CSC12107 – Information Systems for Business Intelligence

Chapter 5
BI Application

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Goals

- After complete this chapter, student can:
  - explaining the basic of OLAP
  - Build an OLAP cube and operate the basic OLAP technologies using SSAS tool
  - Apply the basic Multi – Dimensional eXpressions (MDX) syntax to query multidimensional data.

Reference

- Vincent Rainardi - Building a Data Warehouse: With Examples in SQL Server
- PRACTICAL MDX QUERIES for MS SSAS.

Main topic

- BI report
- BI OLAP
- BI Dashboard
- BI mining

BI application

(Source: V.Rainardi: Building a DW with SQLserver)

- **Six categories of BI application**:
  - Report application
  - Analytics application
  - Data mining application
  - Dashboard application
  - Alerts application
  - Portal application

BI Application

(Source: Thomas C.Hammergren & Alan R.Simon - Data warehouse for Dummies 2nd)

Type	Information you want
Basic querying and reporting	Tell me what happened
Business analysis (OLAP)	Tell me what happened and why
Data mining	Tell me wwhat might happen or Tell me something interesting
Dashboard and scorecard	Tell me a lot of things but don’t make me work to hard

BI Reports

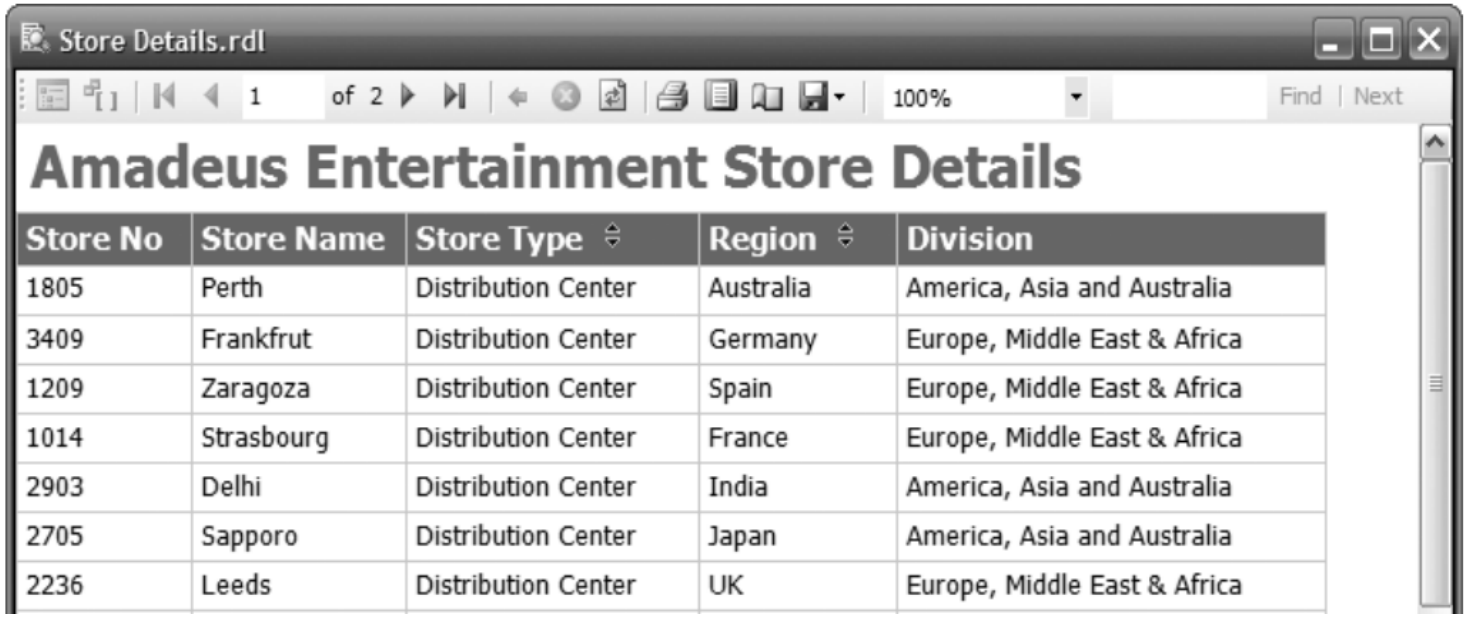
- In the data warehousing context, a report ***is a program that retrieves data from the data warehouse*** and presents it to the users on the screen or on paper:
  - Data quality reports
  - Audit reports
  - Usage reports
  - ODS reports
  - DDS single dimension reports
  - DDS drill-across dimensional reports

BI report

- A simple report retrieves a few columns from a **database table** and presents them in tabular format on the screen

SQL Query for the Store Details Report

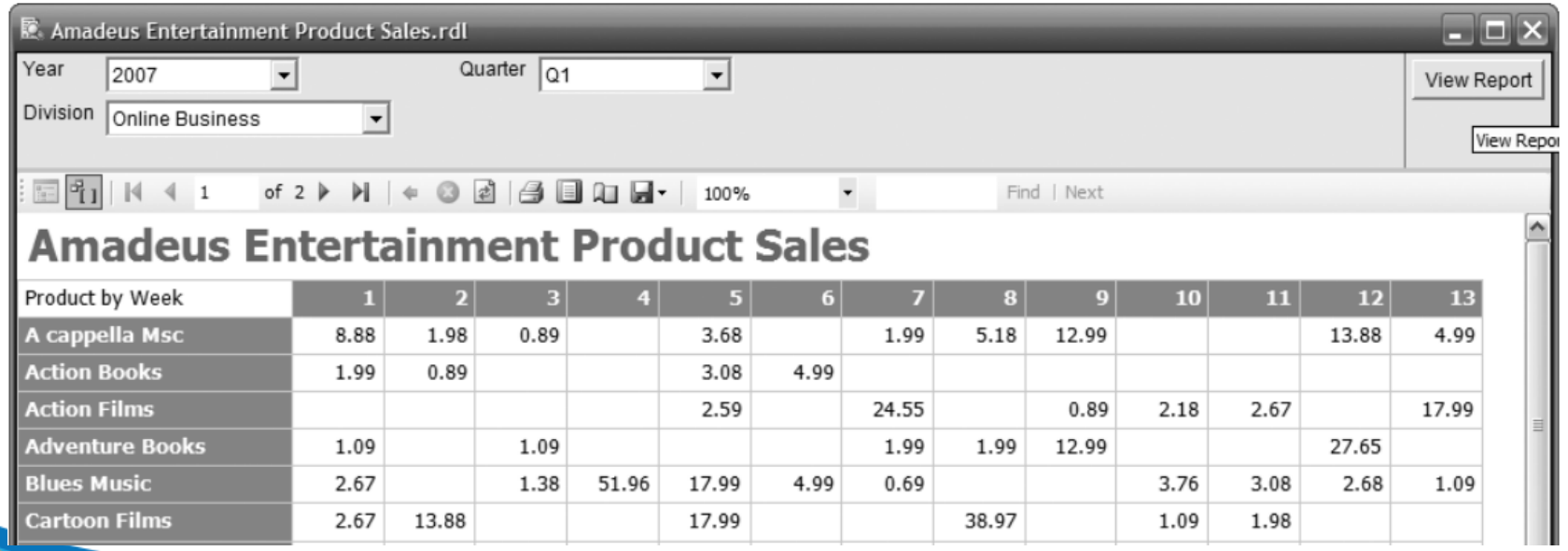
```
Select store_number,store_name,
store_type, region, division
from dim_store
where store_key <> 0
```



Store No	Store Name	Store Type	Region	Division
1805	Perth	Distribution Center	Australia	America, Asia and Australia
3409	Frankfrut	Distribution Center	Germany	Europe, Middle East & Africa
1209	Zaragoza	Distribution Center	Spain	Europe, Middle East & Africa
1014	Strasbourg	Distribution Center	France	Europe, Middle East & Africa
2903	Delhi	Distribution Center	India	America, Asia and Australia
2705	Sapporo	Distribution Center	Japan	America, Asia and Australia
2236	Leeds	Distribution Center	UK	Europe, Middle East & Africa

BI reports

- data warehouse report retrieves data from fact and dimension tables in a **dimensional data store**
  - **shows the weekly product sales of Amadeus Entertainment in a particular quarter for a particular division**



Amadeus Entertainment Product Sales

Product by Week	1	2	3	4	5	6	7	8	9	10	11	12	13
A cappella Msc	8.88	1.98	0.89		3.68		1.99	5.18	12.99			13.88	4.99
Action Books	1.99	0.89			3.08	4.99							
Action Films					2.59		24.55		0.89	2.18	2.67		17.99
Adventure Books	1.09		1.09				1.99	1.99	12.99			27.65	
Blues Music	2.67		1.38	51.96	17.99	4.99	0.69			3.76	3.08	2.68	1.09
Cartoon Films	2.67	13.88			17.99			38.97		1.09	1.98		

BI report

SQL Query for Weekly Product Sales Report

```
SELECT d.week_number, p.product_type, sum(f.sales_value) as [Sales Value]
FROM fact_product_sales f join dim_dated on f.sales_date_key = d.date_key
join dim_product p on f.product_key = p.product_key
join dim_stores on f.store_key = s.store_key
Where d.quarter = @pQuarter and d.year = @pYear and (s.division = @pDivision
or@pDivision = ' All')
Group by d.week_number, p.product_type
```

BI Report

- The main advantage of using reports in BI is their simplicity.
  - Simply to create, to manage, to use.
- Appropriate when the presentation format requirement are simple and static
- Do charting (line chart, bar chart, pie chart, and so on) or present the data in simple tables or pivot table format
  - using parameters and make the report a little bit dynamic
- **Disadvantage**
  - They are not flexible or interactive
  - For viewing data at a higher or lower level → must redesign the report

Analytic application

- **Query the Data warehouse repeatedly and interactively:**
  - To get an overview of the current business performance
  - To compare it with the budget or targets
- **Present the data in flexible formats:**
  - User can go up/down the dimension hierarchy to get different levels of summary
  - Can swap any dimension with another dimension to get a different business perspective

Analytic application

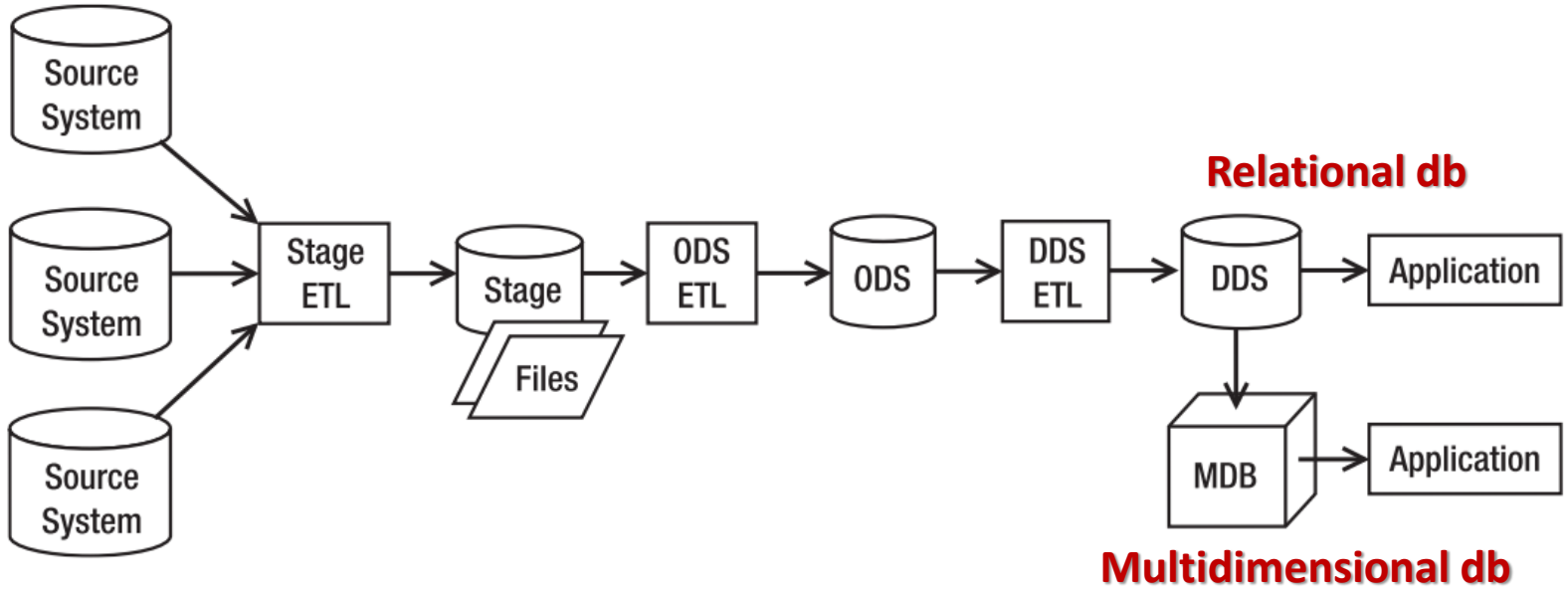
- **OLAP** (Online Analytical Processing) ***is the technology*** behind many Business Intelligence (BI) applications.
  - enables the business users to **go up/drill down** to a particular area of the MDB to view the data at a higher/ more detailed level
  - is an approach to answer multi-dimensional analytical queries

