Data Mining:

Concepts and Techniques

— Chapter 5 —

Jiawei Han
Department of Computer Science
University of Illinois at Urbana-Champaign
www.cs.uiuc.edu/~hanj

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

What Is Frequent Pattern Analysis?

- Frequent pattern: a pattern (itemsets, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Bread and milk?! – frequent itemset
  - What are the subsequent purchases after buying a PC? – camera, memory card – frequent sequential pattern
  - Subgraph, sub tree, sub lattices – frequent structured pattern
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing (mobile phone and artist's ringtone), catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Why Is Freq. Pattern Mining Important?

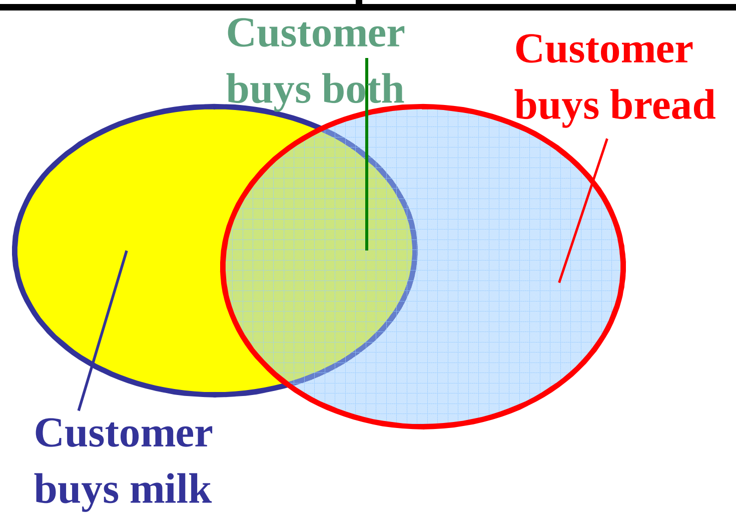
- Discover associations and correlations
- Large data → frequent pattern analysis → enhancement in decision making process
- Input : set of items – each item has a boolean variable - present (1/0)
- Each basket has a boolean vector for all items [1,0,1,0…] – item 1 and 3 are in basket
- Boolean vector is analyzed and association rules are formed
- Computer → antivirus [sup=2%, conf=60%]
- Rule should satisfy min support and min confidence

Why Is Freq. Pattern Mining Important?

- Discloses an intrinsic and important property of data sets
- Forms the foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: associative classification
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

Basic Concepts: Frequent Patterns and Association Rules

Transaction-id TID	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F



- Itemset $X = \{x_1, \ldots, x_k\}$
- Find all the rules $X \rightarrow Y$ with minimum support and confidence
  - support, *s*, probability that a transaction contains $X \cup Y$
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

Let $sup_{min} = 50\%$, $conf_{min} = 50\%$

Freq. Pat.: {*A:3, B:3, D:4, E:3, AD:3*}

Association rules:

$A \rightarrow D$ (60%, 100%)

$D \rightarrow A$ (60%, 75%)

$$support(A \Rightarrow B) = P(A \cup B)$$
$$confidence(A \Rightarrow B) = P(B|A).$$

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

- Occurrence frequency, frequency, support count, count
- **1. Find all frequent itemsets:** By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, *min sup.*
- **2. Generate strong association rules from the frequent itemsets:** By definition, these rules must satisfy minimum support and minimum confidence.

Closed Patterns and Max-Patterns

- A long all pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 = 1.27 \times 10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ⊃ X, *with the same support* as X (proposed by Pasquier, et al. @ ICDT'99)
- An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X (proposed by Bayardo @ SIGMOD'98)
- Closed pattern is a lossless compression of freq. patterns
    - Reducing the # of patterns and rules

Closed Patterns and Max-Patterns

- Exercise. DB of T=

  $\{<a_1, \ldots, a_{100}>, < a_1, \ldots, a_{50}>\}$

  - Min_sup = 1.
- What is the set of closed itemset?
  - $<a_1, \ldots, a_{100}>$: 1
  - $< a_1, \ldots, a_{50}>$: 2
- What is the set of max-pattern?
  - $<a_1, \ldots, a_{100}>$: 1

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Scalable Methods for Mining Frequent Patterns

- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB'94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

Apriori: A Candidate Generation-and-Test Approach

- Apriori property: All nonempty subsets of a frequent itemset must also be frequent
- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method: iterative – levelwise search
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated
- Apriori – Antimonotone - If a set does not pass a test , all its superset will fail the same test as well

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1^{st} scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2^{nd} scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3^{rd} scan

L_3

Itemset	sup
{B, C, E}	2

Generating Association Rules from Frequent Itemsets

- Form *strong* association rules satisfy both minimum support and minimum confidence

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

For each frequent itemset l, generate all nonempty subsets of l.

For every nonempty subset s of l, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq$ min_conf, where min_conf is the minimum confidence threshold.

Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning – uses apriori property
- How to count supports of candidates?
- Example of Candidate-generation
  - L_3=*{abc, abd, acd, ace, bcd}*
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4=*{abcd}*

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Absolute support min_sup=2

Relative support = 22%
(22/100 * Total transactions)
22/100*9 = 2

Scan D for count of each candidate →

C_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Compare candidate support count with minimum support count →

L_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Generate C_2 candidates from L_1 →

Joining

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

Scan D for count of each candidate →

C_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

Compare candidate support count with minimum support count →

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Pruning **{I1, I2, I4} , {I1, I2, I5} - ???**

C_4-{I1,I2,I3,I5} ---?????

Generate C_3 candidates from L_2 →

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

Scan D for count of each candidate →

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Compare candidate support count with minimum support count →

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

(a) Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\} \bowtie$
$\{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$= \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$.

(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of L_2. Therefore, keep {I1, I2, I3} in C_3.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of L_2. Therefore, keep {I1, I2, I5} in C_3.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I1, I3, I5} from C_3.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I4} from C_3.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I5} from C_3.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I4, I5} from C_3.

(c) Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- *min_sup*, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
L_1 = find_frequent_1-itemsets(D);
for (k = 2; L_{k-1} ≠ ϕ; k++) {
    C_k = apriori_gen(L_{k-1});
    for each transaction t ∈ D { // scan D for counts
        C_t = subset(C_k, t); // get the subsets of t that are candidates
        for each candidate c ∈ C_t
            c.count++;
    }
    L_k = {c ∈ C_k | c.count ≥ min_sup}
}
return L = ∪_k L_k;
```

