International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue VI, June 2019- Available at www.ijraset.com

An Efficient Tree based Data Structure for Incremental Frequent Itemset Generation

Archana Gupta[1], Akhilesh Tiwari[2], Sanjeev Jain[3]
[1]*Research Scholar, Madhav Institute of Technology & Science, Gwalior (M.P.), India*
[2]*Department of CSE & IT, Madhav Institute of Technology & Science, Gwalior (M.P.), India*
[3]*PDPM- Indian Institute of Information Technology, Design & Manufacturing, Jabalpur (M.P.), India*

Abstract: ***In this paper, we present an efficient data structure for storing large database to support incremental mining. Incremental algorithms are used to modify the results of earlier mining to derive the results for the incremented database. Various business databases are incremental in nature and knowledge is required from the updated database to infer some decisions and predictions.***
At times it is needed to ignore the previous knowledge obtained from the old database. Thus incremental algorithm should be able to manipulate the inferred rules when some part of the present database is deleted. To implement algorithms for incremental mining, first of all, an efficient data structure to store the data is needed. An easiest and most understandable way of storing the data is in the relational form. But in case of incremental implementation relational form of the data will incur high complexity.
Varieties of tree based data structures are proposed by different researchers in [1] , [2] , [3]. In this paper we will present a novel tree based data structure for storing incremental database and its comparison with the existing ones.

I. INTRODUCTION

In present scenario, database mining has attracted many database communities due to its wide applicability to improve business strategies. Association mining techniques are very important in data mining applications. ARM techniques have very wide scope of applications.
For Example – given a database of sales transaction of retail store, it would be interesting to discover the association among the items such that the presence of some item in the transaction imply the presence of some another items in the same transactions. These association rules are useful in framing various business strategies so as to improve the performance. It is also useful in medical field in various ways.
Many Algorithms have already been proposed for Incremental association rule mining. In incremental approach, when dataset is updated by new information then it has to preserve old information along with new information and it has to mine the knowledge from the entire database.
In many algorithms proposed by different researchers ,it is needed to rescan the old database to infer the results. This approach is not efficient as database size is very huge. So we need some implementation in which there should be no need to rescan the old database again to infer results.
There are two approaches of association rule mining:

A. Apriori based method
B. Tree based method

Tree based methods are more suitable for incremental mining as compared to apriori based methods. The first tree based method for association rule mining is FP Tree based method.
FP Tree is a data structure used to store the information of database in tree form. A database is converted in tree structure and then algorithm is applied on tree data structure to infer the association knowledge. In this method there is no need to scan the original database once it got convert in tree.
That is the advantage of tree based method which need scanning of original database once only.

In incremental mining since database will get update time to time so accordingly FP tree needs to be modified. Thus two types of implementations are possible

1) Rescanning of old database is needed and then it will generate the tree for the entire database.
2) No rescanning of old database is needed and it will generate the tree by updating the old tree with the new incremented information.

Second approach is more efficient as it won't need rescanning of old database whose size is considerably very large. Thus, improving the overall efficiency of the system.

In real life applications, databases are continuously incremented with time. The incremental mining algorithms always need to preserve the old data along with the new data to infer new knowledge. So the main challenge is to design the algorithm for association mining without any need for rescanning the old database.

Many researchers have worked on the same issue by taking into consideration the same factor. [1] proposes DB-Tree and potential frequent pattern Tree (PotFP-Tree) algorithms for mining incremental association rules in updated database. [3] proposes FELINE CATS Tree and AFPIM algorithm for incremental mining. [2] proposes CAN Tree for mining frequent itemsets in incremental database

If we are able to generate the tree for the given database then the information in the database is converted in to tree form. Then the next responsibility is to design the algorithm to retrieve the information from the tree and perform association mining to infer knowledge.

In this paper we are going to present the designing of tree based data structure for storing the incremental database and we will compare our data structure with already designed tree based data structure given in [1] , [2], [3] . We will compare the methods in terms of space requirement, time requirement and the number of scanning needed for generation the data structure.