Analytic application

- **OLAP tools**
  - enable users to analyze multidimensional data interactively from multiple perspectives
- **OLAP operations**:
  - Consolidation (roll-up),
  - Drill-down,
  - Slicing and dicing

Analytic application

- Some analytic applications read from relational databases
- Some analytic applications read from multidimensional databases



Multidimensional database

- The **ER data model** is used in design of relational database.
  - consists of a set of entities or objects, and the relationships between them
  - is appropriate for online transaction processing (OLTP)
- **Multidimensional model** is the most popular data model for data warehouses
  - Star schema
  - Snowflake schema
  - Fact constellation schema

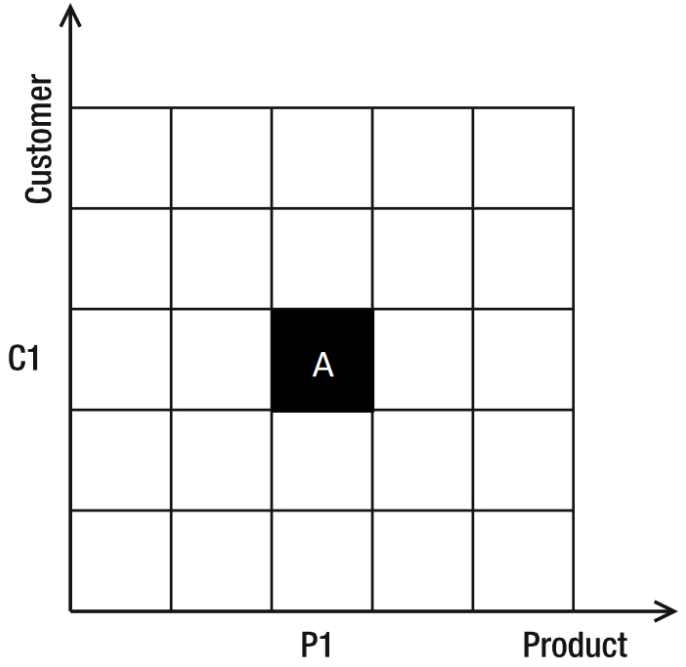
Multidimensional database (MD)

- MD: Is a form of database where
  - The data is stored in **cells**
  - The position of each cell is defined by a number of variables called **dimensions**.
  - Each cell represents a **business event**, and the value of the dimensions shows when and where this event happened
- MDB is populated from DDS

Multidimensional database (MDB)

2 dimensions (Product, Customer) ~ like a matrix

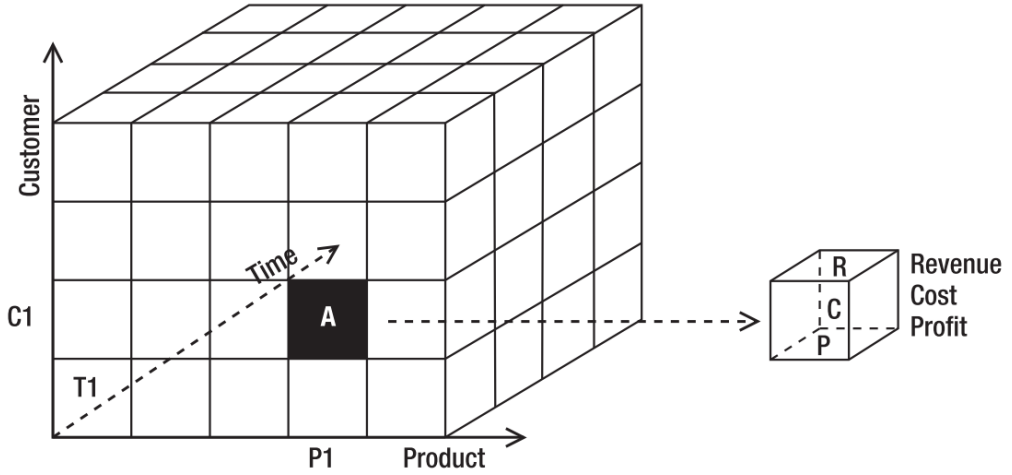
- Each cell represents a business event
- Cell A = Event A is created by
  - customer C1 and
  - product P1
- The cell contains one or more measurement values (or none/empty)



an MDB with two dimensions

Multidimensional database (MDB)

- **An MDB with 3 dimensions ~ a cube**
  - The business event is “A customer buys a product.”
  - Business event A is a transaction where customer C1 buys product P1 at time T1.
  - The cell contains three measure values: the revenue, the cost, and the profit



Multidimensional database (MDB)

- An MDB with four or mor>=4 dimensions is called a hypercube
- Physically it is stored as compressed multidimensional arrays with offset positioning
- MDBs are typically used for online analytical processing (OLAP) and data mining (DM):
  - occupies less disk space compared to a relational dimensional database (like DDS)
  - The aggregates are precalculated
  - Uses multidimensional offsetting to locate the data → minimizes the number of IO operations (disk reads), compared to storing tables in an RDBMS
  - But the drawback is: the processing time required for loading the database and calculating the aggregate values

OLAP

- OLAP concept
- OLAP Cube - multidimensional database
- **MDB / OLAP server**
- OLAP operations

Multidimensional database (MDB)

- Conceptually, a MDB uses the idea of a matrix or a cube
- Physically, an MDB is a file
- **RDBMS** – relational database management system is the system that manages a relational database
- **MDBMS** - a multidimensional database system manages and operates MDBs
  - also known as **OLAP servers** or **cube engines**
  - Microsoft Analysis Services, Hyperion Essbase, and Cognos PowerCube

OLAP SERVER

- **Relational OLAP(ROLAP):**
  - ROLAP is an extended RDBMS along with multidimensional data mapping to perform the standard relational operation.
  - To be able to respond quickly to a query, ROLAP applications store the totals (known as aggregates) in summary tables
- **Multidimensional OLAP (MOLAP)**
  - MOLAP Implementes operation in multidimensional data.
- **Hybrid OnlineAnalytical Processing (HOLAP)**
  - In HOLAP approach the aggregated totals are stored in a multidimensional database while the detailed data is stored in the relational database. This offers both data efficiency of the ROLAP model and the performance of the MOLAP model.

OLAP

- OLAP concept
- OLAP Cube - multidimensional database
- MDB / OLAP server
- **OLAP operations**

Roll/drill up

- Compute the fact based on higher level
  - Roll-up the sale data by product categories
  - Roll-up sales on date from day to quarter

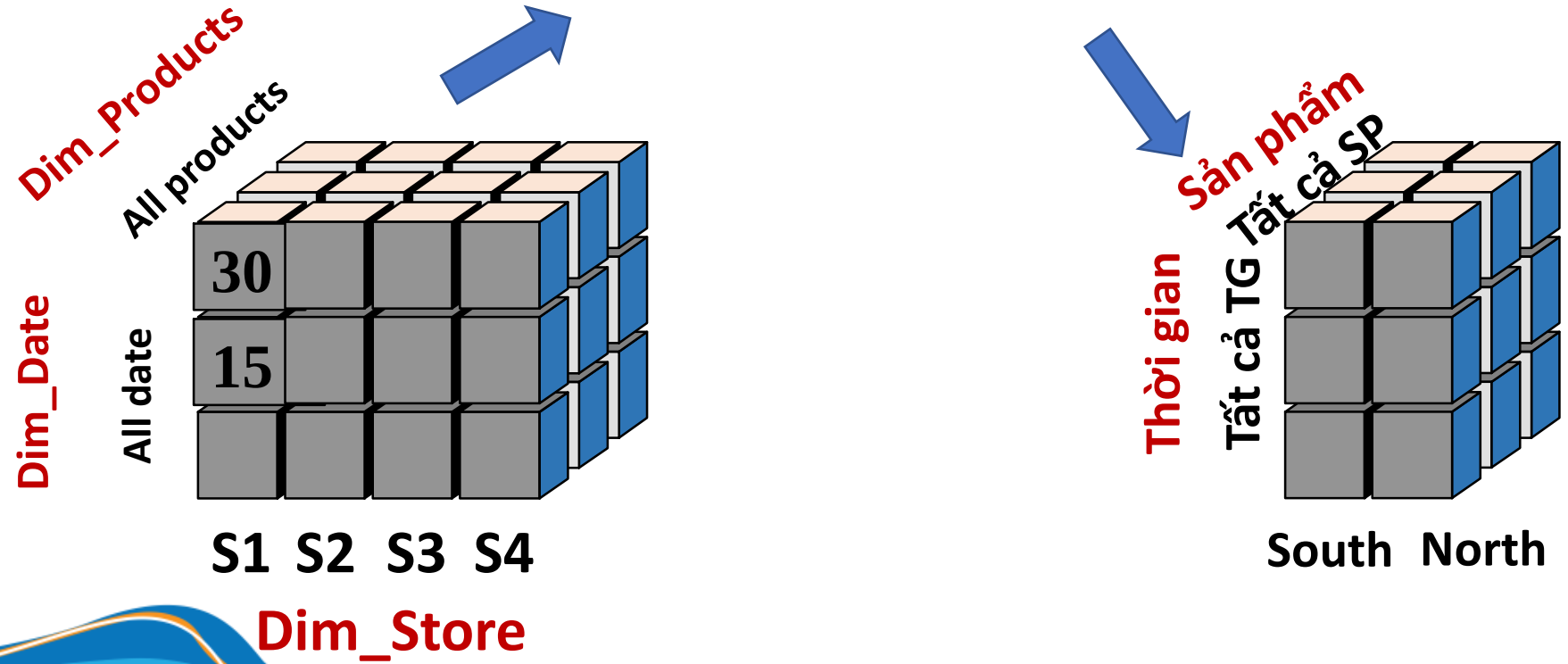
	CalendarYear	CalendarQuarter	QuantitySale
1	2012	4	1120
2	2012	3	809
3	2012	2	744
4	2013	3	14152
5	2010	4	14
6	2013	2	13403
7	2013	1	9202
8	2013	4	16044
9	2014	1	1970
10	2011	3	566

```
select
d.CalendarYear,d.CalendarQuarter,sum(OrderQuantity)
as QuantitySale
from FactInternetSales f join DimDate d on
f.OrderDateKey = d.DateKey
group by d.CalendarYear,d.CalendarQuarter
```

