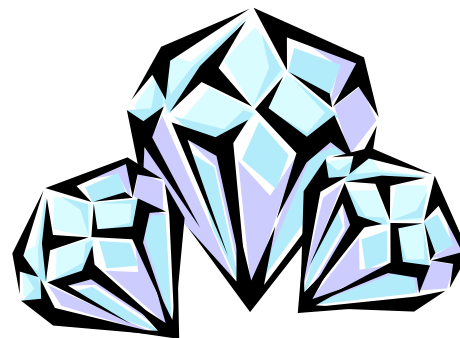


Association Rule Mining (II)

Liaquat Majeed Sheikh

liaquat.majeed@nu.edu.pk

National University of Computer and Emerging Sciences



June 10, 2005

Data Mining: Concepts and
Techniques



Methods to Improve Apriori's Efficiency

- **Transaction reduction:**

A transaction that does not contain any frequent k -itemset is useless in subsequent scans because it can not contain any frequent $(K+1)$ -itemsets. Therefore, such a transaction can be removed from further consideration.

- **Partitioning:**

Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB



Is Apriori Fast Enough? — Performance Bottlenecks

- The core of the Apriori algorithm:
 - Use frequent $(k - 1)$ -itemsets to generate candidate frequent k -itemsets
 - Use database scan and pattern matching to collect counts for the candidate itemsets
- The bottleneck of *Apriori*: candidate generation
 - Huge candidate sets:
 - 10^4 frequent 1-itemset will generate 10^7 candidate 2-itemsets
 - To discover a frequent pattern of size 100, e.g., $\{a_1, a_2, \dots, a_{100}\}$, one needs to generate $2^{100} \approx 10^{30}$ candidates.
 - Multiple scans of database:
 - Needs $(n + 1)$ scans, n is the length of the longest pattern



Multilevel Association Rules

- ❑ It is difficult to find interesting patterns at a too primitive level
 - high support = too few rules
 - low support = too many rules, most uninteresting

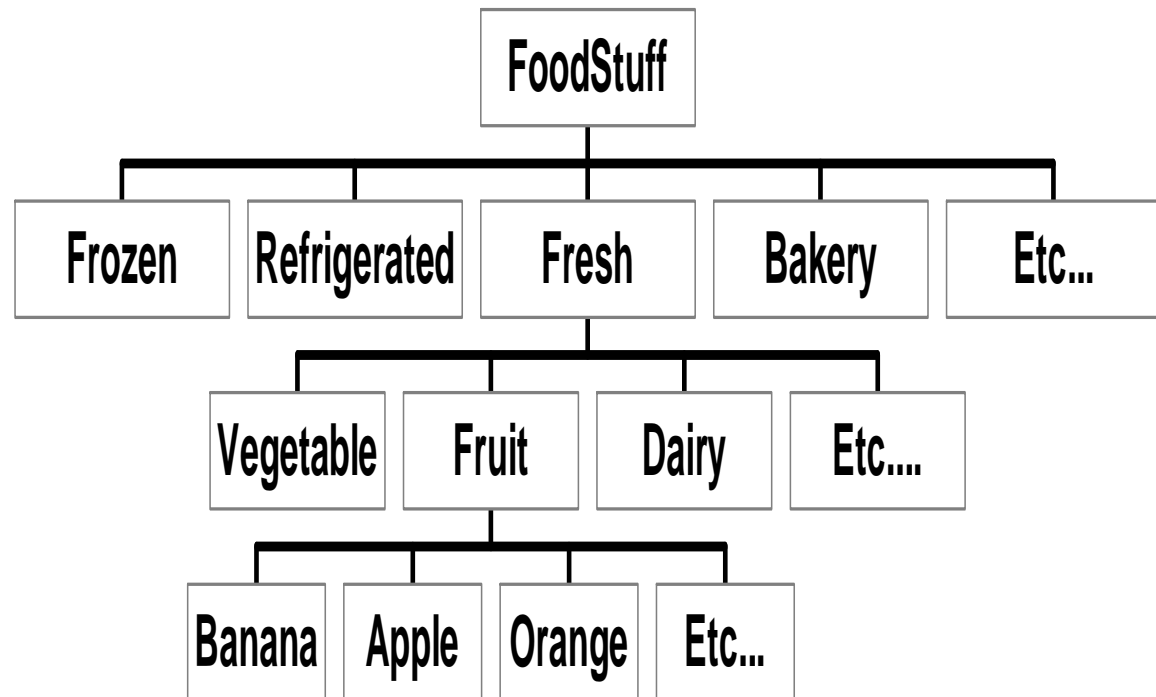
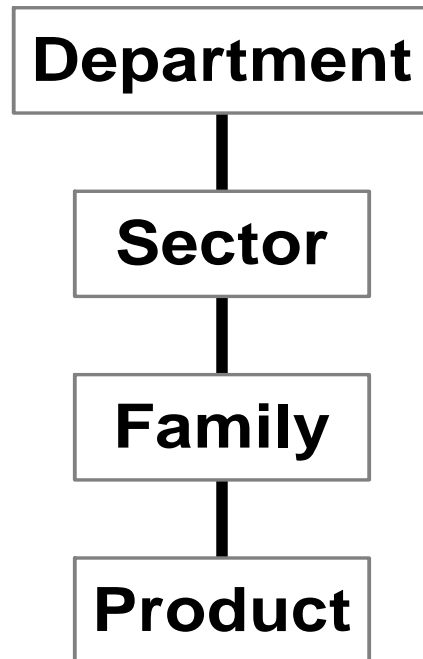
- ❑ Approach: reason at suitable level of abstraction

