# Apriori: a promising data warehouse tool for finding frequent itemset and to define association rules

Bharat S. Dhak, Mausami Sawarkar

Asst. Professor,PJLCE Nagpur, bharat.dhak@gmail.com, mausami_sawarkar@rediffmail.com, 7276010186, 9673344499

**Abstract**— Data Warehouse- not only the source of information or the place to store historical data but now its plays a vital role and act as a solution in today's fastest growing competitive world of IT and Business. It is obvious that if you keep information in proper place then you will definitely get it back in fast, efficient and easy mode. So to do so it is necessary to classify the data before placing it into the data warehouse. And it is also necessary to create analyses and correlate the association between the data items which is available or newly added. In this paper we represent here an algorithm which is used to find frequent item set from the given database which further used to classify items and to find association between. This paper also represents how to use correlation factor to know the actual and accurate bonding between items.

**Keywords**— apriori, association rule, correlation, candidate, dataset, data warehouse, data mining, frequent, lift,

## INTRODUCTION

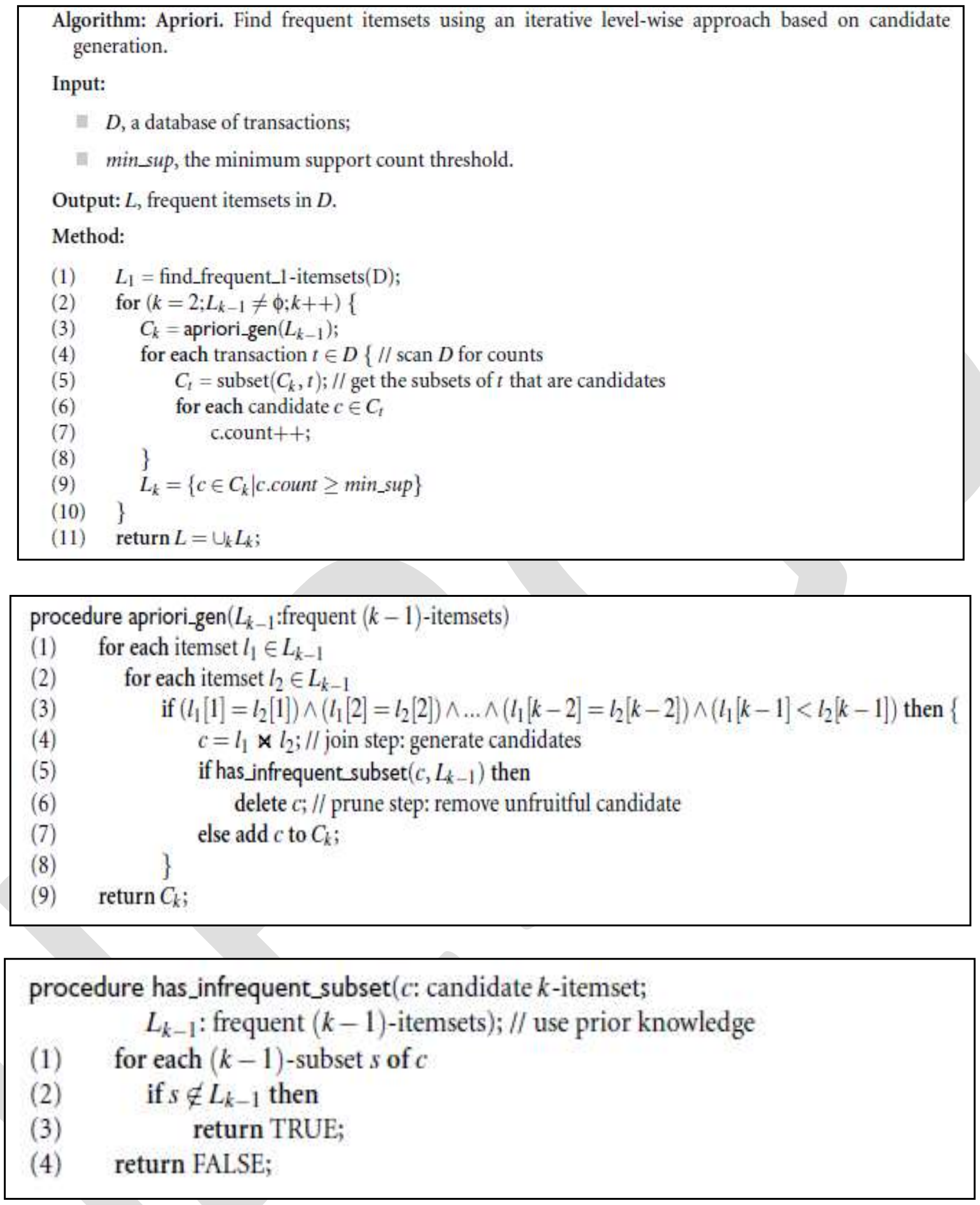
Data Warehousing & mining has recently motivated a database [4] practitioners and researchers mainly for building an application which is based on many fields such as market strategy, financial forecasts and decision support [10]. Many strategies and algorithms and application have been proposed to obtain useful and invaluable information from the large databases. Finding an Association rule is very important factor which is only possible after getting the list of frequent items. An association rule is of the form A => B: [Support=x%, confidence=y%]. And this rule has two very important measurements: support and confidence. The association rule mining problem is to find rules that satisfy user-specified minimum support and minimum confidence. Basically frequent patterns are patterns (such as item sets) that appear in a data set frequently. e.g., a set of items, such as milk & bread, which 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. If a substructure occurs frequently, it is called a (frequent) structured pattern. Finding such frequent patterns plays a vital role in mining associations, correlations. Again, it helps in data classification, clustering, and other data mining tasks as well. Thus, frequent pattern mining has become an important data mining task and a focused factor in data mining research.

