8. New Research and Application Fields

Main Contents

- Data warehouse
- OLAP
- Data mining
- Information retrieval
- Semi-structured data and XML

8.1 Data Warehouse & OLAP

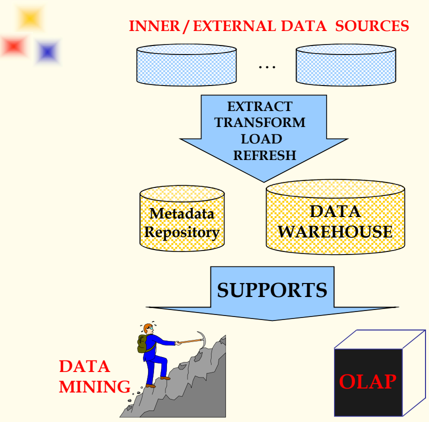
- Challenges come from huge amount of data in network era
  - How to use them efficiently?
  - How to find useful information in such a data ocean?
  - If they cannot be used by human being, they are garbage.
- Data is the most important resources.
- The importance of decision-making scientifically
- Data requirements in decision-making

Data Requirements in Decision-making

- Need summarized data while not detail data
- Need historical data
- Need large amount of external data (multi data source)
- Need decision subject oriented data, while not the data facing daily transaction process
- The data don't need real time updating. They are mainly read
- OLTP and OLAP

What is Data warehouse

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



Database and Data Warehouse

	On-Line Transaction Processing (OLTP)	On-Line Analytical Processing (OLAP)
Data Feature	Detail data	Summarized data
Data prescription	Current data	Current and historical data
Data Source	Inner data of a enterprise	Inner & External data; Distributed; Heterogeneous
Data organization	Surrounding transaction processing	Surrounding decision subject
Data Updating	Updated instantly	Periodical or on demand
Data Amount	Involving less data in one operation	Involving huge amount of data in one operation
Operating Feature	Mainly simple and repeated short transactions	Mainly long transaction processing complex queries

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.

Software Solutions that use Warehouse

- **Online Analytical Processing (OLAP)**
  - enables users to analyses data across multiple dimensions and hierarchies
- **Analysis and Query Reporting Solutions**
  - custom built analysis tools use mathematical models to produce specialized interactive solutions
- **Data Mining**
  - enable users to identify patterns and correlations within a set of data, or to create predictive models from the data

An Example

Color	Size	Number
Light	small	3
Light	Medium	20
Light	Large	10
Dark	Small	4
Light	Small	5
Dark	Small	16
Light	Medium	5
Dark	Large	5
Light	Medium	10
Dark	Medium	5
Dark	Medium	5

	Small	Medium	Large	Total
Light	8	35	10	53
Dark	20	10	5	35
Total	28	45	15	88

Cross-tabulation of *number* by *size* and *color* of sample relation *sales* with the schema Sales(color, size, number).

An Example (Cont.)

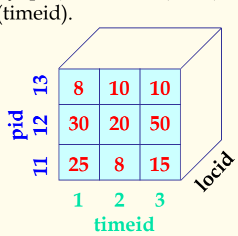
Color	Size	Number
Light	Small	8
Light	Medium	35
Light	Large	10
Light	all	53
Dark	Small	20
Dark	Medium	10
Dark	Large	5
Dark	all	35
all	Small	28
all	Medium	45
all	Large	15
all	all	88

- Can represent subtotals in relational form by using the value all
- E.g. : obtain (Light, all, 53) and (Dark, all, 35) by aggregating individual tuples with different values for size for each color.

Multidimensional Data Model

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

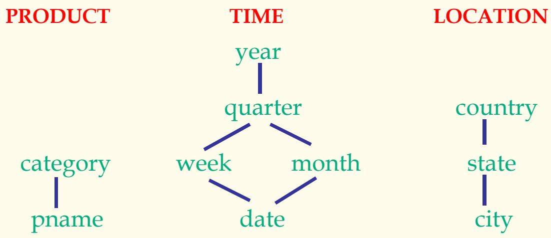
Slice locid=1 is shown:



