OLAP and Data Warehousing

Advanced Topics in Database Management (INFSCI 2711)
Distributed Databases (TELCOM 2326)

Some materials are from INFSCI2710' a Database Management Systems, R. Ramakrishnan and J. Gehrke

Vladimir Zadorozhny, GIST, University of Pittsburgh

Introduction

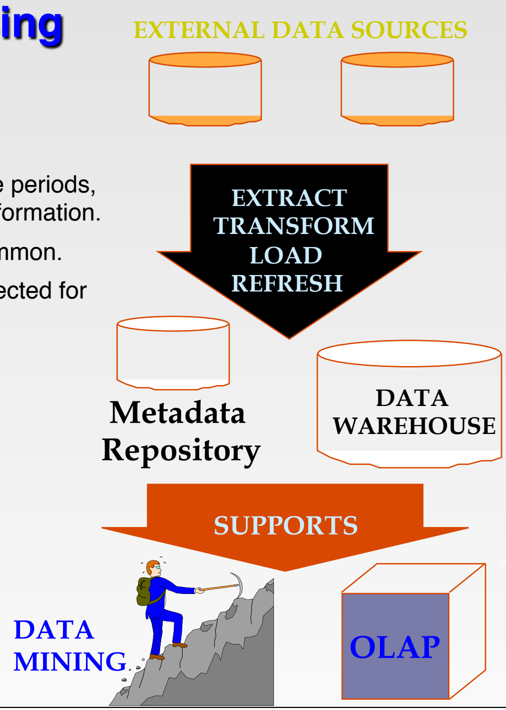
- Increasingly, organizations are analyzing current and historical data to identify useful patterns and support business strategies.
- Emphasis is on complex, interactive, exploratory analysis of very large datasets created by integrating data from across all parts of an enterprise; data is fairly static.
  - Contrast such **On-Line Analytic Processing (OLAP)** with traditional **On-line Transaction Processing (OLTP)**: mostly long queries, instead of short update Xacts.

Three Complementary Trends

- Data Warehousing: Consolidate data from many sources in one large repository.
  - Loading, periodic synchronization of replicas.
  - Semantic integration.
- OLAP:
  - Complex SQL queries and views.
  - Queries based on spreadsheet-style operations and “multidimensional” view of data.
  - Interactive and “online” queries.
- Data Mining: Exploratory search for interesting trends and anomalies (not considered in this class)

Data Warehousing

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



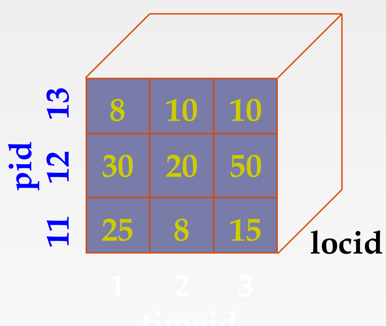
Warehousing Issues

- 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.

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

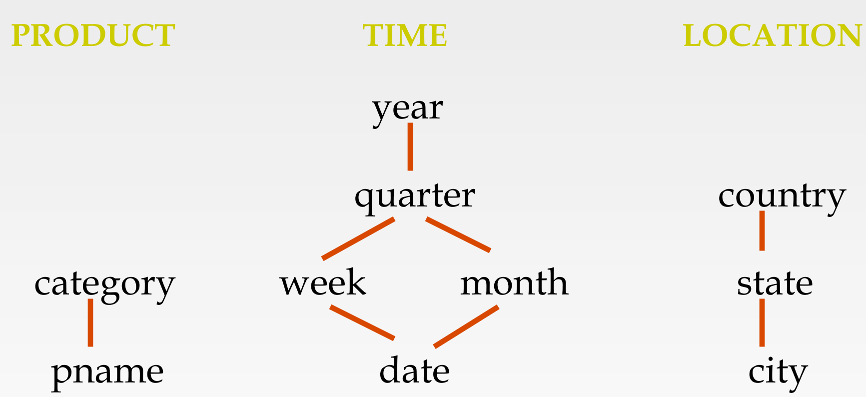
...

