UNIT-IV-II
MINING FREQUENT PATTERNS AND ASSOCIATIONS AND CORRELATIONS

Frequent patterns are patterns that appear in a data set frequently and are as follows

- Itemsets
- Subsequences
- Substructures

Suppose 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***.

A *substructure* can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a ***(frequent) structured pattern.***

Association is the two products are associated if they are dependent.

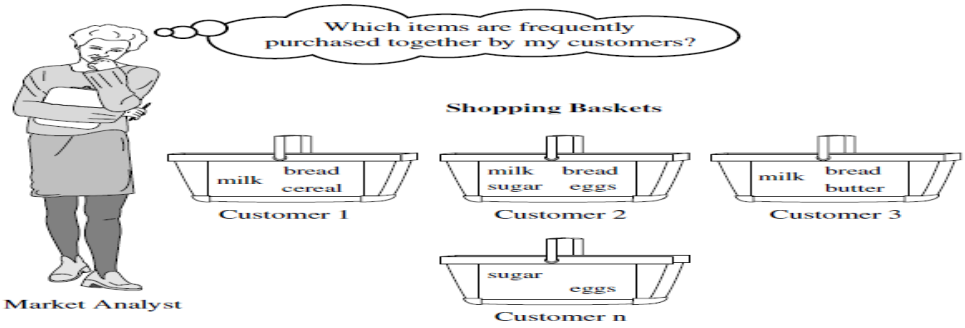
Correlation is a measure which calculates the dependency of the product truly dependent, positive or negative and independent

BASIC CONCEPTS

Frequent pattern mining searches for recurring relationships in a given data set. The basic concepts of frequent pattern mining for the discovery of interesting associations and correlations between item sets in transactional and relational databases.

1. Market Basket Analysis: A Motivating Example

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. For instance, if customers are buying milk, how likely are they to also buy bread (and what kind of bread) on the same trip to the supermarket? Such information can lead to increased sales by helping retailers do selective marketing and plan their shelf space.



2. Frequent Itemsets, Closed Itemsets, and Association Rules:

Frequent Itemsets:

- A set of items is referred to as an **itemset.**
- An itemset that contains *k* items is a ***k*-itemset.**
  Example: The set {*computer, antivirus_ software}* is a 2-itemset.

The **occurrence frequency of an itemset** is the number of transactions that contain the itemset. This is also known as the **frequency, support count**, or **count** of the itemset.

- Frequent itemsets are those that are frequently considering set of items. Each of the itemset will occur atleast as frequently as a pre-determined minimum support count. The set of frequent *k*-itemsets is commonly denoted by L_k.

Closed itemset:

Mining frequent itemset from a large dataset often generates huge number of itemsets satisfying the minimum support especially when minimum support is low. A long itemset will contain maximum number of frequent sub-item sets.

For example, a frequent itemset of length 100, such as $\{a_1, a_2, \ldots, a_{100}\}$, contains $\binom{100}{1} = 100$ frequent 1-itemsets: $a_1, a_2, \ldots, a_{100}$, $\binom{100}{2}$ frequent 2-itemsets: $(a_1, a_2), (a_1, a_3), \ldots, (a_{99}, a_{100})$, and so on. The total number of frequent itemsets that it contains is thus,

$$\binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}.$$

This is too huge a number of itemsets for any computer to compute or store. To overcome this difficulty, we introduce the concepts of ***closed frequent itemset*** and ***maximal frequent itemset.***

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*. An itemset *X* is a **closed frequent itemset** in set *S* if *X* is both closed and frequent in *S*.

An itemset *X* is a **maximal frequent itemset** (or **max-itemset**) in set *S* if *X* is frequent, and there exists no super-itemset *Y* such that *X CY* and *Y* is frequent in *S*.

***Association Rules*:**

Association is the two products are associated if they are dependent. 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.

Association rule mainly uses two measures as **support** and **confidence.**

For example, the information that customers who purchase computers also tend to buy antivirus software at the same time is represented in Association Rule as follows

$$computer \Rightarrow antivirus\_software \ [support = 2\%, confidence = 60\%]$$

A support of 2% for Association Rule means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together and confidence of 60% means that 60% of the customers who purchased a computer also bought the software.

Association rules are considered interesting if they satisfy both a **minimum support threshold** and a **minimum confidence threshold**. Such thresholds can be set by users or domain experts. Measures can be represented as follows

$$support(A \Rightarrow B) = P(A \cup B)$$
$$confidence(A \Rightarrow B) = P(B|A).$$

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

3. Frequent Pattern Mining: A Road Map:

Frequent pattern mining can be classified in various ways, based on the following criteria

i. ***Based on the completeness of patterns to be mined:*** It can mine the complete set of frequent itemsets, the closed frequent itemsets, and the maximal frequent itemsets, given a minimum support threshold. It can also mine constrained frequent itemsets (i.e., those that satisfy a set of user-defined constraints), approximate frequent itemsets (i.e., those that derive only approximate support counts for the mined frequent itemsets), near-match frequent itemsets (i.e., those that tally the support count of the near or almost matching itemsets), top-*k* frequent itemsets (i.e., the *k* most frequent itemsets for a user-specified value, *k*), and so on.

ii. ***Based on the levels of abstraction involved in the rule set:*** Some methods for association rule mining can find rules at differing levels of abstraction.
For example, suppose that a set of association rules mined includes the following rules where *X* is a variable representing a customer:

$$buys(X, \text{“}computer\text{”}) \Rightarrow buys(X, \text{“}HP\_printer\text{”})$$

----→ Rule 1

$$buys(X, \text{“}laptop\_computer\text{”}) \Rightarrow buys(X, \text{“}HP\_printer\text{”})$$

---→ Rule 2

From rule1 and rule2 the items bought are referenced at different levels of abstraction (e.g., “computer” is a higher-level abstraction of “*laptop computer*”). This rule is set as **multilevel association rules**. In a given if do not have reference item, then it is **Single level association rule.**

iii. ***Based on the number of data dimensions involved in the rule:*** If the items or attributes in an association rule reference only one dimension, then it is a **single-dimensional association rule**. For example

$$buys(X, \text{“}computer\text{”}) \Rightarrow buys(X, \text{“}antivirus\_software\text{”})$$

