ISSN: 2321-7782 (Online)
Volume 2, Issue 4, April 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Paper / Case Study
Available online at: www.ijarcsms.com

KSHI: Keyword Search using SQL in Database using Hybrid Indexing to Improve Efficiency

Shikha A. Shah[1]
ME Student, CSE Dept
Parul Institute of Technology
Vadodara – India

Shailendra K. Mishra[2]
Asst.Prof. CSE Dept
Parul Institute of Technology
Vadodara – India

Abstract: In today's emerging world, internet is widely used in every aspect of day to day life. Any government, non-government, corporate firms provide their services through internet. Internet is mainly used for searching, surfing and data collection. For this purpose different websites like Wikipedia, cisco, microsoft, ieee xplore, jstor, springer and many are mainly used. These websites contains large amount of data in the form of files and articles and they have their local database search tools. Main problem with these tools is that they are very slow and output result rate is very low, especially in the case of slow connections like our internet via dial-up connection, on smart phones, 2G wireless internet dongles. This makes the task of surfing and searching, time consuming and tedious for the users. And this might lead to discontinuation of the usage of the particular website. A data search algorithm from websites with Hybrid indexing technique is proposed named KSHI (Keyword Searching using Hybrid Indexing). Two indexes would be saved in the SQL database, which will contain keywords (Keyword index), and links (Link index). These SQL tables will keep the data in the form of a linked list. It will also maintain another table (Runtime table), which will keep the quick information. Speed of the search operation will be increased using the runtime table technique. So this approach will be very useful for fast searching.

Keywords: Relational database, keyword search, Indexing, SQL.

I. INTRODUCTION

Database is an organized collection of information that can easily access, managed and updated. Relational database is a collection of tables of data items that is generally organized as per relational model. As we all know relational database is nothing but a set of tables. Each table contains one or more data categories in columns. Each row contains a unique instance of data for the categories defined by the columns. The standard user and application program interface to a relational database is the *structured query language* (SQL). SQL statements are used both for interactive queries for information from a relational database and for gathering data for reports.

From a long time, the database searching has been doing only via the queries. The Relational DBMS provides its query language SQL that people have been using since inception of the RDBMS. The query language SQL has been based on the relational algebra. The relational algebra is what was defined by Tedd Codd as he was a mathematician. For a long time, it has proved to be the most efficient ways to search database. But then, due to the needs of the interoperations and increasing amount of the database, searching the database using the query has been a very difficult task. Moreover, while getting data from multiple different heterogeneous data sources, it has even become more complex query formation. Moreover, the amount of time required has been increased drastically. This has led the researchers to move towards the informal ways of database querying. Then, the keyword search algorithms for the relational database were formulated. In which, a novice user can search for the data it wants. He may not know the column name or the data structure. He may just type in the information he has to search for the information he is looking for. The input information is the related information for information they require. This gave good results initially and more researchers joined the army of the keyword searching of the databases.

II. Related Literature Review

Keyword search for relational database is there from a long time. It has mainly two approaches. First approach is using the relational algebra and using relational algebra try to formulate the SQL queries based on supply keywords. Another approach is using tree and graph structure. Feng Zao et.al. Proposed a BROAD [1] algorithm which uses tree and graph structure. This algorithm can be useful for XML and RDF databases.

In “Search-As-You-Type: Opportunities and Challenges [12], authors discusses the basic architecture of search-as-you-type. The client has a browser, using which a user can send requests to the server to retrieve results. Each keystroke of the user could invoke a query, which includes the current string the user has typed in. The browser sends the query to the server. The server tokenizes the query string, computes and returns to the user the best answers ranked by their relevancy to the query.