II. OUTLINE OF THE PAPER

First of all, we will present the procedure to generate an efficient tree based data structure from the incremental database. And after that we will compare our tree structure with existing ones and will show which the better one is.

The organization of the rest of the paper is as follows : section 3 presents COMVAN : A novel tree based data structure for the incremental association rule mining. Section 4 presents different Tree based Data structure given by different researchers in [1], [2], [3]. Section 5 presents the comparison among all four methods. And finally, section 6 presents conclusions.

III. A NOVEL TREE BASED DATA STRUCTURE

This method is based on the lexicographic order of the items. First of all, all the attributes must be arranged in the lexicographic order. Then the algorithm will read one item at a time from the record and insert the item in the tree based on the following rules :

A. Root will point to the first item in the first record..
B. Associated with every node in the tree, a list of values is there. A value at a particular position k represents the total number of records in which that item is at kth position.
C. Every next item in the record will be added as a child of the preceding item in the record.
D. If the item is already there in the children of the preceding node then no new node will be added, only the corresponding value in the list will get increment by 1.
E. If the first item of the record is not in the children of the root, then it will search for the first item in the tree using DFS. Once item is found in the tree, then it will follow the same procedure to enter the record in the tree. If that item is not found in the tree then it will be added as a child of the root.
F. The length of the list associated with every node will depend on the level of the tree at which that node is located. The number of values in the list will be same as the number of level. The number at particular position in the list represents the total number of records having that item at this position in reverse. If the length of the list is n, then the value at the n^{th} position represents the number of record in which the position of item is first. Similarly the value at n-1 position represents the number of records in which the position of the item is second. The value at first position represents the number of records with its n^{th} position. So, in general the value at k^{th} position represents the cardinality of the records where the position of that item is (n-k+1).

The sample transactional database for implementing the above algorithm is given in table 1.

Tid	Items bought
101	a, b, c, d, e, g
102	a, b, c, d, h
103	a, b, c
104	a, b, c, d
105	a, b
106	b, c, d, e, g
107	b, c, d, f
108	b, c, d, f
109	b, c, g
110	b, c, d
111	b, d, e
112	b, c, d
113	b, c, d
114	d, e, h
115	d, f
116	d, e
117	c
118	d
119	d, f, h
120	a, b, c, d , f, h

Table 1. The example transactional database

Figure 1 gives the novel tree structure generated with the above algorithm.

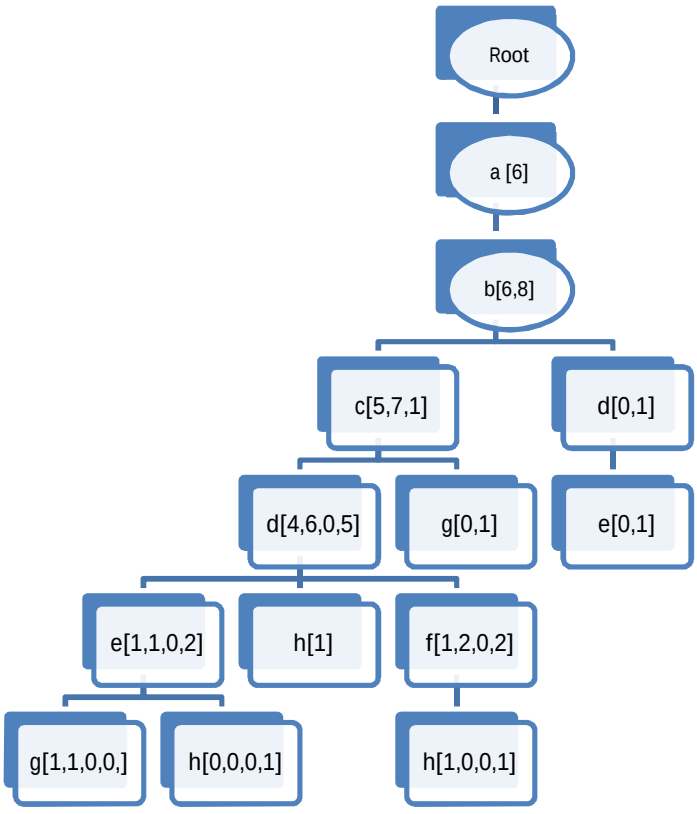

Figure 1.The Novel tree of the example database given in table 1.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue VI, June 2019- Available at www.ijraset.com