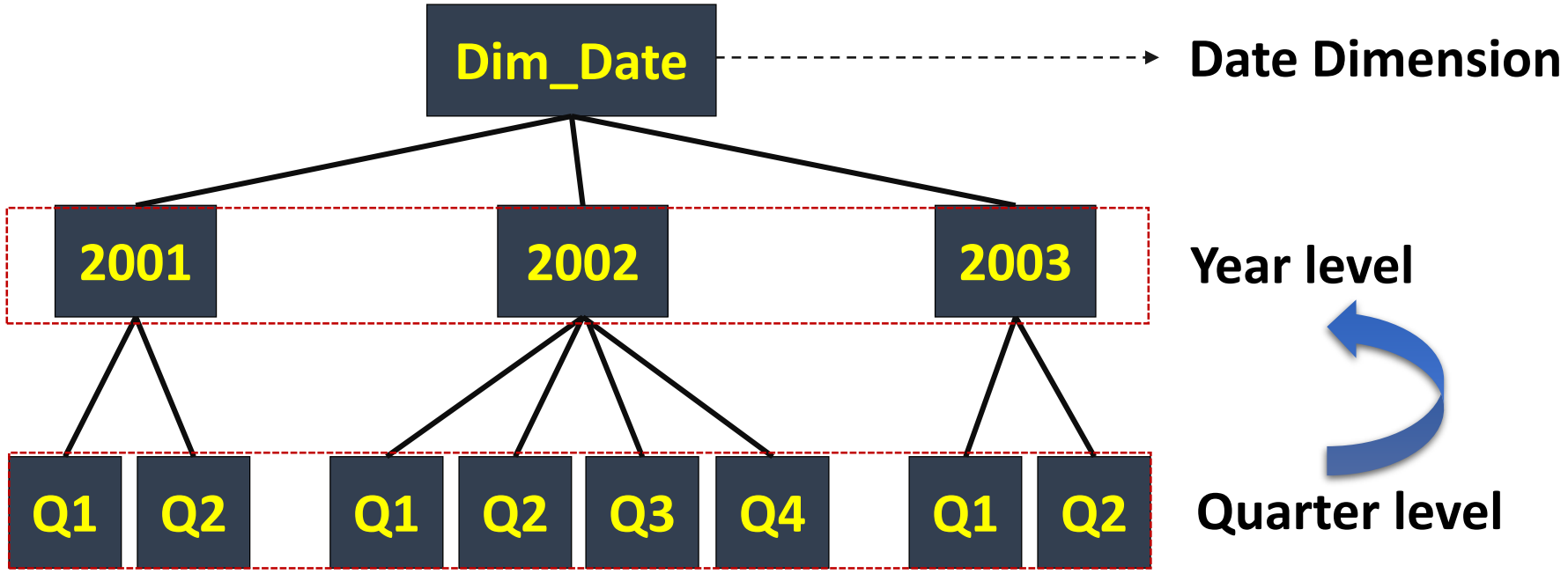
Roll-up

Roll-up

- **Store dimension → Region**



Roll-up

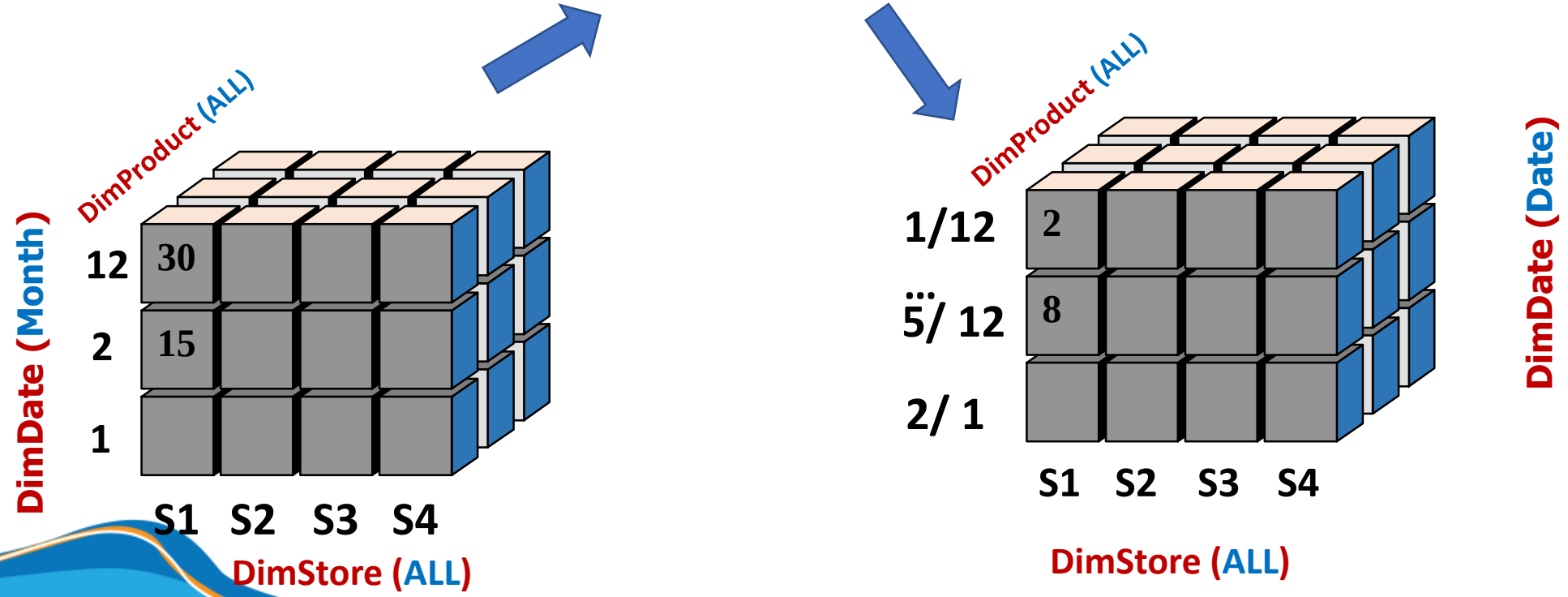


You can roll up a fact from a lower level to a higher level

Drill-down

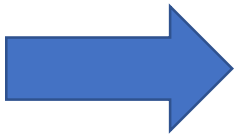
Drill-down

- **Month → day**



Drill-down

	CalendarYear	ProductKey	QuantitySale
1	2012	374	2
2	2013	578	105
3	2014	479	9
4	2012	537	4
5	2011	358	2
6	2013	584	332
7	2011	381	1
8	2013	484	869
9	2013	538	983
10	2012	485	5



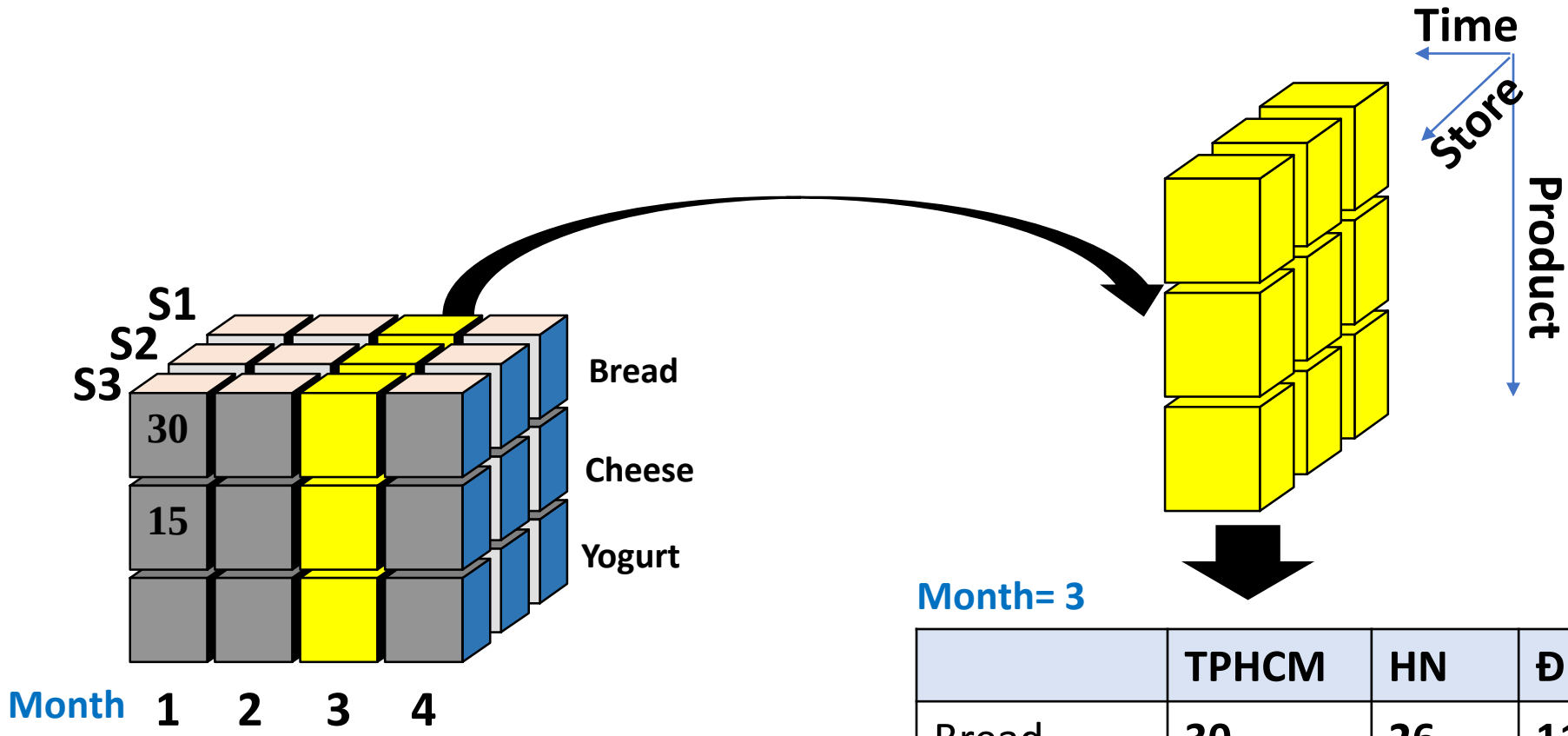
CalendarYear	CalendarQuarter	ProductKey	QuantitySale
2012	3	321	8
2013	2	372	35
2013	2	587	40
2013	3	482	65
2012	1	333	14
2013	2	564	32
2012	4	477	18
2013	1	477	723
2013	2	541	210
2013	4	598	19
2012	4	577	2

Drill down

```
select d.CalendarYear,f.ProductKey,sum(OrderQuantity) as QuantitySale
from FactInternetSales f join DimDate d on f.OrderDateKey = d.DateKey
group by d.CalendarYear,f.ProductKey

select d.CalendarYear, d.CalendarQuarter, f.ProductKey,
sum(OrderQuantity) as QuantitySale
from FactInternetSales f join DimDate d on f.OrderDateKey = d.DateKey
group by d.CalendarYear,CalendarQuarter,f.ProductKey
```

