# A Performance Based Transposition Algorithm for Frequent Itemsets Generation

**Sanjeev Kumar Sharma** *spd50020@gmail.com*
*Research Scholar*
*Devi Ahilya University, Takshashila Campus,*
*Khandwa Road, Indore (M.P.) India*

**Ugrasen Suman** *ugrasen123@yahoo.com*
*Associate Professor*
*Devi Ahilya University,Takshashila Campus,*
*Khandwa Road Indore (M.P.) India*

**Abstract**

Association Rule Mining (ARM) technique is used to discover the interesting association or correlation among a large set of data items. It plays an important role in generating frequent itemsets from large databases. Many industries are interested in developing the association rules from their databases due to continuous retrieval and storage of huge amount of data. The discovery of interesting association relationship among business transaction records in many business decision making process such as catalog decision, cross-marketing, and loss-leader analysis. It is also used to extract hidden knowledge from large datasets. The ARM algorithms such as Apriori, FP-Growth requires repeated scans over the entire database. All the input/output overheads that are being generated during repeated scanning the entire database decrease the performance of CPU, memory and I/O overheads. In this paper, we have proposed a Performance Based Transposition Algorithm (PBTA) for frequent itemsets generation. We will compare proposed algorithm with Apriori and FP Growth algorithms for frequent itemsets generation. The CPU and I/O overhead can be reduced in our proposed algorithm and it is much faster than other ARM algorithms.

**Keywords:** Data Mining, Association Rule Mining (ARM), Association rules.

## 1. INTRODUCTION

There are several organizations in the mainstream of business, industry, and the public sector, which store huge amount of data containing their transaction information online and offline. Such data may contain hidden information that can be used by an organization's decision makers to improve the overall profit. The efficient transformation of these data into beneficial information is thus a key requirement for success in these organizations. Data mining techniques are heavily used to search information and relationships that would be hidden in transaction data. There are various techniques of data mining such as clustering, classification, pattern recognition, correlation, and Association Rule Mining (ARM). The ARM is most important data mining technique that is used to extract hidden information from large datasets. In ARM algorithms, association rules are used to identify relationships among a set of items in database. These relationships are not based on inherent properties of the data themselves (as with functional dependencies), but rather based on co-occurrence of the data items.

The association rules are firstly introduced and subsequently implemented for the generation of frequent itemsets from the large databases [1],[2]. Association rules identify the set of items that are most often purchased with another set of items. For example, an association rule may state that 75% of customers who bought items A and B also bought C and D. The main task of every ARM is to discover the sets of items that frequently appear together called frequent itemsets.

ARM has been used for a variety of applications such as banking, insurance, medicine, website navigation analysis etc.

Frequent itemset can be produced from discovering useful patterns in customer's transaction databases. Suppose T ={$t_1$,$t_2$,$t_3$…..$t_n$} is a customer's transaction database, which is a sequence of transactions where each transaction is an itemset ($t_i \subseteq T$). Let J ={$i_1$,$i_2$,…..$i_n$} be a set of items, and D is a task relevant data**,** which can be a set of database transactions where each transaction T is a set of items such that $T \subseteq J$. Each transaction is associated with identifier called TID. Let A be a set of items and the transaction T is said to contain A if and only if $A \subseteq T$. The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transaction in D that contain $A \cup B$ (i.e. both A and B). This is taken to be the probability $P(A \cup B)$. The rule $A \Rightarrow B$ has confidence c in the transaction set D if c is the percentage of transaction in D containing A that also contain B. This is taken to be the conditional probability P(B|A). Therefore, the Support($A \Rightarrow B$) = $P(A \cup B)$ and Confidence($A \Rightarrow B$) = P(B|A). Those rules that satisfy both minimum support threshold and minimum confidence threshold are called strong. The values for support and confidence have to occur between 0% and 100%. The problem of mining association rules is to generate all rules that have support and confidence greater than some user specified minimum support and minimum confidence thresholds, respectively. This problem can be decomposed into the following sub-problems: i). All itemsets that have support above the user specified minimum support are generated. These itemsets are called the large itemsets. ii). For each large itemset, all the rules that have minimum confidence are generated as follows: for a large itemset X and any $Y \subset X$, if support(X)/support(X - Y) ≥ minimum- confidence, then the rule $X - Y \rightarrow Y$ is a valid rule.