- ❑ A common form of background knowledge is that an attribute may be generalized or specialized according to a hierarchy of concepts

- ❑ Dimensions and levels can be efficiently encoded in transactions

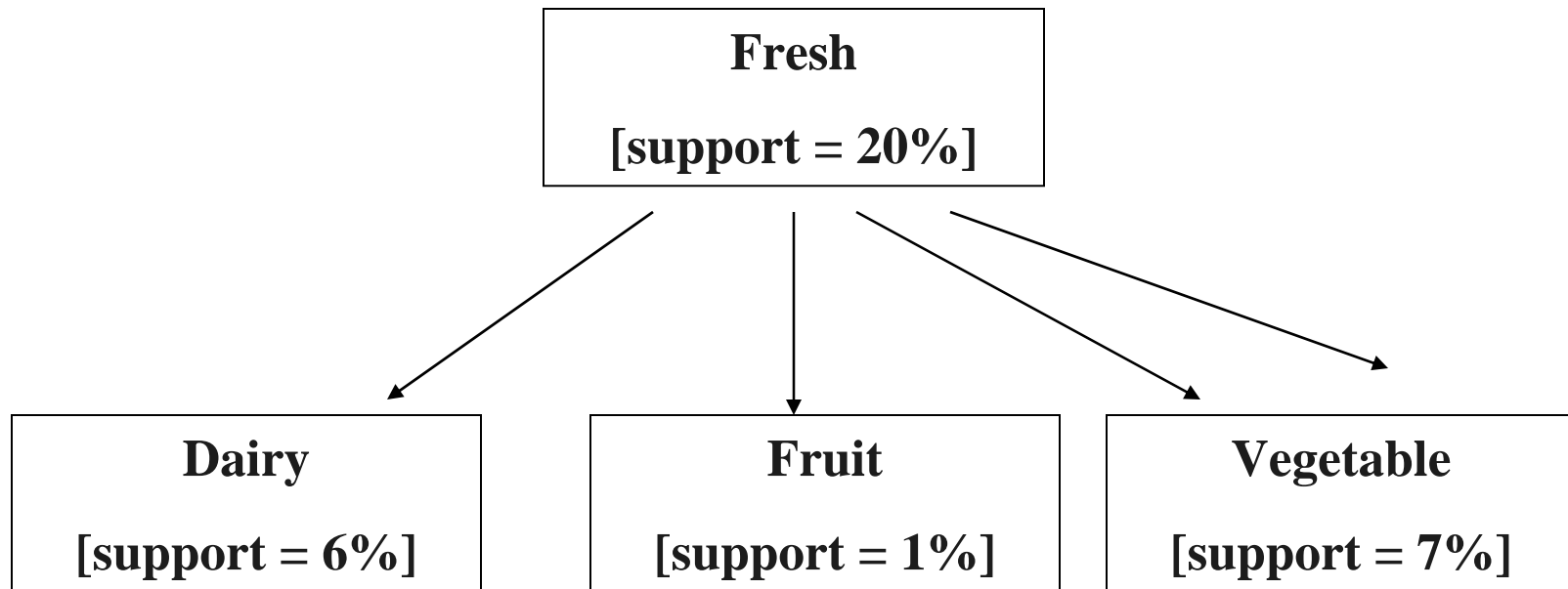
- ❑ Multilevel Association Rules: rules which combine associations with hierarchy of concepts

