On-Line Application Processing

Warehousing
Data Cubes
Data Mining
Overview

- Traditional database systems are tuned to many, small, simple queries.
- Some new applications use fewer, more time-consuming, *analytic* queries.
- New architectures have been developed to handle analytic queries efficiently.
The Data Warehouse

- The most common form of data integration.
  - Copy sources into a single DB (warehouse) and try to keep it up-to-date.
  - Usual method: periodic reconstruction of the warehouse, perhaps overnight.
  - Frequently essential for analytic queries.
OLTP

Most database operations involve On-Line Transaction Processing (OTLP).

- Short, simple, frequent queries and/or modifications, each involving a small number of tuples.

Examples: Answering queries from a Web interface, sales at cash registers, selling airline tickets.
OLAP

◆ *On-Line Application Processing* (OLAP, or “analytic”) queries are, typically:
  - Few, but complex queries --- may run for hours.
  - Queries do not depend on having an absolutely up-to-date database.
OLAP Examples

1. Amazon analyzes purchases by its customers to come up with an individual screen with products of likely interest to the customer.

2. Analysts at Wal-Mart look for items with increasing sales in some region.
   - Use empty trucks to move merchandise between stores.
Common Architecture

- Databases at store branches handle OLTP.
- Local store databases copied to a central warehouse overnight.
- Analysts use the warehouse for OLAP.
A **star schema** is a common organization for data at a warehouse. It consists of:

1. **Fact table**: a very large accumulation of facts such as sales.
   - Often “insert-only.”
2. **Dimension tables**: smaller, generally static information about the entities involved in the facts.
Example: Star Schema

- Suppose we want to record in a warehouse information about every beer sale: the bar, the brand of beer, the drinker who bought the beer, the day, the time, and the price charged.

- The fact table is a relation:
  \[ \text{Sales}(\text{bar, beer, drinker, day, time, price}) \]
Example -- Continued

The dimension tables include information about the bar, beer, and drinker “dimensions”:

Bars(bar, addr, license)
Beers(beer, manf)
Drinkers(drinker, addr, phone)
Visualization – Star Schema

Dimension Table (Bars)

Dimension Table (Drinkers)

Dimension Attrs.

Dependent Attrs.

Fact Table - Sales

Dimension Table (Beers)

Dimension Table (etc.)
Dimensions and Dependent Attributes

Two classes of fact-table attributes:

1. *Dimension attributes*: the key of a dimension table.

2. *Dependent attributes*: a value determined by the dimension attributes of the tuple.
Example: Dependent Attribute

◆ price is the dependent attribute of our example Sales relation.
◆ It is determined by the combination of dimension attributes: bar, beer, drinker, and the time (combination of day and time-of-day attributes).
Approaches to Building Warehouses

1. **ROLAP** = “relational OLAP”: Tune a relational DBMS to support star schemas.

2. **MOLAP** = “multidimensional OLAP”: Use a specialized DBMS with a model such as the “data cube.”
MOLAP and Data Cubes

◆ Keys of dimension tables are the dimensions of a hypercube.
  ✤ **Example:** for the Sales data, the four dimensions are *bar, beer, drinker, and time*.

◆ Dependent attributes (e.g., *price*) appear at the points of the cube.
Visualization -- Data Cubes

- beer
- bar
- drinker
- price
Marginals

- The data cube also includes aggregation (typically SUM) along the margins of the cube.
- The *marginals* include aggregations over one dimension, two dimensions,...
Visualization --- Data Cube w/Aggregation

SUM over all Drinkers

beer
bar
drinker

price
Example: Marginals

◆ Our 4-dimensional *Sales* cube includes the sum of *price* over each bar, each beer, each drinker, and each time unit (perhaps days).

◆ It would also have the sum of *price* over all bar-beer pairs, all bar-drinker-day triples,...
Structure of the Cube

Think of each dimension as having an additional value *.

A point with one or more *’s in its coordinates aggregates over the dimensions with the *’s.

Example: Sales("Joe’s Bar", "Bud", *, *) holds the sum, over all drinkers and all time, of the Bud consumed at Joe’s.
Drill-Down

◆ *Drill-down* = “de-aggregate” = break an aggregate into its constituents.

◆ **Example**: having determined that Joe’s Bar sells very few Anheuser-Busch beers, break down his sales by particular A.-B. beer.
Roll-Up

◆ **Roll-up** = aggregate along one or more dimensions.