There are various algorithm of ARM such as Apriori, FP-growth, Eclat etc. The most important algorithm of ARM is Apriori, which is not only influenced the association rule mining community, but it has affected other data mining fields as well [3]. Association rule and frequent itemset mining has become now a widely research area and hence, faster and faster algorithms have been presented. Numerous of them are Apriori based algorithms or Apriori modifications. Those who adapted Apriori as a basic search strategy, tended to adapt the whole set of procedures and data structures as well [4],[5],[6],[7]. Since the scheme of this important algorithm was not only used in basic association rules mining, but also used in other data mining fields such as hierarchical association rules [8],[9],[10], association rules maintenance [11],[12],[13] , sequential pattern mining [14], episode mining [15] and functional dependency discovery [16],[17] etc. Basically, ARM algorithms are defined into two categories; namely, algorithms respectively with candidate generation and algorithms without candidate generation. In the first category, those algorithms which are similar to Apriori algorithm for candidate generation are considered. Eclat may also be considered in the first category [9]. In the second category, the FP-Growth algorithm is the best–known algorithm. Table-1, defines the comparison among these three algorithms [3].

| Algorithm | Scan | Data Structures |
|---|---|---|
| Apriori | M+1 | HashTable &Tree |
| Eclat | M+1 | HashTable &Tree |
| FP-Growth | 2 | PrefixTree |

**TABLE 1:** Comparison of Algorithms

The main drawback of above discussed algorithms given above is the repeated scans of large database. This may be a cause of decrement in CPU performance, memory and increment in I/O overheads. The performance and efficiency of ARM algorithms mainly depend on three factors; namely candidate sets generated, data structure used and details of implementations [18]. In this paper we have proposed a Performance Based Transposition Algorithm (PBTA) which uses these three factors. Transactional database is considered as a two dimension array which works on boolean value dataset. The main difference between proposed algorithm and other algorithms is that instead of using transactional array in its natural form, our algorithm uses transpose of array i.e. rows and columns of array are interchanged. The advantage of using transposed array

is to calculate support count for particular item. There is no need to repeatedly scan array. Only by finding the row sum of the array will give the required support count for particular item, which ultimately results in increased efficiency of the algorithm. In the first pass of PBTA, we will receive all the support count value for the 1-itemset. Mining of association rules is a field of data mining that has received a lot of attention in recent years.

The rest of this Paper is organized as follows. In Section 2, we will explain the Apriori algorithm through association rules mining. Section 3 introduces our proposed PBTA algorithm with an illustration and compare with other algorithms. Experimental results are shown in Section 4. The concluding remarks are discussed in Section 5.

## 2. APRIORI ALGORITHM

ARM is one of the promising techniques of data mining to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. There are several ARM algorithms such as Apriori, FP-Growth, Eclat. The Apriori algorithm is also called the level-wise algorithm to find all of the frequent sets, which uses the downward closure property. The advantage of the algorithm is that before reading the database at every level, it prunes many of the sets which are unlikely to be frequent sets by using the Apriori property, which states that all nonempty subsets of frequent sets must also be frequent. This property belongs to a special category of properties called anti-monotone in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well. Using the downward closure property and the Apriori property the algorithm works as follows. The first pass of the algorithm counts the number of single item occurrences to determine the $L_1$ or single member frequent itemsets. Each subsequent pass, K, consists of two phases. First, the frequent itemsets $L_k$-1 found in the $(k-1)^{th}$ pass are used to generate the candidate itemsets $C_k$, using the Apriori candidate generation algorithm. Therefore, the database is scanned and the support of the candidates in $C_k$ is determined to ensure that $C_k$ itemsets are frequent itemsets [19].

