Data Analytics Beyond OLAP

Prof. Yanlei Diao
OPERATIONAL DBs

DB 1  DB 2  DB 3

EXTRACT
TRANSFORM
LOAD
(ETL)

METADATA
STORE

DATA
WAREHOUSE

SUPPORTS

OLAP

DATA MINING

INTERACTIVE DATA EXPLORATION
Overview of Topics

- **Data Mining and Knowledge Discovery in Databases**
  - Association rule mining
  - Interesting visualizations

- **Approximate Query Processing**
  - Online aggregation: group by aggregation, wander join
  - Interactive SQL

- **Interactive Data Exploration**
  - Faceted search
  - Semantic windows
  - Explore by example
1. Association Rule Mining

*Fast Algorithms for Mining Association Rules*

Rakesh Agrawal and Ramakrishnan Srikant
VLDB '94
• Example Rules:
  – 98% of customers who purchase tires get automotive services done
  – Customers who buy mustard and ketchup also buy burgers
  – Goal: find these rules from transactional data

• Rules help with decision making
  – E.g., store layout, buying patterns, add-on sales
Association Mining

DB of "Basket Data"

<table>
<thead>
<tr>
<th>TID</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

Association Rules

{1} => {3}
{2,3} => {5}
{2,5} => {3}
Associate Rules

**DB of "Basket Data"**

<table>
<thead>
<tr>
<th>TID</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

**Association Rules**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>=&gt; {3}</td>
</tr>
<tr>
<td>{2,3}</td>
<td>=&gt; {5}</td>
</tr>
<tr>
<td>{2,5}</td>
<td>=&gt; {3}</td>
</tr>
</tbody>
</table>

**Association rule: X => Y**

- X and Y are disjoint itemsets, called **antecedent** (LHS) and **consequent** (RHS)

- **Confidence**: c% of transactions that contain X also contain Y (rule-specific)
- **Support**: s% of all transactions contain both X and Y (relative to all data)

- **Goal**: find all rules that satisfy the confidence and support thresholds.
### Support Example

<table>
<thead>
<tr>
<th>TID</th>
<th>Cereal</th>
<th>Beer</th>
<th>Bread</th>
<th>Bananas</th>
<th>Milk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Support** (Cereal)

\[
\frac{4}{8} = 0.5
\]

**Support** (Cereal => Milk)

\[
\frac{3}{8} = 0.375
\]
## Confidence Example

<table>
<thead>
<tr>
<th>TID</th>
<th>Cereal</th>
<th>Beer</th>
<th>Bread</th>
<th>Bananas</th>
<th>Milk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Confidence(\textbf{Cereal} => \textbf{Milk}) \[\frac{3}{4} = .75\]

Confidence(\textbf{Bananas} => \textbf{Bread}) \[\frac{1}{3} = .33333...\]
Apriori Algorithm and Notation

- \{i_1, i_2, \ldots, i_m\} be the set of literals, known as items
- \{ T_j \} is the set of transactions (database), where each transaction T_j is a set of items s.t.
  - Each transaction has a unique identifier TID
  - The size of an itemset is the number of items
  - Itemset of size k is a k-itemset
- Assume that items in an itemset are sorted in lexicographical order
General Strategy

- **Step I**: Find all itemsets with minimum support (min_sup s)

<table>
<thead>
<tr>
<th>TID</th>
<th>items</th>
<th>support</th>
<th>itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
<td>0.25</td>
<td>{4}, {1,2}, {1,4}, {1,5}, {3,4},</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>{1,3,4}, {1,2,3}, {1,2,5}, {1,3,5},</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>{1,2,3,5}</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
<td>0.5</td>
<td>{1}, {1,3}, {2,3}, {3,5}, {2,3,5}</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
<td>0.75</td>
<td>{2}, {3}, {5}, {2,5}</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Step II**: Generate rules from min_sup'ed itemsets

