CHAPTER 4

Unit IV

Association Rules Mining

Syllabus Topics

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 Various Kinds of Association Rules : Mining multilevel association rules, constraint based association rule mining, Meta rule-Guided Mining of Association Rules.

Syllabus Topic : Market Basket Analysis

4.1 Market Basket Analysis

4.1.1 What is Market Basket Analysis?

- Market basket analysis is a modelling technique which is also called as affinity analysis, it helps identifying which items are likely to be purchased together.
- The market-basket problem assumes we have some large number of items, e.g., "bread", "milk.", etc. Customers buy the subset of items as per their need and marketer gets the information that which things customers have taken together. So the marketers use this information to put the items on different position.
- **For Example :** If someone buys a packet of milk also tends to buy a bread at the same time

  Milk => Bread
- Market basket analysis algorithms are straightforward; difficulties arise mainly in dealing with large amounts of transactional data, where after applying algorithm it may give rise to large number of rules which may be trivial in nature.

4.1.2 How is it Used ?

- Market basket analysis is used in deciding the location of items inside a store, for e.g. if a customer buys a packet of bread he is more likely to buy a packet of butter too, keeping the bread and butter next to each other in a store would result in customers getting tempted to buy one item with the other.
- The problem of large volume of trivial results can be overcome with the help of differential market basket analysis which enables in finding interesting results and eliminates the large volume.
- Using differential analysis it is possible to compare results between various stores, between customers in various demographic groups.
- Some special observations among the rules for e.g. if there is a rule which holds in one store but not in any other (or vice versa) then it may be really interesting to note that there is something special about that store in the way it has organized its items inside the store may be in a more lucrative way. These types of insights will improve company sales.
- Identification of sets of items purchases or events occurring in a sequence, something that may be of interest to direct marketers, criminologists and many others, this approach may be termed as Predictive market basket analysis.

4.1.3 Applications of Market Basket Analysis

➔ (SPPU - Dec. 17)

Q. Explain applications of Market baskets analysis.

Dec. 17, 4 Marks

- Credit card transactions done by a customer may be analysed.
- Phone calling patterns may be analysed.
- Fraudulent Medical insurance claims can be identified.
- For a financial services company :
  - Analysis of credit and debit card purchases.
  - Analysis of cheque payments made.
  - Analysis of services/products taken e.g. a customer who has taken executive credit card is also likely to take personal loan of $5,000 or less.
- For a telecom operator :
  - Analysis of telephone calling patterns.
  - Analysis of value-added services taken together. Rather than considering services taken together at a point in time, it could be services taken over a period of, let's say, six months.
- Various ways can be used to apply market basket analysis :
  - Special combo offers may be offered to the customers on the products sold together.
  - Placement of items nearby inside a store which may result in customers getting tempted to buy one product with the other.
  - The layout of catalogue of an ecommerce site may be defined.
  - Inventory may be managed based on product demands.

Syllabus Topic : Frequent Itemsets

4.2 Frequent Itemsets

- An itemset X is *frequent* if X's support is no less than a *minimum support* threshold.
- A frequent itemset is a set of items that appears at least in a pre-specified number of transactions. Frequent itemsets are typically used to generate association rules.
- Consider a data set S, frequent itemset in S are those items that appear in at least a fraction s of the basket, where s is a chosen constant with a value of 0.01 or 1%.
- To find frequent itemsets one can use the monotonicity principle or a-priori trick which is given as,

  If a set of items say S is frequent then all its subsets are also frequent.
- The procedure to find frequent itemsets :
  - A level wise search may be conducted to find the frequent-1 items(set of size 1), then proceed to find frequent -2 items and so on.
  - Next search for all maximal frequent itemsets.

Syllabus Topic : Closed Itemsets

4.3 Closed Itemsets

➔ (SPPU - May 16, Dec. 16, Aug. 17)

Q. Explain the following terms : Closed and maximal frequent itemsets.

May 16, Dec. 16, Aug. 17, 3 Marks

- An itemset is closed if none of its immediate supersets has the same support as the itemset.
- Consider two itemsets X and Y, if every item of X is in Y but there is at least one item of Y, which is not in X, then Y is not a proper super-itemset of X. In this case, itemset X is closed.
- If X is both closed and frequent, it is known as closed frequent itemset.
- An itemset is maximal frequent if none of its immediate supersets is frequent.
- An itemset X is *maximal frequent itemset or max-itemset* if X is frequent and there exist no super itemset Y such that X is subset of Y and Y is frequent.

Example

Let us consider minimum support = 2.

- The itemsets that are circled with thick lines are the frequent itemsets as they satisfy the minimum support. Fig. 4.3.1, Frequent itemsets are { p,q,r,s,pq,ps,qs,rs,pqs}

- The itemsets that are circled with the double lines are closed frequent itemsets. Fig. 4.3.1, closed frequent itemsets are {p,r,rs,pqs}. For example {rs} is closed frequent itemset as all of its superset {prs ,qrs} have support less than 2.
- The itemsets that are circled with the double lines and shaded are maximal frequent itemsets. Fig. 4.3.1, maximal frequent itemsets are {rs,pqs}. For example {rs} is maximal frequent itemset as none of its immediate supersets like {prs, qrs} is frequent.

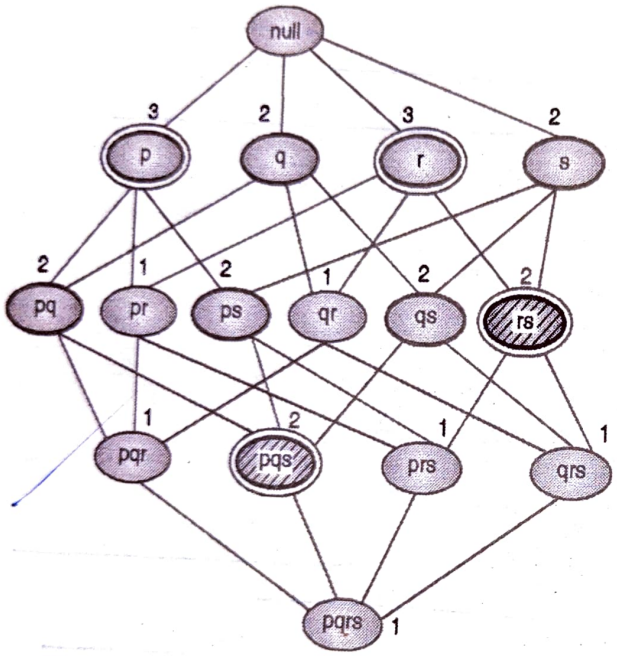


Fig. 4.3.1 : Lattice diagram for maximal, closed and frequent itemsets

Syllabus Topic : Association Rules

4.4 Association Rules

- The items or objects in Relational databases, transactional databases or other information repositories are considered for finding frequent patterns, associations, correlations, or causal structures.
- It searches for interesting relationships among items in a given data set by examining transactions, or shop carts, we can find which items are commonly purchased together. This knowledge can be used in advertising or in goods placement in stores.
- Association rules have the general form

$$I_1 \rightarrow I_2 \text{ (where } I_1 \cap I_2 = 0)$$

Where, I_n are sets of items, for example can be purchased in a store.

- The rule should be read as "Given that someone has bought the items in the set I_1 they are likely to also buy the items in the set I_2".

4.4.1 Finding the Large Itemsets

1. **The Brute Force approach**
   - Find all the possible association rules.
   - Calculate the support and confidence for each rule generated in the above step.
   - The Rules that fail the minsup and minconf are prunned from the above list.
   - The above steps would be a time consuming process, we can have a better approach as given below.
2. **A better approach** : The Apriori Algorithm.

4.4.2 Frequent Pattern Mining

Frequent pattern mining is classified in the various ways based on following criteria :

1. **Completeness of the pattern to be mined :** Here we can mine the complete set of frequent itemset, closed frequent itemset, constrained frequent itemsets.
2. **Levels of abstraction involved in the rule set :** Here we use multilevel association rules based on the levels of abstraction of data.
3. **Number of data dimensions involved in the rule :** Here we use single dimensional association rule, there is only one dimension or multidimensional association rule if there is more than one dimension.
4. **Types of the values handled in the rule :** Here we use Boolean and quantitative association rules.
5. **Kinds of the rules to be mined :** Here we use association rules and correlation rules based on the kinds of the rules to be mined.

6. **Kinds of pattern to be mined :** Here we use frequent itemset mining, sequential pattern mining and structured pattern mining.

4.4.3 Efficient and Scalable Frequent Itemset Mining Method

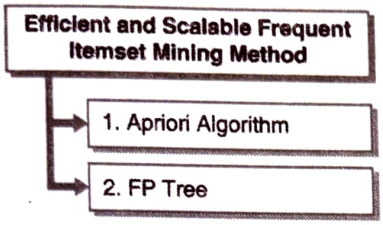


Fig. 4.4.1 : Efficient and Scalable Frequent Itemset Mining Method

Syllabus Topic : A-priori Algorithm

4.5 A-priori Algorithm

→ **(SPPU - Aug. 17)**

Q. Explain the Apriori algorithm for generation of association rules. How candidate keys are generated using apriori algorithm. **Aug. 17, 6 Marks**

Apriori Algorithm for Finding Frequent Itemsets using Candidate Generation

- The Apriori Algorithm solves the frequent item sets problem.
- The algorithm analyzes a data set to determine which combinations of items occur together frequently.
- The Apriori algorithm is at the core of various algorithms for data mining problems. The best known problem is finding the association rules that hold in a basket - item relation.

☞ **Basic idea**

- An itemset can only be a large itemset if all its subsets are large itemsets.
- Frequent itemsets : The sets of items that have minimum support.
- All the subsets of a frequent itemset must be frequent for e.g. {PQ} is a frequent itemset {P} and {Q} must also be frequent.
- Find frequent itemsets frequently with cardinality 1 to k(k-itemset).
- Generate association rules from frequent itemsets.