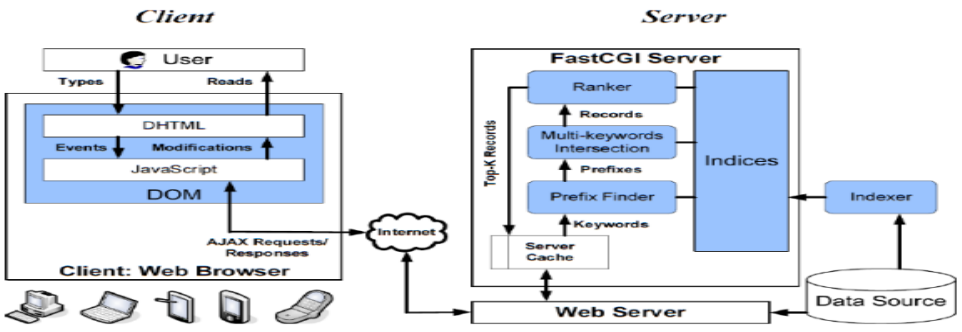


FIGURE 1: SEARCH-AS-YOU-TYPE ARCHITECTURE[12]

In the paper "Keyword Search in Databases: The Power of RDBMS"[2] the other approach based on the relational algebra and SQL was formulated. In this particular authors proposed a system that uses the tuple reduction techniques to provide faster speed and more relevant results. In most of the cases the keyword search algorithm returns many irrelevant results.

The DISCOVER [3], uses the two step process. First is generating the network and second is parsing the relevant data. It finds all the relevant networks of the candidate whose size and data bound using this technique. The drawback of this is time overhead and space complexity associated with this.

There are different approaches which uses combination of relational algebra and graph structure. The keyword search for relational database is proposed in BANKS [4]. The results are retrieved first and then results can be again filtered. This way the database can be browsed to search the data required. The algorithm is further extended to provide keyword data browsing like web browsing and then ranking the results.

Other approach discussed in "Keyword searching and browsing system over relational databases" [5], makes use of indexing. By using indexing search results appear fast but it is useful only when common keywords are searched. After indexing scoring and filtering is used to find most relevant records. Another approach discussed in ,"Efficient relational keyword search system" [6] makes use of ranking. Ranking is maintained using user feedback. It helps less for keywords which are not searched too frequently. The keyword search results were also affected by the functional dependency and the foreign key constraint. This is discussed in "Referenced attribute Functional Dependency Database for visualizing web relational tables" [7]. Authors provided the method to eliminate duplicates.

Coffman said that all keyword search algorithm focus on performance in "A Framework for Evaluating Database Keyword Search Strategies". He found that most of the algorithms are not efficient when the real workload is given. He developed a platform to evaluate different strategies. Coffman [10] has also proposed an empirical model to evaluate the different algorithms. The main criteria are speed of searching, memory usage, and relevance.

In the above discussions, we’ve seen the different approaches, their improvement techniques and the evaluation framework to compare the algorithms. In "Supporting Search-As-You-Type Using SQL in Databases"[11], authors discussed the advanced version of keyword search that provides search-as-you-type facility using SQL. Authors discussed a novel approach for fuzzy search also. Here indexing method is discussed which makes use of Inverted-index table and Prefix table.

Inverted-index table: Given a table T, they assign unique ids to the keywords in table T, following their alphabetical order. They create an inverted-index table IT with records inthe form <kid, rid> where kid is the id of a keyword and rid is the id of a record that contains the keyword. Given a complete keyword, we can use the inverted-index table to find records with the keyword.[11]

Prefix table: Given a table T, for all prefixes of keywords in the table, we build a prefix table PT with records in the form <p,lkid,ukid> where p is a prefix of a keyword, lkid is the smallest id of those keywords in the table T having p as a prefix, and ukid is the largest id of those keywords having p as a prefix.[11]

Thus, given a prefix keyword w, we can use the prefix table to find the range of keywords with the prefix. The following query shoud be fired.

SELECT T.* FROM PT,IT,T

WHERE PT.prefix = “w” AND

PT.ukid ≥ IT.kid AND PT.lkid ≤ IT=T.kid AND

IT.rid = T.rid [11]

The SQL could first use the index on prefix to find the keyword range, and then compute the answers using the indexes on kid and rid. In above literature survey we have gone through different approaches for keyword search in relational database. The following section discusses a novel technique for keyword search in relational database.