<table>
<thead>
<tr>
<th>support</th>
<th>confidence</th>
<th>rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>66%</td>
<td>{3}=&gt;{1}, {3}=&gt;{2}, {2}=&gt;{3}, {3}=&gt;{5},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{5}=&gt;{3}, {5}=&gt;{2,3}, {3}=&gt;{2,5},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{2}=&gt;{3,5}, {5,2}=&gt;{3}, {5,3}=&gt;{2}</td>
</tr>
<tr>
<td>0.5</td>
<td>100%</td>
<td>{1}=&gt;{3}, {5,3}=&gt;{2}, {2,3}=&gt;{5}</td>
</tr>
<tr>
<td>0.75</td>
<td>100%</td>
<td>{5}=&gt;{2}, {2}=&gt;{5}</td>
</tr>
</tbody>
</table>
Step I: Finding Minsup Itemsets

- What is the complexity of finding all subsets of items that satisfy the mini_sup s?
- The power set of the n literals!
- A new algorithmic framework based on anti-monotonicity:
  - For a frequent itemset, all of its subsets are also frequent
  - For an infrequent itemset, all of its supersets must be infrequent
  - Can be used to design efficient pruning of the search space.
Anti-monotonicity

- Adding items to an itemset never increases its support
- For a frequent itemset, all of its subsets are also frequent
- For an infrequent itemset, all of its supersets must be infrequent
Anti-monotonicity

- Adding items to an itemset never increases its support
- For a frequent itemset, all of its subsets are also frequent
- For an infrequent itemset, all of its supersets must be infrequent
Anti-monotonicity

- Adding items to an itemset never increases its support
- For a frequent itemset, all of its subsets are also frequent
- For an infrequent itemset, all of its supersets must be infrequent
Step I: Finding Minsup Itemsets

- **Anti-monotonicity:**
  
  Adding items to an itemset never increases its support

- **Apriori Algorithm:** Proceed inductively on itemset size

  1) Base case: Begin with all minsup itemsets of size 1 \( (L_1) \)

  2) Without peeking at the DB, generate candidate itemsets of size \( k \) \( (C_k) \) from \( L_{k-1} \)

  3) Remove candidate itemsets that contain unsupported subsets

  4) Further refine \( C_k \) using the database to produce \( L_k \)
Task 2) Guess Itemsets

- **Naïve way:**
  - Extend all itemsets with all possible items

- **Apriori:**
  2) Join $L_{k-1}$ with itself, adding only a single, final item

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Itemset</th>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2 3}</td>
<td>{1 2 4}</td>
<td>{1 2 3}</td>
</tr>
<tr>
<td>{1 2 4}</td>
<td>{1 3 4}</td>
<td>{1 2 4}</td>
</tr>
<tr>
<td>{1 3 4}</td>
<td>{1 3 5}</td>
<td>{1 3 4}</td>
</tr>
<tr>
<td>{1 3 5}</td>
<td>{2 3 4}</td>
<td>{1 3 5}</td>
</tr>
<tr>
<td>{2 3 4}</td>
<td></td>
<td>{2 3 4}</td>
</tr>
</tbody>
</table>

$\triangleleft \triangleright$ equal on first $k-1$ items

<table>
<thead>
<tr>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2 3 4}</td>
</tr>
<tr>
<td>{1 3 4 5}</td>
</tr>
</tbody>
</table>
Task 3) Filter Itemsets

- **Apriori:**
  2) Join $L_{k-1}$ with itself, adding only a single, final item
  3) Remove itemsets with an unsupported subset

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Itemset</th>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2 3}</td>
<td>{1 2 4}</td>
<td>{1 2 3}</td>
</tr>
<tr>
<td>{1 2 4}</td>
<td>{1 3 4}</td>
<td>{1 2 4}</td>
</tr>
<tr>
<td>{1 3 4}</td>
<td>{1 3 5}</td>
<td>{1 3 4}</td>
</tr>
<tr>
<td>{1 3 5}</td>
<td>{2 3 4}</td>
<td>{1 3 5}</td>
</tr>
<tr>
<td>{2 3 4}</td>
<td></td>
<td>{2 3 4}</td>
</tr>
</tbody>
</table>

\[\emptyset \supseteq \{1\} \supseteq \{1 2\} \supseteq \{1 2 3\} \supseteq \{1 2 3 4\}\]
Task 4) Finalize k-Itemsets