Q. Write a pseudo code for Apriori algorithm and explain. **Dec. 15, 6 Marks**

Q. Write Apriori Algorithm and explain it with suitable example. **Dec.17, 6 Marks**

☞ **Apriori Algorithm given by Jiawei Han et al.**

Input :

D: a database of transactions;

min_sup : the minimum support count threshold.

Output : L : frequent itemsets in D.

Method :

(1) L_1 = find_frequent_1-itemsets(D);
(2) for ($k = 2; L_{k-1} \neq \phi; k++$){
(3) C_k = apriori_gen (L_{k-1});
(4) for each transaction $t \in D$ { // scan D for counts
(5) C_t = subset (C_k, t);
// get the subsets of t that are candidates
(6) for each candidate $c \in C_t$
(7) c.count ++ ;
(8) }
(9) $L_k = \{c \in C_k \mid c.count \geq min\_sup\}$
(10) }
(11) return $L = \cup_k L_k$;

Procedure apriori_gen (L_{k-1}:frequent (k – 1) - itemsets)

(1) for each itemset $l_1 \in L_{k-1}$
(2) for each itemset $l_2 \in L_{k-1}$
(3) if ($l_1(1) = l_2(1) \wedge (l_1(2) = l_2(2)) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ then {
(4) $c = l_1 \bowtie l_2$; // join step:generate candidates
(5) if has_infrequent_subset(c, L_{k-1}) then
(6) delete c; // prune step: remove unfruitful candidate
(7) else add c to C_k
(8) }

```
(9)   return C_k;

Procedure has_infrequent_subset(c: candidate k-itemset;
L_k-1: frequent (k – 1) –itemsets); // use prior knowledge
(1)   for each (k – 1)-subset s of c
(2)      if s ∉ L_k-1 then
(3)         return TRUE;
(4)   return FALSE;
```

4.5.1 Advantages and Disadvantages of Apriori Algorithm

Some of the advantages and disadvantages of Apriori Algorithm are as follows :

Advantages

1. The algorithm makes use of large itemset property.
2. The method can be easily parallelized.
3. The algorithm is easy from implementation point of view.

Disadvantages

1. Although the algorithm is easy to implement it needs many database scans which reduces the overall performance.
2. Due to Database scans, the algorithm assumes transaction database is memory resident. - Generation of candidate pattern

Syllabus Topic : Generating Association Rules from Frequent Item Sets

4.6 Generating Association Rules from Frequent Item Sets

- Find the frequent itemsets from transaction database.
- Using confidence formula generate strong association rules which satisfy both minimum support and minimum confidence.

Support

- The support of an itemset is the count of that itemset in the total number of transactions, or in other words it is the percentage of the transactions in which the items appear.

If A => B

$$\text{Support } (A \Rightarrow B) = \frac{\#\_\text{tuples\_containing\_both\_A\_and\_B}}{\text{total\_\#\_of\_tuples}}$$

- The support(s) for an association rule X => Y is the percentage of transactions in the database that contains X as well as Y i. e.(X and Y together).
- An itemset is considered to be a *large itemset* if its support is above some threshold called minimum support.

Confidence

- The confidence or strength for an association rule A => B is the ratio of the number of transactions that contain A as well as B to the number of transactions that contain A.
- Consider a rule A => B, it is measure of ratio of the number of tuples containing both A and B to the number of tuples containing A

$$\text{Confidence } (A \Rightarrow B) = \frac{\#\_\text{tuples\_containing\_both\_A\_and\_B}}{\#\_\text{tuples\_containing\_A}}$$

Syllabus Topic : Improving the Efficiency of a-priori

4.7 Improving the Efficiency of a-priori

There are many variations of Apriori algorithm that have been proposed to improve the efficiency, few of them are given as :

- **Hash-based itemset counting :** The itemsets can be hashed into corresponding buckets. For a particular iteration a k-itemset can be generated and hashed into their respective bucket and increase the bucket count, the bucket with a count lesser than the support should not be considered as a candidate set.
- **Transaction reduction :** A transaction that does not contain k-frequent itemset will never have k + 1 frequent itemset, such a transaction should be reduced from future scans.
- **Partitioning :** In this technique only two database scans are needed to mine the frequent itemsets. The algorithm has two phases, in the first phase, the transaction database is divided into non overlapping partitions. The minimum support count of a partition is min support X number of transactions in that partition. Local frequent itemsets are found out in each partition.

- The local frequent itemsets may or may not be frequent with respect to the entire database however a frequent itemset from database has to be frequent in atleast one of the partitions.
- All the frequent itemsets with respect to each partition forms the global candidate itemsets. In the second phase of the algorithm, a second scan of database for actual support of each item is found, these are global frequent itemsets.
- **Sampling :** Rather than finding the frequent itemsets in the entire database D, a subset of transactions are picked up and searched for frequent itemsets. A lower threshold of minimum support is considered as this reduces the possibility of missing the actual frequent itemset due to a higher support count.
- **Dynamic itemset counting :** In this the database is partitioned into blocks and is marked by start points. It maintains a count-so-far, if this count-so-far crosses minimum support, the itemset is added to the frequent itemset collection which can be further used to generate longer candidate itemset.

4.8 Solved Example on Apriori Algorithm

Ex. 4.8.1 : Given the following data, apply the Apriori algorithm. Min support = 50 % Database D.

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

Soln. :

Step 1 : Scan D for count of each candidate. The candidate list is {1, 2, 3,4, 5} and find the support.

C_1 =

Itemset	Sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

Step 2 : Compare candidate support count with minimum support count (i.e. 50%)

L_1 =

Itemset	Sup.
{1}	2
{2}	3
{3}	3
{5}	3

Step 3 : Generate candidate C_2 from L_1

C_2 =

Itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Step 4 : Scan D for count of each candidate in C_2 and find the support

C_2 =

Itemset	Sup.
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

Step 5 : Compare candidate (C_2) support count with the minimum support count

L_2=

Itemset	Sup.
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

Step 6 : generate candidate C_3 from L_2

C_3 =

Itemset
{1,3,5}
{2,3,5}
{1,2,3}

Step 7 : Scan D for count of each candidate in C_3

C_3 =

Itemset	sup
{1,3,5}	1
{2,3,5}	2
{1,2,3}	1

Step 8 : Compare candidate (C_3) support count with the minimum support count

L_3 =

Itemset	sup
{2,3,5}	2

Step 9 : So data contain the frequent itemset(2,3,5)

Therefore the association rule that can be generated from L_3 are as shown below with the support and confidence.

Association Rule	Support	Confidence	Confidence %
2^3=>5	2	2/2=1	100%
3^5=>2	2	2/2=1	100%
2^5=>3	2	2/3=0.66	66%
2=>3^5	2	2/3=0.66	66%
3=>2^5	2	2/3=0.66	66%
5=>2^3	2	2/3=0.66	66%

If the minimum confidence threshold is 70% (Given), then only the first and second rules above are output, since these are the only ones generated that are strong.

Final rules are :

Rule 1: 2^3=>5 and Rule 2 : 3^5=>2

Ex. 4.8.2 : Find the frequent item sets in the following database of nine transactions, with a minimum support 50% and confidence 50%.

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Soln. :

Step 1 : Scan D for count of each candidate. The candidate list is {A,B,C,D,E,F} and find the support

C_1=

Items	Sup.
{A}	3
{B}	2
{C}	2
{D}	1
{E}	1
{F}	1

Step 2 : Compare candidate support count with minimum support count (50%)

L_1=

Items	Sup.
{A}	3
{B}	2
{C}	2

Step 3 : Generate candidate C_2 from L_1

C_2=

Items
{A,B}
{A,C}
{B,C}

Step 4 : Scan D for count of each candidate in C_2 and find the support

C_2=

Items	Sup.
{A,B}	1
{A,C}	2
{B,C}	1

Step 5 : Compare candidate (C_2) support count with the minimum support count

L_2 =

Items	Sup.
{A,C}	2

Step 6 : So data contain the frequent item l(A,C)

Therefore the association rule that can be generated from L are as shown below with the support and confidence

Association Rule	Support	Confidence	Confidence %
A -> C	2	2/3 = 0.66	66 %
C-> A	2	2/2 = 1	100 %

Minimum confidence threshold is 50% (Given), then both the rules are output as the confidence is above 50 %.

So final rules are :

Rule 1 : A -> C

Rule 2 : C-> A

Ex. 4.8.3 : Consider the transaction database given below. Use Apriori algorithm with minimum support count 2. Generate the association rules along with its 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

Soln. :

Step 1 : Scan the transaction Database D and find the count for item-1 set which is the candidate. The candidate list is {I1, I2, I3, I4, I5} and find each candidates support.

C_1=

1-Itemsets	Sup-count
I1	6
I2	7
I3	6
I4	2
I5	2

Step 2 : Find out whether each candidate item is present in at least two transactions (As support count given is 2).

L_1=

1-Itemsets	Sup-count
1	6
2	7
3	6
4	2
5	2

Step 3 : Generate candidate C_2 from L_1 and find the support of 2-itemsets.

C_2=

2-Itemsets	Sup-count
1,2	4
1,3	4
1,4	1
1,5	2
2,3	4
2,4	2
2,5	2
3,4	0
3,5	1
4,5	0

Step 4 : Compare candidate (C_2) generated in step 3 with the support count, and prune those itemsets which do not satisfy the minimum support count.

L_2 =

Frequent 2-Itemsets	Sup-count
1,2	4
1,3	4
1,5	2
2,3	4
2,4	2
2,5	2

Step 5 : Generate candidate C_3 from L_2.

C_3 =

Frequent 3-Itemset
1,2,3
1,2,5
1,2,4

Step 6 : Scan D for count of each candidate in C_3 and find their support count.