If a rule references two or more dimensions, such as the dimensions *age*, *income*, and *buys*, then it is a **multidimensional association rule**. For example

$$age(X, \text{“}30\ldots39\text{”}) \wedge income(X, \text{“}42K\ldots48K\text{”}) \Rightarrow buys(X, \text{“}high\ resolution\ TV\text{”}).$$

iv. ***Based on the types of values handled in the rule:*** If a rule involves associations between the presence or absence of items, it is a **Boolean association rule**.
Examples:

$$computer \Rightarrow antivirus\_software \ [support = 2\%, confidence = 60\%]$$

$$buys(X, \text{“}computer\text{”}) \Rightarrow buys(X, \text{“}HP\_printer\text{”})$$

$$buys(X, \text{“}laptop\_computer\text{”}) \Rightarrow buys(X, \text{“}HP\_printer\text{”})$$

If a rule describes associations between quantitative items or attributes, then it is a **quantitative association rule**. For example

$$age(X, \text{“}30\ldots39\text{”}) \wedge income(X, \text{“}42K\ldots48K\text{”}) \Rightarrow buys(X, \text{“}high\ resolution\ TV\text{”}).$$

v. ***Based on the kinds of rules to be mined:*** Frequent pattern analysis can generate various kinds of rules and other interesting relationships. Categories include mining association rules and correlation rules.

vi. ***Based on the kinds of patterns to be mined:*** Many kinds of frequent patterns can be mined from different kinds of data sets. It focus also frequent item set mining, sequential pattern mining and structured pattern mining.

EFFICIENT AND SCALABLE FREQUENT ITEMSET MINING METHODS

Many efficient and scalable algorithms have been developed for frequent item set mining, from which association and correlation rules can be derived.

1. **The Apriori Algorithm: Finding Frequent Item sets Using Candidate Generation**

Apriori is a seminal algorithm proposed by R.Agrawal and R.Srikant in 1994 for mining frequent itemsets for Boolean association rules. It is also called the level-wise algorithm.

To improve the efficiency of the level-wise generation of frequent item sets, an important property called the Apriori property is used.

Apriori property: All nonempty subsets of a frequent itemset must also be frequent.

The algorithm is a bottom-up search, moving upward level-wise in the lattice

Procedure is as follows

Step1: It is an influential algorithm for mining frequent item set for Boolean association rules, this uses prior knowledge of item set properties.

Step2: It follows iterative approach known as "level-wise" search where 'k' item sets are used to explore (k+1) item sets.

Step3: First, the set of frequent item sets are found denoted by 'L_1', which is used to find 'L_2' which in turn use to find 'L_3' and so on.

Step4: Apriori property says that "All non-empty subsets of a frequent item sets must also be frequent", it is based on the observations that if an item set 'I' does not satisfy the minimum support threshold, it is not frequent.

Step5: If an item 'A' is added to the item set 'I' then resulting item set (I U A) cannot occur more frequently then 'I' i.e.,(I U A) is not found frequent

Therefore Probability of (I U A)<min_sup

P(I U A)<min_sup

Step6: A two step process is followed in generating candidate and frequent item sets such as join and prune actions.

1. Join step – is defined as for (k-1) item set, 'l_i' means that the items sorted such that $l_i[1]<l_i[2]<……..l_i[k-1]$
   Therefore join of L_{k-1} to L_{k-1} i.e.,$L_{k-1} \bowtie L_{k-1}$ is performed
2. Prune step – is defined as C_k is a superset of 'L_k' i.e., it members may or maynot be frequent, but all of frequent item sets are included in 'C_k'.

Pseudo-code is as follows

C_k: Candidate item set of size k and L_k: Frequent item set of size k

L_1 = {frequent items};

for(k=1;L_k!=Ø;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

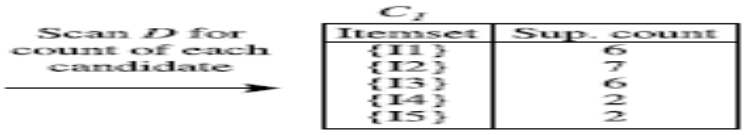
return $L = \cup_k L_k$;

Example1: Consider transactional data for an All Electronics branch as follows and assume min_sup count as 2

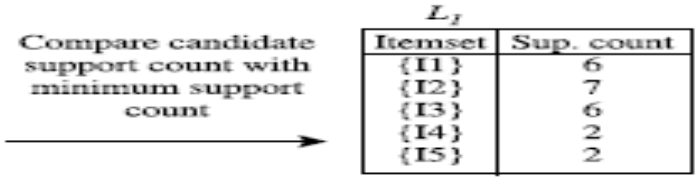
TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

There are nine transactions in the database, that is |D|=9
Apriori algorithm for finding frequent itemsets in D is as follows

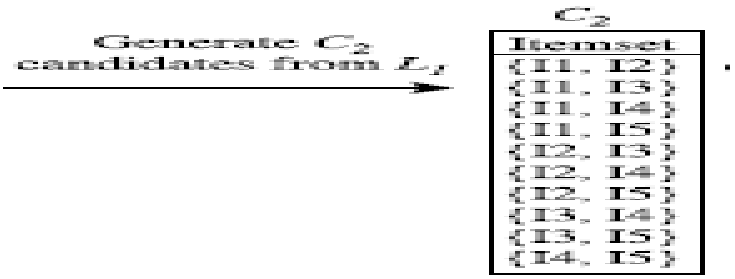
1. In the first iteration of the algorithm, each item is a member of the set of candidate 1- temsets, *C*1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.



