Volume 4, Issue 11, November 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Study on Query form Generation Methods

[1]Gopika S, [2]Jiby J Puthiyadam
[1]Computer & Information Science, CUSAT, India
[2]Asst Professor, Dept of Computer Science and Engineering, India

Abstract— A database becomes useful if its data can be retrieved easily. If a particular user cannot convey what he needs from a database, then the database is of no use. It seems to be difficult to query a database by a particular user who is unfamiliar with a formal query language. One of the simplest methods to query a database is through a form, where a user is allowed to fill in relevant data and obtain required results by submitting that particular form. Designing a good static form is a significant manual task, and the designer needs a deep knowledge of both the data organization and the querying behaviour. Furthermore, form design has two main goals: forms should be easy to understand, and at the same time it must be capable of giving a detailed possible querying capability to the user. This paper mainly covers the different techniques for querying a database through forms. It also identifies some of the difficulties associated with these methods and thereby evaluating the need for creating dynamic query forms for the database queries and the various parameters needed in creating a dynamic query forms.

Keywords— Query Forms, Probabilistic data model, Form interface, Database schema., query customisation

I. INTRODUCTION

We know that databases related to web and scientific data maintain large and heterogeneous data. These types of databases contains a large number of relations[1] and attributes .Query forms are the most popular method used for querying a databases. Traditional query forms are designed and predefined by designers or DBA in various information management systems. Historic predefined query forms are not able to satisfy different on demand queries from a particular user on those databases. Writing well-structured queries [1], using languages such as SQL and XQuery, can create a number of problems. These can be challenging due to a certain number of reasons, including the user's lack of knowledge with the query language and the user's unfamiliarity of the underlying data structure. Forms-based query interfaces are the most popular method used to access databases today. The development of a forms-based interface is often a major step in exploring a database [1] [2].A limited range of queries can only be expressed by such an interface in such forms. All possible queries that any user may have, can be expressed by a set of forms as a whole. Developing an interface that approaches this ideal is surprisingly difficult. Developing a better set of forms by just using the database is a challenging yet another important problem.

The problem identified here is how an efficient query form can be designed to boost the user satisfaction and need in retrieving data from a database [1]. These problems can be solved by developing a good query form with dynamic nature. It means the user can iteratively search to the database until he or she get satisfied with the result obtained.

How to let non-expert users make use of the relational database is a challenging area? A lot of research works are going on database interfaces which help users to query the relational database without a structured query language. QBE (Query-By-Example) and Query Form are two most popular database querying interfaces [3]. Existing database clients and tools make huge efforts to help developers design and generate the query forms, such as Easy Query, Cold Fusion, SAP, Microsoft Access and so on. Visual interfaces to create or customize query forms are provided to the developers by them. One of the major problems of those tools is that, they are given for the professional developers who have well knowledge of their underlying databases and not for the end-user. However, an end-user may not be familiar with the database. If the database schema contains very large number of attributes and relations, it is difficult for them to find appropriate database entities and attributes and to create desired query forms.

At present, query forms have been popularized in most real-world business or scientific information systems. Current research and works mainly focus on how to generate the query forms which can satisfy the user needs.

II. QUERY FORMS

A form-based query interface, which helps the user to fill the query form to particularly specify query parameters [1], is useful which helps users to make access to data. This can be done by any user who is unfamiliar with the formal query languages or the database schema. In most cases, form-based interfaces are used more commonly, but in many cases, each form is designed ad hoc and its applicability is limited to a small set of queries which are already prewritten [1]. As database becomes large and complex, the queries become complex and the development of manual form generation becomes difficult. And in these cases, hidden assumptions will come into act. Another major challenge which arises while creating forms automatically is to satisfy the three important properties such as conciseness, expressivity, and clarity simultaneously. These properties are needed by any form based interfaces.

