An Enhanced FP- Growth Approach To Mining Assosciation Rules In Horizontal Partitioned Database

Anjali V Nair[1] **Jeena P Abraham**[2], **Smita C Thomas**[3]
[1, 2, 3] Dept of Computer Science & Engineering
[1, 2, 3] Mount Zion College of Engineering Kadammanitta ,Pathanamthitta ,India

Abstract- *Data mining is the computing process of discovering patterns in large database system. The goal of data mining process is to extract information from a data set and transform it into understandable structure for further use Association rule mining is a rule based machine learning method for discovering interesting relation between variables in large database. The main disadvantages of FP-Growth is that it may not fit in memory and it is very expensive to build. It have complex data structure. To over come this disadvantage, Enhanced-FP Growth Algorithm is used. This works without any complex data structure or prefix tree. Its main strength is its simplicity. There is no need to representation of transactions. Its transactional database is memory resident.*

Keywords- Data mining; Association rule mining; FP-Growth; Enhanced-FP

I. INTRODUCTION

Data mining is the process of sorting large data sets.Mainly for identifying patterns and relationships to solve problems through data analysis. Data mining tools allow enterprises to predict future trends. In data mining, association rules are created by analyzing frequent data through if/then patterns, and using the support and confidence criteria to locate the most important relationships within the data. Association rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository. Association rules are created by analyzing data for frequent if/then patterns and using the criteria support and confidence to identify the most important relationships. Support is an indication of how frequently the items appear in the database. Confidence indicates the number of times the if/then statements have been found to be true. The FP-Growth Algorithm is the way to find frequent item sets without using candidate generations and performance. For so much it uses a divide-and-conquer strategy . The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information. In simple words, this algorithm works as follows: first creating an FP-tree instance to represent frequent items through compressing the input database . After this first step it divides the compressed database into a set of conditional databases, each one associated with one frequent pattern. Finally, each such database is mined separately. Using this strategy, the FP-Growth reduces the search costs looking for short patterns recursively and then concatenating them in the long frequent patterns, offering good selectivity .In large databases, it's not possible to hold the FP tree in main memory. There are two important basic measure for association rules, support(s) and confidence(c).

Support(s) of an association rule is defined as the percentage/fraction of records that contain X => Y to the total number of records in the database. Suppose the support of an item is 0.2%, it means only 0.2 percent of the transaction contain purchasing of this item.

Confidence of an association rule is defined as the percentage/fraction of the number of transactions that X=>Y to the total number of records that contain X. Confidence is a measure of strength of the association rules.

A. Database Partitioning

Centralized database have the capacity to store the entire database into a single table. Mining is used to extract the important data from a collection of data. sometimes we cant store all the data into a single table, in this case we can split the collection among various parties. The data can't directly share to any other party because privacy is an important factor. Individual parties database is local database and it is a partial database because it cant store all the data. various techniques like vertical, horizontal, and hybrid partitions are present. Vertical partitions are prepared in two types, first is normalization and next is row spliting. Horizontal partitioning means it contains complete schema, but information about dissimilar rows.

B. Third-Party Computation

When a database is portioned among multiple players, Each player wants to know the final output of any particular computation, It is satisfied with the help of a trusted third party or without a third party. if we use a third-party means all other players wants to submit their own data to third party and this third party computes the result and broadcast it to all others. So for this one we have to use a big function for computation.x1.........xm are the private inputs of the players. Each player wants to compute the output for a public function ’f’ by using some protocol. While performing a computation the information collected by a player without the use of a third party is lesser. Each player wants to know the local input and global output.

Section II describes about the Literature Survey, Section III describes the Existing System, Section IV describes the Proposed System, Section V Concludes the paper.

II. LITERATURE SURVEY

Several researchers proposed many techniques for explaining the privacy while the computation of associations rules when the database is partitioned horizontally. Each technique is explained as follows.

Shubhra Rana et.al [1] has proposed a solution named Paillier additive homomorphism cryptosystem. Pattern count tree is used for making the data structure. The main advantages of this system is that it is built in single I/O scan. The Hierarchical Homomorphic Encryption scheme is used for providing security.

Tamir Tassa [2] also proposed an idea named Fast Distributed Mining for finding frequent item set. This algorithm is a extended version of Apriority. It deals two algorithms called secure multi party algorithm. This solution provide complicated inequality verification to check set inclusion while providing security.

Bettahally N.Keshavamurthy et.al [3] has proposed a paper deals with Genetic algorithm. It explains how to search frequent item set and generate association rules as globally frequent. For preserving the privacy, they have used the trusted third party with two offset. Use of Genetic algorithm is done because it provides robustness while searching in a large search space.

Xuan Canh Nguyen et.al has proposed Enhanced M.Hussein et al’s Scheme [4] technique for privacy preserving association rule mining in horizontally distributed databases.It is an enhanced version of MHS.It uses initiator and combiner.MFI approach is generated.

