Search As You Type in Database

Parvathi R, Syama R

Abstract—A search-as-you-type system computes answers on-the-fly as a user types in a keyword query character by character. Search-as-you-type support study on data residing in a relational DBMS. And also focus on how to support this type of search using the native database language, SQL. A main challenge is how to leverage existing database functionalities to meet the high-performance requirement to achieve an interactive speed. Study on how to use auxiliary indexes is stored as tables to increase the search performance. Solutions for both single-keyword queries and multi-keyword queries are presented, and develop novel techniques for fuzzy search using SQL by allowing mismatches between query keywords and answers. Experiments on large, real data sets show that techniques enable DBMS systems on a commodity computer to support search-as-you-type on tables with millions of records. The main consideration was to increase the speed by using auxiliary indexes stored as tables. The search is done based on both single and multi-keyword. Exact search for single keyword queries are done using UDF, LIKE predicate and inverted-index table and the prefix table. Exact search for multi keyword queries are done using UDF, LIKE predicate, full-text indexes and UDF (called "FI+UDF"), full-text indexes and the LIKE predicate (called "FI+LIKE"), the inverted-index table with prefix table and word-level incremental method. Fuzzy search for single keyword queries are implemented using UDF, gram-based method, neighborhood-generation-based method, character-level incremental algorithms. Fuzzy search for multi keyword queries are implemented using word-level incremental algorithms, called NGB+ and Incre+. The approach using inverted index tables and prefix tables supports prefix, fuzzy search and achieve the best performance. The experimental results on large, real data sets showed that the proposed techniques can enable DBMS systems to support search-as-you-type on large tables.

Index Terms— Fuzzy Search, Type Ahead, Prefix search, edit distance

I. INTRODUCTION

Data mining is the computational process of extraction of knowledge from large amounts of data. Many websites and search engines maintain auto complete search that offer numerous answers to the queries, provided by the user. This aspect is known as answers on the fly. Search-as-you-type system work out answers on the fly as a user types a keyword query.

Majority of online search and search engines sustain auto completion that indicates recommended queries or even answers the queries on the fly as a user types a keyword query. In addition to that, they help the users to avoid making spelling mistakes. Many search systems gather their information in a backend relational DBMS. Some databases such as Oracle and SQL server previously maintain prefix search, and so this feature can be used to do search-as-you-type. On the other hand, every database does not grant this feature. On behalf of this reason, contemporary methods can be used in all databases. Consider an example; Netflix, Inc. is a provider of on-demand Internet streaming media available to viewers in all of North America, South America and parts of Europe, and of flat rate DVD-by-mail in the United States, where mailed DVDs are sent via Permit Reply Mail. When the user searches videos on the database of Netflix, the user will get help from database to understand the actual query of the user.

In conventional search systems, answers are retrieved only after the submission of the entire query. If the user is having limited information about the specific data to be searched, they often feel "left in dark" and they need to use, try and see approach for finding the information. Latest approach is to develop a separate application layer on the database to construct indexes, and implement algorithms for answering queries. Advantage of the conventional approaches was getting a high performance, but its major disadvantage is duplicating indexes and extra hardware costs. Now-a-days many information systems follow better search experiences as the user receives the result as soon as the search queries are generated. This instant feedback helps the user to understand the respective data. This particular type of search is usually known as search-as-you-type or type-ahead search.

In "Search-As-You-Type in Databases", the advanced version of keyword search that provides search-as-you-type facility using SQL [1]. Here indexing method makes use of Inverted-index table and Prefix table. The difficulty of using SQL to support search-as-you-type in databases is how to influence existing DBMS functionalities to meet up the high-performance obligation to accomplish an interactive speed. To support prefix matching, use of auxiliary tables as index structures is proposed as solution and SQL queries used to support search-as-you-type. The most important contemplation in Search as-you-type was to improve the speed with auxiliary indexes stored as tables. Based on both single and multi keyword, searching is done.

Main challenge is to extract maximum existing database functionalities to meet the high presentation requirement to obtain an interactive speed. In order to increase search performance, auxiliary indexes are stored as tables. Major goal is to employ the built-in query engine of the database system as much as possible. So that it can reduce the programming efforts to support search-as-you-type. In addition to that, solution which has developed for one database can also be used by the other databases that sustain the similar standard.