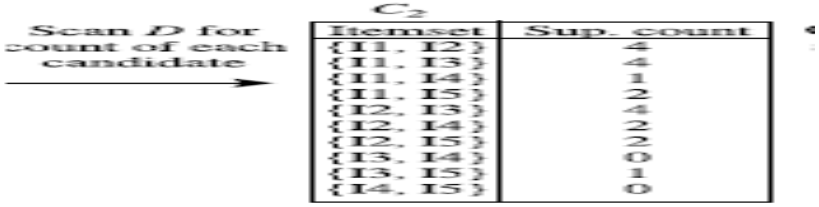
2. Suppose that the minimum support count required is 2, that is, *min sup* = 2. The set of frequent 1-itemsets, *L*1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support.



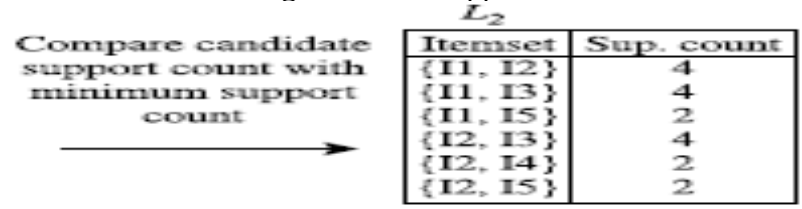
3. Use the join $L1 \bowtie L1$ to generate a candidate set of 2-itemsets, *C*2.



4. The transactions in *D* are scanned and the support count of each candidate itemset in*C*2 is accumulated.



5. The set of frequent 2-itemsets, *L*2, is then determined, consisting of those candidate 2-itemsets in *C*2 having minimum support.



6. Generation the set of candidate 3-itemsets *C*3

$C_3=L_2 \bowtie L_2=\{\{I_1,I_2\},\{I_1,I_3\},\{I_1,I_5\},\{I_2,I_3\},\{I_2,I_4\},\{I_2,I_5\} \bowtie \{I_1,I_2\},\{I_1,I_3\},\{I_1,I_5\},\{I_2,I_3\},\{I_2,I_4\},\{I_2,I_5\}\}$

$=\{\{I_1,I_2,I_3\},\{I_1,I_2,I_5\},\{I_1,I_3,I_5\},\{I_2,I_3,I_4\},\{I_2,I_3,I_5\},\{I_2,I_4,I_5\}\}$

Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent.

Generate C_3 candidates from L_2 →

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

7. The transactions in D are scanned in order to determine $L3$, consisting of those candidate 3-itemsets in $C3$ having minimum support.

Scan D for count of each candidate →

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Compare candidate support count with minimum support count →

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

8. Use $L3 \bowtie L3$ to generate a candidate set of 4-itemsets, $C4$

C4

Generate C4 Candidate from L3 →

Itemset	SupportCount
$\{I_1,I_2,I_3,I_5\}$	1

This itemset is pruned because its subset $\{\{I_1,I_3,I_5\}\}$ in not frequent.

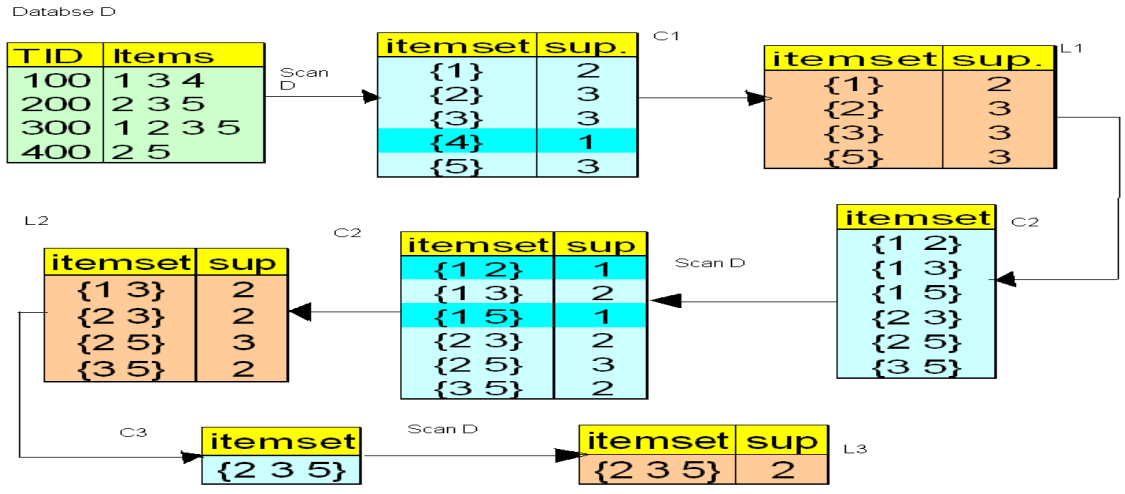
L_4

Itemset	Sup. count
----	------

$L_4=\{\emptyset\}$

Therefore $L=L_1 \cup L_2 \cup L_3$

Example2: Minimum support count be 2 then



$L_4=\{\emptyset\}$

Therefore $L=L_1 U L_2 U L_3$

2. Generating Association Rules from Frequent Itemsets:

Once the frequent itemsets from transactions in a database *D* have been found, it is straightforward to generate strong association rules from them

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

The conditional probability is expressed in terms of itemset support count, where *Support_ count*(*AUB*) is the number of transactions containing the itemsets *AUB*, and *Support_ count*(*A*) is the number of transactions containing the itemset *A*. Based on this equation, association rules can be generated as follows:

- For each frequent itemset *l*, generate all nonempty subsets of *l*.
- For every nonempty subset *s* of *l*, output the rule

$$\text{“}s \Rightarrow (l-s)\text{” if } \frac{support\_count(l)}{support\_count(s)} \geq min\_conf$$

where *min conf* is the minimum confidence threshold.

Example: Generating association rules for AllElectronics

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

Suppose data contains frequent itemset $l = \{I1, I2, I5\}$

Nonempty subsets of l are

$\{I1, I2\}$, $\{I1, I5\}$, $\{I2, I5\}$, $\{I1\}$, $\{I2\}$, and $\{I5\}$.