MOLAP vs ROLAP

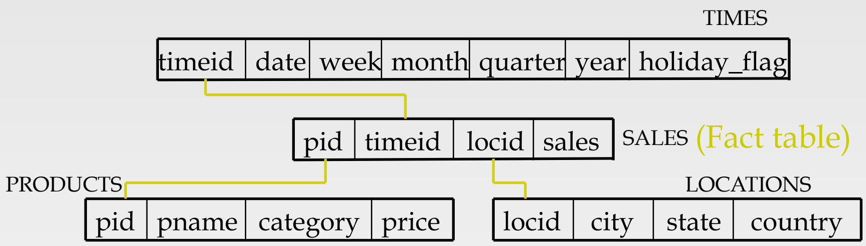
- Multidimensional data can be stored physically in a (disk-resident, persistent) array; called MOLAP systems. Alternatively, can store as a relation; called ROLAP systems.
- The main relation, which relates dimensions to a measure, is called the fact table. Each dimension can have additional attributes and an associated dimension table.
  - E.g., **Products(pid, pname, category, price)**
  - Fact tables are *much* larger than dimensional tables.

Dimension Hierarchies

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



Star Schema Design



- Fact table is large, updates are frequent; dimension tables are small, updates are rare.
- 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.

OLAP Queries

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

	WI	CA	Total
1995	63	81	144
1996	38	107	145
1997	75	35	110
Total	176	223	339

- Slicing and Dicing: Equality and range selections on one or more dimensions.

Using SQL for Pivoting

- The cross-tabulation obtained by pivoting can also be computed using a collection of SQLqueries:

```
SELECT SUM(S.sales)
FROM   Sales S, Times T, Locations L
WHERE  S.timeid=T.timeid AND S.timeid=L.timeid
GROUP BY T.year, L.state
```

```
SELECT SUM(S.sales)
FROM   Sales S, Times T
WHERE  S.timeid=T.timeid
GROUP BY T.year
```

```
SELECT SUM(S.sales)
FROM   Sales S, Location L
WHERE  S.timeid=L.timeid
GROUP BY L.state
```

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:

Lots of work on optimizing the CUBE operator!

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

Views and Decision Support

- OLAP queries are typically aggregate queries.
  - Precomputation is essential for interactive response times.
  - The CUBE is in fact a collection of aggregate queries, and precomputation is especially important: lots of work on what is best to precompute given a limited amount of space to store precomputed results.
- Warehouses can be thought of as a collection of asynchronously replicated tables and periodically maintained views.
  - Has renewed interest in view maintenance!

View Modification (Evaluate On Demand)

View

```
CREATE VIEW RegionalSales(category,sales,state)
        AS SELECT P.category, S.sales, L.state
           FROM Products P, Sales S, Locations L
          WHERE P.pid=S.pid AND S.locid=L.locid
```

Query

```
SELECT R.category, R.state, SUM(R.sales)
FROM RegionalSales AS R GROUP BY R.category, R.state
```

Modified Query

```
SELECT R.category, R.state, SUM(R.sales)
FROM (SELECT P.category, S.sales, L.state
        FROM Products P, Sales S, Locations L
        WHERE P.pid=S.pid AND S.locid=L.locid) AS R
GROUP BY R.category, R.state
```

View Materialization (Precomputation)

- Suppose we precompute RegionalSales and store it.
- Then, previous query can be answered more efficiently (modified query will not be generated).

Issues in View Materialization

- What views should we materialize, and what indexes should we build on the precomputed results?
- Given a query and a set of materialized views, can we use the materialized views to answer the query?
- How frequently should we refresh materialized views to make them consistent with the underlying tables? (And how can we do this incrementally?)

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.
- Commonly requires integrating DISTRIBUTED HETEROGENEOUS DATA