

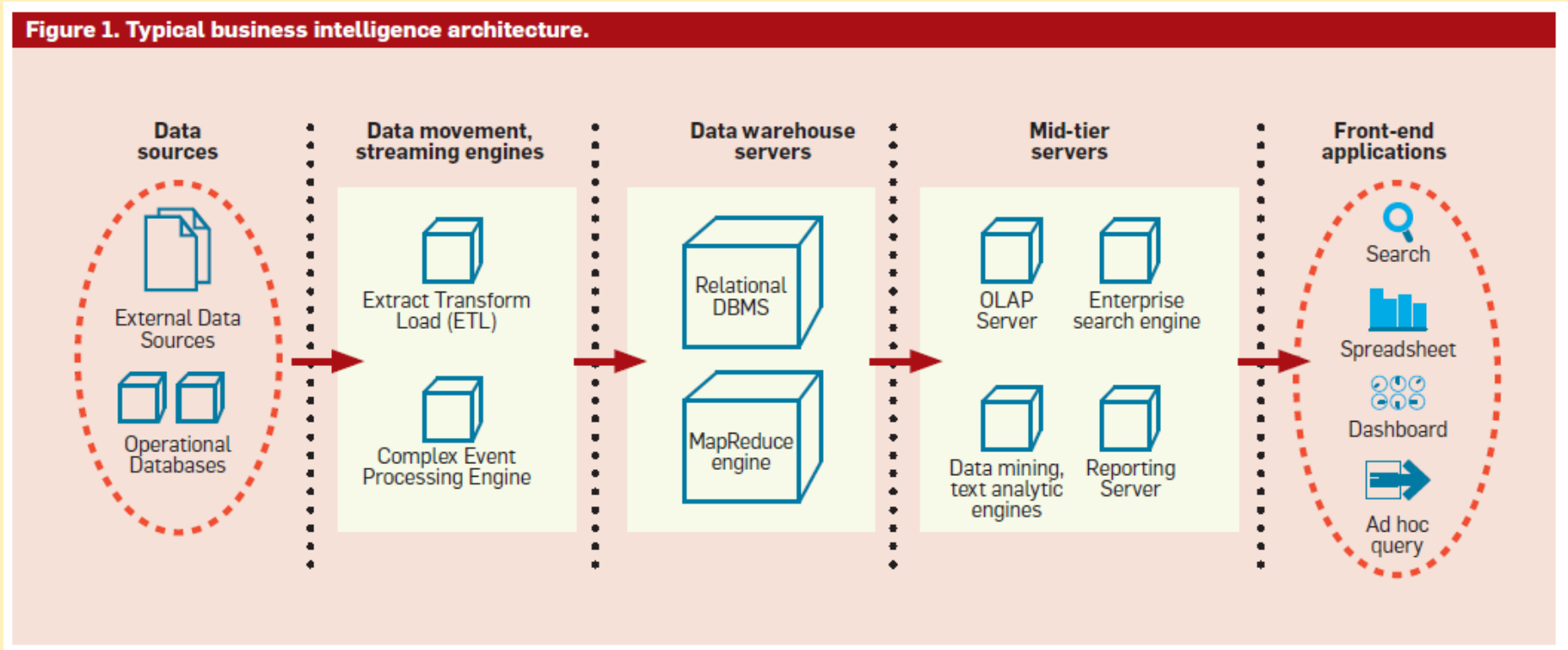
BUSINESS INTELLIGENCE

Data Analysis using SQL



BI Architecture

2



DATA ANALYSIS USING SQL

A Data warehouse is all about **getting answers** to business questions, in the form of **reports**.

Reports must communicate pertinent information clearly and concisely.

Good reporting is imperative: Even the best schema design cannot guarantee success if answers are not delivered with useful reports.

Three ways to **present** information.

- Traditional reports.

- Pivot tables.

- Charts.

There are several kinds of **reporting tools** on the market.

Cuboids in SQL

Order or pivoting

Aggregate

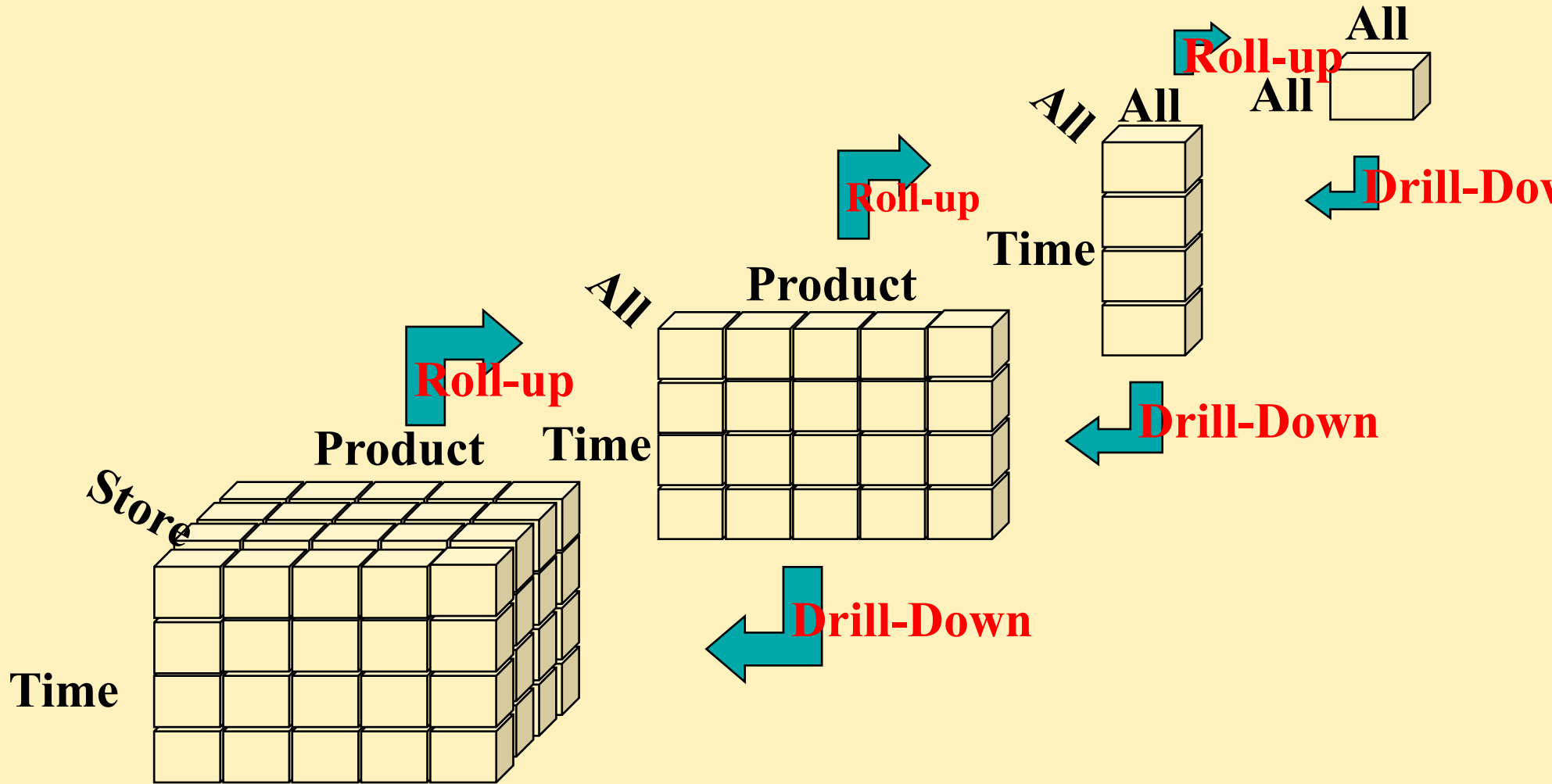
Measure

```
SELECT L.city, I.brand, T.month, SUM(dollars_sold)
FROM fact AS F, location AS L, time AS T, item AS I
WHERE F.location_key = L.location_key AND F.time_key =
  T.time_key AND
  F.item_key = I.item_key
GROUP BY L.city, I.brand, T.month
```

