International Journal of Science Engineering and Advance Technology

A Study To Support First-N Queries And Incremental Updates To Answer Multi Keyword Queries

[1]Ala Rajitha, [2]Mr. Fasi Ahmed Parvez
[1]arajitha222@gmail.com
1,2 Balaji Institute of Technology & Science Narsampet warangal

Abstract:
Most search engines and online search forms maintain auto completion which demonstrates suggested queries or even answers on the fly as a user types in a keyword query character by character. As many search systems accumulate their information in a backend relational 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. Still not all databases provide this feature. For this reason we study new methods that can be used in all databases. One approach is to expand a separate application layer on the database to construct indexes and execute algorithms for answering queries. While this approach has the benefit 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 permit developers to implement new functionalities to a DBMS. This approach is not possible for databases that do not provide such an extender interface such as MySQL. Because 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 particularly those implemented in C/C++ could cause severe dependability and security problems to database engines.
Keywords: Search-as-you-type, databases, SQL, fuzzy search.

Introduction:
A search-as-you-type system work out answers on-the-fly as a user types in a keyword query character by character. We study how to support search-as-you-type on data exist in a relational DBMS. We focus on how to support this type of search using the native database language SQL. A main confront is how to leverage existing database functionalities to meet the high presentation requirement to attain an interactive speed. We study how to use auxiliary indexes stored as tables to increase search performance. We present solutions for both single-keyword queries and multi keyword queries and develops novel methods for fuzzy search using SQL by allowing mismatches between query keywords and answers. We present techniques to answer first-N queries and discuss how to support updates resourcefully. Experiments on huge real data sets show that our techniques enable DBMS systems on a product computer to support search-as-you-type on tables with millions of records. Many information systems these days get better user search experiences by providing immediate feedback as users formulate search queries. The instant feedback helps the user not only in formulating the query but also in understanding the underlying data. This type of search is usually called search-as-you-type or type-ahead search.

Related Work:
There have been recent studies to support well-organized estimated string search which given a set of strings and a query string all strings in the set that are comparable to the query string. Many studies used gram-based index structures to support approximate string search. The experiments showed that these approaches are not as competent as trie-based methods for fuzzy search. An auto completion system can forecast a word or phrase that a user may type in next based on the partial string the user has already typed. Nandi and Jagadish studied phrase prediction which took the query string as a single keyword and computed all sentences with a phrase of the query string. Bast et al. proposed HYB indexes to support search-as-you-type. Ji et al. extended auto completion to support fuzzy full-text instant search. Chaudhuri and Kaushik studied how to find comparable strings interactively as users' type in a query string and they did not study how to support multi keyword queries.

Existing System:
The existing system data provider a member would not suppose free or complete sharing with others because its data is lawfully private or commercially

proprietary or both. As a substitute it is required to hold full control over the data and access to the data. In the meantime as a consumer a health provider requesting data from other providers wait for to protect private information e.g. requestor's identity, interests in the querying process.

Disadvantages:

The database with the tuple data does not be maintained confidentially. The existing systems another person to easily access database.

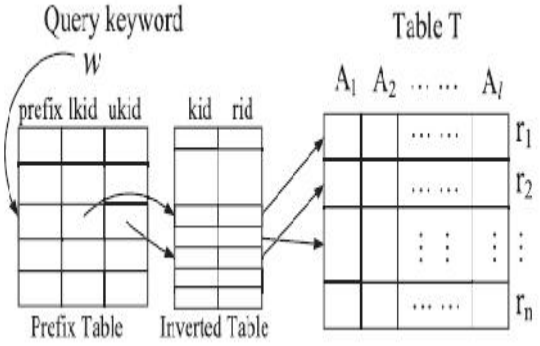
Proposed System:

Co-Ordinator and broker carry out the main vital role and pass the data between the two users. The data which give in from one user will be viewed by another user in the privacy and secured manner. All the data will be saved in the xml type travels in the XPATH. In proposed system we don't have a communication between the two end users.

Advantages:

High security and data privacy between users. In the System all the data will be saved in the xml document with high security.

Inverted-Index Table And Prefix Table:



SELECT $T.*$ FROM P_T, I_T, T
WHERE P_T.prefix = "w" AND
$P_T.ukid \geq I_T.kid$ AND $P_T.lkid \leq I_T.kid$ AND
$I_T.rid = T.rid$.

For example, assuming a user types in a partial query "sig" on table dblp (Table 1), we issue the following SQL:

SELECT dblp.* FROM P_{dblp}, I_{dblp}, dblp
WHERE P_{dblp}.prefix = "sig" AND
$P_{dblp}.ukid \geq I_{dblp}.kid$ AND $P_{dblp}.lkid \leq I_{dblp}.kid$ AND
$I_{dblp}.rid$ = dblp.rid,

To respond the SQL query resourcefully we produce built-in indexes on attributes prefix, kid and rid. The SQL could first utilize the index on prefix to find the keyword range and then calculate the answers using the indexes on kid and rid. For illustration assuming a user types in a partial query "sig" on table dblp we first get the keyword range of "sig" using the index on prefix and then find records r3, r6, and r9 using the index on kid.

Co-Ordinator Module:

The co-coordinator executes the global service between the two end users. To begin with the data owner needs to submit the details of the patient in the server. Data Users needs to look for the data which is stored in the servers and they give request for the data and the co-Ordinator sends the key to the Data users and the Data will be passed by the broker Way.

