AN APPROCH TO FREQUENT ITEMSET MINING USING DIC & MINHASH TECHNIQUE

[1] KRISHNA H. ODEDARA, [2] DR. VIPUL VEKARIYA, [3] PROF. DAXA V. VEKARIYA

[1] Student of M.E. (CE), Noble group of institutions, Junagadh, Gujarat, India
[2] Associate professor, Computer Department, Noble group of institutions, Junagadh, Gujarat, India
[3] Assistant professor, Computer engineering, Noble group of institutions, Junagadh, Gujarat, India

Krishnaodedra123@gmail.com, vp@ngivbt.edu.in, daxa.vekariya@ngivbt.in

ABSTRACT : *Association rule mining is used for finding frequent itemset form the database.There are many efficient algorithm used for finding support of the itemset which denote that itemset is frequent or not. From that algorithm. Ecat algorithm is used intersection for the support counting but it give low efficiency when number of transaction are large. And it only used when itemset is static. In our algorithm considers the DIC(variation of apriori) approach it reduce the number of passes made over the transaction database and dynamically item can be added in dataset. DIC maintains four itemsets dashed circle, dashed box, Solid circle and solid box for calculating the frequent itemset And used hash based technique for reduce the size of candidate itemsets. Besides other efficiency improving methods the DIC give confirmed frequent itemset form transactional dataset.*

Keywords— frequent itemset, transactional database, DIC, MinHash

I. INTRODUCTION

Association rule mining was introduced by Agrawal, and initially used in large-scale transaction data recorded in supermarket to discovering interesting relations between variables in large databases. There are some efficient algorithms uses the downward-closure property of support which guarantees that for a frequent itemset. For example, one of properties used by the Apriori algorithm is that all subsets of a frequent itemset must also be frequent. Finding association rules is the core process of data mining and it is the most popular technique has been studied by many researchers.. It is mining for association rules in database of sales transactions between items which is important field of the research in dataset .Using different algorithm through finding frequent itemset from large transactional dataset.

Frequent itemset mining has wide applications. The research in this field is started many years before but still emerging. This is a part of many data mining techniques like association rule mining, classification, clustering, web mining and correlations. The same technique is applicable to generate frequent sequences also. In general, frequent patterns like tree structures, graphs can be generated using the same principle. There are many applications where the frequent itemset mining is applicable. In short, they can be listed as market-basket analysis, bioinformatics, networks and most in many analyses.

In this paper we uses minhash technique for finding frequent itemset. And using bucket address provide the address space to that frequent itemset. .And using Rehashing technique resolves the collisions that are encountered during various collision resolution techniques used in open addressing starategy.This is done by increasing the size of a hash table, and restoring all of the items into the hash table using the hash function h(k)=k%m where m is the new length of the hash table after increasing it..And also used dynamic counting method for adding item dynamically if required.

II. BACKGROUND

A. Minhash technique

A hash table (hash map) is a data structure used to implement an associative array, a structure that can map keys to values. A hash table uses a hash function to compute an *index* into an array of *buckets* or *slots*, from which the correct value can be found. Hash functions are primarily used in hash tables, to quickly locate a data record given its search key. Hash technique through finding the bucket address and put that itemset on that index.But whenever collisions occur after mapping the frequent item sets then an immediate check for the number of buckets still vacant in the hash table must be done. If it is observed that the hash table is either half –filled or is more than half of the size of the hash table is occupied then it is appropriate to apply rehashing technique using which we can double the size of the hash table thus providing enough buckets for all frequent item sets without any collisions

B. Rehashing technique

Rehashing is a technique used in hash tables to overcome hash collisions, when two different values to be searched for producing the same hash key. It is a popular collision resolution technique used on hash tables. Like linear probing ,it uses one hash value as a starting point and then repeatedly steps forward an interval ,until the desired value is located; an empty location is reached , or the entire table has been searched. .In Linear probing, Quadratic probing and Double hashing, we have to guess the number of elements we need to insert into a hash table.

C. Dynamic itemset counting

This is an alternative to Apriori Itemset Generation. In this itemsets are dynamically added and deleted as transactions are read. It is based on the downward release property in which this calculates the itemsets to different point of time regarding the scan. This algorithm also used to ease the number of database for discovering the frequent itemsets by just counting the new element at any fact of time finished the run time

DIC maintains four sets of itemsets, namely Dashed Circle,Dashed Box, Solid Circle and Solid Box. Itemsets in the"dashed" sets are subjects for support counting while itemsetsin the "solid" sets do not need to be counted. "Circles" contain infrequent itemsets while "boxes" contain frequent itemsets.

