SQL Expressive Powers

1. Relational Algebra or Calculus
2. Aggregation / Grouping
3. Deductive Logics / Analytic Functions (Windowing)
4. Data Mining Features

Ch 26. Data Mining
(26.1, 26.2.1, 26.3.1, 26.3.2)

Sang-Won Lee

http://icc.skku.ac.kr/~swlee

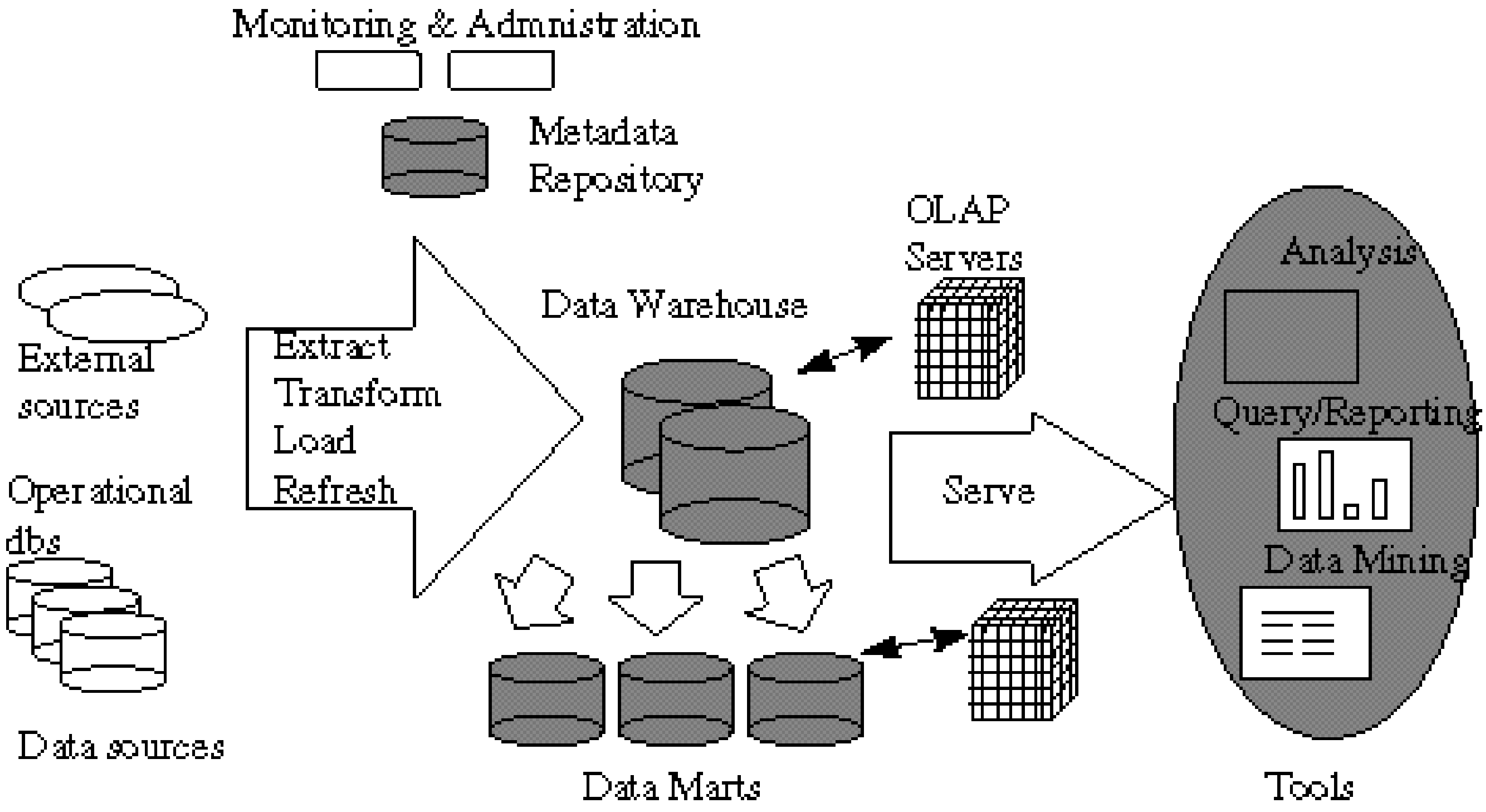
SKKU VLDB Lab.

