# Unit 5
# Mining Frequent Itemsets

**Prepared By**

**Arjun Singh Saud, Asst. Prof. CDCSIT**

# Frequent Patterns

- **Frequent patterns** are patterns that appear in a data set frequently. Frequent Patterns can be frequent itemsets, Frequent subsequences, or Frequent substructures
- For example, a set of items, such as milk and bread, that appear frequently together in a transaction data set is a *frequent itemset*.
- A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a *frequent sequential pattern or frequent subsequence*.

# Frequent Patterns

- A *substructure* can refer to different structural forms, such as sub-graphs or sub-trees. If a substructure occurs frequently, it is called a *frequent structured pattern or frequent substructure*.
- Finding such frequent patterns plays an essential role in mining associations, correlations, and many other interesting relationships among data.

# Market Basket Analysis

- Frequent itemset mining leads to the discovery of associations and correlations among items in large transactional or relational data sets.
- A typical example of frequent itemset mining is market basket analysis. This process analyzes customer buying habits by finding associations between the different items that customers place in their shopping baskets.
- The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers.

# Market Basket Analysis

- Such information can lead to increased sales by helping retailers do selective marketing and design different store layouts.
- Items that are frequently purchased together can be placed in proximity in order to further encourage the sale of such items together.
- Market basket analysis can also help retailers plan which items to put on sale at reduced prices. If customers tend to purchase computers and printers together, then having a sale on computer may encourage the sale of Computers *as well as* printers.

# Association Rules

- Association rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository.
- An example of an association rule would be "*If a customer buys a dozen eggs, he/she is 80% likely to purchase milk.*"
- An association rule has two parts, an antecedent (if part) and a consequent (then part). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent.

# Association Rules

- Let $I = \{I1, I2.....In\}$ be a set of *items*. Let $D = \{T1, T2.....Tm\}$ be a set of transactions called the *database*. A *rule* is defined as an implication of the form:

  $X => Y$

  Where $X, Y \subseteq I$ and $X \cap Y = \Phi$

- An example rule for the supermarket could be $\{butter, bread\} => \{milk\}$ meaning that if a customer purchases butter and bread then he/she is also likely to buy milk.

# Association Rules

- Association rule mining is a method for discovering frequent patterns in large databases.
- It is intended to identify strong rules discovered in databases using different measures of interestingness.
- Some applications of association rule mining are: Market-Basket analysis, cross-marketing, catalog design, clustering, classification, etc.

# Association Rules

- In general, association rule mining can be viewed as a two-step process:

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

# Support and Confidence

- In order to select interesting rules from the set of all possible rules, constraints on various measures of significance and interest are used. The best-known constraints are minimum thresholds on *support and confidence*.
- **Support:** *Support* of association rule $A => B$ is the probability that the database contains both A and B.

$$Support(A => B) = P(A \cup B)$$

# Support and Confidence

- **Confidence**: *Confidence* of association rule $A => B$ is conditional probability that the transactions that contains item A also contains item B.

$$Confidence(A => B) = P(B \mid A) = P(A \cup B) / P(A) = Support(A => B) / P(A)$$

# Support and Confidence

- **Example 1:** Consider the example given below and Calculate support and confidence of rule $bread \Rightarrow milk$

| Transaction ID | Milk | Bread | Butter | Beer | Diaper |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 0 |

**Solution**

$$\text{Support}\ (bread \Rightarrow milk) = 2/5 = 0.4 = 40\%$$

$$\text{Confidence}(bread \Rightarrow milk) = 2/3 = 0.66 = 66\%$$

# Support and Confidence

- **Example 2:** Consider the example given below and Calculate support and confidence of rule $A => C$

| Tid | Items |
|---|---|
| 1 | A, B, C |
| 2 | A, C |
| 3 | A, D |
| 4 | B, E, F |

**Solution**

Support $(A => C)$=2/4=0.5=50%

Confidence( $A => C$ )=0.5/(3/4)=0.5/0.75=0.66=66%

# Frequent Itemsets and Closed Itemsets

- A set of items is referred to as an itemset. An itemset that contains *k* items is called *k*-itemset. For example, set (*computer, antivirus-software*) is a 2-itemset and set (*Milk, Bread, Cheese*) is a 3-itemset.
- A major challenge in mining frequent itemsets from a large data set is the fact that such mining often generates a huge number of itemsets satisfying the minimum support threshold, especially when minimum support is set low. This is because if an itemset is frequent, each of its subsets is frequent as well.

