**ABSTRACT**

A search-as-you-type system computes 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 residing in a relational DBMS. We focus on how to support this type of search using the native database language, SQL. A main challenge is how to leverage existing database functionalities to meet the high performance requirement to achieve 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 develop novel techniques 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 efficiently. Experiments on large, real data sets show that our techniques enable DBMS systems on a commodity computer to support search-as-you-type on tables with millions of records.

**SYSTEM ANALYSIS**

**EXISTING SYSTEM:**

Most 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.

Since many search systems store their information in a backend relational DBMS, a question arises naturally: how to support search-as-you-type on the data residing in a 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. However, not all databases provide this feature. For this reason, we study new methods that can be used in all databases. One approach is to develop a separate application layer on the database to construct indexes, and implement algorithms for answering queries.

**DISADVANTAGES OF EXISTING SYSTEM:**

* In an existing systems are not specially designed for keyword queries, making it more challenging to support search-as-you-type.
* SQL meet the high performance requirement to implement an interactive search interface.
* Some important functionality to support search-as-you-type requires join operations, which could be rather expensive to execute by the query engine.

**PROPOSED SYSTEM:**

In this paper, we develop various techniques to address these challenges. we propose two types of methods to support search-as-you-type for single-keyword queries, based on whether they require additional index structures stored as auxiliary tables.

We discuss the methods that use SQL to scan a table and verify each record by calling a user-defined function (UDF) or using the LIKE predicate. We study how to support fuzzy search for single-keyword queries.

We discuss a gram-based method and a UDF-based method. As the two methods have a low performance, we propose a new neighborhood-generation based method, using the idea that two strings are similar only if they have common neighbors obtained by deleting characters.

We extend the techniques to support multi-keyword queries. We develop a word-level incremental method to efficiently answer multi-keyword queries. Notice that when deployed in a Web application, the incremental-computation algorithms do not need to maintain session information, since the results of earlier queries are stored inside the database and shared by future queries.

**ADVANTAGES OF PROPOSED SYSTEM:**

* A main challenge is how to utilize the limited expressive power of the SQL language (compared with other languages such as C++ and Java) to support efficient search.
* We study how to use the available resources inside a DBMS, such as the capabilities to build auxiliary tables, to improve query performance.
* An interesting observation is that despite the fact we need SQL queries with join operations, using carefully designed auxiliary tables, built-in indexes on key attributes, foreign key constraints, and incremental algorithms using cached results, these SQL queries can be executed efficiently by the DBMS engine to achieve a high speed.