C_3 =

Frequent 3-Itemset	Sup-count
1,2,3	2
1,2,5	2
1,2,4	1

Step 7 : Compare candidate (C_3) support count with the minimum support count and prune those itemsets which do not satisfy the minimum support count.

L_3 =

Frequent 3-Itemset	Sup-count
1,2,3	2
1,2,5	2

Step 8 : Frequent itemsets are {I1,I2,I3} and {I1,I2,I5}

Let us consider the frequent itemsets ={I1, I2, I5}. Following are the Association rules that can be generated shown below with the support and confidence.

Association Rule	Support	Confidence	Confidence %
I1^I2=>I5	2	2/4	50%
I1^I5=>I2	2	2/2	100%
I2^I5=>I1	2	2/2	100%
I1=>I2^I5	2	2/6	33%
I2=>I1^I5	2	2/7	29%
I5=>I1^I2	2	2/2	100%

Suppose if the minimum confidence threshold is 75% then only the following rules will be considered as output, as they are strong rules.

Rules	Confidence
I1^I5=>I2	100%
I2^I5=>I1	100%
I5=>I1^I2	100%

Ex. 4.8.4 : Consider the following transactions :

TID	Items
01	1, 3, 4, 6
02	2,3,5,7
03	1,2,3,5,8
04	2,5,9,10
05	1,4

Apply the Apriori with minimum support of 30% and minimum confidence of 75% and find large item set L.

Soln. :

Step 1 : Scan the transaction Database D and find the count for item-1 set which is the candidate. The candidate list is {1,2,3,4,5,6,7,8,9,10} and find the support.

C_1 =

Itemset	Sup-count
1	3
2	3
3	3
4	2
5	3
6	1
7	1
8	1
9	1
10	1

Step 2 : Find out whether each candidate item is present in at least 30% of transactions (As support count given is 30%).

L_1 =

Itemset	Sup-count
1	3
2	3
3	3
4	2
5	3

Step 3 : Generate candidate C_2 from L_1 and find the support of 2-itemsets.

C_2 =

Itemset	Sup-count
1,2	1
1,3	2
1,4	2
1,5	1
2,3	2
2,4	0
2,5	3
3,4	1
3,5	2

Step 4 : Compare candidate (C_2) generated in step 3 with the support count, and prune those itemsets which do not satisfy the minimum support count.

L_2 =

Itemset	Sup-count
1,3	2
1,4	2
2,3	2
2,5	3

Step 5 : Generate candidate C_3 from L_2 and find the support.

C_3 =

Itemset	Sup-count
1,2,3	1
2,3,5	2
1,3,4	1

Step 6 : Compare candidate (C_3) support count with min support.

L_3 =

Itemset	Sup-count
2,3,5	2

Therefore the database contains the frequent itemset{2,3,5}.

Following are the association rules that can be generated from L3 are as shown below with the support and confidence.

Association Rule	Support	Confidence	Confidence %
2^3=>5	2	2/2=1	100%
3^5=>2	2	2/2=1	100%
2^5=>3	2	2/3=0.66	66%
2=>3^5	2	2/3=0.66	66%
3=>2^5	2	2/3=0.66	66%
5=>2^3	2	2/3=0.66	66%

Given minimum confidence threshold is 75%, so only the first and second rules above are output, since these are the only ones generated that are strong.

Final Rules are :

Rule 1: 2^3=>5 and Rule 2 : 3^5=>2

Ex. 4.8.5 : A database has four transactions. Let min sup=60% and min conf= 80%

TID	Date	Items-bought
T100	10/15/99	{K, A, D, B}
T200	10/15/99	{D, A, C, E, B}
T300	10/19/99	{C,A,B,E}
T400	10/22/99	{B, A, D}

Find all frequent itemsets using apriori algorithm

List strong association rules(with supports S and confidence C).

Soln. :

Step 1 : Scan D for count of each candidate. The candidate list is {A,B,C,D,E,K} and find the support.

C_1 =

Itemset	Sup-count
A	4
B	4
C	2
D	3
E	2
K	1

Step 2 : Compare candidate support count with minimum support count (i.e. 60%).

L_1 =

Itemset	Sup-count
A	4
B	4
D	3

Step 3 : Generate candidate C_2 from L_1.

C_2 =

Itemset
A,B
A,D
B,D

Step 4 : Scan D for count of each candidate in C_2 and find the support.

C_2 =

Itemset	Sup-count
A,B	4
A,D	3
B,D	3

Step 5 : Compare candidate (C_2) support count with the minimum support count.

L_2 =

Itemset	Sup-count
A,B	4
A,D	3
B,D	3

Step 6 : Generate candidate C_3 from L_2.

C_3 =

Itemset
A,B,D

Step 7 : Scan D for count of each candidate in C_3.

C_3 =

Itemset	Sup
A,B,D	3

Step 8 : Compare candidate (C_3) support count with the minimum support count.

L_3 =

Itemset	Sup
A,B,D	3

Step 9 : So data contain the frequent itemset(A,B,D).

Therefore the association rule that can be generated from frequent itemsets are as shown below with the support and confidence.

Association Rule	Support	Confidence	Confidence %
A^B=>D	3	3/4=0.75	75%
A^D=>B	3	3/3=1	100%
B^D=>A	3	3/3=1	100%
A=>B^D	3	3/4=0.75	75%
B=>A^D	3	3/4=0.75	75%
D=>A^B	3	3/3=1	100%

If the minimum confidence threshold is 80% (Given), then only the SECOND, THIRD AND LAST rules above are output, since these are the only ones generated that are strong.

Ex. 4.8.6 : Apply the Apriori algorithm on the following data with Minimum support = 2

TID	List of item_IDs
T100	I1,I2,I4
T200	I1,I2,I5
T300	I1,I3,I5
T400	I2,I4
T500	I2,I3
T600	I1,I2,I3,I5
T700	I1,I3
T800	I1,I2,I3
T900	I2,I3
T1000	I3,I5

Soln. :

Step 1 : Scan Dfor count of each candidate. The candidate list is {I1, I2, I3, I4, I5} and find the support.

C_1 =

I-Itemsets	Sup-count
I1	6
I2	7
I3	7
I4	2
I5	4

Step 2 : Compare candidate support count with minimum support count (i.e. 2).

L_1 =

I-Itemsets	Sup-count
1	6
2	7
3	6
4	2
5	2

Step 3 : Generate candidate C_2 from L_1 and find the support.

C_2 =

2-Itemsets	Sup-count
1,2	4
1,3	4
1,4	1
1,5	3
2,3	4
2,4	2
2,5	2
3,4	0
3,5	3
4,5	0

Step 4 : Compare candidate (C_2) support count with the minimum support count.

L_2 =

2-Itemsets	Sup-count
1,2	4
1,3	4
1,5	3
2,3	4
2,4	2
2,5	2
3,5	3

Step 5 : Generate candidate C_3 from L_2.

C_3 =

Frequent 3-Itemset
1,2,3
1,2,5
1,2,4
1,3,5
2,3,5

Step 6 : Scan D for count of each candidate in C_3.

C_3 =

Frequent 3-Itemset	Sup-count
1,2,3	2
1,2,5	2
1,2,4	0
1,3,5	2
2,3,5	0

Step 7 : Compare candidate (C_3) support count with the minimum support count.

L_3 =

Frequent 3-Itemset	Sup-count
1,2,3	2
1,2,5	2
1,3,5	2

Step 8 : So data contain the frequent itemsets are {I1,I2,I3} and {I1,I2,I5} and {I1,I3,I5}.

Let us assume that the datacontains the frequent itemset={I1,I2,I5} then the association rules that can be generated from frequent itemset are as shown below with the support and confidence.

Association Rule	Support	Confidence	Confidence %
I1^I2=>I5	2	2/4	50%
I1^I5=>I2	2	2/2	100%
I2^I5=>I1	2	2/2	100%
I1=>I2^I5	2	2/6	33%
I2=>I1^I5	2	2/7	29%
I5=>I1^I2	2	2/2	100%

If the minimum confidence threshold is 70% (Given), then only the SECOND,THIRD AND LAST rules above are output, since these are the only ones generated that are strong.

Similarly do for frequent itemset {I1,I2,I3} and {I1,I3,I5}.

Ex. 4.8.7 : A Database has four transactions. Let Minimum support and confidence be 50%.

T_{id}	Items
100	1, 3, 4
200	2, 3, 5
300	1, 2, 3, 5
400	2, 5

T_{id}	Items
500	1,2,3
600	3,5
700	1,2,3,5
800	1,5
900	1,3

Soln. :

Step 1 : Scan D for count of each candidate. The candidate list is {1,2,3,4,5}and find the support.

C_1 =

Itemset	Sup-count
1	6
2	5
3	7
4	1
5	6

Step 2 : Compare candidate support count with minimum support count (i.e. 50%).

L_1 =

Itemset	Sup-count
1	6
2	5
3	7
5	6

Step 3 : Generate candidate C_2 from $L_1$and find the support.

C_2 =

Itemset	Sup-count
1,2	3
1,3	5
1,5	3
2,3	4
2,5	4
3,5	4

Step 4 : Compare candidate (C_2) support count with the minimum support count.

L_2 =

Itemset	Sup-count
1,3	5

So data contain the frequent itemset= {1,5}.

Therefore the association rule that can be generated from L_2 are as shown below with the support and confidence.

Association Rule	Support	Confidence	Confidence %
1=>3	5	5/6=0.83	83%
3=>1	5	5/7=0.71	71%

Given minimum confidence threshold is 50% , so both the rules are strong.

Final rules are :

Rule 1: 1=>3 and Rule 2 : 3=>1

Ex. 4.8.8 : Consider the five transactions given below. If minimum support is 30% and minimum confidence is 80%, determine the frequent itemsets and association rules using the a priori algorithm.

Transaction	Items
T1	Bread, Jelly, Butter
T2	Bread, Butter
T3	Bread, Milk, Butter
T4	Coke, Bread
T5	Coke, Milk

Soln. :

Step 1 : Scan D for Count of each candidate.