# Frequent Itemsets and Closed Itemsets

- To overcome this difficulty, concept of closed frequent itemset is introduced.
- An itemset $X$ is closed in a data set $S$ if there exists no proper super-itemset $Y$ such that $Y$ has the same support count as $X$ in $S$.
- For example, let X={a,b} a frequent itemset with support count 4 and Y={a,b,c} is a frequent itemset with support count 3. Here X is closed itemset.
- An itemset $X$ is a closed frequent itemset in set $S$ if $X$ is both closed and frequent in $S$.

# Apriori Algorithm

- It is a classic algorithm used in data mining for learning association rules.
- Mining association rules basically means finding the items that are purchased together more frequently than others.
- The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent item set properties.
- Apriori employs an iterative approach known as a *level-wise* search, where frequent *k*-itemsets are used to explore frequent (*k*+1)-itemsets.

# Apriori Algorithm

- First, the set of frequent 1-itemsets that satisfy minimum support is found by scanning the database. The resulting set is denoted $L_1$.
- Next, $L_1$ is used to find $L_2$, the set of frequent 2-itemsets, which is used to find $L_3$, and so on, until no more frequent $k$-itemsets can be found.
- The finding of each $L_k$ requires one full scan of the database. To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property is used. *Apriori Property states that any subset of frequent item set must be frequent*.

# Apriori Algorithm

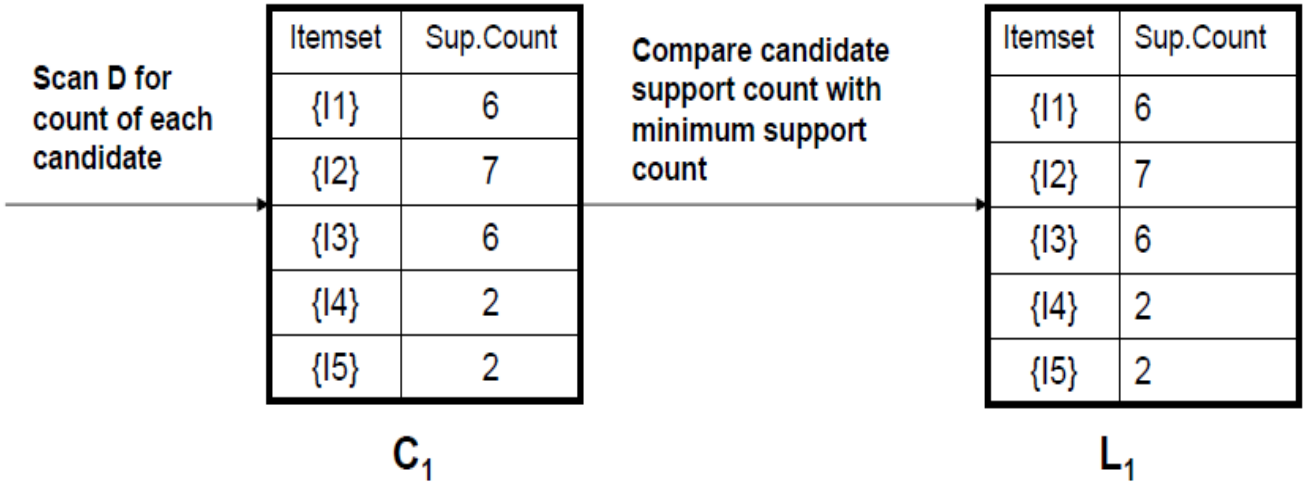
- Example: Consider the database, consisting of 9 transactions. Suppose min. support count required is 2 (i.e. min-sup = 2/9 = 22 %). Let minimum confidence required is 70%. Find out the frequent item sets using Apriori algorithm. Then generate association rules using min. support & min. confidence.