```
procedure apriori_gen(L_{k-1}:frequent (k − 1)-itemsets)
    for each itemset l1 ∈ L_{k-1}
        for each itemset l2 ∈ L_{k-1}
            if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
                c = l1 ⋈ l2; // join step: generate candidates
                if has_infrequent_subset(c, L_{k-1}) then
                    delete c; // prune step: remove unfruitful candidate
                else add c to C_k;
            }
    return C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
            L_{k-1}: frequent (k − 1)-itemsets); // use prior knowledge
    for each (k − 1)-subset s of c
        if s ∉ L_{k-1} then
            return TRUE;
    return FALSE;
```

Generating association rules. Let's try an example based on the transactional data for *AllElectronics* shown in Table 5.1. Suppose the data contain the frequent itemset $l = \{I1, I2, I5\}$. What are the association rules that can be generated from l? The nonempty subsets of l are $\{I1, I2\}$, $\{I1, I5\}$, $\{I2, I5\}$, $\{I1\}$, $\{I2\}$, and $\{I5\}$. The resulting association rules are as shown below, each listed with its confidence:

$$
\begin{array}{ll}
I1 \wedge I2 \Rightarrow I5, & confidence = 2/4 = 50\% \\
I1 \wedge I5 \Rightarrow I2, & confidence = 2/2 = 100\% \\
I2 \wedge I5 \Rightarrow I1, & confidence = 2/2 = 100\% \\
I1 \Rightarrow I2 \wedge I5, & confidence = 2/6 = 33\% \\
I2 \Rightarrow I1 \wedge I5, & confidence = 2/7 = 29\% \\
I5 \Rightarrow I1 \wedge I2, & confidence = 2/2 = 100\%
\end{array}
$$

If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules above are output, because these are the only ones generated that are strong. Note that, unlike conventional classification rules, association rules can contain more than one conjunct in the right-hand side of the rule. ■

L={I1, I2, I3} → association rules ????

Efficient Implementation of Apriori in SQL

- Hard to get good performance out of pure SQL (SQL-92) based approaches alone
- Make use of object-relational extensions like UDFs, BLOBs, Table functions etc.
  - Get orders of magnitude improvement
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. In SIGMOD'98

Challenges of Frequent Pattern Mining

- Challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

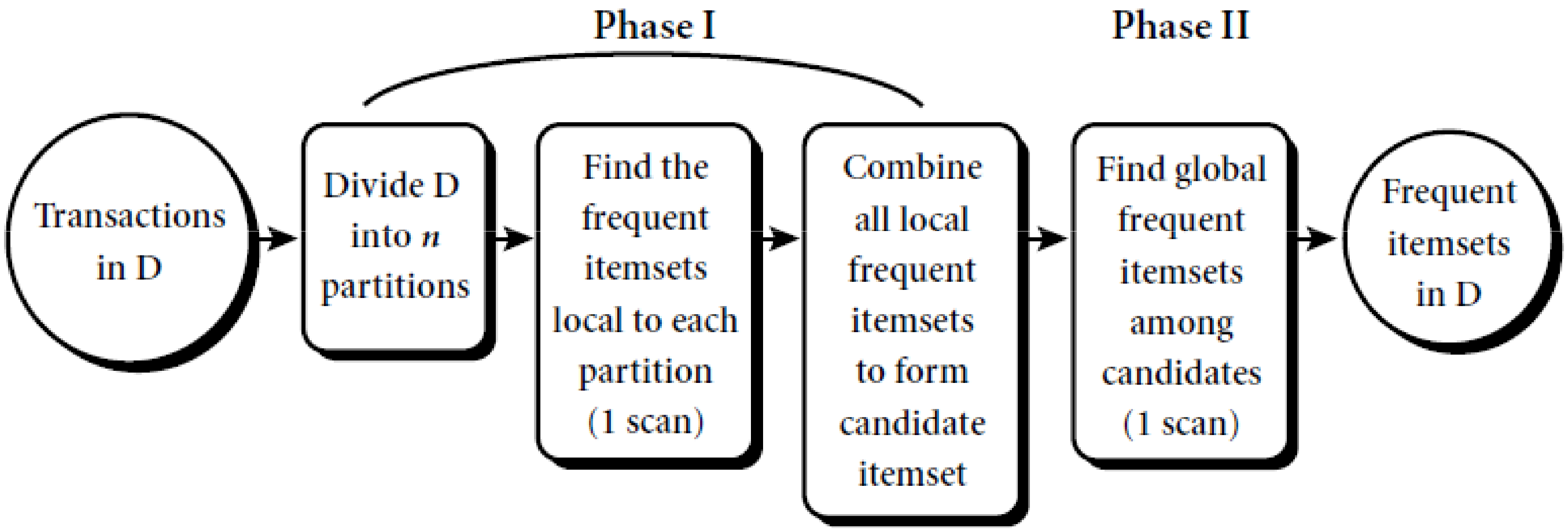
Transaction Reduction

- Reduce the number of transactions scanned in future iterations
- A transaction that does not contain any frequent *k-itemsets cannot contain any* frequent (*k+1)-itemsets*

Partition: Scan Database Only Twice

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association in large databases. In *VLDB'95*

Partition: Scan Database Only Twice



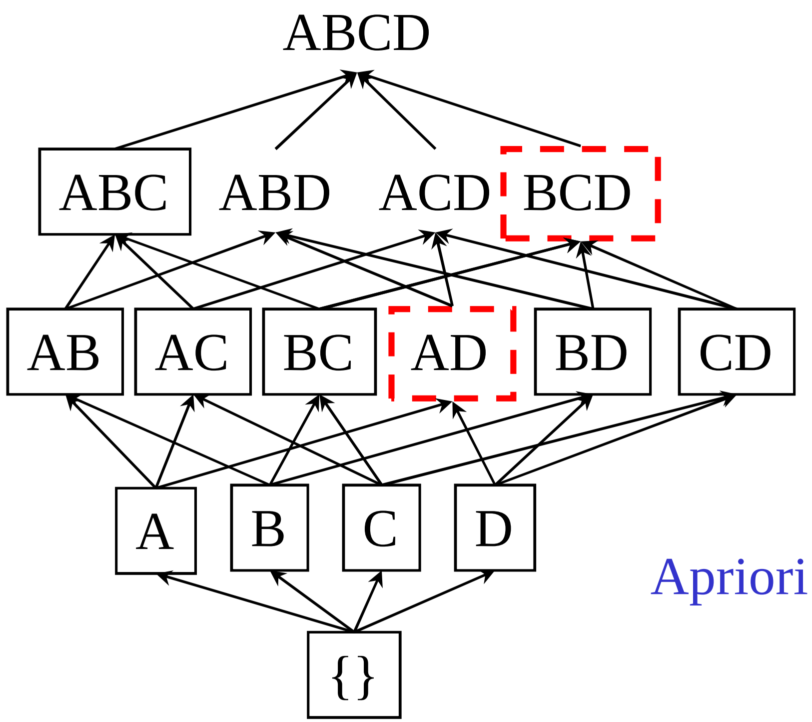
Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
- Scan database again to find missed frequent patterns
- Use lower support threshold than min support – to avoid frequent itemset miss in sample and DB
- H. Toivonen. Sampling large databases for association rules. In *VLDB'96*

DHP: Reduce the Number of Candidates

- A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - Candidates: a, b, c, d, e
  - Hash entries: {ab, ad, ae} {bd, be, de} …
  - Frequent 1-itemset: a, b, d, e
  - ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below support threshold
  - Direct Hashing and pruning
- J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. In *SIGMOD'95*

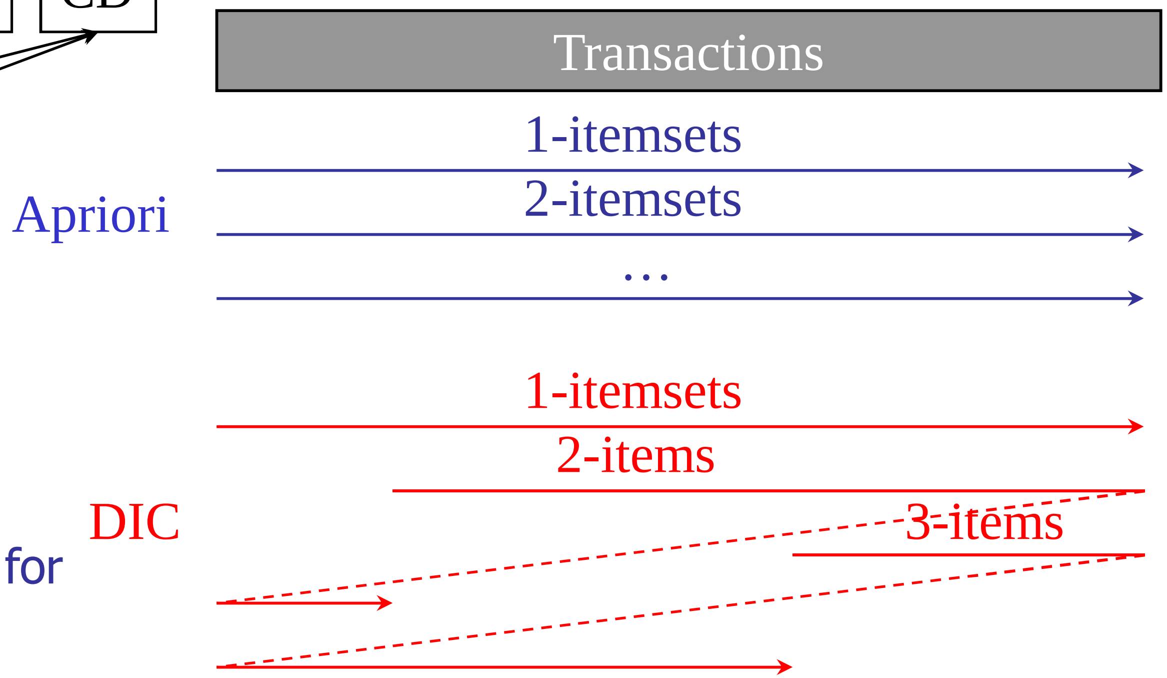
DIC: Reduce Number of Scans



Itemset lattice

S. Brin R. Motwani, J. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD'97*

- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins



Bottleneck of Frequent-pattern Mining

- Multiple database scans are costly
- Mining long patterns needs many passes of scanning and generates lots of candidates
  - To find frequent itemset $i_1 i_2 \ldots i_{100}$
    - # of scans: 100
    - # of Candidates: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 = 1.27*10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?

Mining Frequent Patterns Without Candidate Generation

- Divide and conquer
- It compresses DB of frequent items into frequent pattern tree or FP tree
- It then divides the compressed DB into a set o conditional databases(projected DB), each associated with one frequent item or pattern fragment and mines separately
- Grow long patterns from short ones using local frequent items
  - "abc" is a frequent pattern
  - Get all transactions having "abc": DB|abc
  - "d" is a local frequent item in DB|abc → abcd is a frequent pattern

Construct FP-tree from a Transaction Database

TID	*Items bought*	*(ordered) frequent items*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o, w*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

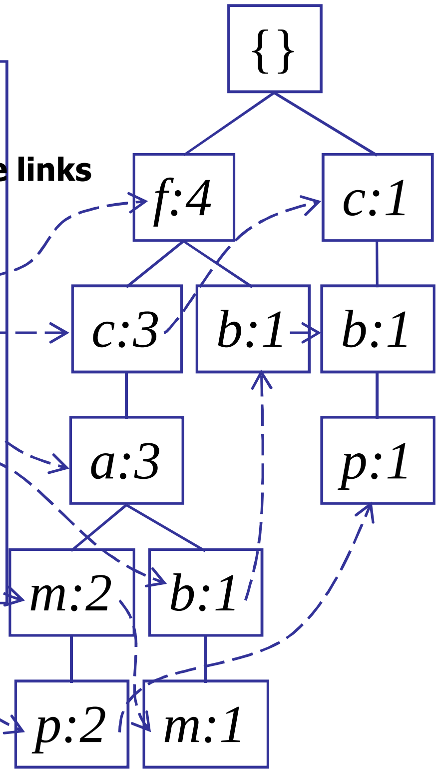
min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

Header Table

Item	*frequency*	*head*
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	

Node links



F-list=f-c-a-b-m-p

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

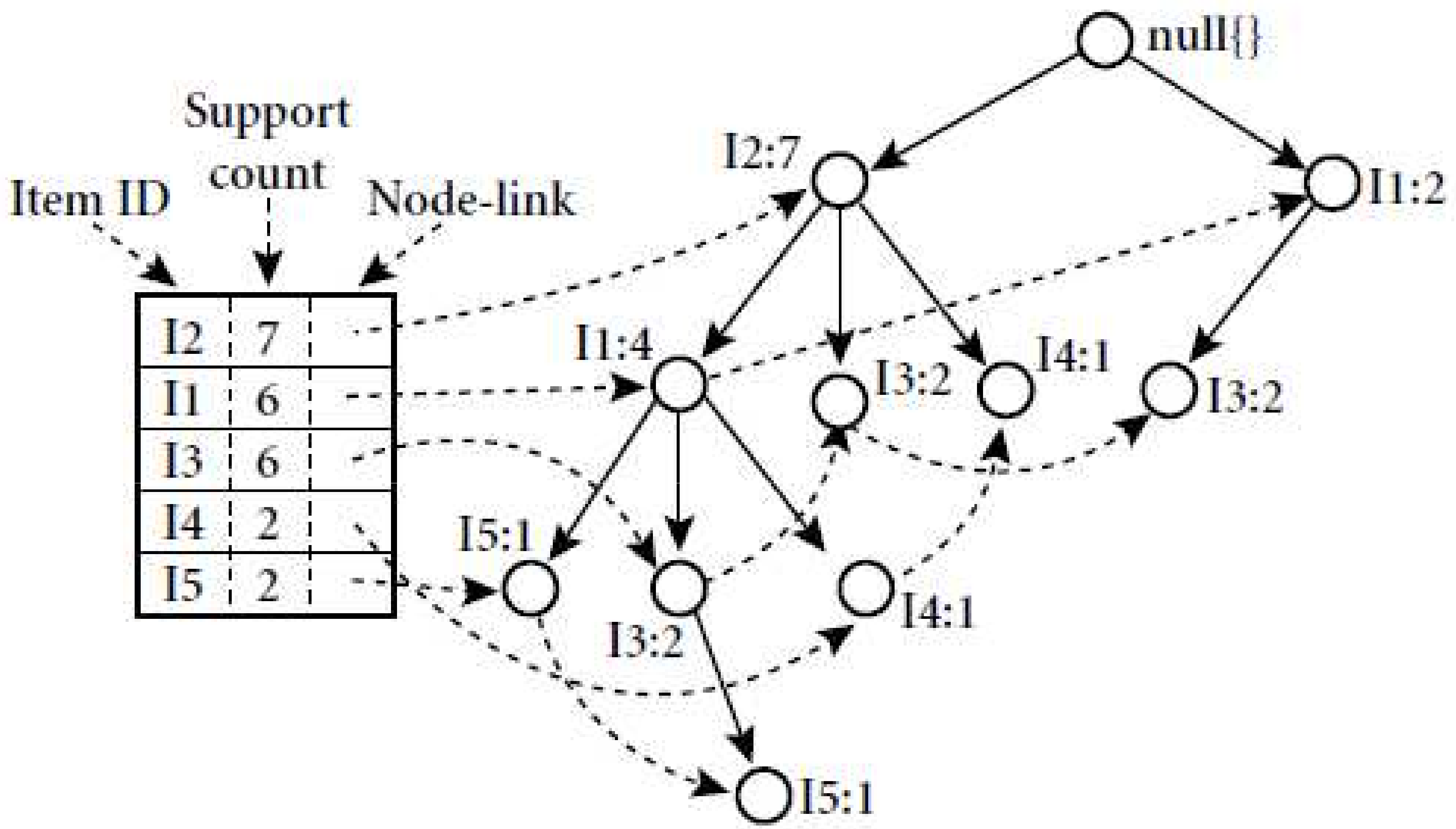
Absolute support min_sup=2

Relative support = 22%
(22/100 * Total transactions)
22/100*9 = 2

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

I2	7
I1	6
I3	6
I4	2
I5	2

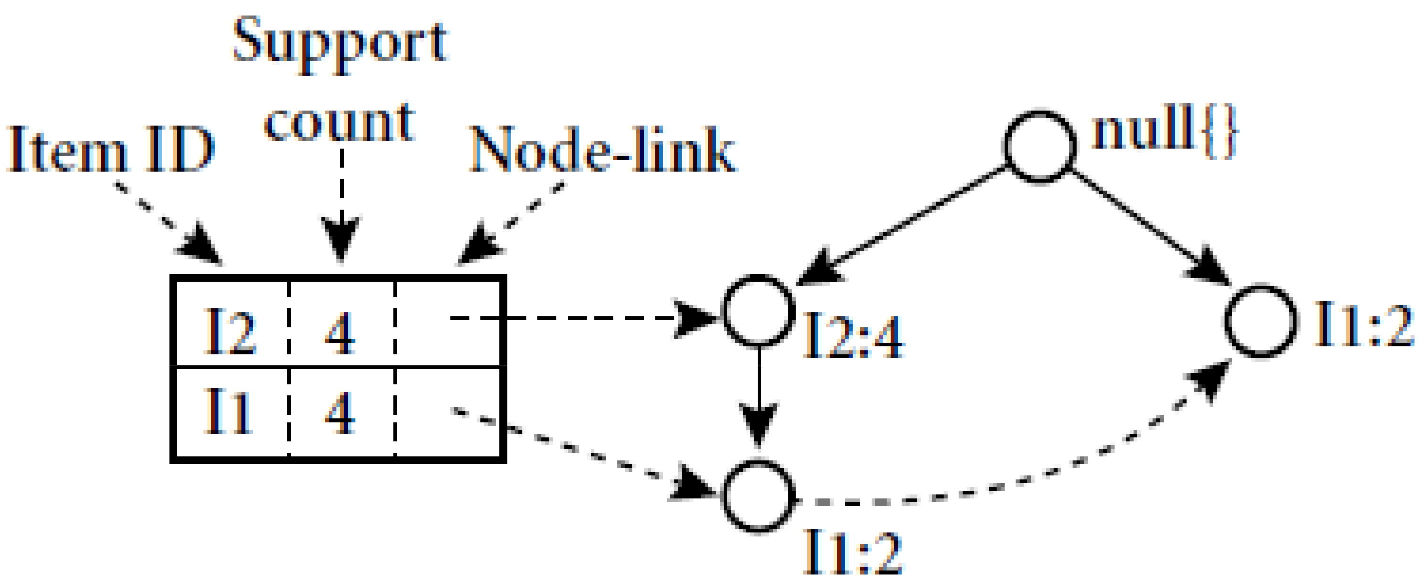
- T100 – {I2, I1, I5}
- T200 - {I2,I4}
- T300 - {I2, I3}
- T400 – {I2,I1,I4}
- T500 – {I1,I3}
