Association Rule

<u>Frequent Pattern Approach</u>

FP Growth Algorithm

- NO candidate Generation
- A divide-and-conquer methodology: decompose mining tasks into smaller ones
- Requires 2 scans of the Transaction DB
- 2 Phase algorithm
  - Phase I

    Construct FP tree (Requires 2 TDB scans)
  - Phase II

    see FP tree (TDB is not used)
    FP tree contains all information about FIs

FP Tree Construction

- Item Prefix Tree
- FI Header Table
- Dependent on ordering of items
- Sort items in decreasing order of support count
  Non FIs are ignored
- Each Tr. is viewed as a list of FIs in descending
  order of support count

Construct FP-tree from a Transaction DB

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}	{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}

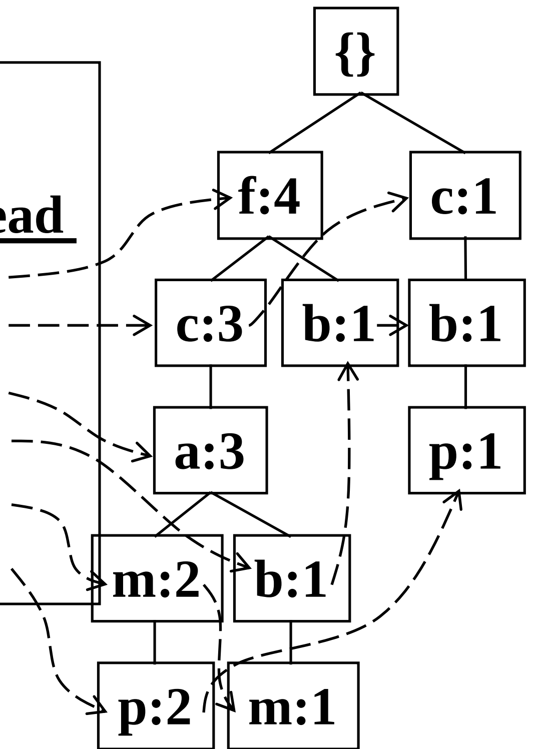
min_support = 0.5

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**

Header Table

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



Major Steps to Mine FP-tree

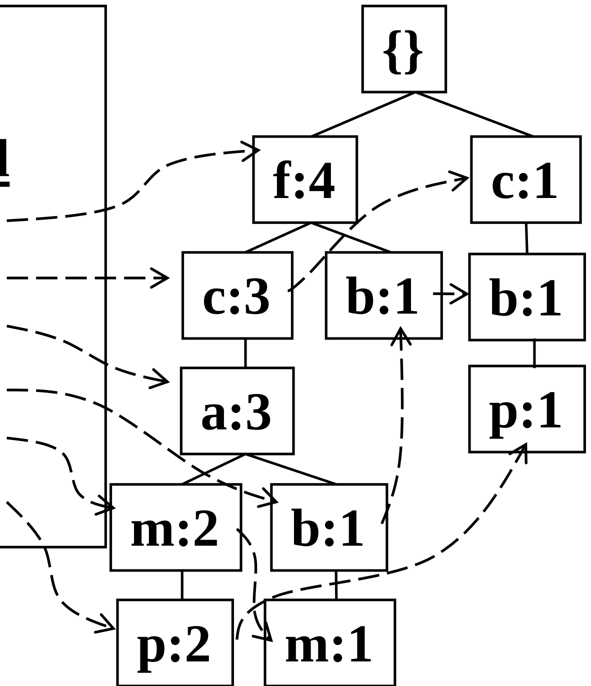
1. **Construct conditional pattern base for each node in the FP-tree**
2. **Construct conditional FP-tree from each conditional pattern-base**
3. **Recursively mine conditional FP-trees and grow frequent patterns obtained so far**
   - **If the conditional FP-tree contains a single path, simply enumerate all the patterns**