| *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 |

# Apriori Algorithm

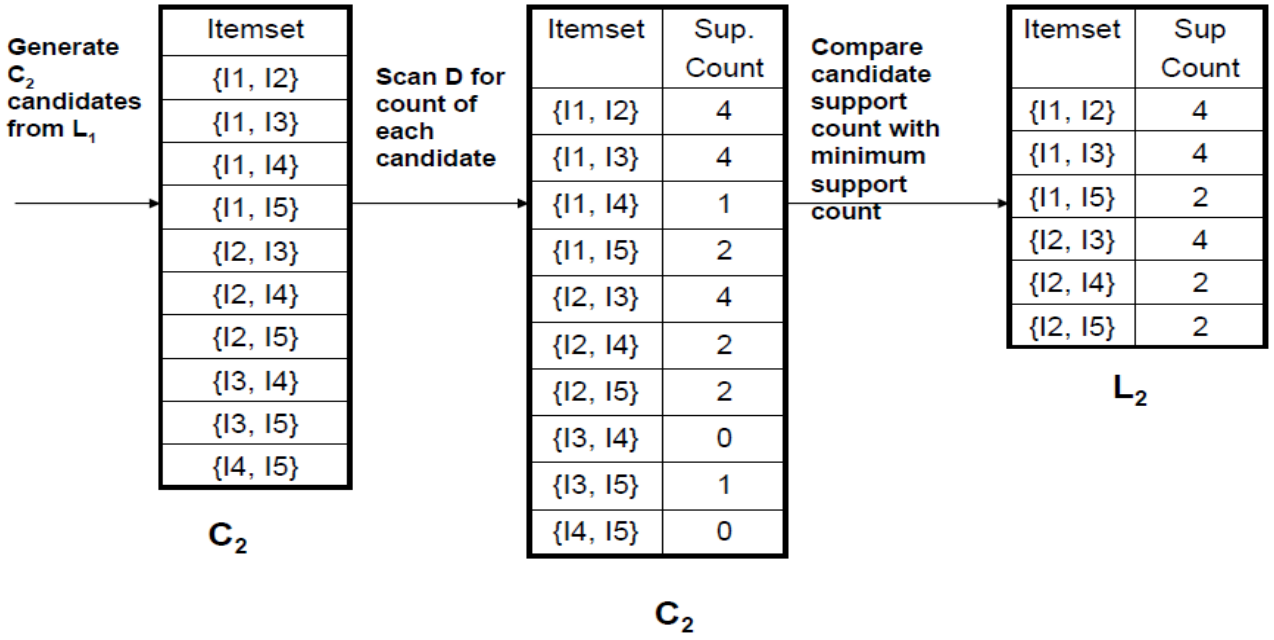
*Solution*

- **Step 1**: Generate 1-itemset Frequent Pattern
- Initially, each item is a member of the set of candidate (C1) itemset. Next, compute support count for each candidate itemset in C1. Frequent 1-itemsets (L1) is then determined by using minimum support.