Rachit V.Adhvaryu et.al has proposed modified EMHS [5] algorithm. This one has been proposed to improve the efficiency of EMHS.It uses RSA and ELGamal algorithms.

III. EXISTING SYSTEM

The existing system contains a large horizontally partitioned data base with M players. Items in the transactions are encrypted to provide security from unauthorised access .FP growth algorithm is used here to avoid multiple database scan and use secure protocol. using this algorithm to find frequent item set at each partition .for hiding the original identity of data, encode the item set into binary vector. In this each player share their information only to the next party instead of all others. Each player added its own share to previous one. All player except the last one sent their share to first player. First player adds all the data it received .Now in this case, first and last player holds the additive shares of the sum vector. There for, to check for a threshold protocol it must satisfy the inequality securely. Verification of this Consider a large database D with M players such that D=D1 U D2 U......DM.We have to find all association rules which are frequent globally. The secret information about the individual partition should not be disclosed. Rule should also satisfy minimum support and confidence in database. To achieve this goal there are two methods present either use third party or no use of third party. If it use a third party, it will compute result for all players and distribute along all others. Then the players submit it to the third party. there is a need of secure protocol for this. The existing system has proposed two secure multi-party computation, that is threshold function and set inclusion protocol. We try to overcome some of these challenges like large key size, frequent item set generation, use of third party and multiple database scan presented in literature survey using our proposed system.

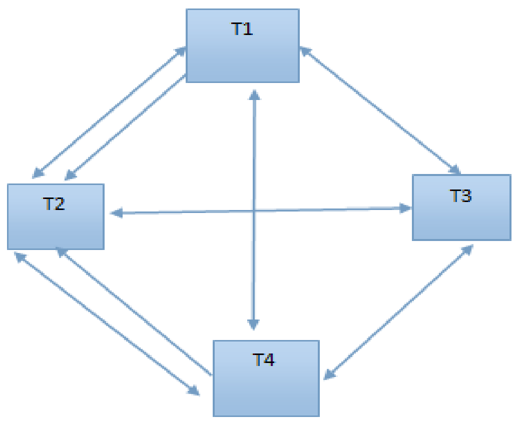

Figure 1. Share Transfer

IV. PROPOSED SYSTEM

In this section we propose a system which tries to cover the challenges presented in Existing system. In the proposed system will provide more security and work without any complex data structure. Its strength is its simplicity.

A. Procedure of Enhanced FP-Growth Algorithm

Enhanced FP-Growth algorithm works without using any prefix tree and complex data structure. It processes the transaction directly so the main advantage of this one is it's simplicity. The pseudo code of this algorithm is depicted on Figure 2.

B. Step by step process of Enhanced FP-Growth algorithm

Step1: In the first scan, The support of items are calculated. The low support count items are discarded and is specified as frequent item set. With respect to their support the items present in the database are sorted in ascending order.Let us consider a horizontal database D and minimum support count is 3.

Transaction database(left), item support(middle), reduced transaction database with items in transactions sorted ascending order with respect to their support.
F and G are the discarded items.

Step2: The transaction database is first converted to transaction list, one list for each item. These list can be sorted into an array and each item contains a pointer to the head.
The items can have a successor pointer and a pointer to the transaction. Using the leading item and index we can inserted the transactions one by one.

Step3: The first list contains the second, seventh ,and eighth transactions with item e, removed. The counter present in the array elements states transactions contains the corresponding item. For finding all the frequent item set we have to traverse it from left to right. Before performing the transaction, first we check all the support count of item set, if it exceeds than minimum support count. The processing of transactions list is as follows: For each list element the leading item of its transaction is retrieved and used as an index. In such type of reassignment, the leading item is also removed from the transaction. In the second array of transaction lists a copy of the list elements is inserted. Now the second array collects the subset of transactions that contain a specific item. This set of transaction lists is then processed recur lively, noting the item associated with the list it was generated from as a common prefix of all frequent item sets found in the recursion. After the recursion the next transaction list is reassigned, copied and processed. In first all items sets Contains a specific item and represents them as a set of transaction list. This set of transaction lists is then processed recur lively, note the item associated with the list is then

```
Function Enhanced-FP (a: array of transaction lists,
    p: set of items,
    s min: int) : int
Var i; k: item;
    s: int;
    n: int;
    b: array of transaction lists;
    t; u: transaction list element;
begin
    n := 0;
    While a is not empty do
        i := last item of a; s := a[i].wgt;
        If s >= smin then
            p := p ∪{i};
            Report p with support s;
            p: = p – {i};
        end;
        If s >= smin then
            b: = array of transaction lists;
            t: = a[i].head;
            While t = nil do
                u: = copy of t; t: = t.succ;
                k: = u.items[0];
                remove k from u.items;
                if u.items is not empty
                then u.succ = b[k].head; b[k].head = u; end;
                b[k].wgt := b[k].wgt +u.wgt;
            end;
            n := n + 1 + Enhanced-FP(b; p; smin);
        end;
        t := a[i].head;
        while t=nil do
            u := t; t := t.succ;
            k := u .items[0]
            remove k from u .items;
            if u .items is not empty
            then u .succ = a[k].head; a[k].head = u; end;
            a[k]. wgt := a[k].wgt +u.wgt;
        end; remove a[i] from a;
    end;
    return n;
end;
```