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

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



Conditional pattern bases

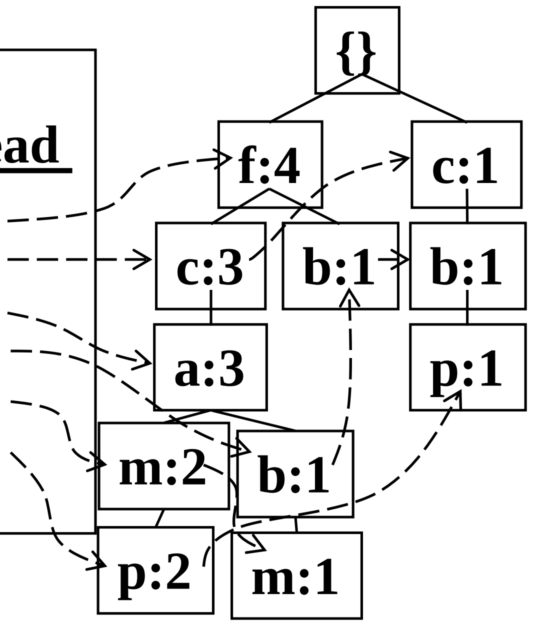
item	cond. pattern base
c	f:3
a	fc:3
b	fca:1, f:1, c:1
m	fca:2, fcab:1
p	fcam:2, cb:1

Step 2: Construct Conditional FP-tree

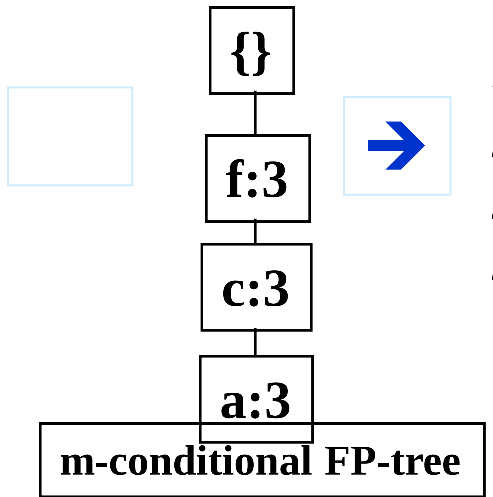
- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base

Header Table

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



m-conditional pattern base:
fca:2, fcab:1



All frequent patterns concerning *m*

m,

fm, cm, am,

fcm, fam, cam,

fcam

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

Principles of Frequent Pattern Growth

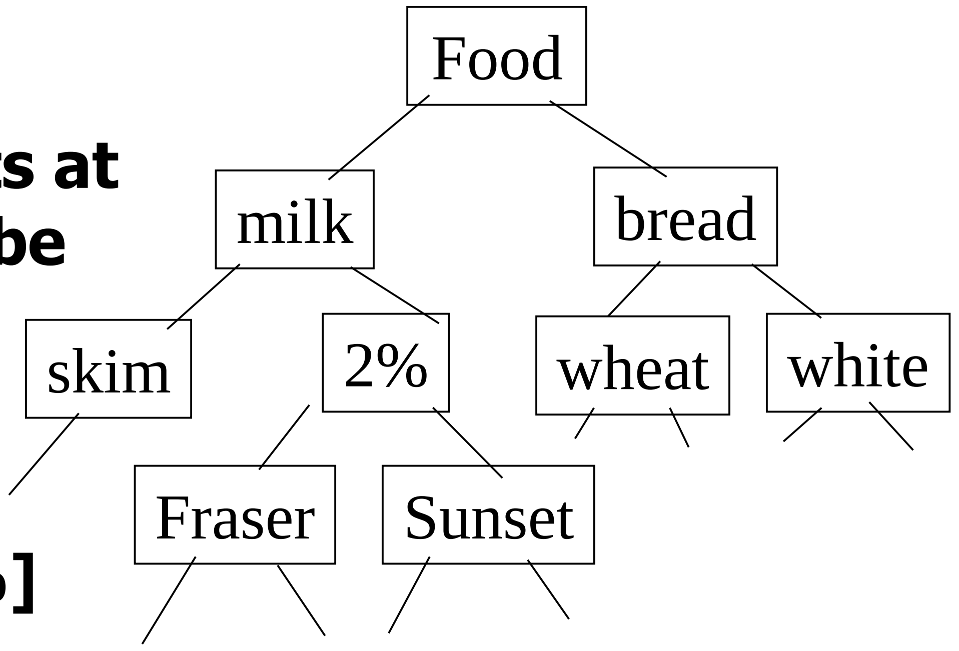
- **Pattern growth property**
  - **Let α be a frequent itemset in DB, B be α's conditional pattern base, and β be an itemset in B. Then $\alpha \cup \beta$ is a frequent itemset in DB iff β is frequent in B.**
- **"abcdef " is a frequent pattern, if and only if**
  - **"abcde " is a frequent pattern,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