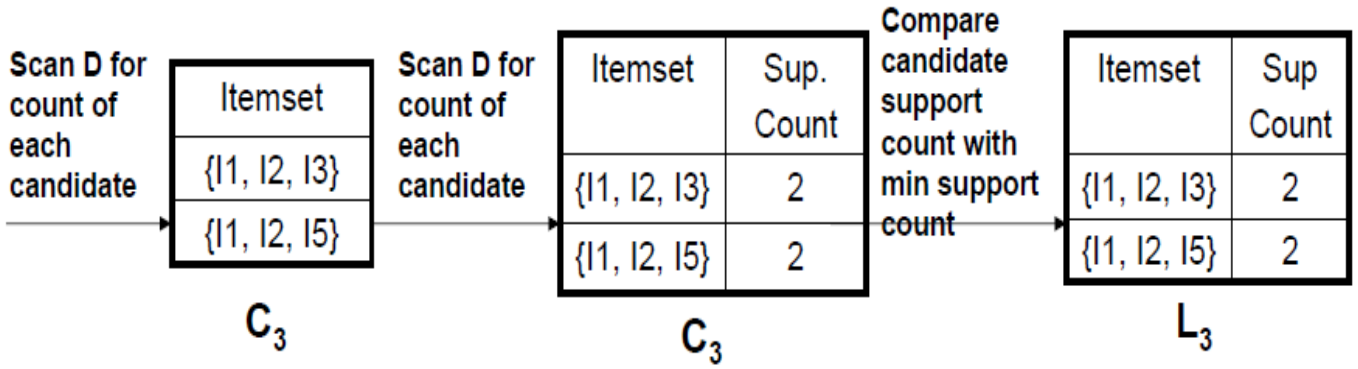
# Apriori Algorithm

- **Step 2**: Generate 2-itemset Frequent Pattern
- Use L1 *Join* L1 to generate a candidate set of 2-itemsets (C2). Next, compute support count for each candidate itemset in C2. Frequent 2-itemsets (L2) is then determined by using minimum support.



# Apriori Algorithm

- **Step 3**: Generate 3-itemset Frequent Pattern
- Generate 3-itemset as C3= L2 *Join* L2. Then use Apriori property to prune the members of C3. Finally generate frequent 3-temset by using minimum support.



# Apriori Algorithm

- **Step 4**: Generate 4-itemset Frequent Pattern
- The algorithm uses L3 *Join* L3 to generate a candidate set of 4-itemsets (C4). Although the join results in {{I1, I2, I3, I5}}, this item set is pruned since its subset {{I2, I3, I5}} is not frequent. Thus, C4= φ, and algorithm terminates, having found all of the frequent items.
- Thus, Frequent Itemsets are: {{I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.

# Apriori Algorithm

- **Step 5:** Generating Association Rules from Frequent Itemsets
- For each frequent item set ***l,*** generate all nonempty subsets of ***l.*** For every nonempty subset ***s*** of ***l,*** output the rule $s \Rightarrow l - s$ if confidence of the rule is greater or equal to minimum confidence.
- We had L = {{I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.
- Let's take ***l*** = {I1, I2, I5}. It's all nonempty subsets are {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, {I5}.

# Apriori Algorithm

- **R1: I1 ^ I2 =>I5** Confidence = 2/4 = 50%, R1 is Rejected.
- **R2: I1 ^ I5 =>I2** Confidence = 2/2 = 100%, R2 is Selected.
- **R3: I2 ^ I5 =>I1** Confidence = 2/2 = 100%, R3 is Selected.
- **R4: I1 =>I2 ^ I5** Confidence = 2/6 = 33%, R4 is Rejected.
- **R5: I2 =>I1 ^ I5** Confidence = 2/7 = 29%, R5 is Rejected.
- **R6: I5 =>I1 ^ I2** Confidence = 2/2 = 100%, R6 is Selected.
- In this way, we have found three strong association rules. We need to repeat same process for every frequent itemset.

# Improving Efficiency of Apriori Algorithm

**Hash Based Technique**

- A hash-based technique can be used to reduce the size of the candidate *k*-itemsets, for $k > 1$.
- For example, when scanning each transaction in the database to generate the frequent 1-itemsets, we can generate all the 2-itemsets for each transaction, hash them into the different *buckets* of a *hash table*.

# Improving Efficiency of Apriori Algorithm

## Hash Based Technique

| 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 |

Create hash table $H_2$ using hash function $h(x, y) = ((order\ of\ x) \times 10 + (order\ of\ y))\ mod\ 7$

$H_2$

| bucket address | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| bucket count | 2 | 2 | 4 | 2 | 2 | 4 | 4 |
| bucket contents | {I1, I4}<br>{I3, I5} | {I1, I5}<br>{I1, I5} | {I2, I3}<br>{I2, I3}<br>{I2, I3}<br>{I2, I3} | {I2, I4}<br>{I2, I4} | {I2, I5}<br>{I2, I5} | {I1, I2}<br>{I1, I2}<br>{I1, I2}<br>{I1, I2} | {I1, I3}<br>{I1, I3}<br>{I1, I3}<br>{I1, I3} |

*If the minimum support count is, say, 3, then the itemsets in buckets 0, 1, 3, and 4 cannot be frequent and so they should not be included in $C_2$.*