Resulting association rules are as follows, each listed with its confidence

$I1 \wedge I2 \Rightarrow I5,$

Confidence= support_count{I1,I2,I5}/ support_count{I1,I2} =2/4=50%

So it is rejected

$I1 \wedge I5 \Rightarrow I2,$

Confidence=support_count{I1,I2,I5}/support_count{I1,I5}=2/2=100%

So it is selected

$I2 \wedge I5 \Rightarrow I1$

Confidence=support_count{I1,I2,I5}/support_count{I2,I5}=2/2=100%

So it is selected

$I1 \Rightarrow I2 \wedge I5$

Confidence= support_count{I1,I2,I5}/support_count{I1}=2/6=33%

So it is rejected

$I2 \Rightarrow I1 \wedge I5$

Confidence= support_count{I1,I2,I5}/support_count{I2}=2/7=29%

So it is rejected

$I5 \Rightarrow I1 \wedge I2$

Confidence= support_count{I1,I2,I5}/support_count{I5}=2/2=100%

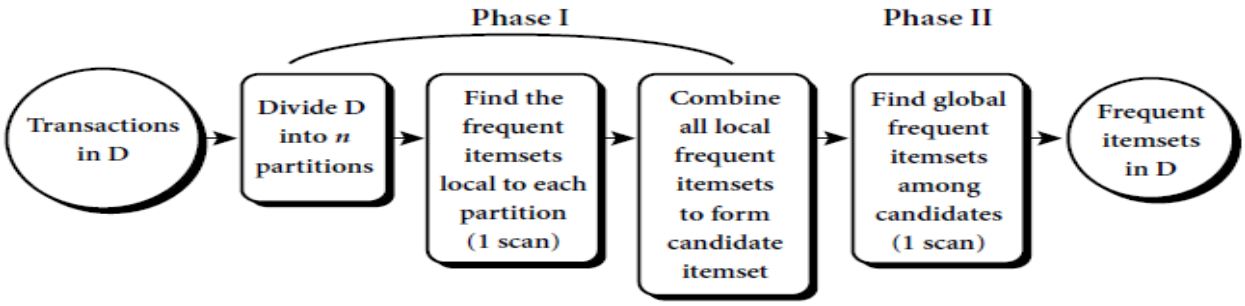
So it is selected

In this way there are **three strong association rules**.

3. Improving the Efficiency of Apriori:

Many variations of the Apriori algorithm have been proposed that focus on improving the efficiency of the original algorithm. Several of these variations are summarized as follows:

- **Hash-based itemset counting:** A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent.
- **Transaction reduction**: A transaction that does not contain any frequent k-itemset is useless in subsequent scans.
- **Partitioning:** Any itemset that is potentially frequent in database must be frequent in atleast one of the partitions of database. For example consists two phases as follows



- **Sampling**: Mining on a subset of given data, lower support threshold plus a method to determine the completeness.
- **Dynamic itemset counting**: Add new candidate itemsets only when all of their subsets are estimated to be frequent.

4. Mining Frequent Itemsets without Candidate Generation:

Apriori candidate generate-and-test method significantly reduces the size of candidate sets, leading to good performance gain, but it can suffer from two nontrivial costs:

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

A method that mines the complete set of frequent itemsets **without candidate generation** is **frequent-pattern growth**, or simply **FP-growth**, which adopts a *divide-and-conquer* strategy as follows

- It compresses the database representing frequent items into a **frequent-pattern tree**, or **FP-tree**,which retains the itemset association information.
- It then divides the compressed database into a set of ***conditional databases*** each associated with one frequent item or "pattern fragment," and mines each such database separately.
- Scan the database to derive set of itemsets and their support_counts. The resulting set is denoted as 'L' in descending support_count order.
- Create root of the tree labeled with null. The items in each transaction are processed in 'L' order and a branch is created for each transaction.

Construction of FP-tree is as follows

- First, create the **root** of the tree, labeled with "**null**".
- Scan the database D a **second time.** (First time scan it to create 1-itemset and the L)
- The items in each transaction are processed in L order (i.e., sorted order)
- A branch is created for **each transaction** with items having their support cont separated by colon.
- Whenever the same node is encountered in another transaction, just **increment** the support count of the same node or prefix.
- To facilitate tree traversal, **an item header table** is built so that each item points to its occurrences in the tree via a chain of node-links

- Now, **the problem of mining frequent patterns in database is transformed to that of mining the FP-tree**.

Mining the FP-tree by creating conditional (sub) pattern bases is as follows

1. Start from each **frequent length-1 pattern** (as as initial suffix pattern)
2. Construct its **conditional pattern base** which consists of the set of prefix paths in the FP-tree co-occurring with suffix pattern.
3. Then, construct its **conditional FP-tree** and perform mining on such a tree.
4. The **pattern growth** is achieved by concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.
5. The union of all frequent patterns (generated by step4) gives the **required frequent itemset.**

Example: Finding frequent itemsets without candidate generation i.e., FP-growth algorithm

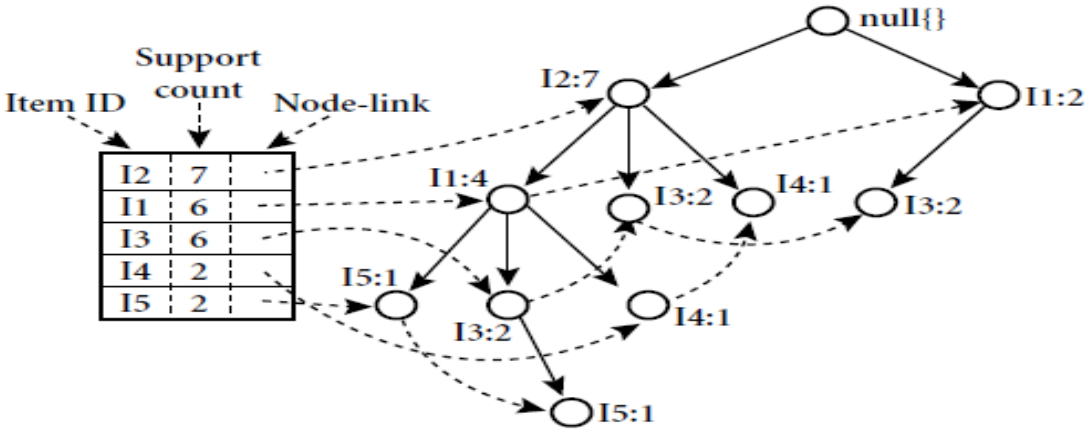
TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

- In this database D there are 9 transactions
- Suppose minimum support is 2.
- First scan the database as Apriori, which derives the set of 1-itemsets and their support counts.
- The set of frequent items is sorted in the order of descending support count.