(http://vldb.skku.ac.kr/)

Motivation

- Data explosion problem
  - Automated data collection tools and mature database technology lead to tremendous amounts of data stored in databases, data warehouses and other information repositories
- We are drowning in data, but starving for knowledge!
- Solution: data warehousing/OLAP and data mining
- Data mining
  - The process of finding interesting (non-trivial, implicit, previously unknown and potentially useful) information, trend, or patterns from data in large databases in order to guide decision
  - vs. statistics, machine learning / knowledge discovery in AI
    - ✓ SCALABILITY with respect to data size (big data)

Data Mining



Source: An overview of data warehousing and OLAP technology, SIGMOD record, March 1997

Definitions

Query & Reporting	OLAP	Data Mining
Extraction of detailed and summary data	Summaries, trends and forecasts	Knowledge discovery of hidden patterns and insights
"Fact & Information"	"Analysis"	"Insight and Prediction"
Who purchased mutual funds in the last 3 years?	What is the average income of mutual fund buyers by region by year?	Who will buy a mutual fund in the next 6 months and why?

Data Mining: Applications

- Market analysis and management
  - target marketing, customer relation management, market basket analysis, cross selling, market segmentation
- Risk analysis and management
  - Forecasting, customer churn and retention, quality control, …
- Fraud detection and management: card fraud
- Sports: IBM Advanced Scout analyzed NBA game statistics (shots blocked, assists, and fouls) to gain competitive advantage for New York Knicks and Miami Heat

Data Mining: An Example Database of Customer Transaction

- Market basket

transid	*custid*	*date*	*item*	*qty*
111	201	5/1/99	pen	2
111	201	5/1/99	ink	1
111	201	5/1/99	milk	3
111	201	5/1/99	juice	6
112	105	6/3/99	pen	1
112	105	6/3/99	ink	1
112	105	6/3/99	milk	1
113	106	5/10/99	pen	1
113	106	5/10/99	milk	1
114	201	6/1/99	pen	2
114	201	6/1/99	ink	2
114	201	6/1/99	juice	4
114	201	6/1/99	water	1

Figure 26.1 The Purchase Relation

Data Mining: Techniques

- Co-ocurrences (that is, frequent itemset counting) & Association Rules
  - 60% of all customers purchase items {X, Y, Z} together
  - 75% of CU male customers who purchase diapers also buy beer
- Sequential Patterns
  - 60% of Amazon customers who buy Harry Porter book purchase its DVD in 3 weeks
- Classification
  - People with (age less than 25 and salary > 40k) mostly drive (sports cars)

Data Mining: Techniques (2)

- Clustering(or Segmentation)
  - Our customers are clustered into N segments
- Similar time sequences
  - Stocks of companies A and B perform similarly
- Outlier Discovery
  - Residential customers for telecom company with businesses at home
- Text/Web Mining, Similar Images

Data Mining: Algorithms

- Association Rules: **A-priori**, Bayesian Network

- Classification: Decision Tree, Sprint
- Segmentation (Clustering): **K-means**, EM, Birch, Cure
- Sequential Patterns: Sprint, Blast
- and many more ...