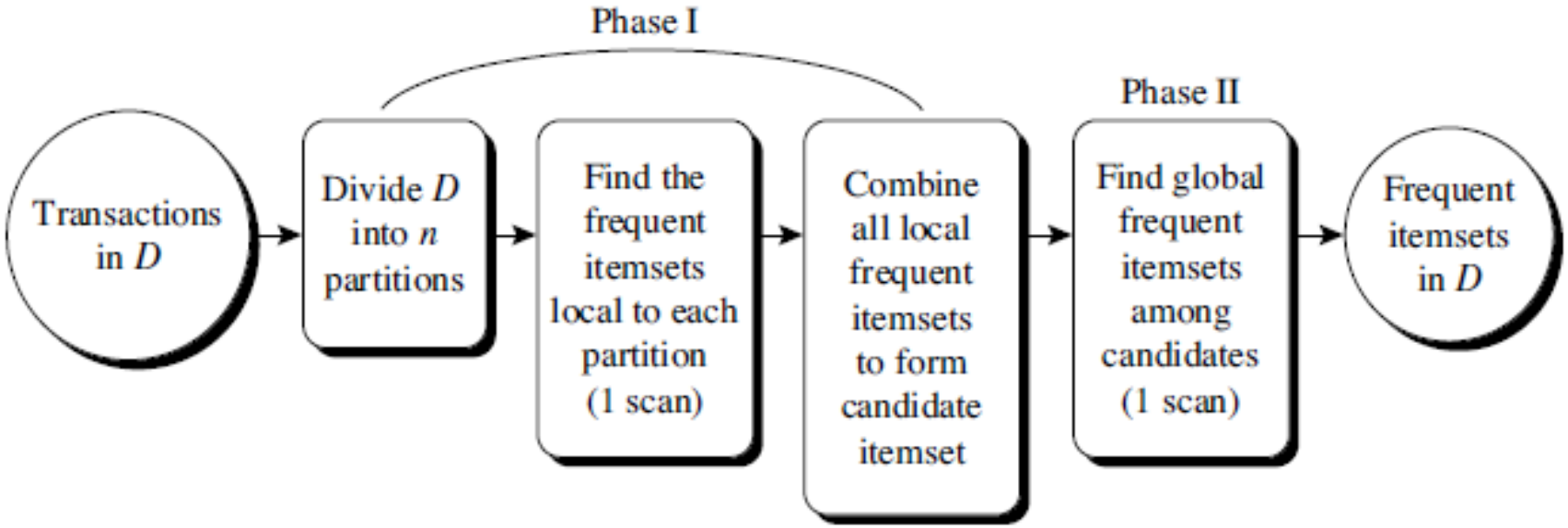
# Improving Efficiency of Apriori Algorithm

**Transaction reduction**

- A transaction that does not contain any frequent *k*-itemsets cannot contain any frequent (*k*+1)-itemsets. Therefore, such a transaction can be marked or removed from further consideration

# Improving Efficiency of Apriori Algorithm

## Partitioning



# Improving Efficiency of Apriori Algorithm

**Sampling**

- The basic idea of the sampling approach is to pick a random sample *S* of the given data *D*, and then search for frequent itemsets in *S* instead of *D*. In this way, we trade off some degree of accuracy against efficiency.
- The *S* sample size is such that the search for frequent itemsets in *S* can be done in main memory, and so only one scan of the transactions in *S* is required overall. Because we are searching for frequent itemsets in *S* rather than in *D*, it is possible that we will miss some of the global frequent itemsets.

# Improving Efficiency of Apriori Algorithm

**Sampling**

- The basic idea of the sampling approach is to pick a random sample *S* of the given data *D*, and then search for frequent itemsets in *S* instead of *D*. In this way, we trade off some degree of accuracy against efficiency.
- The *S* sample size is such that the search for frequent itemsets in *S* can be done in main memory, and so only one scan of the transactions in *S* is required overall. Because we are searching for frequent itemsets in *S* rather than in *D*, it is possible that we will miss some of the global frequent itemsets.

