Frequent Pattern Mining using Double Hash Technique
Ms. Priyanka D. Mali1 Ms. Shruti B. Yagnik2

1,2 P.G Student
1,2 Department of Computer Engineering
1,2 L.J. College of Engineering & Technology, Ahmedabad, India

Abstract— The core concept of data mining is finding frequent pattern and generate association rule based on support and confidence. There are many techniques for mining frequent patterns like apriori, FP growth algorithm, ECLAT, sampling algorithm, partitioning algorithm, H-mine algorithm, direct hash algorithm (DHA) which use hashing technique and double hash technique to find frequent itemset and store data in array structure. Limitation of hashing technique is that hash collision occurs. Direct hash generates primary clusters. To avoid collision open addressing is used. Quadratic probing generates secondary clustering. Both primary and secondary clustering is removed by double hash technique. In this paper method is suggested to improve insertion, deletion and searching in double hash table. 

Key words: Apriori, Hash, Frequent pattern mining, Double hash technique, Support

I. INTRODUCTION

Data mining can be defined as knowledge discovery from data. The KDD process of data mining involves different steps like data cleaning, data integration, data selection, data transformation, mining of data, pattern evaluation and knowledge presentation identify frequent pattern then it will be helpful for market basket analysis, understanding disease behavior and predicting the market previously. Suppose the stoke of one store is analyzed to mine frequent pattern and they found milk and bread is frequently purchased together so the shopper can give discount on purchase of these items and can put them together to increase the sell. It will also be easy for customer to purchase items from same place. Similarly online database query from user is mined and frequent pattern is generated so we get idea about what kind of information users want to surf.

There are many data mining techniques like association rule mining, clustering, classification, outlier detection etc. Various application of data mining techniques are market analysis, medical analysis, business, bioinformatics and other areas which are beneficial for human’s social and commercial activity like fraud detection, customer relationship management, ecommerce systems. Frequent pattern mining is useful to generate association rule. The bank database, different organizations database, web data, text data has been processed to generate frequent patterns and getting different interesting patterns for analysis which is beneficial in decision making purpose.

The frequency of some transaction in given database is determined by its support. The support is calculated by ratio of frequency and database size. The pattern is called frequent if its support is greater or equal to minimum support threshold value. The basic frequent pattern mining algorithm is apriori. The apriori algorithm generates candidate sets and uses the minimum support threshold value to find the interesting frequent pattern .Here the term interesting may differ from user to user. Some frequent pattern may be important for specific user while the same pattern is not useful to other user.

II. RELATED WORK

A. Load Factor:
The presentation of collision resolution method does not depend openly on the number n of stored entries but also dependent relative on the table’s load factor the load factor is the ration between n and size s of its bucket array. The standard cost of lookup through a good quality hash function, is practically constant as the load factor increases from 0 up to 0.7 or so. Further than these points the likelihood of collision and their cost as well for handling the both increase [6]

B. Rehashing:
Like Array-Lists, we have to guess the number of elements we need to insert into a hash table. Whatever our collision policy is, the hash table becomes inefficient when load factor is too high. To alleviate load, rehash.

C. When to rehash?:
For quadratic probing, insert may fail if load > 1/2

1. Rehash as soon as load > 1/2
2. Or, can rehash only when insert fails
3. Heuristically choose a load factor threshold
4. Rehash when threshold breached

Hashing mainly includes four concepts:

1. Hash table
2. Hash function
3. Collision
4. Collision removal techniques

This technique use hash uses hash table structure which will prune the candidate set of infrequent items in early stage. Thus the database scan will be reduced and we get better efficiency than apriori algorithm. It stores the candidate items in bucket form and calculate bucket count. If the bucket count is less than minimum support then it will remove that bucket. Hashing is used in huge quantity of data.

The various Hash table operations are as below:

1. Initialization
2. Insertion
3. Retrieval
4. Deletion

Hash function is:
H(k) = ((order of item X) * 10 + order of item Y) mod n \[2\]

(2.1)

Where n=2m+1

M=table size

There are two techniques to remove hash collision:

- Chaining
- Open addressing:

Which include 3-types as follow:

1) Linear probing:
The interval of probe is fixed.

    Limitation: Linear probing is that it generates primary clusters.

2) Quadratic probing:
Interval between probe are increased using addition of successive output.in this quadratic polynomial is used.

