Performance And Efficiency Analysis Of Keyword Query Routing Over Relational Databases

P. Anand

PG Student, Department of Computer Science and Engineering, Madanapalle Institute of Technology & Science, India.

Abstract: Keyword search is familiar and effective way to find the Information inside the relational databases. Commercial RDBMS provides query the capabilities of textual attributes that are join the state of art of Information Retrieval (IR) relevant ranking tactics. This searching requires the exact column to a given list of keywords are matched. In this we are mainly focusing on keywords based upon the user preferences, the results will be retrieved. This system Improves performance and efficiency of scores depends upon the given query processing based upon ranking function. Here we propose a keyword query routing for, only reduce the processing searching a keyword queries over different sources. We are mainly aiming on a keyword element relationship summary that are closely specifies relationship between keywords and data elements. We may challenging the improve performance and efficiency and effective of total keywords search over a large relational databases

Keywords: Web mining, keywords search Indexing, SQL, Performance, Relational database, Routing, Efficiency.

I. INTRODUCTION:

The web is no longer to gathering of textual data but also a web of inter linked records source. This relationship is considered at the various levels such as keyword level, element level, set level etc. There are two types of databases text databases and relational databases. In text databases, is searched by the user upon documents. In relational databases, is searched as in the form of columns, tables and primary key to foreign key relationships. It is hard to a non-technical web-users to get the documents. Technical users have knowledge of SQL language, so that he can easily exploit the web data. Every user can get data using keyword search. Keyword search do not required any knowledge of structured queries i.e. query languages, the schema, or the underlying data. Many databases, say, SQL server and Oracle support type ahead search. But there are many challenges which are to be considered and more over all databases do not support this feature. Constructing indexes on databases using separate application layer can be used to maintain indexes. However this feature has a benefit that it can be used to achieve performance, on the contrary it has a major problem of duplication of data and indexes that may result in additional hardware costs Database extenders, say, Informix Data Blades, MS SQL Server CLR integration allow developers to provide some additional features, but the point of concern is non-availability of extender feature in databases such as SQL. To use standard SQL techniques which are also portable to other databases.

II. PERFORMANCE ANALYSIS OF KEYWORD QUERY ROUTING:

Discovered Work:

Keyword Query Search can be divided into two directions of work. They are:

- It calculate the most related structured results and
- Solutions for source selection calculate the most relevant sources.

Keyword Search:

In the keyword searching, we mainly follow two methods.

In **schema-based** methods we employed on top off-the-shelf databases. It is a process of mapping with keywords to elements in the database (called keyword elements).

In Schema-agnostic methods work directly on the data.

In particular, we mostly observed, on multi-keyword search. In multi-keyword search techniques, a user types in query containing multiple keywords, and find tuple\s that are similar to these keywords irrespective of the location of keywords. for example, if a user types in "Operating Machines"‖ to find out a book by "R.K.Mishran"‖ with a title including "Operating" and "Machines"‖ irrespective of the locations.

A. Architectural Description:

Initially the user/end-user's gives the keyword in the user interface. The query given by the user will be sent to the evaluation engine for evaluation purpose. queries will be processed by the candidate network and the then it is sent to the search engine for the keyword by executing they match. The searched results are sending to the ranker which will rank the results based on the calculated score. Finally the generated result is send to the user interface to view the result by the user; the obtained results are stored in summary database. When we search similar searches the system automatically checks the initially checked results are performed. The results are retrieved from the summary databases. Updated values are integrated with the database. Most of these applications are problematic in ranking i.e., (IR)-information –retrieval system is done. While we retrieve the documents from the RDBMS system based on user query then, ranking shows a significant role to carry their document in an arranged manner to build upon differential constraints. Mainly, computations are distributed into four phases:

ISSN: 2278 – 7798

- Computing route graph,
- Combinational route graph.
- Route plan are based upon ranking query.
- Analysis the results.