Market Basket Analysis

transid	custid	date	item	qty
111	201	5/1/99	pen	2
111	201	5/1/99	ink	1
111	201	5/1/99	milk	3
111	201	5/1/99	juice	6
112	105	6/3/99	pen	1
112	105	6/3/99	ink	1
112	105	6/3/99	milk	1
113	106	5/10/99	pen	1
113	106	5/10/99	milk	1
114	201	6/1/99	pen	2
114	201	6/1/99	ink	2
114	201	6/1/99	juice	4
114	201	6/1/99	water	1

- E.g. shopping cart filled with several items
  - Either offline or online

- A common goal for retailers
  1. To identify items that are purchased together (i.e., frequent itemset)
  2. THEN, to improve the layout of goods in a store or the layout of (online) catalog pages

- Market basket analysis tries to answer the following questions:
  - What items do customers buy together? frequent itemset
  - In what order do customers purchase items? sequential pattern
  - Who makes purchases? Classification / clustering

26.2.1 Frequent Itemset

Given:

- A database of customer transactions
- Each transaction is a set of items
  - e.g. TX111 = {Pen, Ink, Milk, Juice}

Transid	Items
111	{pen, ink, milk, juice}
112	{pen, ink, milk}
113	{pen, milk}
114	{pen,ink,jouce, water}

transid	*custid*	*date*	*item*	*qty*
111	201	5/1/99	pen	2
111	201	5/1/99	ink	1
111	201	5/1/99	milk	3
111	201	5/1/99	juice	6
112	105	6/3/99	pen	1
112	105	6/3/99	ink	1
112	105	6/3/99	milk	1
113	106	5/10/99	pen	1
113	106	5/10/99	milk	1
114	201	6/1/99	pen	2
114	201	6/1/99	ink	2
114	201	6/1/99	juice	4
114	201	6/1/99	water	1

Figure 26.1

Frequent Itemset

Transid	Items
111	{pen, ink, milk, juice}
112	{pen, ink, milk}
113	{pen, milk}
114	{pen,ink,jouce, water}

- Itemset = a set of items
  - e.g. {pen}, {pen, ink, milk}
- Support of an itemset = the fraction of transactions that contain all the items in the itemset
  - Sup ({pen,ink}) = 3 / 4 = 75% (frequent?)
  - Sup ({milk, juice}) = 1 / 4 = 25% (infrequent?)
- Frequent Itemsets
  - Given a user-specified minimum support (***minsup***), all itemsets whose support is higher than *minsup*
  - E.g.: minsup = 70%
    - ✓ {Pen}, {Ink}, {Milk},

Frequent Itemset: A Naïve Algorithm

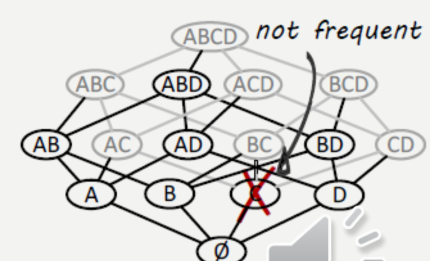
- D: database
- N = # of all transactions T in D;
- I = set of all items in D;
- min-sup = minimum support;

```
for each subset s of I do { // 2^I iterations
    count = 0;
    for each T of N transactions in D do
        if s is a subset of T then
            count++;
        if min-sup <= count/N then
            add s to frequent-itemsets
    end for;
end for;
```

- Analysis of naïve algorithm
  - O(2^I) subsets: exponential growth!
  - Scan N transactions for each of 2^I subsets
  - O(2^I * N) tests of s being subset of T
  - This algorithm becomes **infeasible** quickly..
- Can we do **better**?

(Source: Prof. Dr. Thomas Seidl)

Main idea of the Apriori algorithm: Prune the exponential search space using anti-monotonicity



Frequent Itemset: An Optimization

- The **A-priori** Property: *"Every non-empty subset, S', of a frequent itemset, S, is also a frequent itemset"*
  - Makes use of prior knowledge of subset support properties: Proof:

    - Let $\emptyset \neq S' \subseteq S \subseteq I$ */* where I = set of all items in D */*
    - For any transaction $T \subseteq I$ in database D, we have:

    $$S \subseteq T \Rightarrow S' \subseteq T$$

    - Thus, it holds that

    $$\{T \in D \mid S \subseteq T\} \subseteq \{T \in D \mid S' \subseteq T\}$$

    and consequently

    $$support(S) = |\{T \in D \mid S \subseteq T\}| \leq |\{T \in D \mid S' \subseteq T\}| = support(S')$$

{Pen, Ink, Milk} Frequent?

