A New Approach for Mining Frequent K-itemset

H. Ravi Sankar and M.M. Naidu

Abstract—Discovery of frequent itemsets is an important problem in Data Mining. Most of the previous research based on Apriori, which suffers with generation of huge number of candidate itemsets and performs repeated passes for finding frequent itemsets. To address this problem, we propose an algorithm for finding frequent K-itemsets in which the itemsets whose length is less than K will be pruned from the database and will not be considered for further processing which reduces the size and number of comparisons to be performed. In addition to this, it generates 1-itemset as a data pre processing step which saves time and makes execution fast. The experimental results are included.

Index Terms— **Frequent itemset, algorithm, database, apriori, support**

I. Introduction

Association rule mining is a focused area in today's data mining research. It usually consists of two phases viz., discovery of frequent itemsets and generation of rules from the discovered frequent itemsets. Finding frequent itemsets has gained popularity because it has more number of applications viz., market basket analysis, catalog design, add-on sales, and store layout and customer segmentation.

The efficiency of any algorithm to find the frequent itemset is based on three factors viz., generation of candidate keys, data structures used and way of implementation. Frequent Itemset Mining (FIM) is mainly based on minimum support value, which finds all itemsets with supports no less than a user-specified minimum support threshold. Several algorithms have been proposed [7, 8, 9, 11, 12, 16, 19, 20, 22, 23, 24, 28] in this area.

Apriori like algorithms performs more number of scans and generates huge number of candidate keys. To find the frequent K-itemset, it is necessary to start the algorithm from frequent 1-itemset, 2-itemset……. to frequent K-itemset. The proposed method generates frequent K-itemset with a minimum support directly from the database. At the data warehousing level, 1-itemset will be generated while processing a transaction. Prior to the execution of the proposed algorithm, 1-itemset is available which prunes the first pass from the apriori-like algorithm which in turn saves time.

The outline of this paper is as follows: Section II presents definitions of frequent itemset and an outline of apriori algorithm for finding frequent itemsets. Section III reviews the related work done in this area. Section IV explains the motivation for proposing the algorithm. Section V presents the proposed algorithm, database schema and an illustrative example. Section VI gives the experimental results and the conclusions of study are given in section VII.

II. Background

The definitions of frequent itemset are follows.

Definition 1.1 [30]**:** Let 'I' be a finite set of attributes called items and D be a finite multi set of transactions. Each transaction T⊆ D is a set of items is usually called an itemset. The length or size of an itemset is the number of items that contains. An itemset of length 'K' is referred to as K-itemset.

Definition 1.2 [26]: Let L= {I1, I2 …Im} be a set of literals, called items. Let a non empty set of items T be called an itemset. Let D be a set of variable length itemsets, where each itemset T⊆ L. We say that an itemset T supports an item x∈ L if x is in T. We say that an itemset T supports an itemset X⊆L if T supports every item in the set X. Each itemset has an associated measure of its statistical significance, called support. The support of the itemset T in the set D is:

$$\text{Support}(X, D) = \frac{|\{T \in D \mid T \text{ supports } X\}|}{|D|}$$

In other words, the itemset 'X' holds in the set 'D' with support's', if's' is the fraction of itemsets in 'D' supporting 'X'. A frequent itemset is an itemset, whose support is above a user-defined threshold.

Introduction to Apriori:

It is an influential algorithm for mining frequent itemsets for Boolean association rules. This algorithm uses prior knowledge of frequent itemset properties. This algorithm iteratively finds all possible itemsets that have support greater or equal to a given minimum support value. The first pass of the algorithm counts item occurrences to determine the frequent 1-itemsets. In each of the next passes, the frequent itemsets, L_{k-1} is found in the (K-1)th pass are used to generate the candidate itemsets C_K, using apriori-gen function described below. Then the database is scanned and the support of

Manuscript received July 15, 2007. This work was supported in part by the TEQIP.

