DATA MINING 1
Pattern Mining & Association Rule Mining

Dino Pedreschi, Riccardo Guidotti

Revisited slides from Lecture Notes for Chapter 5 “Introduction to Data Mining”, 2nd Edition by Tan, Steinbach, Karpatne, Kumar

UNIVERSITÀ DI PISA

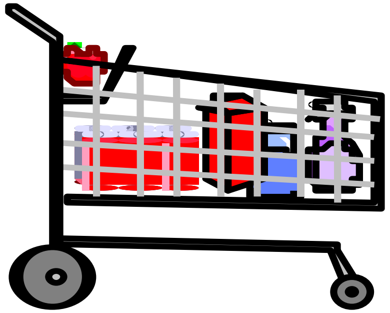
Association Rules - Module Outline

- What are association rules (AR) and what are they used for:
  - The paradigmatic application: Market Basket Analysis
  - The single dimensional AR (intra-attribute)

Market Basket Analysis: The Context

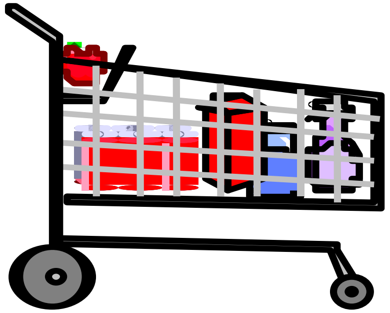
- Analyzing customer purchasing habits by finding associations and correlations between the different items that customers place in their “shopping basket”

Milk, eggs, sugar, bread



Customer1

Milk, eggs, cereal, bread



Customer2

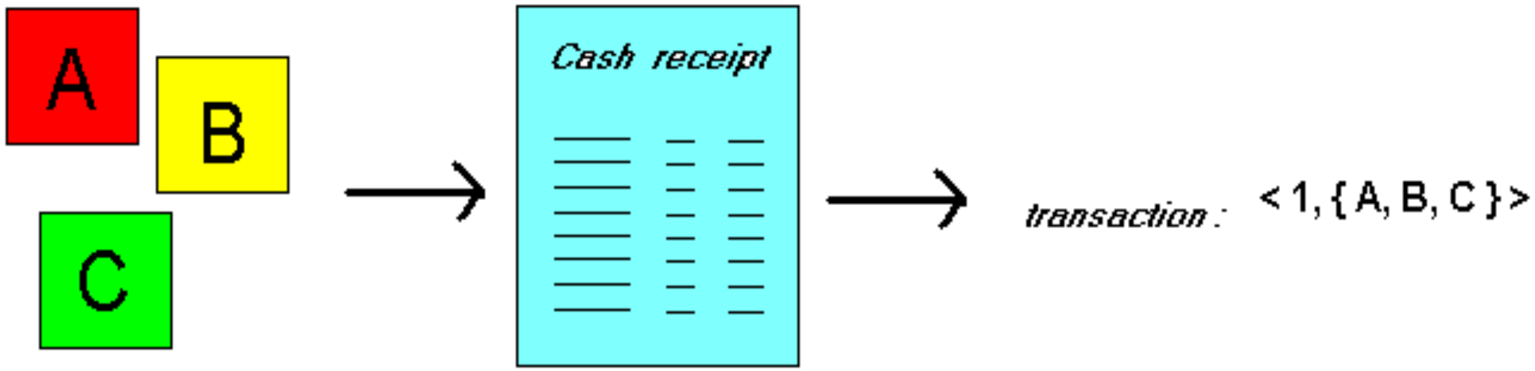
Eggs, sugar

Customer3

Market Basket Analysis: The Context

- Given: a database of customer transactions, where each transaction is a set of items.

- Goal: Find groups of items which are frequently purchased together.



Goal of MBA

- Extract information on purchasing behavior
- Actionable information: can suggest
  - new store layouts
  - new product assortments
  - which products to put on promotion
- MBA applicable whenever a customer purchases multiple things in proximity
  - credit cards
  - services of telecommunication companies
  - banking services
  - medical treatments

MBA: applicable to many other contexts

Telecommunication:

Each customer is a transaction containing the set of customer’s phone calls

Atmospheric phenomena:

Each time interval (e.g. a day) is a transaction containing the set of observed event (rains, wind, etc.)

Etc.

Association Rules

- Express how product/services relate to each other, and tend to group together
- "if a customer purchases three-way calling, then will also purchase call-waiting"
- simple to understand
- actionable information: bundle three-way calling and call-waiting in a single package
- Examples.
  - Rule form: "Body → Head [support, confidence]".
  - buys(x, "diapers") → buys(x, "beers") [0.5%, 60%]
  - major(x, "CS") and takes(x, "DB") → grade(x, "A") [1%, 75%]

Body/Head/Antecedent $X \rightarrow Y$ Head/Tail/Consequent

Useful, trivial, unexplicable

- Useful: “On Thursdays, grocery store consumers often purchase diapers and beer together”.
- Trivial: “Customers who purchase maintenance agreements are very likely to purchase large appliances”.
- Unexplicable: “When a new hardaware store opens, one of the most sold items is toilet rings.”

Apriori

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

Implication means co-occurrence, not causality!

Find groups of items which are frequently purchased together

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

TID	***Items***
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

Definition: Association Rule

- Association Rule
  - An implication expression of the form X → Y, where X and Y are itemsets
  - Example:
    {Milk, Diaper} → {Beer}

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

- Rule Evaluation Metrics
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \{\text{Beer}\}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association rules - module outline

- How to compute AR
    - Basic Apriori Algorithm and its optimizations
    - Multi-Dimension AR (inter-attribute)
    - Quantitative AR

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  ⇒ Computationally prohibitive!

Mining Association Rules

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Mining Association Rules

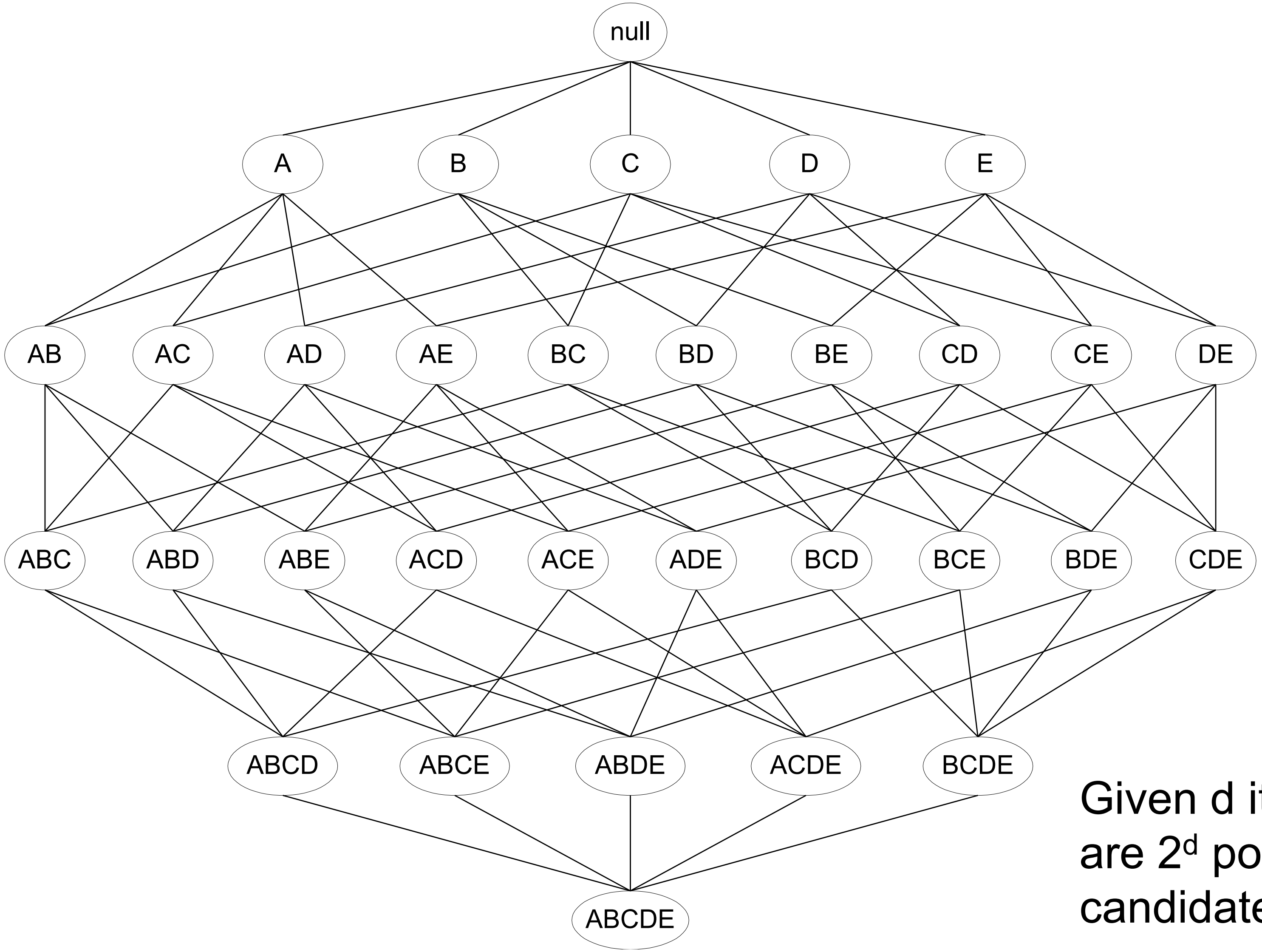
- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

Basic Apriori Algorithm

Problem Decomposition

1. Find the *frequent itemsets*: the sets of items that satisfy the support constraint
   - ◆A subset of a frequent itemset is also a frequent itemset, i.e., if {*A*,*B*} is a frequent itemset, both {*A*} and {*B*} should be a frequent itemset
   - ◆Iteratively find frequent itemsets with cardinality from 1 to *k* *(k*-itemset*)*
2. Use the frequent itemsets to generate association rules.

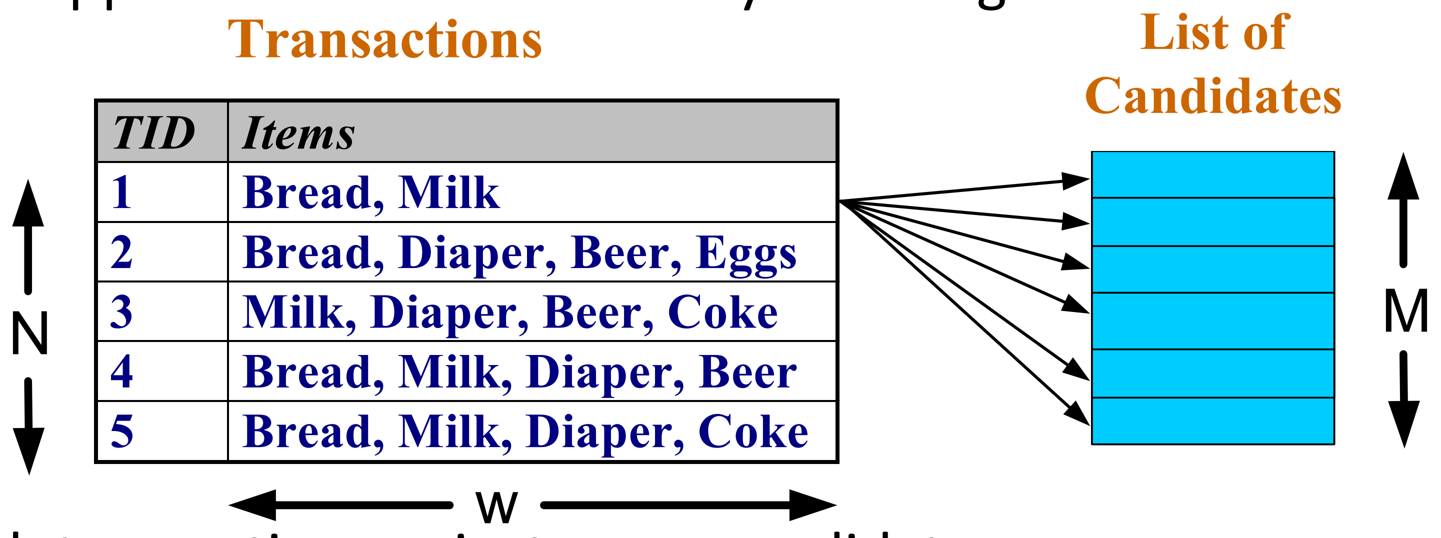
Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database