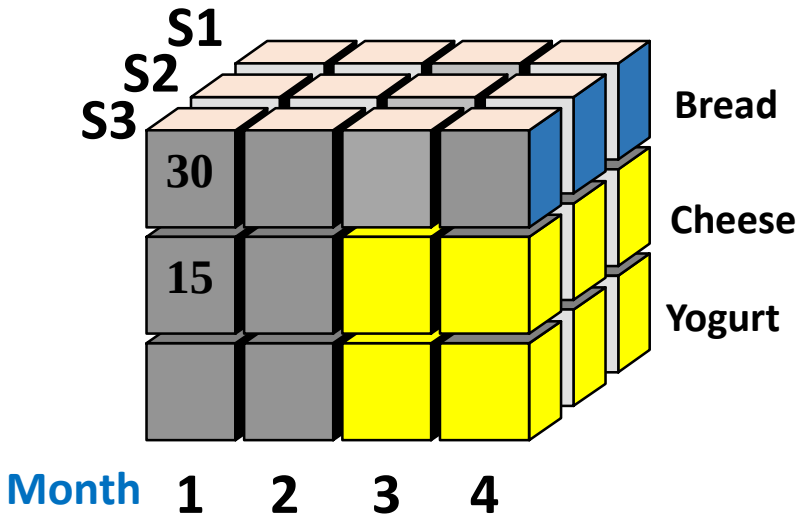
Slice

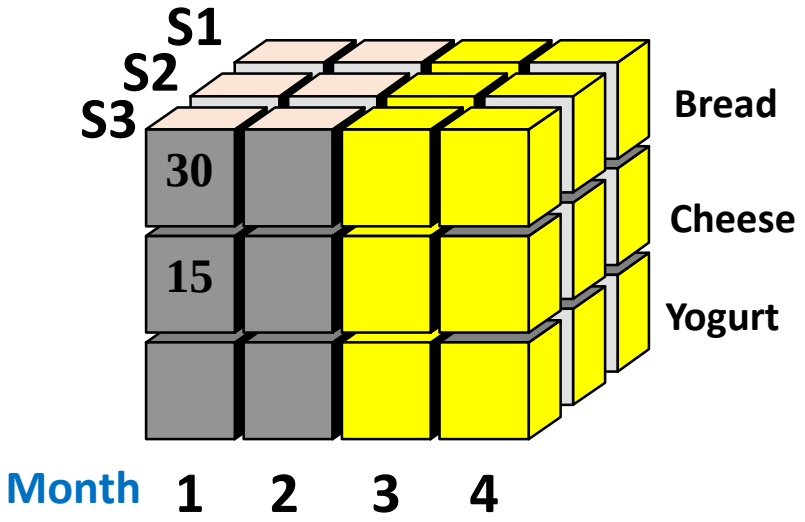


Month= 3

	TPHCM	HN	ĐN
Bread	30	26	11
Cheese	15	35	5
Yogurt	20	18	38

Dice – trích khối con

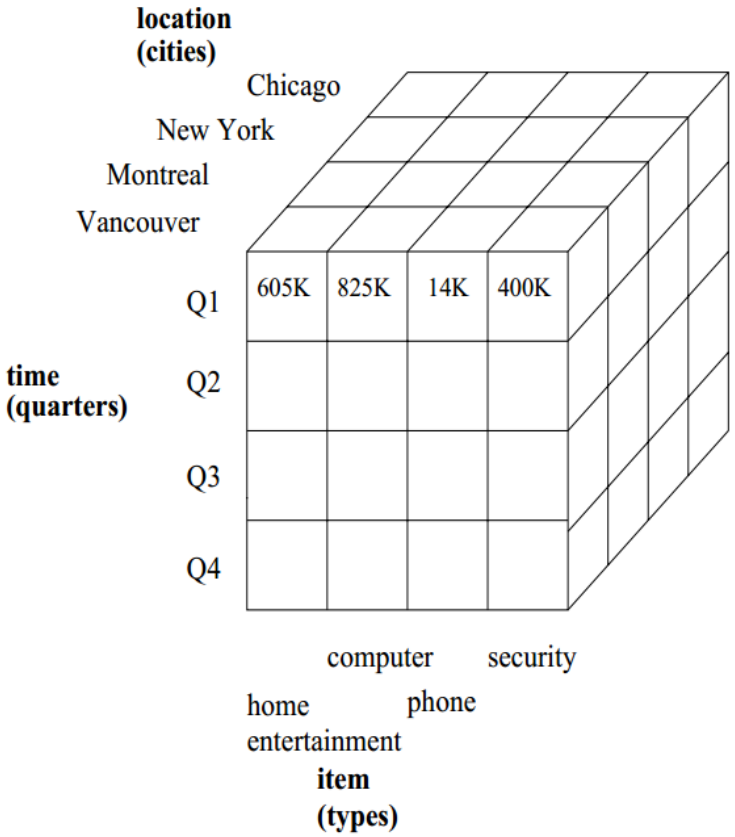




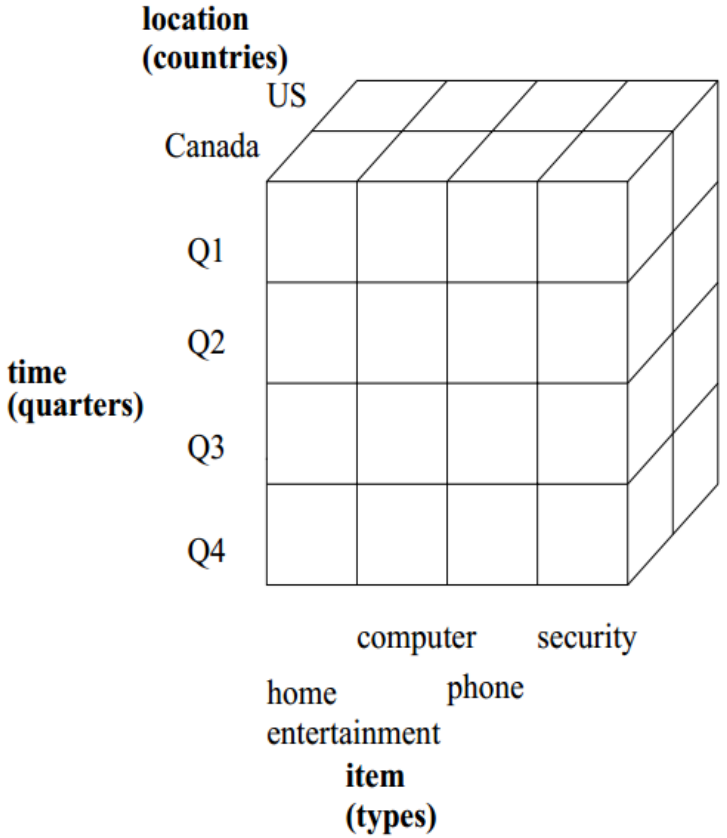
Pivoting (or rotation)

- known as Rotate changes the dimensional orientation of the cube, i.e. rotates the data axes to view the data from different perspectives

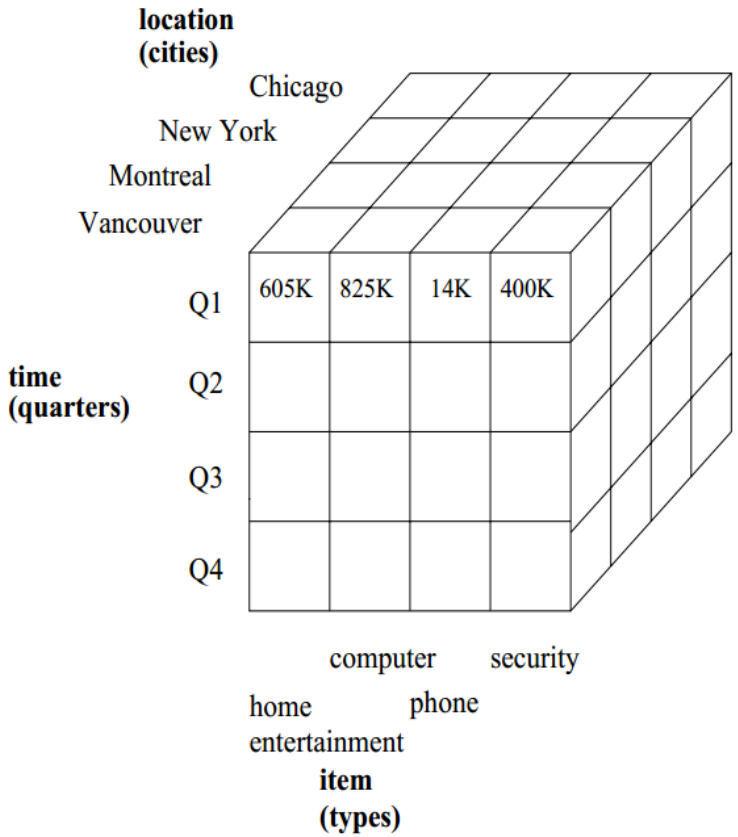
Roll up



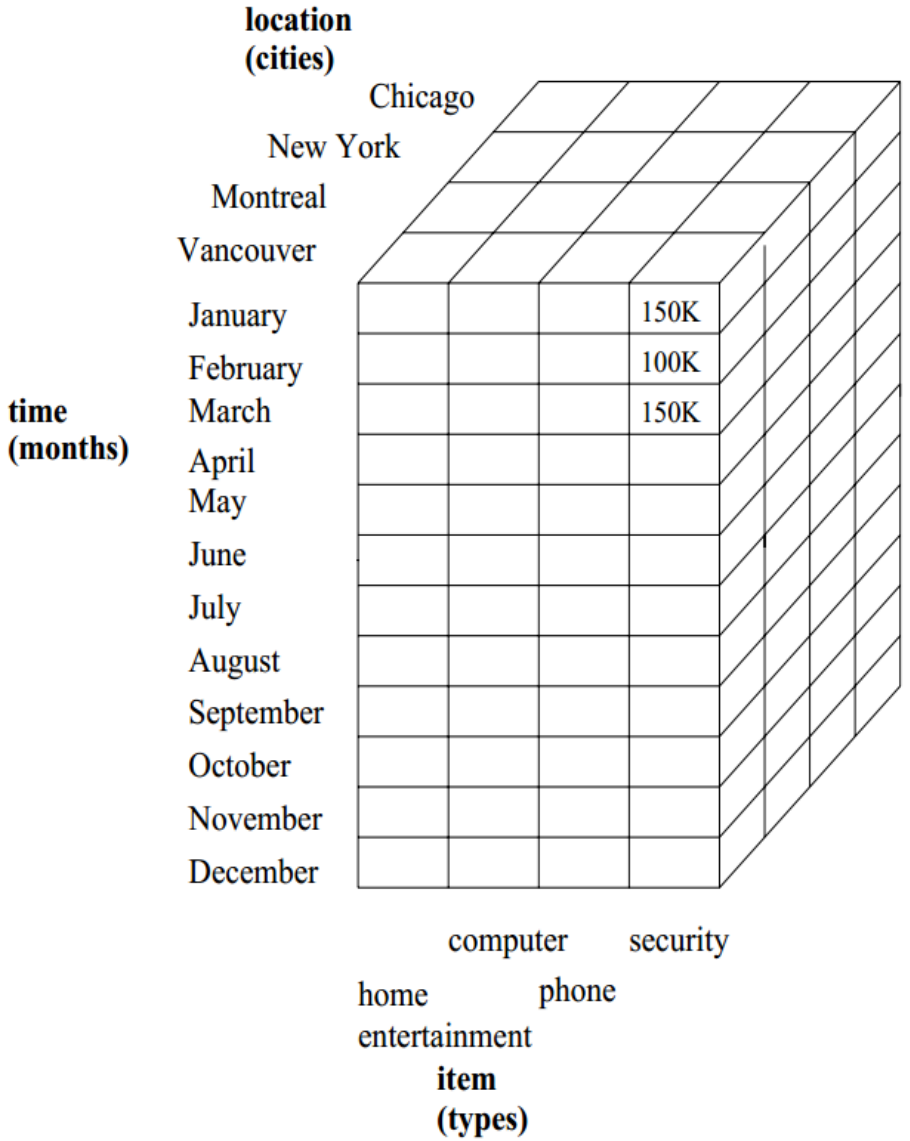
roll-up (from **cities to countries**)



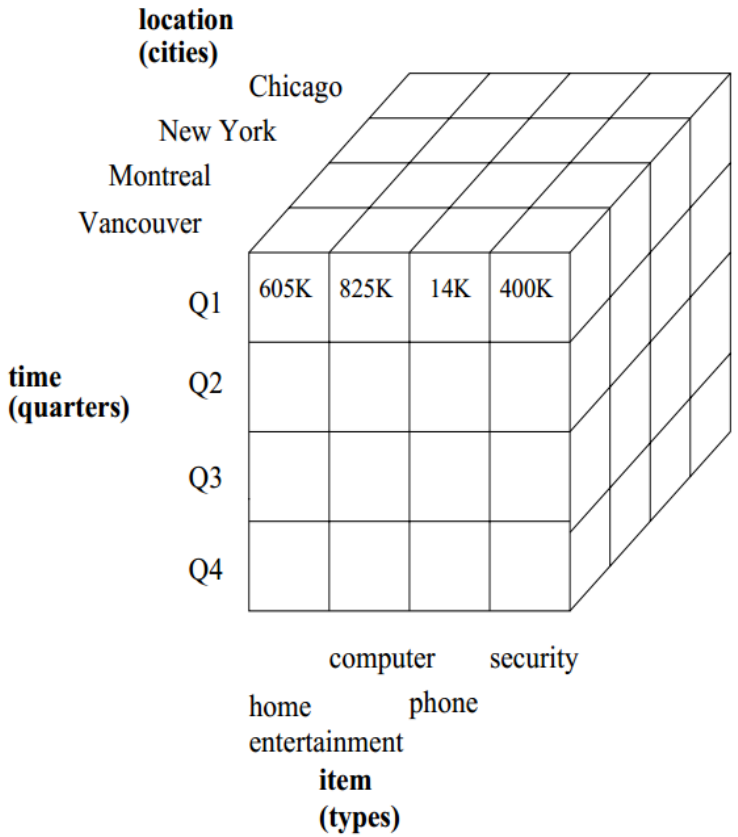
Drill down



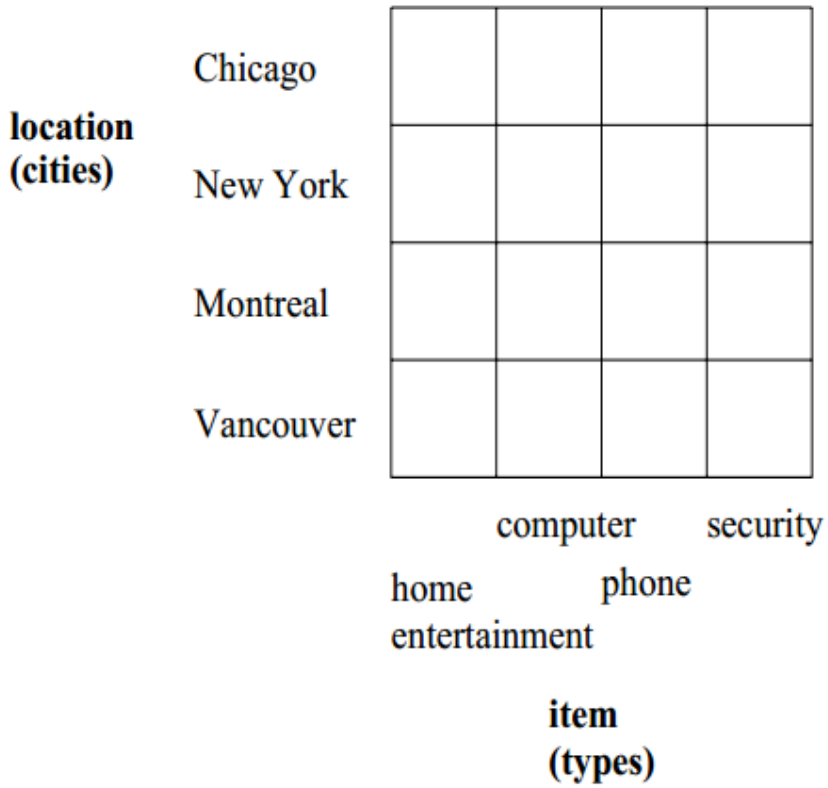
Drill down on time (from **quarters** to **months**)



