Subject : Data Engineering Concepts

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

(Computer Engineering and Technology)
(TYB.Tech)

Unit IV: **Association Rules Mining**

Market basket Analysis, Frequent item set, Closed item set, Association Rules, a-priori Algorithm, Generating Association Rules from Frequent Item sets, Improving the Efficiency of a-priori, Mining Frequent Item sets without Candidate Generation: FP Growth Algorithm; Mining ,Generating Rules

Introduction : Frequent Patterns

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of **frequent item sets and association rule mining**
- Motivation: Finding inherent regularities in data

  What products were often **purchased together**?

  What are the **subsequent purchases after buying a PC**?

  What kinds of **DNA are sensitive to this new drug**?
- **Applications**

  Market basket analysis, cross-marketing, catalog design, sales campaign analysis, Web log (click stream) analysis, and DNA sequence analysis

Market Basket Analysis

Consider shopping cart filled with several items

Market basket analysis tries to answer the following questions:

Who makes purchases?

What do customers buy together?

In what order do customers purchase items?

Introduction to Market Basket Analysis

Def: Market Basket Analysis (Association Analysis) is a mathematical modeling technique based upon the theory that if you buy a certain group of items, you are likely to buy another group of items.

It is **used to analyze the customer purchasing behavior** and **helps in increasing the sales and maintain inventory by focusing on the point of sale transaction data.**

Given a dataset, the Apriori Algorithm trains and identifies product baskets and product association rules

Definitions and Terminology

Transaction is a set of items (Itemset).

Confidence : It is the measure of uncertainty or trust worthiness associated with each discovered pattern.

Support : It is the measure of how often the collection of items in an association occur together as percentage of all transactions

Frequent itemset : If an itemset satisfies minimum support, then it is a frequent itemset.

Strong Association rules: Rules that satisfy both a minimum support threshold and a minimum confidence threshold

In Association rule mining, **we first find all frequent itemsets and then generate strong association rules from the frequent itemsets**

What Is Frequent Pattern Analysis? Introduction

- Frequent pattern: a pattern (a set of items, 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 butter?
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Closed itemset

It is a frequent itemset that has no proper superset with the same support. In other words, a closed itemset is the largest itemset in a set of itemsets with the same support.
For example, consider the following transaction database:

The minimum support (minsup) is set to 3. The frequent itemsets in this database are:
{A}, {B}, {C}, {D}, {A, B}, {A, C}, {A, D}, {B, C}, {B, D}, {C, D}, and {A, B, C}, {A, B, D}, {A, C, D}, and {B, C, D}
However, only the following itemsets are closed:
{A}, {B}, {C}, {D}, {A, B}, {A, C}, {A, D}, {B, C}, and {B, D}

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

- Closed itemsets are useful for association rule mining because they can be used to
- generate all frequent itemsets and their supports. Additionally, closed itemsets are
- often more concise and easier to interpret than frequent itemsets.
- Here are some examples of closed itemsets in different industries:
- Retail: {bread, milk}, {shirt, jeans}, {laptop, printer}
- Healthcare: {fever, cough}, {diabetes, heart disease}, {cancer, chemotherapy}

Association rule mining

- Proposed by Agrawal et al in 1993.
- It is an important data mining model studied extensively by the database and data mining community.
- Assume all data are categorical.
- No good algorithm for numeric data.
- Initially used for Market Basket Analysis to find how items purchased by customers are related.

Bread $\rightarrow$ Milk [sup = 5%, conf = 100%]

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

Market Basket Analysis General Concept: Measures

- **Measures:**
  - **Suport**

Suport = (containing the item combination) /(total number of record.)

Let the rule Is "If a customer purchases Cola, then they will purchase Frozen Pizza"

The support for this

= 2 (number of transaction that include both Cola and Frozen Pizza is) / 5(total records)

= 40%.

  - **Confidence:**

Confidence of a rule = the support for the combination / the support for the condition.

For the rule "If a customer purchases Milk, then they will purchase Potato Chips"

confidence = support for the combination (Potato Chips + Milk) is 20%/ support for the condition (Milk) is 60%,

=33%

Support and confidence are used to select the association rules.

What Is Association Rule Mining?

Frequent patterns: patterns (set of items, sequence, etc.) that occur frequently in a database

Frequent pattern mining: finding regularities in data

What products were often purchased together?

What are the subsequent purchases after buying a car?

Can we automatically profile customers?

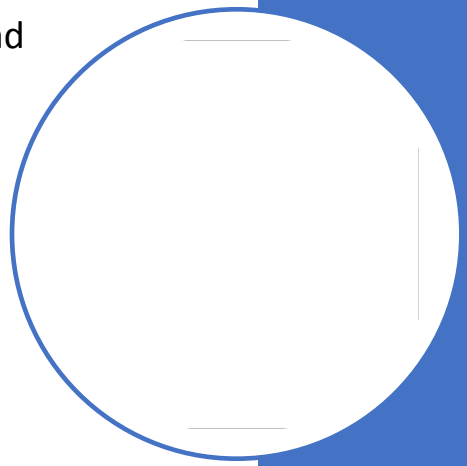
Association Rules

"An association algorithm creates rules that describe how often events have occurred together."

Example: When a customer buys a hammer, then 90% of the time they will buy nails.

Apriori Introduction

- Frequent pattern: a pattern (a set of items, 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**?— Pen and Pencil?!
- What are the **subsequent purchases after buying a PC**?
- What kinds of **DNA are sensitive to this new drug**?
- Can we automatically **classify web documents**?
- **Applications**
- Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis



Examples: 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},
{Beer, Bread} → {Diaper},

Implication means co-occurrence, not causality!

Terms & Definitions: Examples

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count/count/frequency (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 *min_sup* 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 Example

- 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 **all** transactions that contain both X and Y
  - Confidence (c)
    - Measures how often Y appears in transactions that contain X

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

Example:

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

Definition: Association Rule Example

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

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

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

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

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

- Support: “is a measure of what fraction of the population satisfies both the antecedent and the consequent of the rule”.
- Example:
  - People who buy hotdog buns also buy hotdog sausages in 99% of cases. = High Support
  - People who buy hotdog buns buy hangers in 0.005% of cases. = Low support

- Situations where there is high support for the antecedent are worth careful attention
  - E.g. Hotdog sausages should be placed in near hotdog buns in supermarkets if there is also high confidence.

support(X=>Y) = P(XUY)

Association Rules

- Confidence: “is a measure of how often the consequent is true when the antecedent is true.”
- Example:
  - 90% of Hotdog bun purchases are accompanied by hotdog sausages.
  - High confidence is meaningful as we can derive rules.
- Hotdog bun□ Hotdog sausage
- 2 rules may have different confidence levels and have the same support.
- E.g. Hotdog sausage □ Hotdog bun may have a much lower confidence than Hotdog bun □ Hotdog sausage yet they both can have the same support.

Confidence(X=>Y)=P(Y|X)

mset	Temperature	Wind	Humidity	Play
	Warm	Calm	Dry	Yes
	Cold	Calm	Dry	Yes
	Cold	Windy	Raining	No
	Cold	Gale	Dry	No
	Cold	Calm	Raining	No

1. {Cold, Raining} => No
2. {Calm, Dry} => Yes
3. {Dry} => No
4. {Windy} => No

{Cold, Raining} => No

- Support: 2/5 = 40%
- Confidence: 2/2 = 100%
- => Good

{Calm, Dry} => Yes

- Support: 2/5 = 40%
- Confidence: 2/2 = 100%
- => Good

{Dry} => No

- Support: 3/5 = 60%
- Confidence: 1/3 = 33.3%
- => Bad

{Windy} => No

- Support: 1/5 = 20%
- Confidence: 1/1 = 100%
- =>Bad

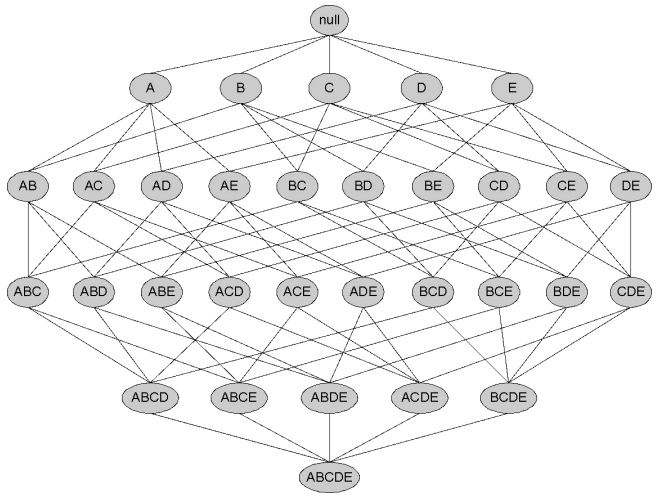
Association Rule Mining Task

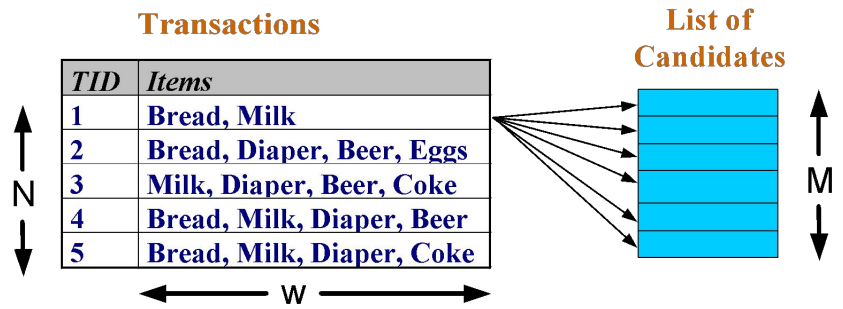
- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- 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

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
  - Match each transaction against every candidate
  - Expensive !!!





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

Mining Association Rules

- Two-step approach:
  - Frequent Itemset Generation
    - Generate all itemsets whose support ≥ minsup
  - 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 computationally expensive

Frequent Itemset Generation Strategies

To Reduce the number of candidates (M) Complete search: $M=2^d$ use pruning techniques

Reducing Number of Candidates

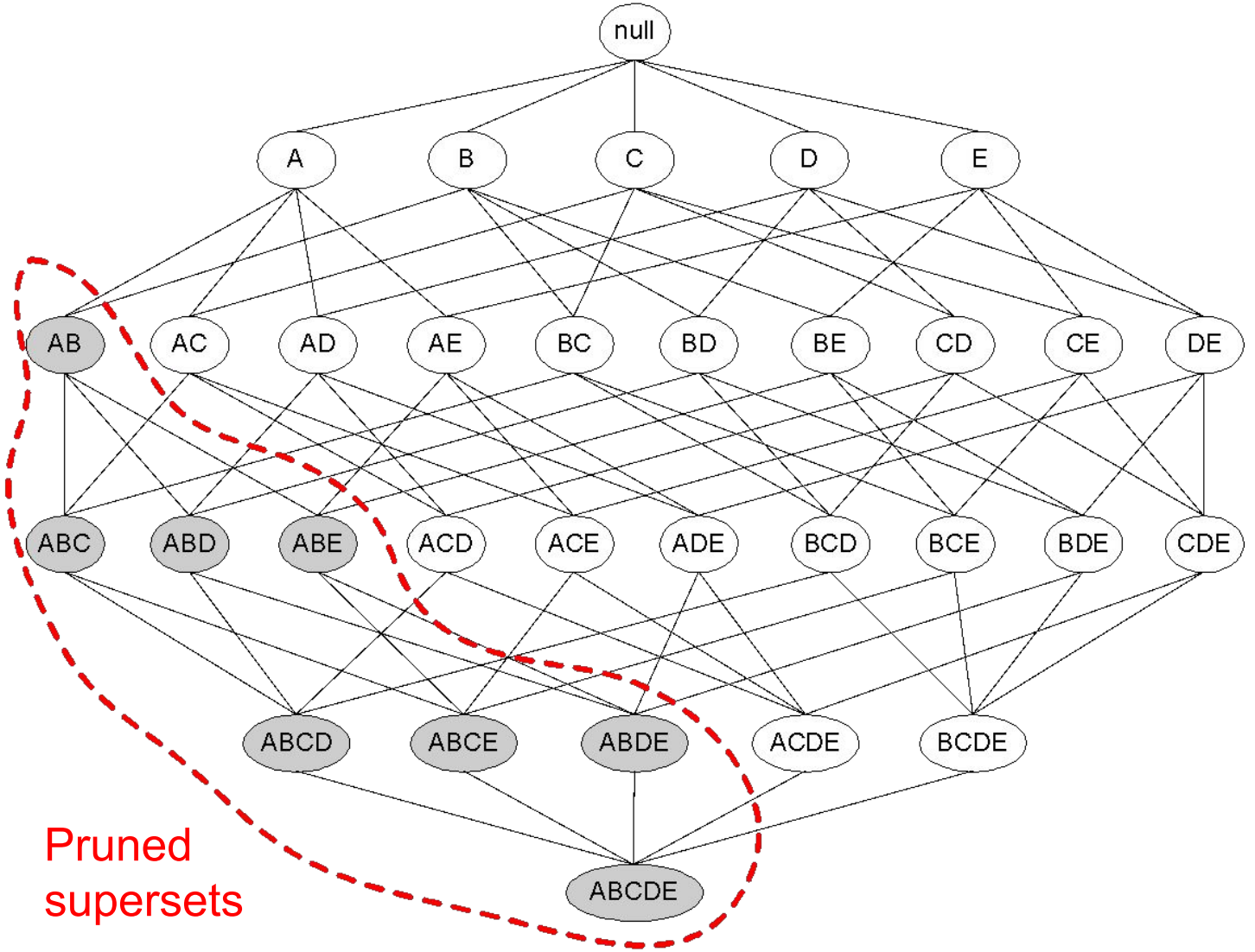
$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- 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:

Support of an itemset never exceeds the support of its subsets.

- This is known as the anti-monotone property of support.

