A Hash based Mining Algorithm for Maximal Frequent Item Sets using Double Hashing

NVB Gangadhara Rao
Head of the Department Computer Science and Engineering, G.I.E.T, Rajanagaram, India.
e-mail:dhar.nalamati@gmail.com

Sirisha Aguru,
Department of Information Technology,
S.V.E.C., Pedatadepalli, Tadepalligudem, India
e-mail:sirib4u1@gmail.com

Abstract
Data mining is the use of automated data analysis techniques to uncover previously undetected relationships among data items. In data mining, association rule learning is a popular and well researched method for discovering interesting relations between variables in large databases. Mining frequent patterns is probably one of the most important concepts in data mining. It plays an essential role in many data mining tasks that try to find interesting patterns from databases, such as association rules, correlations, sequences, classifiers and clusters. In this paper, we propose a new association rule mining algorithm called Hash Based Frequent Item sets-Double Hashing (HBFI-DH) in which hashing technology is used to store the database in vertical data format. To avoid hash collision and secondary clustering problem in hashing, double hashing technique is utilized here. The advantages of this new hashing technique are easy to compute the hash function, fast access of data and efficiency. This algorithm provides facilities to avoid unnecessary scans to the database.

Keywords - Collisions; Hashing; vertical hashing; Double Hashing; Frequent item-sets mining.

I. INTRODUCTION

A. Frequent item-sets

Data mining is the process of extracting patterns from data. It is becoming as an increasingly important tool to transform these data into information. Frequent item set 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. In general, frequent patterns like tree structures, graphs can be generated using the same principle. Agarwal et. al [4] is the first person to state this problem. Later many algorithms were introduced to generate frequent item sets. The Apriori algorithm is the basis for all rule mining. The database in Apriori has to be repeatedly scanned and large number of candidates has to be generated which is a major limitation of this Apriori algorithm. Let \( \{I=I_1, I_2, I_3, I_m\} \) be a set of items. Let \( D \) be the transactional database where each transaction \( T \) is a set if items such that \( T \) is a subset of \( I \). If the support count of an item set \( I \) satisfy the minimum support threshold, then the item set \( I \) is a frequent item set.

B. Proposed Work

In general the structure of the transactional database may be in two different ways – Horizontal data format and Vertical data format. In this paper, transactions of database are stored in the vertical format.

In order to reduce the memory space and to implement fast access, we introduce a new algorithm based on hashing types. The main advantage of this algorithm is to avoid collision and secondary clustering problem. It overcomes the collision problem by double hashing. The main advantage of this algorithm is that, it is scalable with all types of databases regardless of their sizes. However, it is efficient since it requires less memory and time to generate frequent item sets. This proposed algorithm occupies less space.

C. Paper Organization

The remainder of the paper is organized as follows: Section 2 explain about Hashing and some related works. Section 3, explains our Hash Based Frequent Item sets-Double Hashing (HBFI-DH) algorithm. Section 4, provides the performance of these algorithms by showing the experimental results graphically. Section 5 gives the conclusion.
II. HASHING

A hash table or hash map is a data structure that uses a hash function to efficiently map certain identifiers or keys (e.g., person names) to associated values (e.g., their telephone numbers). The hash function is used to transform the key into the index (the hash) of an array element (the slot or bucket) where the corresponding value is to be sought. Collisions are practically unavoidable when hashing a random subset of a large set of possible keys.

