



A Hashing Technique for Supporting Updates Efficiently in Search As You Type SQL Keyword Search

Dinesh Selvaraj^{*}, Ashok Raj. K, Manoj Kumar. V
SITE &VIT University
India

Abstract— *Search-As-You-type is a Technique which assists the user is completing the keywords. Search as you type for database supports searching the keywords in the database and returning the location of the keyword as the user is typing. The Search as you type is implemented in database using SQL. SQL being a native language of database increases the performance and decreases any overhead. The implementation just consists of three table prefix, keyword, Inverted Index table. When new keywords needs to be entered in the middle of the keyword table regenerating the keyword id becomes a issue. If the keyword table needs to be updated frequently causes processing overhead to the system. Hence hashing has been proposed to hash the keywords quickly and separate chaining technique is proposed to solve the problem of adding keyword in the keyword table. The proposed technique minimizes the process overhead when compared to the previous technique.*

Keywords— *SQL, Inverted Index, Prefix, Hashing, Separate Chaining*

I. INTRODUCTION

Various companies like IBM, ORACLE employ different approach to retrieve a data from their database. They follow various procedures, designs & various patterns to increase their search efficiency, they focus on the areas like selection of various kinds of databases, technology and Languages which are used to retrieve data from the database. In the efficient retrieval of data from a database the language they use, type of keyword search technology they use in retrieval are more important and hence the Keyword search forms the most important task in retrieval of those data's. So many companies are focusing on efficient keyword searches which increase the chance of predicting those absolute data's. Traditionally they are many keyword searching techniques employed and they are deployed in databases based on their needs. AUTO-COMplete is the keyword searching technique which predicts the phrase or a word based on the partial string which is typed by the user this resulted in improving key word search up to some extent. COMPLETE SEARCH technique was then introduced it predicts the query keyword at any place in the answer but it was not a appropriate search. TYPE-AHEAD technique[2] was then implemented in Relational databases which provided better results than other techniques but it has an disadvantage that it can work only with Relational database but can't be implemented in the databases containing XML data[4]. See also [10] for complete study on type ahead search. Next comes the SEARCH AS YOU TYPE[1],[3] which uses SQL for providing such function. This method searches the contents when the user types and it also returns the location of the record present in the table using three table. They use SQL itself to retrieve records from table, so that no additional overheads will be happened as in case of using external languages such c++, java. Since this method uses database tables and sql which is native to RBMS. The keyword prediction and updating keywords frequently becomes another tough task when every new data is stored in the database. A better method in updating keyword after every insertion of record would result in efficient keyword updation and parsing contents with database native languages like SQL would even increase efficiency and response time. As all the keywords and ids in a table has to be sorted in order on each update leads to performance degradation, a data storing technique has to be implemented which will be efficient in keyword search .

In this paper, the contributions are as follows :

- We propose a framework for implementing search as you type with new keyword updation technique.
- We present a hashing technique for efficient keyword updation.
- The proposed method is implemented and validated.

The rest of this paper is presented as follows. Section II gives discussions on the related work; Section III describes architecture. Section IV presents Proposed Methods. Section V discusses experimental results. Section VI concludes the paper.

II. RELATED WORK

In the related work of this paper, we survey briefly with the following two aspects: keyword search types, hashing technique.

A. Keyword Search

Topic based Query Terms are suggested on the fly as user types, is a work suggested in [6]. This paper presents a model to identify similarity between terms using a probability function. But this method is not considered as we are

concentrating keyword or query term search using SQL in RDBMS.

In reference [1] they have proposed an idea of using SQL the native database language as SEARCH-AS-YOU-TYPE. They have created three tables. First, KEYWORD TABLE(id,keywords)-which contains keywords and their ids. Second, PREFIX TABLE(pre,lkid,ukid)-consists of the keyword and its range of lower to higher in the inverted index table. Third, the INVERTED-INDEX TABLE(kid,rid)-consists of the id to the keywords and their location in the table.

When a user enters a prefix the sql engine searches for the prefix from the prefix table, where it retrieves the lkid and ukid. The range(lkid-ukid) tells the range of keyword id(kids) where that specific prefix is present. With that range the required keyword is retrieved from the keyword table, then from the retrieved keyword name, the record id(rid) is retrieved which point the corresponding record present in the actual table. This method of search as you type(type ahead search) using SQL Tables showed a remarkable performance than any programming language techniques.

B. Hashing Technique

Hashing is the best technique in case of reduced search time and for information storage and retrieval problems for large data sets. By using hashing technique on tables it has been studied that insertion and deletion makes only a single record in a table has to be accessed for structuring the table[7],[8],[9],[11].