A. Form Generation

First step in designing a form for a declarative query [2], is to analyse it and identify the major constraints and the expected query results. Then the obtained information from this analysis, as well as from the schema of the database, was used to create the required set of form-elements. Finally, arrange these elements in groups, label them suitably, and lay them out in a meaningful way on the form. The form generation technique was extended to design forms for an entire set of queries. Given a set of interesting queries, the naive approach would be to build one form for each one of them. However, queries against a single schema will more often than not have similarities between them, which we can exploit to minimize redundancy [1] [2]. A *form complexity threshold* (FCT) is introduced to control a form's readability. It is a measure of complexity that no form to be exceed. In such cases, even a single query may have complexity that exceeds the FCT. In the general case, FCT is satisfied by splitting a query cluster covered by a complex form into smaller clusters covered by simpler forms. To measure how useful a form is, *expressivity is defined* as how many different queries it can express.

Key features: This method uses a mechanism to overcome the challenges that limit the usefulness of form. For this they used an algorithm to automatically generate forms. An automated self-managing interface builder helps the users to become more familiar with the database and it also maximises the efficiency.

The result entirely depends on the selected form complexity threshold value.

B. Combinig Keyword Search And Forms For Adhoc Querying

To address the problem of difficulty in querying a database, form-based interfaces AND KEYWORD SEARCH HAVE BEEN proposed in this paper [3]. At query time, a user with a question to be answered issues standard keyword search queries; but instead of giving back tuples, the system will give the forms relevant to the question. Then the user may build a structured query with one of these forms and give it back to the system for evaluation. The main idea here is to treat a set of forms as a set of documents, then let users use keyword search to find relevant forms [3](which in turn are used to pose structured queries).However, this task differs from the standard document search problem in several key ways. Perhaps the most important difference is that a form contains parameters, which are undefined until users fill out the form at query time. Furthermore, possible data values for these parameters often do not appear on the forms [2].A keyword search approach that ignores this difference can yield undesirable results. In the first approach, it simply retrieves a form if the form contains at least one (OR semantics) or all (AND semantics) of the terms from a keyword query. If a user specifies a data value and use Naïve-AND, we will get no answers. If a user uses a Naïve-OR, some forms would be returned if at least one schema term (i.e., a term that matches a table or attribute name) is included [2].However, the data terms (i.e., terms that match data values), if any, would be completely ignored, which will not create any output.

Key features: This method combines a set of forms as documents and retrieves forms based on the given keywords.

One of the main disadvantages is that this method fails to retrieve forms when the user inputs data values

C. Form Customisation

A typical form is predefined and can satisfy only a very few set of queries. Without a provision for change, query specification [2] is limited by the developer and target of the interface developer at the time of form creation. If the form which is available cannot satisfy the required query, the users have no use with the available database. Through form customisation [4] a mechanism which allows a user to alter an existing form to express the required query is proposed. Alterations are themselves specified through filling forms to create an expression in an underlying form manipulation expression language we define. The technical difficulties required to alter forms [2][3] is not much greater than form filling[4].A form-element is an object which is designed to change the user's input into a query fragment. A simple example is a labelled text-box for a search query that accepts a keyword from a user and creates a selection predicate for the corresponding table column in the database. For completeness, the most common query operations in a declarative query language are identified and designed form-elements for each of them.

Constraint Specification Element: A form-element which allows a user to specifically enter a value for a data attribute. This will be mapped to a selection predicate in the underlying query. Result Display Element**:** This form element type enables a user to choose which fields to display in the result of the query. It allows users to customize the query's result.

Result Ordering Element: This form element lets users specify how the result of the form should be sorted.

Aggregate Computation Element: This form element denotes an aggregation operation, either a selection or a projection, in the underlying query. Disjunction Element: This form element is a set of constraint specification elements at least one of which must be satisfied. It is mapped to a set of selection predicates separated by *OR*'s in the underlying query.

Join Specification Element: While most forms hide inter-relationships between fields, the flexibility of a form can be increased by the presence of form-elements that allow users to specify these relationships.

Key features: This method allows the user to add more elements to the form.

A major disadvantage is that the forms are already predefined.

D. Query Recommendation For Interactive Database Exploration

Here user interfaces have been developed to assist the user to type the database queries based on the structure of database, query workload and the distribution of data. Different from the work which focuses on query forms, the queries are treated in the forms of SQL and keywords [3][5].The idea is to track the querying behaviour of the particular user and to identify which parts of the database is required for the corresponding data analysis situation, and recommend queries that retrieve the required data. Here introduce a collaborative approach to recommend database query components for

database exploration [5]. They treat SQL queries as item in the collaborative filtering approach and recommend similar queries to related users.

