje ROWS sin tener un eje COLUMNS
- La especificación del **slicer** en la cláusula **WHERE** es opcional; si no se especifica la consulta devuelve la medida por defecto para el cubo



Consultas Básicas de MDX

- La forma más simple de una especificación de un eje: tomar los miembros de una dimensión requerida, incluyendo aquellos de la dimensión especial **Measures**
- ◆ *Mostrar todas las medidas para los clientes a nivel de país*

```
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

- ◆ *Mostrar todas las medidas por año*

```
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 |

- ◆ Para restringir el resultado a Bélgica, podemos escribir

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

- ◆ La consulta anterior sólo cambia los valores devueltos para cada celda



Slicing

- Se añaden múltiples miembros de jerarquías diferentes
- ◆ *Ej. Todas las medidas por año para los clientes de Bélgica que compren productos cuya categoría sea bebidas*

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

- ◆ Para múltiples miembros de la **misma jerarquía** necesitamos incluir un set
- ◆ Valores agregados para Bélgica y Francia en cada celda

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

- ◆ Un conjunto en el **WHERE** agrega implícitamente valores para todos los miembros en el conjunto



Slicing

- ◆ Especificar en la cláusula **WHERE** la medida a mostrar

```
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 |
| ... | ... | ... | ... | ... |

- ◆ Medidas y dimensiones en la cláusula **WHERE**

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



Navegación

- La consulta anterior contiene valores agregados para todos los años incluyendo la columna **All**
- ◆ Para omitir el miembro **All** debemos utilizar la función **CHILDREN**
`SELECT [Order Date].Year.CHILDREN ON COLUMNS, ...`
- ◆ Sin embargo, las filas no incluyen **All**
- ◆ Razón: **Customer.Country.MEMBERS** is la abreviatura de **Customer.Geography.Country.MEMBERS**
 - Así, se seleccionan los miembros de **Country**
 - **All** es el nivel más alto de la jerarquía, no un miembro del nivel **Country**
 - Por tanto, no aparecen en los resultados
- ◆ **Customer** tiene la jerarquía **Company Name**; si se usa la expresión:
`Customer.[Company Name].MEMBERS`
El resultado contendrá el miembro **All**, más los nombres de todos los clientes

Navegación

Cantidad de ventas de los clientes de Francia e Italia por año

```
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-Rhone | € 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 |

- ◆ Nótese otra vez, **All** en columnas (no incluimos **CHILDREN**), no en filas (incluimos **CHILDREN**)



Navegación

- ◆ Para hacer drill-down necesitamos la función **DESCENDANTS**
- ◆ *Ventas totales de las ciudades alemanas*
- ◆ **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 |
| ... | ... | ... | ... | ... |

- ◆ Por defecto, **DESCENDANTS** muestra solo los miembros del nivel especificado en el segundo atributo
- ◆ Un parámetro *opciones* como tercer argumento permite incluir o excluir descendientes o hijos de antes o después del nivel especificado



Navegación: Funciones de calificación

- ◆ **SELF**: función por defecto, muestra los valores para el nivel que se indica en el segundo parámetro (*ciudad* en el ejemplo anterior)
- ◆ **BEFORE**: muestra valores para el nivel estado superior al nivel país
- ◆ **SELF_AND_BEFORE**: muestra valores para el nivel **City** hasta el nivel **Country**
- ◆ **AFTER**: muestra valores para el nivel **Customer**, ya que es sólo si nivel después de **City**
- ◆ **SELF_AND_AFTER**: muestra valores para los niveles **City** y **Customer**
- ◆ **BEFORE_AND_AFTER**: muestra valores desde el nivel **Country** al nivel **Customer**, excluyendo **City**
- ◆ **SELF_BEFORE_AFTER**: muestra valores desde el nivel **Country** al nivel **Customer**
- ◆ **LEAVES**: muestra valores para el nivel **City**, ya que es la única hoja entre los niveles **Country** y **City**
- ◆ Si **LEAVES** se usa sin especificar el nivel como en
`DESCENDANTS(Customer.Geography.Germany, ,LEAVES)`
Se mostrará el nivel hoja de la jerarquía (i.e., **Customer**)