Figure 2.Pseudo code of Enhanced FP Growth processed. After it the next transaction is performed, reassigned ,copied.

In this we have to compare the performance of Enhanced FP-Growth, and FP growth algorithms. Graph shows the execution time of implementations over support.

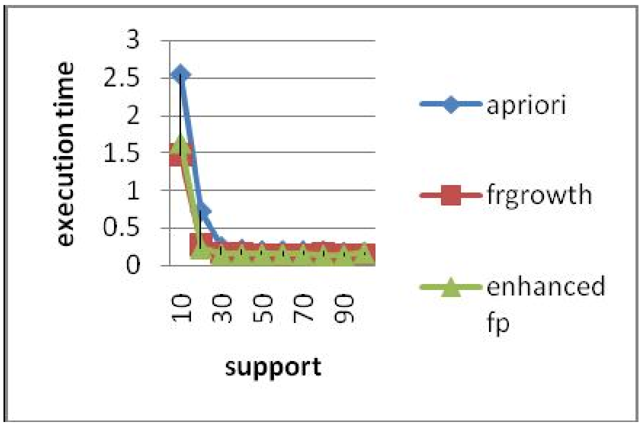

Figure 3. Implementation and Performance Evaluation of different algorithms

According to this graph it will show that, Enhanced FP-Growth is an improvement of FP-Growth. It is mainly designed to eliminate some of the heavy bottle necks in apriori. The algorithm was mainly planned with benefits of map Reduce. It will works with any distributed system focused on any map reduce. FP-Growth solved all the problem using FP tree.

V. CONCLUSION

In this we introduced Enhanced-FP ,which does its work without any complex data structure or prefix tree. Its main strength is its simplicity. There is no need to re-representation of transactions. The Enhanced FP Growth beats FP-Growth by far. It has less memory usage and less runtime. It is more scalable because of its linear running time's based implementations of these three algorithms. By comparing these frequent item set mining algorithms apriori and fp-growth and Enhanced-FP, the strength of Enhanced-FP is analyzed. As the Experimental results show, Enhanced-FP clearly outperforms apriori and FP-Growth. It is faster than apriori and FP-Growth and is not expensive like FP-tree. Its Transactional database is memory resident .As a part of Future scope further work can be done with the help of other secured algorithms in data mining.

VI. ACKNOWLEDGMENT

We would like to thank, first and foremost, Almighty God, without his support this work would not have been possible. We would also like to thank all the faculty members of Mount Zion college of engineering, for their immense support.

REFERENCES

[1] Agrawal R, Imielinski T, Swami A. "Mining Association Rules between Sets of Items in Large Databases." Proc. Conf. on Management of Data, 207- 216.ACM press, New York, NY, USA. June 1993.

[2] Agrawal R, Srikant R. "Fast Algorithms for Mining Association Rules", in Proc.20th Intl. Conf. on very Large Databases (*VLDB'94),*Santiago de Chile,pp.487-99,1994.

[3] Agrawal R.C., Aggarwal C.C., and Prasad V.V., "A Tree Projection Algorithm for Generation of Frequent Item Sets," Parallel and Distributed Computing, pp. 350-371,(2000).

[4] Bodon F. "A fast apriori implementation". In Goethals B. and Zaki M.J.,editors, Proceedings of the IEEE ICDM Workshop on Frequent Item set Mining Implementation (FIMI'03), CEUR Workshop Proceedings Melbourne, Florida, USA, Vol. 90,(2003).

[5] Grahne G. and Zhu J., "Efficiently Using Prefix-Trees in mining Frequent Item sets," Proc. ICDM 2003 Workshop Frequent Item set Mining Implementations, (2003).

[6] Y.Fuand J.Han. "Metal-Rule-guided Mining assosciation rules in relational database.KDOOD's",39-46,Singapore,Dec-1995.

[7] J.Han and Y.Fu."Discovery of multiple level assosciation rules from large databases.VLDB'95,420-431,Zurich,Switzerland.

[8] M.Kamber,J.Han,and J.Y.chiang,"Metarule guided mining of multi-dimensional assosciation rules using data cubes",KDD'97,207-210,Newyork,Callifornia.