International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 2 Issue: 12 3987 - 3989

Survey on Type-Ahead Search

Mr. Salunke Shrikant Dadasaheb
Department of Computer Engineering,
DGOI,FOE,Daund.
Pune Maharashtra India
shrikantsalunke25@gmail.com

Prof. Bere Sachin Sukhadeo
Department of Computer Engineering,
DGOI,FOE,Daund.
Pune Maharashtra India

Mr. Rajpure Amol Subhash
Department of Computer Engineering,
DGOI,FOE,Daund.
Pune Maharashtra India
amolrajpure9@gmail.com

Mr. Tirgul Aniket Nandkumar
Department of Computer Engineering, DGOI,FOE,Daund.
Pune Maharashtra India
anikettirgul333@gmail.com

Abstract - A search-as-you-type system calculates answers on the fly as a user types in a keyword query character by character. We want to study how to support search as you type on data residing in a relational DBMS. We concentrate on how to support this type of search using the native database language and SQL. A main task that tests is how to leverage existing database functionalities to meet the high performance requirement to achieve an interactive speed. We studied how to use auxiliary indexes stored as tables to increase search performance. We presented solutions for both single keyword queries and multi keyword queries and developed novel techniques for fuzzy search using SQL by allowing mismatches between query keywords and answers. We extended 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 and real data sets showed that the proposed techniques can enables DBMS systems to support search as you type on large tables.

Keywords: *Fuzzy search, DBMS, SQL, Keyword query, incremental computation.*

*****

I. INTRODUCTION

Presently information system upgraded with the help of direct feedback. Auto completion of search queries promoted by search engines or online search. Search engines shows answers “on the fly” when user type query character on the keyboard. The web search interface at Netflix also support for search. For example if user type partial query “ university “, the system shows name of universities such as University of Pune, University of Mumbai, University of Toronto so on. So this quick feedback helps user to understand data formulate queries such search known a search-as-you-type or type-ahead search.

Most of the knowledge about search system kept in a backend relational DBMS. It is prominent aspect to find out how Type-ahead support data stored in DBMS? Now-a-days oracle and SQL server are help prefix search, but problem regarding these servers that not all database provide this features. So we need ever new system which support all database. The prominent attempt had been made to develop separate application layer on the database which help to generate interces and implement algorithms for solving queries. Even though this method has high performance but suffer disadvantage such as duplicating data and indexes, increased hardware costs. The current approaches such as DB2, Extenders Informix Data blades, Microsoft SQL server Common Language Runtime (CLR) integration and Oracle cartridges use database extenders. These approaches implement new functionalities to a DBMS. This approach is nor practicable for databases because this unable to provide extender interface, such as MySQL. So it is need to use proprietory interfaces generated by database vendors. Hence a solution for one database may not be portable to another thus an extender – based solution implemented with C/C++ could cause serious reliability and security problems to database engines.

The prominent question arises about this idea is, It is measurable and feasible? By using interactive search interface can SQL provide high performance? The research has proved that each question must be must be answered in 100 milliseconds. DBMS system not only structured for queries of keyboard but also support Type-ahead search with join operations but it may be expensive.

When we consider two features of Search- as-you –type such as multi keyword search and fuzzy search, the scalability became rather more unclear. In multi keyword search, it permit a query sting to have multiple keyword, and search records to match these keywords , even though keyword s appear at different places. For example If user type in a query “ Privacy mining rak” so search book by “Vijay Tendulkar” with title including the keywords “ Privacy” and “Mining ” even though keywords at distinct places . In fuzzy search, it permit small spelling mistake such as “Tendakar” instead of “Tendulkar”. So this principle upgrades search experiences of user and support to do Type-ahead inside DBMS systems.

In present paper, we focus upon how to help Type-ahead search by native query language (SQL). It means SQL search queries when user types in keywords character by character. Our prime aim is to use the built-in query engine of the database system as much as possible. So with this we can minimize programming efforts to help Type-ahead search. The solution generated on one database using Standard SQL methods is movable to other databases which support the same standard. To support this prefix matching we proposed solutions that use auxiliary tables as index structures and SQL queries to support Type-ahead search. We lengthen fuzzy query case and proposed various techniques to upgrade query performance we proposed incremental computation techniques to solve multi-keyboard queries. We studied how to support first N-queries and incremental updates. Thus our result proves that DBMS system support search-as –you-type on large table. The same observations done by Gravano et al and testes et al which use SQL to support similarity join in databases.