- T600 – {I2,I3}
- T700 - {I1,I3}
- T800 – {I2,I1,I3,I5}
- T900 – {I2,I1,I3}



An FP-tree registers compressed, frequent pattern information.

Mining the FP-tree by creating conditional (sub-)pattern bases.

Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}



The conditional FP-tree associated with the conditional node I3.

Algorithm: FP_growth. Mine frequent itemsets using an FP-tree by pattern fragment growth.

Input:

- D, a transaction database;
- *min_sup*, the minimum support count threshold.

Output: The complete set of frequent patterns.

Method:

1. The FP-tree is constructed in the following steps:
   (a) Scan the transaction database D once. Collect F, the set of frequent items, and their support counts. Sort F in support count descending order as L, the *list* of frequent items.
   (b) Create the root of an FP-tree, and label it as "null." For each transaction *Trans* in D do the following.
   Select and sort the frequent items in *Trans* according to the order of L. Let the sorted frequent item list in *Trans* be $[p|P]$, where p is the first element and P is the remaining list. Call insert_tree($[p|P]$, T), which is performed as follows. If T has a child N such that $N.item\text{-}name = p.item\text{-}name$, then increment N's count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link to the nodes with the same *item-name* via the node-link structure. If P is nonempty, call insert_tree(P, N) recursively.
2. The FP-tree is mined by calling FP_growth(*FP_tree*, *null*), which is implemented as follows.

procedure FP_growth($Tree$, α)

if $Tree$ contains a single path P then

  for each combination (denoted as β) of the nodes in the path P

    generate pattern $\beta \cup \alpha$ with $support\_count = minimum\ support\ count\ of\ nodes\ in\ \beta$;

else for each a_i in the header of $Tree$ {

  generate pattern $\beta = a_i \cup \alpha$ with $support\_count = a_i.support\_count$;

  construct β's conditional pattern base and then β's conditional FP_tree $Tree_\beta$;

  if $Tree_\beta \neq \emptyset$ then

    call FP_growth($Tree_\beta$, β); }

Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)
  - For Connect-4 DB, compression ratio could be over 100

Partition Patterns and Databases

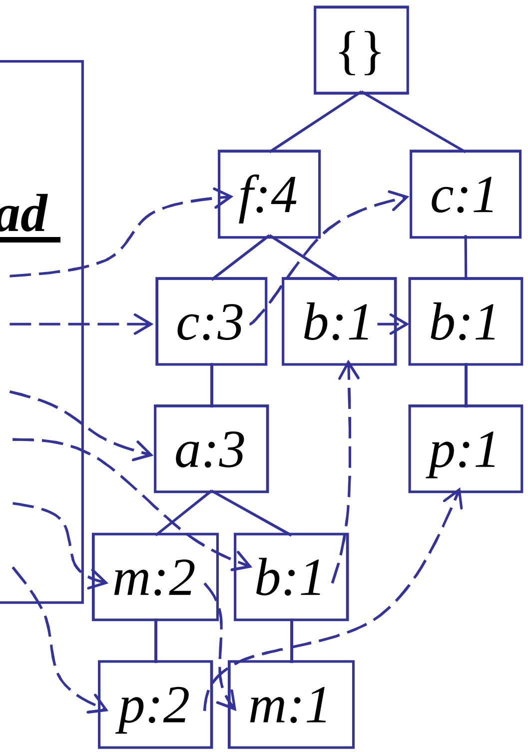
- Frequent patterns can be partitioned into subsets according to f-list
  - F-list=f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - …
  - Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundancy

Find Patterns Having P From P-conditional Database

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item *p*
- Accumulate all of *transformed prefix paths* of item *p* to form *p*'s conditional pattern base

Header Table

Item	frequency	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

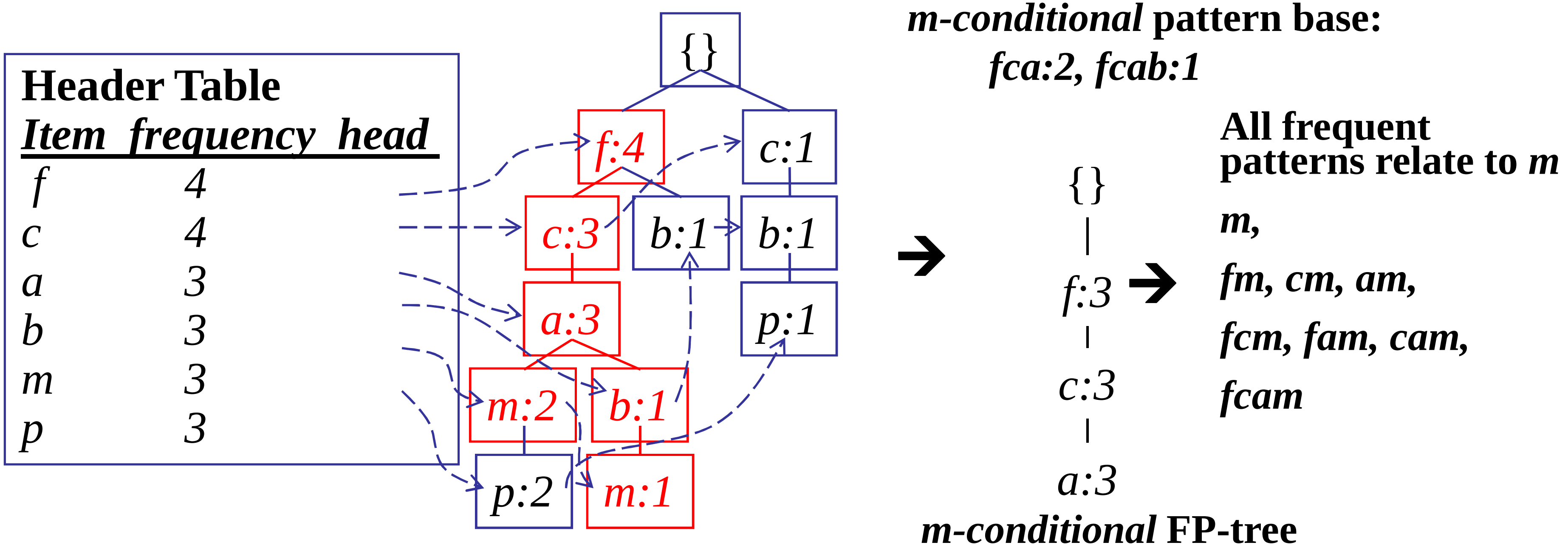


***Conditional* pattern bases**

item	*cond. pattern base*
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

From Conditional Pattern-bases to Conditional FP-trees

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base



Recursion: Mining Each Conditional FP-tree

{}
|
f:3
|
c:3
|
a:3

***m-conditional* FP-tree**

Cond. pattern base of "am": (fc:3)

{}
|
f:3
|
c:3

***am-conditional* FP-tree**

Cond. pattern base of "cm": (f:3)

{}
|
f:3

***cm-conditional* FP-tree**

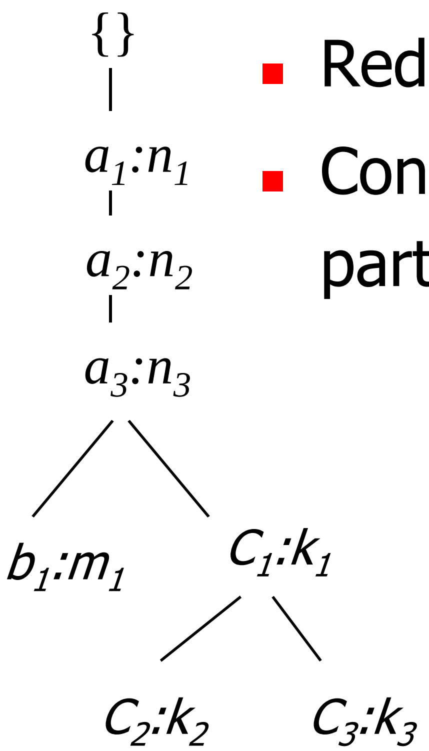
Cond. pattern base of "cam": (f:3)

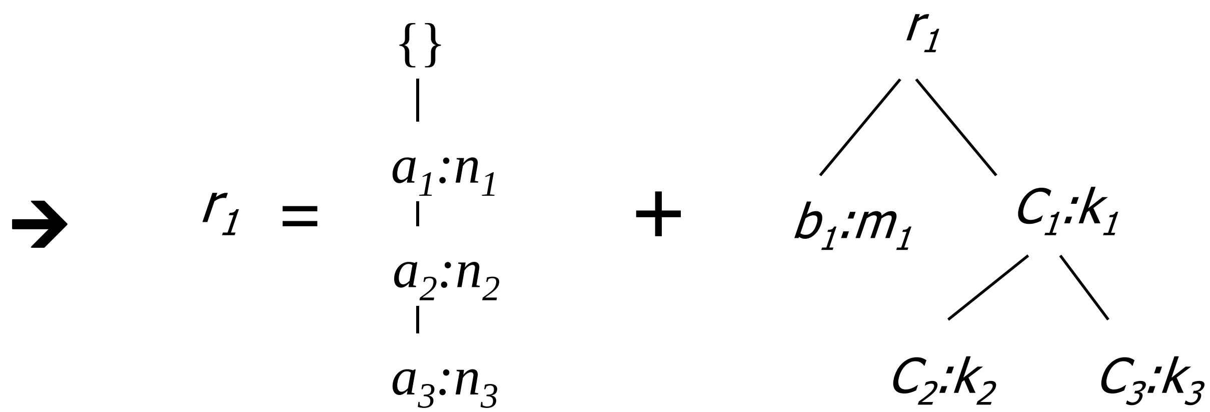
{}
|
f:3

***cam-conditional* FP-tree**

A Special Case: Single Prefix Path in FP-tree

- Suppose a (conditional) FP-tree T has a shared single prefix-path P
- Mining can be decomposed into two parts
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts





Mining Frequent Patterns With FP-trees

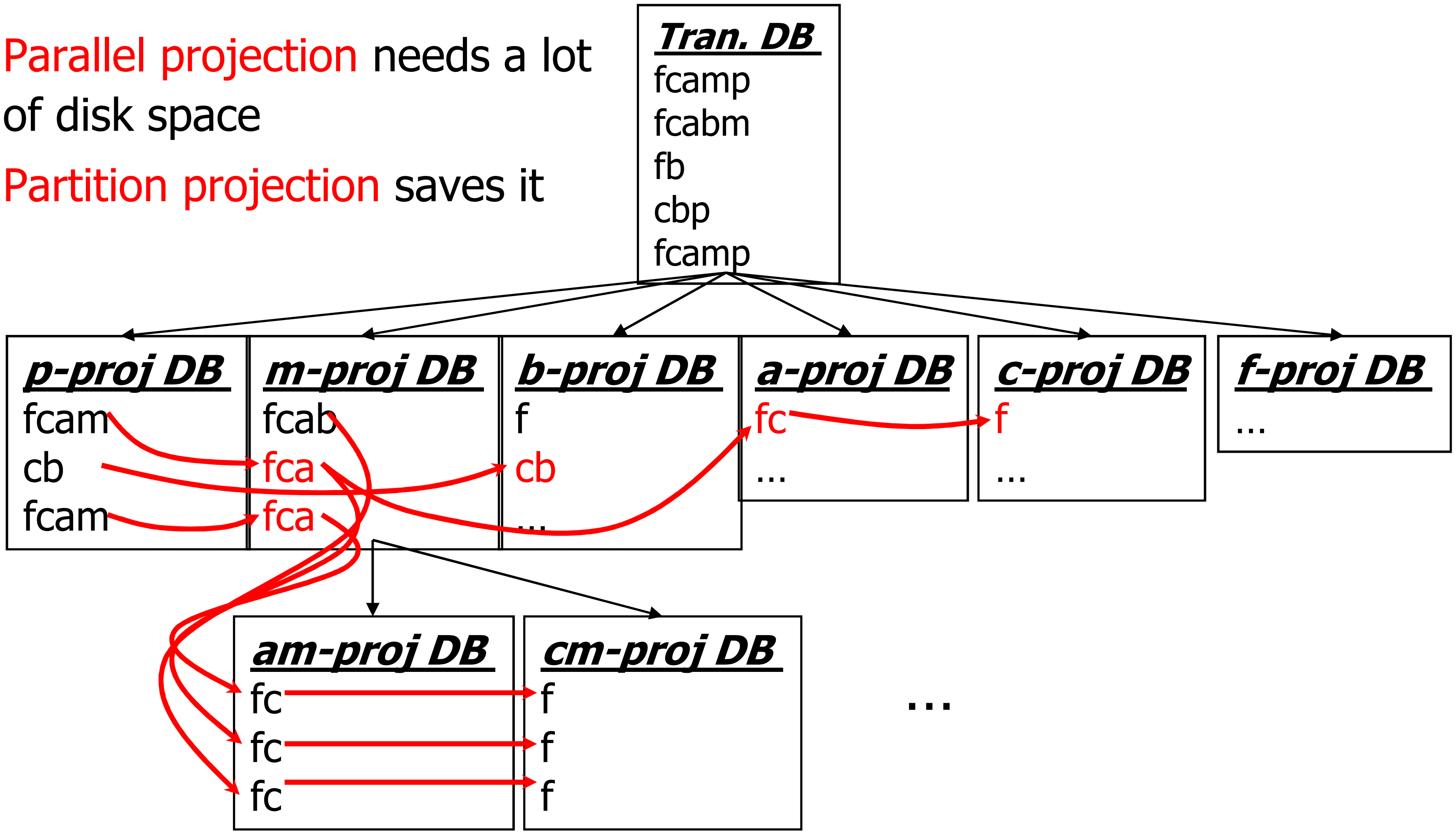
- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

Scaling FP-growth by DB Projection

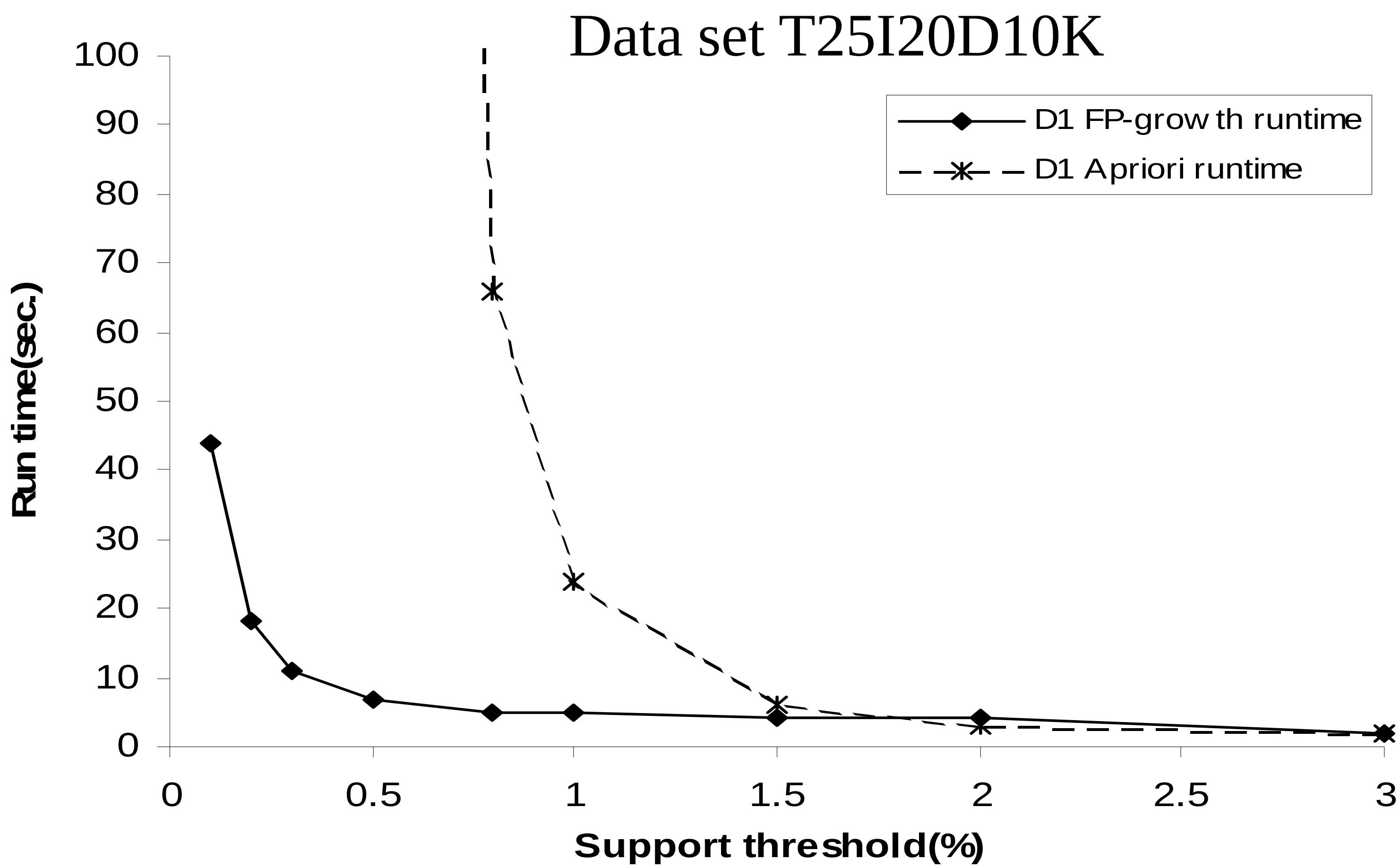
- FP-tree cannot fit in memory?—DB projection
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB
- Parallel projection vs. Partition projection techniques
  - Parallel projection is space costly

Partition-based Projection

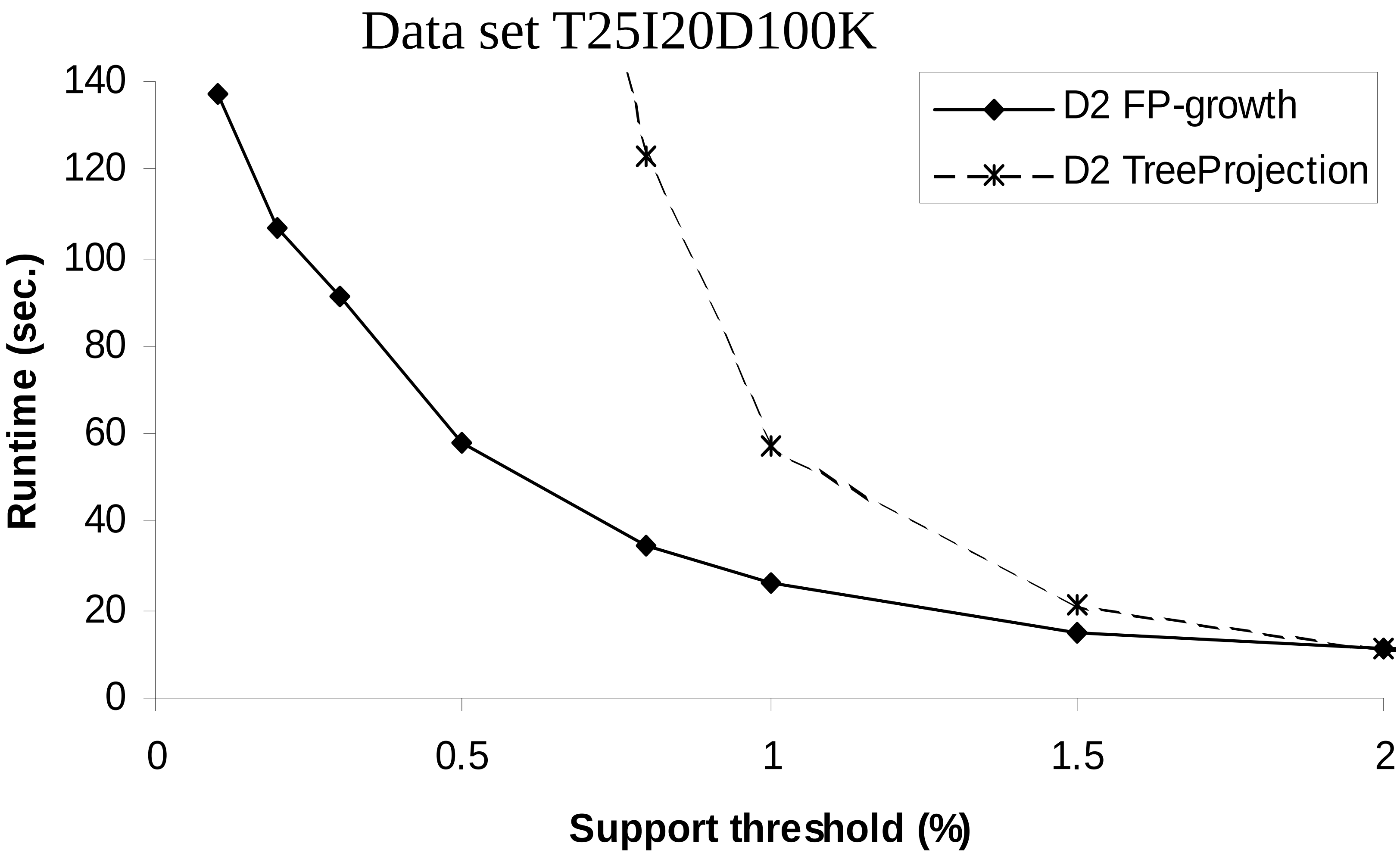
- Parallel projection needs a lot of disk space
- Partition projection saves it



FP-Growth vs. Apriori: Scalability With the Support Threshold



FP-Growth vs. Tree-Projection: Scalability with the Support Threshold



Why Is FP-Growth the Winner?

- Divide-and-conquer:
  - decompose both the mining task and DB according to the frequent patterns obtained so far
  - leads to focused search of smaller databases
- Other factors
  - no candidate generation, no candidate test
  - compressed database: FP-tree structure
  - no repeated scan of entire database
  - basic ops—counting local freq items and building sub FP-tree, no pattern search and matching

Implications of the Methodology

- Mining closed frequent itemsets and max-patterns
  - CLOSET (DMKD'00), CLOSET+,MAXMINER
- Mining sequential patterns
  - FreeSpan (KDD'00), PrefixSpan (ICDE'01)
- Constraint-based mining of frequent patterns
  - Convertible constraints (KDD'00, ICDE'01)
- Computing iceberg data cubes with complex measures
  - H-tree and H-cubing algorithm (SIGMOD'01)

CHARM: Mining by Exploring Vertical Data Format

- Horizontal format : {*TID : itemset*}
- Vertical format: f{*item : TID set*}