III. PROPOSED WORK

In general the structure of the transactional database may be in two different format – Horizontal data format and Vertical data format. In this paper, transactions of database are stored in the vertical format. Vertical data format ,an "Item: TID" format in which "TID" is unique identifier for of a transaction and "Item" is an item in database.

In this paper, We use hash based technique for providing bucket address to frequent mining itemset. But sometime collision occur because size of hash table is less than the required size.so avoid this problem we use Rehashing Based Frequent Item set(RBFI) A new Rehashing technique can be used to increase the size of a hash table, and restoring all of the items into the hash table using the hash function h(k)=k%m where m is the new length of the hash table after.And this paper we also use DIC method for add any no. of transaction in given database when required, which possible using dynamic itemset counting method.

IV. EXAMPLE OF THE PROPOSED WORK

Consider Table 1.Initial Transaction Database.For our convinience, Let us replace these real time items i1,i2,i3,i4,i5 respectively.

Here,take min_support=3

Transaction ID	Itemset
T1	i1,i2,i3,i4
T2	i2,i4
T3	i1,i5
T4	i1,i2,i3
T5	i1,i4
T6	i2,i3
T7	i1,i3
T8	i2,i4
T9	i1,i2
T10	i1,i3

Table : 1 Transactional Database

Itemset	Transaction ID
i1	T1,T3,T4,T5,T7,T9
i2	T1,T2,T4,T6,T8,T9
i3	T1,T4,T6,T7,T10
i4	T1,T2,T5,T8,T10
i5	T2,T3

Table : 2 Vertical form of Transaction Database

The items in the transaction are hashed based on the hash function

h (k) = (order of item k) mod n.

The n value is determined by using the formula **(2m + 1)** where m is the number of items in the database.

Itemset	Transaction ID
{i1,i2}	T1,T4,T9
{i1,i3}	T1,T4,T7
{i1,i4}	T1,T5
{i2,i3}	T1,T4,T6
{i2,i4}	T1,T2,T8
{i3,i4}	T1,T10

Table :3 Vertical form of Transactional Database second level

Itemset	Transaction ID
{i1,i2,i3}	T1,T4
{i1,i2,i4}	T1
{i2,i3,i4}	T1

Table : 4 Vertical form of Transactional Database second level

The item set in the second level from Table.3 are hashed based on the hash function,

h(k)=((order of X)*10+order of Y)mod n.

Here , the item sets are mapped to 1,2,3,1,2,1.Here, there is a collision for {I1,I2}{I2,I3};they are mapped to 1 and {I1,I2}{I3,I4} are mapped to 1 and {I1,I3}{I2,I4} are mapped to 2.

Bucket address	Itemset
0	
1	{i1,i2}{i2,i3}{i3,i4}
2	{i1,i3}{i2,i4}
3	{i1,i4}
4	
5	
6	
7	
8	
9	
10	

Table :5 Bucket address with collision

Rehashing technique is used to overcome this collision. Let h(k) be a hash function that maps element *k* to an integer in [0,j+1] , where

$$j = 2*m+1$$

and *m* is the size of the table.

Here , we increase the size of the hash table by doubling the actual size , so that the resulting hash table size is also a prime number.

Thus the size of the hash table after increasing is

j= (2*m+1),

where m=11 (initial hash table size) .Therefore , *j=23*.Now,we apply the hash function.

***h(k)*=((order of X)*10+order of Y)mod j**

After rehashing the collision is resolved and the 2-itemsets {I1,I2} ,{I1,I3},{I2,I3},{I2,I4},{I3,I4} ,{I1,I4} are mapped to 12 ,13,0,1,11,14 buckets respectively as shown in table :6

In the second level, item sets {I1,I2} ,{I1,I3},{I2,I3},{I2,I4},{I3,I4} whose support counts are greater than or equal to 3 are said to be frequent itemsets.

Bucket address	Itemset
0	{i2,i3}
1	{i2,i4}
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	{i3,i4}
12	{i1,i2}
13	{i1,i3}
14	{i1,i4}
15	
20	

Table :6 After apply rehashing technique avoid collision

The itemsets in the third level are hashed based upon the hash function.

H(k)=((order of X)*100+(order of Y)*10+order of Z)mod j.

Using this hash function the itemsets {I1,I2,I3},{I1,I2,I4},{I2,I3,I4} are mapped to location 8,9 and 4 respectively as shown in the Figure.3.