Transactions

List of Candidates

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

Reducing Number of Candidates

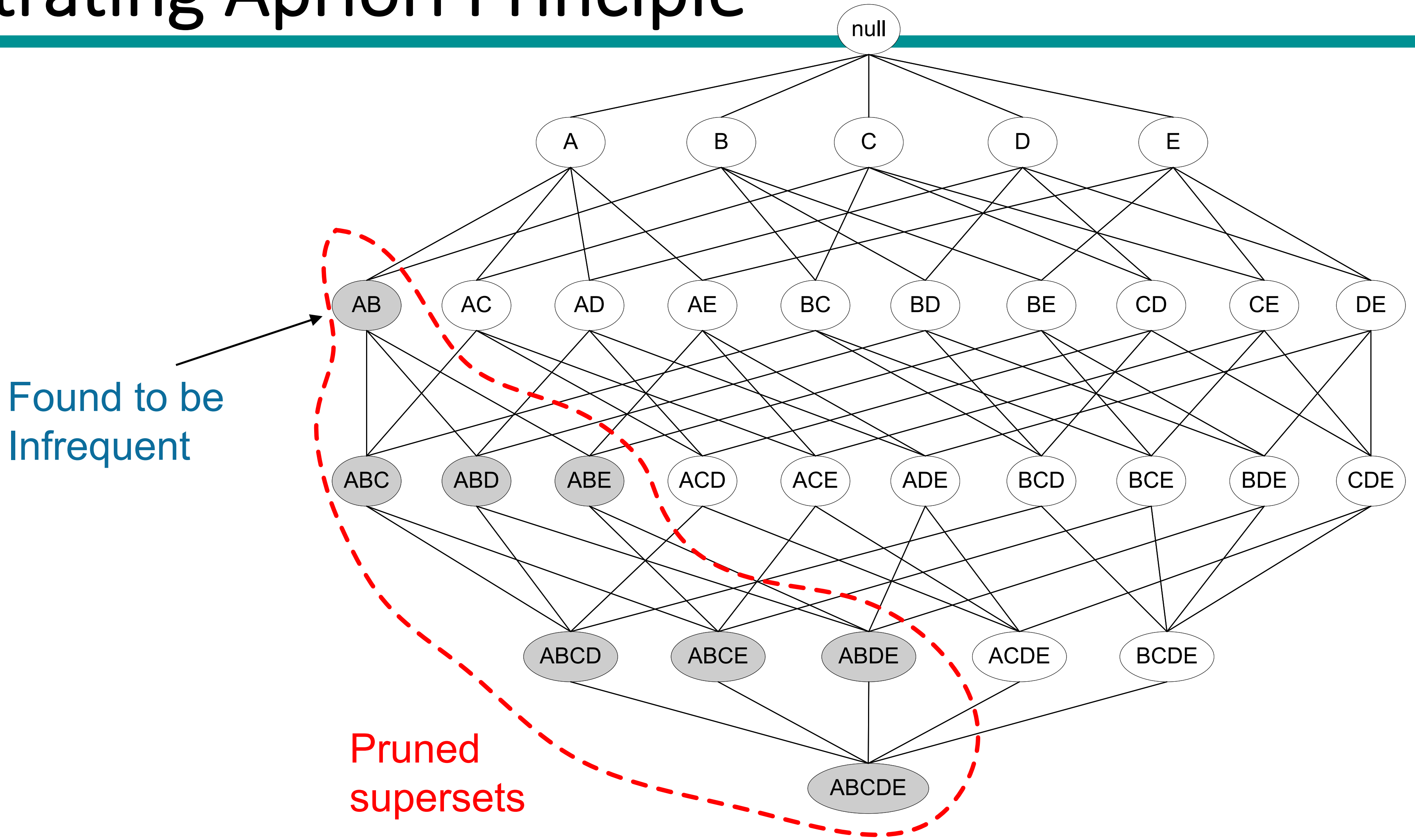
- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent

- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support

Illustrating Apriori Principle



Illustrating Apriori Principle

TID	*Items*
1	**Bread, Milk**
2	**Beer, Bread, Diaper, Eggs**
3	**Beer, Coke, Diaper, Milk**
4	**Beer, Bread, Diaper, Milk**
5	**Bread, Coke, Diaper, Milk**

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Minimum Support = 3

Illustrating Apriori Principle

TID	*Items*
1	**Bread, Milk**
2	**Beer, Bread, Diaper, Eggs**
3	**Beer, Coke, Diaper, Milk**
4	**Beer, Bread, Diaper, Milk**
5	**Bread, Coke, Diaper, Milk**

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Minimum Support = 3

Illustrating Apriori Principle

TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Itemset
{Bread,Milk}
{Bread, Beer }
{Bread,Diaper}
{Beer, Milk}
{Diaper, Milk}
{Beer,Diaper}

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Illustrating Apriori Principle

TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Beer, Bread}	**2**
{Bread,Diaper}	**3**
{Beer,Milk}	**2**
{Diaper,Milk}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Illustrating Apriori Principle

TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Minimum Support = 3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Triplets (3-itemsets)

(No need to generate candidates involving Bread, Beer or Milk, Beer)

Itemset
{ Beer, Diaper, Milk}
{ Beer,Bread,Diaper}
{Bread,Diaper,Milk}
{ Beer, Bread, Milk}

Illustrating Apriori Principle

TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Minimum Support = 3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Triplets (3-itemsets)

Itemset	Count
{ Beer, Diaper, Milk}	**2**
{ Beer,Bread, Diaper}	**2**
{Bread, Diaper, Milk}	**2**
{Beer, Bread, Milk}	**1**

Illustrating Apriori Principle

TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Minimum Support = 3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Triplets (3-itemsets)

Itemset	Count
{ Beer, Diaper, Milk}	**2**
{ Beer,Bread, Diaper}	**2**
{Bread, Diaper, Milk}	**2**
{Beer, Bread, Milk}	**1**

Apriori Algorithm

  - F_k: frequent k-itemsets
  - L_k: candidate k-itemsets

- Algorithm
  - Let k=1
  - Generate F_1 = {frequent 1-itemsets}
  - Repeat until F_k is empty
    - **Candidate Generation**: Generate L_{k+1} from F_k
    - **Candidate Pruning**: Prune candidate itemsets in L_{k+1} containing subsets of length k that are infrequent
    - **Support Counting**: Count the support of each candidate in L_{k+1} by scanning the DB
    - **Candidate Elimination**: Eliminate candidates in L_{k+1} that are infrequent, leaving only those that are frequent => F_{k+1}

Candidate Generation: F_{k-1} x F_{k-1} Method

- Merge two frequent (k-1)-itemsets if their first (k-2) items are identical

- F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE}
  - Merge(**AB**C, **AB**D) = **AB**CD
  - Merge(**AB**C, **AB**E) = **AB**CE
  - Merge(**AB**D, **AB**E) = **AB**DE

  - Do not merge(**A**BD,**A**CD) because they share only prefix of length 1 instead of length 2

Candidate Pruning

- Let F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE} be the set of frequent 3-itemsets

- L_4 = {ABCD,ABCE,ABDE} is the set of candidate 4-itemsets generated (from previous slide)

- Candidate pruning
  - Prune ABCE because ACE and BCE are infrequent
  - Prune ABDE because ADE is infrequent

- After candidate pruning: L_4 = {ABCD}

Illustrating Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread, Diaper, Milk}	**2**

Use of $F_{k-1}xF_{k-1}$ method for candidate generation results in only one 3-itemset. This is eliminated after the support counting step.

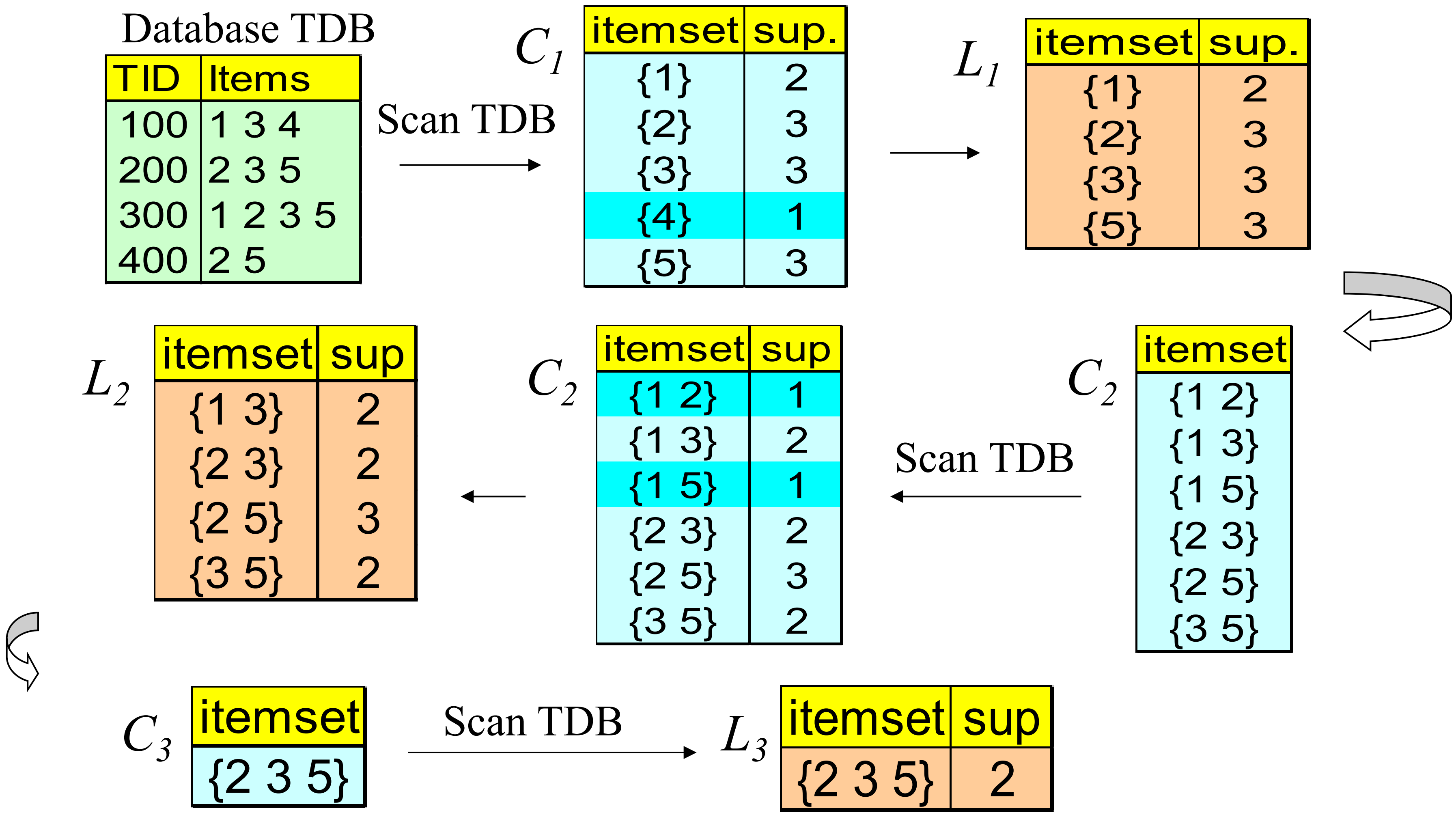
Support Counting of Candidate Itemsets

- Scan the database of transactions to determine the support of each candidate itemset
- Must match every candidate itemset against every transaction, which is an expensive operation

TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Itemset
{ Beer, Diaper, Milk}
{ Beer,Bread,Diaper}
{Bread, Diaper, Milk}
{ Beer, Bread, Milk}

Apriori Execution Example *(min_sup = 2)*



Database TDB

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

Scan TDB

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 TDB

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 TDB

L_3

itemset	sup
{2 3 5}	2

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, candidate rules:

ABC →D, ABD →C, ACD →B, BCD →A,
A →BCD, B →ACD, C →ABD, D →ABC
AB →CD, AC → BD, AD → BC, BC →AD,
BD →AC, CD →AB,

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

Rule Generation

- In general, confidence does not have an anti-monotone property

  $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

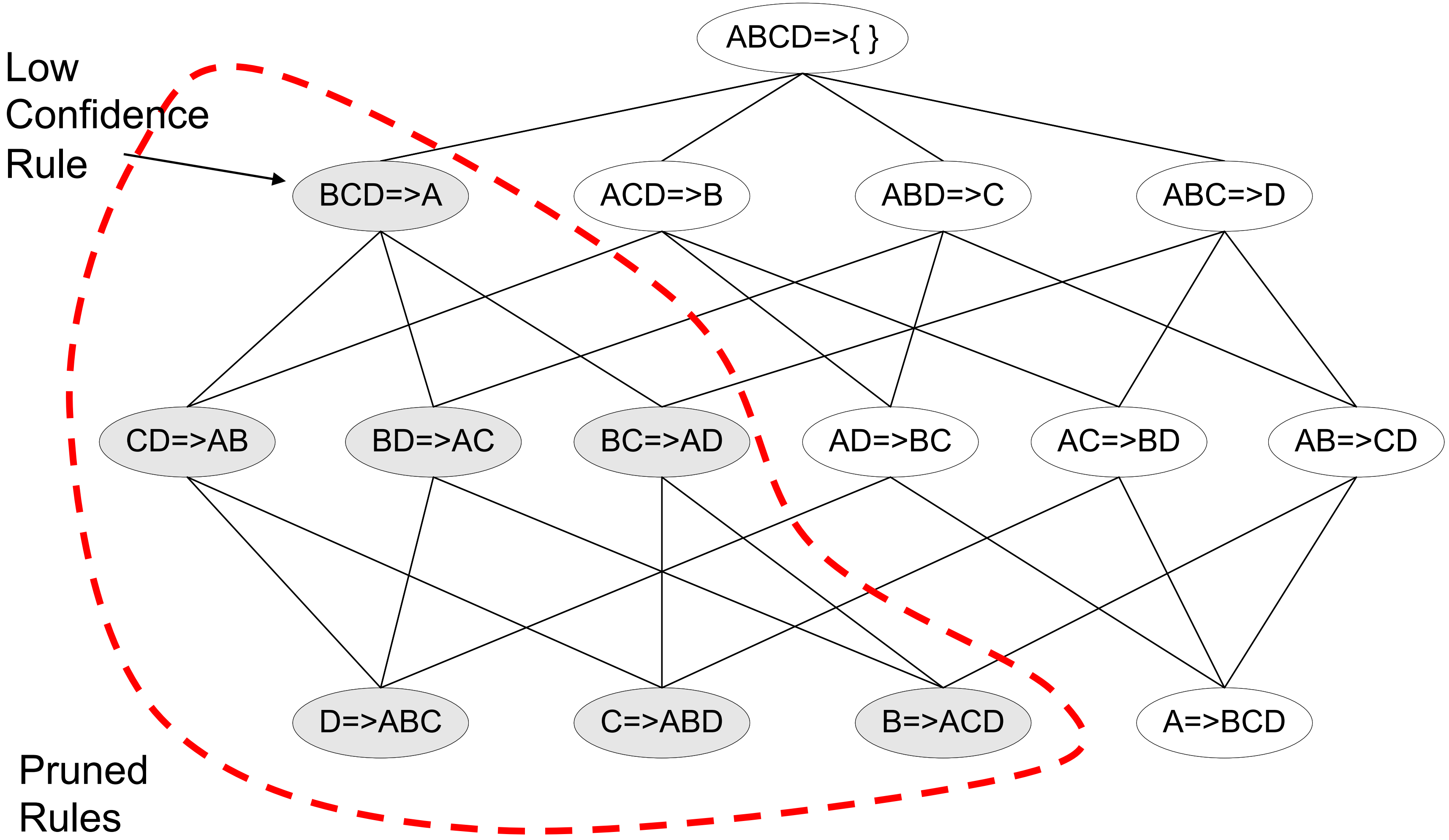
- But confidence of rules generated from the same itemset has an anti-monotone property
  - E.g., Suppose {A,B,C,D} is a frequent 4-itemset:

    $$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

  - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

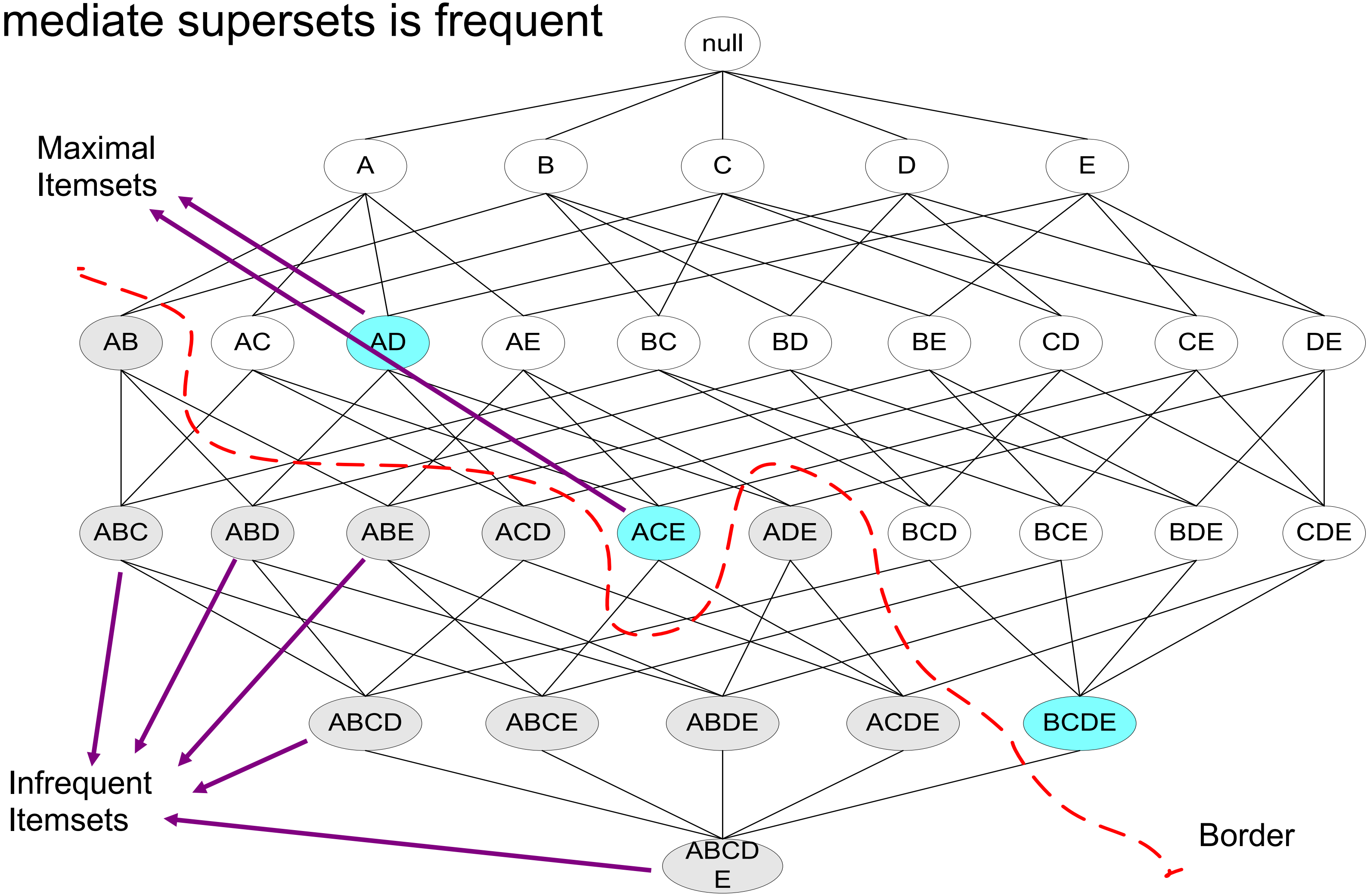
Rule Generation for Apriori Algorithm

Lattice of rules



Maximal Frequent Itemset

An itemset is maximal frequent if it is frequent and none of its immediate supersets is frequent



Closed Itemset

- An itemset X is closed if none of its immediate supersets has the same support as the itemset X.
- X is not closed if at least one of its immediate supersets has support count as X.

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	2
{A,B,C,D}	2

Maximal vs Closed Itemsets

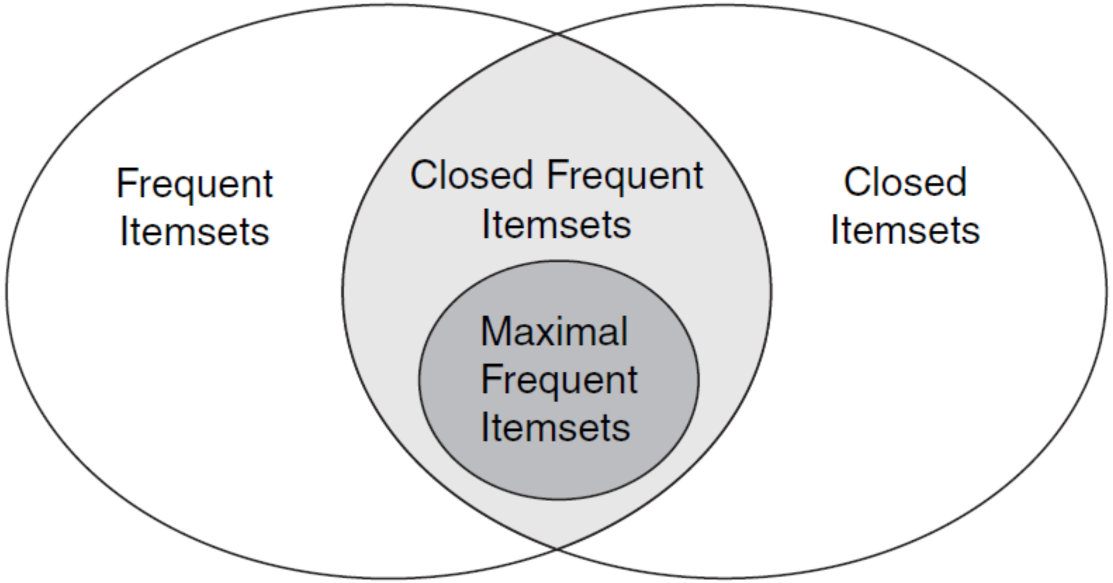


Figure 5.18. Relationships among frequent, closed, closed frequent, and maximal frequent itemsets.

Pattern Evaluation

- Association rule algorithms can produce large number of rules

- Interestingness measures can be used to prune/rank the patterns
  - In the original formulation, support & confidence are the only measures used

Computing Interestingness Measure

- Given X → Y or {X,Y}, information needed to compute interestingness can be obtained from a contingency table

Contingency table

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	N

f_{11}: support of X and Y
f_{10}: support of X and $\overline{Y}$
f_{01}: support of $\overline{X}$ and Y
f_{00}: support of $\overline{X}$ and $\overline{Y}$

Used to define various measures

- support, confidence, Gini, entropy, etc.

Drawback of Confidence

Customers	Tea	Coffee	…
C1	0	1	…
C2	1	0	…
C3	1	1	…
C4	1	0	…
…			

	$Coffee$	$\overline{Coffee}$	
Tea	150	50	200
$\overline{Tea}$	650	150	800
	800	200	1000

Association Rule: Tea → Coffee

Confidence ≅ P(Coffee|Tea) = 150/200 = 0.75

Confidence > 50%, meaning people who drink tea are more likely to drink coffee than not drink coffee

So rule seems reasonable

Drawback of Confidence

	Coffee	$\overline{\text{Coffee}}$	
Tea	150	50	200
$\overline{\text{Tea}}$	650	150	800
	800	200	1000

Association Rule: Tea → Coffee

Confidence= P(Coffee|Tea) = 150/200 = 0.75

but P(Coffee) = 0.8, which means knowing that a person drinks tea reduces the probability that the person drinks coffee!

$\Rightarrow$ Note that P(Coffee|$\overline{\text{Tea}}$) = 650/800 = 0.8125

Measure for Association Rules

- So, what kind of rules do we really want?
  - Confidence(X → Y) should be sufficiently high
    - To ensure that people who buy X will more likely buy Y than not buy Y
  - Confidence(X → Y) > support(Y)
    - Otherwise, rule will be misleading because having item X actually reduces the chance of having item Y in the same transaction
    - Is there any measure that capture this constraint?
      - Answer: Yes. There are many of them.

Statistical Relationship between X and Y

- The criterion

  $$\text{confidence}(X \rightarrow Y) = \text{support}(Y)$$

  is equivalent to:

  - $P(Y|X) = P(Y)$
  - $P(X,Y) = P(X) \times P(Y)$ (X and Y are independent)

  If $P(X,Y) > P(X) \times P(Y)$: X & Y are positively correlated

  If $P(X,Y) < P(X) \times P(Y)$: X & Y are negatively correlated

Measures that take into account statistical dependence

$$\left.\begin{aligned} Lift &= \frac{P(Y \mid X)}{P(Y)} \\[2ex] Interest &= \frac{P(X,Y)}{P(X)P(Y)} \end{aligned}\right\}$$

lift is used for rules while interest is used for itemsets

$$PS = P(X,Y) - P(X)P(Y)$$

$$\phi - coefficient = \frac{P(X,Y) - P(X)P(Y)}{\sqrt{P(X)[1-P(X)]P(Y)[1-P(Y)]}}$$

Example: Lift/Interest

	Coffee	$\overline{\text{Coffee}}$	
Tea	150	50	200
$\overline{\text{Tea}}$	650	150	800
	800	200	1000

Association Rule: Tea → Coffee

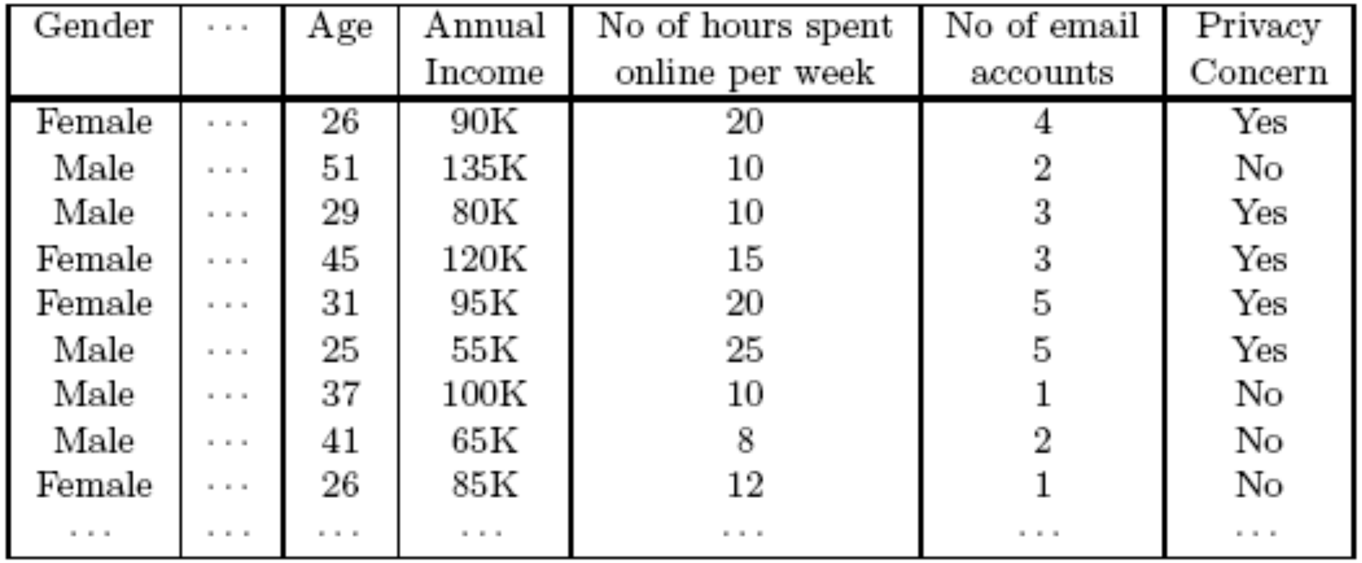
Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.8

⇒ Interest = 0.15 / (0.2×0.8) = 0.9375 (< 1, therefore is negatively associated)

Continuous and Categorical Attributes

How to apply association analysis to non-asymmetric binary variables?