  - $t(AB) = \{T_{11}, T_{25}, \ldots\}$
  - tid-list: list of trans.-ids containing an itemset
- Deriving closed patterns based on vertical intersections
  - $t(X) = t(Y)$: X and Y always happen together
  - $t(X) \subset t(Y)$: transaction having X always has Y
- Using diffset to accelerate mining
  - Only keep track of differences of tids
  - $t(X) = \{T_1, T_2, T_3\}$, $t(XY) = \{T_1, T_3\}$
  - Diffset $(XY, X) = \{T_2\}$
- Eclat/MaxEclat (Equivalent CLAss Transformation) (Zaki et al. @KDD'97), VIPER (*Vertical Itemset* Partitioning for Efficient Rule-extraction) (P. Shenoy et al.@SIGMOD'00), CHARM (Closed Association Rule Mining) (Zaki & Hsiao@SDM'02)

Scan DB once

The vertical data format of the transaction data set *D* of Table 5.1.

itemset	*TID_set*
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

The 2-itemsets in vertical data format.

itemset	*TID_set*
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

The 3-itemsets in vertical data format.

itemset	*TID_set*
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

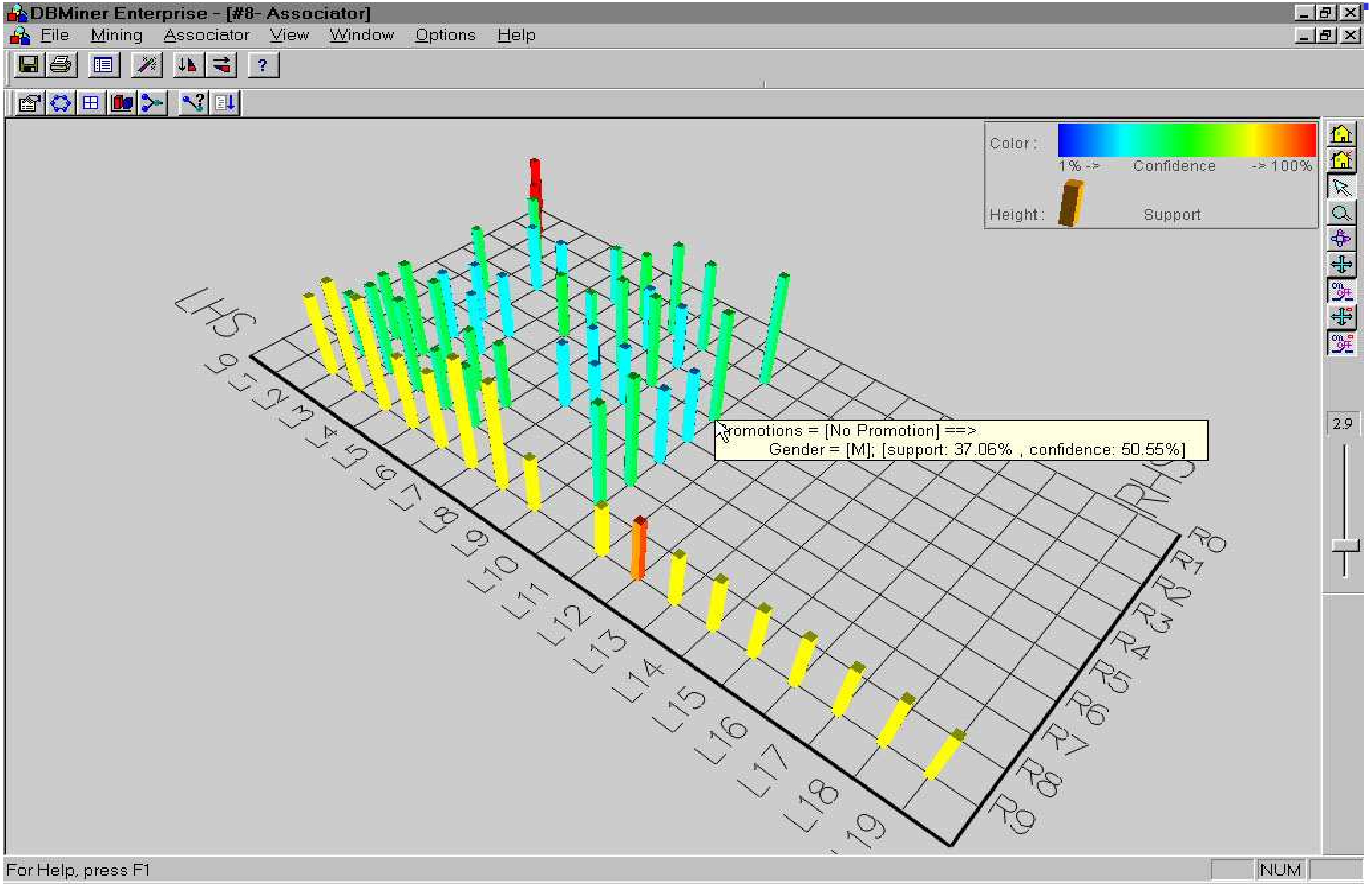
-
- {I3,I4}
- {I4,I5}

* uses Apriori property
* no need to scan DB to find support of (K+1) itemset
* can use diffset

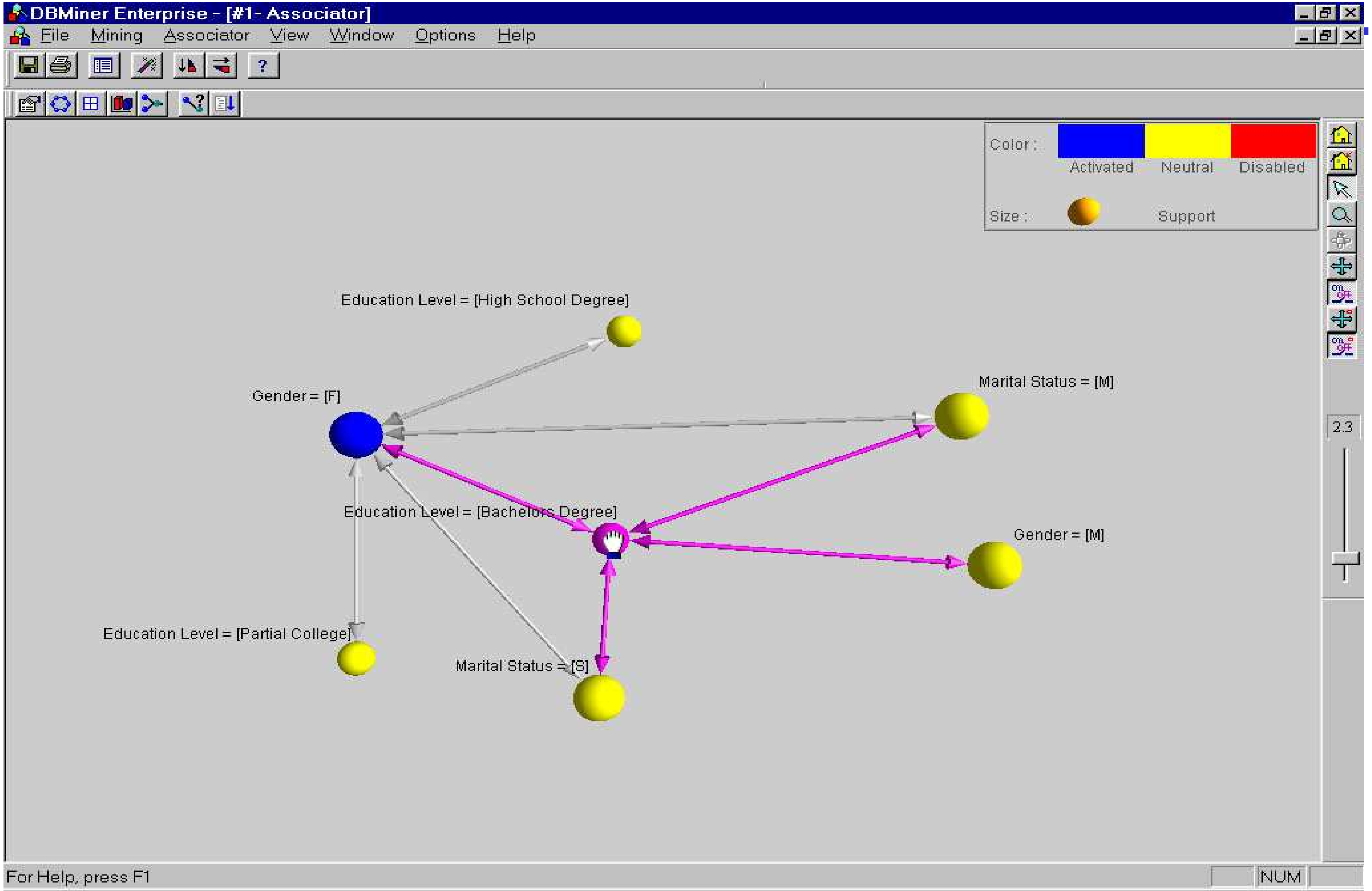
Further Improvements of Mining Methods

- AFOPT (Liu, et al. @ KDD'03)
  - A "push-right" method for mining condensed frequent pattern (CFP) tree
- Carpenter (Pan, et al. @ KDD'03)
  - Mine data sets with small rows but numerous columns
  - Construct a row-enumeration tree for efficient mining

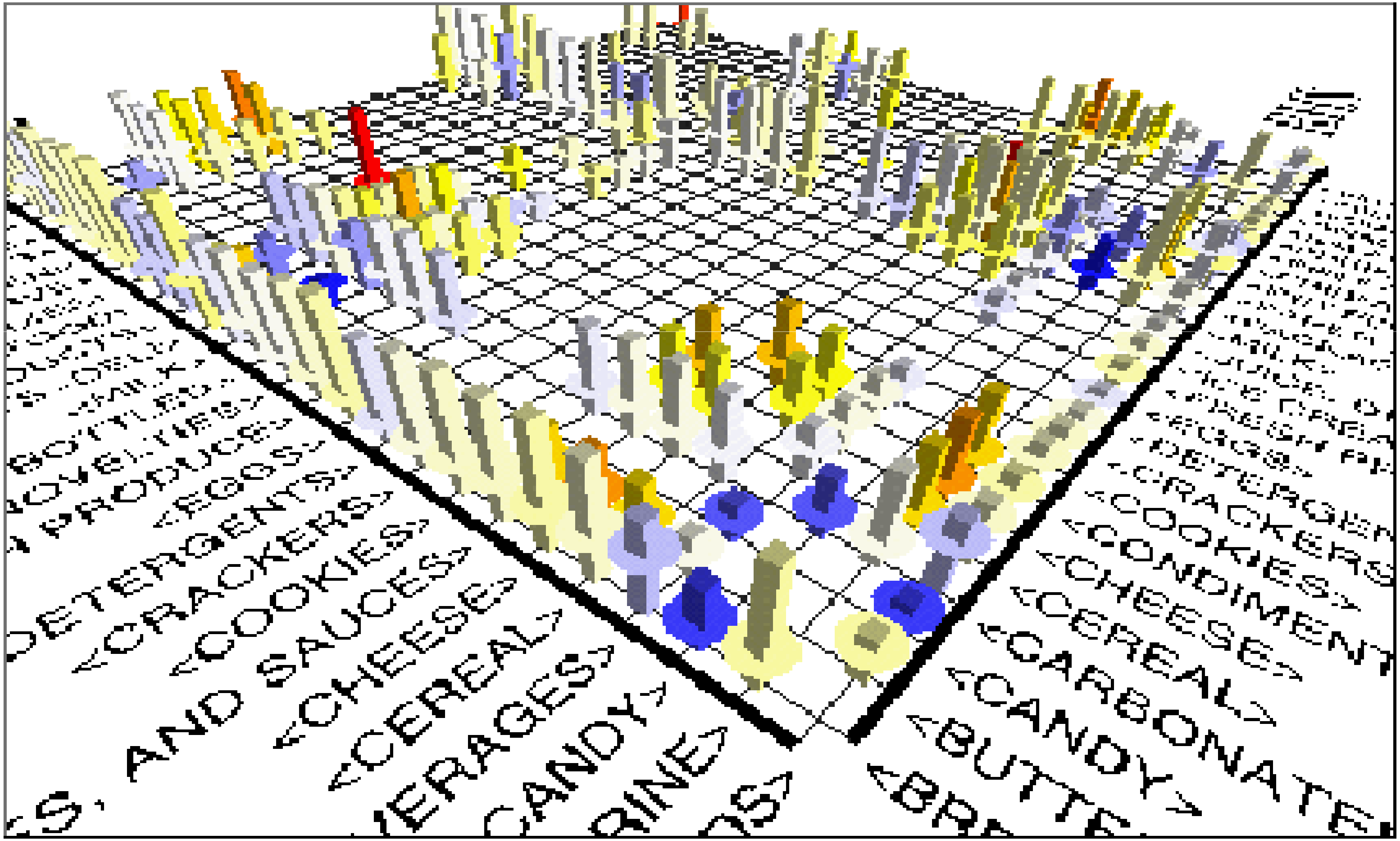
Visualization of Association Rules: Plane Graph



Visualization of Association Rules: Rule Graph



Visualization of Association Rules (SGI/MineSet 3.0)



Chapter 5: Mining Frequent Patterns, Association and Correlations

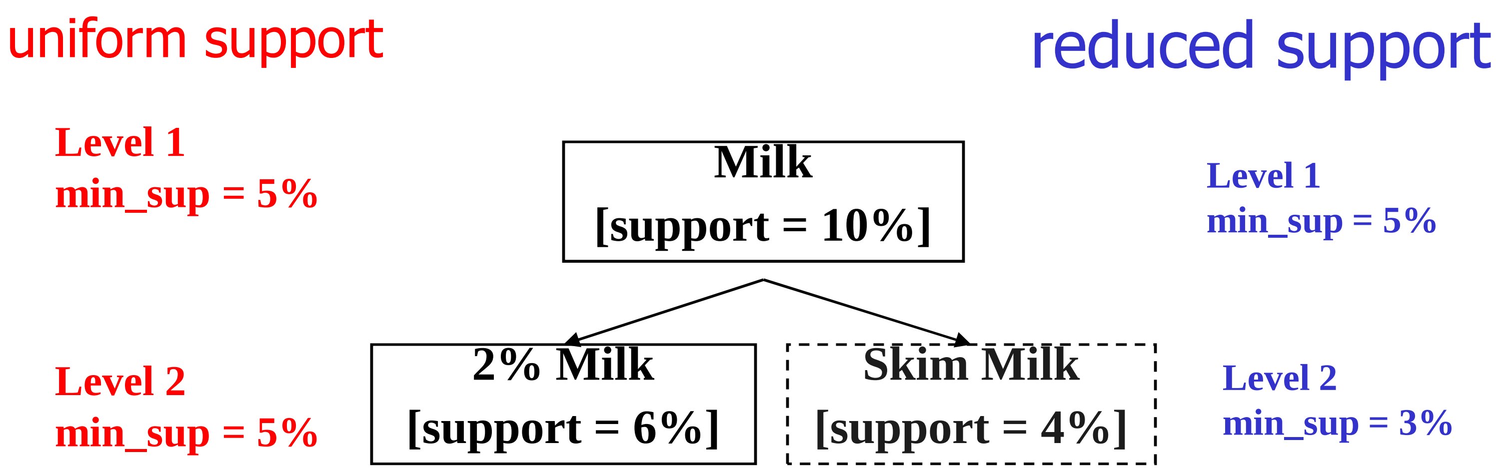
- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

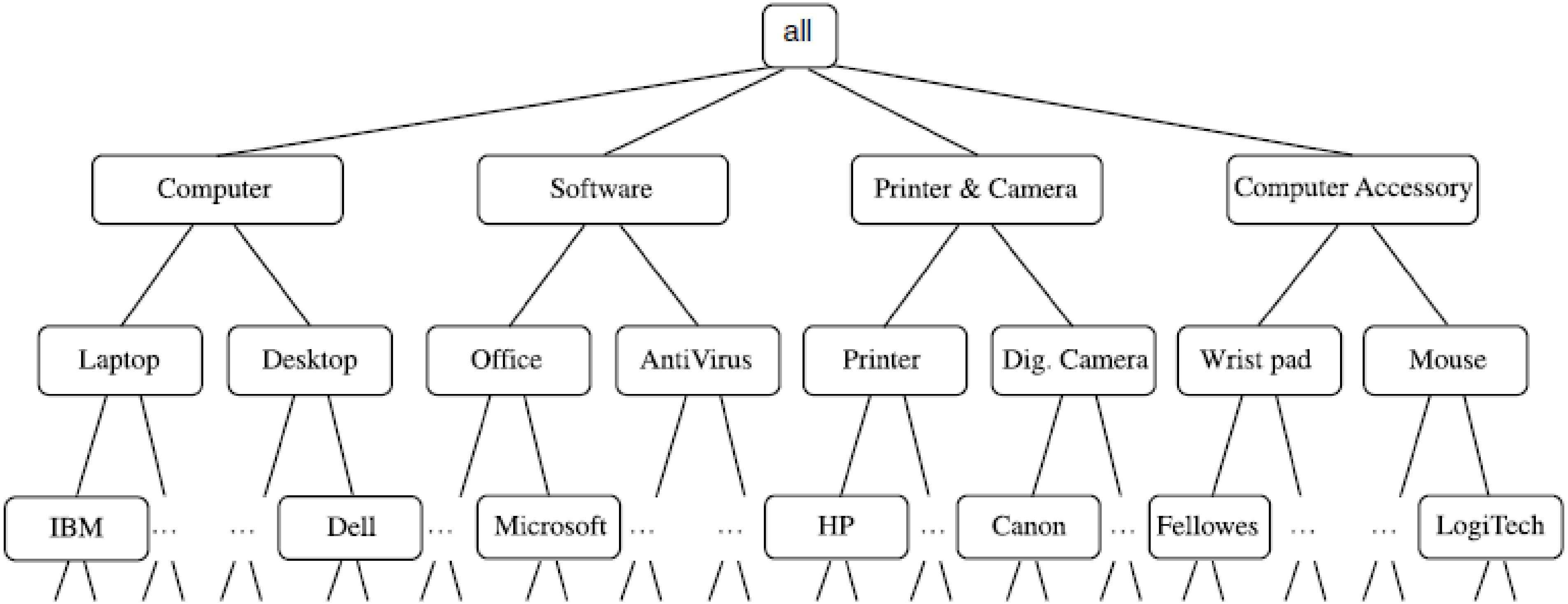
Mining Various Kinds of Association Rules

- Mining multilevel association - abstraction
- Mining multidimensional association – 2 or more predicates or attributes
- Mining quantitative association – numeric attributes with implicit ordering (age)
- Mining interesting correlation patterns

Mining Multiple-Level Association Rules

- Items often form hierarchies
- Flexible support settings
  - Items at the lower level are expected to have lower support
- Exploration of *shared* multi-level mining (Agrawal & Srikant@VLB'95, Han & Fu@VLDB'95)
- Uniform, reduced, group based support





A concept hierarchy for *AllElectronics* computer items.

- Categorical attributes – finite no. of possible values (Nominal, Ordinal) – implicit concept hierarchy
- Numeric or quantitative – implicit ordereing among values – concept hierarchy by discretization, clustering

Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to "ancestor" relationships between items.
- Example
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]
- We say the first rule is an ancestor of the second rule.
- A rule is redundant if its support is close to the "expected" value, based on the rule's ancestor.

Mining Multi-Dimensional Association

- Single-dimensional rules:intra dimensional

  buys(X, "milk") ⇒ buys(X, "bread")

- Multi-dimensional rules: ≥ 2 dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)

    age(X,"19-25") ∧ occupation(X,"student") ⇒ buys(X, "coke")

  - hybrid-dimension assoc. rules (*repeated predicates*)

    age(X,"19-25") ∧ buys(X, "popcorn") ⇒ buys(X, "coke")

- Categorical Attributes: finite number of possible values, no ordering among values—data cube approach (occupation, color, brand)
- Quantitative Attributes: numeric, implicit ordering among values—discretization, clustering, and gradient approaches (age, income, price)
- Finding frequent predicate sets instead of frequent item sets

Mining Quantitative Associations

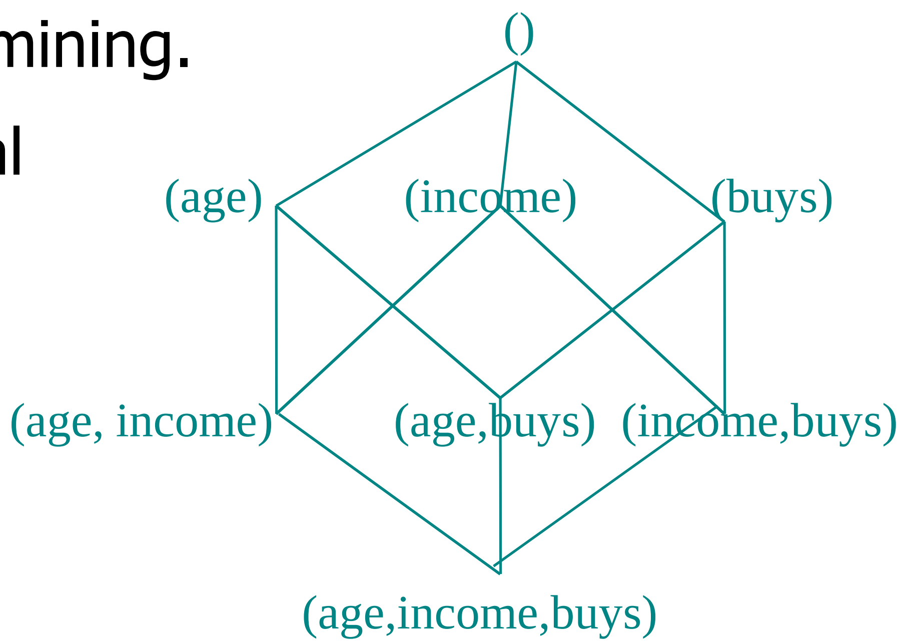
- Techniques can be categorized by how numerical attributes, such as age or salary are treated

1. Static discretization based on predefined concept hierarchies (data cube methods)
2. Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)
3. Clustering: Distance-based association (e.g., Yang & Miller@SIGMOD97)
   - one dimensional clustering then association
4. Deviation: (such as Aumann and Lindell@KDD99)

   Sex = female => Wage: mean=$7/hr (overall mean = $9)

Mining Multi Dimensional AR using Static Discretization of Quantitative Attributes

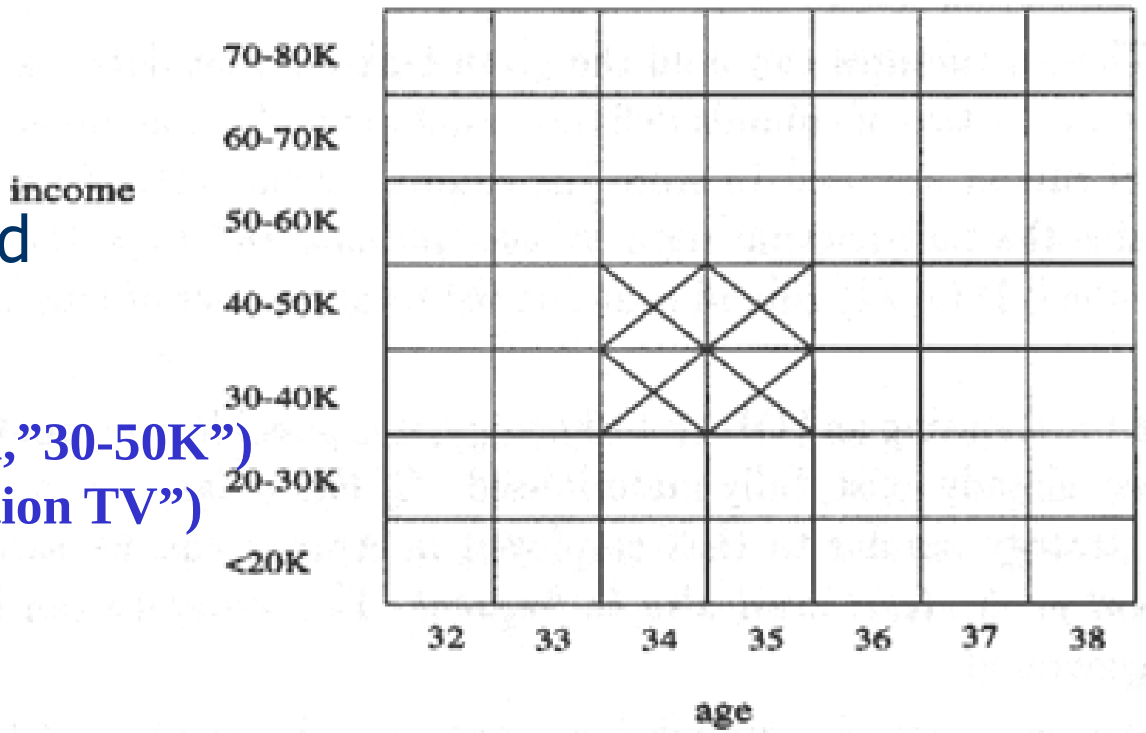
- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- In relational database, finding all frequent k-predicate sets will require *k* or *k*+1 table scans.
- Data cube is well suited for mining.
- The cells of an n-dimensional cuboid correspond to the predicate sets.
- Mining from data cubes can be much faster.



Quantitative Association Rules

- Proposed by Lent, Swami and Widom ICDE'97
- Numeric attributes are *dynamically* discretized
  - Such that the confidence or compactness of the rules mined is maximized
- 2-D quantitative association rules: $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
- Cluster *adjacent* association rules to form general rules using a 2-D grid
- Example

age(X,"34-35") $\wedge$ income(X,"30-50K") $\Rightarrow$ buys(X,"high resolution TV")



Mining Other Interesting Patterns