Pass 1
1. Generate the candidate itemsets in $C_1$
2. Save the frequent itemsets in $L_1$

Pass k

1. Generate the candidate itemsets in $C_k$ from the frequent itemsets in $L_{k-1}$
   a) Join $L_{k-1}$ $p$ with $L_{k-1}$ q, as follows:
      insert into $C_k$
      select $p$.item1, $p$.item2, . . . , $p$.item$_{k-1}$, $q$.item$_{k-1}$
      from $L_{k-1}$ $p$, $L_{k-1}$q
      where $p$.item1 = $q$.item1, . . . $p$.item$_{k-2}$ = $q$.item$_{k-2}$,
      $p$.item$_{k-1}$ < $q$.item$_{k-1}$
   b) Generate all ($k$-1)-subsets from the candidate itemsets in $C_k$
   c) Prune all candidate itemsets from $C_k$ where some ($k$-1)-subset of the candidate itemset is not in the frequent itemset $L_k$-1
2. Scan the transaction database to determine the support for each candidate itemset in $C_k$
3. Save the frequent itemsets in $L_k$

We will use Apriori algorithm for ARM as a basic search strategy in our proposed algorithm. The proposed algorithm will adapt the whole set of procedures of Apriori but the data structure will be different. Also, the proposed algorithm will use the transposition of transactional database as data structures.

## 3. PERFORMANCE BASED TRANSPOSITION ALGORITHM (PBTA)

In Apriori algorithm, discovery of association rules require repeated passes over the entire database to determine the commonly occurring set of data items. Therefore, if the size of disk and database is large, then the rate of input/output (I/O) overhead to scan the entire database may be very high. We have proposed Performance Based Transposition Algorithm (PBTA), which improves the Apriori algorithm for repeated scanning of large databases for frequent itemsets generation. In PBTA, transaction dataset will be used in the transposed form and the description of proposed algorithm is discussed in the following sub-sections.

### 3.1 Candidate Generation Algorithm

In the candidate generation algorithm, the frequent itemsets are discovered in k-1 passes. If k is the pass number, $L_{k-1}$ is the set of all frequent (k-1) itemsets. $C_k$ is the set of candidate sets of pass k and c denotes the candidate set. $I_1, I_2 \ldots I_k$ are the itemsets[19]. The candidate generation procedure is as follows.

Procedure Gen_candidate_itemsets ($L_{k-1}$)
$C_k = \Phi$
for all itemsets $I_1 \in L_{k-1}$ *do*
for all itemsets $I_2 \in L_{k-1}$ *do*
if $I_1[1] = I_2[1]$ ^ $I_1[2] = I_2[2]$ ^ … ^ $I_1[k-1] < I_2[k-1]$
then $c = I_1[1], I_1[2] \ldots I_1[k-1], I_2[k-1]$
$C_k = C_k \cup \{c\}$

### 3.2 Pruning Algorithm

The pruning step eliminates some candidate sets which are not found to be frequent.

Procedure Prune($C_k$)
*for* all $c \in C_k$
*for* all (k-1)-subsets d of c do
*if* $d \notin L_{k-1}$
*then* $C_k = C_k - \{c\}$

### 3.3 PBTA Algorithm Description

The PBTA uses candidate generation and pruning algorithms at every iteration. It moves from level 1 to level k or until no candidate set remains after pruning. The step-by-step procedure of PBTA algorithm is described as follows.

1. Transpose the transactional database
2. Read the database to count the support of $C_1$ to determine $L_1$ using sum of rows.
3. $L_1$= Frequent 1- itemsets and k:= 2
4. While (k-1 ≠ NULL set) do
   Begin
   $C_k$ := Call Gen_candidate_itemsets ($L_{k-1}$)
   Call Prune ($C_k$)
   for all itemsets $i \in I$ do
   Calculate the support values using dot-multiplication of array;
   $L_k$ := All candidates in $C_k$ with a minimum support;
   K:k+1
   End
5. End of step-4

#### 3.3.1 An Illustaration

Suppose we have a transactional database in which the user transactions from T1 to T5 and items from A1 to A5 are stored in the form of boolean values, which is shown in Table 1. We have assumed that this database can be generated by applying Apriori algorithm for frequent itemsets generation.