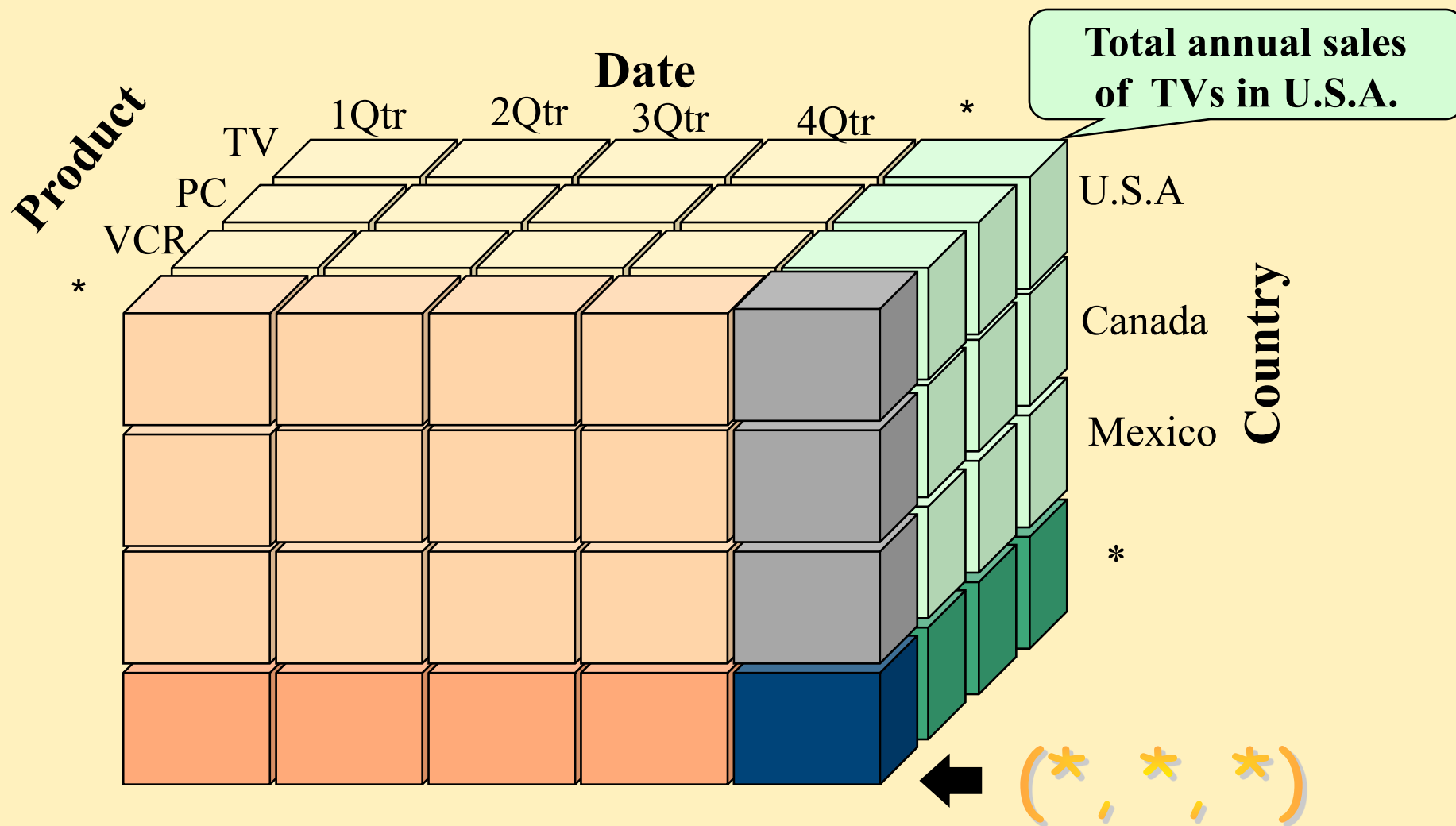
Star-Join

Hierarchy levels

How many cuboids?



Data Cube (extended cube, hypercube)



Data cube in SQL Server

Order or pivoting

Aggregate

Measure

```
SELECT L.city, I.brand, T.month, SUM(dollars_sold)
FROM fact AS F, location AS L, time AS T, item AS I
WHERE F.location_key = L.location_key AND F.time_key =
      T.time_key AND F.item_key = I.item_key
GROUP BY CUBE(L.city, I.brand, T.month)
```