- Flexible support constraints (Wang et al. @ VLDB'02)
  - Some items (e.g., diamond) may occur rarely but are valuable
  - Customized sup_{min} specification and application
- Top-K closed frequent patterns (Han, et al. @ ICDM'02)
  - Hard to specify sup_{min}, but top-k with $length_{min}$ is more desirable
  - Dynamically raise sup_{min} in FP-tree construction and mining, and select most promising path to mine

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Interestingness Measure: Correlations (Lift)

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%.
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: lift

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

$$lift(B,C) = \frac{2000/5000}{3000/5000*3750/5000} = 0.89$$

$$lift(B,\neg C) = \frac{1000/5000}{3000/5000*1250/5000} = 1.33$$

Are *lift* and χ^2 Good Measures of Correlation?

- "*Buy walnuts* $\Rightarrow$ *buy milk* [1%, 80%]" is misleading
  - if 85% of customers buy milk
- Support and confidence are not good to represent correlations
- So many interestingness measures? (Tan, Kumar, Sritastava @KDD'02)

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

	Milk	No Milk	Sum (row)
Coffee	m, c	~m, c	c
No Coffee	m, ~c	~m, ~c	~c
Sum(col.)	m	~m	Σ

$$all\_conf = \frac{\sup(X)}{\max\_item\_\sup(X)}$$

$$coh = \frac{\sup(X)}{|universe(X)|}$$

DB	m, c	~m, c	m~c	~m~c	lift	all-conf	coh	χ2
A1	1000	100	100	10,000	9.26	0.91	0.83	9055
A2	100	1000	1000	100,000	8.44	0.09	0.05	670
A3	1000	100	10000	100,000	9.18	0.09	0.09	8172
A4	1000	1000	1000	1000	1	0.5	0.33	0

Which Measures Should Be Used?

- **lift** *and* χ^2 are not good measures for correlations in large transactional DBs
- **all-conf** or **coherence** could be good measures (Omiecinski@TKDE'03)
- Both **all-conf** and **coherence** have the downward closure property
- Efficient algorithms can be derived for mining (Lee et al. @ICDM'03sub)

symbol	measure	range	formula
ϕ	ϕ-coefficient	-1...1	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
Q	Yule's Q	-1 ... 1	$\frac{P(A,B)P(\overline{A},\overline{B})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{A},\overline{B})+P(A,\overline{B})P(\overline{A},B)}$
Y	Yule's Y	-1 ... 1	$\frac{\sqrt{P(A,B)P(\overline{A},\overline{B})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{A},\overline{B})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}}$
k	Cohen's	-1 ... 1	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
PS	Piatetsky-Shapiro's	-0.25 ... 0.25	$P(A,B)-P(A)P(B)$
F	Certainty factor	-1 ... 1	$\max(\frac{P(B\mid A)-P(B)}{1-P(B)}, \frac{P(A\mid B)-P(A)}{1-P(A)})$
AV	added value	-0.5 ... 1	$\max(P(B\mid A)-P(B), P(A\mid B)-P(A))$
K	Klosgen's Q	-0.33 ... 0.38	$\sqrt{P(A,B)}\max(P(B\mid A)-P(B), P(A\mid B)-P(A))$
g	Goodman-kruskal's	0 ... 1	$\frac{\Sigma_j \max_k P(A_j,B_k)+\Sigma_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-max_k P(B_k)}$
M	Mutual Information	0 ... 1	$\frac{\Sigma_i \Sigma_j P(A_i,B_j) \log \frac{P(A_i,B_j)}{P(A_i)P(B_J)}}{\min(-\Sigma_i P(A_i) \log P(A_i) \log P(A_i), -\Sigma_i P(B_i) \log P(B_i) \log P(B_i))}$
J	J-Measure	0 ... 1	$\max(P(A,B)\log(\frac{P(B\mid A)}{P(B)}) + P(A\overline{B})\log(\frac{P(\overline{B}\mid A)}{P(\overline{B})}))$ $P(A,B)\log(\frac{P(A\mid B)}{P(A)}) + P(\overline{A}B)\log(\frac{P(\overline{A}\mid B)}{P(\overline{A})})$
G	Gini index	0 ... 1	$\max(P(A)[P(B\mid A)^2 + P(\overline{B}\mid A)^2] + P(\overline{A}[P(B\mid \overline{A})^2 + P(\overline{B}\mid \overline{A})^2] - P(B)^2 - P(\overline{B})^2,$ $P(B)[P(A\mid B)^2 + P(\overline{A}\mid B)^2] + P(\overline{B}[P(A\mid \overline{B})^2 + P(\overline{A}\mid \overline{B})^2] - P(A)^2 - P(\overline{A})^2)$
s	support	0 ... 1	$P(A,B)$
c	confidence	0 ... 1	$max(P(B\mid A), P(A\mid B))$
L	Laplace	0 ... 1	$\max(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2})$
IS	Cosine	0 ... 1	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
γ	coherence(Jaccard)	0 ... 1	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
α	all_confidence	0 ... 1	$\frac{P(A,B)}{\max(P(A),P(B))}$
o	odds ratio	0 ... ∞	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(\overline{A},B)P(A,\overline{B})}$
V	Conviction	0.5 ... ∞	$\max(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})})$
λ	lift	0 ... ∞	$\frac{P(A,B)}{P(A)P(B)}$
S	Collective strength	0 ... ∞	$\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{AB})}$
χ^2	χ^2	0 ... ∞	$\Sigma_i \frac{(P(A_i)-E_i)^2}{E_i}$

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Constraint-based (Query-Directed) Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - System optimization: explores such constraints for efficient mining—constraint-based mining

Constraints in Data Mining

- Knowledge type constraint:
  - classification, association, etc.
- Data constraint — using SQL-like queries
  - find product pairs sold together in stores in Chicago in Dec.'02
- Dimension/level constraint
  - in relevance to region, price, brand, customer category
- Rule (or pattern) constraint
  - small sales (price $< \$10$) triggers big sales (sum $> \$200$)
- Interestingness constraint
  - strong rules: min_support $\geq$ 3%, min_confidence $\geq$ 60%

Constrained Mining vs. Constraint-Based Search

- Constrained mining vs. constraint-based search/reasoning
  - Both are aimed at reducing search space
  - Finding all patterns satisfying constraints vs. finding some (or one) answer in constraint-based search in AI