.

| | A1 | A2 | A3 | A4 | A5 |
|---|---|---|---|---|---|
| T1 | 1 | 0 | 0 | 0 | 1 |
| T2 | 0 | 1 | 0 | 1 | 0 |
| T3 | 0 | 0 | 0 | 1 | 1 |
| T4 | 0 | 1 | 1 | 0 | 0 |
| T5 | 0 | 0 | 0 | 0 | 1 |

**TABLE 2:** Transaction Database

Consider the transpose of transactional database of Table 1 is stored in Table 2 by applying metrics arithmetics that can be used in our proposed algorithm (PBTA). Assume the user-specified minimum support is 40%, and then the steps for generating all frequent item sets in proposed algorithm will be repeated until NULL set is reached. In PBTA, transactional dataset will be used in the transposed form. Therefore, candidate set and frequent itemset generation process will be changed as compared to Apriori algorithm. In the first pass, we will receive $L_1$.

{A1}→ 1, {A2}→2, {A3}→1, {A4}→2, {A5}→3
$L_1$ := { {A1}→1, {A2}→2, {A3} →1, {A4} → 2, {A5} → 3}

Then the candidate 2-itemset will be generated by performing dot-multiplication of rows of array, as array consist of boolean values, the resultant cell will be produce in the form of 1. If the corresponding cells of the respective rows have 1, otherwise 0 will be in the resultant cell. In this approach, we will receive a new array consisting of candidate 2-itemsets to get the higher order of itemsets. The above process between rows of array can be performed to find out the results.

| | | | | | |
|---|---|---|---|---|---|
| A1 | 1 | 0 | 0 | 0 | 0 |
| A2 | 0 | 1 | 0 | 1 | 0 |
| A3 | 0 | 0 | 0 | 1 | 0 |
| A4 | 0 | 1 | 1 | 0 | 0 |
| A5 | 1 | 0 | 1 | 0 | 1 |

**TABLE 3:** Transpose Database of Transaction

In the second pass, where k=2 , the candidate set $C_2$ becomes
$C_2$: = {{A1*A2}, {A1*A3}, {A1,*A4}, {A1*A5}, {A2*A3}, {A2*A4},{A2*A5},{A3*A4},{A3*A5},{A4*A5} }
The pruning step does not change $C_2$ as all subsets are present in $C_1$.
Read the database to count the support of elements in $C_2$ to get:
{ {A1*A2}→ 0, {A1*A3}→0, {A1*A4} →0, {A1*A5}→1, {A2*A3}→ 1,
{A2*A4} →1, {A2*A5}→0, {A3*A4}→0, {A3*A5}→0, {A4*A5}→1}} and reduces to
$L_2$ = { {A1*A5}→1, {A2*A3}→1, {A2*A4}→1, {A4*A5}→1}}

In the third pass where k=3, the candidate generation step proceeds:
In the candidate generation step,

- Using {A1*A5} and {A4*A5} it generates {A1*A4*A5}
- Using {A2*A3} and {A2*A4} it generates {A2*A3*A4}
- Using {A2*A4} and {A4*A5} it generates {A2*A4*A5}

Thus, $C_3$:= { {A1*A4*A5}, {A2*A3*A4}, {A2*A4*A5}}

The pruning step prunes {1,4,5}, {2,3,4},{2,4,5} as not all subsets of size 2, i.e., {1,4}, {3,4}, {2,5} are not present in $L_3$.

So $C_{3:}= \Phi$

Hence the total frequent sets becomes $L:= L_1 \cup L_2$.

By comparing both Apriori and proposed algorithm, we found that Apriori algorithm requires multiple passes of the dataset to calculate support count for different itemsets. Therefore, in the case of Apriori, the record pointer moves the order of candidate item set * no of records while in the case of PBTA algorithm, record pointer moves equal to only order of candidate itemsets. For example, if we have to find out support count value for 2-itemset in a dataset having 5 items with 5 records using Apriori algorithm number of time record pointer will be 2*5 i.e. 10 while in case of our proposed algorithm it will be 2 only.

