Combining Keyword Search and Forms for Ad Hoc Querying of Databases

SEMINAR REPORT

CSE 736 – DATABASE SEMINAR

Spring 2010

University at Buffalo

Feb 12th, 2010

Submitted By

Mohan Kumar Padmanabhan
3567-4710

Table of Contents

Overview

The main paper discussed in the seminar proposes a method to create structured queries on structural databases to search for required data. The motivation for the paper is to identify a method to build queries on structured data and fetch the desired search results in a systematic way without the requirement of knowledge of building structural queries. It describes a method where the system makes use of forms to input required parameters and build queries which are later sent to the database to get back the results. The forms are presented based on an initial set of keywords which are provided by the user.

In the paper they have described a step by step methodology to implement their idea. They created a subset of SQL, called SQL', which forms the basis of the queries they create which are later represented as forms. The SQL' is later expanded to the dataset they have considered to contain all types of queries within SQL' which will enable to fetch data from the database they have considered. These queries are stored in the back-end as query forms. They have mentioned that they had created scripts to convert the queries into forms. Later, when a user queries the database using keywords, the keywords are searched in the query forms and matching queries are fetched. These forms are displayed in a question format based on what the query forms return. Depending upon the requirement of the user, he/she selects the appropriate forms and enters the data that is required in the forms. On submitting the form, the query corresponding to the form, along with the values entered in the form, is executed and the results are provided to the user.

The paper considers a DBLife dataset containing 5 entity tables and 9 relationship tables. It also described three algorithms which are used to search the keywords entered within the query forms. Since the keywords may be data terms or query terms, a methodology is required to identify the corresponding query terms for the data terms and then search in the collection of query forms to select the forms. The first algorithm, called Double-Index OR (DI-OR), initially identifies schema terms for all the data specific terms and adds it to the set of the keywords entered. It then picks forms which contain any of the words present in this set. Since this was too inclusive, the second algorithm, called Double-Index AND(DI-AND) augmented the keywords entered with original query by generating all possible queries that result from replacing user-supplied data terms with schema terms. It uses AND semantics for each query, and return the union of the query results to the user. This algorithm created some forms which do not return any output, particularly, the searches involving data specific terms present only in the entity tables and not in relationship tables. So, the third algorithm, called Double-Index Join (DIJ) proposes a method to eliminate these dead forms. This algorithm searches the relationship tables which are identified to be related to an entity table containing a user-specified data term. Those combinations of entity-relationship tables which does not contain the data term in the relationship table are eliminated from the output set of forms.

The results consists of many parameters, most important of them being response time, number of forms returned, total time to fetch required records and ranking of forms. It was easily observable that DIJ performed the best, returning the optimum number of forms. Also, the response time of the entire approach was considerably very good. Grouping forms from same table and same query type puts the required form in better visibility to get selected and create the structured query using the forms. Experiment with additional forms proves the better functionality of the DIJ algorithm.

The second paper discusses about a different approach for viewing and browsing through a structured database. The Browsing ANd Keyword Searching (BANKS) interface provides a way in which all the tuples in a database are represented as nodes and the relationships between the tables as edges of a graph. The results are returned by node identification and a tree is returned as result. The identification of the tree (Steiner tree) based on a keyword is done by a heuristic algorithm. After the tree is displayed, other operations on the results are facilitated by representing the primary keys and column names as hyperlinks.

Detailed Comments

Both the papers have described efficient and faster ways of displaying search results to the user. The first paper tried to achieve this by allowing the user themselves to query on the database without requiring the knowledge to construct queries. Since running keyword matching on a database is not optimal in fetching the exact result and running queries would, the approach discussed here seems to be able to get the closest result much faster. The second paper enabled getting the required results by browsing through the tables by means of primary-key hyperlinks and other functionalities provided in the BANKS interface.

One explanation missing in the primary paper is the approach to generate forms for a given database schema. It has been mentioned that the forms are generated by scripts, but the implementation, efficiency or the functionality of these scripts were not discussed in detail.

The paper on query-forms considers only a subset of SQL in its implementation. But this subset is sufficient to cover the queries on a structured database. Also, as pictures are better than words, representing the parameters as user input is a better way to create queries that writing the queries themselves. Also, the forms have some user understandable question to represent the tables in the query forms which makes identification of forms much easier. The second paper requires a little more effort in identifying the correct result from the database, primarily because the user needs to map the relations and the relationships logically to browse through the database.

The paper on query-forms primarily compares with its related work, the efficiency of being able to create structured queries by the user without the requirement of the knowledge of the database schema or creating queries. Most of the other work either present the schema or demand the knowledge of at least the concept of relational databases but this paper appears to require no knowledge of the database structure.

The questions discussed during the presentation primarily consist of the applications of the query-forms over large unstructured databases. An example quoted in the presentation, which displayed a simple form for a keyword search in Google web search was also discussed as to how that can be extended for at least the example shown in the slides. The three algorithms of query-forms being simple were made understandable my means of proper examples quoted from the paper. A methodology to dynamically generate forms as and when the user selects/fills data was also discussed. The complexity of this methodology was found to be of a complex decision tree which branches out for every field in each of the table present in the database and so would result in a very complex algorithm.