  - Constraint-pushing vs. heuristic search
  - It is an interesting research problem on how to integrate them
- Constrained mining vs. query processing in DBMS
  - Database query processing requires to find all
  - Constrained pattern mining shares a similar philosophy as pushing selections deeply in query processing

Anti-Monotonicity in Constraint Pushing

- Anti-monotonicity
  - *When an intemset S* ***violates*** *the constraint, so does any of its superset*
  - *sum(S.Price)* $\leq$ *v* is anti-monotone
  - *sum(S.Price)* $\geq$ *v* is not anti-monotone
- Example. C: range(S.profit) $\leq$ 15 is anti-monotone
  - Itemset *ab* violates C
  - So does every superset of *ab*

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Monotonicity for Constraint Pushing

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

- Monotonicity
  - *When an intemset S* ***satisfies*** *the constraint, so does any of its superset*
  - *sum(S.Price)* $\geq v$ is monotone
  - *min(S.Price)* $\leq v$ is monotone
- Example. C: range(S.profit) ≥ 15
  - Itemset *ab* satisfies C
  - So does every superset of *ab*

Succinctness

- Succinctness:
  - Given A_1, the set of items satisfying a succinctness constraint C, then any set S satisfying C is based on A_1 , i.e., S contains a subset belonging to A_1
  - Idea: Without looking at the transaction database, whether an itemset S satisfies constraint C can be determined based on the selection of items
  - $min(S.Price) \le v$ is succinct
  - $sum(S.Price) \ge v$ is not succinct
- Optimization: If C is succinct, C is pre-counting pushable

The Apriori Algorithm — Example

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2

Naïve Algorithm: Apriori + Constraint

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

Sum{S.price} < 5

The Constrained Apriori Algorithm: Push an Anti-monotone Constraint Deep

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
~~{5}~~	~~3~~

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
~~{5}~~	~~3~~

C_2

itemset
{1 2}
{1 3}
~~{1 5}~~
{2 3}
~~{2 5}~~
~~{3 5}~~

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
~~{1 5}~~	~~1~~
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
~~{2 3 5}~~

Scan D

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

Sum{S.price} < 5

The Constrained Apriori Algorithm: Push a Succinct Constraint Deep

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
~~{2 3}~~
~~{2 5}~~
~~{3 5}~~

not immediately to be used

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
~~{2 3 5}~~

Scan D

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

min{S.price } <= 1

Converting "Tough" Constraints

- Convert tough constraints into anti-monotone or monotone by properly ordering items
- Examine C: avg(*S*.profit) ≥ 25
  - Order items in value-descending order
    - *<a, f, g, d, b, h, c, e>*
  - If an itemset *afb* violates C
    - So does *afbh, afb**
    - It becomes anti-monotone!

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Strongly Convertible Constraints

- avg(X) ≥ 25 is convertible anti-monotone w.r.t. item value descending order R: <*a, f, g, d, b, h, c, e*>
  - If an itemset *af* violates a constraint C, so does every itemset with *af* as prefix, such as *afd*
- avg(X) ≥ 25 is convertible monotone w.r.t. item value ascending order R^{-1}: <*e, c, h, b, d, g, f, a*>
  - If an itemset *d* satisfies a constraint *C*, so does itemsets *df* and *dfa*, which having *d* as a prefix
- Thus, avg(X) ≥ 25 is strongly convertible

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Can Apriori Handle Convertible Constraint?

- A convertible, not monotone nor anti-monotone nor succinct constraint cannot be pushed deep into the an Apriori mining algorithm
  - Within the level wise framework, no direct pruning based on the constraint can be made
  - Itemset df violates constraint C: avg(X)>=25
  - Since adf satisfies C, Apriori needs df to assemble adf, df cannot be pruned
- But it can be pushed into frequent-pattern growth framework!

Item	**Value**
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

Mining With Convertible Constraints

- C: avg(X) >= 25, min_sup=2
- List items in every transaction in value descending order R: <a, f, g, d, b, h, c, e>
  - C is convertible anti-monotone w.r.t. R
- Scan TDB once
  - remove infrequent items
    - Item h is dropped
  - Itemsets a and f are good, …
- Projection-based mining
  - Imposing an appropriate order on item projection
  - Many tough constraints can be converted into (anti)-monotone

Item	Value
a	40
f	30
g	20
d	10
b	0
h	-10
c	-20
e	-30

TDB (min_sup=2)

TID	Transaction
10	a, f, d, b, c
20	f, g, d, b, c
30	a, f, d, c, e
40	f, g, h, c, e

Handling Multiple Constraints

- Different constraints may require different or even conflicting item-ordering
- If there exists an order *R* s.t. both C_1 and C_2 are convertible w.r.t. *R*, then there is no conflict between the two convertible constraints
- If there exists conflict on order of items
  - Try to satisfy one constraint first
  - Then using the order for the other constraint to mine frequent itemsets in the corresponding projected database

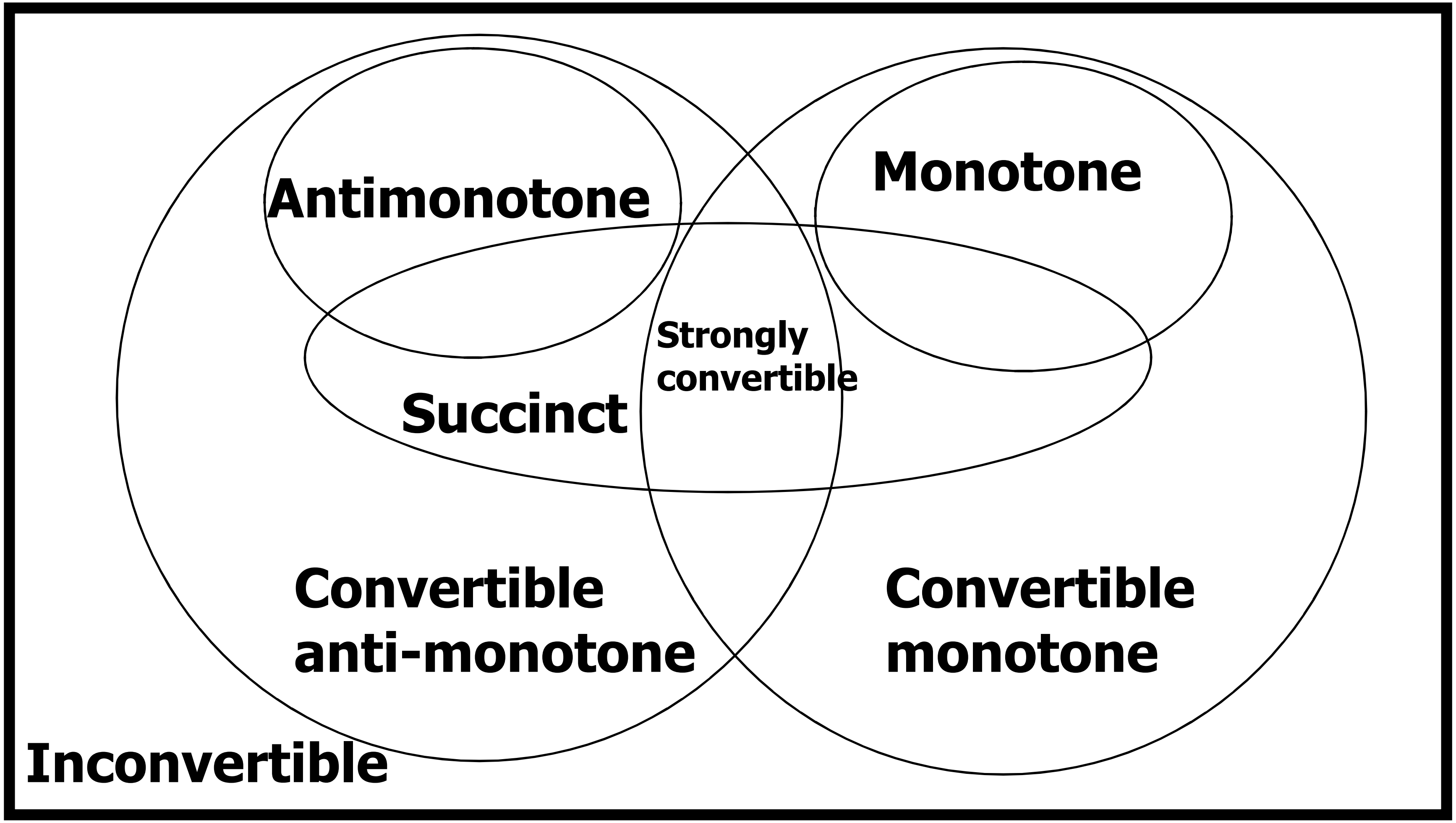
What Constraints Are Convertible?

Constraint	Convertible anti-monotone	Convertible monotone	Strongly convertible
avg(S) ≤ , ≥ v	Yes	Yes	Yes
median(S) ≤ , ≥ v	Yes	Yes	Yes