Bucket address	Itemset
0	
1	
2	
3	
4	{i2,i3,i4}
5	
6	
7	
8	{i1,i2,i3}
9	{i1,i2,i4}
10	
11	
12	
13	
14	
15	
20	

Table:7 Bucket Address for third level transaction

Now we add transactions using dynamic method,

Add Transactions

T11->i3,i4 & T12->i1,i3

Transaction ID	Itemset
T1	i1,i2,i3,i4
T2	i2,i4
T3	i1,i5
T4	i1,i2,i3
T5	i1,i4
T6	i2,i3
T7	i1,i3
T8	i2,i4
T9	i1,i2
T10	i1,i3
T11	i3,i4
T12	i1,i3,i5

Table: 8 After Adding transaction T11,T12

Itemset	Transaction ID
i1	T1,T3,T4,T5,T7,T9,T12
i2	T1,T2,T4,T6,T8,T9
i3	T1,T4,T6,T7,T10,T11,T12
i4	T1,T2,T5,T8,T10,T11
i5	T2,T3,T12

Table :9 vertical form of dtabase after transactions T11,T12

Itemset	Transaction ID
{i1,i2}	T1,T4,T9
{i1,i3}	T1,T4,T7,T12
{i1,i4}	T1,T5
{i1,i5}	T12
{i2,i3}	T1,T4,T6
{i2,i4}	T1,T2,T8
{i2,i5}	T2
{i3,i4}	T1,T10,T11
{i3,i5}	T12
{i4,i5}	T2

Table :10 Vertical form of database in the second level

Itemset	Transaction ID
{i1,i2,i3}	T1,T4
{i1,i2,i4}	T1
{i2,i3,i4}	T1

Table :11 Vertical form of database in the third level

Here we can see that there is no changes in the third level of transaction table after adding transaction T11,T12. So,there is no require to calculate bucket address. Because it not any change in third level database. Somtime it change when adding itemset count is equal or greater than the min support then require to calculate it and at that time whenever collision occur then we can remove collision using rehashing technique which is seen in above method. And put that itemset at appropriate bucket address.

V. CONCLUSION

In this work, we proposed an approach to find frequent itemset using hashing & dynamic method . Here, We presented the frequent itemset finding using hash based

technique .And put that itemset in their bucket address but some collision occur due to one key put at same location.Our method we can overcome this collision problem using rehashing technique through and get the unique bucket address for each frequent mining itemset.And also use dynamic itemset counting method for adding transaction whenever required.For the further improvement ,some other optimization methods can also be used in our framework

REFERENCES

[1] R Agrawal, T Imielinski, and A Swami. Mining association rules
between sets of items in large database[C]. Proceedings of the 1993 ACM-SIGKDD international conference on management of data(SIGMOD'93), New York, NY, USA: ACM, 1993.

[2] M.S.V.K. Pang-Ning Tan, ―*Data mining, in Introduction to datamining"*, Pearson International Edition, 2006.

[3] A Hash based Mining Algorithm for Maximal Frequent Itemsets using Linear Probing‖. Infocomp Journal of Computer Science 2009, Vol.8, No.1, pp.14-19..

[4] M. Krishnamurthy, A. Kannan , R. Deepalakshmi ―Frequent Item set Generation Using Hashing-Quadratic Probing Technique‖ ―European Journal of Scientific Research ISSN 1450-216X Vol.50 No.4 (2011), pp. 523-532.

[5] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiagode Chile, Chile, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann, 1994, pp. 487–499.

[6] "Dynamic itemset counting and implication rules for market basket data," in SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA., J. Peckham, Ed. ACM Press, 1997, pp. 255–264.

[7] J. Han―*Data Mining: Concepts and Techniques 3rdedition*‖, Morgan Kaufmann Publishers, 2013.

[8] M. Holsheimer, M. L. Kersten, H. Mannila, and H.Toivonen, ―A Perspective on Databases and Data Mining‖, KDD 1995: 150-155.

[9] A Hash based Mining Algorithm for Maximal Frequent Itemsets using Linear Probing‖. Infocomp Journal of Computer Science 2009.

[10] M. J. Zaki, ―Scalable Algorithms for Association Mining‖, IEEE Transactions on Knowledge and Data Engineering,s May/June 2000.

[11] M. Krishnamurthy, R. Deepalakshmi ―Frequent Item set Generation Using Hashing-Quadratic Probing Technique‖ ―European Journal of Scientific Research ISSN 1450-216X Vol.50 No.4 (2011), pp. 523-532.

[12] A. Savasere, E. Omiecinski, and S. Navathe, ―An efficient algorithm for mining association rules in large databases‖, 21st VLDB Conference, 1995.