III. PROPOSED ALGORITHM

Here we are discussing a new technique for keyword search in relational database which makes use of hybrid indexing technique. Hybrid indexing here means permanent indexing as well as run-time or temporary indexing. For indexing we are using link-based approach. The technique or algorithm is described below.

The algorithm (KSHI) has following steps:

1) User will input the first word of query.

Example: data

2) The system creates a Hybrid Search Index (temporary table) in the memory for that group number. The system will match and create a Hybrid Search Index

Example: Group 2 in Hybrid Search Index in figure 2

This table was created by selecting the complete group entities from the *Link reference index table* and *Keyword reference index table.*

3) It will wait the second keyword to be entered.

4) Till the time there is not second keyword entered, it will show the “All entries in the Hybrid Search Index” as the search results to the user.

Note: If table is heavily populated, then It will show only first 10 entries and show a next button to view next entities. This will reduce the search algorithm overhead created by preventing the search list overloading (Search list overloading means showing all of the search results at the same time).

5) When the second keyword will be entered, only matching search results would be listed.

6) If second keyword does not match any option, the request will be sent to the content search module, which will search the content for the new keyword.

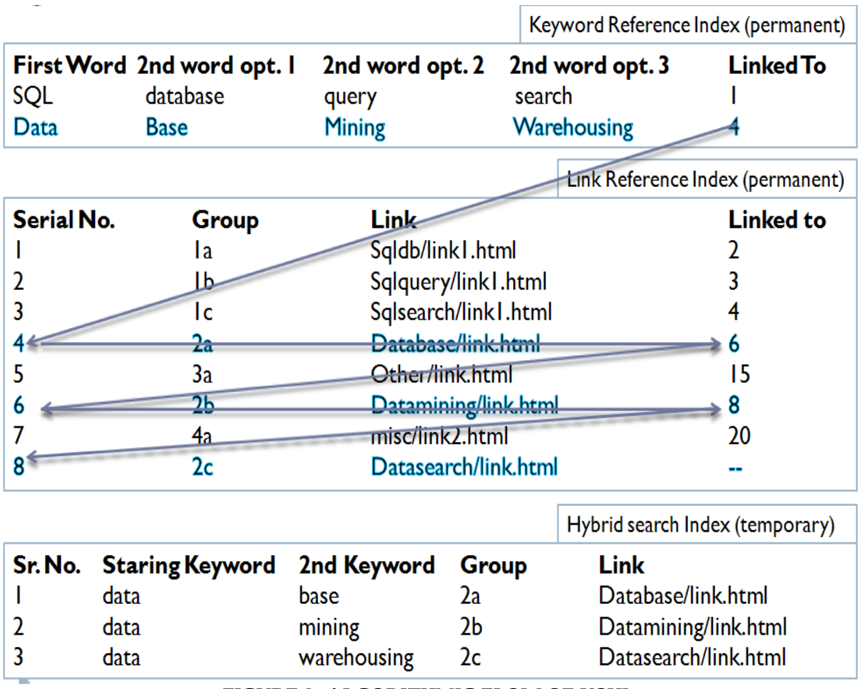
7) If search result found, content search module returns true and handover all of the details to the index search module.

8) It will add the keyword and its link in the Keyword Reference Index and Link Reference Index.

9) In the meanwhile, the search result will be shown to the user.

10) If content search module return false, index search algorithm will show "no results found" to the user.

11) When user session will end, it will flush the temporary table, if not in use by any other user also.