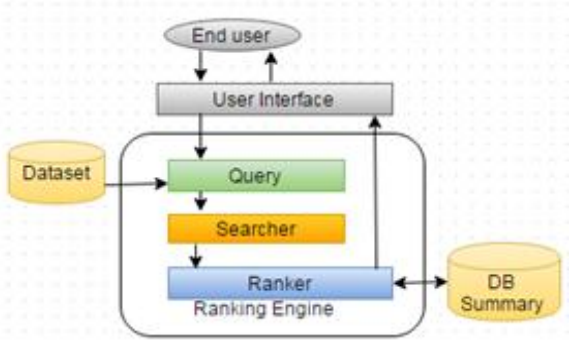

Figure 1: Architecture of Query processing

Computing route graph are based upon the keywords search graph. The keywords are routing in different ways and finalize the routing based upon ranking and analysis the results.

B. Methodology:

- Given a query keyword based on this computing a routing plan [RP].The determined joined plan specifies the, KERG relationships are retrieval for every pair of keywords.
- T is the intermediate table, join sequence of KERG are depends upon the tuple captures.
- When JP is off combining the total value are calculated for every Tuple in T. Routing graphs, specifies the same sources, and finally single combined result are maintained. May analysis the final outputs only the unique combination of sources. According to the routing graphs the calculated the score based upon the ranking system, i.e., Keyword-element Score. Some ranking functions are used for time frequency and inverse document frequency.
- Our test to explore quality of routing plans. We produce some results based upon that results, how they are matched based on some information, i.e., we evaluates together weight and relevant of route plan. Again we remember the KERG, may arrest altogether valid plans, score manner depends upon relevance plans. As we properly precise search space based on KERG. We properly calculate score based upon this we improve performance that we have seen. See below it's given some formulas for score.

$$tf(k_i, n'_k(k_i, n'_k, g)) == |\{n \mid n \in n', k_i \in \varepsilon_A(n)\}|$$

$$idf(k_i) == \ln \frac{|N'_k|}{|\{n'_k \mid n'_k \in N'_k, tf(k_i, n'_k) > 0\}|}$$

Figure 2: Score calculation formulae

- Where TF-IDF stands for inverse document frequency and term frequency is a statistical numeric value of each word in a document as important, later used as weighing factor to increase the number of times a word appear in the document.

III. EFFICIENCY ANALYSIS OF KEYWORDS IN RELATIONAL DATABASES

Keyword search for relational database is there from a long time. It has mainly two approaches. First approach is 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 [8] algorithm which uses tree and graph based structure. Algorithms can be useful for XML and RDF databases.

In "Search-As-You-Type: Opportunities and Challenges [9], authors discusses the basic architecture of search-as-you-type. The client send a request to the server to getting results. Each and every time the user the user might be invoked a query. The browser request sends the query to the server side. The processing queries string that are tokenized and return back to the user may select the better result based upon their ranks to the relevant query

Other approach discussed in "Keyword searching system over relational databases" [5], makes use of indexing. By using indexing search results appear fast but it is very useful for only whenever you search the common keywords.

Coffman said that all keyword search algorithm focus on performance in "A Framework for Evaluating Database Keyword Search Strategies ". He found that the given real workload algorithms are not efficient. He developed a plat form to evaluate different strategies. Empirical model proposed by the Coffman [11] to calculate the different relevant algorithms. Major thing is to speed up the performance and efficiency of search, usage of memory and relevance.

In "Supporting Search-As-You-Type Using SQL in Databases"[9], authors discussed the keyword search facilitates the advanced searching keywords 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:

Consider a table T, contains a keywords as unique. They created a inverted-index table IT contain records in the form <kid, rid> i.e., kid means keywords ids, rid means records ids contains keywords.

Prefix-table:

Given a table T, for all prefixes of keywords, we built for prefixes called prefix table PT with records in the form <p, lkid, ukid> where p is keyword prefix, lkid is the smallest keyword id., ukid is the largest keyword id.