## 4. EXPERIMENTAL EVALUATIONS

The performance comparison of PBTA with classical frequent pattern-mining algorithms such as Apriori, FP-Growth is presented in this Section. All the experiments are performed on 1.50 Ghz Pentium-iv desktop machine with 256 MB main memory, running on Windows-XP operating system. The program for Apriori, FP-Growth and proposed algorithm PBTA were developed in Java JDK1.5 environment. We report the experimental results on three synthetic boolean datasets with 300K, 500K and 700K records, each having 130 columns. The datasets consists of boolean values are shown in table 3, table 4 and table 5. The performances results of Apriori, FP-growth and PBTA are shown with Fig. 1, Fig. 2 and Fig. 3 with data size 300K, 500K, and 700K represented in the graphical form. The X-axis in these graphs represents the support threshold values while the Y-axis represents the response times (in milliseconds) of the algorithms being evaluated as shown.

| | Response Time (in ms) | | |
|---|---|---|---|
| **Support Count (in %)** | **Apriori** | **FP-Growth** | **PBTA** |
| 50 | 50 | 40 | 10 |
| 40 | 100 | 80 | 40 |
| 30 | 150 | 100 | 150 |
| 20 | 300 | 230 | 60 |
| 10 | 500 | 400 | 80 |

**TABLE 4:** Response Time Comparison of algorithms with 300k database

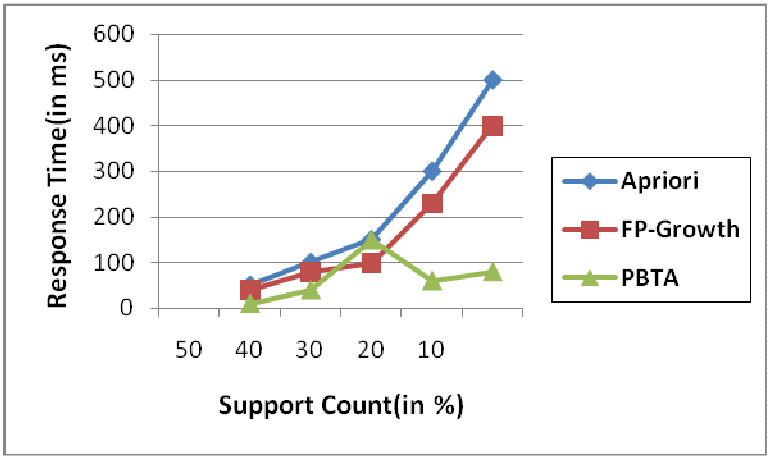


**FIGURE 1:** Performance analysis of algorithms (300k database)

In the first case, we have considered the transactional database with 300k in size as it is shown in Fig. 1. We have compared the performance of Apriori, FP-Growth with PBTA on the basis of response time. The observation shows that as the support count will be decreased and the response time taken by PBTA is much lesser then Apriori and FP-Growth algorithm.

| | Response Time (in ms) | | |
|---|---|---|---|
| **Support Count (in %)** | **Apriori** | **FP-Growth** | **PBTA** |
| 50 | 84 | 67 | 17 |
| 40 | 167 | 134 | 67 |
| 30 | 251 | 167 | 251 |
| 20 | 501 | 384 | 100 |
| 10 | 835 | 668 | 134 |