1. Placing the items in L-order (descending order or less min_count)

ItemID	Sup_count
I2	7
I1	6
I3	6
I4	2
I5	2

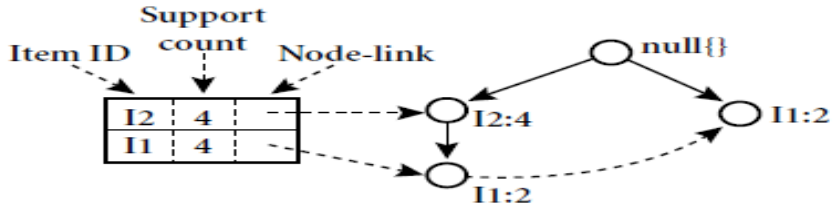
2. Construction of FP-tree registers compresses, frequent pattern information



3. 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}

- Start from I5. The I5 is involved in 2 branches namely {I2,I1,I5:1} and {I2,I1,I3,I5:1}. Therefore considering 15 as suffix, its corresponding prefix paths would be {I2,I1:1} and {I2,I1,I3:1}, which forms its conditional pattern base. Out of these, only I1 and I2 is selected in the conditional FP-tree because I3 is not satisfying the minimum support count.
  For I1, support count in conditional pattern base = 1+1=2
  For I2, support count in conditional pattern base = 1+1=2
  For I3, support count in conditional pattern base = 1
  Thus support count for I3 is less than required min_sup which is 2 here.
  Frequent patterns {I2,I5:2}, {I1,I5:2}, {I2,I1,I5:2}
- The I4 prefix path from conditional pattern base {I2,I1:1},{I2:1} has a single node conditional FP-tree {I2:2} and derives one frequent pattern {I2,I1:2}.
- The I3 conditional pattern base is {{I2,I1:2},{I2:2},{I1:2}}. Conditional FP-tree has two brances {I2:4,I1:2} and {I1:2}.



Generates set of patterns {I2,I3:4}, {I1,I3:4},{I2,I1,I3:2}

- The I1 conditional pattern base is {I2:4}, whose FP-tree contains only one node {I2:4}, which generates one frequent pattern {I2,I1:4}.

5. Mining Frequent Itemsets Using Vertical Data Format(VDF):
(ECLAT (Equivalence CLASS Transformation)

Both the Apriori and FP-growth methods mine frequent patterns from a set of transactions in *TID-itemset* format (that is, {*TID* : *itemset}*), where *TID* is a transaction-id and *itemset* is the set of items bought in transaction *TID*. This data format is known as **horizontal data format**.

T_{ID}_Itemset => {T_{ID}:Itemset}

Alternatively, data can also be presented in *item-TID set* format (that is, {*item* : *TID_ set}*), where *item* is an item name, and *TID_ set* is the set of transaction identifiers containing the item. This format is known as **vertical data format**.

Item_T_{ID}_set => {Item:T_{ID}_set}

Example: Consider transactional data for an AllElectronics branch as follows

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

1. The vertical data format of the transaction data set *D* of

itemset	*TID_set*
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

2. Mining frequent itemsets using Vertical data format. Mining can be performed by intersecting T_{ID}sets of every pair of frequent items with minimum support (min_sup) count as 2. The 2-itemset in Vertical data format as

itemset	*TID_set*
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3,I4}	{ }
{I3,I5}	{T800}
{I4,I5}	{ }

3.Notice that because the itemsets {I1,I4} and {I3,I5} each contains only one transaction, they do not belong to the set of frequent 2-itemsets.

itemset	*TID_set*
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

If everyone of 2-itemset is frequent then generate 3-itemset in vertical format

itemset	*TID_set*
{I1,I2,I3}	{T800,T900}
{I1,I2,I5}	{T100,T800}
{I1,I2,I4}	{T400}
{I2,I3,I4}	{ }
{I2,I3,I5}	{T800}
{I2,I4,I5}	{ }

===>

itemset	*TID_set*
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

Therefore, the sup_count of an itemset is simply the length of TID_set of items. So, VDF specifies association between items as well as transactions.

6. Mining Closed Frequent Itemsets:

A recommended methodology is to search for closed frequent itemsets directly during the mining process. This requires us to prune the search space as soon as we can identify the case of closed itemsets during mining. Pruning strategies include the following:

- **Item merging**: If every transaction containing a frequent itemset X also contains an itemset Y but not any proper superset of Y, then X [Y forms a frequent closed itemset and there is no need to search for any itemset containing X but no Y.

- **Sub-itemset pruning**: If a frequent itemset X is a proper subset of an already found frequent closed itemset Y and support count(X) = support count(Y), then X and all of X's descendants in the set enumeration tree cannot be frequent closed itemsets and thus can be pruned.
- **Item skipping**: In the depth-first mining of closed itemsets, at each level, there will be a prefix itemset X associated with a header table and a projected database. If a local frequent item p has the same support in several header tables at different levels, we can safely prune p from the header tables at higher levels.

Besides pruning the search space in the closed itemset mining process, another important optimization is to perform efficient checking of a newly derived frequent itemset to see whether it is closed, because the mining process cannot ensure that every generated frequent itemset is closed.