Navegación

- ◆ **ASCENDANTS** devuelve un conjunto que incluye todos los antecesores de un miembro y el propio miembro

- ◆ *Total de ventas para un cliente en particular y todos sus antecesores*

```
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 |

- ◆ Para obtener el resultado para un ancestro de un nivel específico, se puede utilizar la función **ANCESTOR**:

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



Cross Join

- Combina varias dimensiones en un eje simple para mostrar más de dos ejes
- ◆ *Cantidad de ventas por categoría de producto, país y trimestre (cliente y tiempo combinado en el mismo eje)*

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

- ◆ Es posible usar '*'

```
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

Más de dos cross join

```
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 | ... |
| ... | ... | ... | ... | ... | ... | ... |



Subconsultas

- ◆ La cláusula **WHERE** aplica un *slice* al cubo

- No sólo aplicado para seleccionar la medida a mostrar si no también para las dimensiones

- ◆ *Ventas de las categorías de bebidas y condimentos por trimestre*

- ◆ Consulta clásica **WHERE**

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

- ◆ Alternativa con **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 )
```

- ◆ A diferencia de SQL, en la consulta más externa podemos acceder a atributos **no seleccionados** en una subconsulta

- ◆ **Diferencia clave:**

- Si la jerarquía **Category** está en la cláusula **WHERE**, ésta no puede aparecer en cualquier eje
- No ocurre igual en el caso de la subconsulta (si se puede utilizar en un eje)



Subconsultas

◆ Diferencia entre las dos propuestas anteriores:

```
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 |
| ... | ... | ... |

- ◆ Los miembros de la jerarquía **Category** ahora son sólo las categorías de bebidas y condimentos y no el resto, pues se han seleccionado expresamente las anteriores

Subconsultas

- ◆ No hay restricción a una sola dimensión en las subconsultas:

```
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 |

- ◆ Anidamiento múltiple

```
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** ordena un conjunto en orden descendente con respecto a la expresión dada como tercer parámetro, devolviendo el número específico de elementos con los valores más altos

Miembros calculados y conjuntos con nombre

- ◆ **Los miembros calculados** definen nuevos miembros en una dimensión, o nuevas medidas calculadas en tiempo de ejecución

WITH MEMBER Parent.MemberName AS (expression)

- ◆ **Los *named sets*** definen nuevos conjuntos

- ◆ WITH SET SetName AS (expression)

- ◆ Ejemplo: Juna medida que calcula el porcentaje de beneficio en las ventas

```
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% |
| ... | ... | ... | ... |

Conjuntos estáticos y dinámicos

- ◆ **Conjunto estático:** Nordic Countries se componen de Dinamarca, Finlandia, Noruega y Suecia

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 |
| | € 54.41 | 848 | 9.09 % | € 17,530.05 | € 827.45 | 51 |
| Finland | € 97.95 | 152 | 0.00 % | € 5,321.15 | € 257.45 | 15 |
| Norway | € 68.73 | 2,149 | 19.57 % | € 51,292.64 | € 3,032.12 | 94 |
| Sweden | | | | | | |

- ◆ **Conjunto dinámico:** Los cinco productos con mayores ventas

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 |

Navegación relativa

- ◆ Métodos para recorrer una jerarquía: **CURRENTMEMBER**, **PREVMEMBER**, **NEXTMEMBER**, **PARENT**, **FIRSTCHILD**, **LASTCHILD**
- ◆ *Ventas de un miembro de la jerarquía Geography como porcentaje de las ventas de su padre*
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** devuelve el miembro actual a lo largo de una dimensión durante una iteración; de forma abreviada:
(Measures.[Sales Amount]) / (Measures.[Sales Amount], Customer.Geography.CURRENTMEMBER.PARENT)

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



Navegación relativa

- ◆ Las consultas previas no funcionan para el miembro **All** (**All** no tiene padre); se requiere una expresión condicional:

```
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** tiene tres parámetros:

(1) Una condición booleana, en este caso:

(Measures.[Sales Amount], Customer.Geography.CURRENTMEMBER.PARENT)=0

(2) El valor devuelto si la condición es verdadera: '1' en este caso, ya que **All** no tiene padre, que se corresponderá con el porcentaje de ventas

(3) El valor devuelto si la condición es falsa

- ◆ **PREVMEMBER** se utiliza para mostrar crecimiento sobre un período de tiempo

- ◆ *Mostrar las ventas por internet y el incremento mensual para todos los meses en 1996*

