International Journal of Scientific Engineering and Research (IJSER)
ISSN (Online): 2347-3878
Index Copernicus Value (2015): 62.86 | Impact Factor (2015): 3.791

Different Techniques of On-the-Fly Search on SQL Relational Database: Survey

Ansari Aadil S.[1], Ujwala M. Patil[2]

[1]Computer Science and Engineering, R. C. Patel Institute of Technology, Shirpur, India
[2]Associate Professor, Dept of Computer Engineering, R. C. Patel Institute of Technology, Shirpur, India

Abstract: ***Searching in a database was harder task in earlier times. In order to make search faster and easy, different techniques were developed. Main aim was to enter a query, even by an ignorant user so that he can get the results easily and also the cost of retrieval should be low. So we have selected some of the techniques and evaluated them. When the size of the increases it becomes even tough to retrieve the results. An Optimal search technique must render the requested data in a stipulated time based on the user query. Since large amount of data has to be processed, there has to be some order for ranking the queries to make the search a more efficient one. On-the-fly search is a method which gives answers when user types in a keyword query, one after the another character by character. This system which retrieves answers when a user types a keyword query is search-as-you-type The survey is based on the different techniques for search-as-you-type on the data present in a relational DBMS. The main aim of different methods is how to implement search as-you-type using the native language for database mainly SQL. Improving actual database functionalities to achieve improved performance in order to get an interactive speed is the main problem. Performance of the search can be increased using auxiliary indexes.***

Keywords: component, DBMS, SQL, On-the-fly

1. Introduction

Extracting knowledge from the large amounts of data is called as Data mining. In Relational database, information and multiple dataset are stored. These Datasets are represented in tables and records through rows and columns. Currently keyword search handle with single databases. Keyword Search is latest technique in database search. User simply inserts a keyword for looking out and gets a result associated with that keyword. The Solution of the tuples which are connected to database keys like primary key and foreign keys lies in keyword search on relational dataset. Traditional information systems return answer only after the submission of the entire query. When the user does not have enough knowledge about data lying below they often feel "left blind", and have to use the luck approach for collecting information.

New approach is to develop a separate application layer on the database to construct indexes, and implement algorithms for answering queries. Conventional approaches have the advantage of getting a high performance, but its major drawback is duplicating indexes and data which results in additional hardware costs. User search experiences are improved by the system which provides instant reaction as users prepares the search queries. Autocompletion is supported by nearly all search engines and online search forms, which shows recommended queries or answers "on the fly" as a user types in a keyword query character by character. The user searches something in the system. The database gives the results with already searched or with the queries mostly searched. . Consider an example Netflix, Inc. is a provider of on-demand Internet streaming media available to viewers in all of parts of Europe, South America, North America, and of flat rate DVD by-mail in the United States, where mailed DVDs are sent via Permit Reply Mail. Consider an example in Netflix Database, when the user searches videos, the user will get help from database to understand the actual query of the user.

2. Literature Survey

We have studied different techniques to search on SQL Relational databases below are some of them.

Sr No	*Author*	*Title*	*Publication*	*Year*	*Research*
1	A. Nandi and H.V. Jagadish	"Effective Phrase Prediction"	VLDB	2007	They Studied the problem of autocompletion not just at the level of a single "word", but at the level of a multi-word "phrase". They found couple of major challenges, one is that the number of phrases i.e. both the number possible and the number actually observed in a corpus is combined larger than the number of words; other is that a "phrase", not like a "word", does not have a well-defined limits, so that the autocompletion system has to decide not just what to predict, but also how far it can go. They introduced a Fussy Tree structure to address the first challenge and the concept of a significant phrase to address the second [1].
2.	H. Bast, A. Chitea, F.M. Suchanek, and I. Weber	"ESTER: Efficient Search on Text, Entities, and Relations"	SIGIR	2007	They proposed a modular and highly efficient system for combined full-text and ontology search. Their system is a query engine that supports basic operations of prefix search and join. These can be implemented very efficiently with a compact index, and in combination provide powerful

