A Review on Search-As-You-Type Techniques in Relational Databases

Kaushik G. Vaghani  
PG-Scholar  
Computer Science & Engineering Dept  
Parul Institute of Technology  
Gujarat Technological University, India

Pratik A. Patel  
Assistant Professor  
Computer Science & Engineering Dept  
Parul Institute of Technology  
Gujarat Technological University, India

Abstract—Nowadays many of search engines in Internet have used keyword-based search concept. The RDBMS do not allow keyword-based search while they provides powerful query languages. Search systems which are already used that are based on keyword in RDBMS require users to submit a complete query to compute answers. Often users have partial knowledge about the data, and have to try and see the method to modify queries and find the answers. Search-as-you-type is a search system that allows the user to submit the prefix of the keyword and system will compute the answer as user type keyword character by character for data stored in a relational database management systems. A main challenge is how to influence existing database functionalities to achieve high-performance in searching speed. Various methods are used that supports the search-as-you-type functionality in the relational databases.

Keywords—Search-As-You-Type, Type-ahead Search, Keyword Search, Databases, SQL

I. INTRODUCTION

It has become extremely popular to provide users with flexible ways to search information over databases as simple as keyword search like Google search. Now a day, the relational databases are broadly used by applications from different areas and different search paradigms needed by different users. Knowledgeable users, such as database administrators, need a search paradigm that can provide them accurate and fully functional accessing abilities. In contrast, most inexperienced users, as casual Internet users expect to search databases as easily as possible. In addition, some users, such as systems analysts, call for new paradigms for search that influence usability and functionality.

Most information systems currently improve user search experiences by providing immediate feedback as users create search queries. Many search engines and online search forms support auto completion, which shows suggested queries or even answers “on the fly” as a user types in a keyword query character by character[5].

Keyword based searching is generally accepted way for querying relational databases and document systems[2]. Relational databases are generally searched using structured query languages. The user must know the database schema to be able to ask queries. Over the Web most search engines have popularized an alternative unstructured querying and browsing paradigm that is simple and user-friendly. Users type in keywords and follow hyperlinks to navigate from one document to another. No knowledge of the schema is required[4]. With the expansion of the WWW, there has been a fast increase in the number of users who need to access online databases without having a detailed knowledge of schema or query languages; even comparatively simple query languages designed for non-experts are too complicated for such users. Query languages for semi-structured/XML data are even more complex, increasing the impedance mismatch further.

There are basically two types of searching over relational Databases:

1. Keyword Search over Relational Databases

   Keyword search is process of searching the record from the relational databases that contains the full keyword provided by user.

2. Search-as-you-type in Relational Databases

   Search-as-you-type is process of providing the user with the answers “on the fly” as a user types a keyword character by character.

Keyword based search techniques that are used for searching information from collections of Web documents cannot be used on data that stored in relational databases[4]. In relational databases, information needed to answer a keyword query is often split across the tables/tuples, due to normalization.

Because most search systems keep their information in a backend relational database management system, a question arises obviously: how to bear search-as-you-type on the data stored in a DBMS? Some databases already support prefix search such as Oracle and SQL server, and we might use this feature to do search-as-you-type. But, all databases do not provide this quality. Because of this cause, we require new methods that can be used in all databases.
II. RELATED WORK

Search based on keyword is a well studied problem in the world of Internet search engines and text based documents. There have been many studies on keyword search in relational databases. Most of them employed tree based methods[10,12,13]. Other methods[11,14] generated answers composed of relevant tuples by generating and extending a candidate network following the primary-foreign-key relationship. The main objective to develop new techniques is all about turning intentions into actions in the blink of an eye. In online retail, having a quick and user-friendly website it helps to increase sales and conversion rates on merchant websites[6]. Search-as-you-type is a user-friendly feature which can reduce the efforts of users to process their queries by returning the results immediately as users type keyword character by character.

Keyword Search over relational databases uses DBXplorer[1], DISCOVER, BANKS[4] system that support keyword search on relational databases.

A main requirement of search-as-you-type on huge amounts of relational data is the need of a high interactive speed for searching. Every keystroke from a user can invoke a query to the system that needs to calculate the answers within the milliseconds. Many few techniques have been implemented for Search-as-you-type for the data stored in the relational database systems. Recently type-ahead search on relational databases uses TASTIER approach[2] and another techniques are by using existing functionality of query engine of database systems as much as possible and requires additional index structures stored as auxiliary tables[5].

III. DIFFERENT TECHNIQUES OF SEARCHING OVER RELATIONAL DATABASE MANAGEMENT SYSTEM

The various techniques that I have reviewed for searching records over relational database management system are explained in below sections.

1. No-Index Methods[5]
   One simple way to support search-as-you-type is to execute a SQL query that scan every record and checks whether the record is an answer of the query. There are two methods to do so :

   a. Calling User-Defined Functions (UDFs)
      We can add functions into databases to verify whether a record contains the query keyword.

   b. Using the LIKE predicate
      Databases provide a LIKE predicate that allow users to achieve keyword matching. We can use LIKE predicate to ensure whether a record contains the query keyword.

   Advantages: This two no-index methods do not require additional spaces in database.
   Disadvantages: They may not use because they need to scan all records in the table.

2. Index-Based Method[5]
   This method uses additional index structure stored as auxiliary tables to facilitate prefix search. This method can be used in all databases. A description of the additional auxiliary tables is as follows:

   Inverted-index table: Given a table T with assign unique ids to the keywords in table T, following their alphabetical order. Inverted-index table IT 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.