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

Dimension Hierarchies

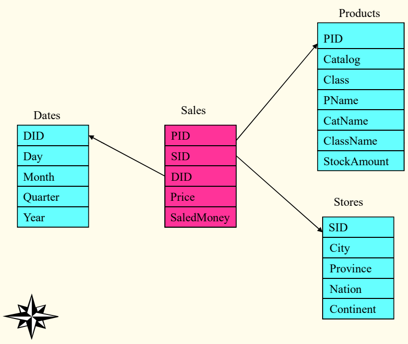
- For each dimension, the set of values can be organized in a hierarchy:



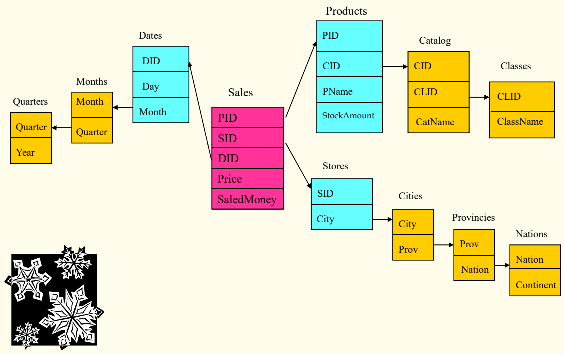
MOLAP vs ROLAP

- Multidimensional data can be stored physically in an (disk-resident, persistent) array; called MOLAP systems. Alternatively, can store as a relation; called ROLAP systems.
- 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.

Star Schema



Snowflake Schema



Materialized View

- Star or snowflake schema are common storing schema in data warehouse. But decision-making are generally not based on star or snowflake schema directly. They are based on different kinds of summarized data computed from star or snowflake schema. Because the computation of aggregation function is very time-consuming, the computing results are often stored as *materialized view* in data warehouse.
- Take P (Products), S (Stores), D (Dates) as example. Their star schema as above.

PSD View

- Take the computation of SUM function as example. Other aggregation functions are similar.
- *Sales of every product in every store at every day.*

```
CREATE VIEW PSD (PID, SID, DID, TotalSales) AS
SELECT PID, SID, DID, SUM (SaledMoney) AS TotalSales
FROM Sales
GROUP BY PID, SID, DID;
```

PS, SD, and PD View

- *Total sales of every product in every store (for all times)*
- SD and PD views are similar (for all products or for all stores)

```
CREATE VIEW PS (PID, SID, TotalSales)  AS
SELECT PID, SID, SUM(TotalSales) AS TotalSales
FROM PSD
GROUP BY PID, SID;
```

- PS View can be expressed as PS ALL

P, S, and D View

- *Total sales of every product (for all stores and all times)*
- S and D views are similar (aggregated according to store or date respectively)

```
CREATE VIEW P (PID, TotalSales)  AS
SELECT PID, SUM(TotalSales) AS TotalSales
FROM PS            /* or PD */
GROUP BY PID;
```

- P View can be expressed as P ALL ALL

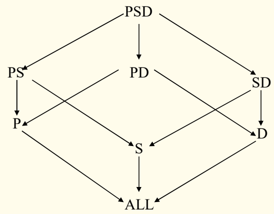
ALL View

- *Total sales for all products and all stores and all times*

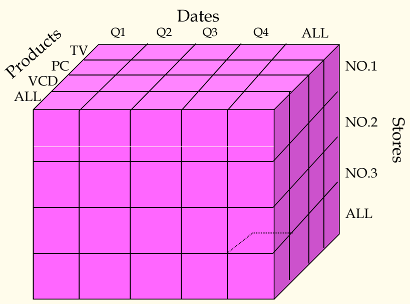
```
CREATE VIEW ALL (TotalSales)  AS
SELECT SUM(TotalSales) AS TotalSales
FROM P            /* or S or D */
```


Reliant Relationships Between Views

- There are three dimensions in Sales: {P, S, D}, every sub-set of it is corresponding to a view. These are equivalent to 2^3 elements of power-set ρ({P, S, D}).
- The following is the reliant relationships between these materialized views:



A Data Cube Example



Products (P)
Stores (S)
Dates (D)

Measure: SaledMoney

OLAP Queries

- 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 top five products ranked by total sales.
- Roll-up: Aggregating at different levels of a dimension hierarchy.
  - E.g., Given total sales by city, we can roll-up to get sales by state.

