Supporting Ranking Queries for Search-As-You-Type in Databases using WIB

NAGURU.CHENGAIAH  
Dept. of CSE  
Nova College of Engineering and Technology  
Vijayawada, A.P., India.

P RADHA KRISHNA  
Associate Prof & HOD, Dept. of CSE  
Nova College of Engineering and Technology  
Vijayawada, A.P., India.

Dr. K.RAMA KRISHNAIAH  
Professor  
NVR College of Engineering and Technology  
Tenali, A.P., India.

Abstract: Searching is one of the most important technique to search the required data from various sources. Search as you type is the current study to search the user requirement data from RDBMS. The main aim is to provide the high performance and to achieve high speed to the proposed system. In this paper, to improve the performance of the existing system is integrated with auxiliary indexes stored as tables to increase search performance and single keyword queries and multiple keyword queries and develop novel techniques for fuzzy search using SQL by allowing mismatches between query keywords and answers. The proposed system shows the performance of the auxiliary table and WIP-based searching that supports ranking queries.

Keywords: Search-As-You-Type, Fuzzy Search, First-N Queries, Weighted Index-Based Technique

I. INTRODUCTION

MANY information systems nowadays improve user search experiences by providing instant feedback as users formulate search queries. Most search engines and online search forms support auto completion, which shows suggested queries or even answers “on the fly” as a user types in a keyword query character by character. For instance, consider the Web search interface at Netflix, which allows a user to search for movie information. If a user types a partial query “mad,” the system shows movies with a title matching this keyword as a prefix, such as “Madagascar” and “Mad Men: Season 1.” The instant feedback helps the user not only in formulating the query, but also in understanding the underlying data. This type of search is generally called search-as-you-type or type-ahead search.

Since many search systems store their information in a backend relational DBMS, a question arises naturally: how to support search-as-you-type on the data residing in a DBMS? Some databases such as Oracle and SQL server already support prefix search, and we could use this feature to do search-as-you-type. However, not all databases provide this feature. For this reason, we study new methods that can be used in all databases. One approach is to develop a separate application layer on the database to construct indexes, and implement algorithms for answering queries. While this approach has the advantage of achieving a high performance, its main drawback is duplicating data and indexes, resulting in additional hardware costs. Another approach is to use database extenders, such as DB2 Extenders, Informix Data Blades, Microsoft SQL Server Common Language Runtime (CLR) integration, and Oracle Cartridges, which allow developers to implement new functionalities to a DBMS. This approach is not feasible for databases that do not provide such an extender interface, such as MySQL. Since it needs to utilize proprietary interfaces provided by database vendors, a solution for one database may not be portable to others. In addition, an extender-based solution, especially those implemented in C/C++, could cause serious reliability and security problems to database engines.

II. RELATED WORK

In this related work we are going to discuss different possible methods that are supports our approach support search-as-you-type and give their limitations and advantages. Using a separate application layer is the first method which can get very high performance as it can use various complex data structures and programming languages. Nevertheless, it is isolated from the RDBMS systems. Database extenders is the second method. Nevertheless, this extension-based method is “not safe” for the query engine, which could cause security and reliability problems to the database engine. This method depends on the API of a specific DBMS being used, and the different DBMS systems have different APIs. Furthermore, this method does not work if a DBMS system has no extender feature, ex., MySQL. Using SQL is the third method. The SQL-based method is more compatible as it is using the standard SQL. Even if DBMS systems do not provide search-as-you-type extension feature (indeed no Database Management Systems provide such an extension) the SQL-based
method can also be used in this particular case. So, the SQL-based method is more portable to the different platform than the first two methods. A simple way to support search-as-you-type is to issue a SQL query that scans every record and verifies whether record is an answer to the query or not. There are 2 ways to do the checking: one is Calling User-Defined Functions i.e UDFs. We could add functions into the databases to verify whether a record contains query keyword; and second is: Using LIKE predicate. Databases provide the LIKE predicate to enable users to perform string matching. We can use LIKE predicate to check if a record contains the query keyword or not. This method might introduce false positives, example, keyword “publication” contains the query the string “ic,” but the keyword does not have query string “ic” as a prefix. We can remove these false positives by calling the UDFs. The two noindex methods needs no additional space, but they may not scale as they need to scan all the records in the table.In this section, we propose to keep auxiliary tables as index structures to facilitate the prefix search. Some databases such as SQL server and Oracle have already support prefix search, and we can use this feature to do the prefix search. Nevertheless, not all the databases provide this feature. For this particular reason, we are developing a new method that could be used in all databases. Furthermore, we are maintaining inverted table that contains each keyword with specific unique id. Based on this specific keyword we can give the result to the user who is giving the request on-the-fly. On the other hand ranking adaptation is closely related to the classifier adaptation, which has shown its efficiency for many learning tasks. Nevertheless, to the best of our knowledge, there are no prior works on adaptation for the ranking problem. Furthermore the general difficulties faced by classifier adaptation, such as the covariate shift (or namely sample selection bias) and the concept drifting, ranking adaptation is relatively more challenging. Unlike the classifier adaptation, which mainly deals with the binary targets, and ranking adaptation desires to adapt model which is used to predict rankings for a collection of documents. Though documents are normally labeled with the several relevance levels, which seems to be handled by multi-class regression or classification, it is still difficult to directly use the classifier adaption for ranking. The reason lies in two-fold: one: in ranking, the mainly concentration is about the preference of the two documents or ranking of a collection of documents, which is a difficult to be modeled by the regression or classification; two: the relevance levels in between different domains are sometimes varied and need to be aligned. In this paper, we are also focusing on the adaptation of ranking models, instead of utilizing labeled data from the auxiliary domains directly, which might be inaccessible due to the privacy issue and data missing. Furthermore, Model adaptation is more advisable than data adaptation, because learning complexity is now only correlated to size of the target domain training set, which should be more smaller than size of auxiliary dataset.