Key features: The querying behaviour of user can be easily tracked and it recommends query based on these choices.

It fails to identify and retrieve similar queries in terms of their structure.

E. Ranking Of Database Query Results

The problem of ranking answers to a database query when many tuples are returned are addresses in this paper [5][6].They adapt and apply principles of probabilistic models from Information Retrieval for structured data. The proposed solution [7] is independent of the domain. It maximises data and workload statistics and correlations. The ranking functions can be further customized for different applications. The Many-Answers Problem has been investigated outside the database area, especially in Information Retrieval (IR), where many documents often satisfy a given keyword-based query. An approach to overcome these problems ranges from *query reformulation* techniques *automatic ranking* of the query results by their degree of "relevance" to the query and returning only the top-*K* subset.

Key features: Query reformulation technique is used to solve the many many answer problems**.**

When the database schema becomes complex, it cannot handle many many answer problem.

F. Usher

Data quality is a major problem in modern databases. Data entry forms provide the best opportunity for detecting and reducing the errors, but there has been little research for automating the methods for improving data quality at entry time. USHER, an end to-end system for the design of form, data quality assurance and form entry [7][8] was proposed to improve the data quality. USHER [8] learns a probabilistic model over the questions of the form using previous form submissions. USHER then uses the probabilistic model at each step of the data entry process to improve data quality. Before entry, it generates a layout of the form that captures the most important data values of a form instance as fast as possible. During entry, it dynamically adapts the form to the values which are entered, and provides a real time feedback to assist the data enterer toward their intended values [7][8]. After entry, it re-asks questions that it seems likely to have been entered as not suitable to the criteria. All the three components of USHER are evaluated using two real world data sets. The results evaluated shows that each component has the potential to improve data quality considerably, at a low cost when compared to the other methods.

Key features: Improved data quality by using probabilistic data models.

Time complexity is a major problem in this technique**.**

G. Query By Example

Query by Example (QBE) is an industry standard language for querying data to obtain information from relational database systems.

Query by Example (QBE) was developed in tandem with SQL in the 1970s by IBM (Zloohf) where the technique of using search terms was used to filter data based on data content graphically. This avoided the user having to know or learn SQL or any other query language as QBE itself included the language necessary to perform the searches in order to achieve results quickly on built in logical conditions. Microsoft QBE, as used by Microsoft Access, is mainly used for obtaining the results from search queries. QBE language can be used for the creation or modification of data.

QBE and SQL are largely interchangeable. Performing a sample query in Microsoft Access using QBE and using its graphical tools to establish the query parameters gives a graphical representation of the query and on viewing the SQL within the program the user is presented with an accurately written piece of equivalent SQL code that performs an identical query.

The advantage versus disadvantage discussion of QBE and SQL is largely dependent on the technical skills (and background) of the user and the time available to develop the query in question. It must be remembered that Microsoft Access has the largest database user base in the world for an extremely good reason. It is graphical and easy to use. Microsoft QBE is therefore exactly the same yet it can become the technical procedure that some may prefer if the user has the technical preference to develop their own SQL queries. The Microsoft QBE language however is an interpretation of QBE as standards were never developed to ensure conformity across systems. This can lead to such interpretations as Microsoft QBE differing from graphical based product to product and hence there is a risk of different results being obtained from seemingly identical queries in a limited number of cases. As SQL has written and constantly developing standards and is ubiquitous in the relational database industry then the technical ability to write code based SQL queries ensures that queries, if properly written according to those standards, will yield the same results across products and systems and therefore we can place reliance on its integrity. Whilst Microsoft Access QBE is largely sufficient for the target market it serves (small standalone single-user databases), the QBE lack of portability makes SQL much more advantageous in my opinion.

Key features: Includes the language necessary to perform the searches in order to achieve results quickly on built in logical conditions. This method entirely depends on the technical skills of the user.