OLAP Queries

- Drill-down: The inverse of roll-up.
  - E.g., Given total sales by state, can drill-down to get total sales by city.
  - E.g., Can also drill-down on different dimension to get total sales by product for each state.
- Pivoting: Aggregation on selected dimensions.
- Slicing and Dicing: Equality and range selections on one or more dimensions.

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, SID, DID 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:

Lots of work on optimizing the CUBE operator!

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


Examples of Queries on Cube

```
SELECT PID, SID, Quarter, SUM (SaledMoney) AS TotalSales
FROM Sales S, Dates D
WHERE S.DID=D.DID
CUBE BY  PID, SID, Quarter;
```
--- *Generate a data cube including 2^3 views.*

- CUBE(TV,No1,Q1)
  - *Query TV's total sales of store No1 in first quarter.*
- CUBE(ALL,No1,Q1)
  - *Query total sales of store No1 in first quarter.*
- CUBE(ALL,ALL,Q1)
  - *Query total sales in first quarter.*
- CUBE(ALL,ALL,ALL)
  - *Query total sales in whole year.*
- CUBE(TV,No1,Q1) + CUBE(VCD,No1,Q1)
- CUBE(ALL,ALL,Q1) / CUBE(ALL,ALL,ALL) * 100%

Examples of Roll-up / Drill-down

```
SELECT PID, SID, DID, SUM (SaledMoney) AS TotalSales
FROM Sales
GROUP BY  PID, SID, DID ROLLUP;
```
--- *Generate four views : PSD, PS, P, and ALL*

```
SELECT PID, SID, DID, SUM (SaledMoney) AS TotalSales
FROM Sales S, Dates D
WHERE S.DID=D.DID AND Year BETWEEN 1997 AND 1998
CUBE BY  PID, SID, DID
ROLLUP Day, Month, Quarter, Year;
```

- CUBE(TV, No1, 1998)
- CUBE(TV, No1, 1998.Q4)
- CUBE(TV, No1, 1998.12)

Design Issues

TIMES

timeid	date	week	month	quarter	year	holiday_flag

SALES (Fact table)

pid	timeid	locid	sales

PRODUCTS

pid	pname	category	price

LOCATIONS

locid	city	state	country

- Fact table in BCNF; dimension tables un-normalized.
  - Dimension tables are small; updates/inserts/deletes are rare. So, anomalies less important than query performance.
- This kind of schema is very common in OLAP applications, and is called a star schema; computing the join of all these relations is called a star join.

Implementation Issues

- New indexing techniques: Bitmap indexes, Join indexes, array representations, compression, precomputation of aggregations, etc.
- E.g., Bitmap index:

Bit-vector:
1 bit for each possible value.
Many queries can be answered using bit-vector ops!

F M

sex	custid	name	sex	rating	rating
10	112	Joe	M	3	00100
10	115	Ram	M	5	00001
01	119	Sue	F	5	00001
10	112	Woo	M	4	00010

Bitmap index – index itself is data

Date	Store	State	Class	Sales
3/1/96	32	NY	A	6
3/1/96	36	MA	A	9
3/1/96	38	NY	B	5
3/1/96	41	CT	A	11
3/1/96	43	NY	A	9
3/1/96	46	RI	B	3
3/1/96	47	CT	B	7
3/1/96	49	NY	A	12

Bitmap Index for Sales

8bit	4bit	2bit	1bit
0	1	1	0
1	0	0	1
0	1	0	1
1	0	1	1
1	0	0	1
0	0	1	1
0	1	1	1
1	1	0	0

Bitmap Index for State

AK	AR	CA	CO	CT	MA	NY	RI	… …
0	0	0	0	0	0	1	0	
0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	
0	0	0	0	1	0	0	0	
0	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	1	
0	0	0	0	1	0	0	0	
0	0	0	0	0	0	1	0	

for Class

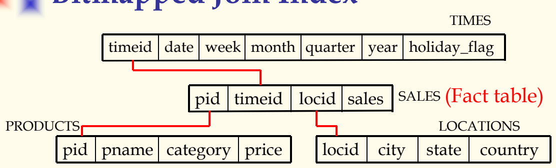
A	B	C
1	0	0
1	0	0
0	1	0
1	0	0
1	0	0
0	1	0
0	1	0
1	0	0