Hierarchy of concepts





Multilevel Association Rules



- ❑ Fresh \Rightarrow Bakery [20%, 60%]
- ❑ Dairy \Rightarrow Bread [6%, 50%]
- ❑ Fruit \Rightarrow Bread [1%, 50%] is not valid



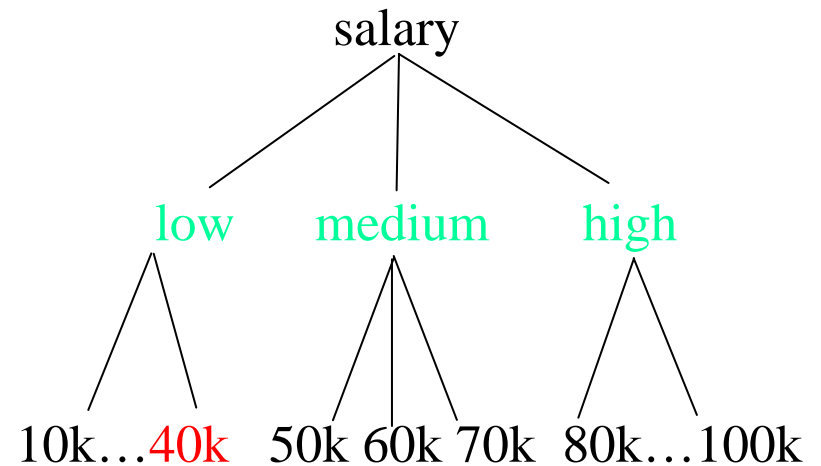
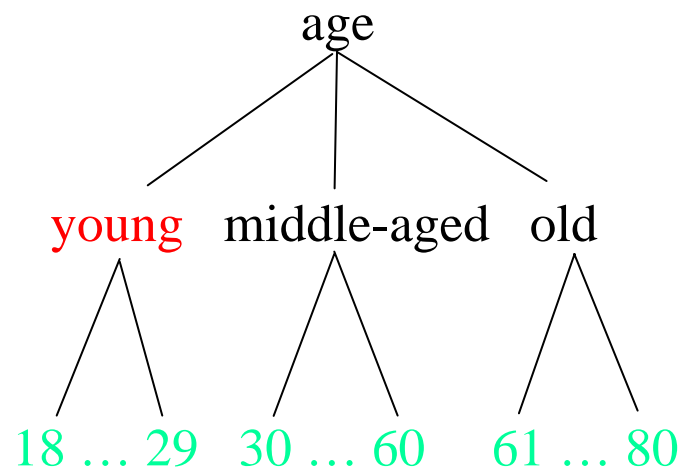
Support and Confidence of Multilevel Association Rules