Illustrating Apriori Principle



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
    - **Support Counting**: Count the support of each candidate in L_{k+1} by scanning the DB
    - **Candidate Pruning: Prune candidate itemsets in Lk+1 containing subsets of length k that are infrequent .** Eliminate candidates in L_{k+1} that are infrequent, leaving only those that are frequent => F_{k+1}

Frequent itemset Generation Method

HINT : Merge two frequent itemsets with size n if their first n-1 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

Apriori Algorithm

The Apriori Algorithm: Basics

The Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules.

Key Concepts :

- Frequent Itemsets: The sets of item which has minimum support (denoted by L_i for i^{th}-Itemset).
- Apriori Property: Any subset of frequent itemset must be frequent.
- Join Operation: To find L_k , a set of candidate k-itemsets is generated by joining L_{k-1} with itself.

The Apriori Algorithm (Pseudo-Code)

- C_k: Candidate itemset of size k
- L_k : frequent itemset of size k

- L_1 = {frequent items};
- **for** (k = 1; L_k !=∅; k++) **do begin**
- C_{k+1} = candidates generated from L_k;
- **for each** transaction t in database do
  - increment the count of all candidates in C_{k+1} that are contained in t
- L_{k+1} = candidates in C_{k+1} with min_support
- **end**
- **return** $\cup_k L_k$;

Implementation of Apriori

- **How to generate candidates?**
  - **Step 1: self-joining L_k**
  - **Step 2: pruning**
- **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*}**

Apriori Example

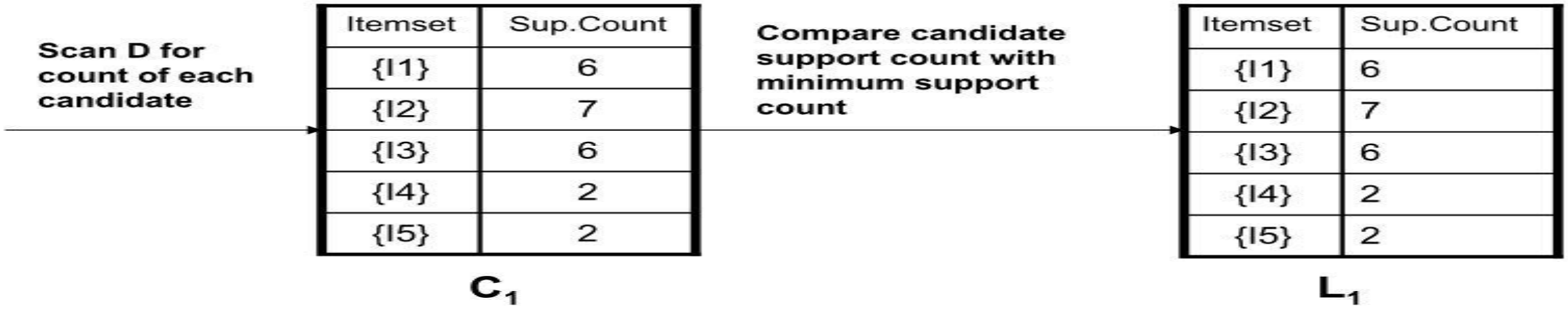
Transactional Data for an *AllElectronics* Branch

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

- Consider a database, D , consisting of 9 transactions.
- Suppose min. support count required is 2 (i.e. min_sup = 2/9 = 22 %)
- Let minimum confidence required is 70%.
- We have to first find out the frequent itemset using Apriori algorithm.
- Then, Association rules will be generated using min. support & min. confidence.

Example –Step 1

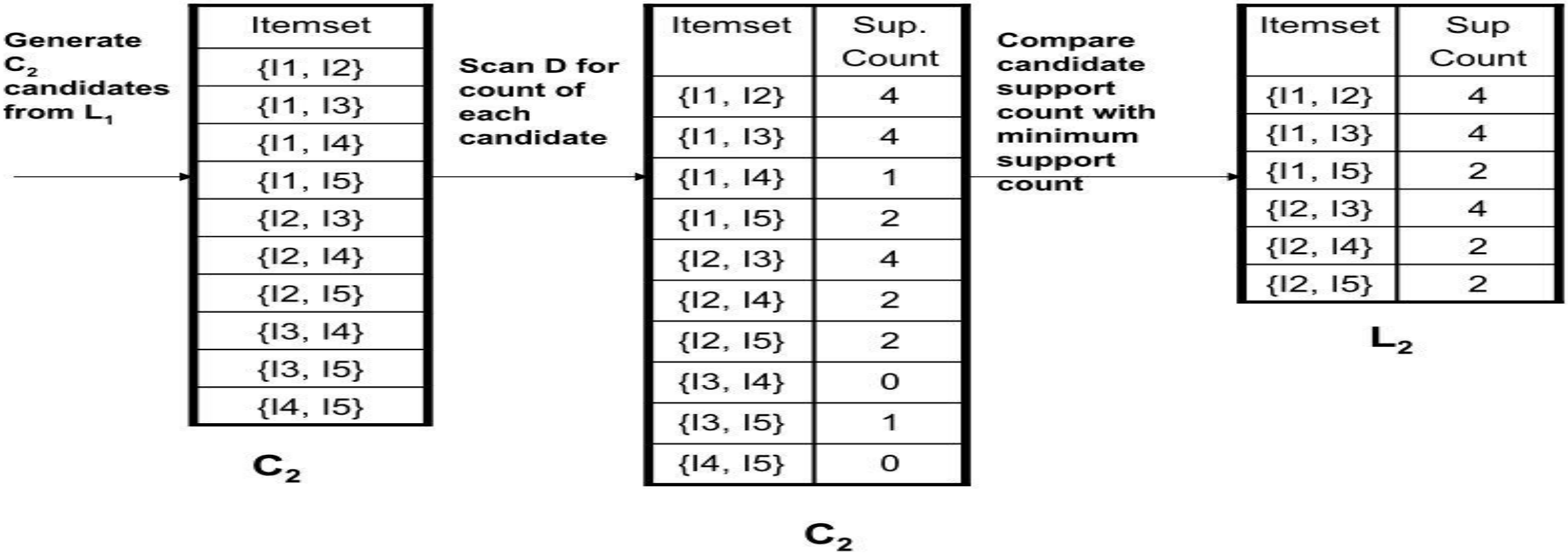
Step 1: Generating 1-itemset Frequent Pattern



- The set of frequent 1-itemsets, L_1 , consists of the candidate 1-itemsets satisfying minimum support.
- In the first iteration of the algorithm, each item is a member of the set of candidate.

Example – Step 2

Step 2: Generating 2-itemset Frequent Pattern

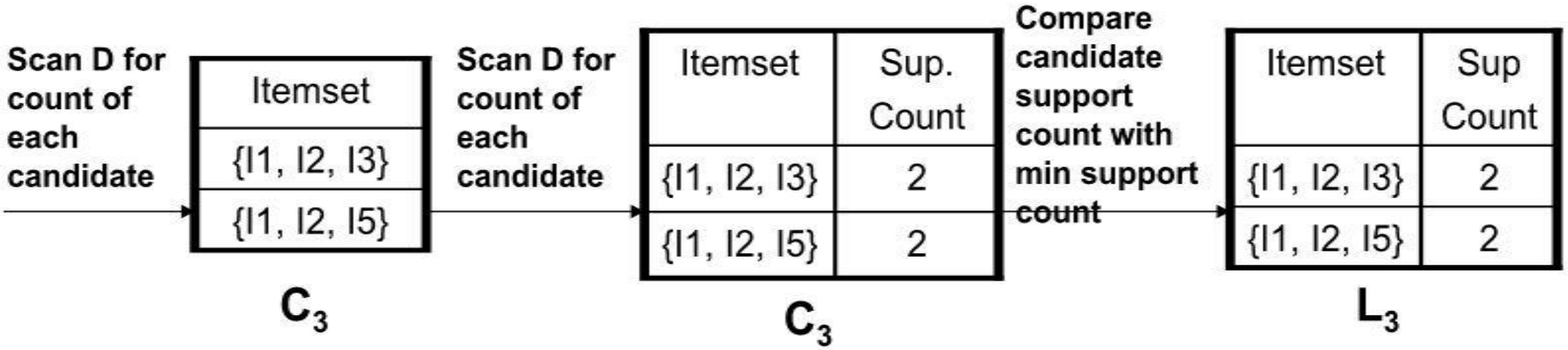


Example – Step 2

Step 2: Generating 2-itemset Frequent Pattern

- To discover the set of frequent 2-itemsets, L_2 , the algorithm uses L_1 *Join* L_1 to generate a candidate set of 2-itemsets, C_2.
- Next, the transactions in D are scanned and the support count for each candidate itemset in C_2 is accumulated (as shown in the middle table).
- The set of frequent 2-itemsets, L_2 , is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.
- Note: We haven't used Apriori Property yet.

Example – Step 3



- The generation of the set of candidate 3-itemsets, C_3 , involves use of the Apriori Property.
- In order to find C_3, we compute L_2 *Join* L_2.
- C_3 = L2 *Join* L2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}

Step 3: Generating 3-itemset Frequent Pattern

- Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that four latter candidates cannot possibly be frequent. How ?
- For example , lets take {I1, I2, I3}. The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L_2, We will keep {I1, I2, I3} in C_3.
- Lets take another example of {I2, I3, I5} which shows how the pruning is performed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3,I5}.
- BUT, {I3, I5} is not a member of L_2 and hence it is not frequent violating Apriori Property. Thus We will have to remove {I2, I3, I5} from C_3.
- Therefore, C_3 = {{I1, I2, I3}, {I1, I2, I5}} after checking for all members of result of Join operation for Pruning.
- Now, the transactions in D are scanned in order to determine L_3, consisting of those candidates 3-itemsets in C_3 having minimum support.

Example – Step 4

Step 4: Generating 4-itemset Frequent Pattern

- The algorithm uses L_3 *Join* L_3 to generate a candidate set of 4-itemsets, C_4. Although the join results in {{I1, I2, I3, I5}}, this itemset is pruned since its subset {{I2, I3, I5}} is not frequent.
- Thus, $C_4 = \varphi$, and algorithm terminates, having found all of the frequent items. This completes our Apriori Algorithm.
- What's Next ?

  These frequent itemsets will be used to generate strong association rules (where strong association rules satisfy both minimum support & minimum confidence).

Example – Step 5

Step 5: Generating Association Rules from Frequent Itemsets

- Procedure:
  - For each frequent itemset ***"l"***, generate all nonempty subsets of ***l***.
  - For every nonempty subset ***s*** of ***l***, output the rule "**s → (l-s)**" if **support_count(l) / support_count(s) >= min_conf** where min_conf is minimum confidence threshold.

- Back To Example:

We had L = {{I1}, {I2}, {I3}, {I4}, {I5}, {I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.

  - Lets take ***l*** = {I1,I2,I5}.
  - Its all nonempty subsets are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}.

- Let minimum confidence threshold is , say 70%.
- The resulting association rules are shown below, each listed with its confidence.
  - R1: I1 ^ I2 → I5
    - Confidence = sc{I1,I2,I5}/sc{I1,I2} = 2/4 = 50%
    - R1 is Rejected.
  - R2: I1 ^ I5 → I2
    - Confidence = sc{I1,I2,I5}/sc{I1,I5} = 2/2 = 100%
    - R2 is Selected.
  - R3: I2 ^ I5 → I1

Step 5: Generating Association Rules from Frequent Itemsets

- R4: I1 → I2 ^ I5
  - Confidence = sc{I1,I2,I5}/sc{I1} = 2/6 = 33%
  - R4 is Rejected.
- R5: I2 → I1 ^ I5
  - Confidence = sc{I1,I2,I5}/{I2} = 2/7 = 29%
  - R5 is Rejected.
- R6: I5 → I1 ^ I2
  - Confidence = sc{I1,I2,I5}/ {I5} = 2/2 = 100%
  - R6 is Selected.

  In this way, We have found three strong association rules.

The following are the main steps of the apriori algorithm in data mining:

- Set the minimum support threshold - min frequency required for an itemset to be "frequent".
- Identify frequent individual items - count the occurence of each individual item.
- Generate candidate itemsets of size 2 - create pairs of frequent items discovered.
- Prune infrequent itemsets - eliminate itemsets that do no meet the threshold levels.
- Generate itemsets of larger sizes - combine the frequent itemsets of size 3,4, and so on.
- Repeat the pruning process - keep eliminating the itemsets that do not meet the threshold levels.
- Iterate till no more frequent itemsets can be generated.
- Generate association rules that express the relationship between them - calculate measures to evaluate the strength & significance of these rules.

- **Improve the Apriori Algorithm's efficiency**
- Many methods are available for improving the efficiency of apriori algorithm.
- **Hash-Based Technique**: This method uses a hash-based structure called a hash table for generating the k-itemsets and their corresponding count. It uses a hash function for generating the table.
- **Transaction Reduction**: This method reduces the number of transactions scanned in iterations. The transactions which do not contain frequent items are marked or removed.
- **Partitioning**: This method requires only two database scans to mine the frequent itemsets. It says that for any itemset to be potentially frequent in the database, it should be frequent in at least one of the partitions of the database.
- **Sampling**: This method picks a random sample S from Database D and then searches for frequent itemset in S. It may be possible to lose a global frequent itemset. This can be reduced by lowering the min_sup.
- **Dynamic Itemset Counting**: This technique can add new candidate itemsets at any marked start point of the database during the scanning of the database.

