Database Systems and Administration
CS3543

Decision-Support Systems: Data Warehousing and advanced SQL

Suprio Ray
University of New Brunswick, Fredericton, Canada

Acknowledgement

Thanks to Ramakrishnan and Gehrke, C. J. Date, Silberchatz, Korth and Sudarshan and many articles for some of the materials in these slides.

Introduction

- Increasingly, organizations are analyzing current and historical data to identify useful patterns and support business strategies.

- Emphasis is on complex, interactive, exploratory analysis of very large datasets created by integrating data from across all parts of an enterprise; data is fairly static.
  - Contrast such On-Line Analytic Processing (OLAP) with traditional On-line Transaction Processing (OLTP): mostly long queries, instead of short update transactions.

OLTP vs. OLAP

	OLTP	**OLAP**
users	clerk, IT professional	knowledge worker, executives
function	day to day operations	**decision support**
DB design	application-oriented	subject-oriented
data	current, up-to-date detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
access	read/write index/hash on prim. key	lots of scans
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
#users	thousands	hundreds
DB size	100MB-GB	100GB-TB
metric	transaction throughput	query throughput, response

Src: C J Date

Decision Support Systems

- Decision-Support systems are used to make business decisions often based on data collected by on-line transaction-processing systems.

- Examples of business decisions:
  - What items to stock?
  - What insurance premium to change?
  - Who to send advertisements to?

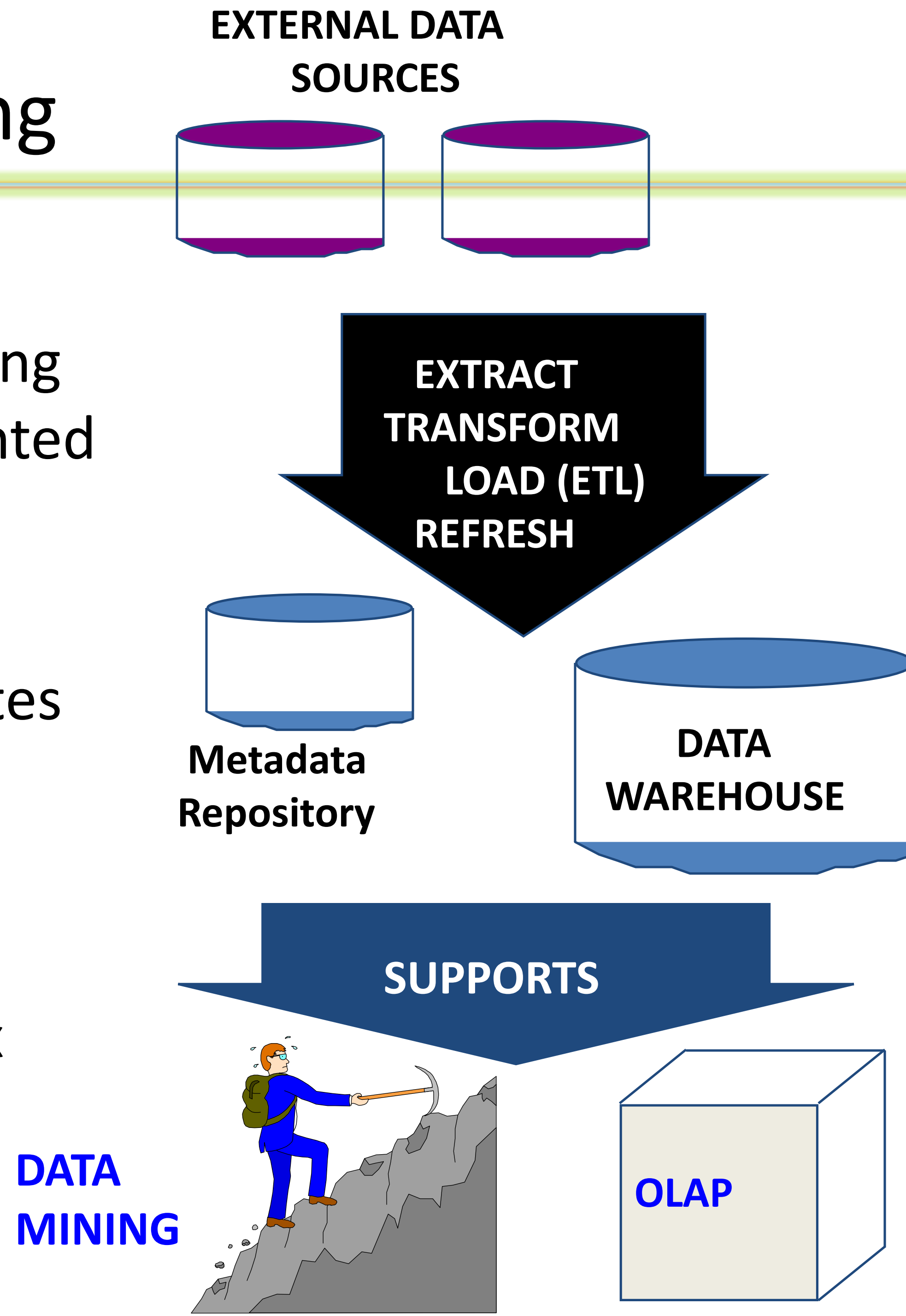
- Examples of data used for making decisions
  - Retail sales transaction details
  - Inventory transaction details
  - Customer profiles (income, age, sex, etc.)