F. A. H. Ravi sankar is with the CTRI, Rajahmundry, India, Phone: 91-9849418571, email : hravisankar@india.com

S. B. M.M. Naidu was with Sri Venkateswara Univesity, Tirupati, India. He is now the Vice-Principal and Professor, Department of Computer science and engineering, Phone: 91-9848348473., e-mail: mmnaidu@yahoo.com.

candidates in C_K is counted. The output of the first phase of the apriori algorithm consists of a set of K-itemsets (K=1,2,……..) that have support greater or equal to a given minimum support value. Figure 1 presents a description of the algorithm.

```
Scan D to find L1;
For (k=2; L_{k-1} <> 0; k++) do begin
        C_k = apriori_gen (L_{k-1});
   For all transactions t ∈ D do begin
     C_t = subset (C_k, t);
      For all candidates c∈ C_t do
            c.count ++;
    end
   L_k = { c ∈ C_k | c.count ≥ minsup } ;
end
   Answer = U_k L_k ;
```

Figure 1: Apriori Algorithm

III. Related work

FIM is first introduced by Agrawal et al [21] in 1993. Most of the researchers designed various algorithms using bottom-up approach, top-down approach, hashing, indexing and sampling. Similarly they represented the data either in horizontal, vertical, bit vector, or trie, which is a rooted directed tree [27]. Later many researchers extend the research on closed frequent itemsets [14] and designed various algorithms viz., TFP [13], CLOSET, MAFIA [10], CHARM [15], CLOSET+[17], LCM[18]. An itemset is said to be closed if it has no proper superset with the same support.

The first algorithm proposed is AIS and later it was improved and named it as 'Apriori [1]'. Apriori uses bottom-up approach which performs a breadth-first search by generating candidate k+1 itemsets from frequent k-itemsets. The Apriori algorithm uses the Ariori-gen algorithm repeatedly to generate candidates and then count their supports by reading the entire database once. Later many variations of Apriori have been proposed to reduce the number of scans or the number of candidates to be generated. AprioriTid [25] is generated in which the database is not used at all for counting the support of candidate itemsets after the first pass. Rather, an encoding of the candidate itemsets used in the previous pass is employed for this. In later passes, the size of the encoding can become much smaller than the database, thus saving much reading effort.

Based on Apriori algorithm various algorithms viz., OCD [4], Direct Hashing and Pruning [5] (DHP), Dynamic Itemset counting [6] (DIC), Partition [3] and Sampling [2] algorithms have been developed.

Recent studies in frequent itemsets concentrated on Maximal frequent itemsets (MFI). A frequent itemset is said to be maximal if it has no proper supersets and that are themselves frequent. The problem of discovering the frequent set can be reduced to the problem of discovering the MFI.

IV. Motivation

Apriori algorithm needs to scan the database multiple times. When mining a huge database, multiple database scans are costly. One feasible strategy to improve the efficiency of Apriori algorithm is to reduce the number of database scans.

The Apriori algorithm has to generate a huge number of candidates. Storing and counting these candidates are tedious. To attack this problem, some studies focus on reducing the number of candidates.

Apriori-like algorithms use full database scan once to find the 1-itemset. If the database consists of more number of transactions, for example, 1000, the time takes to execute for generating 1-itemset is more. For avoiding pass-1, it is proposed to generate 1-itemset at the data warehousing level. In each iteration of the apriori, 'join' operation is to be performed on all itemsets to generate candidate keys for the next iteration. It is expensive to generate a huge number of candidate sets. For finding frequent 4-itemsets, apriori starts the algorithm by generating frequent 1-itemset, frequent 2-itemset, frequent 3-itemset followed by frequent 4-itemset which takes more number of scans and time.

Research has been done widely for finding all frequent itemsets, maximal frequent itemsets and closed frequent itemsets and most of the algorithms are based on apriori which iterative in nature. No algorithm suggested a method to find the frequent K- itemset directly from the database without generating 1 to (N-1) frequent itemsets. So, finding frequent K-itemset without generating K-1 itemsets is an open area of the research which reduces number of scans, time and generating huge number of candidate itemsets. With this aim, a novel method is proposed in which itemsets whose length is less than K will be pruned from the database at the initial stage of the algorithm that reduces further processing overhead. With this algorithm, the number of scans and time will be reduced. The efficiency of the algorithm increases if the value of K > (N/2) where N is the longest itemset in the database.

V. PROPOSED METHOD

In this section, a new method for finding frequent K-itemset is presented. In this method, 1-itemset will be generated directly from the transaction database which performs efficient data preprocessing at the data warehousing level while processing a transaction. To find the frequent K-itemset, the algorithm starts the searching of itemsets whose length is at least K, i.e., the itemsets whose length is less than K will be pruned from the database and will not be considered for further processing which reduces the size and number of comparisons to be performed.

a) Notation:

The following notation is used in development of proposed algorithm.