   Prefix table: Given a table T, for all prefixes of keywords in the table, a prefix table PT 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.

   Below fig 1 shows that how index based method works to find the records by using these additional tables.

![Fig 1: Using inverted-index table and prefix table to support search-as-you-type[5]](image-url)
Given a partial keyword \( w \), we first get its keyword range \([lkid, ukid]\) using the prefix table \( P_T \), and then find the records that have a keyword in the range through the inverted-index table \( I_T \) as shown in Figure 1. We use the following SQL to answer the prefix-search query \( w \):

\[
\text{SELECT } T.* \text{ FROM } P_T, I_T, T \\
\text{WHERE } P_T.prefix = "w" \text{ AND } \\
P_T.ukid >= I_T.kid \text{ AND } P_T.lkid <= I_T.kid \text{ AND } \\
I_T.rid = T.rid
\]

**Advantages:** This method does not require the scan of whole records in databases. So execution time for query is very small as compared to No-index based methods.

**Disadvantages:** This method does not support ranking queries as well it does not use for records that are stored in multiple tables.

3. **A Tastier Approach[2]**

This is a new approach to keyword search in the relational database management system, called Tastier. A Tastier system can convey instant satisfaction to users by supporting type-ahead search as it finds answers instantly as the user types in keyword keywords character by character. The major challenge is how to accomplish a high interactive speed for large amounts of data in multiple tables, so that a query can be answered efficiently within milliseconds. This technique offers efficient index structures and algorithms for finding relevant answers on-the-fly by joining tuples in the database. It uses a graph partition-based method to improve query performance by grouping relevant tuples and pruning irrelevant tuples efficiently. This also uses a technique to answer a query efficiently by predicting highly relevant complete queries for the user if user just inserts a prefix for the keyword.

This approach makes use of a new information-access concept that supports type-ahead search in relational databases with multiple tables. It uses efficient index structures and algorithms for incrementally computing answers to queries in order to achieve an interactive speed on large data sets. They have conducted a detailed experimental study of the algorithms. The results show the high efficiency of these new searching systems.

**Advantages:** It provides high interactive speed for searching and it supports multiple tables.

4. **DBXplorer Approach[1]**

Most of the search engines in the internet uses keyword-based search paradigm. While relational database management systems that offer powerful query languages do not allow keyword-based search. This technique introduces DBXplorer approach that allows keyword-based search in relational databases. It implemented DBXplorer using a commercial relational database and web server and enables users to interact via a browser front-end. Enabling keyword search in databases that do not need knowledge of the schema is a challenging job.

DBXplorer approach for searching keyword makes use of the symbol table that stores information at row level granularity, i.e., for each keyword they keep the list of rows that contains the keyword. This approach also describes the search component that answers keyword searches once the symbol table has been built. This paper only considers the exact match problem that is each keyword in a query must match the value of an attribute in a row of a table.

**Disadvantages:** It is not efficient to maintain the symbol table for the large amount of records stored in a relational database.

5. **Banks Approach[4]**

Today there has been a fast increase in the number of users who need to access online databases without having a detailed knowledge of the schema or of query languages. This technique describes BANKS, a system which enables keyword-based search on relational databases. BANKS enables users to extract information in a simple manner without any need for writing complex queries or any knowledge of the schema. User can get information by typing a few keywords. They model the database as a graph, with the tuples as nodes and cross references between them as edges.

BANKS allows query keywords to match data and metadata.

BANKS, is an integrated browsing and keyword querying system for relational databases. BANKS reduces the effort to a great extent involved in publishing relational data on the WWW and making it searchable. In future they want to extend the BANKS system to handle browsing and keyword searching of XML data. They also want to review the output, i.e., group the output tuples into sets that have the same tree structure, and allows the user to look for further answers with a particular tree structure.

**Advantages:** It greatly reduce the effort involved in publishing relational data on web and making it searchable.

**Disadvantages:** It is very difficult to model the large database as a graph.

**IV. CONCLUSIONS**

In this paper, various techniques for searching data from relational database are explained. Each technique uses their own algorithm for providing interactive speed to user as user types a keyword. Some of the approaches also provides search-as-you-type functionality as it provides answers on fly as user types keyword character by character. In order to
provide excellent keyword based search-as-you-type speed at a minimal time, all information systems need diagnostic procedures that are fast, efficient, and accurate. In addition, the procedures should not be requiring additional overhead in relational database management systems.

**ACKNOWLEDGMENT**

With the cooperation of my guide, I am highly indebted to Asst. Prof. Pratik Patel, for his valuable guidance and supervision regarding my topic as well as for providing necessary information regarding materials.

**REFERENCES**


[7] Daniel Suelmann, “Keyword-based Search in a Relational Database” Bachelor’s Thesis, Department of Information Science Faculty of Arts University of Groningen August 2009


**AUTHOR PROFILE**

**Kaushik Vaghani** received the Bachelor of Engineering degree in Computer Engineering from Sarvajanik College Of Engineering & Technology under Veer Narmad South Gujarat University in 2011, Gujarat. During 2011-2012, he worked for Triz Innovation Pvt Ltd. as a software developer. He is pursuing Master of Engineering degree in Computer Science & Engineering from Parul Institute of Technology under Gujarat Technological University.