In a table keyword prefix w,the prefix table to find the range of keywords. The following query should 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 [12]

ISSN: 2278 – 7798

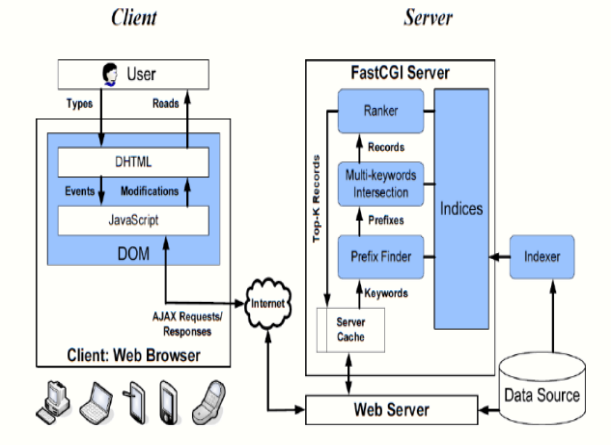

Figure 3: Search Type Architecture

To find the range of a keyword initially SQL uses the index on prefix table calculate the results based on kid and rid. In above literature survey we have gone through different approaches for searching keywords in relational database. The following section discusses keyword search based on novel technique in relational database.

Proposed Algorithm

Here we are discussing a keyword search based on novel technique 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.

1) Input given by the user as a query.

Example: *data*

2) We maintain a Hybrid Indexed \search table For memory management to assign group numbers. The system will match and create a Hybrid Search Index

Example: Group 2 in Hybrid Search Index in fig 5. This table called Link reference index table and Keyword reference index table.

3) Enter the second keyword till wait for some time.

4) Till wait for some time to enter second keyword

5) Matching results will be listed whenever enter the second keyword.

6) Entered second word not matched on that time send to search module to a new keyword.

7) Search module true, handover all details to indexed search module.

8) List of keywords and add to the link keyword and reference index

9) On time the results will shown to the user.

10) Search module false, show "no results found".

11) User session expires, the temporary table blush, means not access to other users.

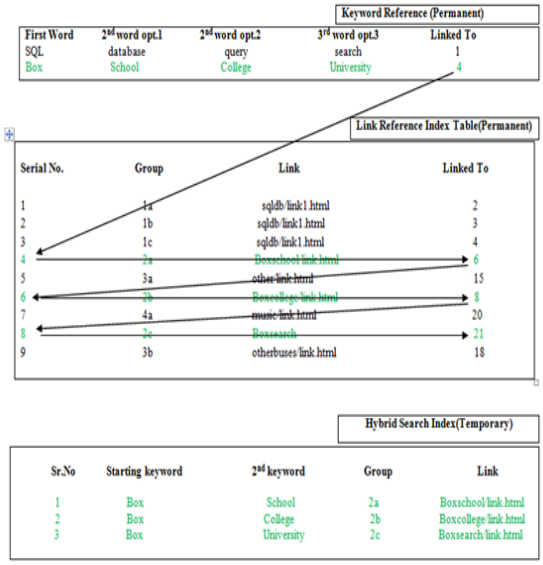

Figure 5: Algorithmic Flow of Kshi

IV. RESULT ANALYSIS

For implementation of proposed algorithm we are going to use PHP with MySQL instances as laptop computers with 500.0 GHz Intel(R) Core(TM) i3-3110M CPU, 500.0 GHZ and 8.00 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 INDIA cit ies. It has over 50,000 records one can get this database from http://indiapostal.com/

As shown in below figure, if we enter "Boxers" in search box then the cities with their zip codes are displayed which contains "Boxers" as prefix. The time taken by t his 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 t o the user.

No. Of Characters entered	Words	Time taken existing technique in Ms	Time taken by pro.algo.(KSHI) in ms
3	Box	0.8985471725	0.001492977
4	Boxer	0.069954156	0.001060962
5	Boxere	0.064224403	0.000570058

Table 1: Comparison Of Result In Time

The algorithm proceeds as shown in figure 5. 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 graph X- axis shows the keywords and Y-axis shows time (ms). The result assessment graph is shown here.