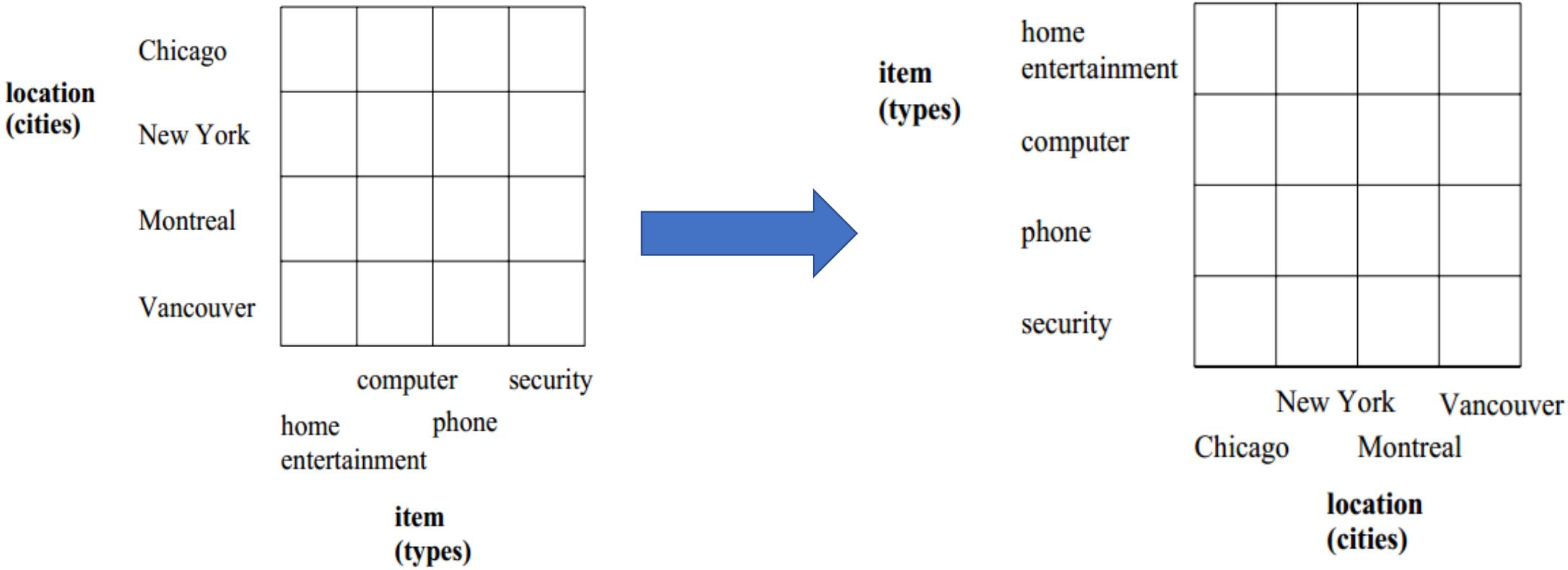
Slice



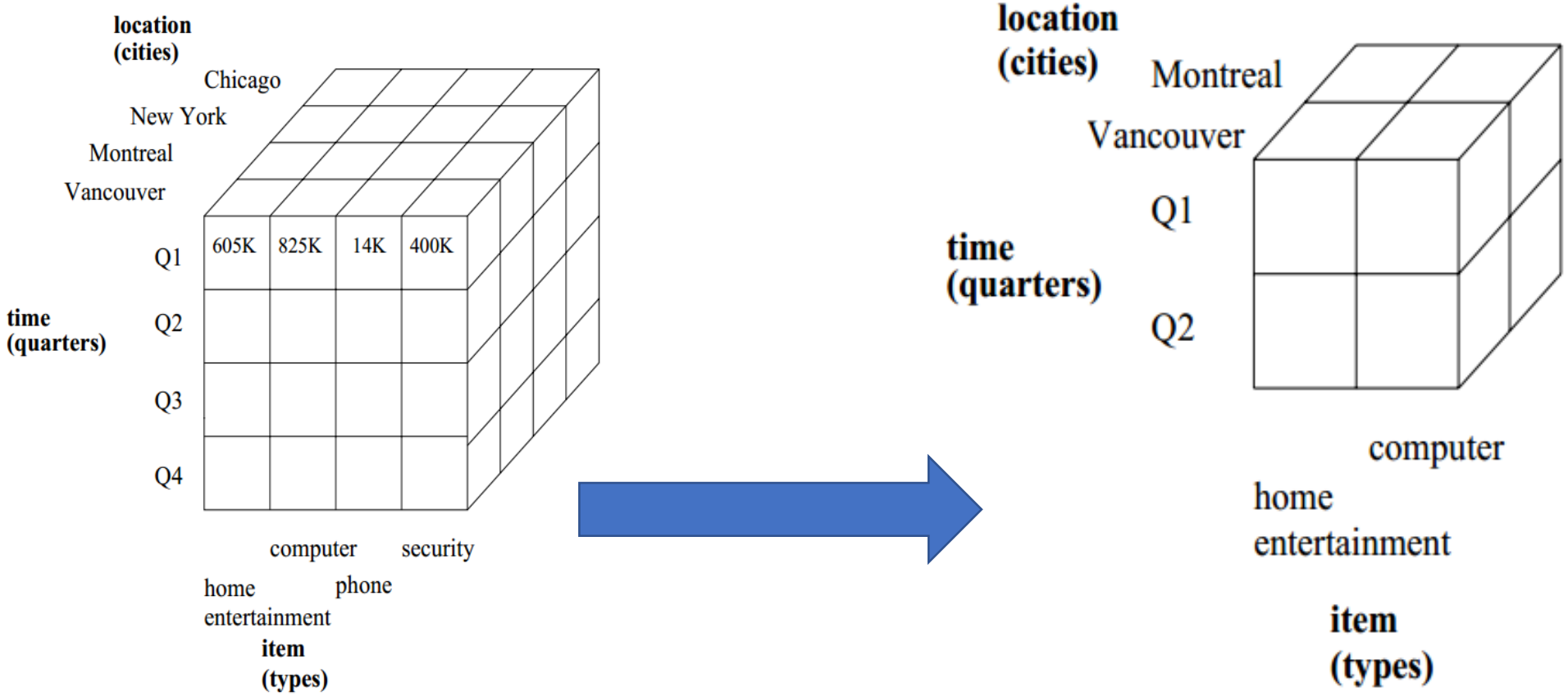
Slice
for time="Q2"



Pivot


location
(cities)
Chicago
New York
Montreal
Vancouver
computer
security
home
entertainment
phone
item
(types)
item
(types)
home
entertainment
computer
phone
security
New York
Vancouver
Chicago
Montreal
location
(cities)

Dice



dice for

(location="Montreal" or "Vancouver") And

(time="Q1" or "Q2") and

(item="home entertainment" or "computer")

SQL extension for OLAP

https://web.stanford.edu/class/cs345d-01/rl/olap.pdf

- SQL has several aggregate operators:
  - sum(), count(), avg(), min(), max()
- The basic idea is:
  - Combine all values in a column into a single scalar value
- **Syntax:**

  Select ProductKey,sum(OrderQuantity) as QuantitySale
  From FactInternetSales
  Group by ProductKey

	ProductKey	QuantitySale
1	593	39
2	355	392
3	570	48
4	378	147
5	384	199
6	361	427
7	576	147
8	564	140
9	324	16
10	344	58

SQL extension for OLAP

- Group By allows aggregates over table sub-groups
- Result is a new table

	CalendarQuarter	OrderQuantity
9	4	1
10	4	1
11	4	1
12	4	1
13	4	1
14	4	1
15	1	1
16	1	1
17	1	1

	CalendarQuarter	SaleQuantity
1	3	15527
2	1	12334
3	4	17829
4	2	14708

SQL extension for OLAP

- Certain forms of data analysis are difficult if not impossible with the SQL constructs.
- Crosstab?

	Sales Amount	Order Quantity
All	29359797.220701	60398
Bib-Shorts	(null)	(null)
Bike Racks	39360	328
Bike Stands	39591	249
Bottles and Cages	56798.1899999963	7981
Bottom Brackets	(null)	(null)
Brakes	(null)	(null)
Caps	19688.1000000002	2190
Chains	(null)	(null)

SQL extension for OLAP

- SQL also introduced additional options in the Group By clause:
  - GROUPING SETS
  - Rollup
  - Cube

GROUPING SET

- Is a set of columns by which you group using the GROUP BY clause.
- Normally, a single aggregate query defines a single grouping set.

```
SELECT warehouse, product, SUM (quantity) qty
FROM inventory
GROUP BY warehouse, product;
```

	warehouse	product	qty
1	San Fransisco	iPhone	260
2	San Fransisco	Samsung	300
3	San Jose	iPhone	300
4	San Jose	Samsung	350

https://www.sqltutorial.org/sql-grouping-sets/

GROUPING SET

- we have **four grouping sets**:
  - (warehouse, product),
  - (warehouse),
  - (product),
  - and ()

```
SELECT warehouse, product, SUM (quantity) qty
FROM inventory
GROUP BY warehouse, product;
```

```
SELECT warehouse, SUM (quantity) qty
FROM inventory
GROUP BY warehouse;
```

```
SELECT product, SUM (quantity) qty
FROM inventory
GROUP BY product;
```

```
SELECT SUM(quantity) qty
FROM inventory;
```

GROUPING SET

- To return all grouping sets using a single query, you can use the UNION ALL operator to combine all the queries above

```
SELECT warehouse, product, SUM (quantity) qty FROM inventory GROUP BY warehouse, product
UNION ALL
SELECT warehouse, null, SUM (quantity) qty FROM inventory GROUP BY warehouse
UNION ALL
SELECT null, product, SUM (quantity) qty FROM inventory GROUP BY product
UNION ALL
SELECT null, null, SUM(quantity) qty FROM inventory;
```

GROUPING SET

- the query is difficult to read because it is lengthy.
- it has a performance issue because the database system has to scan the inventory table multiple times.
- ➔ **GROUPING SET**

	warehouse	product	qty
1	San Fransisco	iPhone	260
2	San Fransisco	Samsung	300
3	San Jose	iPhone	300
4	San Jose	Samsung	350
5	San Fransisco	NULL	560
6	San Jose	NULL	650
7	NULL	iPhone	560
8	NULL	Samsung	650
9	NULL	NULL	1210

```
SELECT c1, c2, aggregate (c3)
FROM table
GROUP BY GROUPING SETS ((c1, c2), (c1), (c2), () );
```

GROUPING SET

```
SELECT warehouse, product, SUM (quantity) qty
FROM inventory
GROUP BY GROUPING SETS ((warehouse,product), (warehouse), (product), () );
```

- Here is the output:

warehouse	product	
San Fransisco	iPhone	
San Jose	iPhone	
NULL	iPhone	Inventory by iPhone
San Fransisco	Samsung	
San Jose	Samsung	
NULL	Samsung	Inventory by Samsung
NULL	NULL	All Inventory
San Fransisco	NULL	Inventory By San Fransisco
San Jose	NULL	Inventory By San Jose

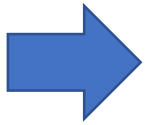
GROUP BY ROLLUP

- **ROLLUP(d1,d2,d3)** creates only 4 grouping sets:
  - (d1, d2, d3)
  - (d1, d2)
  - (d1)
  - ()
- **Syntax**:

```
SELECT d1, d2, d3, aggregate_function(c4)
FROM table_name
GROUP BY ROLLUP (d1, d2, d3);
```