A. Related Works

There are many works in the literature that discuss about Association rules, hash based technique and Frequent Item sets. The Association Rule mining raised by R. Agarwal is an important research in data mining field. His Apriori algorithm can discover meaningful item sets and build association rules. However, a large number of candidate sets are generated and the database needs repeated scanning. In order to reduce the database scanning various studies were undergone. Further studies in data mining have presented many efficient algorithms for discovering association rules. The discovery of association rules has been discussed in the past using two steps namely (1) Finding out all the frequent item sets which are greater than or equal to user-specified minimum support threshold (2) generating association rules from frequent item sets[2]. The transaction-marked DHP algorithm (TMDHP) to mining frequent item sets in pervasive computing. Each element of the item sets and the transactions ID will be stored together in the hash-table. We need to access database once and avoids producing a candidate item set [3]. Many algorithms focus on how to find the frequent item sets quickly such as the Pincer-Search [8], FP-growth[5], LPMiner[6], ICI[6] and R. Agrawal and R. Srikanth [9]. Among those algorithms, Pincer-Search combines both the bottom-up and top-down directions to reduce the number of candidate sets, and then increases the efficiency. In addition, because FP-growth uses the structure of FP-tree to store the items of database, the algorithm only needs to scan the database two times task without generating any candidate sets in mining frequent item sets. Thus FP-growth can quickly finish the mining task. The HBMFI-LP[11] method uses hashing technology to store the database in vertical data format. To avoid hash collisions, linear probing technique is used. The HBMFI-QP[13] method also uses hashing technology to store the database in vertical data format. To avoid hash collision, that occur during linear probing, quadratic probing technique is used. It generates the exact set of maximal frequent item sets directly and removes the subset that is not frequent, which based on the classical Apriori[12] instead of checking whether these candidates are frequent item sets after generating new candidates.

Comparing with the previous work our algorithm reduces these problems and the performance is considerably increased. Hence, the scanning cost and computational cost is also reduced by using our new proposed algorithm.

III. PROPOSED ALGORITHM

In order to reduce the space constraint as well as to address scalability, we introduce a new algorithm called Improved HBFI-DH Algorithm to perform mining in legacy database which is basically different from all the previous algorithms.

A. HBFI-DH Algorithm

Input: D, a database of transactions where all are represented as vertical hash table.
Process logic: Finding the frequent item sets.
Output: Generating the frequent item sets.

1. Get the minimum support, min_sup;
2. begin
3. m=0;
4. k=0;
5. Generate the new database in (Items, Tidset) format for all Items
6. Increment m;
7. n=2*m+1;
8. D_k = D;
9. D_k^0
10. begin
11. Make a hash table of size n.
12. Map items on to the buckets if collision occurs then use double hashing technique. Create a linked list for the kth level to maintain the transaction from the database D_k.
13. for all Items I \in D do
14. begin
15. Generate a subset of items.
16. end.
17. Find common transaction between the subsets in the kth level.
18. Eliminate the subset <= min_sup.
19. D_k = Items >= min_sup.
20. Increment k.
21. End until frequent item set is found.
In this paper, we propose a new HBFI-DH algorithm for vertical data format from the transactional database. In this format, the data is represented as Tid set and item. To avoid collision and Secondary clustering problem double hashing technique is used. Secondary clustering occurs with quadratic probing because if two items have the same initial hash value, their entire probe sequences will be the same. The probe sequences generated by Quadratic probing are always based on the bin positions but not on the item. Thus for every item for which a bin is to be searched always follows the same sequence and if there are any bins empty other than these bins in the sequence, then they cannot be occupied by this item. Instead of searching through the same sequence every time as done, the problem of secondary clustering can be avoided by computing the hash function twice.

**TABLE-1 : INITIAL TRANSACTION DATABASE**

<table>
<thead>
<tr>
<th>TID</th>
<th>ITEM SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>I1,I2,I3,I4</td>
</tr>
<tr>
<td>T2</td>
<td>I2,I4</td>
</tr>
<tr>
<td>T3</td>
<td>I1,I5</td>
</tr>
<tr>
<td>T4</td>
<td>I1,I2,I3</td>
</tr>
<tr>
<td>T5</td>
<td>I1,I4</td>
</tr>
<tr>
<td>T6</td>
<td>I2,I3</td>
</tr>
<tr>
<td>T7</td>
<td>I1,I3</td>
</tr>
<tr>
<td>T8</td>
<td>I2,I4</td>
</tr>
<tr>
<td>T9</td>
<td>I1,I2</td>
</tr>
<tr>
<td>T10</td>
<td>I3,I4</td>
</tr>
</tbody>
</table>