- ❑ Generalizing/specializing values of attributes affects support and confidence
- ❑ from **specialized** to **general**: support of rules **increases** (new rules may become valid)
- ❑ from **general** to **specialized**: support of rules **decreases** (rules may become not valid, their support falls under the threshold)

Mining Multilevel Association Rules

Hierarchical attributes: age, salary

Association Rule: (age, young) → (salary, 40k)



Candidate Association Rules:

(age, 18) → (salary, 40k),

(age, young) → (salary, low),

(age, 18) → (salary, low)



Mining Multilevel Association Rules

- ❑ Calculate frequent itemsets at each concept level, until no more frequent itemsets can be found

- ❑ For each level use Apriori

- ❑ A top_down, progressive deepening approach:
 - First find high-level strong rules:
fresh → bakery [20%, 60%].
 - Then find their lower-level “weaker” rules:
fruit → bread [6%, 50%].

- ❑ Variations at mining multiple-level association rules.
 - Level-crossed association rules:
fruit → *wheat bread*



Multi-level Association: Uniform Support vs. Reduced Support

- Uniform Support: the same minimum support for all levels
 - One minimum support threshold. No need to examine itemsets containing any item whose ancestors do not have minimum support.
 - If support threshold
 - too high \Rightarrow miss low level associations.
 - too low \Rightarrow generate too many high level associations.

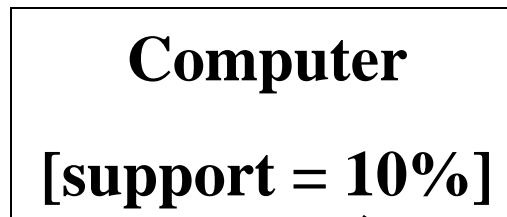
- Reduced Support: reduced minimum support at lower levels - different strategies possible.



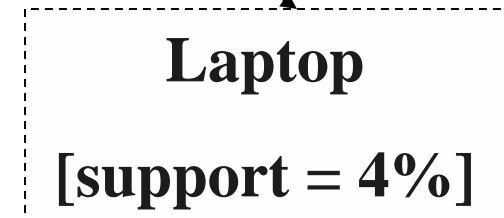
Uniform Support

Multi-level mining with uniform support

Level 1
min_sup = 5%



Level 2
min_sup = 5%





Reduced Support

Multi-level mining with reduced support

Level 1
min_sup = 5%

Computer
[support = 10%]

Level 2
min_sup = 3%

Desktop
[support = 6%]

Laptop
[support = 4%]



Multi-Dimensional Association: Concepts

- Single-dimensional rules:

$\text{buys}(X, \text{"milk"}) \Rightarrow \text{buys}(X, \text{"bread"})$

- Multi-dimensional rules:

Inter-dimension association rules (*no repeated predicates*)

$\text{age}(X, \text{"19-25"}) \wedge \text{occupation}(X, \text{"student"}) \Rightarrow$
 $\text{buys}(X, \text{"coke"})$

- hybrid-dimension association rules (*repeated predicates*)

$\text{age}(X, \text{"19-25"}) \wedge \text{buys}(X, \text{"popcorn"}) \Rightarrow \text{buys}(X,$
 $\text{"coke"})$



Summary

- Association rule mining
 - A large number of papers have been published
- Many interesting issues have been explored
- An interesting research direction
 - Association analysis in other types of data: spatial data, multimedia data, time series data, etc.



Mining Frequent Patterns Without Candidate Generation

- Compress a large database into a compact, Frequent-
Pattern tree (FP-tree) structure
 - highly condensed, but complete for frequent pattern mining
 - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
 - A divide-and-conquer methodology: decompose mining tasks into smaller ones
 - Avoid candidate generation: sub-database test only!