The candidate list is {Bread, Jelly, Butter, Milk, Coke}

C_1 =

I-Itemlist	Sup-Count
Bread	4
Jelly	1
Butter	3
Milk	F2
Coke	2

Step 2 : Compare candidate support count with minimum support count (i.e. 2)

I-Itemlist	Sup-Count
Bread	4
Butter	3
Milk	2
Coke	2

Step 3 : Generate C2 from L1 and find the support

C_2 =

I-Itemlist	Sup Count
{Bread, Butter}	3
{Bread, Milk}	1
{Bread, Coke}	1
{Butter, Milk}	1
{Butter, Coke}	0
{Milk, Coke}	1

Step 4 : Compare candidate (C2) support count with the minimum support count

L_2 =

Frequent 2 - Itemset	Sup – Count
{Bread, Butter}	3

Step 5 : So data contain the frequent itemset is {Bread, Butter}

Association Rule	Support	Confidence	Confidence %
Bread⟶Butter	3	3/4	75%
Butter⟶Bread	1	3/3	100%

Minimum confidence threshold is 80% (Given)

Final rule is

Butter→Bread

Ex. 4.8.9 : Consider the following transaction database.

TID	Items
01	A, B, C, D
02	A, B, C, D, E, G
03	A, C, G, H, K
04	B, C, D, E, K
05	D, E, F, H, L
06	A, B, C, D, L
07	B, I, E, K, L
08	A, B, D, E, K
09	A, E, F, H, L
10	B, C, D, F

Apply the Apriori algorithm with minimum support of 30% and minimum confidence of 70%, and find all the association rules in the data set.

Soln. :

Step 1 : Generate single item set :

Items	Support
A	6
B	7
C	6
D	7
E	6
F	3
G	2
H	3
I	1
K	4
L	4

Item set above 30 % support	
A	6
B	7
C	6
D	7
E	6
F	3
H	3
K	4
L	4

Step 2 : Generate 2 item set :

Item	Support
AB	4
AC	4
AD	4
AE	3
AF	1
AH	2
AK	2
AL	2
BC	5
BD	6
BE	4
BF	1
BH	0
BK	3
BL	2
CD	5
CE	2
CF	1

Item	Support
CH	1
CK	2
CL	1
DE	4
DF	2
DH	1
DK	2
DL	2
EF	2
EH	2
EK	3
EL	3
FH	2
FK	0
FL	2
HK	1
HL	2
KL	1

Item set above 30 % support	
AB	4
AC	4
AD	4
AE	3
BC	5
BD	6
BE	4
BK	3
CD	5
DE	4
EK	3
EL	3

Step 3 : Generate 3 item set :

Item sets of 3 items

Item set	Support
ABC	3
ABD	4
ABE	2
ABK	1
ACD	3
ACE	1
ADE	2
AEK	1
AEL	1
BCD	5
BCE	2
BCK	1
BDE	3
BDK	2
BEK	2
BEL	1
CDE	2
DEK	2
DEL	1

Item set above 30 % support

Item set	Support
ABC	3
ABD	4
ACD	3
BCD	5
BDE	3

Step 4 : Generate 4 item set

Item set	Support
ABCD	3
ABDE	2
BCDE	2

Therefore ABCD is the large item set with minimum support 30%.

Following Rules generated

Rule	Confidence	Confidence %
A → BCD	3/6 = 0.5	50%
B → ACD	3/7 = 0.43	43%
C → ABD	3/6 = 0.5	50%
D → ABC	3/7 = 0.43	43%
AB → CD	3/4 = 0.75	75%
BC → AD	3/5 = 0.6	60%
CD → AB	3/5 = 0.6	60%
AC → BD	3/4 = 0.75	75%
AD → BC	3/4 = 0.75	75%
BCD → A	3/5 = 0.6	60%
ACD → B	3/3 = 1	100%
ABD → C	3/4 = 0.75	75%
ABC → D	3/3 = 1	100%

From the above Rules generated, only the rules having greater than 70% are considered as final rules. So final Rules are,

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

Ex. 4.8.10 : Consider the following :

Transaction	Items
t_1	Bread, Jelly, Peanut Butter
t_2	Bread, Peanut Butter
t_3	Bread, Milk, Peanut Butter
t_4	Beer, Bread
t_5	Beer, Milk

Calculate the support and confidence for the following association rules :

i) Bread → Peanut Butter

ii) Jelly → Milk,

iii) Beer → Bread.

SPPU - Dec. 15, 6 Marks

Soln. :

Consider Minimum support count = 2 and Minimum confidence = 80%

Step 1 : Scan D for Count of each candidate.

The candidate list is {Bread, Jelly, Peanut Butter, Milk, Beer}

C1 =

I-Itemlist	Sup-Count
Bread	4
Jelly	1
Peanut Butter	3
Milk	2
Beer	2

Step 2 : Compare candidate support count with minimum support count (i.e. 2)

I-Itemlist	Sup-Count
Bread	4
Peanut Butter	3
Milk	2
Beer	2

Step 3 : Generate C2 from L1 and find the support

C2 =

I-Itemlist	Sup Count
{Bread, Peanut Butter}	3
{Bread, Milk}	1
{Bread, Beer}	1
{Peanut Butter, Milk}	1
{Peanut Butter, Beer}	0
{Milk, Beer}	1

Step 4 : Compare candidate (C2) support count with the minimum support count

L2 =

Frequent 2 - Itemset	Sup – Count
{Bread, Peanut Butter}	3

Step 5 : So data contain the frequent itemset is {Bread, Peanut Butter}

Association Rule	Support	Confidence	Confidence %
Bread ⟶ Peanut Butter	3	3/4	75%
Bread ⟶ Peanut Butter	1	3/3	100%

Minimum confidence threshold is 80%

Final rule is : Peanut Butter ⟶Bread

Similarly Confidence and support for the following association rules are :

Association Rule	Support	Confidence	Confidence %
Bread →Peanut Butter	3	3/4	75%
Jelly → Milk	0	0/1	0 %
Beer → Bread	1	1/2	50%

Ex. 4.8.11 : Consider the market basket transactions shown below :

Transaction ID	Items-bought
T1	{Mango, Apple, Banana, Dates}
T2	{Apple, Dates, Coconut, Banana, Fig}
T3	{Apple, Coconut, Banana, Fig}
T4	{Apple, Banana, Dates}

Assuming the minimum support of 50% and minimum confidence of 80%

(i) Find all frequent itemsets using Apriori algorithm.

(ii) Find all association rules using Apriori algorithm. **SPPU - May 16, 6 Marks**

Soln. :

(i) Let us assume, Mango = M, Apple = A, Banana = B, Dates = D, Coconut = C and Fig = F

Step 1 : Scan D for count of each candidate. The candidate list is {A,B,C,D,F,M} and find the support.

C_1 =

Itemset	Sup-count
A	4
B	4
C	2
D	3
F	2
M	1

Step 2 : Compare candidate support count with minimum support count (i.e. 50%).

L_1 =

Itemset	Sup-count
A	4
B	4
C	2
D	3
F	2

Step 3 : Generate candidate C_2 from L_1.

C_2 =

Itemset
A,B
A,C
A,D
A,F
B,C
B,D
B,F
C,D
C,F
D,F

Step 4 : Scan D for count of each candidate in C_2 and find the frequent itemset.

L_2 =

Itemset	Sup-count
A,B	4
A,C	2
A,D	3
A,F	2
B,C	2
B,D	2
B,F	2
C,F	2

Step 5 : Generate candidate C_3 from L_2.

C_3 =

Itemset
A,B,C
A,B,D
A,B,F
A,D,F
B,C,D
B,C,F
B,D,F
A,C,F

Step 6 : Compare candidate (C_3) support count with the minimum support count.

L_3 =

Itemset	Sup-count
A,B,C	2
A,B,D	3
A,B,F	2
B,C,F	2
A,C,F	2

Step 7 : Generate candidate C_4 from L_3.

C_4 =

Itemset
A,B,C,D
A,B,C,F

Step 8 : Compare candidate (C_4) support count with the minimum support count.

L_4 =

Itemset	Sup-count
A,B,C,D	1
A,B,C,F	2

Step 9 : So data contain the frequent itemset (A,B,C,F).

(ii) Therefore the association rule that can be generated from frequent itemsets are as shown below with the support and confidence.

Association Rule	Support	Confidence	Confidence %
A,B,C → F	2	2/2	100
A,C,F → B	2	2/2	100
B,C,F→ A	2	2/2	100
A,B,F → C	2	2/2	100

Ex. 4.8.12 : A database has five transactions. Let minimum support is 60%.

TID	Items
1	Butter, Milk
2	Butter, Dates, Balloon, Eggs
3	Milk, Dates, Balloon, Cake
4	Butter, Milk, Dates, Balloon
5	Butter, Milk, Dates, Cake

Find all the frequent item sets using Apriori algorithm. Show each step.

SPPU - Oct. 16, 6 Marks

Soln. :

Step 1 : Scan database for count of each candidate. The candidate list is {Butter, milk, Dates, Balloon, Eggs, cake} and find the support

c_1 =

Itemset	Support
{ Butter }	4
{ Milk }	4
{ Dates }	4
{ Balloon }	3
{ Eggs }	1
{ Cake }	2

Step 2 : Compare candidate support count with minimum support (i.e. 60%)

L_1 =

Itemset	Support
{ Butter }	4
{ Milk }	4
{ Dates }	4
{ Balloon }	3

Step 3 : Generate candidate C_2 from L_1

C_2 =

Itemset
{ Butter, Milk }
{ Butter, Dates}
{Butter, Balloon }
{Milk, Dates}
{ Milk, Balloon }
{Dates, Balloon }

Step 4 : Scan D for count of each candidate to find the support C_2.