Decision-Support Systems: Overview

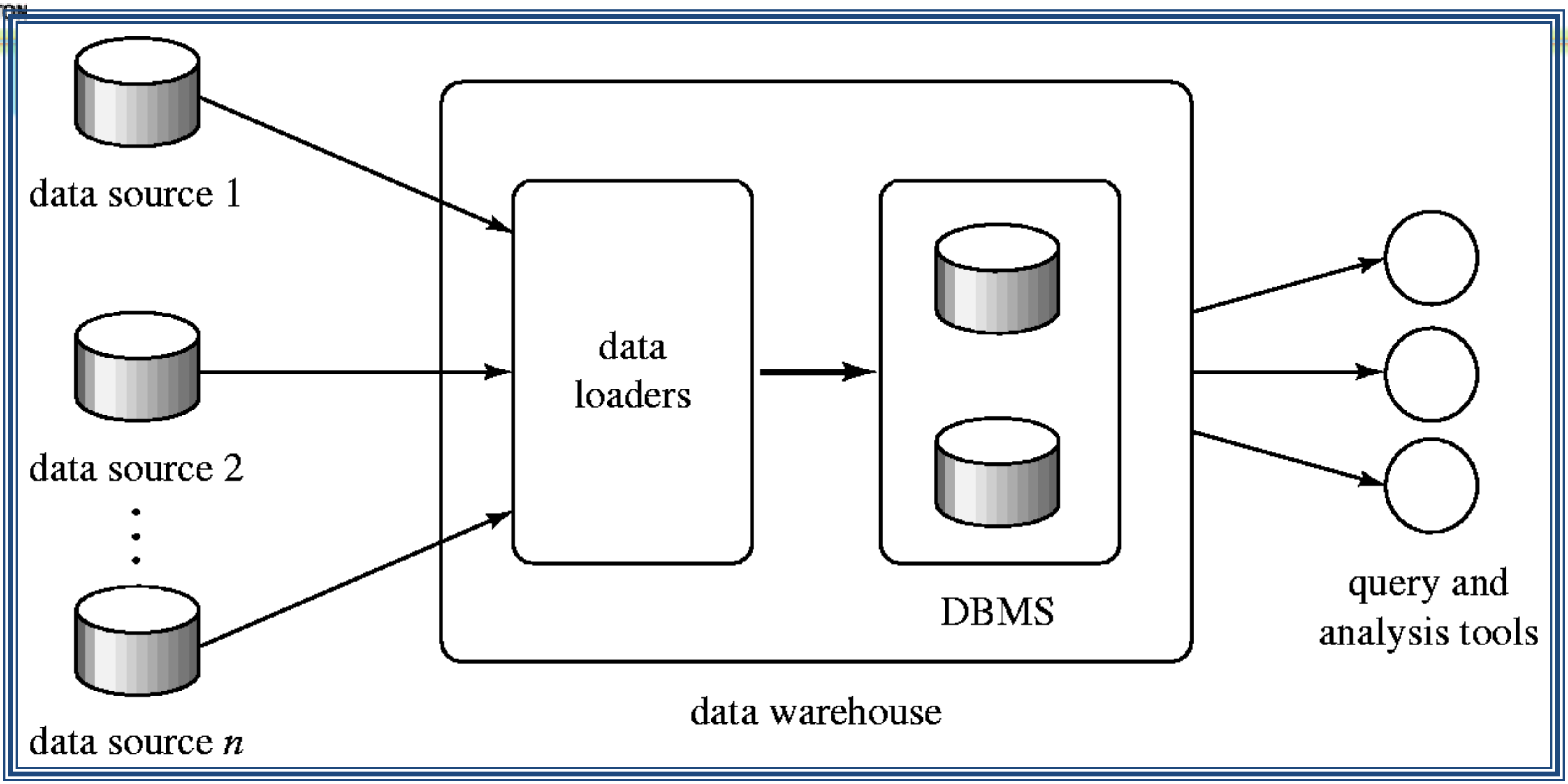
- Data Warehousing: Consolidate data from many sources in one large repository.
  - Loading, periodic synchronization of replicas (ETL)
  - Semantic integration
- OLAP: View data “dimensionally”, explore, navigate
  - Complex SQL queries and views.
  - Queries based on spreadsheet-style operations and “multidimensional” view of data.
  - Interactive and “online” queries.
- Data Mining: Exploratory search and analysis for interesting trends and anomalies.

Data Warehousing

- Integrated data spanning long time periods, often augmented with summary information.
- Several gigabytes to terabytes common.
- Need: interactive response times expected for complex queries; ad-hoc updates uncommon.



Data Warehousing (Cont.)



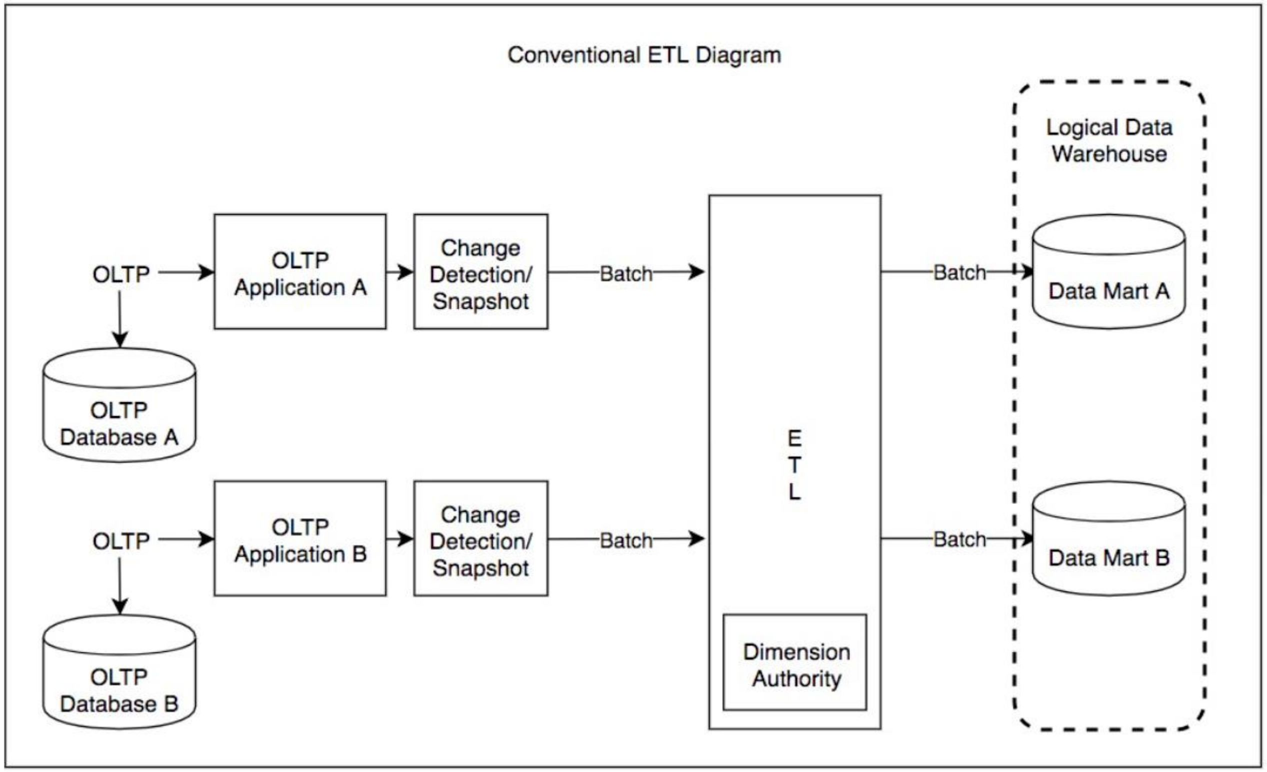
- A data warehouse is a repository of information gathered from multiple sources.