By taking the minimum support=3, the following table table-2 is obtained

**TABLE-2: VERTICAL FORMAT OF TRANSACTIONAL DATABASE**

<table>
<thead>
<tr>
<th>ITEM SET</th>
<th>TID</th>
</tr>
</thead>
<tbody>
<tr>
<td>{I1, I2}</td>
<td>T1, T4, T9</td>
</tr>
<tr>
<td>{I1, I3}</td>
<td>T1, T4, T7</td>
</tr>
<tr>
<td>{I1, I4}</td>
<td>T1, T5</td>
</tr>
<tr>
<td>{I2, I3}</td>
<td>T1, T4, T6</td>
</tr>
<tr>
<td>{I2, I4}</td>
<td>T1, T2, T8</td>
</tr>
<tr>
<td>{I3, I4}</td>
<td>T1, T10</td>
</tr>
</tbody>
</table>

Now, the item set in the second level are hashed based on the hash function, \( h(k) = (\text{order of X})*10 + \text{order of Y}) \mod n. Here, the item sets are mapped to 1, 2, and 3,1,2,1. Here, there is a collision for \{I1, I2\} {I2, I3} are mapped to 1 and \{I1, I2\} {I3, I4} are mapped to 2. Double Hashing technique is used to overcome collision.

**TABLE-3: VERTICAL FORMAT OF THE TRANSACTIONAL DATABASE IN THE SECOND LEVEL**

<table>
<thead>
<tr>
<th>ITEM SET</th>
<th>TID</th>
</tr>
</thead>
<tbody>
<tr>
<td>{I1, I2}</td>
<td>T1, T4, T9</td>
</tr>
<tr>
<td>{I1, I3}</td>
<td>T1, T4, T7</td>
</tr>
<tr>
<td>{I1, I4}</td>
<td>T1, T5</td>
</tr>
<tr>
<td>{I2, I3}</td>
<td>T1, T4, T6</td>
</tr>
<tr>
<td>{I2, I4}</td>
<td>T1, T2, T8</td>
</tr>
<tr>
<td>{I3, I4}</td>
<td>T1, T10</td>
</tr>
</tbody>
</table>

Let \(h(k)\) be a hash function that maps an element \(k\) to an integer in \([0, m-1]\), where \(m\) is the size. Let \(h(k, i)\) be the \(i\)th probe position for a value \(k\) be given by the function \(h(k, i) = h(k) + i \cdot h_2(k) \mod m\). That is here we use a second hash function to obtain the next slot, where \(h(k)\) denotes the current “hash-value” and \(h_2(k)\) is obtained from the equation \(h_2(k) = \left\lfloor \frac{R - (k \mod R)}{2} \right\rfloor\).
$R$ is a prime number that should be less than $m$.

The probing sequence will now be:

$$h(k), h(k) + h_2(k), h(k) + 2h_2(k), h(k) + 3h_2(k) \ldots$$

Fig. 1.: Hash table including links for the transactional database in the first level.

```
0   1   2   3   4   5   6   7   8   9   10
I1  I2  I3  I4  I5  
I6  I7  I8  I9  T1  
T2  T3  T4  T5  T6  
T7  T8  T9  T10 T11
```

Then using Double-Hashing technique, \{I2, I3\} is mapped to location 7, \{I3, I4\} is mapped to location 8, \{I2, I4\} is mapped to location 6. Double Hashing thus takes the original hash value and goes on adding the successive second hash functions to the starting value.

Initially, the value of $i$ is 1 and by using this value if collision again exists then the value of $i$ is incremented by until a new location is found. It provides good memory caching because it preserves the locality of reference. It better avoids the clustering problem that can occur with linear probing and quadratic probing. The choice of $h_2$ is important.

(a) It should never evaluate to zero. (b) Must make sure that all cells can be probed. The hash table for the second level is shown in Figure 2.