# Improving Efficiency of Apriori Algorithm

**Sampling**

- The basic idea of the sampling approach is to pick a random sample *S* of the given data *D*, and then search for frequent itemsets in *S* instead of *D*. In this way, we trade off some degree of accuracy against efficiency.
- The *S* sample size is such that the search for frequent itemsets in *S* can be done in main memory, and so only one scan of the transactions in *S* is required overall. Because we are searching for frequent itemsets in *S* rather than in *D*, it is possible that we will miss some of the global frequent itemsets.

# Limitations of Apriori Algorithm

- It needs to generate a huge number of candidate sets.
- It may need to repeatedly scan the whole database and check a large set of candidates by pattern matching.

# FP: Growth Algorithm

- The FP-Growth Algorithm is an alternative way to find frequent itemsets without using candidate generations, thus improving performance.
- 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 itemset association information. It uses two step approach:
  - **Step 1:** Build a compact data structure called the FP-tree. It is built using two passes over the data-set.
  - **Step 2:** Extracts frequent itemsets directly from the FP-tree by traversing through FP-Tree

# FP: Growth Algorithm

**Construction of FP-tree**

**Pass1**

- Scan data and find support for each item.
- Discard infrequent items.
- Sort frequent items in decreasing order based on their support

# FP: Growth Algorithm

**Construction of FP-tree**

**Pass2**

- Create the root of the tree, labeled with null
- Scan database again. Process items in each transaction in descending order of support count and create branch for each transaction
- If some transaction shares common prefix with already processed transactions, increment counter for each item in common prefix and create branches for items that are not common.
- To facilitate tree traversal, an item header table is also built so that each item points to its occurrences in the tree via a chain of node-links.

# FP: Growth Algorithm

**Extract Frequent Item-sets from FP-tree**

- Extract frequent patterns from FP-Tree by using bottom-up algorithm from leaves towards the root.
- Construct its conditional pattern base, which is subset of database that contains the set of *prefix paths* in the FP-tree.
- Construct its (*conditional*) FP-tree. Conditional FP-tree of item I is FP-tree constructed only by considering transaction that contains item I and then removing the Item I from all transactions. Then perform mining recursively on such a tree. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

# FP: Growth Algorithm

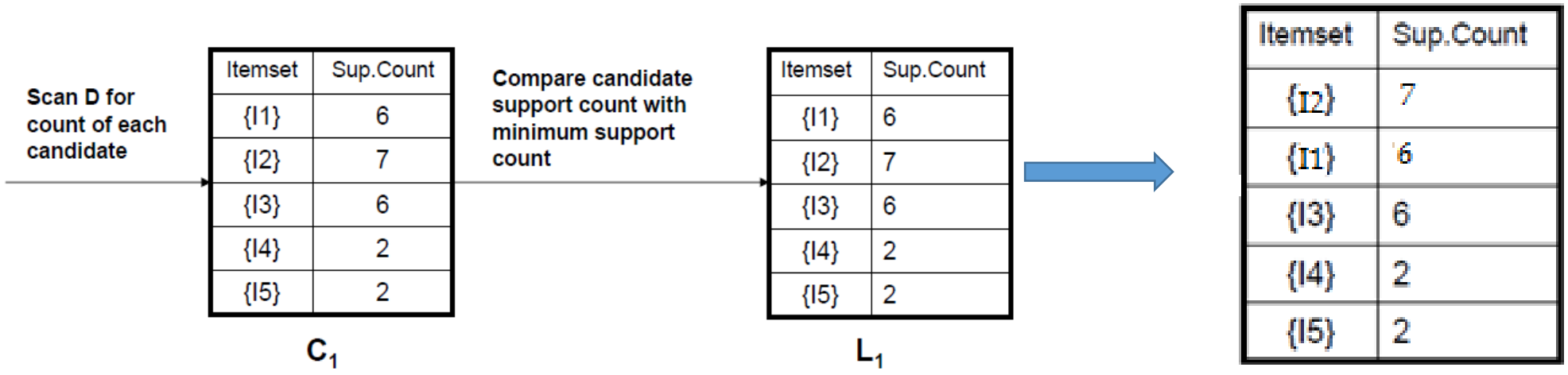
- Example: Consider the database, consisting of 9 transactions. Suppose min. support count required is 2 (i.e. min-sup = 2/9 = 22 %). Let minimum confidence required is 70%. Find out the frequent item sets using FP-growth algorithm. Then generate association rules using min. support & min. confidence.