Why Is FP-Growth Fast?

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

Multiple-Level Association Rules

- Items often form hierarchy.
- Items at the lower level are expected to have lower support.
- Rules regarding itemsets at appropriate levels could be quite useful.



milk ⇒ bread [20%, 60%]

2% milk ⇒ wheat bread [6%, 50%].

Multiple-Level Association Rules

mining multilevel association rules.

2% milk $\Rightarrow$ wheat bread

2% milk $\Rightarrow$ bread

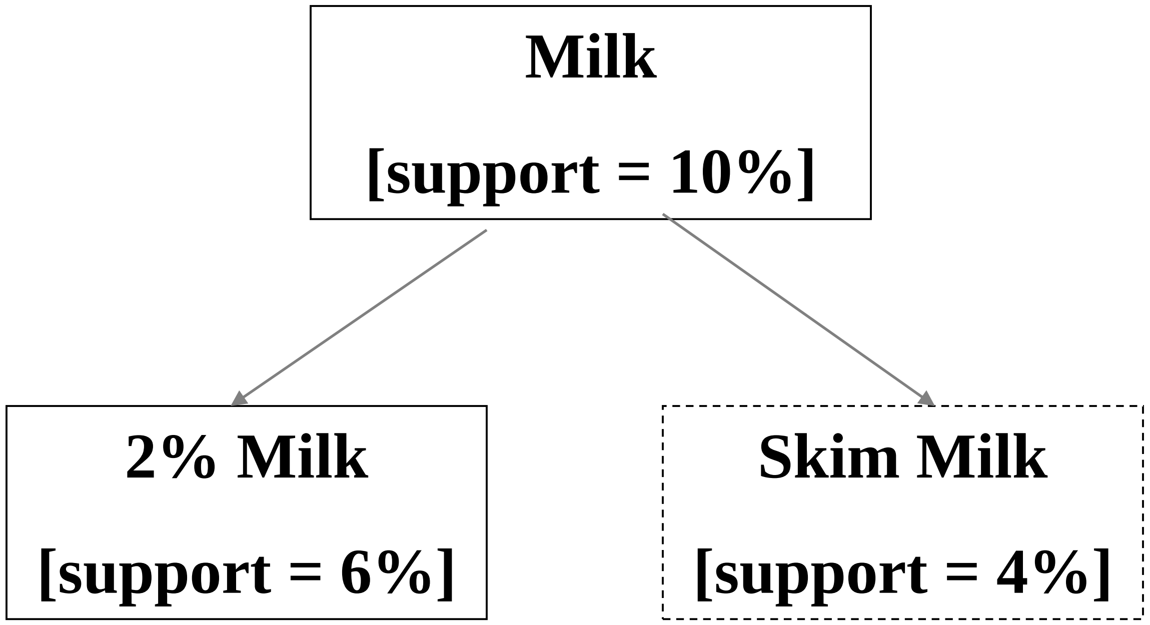
Multi-level Association: Uniform Support vs. Reduced Support