Data Warehousing (Cont.)

- Provides a single consolidated interface to data
- Data stored for an extended period, providing access to historical data
- Running large queries at the warehouse ensures that OLTP systems are not affected by the decision-support workload.
- Data/updates are periodically downloaded form online transaction processing (OLTP) systems.
  - Typically, download happens each night.
  - Data may not be completely up-to-date, but is recent enough for analysis.

Data Warehousing (Cont.)

- Data/updates are periodically downloaded form online transaction processing (OLTP) systems.
  – Typically, download happens each night.
  – Data may not be completely up-to-date, but is recent enough for analysis.



Src: wikipedia

Data Warehousing Issues

- Semantic Integration: When getting data from multiple sources, must eliminate mismatches, e.g., different currencies, schemas.
- Heterogeneous Sources: Must access data from a variety of source formats and repositories.
  - Replication capabilities can be exploited here.
- Load, Refresh, Purge: Must load data, periodically refresh it, and purge too-old data.
- Metadata Management: Must keep track of source, loading time, and other information for all data in the warehouse.

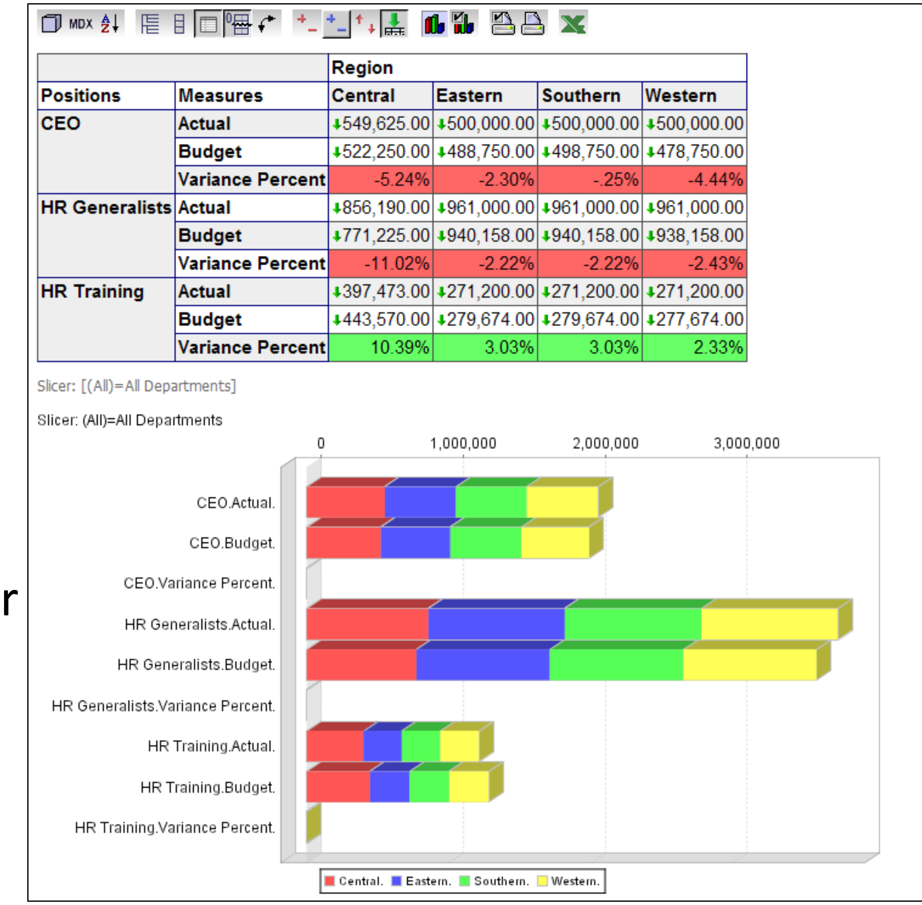
OLAP

- View data "dimensionally"
  - i.e. Sales by region, by channel, by time period

pid	timeid	locid	sales
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35

OLAP

- View data "dimensionally"
  - i.e. Sales by region, by channel, by time period
- Navigate and explore
  - Ad Hoc analysis
  - "Drill-down" from year to quarter
  - Pivot
  - Select specific members for analysis
- Interactive response



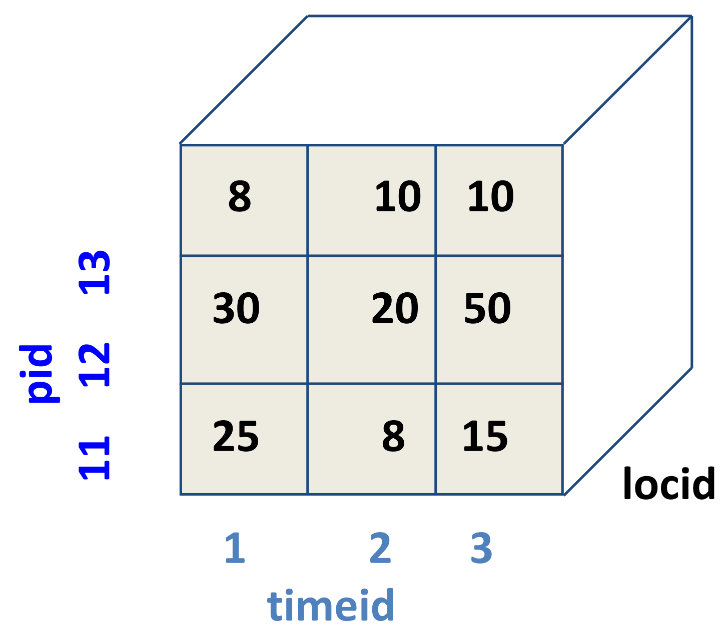
		Region			
Positions	Measures	Central	Eastern	Southern	Western
CEO	Actual	549,625.00	500,000.00	500,000.00	500,000.00
	Budget	522,250.00	488,750.00	498,750.00	478,750.00
	Variance Percent	-5.24%	-2.30%	-.25%	-4.44%
HR Generalists	Actual	856,190.00	961,000.00	961,000.00	961,000.00
	Budget	771,225.00	940,158.00	940,158.00	938,158.00
	Variance Percent	-11.02%	-2.22%	-2.22%	-2.43%
HR Training	Actual	397,473.00	271,200.00	271,200.00	271,200.00
	Budget	443,570.00	279,674.00	279,674.00	277,674.00
	Variance Percent	10.39%	3.03%	3.03%	2.33%