GROUP BY ROLLUP

	warehouse	product	model	quantity
1	San Fransisco	iPhone	6s	50
2	San Fransisco	iPhone	7	10
3	San Fransisco	iPhone	X	200
4	San Fransisco	Samsung	Galaxy S	200
5	San Fransisco	Samsung	Note 8	100
6	San Jose	iPhone	6s	100
7	San Jose	iPhone	7	50
8	San Jose	iPhone	X	150
9	San Jose	Samsung	Galaxy S	200
10	San Jose	Samsung	Note 8	150



	warehouse	product	model	Quantity
1	San Fransisco	iPhone	6s	50
2	San Fransisco	iPhone	7	10
3	San Fransisco	iPhone	X	200
4	San Fransisco	iPhone	NULL	260
5	San Fransisco	Samsung	Galaxy S	200
6	San Fransisco	Samsung	Note 8	100
7	San Fransisco	Samsung	NULL	300
8	San Fransisco	NULL	NULL	560
9	San Jose	iPhone	6s	100
10	San Jose	iPhone	7	50
11	San Jose	iPhone	X	150
12	San Jose	iPhone	NULL	300
13	San Jose	Samsung	Galaxy S	200
14	San Jose	Samsung	Note 8	150
15	San Jose	Samsung	NULL	350
16	San Jose	NULL	NULL	650
17	NULL	NULL	NULL	1210

```
select * from inventory
```

```
SELECT warehouse,product,model,sum(quantity) as Quantity
FROM inventory
GROUP BY ROLLUP (warehouse,product, model);
```

GROUP BY CUBE

- The **CUBE (d1,d2,d3)** defines 8 possible grouping sets:
  1. (d1, d2, d3)
  2. (d1, d2)
  3. (d2, d3)
  4. (d1, d3)
  5. (d1)
  6. (d2)
  7. (d3)
  8. ()

https://www.sqlservertutorial.net/sql-server-basics/sql-server-rollup/

GROUP BY CUBE

- **Syntax**:

```
SELECT d1, d2, d3, aggregate_function(c4)
FROM table_name
GROUP BY CUBE (d1, d2, d3);
```

Exercises

- https://www.mssqltips.com/sqlservertip/6315/group-by-in-sql-sever-with-cube-rollup-and-grouping-sets-examples/
- https://www.sqlservertutorial.net/sql-server-basics/sql-server-cube/
- https://simonlearningsqlserver.wordpress.com/2018/03/25/grouping-sets-and-rollup-cube/

Data mart

- A data mart is a set of dimensional tables supporting a business process
- The centralized data warehouse releases data marts. These data marts are often described as built to answer a business question
- The DDS consists of one or several dimensional data marts. A dimensional data mart is a group of related fact tables and their corresponding dimension tables containing the measurements of business events,categorized bytheir dimensions

MDX

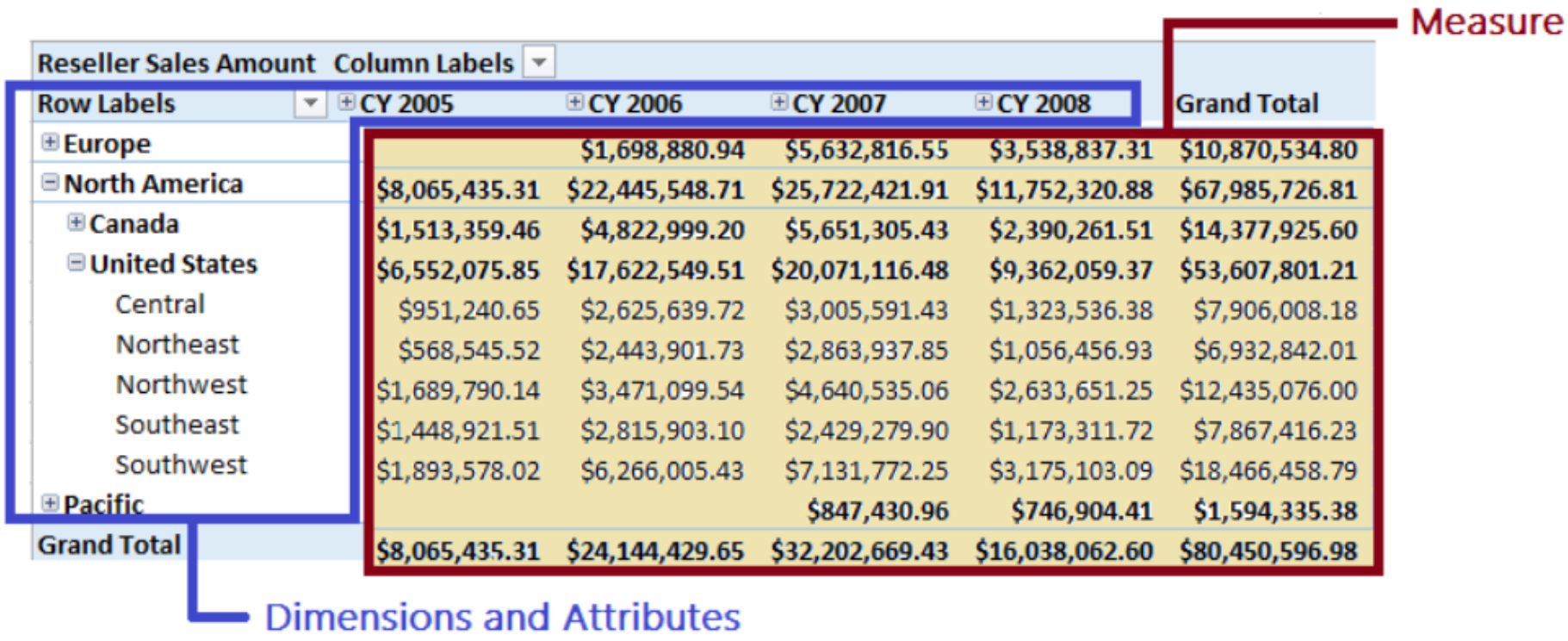
- MDX concept
- MDX syntax
- Some MDX function

MDX

- SQL syntax is typically used with relational databases
- MDX is not an extension of the SQL language and is different from SQL in many ways
- **MDX: Multi-Dimensional eXpressions**
  - Is the query language that you use to work with and retrieve multidimensional data

MDX

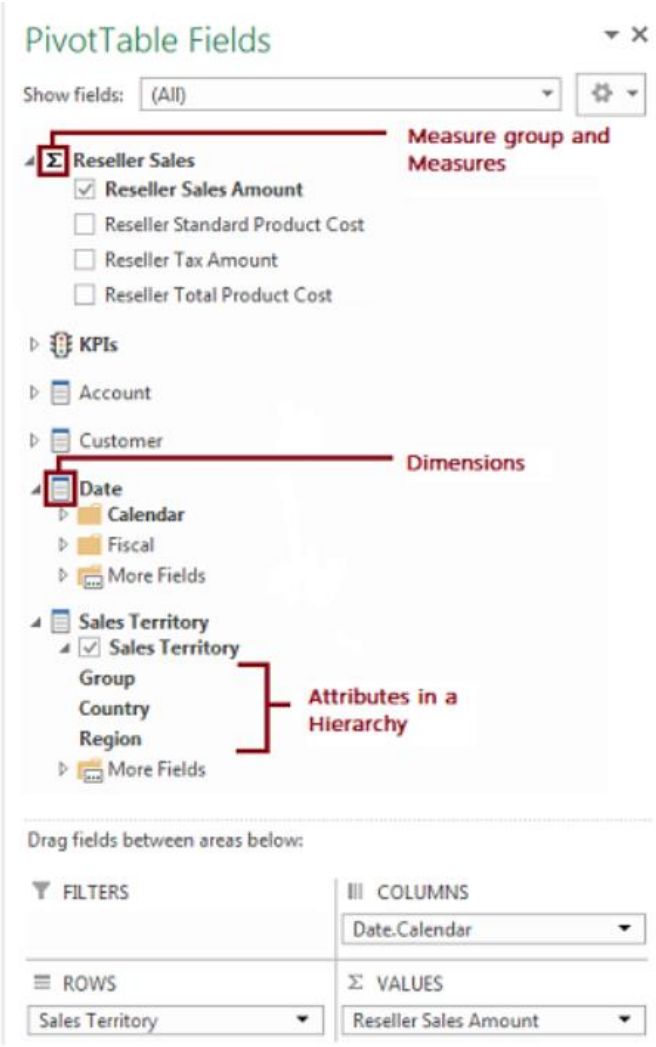
- Sample cube:



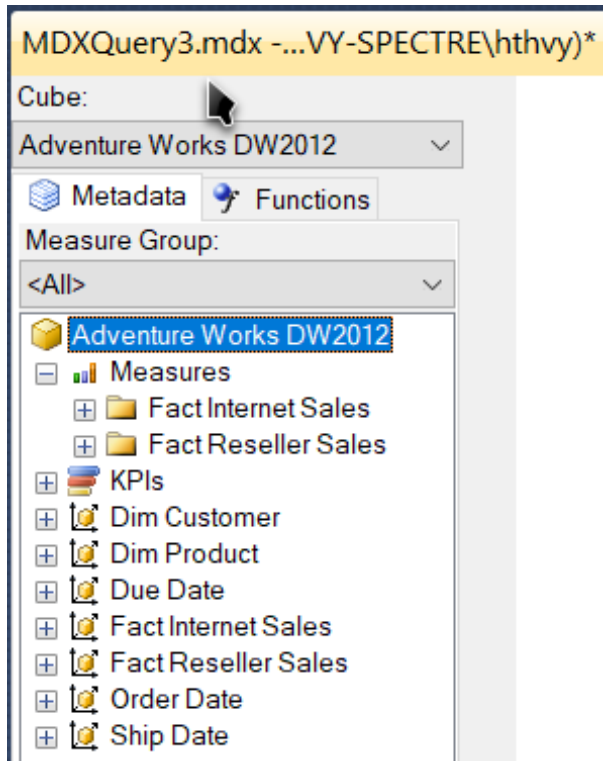
Reseller Sales Amount	Column Labels				
Row Labels	CY 2005	CY 2006	CY 2007	CY 2008	Grand Total
Europe		$1,698,880.94	$5,632,816.55	$3,538,837.31	$10,870,534.80
North America	$8,065,435.31	$22,445,548.71	$25,722,421.91	$11,752,320.88	$67,985,726.81
Canada	$1,513,359.46	$4,822,999.20	$5,651,305.43	$2,390,261.51	$14,377,925.60
United States	$6,552,075.85	$17,622,549.51	$20,071,116.48	$9,362,059.37	$53,607,801.21
Central	$951,240.65	$2,625,639.72	$3,005,591.43	$1,323,536.38	$7,906,008.18
Northeast	$568,545.52	$2,443,901.73	$2,863,937.85	$1,056,456.93	$6,932,842.01
Northwest	$1,689,790.14	$3,471,099.54	$4,640,535.06	$2,633,651.25	$12,435,076.00
Southeast	$1,448,921.51	$2,815,903.10	$2,429,279.90	$1,173,311.72	$7,867,416.23
Southwest	$1,893,578.02	$6,266,005.43	$7,131,772.25	$3,175,103.09	$18,466,458.79
Pacific			$847,430.96	$746,904.41	$1,594,335.38
Grand Total	$8,065,435.31	$24,144,429.65	$32,202,669.43	$16,038,062.60	$80,450,596.98

MDX

- MDX return a subset of multidimensional data from cube
  - **EX**: a multidimensional model to a PivotTable in Excel.
-