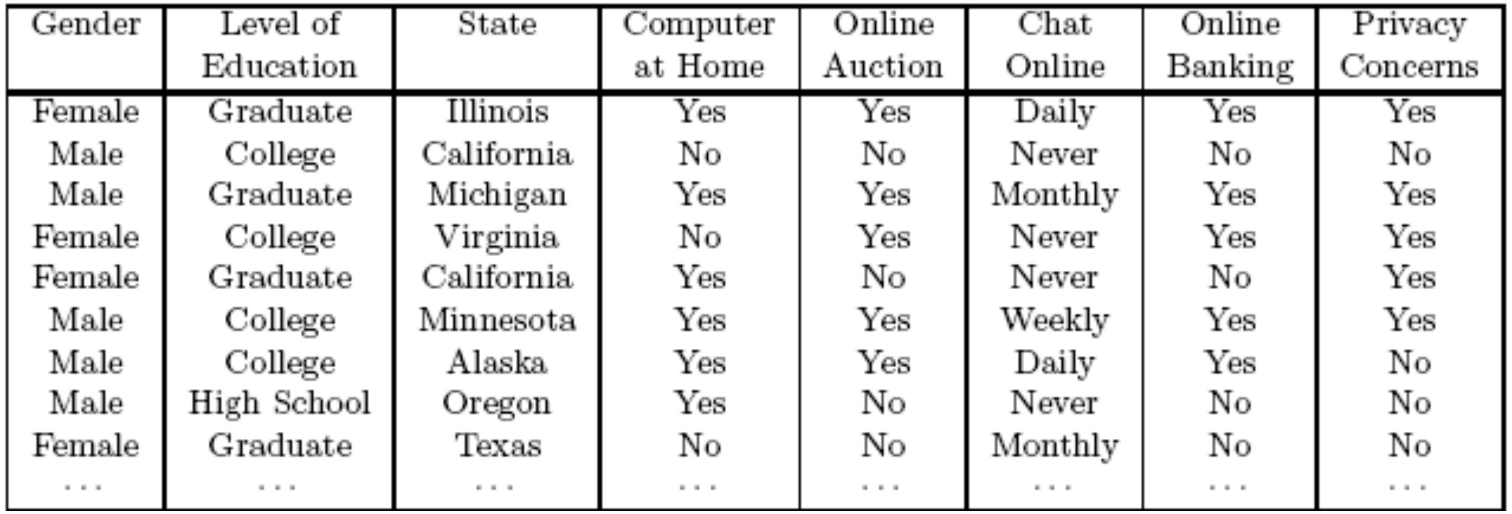
Gender	···	Age	Annual Income	No of hours spent online per week	No of email accounts	Privacy Concern
Female	···	26	90K	20	4	Yes
Male	···	51	135K	10	2	No
Male	···	29	80K	10	3	Yes
Female	···	45	120K	15	3	Yes
Female	···	31	95K	20	5	Yes
Male	···	25	55K	25	5	Yes
Male	···	37	100K	10	1	No
Male	···	41	65K	8	2	No
Female	···	26	85K	12	1	No
···	···	···	···	···	···	···

Example of Association Rule:

{Gender=Male, Age $\in$ [21,30)} $\rightarrow$ {No of hours online $\geq$ 10}

Handling Categorical Attributes

- Example: Internet Usage Data

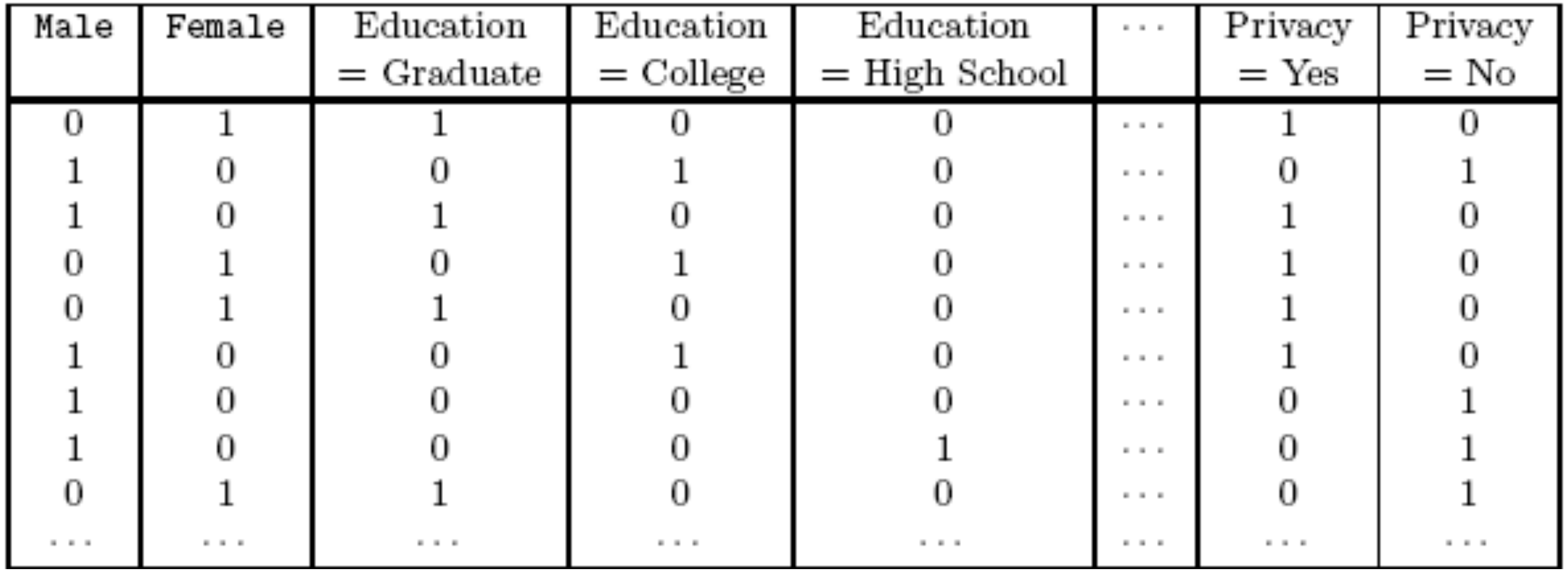


Gender	Level of Education	State	Computer at Home	Online Auction	Chat Online	Online Banking	Privacy Concerns
Female	Graduate	Illinois	Yes	Yes	Daily	Yes	Yes
Male	College	California	No	No	Never	No	No
Male	Graduate	Michigan	Yes	Yes	Monthly	Yes	Yes
Female	College	Virginia	No	Yes	Never	Yes	Yes
Female	Graduate	California	Yes	No	Never	No	Yes
Male	College	Minnesota	Yes	Yes	Weekly	Yes	Yes
Male	College	Alaska	Yes	Yes	Daily	Yes	No
Male	High School	Oregon	Yes	No	Never	No	No
Female	Graduate	Texas	No	No	Monthly	No	No
...	...	...	...	...	...	...	...

{Level of Education=Graduate, Online Banking=Yes}
→ {Privacy Concerns = Yes}

Handling Categorical Attributes

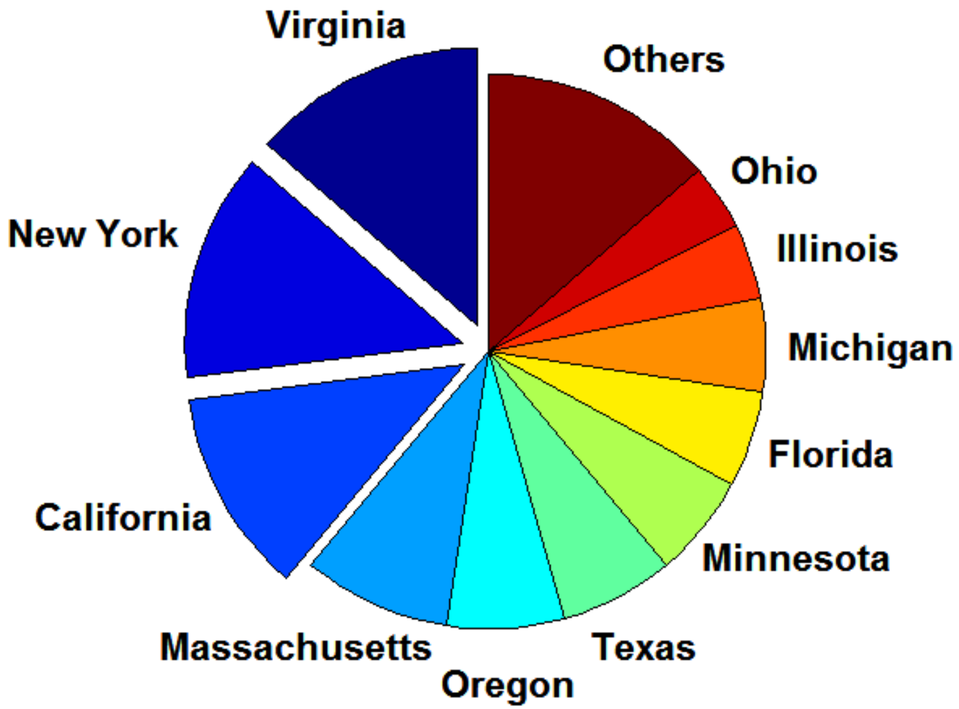
- Introduce a new “item” for each distinct attribute-value pair



Male	Female	Education = Graduate	Education = College	Education = High School	···	Privacy = Yes	Privacy = No
0	1	1	0	0	···	1	0
1	0	0	1	0	···	0	1
1	0	1	0	0	···	1	0
0	1	0	1	0	···	1	0
0	1	1	0	0	···	1	0
1	0	0	1	0	···	1	0
1	0	0	0	0	···	0	1
1	0	0	0	1	···	0	1
0	1	1	0	0	···	0	1
···	···	···	···	···	···	···	···

Handling Categorical Attributes

- Some attributes can have many possible values
  - Many of their attribute values have very low support
    - Potential solution: Aggregate the low-support attribute values

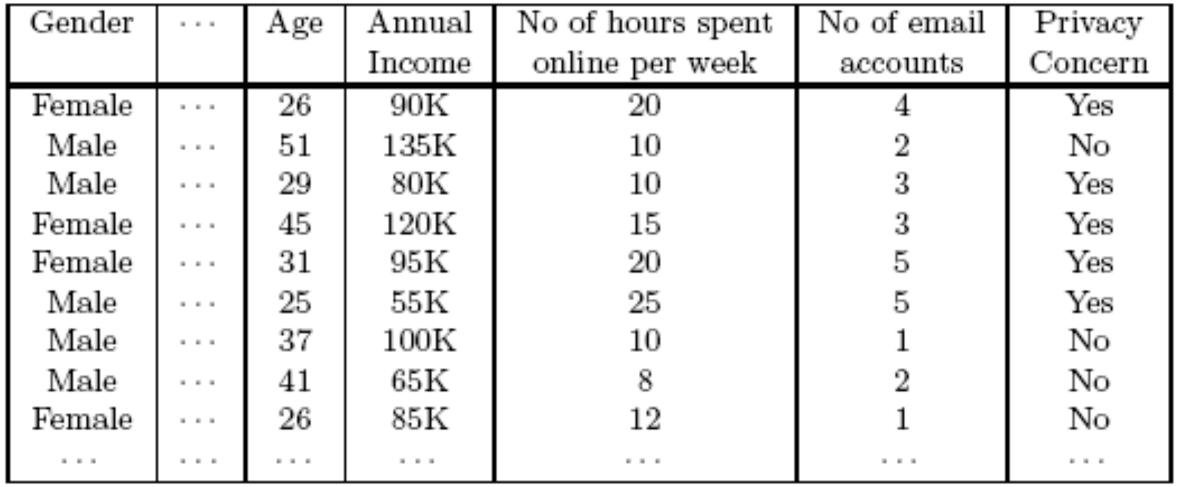


Handling Continuous Attributes

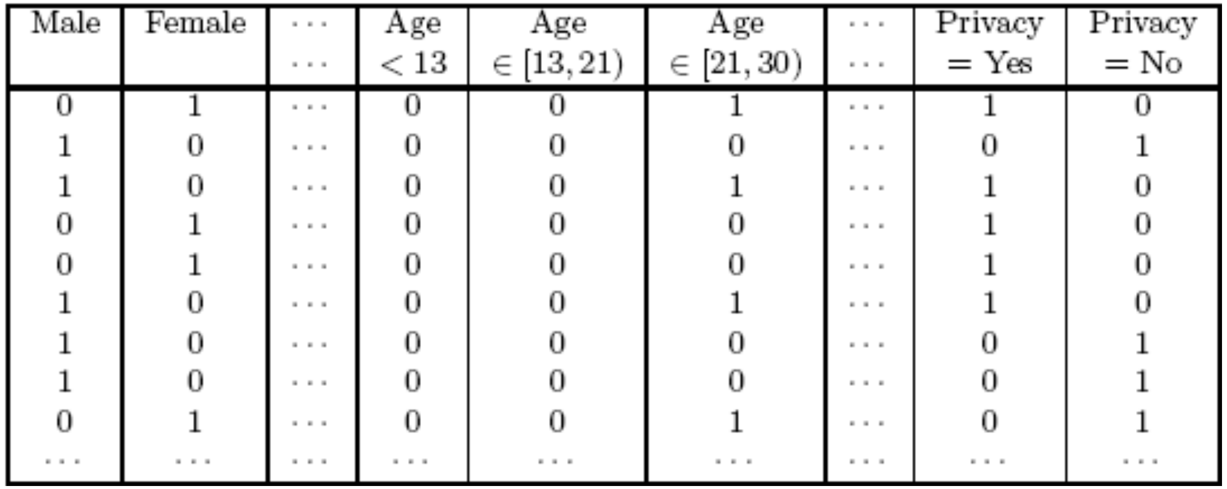
- Different methods:
  - Discretization-based
  - Statistics-based
  - Non-discretization based
    - minApriori

- Different kinds of rules can be produced:
  - {Age$\in$[21,30), No of hours online$\in$[10,20)} $\rightarrow$ {Chat Online =Yes}
  - {Age$\in$[21,30), Chat Online = Yes} $\rightarrow$ No of hours online: μ=14, σ=4