Src: J Hyde

Multidimensional Data Model

- Collection of numeric measures, which depend on a set of dimensions.
  - E.g., measure **Sales**, Revenue, Budget, Expenditure
  - dimensions **Product** (key: pid), **Location** (locid), and **Time** (timeid).

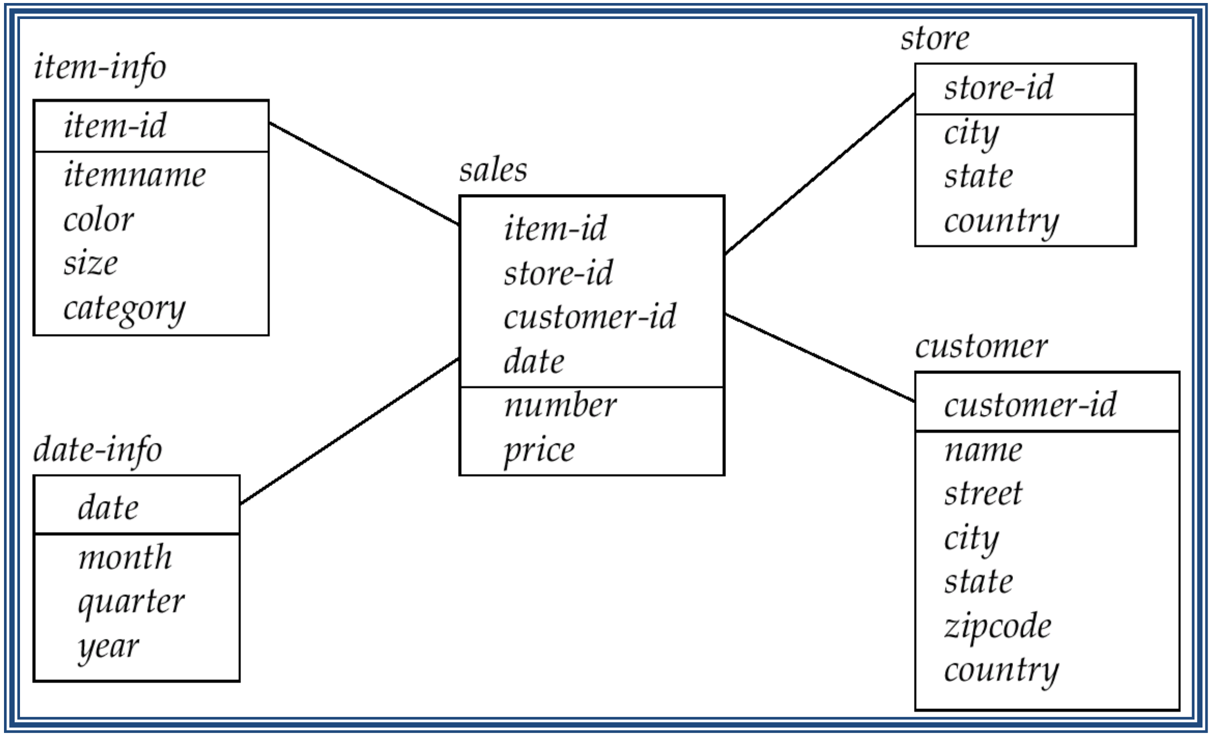
Slice locid=1 is shown:



pid	timeid	locid	sales
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35

OLAP - Star Schema

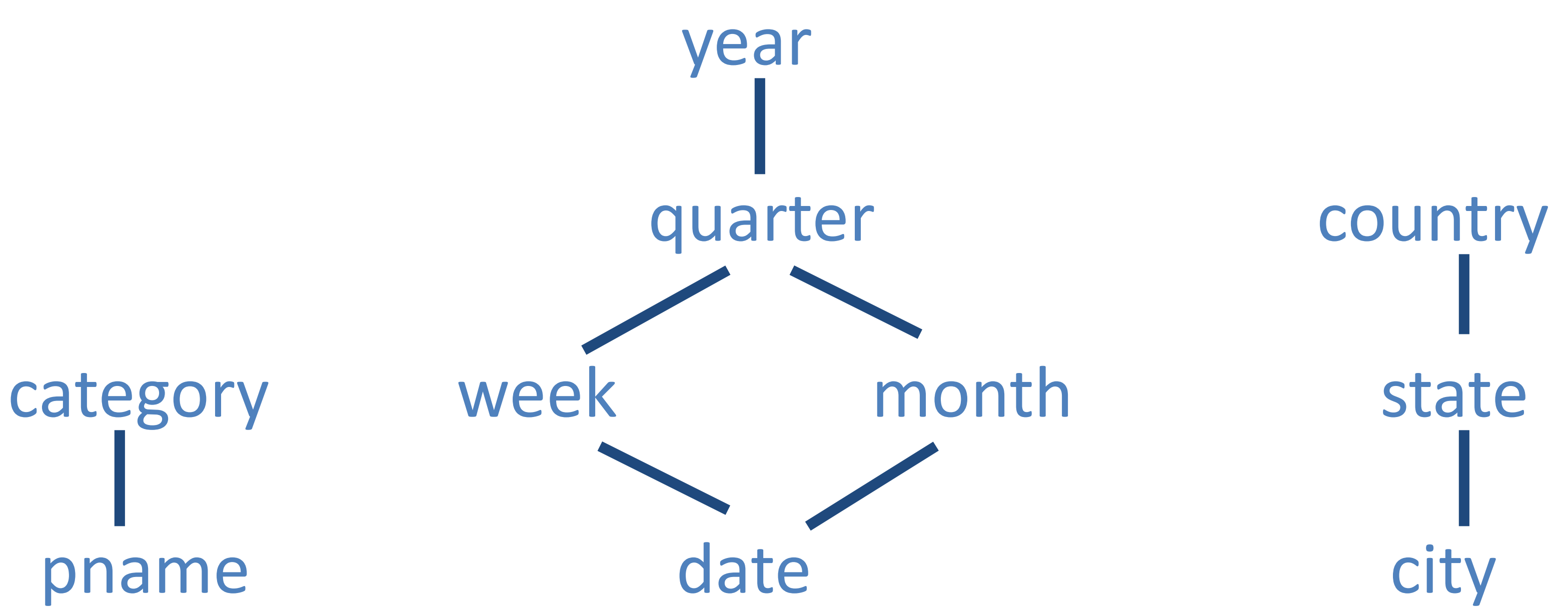
- The main relation, which relates dimensions to a measure, is called the fact table. Each dimension can have additional attributes and an associated dimension table.
  - E.g., **Products(pid, pname, category, price)**
  - Fact tables are *much* larger than dimensional tables.