II. LITERATURE SURVEY

A. Keyword Searching and Browsing in Databases using BANKS

Bhalotia et at [2] have developed to reduce the cost of query

Revised Version Manuscript Received on August 24, 2015.

Parvathi R, Department of Information Technology, University of Kerala/ SCT College of Engineering, Trivandrum, India.

Syama R, Department of Computer Science, University of Kerala / SCT College of Engineering, Trivandrum, India.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

by taking advantage of the overlap of tokens surrounded by the set of entities. BANKS allow ignorant users to browse relational database with ease. Here each foreign-key–primary-key link is modeled as a directed edge between the matching tuples. Naturally, an answer to a query should be a subgraph linking nodes matching the keywords. Consider a directed tree containing a directed path from the root to each query node and the root node is an information node and the tree a connection tree. A reply to a query is a rooted directed tree. In the BANKS system relational database is modeled as a graph. Each node in the graph is denoted as tuple in the database.

Each foreign-key–primary-key link is modeled as a directed edge between the corresponding tuples. This can be easily extended to new type of connections. Instinctively, an answer to a query should be a subgraph connecting nodes matching the keywords. By looking at a subgraph it is not clear what information it conveys. So, it is necessary to find a node in the graph as a central node which connects all the keyword nodes, and efficiently reflects the relationship between them. Answer is considered to be a directed path from the root to each keyword node in a rooted directed tree. Ignoring directionality would cause problems because of "hubs" which are connected to a large numbers of nodes. As a result, the efficiency of proximity-based scoring is abridged, as many nodes that are within a small distance of many other nodes.

B. DISCOVER: Keyword Search in Relational Databases

Hristidis et at [3] have presented a system that performs keyword search in relational databases. It consists of mainly three steps. Initially, it produces the minimum set of candidate networks. Secondly, the greedy algorithm estimates the set of candidate networks which forms a near-optimal execution plan. Finally, DBMS processes the execution plan created. DISCOVER determines the candidate networks which allocate join expressions. It can bid an opportunity to develop intermediary results and can be used for numerous candidate networks computation. The Plan Generator which produces an execution plan that uses transitional results for evaluating the candidate networks. Ultimately for each line of the execution plan, an SQL statement is formed and is accepted to the DBMS. The DISCOVER, that uses two steps for processing. First is for the creation of the network and second is parsing the applicable data. The drawback of this is space complexity and time overhead.

C. Efficient Interactive Fuzzy Keyword Search

Li et at [4] recommended a information access paradigm, called "interactive fuzzy search," in which the system searches the underlying data "on the fly" as the user types in query keywords. Two unique features are: (1) Interactive: Best answers would be searched by the system as the user types keyword query; (2) Fuzzy: While searching for significant records, the system also tries to find the words similar to the keywords in the query. Some of the contributions are as follows: (1) Studies regarding the case of single keyword queries are offered and incremental algorithm used for computing keyword prefixes related to a prefix keyword. (2) For multiple keywords queries, the goal is to compute the records with keywords whose prefixes are similar to keywords query. In order to compute the intersection of the inverted lists for keywords, efficient and novel algorithm for computing the results is computed .Main aim for checking whether a record matches query keyword conditions is to use forward lists of keyword IDs .Novel on-demand caching technique is used for incremental search. It extends auto complete interfaces by allowing keywords to become visible in various attributes of the essential data, and then it finds significant records that have keywords corresponding query keywords approximately. The helpful algorithms for incrementally computing answers to queries by using cached results of earlier queries in order to accomplish an interactive swiftness on large data sets.

III. IMPLEMENTATION DETAILS

Two types of searching are performed for single and multi keyword that is Exact Search and Fuzzy search. Exact Search (Prefix Search) as a user types query or a partial keyword the records containing those initial characters or the keywords will be displayed immediately. Fuzzy Search (Approximate Search) It allows minor mismatches. The system finds the record with keywords that are similar to the query keyword. In Fuzzy search, auxiliary tables are created as index structures to react to a query.

Exact and Fuzzy Search for Single Keyword

Searching comprises of two methods,

1. No-Index method
2. Index method

NO INDEX METHOD:

1. Exact search

Here it give arise an SQL query to facilitate scanning of each record and verify whether the obtained record is answer to the given query. Two techniques are operated here as follows:

By using the LIKE predicate, Databases would allow users to perform string matching [5]. LIKE predicate can be used to check whether a record contains the query keyword, this method introduces false positives, these false positive can be removed by calling UDFs.