| *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 Algorithm

## Solution

**Step 1:** Generate 1-itemset Frequent Pattern and then Sort 1-itemset frequent pattern by using minimum support count




| Itemset | Sup.Count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

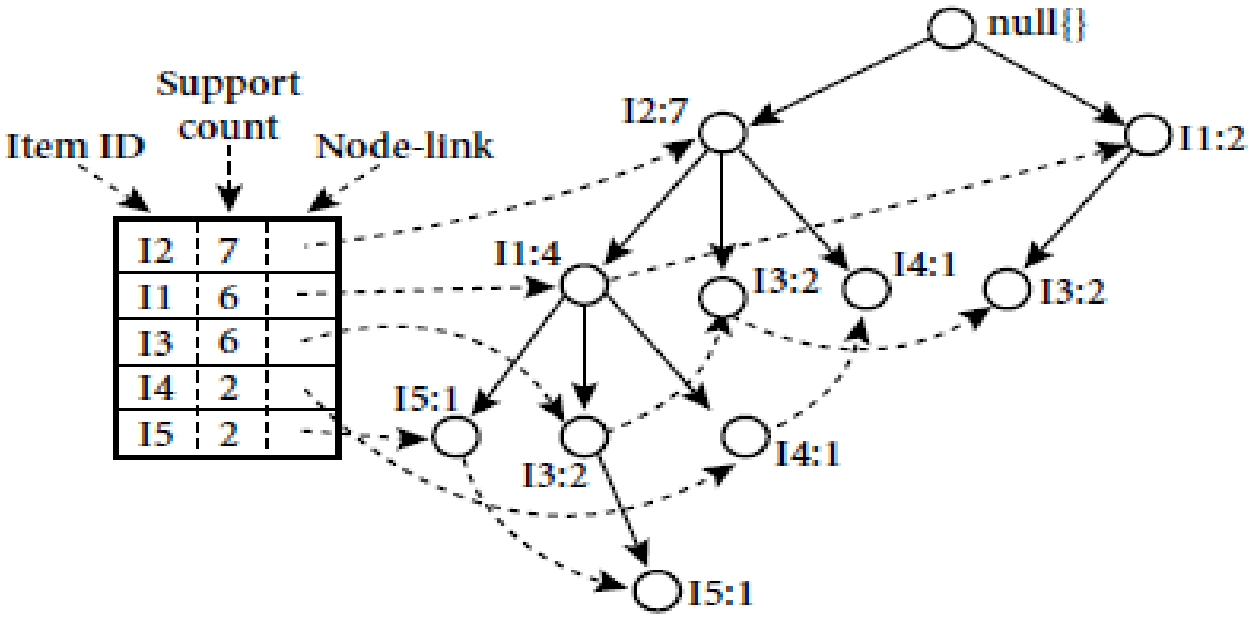
| Itemset | Sup.Count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

| Itemset | Sup.Count |
|---|---|
| {I2} | 7 |
| {I1} | 6 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

# FP: Growth Algorithm

## Solution

**Step 2:** Construct FP-Tree



# FP: Growth Algorithm

## Solution

### Step 3: Mine FP-Tree by using Conditional 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} |

- **Step 4: Generate Association Rules**

  Same as in Apriori algorithm

# Types of Association Rules

**Single Dimensional Association Rules**

- Association rules involving single predicate repeated multiple times are called single dimensional or intra-dimensional rules.
- Consider the examples given below, which is single dimensional association rule:

$$buys(X, \text{“}digital\ camera\text{”})) \Rightarrow buys(X, \text{“}HP\ printer\text{”})$$