Discretization-based Methods

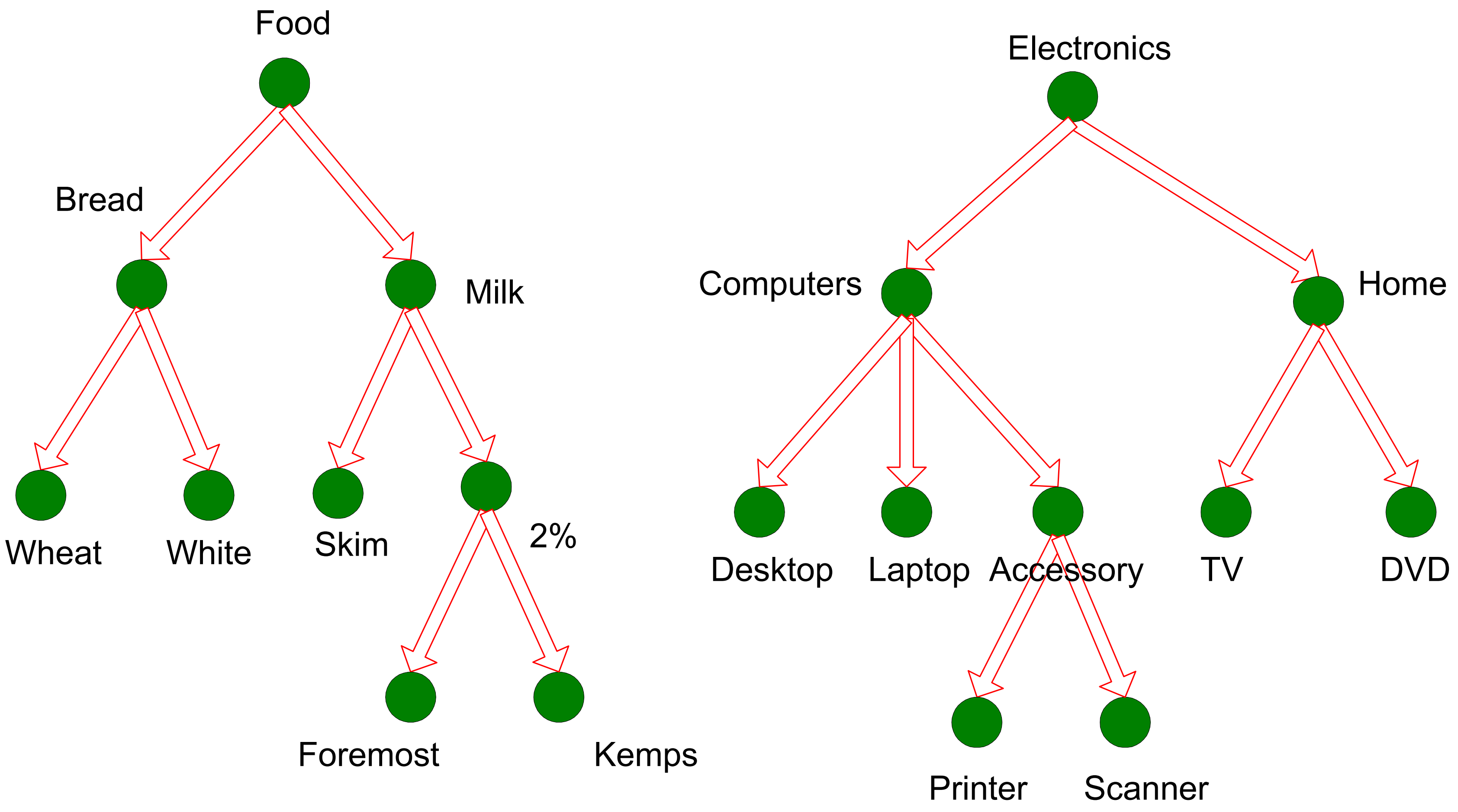


Gender	···	Age	Annual Income	No of hours spent online per week	No of email accounts	Privacy Concern
Female	···	26	90K	20	4	Yes
Male	···	51	135K	10	2	No
Male	···	29	80K	10	3	Yes
Female	···	45	120K	15	3	Yes
Female	···	31	95K	20	5	Yes
Male	···	25	55K	25	5	Yes
Male	···	37	100K	10	1	No
Male	···	41	65K	8	2	No
Female	···	26	85K	12	1	No
···	···	···	···	···	···	···



Male	Female	··· ···	Age < 13	Age $\in [13, 21)$	Age $\in [21, 30)$	··· ···	Privacy = Yes	Privacy = No
0	1	···	0	0	1	···	1	0
1	0	···	0	0	0	···	0	1
1	0	···	0	0	1	···	1	0
0	1	···	0	0	0	···	1	0
0	1	···	0	0	0	···	1	0
1	0	···	0	0	1	···	1	0
1	0	···	0	0	0	···	0	1
1	0	···	0	0	0	···	0	1
0	1	···	0	0	1	···	0	1
···	···	···	···	···	···	···	···	···

Concept Hierarchies



Multi-level Association Rules

- Why should we incorporate concept hierarchy?
  - Rules at lower levels may not have enough support to appear in any frequent itemsets
  - Rules at lower levels of the hierarchy are overly specific
    - e.g., skim milk $\rightarrow$ white bread, 2% milk $\rightarrow$ wheat bread,
      skim milk $\rightarrow$ wheat bread, etc.

      are indicative of association between milk and bread
  - Rules at higher level of hierarchy may be too generic

Multi-level Association Rules

- Approach 1: Extend current association rule formulation by augmenting each transaction with higher level items

  Original Transaction: {skim milk, wheat bread}

  Augmented Transaction:
  {skim milk, wheat bread, milk, bread, food}

- Issues:
  - Items that reside at higher levels have much higher support counts
  - if support threshold is low, too many frequent patterns involving items from the higher levels
  - Increased dimensionality of the data

Multi-level Association Rules

- Approach 2:
  - Generate frequent patterns at highest level first
  - Then, generate frequent patterns at the next highest level, and so on

- Issues:
  - I/O requirements will increase dramatically because we need to perform more passes over the data
  - May miss some potentially interesting cross-level association patterns

Is Apriori Fast Enough?

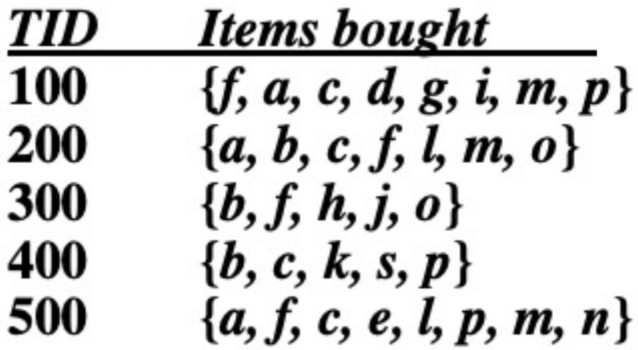
- The core of the Apriori algorithm:
  - Use frequent (k – 1)-itemsets to generate candidate frequent k-itemsets
  - Use database scan and pattern matching to collect counts for the candidate itemsets
- The bottleneck of Apriori: candidate generation
  - Huge candidate sets:
    - 10^4 frequent 1-itemset will generate 10^7 candidate 2-itemsets
    - To discover a frequent pattern of size 100, e.g., $\{a_1, a_2, …, a_{100}\}$, 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

FP-Growth

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!

How to construct a FP-tree

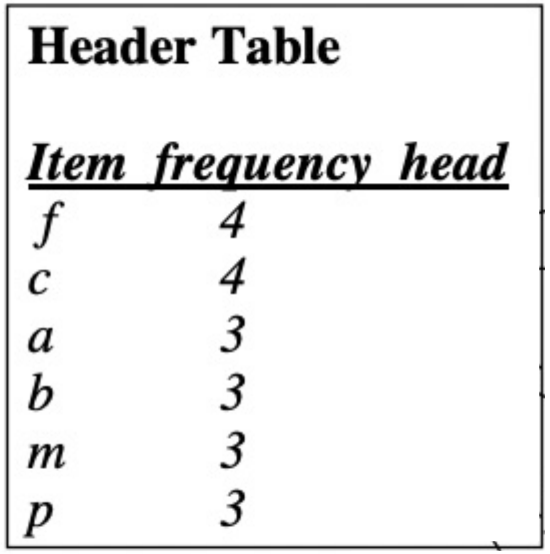


TID	*Items bought*
100	*{f, a, c, d, g, i, m, p}*
200	*{a, b, c, f, l, m, o}*
300	*{b, f, h, j, o}*
400	*{b, c, k, s, p}*
500	*{a, f, c, e, l, p, m, n}*

min_support = 3

Steps:

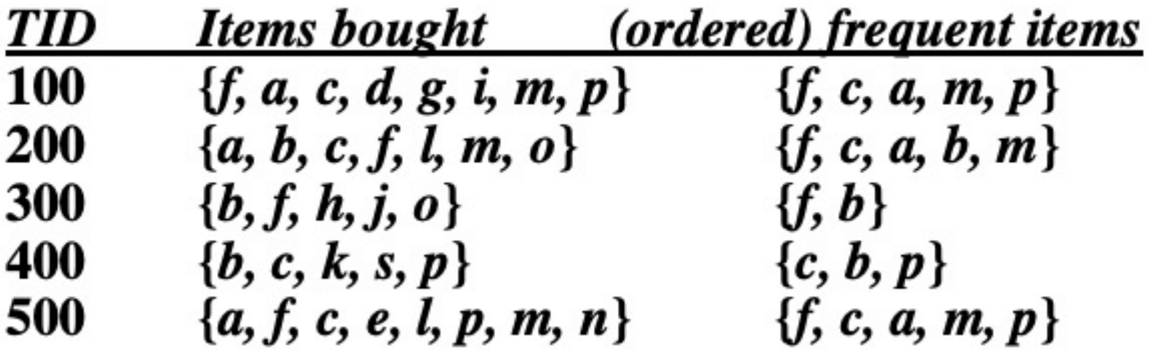
1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Order frequent items in frequency descending order
3. Scan DB again, construct FP-tree



Header Table

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

How to construct a FP-tree

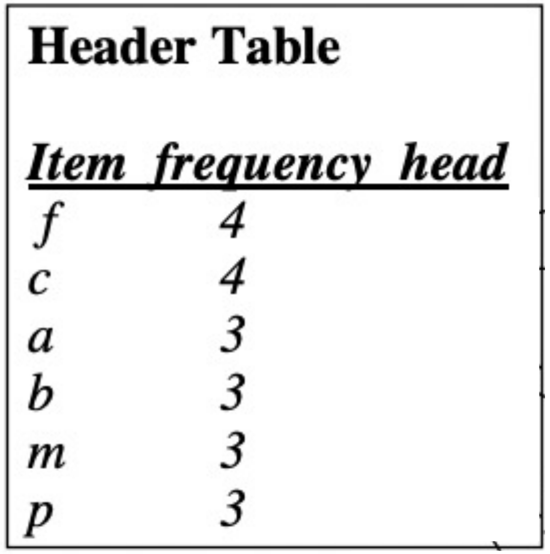


TID	*Items bought*	*(ordered) frequent items*
100	*{f, a, c, d, g, i, m, p}*	*{f, c, a, m, p}*
200	*{a, b, c, f, l, m, o}*	*{f, c, a, b, m}*
300	*{b, f, h, j, o}*	*{f, b}*
400	*{b, c, k, s, p}*	*{c, b, p}*
500	*{a, f, c, e, l, p, m, n}*	*{f, c, a, m, p}*