Fig. 2: Hash table including links for the transactional database in the second level.

```
0   1   2   3   4   5   6   7   8   9   10
I1  I2  I3  I4  I5  
I6  I7  T1  T2  T3  
T4  T5  T6  T7  T8  
T9  T10 T11 T12 T13
```

In the second level, items \{I1, I2\}, \{I1, I3\}, \{I2, I4\}, \{I2, I3\} are frequent item sets which can be observed from TABLE-3 and Figure-2 i.e. there support counts are greater than or equal to 3. Now, total frequent item sets are \{I1, I2, I3\}, \{I1, I2\}, \{I1, I3\}, \{I2, I3\}, \{I2, I4\}. From these, maximally frequent item sets are \{I1, I2\}, \{I1, I3\}, \{I2, I3\}, \{I2, I4\}.

3-itemsets are generated from the maximal frequent item sets of second level. They are shown in TABLE-4. The item sets in the second level are hashed based on the hash function, \( h(k) = \text{floor} \left( (\text{order of } X) \times 100 + (\text{order of } Y) \times 10 + \text{order of } Z \right) \mod n \). Using the hash function, item sets \{I1, I2, I3\}, \{I1, I2, I4\}, \{I2, I3, I4\} are mapped to locations 2, 3, 3 respectively.

<table>
<thead>
<tr>
<th>ITEM SET</th>
<th>TI</th>
</tr>
</thead>
<tbody>
<tr>
<td>{I1, I2, I3}</td>
<td>T1,</td>
</tr>
<tr>
<td>{I1, I2, I4}</td>
<td>T4</td>
</tr>
<tr>
<td>{I2, I3, I4}</td>
<td>T1</td>
</tr>
</tbody>
</table>

The proposed algorithm (HBFI-DH) performs better because FI is calculated in a simplest way. The structure of transactional database is vertical data format. This makes easy to perform several tasks. In this format, support also need not be calculated separately. In this case, support is directly given by the number of transactions in the tidlist of each FI or it can be obtained from the count value in the header node of the corresponding linked list. It is about 2 to 3 times faster than other hash based technique. It quickly finds an empty location in the hash table to map the items. The HBFI-DH performs better with large number of transactions and long item sets. Here, this algorithm doesn’t require performing separate pruning. Hash data structure can be maintained to store the database.

Here in Table- 4, again there are collisions at \{I1,I2,I4\} and \{I2,I3,I4\}. Then using Double hashing technique \{I2,I3,I4\} can be mapped to location 7. The hash table for third level is in Figure- 3.

IV. EXPERIMENTAL RESULTS

Fig.4 shows a time comparison between Apriori algorithm and HBFI-DH algorithms for various values of thresholds. From the diagram it can be seen that the time taken for HBFI-DH is considerably reduced. In this method the time taken to hash items in to vertical hash table is comparatively very low. For various support counts the time taken to find a frequent item set is less compared with Apriori.

Fig.3: Hash table including links for the transactional database in the second level

<table>
<thead>
<tr>
<th>1 2 3 4 5 6 7 8 9 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1, I2, I3</td>
</tr>
<tr>
<td>I1, I2, I4</td>
</tr>
<tr>
<td>I2, I3, I4</td>
</tr>
</tbody>
</table>
V. CONCLUSION

In this paper, an effective algorithm for the initial candidate set generation has been proposed. Our experimental results demonstrate that it is better than Apriori and HBFI-QP.

HBFI-QP is a hash based algorithm and it is very effective for the generation of candidate item sets and it eliminates the items which are not needed for the generation of frequent item sets before the generation of candidate 2-item sets. The algorithm works well but it suffers from secondary clustering problem.

We presented HBFI, an algorithm for finding frequent item sets. Our experimental results demonstrate that HBFI-DH is better than Apriori and other hash based methods because it efficiently maps the item sets in the hash table and it also avoids the primary clustering problem and secondary clustering. The vertical data format representation of the database leads to the easy manipulations on hash data structure.

HBFI-DH uses all the bins and hence the phenomenon of secondary clustering will not occur with double hashing.

REFERENCES

[1] Mahima Singh, Dr. Deepak Garg, “Choosing Best Hashing Strategies and Hash Functions”, International Advance computing conference (IACC 09), March 2009 at Thapar University, Patiala


[5] J. Han, J. Pei, and Y. Yin, “Mining Frequent Patterns Without Candidate Generation,” ACM SIGMOD Conference, 2000, pp. 1-12.