III. RESULT ANALYSIS

In Assisted querying using instant response interfaces, a novel user interface have been developed to assist the user to type the database queries based on the structure of database, distribution of data and the query workload. In this method size is a major constraint. As the complexity of database increases, the time required for retrieving data from the database

seems to be non-feasible. In automated creation of a form-based database query interface and in expressive query specification through form customization, the problem is that, if the database schema is very large, it is difficult to find the most suitable database entities and attributes to create a relevant query form. In combining keyword search and forms for adhoc querying of database, it automatically generate a lot of predefined query forms. Here several keywords are used by the user to find relevant query forms. This leads to the conclusion that a query rewrite by mapping data values to schema values during keyword search. Another one is that, simply displaying the returned form as a flat list.

In automating the design and construction of query forms, it is a work-load driven model. It applies clustering algorithm to find representative queries. One of the disadvantages is that if we generate lots of query forms in advance, there are still user queries that cannot be satisfied by one of the query form.

Query recommendation for interactive database exploration introduces a collaborative approach to recommend database query components for database exploration. One of the problems is that they do not consider the goodness of query result.

Building dynamic faceted search systems over database present relevant facets for the users according to navigation path. Dynamic faceted search engine are similar to the dynamic query form if we only consider selection components in query.

In Usher: Improving Data Quality with dynamic forms, Data quality is a critical problem in modern database. USHER is an end to end system for form design, entry and data quality assurance.

Table 1: Comparison of different methods

Method	Precision	Recall	Database Size	Response Time
Combining keyword search and forms for adhoc querying	56	52	Not affected	Fast
Form customization	80	74	Not affected	Medium
Query recommendation for interactive database exploration	68	65	Not suitable for large databases	Fast
Ranking of database query results	72	72	Not suitable for Complex databases	Fast
Usher	64	60	Not affected	Fast
Query by example	Equal	Equal	Not affected	Medium

V. CONCLUSION

The advantages of using a query form for the database queries are evaluated. The major steps in developing a query form for the database queries are evaluated. Even though the query forms have many advantages, the need for developing a dynamic query form for the database is seen. A forms-based interface is the gateway to many a database and it ultimately determines the availability and usefulness of the data. Designing a good set of forms is hard. Based on this survey, the method used in ranking of database query results seems to be a better technique in creation of query forms for the database queries. In this paper, we have seen the methods to generate forms and the difficulties associated with these methods and thereby the need of creating a dynamic query forms for retrieving data from database.

REFERENCE

[1] A. Nandi and H.V. Jagdish. *"Assisted querying using instant response interfaces"*. In proceedings of ACMSIGMOD, pages 1156-1158, 2007.

[2] M. Jayapandian and H. V. Jagdish. *"Automating the design and construction of query forms".* IEEE TKDE, 21(10): 1389-1402, 2009.

[3] Eric Chu, Akanksha Baid Xiaoyong Chai AnHai Doan Jeffrey Naughton*"Combining Keyword Search and Forms for Ad Hoc Querying of Databases"*

[4] Magesh Jayapandian, H.V Jagadish,*"Expressive Query Specification through Form Customization", EDBT'08,* March 25–30, 2008

[5] G. Chatzopoulou, M. Eirinaki and N. Polyzotis *"Query recommendations for interactive database exploration"*. In proceedings of SSDBM, pages 3-18, New Orleans, LA, USA, June 2009

[6] S. B. Roy, H. Wang, U. Nambiar, G. Das and M. K. Mohsnia. Dynacet: *"Building faceted search systems over databases"*. In proceedings of ICDE Conference, pages 1463-1466, Shanghai, China, March 2009.

[7] Surajit Chaudhuri, Gautam Das, Vagelis Hristidis,"Probabilistic Ranking of Database Query Results", Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

[8] Kuang Chen, Harr Chen, Neil Conway, Joseph M. Heller stein, Tapan S. Parikh, "USHER: Improving Data Quality with Dynamic Forms",

[9] N. Khoussainova, M. Balazinska, W. Gatterbauer, Y. Kwon, and D. Suciu "A case for a collaborative query management system". In Proceedings of CIDR, Asilomar, CA, USA, January 2009.

[10] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu. "Snipsuggest: Context-aware autocompletion for sql". PVLDB, 4(1):22–33, 2010.

[11] B. Liu and H. V. Jagadish. "Using trees to depict a forest". PVLDB, 2(1):133–144, 2009.

[12] D. Rafiei, K. Bharat, and A. Shukla. "Diversifying web search results", In Proceedings of WWW, pages 781–790, Raleigh, North Carolina, USA, April 2010.