## MARKET BASKET ANALYSIS

Market basket analysis [5] is a motivational example for frequent itemset mining which leads to the finding of associations and correlations among items in large transactional or relational data sets. With large amounts of data continuously being collected and stored, many industries are becoming interested in mining such patterns from their databases. The discovery of interesting correlation relationships among huge amounts of business transaction records can help in many business decision making processes, such as catalog design, cross-marketing, and customer shopping behavior analysis. A typical example of frequent itemset mining is market basket analysis. In market basket analysis we initiated with analyzing customer buying habits by finding associations between different items that customers place in their "shopping cart". These results help retailers and shopkeeper to develop business strategies by deciding that which items are frequently bought together by customers. For example, if customers are buying computer, then what is the percentage that they also tends to buy pen drive on the same trip to the supermarket? Such analysis can lead to increased sales by helping retailers do selective marketing and plan their shelf space. On the basis of same market basket analysis there exists an algorithm known as apriori algorithm which is going to be discussing in next section.

## THE APRIORI ALGORITHM:

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 [5] for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties. In apriori algorithm it actually do the iterative search of k-itemsets are used to explore (k + 1)-itemsets. In this we have to first need to find the set of frequent 1-itemsets by scanning the database to count for frequency of each items, and then checking for minimum support threshold. The resultant set is denoted by L1. Next, L 1 is used to find L 2, the set of frequent 2- itemsets, which is used to find L 3, and so on, until the system exhaust or no more is remaining. The finding of each L k requires one full scans of the database. The Apriori property is actually used to preserve the efficiency of algorithm. The Apriori property said that -All nonempty subsets of a frequent itemset must also be frequent. i.e., if {PQ} is a frequent itemset, then both {P} and {Q} should be a frequent itemset. Once the frequent pattern itemset found then we can find an association between them. To elaborate this; we have an example based on apriori which is used to find frequent patterns and then used to build an association rule between an itemset.