- 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**
  - **– Lower level items do not occur as frequently. 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
  - There are 4 search strategies:
    - **Level-by-level independent**
    - **Level-cross filtering by k-itemset**
    - **Level-cross filtering by single item**
    - **Controlled level-cross filtering by single item**

Uniform Support

Level 1
min_sup = 5%

Level 2
min_sup = 5%



Reduced Support

Level 1
min_sup = 5%

Level 2
min_sup = 3%

$$\begin{tikzcd}
 & \boxed{\begin{matrix}\textbf{Milk}\\ \textbf{[support = 10\%]}\end{matrix}} \arrow[dl] \arrow[dr] & \\
\boxed{\begin{matrix}\textbf{2\% Milk}\\ \textbf{[support = 6\%]}\end{matrix}} & & \boxed{\begin{matrix}\textbf{Skim Milk}\\ \textbf{[support = 4\%]}\end{matrix}}
\end{tikzcd}$$

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.

Multi-Dimensional Association: Concepts

- **Single-dimensional rules:**

  buys(X, "milk") ⇒ buys(X, "bread")

- **Multi-dimensional rules: ❍ 2 dimensions or predicates**
  - Inter-dimension association rules (no repeated predicates)

    age(X,"19-25") ∧ occupation(X,"student") ⇒ buys(X,"coke")
  - hybrid-dimension association rules (repeated predicates)

    age(X,"19-25") ∧ buys(X, "popcorn") ⇒ buys(X, "coke")
- **Categorical Attributes**
  - finite number of possible values, no ordering among values
- **Quantitative Attributes**
  - numeric, implicit ordering among values

Techniques for Mining MD Associations

- **Search for frequent k-predicate set:**
  - Example: **{age, occupation, buys}** is a 3-predicate set.
  - Techniques can be categorized by how **age** are treated.

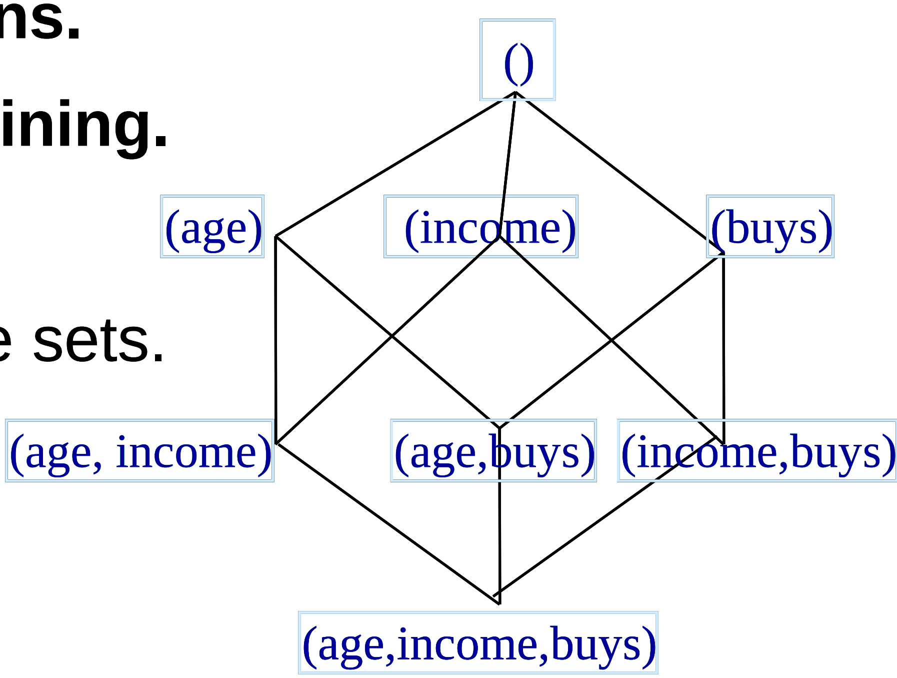
1. **Using static discretization of quantitative attributes**
   - Quantitative attributes are statically discretized by using predefined concept hierarchies.
2. **Quantitative association rules**
   - Quantitative attributes are dynamically discretized into "bins"based on the distribution of the data.
3. **Distance-based association rules**
   - This is a dynamic discretization process that considers the distance between data points.