					querying abilities. They demonstrated how they can answer basic graph pattern queries by reducing them to a small number of these basic operations. Natural blend of such semantic queries with ordinary full-text queries was supported by their system. Moreover, the prefix search operation allows for a fully interactive and proactive user interface, which suggests to the user possible semantic interpretations of his query, and speculatively executes the most likely of these interpretations [2].
3.	H. Bast and I. Weber	"Type Less, Find More: Fast Autocompletion Search with a Succinct Index,"	SIGIR	2006	They Proposed an Indexing data structure which uses a state of-the-art compressed inverted index and yields an order of magnitude faster query processing times. They even achieve large TREC Terabyte collection, which comprises over 25 million documents, on a single machine and with the index on disk, average response times of one tenth of a second. They have built a full-edged, interactive search engine that realizes the proposed autocompletion feature combined with support for proximity search, semi-structured (XML) text, sub word and phrase completion, and semantic tags. They had introduced an autocompletion feature for full text search, and presented a new compact indexing data structure for supporting this feature with very fast response times [3].
4.	S. Ji, G. Li, C. Li, and J. Feng	"Efficient Interactive Fuzzy Keyword Search,"	SIGMOD	2009	They Proposed a information-access paradigm, called "interactive, fuzzy search," the system which searches the underlying data "on the fly" as the user types in query keywords. Autocomplete interfaces was extended by allowing keywords to appear in multiple attributes (in an arbitrary order) of the underlying data; and finding relevant records that have keywords matching query keywords approximately. Their framework allowed users to explore data as they type. They studied research challenges for large amounts of data. Various incremental search algorithms were developed, using previously computed and cached results in order to achieve an interactive speed [4].
5.	S.Chaudhuri and R. Kaushik,	"Extending Autocompletion to Tolerate Errors"	SIGMOD	2009	They Proposed a method capture input typing errors via edit distance. They show that a native approach of invoking an offline edit distance matching algorithm at each step performs poorly and present more efficient algorithms. Their study demonstrated the effectiveness of algorithms. However they focused on the algorithmic aspects of error-tolerant autocompletion which are relevant regardless of the specific application. Similarity function like, issue of performing error tolerant autocompletion also needed to be addressed [5].
6.	G. Li, S. Ji, C. Li, and J. Feng,	"Efficient Type-Ahead Search on Relational Data: A Tastier Approach"	SIGMOD	2009	They Proposed a novel approach to keyword search in the relational world, called Tastier. A Tastier system can bring instant gratification to users by supporting type-ahead search, it find answers "on the fly" as the user types in query keywords. Their main challenge is how to achieve a high interactive speed for large amounts of data in multiple tables, so that a query can be answered efficiently within milliseconds. To find appropriate results on-the-fly by joining tuples in the database they developed an index structures and algorithms. They improved the query performance by grouping relevant tuples and pruning irrelevant tuples efficiently also proposed a method to answer a query efficiently by predicting highly relevant complete queries for the user. They developed a graph-partition-based method and a query prediction technique to improve search efficiency [6].
7.	L. Qin, J.X. Yu, and L. Chang,	"Keyword Search in Data Bases: The Power of Rdbms"	SIGMOD	2009	They Proposed a system by using SQL to compute all the interconnected tuple structures for a given keyword query. To control the size of the structures they used three different types of interconnected tuple structures. The main idea behind their approach is tuple reduction. A middleware free approach to compute such m-keyword queries on RDBMSs using SQL only. Their middleware free approach makes it possible to fully utilize the functionality of RDBMSs to support keyword queries in the same framework of RDBMSs [7].
8.	G. Li, J. Fan, H. Wu, J. Wang, and J. Feng	"Dbease: Making Data Bases User-Friendly and Easily Accessible"	CIDR	2011	They Proposed a search method DBease which make databases user-friendly and easily available. It allows users to explore data "on the fly" as they type in keywords, even in the presence of minor errors. DBease made databases user friendly and easily accessible. They developed various techniques to improve keyword search, form-based search, and SQL-based search for enhancing user experiences. Search as-you-type help user's on-the-fly to explore the underlying data. Form-based search can provide on-the-fly faceted search. SQL suggestion can help various users to formulate SQLs based on limited keywords [8].
9.	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"	VLDB	2001	They Proposed a method for building approximate string join capabilities for commercial databases by exploiting facilities. Their technique relies on matching short substrings of length called q-grams at the core, and taking into account both positions of individual matches and the total number of such matches. Their approach applies to both full string matching and substring matching, with a different possibility of edit distance functions. They demonstrated experimentally the benefits of our technique over the direct use of UDFs, using commercial database