Itemset	Support
{ Butter, Milk }	3
{ Butter, Dates}	3
{Butter, Balloon }	2
{Milk, Dates}	3
{ Milk, Balloon }	2
{Dates, Balloon }	3

Step 5 : Compare candidate C_2 support count with minimum support count

Itemset	Support
{ Butter, Milk }	3
{ Butter, Dates}	3
{Milk, Dates}	3
{ Dates, Balloon }	3

Step 6 : Generate candidate C_3 from L_2

{Balloon, Milk, Dates }

Associative rules from this –

Itemset	Confidence %
{Milk, Dates } → {Balloon }	0.67
{ Milk, Balloon} → {Dates }	1.00
{Dates, Balloon} → {Milk }	0.67
{ Balloon } → { Milk, Dates }	0.67
{ Dates } → { Milk, Balloon }	0.5
{ Milk} → { Dates, Balloon }	0.5

Ex. 4.8.13 : Consider the market basket transaction shown below :

Transaction ID	Items bought
T1	{M, A, B, D}
T2	{A, D, C, B, F}
T3	{A, C, B, F}
T4	{A, B, D}

Assuming the minimum support of 50% and minimum confidence of 80%

Find all frequent items using Apriori algorithm.

Find all association rules using Apriori algorithm.

Soln. :

Step 1 : Scan D for count of each candidate. The candidate list is {A,B,C,D,F,M} and find the support.

C_1 =

Itemset	Sup-count
A	4
B	4
C	2
D	3
F	2
M	1

Step 2 : Compare candidate support count with minimum support count (i.e. 50%).

L_1 =

Itemset	Sup-count
A	4
B	4
C	2
D	3
F	2

Step 3 : Generate candidate C_2 from L_1

C_2 =

Itemset
A,B
A,C
A,D
A,F
B,C
B,D
B,F
C,D
C,F
D,F

Step 4 : Scan D for count of each candidate in C_2 and find the frequent itemset.

L_2 =

Itemset	Sup-count
A,B	4
A,C	2
A,D	3
A,F	2
B,C	2
B,D	2
B,F	2
C,F	2

Step 5 : Generate candidate C_3 from L_2.

C_3 =

Itemset
A,B,C
A,B,D
A,B,F
A,D,F
B,C,D
B,C,F
B,D,F
A,C,F

Step 6 : Compare candidate (C_3) support count with the minimum support count.

L_3 =

Itemset	Sup-count
A,B,C	2
A,B,D	3
A,B,F	2
B,C,F	2
A,C,F	2

Step 7 : Generate candidate C_4 from L_3.

C_4 =

Itemset
A,B,C,D
A,B,C,F

Step 8 : Compare candidate (C_4) support count with the minimum support count.

L_4 =

Itemset	Sup-count
A,B,C,D	1
A,B,C,F	2

Step 9 : So data contain the frequent itemset(A,B,C,F).

Therefore the association rule that can be generated from frequent itemsets are as shown below with the support and confidence.

Association Rule	Support	Confidence	Confidence %
A,B,C → F	2	2/2	100
A,C,F → B	2	2/2	100
B,C,F→ A	2	2/2	100
A,B,F → C	2	2/2	100

Syllabus Topic : Mining Frequent Item sets without Candidate Generation : FP Growth Algorithm

4.9 Mining Frequent Item sets without Candidate Generation : FP Growth Algorithm

☞ **Definition of FP-tree**

An FP-tree is a tree structure which consists of :

- One root labeled as "null".
- A set of item prefix sub-trees with each node formed by three fields : item-name, count, node-link.
- A frequent-item header table with two fields for each entry : item-name, head of node-link.
- It contains the complete information for frequent pattern mining.
- The size of the FP-tree is bounded by the size of the database, but due to frequent items sharing, the size of the tree is usually much smaller than its original database.
- High compaction is achieved by placing more frequently items closer to the root (being thus more likely to be shared).
- The FP-Tree contains everything from the database we need to know for mining frequent.

☞ **Patterns**

- The size of the FP-tree is ≤ the candidate sets generated in the association rule mining.
- This approach is very efficient due to :
  - Compression of a large database into a smaller data structure.
  - It is a frequent pattern growth mining method or simply FP-growth.
  - It adopts a divide-and-conquer strategy.
- The database of frequent items is compressed into a FP-Tree, and the association information of items is preserved.
- Then mine each such database separately.

4.9.1 FP-Tree Algorithm

☞ **FP-tree construction algorithm given by Jiawei Han et al.**

- **FP-Growth :** Allows frequent itemset discovery without candidate itemset generation.
- Once the FP tree is generated, it is mined by calling FP_growth(FP_tree,null).
- **Algorithm :** FP growth, Mine frequent itemsets using an FP-tree by pattern fragment growth.

Input

- D, a transaction database.
- min_sup, the minimum support count threshold.

Output : The complete set of frequent patterns.

Method

1. **A FP tree is constructed in the following steps**

(a) Scan the transaction database D once, Collect F, the set of frequent items, and their support counts. Sort F by support count in descending order as L, the list of frequent items.

(b) Create the root of an FP tree, and label it as "null". For each transaction Trans D do the following.

Select and sort the frequent items in Trans according to the order of L. Let the sorted frequent item list in Trans be [p | P], where p is the first element and P is the remaining list. Call insert_tree ([p | P]·T), which is performed as follows. If T has a child N such that N_item·name = p·item·name, then increment N's count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link to the nodes with the same item·name via the node·link structure. If P is nonempty, call insert_tree (P, N) recursively.

2. **The FP-tree is mined by calling FP growth. FP tree, null/, which is implemented as follows :**

Procedure FP_growth (Tree, α)

```
(1) if Tree contains a single path P then
(2) for each combination (denoted as β) of the nodes in the path P
(3) generate pattern β∪α with support_count = minimum support
    count of nodes in β;
(4) else for each a_i in the header of Tree {
(5) generate pattern β = a_i ∪ αwith support_count
    = a_i.support_count;
(6) construct β's conditional pattern base and then
    β's conditional FP_tree Treeβ;
(7) if Treeβ≠ϕ then
(8) call FP_growth (Treeβ, β) ; }
```

Analysis

- Two scans of the DB are necessary. The first collects the set of frequent items and the second constructs the FP-tree.
- The cost of inserting a transaction Trans into the FP-tree is O(|Trans|), where |Trans| is the number of frequent items in Trans.

4.9.2 FP-Tree Size

- Many transactions share items due to which the size of the FP-Tree can have a smaller size compared to uncompressed data.
- **Best case scenario :** All transactions have the same set of items which results in a single path in the FP Tree.
- **Worst case scenario :** Every transaction has a distinct set of items, i.e. no common items
  - FP-tree size is as large as the original data.
  - FP-Tree storage is also higher, it needs to store the pointers between the nodes and the counter.
- FP-Tree size is dependent on the order of the items. Ordering of items by decreasing support will not always result in a smaller FP -Tree size (it's heuristic).

4.9.3 Example of FP Tree

Ex. 4.9.1 : Transactions consist of a set of items I = {a, b, c, ...} , min support = 3

TID	Items Bought
1	f, a, c, d, g, i, m, p
2	a, b, c, f, l, m, o
3	b, f, h, j, o
4	b, c, k, s, p
5	a, f, c, e, l, p, m, n

SPPU - Oct. 16, 5 Marks

Soln. :

Step 1 : Find the minimum support of each item.

Item	Sup.
a	3
b	3
c	4
d	1
e	1
f	4
g	1
h	1
i	1
j	1
k	1
l	2

Item	Sup.
m	3
n	1
o	2
p	3

Consider items with min support = 3 (given)

Item	Sup.
a	3
b	3
c	4
f	4
m	3
p	3

Step 2 : Order all items in itemset in frequency descending order (min support = 3)

(**Note :** Consider only items with min support = 3)

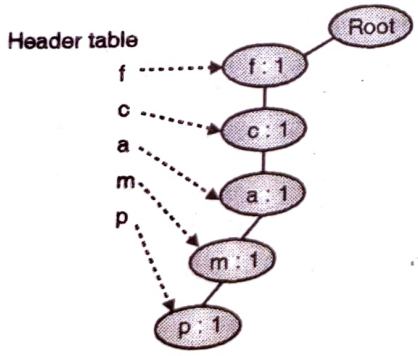
TID	Items Bought	(Ordered frequent items)
1	f, a, c, d, g, i, m, p	f, c, a, m, p
2	a, b, c, f, l, m, o	f, c, a, b, m
3	b, f, h, j, o	f, b
4	b, c, k, s, p	c, b, p
5	a, f, c, e, l, p, m, n	f, c, a, m, p

(f:4, c:4, a:3, b:3, m:3, p:3)

Step 3 : FP Tree construction

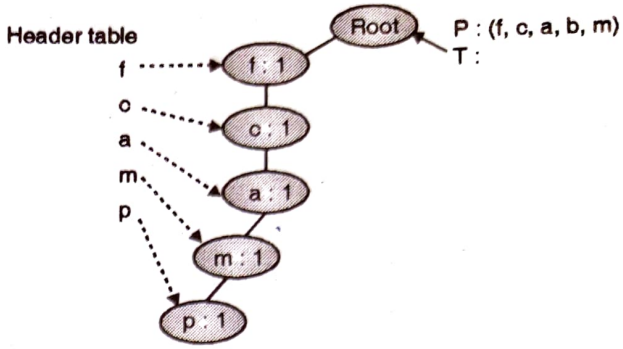
Originally Empty 

Step 4 : Insert the first Transaction (f, c, a, m, p)

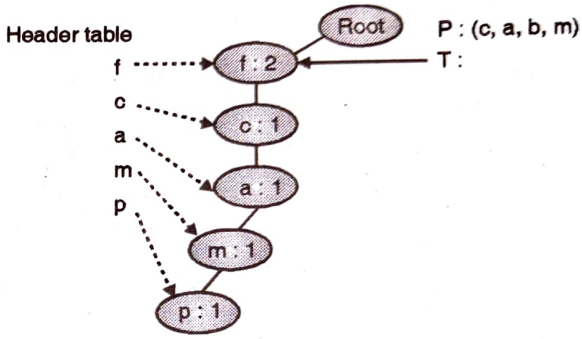


Step 5 : Start the insertion of Second transaction (f,c,a,b,m)

(i) The transaction T is pointing to the root node,

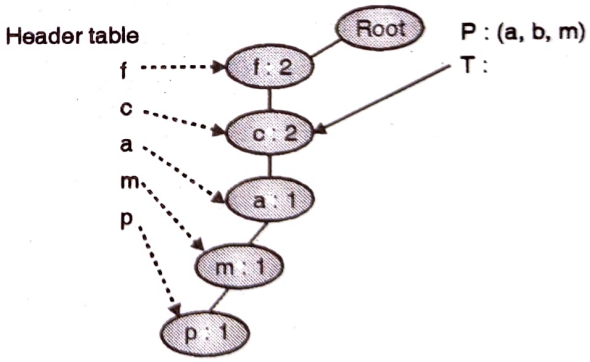


(ii) Consider the first item in the second transaction i.e. f and add it in the tree.