min_support = 3

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	

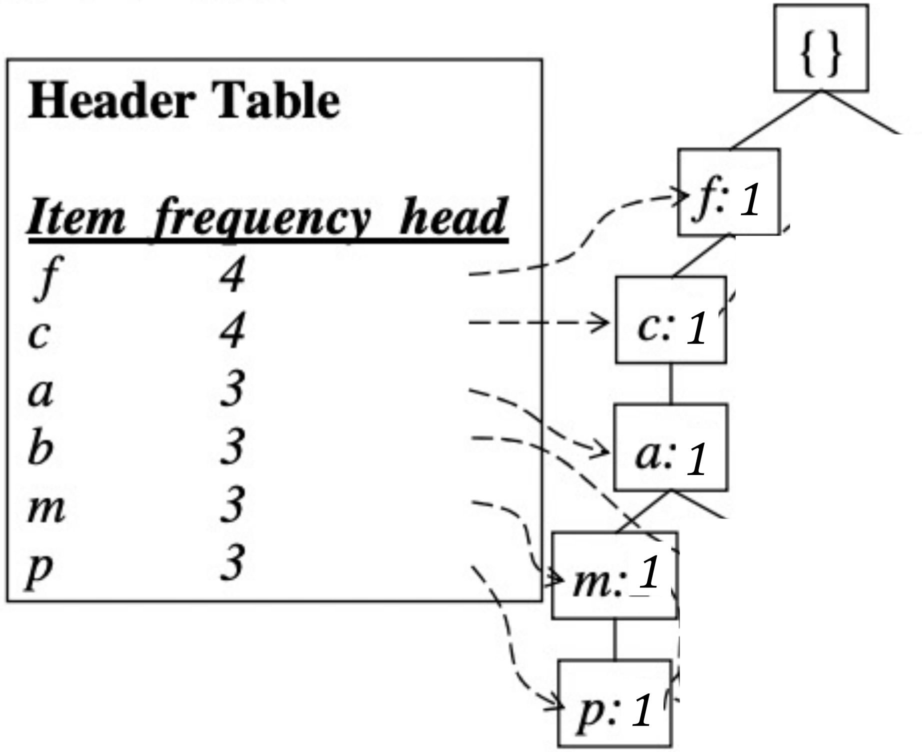
How to construct a FP-tree

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 = 3

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	

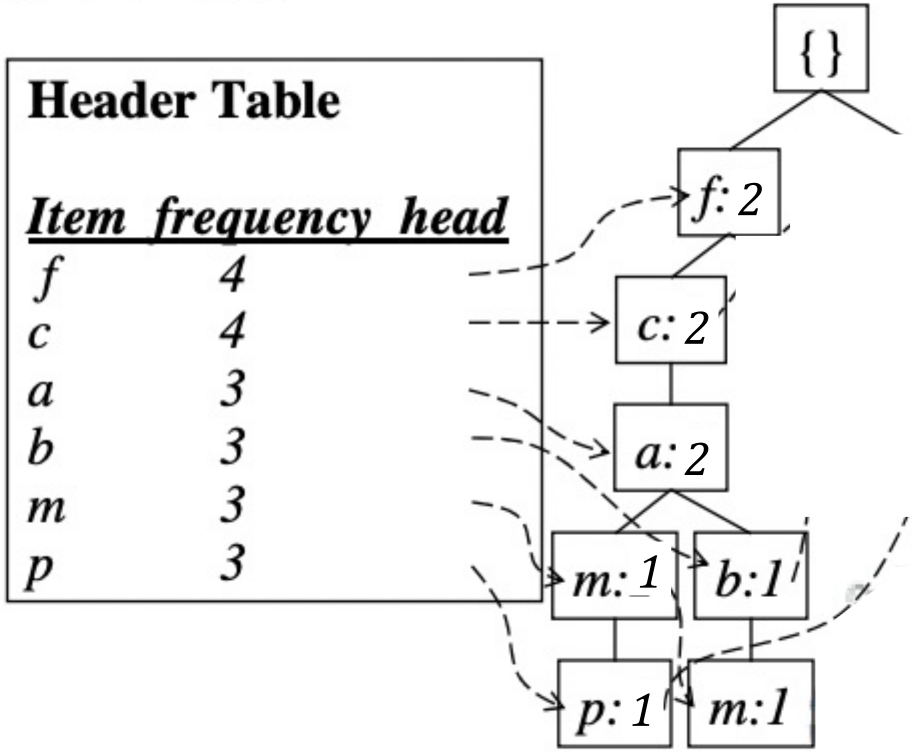
How to construct a FP-tree

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 = 3

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	

How to construct a FP-tree

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}*

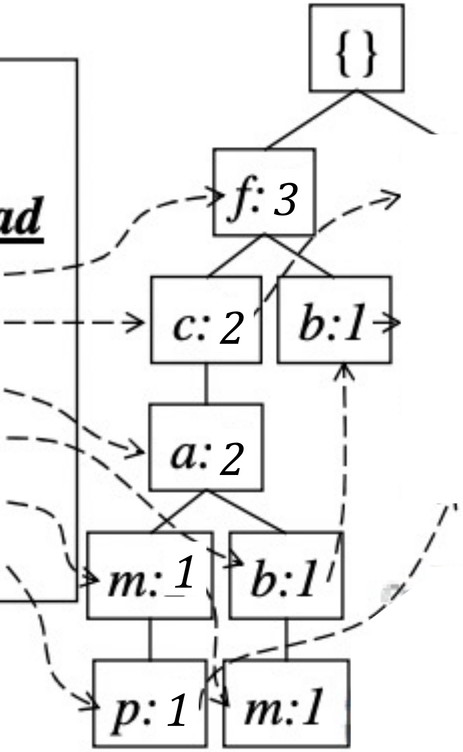
min_support = 3

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*	



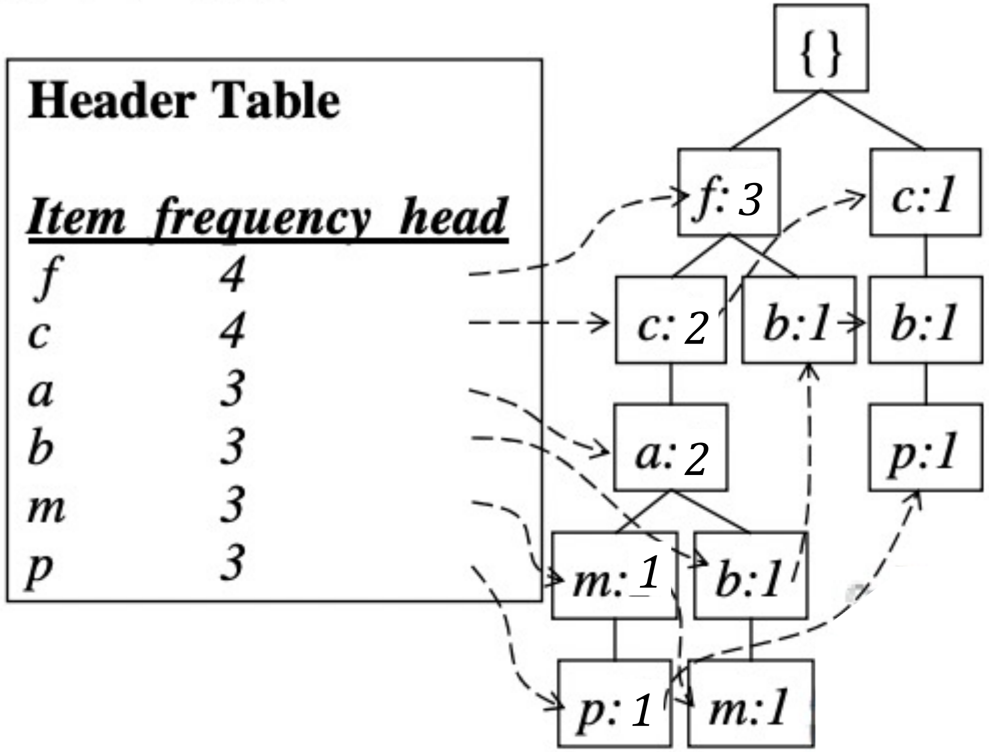
How to construct a FP-tree

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 = 3

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	

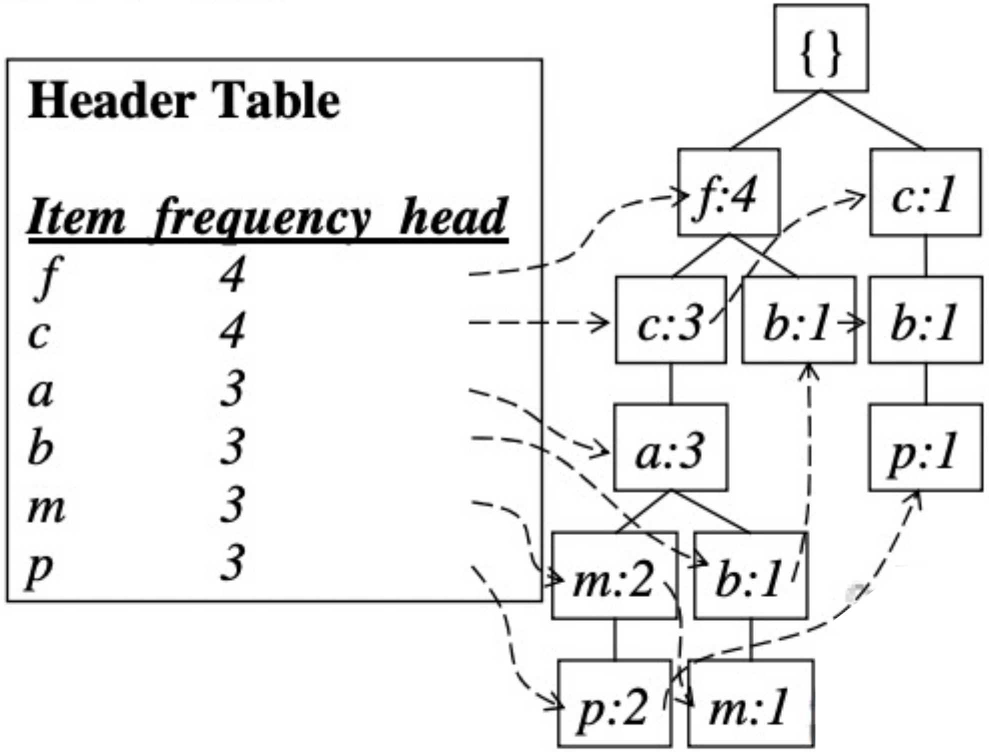
How to construct a FP-tree

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 = 3

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	

Benefits of the FP-tree Structure

- Completeness:
  - never breaks a long pattern of any transaction
  - 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

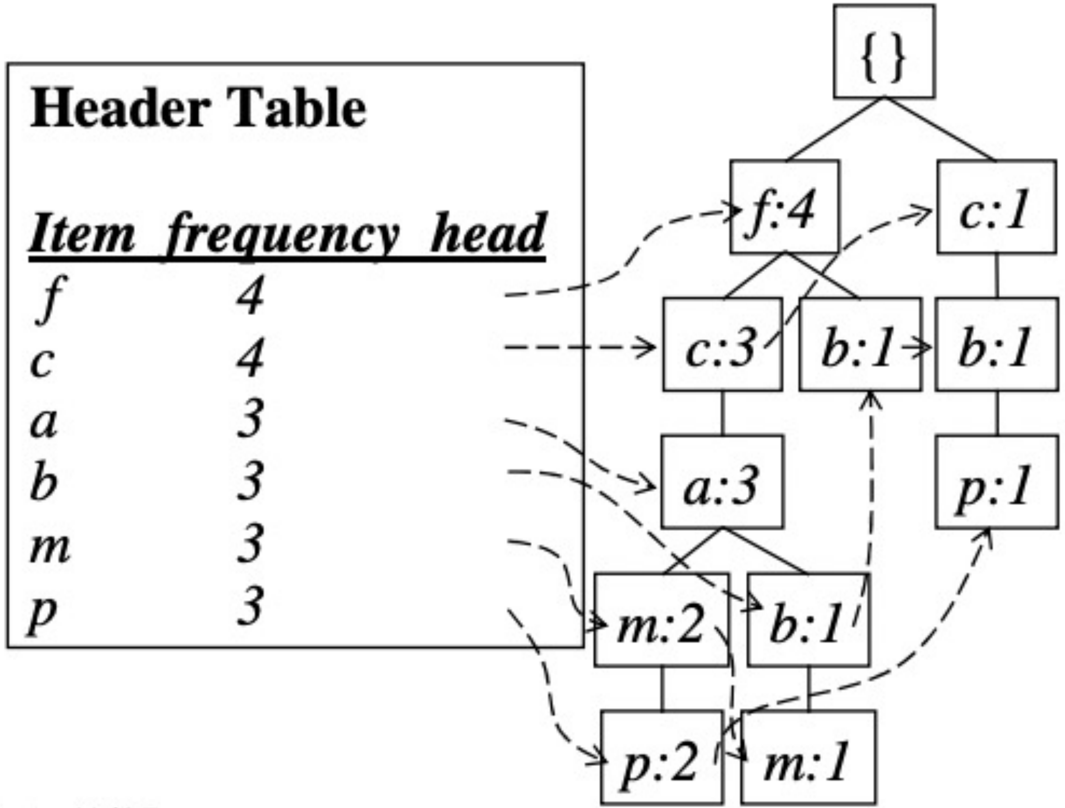
- Method
    - For each item, construct its *conditional pattern-base*, and then its *conditional FP-tree*
    - Repeat the process on each newly created conditional FP-tree
    - Until the resulting FP-tree is *empty*, or it contains only one path (single path will generate all the combinations of its 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
4. If the conditional FP-tree contains a single path, simply enumerate all the patterns

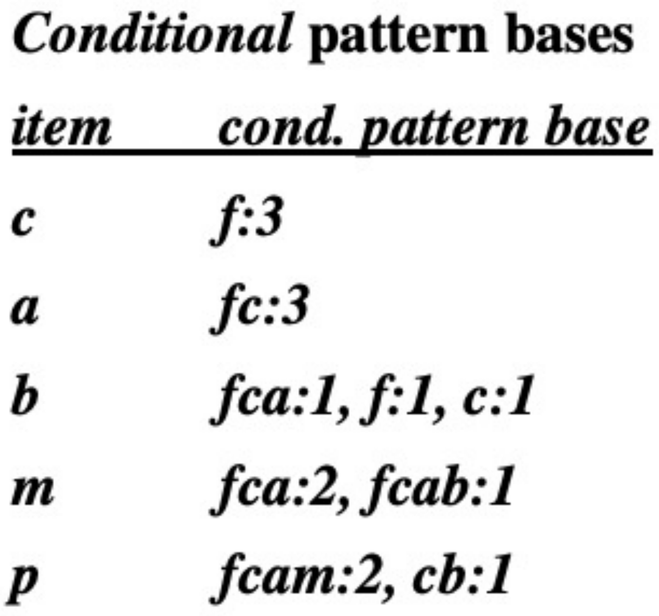
Step 1: From FP-tree to Conditional Pattern Base

- 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**

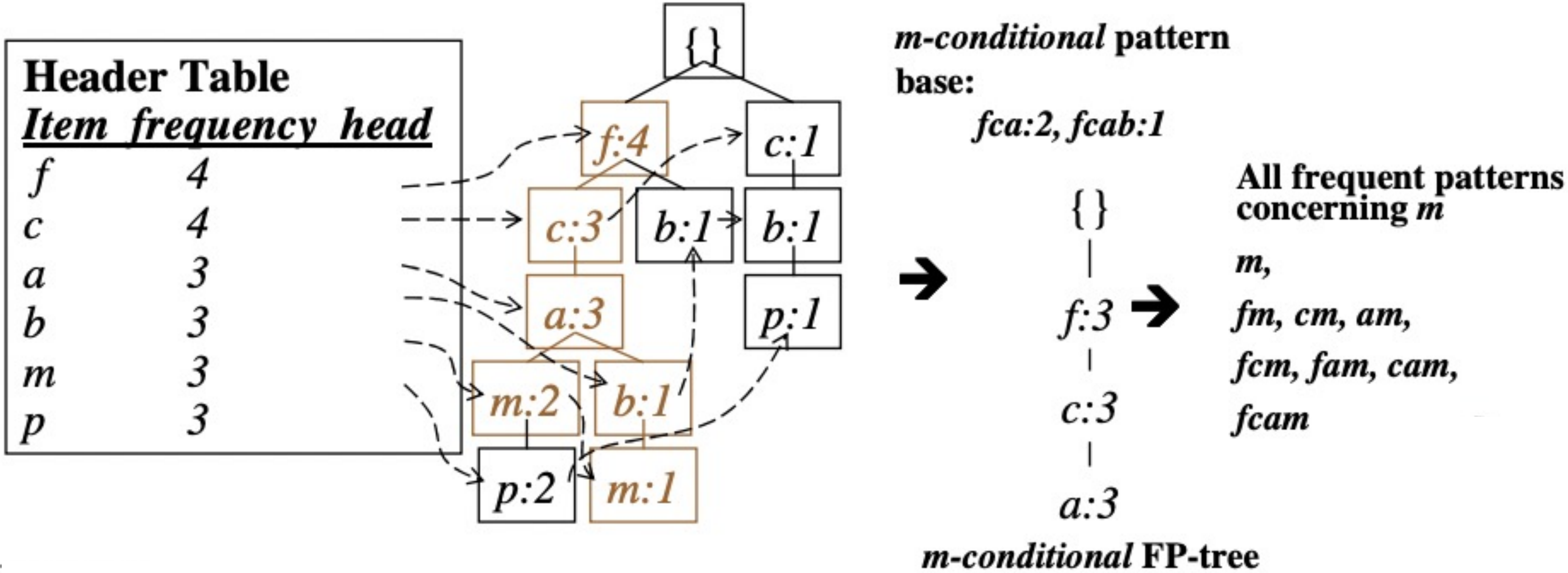
item	*cond. pattern base*
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