By using User-Defined Functions (UDFs), Functions can be added into databases to authenticate whether a record contains the query keyword. These both methods do not require additional space but they may not scale since they need to scan all records in database table.

2. Fuzzy search

UDFs are used in fuzzy search for the Single keyword. Performance can be improved by doing early termination in dynamic programming computation using edit-distance threshold. If there is a keyword in string having prefixes with an edit distance within a value, then it returns true. Thus it issues a SQL query which scans each record and calls UDF to verify the record.

INDEX METHOD

1. Exact search

Index-Based Methods create auxiliary tables as index structures to facilitate prefix search.

Every database does not support this prefix search so that a new method is require to develop, therefore it can be used in all database, and which also performs more efficiently. Descriptions of the additional auxiliary tables are described below as follows:

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

Inverted-index table:

Given a table T, it allocates unique ids and keywords in table T, following their alphabetical order. Inverted-index tables I_T with records in the form <kid, rid>, where kid is the id of the keyword and rid is the id of a record that contains the keyword. Inverted index table shown in figure 1(b).

Prefix table:

Given a table T, for all prefixes of keywords in the table, a prefix table P_T 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 prefix.
Prefix table shown in figure 1(c).

(a) Keywords

kid	keyword
k1	icde
k2	icdt
k3	preserving
k4	privacy
k5	publishing
k6	pvldb
k7	sigir
k8	sigmod
k9	vldb
k10	vldbj
…	…

(b) Inverted-index Table

kid	rid
K2	r10
K5	r6
K5	r8
K5	r10
K6	r1
K7	r9
K8	r3
k8	r6
k9	r8
k10	r4
…	…

(c) Prefix Table

prefix	lkid	ukid
ic	k1	k2
p	k3	k6
pr	k3	k4
pri	k4	k4
pu	k5	k5
pv	k6	k6
pvl	k6	k6
sig	k7	k8
v	k9	k10
vl	k9	k10
…	…	…

Figure 1: Inverted Index Table and Prefix Table

Figure 2 shows that how index based method works, to find the records by using these additional tables. Given a partial keyword w, initially its keyword range [lkid,ukid] is obtained by using the prefix table P_T, and subsequently achieve the records that have a keyword in the range through the inverted-index table I_T as shown in the figure

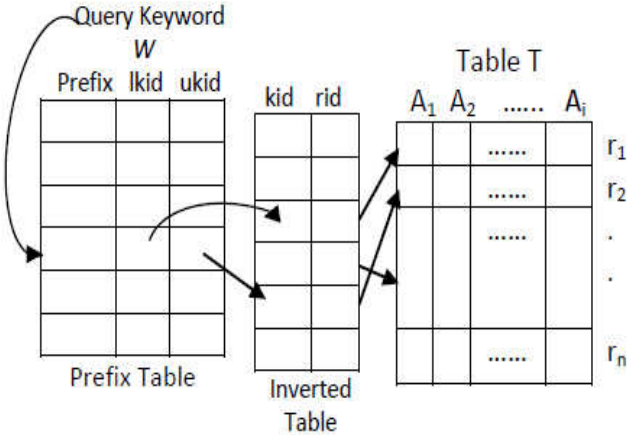

Figure: 2 Inverted-index table and prefix table

2. Fuzzy Search

Use of inverted-index table and prefix table are extended to support fuzzy search. For a given keyword, it first compute its similar prefixes from the prefix table get the keyword ranges of these similar prefixes, and then computes the answer based on these ranges using the inverted-index table.

2.1. Gram-based Method

There are various q-gram-based techniques that support approximate string search [5]. For a given string w, and its q-grams are its substrings of length q.In order to acquire the similar prefixes of the given keyword, it is required to conceive a q-gram table G_T with records<p,qgram> in addition to the inverted-index table and the prefix table whereas p is the prefix in the prefix table and q-gram is a q-gram of p.UDF method is used to verify the candidates to get the similar prefixes of w. Moreover, this method is inefficient for short query keywords [6], as short keywords have smaller numbers of q-grams and the method has low pruning power.