{Pen, Ink}, {Ink, Milk}, {Pen, Milk}

(Source: Prof. Dr. Thomas Seidl)

- It gives
  - Efficient execution: the number of candidate frequent itemsets can be significantly reduced by considering only itemsets obtained by enlarging frequent itemsets.
  - Complete computation: any frequent itemset is not missed.

A-Priori Algorithm

- Simple version

```
foreach item,                                                        // Level 1
    check if it is a frequent itemset  // appears in > minsup transactions
k = 1
repeat           // Iterative, level-wise identification of frequent itemsets
    foreach new frequent itemset I_k with k items                 // Level k + 1
        generate all itemsets I_{k+1} with k + 1 items, I_k ⊂ I_{k+1}   // aprior-gen step
    Scan all transactions once and check if
    the generated k + 1-itemsets are frequent
    k = k + 1
until no new frequent itemsets are identified
```

Figure 26.2 An Algorithm for Finding Frequent Itemsets

- Further optimization
  - Reduce # of candidate itemsets further using the **A-priori** property

Frequent Itemset: A-priori Algorithm

Source: "Fast algorithms for Mining Association Rules" (VLDB 1994)

```
L_1 = {large 1-itemsets};
for ( k = 2; L_{k-1} ≠ ∅; k++ ) do begin
    C_k = apriori-gen(L_{k-1});  // New candidates
    forall transactions t ∈ D do begin
        C_t = subset(C_k, t);  // Candidates contained in t
        forall candidates c ∈ C_t do
            c.count++;
    end
    L_k = {c ∈ C_k | c.count ≥ minsup}
end
Answer = ⋃_k L_k;
```

Figure 1: Algorithm Apriori

1. join step for new candidate generation

```
insert into C_k
select p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}
from L_{k-1} p, L_{k-1} q
where p.item_1 = q.item_1, . . ., p.item_{k-2} = q.item_{k-2},
      p.item_{k-1} < q.item_{k-1};
```

2. Pruning invalid candidates

```
forall itemsets c ∈ C_k do
    forall (k−1)-subsets s of c do
        if (s ∉ L_{k-1}) then
            delete c from C_k;
```

Frequent Itemset

- The A-priori algorithm: min. sup. 70% (from textbook)

Database D

Transid	Items
111	{pen, ink, milk, juice}
112	{pen, ink, milk}
113	{pen, milk}
114	{pen,ink,juice, water}

1st Scan D →

C_1

Itemset	Sup.
{pen}	4
{ink}	3
{milk}	3
{juice}	2
{water}	1

L_1

Itemset	Sup.
{pen}	4
{ink}	3
{milk}	3

C_2

Itemset
{pen, ink}
{pen, milk}
{ink, milk}

2nd Scan D →

C_2

Itemset	Sup.
{pen, ink}	3
{pen,milk}	3
{ink,milk}	2

L_2

Itemset	Sup.
{pen,ink}	3
{pen,milk}	3

C3

Itemset
{pen, ink, milk}

3rd Scan D →

C_3

Itemset	Sup.

L_3

Simple version:
EMPTY == > No new frequent itemset == > STOP!