OLAP: Dimension Hierarchies

- For each dimension, the set of values can be organized in a hierarchy:

PRODUCT **TIME** **LOCATION**



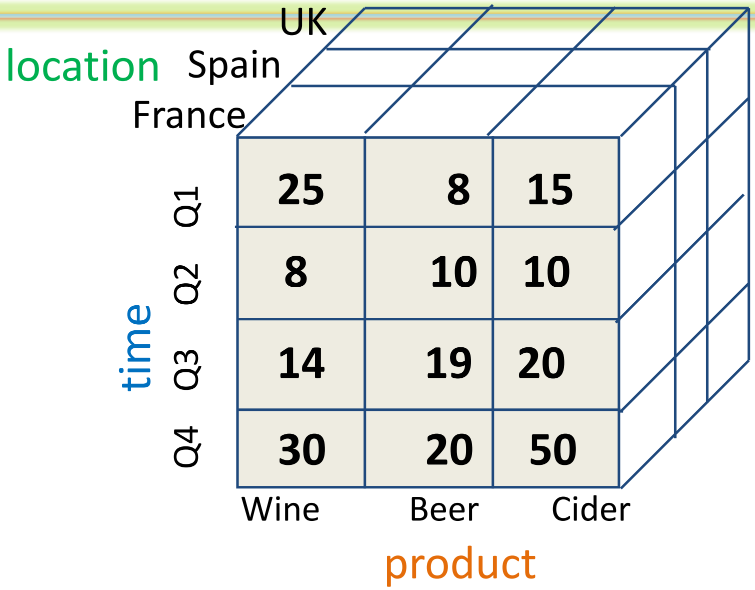
OLAP: Dimension Hierarchies

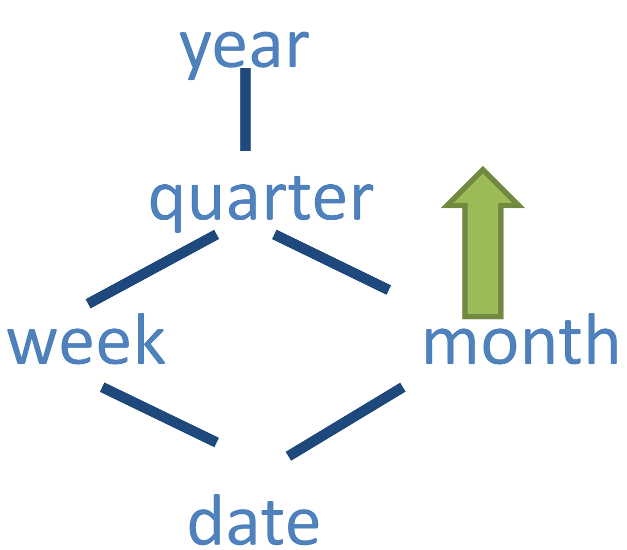
- Influenced by SQL and by spreadsheets.

- A common operation is to aggregate a measure over one or more dimensions.
  - Find total sales.
  - Find total sales for each city, or for each state.
  - Find total sales for each quarter, or for each month.
  - Find top five products ranked by total sales.

OLAP operations

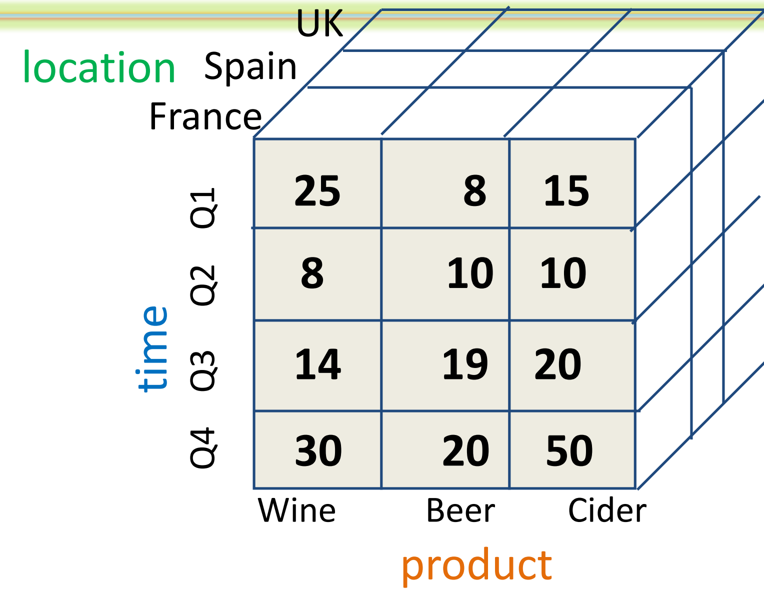
1. <u>Roll-up:</u> Aggregating at different levels of a dimension hierarchy
   - Going from lower to higher in that hierarchy
   - E.g., Given total sales by month, we can *roll-up* to get total sales by quarter (Month -> Quarter)