```
WITH MEMBER Measures.[Net Sales Growth] AS
    (Measures.[Net Sales]) - (Measures.[Net Sales], [Order Date].Calendar.PREVMEMBER),
    FORMAT_STRING = 'e ###,##0.00; e -###,##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
```

Conjuntos generados

- ◆ **GENERATE**: Itera sobre los miembros de un conjunto, utilizando un segundo conjunto como plantilla para el conjunto resultante

◆ *Ventas por categoría para todos los clientes in Bélgica y Francia*

- ◆ La función **GENERATE** evita enumerar todos los clientes para cada país

```
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 | ... |
| BlondesddsI 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 | ... |
| ... | ... | ... | ... | ... | ... |

Funciones de tiempo

• **PARALLELPERIOD**: compara valores de un miembro especificado con los de un miembro en la misma posición relativa en un período anterior

```
WITH MEMBER Measures.[Previous Year] AS
    (Measures.[Net Sales], PARALLELPERIOD([Order Date].Calendar.Quarter, 4)),
    FORMAT_STRING = 'e ###,##0.00'
MEMBER Measures.[Net Sales Growth] AS
    Measures.[Net Sales] - Measures.[Previous Year],
    FORMAT_STRING = 'e ###,##0.00; e -###,##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 | e 67,531.59 | (null) | e 67,531.59 |
| Q4 1996 | e 114,843.27 | (null) | e 114,843.27 |
| Q1 1997 | e 125,174.40 | (null) | e 125,174.40 |
| Q2 1997 | e 121,518.78 | (null) | e 121,518.78 |
| Q3 1997 | e 133,636.32 | e 67,531.59 | e 66,104.73 |
| Q4 1997 | e 159,989.61 | e 114,843.27 | e 45,146.34 |
| Q1 1998 | e 259,322.36 | e 125,174.40 | e 134,147.95 |
| Q2 1998 | e 104,552.03 | e 121,518.78 | e -16,966.75 |

Funciones de agregación

- ◆ MDX proporciona varias funciones de agregación: **SUM**, **AVG**, **MEDIAN**, **MAX**, **MIN**, **VAR**, y **STDDEV**

- ◆ *Total, máximo, mínimo y media de las ventas al mes en 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 | € 221,157.31 | € 12,992.48 | € 5,584.84 | € 9,119.26 |
| Products | € 80,870.58 | € 6,012.65 | € 1,891.00 | € 4,193.64 |
| Grains/Cereals | € 139,428.18 | € 14,110.16 | € 1,029.00 | € 6,217.45 |
| Meat/Poultry | € 90,216.14 | € 12,157.90 | € 1,650.00 | € 4,429.52 |
| Produce | € 122,307.02 | € 8,448.86 | € 1,587.11 | € 5,263.19 |
| Seafood | | | | |

Funciones de agregación

- ◆ *Obtener el máximo de ventas por categoría en 1997, además del mes en el que se ha obtenido*

```
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** obtiene la **tupla** correspondiente al máximo de ventas realizadas; una vez obtenida, **ITEM** devuelve el primer elemento de una tupla especificada, y finalmente, **NAME** obtiene el nombre de ese miembro