    Limitation: Maximum half of the table can be used as substitute location to resolve collisions. Once the table gets more than half full and hard to fill unfilled location spot. Second disadvantage is that it generates secondary clustering.

D. Double hashing:
Interval between probe are decided using hash function.

As the load factor approaches zero, the size of hash table increases with little improvement in the search cost and memory is wasted.

This graph compares the average number of cache misses required to lookup in tables with chaining and linear probing. as the table passes the 80%-full mark, linear probing performance drastically degrades.

![Fig. 1: Comparison of chaining and linear probing][1]

III. APPROACH OVERVIEW

Double hashing is useful for collision resolutions. Starting point it uses hash value and key and then go through using different probing technique until desired value is placed in table. The insertion in hash sequence is done linearly probing or quadratic probing.

Here h1 and h2 are hash functions k is value and j is associated location m is hash table size. Various hash functions are:

\[ h(k,j)=[h1(k)+i*h2(k)]mod m \]

(3.1)

Probing equation for ith probe:

\[ i\text{-th probe}=[h(k)+g(k)]mod m \]

(3.2)

Where m=table size

h(k)=k mod m

\[ g(k)=1+(k \text{ mod (m-1)}) \]

(3.3)

(3.4)

Existing algorithm\[^{10}\]:

Input: transactions from hash table

Output: frequent pattern

1) start
2) i=0, j=0
3) min_support is given
4) generate a database like (itemset,tidset)
5) for all I belong to D
6) increment m
7) n=2*m+1
8) Dk=D
9) Begin
10) Make a hash table of n size
11) map items to bucket
12) if collision occurs use double hashing technique
13) create a linked list to maintain the transactions
14) for all I belongs to Dk
15) do
16) begin
17) for all items I generate subset
18) end
19) find common transaction between subset
20) eliminate the subset if less than min_support
21) increment k
22) end until frequent itemset found

IV. PROPOSED WORK

- Step-1: Start
- Step-2: Simply scan the given transactional database to make table of items with item count and their corresponding transactions.
- Step-3: Generate table L1
- Step-4: This is similar to apriori join step. For ith level combine items to generate all possible ith level transaction in table Ck by using table Lk-1. Then frequency count is discovered for each combinations and generate linked list structure and allocate items in structure.
- Step-5: Apply hash function for each item in Ck
- Step-6: If collision not occurs at point p
- Go to step 4
- Step-7: If collision occurs
- Apply second hash function at position p to move existing value on empty slot in hash table.
- Step-8: Check the new position in hash table is empty?
- If empty:
- Put the value in that place
- Else if it a collision so go to step 6
- Step-9: Place new item in hash table
- Step-10: End the process when frequent itemset is found.

All rights reserved by www.ijsrd.com

[1]: www.ijsrd.com

[Fig. 1: Comparison of chaining and linear probing][7]