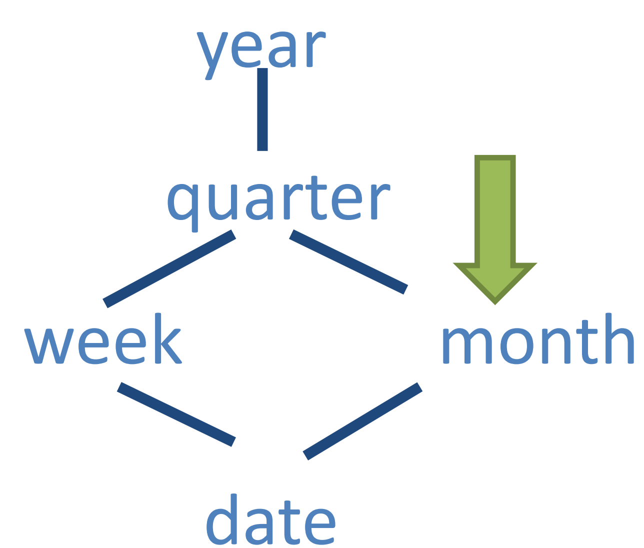


OLAP operations

1. Roll-up: Aggregating at different levels of a dimension hierarchy
   - Going from lower to higher in that hierarchy
   - E.g., Given total sales by month, we can *roll-up* to get total sales by quarter (Month -> Quarter)

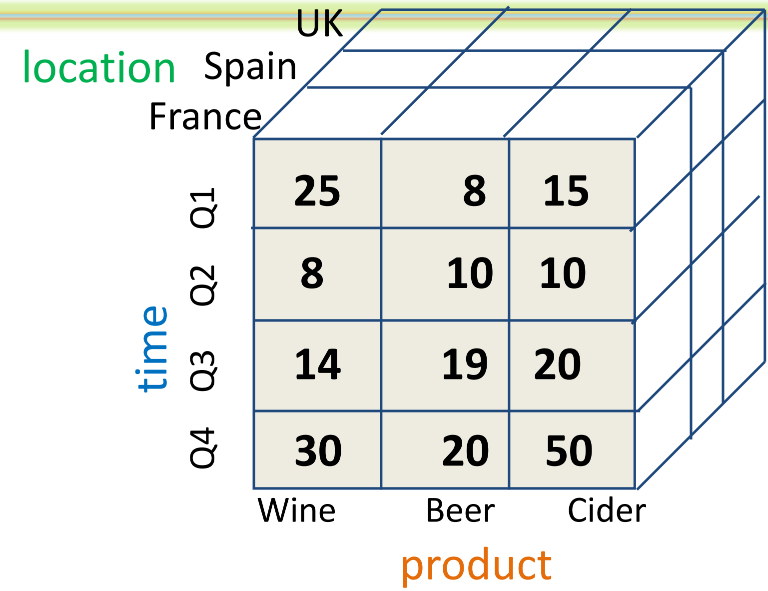


2. Drill-down: The inverse of roll-up: Going from higher to lower in the aggregation hierarchy
   - E.g., Given total sales by quarter, can *drill-down* to get total sales by Month (Quarter -> Month)

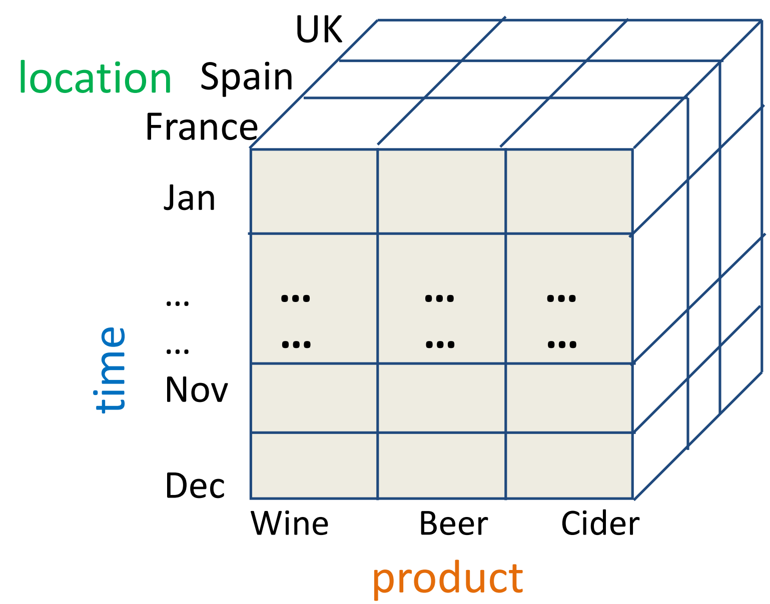


OLAP operations

1. Roll-up: Aggregating at different levels of a dimension hierarchy
   - Going from lower to higher in that hierarchy
   - E.g., Given total sales by month, we can *roll-up* to get total sales by quarter (Month -> Quarter)



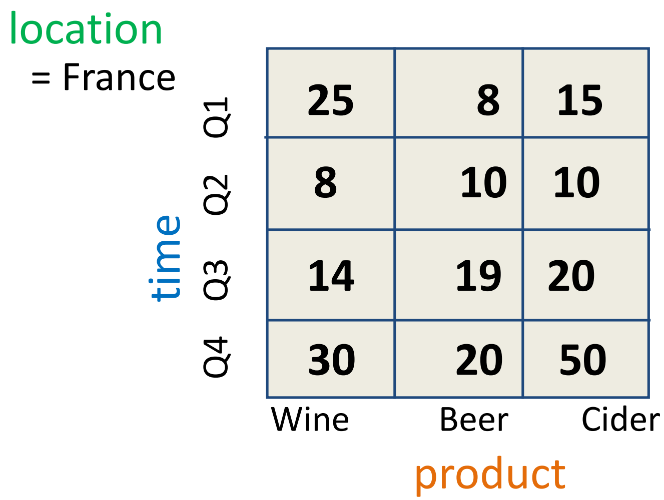
2. Drill-down: The inverse of roll-up: Going from higher to lower in the aggregation hierarchy
   - E.g., Given total sales by quarter, can *drill-down* to get total sales by Month (Quarter -> Month)