TID = Transaction code for identifying each transaction uniquely
ICODE = Item code for identifying each item uniquely
ICOUNT = Number of occurrences of an item for a set of transactions
NCOUNT = Number of items of a transaction
M = Minimum support threshold
I = Item table
I_1 = Frequent 1-itemset table
K = length of itemset
N = Itemset table
L_K = K-itemset table where $K \geq 1$
N_K = Pruned itemset table where NCOUNT $\geq$ K
D = Transaction table
D_K = Pruned transaction table based on K

b) Database Schema:

The initial database schema is shown in fig. 2.

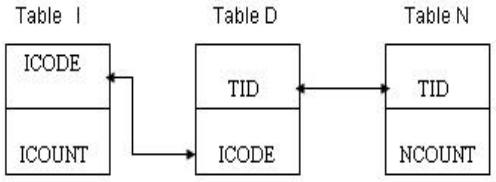

Figure 2. Database schema

- ICODE is the primary key for table I.
- TID is the primary key for table N.
- {TID, ICODE} is the primary key for table D where TID is the foreign key that references table N and ICODE is the foreign key that references the table I.

As and when a transaction is processed, ICOUNT is updated and a record is appended to itemset table.

c) Algorithm

1. Generate I_1 from I by applying M
2. Generate L_K using join operation K times on I_1
3. Prune the tuples from N whose NCOUNT < K that results N_K
4. Delete from D such of those tuples with TID associated with NCOUNT < K that results D_K
5. Find the occurrence frequency for each itemset of L_K using D_K. If it is greater than or equal to M, then that K-itemset is frequent K-itemset.

d) Illustrative Example: Find Frequent 3-Itemsets with Minimum support of 2.

TABLE I

Transaction table, D

TID	ICODE
T100	I1
T100	I2
T100	I5
T200	I2
T200	I4
T300	I2
T300	I3
T400	I1
T400	I2
T400	I3
T500	I1
T500	I3
T600	I2
T600	I3
T700	I1
T700	I3
T800	I1
T800	I2
T800	I3
T800	I5
T900	I1
T900	I2
T900	I3

TABLE II

Item table, I

ICODE	ITEMCOUNT
I1	6
I2	7
I3	7
I4	1
I5	2

TABLE III

Itemset table, N

TID	NCOUNT
T100	3
T200	2
T300	2
T400	3
T500	2
T600	2
T700	2
T800	4
T900	3

The transaction table, D, consisting of 9 transactions, the item table, I, consisting of 5 records and the itemset table, N, consisting of 9 records are shown in tables I, II and III respectively. Following the step-1 of the algorithm, the frequent 1-itemset table, I_1 is generated as shown in table IV. The 3-itemset table, L_3 is generated using join operations 3 times on I_1 and shown in table V. After deleting the tuples from N whose NCOUNT < 3 resulted N_3 as shown in table VI. After deleting the tuples, D, whose NCOUNT < 3 resulted D_3 as

shown in table VII. Using step 5, frequent 3-itemsets are found as I1I2I3 and I1I2I5.

TABLE IV

Frequent 1-itemset, I_1

ICODE	ITEMCOUNT
I1	6
I2	7
I3	7
I5	2

TABLE V

K-itemset, L_K

ICODE
I1I2I3
I1I2I5
I1I3I5
I2I3I5

TABLE VI

Pruned itemset table, N_K

TID	NCOUNT
T100	3
T400	3
T800	4
T900	3

TABLE VII

Pruned transaction table, D_K

TID	ICODE
T100	I1
T100	I2
T100	I5
T400	I1
T400	I2
T400	I3
T800	I1
T800	I2
T800	I3
T800	I5
T900	I1
T900	I2
T900	I3

VI. Experimental Results

In this section we show the results of our computational experiments. The proposed method was implemented on HCL PC powered by PIII 850 MHZ, 128MB RAM, Windows 98 2nd Edition, FoxPro 2.6. The transaction data was collected from a retail shop and a transaction database was designed which consists of 250 transactions, 825 records, 15 items and maximum of 5 itemsets. Minimum support is plotted on X-axis and CPU time (in seconds) on Y-axis as shown in Fig. 3. With a minimum support of 2% the proposed method takes 86 seconds for execution, while the apriori algorithm takes 152 seconds for finding the frequent 3-itemsets. Similarly for a minimum support of 10%, the proposed method takes 45 seconds while the Apriori takes 81 seconds.