sum(S) ≤ v (items could be of any value, v ≥ 0)	Yes	No	No
sum(S) ≤ v (items could be of any value, v ≤ 0)	No	Yes	No
sum(S) ≥ v (items could be of any value, v ≥ 0)	No	Yes	No
sum(S) ≥ v (items could be of any value, v ≤ 0)	Yes	No	No
……			

Constraint-Based Mining—A General Picture

Constraint	Antimonotone	Monotone	Succinct
$v \in S$	no	yes	yes
$S \supseteq V$	no	yes	yes
$S \subseteq V$	yes	no	yes
$\min(S) \leq v$	no	yes	yes
$\min(S) \geq v$	yes	no	yes
$\max(S) \leq v$	yes	no	yes
$\max(S) \geq v$	no	yes	yes
$count(S) \leq v$	yes	no	weakly
$count(S) \geq v$	no	yes	weakly
$sum(S) \leq v$ ($a \in S, a \geq 0$)	yes	no	no
$sum(S) \geq v$ ($a \in S, a \geq 0$)	no	yes	no
$range(S) \leq v$	yes	no	no
$range(S) \geq v$	no	yes	no
$avg(S)\ \theta\ v, \theta \in \{ =, \leq, \geq \}$	convertible	convertible	no
$support(S) \geq \xi$	yes	no	no
$support(S) \leq \xi$	no	yes	no

A Classification of Constraints



Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth, CLOSET+, ...)
  - Vertical format approach (CHARM, ...)
- Mining a variety of rules and interesting patterns
- Constraint-based mining
- Mining sequential and structured patterns
- Extensions and applications

Frequent-Pattern Mining: Research Problems

- Mining fault-tolerant frequent, sequential and structured patterns
  - Patterns allows limited faults (insertion, deletion, mutation)
- Mining truly interesting patterns
  - Surprising, novel, concise, …
- Application exploration
  - E.g., DNA sequence analysis and bio-pattern classification
  - "Invisible" data mining

Ref: Basic Concepts of Frequent Pattern Mining

- (Association Rules) R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. SIGMOD'93.
- (Max-pattern) R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98.
- (Closed-pattern) N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99.
- (Sequential pattern) R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95

Ref: Apriori and Its Improvements

- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'94.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. VLDB'95.
- J. S. Park, M. S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95.
- H. Toivonen. Sampling large databases for association rules. VLDB'96.
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97.
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98.

Ref: Depth-First, Projection-Based FP Mining

- R. Agarwal, C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent itemsets. J. Parallel and Distributed Computing:02.
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD' 00.
- J. Pei, J. Han, and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. DMKD'00.
- J. Liu, Y. Pan, K. Wang, and J. Han. Mining Frequent Item Sets by Opportunistic Projection. KDD'02.
- J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining Top-K Frequent Closed Patterns without Minimum Support. ICDM'02.
- J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. KDD'03.
- G. Liu, H. Lu, W. Lou, J. X. Yu. On Computing, Storing and Querying Frequent Patterns. KDD'03.

Ref: Vertical Format and Row Enumeration Methods

- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel algorithm for discovery of association rules. DAMI:97.
- Zaki and Hsiao. CHARM: An Efficient Algorithm for Closed Itemset Mining, SDM'02.
- C. Bucila, J. Gehrke, D. Kifer, and W. White. DualMiner: A Dual-Pruning Algorithm for Itemsets with Constraints. KDD’02.
- F. Pan, G. Cong, A. K. H. Tung, J. Yang, and M. Zaki , CARPENTER: Finding Closed Patterns in Long Biological Datasets. KDD'03.

Ref: Mining Multi-Level and Quantitative Rules

- R. Srikant and R. Agrawal. Mining generalized association rules. VLDB'95.
- J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. VLDB'95.
- R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. SIGMOD'96.
- T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. SIGMOD'96.
- K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Computing optimized rectilinear regions for association rules. KDD'97.
- R.J. Miller and Y. Yang. Association rules over interval data. SIGMOD'97.
- Y. Aumann and Y. Lindell. A Statistical Theory for Quantitative Association Rules KDD'99.

Ref: Mining Correlations and Interesting Rules

- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94.
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97.
- C. Silverstein, S. Brin, R. Motwani, and J. Ullman. Scalable techniques for mining causal structures. VLDB'98.
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02.
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03.
- Y. K. Lee, W.Y. Kim, Y. D. Cai, and J. Han. CoMine: Efficient Mining of Correlated Patterns. ICDM'03.

Ref: Mining Other Kinds of Rules

- R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. VLDB'96.
- B. Lent, A. Swami, and J. Widom. Clustering association rules. ICDE'97.
- A. Savasere, E. Omiecinski, and S. Navathe. Mining for strong negative associations in a large database of customer transactions. ICDE'98.
- D. Tsur, J. D. Ullman, S. Abitboul, C. Clifton, R. Motwani, and S. Nestorov. Query flocks: A generalization of association-rule mining. SIGMOD'98.
- F. Korn, A. Labrinidis, Y. Kotidis, and C. Faloutsos. Ratio rules: A new paradigm for fast, quantifiable data mining. VLDB'98.
- K. Wang, S. Zhou, J. Han. Profit Mining: From Patterns to Actions. EDBT'02.

Ref: Constraint-Based Pattern Mining