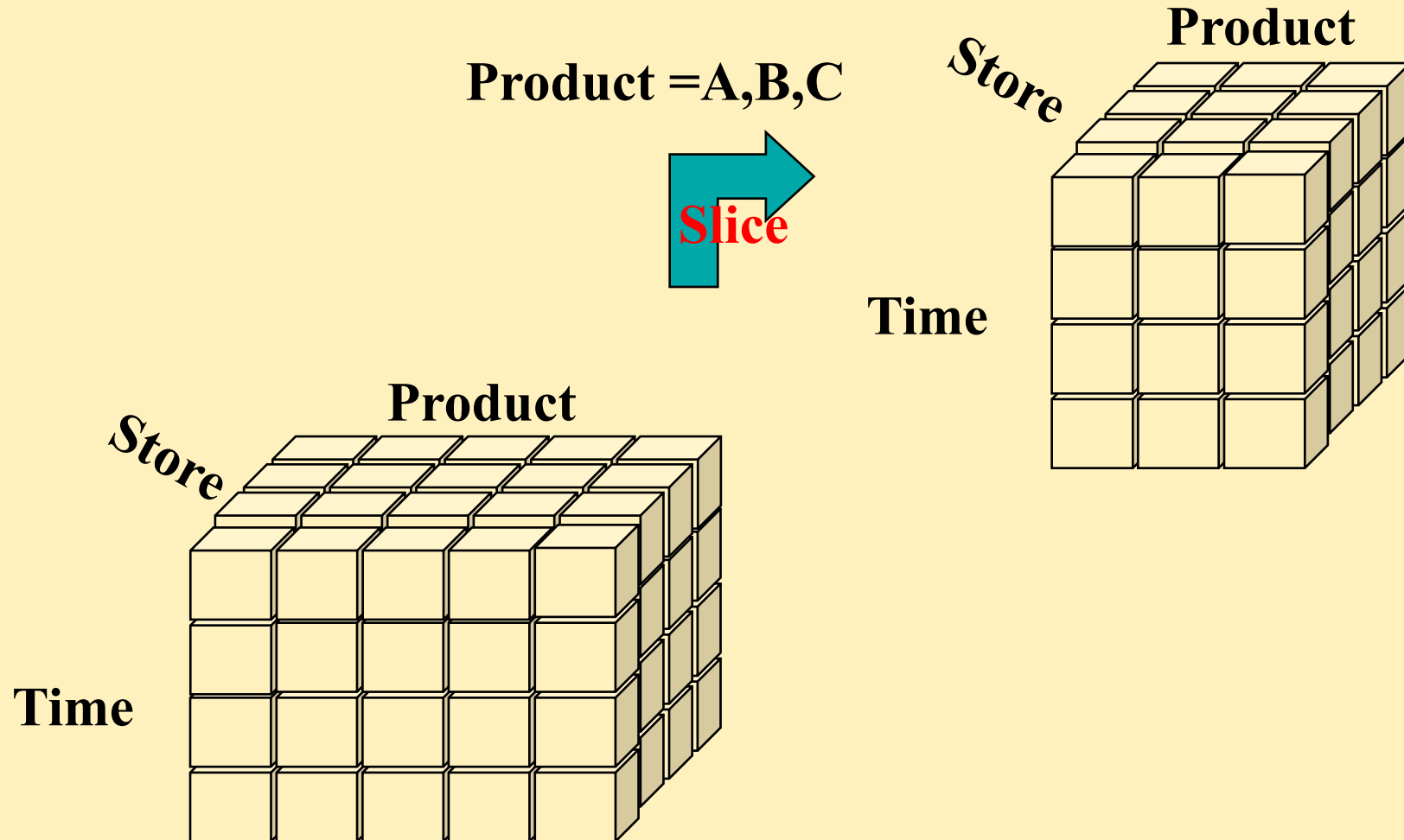
Star-Join

Hierarchy levels

```
GROUP BY ROLLUP(L.city, I.brand, T.month)
- all initial subsequences of the group-by attributes
```

Slice and Dice

8



Slice in SQL Server

9

Order or
pivoting

Aggregate

Measure

```
SELECT L.city, I.brand, T.month, SUM(dollars_sold)
FROM fact AS F, location AS L, time AS T, item AS I
WHERE F.location_key = L.location_key AND F.time_key =
T.time_key AND
F.item_key = I.item_key AND
T.year = 2016
GROUP BY CUBE(L.city, I.brand, T.month)
```

Slice

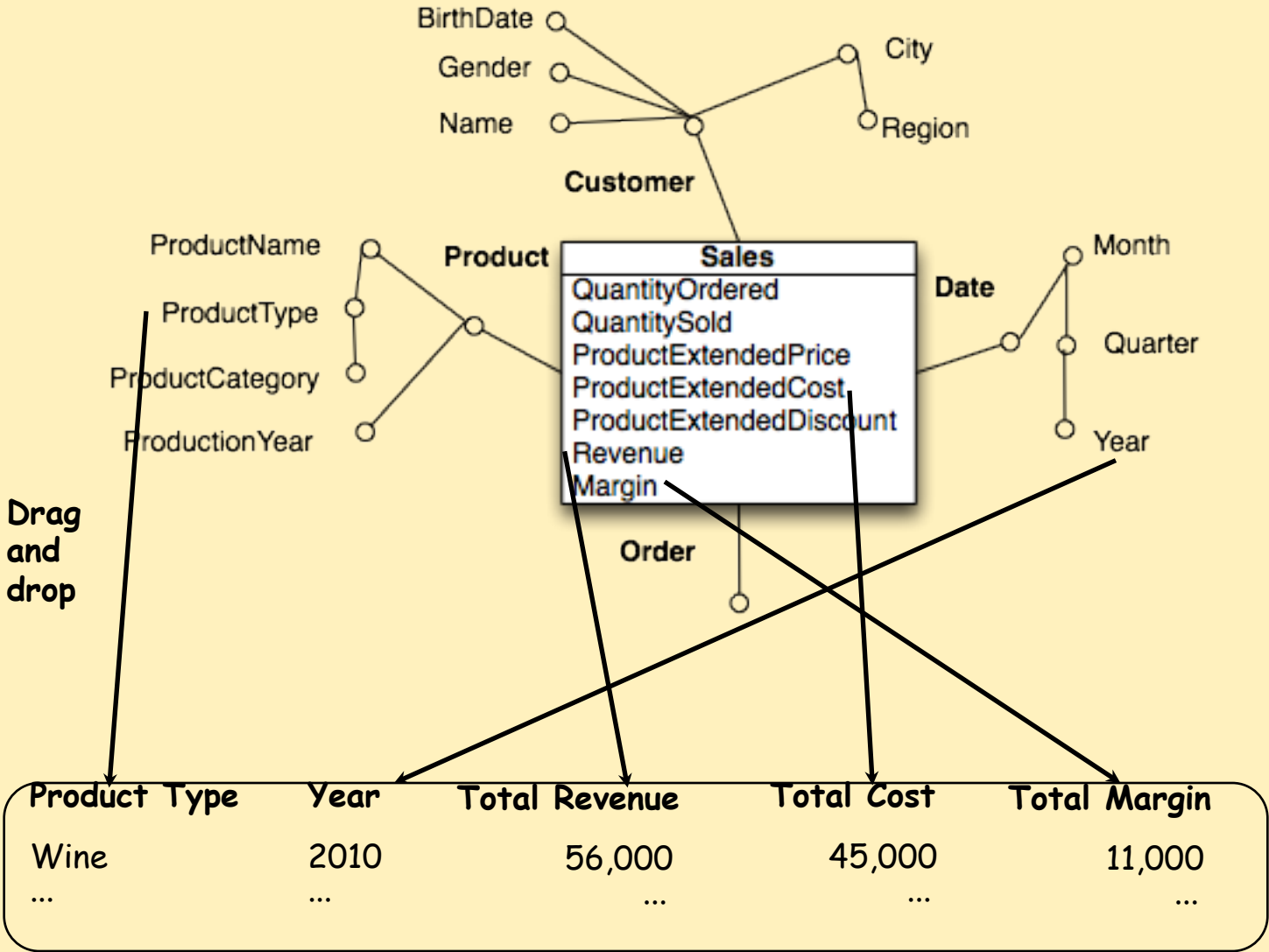
Star-Join

Hierarchy levels

Star-join executions in SQL Server

- ¹⁰ Star-join optimization
 - automatically detected (vs to be setup in Oracle)
- Bitmap join indexes
 - not available (vs available in Oracle)
- Columnstore indexes (since SQL Server 2012)
 - see docs
 - <http://msdn.microsoft.com/en-us/library/gg492088.aspx>
 - Example (on a copy of sales_fact)
 - CREATE CLUSTERED COLUMNSTORE INDEX cci_sales ON sales_fact_copy