The problem identified from the existing work is, the keyword table has to be globally reordered every time on insertion of a new record into the master table when the reserved space for that keyword table is already consumed. Therefore hashing techniques such as separate chaining etc... can be used to store the records, which performs good space utilization.

III. SYSTEM ARCHITECTURE

The keyword table consists of a index for every alphabets identified by 1 to 26. The hashed keyword table has a HASH function which categories the keywords based on the index value. The hash function parses the first letter of every keyword and categories the keywords based on the key value in the keyword table. For the index which has more than one keyword hashing to it, a separate linked list is created and the keywords are added to it. Initially all the index points to a null value. When the keywords are added to the particular index at first a linked list is created and the keywords are added to the linked list and then index value is made to point to the linked list. The architecture diagram of the proposed HASHED KEYWORD table is shown in figure 2.

A. The Node structure

The node structure of the linked list is given in figure 1.

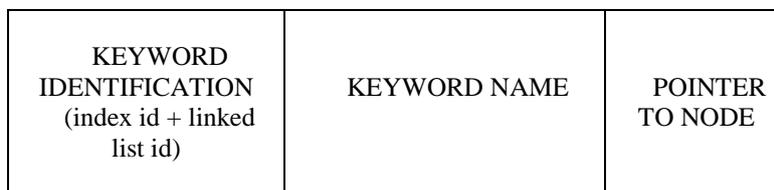


Fig. 1 Node Structure of Linked List

B. Keyword identification

The keyword id in the hashed keyword table consists of combining the keyword id of the index and its keyword id in the linked list. For example the index key id is 5 and the id of the keyword in the linked list is 5. Then by combining them we get 5.5. The keyword identification uniquely identifies the keywords in the keyword table. This method also is useful when the keywords are inserted into the keyword table.

C. Keyword Name

The keyword name is the name of the keyword which is inserted into the keyword table. The keywords in the keyword table are sorted using alphabetical order.

D. Pointer to node

The pointer node which points to another node which is present in the linked list.

IV. EXPERIMENTAL RESULTS

The experimental result is conducted by inserting the words into a normal keyword table and hashed keyword table the results are compared for insert, update, and deleting in a keyword table and a hashed keyword table. In the table 1 time complexity is calculated and compared for the various operation in the different tables.

A. Insertion in keyword table

When inserting the keywords in the middle of the keywords table. The id values of the keyword table are regenerated. The changes are also propagated to the prefix and the inverted index table. If these changes made are frequently and large then the performance of the system if affected severely. When a keyword needs to be inserted the time complexity to find the keyword is $O(n)$ and to add the keyword to the keyword table is $O(1)$ and to update the remaining indexes is $O(n)$ hence the total time complexity for insertion is $O(n^2)$. The Update and deletion of the keyword in the keyword table also consists of the same complexity $O(n^2)$.

B. Insertion of keyword in Hashed keyword table

Inserting a keyword in the hashed keyword table, the key index is easily found by the hash function and insertion process involves find the correct position by the alphabetical order. The insertion of keyword has a time complexity of $O(n)$. Update and deletion consists of $O(n)$. Table 1 shows the time complexity of the various operations between the keyword table and hashed keyword table.