**TABLE 5:** Response Time Comparison of algorithms with 500k database

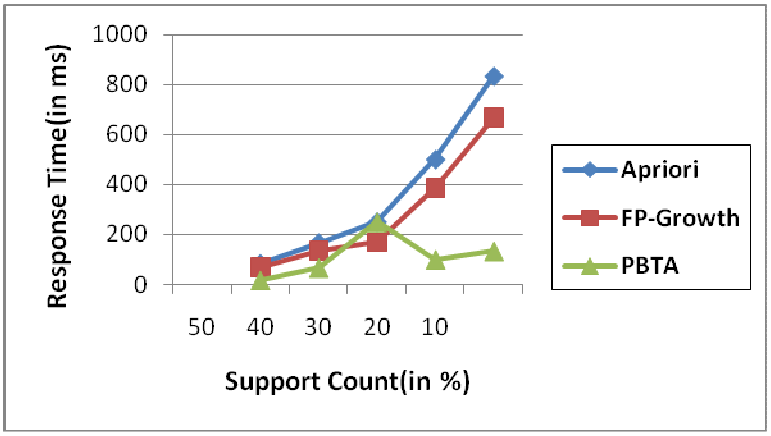


**FIGURE 2:** Performance analysis of algorithms (500k database)

In the another case, the transactional database with 500k size is considered, which is shown in Fig. 2. Hence, we have observed that as the support count threshold is reduced and the response time taken by PBTA is much lesser then Apriori and FP-Growth algorithm.

| | Response Time (in ms) | | |
|---|---|---|---|
| **Support Count (in %)** | **Apriori** | **FP-Growth** | **PBTA** |
| 50 | 117 | 93 | 23 |
| 40 | 233 | 186 | 93 |
| 30 | 350 | 233 | 350 |
| 20 | 699 | 536 | 140 |
| 10 | 1165 | 932 | 186 |

**TABLE 6:** Response Time Comparison of algorithms with 700k database

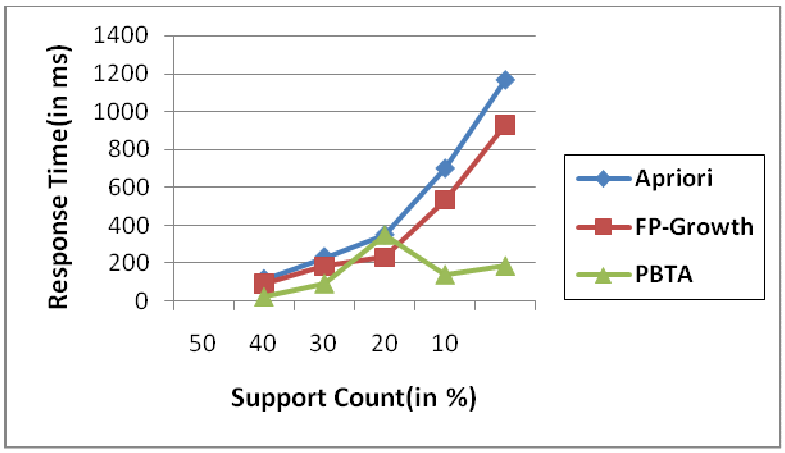


**FIGURE 3:** Performance analysis of algorithms with 700k database

In the Fig. 3, the transactional database with 700k is used. Here, we observed that in comparison to Apriori and FP-Growth, PBTA will take lesser response time while support threshold is reduced. This PBTA algorithm may be used for extraction of useful frequent hyperlinks or URLs for web recommendation [20].

## 5. CONCLUSIONS

ARM algorithms are important to discover frequent itemsets and patterns from large databases. In this paper, we have designed a Performance Based Transposition Algorithm (PBTA) for generation of frequent itemsets similar to Apriori algorithm. The proposed algorithm can improve the efficiency of Apriori algorithm and it is observed to be very fast. Our algorithm is not only efficient but also very fast for finding association rules in large databases. The proposed algorithm drastically reduces the I/O overhead associated with Apriori algorithm and retrieval of support of an itemset is quicker as compared to Apriori algorithm. This algorithm may be useful for many real-life database mining scenarios where the data is stored in boolean form. At present this algorithm is implemented for only boolean dataset that can also be extend to make it applicable to all kind of data sets.

## 6. REFERENCES

[1] R.Agrawal, T. Imielinski, and A.Sawmi, “Mining association rules between sets of items in large databases” in proc. of the ACM SIGMOD Conference on Management of Data, pages (207-216), 1993.

[2] D.W-L.Cheung, S.D.Lee, and B.Kao, “A general incremental technique for maintaining discovered association rules” in Database Systems for advanced Applications, pages 185-194, 1997.