Exercises & Solutions

Exercise 01

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

Illustrating Apriori Principle : Candidate Generation for n=1

Minimum Support = 3

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)

Items
Bread
Coke
Milk
Beer
Diaper
Eggs

Illustrating Apriori Principle : Support Counting

Minimum Support = 3

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

Illustrating Apriori Principle : Prune infrequent Candidate

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

Minimum Support = 3

Items (1-itemsets)

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

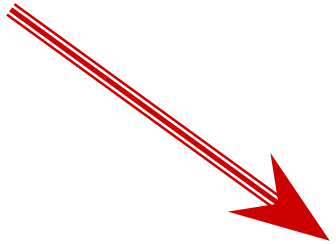
Candidate Generation(n=2)

Minimum Support = 3

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



Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

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

HINT : Merge two frequent itemsets with size n if their first n-1 items are identical

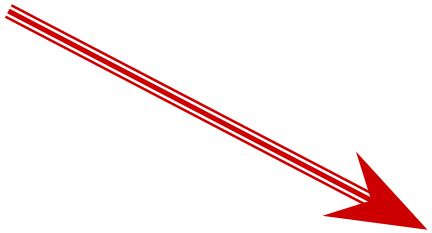
Support Counting

Minimum Support = 3

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



Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

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

Prune infrequent Candidate and Candidate Generation for n=3

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)

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
{Bread, Diaper, Milk}	2

HINT : Merge two frequent itemsets with size n if their first n-1 items are identical

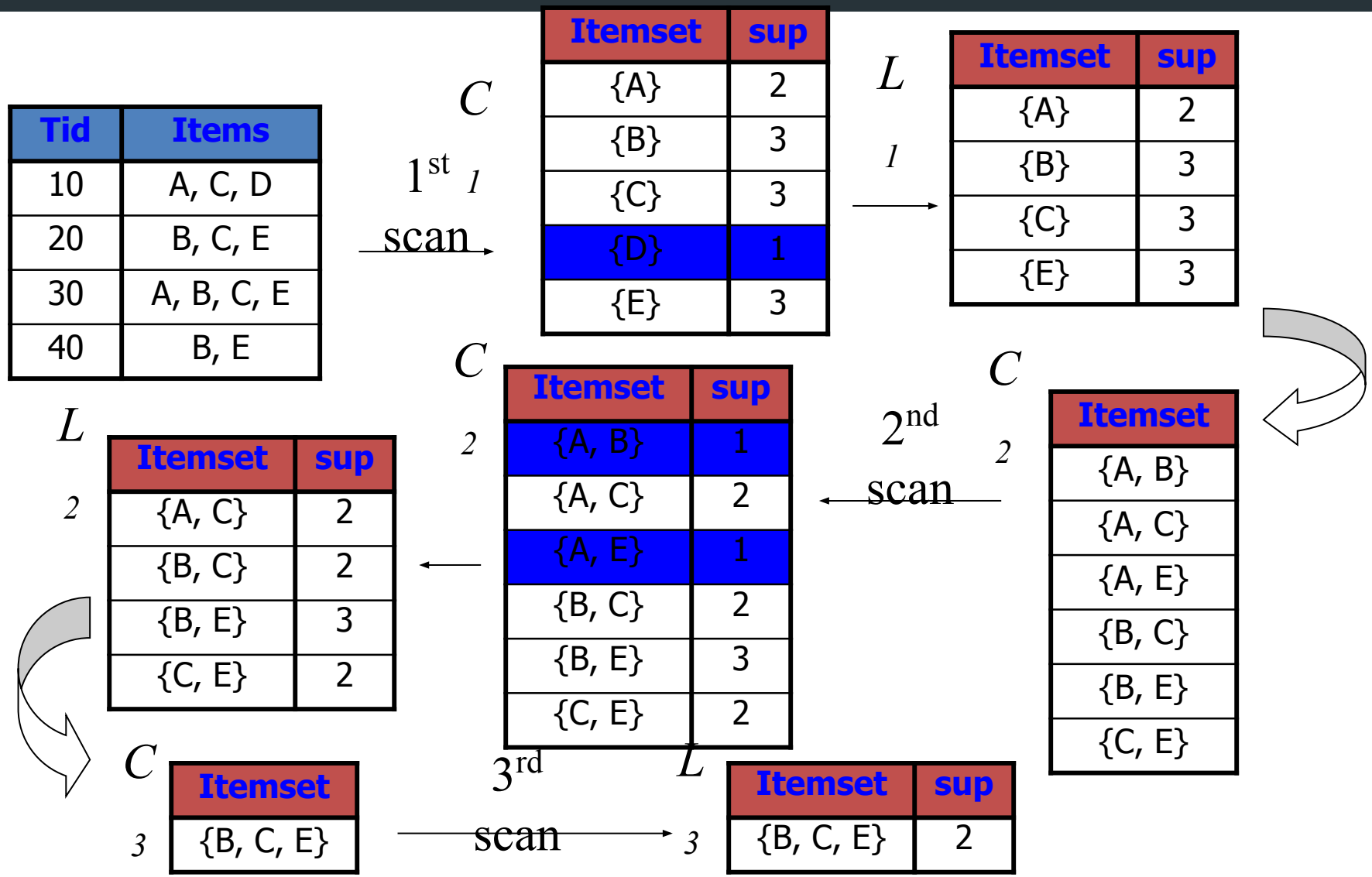
Exercise 02

$Sup_{min} = 2$

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

Example

$Sup_{min} = 2$



Generating rules: an example

- Suppose {2,3,4} is frequent, with sup=50%
  - Proper nonempty subsets: {2,3}, {2,4}, {3,4}, {2}, {3}, {4}, with sup=50%, 50%, 75%, 75%, 75%, 75% respectively
  - These generate these association rules:
    - 2,3 → 4, confidence=100%
    - 2,4 → 3, confidence=100%
    - 3,4 → 2, confidence=67%
    - 2 → 3,4, confidence=67%
    - 3 → 2,4, confidence=67%
    - 4 → 2,3, confidence=67%
    - All rules have support = 50%

Association 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, then 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 $\varnothing \rightarrow L$)

What are the Association Rules for Below, calculate Confidence of Each Rule

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

Itemset	sup
{B, C, E}	2

- BC--> E
- B-->CE
- CE-->B
- E-->BC
- BE-->C

FP Growth

In Data Mining, finding frequent patterns in large databases is very important and has been studied on a large scale.

The FP-Growth Algorithm is an alternative way to find frequent item sets without using candidate generations, thus improving performance.

For so much, it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.

FP Growth Allows frequent itemset discovery without candidate generation

Two step Process

1.Build a compact data structure called the FP-tree : 2 passes over the database [Only !!!]

2.Extracts frequent itemset directly from the FP-tree: Traverse through FP-tree

Use a compressed representation of the database using an FP-tree

Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets.

Building the FP-Tree

1. Scan data to determine the support count of each item.

   Infrequent items are discarded, while the frequent items are sorted in decreasing support counts.

2. Make a second pass over the data to construct the FPtree.

   As the transactions are read, before being processed, their items are sorted according to the above order.

Difference of Apriori and FP Growth

Apriori	FP Growth
Apriori generates frequent patterns by making the itemsets using pairings such as single item set, double itemset, and triple itemset.	FP Growth generates an FP-Tree for making frequent patterns.
Apriori uses candidate generation where frequent subsets are extended one item at a time.	FP-growth generates a conditional FP-Tree for every item in the data.
Since apriori scans the database in each step, it becomes time-consuming for data where the number of items is larger.	FP-tree requires only one database scan in its beginning steps, so it consumes less time.
A converted version of the database is saved in the memory	A set of conditional FP-tree for every item is saved in the memory
It uses a breadth-first search	It uses a depth-first search.

FP-Tree Definition

- FP-tree is **a frequent pattern tree** . Formally, FP-tree is a tree structure defined below:

  1. One root labeled as “null", a set of *item prefix sub-trees* as the children of the root, and a *frequent-item header table*.

  2. Each node in *the item prefix sub-trees* has three fields:
     - item-name : register which item this node represents,
     - count, the number of transactions represented by the portion of the path reaching this node,
     - node-link that links to the next node in the FP-tree carrying the same item-name, or null if there is none.

  3. Each entry in the *frequent-item header table* has two fields,
     - item-name, and
     - head of node-link that points to the first node in the FP-tree carrying the item-name.

Construct FP-tree

Two Steps:

1. Scan the transaction DB for the first time, find frequent items (single item patterns) and order them into a list L in frequency descending order.

   e.g., L={f:4, c:4, a:3, b:3, m:3, p:3}

   In the format of (item-name, support)

2. For each transaction, order its frequent items according to the order in L; Scan DB the second time, construct FP-tree by putting each frequency ordered transaction onto it.