Properties of FP-tree for Conditional Pattern Base Construction

- Node-link property: For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header
- Prefix path property: To calculate the frequent patterns for a node a_i in a path *P*, only the prefix sub-path of a_i in *P* need to be accumulated, and its frequency count should carry the same count as node a_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



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,

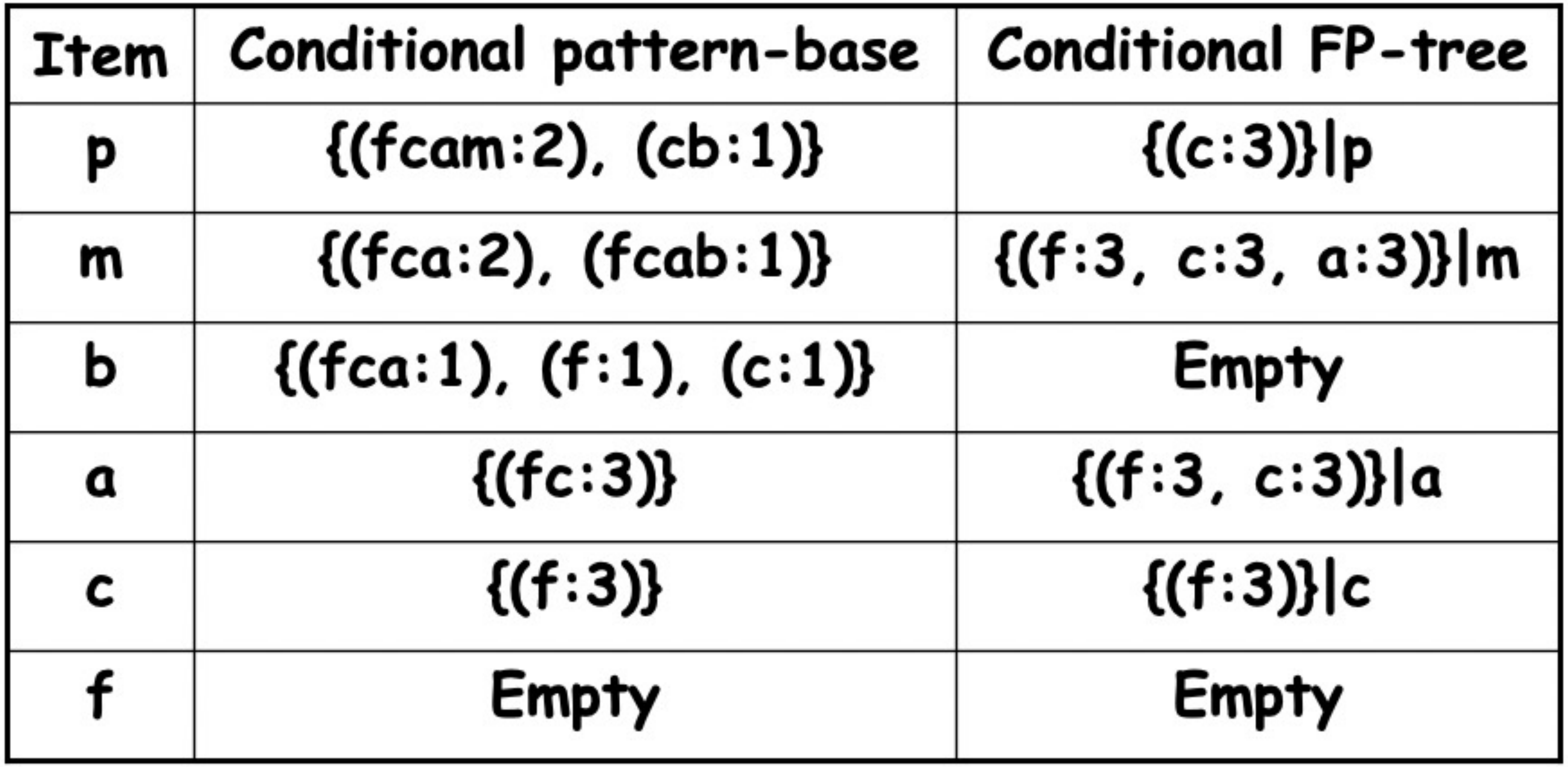
fm, cm, am,

fcm, fam, cam,

fcam

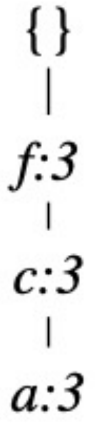
m-conditional **FP-tree**

Mining Frequent Patterns by Creating Conditional Pattern Bases



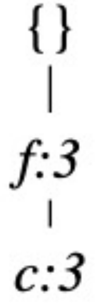
Item	Conditional pattern-base	Conditional FP-tree
p	{(fcam:2), (cb:1)}	{(c:3)}|p
m	{(fca:2), (fcab:1)}	{(f:3, c:3, a:3)}|m
b	{(fca:1), (f:1), (c:1)}	Empty
a	{(fc:3)}	{(f:3, c:3)}|a
c	{(f:3)}	{(f:3)}|c
f	Empty	Empty

Step 3: recursively mine the conditional FP-tree



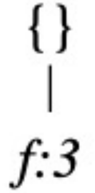
m-conditional **FP-tree**

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



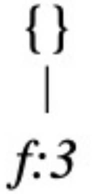
am-conditional **FP-tree**

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



cm-conditional **FP-tree**

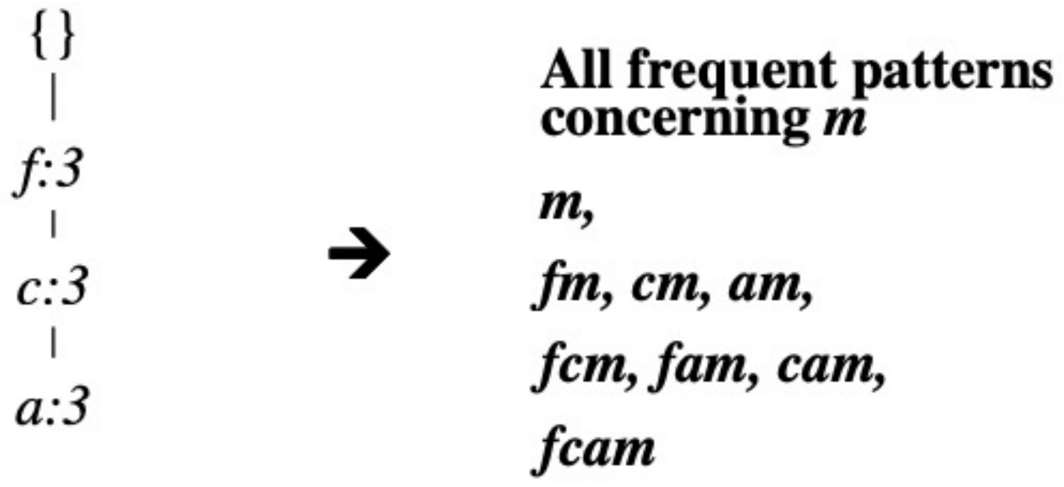
Cond. pattern base of "cam": (f:3)



cam-conditional **FP-tree**

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*



All frequent patterns concerning *m*

m,

fm, cm, am,

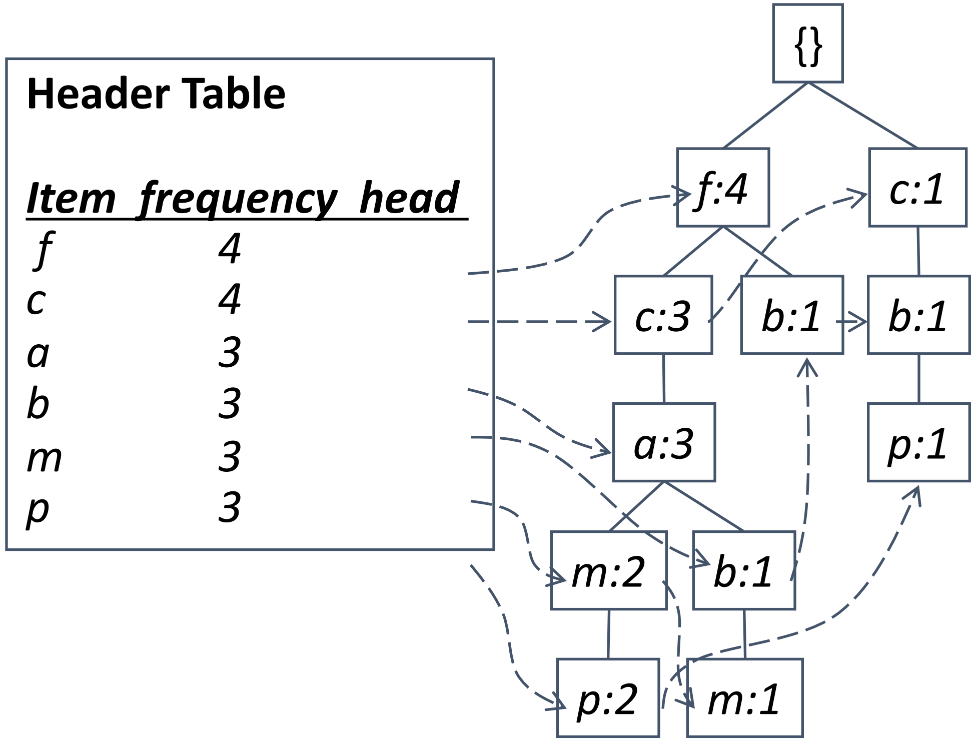
fcm, fam, cam,

fcam

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

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



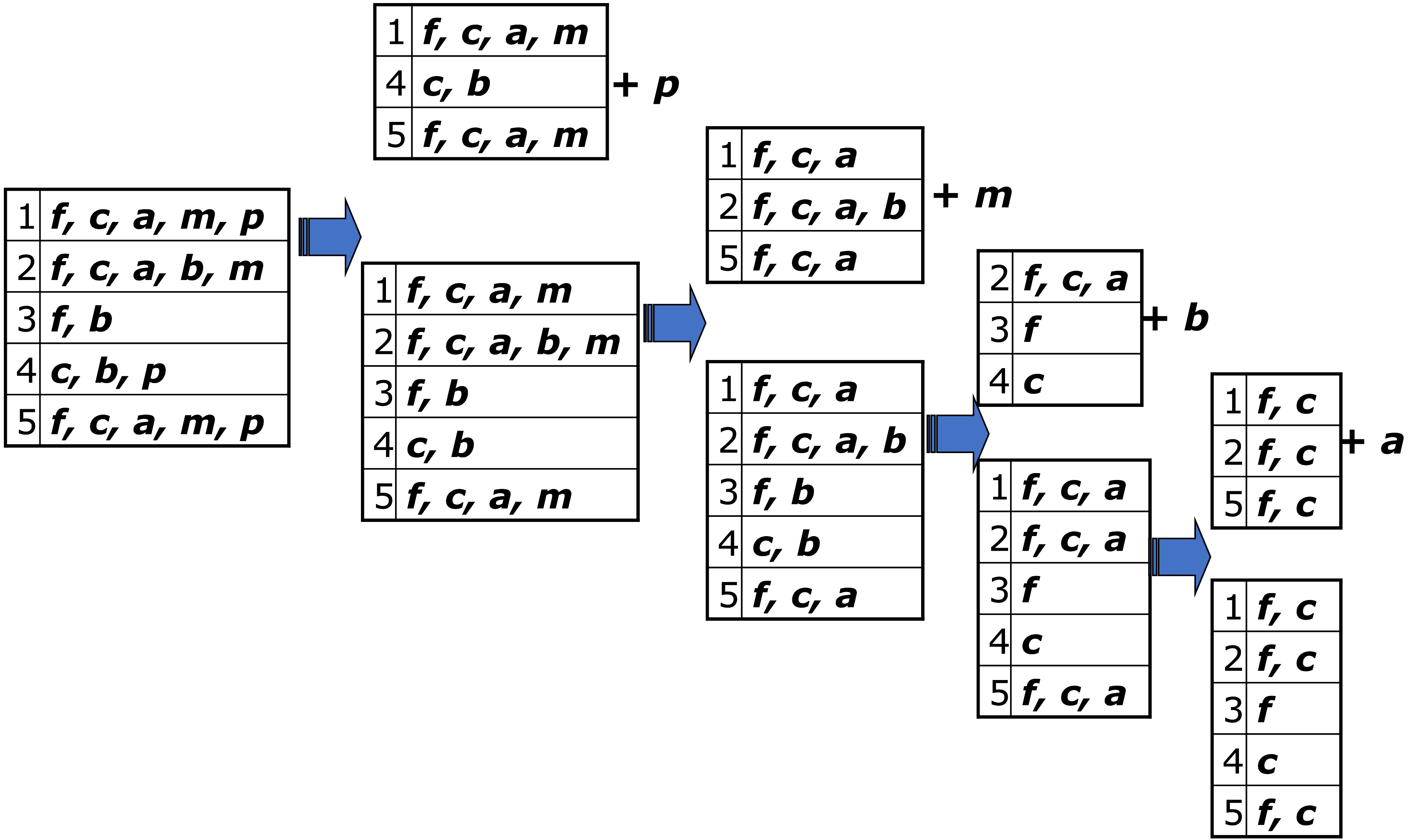
Header Table

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***

FP-Growth



1	***f, c, a, m, p***
2	***f, c, a, b, m***
3	***f, b***
4	***c, b, p***
5	***f, c, a, m, p***

1	***f, c, a, m***
4	***c, b***
5	***f, c, a, m***

+ *p*

1	***f, c, a, m***
2	***f, c, a, b, m***
3	***f, b***
4	***c, b***
5	***f, c, a, m***