Keyword Reference Index (permanent)

First Word	2nd word opt. 1	2nd word opt. 2	2nd word opt. 3	Linked To
SQL	database	query	search	1
Data	Base	Mining	Warehousing	4

Link Reference Index (permanent)

Serial No.	Group	Link	Linked to
1	1a	Sqldb/link1.html	2
2	1b	Sqlquery/link1.html	3
3	1c	Sqlsearch/link1.html	4
4	2a	Database/link.html	6
5	3a	Other/link.html	15
6	2b	Datamining/link.html	8
7	4a	misc/link2.html	20
8	2c	Datasearch/link.html	--

Hybrid search Index (temporary)

Sr. No.	Staring Keyword	2nd Keyword	Group	Link
1	data	base	2a	Database/link.html
2	data	mining	2b	Datamining/link.html
3	data	warehousing	2c	Datasearch/link.html

FIGURE 2: ALGORITHMIC FLOW OF KSHI

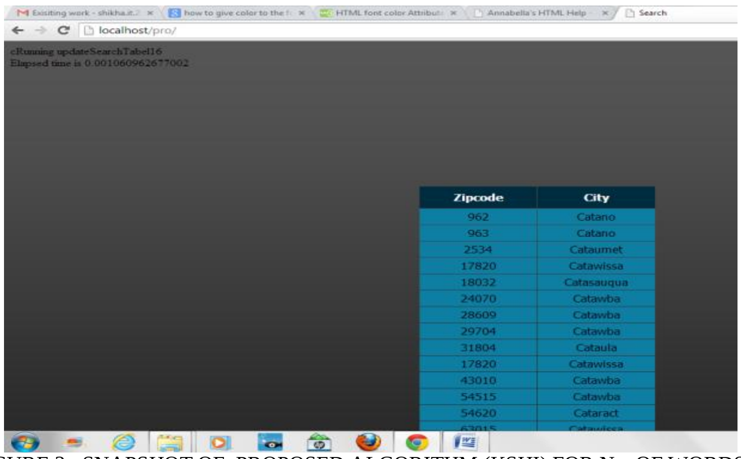
The algorithm proceeds as described above. Here we are giving the comparison between different existing indexing methods and our proposed indexing method.

The existing methods which use indexing methods, stores index on permanent basis. So there is wastage of memory. Here in our proposed indexing method we are using run-time index which is maintained temporary: as soon as user session ends it will remain in the memory for sometimes and then it is flushed out. So wastage of memory is less compared to other existing techniques

IV. Result and Analysis

For implementation of proposed algorithm we are going to use PHP with MySQL instances as laptop computers with 2.0 GHz Intel(R) Core(TM) i3-3110M CPU, 2.40 GHZ and 4.00 GB RAM having Windows 7 (64 bit) Operating system.

For implementation of the algorithm the database we have used is MySQL. It is real world database containing information about city zipcode, area code, country, state etc. of USA cities. It has over 45,000 records one can get this database from http://zipsdb.com/



```
cRunning updateSearchTabel16
Elapsed time is 0.00106096267 7002
```

Zipcode	City
962	Catano
963	Catano
2534	Cataumet
17820	Catawissa
18032	Catasauqua
24070	Catawba
28609	Catawba
29704	Catawba
31804	Cataula
17820	Catawissa
43010	Catawba
54515	Catawba
54620	Cataract

FIGURE 3 : SNAPSHOT OF PROPOSED ALGORITHM (KSHI) FOR No. OF WORDS-5

As shown in above figure, if we enter “cataw” in search box then the cities with their zipcodes are displayed which contains “cataw” as prefix. The time taken by this search is around 0.00106 ms.

The performance of the proposed algorithm may be measured on the basis of time complexity. Time complexity defines the response time for a user. Response time is defined as the time taken by the algorithm to search and give answers to the user.

No. Of chatacters entered	Words	Time taken existing technique in ms	Time taken by proposed algo.(KSHI) in ms
3	cat	0.8985471725	0.001492977
4	cata	0.069954156	0.001060962
5	cataw	0.064224403	0.000570058

TABLE 1 : COMPARISON OF RESULT IN TIME

The algorithm proceeds as shown in figure 2. In the first step as soon as the first word is entered, hybrid search index table is created at the run-time with the use of keyword reference index table and link reference index table .As stated above hybrid search index table is created remains temporary in the memory. So memory wastage should be less.Response time should be measured in milliseconds. Response time of our proposed method should be less compared to existing method. Existing method may give result in 1 ms, our proposed method can give result in less than 1ms because of above stated reason and it is clear from above table1. The result assessment graph is shown here.