IV. DIFFERENT TREE BASED DATA STRUCTURE GIVEN BY DIFFERENT RESEARCHERS

This section presents proposed data structure for incremental ARM implementation. Section 4.1 discuss the DB Tree proposed in [1]. Section 4.2 presents the designing of CATs tree and section 4.3 presents designing of CAN Tree..

A. DB tree

DB Tree proposed in [1] is a generalized form of FP Tree. It stores all items in the database as well as frequency count of all items in decreasing order of support. It is constructed in the same way as FP Tree except it includes all items instead of only frequent 1-iutemsets as in the FP Tree. Same as FP Tree , it needs exactly two scan of the transactional database to construct DB Tree. It will have more nodes as compare to the FP Tree as it includes all item of the transactional database irrespective of their count. It can be consider as special FP Tree with support = 0. Then also DB Tree is much compact then the original database because many transactions share the common path in the DB Tree. Figure 2 gives the DB Tree for the example transaction database given in table 1.

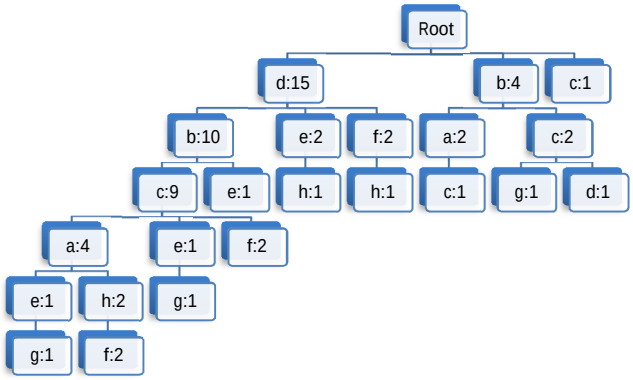

Figure 2 : The DB-Tree for the example database given in Table 1

B. CATs tree

Cheung and Zaine[10] designed the CATS tree (Compressed and Arranged Transaction Sequences tree) mainly for interactive mining. It extends the concept of the FP-tree to improve storage compression, and allows frequent pattern mining without candidate itemset generation.

The CATS tree for the given transaction database is constructed as follows:

It needs to scan the database once only to construct the tree. First of all it reads the first transaction (tuple) and first item will be connected with the root directly and then subsequent items are added as child of the previous item and forms a path in the tree. Then it will read nect transaction from the database. If there is already a path same as the items of the new transaction, the new transaction will be merged with the node at the highest frequency level. The remainder of the transaction is then added to the merged nodes and this process is repeated recursively until all common items are found. If some of the items in the transaction left then the remaining items are added as a new branch to the last merged node. If the frequency of a node becomes higher than its ancestors then it has to swap with the ancestor so as to ensure that its frequency should be lower than or equal to the frequencies of its ancestors. Figure 3 shows the CATS tree for the transaction database given in table 1.