**Algorithm: Apriori.** Find frequent itemsets using an iterative level-wise approach based on candidate generation.

**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)       $L_k = \{c \in C_k | c.count \geq min\_sup\}$
(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 $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge ... \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ 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 $s \notin L_{k-1}$ then
(3)           return TRUE;
(4)   return FALSE;
```

Fig. 1 The Apriori algorithms for discovering frequent itemsets [5]

**WORKING OF AN APRIORI ALGORITHM:**

In fig. 1 we can see the general algorithm used to generate candidates and then to check and make itemset efficient. In general, Apriori Algorithm can be viewed as a two-step process[5]: (i) Generating all item sets having support factor greater than or equal to, the user specified minimum support. (ii) Generating all rules having the confidence factor greater than or equal to the user specified minimum confidence.

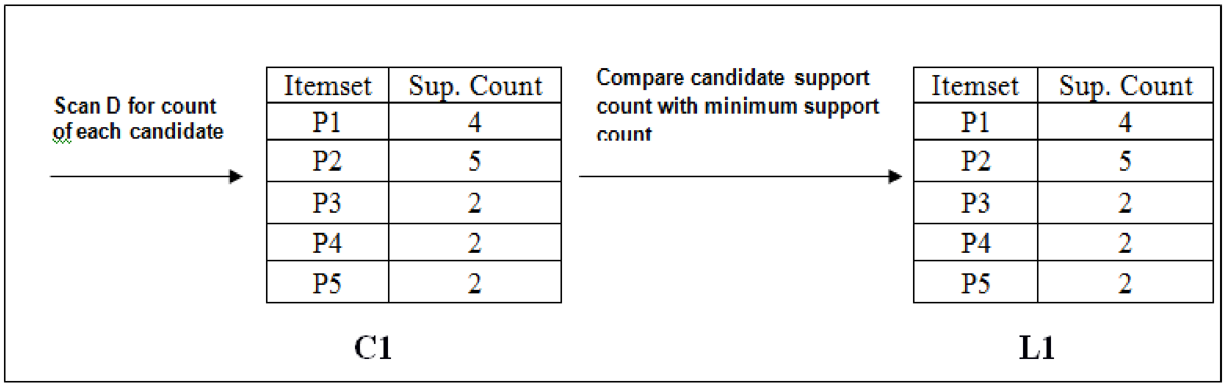
**EXAMPLE OF AN APRIORI ALGORITHM:**

In fig. 2, we have one example to consider which show the basic working of Apriori Algorithm, Consider a database, D, consisting of 5 transactions.
Consideration for example:
-Suppose min. support count required is 2 (i.e. min_sup = 2/5 = 40 %)
-Let minimum confidence required is 70%.
-We have to first find out the frequent itemset using Apriori algorithm.
-Then, we have to generate an Association rules using min. support & min. confidence.

| Transaction ID | List of Products purchased by customer |
|---|---|
| T101 | P1, P2, P5 |
| T102 | P2, P4 |
| T103 | P1, P2, P4 |
| T104 | P1, P2, P3, P5 |
| T105 | P1, P2, P3 |

Fig. 2 a sample transaction snapshot for consideration.

## Step 1: Generating 1-itemset Frequent Pattern


Scan D for count of each candidate
Compare candidate support count with minimum support count

| Itemset | Sup. Count |
|---|---|
| P1 | 4 |
| P2 | 5 |
| P3 | 2 |
| P4 | 2 |
| P5 | 2 |

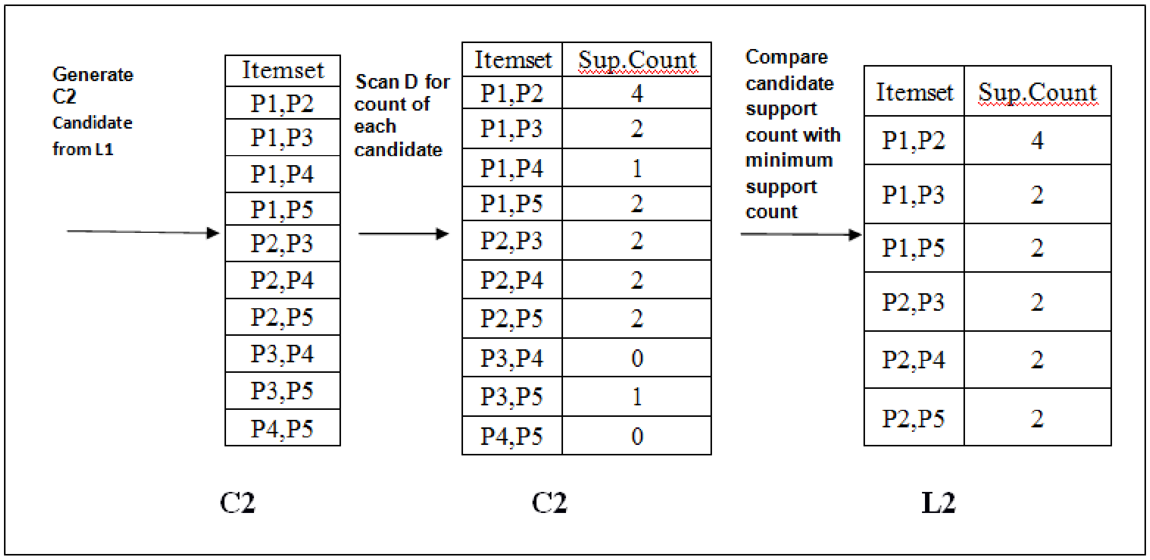
C1

| Itemset | Sup. Count |
|---|---|
| P1 | 4 |
| P2 | 5 |
| P3 | 2 |
| P4 | 2 |
| P5 | 2 |

L1

Here in 1st step, which is an iterative process, the dataset D is scanned and the support count (also called as frequency, count, occurrences) of individual product is calculated called as Candidate generation C1. Then we have to compare this with minimum support count which is given i.e. 2 (40%). Here we can see in the first iteration of the algorithm, each item is a member of the set of candidate.

## Step 2: Generating 2-itemset Frequent Pattern


Generate C2 Candidate from L1
Scan D for count of each candidate
Compare candidate support count with minimum support count
C2
C2
L2

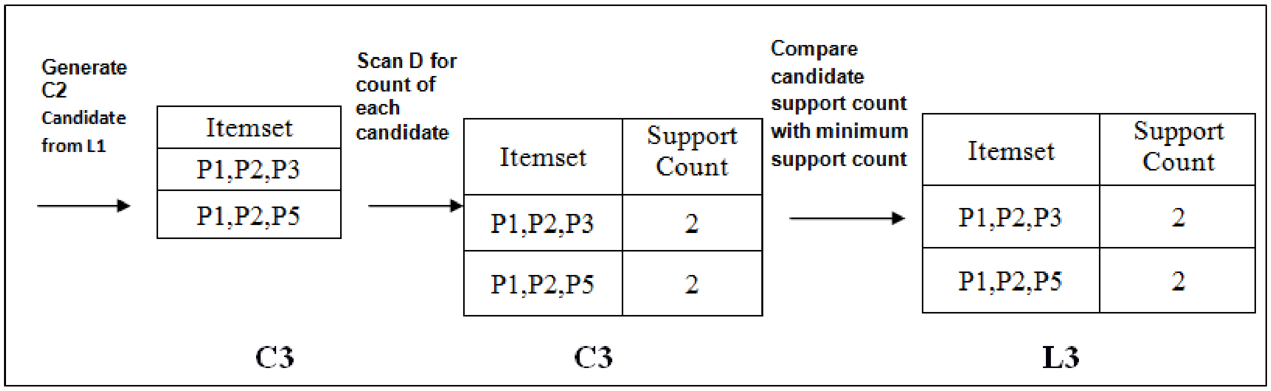
| Itemset |
|---|
| P1,P2 |
| P1,P3 |
| P1,P4 |
| P1,P5 |
