Data Warehousing & Mining

CPS 196.3
Introduction to Database Systems

Data integration

- Data resides in many distributed, heterogeneous OLTP (On-Line Transaction Processing) sources
  - Sales, inventory, customer, …
  - NC branch, NY branch, CA branch, …
- Need to support OLAP (On-Line Analytical Processing) over an integrated view of the data
- Possible approaches to integration
  - Eager: integrate in advance and store the integrated data at a central repository called the data warehouse
  - Lazy: integrate on demand; process queries over distributed sources

OLTP versus OLAP

<table>
<thead>
<tr>
<th>OLTP</th>
<th>OLAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly updates</td>
<td>Mostly reads</td>
</tr>
<tr>
<td>Short, simple transactions</td>
<td>Long, complex queries</td>
</tr>
<tr>
<td>Clerical users</td>
<td>Analysts, decision makers</td>
</tr>
<tr>
<td>Goal: ACID, transaction</td>
<td>Goal: fast queries</td>
</tr>
<tr>
<td>throughput</td>
<td></td>
</tr>
</tbody>
</table>
Eager versus lazy integration

Eager (warehousing)
- In advance: before queries
- Copy data from sources

Lazy
- On demand: at query time
- Leave data at sources

Maintaining a data warehouse

- The "ETL" process
  - Extraction: extract relevant data and/or changes from sources
  - Transformation: transform data to match the warehouse schema
  - Loading: integrate data/changes into the warehouse

- Approaches
  - Recomputation
    - Easy to implement; just take periodic dumps of the sources, say, every night
    - What if there is no "night," e.g., a global organization?
    - What if recomputation takes more than a day?
  - Incremental maintenance
    - Compute and apply only incremental changes; fast if changes are small
    - Not easy to do for complicated transformations
    - Need to detect incremental changes at the sources

“Star” schema of a data warehouse

Product

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>beer</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>wine</td>
<td>1</td>
</tr>
</tbody>
</table>

Sales

<table>
<thead>
<tr>
<th>Date</th>
<th>Prod ID</th>
<th>Cust ID</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/23/2001</td>
<td>1</td>
<td>1006</td>
<td>1</td>
</tr>
<tr>
<td>01/24/2001</td>
<td>2</td>
<td>1007</td>
<td>2</td>
</tr>
</tbody>
</table>

Customer

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>100 Main St.</td>
<td>Durham</td>
</tr>
<tr>
<td>Ben</td>
<td>102 Main St.</td>
<td>Durham</td>
</tr>
<tr>
<td>Coy</td>
<td>800 Eighth St.</td>
<td>Durham</td>
</tr>
</tbody>
</table>

Fact table

- Big
- Constantly growing
- Stores measures (often aggregated in queries)

Dimension table

- Small
- Updated infrequently
Data cube
Simplified schema: Sale (CID, PID, SID, qty)

Completing the cube—plane
Total quantity of sales for each product in each store
SELECT PID, SID, SUM(qty) FROM Sale
GROUP BY PID, SID;

Completing the cube—axis
Total quantity of sales for each product
SELECT PID, SUM(qty) FROM Sale GROUP BY PID;
Completing the cube—origin

![Graph showing total quantity of sales and further project points onto the origin.]

CUBE operator

- Sale (CID, PID, SID, qty)
- Proposed SQL extension:
  SELECT SUM(qty) FROM Sale
  GROUP BY CUBE CID, PID, SID;
- Output contains:
  - Normal groups produced by GROUP BY
    - (c1, p1, s1, sum), (c1, p2, s3, sum), etc.
  - Groups with one or more ALL’s
    - (ALL, p1, s1, sum), (c2, ALL, ALL, sum), (ALL, ALL, ALL, sum), etc.
- Can you write a CUBE query using only GROUP BY’s?

Gray et al., "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total." ICDE 1996

Automatic summary tables

- Computing GROUP BY and CUBE aggregates is expensive
- OLAP queries perform these operations over and over again
  - Idea: precompute and store the aggregates as automatic summary tables (a DB2 term)
    - Maintained automatically as base data changes
    - Same as materialized views
Aggregation view lattice

Roll up

GROUP BY ∅

GROUP BY CID

GROUP BY PID

GROUP BY SID

GROUP BY CID, PID

GROUP BY CID, SID

GROUP BY PID, SID

GROUP BY CID, PID, SID

A parent can be computed from any child

Selecting views to materialize

- Factors in deciding what to materialize
  - What is its storage cost?
  - What is its update cost?
  - Which queries can benefit from it?
  - How much can a query benefit from it?

- Example
  - GROUP BY ∅ is small, but not useful to most queries
  - GROUP BY CID, PID, SID is useful to any query, but too large to be beneficial

Hazinrayar and al., "Implementing Data Cubes Efficiently," SIGMOD 1996

Data mining

- Data → knowledge
- DBMS meets AI and statistics
- Clustering, prediction (classification and regression), association analysis, outlier analysis, evolution analysis, etc.
  - Usually complex statistical "queries" that are difficult to answer → often specialized algorithms outside DBMS

- We will focus on frequent itemset mining
Mining frequent itemsets

- Given: a large database of transactions, each containing a set of items
  - Example: market baskets
- Find all frequent itemsets
  - A set of items $X$ is frequent if no less than $\min\%$ of all transactions contain $X$
  - Examples: {diaper, beer}, {scanner, color printer}