# Types of Association Rules

**Multidimensional Association Rules**

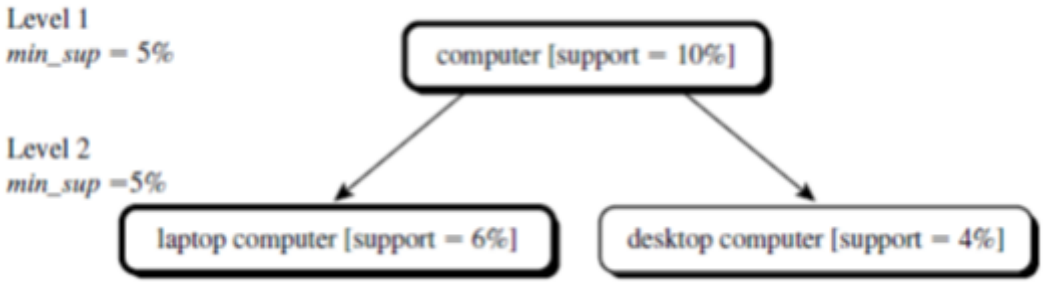
- Association rules that involve two or more predicates are referred as multidimensional association rules.
- Consider the rule given below that contains three predicates (*age, occupation,* and *buys*), each of which occurs *only once* in the rule.

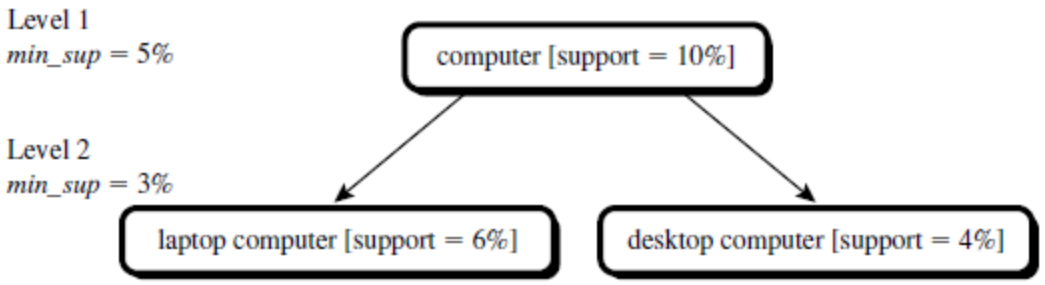
$$age(X, \text{“20:::29”}) \wedge occupation(X, \text{“}student\text{”}) \Rightarrow buys(X, \text{“}laptop\text{”})$$

# Types of Association Rules

## Multilevel Association Rules

- Association rules generated from mining data at multiple levels of abstraction are called multiple-level or multilevel association rules. Multilevel association rules can be mined efficiently using concept hierarchies under a support-confidence framework.



# Types of Association Rules

## Quantitative Association Rules

- Database attributes can be categorical or quantitative. Categorical attributes have a finite number of possible values, with no ordering among the values (e.g., *occupation, brand, color*). Quantitative attributes are numeric and have an implicit ordering among values (e.g., *age, income, price*).
- Association rules mined from quantitative attributes are referred as quantitative association rules.

# From Association Mining to Correlation Analysis

- Most association rule mining algorithms employ a support-confidence framework.
- Although minimum support and confidence thresholds helps to exclude the exploration of a good number of uninteresting rules, many rules so generated are still not interesting to the users.
- This is especially true *when mining at low support thresholds or mining for long patterns*.
- Support-confidence framework can be supplemented with additional interestingness measures based on statistical significance and correlation analysis.

# From Association Mining to Correlation Analysis

- Some association rules {A⇒B} that satisfy minimum support and threshold may be uninteresting if 'A' and 'B' are negatively correlated with each other. This type of rules can be excluded by using the measure correlation analysis. Lift is the measure that measures correlation.
- **Lift:** The lift between the occurrence of $A$ and $B$ can be measured as below:

$$Lift(A,B) = \frac{P(A \cup B)}{P(A)P(B)} = \frac{Confidence(A \Rightarrow B)}{Support(B)}$$

# From Association Mining to Correlation Analysis

- If the resulting value of above equation is less than 1, then the occurrence of $A$ is *negatively correlated with* the occurrence of $B$. If the resulting value is greater than 1, then $A$ and $B$ are *positively correlated*, meaning that the occurrence of one implies the occurrence of the other. If the resulting value is equal to 1, then $A$ and $B$ are *independent* and there is no correlation between them.