1	***f, c, a***
2	***f, c, a, b***
5	***f, c, a***

+ *m*

1	***f, c, a***
2	***f, c, a, b***
3	***f, b***
4	***c, b***
5	***f, c, a***

2	***f, c, a***
3	***f***
4	***c***

+ *b*

1	***f, c, a***
2	***f, c, a***
3	***f***
4	***c***
5	***f, c, a***

1	***f, c***
2	***f, c***
5	***f, c***

+ *a*

1	***f, c***
2	***f, c***
3	***f***
4	***c***
5	***f, c***

FP-Growth

1	***f, c, a, m, p***
2	***f, c, a, b, m***
3	***f, b***
4	***c, b, p***
5	***f, c, a, m, p***

1	***f, c, a, m***
4	***c, b***
5	***f, c, a, m***

+ *p*

(1)

1	***f, c, a***
2	***f, c, a, b***
5	***f, c, a***

+ *m*

(2)

2	***f, c, a***
3	***f***
4	***c***

+ *b*

(3)

1	***f, c***
2	***f, c***
5	***f, c***

+ *a*

(4)

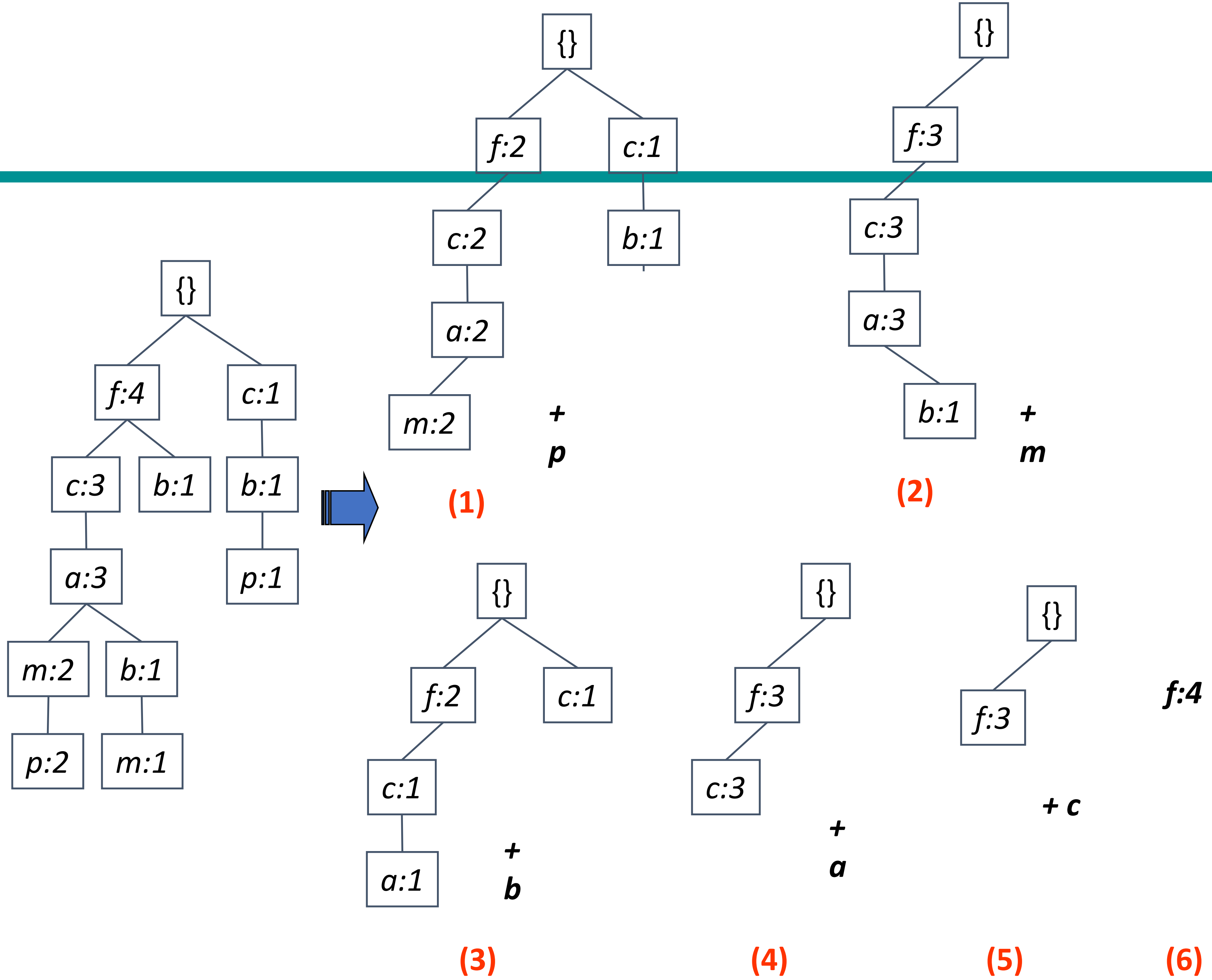
1	***f***
2	***f***
4	
5	***f***

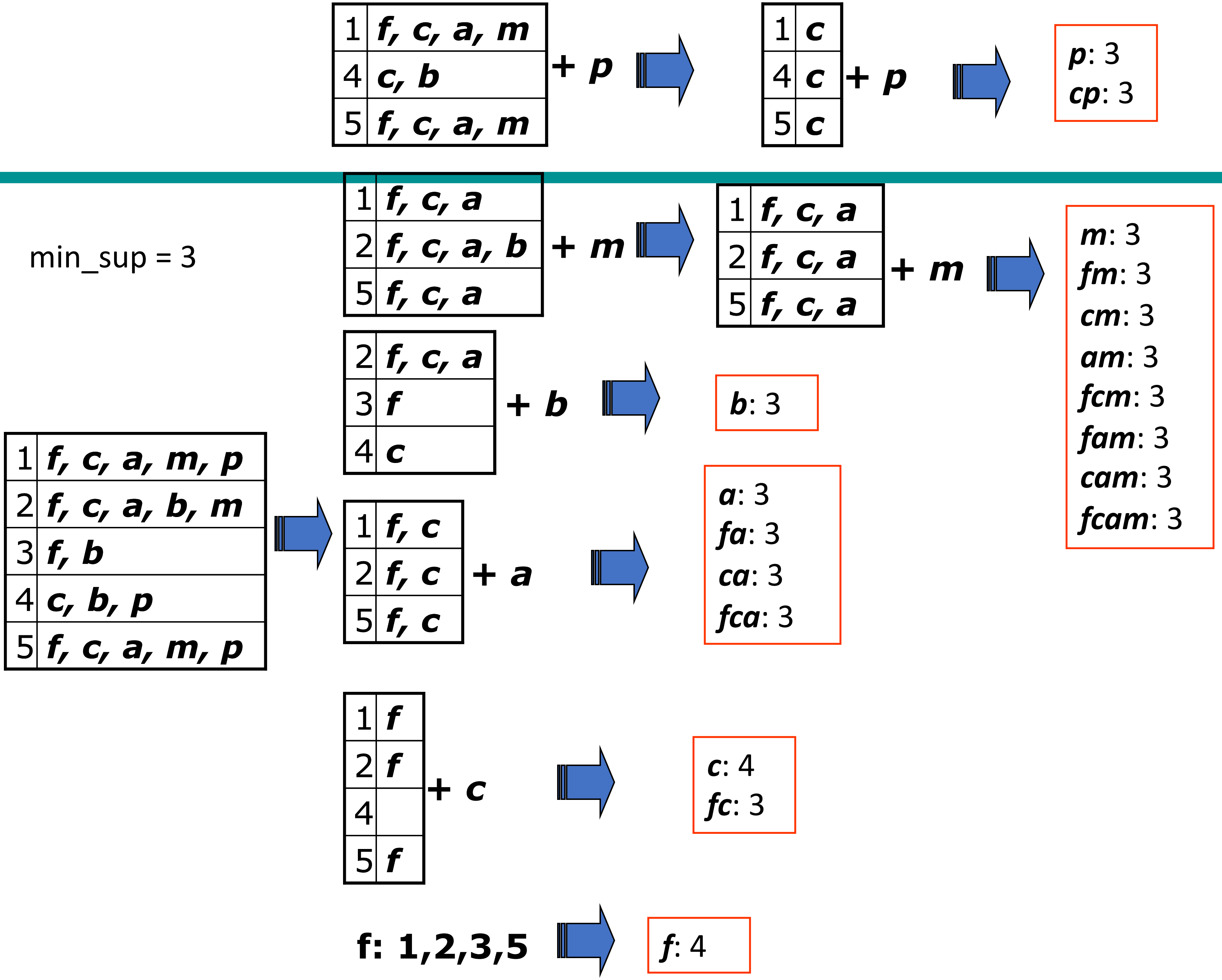
+ *c*

(5)

f: 1,2,3,5

(6)


{}
f:4
c:1
c:3
b:1
b:1
a:3
p:1
m:2
b:1
p:2
m:1
{}
f:2
c:1
c:2
b:1
a:2
m:2
+
p
(1)
{}
f:3
c:3
a:3
b:1
+
m
(2)
{}
f:2
c:1
c:1
a:1
+
b
(3)
{}
f:3
c:3
+
a
(4)
{}
f:3
+ c
(5)
f:4
(6)



min_sup = 3

1	***f, c, a, m, p***
2	***f, c, a, b, m***
3	***f, b***
4	***c, b, p***
5	***f, c, a, m, p***

1	***f, c, a, m***
4	***c, b***
5	***f, c, a, m***

+ *p*

1	***c***
4	***c***
5	***c***

+ *p*

p: 3
cp: 3

1	***f, c, a***
2	***f, c, a, b***
5	***f, c, a***

+ *m*

1	***f, c, a***
2	***f, c, a***
5	***f, c, a***

+ *m*

m: 3
fm: 3
cm: 3
am: 3
fcm: 3
fam: 3
cam: 3
fcam: 3

2	***f, c, a***
3	***f***
4	***c***

+ *b*

b: 3

1	***f, c***
2	***f, c***
5	***f, c***

+ *a*

a: 3
fa: 3
ca: 3
fca: 3

1	***f***
2	***f***
4	
5	***f***

+ *c*

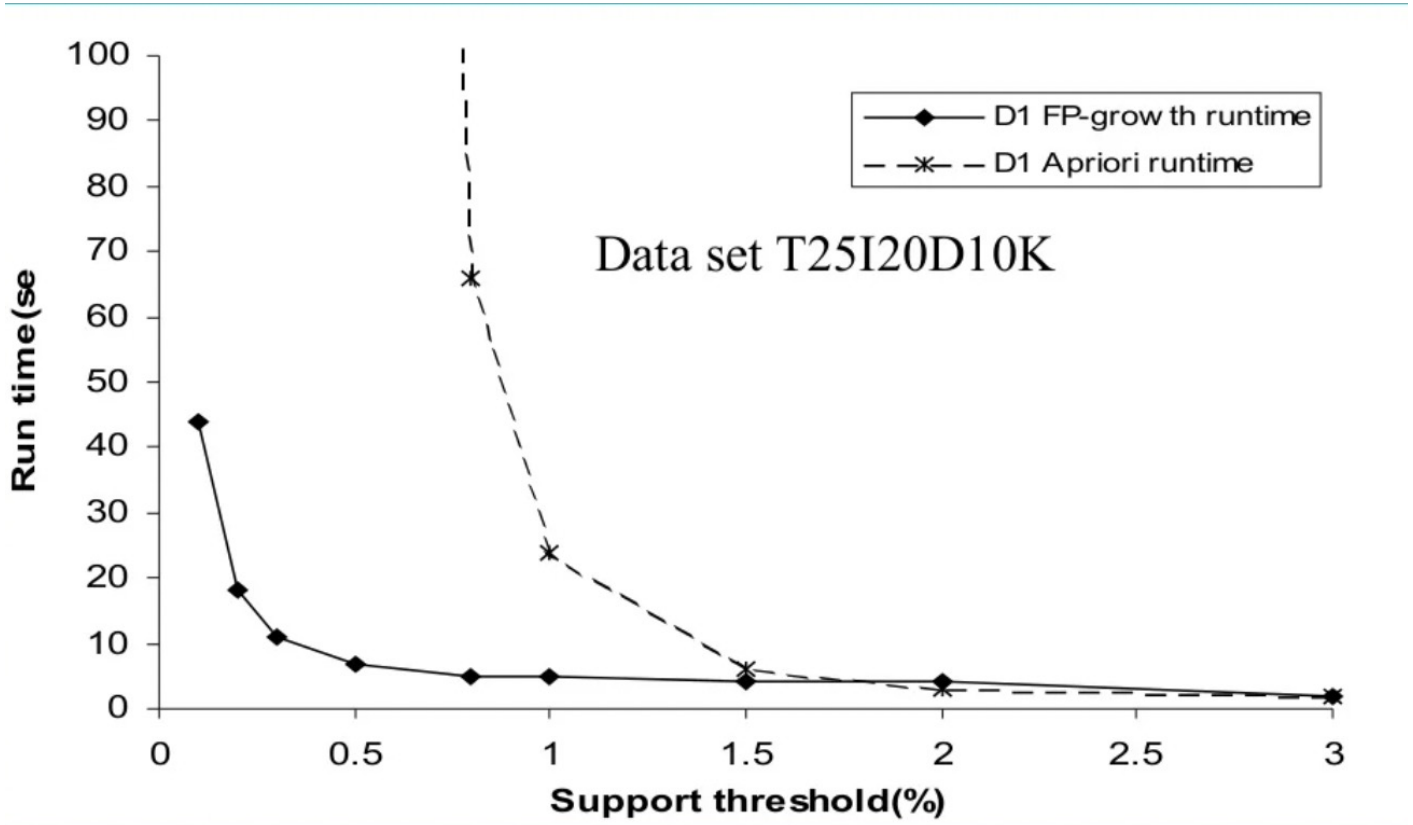
c: 4
fc: 3

f: 1,2,3,5

f: 4

Why is FP-Growth Fast?

- FP-Growth is an order of magnitude faster than Apriori
  - No candidate generation, no candidate test
  - Use compact data structure
  - Eliminate repeated dataset scan
  - Basic operation is counting and FP-tree building



#Transactions	Items	Average Transaction Length
250,000	1000	12

References

- Pattern Mining. Chapter 5. Introduction to Data Mining.