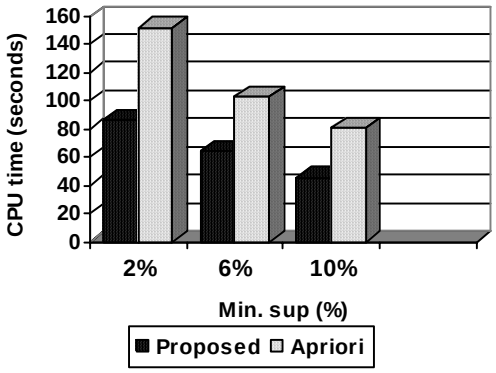

Figure 3. Frequent 3-itemset

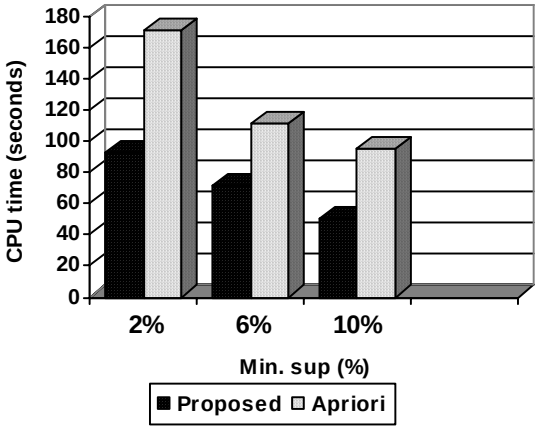

Figure 4. Frequent 4-itemset

The computational time for frequent 4-itemsets with a minimum support of 2%, 6% and 10% is shown in fig. 4. The proposed method takes 92 seconds for finding frequent 4-itemset with a minimum support of 2%, while the Apriori takes 171 seconds; means the efficiency of the proposed algorithm is increased in terms of execution time when the minimum support is low.

VII CONCLUSION

Finding frequent itemsets using Apirori algorithm requires multiple scans of the database and generation of candidate keys for each iteration. Based on our performance study, the proposed algorithm achieves high accuracy and efficiency which can be credited to the following distinguished features. 1) Pruning of 1-itemset generation which will be generated at the time of transaction 2) All itemsets whose length is less than K will be pruned from the transaction database which reduces the size of the database and further processing overhead.

VII. References

[1] Agarwal, R., and Srikant R., 1994, "Fast algorithms for mining association rules," Proc. 20th VLDB, Santiago, Chile.

[2] Toivonen, H.., 1996, "Sampling large databases for Association Rules," Proc. VLDB conference.

[3] Sarasere, A., Omiecinsky, E., and Navathe, S., 1995. "An efficient Algorithm for Mining Association Rules in Large Databases", Proc. 21st VLDB conference, Swizerland..

[4] Mannila, H., Toivonen, H. and Verkamo, A., 1994, "Improved methods for finding association rules," Proc. AAAI Workshop Knowledge Discovery.

[5] Park, J., Chen, M., and Yu. P., 1995, "An effective hash based algorithm for mining association rules," Proc. ACM Special Interest Group on Management of Data (SIGMOD).

[6] Brin, S., Motwani, R., Ullman, J., and Tsur, S., 1997, "Dynamic Itemset Counting and Implication Rules for Market Basket Data", Proc. ACM Special Interest Grou on Management of Data (SIGMOD).

[7] Gopalan, R., and Sucahyo, Y.G., 2003, "Fast Frequent Itemset Mining using compressed data representation," Proc. IASTED International Conference on Databases and Applications (DBA'2003), Innsbruck, Austria.

[8] Fu, H., and Nguifo, E.M., 2003, "A Fast Scalable Algorithm to Build Closed Itemsets from Large Data" Proc. Third IASTED International Conference on Artificial Intelligence and Applications (AIA03), Benalmádena, Spain, pp 210.

[9] Szymon, Jaroszewicz and Simovici, Dan A., 2004, "Interestingness of frequent itemsets using Bayesian networks as background knowledge" Proc. Tenth ACM SIGKDD International conference on Knowledge discovery and data mining, USA, pp: 178-186.