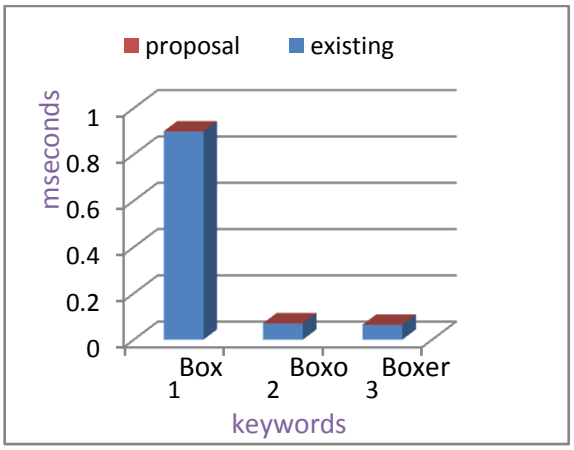

Figure 6: Result analysis of proposed system

v. CONCLUSION:

In this paper, we studied different Approaches (i.e. schema based and schema agnostic approach). As we specify a model keyword search system some sort of performance gain can be achieved. We propose a technique 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 performed the keyword based search and top-k results will be retrieved based upon the indexing. We have implemented the algorithm on large real dataset. 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. Many researches develop many techniques but still a challenge to give better results.

VI. REFERENCES:

[1] V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient IR-Style Keyword Search over Relational Databases," *Proc. 29th Int'l Conf. Very Large Data Bases (VLDB)*, pp. 850-861, 2003

[2] H. He, H. Wang, J. Yang, and P.S. Yu, "Blinks: Ranked Keyword Searches on Graphs," *Proc. ACM SIGMOD Conf.*, pp. 305-316, 2007.B. Kimelfeld and Y. Sagiv. Finding and approximating top-k answers.

[3] Y. Luo, X. Lin, W. Wang, and X. Zhou, "Spark: Top-K Keyword Query in Relational Databases," *Proc. ACM SIGMOD Conf.*,pp. 115-126, 2007.

[4] M. Sayyadian, H. LeKhac, A. Doan, and L. Gravano, "Efficient Keyword Search Across Heterogeneous Relational Databases," *Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE)*, pp. 346-355, 2007.[

[5] G. Li, S. Ji, C. Li, and J. Feng, "Efficient Type-Ahead Search on Relational Data: A Tastier Approach," *Proc. ACM SIGMOD Conf.*, pp. 695-706, 2009.

[6] G. Li, B.C. Ooi, J. Feng, J. Wang, and L. Zhou, "Ease: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-Structured and Structured Data," *Proc. ACM SIGMOD Conf.*, pp. 903-914, 2008.

[7] T. Tran, H. Wang, and P. Haase, "Hermes: Data Web Search on a Pay-as-You-Go Integration Infrastructure," *J. Web Semantics*, vol. 7, no. 3, pp. 189-203, 2009.

[8] Feng Zao, Xiaolong, Anthony, Gand "BROAD: Diversified Keyword Search in Databases" NUS SOC Technical Report Number TRD3/11 March 18,2011.

[9] Chen Li, Guoliang Li, "Search-As-You-Type: Opportunities and Challenges" Bulletin of the IEEE Computer Society Technical Committee on DataEngineering, 2010.

[10] Arvind, Gaurav, Churata, Soumen "Keyword Searching and Browsing in Databases using BANKS" ICDE,San Jose, California, 2002.

[11] 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.

[12] Li, Guoliang "Supporting Search-As-You-Type Using SQL in Databases " IEEE Transactions on Knowledge and Data Engineering, ISSN : 1041-4347,February 2013.

[13] http://en.wikipedia.org/wiki/database

ISSN: 2278 – 7798