- **Apriori:**
  2) Join $L_{k-1}$ with itself, adding only a single, final item
  3) Remove itemsets with an unsupported subset
  4) Use the database to further refine $C_k$
    - Count precisely the occurrence of each itemset in the dataset, to see if it is indeed larger than $\text{min}_\text{sup}$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Itemset</th>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2 3}</td>
<td>{1 2 3}</td>
<td>{1 2 3 4}</td>
</tr>
<tr>
<td>{1 2 4}</td>
<td>{1 2 4}</td>
<td>{1 3 4}</td>
</tr>
<tr>
<td>{1 3 4}</td>
<td>{1 3 4}</td>
<td>{1 3 5}</td>
</tr>
<tr>
<td>{1 3 5}</td>
<td>{1 3 5}</td>
<td>{1 3 4 5}</td>
</tr>
<tr>
<td>{2 3 4}</td>
<td>{2 3 4}</td>
<td>{2 3 4}</td>
</tr>
</tbody>
</table>

equal on first $k-1$ items
Repeat for larger values of $k$

- **Apriori Algorithm:** Proceed inductively on itemset size

1) Base case: Begin with all minsup itemsets of size 1 ($L_1$)

2) Without peeking at the DB, generate candidate itemsets of size $k$ ($C_k$) from $L_{k-1}$

3) Remove candidate itemsets that contain unsupported subsets

4) Further refine $C_k$ using the database to produce $L_k$

repeat

until $L_k$ is empty
General Strategy

- **Step I**: Find all itemsets with *minimum support* (*min_sup* s)

<table>
<thead>
<tr>
<th>TID</th>
<th>items</th>
<th>support</th>
<th>itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
<td>0.25</td>
<td>{4}, {1,2}, {1,4}, {1,5}, {3,4},</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>{1,3,4}, {1,2,3}, {1,2,5}, {1,3,5},</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>{1,2,3,5}</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
<td>0.5</td>
<td>{1}, {1,3}, {2,3}, {3,5}, {2,3,5}</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
<td>0.75</td>
<td>{2}, {3}, {5}, {2,5}</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Step II**: Generate rules from *min_sup*'ed itemsets

<table>
<thead>
<tr>
<th>support</th>
<th>confidence</th>
<th>rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>66%</td>
<td>{3}=&gt;{1}, {3}=&gt;{2}, {2}=&gt;{3}, {3}=&gt;{5}, {5}=&gt;{3}, {5}=&gt;{2,3}, {3}=&gt;{2,5}, {2}=&gt;{3,5}, {5,2}=&gt;{3}, {5,3}=&gt;{2}</td>
</tr>
<tr>
<td>0.5</td>
<td>100%</td>
<td>{1}=&gt;{3}, {5,3}=&gt;{2}, {2,3}=&gt;{5}</td>
</tr>
<tr>
<td>0.75</td>
<td>100%</td>
<td>{5}=&gt;{2}, {2}=&gt;{5}</td>
</tr>
</tbody>
</table>
Full Algorithm

\textbf{Apriori}\{

L_1 = \{\text{large 1-itemsets}\}

\text{for (k = 2; L_{k-1} \neq \emptyset ; k++) } \{ //\text{Generate large k-itemsets}

\text{//Step 1: generate candidate k-itemsets from large (k-1)-itemsets}
C_k = \text{apriori-gen}(L_{k-1});

\text{//Step 2: count support of each candidate set}
\text{forall transactions t in the database } \{

\text{C}_t = \text{subset}(C_k, t); // Candidates from C_k contained in t}
\text{forall candidates c in } C_t

\text{c.count++;
}

\}
L_k = \{c \in C_k \mid \text{c.count} \geq \text{mins}\}

\}
\}

\text{Answer} = \bigcup_k L_k
apriori-gen(Lk-1) {

    // Intuition: every subset of a large itemset must be large.
    // So combine almost-matching pairs of large (k-1)-itemsets,
    // and prune out those with non-large (k-1)-subsets.

    join:

    insert into Ck
    select p.item1, ..., p.itemk-1, q.itemk-1
    from Lk-1 p, Lk-1 q
    where p.item1 = q.item1 and ... and p.itemk-2 = q.itemk-2 and p.itemk-1 < q.itemk-1;

    prune:

    // delete itemsets such that some (k-1)-subset is not in Lk-1
    forall itemsets c in Ck

        forall (k-1)-subsets s of c

            if (s not in Lk-1) {
                delete c from Ck;
                break;
            }