[3] M.H.Margahny and A.A.Mitwaly, “Fast Algorithm for Mining Association Rules” in the conference proceedings of AIML, CICC, pp(36-40) Cairo, Egypt, 19-21 December 2005.

[4] J.S.Park, M-S. Chen and P.S.YU, “An effective hash based algorithm for mining association rules” in M.J. Carey and D.A. Schneider, editors, Proceedings of the 1995 ACM SIG-MOD International Conference on Management of Data, pages 175-186, San Jose, California, 22-25. 1995.

[5] S.Brin, R.Motwani, J.D.Vllman, and S.Tsur, “Dynamic itemset counting and implication rules for market basket data” SIGMOD Record (ACM Special Interest Group on Management of Data), 26(2): 255, 1997.

[6] H.Toivonen, “Sampling large databases for association rules” in the VLDB Journal, pages 134-145, 1996.

[7] A.Sarasere,E. Omiecinsky, and S.Navathe, “An efficient algorithm for mining association rules in large databases” in Proceedings of 21St International Conference on Very Large Databases (VLDB) , Zurich, Switzerland, Also Catch Technical Report No. GIT-CC-95-04 1995.

[8] Y.F.Jiawei Han., ”Discovery of multiple-level association rules from large databases” In the Proceedings of AIML 05 Conference, 19-21 December 2005, CICC, Cairo, Egypt the 21St International Conference on Very Large Databases (VLDB), Zurich, Switzerland, 1995.

[9] Y.Fu., “Discovery of multiple-level rules from large databases”, 1996.

[10] D.W-L.Cheung, J.Han, V.Ng, and C.Y.Wong, “Maintenance of discovered association rules in large databases : An incremental updating techniques” In ICDE, pages 106-114,1996.

[11] D.W-L.Cheung, S.D.Lee, and B.Kao, “A general incremental technique for maintaining discovered association rules” in Database Systems for advanced Applications, pages 185-194, 1997.

[12] S.Thomas, S.Bodadola, K.Alsabti, and S.Ranka, “An efficient algorithm for incremental updation of association rules in large databases” in Proc. KDD'97, Page 263-266, 1997.

[13] N.F.Ayan, A.U. Tansel, and M.E.Arkm, ”An efficient algorithm to update large itemsets with early pruning”, in Knoweldge discovery and Data Mining, pages 287-291,1999.

[14] R.Agrawal and R.Srikant, “Mining sequential patterns”, In P.S.Yu and A.L.P. Chen, editors, Proc.11the Int. Conf. Data engineering. ICDE, pages 3-14. IEEE pages, 6-10, 1995.

[15] H.Mannila, H.Toivonen, and A.I.Verkamo, “Discovering frequent episodes in sequences” in proceedings of the First International Conference on knowledge Discovery and Data mining”, pages 210- 215. AAAI pages, 1995.

[16] Y.Huhtala, J.Karkkainen, P.Pokka, and H.Toivonen, ”TANE: An efficient algorithm for discovering functional and approximate dependencies”,. The computer Journal, 42(2): 100-111, 1999.

[17] Y.Huhtala, J.Kinen, P.Pokka, and H.Toivonen, “Efficient discovery of functional and approximate dependencies using partitions” in ICDE, pages 392- 401, 1998.

[18] F.Bodon, “A Fast Apriori Implementation”, in the Proc.1st IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI2003, Melbourne,FL).CEUR Workshop Proceedings 90, A acheme, Germany 2003.

[19] Akhilesh Tiwari, Rajendra K. Gupta, and Dev Prakash Agrawal, “Cluster Based Partition Approach for Mining Frequent Itemsets” in the International Journal of Computer Science and Network Security(IJCSNS), VOL.9 No.6,pp(191-199) June 2009.

[20] Sanjeev Kumar Sharma, Ugrasen Suman, “A Semantic Enhance Data-Mining Framework for Web Personalization”, In the proceedings of International conference on Data Analysis, Data Quality and Metadata Management(DAMD-2010), pp(49-57),Singapore,2010.