REPORTING TOOLS



SIMPLE REPORTS WITH SQL

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

Margin by Brand and by Product Year 2009				
Brand	Product	Revenue (€)	Margin (€)	Margin% (%)
B1	P1	2 100	273	13
	P2	3 720	624	17
	P3	15 300	1 803	12
B2	P4	12 600	756	6
	P5	22 500	2 196	10
	P6	48 300	4 496	9

```

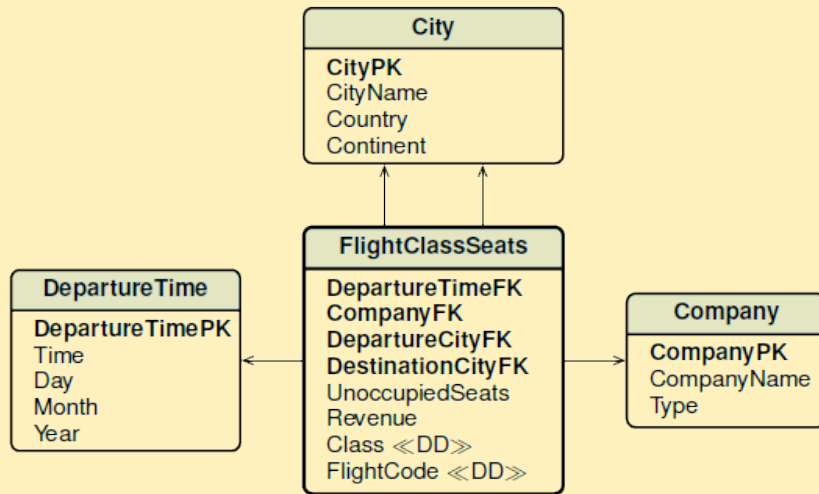
SELECT Brand, Product, SUM(Revenue) AS Revenue,
SUM(Margin) AS Margin,
ROUND(100*SUM(Margin)/SUM(Revenue)) AS Margin%
FROM Sales
WHERE YEAR(Date) = 2009
GROUP BY Brand, Product
ORDER BY Brand, Product;
    
```

Slice

Rollup & drill-down

Pivoting

AIRLINE COMPANIES: DATA ANALYSIS



Requirements analysis

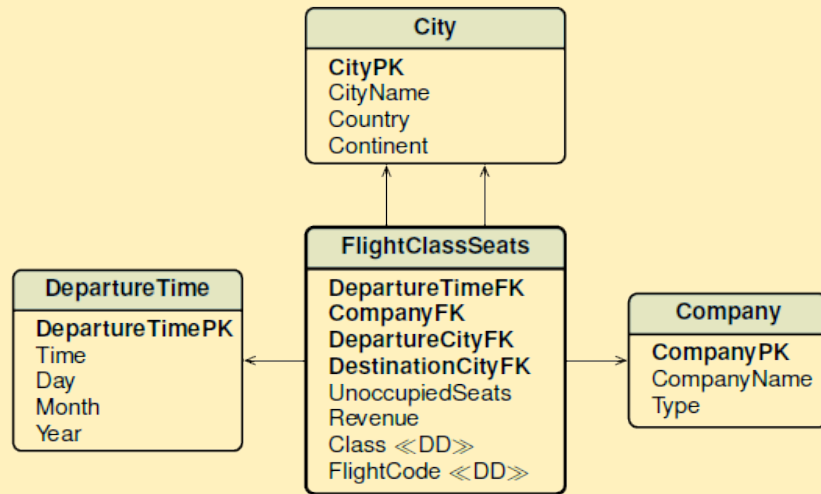
Number of unoccupied seats in a given year, by flight code, by company name (or type), by class, by departure time (time, day, month, year)

Number of unoccupied seats in a given class and year, by flight code, by company name, by class, by departure (destination) city (country, continent).

Number of unoccupied seats and revenue of the Alitalia company, by year, by month, by destination country.

```
SELECT FlightCode, CompanyName, Class, Time, SUM(UnoccupiedSeats) As TotalUnoccupiedSeats
FROM FlightClassSeats f, DepartureTime t, Company c
WHERE f.DepartureTimeFK = t.DepartureTimePK AND f.CompanyFK = c.CompanyPK and year = 2015
GROUP BY FlightCode, CompanyName, Class, Time,
```

AIRLINE COMPANIES: DATA ANALYSIS



Requirements analysis

Number of unoccupied seats in a given year, by flight code, by company name (or type), by class, by departure time (time, day, month, year)

Number of unoccupied seats in a given class and year, by flight code, by company name, by class, by departure (destination) city (country, continent).

Number of unoccupied seats and revenue of the Alitalia company, by year, by month, by destination country.

```
SELECT FlightCode, CompanyName, Class, City, SUM(UnoccupiedSeats) As TotalUnoccupiedSeats
FROM FlightClassSeats f, DepartureTime t, City c
WHERE f.DepartureTimeFK = t.DepartureTimePK AND f.DepartureCityFK = c.CityPK
      AND Class='Business' AND year = 2015
GROUP BY FlightCode, CompanyName, Class, City
```

```
SELECT year, month, country, SUM(UnoccupiedSeats) As TotalUnoccupiedSeats,
      SUM(Revenue) As TotalRevenue
FROM FlightClassSeats f, DepartureTime t, City c
WHERE f.DepartureTimeFK = t.DepartureTimePK AND f.DestinationCityFK= c.CityPK
      AND CompanyName='Alitalia'
GROUP BY year, month, country
```

SIMPLE REPORTS WITH SUBTOTALS

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

Margin by Brand and by Product Year 2009				
Brand	Product	Revenue (€)	Margin (€)	Margin% (%)
B1	P1	2 100	273	13
	P2	3 720	624	17
	P3	15 300	1 803	12
B1	Total	21 120	2 700	13
B2	P4	12 600	756	6
	P5	22 500	2 196	10
	P6	48 300	4 496	9
B2	Total	83 400	7 448	9
Total		104 520	10 148	10

SIMPLE REPORTS WITH SUBTOTALS IN SQL

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

Margin by Brand and by Product Year 2009				
Brand	Product	Revenue (€)	Margin (€)	Margin% (%)
B1	P1	2 100	273	13
	P2	3 720	624	17
	P3	15 300	1 803	12
B1	Total	21 120	2 700	13
B2	P4	12 600	756	6
	P5	22 500	2 196	10
	P6	48 300	4 496	9
B2	Total	83 400	7 448	9
Total		104 520	10 148	10

SQL: OPERATOR ROLLUP

GROUP BY ROLLUP(A, B)

Important the (attributes order)

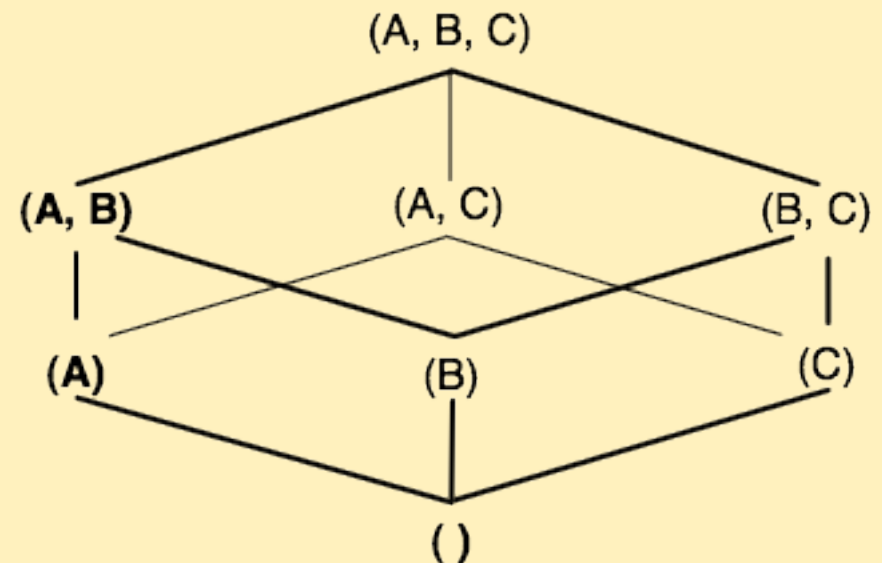
Semantics: Union of 3 groupings:

(A, B)

(A) subtotals

() totals

ROLLUP compute one path
through lattice



SIMPLE REPORTS WITH SUBTOTALS: ROLLUP

Margin by Brand and by Product
Year 2009

Brand	Product	Revenue (€)	Margin (€)	Margin% (%)
B1	P1	2 100	273	13
	P2	3 720	624	17
	P3	15 300	1 803	12
B1	Total	21 120	2 700	13
B2	P4	12 600	756	6
	P5	22 500	2 196	10
	P6	48 300	4 496	9
B2	Total	83 400	7 448	9
Total		104 520	10 148	10

(Brand, Product)

1

2

(Brand)

3

()

```

SELECT      Brand, Product, SUM(Revenue) AS Revenue,
            SUM(Margin) AS Margin,
            ROUND(100*SUM(Margin)/SUM(Revenue)) AS Margin%
FROM        Sales
WHERE       YEAR(Date) = 2009
GROUP BY   ROLLUP (Brand, Product)
ORDER BY   Brand, Product;
    
```

SIMPLE REPORTS WITH SUBTOTALS: CROSS-TABULATION

Product	Store			Total
	S1	S2	S3	
P1	300	500	50	850
P2	30	50	400	480
Total	330	550	450	1330

Margin by Brand and by Product Year 2009				
Brand	Product	Revenue (€)	Margin (€)	Margin% (%)
B1	P1	2 100	273	13
	P2	3 720	624	17
	P3	15 300	1 803	12
Total B1		21 120	2 700	13
B2	P4	12 600	756	6
	P5	22 500	2 196	10
	P6	48 300	4 496	9
Total B2		83 400	7 448	9
Total P1		2 100	273	13
Total P2		3 720	624	17
Total P3		15 300	1 803	12
Total P4		12 600	756	6
Total P5		22 500	2 196	10
Total P6		48 300	4 496	9
Total		104 520	10 148	10

SQL: OPERATOR CUBE

GROUP BY CUBE(A,B)

Important: the (attributes order) **doesn't matter**

Semantics: Union of 4 groupings:

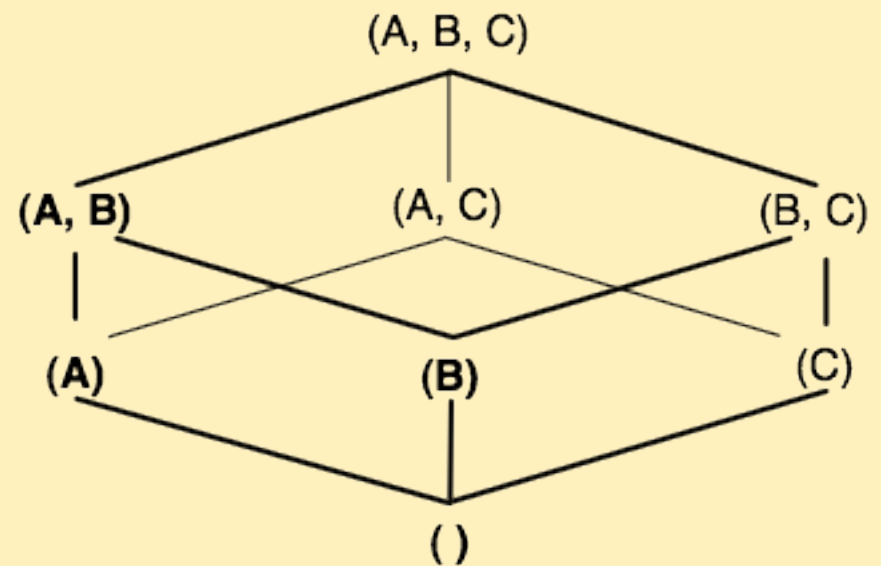
(A,B)

(A) subtotals

(B) subtotals

() totals

CUBE compute a sub-lattice



SIMPLE REPORTS WITH SUBTOTALS: CUBE

Margin by Brand and by Product Year 2009				
Brand	Product	Revenue (€)	Margin (€)	Margin% (%)
B1	P1	2 100	273	13
	P2	3 720	624	17
	P3	15 300	1 803	12
Total B1		21 120	2 700	13
B2	P4	12 600	756	6
	P5	22 500	2 196	10
	P6	48 300	4 496	9
Total B2		83 400	7 448	9
	Total P1	2 100	273	13
	Total P2	3 720	624	17
	Total P3	15 300	1 803	12
	Total P4	12 600	756	6
	Total P5	22 500	2 196	10
	Total P6	48 300	4 496	9
Total		104 520	10 148	10

Diagram annotations:

- 1 (Brand, Product): Yellow arrows pointing to the Brand and Product columns.
- 2 (Brand): Pink arrows pointing to the Brand column.
- 3 (Product): Blue arrow pointing to the Product column.
- 4 (): Green arrow pointing to the Total row.

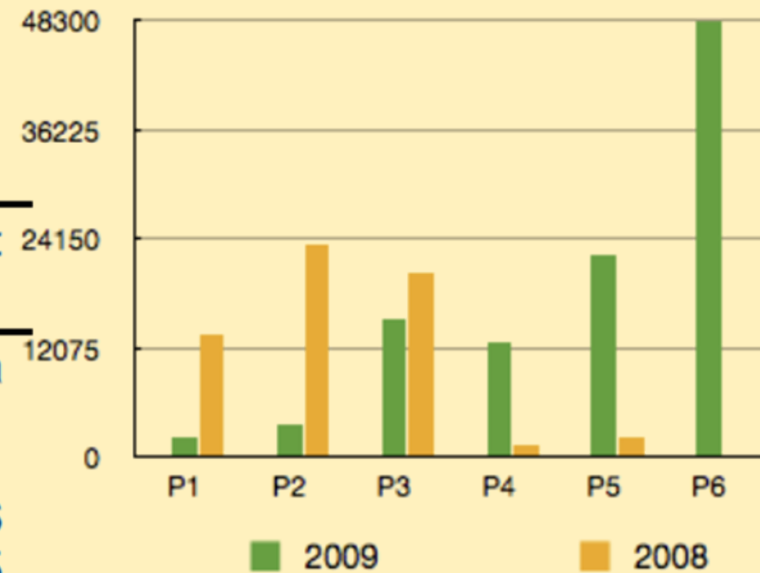
```

SELECT Brand, Product, SUM(Revenue) AS Revenue,
SUM(Margin) AS Margin,
ROUND(100*SUM(Margin)/SUM(Revenue)) AS Margin%
FROM Sales
WHERE YEAR(Date) = 2009
GROUP BY CUBE (Brand, Product)
ORDER BY Brand, Product;
    
```

MODERATELY DIFFICULT REPORTS WITH COMPARISON BETWEEN COLUMNS (VARIANCE REPORT)

Comparison between Revenue by Brand and by Product
2009 – 2008

Brand	Product	Revenue (€) 2009	Revenue (€) 2008	Delta (%)
B1	P1	2 100	13 560	-546
	P2	3 720	23 640	-535
	P3	15 300	20 340	-33
B2	P4	12 600	1 440	89
	P5	22 500	2 100	91
	P6	48 300		100



$$\text{Delta} = 100 \times (\text{Revenue}_{2009} - \text{Revenue}_{2008}) / \text{Revenue}_{2009}$$

A product may have been sold in one year, but not in the other !