---

[1]: www.ijsrd.com

[7]: www.ijsrd.com

[3]: www.ijsrd.com

[5]: www.ijsrd.com

[6]: www.ijsrd.com

[8]: www.ijsrd.com

[9]: www.ijsrd.com

[10]: www.ijsrd.com

[11]: www.ijsrd.com

[12]: www.ijsrd.com

[13]: www.ijsrd.com

[14]: www.ijsrd.com

[15]: www.ijsrd.com

[16]: www.ijsrd.com

[17]: www.ijsrd.com

[18]: www.ijsrd.com

[19]: www.ijsrd.com

[20]: www.ijsrd.com

[21]: www.ijsrd.com

[22]: www.ijsrd.com

[23]: www.ijsrd.com

[24]: www.ijsrd.com

[25]: www.ijsrd.com

[26]: www.ijsrd.com

[27]: www.ijsrd.com

[28]: www.ijsrd.com

[29]: www.ijsrd.com

[30]: www.ijsrd.com

[31]: www.ijsrd.com

[32]: www.ijsrd.com

[33]: www.ijsrd.com

[34]: www.ijsrd.com

[35]: www.ijsrd.com

[36]: www.ijsrd.com

[37]: www.ijsrd.com

[38]: www.ijsrd.com

[39]: www.ijsrd.com

[40]: www.ijsrd.com

[41]: www.ijsrd.com

[42]: www.ijsrd.com

[43]: www.ijsrd.com

[44]: www.ijsrd.com

[45]: www.ijsrd.com

[46]: www.ijsrd.com

[47]: www.ijsrd.com

[48]: www.ijsrd.com

[49]: www.ijsrd.com

[50]: www.ijsrd.com

[51]: www.ijsrd.com

[52]: www.ijsrd.com

[53]: www.ijsrd.com

[54]: www.ijsrd.com

[55]: www.ijsrd.com

[56]: www.ijsrd.com

[57]: www.ijsrd.com

[58]: www.ijsrd.com

[59]: www.ijsrd.com

[60]: www.ijsrd.com

[61]: www.ijsrd.com

[62]: www.ijsrd.com

[63]: www.ijsrd.com

[64]: www.ijsrd.com

[65]: www.ijsrd.com

[66]: www.ijsrd.com

[67]: www.ijsrd.com

[68]: www.ijsrd.com

[69]: www.ijsrd.com

[70]: www.ijsrd.com

[71]: www.ijsrd.com

[72]: www.ijsrd.com

[73]: www.ijsrd.com

[74]: www.ijsrd.com

[75]: www.ijsrd.com

[76]: www.ijsrd.com

[77]: www.ijsrd.com

[78]: www.ijsrd.com

[79]: www.ijsrd.com

[80]: www.ijsrd.com

[81]: www.ijsrd.com

[82]: www.ijsrd.com

[83]: www.ijsrd.com

[84]: www.ijsrd.com

[85]: www.ijsrd.com

[86]: www.ijsrd.com

[87]: www.ijsrd.com

[88]: www.ijsrd.com

[89]: www.ijsrd.com

[90]: www.ijsrd.com

[91]: www.ijsrd.com

[92]: www.ijsrd.com

[93]: www.ijsrd.com

[94]: www.ijsrd.com

[95]: www.ijsrd.com

[96]: www.ijsrd.com

[97]: www.ijsrd.com

[98]: www.ijsrd.com

[99]: www.ijsrd.com

[100]: www.ijsrd.com
Here main two modifications are done. First is the second hash function is made quadratic instead of linear. After modification the second function will be like,

\[ h(k,i)=h(k)+(i^2)*hp(k) \mod m \quad (4.1) \]

In equation 4.1 hp is probing function

And if there any collision occurs the insertion of new item should be done as per its hash function value, for the existing value in array calculate the position by second hash function which is quadratically increasing. Increase the value of I in equation 4.1 accordingly which indicates number of times the collision occurred with the same value.

V. RESULTS AND ANALYSIS

![Graph showing comparison of old and improved double hash techniques for frequent itemset mining](image)

Fig. 2: comparison graph of old technique and improved double hash technique for frequent itemset mining

In order to appraise the performance of the Improved Double Hash algorithm, we conducted an experiment using the old double hash technique and the Improved algorithm. The algorithms can be implemented in Netbeans or Eclipse and tested on a Windows 7 platform.

- The test database is transactional. Here can consider different parameters like size of transactional file, Support (user threshold value) etc.
- \( |T| \) is the number of transactions or rows in given file, and \( |I| \) is total unique items of transaction.
- The experimental results for different numbers of minimum supports have shown in the graph. The results show that the performance of the Improved Double hash algorithm is much better than that of the old algorithm.
- The better the performance efficiency of double hash algorithm is, the smaller the minimum support is. This is because the smaller the minimum support, the more candidate itemsets the algorithm has to determine, and also the join and pruning processes take more time to execute.
- By considering these different parameters here can give different input transactional file. Here size of file may vary. For the same dataset we run both the algorithm and calculate the time.
- From graph can see here time required for improved algorithm is less than old algorithm for different threshold values.

VI. CONCLUSION AND FUTURE WORK

A. Conclusion:

Proposed Model in this thesis may solve some of the problem of the existing double hashing technique. Insertion, Deletion and Searching may become more efficient by reducing probing sequence in hash table. It can deal with primary and secondary clustering problem. Proposed system reduce time complexity.

B. Future Extension:

The algorithm should be proposed to efficiently work for incremental database also. The problem should be overcomes to give efficient result even when the support changes.

REFERENCES

[10] P. Asha, Dr. T. T. Jebarajan, Varun Cyriac Thomas “Efficient mining of Frequent Item sets and Association