}
2. Interesting Visualizations

SeeDB: efficient data-driven visualization recommendations to support visual analytics

Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, Neoklis Polyzotis
VLDB ’14
Visualization Recommendation

Given a dataset and a task, automatically produce a set of visualizations that are the most “interesting” given the task
For simplicity, assume a single table (star schema)

Visualizations = agg. + grp. by queries

Vi = SELECT d, f(m)
    FROM table
    WHERE ___
    GROUP BY d

(d, m, f):
dimension, measure, aggregate
Space of Visualizations

\[ V_i = \text{SELECT } d, f(m) \]
\[ \text{FROM table} \]
\[ \text{WHERE ___} \]
\[ \text{GROUP BY } d \]

\((d, m, f)\):
- dimension, measure, aggregate
- \{d\} : race, work-type, sex etc.
- \{m\} : capital-gain, capital-loss, hours-per-week
- \{f\} : COUNT, SUM, AVG
Key Questions

I. *Interestingness*: How do we determine if a visualization is interesting?
   
   – Utility Metric

II. *Scale*: How to make recommendations efficiently and interactively?

   – Optimizations
**Deviation-based Utility Metric**

An *interesting* visualization displays *a large deviation from a reference*

**Task:** compare *unmarried* adults with *all adults*

\[
V_1 = \text{SELECT } d, f(m) \text{ FROM table WHERE } \text{target} \text{ GROUP BY } d
\]
\[
V_2 = \text{SELECT } d, f(m) \text{ FROM table WHERE } \text{reference} \text{ GROUP BY } d
\]
Deviation-based Utility Metric

An interesting visualization displays a large deviation from a reference

Many metrics for computing distance between distributions

$D [P(V_1), P(V_2)]$

- Earth mover’s distance
- L1, L2 distance
- K-L divergence

Any distance metric b/n distributions is OK!
Key Questions

I. **Interestingness**: How do we determine if a visualization is interesting?

   - Utility Metric

II. **Scale**: How to compute efficiently and interactively?

   - Need to search through different combinations of \((d, m, f)\):
     
     \(d\): race, work-type, sex etc.
     
     \(m\): capital-gain, capital-loss, hours-per-week
     
     \(f\): COUNT, SUM, AVG

   - Optimizations include: (i) shared execution of queries, (ii) early pruning of non top-k patterns
An Extension: see Project 1

Extracting Top-K Insights from Multi-dimensional Data

Bo Tang, Shi Han, Man Lung Yiu, Rui Ding, Dongmei Zhang. SIGMOD ’17

- Transformations (delta_prev, %, rank, etc.) of data and combinations
- Types of insights: point or shape
- Optimizations
Overview of Topics

- Data Mining over Databases
  - Association rule mining
  - Interesting visualizations

- Approximate Query Processing
  - Online aggregation: group by aggregation, wander join
  - Interactive SQL

- Interactive Data Exploration
  - Faceted search
  - Semantic windows
  - Explore by example
3. Interactive Data Exploration

Explore-by-example: an automatic query steering framework for interactive data exploration

Kyriaki Dimitriadou, Olga Papaemmanouil, Yanlei Diao
SIGMOD ’14
Web Applications: e.g., House Hunting

- 2+ bedrooms, ≤ 0.5 million
- 2+ bedrooms, price depending on the size
- 2+ bedrooms, price depending on the size, location
- 2+ bedrooms, price depending on the size, location, quietness (away from street, higher floor)
- 2+ bedrooms, price depending on the size, location, quietness, school district, building structure...
An Explore-by-Example Framework

- Sample Acquisition
- Sample
- Initial $d$ attributes
- Exploration Queries
- Relevant examples
- Irrelevant examples
- Relevance Feedback
- Classification Model
- User Model
- Query Formulation
- Final Data Extraction Query
- Space Exploration
- SQL
An active learner requires fewer labeled examples than a traditional learner.
Optimization for active learning-based interactive database exploration

E Huang, L Peng, LD Palma, A Abdelkafi, A Liu, Y Diao
Proceedings of the VLDB Endowment (PVLDB), 12 (1), 71-84, 2018