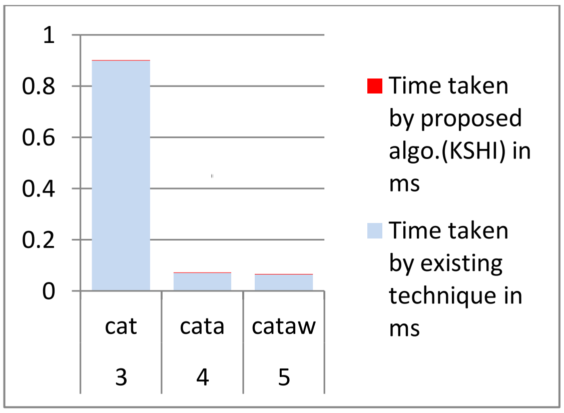


FIGURE 4 : RESULT AND COMPARISON OF EXISTING WORK AND PROPOSED ALGORITHM(KSHI)

V. Conclusion

In this paper, we have proposed a new technique for keyword search in database that makes use of permanent indexing as well as temporary indexing. We focus on how to support keyword searching efficiently with less memory wastage. We have implemented the algorithm on large real dataset. By experiments we will see that the response time should be less as compared to previous indexing techniques. The result and analysis will prove that our technique is more efficient compared to existing technique and it provides search result faster. So the response time should be less and efficiency is improved.

References

1. Feng Zao, Xiaolong, Anthony, Gand "BROAD: Diversified Keyword Search in Databases" NUS SOC Technical Report Number TRD3/11 March 18, 2011.
2. Lu Qin, Jeffrey Xu Yu, Lijun Chang "Keyword Search in Databases: The Power of RDBMS" ACM 978-1-60558-551-2/09/06 SIGMOD'09, June 29–July 2, 2009.
3. Vegelis & Yannis "DISCOVER: Keyword Search in Relational Databases" Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.
4. Arvind, Gaurav, Churata, Soumen "Keyword Searching and Browsing in Databases using BANKS" ICDE,San Jose, California, 2002.
5. Khine, Phyo Thu Thu, "Keyword searching and browsing system over relational databases" IEEE 2011 Sixth International Conference on Digital Information Management (ICDIM)26-28 Sept. 2011 Page(s): 121 – 126
6. Khine, Phyo Thu Thu,"Efficient relational keyword search system" 2011 IEEE International Conference on Intelligent Computer Communication and Processing , 25-27 Aug. 2011, Page(s): 65 – 70
7. Jayanthi, S. K. "Referenced attribute Functional Dependency Database for visualizing web relational tables" E-ISBN: 978-1-4244-8679-3, 2011 3rd International Conference on Electronics Computer Technology (ICECT), April 2011.
8. G. Li, J. Fan, H. Wu, J. Wang, and J. Feng, "Dbease: Making DataBases User-Friendly and Easily Accessible", 5th Biennial Conference on Innovative Data System Research(CIDR'11) January 9-12, 2011.
9. Coffman, J. "A Framework for Evaluating Database Keyword Search Strategies" ACM 978-1-4503-0099-5/10/10, CIKM'10, Toronto, Ontario, Canada. October 26–30, 2010.
10. Coffman, J. "An Empirical Performance Evaluation of Relational Keyword Search Techniques" IEEE Transactions on Knowledge and Data Engineering, Page(s): 1, ISSN : 1041-4347, November 2012.
11. Li, Guoliang "Supporting Search-As-You-Type Using SQL in Databases " IEEE Transactions on Knowledge and Data Engineering, ISSN : 1041-4347, February 2013.
12. Chen Li, Guoliang Li, "Search-As-You-Type: Opportunities and Challenges" Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2010.
13. "Database system concepts" Sixth Edition By Abraham Silberschatz, Henry F. Korth and S.Sudarshan.
14. http://en.wikipedia.org/wiki/database