FP-tree Example: step 1

Step 1: Scan DB for the first time to generate L

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}

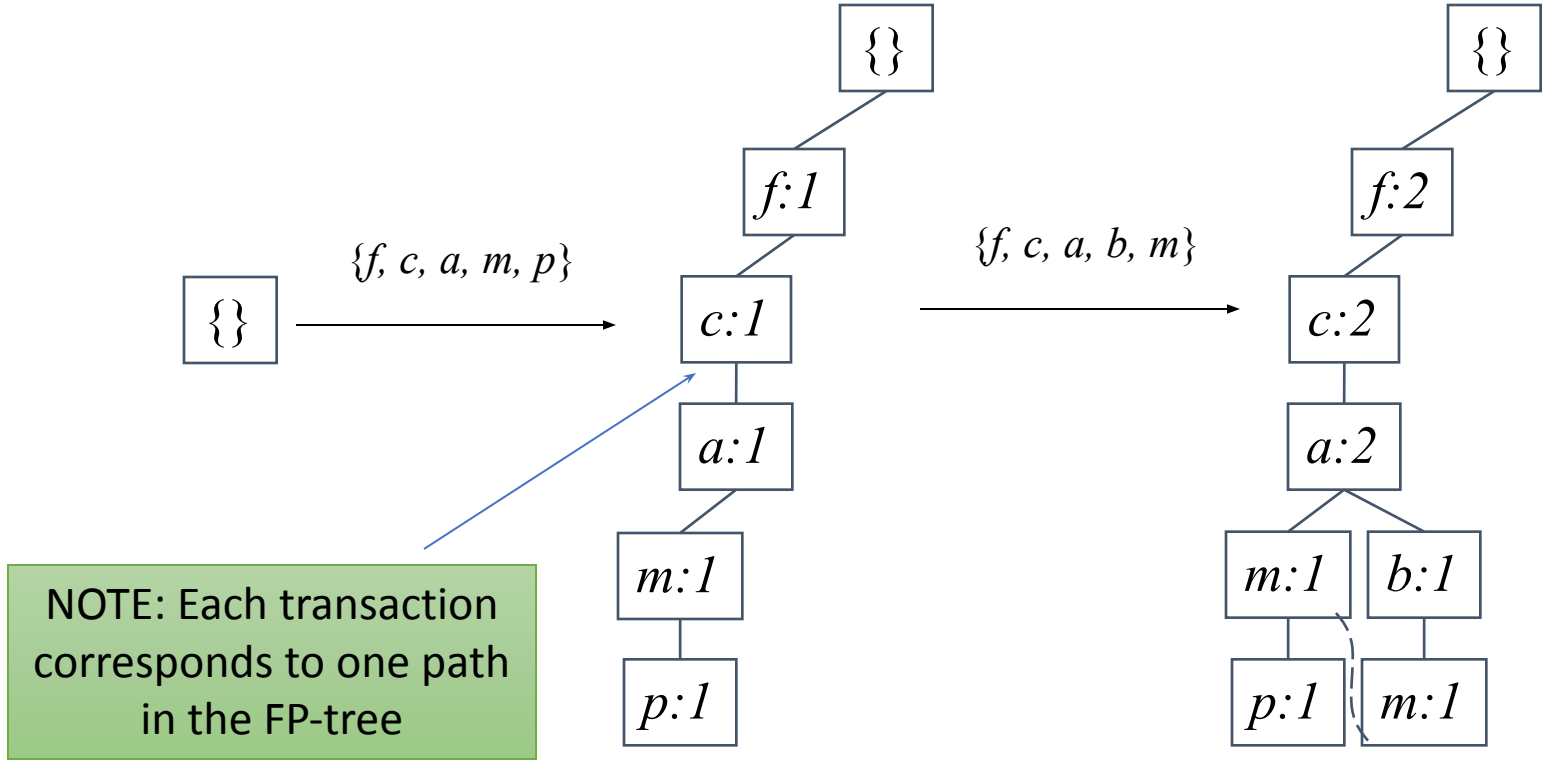
L

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

By-Product of First Scan of Database

FP-tree Example: step 2

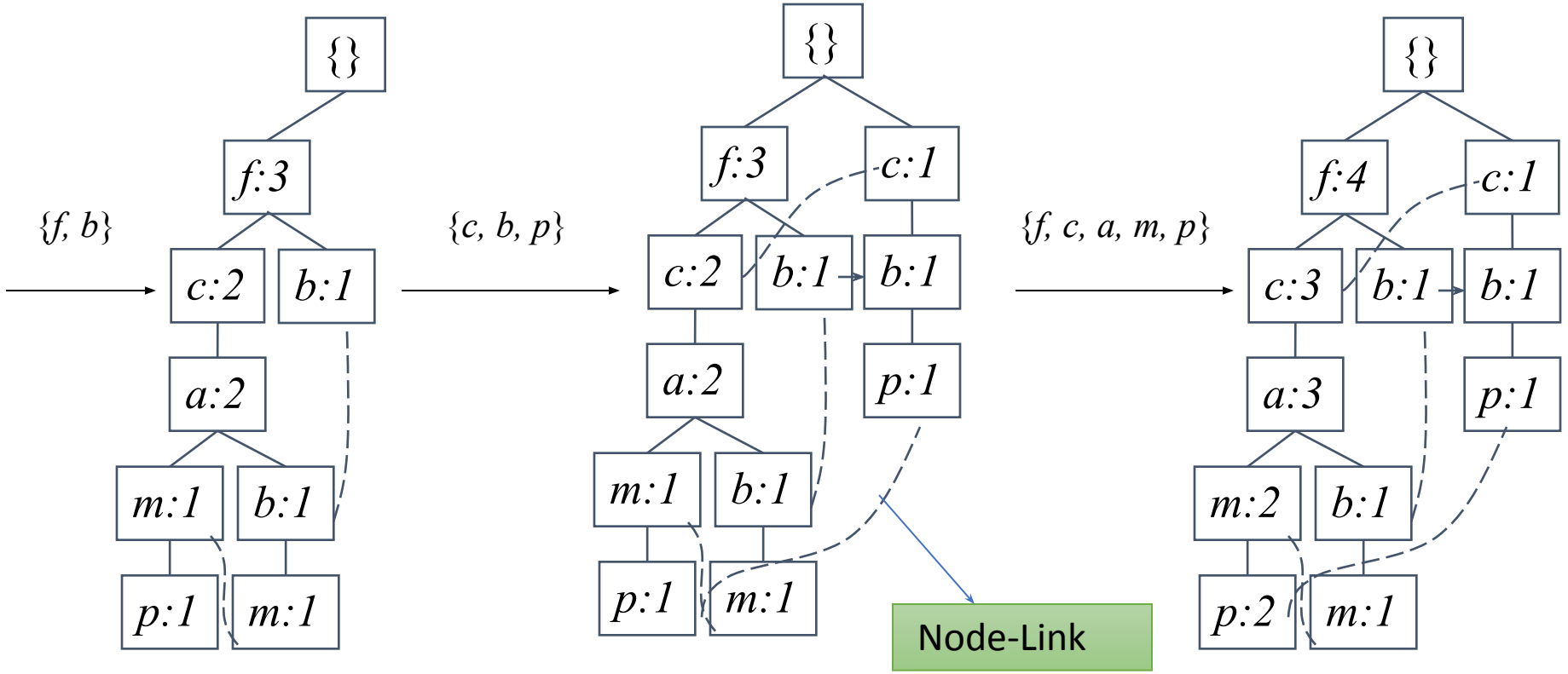
Step 2: construct FP-tree



NOTE: Each transaction corresponds to one path in the FP-tree

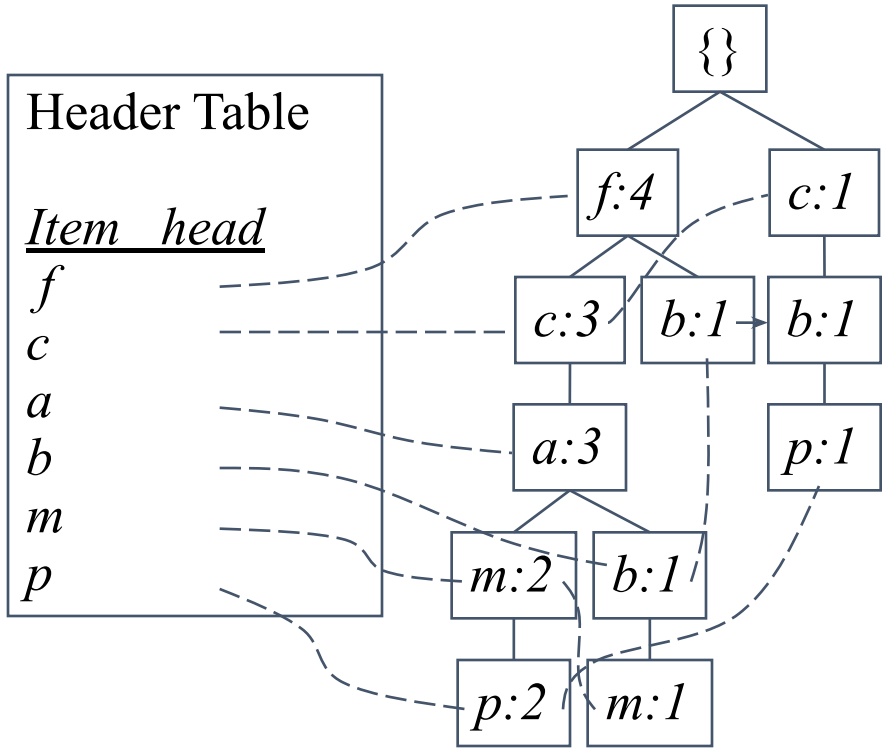
FP-tree Example: step 3

Step 3: construct FP-tree



Construction Example

Final FP-tree



FP-growth:
Mining Frequent Patterns
Using FP-tree

Mining Frequent Patterns Using FP-tree

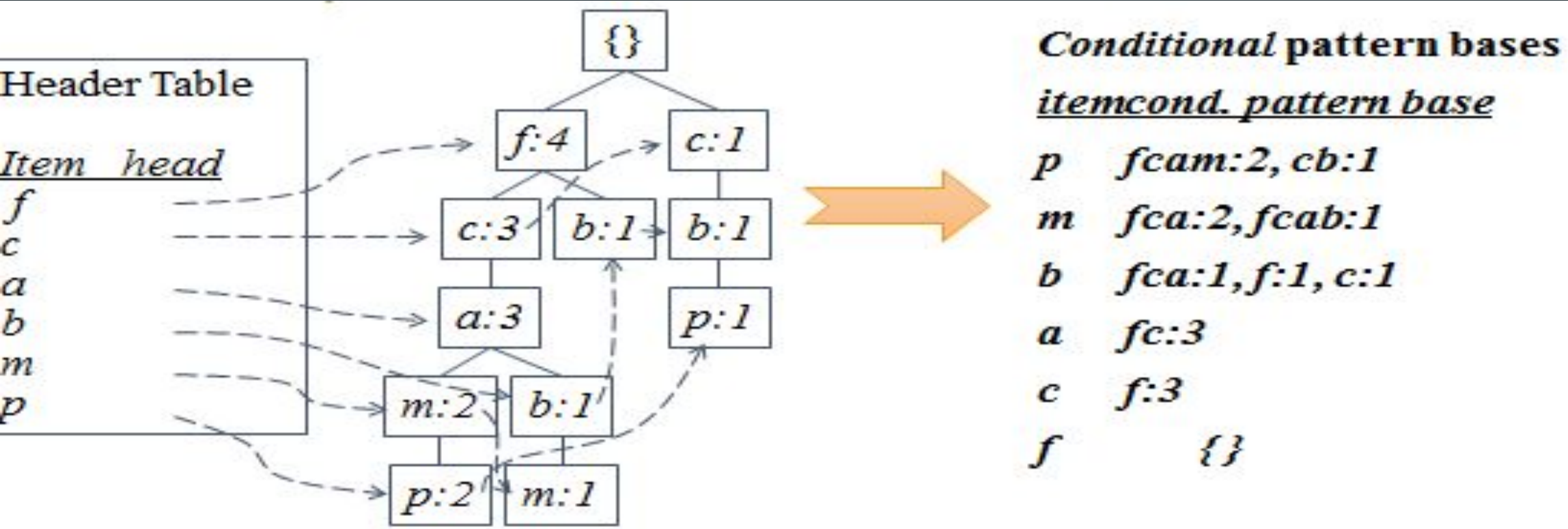
- General idea (divide-and-conquer)

  Recursively grow frequent patterns using the FP-tree: looking for shorter ones recursively and then concatenating the suffix:

  - 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)

3 Major Steps

Starting the processing from the end of list L:	
Step 1:	Construct **conditional pattern base** for each item in the header table
Step 2	Construct **conditional FP-tree** from each conditional pattern base
Step 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

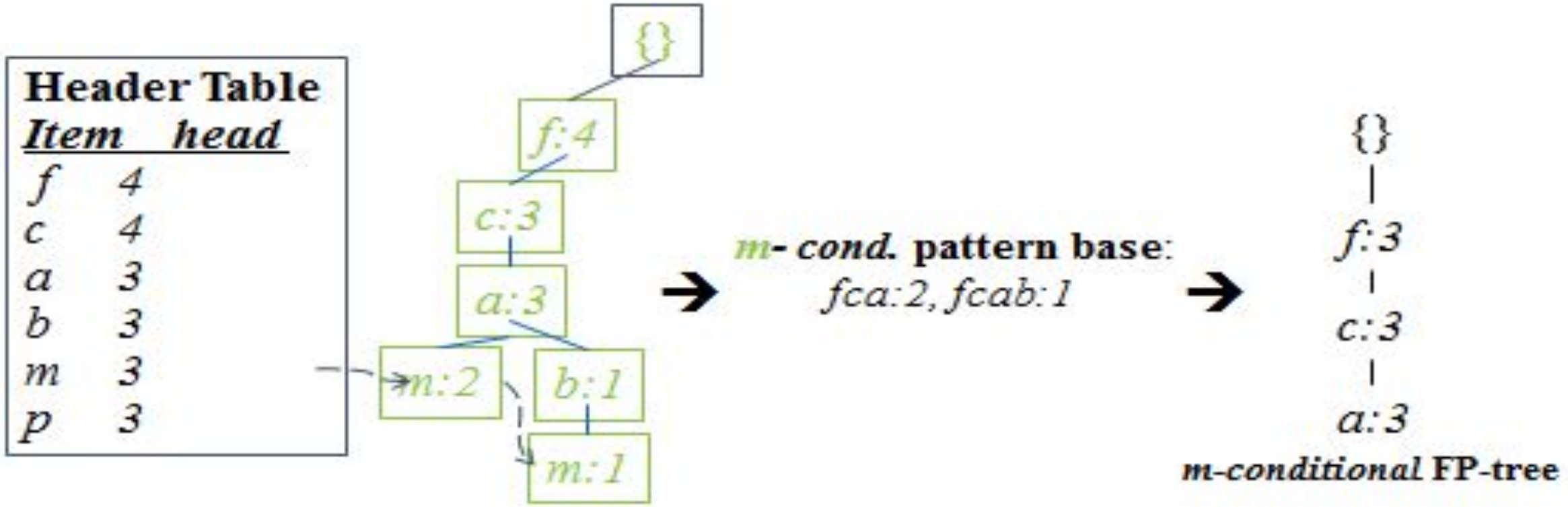


Step 1: Construct Conditional Pattern Base

- Starting at the bottom of frequent-item 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**

Properties of FP-Tree

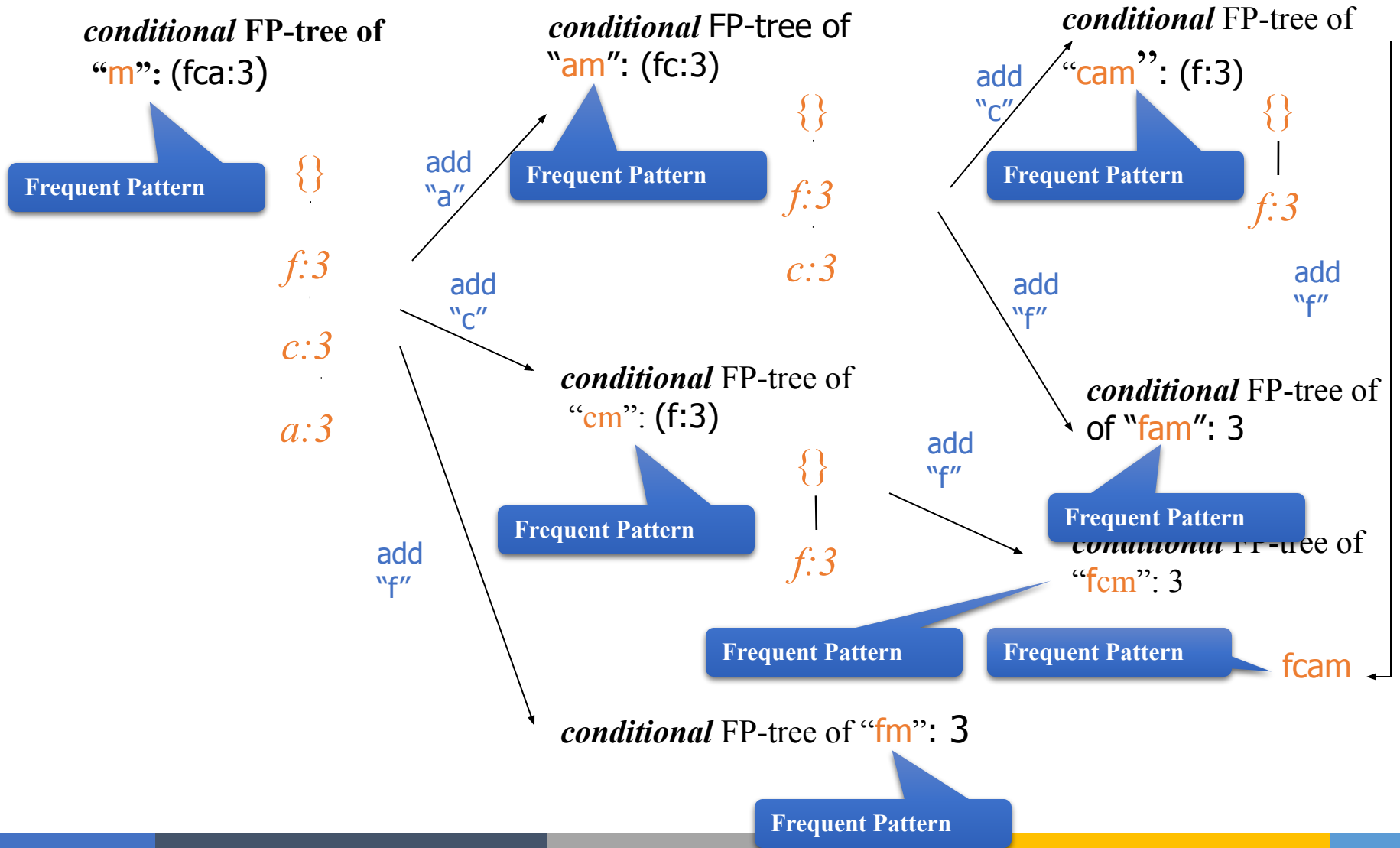
Node-link property	For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header.
Prefix path property	To calculate the frequent patterns for a node a_i in a path P, only the prefix sub-path 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 conditional FP-tree for the frequent items of the pattern base

Step 3: Recursively mine the conditional FP-tree



Principles of FP-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 α ∪ β is a frequent itemset in DB iff β is frequent in B.

Is "*fcabm*" a frequent pattern?

- "fcab" is a branch of m's conditional pattern base
- "b" is **NOT** frequent in transactions containing "fcab"
- "bm" is **NOT** a frequent itemset.