| P2,P3 |
| P2,P4 |
| P2,P5 |
| P3,P4 |
| P3,P5 |
| P4,P5 |

| Itemset | Sup.Count |
|---|---|
| P1,P2 | 4 |
| P1,P3 | 2 |
| P1,P4 | 1 |
| P1,P5 | 2 |
| P2,P3 | 2 |
| P2,P4 | 2 |
| P2,P5 | 2 |
| P3,P4 | 0 |
| P3,P5 | 1 |
| P4,P5 | 0 |

| Itemset | Sup.Count |
|---|---|
| P1,P2 | 4 |
| P1,P3 | 2 |
| P1,P5 | 2 |
| P2,P3 | 2 |
| P2,P4 | 2 |
| P2,P5 | 2 |

To find out the set of frequent 2-itemsets, L2, we have to do L1 *Join* L1to generate a candidate set of 2-itemsets, C2. further, the transactions in D are scanned and the support count for each candidate itemset in C2 is gathered. The set of frequent 2-itemsets, L2, is then found out, consisting of those candidate 2-itemsets in C2 which is having minimum support.

**Step 3: Generating 3-itemset Frequent Pattern**


Generate C2 Candidate from L1
Scan D for count of each candidate
Compare candidate support count with minimum support count
C3
C3
L3

| Itemset |
|---|
| P1,P2,P3 |
| P1,P2,P5 |

| Itemset | Support Count |
|---|---|
| P1,P2,P3 | 2 |
| P1,P2,P5 | 2 |

| Itemset | Support Count |
|---|---|
| P1,P2,P3 | 2 |
| P1,P2,P5 | 2 |

The generation of the set of candidate 3-itemsets, C3, involves use of the Apriori Property. In order to find C3, we calculate L2*Join*L2.

-C3= L2 *Join*L2 = {{P1, P2, P3}, {P1, P2, P5}, {P1, P3, P5}, {P2, P3, P4}, {P2, P3, P5}, {P2, P4, P5}}.

-Here, Join step is completed and Prune step will be used to reduce the size of C3. Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can decide that four final candidates cannot possibly be frequent as there subsets are not frequent.

**Step 4: Generating 4-itemset Frequent Pattern**

Now, to generate C4 i.e. set of 4-itemset, we have to perform L3 Join L3. The join results is {{P1, P2, P3, P5}}, this itemset is pruned since its subset {{P2, P3, P5}} is not frequent. Thus, C4= φ, and algorithm terminates, having found all of the frequent items. Now our apriori algorithm is completed. Now, these frequent itemsets are used to generate strong association rules which further used to classify the data set or data items.

**Step 5: Generating Association Rules from Frequent Itemsets**

The concept of generating association rule [1, 2, 5] is very simple. Now, for each frequent itemset ***"l"***, generate all nonempty subsets of ***s***.

- For every nonempty subset ***s*** of ***l***, output the rule **"s-> (*l* - s)**

So here in our case,
We have L = {{P1}, {P2}, {P3}, {P4}, {P5}, {P1, P2}, {P1, P3},
{P1, P5}, {P2, P3}, {P2, P4}, {P2, P5}, {P1, P2, P3}, {P1, P2, P5}}.

–Let's take ***l*** = {P1, P2, P3}.
–then it's all nonempty subsets are: {P1, P2}, {P1, P3}, {P2, P3}, {P1}, {P2}, {P5}.
As mentioned before the minimum confidence threshold is 70%. Then with the consideration of above sample *l* = {P1, P2, P3}.

**Following are the generated Association rule:**

| Rule No | Rule | Confidence | Result | Remark |