JOIN

R

A	B
1	a
2	b

S

A	C
1	x
2	y

**SELECT *
FROM R
NATURAL JOIN
S;**

A	B	C
1	a	x
2	b	y

R

A	B
1	a
2	b
3	c

S

A	C
1	x
3	y
5	z

**SELECT *
FROM R
NATURAL JOIN
S;**

A	B	C
1	a	x
3	c	y

JOIN

R

A	B
1	a
2	b
3	c

S

A	C
1	x
3	y
5	z

```
SELECT *  
FROM R  
NATURAL FULL JOIN  
S;
```

A	B	C
1	a	x
2	b	
3	c	y
5		z

```
SELECT *  
FROM R  
FULL JOIN  
S USING (A);
```


SOLUTION USING VIEWS

Comparison between Revenue by Brand and by Product
2009 – 2008

Brand	Product	Revenue (€) 2009	Revenue (€) 2008	Delta (%)
B1	P1	2 100	13 560	-546
	P2	3 720	23 640	-535
	P3	15 300	20 340	-33
B2	P4	12 600	1 440	89
	P5	22 500	2 100	91
	P6	48 300		100

```
CREATE VIEW VRevenue09 AS
SELECT Brand, Product, SUM(Revenue) AS Revenue2009
FROM Sales WHERE Year(Data) = 2009
GROUP BY Brand, Product;
```

```
CREATE VIEW VRevenue08 AS
SELECT Brand, Product, SUM(Revenue) AS Revenue2008
FROM Sales WHERE Year(Data) = 2008
GROUP BY Brand, Product;
```

```
SELECT VRevenue09.Brand AS Brand, VRevenue09.Product AS Product, Revenue2009, Revenue2008,
CASE
    WHEN Revenue2009 IS NULL THEN -100
    WHEN Revenue2008 IS NULL THEN 100
    ELSE ROUND(100*(Revenue2009 - Revenue2008)/Revenue2009) END AS Delta
FROM VRevenue09 FULL JOIN VRevenue08 USING(Brand, Product)
ORDER BY Brand, Product
```

SOLUTION USING 'WITH' CLAUSE

```
WITH Revenue09 AS
( SELECT Brand, Product, SUM(Revenue) AS Revenue2009
  FROM Sales
  WHERE YEAR(Date) = 2009
  GROUP BY Brand, Product
)
, Revenue08 AS
( SELECT Brand, Product, SUM(Revenue) AS Revenue2008
  FROM Sales
  WHERE YEAR(Date) = 2008
  GROUP BY Brand, Product
)

SELECT Revenue09.Brand AS Brand, Revenue09.Product AS Product
, Revenue2009
, Revenue2008
, CASE
  WHEN Revenue2009 IS NULL THEN -100
  WHEN Revenue2008 IS NULL THEN 100
  ELSE ROUND(100*(Revenue2009 - Revenue2008)/ Revenue2009)
END AS Delta
FROM Revenue09 FULL JOIN Revenue08 USING (Brand, Product)
ORDER BY Brand, Product;
```

EXERCISE: MODERATELY DIFFICULT REPORTS WITH COMPARISON ACROSS AGGREGATION LEVELS

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

Revenue by Brand and Product January 2008				
Brand	Product	Revenue (€)	Percent of Brand Revenue	Percent of Total Revenue
M1	P1	175,000	45%	21%
	P2	96,000	25%	12%
	P3	114,000	30%	14%
M1	All products	385,000	100%	47%
M2	P4	102,400	23%	12%
	P5	96,200	22%	12%
	P6	124,000	28%	15%
	P7	120,000	27%	14%
M2	All products	442,600	100%	53%
All brands		827,000		100%

VERY DIFFICULT REPORTS WITHOUT ANALYTIC SQL: RUNNING TOTALS

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

Product P1 Revenue by Quarter and Month Year 2009				
Quarter	Month	Revenue (€)	Revenue QtoD (€)	Revenue YtoD (€)
Q1	January	16 500	16 500	16 500
Q1	February	14 220	30 720	30 720
Q1	March	27 480	58 200	58 200
Q2	April	7 920	7 920	66 120
Q2	May	1 200	9 120	67 320
Q2	June	1 260	10 380	68 580
Q3	July	5 400	5 400	73 980
Q3	August	11 730	17 130	85 710
Q3	September	10 860	27 990	96 570
Q4	October	5 850	5 850	102 420
Q4	November	2 100	7 950	104 520
Q4	December			

VERY DIFFICULT REPORTS WITHOUT ANALYTIC SQL: RANK

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

Revenues and Ranks in the 2009 by Region and by Product				
Region	Product	Total Revenue	Product Rank by Region	Product Rank Global
Lazio	P3	2880	3	4
	P2	960	5	8
	P4	2700	4	5
	P1	480	6	10
	P5	4800	2	2
	P6	11400	1	1
Toscana	P1	120	6	12
	P6	3600	1	3
	P3	1800	2	6
	P5	1500	3	7
	P4	900	4	9
	P2	240	5	11

Which are the **best 5** products sold in Toscana?

VERY DIFFICULT REPORTS WITHOUT ANALYTIC SQL

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

We want to partition the customers into four groups:

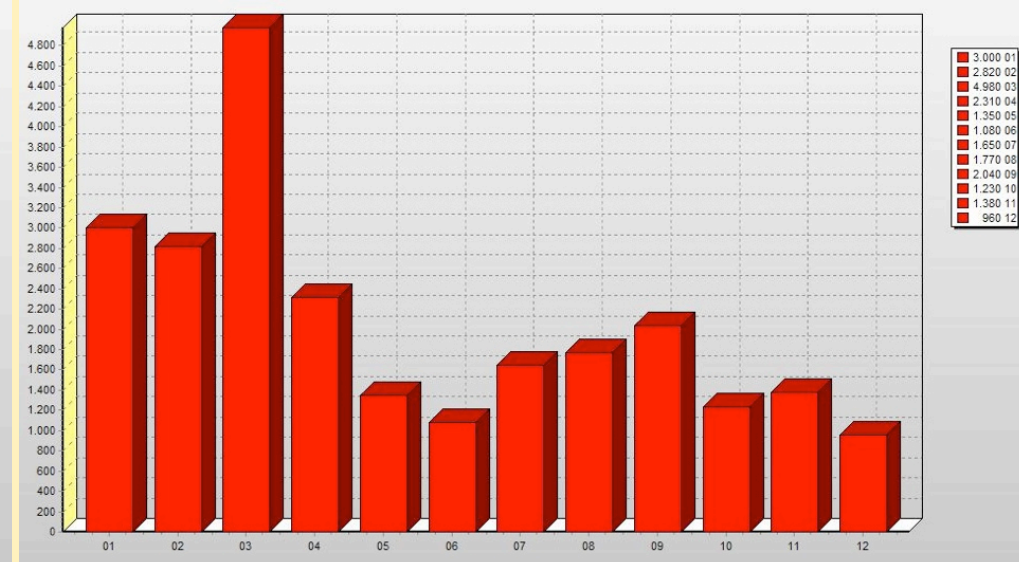
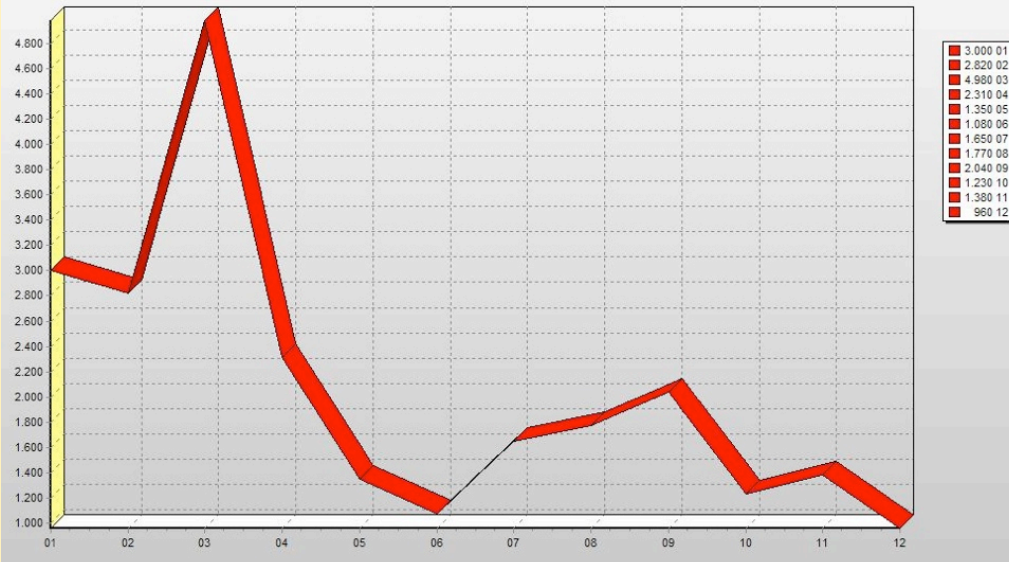
- **Top5%**, with 5% of customers with the highest amount of revenues.
- **Next15%**, with 15% of other customers with the highest amount of revenues.
- **Middle30%**, with 30% of other customers with the highest amount of revenues.
- **Bottom50%**, with 50 % of the customers with the lowest amount of revenues.