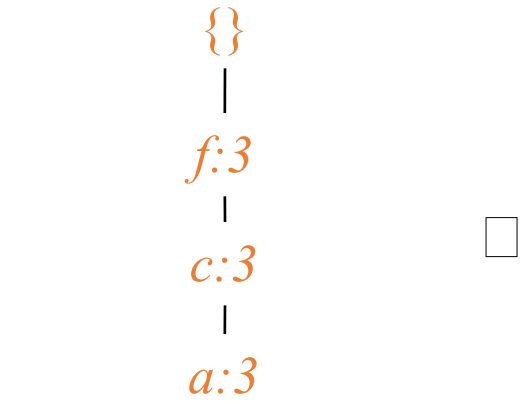
Conditional Pattern Bases and Conditional FP-Tree

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

order of L

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



m-conditional FP-tree

All frequent patterns concerning *m*: combination of {f, c, a} and *m*

m,

fm, cm, am,

fcm, fam, cam,

fcam

Advantages of the FP-tree Structure

The most significant advantage of the FP-tree

- Scan the DB only twice and twice only.

Completeness:

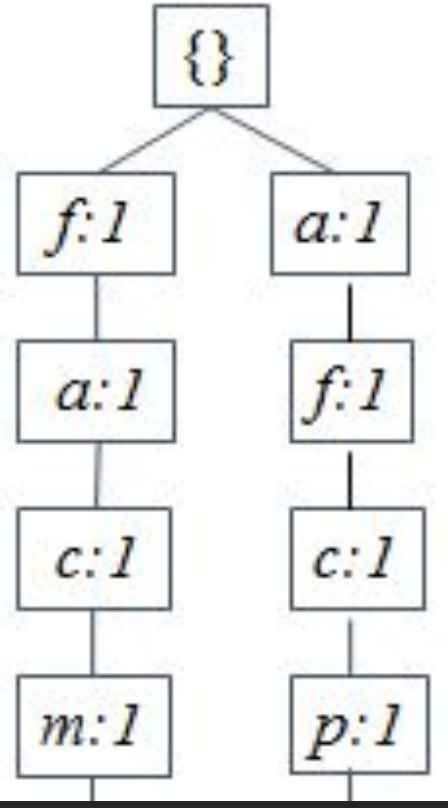
- the FP-tree contains all the information related to mining frequent patterns (given the min-support threshold). Why?

Compactness:

- The size of the tree is bounded by the occurrences of frequent items
- The height of the tree is bounded by the maximum number of items in a transaction

- Why descending order?
- Example 1:

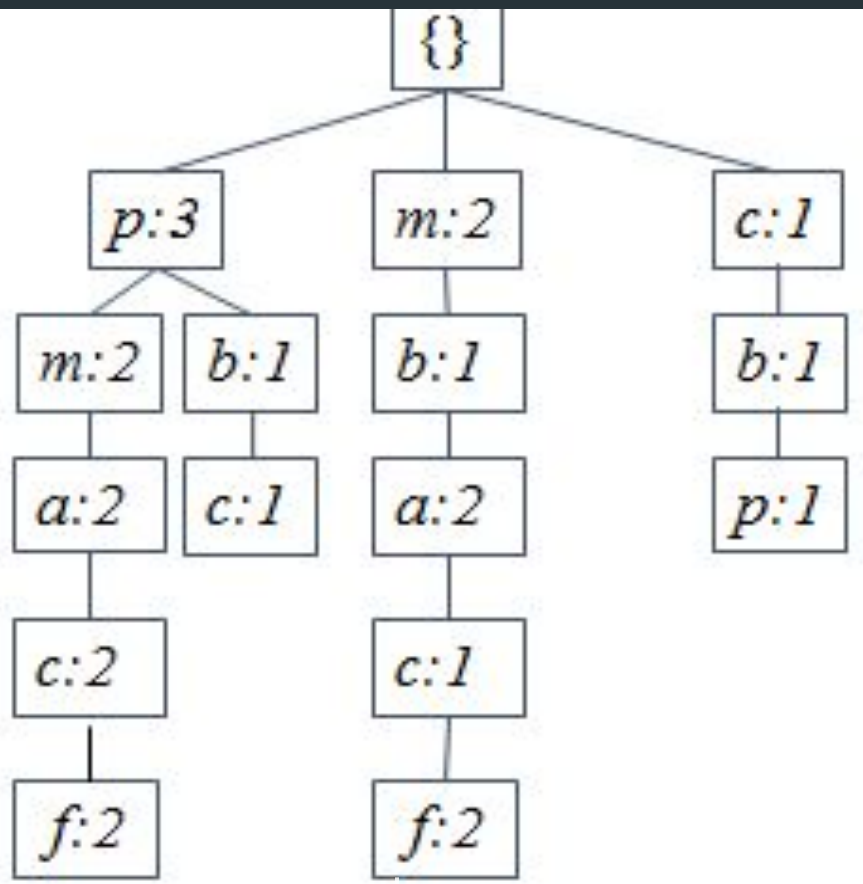
TID	(unordered) frequent items
100	{f, a, c, m, p}
500	{a, f, c, p, m}



Question

- Example 2:

TID	(ascended) frequent items
100	{p, m, a, c, f}
200	{m, b, a, c, f}
300	{b, f}
400	{p, b, c}
500	{p, m, a, c, f}



This tree is larger than FP-tree, because in FP-tree, more frequent items have a higher position, which makes branches less

Question

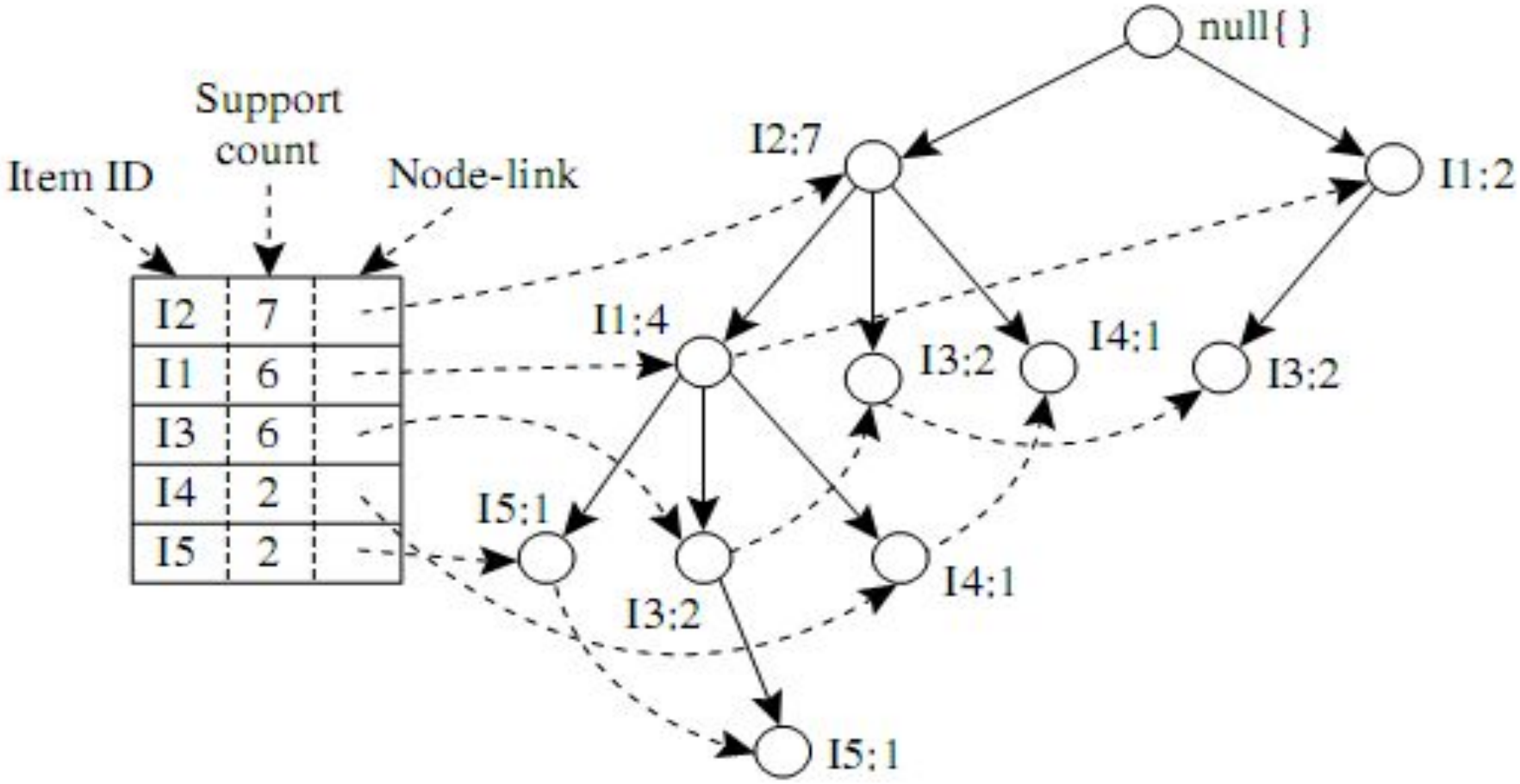
Exercises

Exercise 01
Support=2

Transactional Data for an *AllElectronics* Branch

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

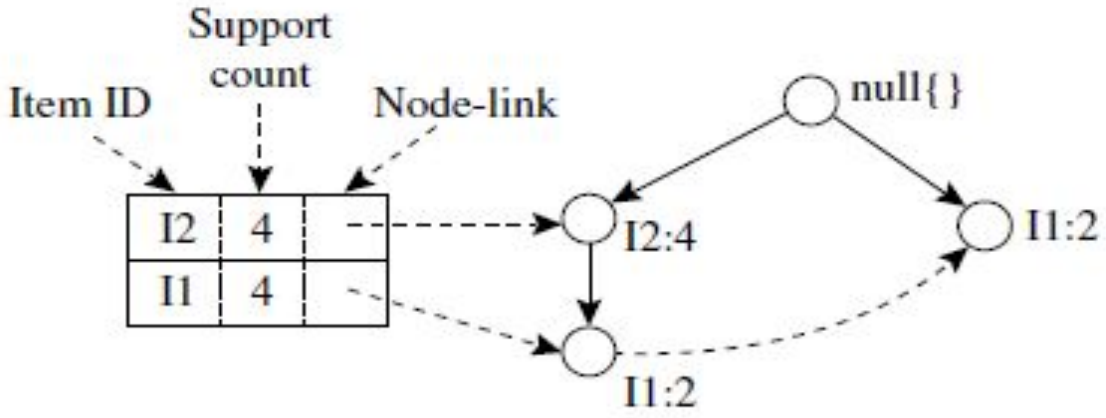
FP Growth: Solution



FP Growth: Solution

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}



FP Exexrcise 02

Transactions

A B C E F O

A C G

E I

A C D E G

A C E G L

E J

A B C E F P

A C D

A C E G M

A C E G N

Freq. 1-Itemsets.
Supp. Count ≥2

A:8	
C:8	
E:8	
G:5	
B:2	
D:2	
F:2	

Transactions with items sorted based on frequencies, and ignoring the infrequent items.