|---|---|---|---|---|
| R1 | P1 ^ P2 → P3 | SC {P1, P2, P3}/SC {P1,P2} | 2 / 4 = 50% | R1 is rejected. |
| R2 | P1 ^ P3 → P2 | SC {P1, P2, P3}/SC {P1, P3} | 2 / 2 = 100% | R2 is selected. |
| R3 | P2 ^ P3 → P1 | SC{P1, P2, P3}/SC {P2, P3} | 2 / 2 = 100% | R2 is selected. |
| R4 | P3 → P1 ^ P2 | SC {P1, P2, P3}/SC {P3} | 2 / 2 = 100% | R2 is selected. |
| R5 | P2 → P1 ^ P3 | SC {P1, P2, P3}/SC {P2} | 2 / 5 = 40% | R1 is rejected. |
| R6 | P1 → P2 ^ P3 | SC {P1, P2, P3}/SC {P1} | 2 / 4 = 50% | R1 is rejected. |

So here we can find that only ***Rule No. R2, R3 and R4*** is strong association rule which are further useful for classification. Now, it is also true that the strong association is not all the time useful.

## Efficiency of an Apriori algorithm:

Yes definitely an apriori algorithm is most suitable, important and easy to implement algorithm for finding frequent patterns item set. Here are some methods [8] available for improving efficiency of an apriori algorithm [5].

A) Hash-based technique [5, 6]: where hashing method is used and itemsets dump into corresponding buckets for efficient use.
B) Transaction reduction: in this method we reduce the number of transactions which are scanned in the future iterations.
C) Partitioning: as the name suggest this method do the partition of the data to find candidate itemsets.
D) Sampling: sampling is the method where we have to take a subset of the given data and then applying all methodology onto it. Definitely applying apriori method on small subset will give us the result in less time and it also confirm us that the superset contains frequent patterns or not.
E) Dynamic itemset counting: in this method we have to add candidate itemsets at different points during a scanning of whole dataset.

Obviously every system has some drawback so this too. This apriori method is really the basic and easy to implement but it generates huge number of candidate sets and it may also need to continually scan the database and make sure a large set of candidates by pattern Matching. And it is obvious that to go for each transaction to calculate support those items. To overcome this drawback many scholars have proposed some methods which includes FP-Growth (Frequent Pattern) itemset mining in which we generate frequent patterns without generating candidates [13]. Another technique is Vertical data format; both apriori and FP-growth uses horizontal data format for analysis but we also can view the in vertical format like *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 this is the essence of the ECLAT (Equivalence CLASS Transformation) algorithm developed by Zaki.[7]

## CONCLUSION AND FUTURE WORK

The mentioned algorithm for mining frequent patterns and for defining association rules is definitely simple and easy to implement method for every data set and also useful for large amount of database [9] if used with any improving method mentioned above. The apriori method and its simple technique are also helpful to determine frequent patterns in sequential data and time-series database.