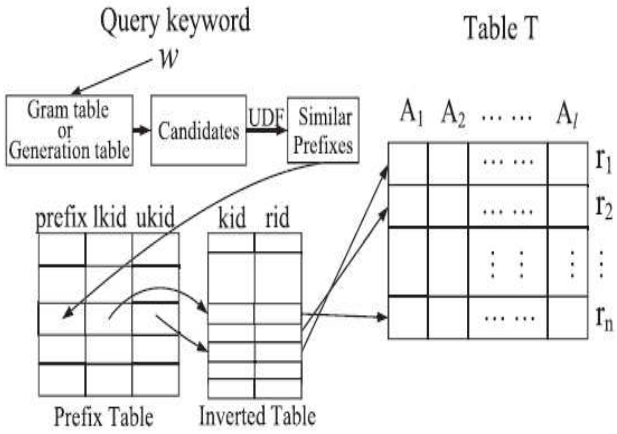

Figure 3: Q-gram table and the neighborhood generation table

2.2 Neighborhood Generation

Given a query w, the substrings of w can be obtained by deleting i characters are called " i-deletion neighborhoods" of w. Let $D_i(w)$ denote the set of i-deletion neighborhoods of w and $D_T(w) = \bigcup_{i=0}^{T} D_i(w)$.For example, given a string "pvldb,". D_0 (pvldb)= pvldb, and D_1(pvldb)={ vldb, pldb, pvdb, pvlb, pvld}.Whenever the user inputs the keyword w "pvldb".initially similar prefixes can be obtained in D_T which consists i-deletion neighborhoods $D_T(w)$= { vldb, pldb, pvdb, pvlb, pvld }.Now with edit distance 1 it is found that "vldb" similar to "pvldb".

Edit distance is used to find the similar prefixes of the given keyword w.so that all the prefixes can be prune if there is no common i-deletion neighbourhoods with w[8][9].To this end ,it is necessary to create a neighbourhood generation table $D_T$with columns as <p,i-deletion,i>,where p is the prefix of the P_T (Prefix Table). Figure 4 shows the neighbourhood generation table.

prefix	*i-deletion*	*i*
vldb	vldb	0
vldb	ldb	1
vldb	vdb	1
vldb	vlb	1
vldb	vld	1
…	…	…

Figure 4: Neighborhood generation table

For a keyword w ,initially similar prefixes should be acquired in $D_T$which have neighborhoods of w in D_T(W).Then UDF is used to verify the candidates in order to obtain the similar prefixes. This method is not efficient for long strings, especially for large edit distance threshold because as it requires very large storage space for its neighborhoods. Thus

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

this method is only effective for strings with short length.

2.3 Incremental Computation Method

It is trie structure to incrementally compute similar prefixes [4][7]. Effective index structure is created using the auxiliary tables and in order to achieve high interactive speed pruning techniques are used. When the user enters the keyword and submits a new query. S_t^w table is used to compute the similar prefixes of the keyword in $S_t^{w'}$ by joining the similar prefix table $S_t^{w'}$ and the prefix table P_T and compute the answers of input string w using the inverted index table I_T.

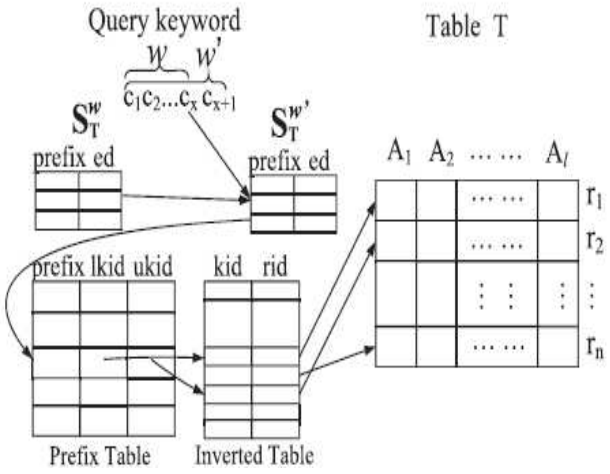

Figure 5: Character-level incremental method

IV. Experimental Results and Analysis

In this section, three methods are compared, namely Q-gram method, Neighborhood generation method, incremental method. Datasets with four different sizes (mb) are used for experimental analysis. Graph representation in figure 6 shows the execution time of each method, where x axis shows the dataset size(mb) and y axis indicates time(ms).