◆ **Example**: given a table of how much Bud each drinker consumes at each bar, roll it up into a table giving total amount of Bud consumed by each drinker.
Example: Roll Up and Drill Down

### $ of Anheuser-Busch by drinker/bar

<table>
<thead>
<tr>
<th></th>
<th>Jim</th>
<th>Bob</th>
<th>Mary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe’s Bar</td>
<td>45</td>
<td>33</td>
<td>30</td>
</tr>
<tr>
<td>Nut-House</td>
<td>50</td>
<td>36</td>
<td>42</td>
</tr>
<tr>
<td>Blue Chalk</td>
<td>38</td>
<td>31</td>
<td>40</td>
</tr>
</tbody>
</table>

### $ of A-B / drinker

<table>
<thead>
<tr>
<th></th>
<th>Jim</th>
<th>Bob</th>
<th>Mary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>133</td>
<td>100</td>
<td>112</td>
</tr>
</tbody>
</table>

#### Roll up by Bar

#### Drill down by Beer

### $ of A-B Beers / drinker

<table>
<thead>
<tr>
<th></th>
<th>Jim</th>
<th>Bob</th>
<th>Mary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bud</td>
<td>40</td>
<td>29</td>
<td>40</td>
</tr>
<tr>
<td>M’lob</td>
<td>45</td>
<td>31</td>
<td>37</td>
</tr>
<tr>
<td>Bud Light</td>
<td>48</td>
<td>40</td>
<td>35</td>
</tr>
</tbody>
</table>
Data Mining

- **Data mining** is a popular term for queries that summarize big data sets in useful ways.

- **Examples:**
  1. Clustering all Web pages by topic.
  2. Finding characteristics of fraudulent credit-card use.
Course Plug

❄ Winter 2007-8: Anand Rajaraman and Jeff Ullman are offering CS345A *Data Mining*.
  - MW 4:15-5:30, Herrin, T185.
Market-Basket Data

◆ An important form of mining from relational data involves market baskets = sets of “items” that are purchased together as a customer leaves a store.

◆ Summary of basket data is frequent itemsets = sets of items that often appear together in baskets.
Example: Market Baskets

-if people often buy hamburger and ketchup together, the store can:

1. Put hamburger and ketchup near each other and put potato chips between.
2. Run a sale on hamburger and raise the price of ketchup.
Finding Frequent Pairs

- The simplest case is when we only want to find “frequent pairs” of items.
- Assume data is in a relation \text{Baskets(basket, item)}.
- The support threshold $s$ is the minimum number of baskets in which a pair appears before we are interested.
Frequent Pairs in SQL

SELECT b1.item, b2.item
FROM Baskets b1, Baskets b2
WHERE b1.basket = b2.basket
AND b1.item < b2.item
GROUP BY b1.item, b2.item
HAVING COUNT(*) >= s;

Look for two Basket tuples with the same basket and different items. First item must precede second, so we don’t count the same pair twice.

Create a group for each pair of items that appears in at least $s$ times.

Throw away pairs of items that do not appear at least $s$ times.
A-Priori Trick – (1)

- Straightforward implementation involves a join of a huge Baskets relation with itself.
- The *a-priori algorithm* speeds the query by recognizing that a pair of items \{i, j\} cannot have support \(s\) unless both \{i\} and \{j\} do.
A-Priori Trick – (2)

Use a materialized view to hold only information about frequent items.

```
INSERT INTO Baskets1(basket, item)
SELECT * FROM Baskets
WHERE item IN (SELECT item FROM Baskets GROUP BY item HAVING COUNT(*) >= s);
```

Items that appear in at least $s$ baskets.
A-Priori Algorithm

1. Materialize the view Baskets1.
2. Run the obvious query, but on Baskets1 instead of Baskets.

- Computing Baskets1 is cheap, since it doesn’t involve a join.
- Baskets1 probably has many fewer tuples than Baskets.
  - Running time shrinks with the square of the number of tuples involved in the join.
Example: A-Priori

◆ Suppose:
  1. A supermarket sells 10,000 items.
  2. The average basket has 10 items.
  3. The support threshold is 1% of the baskets.
◆ At most 1/10 of the items can be frequent.
◆ *Probably*, the minority of items in one basket are frequent -> factor 4 speedup.