Data Warehousing and Decision Support

ENEE 752
Spring 2005
On-Line Analytic Processing (OLAP)

- Consolidate data from many databases maintained by different business units – issues in loading, synchronization, and semantic integration.
- Interactive exploratory analysis of consolidated data using aggregation operators, complex Boolean conditions, and statistical functions.
- Used for decision support and identifying trends.
Data Consolidation

• Integrate data spanning long time periods, often augmented with summary information.

• Issues:
  – Semantic integration of data from multiple sources. How do we handle heterogeneous sources?
  – Load, refresh, and purge policies
  – Metadata management – must keep track of source, loading time, …
Multidimensional Data Model

• A set of numeric measures, each depending on a set of dimensions.

• Example: Sales measure that depends on Product (key: pid), Location (key: locid), and Time (key: timeid) resulting in a 3-D cube.

• Can be stored as a multi-dimensional array mapped onto a disk – MOLAP.
Relational OLAP (ROLAP)

- Store main relation relating dimensions to a measure as a **fact table**.
- Each dimension can have its associated dimension table.
- Example:
  - Locations (locid, city, state, country)
  - Products (pid, pname, category, price)
  - Time (timeid, date, week, year, holiday_flag)
- **Star Scheme** – dimension tables are much smaller than fact tables.
Dimension Hierarchies

• For each dimension, it may be desirable to organize the set of values into a hierarchy.

• Examples:
  – Time: date -> week -> month -> quarter -> year
  – Location: city -> state -> country
  – Product: pname -> category

• Roll-up queries: aggregation at different levels of a dimension hierarchy. Drill-down is the inverse.
OLAP Queries

• Most common: aggregate a measure over one or more dimensions:
  – Determine total sales for each city
  – Determine annual total sales
  – Determine top five products ranked by total sales

• Pivoting: aggregation over selected dimensions (cross-tabulation)

• Slicing and Dicing: equality and range selection on one or more dimensions.
SQL Queries

• Pivoting on Location and Time:
  
  • SELECT T.year, L.state, SUM(S.sales)
    FROM Sales S, Times T, Locations L
    WHERE S.timeid=T.timeid AND S.locid=L.locid
    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, Locations L
    WHERE S.locid=L.locid
    GROUP BY L.state
The Cube Operator

- We have $2^k$ possible Group By queries that can be generated through pivoting over $k$ dimensions.
- CUBE operator over a list of dimensions computes specified operation over all subsets of these dimensions.

Example:
- SELECT T.year, L.state, SUM(S.sales)
- FROM Sales S, Times T, Locations L
- WHERE S.timeid=T.timeida AND S.locid=L.locid
- GROUP BY CUBE (T.year, L.state)
Views

• Precomputing views is critical for interactive response times.
• Issue: Which views should be materialized?
• Standard strategy:
  – 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 R
    GROUP BY R.category, R.state
  – RegionalSales is computed for each query.
View Materialization

• Suppose we precompute RegionSales, then we can handle the previous query much more efficiently.

• A view whose tuples are stored in the database is said to be *materialized*.
  – Provides fast access similar to caches
  – Need to update the view as the underlying tables change (incremental view maintenance?)

• Close relationship to maintaining distributed databases, checking integrity constraints, and evaluating rules and triggers.
Issues in View Materialization

• Which views should we materialize? And how should they be indexed?
• Given a query and a set of materialized views, how should we use the views to answer the query?
• How frequently should we refresh the materialized views? Can we do it incrementally?
View Maintenance

• Two Steps:
  – Propagate: compute changes when data changes
  – Refresh: apply changes to the materialized view

• Maintenance Policy – When to Refresh
  – Immediate: As part of the transaction that modifies the underlying tables
  – Deferred: Some time later, under a separate transaction:
    • Lazy: delay until next query on view; then refresh
    • Periodic: refresh periodically – queries possibly answered using outdated views (widely used for replication in distributed databases)
    • Event-Based: refresh when a specified event occurs.