When a new frequent itemset is derived, it is necessary to perform two kinds of closure checking:

(1) ***superset checking:*** checks if this new frequent itemset is a superset of some already found closed itemsetswith the same support, and

(2) ***subset checking:*** checks whether the newly found itemset is a subset of an already found closed itemset with the same support

MINING VARIOUS KINDS OF ASSOCIATION RULES

Association rules are mainly used because

- Express how items or objects are related to each other, and how they tend to group together
- Simple to understand (comprehensibility).
- Provide useful information (utilizability)
- Efficient discovery algorithms exist (efficiency)

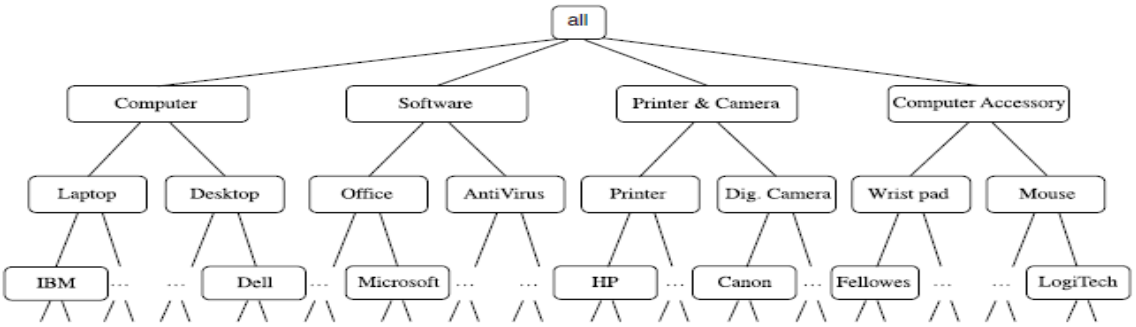
Different kinds of association rules are as follows

- **Multilevel association rule**: Associations between items or attributes at different levels of abstraction. i.e., at different levels of concept hierarchy.
- **Multidimensional association rule**: It involves more than one dimension or predicate.
- **Quantitative association rule**: It involves numeric attributes that have an implicit ordering among value.

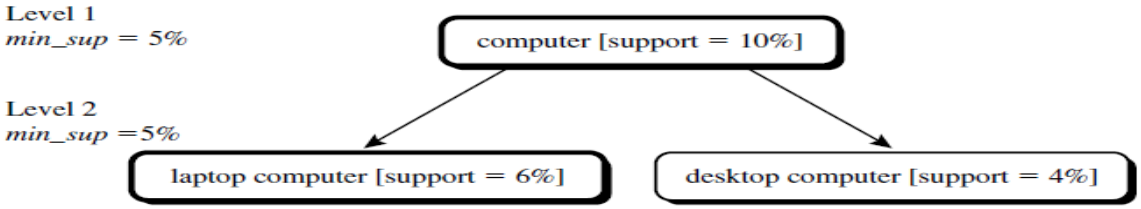
1. Mining Multilevel Association Rules:

Items often form hierarchies. Items at the lower level are expected to have lower support. Rules regarding itemsets at appropriate levels of the hierarchy could be quite useful. It can be mined using concept hierarchies under a support and confidence. It follows top down strategy.

Example: A concept hierarchy for *AllElectronics* computer items

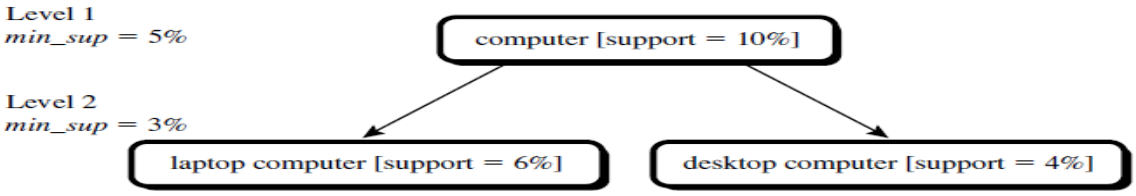


Using uniform minimum support for all levels (referred to as uniform support): All the levels have same support count (min_sup) which is used when mining at each level of abstraction. When a uniform minimum support threshold is used, the search procedure is simplified. The method is also simple in that users are required to specify only one minimum support threshold. An Apriori-like optimization technique can be adopted, based on the knowledge that an ancestor is a superset of its descendants.



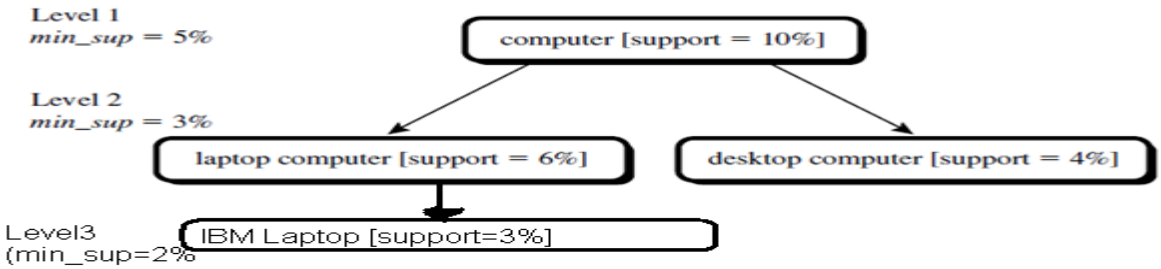
From above figure a minimum support threshold of 5% is used throughout (e.g., for mining from *"computer"* down to *"laptop computer"*). Both *"computer"* and *"laptop computer"* are found to be frequent, while *"desktop computer"* is not. (4% < 5%)

- **Using reduced minimum support at lower levels (referred to as reduced support):** Each level of abstraction has its own minimum support threshold. The deeper the level of abstraction, the smaller the corresponding threshold is.



From figure the minimum support thresholds for levels 1 and 2 are 5% and 3%, respectively, i.e.,min_sup is reduced there is possibility of making the item1 frequent. In this way, *"computer," "laptop computer,"* and *"desktop computer"* are all considered frequent.