Optimized version:
{ink,milk} not in L2 (not frequent)
== > {pen,ink,milk} not frequent
== > pruned == > skip 3rd scan of D

Frequent Itemset

- The A-priori algorithm: another example with min. sup. = 2

Database D

Transid	Items
100	{1, 3, 4}
200	{2, 3, 5}
300	{1, 2, 3, 5}
400	{2,5}

Scan D →

C1

Itemset	Sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L1

Itemset	Sup.
{1}	2
{2}	3
{3}	3
{5}	3

C2

Itemset
{1, 2}
{1, 3}
{1, 5}
{2, 3}
{2, 5}
{3, 5}

Scan D →

C2

Itemset	Sup.
{1, 2}	1
{1, 3}	2
{1, 5}	1
{2, 3}	2
{2, 5}	3
{3, 5}	2

L2

Itemset	Sup.
{1, 3}	2
{2, 3}	2
{2, 5}	3
{3, 5}	2

C3

Itemset
{2, 3, 5}

Scan D →

C3

Itemset	Sup.
{2,3,5}	2

L3

Itemset	Sup.
{2,3,5}	2

What about the itemset {1,3,5}?

{1,5} not in L2 == > {1,3,5} was pruned!

26.3.1 Mining Association Rules

- Given:
  - A database of customer transactions, each of which is a set of items

- Find all rules X ==> Y that correlate the presence of one set of items X with another set of items Y
  - e.g. {diaper, baby_food} ==> {beer} with the probability of 60%
  - Any number of items in the consequent(Y)/antecedent(X) of a rule

- Sample Applications
  - market basket analysis, attached mailing in direct marketing, fraud detection for medical insurance, cross-selling (department store floor/shelf planning, Web-site layout)

Support and Confidence

- A rule must have some minimum user-specified ***support***
  - Rule "X ==> Y" has support **s** if P(XY) = s
  - E.g. {1 , 2} ==> {3} should hold in some minimum percentage of transactions to have business value

- A rule must have some minimum user-specified ***confidence***
  - rule X ==> Y has confidence **c** if P(Y|X) = c
  - e.g. {1, 2} ==> {3} has 90% confidence if 90% of the transactions that contain the itemset {1,2} also contain the itemset {3}.

Very Large Data Bases

Support and Confidence (2)

Association rules

Examples:

- {pen} ==> {milk}: sup. 75%, conf. 75%
- {ink} ==> {pen}: sup. 75%, conf. 100%
- {pen} ==> {ink}: sup. 75%, conf. 75%

transid	*custid*	*date*	*item*	*qty*
111	201	5/1/99	pen	2
111	201	5/1/99	ink	1
111	201	5/1/99	milk	3
111	201	5/1/99	juice	6
112	105	6/3/99	pen	1
112	105	6/3/99	ink	1
112	105	6/3/99	milk	1
113	106	5/10/99	pen	1
113	106	5/10/99	milk	1
114	201	6/1/99	pen	2
114	201	6/1/99	ink	2
114	201	6/1/99	juice	4
114	201	6/1/99	water	1

- Example: find all association rules with sup. >= 50% and conf. >= 75%

26.3.2 An Algorithm for Finding Association Rules

- "Association Rule Finding" works in two phases

  Phase 1: Given ***minsup***, find all frequent itemsets

  - ✓ Use **"A-priori algorithm"**
  - ✓ Most expensive phase!

  Phase 2: Given **confidence C**, use the frequent itemsets to generate the desired association rules

  - ✓ Generation is straightforward (see the latter part in Section 26.3.2)
    1. For each frequent itemset X, divide X into LHS and RHS
    2. Conf (LHS → RHS) = Support(X) / Support (LHS)
       - Support(X), Support (LHS): already computed in Phase1
    3. Compare ***Conf (LHS → RHS)*** with **C**

Oracle Data Mining

- Oracle Data Mining Concepts 12c Release 2 (12.2)

Mining Functions	Mining Algorithms
Regression Classification Anomaly Detection Clustering Association Feature Selection and Extraction	**Apriori** Decision Tree Expectation Maximization Explicit Semantic Analysis Generalized Linear Models k-Means Minimum Description Length Naive Bayes Non-Negative Matrix Factorization O-Cluster Singular Value Decomposition Support Vector Machines