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.

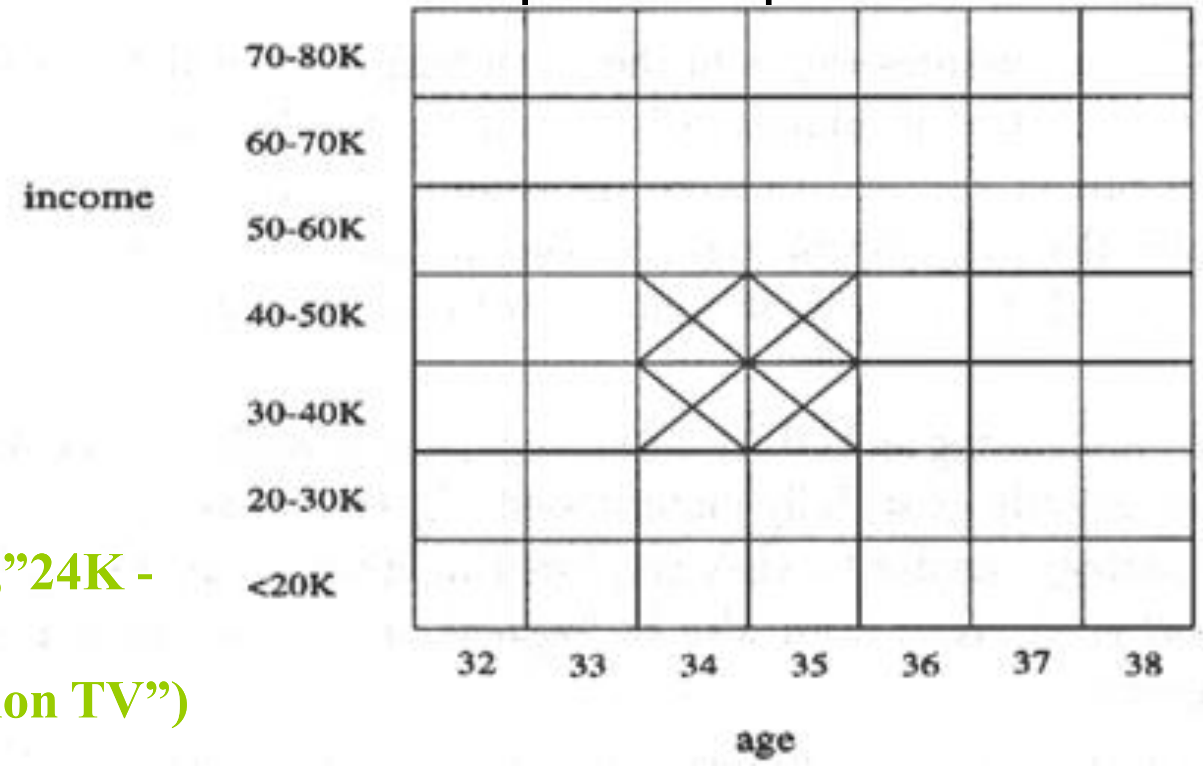
- **Mining from data cubes can be much faster.**



Quantitative Association Rules

- **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,"30-34") ∧ income(X,"24K - 48K")
⇒ buys(X,"high resolution TV")


70-80K
60-70K
income
50-60K
40-50K
30-40K
20-30K
<20K
32
33
34
35
36
37
38
age