The current methods used for mining frequent patterns including apriori generate their result according to their algorithms without considering any other parameters. Some where we got the result where the items included in group are not associated with each other. Simply they are not correlate with each other. Yes this can be reviewed by using some correlation method like "lift" [5] but still some improvement is required in this field. An apriori algorithm can also be used for finding association between the satellite images taken by the shuttle for various purposes like images of earth taken before, on and after the time of disaster to prevent future calamities.

## REFERENCES:

[1] Agrawal, R et.al.,(1994) Fast algorithms for mining association rules. In Proc. 20th Int. Conf. Very Large Data Bases.

[2] Agrawal,R et al.,(1996), "Fast Discovery of Association Rules," Advances in Knowledge Discovery and Data Mining.

[3] Agrawal, R., Imielinski, T., and Swami et. al (1993). Mining association rules between sets of items in large databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data, 207-216.

[4] M.S. Chen, J. Han, P.S. Yu, "Data mining: an overview from a database perspective", IEEE Transactions on Knowledge and Data Engineering, 1996, 8, pp. 866-883.

[5] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publisher, San Francisco, CA, USA, 2001.

[6] J. Park, M. Chen and Philip Yu, "An Effective Hash-Based Algorithm for Mining Association Rules", Proceedings of ACM

Special Interest Group of Management of Data, ACM SIGMOD'95, 1995.

[7] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New Algorithms for Fast Discovery of Association Rules", Proc. 3$^{rd}$ ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'97, Newport Beach, CA), 283-296 AAAI Press, Menlo Park, CA, USA 1997.

[8] Shruti Aggarwal, Ranveer Kaur, "Comparative Study of Various Improved Versions of Apriori Algorithm", International Journal of Engineering Trends and Technology (IJETT) - Volume4Issue4- April 2013

[9] Agrawal, R., T. Imielin´ ski, and A. Swami (1993). "Mining association rules between sets of items in large databases". In Proceedings of the 1993 ACM SIGMOD International Conferenceon Management of Data, SIGMOD '93, New York, NY, USA, pp. 207–216. ACM.

[10] Association Rule Learning – Wikipedia, the free encyclopedia.

[12] Zawaidah, Jbara and Zanona (2011) " An Improved Algorithm for Mining Association Rules in Large Databases" World of Computer Science and Information Technology Journal (WCSIT) ISSN: 2221-0741 Vol. 1, No. 7, 311-316, 2011.

[13] J. Han, J. Pei, and Y. Yin. Mining Sequential Patterns without Candidate Generation (PDF), (Slides), Proc. 2000