- R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. KDD'97.
- R. Ng, L.V.S. Lakshmanan, J. Han & A. Pang. Exploratory mining and pruning optimizations of constrained association rules. SIGMOD’98.
- M.N. Garofalakis, R. Rastogi, K. Shim: SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. VLDB’99.
- G. Grahne, L. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. ICDE'00.
- J. Pei, J. Han, and L. V. S. Lakshmanan. Mining Frequent Itemsets with Convertible Constraints. ICDE'01.
- J. Pei, J. Han, and W. Wang, Mining Sequential Patterns with Constraints in Large Databases, CIKM'02.

Ref: Mining Sequential and Structured Patterns

- R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. EDBT'96.
- H. Mannila, H Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. DAMI:97.
- M. Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning:01.
- J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. ICDE'01.
- M. Kuramochi and G. Karypis. Frequent Subgraph Discovery. ICDM'01.
- X. Yan, J. Han, and R. Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. SDM'03.
- X. Yan and J. Han. CloseGraph: Mining Closed Frequent Graph Patterns. KDD'03.

Ref: Mining Spatial, Multimedia, and Web Data

- K. Koperski and J. Han, Discovery of Spatial Association Rules in Geographic Information Databases, SSD'95.
- O. R. Zaiane, M. Xin, J. Han, Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. ADL'98.
- O. R. Zaiane, J. Han, and H. Zhu, Mining Recurrent Items in Multimedia with Progressive Resolution Refinement. ICDE'00.
- D. Gunopulos and I. Tsoukatos. Efficient Mining of Spatiotemporal Patterns. SSTD'01.

Ref: Mining Frequent Patterns in Time-Series Data

- B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. ICDE'98.
- J. Han, G. Dong and Y. Yin, Efficient Mining of Partial Periodic Patterns in Time Series Database, ICDE'99.
- H. Lu, L. Feng, and J. Han. Beyond Intra-Transaction Association Analysis: Mining Multi-Dimensional Inter-Transaction Association Rules. TOIS:00.
- B.-K. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online Data Mining for Co-Evolving Time Sequences. ICDE'00.
- W. Wang, J. Yang, R. Muntz. TAR: Temporal Association Rules on Evolving Numerical Attributes. ICDE’01.
- J. Yang, W. Wang, P. S. Yu. Mining Asynchronous Periodic Patterns in Time Series Data. TKDE’03.

Ref: Iceberg Cube and Cube Computation

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB'96.
- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidi-mensional aggregates. SIGMOD'97.
- J. Gray, et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. DAMI: 97.
- M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. VLDB'98.
- S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. EDBT'98.
- K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. SIGMOD'99.

Ref: Iceberg Cube and Cube Exploration

- J. Han, J. Pei, G. Dong, and K. Wang, Computing Iceberg Data Cubes with Complex Measures. SIGMOD' 01.
- W. Wang, H. Lu, J. Feng, and J. X. Yu. Condensed Cube: An Effective Approach to Reducing Data Cube Size. ICDE'02.
- G. Dong, J. Han, J. Lam, J. Pei, and K. Wang. Mining Multi-Dimensional Constrained Gradients in Data Cubes. VLDB'01.
- T. Imielinski, L. Khachiyan, and A. Abdulghani. Cubegrades: Generalizing association rules. DAMI:02.
- L. V. S. Lakshmanan, J. Pei, and J. Han. Quotient Cube: How to Summarize the Semantics of a Data Cube. VLDB'02.
- D. Xin, J. Han, X. Li, B. W. Wah. Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration. VLDB'03.

Ref: FP for Classification and Clustering

- G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. KDD'99.
- B. Liu, W. Hsu, Y. Ma. Integrating Classification and Association Rule Mining. KDD'98.
- W. Li, J. Han, and J. Pei. CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. ICDM'01.
- H. Wang, W. Wang, J. Yang, and P.S. Yu. Clustering by pattern similarity in large data sets. SIGMOD' 02.
- J. Yang and W. Wang. CLUSEQ: efficient and effective sequence clustering. ICDE'03.
- B. Fung, K. Wang, and M. Ester. Large Hierarchical Document Clustering Using Frequent Itemset. SDM'03.
- X. Yin and J. Han. CPAR: Classification based on Predictive Association Rules. SDM'03.

Ref: Stream and Privacy-Preserving FP Mining

- A. Evfimievski, R. Srikant, R. Agrawal, J. Gehrke. Privacy Preserving Mining of Association Rules. KDD'02.
- J. Vaidya and C. Clifton. Privacy Preserving Association Rule Mining in Vertically Partitioned Data. KDD'02.
- G. Manku and R. Motwani. Approximate Frequency Counts over Data Streams. VLDB'02.
- Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-Dimensional Regression Analysis of Time-Series Data Streams. VLDB'02.
- C. Giannella, J. Han, J. Pei, X. Yan and P. S. Yu. Mining Frequent Patterns in Data Streams at Multiple Time Granularities, Next Generation Data Mining:03.
- A. Evfimievski, J. Gehrke, and R. Srikant. Limiting Privacy Breaches in Privacy Preserving Data Mining. PODS'03.

Ref: Other Freq. Pattern Mining Applications

- Y. Huhtala, J. Kärkkäinen, P. Porkka, H. Toivonen. Efficient Discovery of Functional and Approximate Dependencies Using Partitions. ICDE’98.
- H. V. Jagadish, J. Madar, and R. Ng. Semantic Compression and Pattern Extraction with Fascicles. VLDB'99.
- T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk. Mining Database Structure; or How to Build a Data Quality Browser. SIGMOD'02.