A Novel Approach for Caching Item Sets in Database Systems Using Similarity Graph and Page Rate Algorithm

Azeem Mohammad, Ali Syed and Mahdi Bazarganigilani

Charles Sturt University, School of Business and Economics,
a.mohammad@studygroup.com
asyed@studygroup.com
mahdi62b@yahoo.com

***Abstract*:** In this paper, we present a novel approach for caching large data sets. We make a similarity graph and rank the items according to their relationships and importances. Finally we compare our method with Least Recently Used (LRU) and Least Frequently Used (LFU) Algorithms. Results show great performance of our approach.

***Keywords*-component;** Caching Algorithm; Repetitive Items Sets ;Page Rate Algorithm; Items Sets Graph.

1. Introduction

Caching important items in database systems is very important issue in large datasets. Caching algorithm can be more intelligent toward importance and effectiveness of items. Moreover, this enhancement improves database reliability and decreases the over head of repetitive accesses to slow storages. Discovering the most important repetitive items plays a major role in our caching algorithm.

There are many algorithms to discover the repetitive items. Apriori-category algorithm and FP tree is two important ones. Apriori-category algorithm is based on candid items. Moreover, it needs too many traverse of the database. On the other hand, FP tree algorithm doe not need keeping many candid items and it just traverses the database twice. While the FP tree algorithm shows great performance in comparison to Apriori-category algorithm, it is inefficient when the repetitive item sets are too much and the support threshold of repetitive items is low [1].

The demerits of previous algorithms show a need for a more efficient and faster algorithm. In this paper, we present a dynamic algorithm to discover the most frequent items according to the similarity graph and page rate algorithm. Our algorithm needs two traverses for constructing the graph items and ranking them. Like FP Tree algorithm [3], we don't need any candid items and we don't consider any threshold, since we don't restrict the number of caching items.

In the following sections, we describe the constructing of Items Graph [2] and Page Rate algorithm [4]. This algorithm engages the relation among pages in addition to their frequencies. This leads to improvement in accuracy of the prediction pages. Finally we perform our experiments and compare our results.

2. Repetitive Items Sets Graph Construction

Items Sets Graph is an undirected graph. The vertexes are sets of $V = \{V_1, V_2, ..., V_n\}$. Every vertex has two properties. The name, which denotes to the name of the item. Moreover, the number, which counts the number of occurrences of items in database transactions. E denotes to the edges of the graph. $E < i, j >$ shows that two items have been repeated in a transaction. The number represents the number of repetitions.

We traverse the database transactions for constructing the graph. Every transaction is consisting of: $T = \{TID, itemsets\}$. TID is transaction ID and $itemsets$ are repetitive items. If two items A, B occurs in one transaction, it will have one edge in similarity graph. Moreover, the number shows the number of common repetitions in other transactions.

Suppose, we have ten transactions as shown in the following table.

Table 1. Repetitive items list

Transaction Number	Items List
T1	I1,I2,I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4

T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3
T10	I1,I4

The following table shows the Items and their supporting transactions.

Table 2. Items and Supporting transactions

Items	Transaction
I1	T4, T5, T7, T8, T9,T10
I2	T1, T2, T3, T4, T6, T8, T9
I3	T3, T5, T6, T7, T8, T9
I4	T2, T4,T10
I5	T1, T8

The following Figure shows the result of our sample.

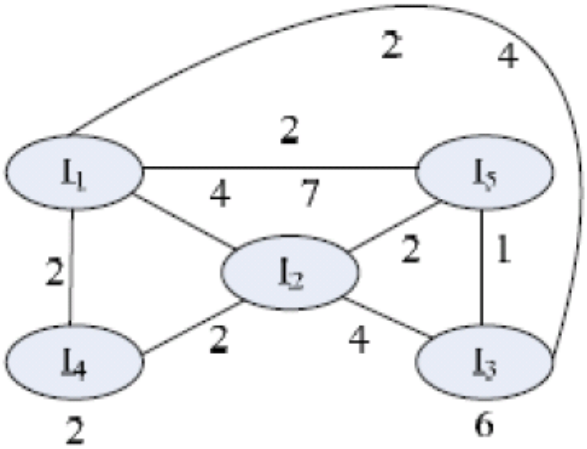

Figure 1. Item Sets Similarity Graph

3. Page Rate Algorithm

After, finding the relations and importance of items in our transactions, we rank the pages for better caching algorithm. The algorithm we use to rank pages affects highly on the caching efficiency. Therefore, we need to engage the relationships among items in addition to importance and frequencies of them. We use Page Rate algorithm as described in [4]. Ranking of items are based on the following
formula:

$$Rank(I_k) = \eta * Rank(I_k) + \sum_{i=1, i\neq k}^{n} Rank(I_k) * L(i,k)$$

η is the factor of effectiveness of number of transactions which is high in our implementations. $L(i,k)$ is the edge of I_i to I_k in the graph. This ranking performed recursively to reach a predefined accuracy. We select best ranked items according to our caching size.