					systems and real data [9].
10.	S. Chaudhuri, K. Ganjam, V. Ganti, R. Kapoor, V. Narasayya, and T. Vassilakis,	"Data Cleaning in Microsoft SQL Server 2005"	SIGMOD	2005	They Proposed couple of new data cleansing operators, one is Fuzzy Lookup and other one is Fuzzy Grouping, it address problems in a scalable and domain-independent manner. These operators were implemented within Microsoft SQL Server 2005 Integration Services. They demonstrated their functionality and highlighted multiple real world scenarios in which they can be used to achieve high data quality [10].
11.	S. Agrawal, K. Chakrabarti, S. Chaudhuri, and V. Ganti	"Scalable Ad-Hoc Entity Extraction from Text Collections,"	VLDB	2008	They Proposed the entity extraction task where entities of interest are bounded 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. 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. They demonstrated the efficiency of our techniques on real datasets. Their main observation is that in many scenarios, there exists a significant overlap of tokens among entities they exploit this observation to develop techniques to efficiently identify a set of documents which need to be processed for entity extraction. Through an extensive empirical evaluation using real datasets, and demonstrated that their techniques result in significant improvements over prior approaches [11].
12.	G. Li, J. Feng, X. Zhou, and J. Wang	"Providing Built-in Keyword Search Capabilities in Rdbms"	VLDB	2011	They Proposed a concept called Compact Steiner Tree (CSTree), which can be used to approximate the Steiner tree problem for answering top-k keyword queries efficiently. A novel structure- aware index, together with an effective ranking mechanism for fast, progressive and accurate retrieval of top-k highest ranked CSTrees. Their proposed techniques can be implemented using a standard relational RDBMS to benefit from its indexing and query-processing capability. This techniques was implemented in MYSQL, which can provide built-in keyword-search capabilities using SQL. The experimental results showed a significant improvement in both search efficiency and result quality comparing to existing state of- the-art approaches [12].

3. Conclusion

Keyword search in different scenarios enables information discovery without requiring from the user to know the schema of the database. Undergoing with the above survey we draw a conclusion, to develop a separate application layer on the database to construct indexes, and implement algorithms for answering queries. In this article, we studied the problem of using the SQL to support the system search-as-you-type in data bases and also studied 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 can propose a solution that uses the auxiliary tables as index structures and SQL queries to support the search-as-you-type. We can enhance the techniques in the case of fuzzy queries, and can propose various techniques to improve the query performance. We can also propose multi keyword queries search, and study how to support first-N queries and the incremental updates.

References

[1] A. Nandi and H.V. Jagadish, "Effective Phrase Prediction," Proc.33rd Int'l Conf. Very Large Data Bases (VLDB '07), pp. 219-230, 2007.

[2] 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.

[3] 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.

[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] 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.

[6] 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.

[7] L. Qin, J.X. Yu, and L. Chang, "Keyword Search in Data Bases: The Power of Rdbms," Proc. 35th ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '09), pp. 681-694, 2009.

[8] G. Li, J. Fan, H. Wu, J. Wang, and J. Feng, "Dbease: Making Data Bases User-Friendly and Easily Accessible," Proc. Conf. Innovative Data Systems Research (CIDR), pp. 45-56, 2011.

[9] 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.

[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. 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.

[12] G. Li, J. Feng, X. Zhou, and J. Wang, "Providing Built-in Keyword Search Capabilities in Rdbms," VLDB J., vol. 20, no. 1, pp. 1-19, 2011.

Author Profile

Aadil S. Ansari received the B.E. degree from NMU Jalgaon, Maharashtra, and M.E. degree in Computer Science Engineering pursuing from R. C. Patel Institute of Technology, Maharashtra, India respectively. From 2014-present, working as a lecturer in MMANTC, Mansoora Malegaon, in the Applied Science Department.

Mrs Ujwala M. Patil, she is an Associate Professor in RCPIT, Shirpur, Maharashtra, India in department of Computer Engineering.