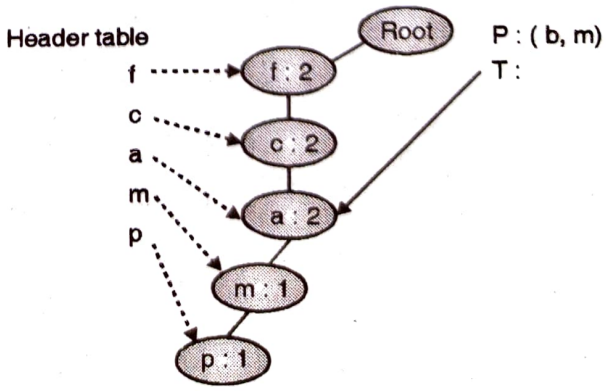


After this step we get f:2, finished adding f in the above tree.

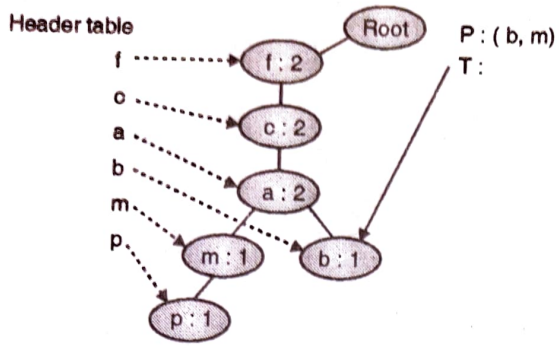
(iii) Now consider the second item in the above transaction .i.e. c.



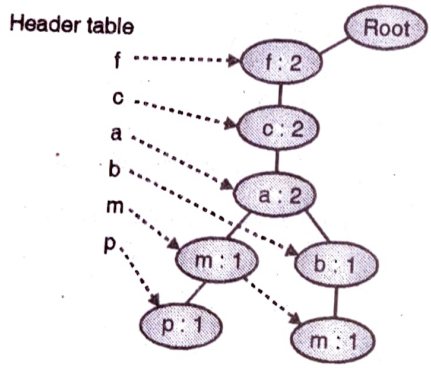
(iv) Similarly consider the next item a.



(v) Since we do not have a node b, we create one node for b below the node a (note : to maintain the path).

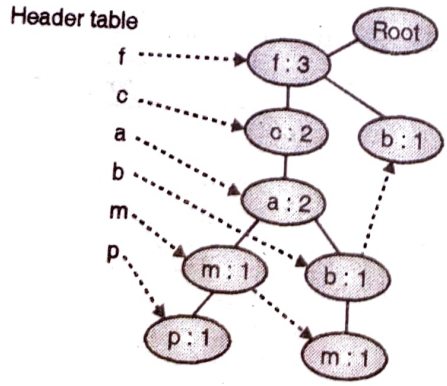


(vi) Now only m of second transaction is left. Though a node m is already exists still we can't increase its count of the existing node m as we need to represent the second transaction in FP tree, so add new node m below node b and link it with existing node m.

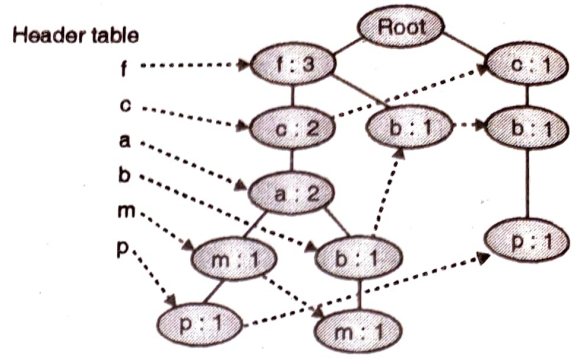


Second transaction is complete.

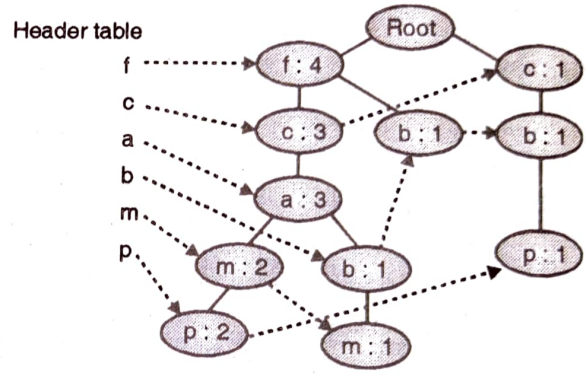
Step 6 : Similarly insert the third transaction(f,b) as explained in step 5. So After the insertion of third transaction (f,b)



Step 7 : After the insertion of fourth transaction(c, b, p)



Step 8 : After the insertion of fifth Transaction (f,c, a, m, p)



This is the final FP-Tree.

4.9.4 Mining Frequent Patterns from FP Tree

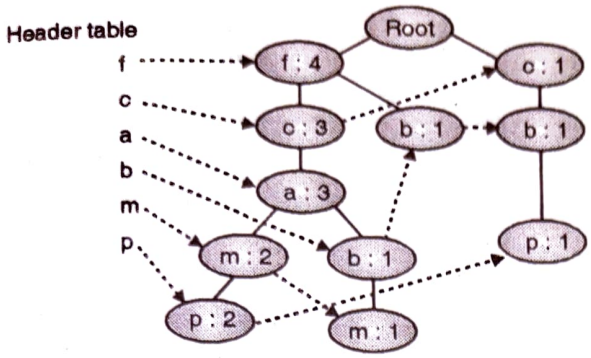
General idea (divide-and-conquer)

- Use the FP Tree and recursively grow frequent pattern path.

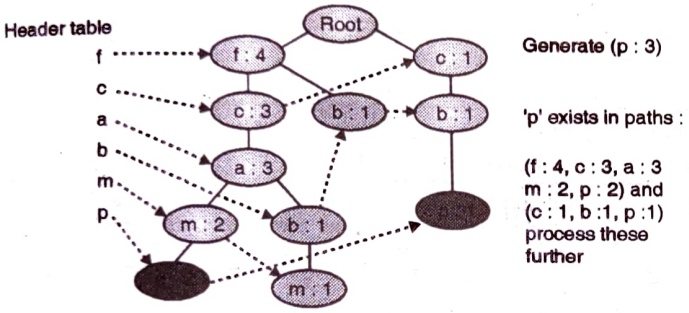
Method

- For each item, conditional pattern-base is constructed, and then it's conditional FP-tree.
- On each newly created conditional FP-tree, repeat the process.
- The process is repeated until the resulting FP-tree is empty, or it has only a single path(All the combinations of sub paths will be generated through that single path, each of which is a frequent pattern).

Example : Finding all the patterns with 'p' in the FP tree given below :



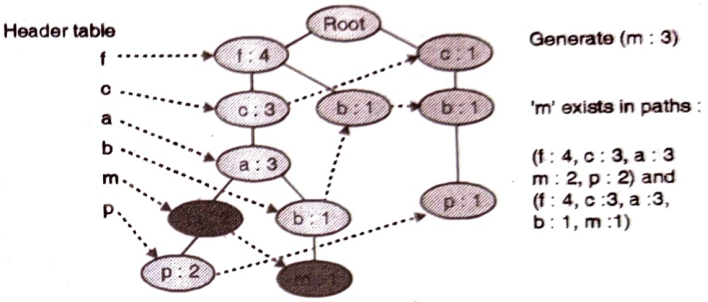
- Starting from the bottom of the header table.



- Following are the paths with 'P'
  - We got (f:4, c:3, a:3, m:2, p:2) and (c:1, b:1, p:1)
  - The transactions containing 'p' have p.count
  - Therefore we have (f:2, c:2, a:2, m:2, p:2) and (c:1, b:1, p:1)
  - Since 'p' is part of these we can remove'p'
- Conditional Pattern Base (CPB)
  - After removing P we get : (f:2, c:2, a:2, m:2) and (c:1, b:1)
  - Find all frequent patterns in the CPB and add 'p' to them, this will give us all frequent patterns containing 'p'.
  - This can be done by constructing a new FP-Tree for the CPB.
- Finding all patterns with 'P'.
  - We again filter away all items < minimum support threshold (i.e. 3)
  - (f:2, c:2, a:2, m:2), (c:1, b:1) => (c:3)
  - We generate (cp:3) (Note : we are finding frequent patterns containing item p, so we append p to c as c is only item that has min support threshold.)
  - Support value is taken from the sub-tree
  - Frequent patterns thus far: (p:3, cp:3)

Example : Finding Patterns with 'm' but not 'p'.