4. Efficiency Analysis of the algorithm

For evaluating the cost of our algorithm, we consider time and place complexity. The amount of place our algorithm occupies is for saving the graph which is the number of transactions n and the number of edges k .this is much more efficient than Apriori-category algorithm. Moreover, considering the time complexity, it takes $O(n)$ for constructing the graph and $O(n*k)$ for ranking the items. k is the number of neighbor vertexes. Therefore, this is much more efficient than Apriori-category algorithm and its similar methods.

5. Experimental Results

In this section, we evaluate our caching algorithm We use data base of MovieLens [5]. Movie Lens is a web system created in 2003. This site has 4000 active users and 6000 products. We have selected 10000 records of them with 1700 products and put 8000 of them for training set and 2000 for test data set.
We use Hit rate parameter as follow:

$$HitRate = \frac{\sum_{i} Item_i \; i \in testset \quad and \quad i \in cachingset}{\sum_{i} Item_i \; i \in testset}$$

We use Least frequently method and Least recently method which discard the least frequent and recent items. Then, we compare them with our proposed methods. As results show, with increasing number of caching items, the hit rate is improved. While, in small caching sizes, our proposed method is more efficient and accurate.
The following table shows our comparisons.

Table 3. Items and Supporting transactions

Caching Size	Hit Rate (%)		
	Our proposed Method	LRU	LFU
100	0.295128292	0.160224	0.284955
200	0.466203233	0.289251	0.453261
300	0.59494744	0.433876	0.585679
400	0.685995253	0.548774	0.683113
500	0.762461851	0.646038	0.749294

600	0.822086583	0.714197	0.808975
700	0.86605629	0.765457	0.857635
800	0.897705437	0.829547	0.894201
900	0.926246185	0.877247	0.924099
1000	0.946535549	0.921612	0.943766
1100	0.959534305	0.946253	0.959195
1200	0.971133311	0.960269	0.970046
1300	0.97903244	0.975528	0.976659
1400	0.984684074	0.983893	0.984401
1500	0.988527184	0.987849	0.988979
1600	0.991013903	0.991127	0.991579
1700	1	1	1

The following Figure illustrates the comparison.

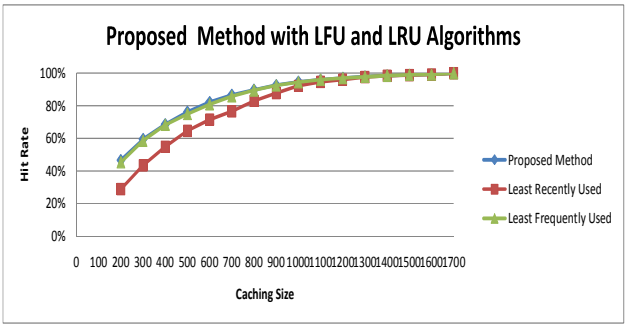

Figure2 . Proposed method with LRU and LFU caching Algorithms.

Conclusions

This paper introduced a new method for caching repetitive items in large databases. We used similarity graph and our Page rate algorithm to include the best items for caching. We compared our results with two famous caching algorithms, LFU and LRU. Results have shown better performance of our proposed method. This privilege is higher when the cache size is more limited.

References

[1] Agrawal R, Imielinski T, Swami "A. Mining Association Rules between Sets of Items in Very Large Databases"[C]. Proceedings of the ACM SIGMOD Conference on Management of Data, Washington, USA, 1993-05: 207-216.

[2] Ebrahimi, A, Bahrbegi H, 2004. "A Novel Datamining Algorithm. " Proc of the *1st Conference on Electrical and Computer Engineering Applications*, *Lahijan*, *Iran*, March 2004.

[3] L Guan, S Cheng, and R Zhou, "Artificial neural Han J, Pei J, Yin Y. Mining Frequent Patterns without Candiate Generation[C]. Proc. of SIGMOD'00, Dallas, 2000:1-12

[4] Jianhan,zhu.,"Mining Web Site Link Structures for *Adaptive Web Site Navigation and Search*,",Ph.D Thesis, University of Ulster at Jordanstown,October 2003.

[5] Khosravi, M. Nematbakhsh, "Designing a catalog management system- an ontology approach", Malaysian Journal of Computer Science, Vol. 20(1), 2007, pp. 35 – 50.