Broker Module:

The broker acts upon the role who can act between the Co-coordinator and the data Users. The request which is all submitted from the data user will be established and thus it will be passed to the co-coordinator. The data will be passed from the co-coordinator and thus it will be submitted to the End Users (Data Users).

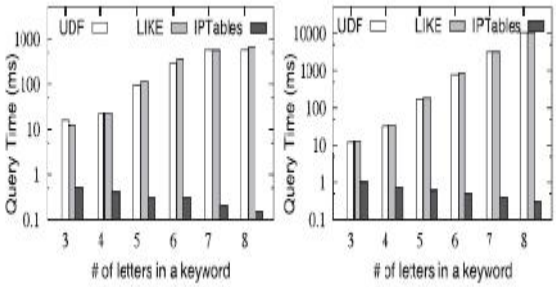
User Module:

The Users are classified into two types they are Data Users and Data Owner depends on the constraint the data will be passed to the Co-coordinator. The co-coordinator pass the details via broker and the data will be ensured with the secret key and thus it will display for the users.

Admin Module:

To organize the database based on the patient and doctor details and records. The admin needs to register and register the Organization and Users Forms.

Experiment Results:



We see that both the UDF-based method and the LIKE based method had a low search performance as they needed to scrutinize records. IP Tables accomplish a high performance by using indexes. As the keyword length increased the presentation of the first two methods decreased since the keyword became more selective and the two methods needed to examine more records in order to find the same number (N) of answers. As the keyword length increased IP Tables had a higher

performance since there were fewer complete keywords for the query and the query needed fewer join operations.

Conclusion:

To maintain prefix matching we proposed solutions that use supplementary tables as index structures and SQL queries to support search-as-you-type. We comprehensive the techniques to the case of fuzzy queries and proposed various techniques to improve query performance. We proposed incremental-computation techniques to answer multi keyword queries, and studied how to support first-N queries and incremental updates. Our 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. There are several open problems to support search-as you- type using SQL. One is how to support ranking queries resourcefully. Another one is how to support multiple tables. We studied the difficulty of using SQL to support search-as-you-type in data bases. We focused on the challenge of how to leverage existing DBMS functionalities to meet the high-performance requirement to achieve an interactive speed.

References:

[1] S. Agrawal, K. Chakrabarti, S. Chaudhuri, and V. Ganti, "Scalable Ad-Hoc Entity Extraction from Text Collections," Proc. VLDB Endowment, vol. 1, no. 1, pp. 945-957, 2008.

[2] S. Agrawal, S. Chaudhuri, and G. Das, "DBXplorer: A System for Keyword-Based Search over Relational Data Bases," Proc. 18th Int'l Conf. Data Eng. (ICDE '02), pp. 5-16, 2002.

[3] A. Arasu, V. Ganti, and R. Kaushik, "Efficient Exact Set-Similarity Joins," Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB '06), pp. 918-929, 2006.

[4] H. Bast, A. Chitea, F.M. Suchanek, and I. Weber, "ESTER: Efficient Search on Text, Entities, and Relations," Proc. 30th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '07), pp. 671-678, 2007.

[5] H. Bast and I. Weber, "Type Less, Find More: Fast Autocompletion Search with a Succinct Index," Proc. 29th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '06), pp. 364-371, 2006.

[6] H. Bast and I. Weber, "The Complete Search Engine: Interactive, Efficient, and Towards IR & DB Integration," Proc. Conf. Innovative Data Systems Research (CIDR), pp. 88-95, 2007.

[7] R.J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all Pairs Similarity Search," Proc. 16th Int'l Conf. World Wide Web (WWW '07), pp. 131- 140, 2007.

[8] 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.

[9] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin, "An Efficient Filter for Approximate Membership Checking," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '08), pp. 805- 818, 2008.

[10] S. Chaudhuri, K. Ganjam, V. Ganti, R. Kapoor, V. Narasayya, and T. Vassilakis, "Data Cleaning in Microsoft SQL Server 2005," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '05), pp. 918-920, 2005.

[11] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, "Robust and Efficient Fuzzy Match for Online Data Cleaning," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03), pp. 313- 324, 2003.

[12] S. Chaudhuri, V. Ganti, and R. Kaushik, "A Primitive Operator for Similarity Joins in Data Cleaning," Proc. 22nd Int'l Conf. Data Eng. (ICDE '06), pp. 5-16, 2006.

[13] S. Chaudhuri, V. Ganti, and R. Motwani, "Robust Identification of Fuzzy Duplicates," Proc. 21st Int'l Conf. Data Eng. (ICDE), pp. 865- 876, 2005.

[14] 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.

[15] B.B. Dalvi, M. Kshirsagar, and S. Sudarshan, "Keyword Search on External Memory Data Graphs," Proc. VLDB Endowment, vol. 1, no. 1, pp. 1189-1204, 2008.

[16] B. Ding, J.X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding Top-K Min-Cost Connected Trees in Data Bases," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE '07), pp. 836-845, 2007.

[17] 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.

[18] M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava, "Fast Indexes and Algorithms for Set Similarity Selection Queries," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08), pp. 267-276, 2008.