- Find 'm' from the header table



- Conditional Pattern Base :
  - Path 1 : (f:4, c:3, a:3, m:2, p:2) → (f:2, c:2, a:2)
  - In the above transaction we need to consider m:2, based on this we get f:2 and so on. Exclude p as we don't want p i.e. given in example.
  - Path 2 : (f:4, c:3, a:3, b:1, m:1) → (f:1, c:1, a:1, b:1)
- Build FP tree using (f:2, c:2, a:2) and (f:1, c:1, a:1, b:1)
- Now we got (f:3, c:3, a:3 ,b:1)
- Initial Filtering removes b:1 (We again filter away all items < minimum support threshold).
- Mining Frequent Patterns by Creating Conditional Pattern-Bases.

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

Ex. 4.9.2 : Transaction item list is given below. Draw FP tree.

T1 = b, e

T2 = a, b, c, e

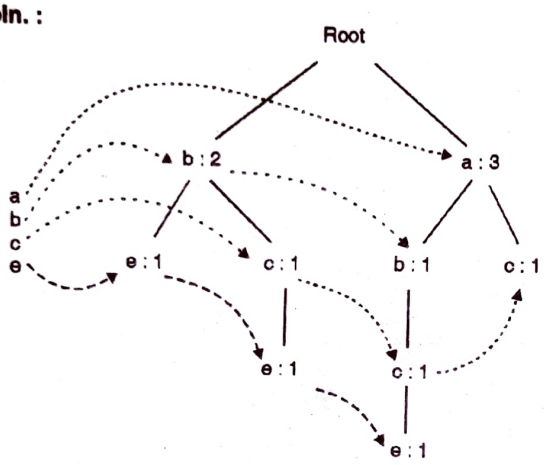
T3 = b, c, e

T4 = a, c

T5 = a

Given : minimum support = 2

Soln. :

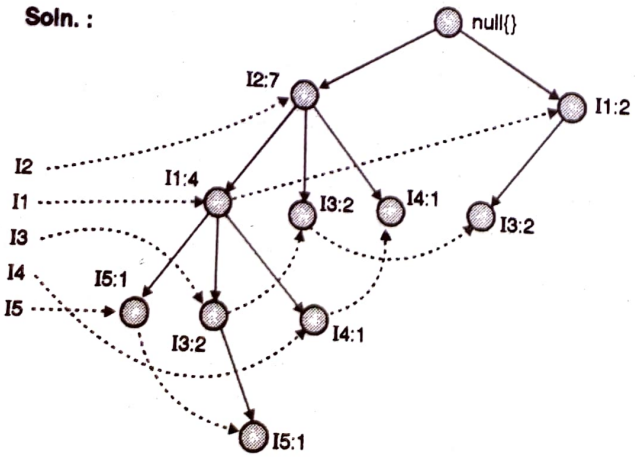


Ex. 4.9.3 : Transaction database is

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

Min support = 2

Soln. :



Item ID	Support Count
I2	7
I1	6
I3	6
I4	2
I5	2

Mining the FP-Tree by creating conditional (sub) pattern bases.

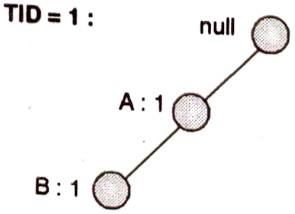
Item	Conditional pattern base	Conditional FP-tree	Frequent patterns generated
15	{(I2 I1 : 1), (I2 I1 I3 : 1) }	{I2 : 2, I1 : 2}	I2 I5 : 2, I1 I5 : 2, I2 I1 I5 : 2
14	{(I2 I1 : 1), (I2 : 1)}	{I2 : 2}	I2 I4 : 2
13	{(I2 I1 : 2), (I2 : 2), (I1 : 2) }	(I2 : 4, I1 : 2), (I1 : 2)	I2 I3 : 4, I1, I3 : 2, I2 I1 I3 : 2
11	{(I2 : 4)}	{(I2 : 4)}	I2 I1 : 4

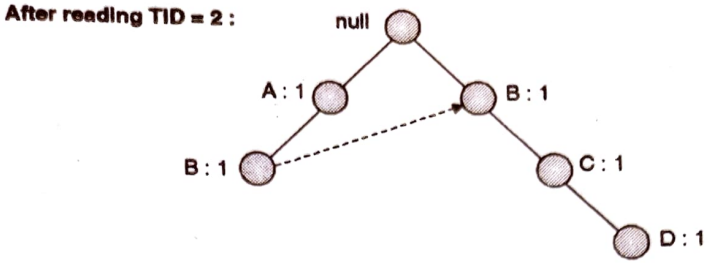
Ex. 4.9.4 : Consider the following dataset of frequent itemsets. All are sorted according to their support count. Construct the FP-Tree and find Conditional Pattern base for D.

TID	Items
1	{A, B}
2	{B, C, D}
3	{A, C, D, E}
4	{A, D, E}
5	{A, B, C}
6	{A, B, C, D}
7	{B, C}
8	{A, B, C}
9	{A, B, D}
10	{B, C, E}

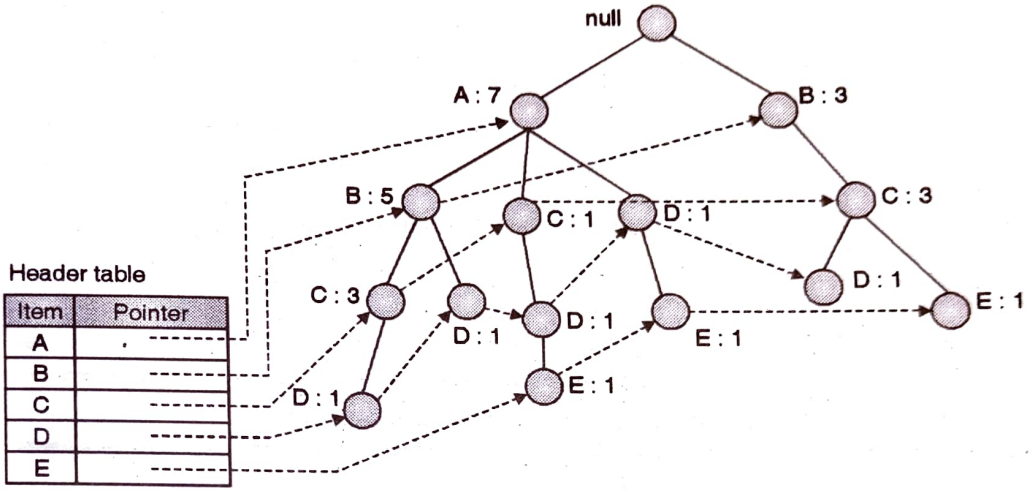
Soln. :

After reading TID = 1 :





Similarly for all the remaining transactions, FP tree is given below.



Conditional Pattern base for D

P = {(A:1,B:1,C:1), (A:1,B:1), (A:1,C:1), (A:1), (B:1,C:1)}

- We have the following paths with 'D'

  P = {(A:1,B:1,C:1),(A:1,B:1), (A:1,C:1), (A:1)} and {(B:1,C:1)}

- Support count of D = 1.
- Conditional Pattern Base (CPB)
  - To find all frequent patterns containing 'D' we need to find all frequent patterns in the CPB and add 'D' to them.
  - We can do this by constructing a new FP-Tree for the CPB
- Finding all patterns with 'D'
  - Again filter away all items < minimum support threshold

    (i.e.1 as Support of D =1)
  - Consider First Branch

    {(A:1,B:1,C:1),(A:1,B:1),

    (A:1,C:1), (A:1)} => {(A:4,B:2,C:2)}

    So append ABC with D

    We generate ABCD:1

- Similarly for other branch of the tree

  {(B:1,C:1)} => {(B:1,C:1)}

  So append BC with D

  We generate BCD : 1

- Recursively apply FP-growth
- So Frequent Itemsets found (with sup > 1): AD, BD, CD, ACD, BCD which are generated from CPB on conditional node D.

4.9.5 Benefits of the FP-Tree Structure

☞ **Completeness**

- The Long pattern of any transaction is never broken.
- For frequent pattern mining complete information is preserved.
- The method can mine short as well as long frequent patterns and it is highly efficient.
- FP-Growth algorithm is much faster than Apriori Algorithm.
- The search cost is reduced.

Syllabus Topic : Mining Various Kinds of Association Rules

4.10 Mining Various Kinds of Association Rules

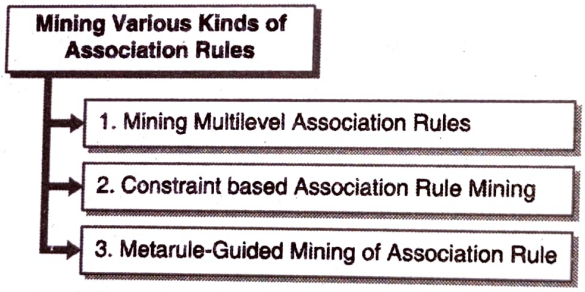


Fig. 4.10.1 : Mining Various Kinds of Association Rules

4.10.1 Mining Multilevel Association Rules

→ **(SPPU - May 16)**

Q. Explain the following terms : Multilevel association rules. **May 16, 3 Marks**

- Items are always in the form of hierarchy.
- Items which are at leaf nodes are having lower support.
- An item can be either generalized or specialized as per the described hierarchy of that item and its levels can be powerfully preset in transactions.
- Rules which combine associations with hierarchy of concepts are called Multilevel Association Rules.