- Total sales = ? (4*8+4*4+4*2+6*1=62)
- How many class A store in NY ? (3)
- Sales of class A store in NY = ? (2*8+2*4+1*2+1*1=27)
- How many stores in CT ? (2)
- Join operation (query product list of class A store in NY)

Join Indexes

- Consider the join of Sales, Products, Times, and Locations, possibly with additional selection conditions (e.g., country="USA").
  - A join index can be constructed to speed up such joins. The index contains [s,p,t,l] if there are tuples (with sid) s in Sales, p in Products, t in Times and l in Locations that satisfy the join (and selection) conditions.
- Problem: Number of join indexes can grow rapidly.
  - A variation addresses this problem: For each column with an additional selection (e.g., country), build an index with [c,s] in it if a dimension table tuple with value c in the selection column joins with a Sales tuple with sid s; if indexes are bitmaps, called bitmapped join index.

Bitmapped Join Index

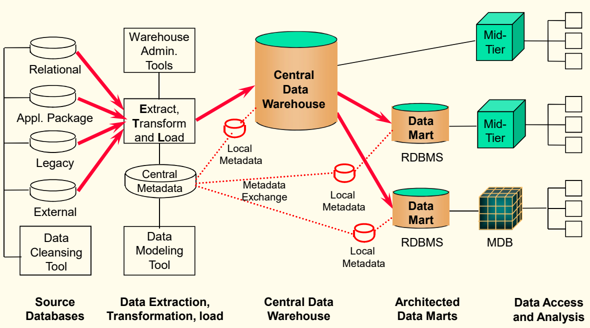


- Consider a query with conditions price=10 and country="USA". Suppose tuple (with sid) s in Sales joins with a tuple p with price=10 and a tuple l with country ="USA". There are two join indexes; one containing [10,s] and the other [USA,s].
- Intersecting these indexes tells us which tuples in Sales are in the join and satisfy the given selection.

The Procedure of Data Warehouse Engineering

- Requirements Analysis (Project Plan)
- Data Warehouse Architecture Design
- Environments Construction
- Data Warehouse Schema Design
- ETL Processing of Data
- Meta Data Management
- Front Applications Design & Implementation
- Testing
- Running & Maintaining

Data Warehouse Architecture Design

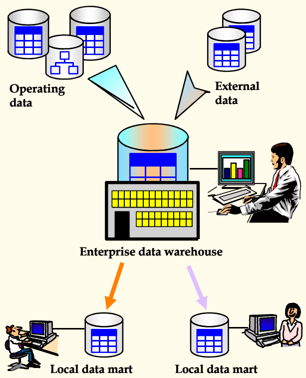


Data Warehouse Schema Design

- Top Down
- Bottom Up
- Mixed Method

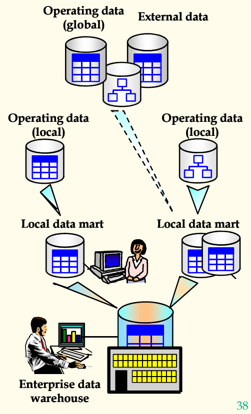
Top Down

- Construct enterprise data warehouse (EDW)
  - ✓ Common central data schema
  - ✓ One-off data reconstruction
  - ✓ Minimized data redundancy and inconsistency
  - ✓ Detail and historical data; global data exploration
- Construct data marts from EDW
  - ✓ Sub-set in EDW which related with a department
  - ✓ Summarized data
  - ✓ Rely on the availability of data in EDW



Bottom Up

- Construct data mart of department
  - ✓ Limited in a subject area
  - ✓ Local business requirements are fulfilled rapidly
  - ✓ Department autonomy
  - ✓ A good guidance to the construction of data marts of other departments
  - ✓ Need to do data reconstruction for every department
  - ✓ Have redundancy and inconsistency at a certain extent
  - ✓ A feasible way
- Enlarged to enterprise data warehouse (EDW)
  - ✓ Take the construction of EDW as a long-term target



Summary

- Decision support is an emerging, rapidly growing subarea of databases.
- Involves the creation of large, consolidated data repositories called data warehouses.
- Warehouses exploited using sophisticated analysis techniques: complex SQL queries and OLAP "multidimensional" queries (influenced by both SQL and spreadsheets).
- New techniques for database design, indexing, view maintenance, and interactive querying need to be supported.