<table>
<thead>
<tr>
<th>TID</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T001</td>
<td>diaper, milk, candy</td>
</tr>
<tr>
<td>T002</td>
<td>milk, egg</td>
</tr>
<tr>
<td>T003</td>
<td>milk, beer</td>
</tr>
<tr>
<td>T004</td>
<td>diaper, milk, egg</td>
</tr>
<tr>
<td>T005</td>
<td>diaper, beer</td>
</tr>
<tr>
<td>T006</td>
<td>milk, beer</td>
</tr>
<tr>
<td>T007</td>
<td>diaper, beer</td>
</tr>
<tr>
<td>T008</td>
<td>diaper, milk, beer, candy</td>
</tr>
<tr>
<td>T009</td>
<td>diaper, milk, beer</td>
</tr>
</tbody>
</table>

First try

- A naïve algorithm
  - Keep a running count for each possible itemset
  - For each transaction $T$, and for each itemset $X$, if $T$ contains $X$ then increment the count for $X$
  - Return itemsets with large enough counts
- Problem:

  - Think: How do we prune the search space?

The Apriori property

- All subsets of a frequent itemset must also be frequent
  - Because any transaction that contains $X$ must also contains subsets of $X$
  - If we have already verified that $X$ is infrequent, there is no need to count $X$'s supersets because they must be infrequent too
The Apriori algorithm

Multiple passes over the transactions
- Pass $k$ finds all frequent $k$-itemsets (itemset of size $k$)
- Use the set of frequent $k$-itemsets found in pass $k$ to construct candidate $(k+1)$-itemsets to be counted in pass $(k+1)$
  - A $(k+1)$-itemset is a candidate only if all its subsets of size $k$ are frequent

Example: pass 1

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>A, B, E</td>
</tr>
<tr>
<td>002</td>
<td>B, D</td>
</tr>
<tr>
<td>003</td>
<td>B, C</td>
</tr>
<tr>
<td>004</td>
<td>A, B, D</td>
</tr>
<tr>
<td>005</td>
<td>A, C</td>
</tr>
<tr>
<td>006</td>
<td>B, C</td>
</tr>
<tr>
<td>007</td>
<td>A, C</td>
</tr>
<tr>
<td>008</td>
<td>A, B, C, E</td>
</tr>
<tr>
<td>009</td>
<td>A, B, C</td>
</tr>
<tr>
<td>010</td>
<td>F</td>
</tr>
</tbody>
</table>

Transactions $\delta_{\text{min}} = 20\%$

Frequent 1-itemsets
- Itemset {F} is infrequent

Example: pass 2

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>A, B, E</td>
</tr>
<tr>
<td>002</td>
<td>B, D</td>
</tr>
<tr>
<td>003</td>
<td>B, C</td>
</tr>
<tr>
<td>004</td>
<td>A, B, D</td>
</tr>
<tr>
<td>005</td>
<td>A, C</td>
</tr>
<tr>
<td>006</td>
<td>B, C</td>
</tr>
<tr>
<td>007</td>
<td>A, C</td>
</tr>
<tr>
<td>008</td>
<td>A, B, C, E</td>
</tr>
<tr>
<td>009</td>
<td>A, B, C</td>
</tr>
<tr>
<td>010</td>
<td>F</td>
</tr>
</tbody>
</table>

Transactions $\delta_{\text{min}} = 20\%$

Frequent 1-itemsets
- Itemset {F} is infrequent

Candidate 2-itemsets
- Generate candidates
- Scan and count
- Check min. support

Frequent 2-itemsets
Example: pass 3

Transactions
\( \text{min}\% = 20\% \)

\[
\begin{array}{l}
\text{TID} & \text{items} \\
001 & A, B, E \\
002 & B, D \\
003 & C \\
004 & A, B, D \\
005 & A, C \\
006 & A, C \\
007 & A, C \\
008 & A, B, C, E \\
009 & A, C \\
010 & F
\end{array}
\]

Generate candidates

\[
\begin{array}{l}
\text{itemset count} \\
\{A, B\} & 4 \\
\{A, C\} & 4 \\
\{A, E\} & 2 \\
\{B, D\} & 2 \\
\{B, E\} & 2 \\
\{B, C\} & 4 \\
\{B, E\} & 2 \\
\end{array}
\]

Frequent 2-itemsets

Example: pass 4

Transactions
\( \text{min}\% = 20\% \)

\[
\begin{array}{l}
\text{TID} & \text{items} \\
001 & A, B, E \\
002 & B, D \\
003 & B, C \\
004 & A, B, D \\
005 & A, C \\
006 & B, C \\
007 & A, C \\
008 & A, B, C, E \\
009 & A, B, C \\
010 & F
\end{array}
\]

Generate candidates

\[
\begin{array}{l}
\text{itemset count} \\
\{A, B, C\} & 2 \\
\{A, B, E\} & 2 \\
\end{array}
\]

Frequent 3-itemsets

Example: final answer

\[
\begin{array}{l}
\text{itemset count} \\
\{A\} & 6 \\
\{B\} & 7 \\
\{C\} & 6 \\
\{D\} & 2 \\
\{E\} & 2 \\
\end{array}
\]

Frequent 1-itemsets

\[
\begin{array}{l}
\text{itemset count} \\
\{A, B\} & 4 \\
\{A, C\} & 4 \\
\{A, E\} & 2 \\
\{B, D\} & 2 \\
\{B, E\} & 2 \\
\end{array}
\]

Frequent 2-itemsets

\[
\begin{array}{l}
\text{itemset count} \\
\{A, B, C\} & 2 \\
\{A, B, E\} & 2 \\
\end{array}
\]

Frequent 3-itemsets