- **Using item or group-based minimum support (referred to as group-based support):** Using item or group based it is sometimes more desirable to set up user-specific, item, or group based minimal support thresholds when mining multilevel rules. For example, a user could set up the minimum support thresholds based on product price, or on items of interest, such as by setting particularly low support thresholds



2. Mining Multidimensional Association Rules from Relational Databases and DataWarehouses:

- If an association rule contains a single predicate i.e., the rule which carries exactly one attribute or two attributes with no two predicates on the left side is called **Single dimensional association rule** or **intradimensional association rule**.
  Example: Single predicate is buys as follows
  $$buys(X, \text{“}digital\ camera\text{”}) \Rightarrow buys(X, \text{“}HP\ printer\text{”}).$$
- **Multidimensional association rule** involves more than one dimension or predicate. This rule involve two or more dimensions, where each of which occurs only once in the rule and hence it has no repeated predicates. Example is as follows
  $$age(X, \text{“}20...29\text{”}) \wedge occupation(X, \text{“}student\text{”}) \Rightarrow buys(X, \text{“}laptop\text{”})$$
- Multidimensional association rules with no repeated predicates are called **interdimensional association rules.**
- Mining multidimensional association rules with repeated predicates, which contain multiple occurrences of some predicates. These rules are called **hybrid-dimensional association rules**. Example is as follows
  $$age(X, \text{“}20...29\text{”}) \wedge buys(X, \text{“}laptop\text{”}) \Rightarrow buys(X, \text{“}HP\ printer\text{”})$$

The 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*). Categorical attributes are also called nominal attributes, because their values are “names of things.”
- **Quantitative attributes** are numeric and have an implicit ordering among values (e.g., *age, income, price*).

Mining Quantitative Association Rules:

Quantitative association rules are multidimensional association rules in which the numeric attributes are *dynamically* discretized during the mining process so as to satisfy s ome mining criteria, such as maximizing the confidence or compactness of the rules mined.

Quantitative attributes can have a very wide range of value defining their domain. The partitioning process is referred to as **binning(**Quantitative attributes can have a very wide range of values defining their domain), that is, where the intervals are considered “bins.”

Three common binning strategies area as follows:

- **Equal-width binning**, where the interval size of each bin is the same
- **Equal-frequency binning**, where each bin has approximately the same number of tuples assigned to it,
- **Clustering-based binning,** where clustering is performed on the quantitative attribute to group *neighboring points* (judged based on various distance measures) into the same bin

$$age(X, 34) \wedge income(X, \text{“}31K...40K\text{”}) \Rightarrow buys(X, \text{“}HDTV\text{”})$$
$$age(X, 35) \wedge income(X, \text{“}31K...40K\text{”}) \Rightarrow buys(X, \text{“}HDTV\text{”})$$
$$age(X, 34) \wedge income(X, \text{“}41K...50K\text{”}) \Rightarrow buys(X, \text{“}HDTV\text{”})$$
$$age(X, 35) \wedge income(X, \text{“}41K...50K\text{”}) \Rightarrow buys(X, \text{“}HDTV\text{”}).$$

Replace four rules as

$$age(X, \text{“}34...35\text{”}) \wedge income(X, \text{“}31K...50K\text{”}) \Rightarrow buys(X, \text{“}HDTV\text{”})$$

ARCS works as follows

1. Binning
2. Find frequent predicate set
3. Clustering

4. Optimize

FROM ASSOCIATION MINING TO CORRELATION ANALYSIS

Association rule mining employs support-confidence work. When mining long patterns at low support threshold many rule generated which are not stored in database and also not interesting. Hence

"All association rules are not strong rules"

"Strong rules are not necessarily interesting"

Example: Suppose the transaction s of purchasing computer games and videos are analyzed such that of the 10,000 transaction includes computer games, while 7,500 included videos and 4000 included games and videos. Suppose a data mining program for discovering association rules is run on the data, using a minimum support of say 40% and a minimum confidence of 60%.

Single Association rules is as

$$buys(X, \text{“computer games”}) \Rightarrow buys(X, \text{“videos”}) \quad [support = 40\%, confidence = 66\%]$$

Above rule is strong association rule and would therefore be reported, since its support value of 4000/10000=40% and confidence value of 4000/6000=66% satisfy the minimum support and minimum confidence value.

From Association Analysis to Correlation Analysis:

As the measures of support and confidence measures are insufficient at filtering out uninteresting association rules, a correlation measure can be used to augment the support-confidence framework for association rules. This leads to ***correlation rules*** of the form

$$A \Rightarrow B \; [support, confidence. correlation].$$

That is, a correlation rule is measured not only by its support and confidence but also by the correlation between itemsets *A* and *B*. Various correlation measures to determine which would be good for mining large data sets are as follows

1. Lift Measure: Lift is a simple correlation measure that is given as follows. The occurrence of itemset *A* is independent of the occurrence of itemset *B* if $P(A \cup B) = P(A)P(B)$; otherwise, itemsets *A* and *B* are dependent and correlated as events. This definition can easily be extended to more than two itemsets. The lift between the occurrence of *A* and *B* can be measured by computing

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)}.$$

- If the resulting value 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.

Example: Among 5000students 3000 plays basketball, 3750 eat cereal and finally 2000 plays basketball and eat cereal.

- *play basketball* ⇒ *eat cereal* [40%, 66.7%] is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%.
- *play basketball* ⇒ *not eat cereal* [20%, 33.3%] is far more accurate, although with lower support and confidence

Measure the correlated/dependent events by using lift as follows

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

$$lift(B, C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89$$

$$lift(B, \neg C) = \frac{1000/5000}{3000/5000 * 1250/5000} = 1.33$$

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

2. **Correlation analysis by χ^2:**

The observed and expected values for each start of the table [rows/columns].

$$\chi^2 = \Sigma \frac{(observed - expected)^2}{expected}$$

Example:

	game	$\overline{game}$	Σ_{row}
video	4,000 (4,500)	3,500 (3,000)	7,500
$\overline{video}$	2,000 (1,500)	500 (1,000)	2,500
Σ_{col}	6,000	4,000	10,000

To compute the correlation using χ^2 analysis, the observed value and expected value (displayed in parenthesis) for each slot of the above table. From above compute the χ^2 value as follows

$$\chi^2 = \Sigma \frac{(observed - expected)^2}{expected} = \frac{(4{,}000 - 4{,}500)^2}{4{,}500} + \frac{(3{,}500 - 3{,}000)^2}{3{,}000} + \frac{(2{,}000 - 1{,}500)^2}{1{,}500} + \frac{(500 - 1{,}000)^2}{1{,}000} = 555.6.$$