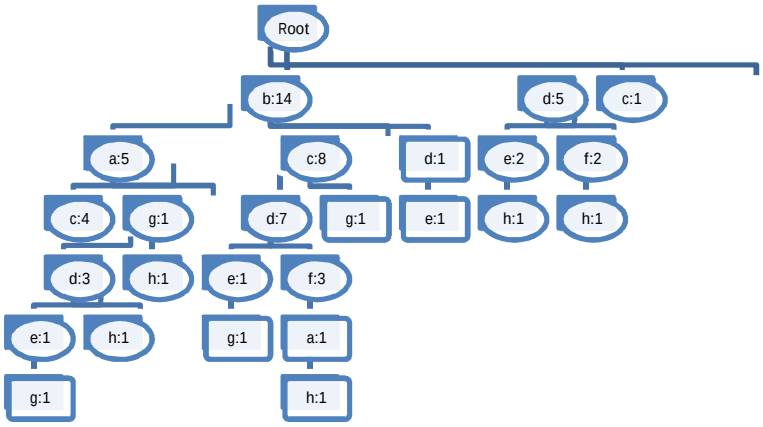

Figure 3: The CATS Tree for the example database given in Table 1

C. CAN tree

The construction of the CAN tree [13] requires only one scanning of the transaction database unlike FP-Tree constructions where two scanning of database are required. In CAN tree constructions, all items in the transactions must be arranged in some canonical order, which is determined by the user prior to the mining process. Items can be arranged in some alphabetic order or in some lexicographic order. Items can be arranged in some specific order depending on their properties such as their price values, their validity of some constraints, etc. the above ordering methods are frequency independent, but the items can also be arranged according to some fixed frequency related ordering such as in descending order of the global frequency of the original database. Once the ordering of the items are decided then all transaction will follow the same order, even in the subsequently incremented database independent of the frequency ordering of items in the incremented database.

Figure 4 shows the CAN tree for the transaction database given in table 1. It will add a transaction in the tree by forming a pathfrom root to the leaf node. To add a transaction in the tree, it will check the list of items in the new transaction. If first item is already present at level 1 then it will simple increment the frequency count of that item by 1 else add a completely new path for the given transaction. Now if first item is present at level 1 then it will search for the second item in the children of first item , if it present then it will increment the count else it will add a completely new path for the remaining transaction. In general , if ith item is present in the children of i-1th item then increment its count else add a new path for ith item to nth item in the transaction at i-1th node.

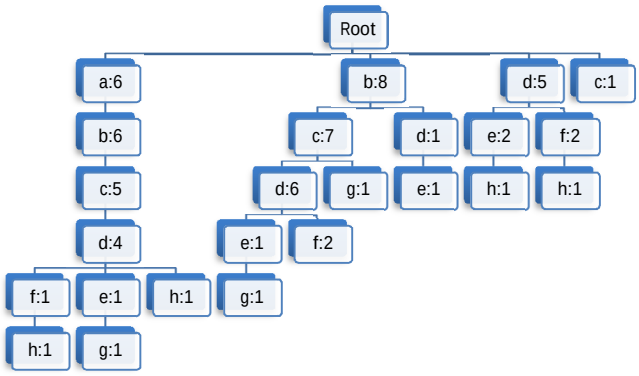

Figure 4 : The CAN Treefor the Example database given in table 1

D. Comparisons

DB Tree	CATs Tree	CAN Tree	Proposed Tree
It needs to arrange the items in the order of their global frequency in the database.	There is no need to change the order of items in the transaction	All the items should be arrange in some predecided canonical order	All the items should be arrange in some predecided canonical order
Overhead increases if database gets update.	No overhead in case of updation.	No overhead in case of updation.	No overhead in case of updation.
It grows rapidly.	It grows rapidly.	It grows rapidly.	It grows less comparatively.
Number of nodes in the tree is high.	Number of nodes in the tree is high.	Number of nodes in the tree is high.	Number of nodes in the tree is less.
No swapping is needed.	It needs swapping of the nodes if the count of a child exceeds the count of the parent.	No swapping is needed.	No swapping is needed.
Frequency count of any node will always be either less than or equal to the count of its ancestors.	Frequency count of any node will always be either less than or equal to the count of its ancestors.	Frequency count of any node will always be either less than or equal to the count of its ancestors.	No constraint on the frequency count.
It follows the concept of the FP-Tree	It doesn't follow the concept of FP-tree.	It doesn't follow the concept of FP-tree.	It doesn't follow the concept of FP-tree.