For each customer group we want to know their number, and the percentage of the sum of their revenues compared to total revenue of all sales.

Group	Number of customers	Percent of total revenue
Top5%	1	20
Next15%	3	50
Middle30%	6	20
Bottom50%	10	10

VERY DIFFICULT REPORTS WITHOUT ANALYTIC SQL

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)



Monthly Total Revenue



Moving Average Monthly Total Revenue (Window 3 or 5)

Syntax

SELECT Select Attributes (S_A), Select Aggregation Functions (S_{AF}),

FROM Fact table (F) and a dimension table (D1)

WHERE Where condition (W_C)

GROUP BY Grouping Attributes (G_A)

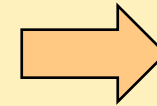
HAVING Having condition (H_C) with aggregation functions (H_{AF})

ORDER BY Sorting attributes (O_A);

Intuition: Partition By

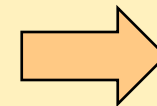
R	
P	...
P1	...
P1	...
P2	...
P2	...
P2	...
P2	...
P2	...

```
SELECT P, COUNT(*) AS No
FROM R
GROUP BY P;
```



P	No
P1	2
P2	5

```
SELECT P,
COUNT(*) OVER (PARTITION BY P) AS No
FROM R
ORDER BY P;
```

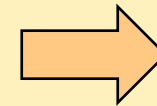


P	No
P1	2
P1	2
P2	5
P2	5
P2	5
P2	5

Intuition: without Partition By

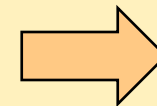
R	
P	...
P1	...
P1	...
P2	...
P2	...
P2	...
P2	...
P2	...

```
SELECT COUNT(*) AS No  
FROM R
```



No
7

```
SELECT P, COUNT(*) OVER() AS No  
FROM R  
ORDER BY P;
```



P	No
P1	7
P1	7
P2	7
P2	7
P2	7
P2	7
P2	7

ANALYTIC SQL

Execution order

ORDER BY O_A

SELECT $S_A, S_{AF},$

A_F OVER (...)

HAVING H_C

GROUP BY G_A

WHERE W_C

FROM F, D1

SELECT Select Attributes (S_A), Select Aggregation Functions (S_{AF}),
Analytic Function (A_F) OVER(
[PARTITION BY <attribute list>]
[ORDER BY <sort attribute list>
[<window clause>]])
FROM Fact table (F) and a dimension table (D1)
WHERE Where condition (W_C)
GROUP BY Grouping Attributes (G_A)
HAVING Having condition (H_C) with aggregation functions (H_{AF})
ORDER BY Sorting attributes (O_A);

RANK

```
SELECT Customer, Product, SUM(Revenue) AS TotalRev,  
       RANK ( ) OVER (ORDER BY SUM(Revenue) ) AS Rank  
FROM Sales WHERE Customer IN ('C1', 'C2')  
GROUP BY Customer, Product ORDER BY TotalRev DESC;
```

Customer	Product	TotalRev	Rank
C1	P1	1100	7
C1	P3	1000	6
C2	P1	1000	5
C2	P2	900	4
C2	P4	800	3
C1	P2	200	2
C2	P3	200	1

RANK WITH PARTITIONS

```
SELECT Customer, Product, SUM(Revenue) AS TotalRevenue,  
      RANK ( ) OVER (PARTITION BY Customer  
      ORDER BY SUM(Revenue) DESC) AS Rank  
FROM Sales WHERE Customer IN ('C1', 'C2')  
GROUP BY Customer, Product;
```

Customer	Product	TotalRev	Rank
C1	P1	1100	1
C1	P3	1000	2
C1	P2	200	3
C2	P1	1000	1
C2	P2	900	2
C2	P4	800	3
C2	P3	200	4

RANK vs DENSE_RANK vs ROW_NUMBER

<RankFunction>()

OVER(

[PARTITION BY <attribute list>]

ORDER BY <sort attribute list>

) [AS lde]

- Consider the values in the ascending order
 - (10; 20; 20; 30; 30; 40)
- RANK() of a value is 1 + the number of values that strictly precedes it
 - ranks (1; 2; 2; 4; 4; 6)
- DENSE_RANK() of a value is 1 + the number of distinct values that precedes it
 - dense ranks (1; 2; 2; 3; 3; 4)
- PERCENT_RANK() is $(RANK() - 1) / (TotalRows - 1)$
 - percent ranks (0; 0.2; 0.2; 0.6; 0.6; 1)
- ROW_NUMBER() is the row number
 - row numbers (1; 2; 3; 4; 5; 6)
- CUME_DIST() is $ROW_NUMBER() / TotalRows$
 - cumulative distribution (0.16; 0.33; 0.5; 0.67; 0.83; 1)
- NTILE(3) is the tertile of the value (3 is a parameter, can be any integer)
 - tertiles (1; 1; 2; 2; 3; 3)

OTHER ANALYTIC FUNCTIONS

- COUNT(), SUM(), AVG(), MIN(), MAX() ... and all standard aggregates

Sales(Brand, Product, Revenue)