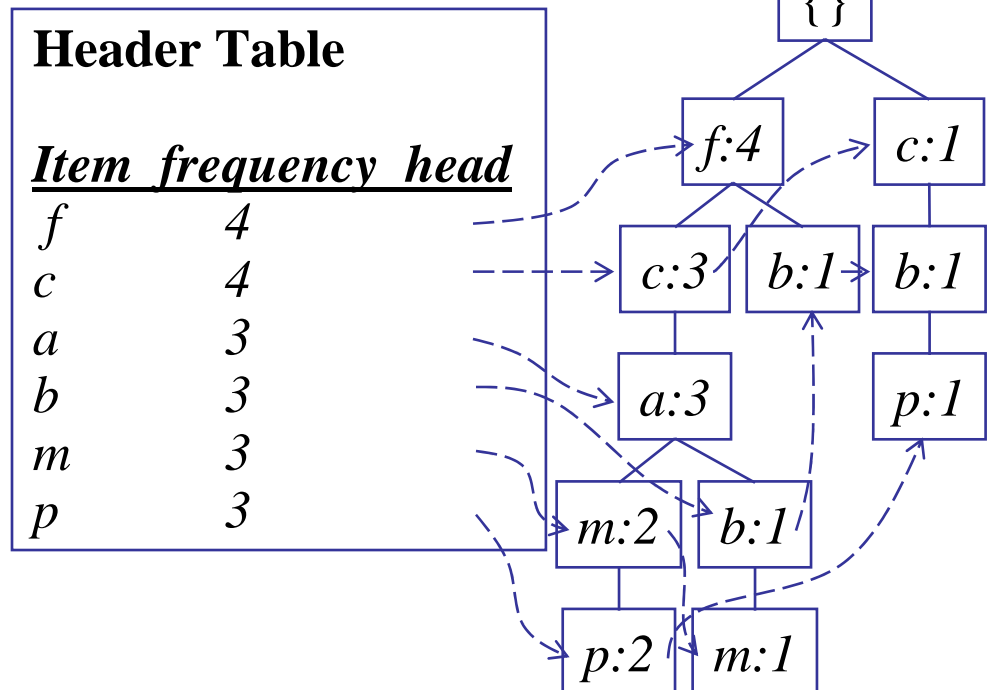
Construct FP-tree from a Transaction DB

<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
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}	{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 = 0.5

Steps:

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





Benefits of the FP-tree Structure

- Completeness:
 - preserves complete information for frequent pattern mining
- Compactness
 - reduce irrelevant information—infrequent items are gone
 - frequency descending ordering: more frequent items are more likely to be shared
 - never be larger than the original database (if not count node-links and counts)



Mining Frequent Patterns Using FP-tree

- General idea (divide-and-conquer)
 - Recursively grow frequent pattern path using the FP-tree
- Method
 - For each 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)



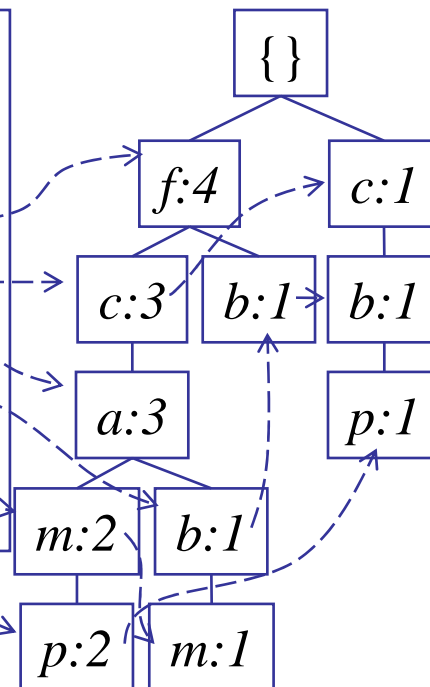
Major Steps to Mine FP-tree

- 1) Construct conditional pattern base for each node in the FP-tree
- 2) Construct conditional FP-tree from each conditional pattern-base
- 3) Recursively mine conditional FP-trees and grow frequent patterns obtained so far