A C E B F

A C G

E

A C E G D

A C E G

E

A C E B F

A C D

A C E G

A C E G

FP-Tree after reading 1st transaction

A C E B F

A C G

E

A C E G D

A C E G

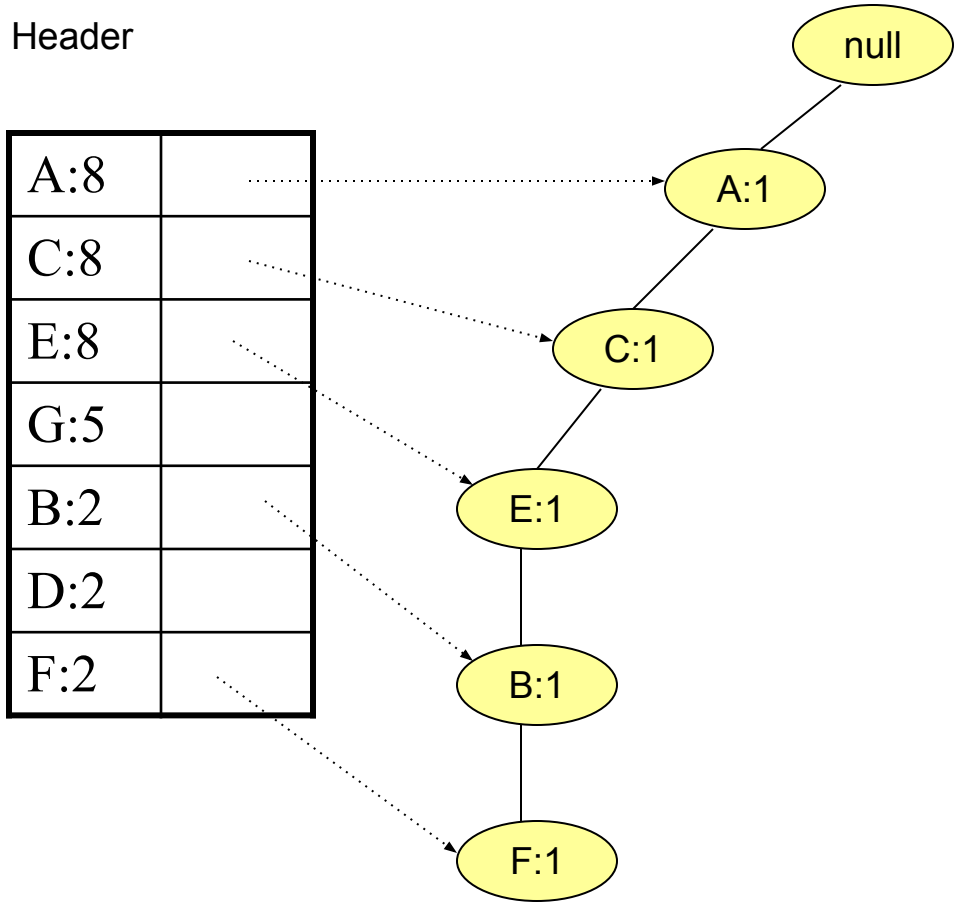
E

A C E B F

A C D

A C E G

A C E G



FP-Tree after reading 2nd transaction

A C E B F

A C G

E

A C E G D

A C E G

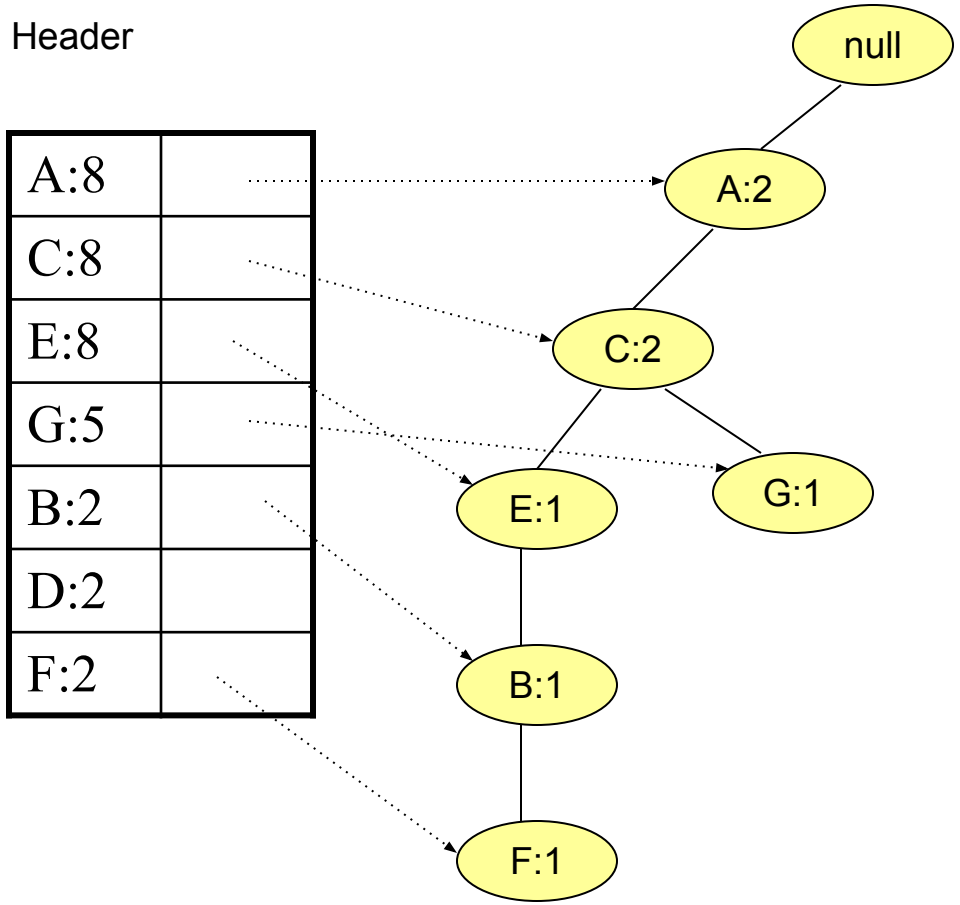
E

A C E B F

A C D

A C E G

A C E G



FP-Tree after reading 3rd transaction

A C E B F

A C G

E

A C E G D

A C E G

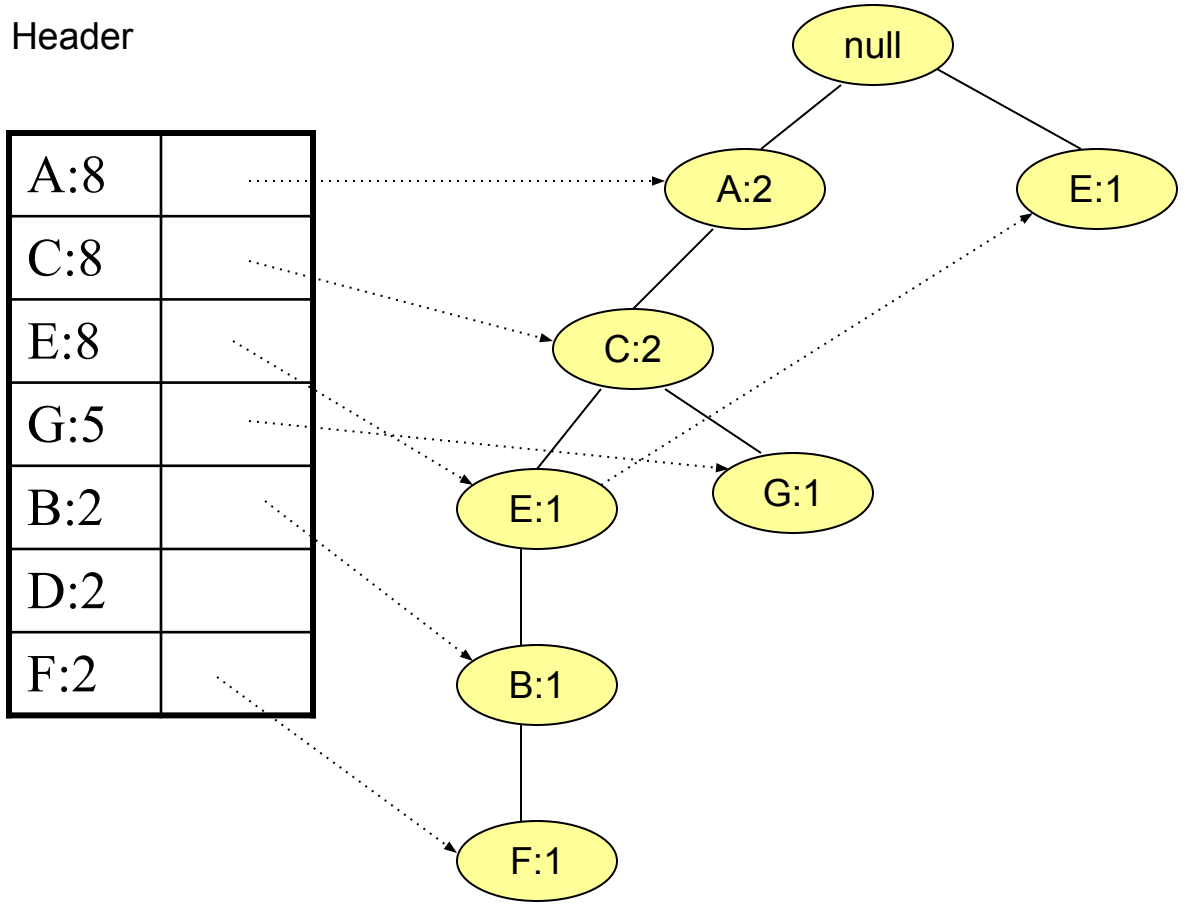
E

A C E B F

A C D

A C E G

A C E G



FP-Tree after reading 4th transaction

A C E B F

A C G

E

A C E G D

A C E G

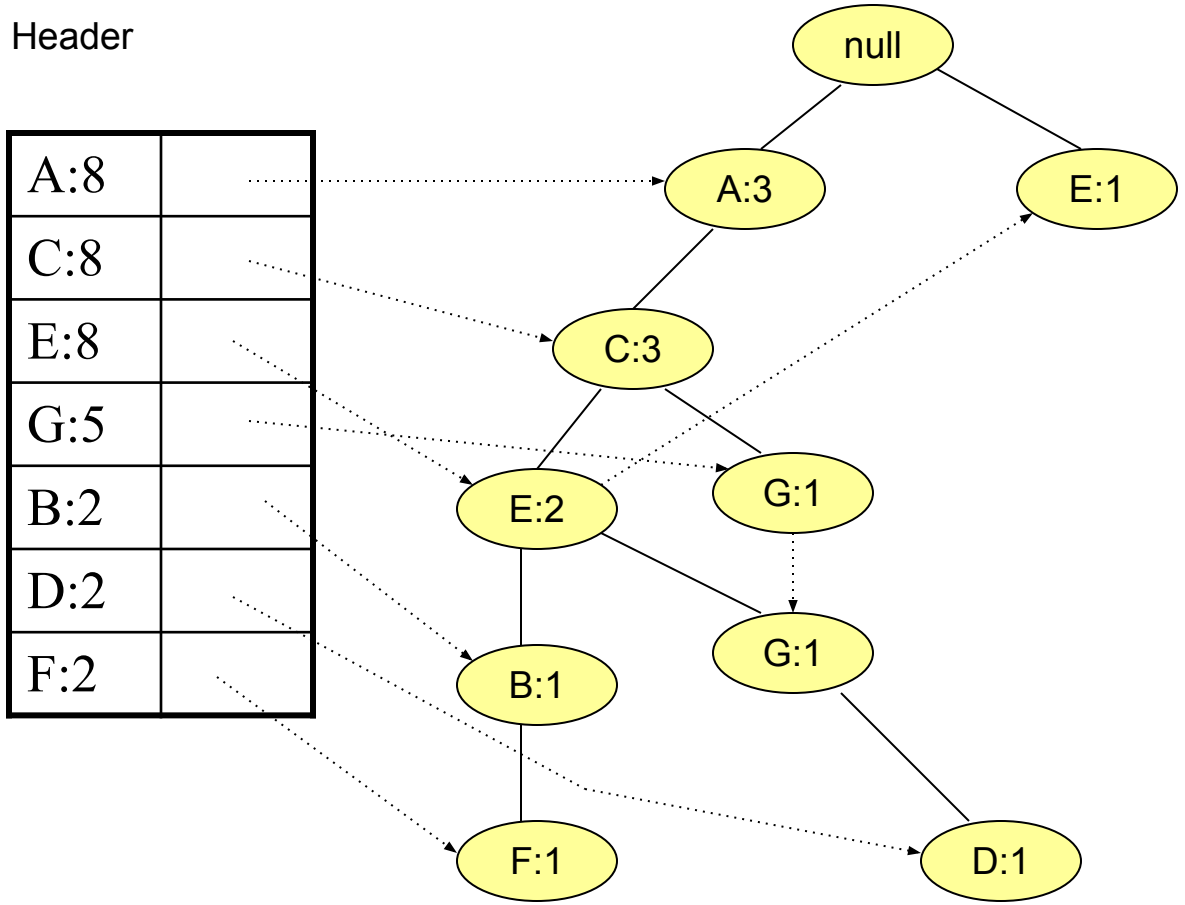
E

A C E B F

A C D

A C E G

A C E G



FP-Tree after reading 5th transaction

A C E B F

A C G

E

A C E G D

A C E G

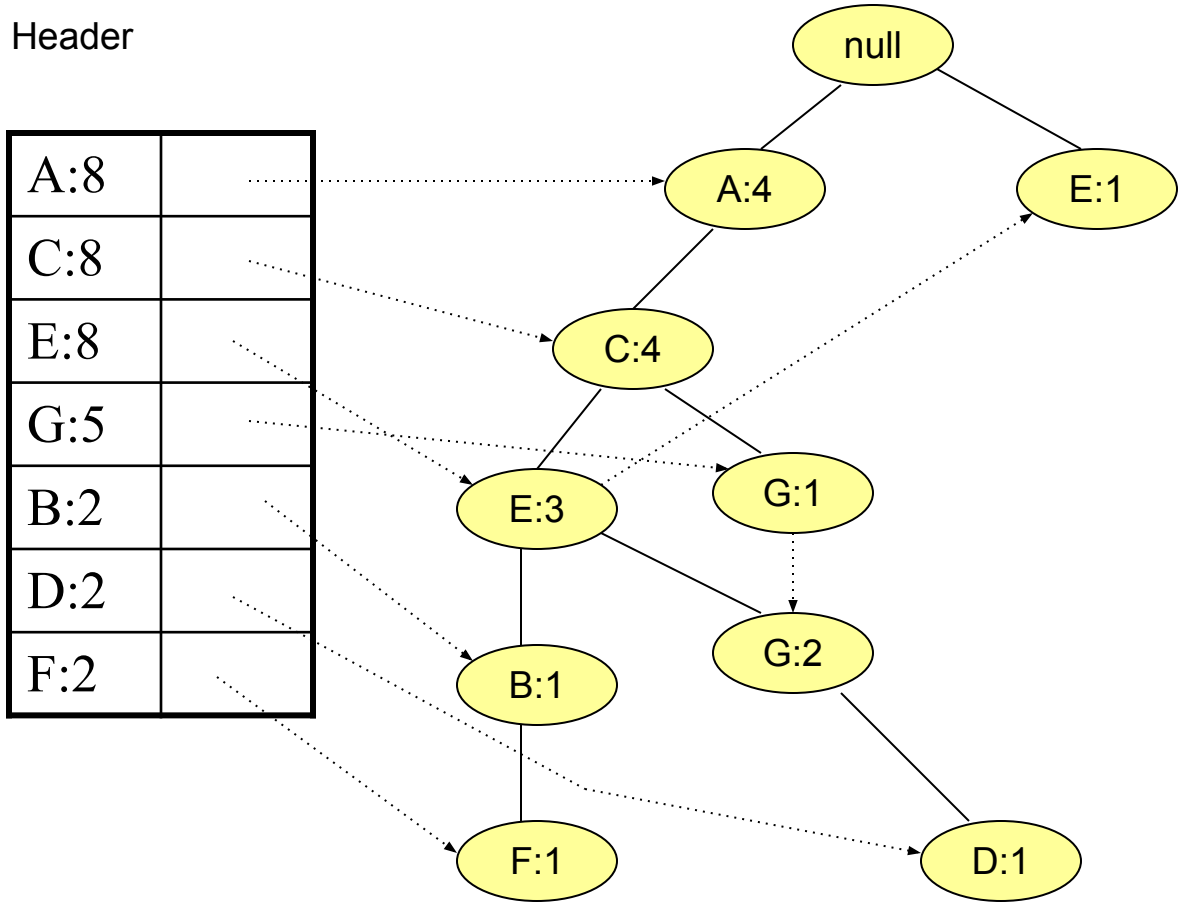
E

A C E B F

A C D

A C E G

A C E G



FP-Tree after reading 6th transaction

A C E B F

A C G

E

A C E G D

A C E G

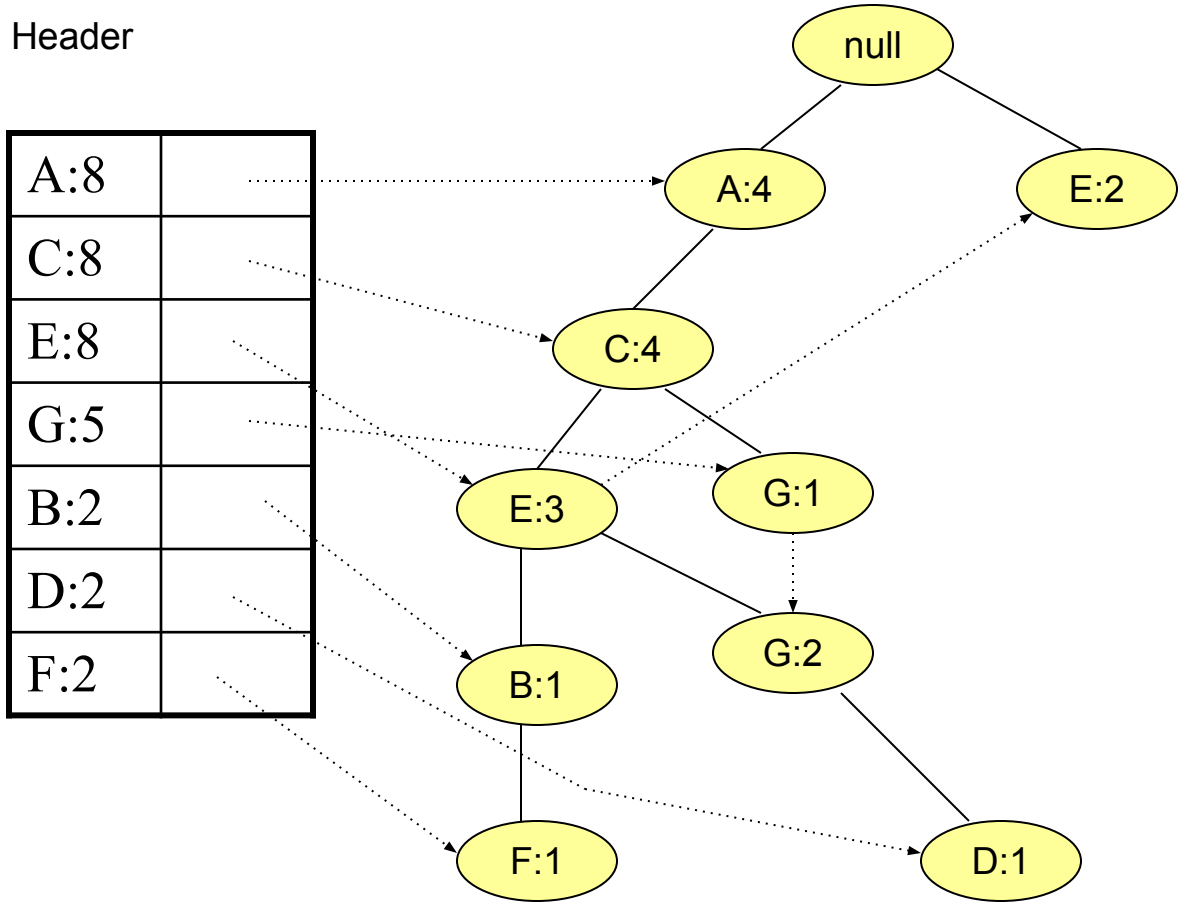
E

A C E B F

A C D

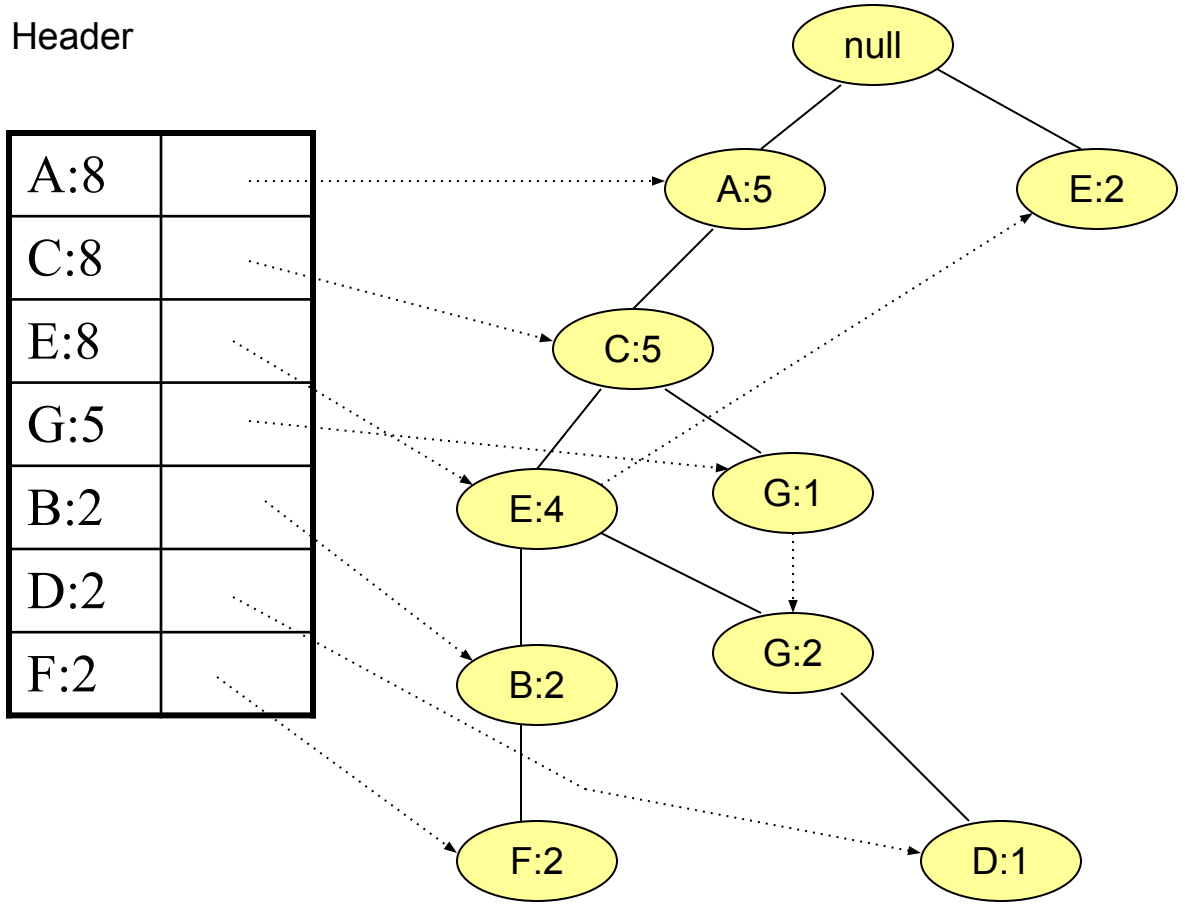
A C E G

A C E G



FP-Tree after reading 7th transaction

A C E B F
A C G
E
A C E G D
A C E G
E
A C E B F
A C D
A C E G
A C E G



FP-Tree after reading 8th transaction

A C E B F

A C G

E

A C E G D

A C E G

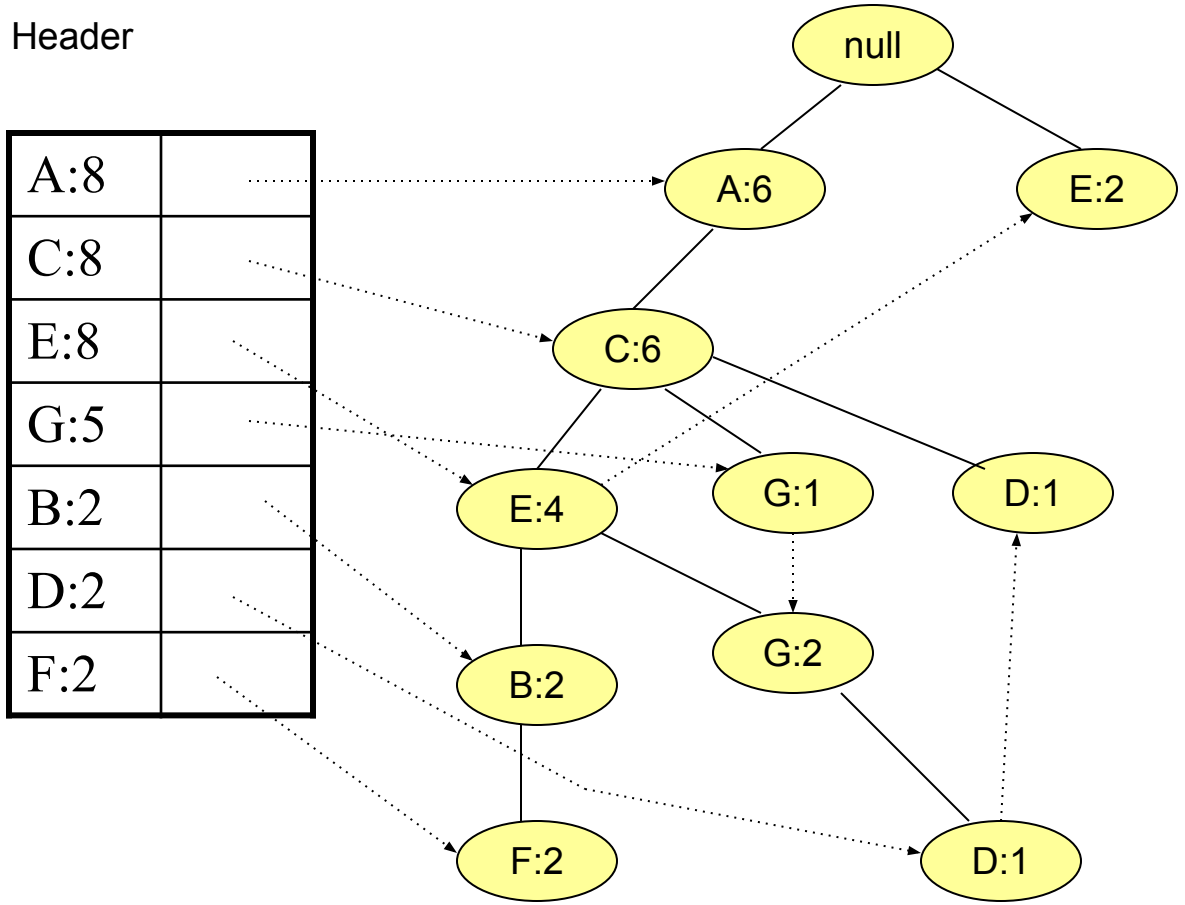
E

A C E B F

A C D

A C E G

A C E G



FP-Tree after reading 9th transaction

A C E B F

A C G

E

A C E G D

A C E G

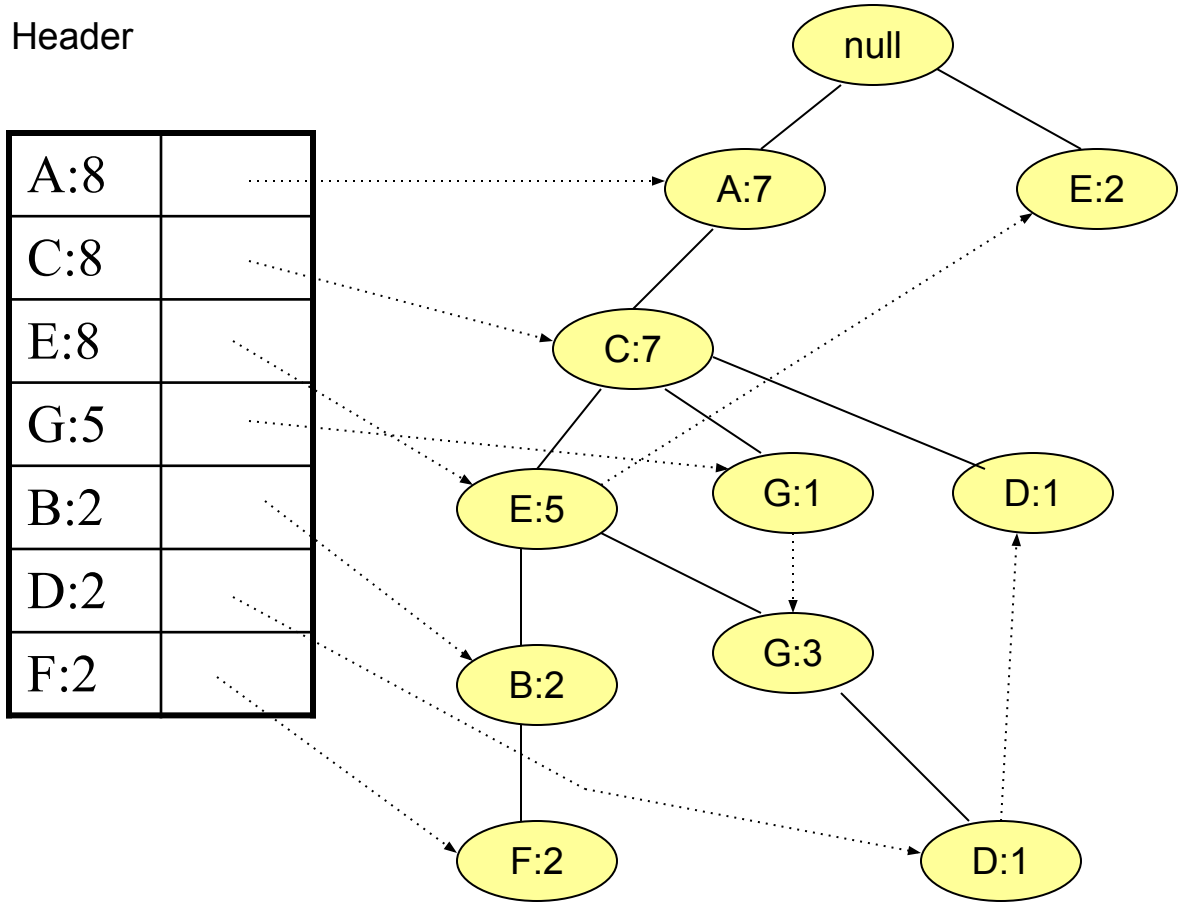
E

A C E B F

A C D

A C E G

A C E G



FP-Tree after reading 10th transaction

A C E B F

A C G

E

A C E G D

A C E G

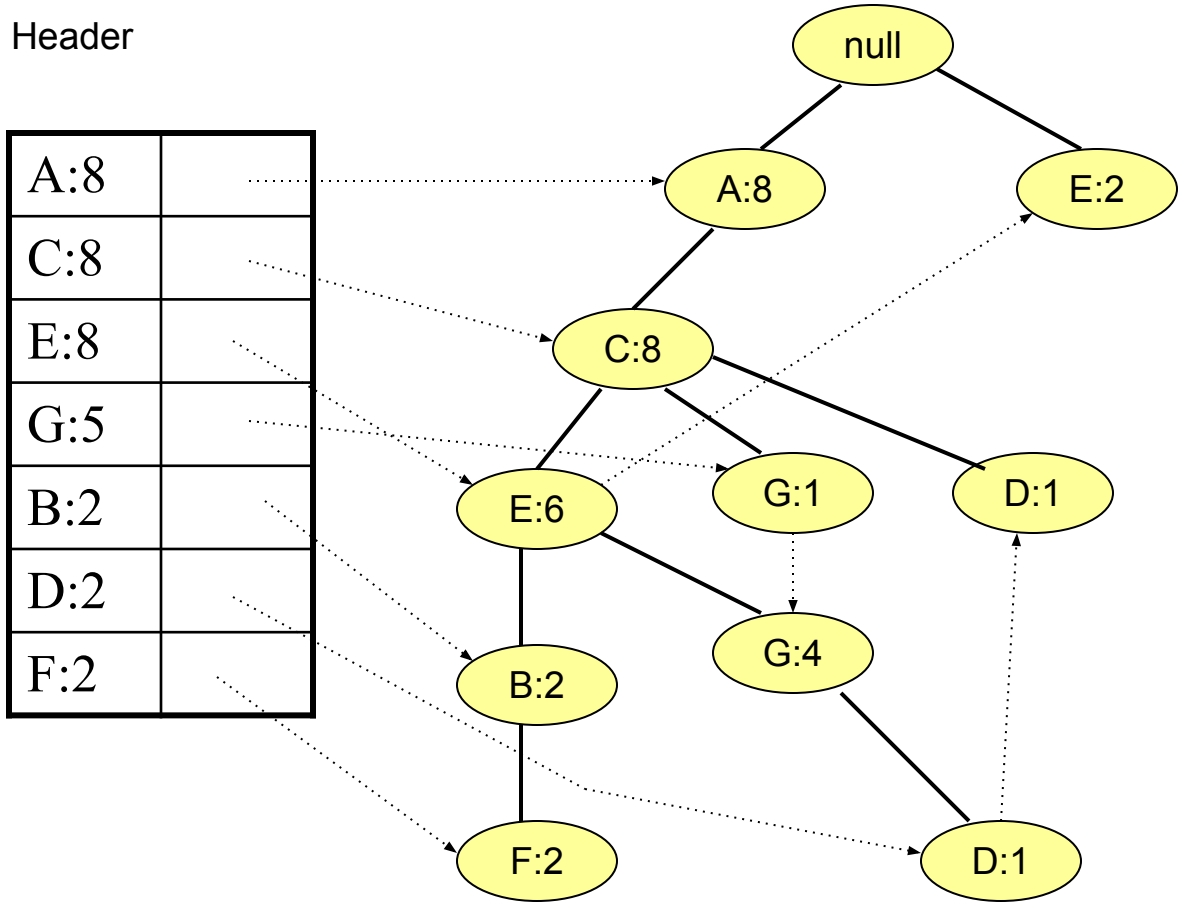
E

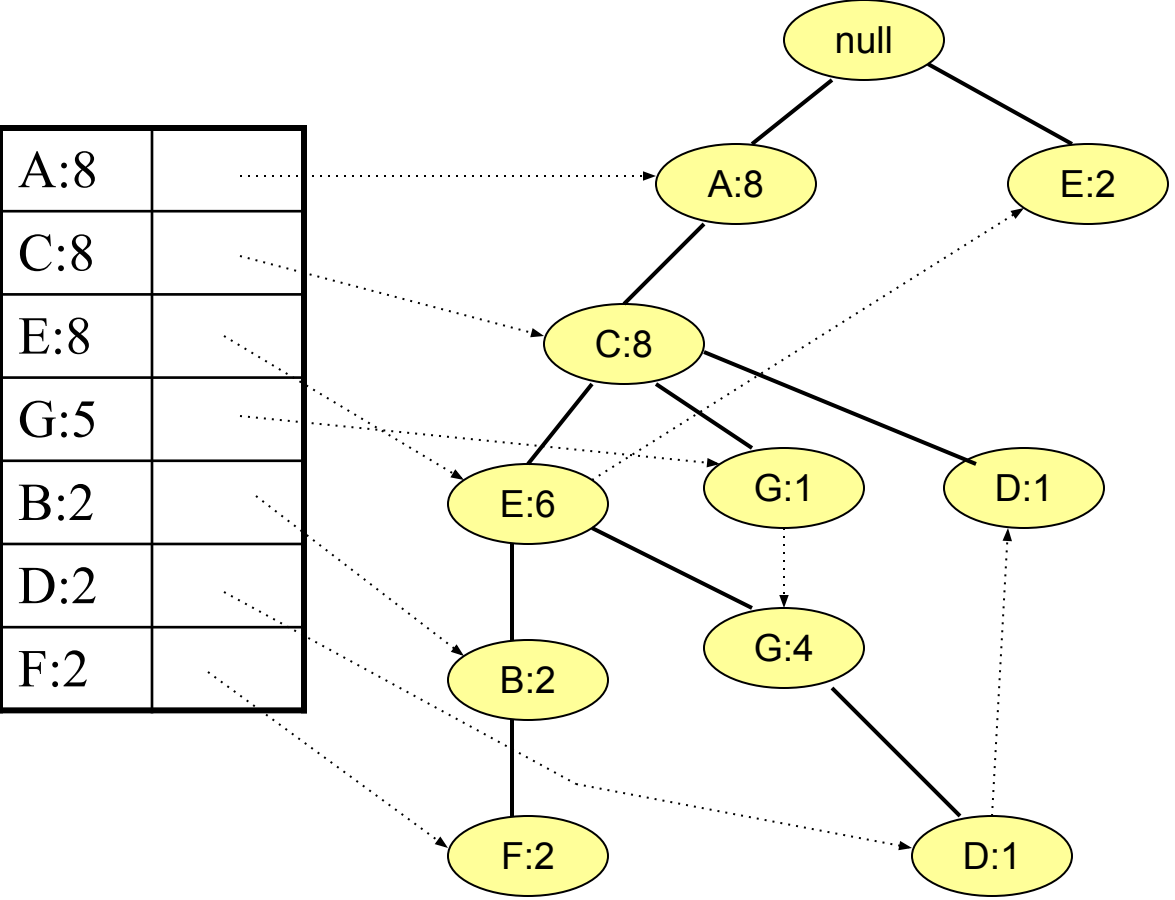
A C E B F

A C D

A C E G

A C E G





Items Arranged in ascending order of support	Conditional Pattern Base	Conditional Frequent Pattern Tree	Frequent Pattern Generated
F	{A,C,E,B :2}	{A:2,C:2,E:2,B:2}	{A,F:2} , {C,F:2} ,{E,F :2} ,{B,F:2} {A,C,E,B,F:2}
D	{A,C:1} {A,C,E,G:1}	{A:2 ,C:2}	{A,D:2} , {C,D:2} {A,C,D:2}
B	{A, C,E:2}	{A:2,C:2,E:2}	{A,B:2} {C,B:2} {E,B:2} {A,C,E,B:2}
G	{A,C:1} {A,C,E:4}	{A,C:2} {E:4}	{A,G:2} {C,G:2} {A.C.G:2} {E,G:4}