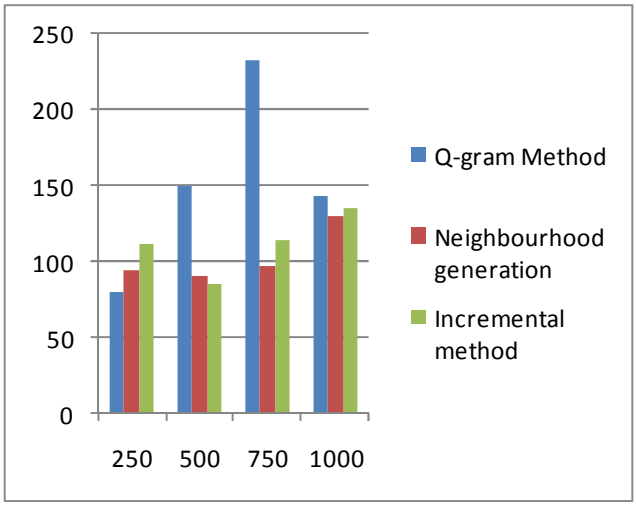

Figure 6: Execution time analysis

V. Conclusion

In this Paper, we studied the problem of using the SQL to support the search-as-you-type in databases. In order to maintain prefix matching, supplementary tables are used as index structures. Main Challenge is to extract the DBMS functionalities to meet the high performance to attain an interactive speed. Other than the Prefix matching, techniques are extended for fuzzy queries to improve the query performance. Multi keyword searching can also be performed by the fuzzy search. However, there are several problems for search as you type in databse. For example, one is how to support queries efficiently and other is how to support multiple tables.

References

1. Guoliang Li, JianhuaFeng, "Supporting Search-As-You-Type Using SQL in Databases", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 25, NO. 2, FEBRUARY 2013
2. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword Searching and Browsing in Data Bases Using Banks," Proc. 18th Int'l Conf. Data Eng. (ICDE '02), pp. 431- 440, 2002.
3. Hristidis and y. Papakonstantinou, "DISCOVER: Keyword Search In Relational Data Bases," PROC. 28TH INT'L CONF. Very Large Data Bases (VLDB '02), PP. 670-681, 2002.
4. S. Ji, G. Li, C. Li, and J. Feng, "Efficient Interactive Fuzzy Keyword Search," Proc. 18th ACM SIGMOD Int'l Conf. World Wide Web (WWW), pp. 371-380, 2009.
5. L. Gravano, P.G. Ipeirotis, H.V. Jagadish, N. Koudas, S.Muthukrishnan, and D. Srivastava, "Approximate String Joins in a Data Base (Almost) for Free," Proc. 27th Int'l Conf. Very Large Data Bases (VLDB '01), pp. 491-500, 2001.
6. J. Wang, G. Li, and J. Feng, "Trie-Join: Efficient Trie-Based String Similarity Joins with Edit-Distance Constraints," Proc. VLDB Endowment, vol. 3, no. 1, pp. 1219-1230, 2010.
7. S. Chaudhuri and R. Kaushik, "Extending Autocompletion to Tolerate Errors," Proc. 35th ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '09), pp. 433-439, 2009.
8. C. Li, J. Lu, and Y. Lu, "Efficient Merging and Filtering Algorithms for Approximate String Searches," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08), pp. 257-266, 2008.
9. H. Lee, R.T. Ng, and K. Shim, "Extending Q-Grams to Estimate Selectivity of String Matching with Low Edit Distance," Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB '07), pp. 195-206, 2007.
10. G. Li, S. Ji, C. Li, and J. Feng, "Efficient Type-Ahead Search on Relational Data: A Tastier Approach," Proc. 35th ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '09), pp. 695-706, 2009.

Parvathi R, is currently doing his MTech in Computer Science and Engineering at Sree Chitra Thirunal College of Engineering under Kerala University, Trivandrum, Kerala, India. Parvathi received her B Tech Degree in Information Technology from Mohandas College of Engineering under Kerala University, Kerala, India in 2010

Syama R, is working as Assistant professor at the department of computer science and engineering, Sree Chitra Thirunal College of Engineering, Trivandrum, Kerala.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.