Step 1: From FP-tree to Conditional Pattern Base

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

Header Table		
<i>Item</i>	<i>frequency</i>	<i>head</i>
<i>f</i>	4	
<i>c</i>	4	
<i>a</i>	3	
<i>b</i>	3	
<i>m</i>	3	
<i>p</i>	3	

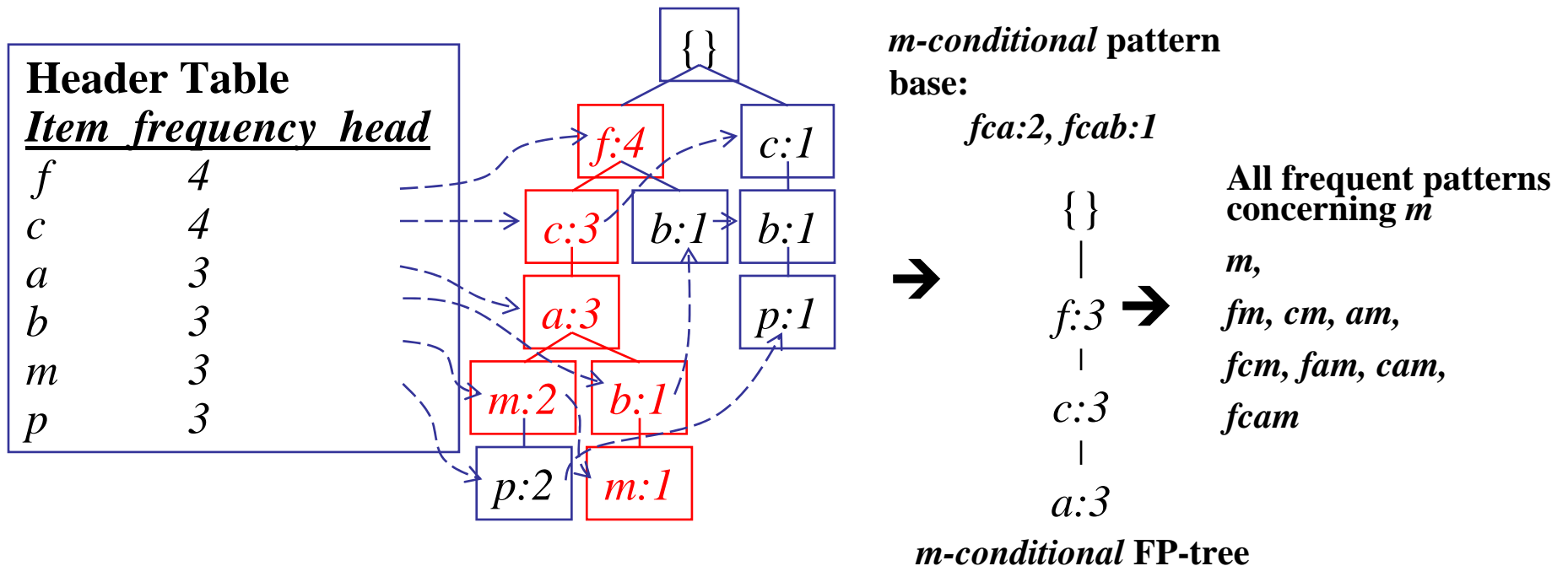


Conditional pattern bases

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

Step 2: Construct Conditional FP-tree

- 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





Mining Frequent Patterns by Creating Conditional Pattern-Bases

Item	Conditional pattern-base	Conditional FP-tree
p	{(fcam:2), (cb:1)}	{(c:3)} p
m	{(fca:2), (fcab:1)}	{(f:3, c:3, a:3)} m
b	{(fca:1), (f:1), (c:1)}	Empty
a	{(fc:3)}	{(f:3, c:3)} a
c	{(f:3)}	{(f:3)} c
f	Empty	Empty