OLAP operations

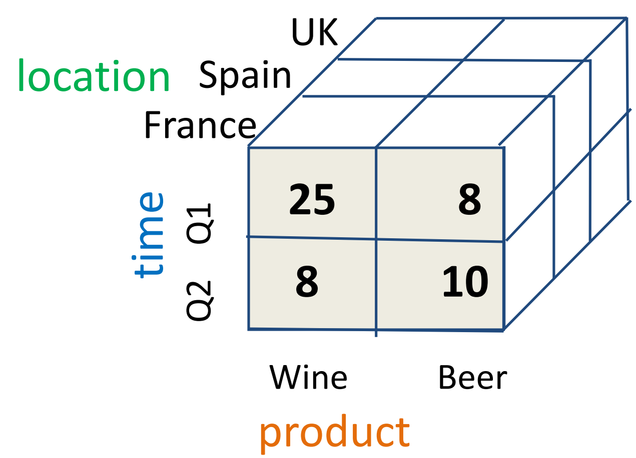
3. Slicing: Equality selections on one or more dimensions.

- Selecting a single dimension results in a new sub-cube creation
- E.g., total sales by location=France.



4. Dicing: Range selections on one or more dimensions

- Selecting a sub-cube by selecting two or more dimensions
- E.g., total sales by time in (Q1, Q2) and product in (Wine, Beer).



OLAP operations

5. Pivoting: Aggregation on selected dimensions. Then rotating the current view to get a new view.

– E.g., Pivoting on Location and Time yields **cross-tabulation**:

size: **all**

color

item-name		dark	pastel	white	Total
	skirt	8	35	10	53
	dress	20	10	5	35
	shirt	14	7	28	49
	pant	20	2	5	27
	Total	62	54	48	164

Illustration: flat data table

item-name	*color*	*number*
skirt	dark	8
skirt	pastel	35
skirt	white	10
skirt	**all**	53
dress	dark	20
dress	pastel	10
dress	white	5
dress	**all**	35
shirt	dark	14
shirt	pastel	7
shirt	white	28
shirt	**all**	49
pant	dark	20
pant	pastel	2
pant	white	5
pant	**all**	27
all	dark	62
all	pastel	54
all	white	48
all	**all**	164

Cross Tabulation of *sales* by *item-name* and *color*

size: **all**

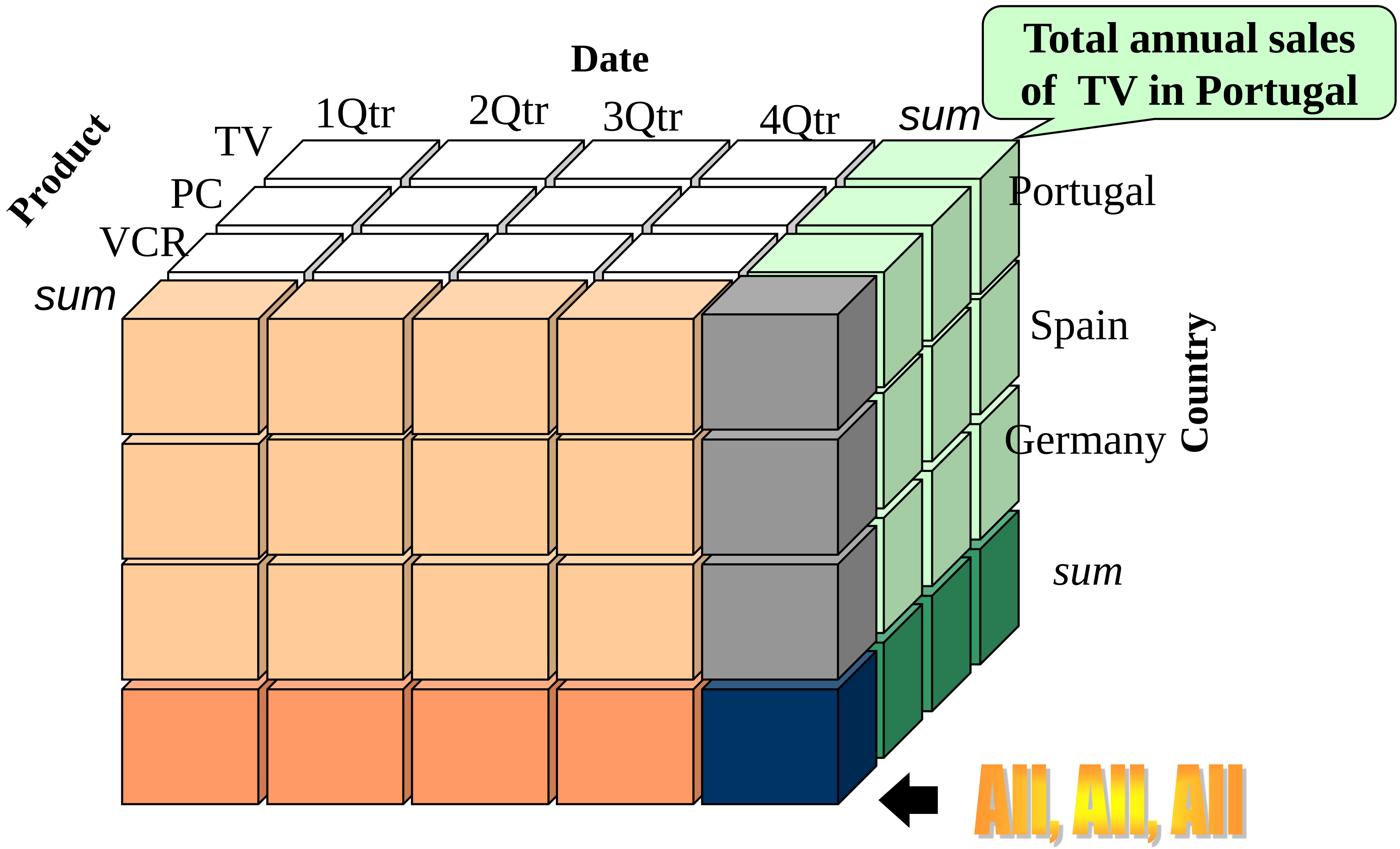
color

item-name	dark	pastel	white	Total
skirt	8	35	10	53
dress	20	10	5	35
shirt	14	7	28	49
pant	20	2	5	27
Total	62	54	48	164

- The table above is an example of a cross-tabulation (or cross-tab) also referred to as a pivot-table. *In general, a cross-table is a table where values for one attribute form the row headers, values for another attribute form the column headers*, and the values in an individual cell are derived as above.