Because the χ^2 value is greater than one, and the observed value of the slot (*game*, *video*) = 4,000, which is less than the expected value 4,500, *buying game* and *buying video* are *negatively correlated*

3. Other Correlation measures:

The **all_confidence** value is as follows

Given an itemset $X = \{i1, i2, : : : , ik\}$, the all confidence of X is defined as

$$all\_conf(X) = \frac{sup(X)}{max\_item\_sup(X)} = \frac{sup(X)}{max\{sup(i_j) | \forall i_j \in X\}},$$

where $max\{sup(i_j)|\forall i_j \in X\}$ is the maximum (single) item support of all the items in X, and hence is called the **max_item_sup** of the itemset X. The *all_confidence* of X is the minimal confidence among the set of rules $i_j \rightarrow X - i_j$, where $i_j \in X$.

The **cosine** measure is as follows

$$cosine(A, B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}}$$

The *cosine* measure can be viewed as a harmonized *lift* measure: the two formulae are similar except that for cosine; the *square root* is taken on the product of the probabilities of *A* and *B*. This is an important difference, however, because by taking the square root, the cosine value is only influenced by the supports of *A*, *B*, and *A* U *B*, and not by the total number of transactions.

CONSTRAINT-BASED ASSOCIATION MINING

It is difficult to the mining process to cover all the rules from a given set of data, most of which ends up being unrelated to the users thus a good heuristic is to have the users to specify some expectations as "**constraints**" to confine the search space. This strategy is known as **Constraint based mining**. The constraint can include the following

- **Knowledge type constraints**: These specify the type of knowledge to be mined, such as association or correlation.
- **Data constraints**: These specify the set of task-relevant data.
- **Dimension/level constraints**: These specify the desired dimensions (or attributes) of the data, or levels of the concept hierarchies, to be used in mining.
- **Interestingness constraints**: These specify thresholds on statistical measures of rule interestingness, such as support, confidence, and correlation.
- **Rule constraints**: These specify the form of rules to be mined

Metarule-Guided Mining of Association Rules: Metarules allow users to specify the syntactic form of rules that they are interested in mining. The rule forms can be used as constraints to help improve the efficiency of the mining process. Metarules may be based on the analyst's experience, expectations, or intuition regarding the data or may be automatically generated based on the database schema.

$P_1(X,Y) \wedge P_2(X,W) \Rightarrow P(X,Z)$

Where P_1,P_2 are predicate sets or attributes X is a variable representing a customer Y,W,Z tables values of the attributes.

Example:

$$P_1(X, Y) \wedge P_2(X, W) \Rightarrow buys(X, \text{"office software"}).$$

where *P*1 and *P*2 are predicate variables that are instantiated to attributes from the given database during the mining process, *X* is a variable representing a customer, and *Y* and *W* take on values of the attributes assigned to *P*1 and *P*2, respectively

A metarule forms a hypothesis regarding the relationships that the user is interested in probing or confirming. The data mining system can then search for rules that match the given metarule

$$age(X, \text{"30...39"}) \wedge income(X, \text{"41}K\text{...60}K\text{"}) \Rightarrow buys(X, \text{"office software"})$$

For mining interdimensional association rules, the metarule is a rule template of the form

$$P_1 \wedge P_2 \wedge \cdots \wedge P_l \Rightarrow Q_1 \wedge Q_2 \wedge \cdots \wedge Q_r$$

where *Pi* (*i*=1,. . . , *l*) and *Qj* (*j* =1,. . . , *r*) are either instantiated predicates or predicate variables. Let the number of predicates in the metarule be $p = l + r$.

Constraint Pushing: Mining Guided by Rule Constraints

Rule constraints specify expected set/subset relationships of the variables in the mined rules, constant initiation of variables, and aggregate functions. Rule constraints can be classified into five categories with respect to frequent itemset mining

1. **Antimonotonic**: If an itemset doesnot satisfy the value constraint, then none of the supersets can satisfy the constraint. So it is called antimonotonic.
   Example:
   $\min(price) \geq 500$, and $count(I) \leq 10$,
   Any itemset that violates either of these constraints can be discarded. Since adding more items to such constraint can never satisfy the constraints.
2. **Monotonic:** If an itemset satisfy the rule constraint,
   $\min(price) \geq 500$,
   With single condition then it is monotonic.
3. **Succint**: Enumerate all and only those sets that are guaranteed to satisfy the constraint. Directly generate precisely the sets that satisfy it.
4. **Convertible**: if the items in the itemset are arranged in a particular order, the constraint may become monotonic or antimonotonic with regard to the frequent itemset mining process.
5. **Inconvertible**: Every constraint is not convertible and each element in a set would be of any real value which are not convertible, such constraints are inconvertible.

Characterizations of commonly used SQL-based constraints are as follows

Constraint	Antimonotonic	Monotonic	Succinct
$v \in S$	no	yes	yes
$S \supseteq V$	no	yes	yes
$S \subseteq V$	yes	no	yes
$min(S) \leq v$	no	yes	yes
$min(S) \geq v$	yes	no	yes
$max(S) \leq v$	yes	no	yes
$max(S) \geq v$	no	yes	yes
$count(S) \leq v$	yes	no	weakly
$count(S) \geq v$	no	yes	weakly
$sum(S) \leq v \ (\forall a \in S, a \geq 0)$	yes	no	no
$sum(S) \geq v \ (\forall a \in S, a \geq 0)$	no	yes	no
$range(S) \leq v$	yes	no	no
$range(S) \geq v$	no	yes	no
$avg(S) \ \theta \ v, \theta \in \{\leq, \geq\}$	convertible	convertible	no
$support(S) \geq \xi$	yes	no	no
$support(S) \leq \xi$	no	yes	no
$all\_confidence(S) \geq \xi$	yes	no	no
$all\_confidence(S) \leq \xi$	no	yes	no