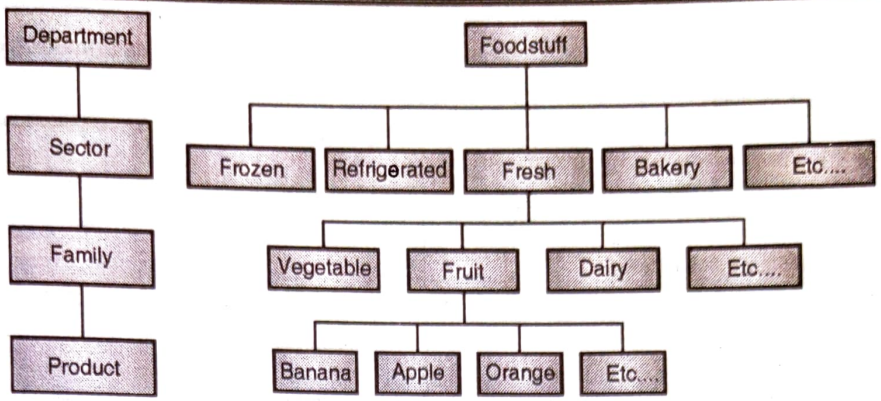


Fig. 4.10.2 : Hierarchy of concept

Support and confidence of multilevel association rules

- The support and confidence of an item is affected due to its generalization or specialization value of attributes.
- The support of generalized item is more than the support of specialized item
- Similarly the support of rules increases from specialized to generalized itemsets.
- If the support is below the threshold value then that rule becomes invalid
- Confidence is not affected for general or specialized.

Two approaches of multilevel association rule

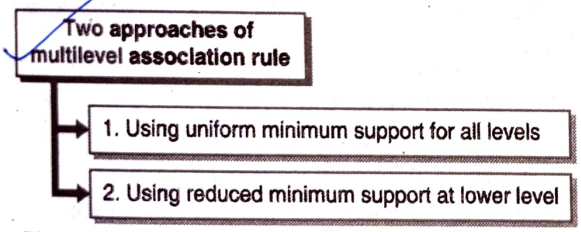


Fig. 4.10.3 : Two approaches of multilevel association rule

1. Using uniform minimum support for all levels

- Consider the same minimum support for all levels of hierarchy.
- As only one minimum support is set, so there is no necessity to examine the items of itemset whose ancestors do not have minimum support.
- If very high support is considered then many low level association may get missed.
- If very low support is considered then many high level association rules are generated.

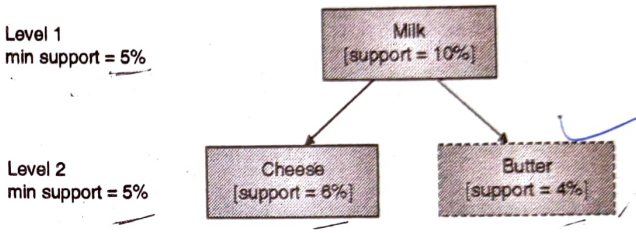


Fig. 4.10.4 : Example of uniform minimum support for all levels

2. Using reduced minimum support at lower level

- Consider separate minimum support at each level of hierarchy.
- As every level is having its own minimum support, the support at lower level reduces.

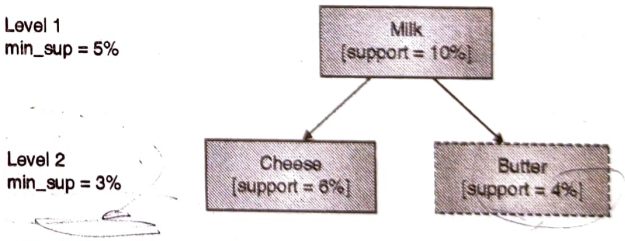


Fig. 4.10.5 : Example of reduced minimum support for lower levels

There are 4 search strategies :

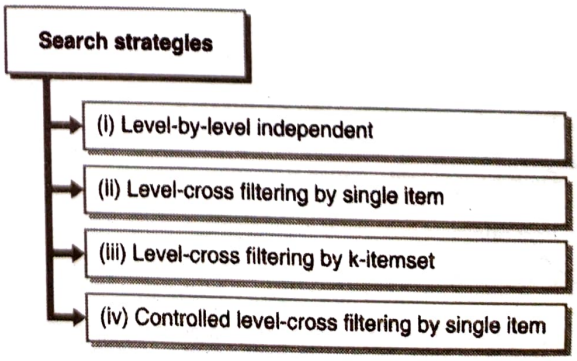


Fig. 4.10.6 : Search strategies

➜ **(i) Level-by-level independent**

- Its a full-breadth search method.
- The parent node is checked whether it's frequent or not frequent and based on that node is examined.

➜ **(ii) Level-cross filtering by single item**

The children of only frequent nodes are checked.

➜ **(iii) Level-cross filtering by k-itemset**

- Find the frequent k itemset at the parent level.
- Only the k itemset at next level is checked.

➜ **(iv) Controlled level-cross filtering by single item**

- This is the modified version of Level-cross filtering by single item.
- Some minimum support threshold is set for lower level.
- So the items which do not satisfy minimum support are checked for minimum support threshold this is also called "Level Passage Threshold".

Syllabus Topic : Constraint based Association Rule Mining

4.10.2 Constraint based Association Rule Mining

➜ **(SPPU - Dec. 16)**

Q. Explain the following terms : Constraints-based rule mining. **Dec. 16, 3 Marks**

Mining performed based on user specific constraints is called constraint-based mining.

Forms of constraints

1. Knowledge type constraints
2. Data constraints
3. Dimension / Level constraints
4. Interestingness constraints
5. Rule constraints

Mining query optimizer' must be incorporated in the mining process to exploit the constraints specified.

Syllabus Topic : Metarule-Guided Mining of Association Rule

4.10.3 Metarule-Guided Mining of Association Rule

- Specifies the *syntactic form* of the rules in which we are interested.
- Syntactic forms serves as the constraint.
- It is based on analysts experience, expectation, or intuition regarding data.
- To analyze the customers behaviour leading to the purchase of *Apple Products,* meta rule will be $P_1(C,Y)$ and $P_2(C,Z) \rightarrow$ buys (C, "Apple Products")

  Where, P_1, P_2 are the predicates on customer C for values Y and Z of predicates P_1 and P_2.
- Data mining system looks for the patterns which matches the given metarules. For example if two predicates Age and Salary are given to analyse whether the customer buys "Apple Product"

  age (C, "30..40") Λ Salary (C, "30K..50K") $\rightarrow$buys (C, "Apple Product")
- So generalise the metarule Guided association rule as a template like

  $P_1 \Lambda P_2 \Lambda \ldots \Lambda P_n \rightarrow Q_1 \Lambda Q_2 \Lambda \ldots \Lambda Q_r$

  Where, each P_i's and Q_j's are predicates

  And the number of predicates in the merarule is $p = n + r$

4.11 Solved University Question and Answer

Q. 1 Differentiate between : **(Oct. 16, 4 Marks)**

(i) Multilevel and multidimensional associations

(ii) Pattern-pruning and data-pruning constraints

Ans. :

(i) Multilevel and multidimensional associations

Multilevel associations

- Items are always in the form of hierarchy.
- tems which are at leaf nodes are having lower support.

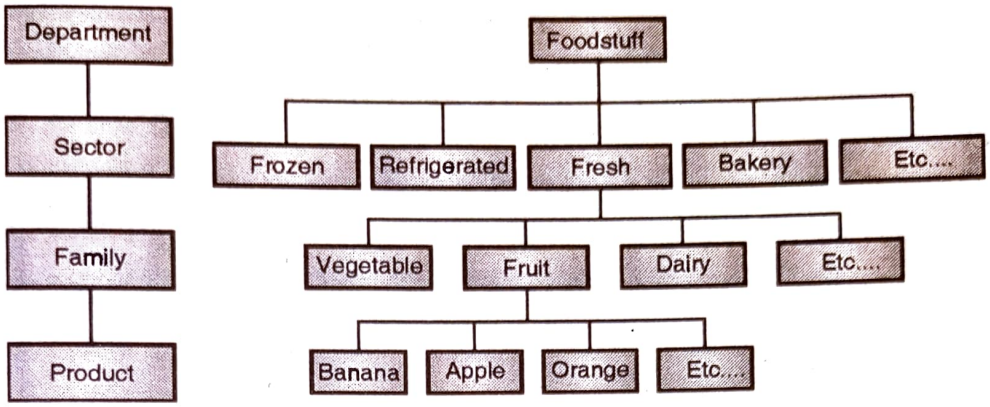


Fig. 1 : Hierarchy of concept

- An item can be either generalized or specialized as per the described hierarchy of that item and its levels can be powerfully preset in transactions.
- Rules which combine associations with hierarchy of concepts are called Multilevel Association Rules.

Support and confidence of multilevel association rules

- The support and confidence of an item is affected due to its generalization or specialization value of attributes.
- The support of generalized item is more than the support of specialized item
- Similarly the support of rules increases from specialized to generalized itemsets.
- If the support is below the threshold value then that rule becomes invalid
- Confidence is not affected for general or specialized.

Multidimensional associations

- **Single-dimensional rules :** The rule contains only one distinct predicate. In the following example the rule has only one predicate "buys".

  buys(X, "Butter") ⇒ buys(X, "Milk")

- **Multi-dimensional rules :** The rule contains two or more dimensions or predicates.
  - **Inter-dimension association rules :** The rule doesn't have any repeated predicate

gender(X,"Male") ∧ salary(X, "High") ⇒ buys(X, "Computer")

  - **Hybrid-dimension association rules :** The rule have many occurrences of same predicate i.e. buys.

gender(X,"Male") ∧ buys(X, "TV") ⇒ buys(X, "DVD")

- **Categorical attributes :** This have finite number of possible values and there is no ordering among values. Example : brand, color.
- **Quantitative attributes :** These are numeric values and there is implicit ordering among values. Example : age, income.

(ii) Pattern-pruning and data-pruning constraints

Pattern-pruning	data-pruning
If we can prune a frequent pattern P after checking constraints on it, then the entire subtree rooted at P in the pattern tree model will not be grown.	If we can prune Graph G from the data space search of P after data pruning checking G, will be pruned from the data search space of all nodes in the subtree rooted at P.
Pattern-pruning should be performed when $T_c(P) <=_p Tp$	We would perform data pruning checking for G if $T_d(P,G) <_q T_p$

□□□