III. EXISTING SYSTEM

Search engines and other online web apps will support the auto completion, which helps the user to answer “on the fly” as a user types in a keyword query character by character.

Since numerous search frameworks store their data in a backend RDBMS, an inquiry emerges actually: how to bolster seek as-you-write on the information living in a DBMS? A few databases, for example, Oracle and SQL server as of now bolster prefix inquiry, and we could utilize this component to hunt as-you-write. Be that as it may, not all databases give this element. Consequently, we concentrate new systems that can be utilized as a part of all databases. One methodology is to build up a different application layer on the database to develop records, and execute calculations for noting questions.

DISADVANTAGES OF EXISTING SYSTEM:

- In an existing systems are not specially designed for keyword queries, making it more challenging to support search-as-you-type.
- SQL meet the high performance requirement to implement an interactive search interface.
- Some important functionality to support search-as-you-type requires joint operations, which could be rather expensive to execute by the query engine.

IV. PROPOSED SYSTEM

We produce totally different methods to handle these difficulties. We have a tendency to propose 2 styles of methods to bolster scan as-you-write for single-decisive word inquiries, taking under consideration whether or not they oblige additional file structures place away as assistant tables.

We point out the routines that utilization SQL to output a table and check each record by line of work a consumer characterized capability (UDF) or utilizing sort predicate. We have a tendency to concentrate a way to bolster soft go after single-magic word inquiries.

We discuss a gram-based strategy and a UDF-based system. As the two methods have an occasional execution, we have a tendency to propose another neighborhood-era primarily based technique, utilizing the thought that 2 strings area unit comparative simply within the event that they
need regular neighbors gotten by erasing characters.

We extend the methods to support multi-essential word queries. we have a tendency to increase a word-level progressive strategy to proficiently answer multi keyword queries. Notice that once sent during a web application, the incremental-computation algorithms haven’t got to stay up session information, since the results of previous inquiries area unit place away within the information and shared by future queries.

**ADVANTAGES OF PROPOSED SYSTEM:**

- A main challenge is how to utilize the limited expressive power of the SQL language (compared with other languages such as C++ and Java) to support efficient search.
- We study how to use the available resources inside a DBMS, such as the capabilities to build auxiliary tables, to improve query performance.

**V. ENHANCEMENT**

A important challenge is that the thanks to impact existing info functionalities to accomplish superior in seeking pace and the way to support ranking queries that offer the foremost as usually as potential looked results at prime position in registered result. The planned strategy demonstrates to utilize weights of records place away as Associate in auxiliary tables to expand look execution. I actually have planned answers for single keyword queries and raise another procedure, weighted index-based technique called WIP-based searching that backings positioning inquiries down trying records in lightweight of prefix of catchphrases by utilizing additional weight table place away as helper table. My principle methodology is to push the burden needs into list primarily based ways. By this new WIP-based technique of pursuit as-you-write, question result offers the records endless provider of use.

**VI. RESULTS**

Search-as-you-type for single keyword: Exact Search: As a user types keyword w in the search box character by character, the system we are developing search as-you-type on-the-fly finds bunch of records that contain keywords with a prefix w. We call this search paradigm as prefix search. Without loss of generality, every tokenized keyword in data set and queries is assumed to use the lower case characters. For example, consider the data in Table 1, A1 ≈ title, A2 ≈ authors, A3 ≈ book title, and A4 ≈ year. In this exact search the keyword entered by the user is undergone to the DBMS engine and finds appropriate query with which it is started. If it found any matching then it will gives the query as suggestion to the user. By seeing this suggested result user could do the search more easily. In a particular situation user might not have proper idea about the query. In that situation our system helps more. It reduces the user burden by giving on-the-fly suggestion.