II. LITERATURE SURVEY

In this section we are presenting the different methods those are presented to solve keyword search:

• S. Agrawal, K. Chakrabarti, S. Chaudhuri, and V. Ganti, Scalable Ad-Hoc Entity Extraction from Text Collections [1]: In this paper, the author introduces the ad-hoc entity extraction task where entities of interest are constrained to be from a list of entities that is specific to the task. In such scenarios, traditional entity extraction techniques that process all the documents for each ad-hoc entity extraction task can be significantly expensive. In that paper, they proposed an efficient approach that leverages the inverted index on the documents to identify the subset of documents relevant to the task and processes only those documents. The author demonstrates the efficiency of their techniques on real datasets.

• S. Agrawal, S. Chaudhuri, and G. Das, DBXplorer: A System for Keyword-Based Search over Relational Data Bases [2]: Internet search engines have popularized the keyword based search paradigm. While traditional database management systems offer powerful query languages, they do not allow keyword-based search. In this paper, we discuss DBXplorer, a system that enables keyword based search in relational databases. DBXplorer has been implemented using a commercial relational database and web server and allows users to interact via a browser front-end. We outline the challenges and discuss the implementation of their system including results of extensive experimental evaluation.

• Arasu, V. Ganti, and R. Kaushik, Efficient Exact Set-Similarity Joins [3]: Given two input collections of sets, a set-similarity join (SSJoin) identifies all pairs of sets, one from each collection, that have high similarity. Recent work has identified SSJoin as a useful primitive operator in data cleaning. In this paper, the author proposes new algorithms for SSJoin. Their algorithms have two important features: They are exact, i.e., they always produce the correct answer, and they carry precise performance guarantees. The author believes their algorithms are the first to have both features; previous algorithms with performance guarantees are only probabilistically approximate. They demonstrate the effectiveness of their algorithms using a thorough experimental evaluation over real-life and synthetic data sets.

• H. Bast, A. Chitea, F.M. Suchanek, and I. Weber, ESTER: Efficient Search on Text, Entities, and Relations [4]: The author presents ESTER, a modular and highly efficient system for combined full-text and ontology search. ESTER builds on a query engine that supports two basic operations: prefix search and join. Both of these can be implemented very efficiently with a compact index, yet in combination provide powerful querying capabilities. The author show how ESTER can answer basic SPARQL graph pattern queries on the ontology by reducing them to a small number of these two basic operations. ESTER further supports a natural blend of such semantic queries with ordinary full text queries. Moreover, the prefix search operation allows for a fully interactive and proactive user interface, which after every keystroke suggests to the user possible semantic interpretations of his or her query, and speculatively executes the most likely of these interpretations. As a proof of concept, they applied ESTER to the English Wikipedia, which contains about 3 million documents, combined with there cent YAGO ontology, which contains about 2.5 million facts. For a variety of complex queries, ESTER achieves worst case query processing times of a fraction of a second, on a single machine, with an index size of about 4 GB.

• H. Bast and I. Weber, Type Less, Find More: Fast Auto completion Search with a Succinct Index [5]: The author consider the following full-text search auto completion feature. Imagine a user of a search engine typing a query. Then with every letter being typed, user would like an instant display of completions of the last query word which would lead to good hits. At the same time, thebe shits for any of these completions should be displayed. Known indexing data-structures that apply to this problem either incur large processing times for a substantial class of queries, or they use a lot of space. They present a new indexing data-structure that uses no more space than a state-of-the art compressed inverted index, but that yields an order of magnitude faster query processing times. Even on the large TREC Terabyte collection, which comprises over 25 million documents, they achieve on a single machine and with the index on disk, average response time of one tenth of a second. They have built a full-edged, interactive search engine that realizes the proposed auto completion feature combined with support for proximity search semi-structured (XML) text, sub word.

III. CONCLUSION

In this paper, we studied the problem 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. To support prefix matching, we proposed solutions that use auxiliary tables as index structures and SQL queries to support search-as-you-type. We extended the techniques to the case of fuzzy queries, and proposed various techniques to improve query performance.

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] 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.