E	{A,C:6}	{A,C:6}	{A,E:6} {C,E:6} {A,C.E:6}
C	{A:8}	{A:8}	{A,C:8}

Exercise 03

min_sup:3

Transaction ID	Items
T1	{E,K,M,N,O,Y}
T2	{D,E,K,N,O,Y}
T3	{A,E,K,M}
T4	{C,K,M,U,Y}
T5	{C,E,I,K,O,O}

Solution

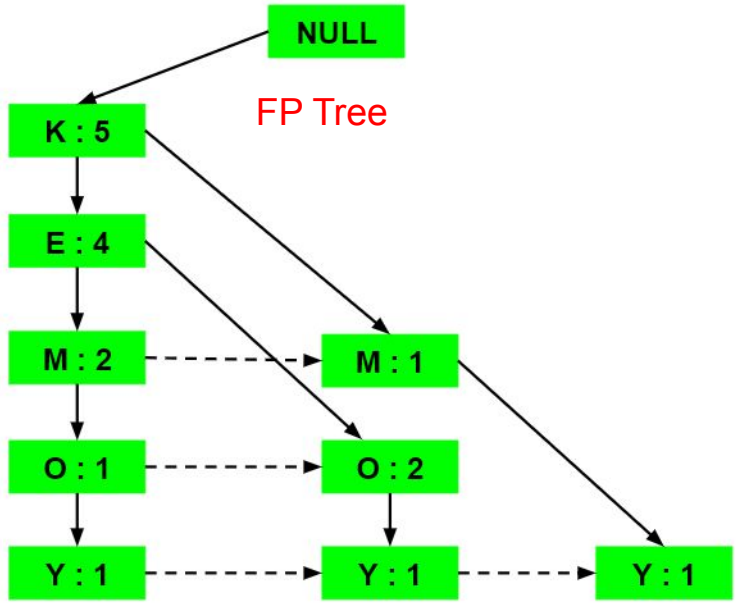
Original Set

Transaction ID	Items
T1	{E,K,M,N,O,Y}
T2	{D,E,K,N,O,Y}
T3	{A,E,K,M}
T4	{C,K,M,U,Y}
T5	{C,E,I,K,O,O}

Ordered Set as per Support Count

Transaction ID	Items	Ordered-Item Set
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}

FP Tree



Item Set with min_sup:3 = {K : 5, E : 4, M : 3, O : 3, Y : 3}

Items	Conditional Pattern Base
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}
O	{{K,E,M : 1}, {K,E : 2}}
M	{{K,E : 2}, {K : 1}}
E	{K : 4}
K	

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{K : 3}
O	{{K,E,M : 1}, {K,E : 2}}	{K,E : 3}
M	{{K,E : 2}, {K : 1}}	{K : 3}
E	{K : 4}	{K : 4}
K		

Items	Frequent Pattern Generated
Y	{<K,Y : 3>}
O	{<K,O : 3>, <E,O : 3>, <E,K,O : 3>}
M	{<K,M : 3>}
E	{<E,K : 3>}
K	

Advantages Disadvantages of FP Growth

Advantages Of FP Growth Algorithm

1.This algorithm needs to scan the database only twice when compared to Apriori which scans the transactions for each iteration.
2.The pairing of items is not done in this algorithm and this makes it faster.
3.The database is stored in a compact version in memory.
4.It is efficient and scalable for mining both long and short frequent patterns.

Disadvantages Of FP-Growth Algorithm

1.FP Tree is more cumbersome and difficult to build than Apriori.
2.It may be expensive.
3.When the database is large, the algorithm may not fit in the shared memory.

Comparison of FP Growth with Apriori

FP Growth	Apriori
Pattern Generation	
FP growth generates pattern by constructing a FP tree	Apriori generates pattern by pairing the items into singletons, pairs and triplets.
Candidate Generation	
There is no candidate generation	Apriori uses candidate generation
Process	
The process is faster as compared to Apriori. The runtime of process increases linearly with increase in number of itemsets.	The process is comparatively slower than FP Growth, the runtime increases exponentially with increase in number of itemsets