[10] Doug Burdick, Manuel Calimlim, Johannes Gehke, 2001, "MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases", Proc. 2001 International Conference on Data Engineering (ICDE'01), pp:443-452.

[11] Han, J., Pei, J., Yin, Y., and Mao, R., 2004, " Mining Frequent patterns without candidate generation: A Frequent-Pattern Tree Approach". Data Mining and Knowledge Discovery, Vol. 8(1), Pp: 53-87.

[12] Sriphaew, K., and Theeramunkong, T., 2002, "A new method for finding generalized frequent itemsets in generalized association rule mining". Proc. Seventh International Symposium on Computers and Communications, pp: 1040-1045.

[13] Wang, J., Han, J., Lu, Y., and Tzvetkov, P., 2005, "TFP: An Efficient Algorithm for Mining Top-K Frequent Closed Itemsets". IEEE Transactions on Knowledge and Data Engineering. Vol. 17(5), pp. 652-664.

[14] Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L., 1999, "Discovering Frequent Closed Itemsets for Association Rules". Proc. Seventh International Conference on Database Theory (ICDT'99), pp. 398-416.

[15] Zaki, M.J., and Hsiao, C.J., 2002, "CHARM: An Efficient Algorithm for Closed Itemset Mining", Proc. 2002 SIAM International Conference on Data Mining (SDM '02), pp. 457-473.

[16] Lin, D., and Kedem, M., 2002, "Pincer-Search: An Efficient Algorithm for Discovering the Maximum Frequent Set", IEEE Transactions on Knowledge and Data Engineering. Vol. 14(3).

[17] Wang, J., Pei, J., and Han J., 2003, " Closet+: Searching for the best strategies for mining frequent closed itemsets", SIGKDD.

[18] Uno, T., Asai, T., Uchida, Y., and Arimura, H., 2003, " LCM : An efficient algorithm for enumerating frequent closed itemsets", SDM.

[19] Bodon, F., 2003, " A fast apriori implementation", Informatics Laboratory, Hungary.

[20] Borgelt, C and Kruse, R, 2003, " Induction of Association rules: Apriori implementation", Department of Knowledge processing and Language Engineering, Germany.

[21] Agarwal, R., Imielinski, T., and Swami, A., 1993, "Mining association rules between sets of items in large databases", Proc. ACM SIGMOD conference on Management of Data, Washington, D.C., May, 1993.

[22] Grahne, G., Zhu, J., 2005, "Fast Algorithms for Frequent Itemset Mining Using FP-Trees", IEEE Transactions on Knowledge and Data Engineering. Vol. 17(10).

[23] Shen, Li., Shen H., and Cheng, L., 2003, " New Algorithms for Efficient Mining of Association Rules", School of computing and IT, Griffith University, Australia.

[24] Gouda, K., Zaki, J.M., 2001, "Efficiently Mining Maximal Frequent Itemsets", IEEE Transactions on Knowledge and Data Engineering.

[25] Chao Li, Z., Lian He, P., and Lei, M, 2005 " A High Efficient AprioriTID algorithm for mining association rule", Proc. Fourth International Conference on Machine Learning and Cybernetics, Guangzhou.

[26] Wojciechowski, M., and Zakrzewicz, M., 2005 " HASH-MINE: A New Framework for Discovery of Frequent Itemsets", Poznan University of Technology, Poland.

[27] Bodon, F., 2005, "Trie based Apriori Implementation for Mining Frequent Itemsequences, SIGKDD Workshop on Open Source Data Mining Workshop (OSDM'05), Chicago, USA.

[28] Seno, M., and Karypis, G., 2001, "LPMiner: An Algorithm for Finding Frequent Itemsets Using Length-Decreasing support constraint", IEEE Transactions on Knowledge and Data Engineering.

[29] Baralis, E., Cerquitelli, T., and Chiusano, S, 2005, "Index Support for Frequent Itemset Mining in a Relational DBMS", Proc. 21st International Conference on Data Engineering (ICDE).

[30] Christos, B., George, T. and Loannis, V, 2005, "Mining for contiguous frequent itemsets in transaction databases", Proc. IEEE workshop on Intelligent data application and advanced computing systems, Sofia, Bulgaria.