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

Ansari Aadil S.¹, Ujwala M. Patil²

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

<table>
<thead>
<tr>
<th>Sr No</th>
<th>Author</th>
<th>Title</th>
<th>Publication</th>
<th>Year</th>
<th>Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A. Nandi and H.V. Jagadish</td>
<td>“Effective Phrase Prediction”</td>
<td>VLDB</td>
<td>2007</td>
<td>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 a 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 Fuzzy Tree structure to address the first challenge and the concept of a significant phrase to address the second [1].</td>
</tr>
<tr>
<td>2</td>
<td>H. Bast, A. Chitea, F.M. Suchanek, and I. Weber</td>
<td>“ESTER: Efficient Search on Text, Entities, and Relations”</td>
<td>SIGIR</td>
<td>2007</td>
<td>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</td>
</tr>
<tr>
<td>No.</td>
<td>Authors</td>
<td>Title</td>
<td>Conference/Volume</td>
<td>Year</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>-------</td>
<td>-------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>S. Chaudhuri and R. Kaushik</td>
<td>“Extending Autocompletion to Tolerate Errors”</td>
<td>SIGMOD</td>
<td>2009</td>
<td></td>
</tr>
</tbody>
</table>

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 have 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].

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].

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 their technique over the direct use of UDFs, using commercial database

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 have 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].

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].

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 their technique over the direct use of UDFs, using commercial database

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].

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 their technique over the direct use of UDFs, using commercial database

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 have 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].

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].

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 their technique over the direct use of UDFs, using commercial database

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].

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 their technique over the direct use of UDFs, using commercial database
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


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].

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 is 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].

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].


Systems and real data [9].
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.