Memory Usage	
A compact version of database is saved	The candidates combinations are saved in memory

FP Growth-Generating Rules

Generating rules in the FP algorithm is the process of identifying strong association rules from the frequent itemsets that have been mined. Association rules are a type of data mining rule that describes the relationship between two or more items. They are typically expressed in the form "if-then" statements, such as "if a customer buys diapers, then they are also likely to buy beer.“

It generates association rules by using the following steps:

- Calculate the support and confidence of each frequent itemset. Support is the proportion of transactions in the database that contain the itemset. Confidence is the proportion of transactions that contain the itemset among those that also contain the antecedent item.
- Filter the frequent itemsets based on minimum support and minimum confidence thresholds. This ensures that only rules that are both common and reliable are considered.

- Generate association rules for each remaining frequent itemset. For each frequent itemset, generate an association rule for each subset of the itemset. The antecedent of the rule is the subset of the itemset, and the consequent is the remaining item in the itemset.

- Evaluate the strength of each association rule using additional metrics, such as lift and conviction. Lift measures how much more likely the consequent is to occur when the antecedent is also present, compared to when the antecedent is not present. Conviction measures the extent to which the antecedent rules out the possibility of the consequent not occurring.
- Select the strongest association rules based on the chosen metrics. This can be done manually or using automated rule selection techniques.
- **Steps:-**
- Find the minimum support of each item.
- Order frequent itemset in descending order.
- Draw an FP tree.
- Minimum frequent pattern from the FP tree.

References

- Data Mining. Concepts and Techniques, 3rd Edition, Han Kamber
- https://www.cs.uic.edu/~liub/WebContentMining.html
- http://dmr.cs.umn.edu/Papers/P2004_4.pdf

Thank you !!