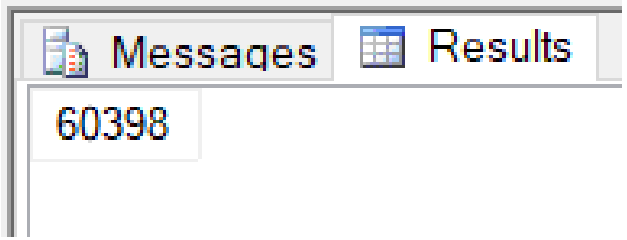


MDX - hello world

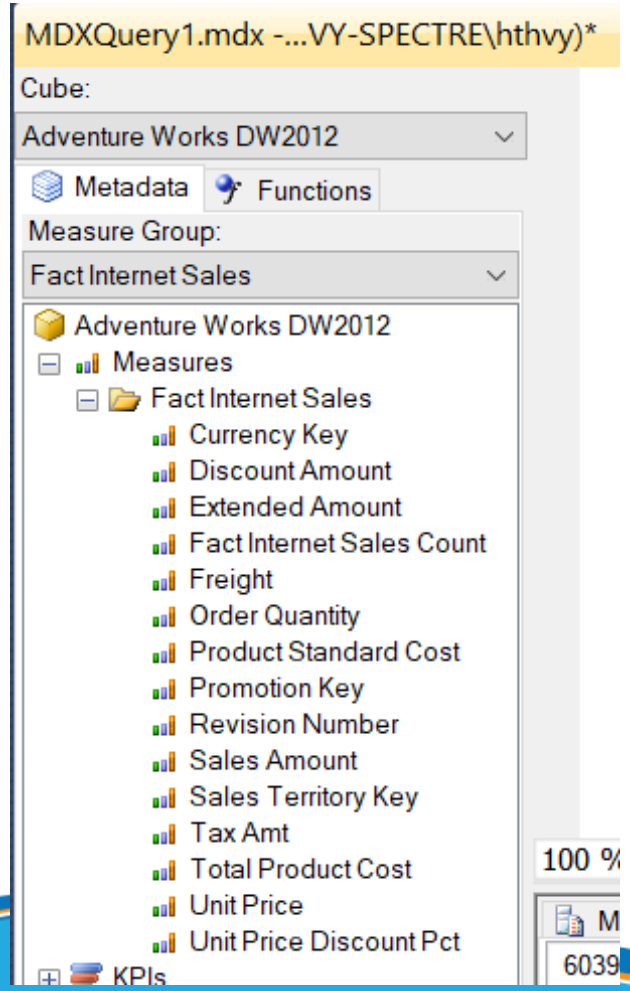


- The following query show the super grand total from the cube
  - That is, for all products and all dates and all customers
  - it's the super grand total for Fact internet Sales

```
select
From [Adventure Works DW2012]
```

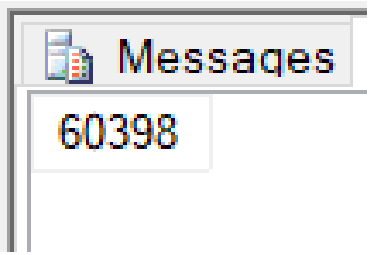


MDX - hello world

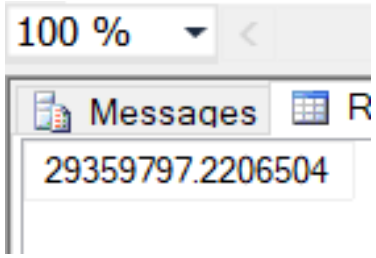


- In MDX, a **Where** clause is called a **slicer**.

```
select
from [Adventure Works DW2012]
Where [Measures].[Order Quantity]
```



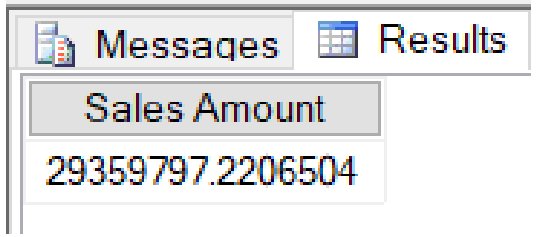

```
select
from [Adventure Works DW2012]
where [Measures].[Sales Amount]
```



MDX - hello world

- No Where clause this time. Instead, the Sales Amount is requested on the columns

```
select [Measures].[Sales Amount] on columns
from [Adventure Works DW2012]
```



MDX - Dimension Data on Rows

- Adding dimension data to columns
- **[Order Date].[Hierarchy].[Year]** is in the format **[Dimension].[Hierarchy].[Level]**

```
select [Order Date].[Hierarchy].[Year] on columns
from [Adventure Works DW2012];
```

Messages | Results

2005	2006	2007	2008	2009	2010	2011	2012	2013	2014
(null)	(null)	(null)	(null)	(null)	14	2216	3397	52801	1970

MDX - Dimension Data on column

- There is now a comma-separated list for the column specification.
- The braces around the two entries are obligatory—the query will fail without them

```
-- and a total column
Select {[Order Date].[Hierarchy].[Year], [Order Date].[Hierarchy]} on columns
From [Adventure Works DW2012];
```

Messages | Results

2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	All
(null)	(null)	(null)	(null)	(null)	14	2216	3397	52801	1970	60398

MDX - Dimension Data on Rows

- Two dimensions on two axes - a comma (,) separates the column specification from the row specification

```
-- adding dimension data to rows
select {[Order Date].[Hierarchy].[Year],[Order Date].[Hierarchy]}
on columns,
[Dim Product].[Hierarchy].[Product Subcategory Key] on rows
from [Adventure Works DW2012];
```

Messages | Results

	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	All
2	(null)	(null)	(null)	(null)	(null)	9	1821	2158	4080	(null)	8068
20	(null)	(null)	(null)	(null)	(null)	(null)	(null)	3	1363	64	1430
21	(null)	(null)	(null)	(null)	(null)	(null)	(null)	8	3189	135	3332
22	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	963	56	1019
23	(null)	(null)	(null)	(null)	(null)	(null)	(null)	2	541	25	568
24	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
25	(null)	(null)	(null)	(null)	(null)	(null)	(null)	1	531	30	562

MDX - Dimension Data on Rows

- Two dimensions on two axes - a comma (,) separates the column specification from the row specification

```
-- a total rows
select {[Order Date].[Hierarchy].[Year],[Order
Date].[Hierarchy]} on columns,
{[Dim Product].[Hierarchy].[Product Subcategory
Key],[Dim Product].[Hierarchy]} on rows
from [Adventure Works DW2012];
```

	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	All
31	(null)	(null)	(null)	(null)	(null)	(null)	(null)	26	6174	240	6440
32	(null)	(null)	(null)	(null)	(null)	(null)	(null)	2	708	23	733
33	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
34	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
35	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
36	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
37	(null)	(null)	(null)	(null)	(null)	(null)	(null)	34	16373	925	17332
4	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
5	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
6	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
7	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
8	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
9	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)
All	(null)	(null)	(null)	(null)	(null)	14	2216	3397	52801	1970	60398

MDX - Hiding Nulls

```
--hidding null
select non empty{[Order
Date].[Hierarchy].[Year],[Order
Date].[Hierarchy]} on columns,
non empty {[Dim Product].[Hierarchy].[Product
Subcategory Key], [Dim Product].[Hierarchy]} on
rows
from [Adventure Works DW2012];
```

Messages | Results

	2010	2011	2012	2013	2014	All
21	(null)	(null)	8	3189	135	3332
22	(null)	(null)	(null)	963	56	1019
23	(null)	(null)	2	541	25	568
25	(null)	(null)	1	531	30	562
26	(null)	(null)	(null)	308	20	328
27	(null)	(null)	1	237	11	249
28	(null)	(null)	38	7718	225	7981
29	(null)	(null)	(null)	869	39	908
3	(null)	(null)	13	2154	(null)	2167
30	(null)	(null)	5	2022	94	2121
31	(null)	(null)	26	6174	240	6440
32	(null)	(null)	2	708	23	733
37	(null)	(null)	34	16373	925	17332
All	14	2216	3397	52801	1970	60398

MDX - Displaying a Different Measure

```
--display different measure
select non empty{[Order Date].[Hierarchy].[Year],[Order
Date].[Hierarchy]} on columns,
non empty [Dim Product].[Hierarchy].[Product Subcategory Key on rows
from [Adventure Works DW2012]
where [Measures].[Sales Amount - Fact Reseller Sales]
```

	2010	2011	2012	2013	All
1	237464.298	9044828.3657999	9040948.22199994	8169443.4906998	26492684.3764996
10	(null)	4638.636	73293.0536000001	(null)	77931.6896000001
11	(null)	2550.9076	58391.2908000001	(null)	60942.1984000001
12	9633.656	863782.753500002	1783601.52640001	2056654.211	4713672.14690001
13	(null)	(null)	12733.4753	134750.4345	147483.9098
14	21892.3044	694553.450599997	1946295.79020001	1187111.79860001	3849853.34380002
15	(null)	(null)	5298.9861	50530.4021000001	55829.3882000001
16	(null)	(null)	151825.9755	1490501.7107	1642327.6862

MDX – Hierachy

- Dimensions contain one or more hierarchies
- Hierarchies contain one or more levels,
  - there is always, by default, an All level for every hierarchy
- Levels contain members.

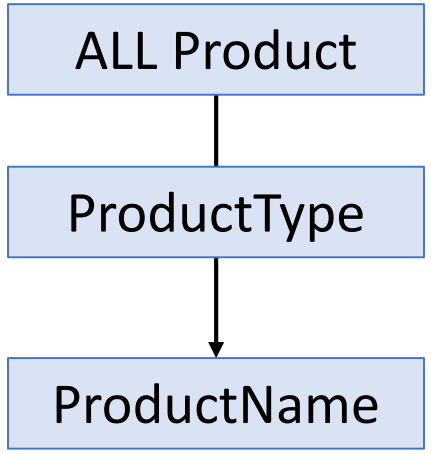
Dimensions

Hierarchies

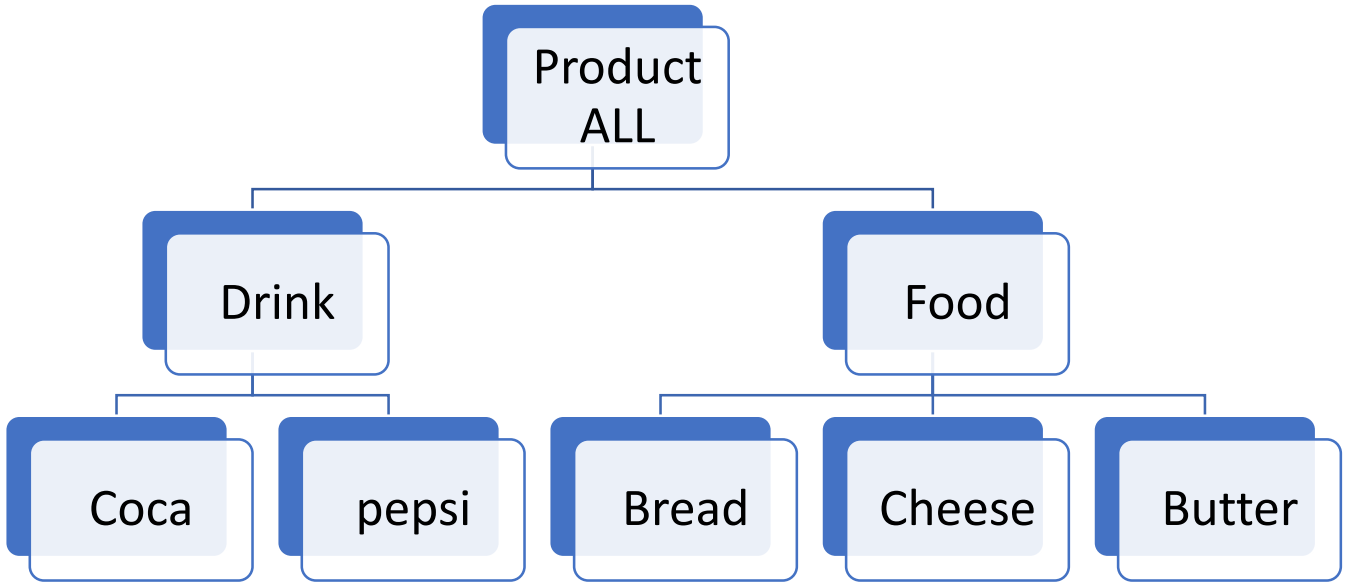
Levels

Members

MDX – Hierachy



Product Dimension

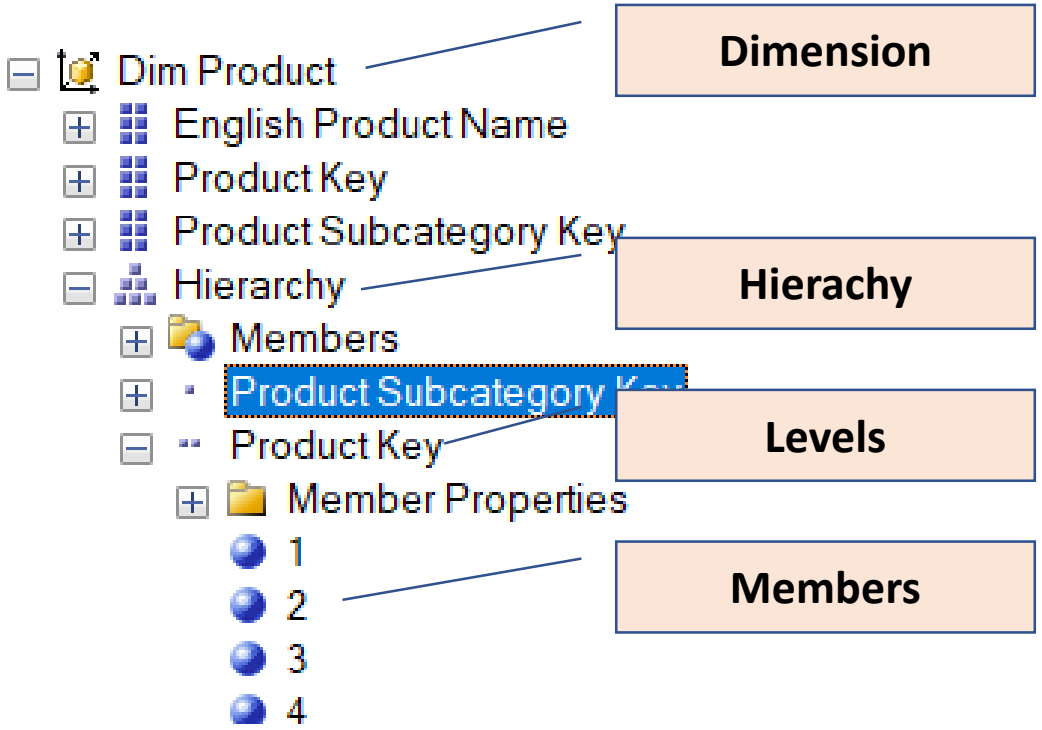


Members of the Product dimension

Return a sets of members:

1. [Product].[ProductType].Pepsi
2. [Product].[ProductName].**Members** = { Coca, pepsi, Bread, Cheese, Butter}
3. [Product].[Drink].**CHILDREN** = {Coca, pepsi}
4. [Product].[Food].[Cheese]:[Butter] = {Cheese, Butter}

MDX – Hierachy




```
--
dimension.hierarchy.level.members
Select non empty [Dim
Product].[Hierarchy].[Product
Subcategory Key].members on
columns
from [Adventure Works DW2012]
```

Messages Results

1	19	2	20	21	22	23	25	26	27	28	29	3	30	31	32	37
4970	2190	8068	1430	3332	1019	568	562	328	249	7981	908	2167	2121	6440	733	17332

combine two levels on the Columns axis

- Combine 2 levels together with **{ }**

```
--combine two level
Select non empty{[Dim Product].[Hierarchy].[Product Subcategory
Key].members,
[Dim Product].[Hierarchy].[ALL]} on columns
from [Adventure Works DW2012]
```

Messages | Results

1	19	2	20	21	22	23	25	26	27	28	29	3	30	31	32	37	All
4970	2190	8068	1430	3332	1019	568	562	328	249	7981	908	2167	2121	6440	733	17332	60398

MDX syntax

```
[ WITH MEMBER <member_name>  AS <value_expr> |
SET <set_name>  AS <set_expr>
SELECT <axis_specification> [, <axis_specification>, …]
FROM    <cube_specification>
[ WHERE <slicer_specification>]
```

Using ORDER

- To sort your rows (or columns), you employ the Order function
  - Order (param 1, param 2, option)
  - Param 1: the set of rows to sort
  - Param 2: is the measure to sort by
  - Option: asc, desc. Default is asc

```
Select non empty[Measures].[Sales Amount -
Fact Reseller Sales] on columns,
order ([Dim Product].[English Product
Name].[English Product
Name],[Measures].[Sales Amount - Fact
Reseller Sales]) on rows
from [Adventure Works DW2012];
```

	Sales Amount - Fact Reseller Sales
Mountain-500 Black, 42	109833.966
Women's Mountain Shorts, L	111367.6472
HL Road Front Wheel	112286.412
Mountain-500 Silver, 48	114647.7708
Women's Mountain Shorts, S	115887.1677
Mountain-500 Silver, 42	115935.948
Mountain-400-W Silver, 42	118193.664
ML Mountain Rear Wheel	119193.5227
Mountain-500 Silver, 40	119664.882
Mountain-400-W Silver, 46	121158.0473
Mountain-500 Black, 48	127329.642

Using ORDER

```
Select non empty[Measures].[Sales Amount - Fact Reseller Sales] on columns,
order ([Dim Product].[English Product Name].[English Product Name],[Measures].[Sales Amount - Fact Reseller Sales]) on rows
from [Adventure Works DW2012];
```

```
Select non empty[Measures].[Sales Amount Fact Reseller Sales] on columns,
order ([Dim Product].[English Product Name].[English Product Name],[Measures].[Sales Amount - Fact Reseller Sales]) on rows
from [Adventure Works DW2012];
```

MDX Filter function

- Let’s filter the product subcategories to hide those with a null (or zero) Internet Sales Amount.
  - **Filter (param_1, param_2)**
    - The first parameter for Filter is the set of members you wish to filter.
    - The second parameter is a Boolean test that returns true or false for each member of the set.

```
Select {[Measures].[Sales Amount],
[Measures].[Sales Amount - Fact Reseller Sales]} on columns,
filter([Dim Product].[Product Subcategory Key].[Product
Subcategory Key],[Measures].[Sales Amount] > 0) on rows
from [Adventure Works DW2012];
```

MDX Filter function

- As an alternative to the Filter function: **Having clause**

```
Select
{[Measures].[Sales Amount],[Measures].[Sales Amount - Fact
Reseller Sales]} on columns,
[Dim Product].[Product Subcategory Key].[Product Subcategory Key]
having [Measures].[Sales Amount] > 0 on rows
FROM [Adventure Works DW2012];
```

Complex Filter with and/or

```
Select
{[Measures].[Sales Amount],[Measures].[Sales Amount - Fact
Reseller Sales]}on columns,
filter([Dim Product].[Product Subcategory Key].[Product
Subcategory Key],[Measures].[Sales Amount] > 20000
and [Measures].[Sales Amount - Fact Reseller Sales] <
75000) on rows
from [Adventure Works DW2012];
```

TopCount

- Get the top five in the list of subcategories
  - TopCount(Set_Expression,Count [,Numeric_Expression])
    - *Set_Expression: a valid* expression that returns a set
    - Count: A valid numeric expression that specifies the number of tuples to be returned
    - *Numeric_Expression:* A valid numeric expression of cell coordinates that return a number.

```
select [Measures].[Sales Amount] on columns,
(topcount([Dim Product].[Product Subcategory Key].[Product Subcategory
Key].members,5,[Measures].[Sales Amount])) on rows
from [Adventure Works DW2012];
```

Calculation – with clause

- Return total sales (that is, Internet or customer sales and reseller or retailer sales)

```
with member [Measures].[Customer Sales] as [Measures]
.[Sales Amount]
member [Measures].[Retailer Sales] as [Measures].[Sales Amount - Fact Reseller Sales]
member [Measures].[Total Sales] as [Measures]
.[Sales Amount]+[Measures].[Sales Amount - Fact Reseller Sales]
select
{[Measures].[Customer Sales],[Measures].[Retailer Sales],
[Measures].[Total Sales]}
on columns,
[Order Date].[Hierarchy].[Year] on rows
from [Adventure Works DW2012];
```

Calculation - With Clause

- MDX allows you to extend the cube functionality temporarily by using a With clause before the Select statement

```
with member [Measures].[My Measure] as "Hello world"
select [Measures].[My Measure] on columns,
 [Order Date].[Hierarchy].[Year] on rows
from [Adventure Works DW2012];
```

	My Measure
2005	Hello world
2006	Hello world
2007	Hello world
2008	Hello world
2009	Hello world
2010	Hello world
2011	Hello world
2012	Hello world
2013	Hello world
2014	Hello world

Crossjoint

- If you crossjoin two sets of members from the same dimension, the two sets must be based on different hierarchies within the same dimension. You can even crossjoin two different dimensions
- **Syntax**:
  - Standard syntax
    - **Crossjoin(Set_Expression1 ,Set_Expression2 [,...n])**
  - Alternate syntax
    - **Set_Expression1 * Set_Expression2 [* ...n]**

Crossjoin

- **Example**

```
-- crossjoin working
select [Measures].[Sales Amount] on columns,
crossjoin([Order Date].[Hierarchy].[Year],
[Order Date].[Month].[Month]) on rows
from [Adventure Works DW2012];
```

Messages | Results

		Sales Amount
2011	April	502073.845800001
2011	August	614557.935000001
2011	December	669431.503100001
2011	February	466334.903
2011	January	469823.9148
2011	July	596746.556800001
2011	June	737839.821400002
2011	March	485198.659400001
2011	May	561681.475800001

```
-- crossjoin not working
select [Measures].[Sales Amount] on columns,
crossjoin([Order Date].[Hierarchy].[Year],[Order Date].[Hierarchy].[Month] )
on rows
from [Adventure Works DW2012];
```

Messages

```
Executing the query ...
Query (2, 1) The Hierarchy hierarchy is used more than once in the Crossjoin function.
Execution complete
```

Crossjoin - adding another measure

```
select
{[Measures].[Sales Amount],[Measures].[Sales
Amount - Fact Reseller Sales]}
on columns,
crossjoin([Order Date].[Hierarchy].[Year],
[Order Date].[Month].[Month])
on rows
from [Adventure Works DW2012];
```

		Sales Amount	Sales Amount - Fact Reseller Sales
2011	April	502073.845800001	(null)
2011	August	614557.935000001	3356069.34399998
2011	December	669431.503100001	2393689.5255
2011	February	466334.903	(null)
2011	January	469823.9148	1538408.3122
2011	July	596746.556800001	713116.694300001
2011	June	737839.821400002	(null)
2011	March	485198.659400001	2010618.07409999
2011	May	561681.475800001	4027080.3403
2011	November	660545.813200001	1001803.7697
2011	October	708208.003200001	2269116.7118
2011	September	603083.497600001	882899.942400001
2012	April	400335.6145	3053816.326
2012	August	523917.3815	1563955.08079999

A Second Crossjoin on a Second Axis

- If you have two axes, you can have a separate crossjoin on each axis

```
select
crossjoin([Dim Product].[Hierarchy].[Product Subcategory Key],
{[Measures].[Sales Amount],[Measures].[Sales Amount - Fact Reseller Sales]})
on columns,
crossjoin([Order Date].[Hierarchy].[Year],
[Order Date].[Month].[Month]) on rows
from [Adventure Works DW2012];
```

		1	1	10	11
		Sales Amount	Sales Amount - Fact Reseller Sales	Sales Amount - Fact Reseller Sales	Sales Amount - Fact Reseller Sales
2010	December	16974.95	237464.298	(null)	(null)
2011	April	88024.74	(null)	(null)	(null)
2011	August	71074.79	1837014.576	(null)	(null)
2011	December	153893.2842	713334.394000001	4638.636	2550.9076
2011	February	101599.7	(null)	(null)	(null)

Remarks

- You can’t have the same hierarchy from the same dimension on more than one axis

```
select
[Order Date].[Hierarchy].[Year] on columns,
[Order Date].[Hierarchy].[Month] on rows
from [Adventure Works DW2012];
```

```
Executing the query ...
The Hierarchy hierarchy already appears in the Axis0 axis.
Execution complete
```

```
--solution
Select non empty [Order Date].[Hierarchy].[Year] on columns,
non empty [Order Date].[Month].[Month] on rows
from [Adventure Works DW2012];
```

Calculated member

- MDX uses the keywords **MEMBER** and **AS** in the **WITH** clause for creating calculated members
- EX: Calculated Profit for every product subcategory by years on the MDX expression:

```
WITH MEMBER MEASURES.[Profit] AS
[Measures].[Sales Amount] - [Measures].[Total Product Cost]
SELECT measures.profit ON COLUMNS,
non empty crossjoin ([Dim Product].[English Product Subcategory
Name].members,[Order Date].[Calendar Year].members) ON ROWS
FROM [DW2008R2];
```

Pentaho BI suite

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

Microsoft BI suite

- SSIS (QL Server Integration Services),
- SSAS (SQL Server Analytical Services)
- SSRS (SQL Server Reporting Services)
- MS PowerBI
- **The Best Alternatives:**
  - **Sisense**
  - **QlikView**
  - **Information Builders**
  - **BusinessObjects (SAP)**
  - **……**