**Search-as-You-Type for Multi keyword Queries:**

Take a multi keyword query Q with m number of keywords they are w1; w2; . . . ; wm, as user is completing the last keyword that is wm, we treat wm as the partial keyword and the other keywords as complete keywords. As a user types in query Q letter by letter, our system search-as-you-type on-the-fly finds records that contain the complete keywords and the keyword with a prefix wm. For an example scenario, if a user types in a query “privacysig,” the system search-as-you-type returns records as r3,r6, and r9. In a particular, r3 contains the complete keyword “privacy” and another keyword “sigmod” with a prefix “sig .”As user types the word it searches in DBMS with same matching pattern. If found correct query related to the user wish then it comes in suggestion box. If user type any one of the similar word and types other word which is not related to the actual query in that case it won’t give any suggestion. As each query is divided into some number of keywords. So unrelated word never found in the DBMS system so the system search-as-you type will not give any suggestion to the user.

**Fuzzy Search:** In some of the cases the information is inserted into DBMS by special words. Here the special words means some of the information is stored with a name which is not related to that particular information. As a result user will never find the information as if he/she go normal search. By having this discussion it is worth full to have fuzzy search. Fuzzy means anonymous. The information is stored with anonymous name; For this in our system search-as-you-type we are developing fuzzy search also. In this type of search admin can upload the information of files with anonymous names. If user types the keyword related to the information file he/she won’t get any kind of suggestions. If user enters the proper anonymous keyword only the files which reside in the database comes as suggestion for user. By making this we can provide little security and only limited persons are allowed to access. So this fuzzy search helps in giving security when compared to the normal search.

**UDF Search:** I this UDF search scenario the query is partitioned by different words in the correspond query with a specific index for each word. All these queries are uploaded by the admin into DBMS. For example a query contains 10 words like w1,w2,w3…w10 as words. For all these word a specific number or unique number is generated. And in other hand the entire query also will have
specific or unique number to identify the query. Now if a user enters a number then it checks whether the number is available in the keywords list or not. If it founds in keyword list then it gives the suggestion as the main query which is connected to the keyword list. By this the inverted table helps in our system search-as-you-type on giving the results on-the-fly. Rank based Suggestions: Apart from all these search methodologies we are also giving rank based suggestions to the user. In this aspect we are taking the user click as feedback and based those feedback we are giving suggestions to the user. For example a user enter a query and for that query out system search-as-you-type gives many suggestions among those suggestion user might interested in any query. Now we are getting which query is further processed as feedback. Based on this feedback we are increasing the rank of that query. If any other user comes do search with the same keyword which is having higher rank then it will be visible on the first row. Therefore based on the ranks of the query our system is going to give response within fraction of seconds.

VII. CONCLUSION

In this article, we studied the problem of using the SQL to support the system search-as-you-type in data bases. And implemented various kinds of search techniques. We mainly concentrated on the challenge of how to make full use of the existing DBMS functionalities to meet high-performance requirement to get an interactive speed. To support the prefix matching, we proposed a solutions that uses the auxiliary tables as index structures and SQL queries to support the search-as-you-type. We enhanced the techniques in the case of fuzzy queries, and proposed various techniques to improve the query performance. We proposed multi keyword queries search, and studied how to support first-N queries and the incremental updates. And we are getting the feedback of the user requested queries and based on that we are giving rank to those queries. This is very helpful in the rank based search.

VIII. REFERENCES

[1]. “Scalable Ad-Hoc Entity Extraction from Text Collections,” by S. Agrawal, K. Chakrabarti, S. Chaudhuri, and V. Ganti,
[3]. “Efficient Exact SetSimilarity Joins,” by A. Arasu, V. Ganti, and R. Kaushik,
[4]. “ESTER: Efficient Search on Text, Entities, and Relations,” by H. Bast, A. Chitea, F.M. Suchanek, and I. Weber,
[5]. “Type Less, Find More: Fast Auto completion Search with a Succinct Index,” by H. Bast and I. Weber,

AUTHORS PROFILE


Mr. P.Radha Krishna did his Master in Computer Application at Lakkireddy Bali reddy college of engineering .He started his teaching career at Anurag Engineering College ,Kodad. Where He had guided be so many batches in their projects. He guided students in doing IBM projects at Anurag.He played an active role in the NBA Accreditation process at Anurag Engineering College in to 2007, He had acted as In charge Hod of MCA at Anurag. Mr. P Radha Krishna is a Member of ISTE, IAENG, ISRD .He had participated many workshops and paper presentations, which include sun-microsystem and open systems (LAMP) workshops .Mr. Radha Krishna is the secretary of Alumni Association of LBRCE .He is now working as the Head of the Department of CSE of Nova college of Engineering & Technology, Jupudi, Ibrahimpatnam.

Dr. K. Rama Krishnaiah is a highly qualified person, an efficient and eminent academician. He is an outstanding administrator; a prolific researcher published 33 research papers in various International Journals and a forward looking educationist. He worked in prestigious K L University for 11.5 years and he contributed his service for NBA accreditation in May 2004, Aug 2007 with ‘record rating’, ISO 9001:2000 in 2004, Autonomous status in 2006, NAAC accreditation of UGC in 2008 and University status in 2009. Later on he worked as Principal at Nova College of Engineering and Technology, Vijayawada for a period of 3.5Yrs. He took charge as the Principal, NVR College of Engineering and Technology, Tenali in May 2014.