Brand	Product	prodRevenue	PctOverBrand	PctOverTot
B1	P1	40	40	20
B1	P2	60	60	30
B2	P3	20	20	10
B2	P4	80	80	40

```
WITH s AS (SELECT Brand, Product, SUM(Revenue) AS prodRevenue
```

```
FROM sales
```

```
GROUP BY Brand, Product)
```

```
SELECT Brand, Product, prodRevenue,
```

```
100 * prodRevenue / SUM(prodRevenue) OVER(PARTITION BY Brand) AS PctOverBrand,
```

```
100 * prodRevenue / SUM(prodRevenue) OVER() AS PctOverTot
```

```
FROM s
```

```
SELECT Brand, Product, SUM(Revenue) AS prodRevenue,
```

```
100 * SUM(Revenue) / SUM(SUM(Revenue)) OVER(PARTITION BY Brand) AS PctOverBrand,
```

```
100 * SUM(Revenue) / SUM(SUM(Revenue)) OVER() AS PctOverTot
```

```
FROM sales
```

```
GROUP BY Brand, Product
```

OTHER ANALYTIC FUNCTIONS

- LAG(attribute, offset, default) and LEAD(attribute, offset, default)
 - The value of attribute in offset rows before (LAG) or after (LEAD)

SELECT Store, Year, TotalRev,

LEAD(TotalRev, 1, 0) OVER(PARTITION BY Store ORDER BY Year DESC) **AS** PrevRev,

FROM TotalSales

ORDER BY Store, Year

Store	Year	TotalRev	PrevRev
S1	2015	1100	1000
S1	2014	1000	200
S1	2013	200	0
S2	2015	1000	900
S2	2014	900	800
S2	2013	800	200
S2	2012	200	0

WINDOWING

```
<AggregateFunction>(<expr>)  
OVER(  
    [PARTITION BY <attribute list>]  
    [ORDER BY <sort attribute list>  
    [<ROWS or RANGE> <window size specification>]]  
    ) [ AS lde ]
```

Windowing functions are used to compute cumulative, moving and centered aggregates.

Window functions add a value to each row that depends on the other rows in the window.

Examples of window specifications:

ROWS UNBOUNDED PRECEDING. The window begin with the first record of the partition and ends with the current record.

ROWS BETWEEN ... PRECEDING AND ... FOLLOWING. The window include all records that fall within the given offset.

WINDOWING EXAMPLE

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

Product P1 Revenue by Quarter and Month Year 2009				
Quarter	Month	Revenue (€)	Revenue QtoD (€)	Revenue YtoD (€)
Q1	January	16 500	16 500	16 500
Q1	February	14 220	30 720	30 720
Q1	March	27 480	58 200	58 200
Q2	April	7 920	7 920	66 120
Q2	May	1 200	9 120	67 320
Q2	June	1 260	10 380	68 580
Q3	July	5 400	5 400	73 980
Q3	August	11 730	17 130	85 710
Q3	September	10 860	27 990	96 570
Q4	October	5 850	5 850	102 420
Q4	November	2 100	7 950	104 520
Q4	December			

WINDOWING EXAMPLE

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

Product P1 Revenue by Quarter and Month Year 2009				
Quarter	Month	Revenue (€)	Revenue QtoD (€)	Revenue YtoD (€)
Q1	January	16 500	16 500	16 500
Q1	February	14 220	30 720	30 720
Q1	March	27 480	58 200	58 200
Q2	April	7 920	7 920	66 120
Q2	May	1 200	9 120	67 320
Q2	June	1 260	10 380	68 580
Q3	July	5 400	5 400	73 980
Q3	August	11 730	17 130	85 710
Q3	September	10 860	27 990	96 570
Q4	October	5 850	5 850	102 420
Q4	November	2 100	7 950	104 520
Q4	December			

EXAMPLE

Sales(Customer, Product, Brand, Date, City, Region, Area, Quantity, Revenue, Margin)

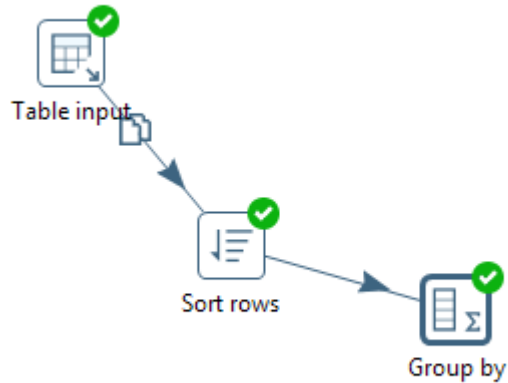
A moving average of total revenue, with a moving window of 3 months, by month.

```
SELECT    MONTH(Date) AS Month
```

```
FROM      Sales  
GROUP BY  MONTH(Date)  
ORDER BY  Month;
```

Result visualization in Oracle...

ANALYTIC FUNCTIONS IN PENTAHO DATA INTEGRATION



Group By

Step name:

Include all rows?

Temporary files directory:

TMP-file prefix:

Add line number, restart in each group

Line number field name:

Always give back a result row

The fields that make up the group:

#	Group field
1	store_id
2	customer_id

Aggregates:

#	Name	Subject	Type	Value
1	N		Number of rows (without field argument)	

Analytic Query

Step name:

The fields that make up the group:

#	Group field
1	product_id

Analytic Functions:

#	New Field Name	Subject	Type	N
1	prev_cust	customer_id	LAG "N" rows BACKWARD in get Subject	1

OR CONNECT TO SQL SERVER 2014 ON KDD.DI.UNIPI.IT

LOGIN: sobigdata PWD: pisa

Table input

Step name: Table input 2
Connection: sql

SQL

```
SELECT store_id, customer_id, rank() over (order by sum(store_sales))  
from sales_fact  
group by store_id, customer_id
```

Line 1 Column 52

Enable lazy conversion
Replace variables in script?
Insert data from step
Execute for each row?
Limit size: 0

Help OK Preview Cancel

Database Connection

General
Advanced
Options
Pooling
Clustering

Connection Name: FoodMart su SQL Server

Connection Type:

- MS Access
- MS SQL Server
- MS SQL Server (Native)**
- MaxDB (SAP DB)
- MonetDB
- MySQL
- Native Mondrian
- Neoview
- Netezza
- OpenERP Server
- Oracle
- Oracle RDB
- Palo MOLAP Server
- Pentaho Data Services

Settings

Host Name: kdd.di.unipi.it
Database Name: foodmart
Instance Name:
Port Number: 1433
User Name: sobigdata
Password: ●●●●
 Use Integrated Security
 Use .. to Separate Schema and Table

Access:
Native (JDBC)
ODBC
JNDI

Prova ta delle featu Esplora

OK Cancel

SUMMARY

SQL is not select-from-where **only**.

Grouping and **aggregation** is a major part of SQL.

SQL has been extended for OLAP operations, because of intensive data warehouse applications during the last decade.

Make sure you understand SQL. It is much more than syntax.

Open Lab

- Redo exercises on ETL using SQL queries instead

Exercise

- FoodMart data mart
 - Write a SQL query that returns all constant customers in June 1998
 - **Constant:** with at least two baskets per month for at least two months in the last four months.

Exercise

```
WITH salesagg AS (  
    SELECT customer_id, COUNT(DISTINCT s.time_id) AS npurchases  
    FROM sales_fact s, time_by_day t  
    WHERE s.time_id = t.time_id and  
           (t.the_year*12 + t.month_of_year) BETWEEN 1998*12+6-3 AND  
           1998*12+6  
    GROUP BY customer_id, the_year, month_of_year  
)  
SELECT customer_id  
FROM salesagg  
WHERE npurchases > 1  
GROUP BY customer_id  
HAVING Count(*) > 1
```