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

Funciones de agregación

Obtener el máximo de ventas por categoría y *país* en 1997, además del mes en el que se ha obtenido

```
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 | e 2,149.40 | December 1997 |
| Beverages | Belgium | e 514.08 | March 1997 |
| Beverages | Denmark | e 10,540.00 | January 1997 |
| Beverages | Finland | e 288.00 | February 1997 |
| Beverages | France | e 915.75 | December 1997 |
| Beverages | Germany | e 8,010.00 | May 1997 |
| ... | ... | ... | ... |



Funciones de agregación

- ◆ **COUNT** cuenta el número de tuplas en un conjunto
- ◆ Dos opciones: incluir o excluir celdas vacías
- ◆ Número de clientes que compraron un producto de una categoría en particular
- ◆ Realizado contando el número de tuplas obtenidas al unir la cantidad de ventas y los nombres de clientes
- ◆ Debe excluir celdas vacías para contar sólo clientes con ventas en la categoría del producto correspondiente

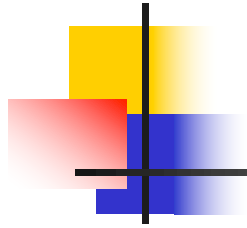
```
WITH MEMBER Measures.[Customer Count] AS
    COUNT({Measures.[Sales Amount] *
    [Customer].[Company Name].MEMBERS}, EXCLUDEEMPTY)
SELECT {Measures.[Sales Amount], [Customer Count]} ON COLUMNS,
    Product.Category.MEMBERS ON ROWS
FROM Sales
```

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



Ejercicio. Consultas en MDX

1. Cantidad total de ventas por cliente, año y categoría de producto.
2. Cantidad de ventas anuales para cada par de país cliente y país proveedor.
3. Ventas mensuales por el estado del cliente comparados con los del año anterior
4. Crecimiento de ventas mensuales por producto, es decir, el total de ventas por producto comparado con las ventas del mes anterior
5. Los tres empleados con mayores ventas.
6. El empleado con mayor venta por producto y año.
7. Total de ventas y media mensual de ventas por empleado y año.



Operaciones Avanzadas

MDX



Chapter 6: Querying the Data Warehouse

Outline

- ◆ Introduction to the MDX Language
- ② **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

- ◆ **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 exceeded 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 | e -2,370.58 | e 6,114.67 | e 8,581.51 | e 7,171.01 | ... |
| Cunewalde | e 6,966.40 | e 2,610.51 | e 8,821.85 | e 7,144.74 | ... |
| London | e 2,088.23 | e 683.88 | e 1,942.56 | e 83.13 | ... |
| Montral | e 9,142.78 | e 2,359.90 | e 213.93 | e 3,609.16 | ... |
| Boise | e 1,871.10 | e 94.84 | e 4,411.46 | e 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 | e 84.77 | 4,644 | 21.71 % | e 115,328.31 | e 6,827.10 | 114 |
| Belgium | e 64.65 | 1,242 | 9.72 % | e 30,505.06 | e 1,179.53 | 49 |
| Denmark | e 70.28 | 1,156 | 17.94 % | e 32,428.94 | e 1,377.75 | 45 |
| Finland | e 54.41 | 848 | 9.09 % | e 17,530.05 | e 827.45 | 51 |
| France | e 64.51 | 3,052 | 11.76 % | e 77,056.01 | e 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 | e 120,361.83 |
| Europe | Condiments | e 60,517.12 |
| Europe | Confections | e 95,690.12 |
| Europe | Dairy Products | e 137,315.75 |
| Europe | Grains/Cereals | e 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 | e 101.46 | 3,616 | 21.40 % | e 103,597.43 | e 4,999.77 | 77 |
| Boise | e 90.90 | 4,809 | 32.41 % | e 102,253.85 | e 6,570.58 | 113 |
| Graz | e 88.00 | 4,045 | 23.57 % | e 93,349.45 | e 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 | e 101.46 | 3,616 | 21.40 % | e 103,597.43 | e 4,999.77 | 77 |
| Boise | e 90.90 | 4,809 | 32.41 % | e 102,253.85 | e 6,570.58 | 113 |
| Graz | e 88.00 | 4,045 | 23.57 % | e 93,349.45 | e 5,725.79 | 92 |
| Top 3 Cities | e 95.46 | 12,470 | 26.69 % | e 299,200.73 | e 17,296.14 | 282 |
| Other Cities | e 38.68 | 33,918 | 0.95 % | e 845,955.13 | e 41,291.35 | 1,649 |
| All Customers | e 134.14 | 46,388 | 27.64 % | e 1,145,155.86 | e 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 | e 103,597.43 |
| Boise | e 102,253.85 |
| Graz | e 93,349.45 |
| London | e 51,169.01 |
| Albuquerque | e 49,290.08 |
| All Customers | e 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 e 10,000.