Single FP-tree Path Generation

- Suppose an FP-tree T has a single path P
- The complete set of frequent pattern of T can be generated by enumeration of all the combinations of the sub-paths of P

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



**All frequent patterns
concerning m**

m,

fm, cm, am,

fcm, fam, cam,

fcam

m-conditional FP-tree



Principles of Frequent Pattern Growth

- Pattern growth property
 - Let α be a frequent itemset in DB, B be α 's conditional pattern base, and β be an itemset in B . Then $\alpha \cup \beta$ is a frequent itemset in DB iff β is frequent in B .
- "*abcdef*" is a frequent pattern, if and only if
 - "*abcde*" is a frequent pattern, and
 - "*f*" is frequent in the set of transactions containing "*abcde*"



Why Is Frequent Pattern Growth Fast?

- Our performance study shows
 - FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection
- Reasoning
 - No candidate generation, no candidate test
 - Use compact data structure
 - Eliminate repeated database scan
 - Basic operation is counting and FP-tree building

Quantitative Association Rules

RecordID	Age	Married	NumCars
100	23	No	1
200	25	Yes	1
300	29	No	0
400	34	Yes	2
500	38	Yes	2

How to get this rule?

Sample Rules	Support	Confidence
<age:30..39> and <married: yes> ==> <numCars:2>	40%	100%
<NumCars: 0..1> ==> <Married: No>	40%	66.70%

Quantitative Association Rules

People

RecordID	Age	Married	NumCars
100	23	No	1
200	25	Yes	1
300	29	No	0
400	34	Yes	2
500	38	Yes	2

Partition for age

Interval
20...24
25...29
25...29
30...34
35...39

Example Continue:

After partitioning Age

RecordID	Age	Married	NumCars
100	20...24	No	1
200	25...29	Yes	1
300	25...29	No	0
400	30...34	Yes	2
500	35...39	Yes	2

Mapping Age

Interval	Integer
20...24	1
25...29	2
30...34	3
35...39	4

Mapping Married

Value	Integer
Yes	1
No	2

Example Continue:

After mapping attributes

RecordID	Age	Married	NumCars
100	1	2	1
200	2	1	1
300	2	2	0
400	3	1	2
500	4	1	2

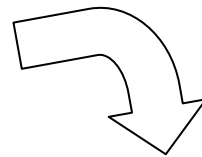
Frequent Itemset: Sample

Itemset	Support
{Age: 20...29}	3
{Age: 30...39}	2
{Married: Yes}	3
{Married: No}	2
{NumCars: 0...1}	3
{Age: 30...39, Married: Yes}	2

Mapping Quantitative to Boolean

- One possible solution is to map the problem to the Boolean association rules:
 - discretizing a non-categorical attribute to intervals
 - Age [20,29], [30,39],....
 - forming Boolean records
 - categorical attributes: each value becomes one item
 - non-categorical attributes: each interval becomes one item

RecordID	Age	Married	NoCars
100	23	No	1
500	38	Yes	2



RecID	Age: 20..29	Age: 30..39	Married: Yes	Married: No	Cars: 0	Cars: 1	Cars: 2
100	1	0	0	1	0	1	0
500	0	1	1	0	0	0	1



Mining Quantitative Association Rules

- Problems with the mapping
 - too few intervals: lost information
 - too low support: too many rules
- Solutions
 - using the supports of an itemset and its generalizations to determine the intervals
 - using interest measure to control the number of association rules



References

- R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In Proceedings of the 1993 International Conference on Management of Data (SIGMOD 93), pages 207-216, May 1993
- R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules (1994) Proc. 20th Int. Conf. Very Large Data Bases, VLDB
- Jiawei Han, Jian Pei, Yiwen Yin. Mining Frequent Patterns without Candidate Generation. SIGMOD Conference 2000: 1-12