By considering the above implementation of all four methods on sample database, it is observed that number of nodes required is minimum in our proposed novel tree structure. In this implementation every node has a list of values associated with it. There is a special procedure to read the values. The position of the value has its own significance. We can easily construct the original database from the given tree. Since it needs lesser number of nodes , hence its memory requirement is always less comparatively. Thus it is more efficient as compared to the other methods. Time requirement will also be less as it has to make lesser number of comparisons.

V. CONCLUSION

This paper presents a novel data structure named as *COMVAN tree*. It has been observed that the Apriori and FP tree based methods are not suitable for incremental mining; as they need rescanning of old data at every increment.

There are many advantages of newly proposed COMVAN tree over most of the existing tree based methods proposed in [1,2,3]. One of the main advantages is, once a COMVAN tree is constructed, frequent pattern mining with different support value can be performed without rebuilding the tree. It gives the many advantages of "Build once, Mine many". This COMVAN tree has great advantage if frequent patterns contain some common data items. Further, the proposed tree based data structure also eliminates the need to create separate tree; in case the support vary. It has also become possible to delete any transaction in easiest manner without the need to rescan the database. Second advantage is that, now it has become possible to construct the original table with the help of COMVAN tree and vice versa. Third advantage is that now it is required to scan the original database only once. Finally, the new tree based data structure allows the insertion and deletion of single transaction at a time; this makes it suitable for real time frequent mining. During the analysis of proposed tree based data structure it has been realized that the proposed structure is not only memory efficient but also succeeded in offering high performance.

A. Conflict of Interest:

The Author(s) declare that there is no conflict of interest regarding the publication of this manuscript.

REFERENCES

[1] I. Ezeife and Y. Su. "Mining Incremental Association Rules withGeneralized FP-Tree". Proceedings of the 15th Canadian Conferenceon Artificial Intelligence, May 2002.
[2] C. K. Leung, Q. I. Khan and T. Hoque."CanTree: A Tree Structure forEfficient Incremental Mining of Frequent Patterns", Proceedings ofthe Fifth IEEE International Conference on Data Mining (ICDM'05), 2005.
[3] W. Cheung and O.R. Zaiane , " incremental mining of frequent patterns without candidate generation or support constraint", Proceedings of IDEAS 2003, pp. 111-116.
[4] J. Pei, J. Han, and L.V.S. Lakshmanan. Mining frequent itemsets with convertible constraints.In Proc. ICDE 2001, pp. 433–442.
[5] C.K.-S. Leung, L.V.S. Lakshmanan, and R.T. Ng. Exploiting succinct constraints using FP-trees. SIGKDD Explorations, 4(1), pp. 40– 49, June 2002.

[6] Agrawal, R., Imielinski, T. and Swami, A. “Mining Association Rules between Sets of Items in LargeDatabase”. Proceedings of the ACM SIGMOD conference on management of data, Washington, D.C, May 26-28, 1993.
[7] Agrawal R and Srikant, R., “Fast algorithms for miningassociation rules”, Proceedings of the 1994 Int. Conf. Very LargeData Bases, pp. 487-499, Santiago, Chile, September1994.
[8] Brin, S., Motwani, R., and Silverstein, C. Beyond market baskets: Generalizing association rules to correlations. SIGMOD 26[2], 265-276. 1997.
[9] Antonie, M.-L. and Zaïane, O. R., Text Document Categorization by Term Association , IEEE ICDM'2002, pp 19-26, Maebashi City, Japan, December 9 - 12, 2002
[10] Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M.-C. FreeSpan: Frequent pattern-projected sequential pattern mining. ACM SIGKDD, 2000.
[11] Beil, F., Ester, M., Xu, X., Frequent Term-Based Text Clustering, ACM SIGKDD, 2002
[12] Orlando, S., Palmerini, P., and Perego, R. Enhancing the Apriori Algorithm for Frequent Set Counting. Proceedings of 3rd International Conference on Data Warehousing and Knowledge Discovery. 2001.
[13] J. Pei, J. Han, and R. Mao, “Closet: An Efficient Algorithm forMining Frequent Closed Itemsets,” Proc. SIGMOD Int’l WorkshopData Mining and Knowledge Discovery, May2000.