- A cross tab with summary rows/columns can be represented by introducing a special value all to represent subtotals.

Another Sample Data Cube



Src: C. J. Date

An example: supplier-and-parts database

S#	P#	QTY
S1	P1	300
S1	P2	200
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200

S# -> supplier#
P# -> parts#
QTY -> quantity

(This example is from C. J. Date's database textbook)

Queries:

1) Get the total shipment (of parts) quantity
2) Get total shipment quantities by supplier
3) Get total shipment quantities by part
4) Get the shipment by supplier and part

Cross Tabulations

	P1	P2	Total
S1	300	200	500
S2	300	400	700
S3	0	200	200
S4	0	200	200
	600	1000	1600

Queries:

1) Get the total shipment quantity
2) Get total shipment quantities by supplier
3) Get total shipment quantities by part

Cross Tabulations benefits

	P1	P2	Total
S1	300	200	500
S2	300	400	700
S3	0	200	200
S4	0	200	200
	600	1000	1600

- Display query results as cross tabulations
  - More readable way
  - Formatted as a simple array
  - Two dimensions (supplier and parts)

Cross Tabulations and SQL queries

1. The total shipment quantity:

```
SELECT SUM(QTY) AS TOTQTY
FROM SP
GROUP BY () ;
```

TOTQTY
1600

	P1	P2	Total
S1	300	200	500
S2	300	400	700
S3	0	200	200
S4	0	200	200
	600	1000	1600

Cross Tabulations and SQL queries

2. The total shipment quantities by suppliers

```
SELECT S#, SUM(QTY)  AS TOTQTY
   FROM SP
 GROUP BY (S#) ;
```

S#	TOTQTY
S1	500
S2	700
S3	200
S4	200

	P1	P2	Total
S1	300	200	500
S2	300	400	700
S3	0	200	200
S4	0	200	200
	600	1000	1600

Cross Tabulations and SQL queries

3. The total shipment quantities by parts

```
SELECT P#,SUM(QTY) AS TOTQTY
 FROM SP
 GROUP BY (P#) ;
```

P#	TOTQTY
P1	600
P2	1000

	P1	P2	Total
S1	300	200	500
S2	300	400	700
S3	0	200	200
S4	0	200	200
	600	1000	1600

Cross Tabulations and SQL queries

4. the shipment by supplier and part

```
SELECT S#, P#, SUM(QTY) AS TOTQTY
 FROM SP
 GROUP BY (S#,P#) ,
```

S#	P#	TOTQTY
S1	P1	300
S1	P2	200
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200

	P1	P2	Total
S1	300	200	500
S2	300	400	700
S3	0	200	200
S4	0	200	200
	600	1000	1600

Considerations with SQL queries

- Formulation of so many similar but many distinct queries is tedious
- Executing the queries is expensive
- Make life easier,
  - more efficient computation
- Single query to replace many distinct queries
  - GROUPING SETS, ROLLUP, CUBE options
  - Added to SQL standard 1999

GROUPING SETS

- Execute several queries simultaneously with a single one

SELECT S#, P#, SUM (QTY) AS TOTQTY

FROM SP

GROUP BY **GROUPING SETS ((S#), (P#))** ;

Single results table

Not a relation !!

null ➔ missing information

S#	P#	TOTQTY
S1	null	500
S2	null	700
S3	null	200
S4	null	200
null	P1	600
null	P2	1000

ROLLUP

SELECT S#,P#, SUM (QTY) AS TOTQTY
FROM SP
GROUP BY **ROLLUP (S#, P#) ;**

GROUP BY GROUPING SETS ((S#, P#), (S#) , ( ))

S#	P#	TOTQTY
S1	P1	300
S1	P2	200
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S1	null	500
S2	null	700
S3	null	200
S4	null	200
null	null	1600

ROLLUP

- The quantities have been “roll up“ for each supplier
- Rolled up “along supplier dimension“

GROUP BY ROLLUP (A,B,...,Z)

(A,B,...,Z)
(A,B,...)
(A,B)
(A)
()

→ GROUP BY ROLLUP (A,B) is not symmetric in A and B !

- Notice, hierarchical rollups starts with the primary group A, followed by the others in hierarchical order. This hierarchy is implied by the order in which the fields appear in the GROUP BY clause: (A,B,...,Z)

CUBE

SELECT S#, P#, SUM (QTY) AS TOTQTY
FROM SP
GROUP **BY CUBE (S#, P#)** ;

GROUP BY GROUPING SETS ((S#, P#), (S#), (P#), ( ))

S#	P#	TOTQTY
S1	P1	300
S1	P2	200
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S1	null	500
S2	null	700
S3	null	200
S4	null	200
null	P1	600
null	P1	1000
null	null	1600

CUBE

- Confusing term CUBE (?)
  - Derived from the fact that in multidimensional terminology: data values are stored in cells of a multidimensional array or a hypercube
    - The actual physical storage **my differ**
  - In our example
    - cube has just two dimensions (supplier, part)
- Means “group“ by all possible subsets of the set {A, B, ..., Z }
  - If there are k dimensions, we have 2^k possible SQL GROUP BY queries that can be generated

The CUBE Operator

- Generalizing the previous example, if there are k dimensions, we have 2^k possible SQL GROUP BY queries that can be generated through pivoting on a subset of dimensions.

- CUBE pid, locid, timeid BY SUM Sales
  - Equivalent to rolling up Sales on all eight subsets of the set {pid, locid, timeid}; each roll-up corresponds to an SQL query of the form:

```
SELECT SUM(S.sales)
FROM   Sales S
GROUP BY grouping-list
```

Lots of work on optimizing the CUBE operator!