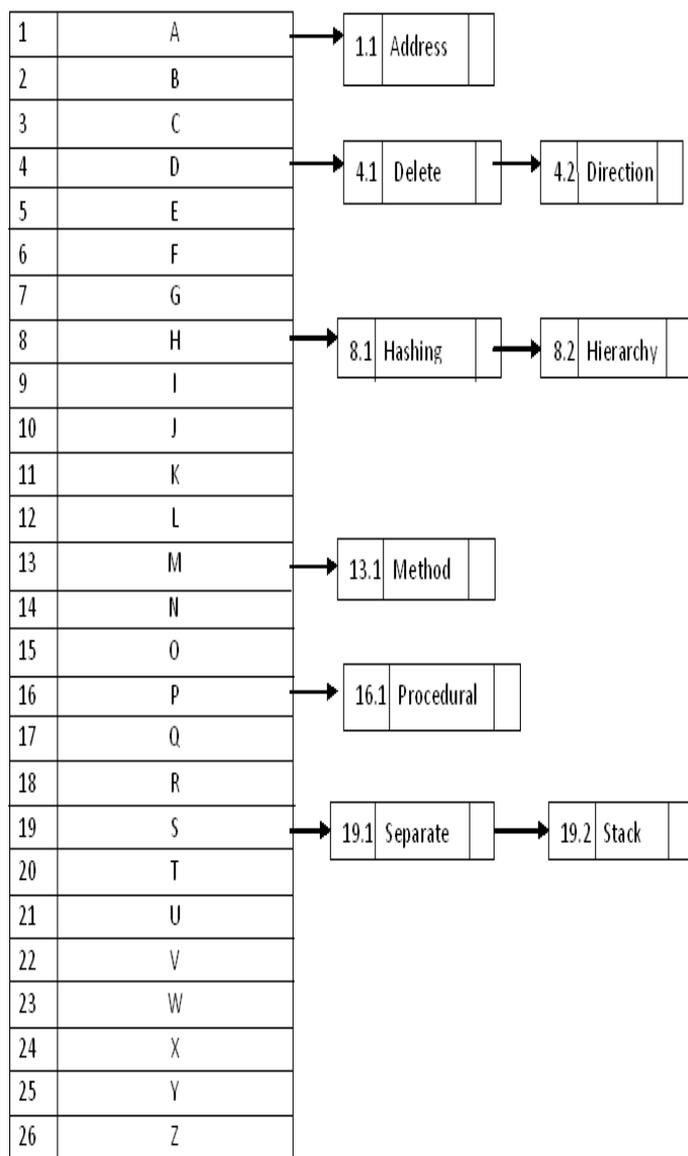


Fig. 2 Hashed Keyword Table

TABLE I
TIME COMPLEXITY DIFFERENCE BETWEEN KEYWORD TABLE AND HASHED KEYWORD TABLE

Operations	Keyword Table	Hashed Keyword Table
Insertion of keyword	$O(n^2)$	$O(n)$
Updation of keyword	$O(n^2)$	$O(n)$
Deletion of keyword	$O(n^2)$	$O(n)$

V. CONCLUSIONS

In this paper, we studied about the problem on insertion of keywords in the keyword table in search as you type keyword search. We proposed effective hashing technique called separate chaining to efficiently support dynamic updates on keyword table. The experimental results showed us the hashed keyword table performs better than the keyword table. The time complexity of the operation performed on hashed keyword table is less than keyword table. This result also holds true for large values of keywords in the hashed keyword table. Future work may be implementing various hashing and other file storing techniques in SQL based search as you type keyword search..

REFERENCES

- [1] Guoliang Li, Jianhua Feng and Chen Li, "Supporting Search As You Type Using Sql in Database", *IEEE transactions on knowledge and data engineering*, Vol. 25, No. 2, pp. 461-475, February 2013.
- [2] Jianhua Feng and Guoliang Li, "Efficiency fuzzy type ahead search in XML Data", *IEEE transactions on knowledge and data engineering*, Vol. 24, No. 5, pp.882-895, May 2012.
- [3] Wu, Hao, Guoliang Li, Chen Li, and Lizhu Zhou. "Seaform: Search-as-you-type in forms." *Proceedings of the VLDB Endowment* 3, No. 1-2, 2010, pp.1565-1568.
- [4] Guoliang Li, Shengyue Ji, Chen Li, Jiannan Wang, Jianhua Feng, "Efficient Fuzzy Type Ahead Search in TASTIER", *IEEE 26th International Conference on Data Engineering (ICDE)*, 2010, pp.1105-1108.
- [5] Adam Kirsch, Michael Mitzenmacher, "On the Performance of Multiple Choice Hash Tables with Moves on Deletes and Inserts", *46th Annual Allerton Conference on Communication, Control, and Computing*, 2008, pp.1284-1290.
- [6] Fan, Hao Wu, Guoliang Li, Lizhu Zhou, "Suggesting Topic-Based Query Terms as You Type", *12th International Asia-Pacific Web Conference*, 2010, pp.61-67.
- [7] Jin-cheng li, Xian, "Boosting Multiple Hash tables to Search" *Proceedings of the 2012 International Conference on Machine Learning and Cybernetics*, (15-17)-July- 2012, pp.57-62.
- [8] Witold Litwin, "Linear Hashing : A New Tool For File And Table Addressing", *IEEE*, 1980, pp.0000-0012.
- [9] Ho, Yuk. "Application of minimal perfect hashing in main memory indexing.", *ACM*, 1994.
- [10] Guolian Li, Shengyue Ji, Chen Li, Jianhua Feng, "Efficient Fuzzy Full-text Type-Ahead Search", *The VLDB Journal—The International Journal on Very Large Data Bases*, Vol 20 Issue 4, pp.617-640, August 2011.
- [11] Ozel, S. A., and H. A. Guvenir. "An algorithm for mining association rules using perfect hashing and database pruning." In *10th Turkish Symposium on Artificial Intelligence and Neural Networks*, 2001, pp. 257-264.