International Journal of Computer Engineering and Information Technology

VOL. 8, NO. 9, September 2016, 174–178
Available online at: www.ijceit.org
E-ISSN 2412-8856 (Online)

Creating Database Queries for Dynamic Query Forms

Prof. Ahamad Husen Shekh[1], Prof. Rosy Bhoi[2] and Prof. Ankita Kadu[3]

[1, 2, 3] Computer Engineering, Theem College Of Engineering, Boisar, Maharastra, India

[1]skahm2013@gmail.com, [2]goldenera13@gmail.com, [3]ankitakadu@gmail.com

ABSTRACT

A form is a database object that you can use to create a user interface for a database application. A "bound" form is one that is directly connected to a data source such as a table or query, and can be used to enter, edit, or display data from that data source. Alternatively, you can create an "unbound" form that does not link directly to a data source, but which still contains command buttons, labels, or other controls that you need to operate your application.

This article focuses primarily on bound forms. You can use bound forms to control access to data, such as which fields or rows of data are displayed. For example, certain users might need to see only several fields in a table with many fields.[1] Providing those users with a form that contains only those fields makes it easier for them to use the database. You can also add command buttons and other features to a form to automate frequently performed actions.

Think of bound forms as windows through which people see and reach your database. An effective form speeds the use of your database, because people don't have to search for what they need. A visually attractive form makes working with the database more pleasant and more efficient, and it can also help prevent incorrect data from being entered.

Modern web databases maintain large and heterogeneous data. These real-world databases contain over hundreds relations and attributes. Traditional predefined query forms are not able to satisfy various ad-hoc queries from users on those databases; a user can also fill the query form and submit queries to view the query result at each iteration. In this way, a query form could be dynamically refined till the user satisfies with the query results.

.

Keywords: *Query Form creation, User Interaction, Database, Query Form.*

1. INTRODUCTION

Traditional data mining technologies cannot work with huge, heterogeneous, unstructured data. Modern web database as well as scientific database maintains tremendous and heterogeneous data. These real word databases may contain hundreds or even thousands of relations and attributes. Query form is one of the most widely used interfaces for querying database. Traditional query forms are designed and pre-defined by developer or DBA in various information management systems. But extracting the useful information with this traditional query form from large dataset and streams of the data is not possible and it is difficult to design set of static query forms to satisfy numerous ad- hoc database queries on those complex databases. Query form is one of the most important user interfaces used by users for querying databases. With the fast development of web information and scientific databases, modern databases became tremendous and complex. Many databases such as Freebase, DBPedia have thousands of structured web entities. This paper proposes DQF, a novel database query form interface, which is able to dynamically generate query forms. The essence of DQF is to capture a user's preference and rank query form components, assisting him/her to make decisions. The generation of a query form is an iterative process and is guided by the user. At each iteration, the system automatically generates ranking lists of form components and the user then adds the desired form components into the query form. The ranking of form components is based on the captured user preference. A user can also fill the query form and submit queries to view the query result at each iteration. In this way, a query form could be dynamically refined till the user satisfies with the query results.[2] We utilize the expected F-measure for measuring the goodness of a query form. A probabilistic model is developed for estimating the goodness of a query form in DQF. Our experimental evaluation and user study demonstrate the effectiveness and efficiency of the system.

2. ALGORITHM

A: PageRank-like algorithm

Page Rank is a topic much discussed by Search Engine Optimization (SEO) experts. At the heart of Page Rank is a mathematical formula that seems scary to look at but is actually fairly simple to understand.

Despite this many people seem to get it wrong! In particular "Chris Ridings of

www.searchenginesystems.net" has written a paper entitled "Page Rank Explained: Everything you've always wanted to know about Page Rank", pointed to by many people, that contains a fundamental mistake early on in the explanation! Unfortunately this means some of the recommendations in the paper are not quite accurate.
By showing code to correctly calculate real PageRank I hope to achieve several things in this response:

1. Clearly explain how PageRank is calculated.
2. Go through every example in Chris' paper, and add some more of my own, showing the correct PageRank for each diagram. By showing the code used to calculate each diagram I've opened myself up to peer review - mostly in an effort to make sure the examples are correct, but also because the code can help explain the PageRank calculations.
3. Describe some principles and observations on website design based on these correctly calculated examples.

Any good web designer should take the time to fully understand how PageRank really works - if you don't then your site's layout could be seriously hurting your Google listings!

B: How is PageRank Used?

PageRank is one of the methods Google uses to determine a page's relevance or importance. It is only one part of the story when it comes to the Google listing, but the other aspects are discussed elsewhere (and are ever changing) and PageRank is interesting enough to deserve a paper of its own.
PageRank is also displayed on the toolbar of your browser if you've installed the Google toolbar (http://toolbar.google.com/). But the Toolbar PageRank only goes from 0 – 10 and seems to be something like a logarithmic scale:

Toolbar PageRank (log base 10)	Real PageRank
0	0 - 10
1	100 - 1,000
2	1,000 - 10,000
3	10,000 - 100,000
4	and so on...

We can't know the exact details of the scale because, as we'll see later, the maximum PR of all pages on the web changes every month when Google does its re-indexing! If we presume the scale is logarithmic (although there is only anecdotal evidence for this at the time of writing) then Google could simply give the highest actual PR page a toolbar PR of 10 and scale the rest appropriately.
Also the toolbar sometimes guesses! The toolbar often shows me a Toolbar PR for pages I've only just uploaded and cannot possibly be in the index yet!
What seems to be happening is that the toolbar looks at the URL of the page the browser is displaying and strips off everything down the last "/" (i.e. it goes to the "parent" page in URL terms). If Google has a Toolbar PR for that parent then it subtracts 1 and shows that as the Toolbar PR for this page. If there's no PR for the parent it goes to the parent's parent's page, but subtracting 2, and so on all the way up to the root of your site. If it can't find a Toolbar PR to display in this way, that is if it doesn't find a page with a real calculated PR, then the bar is greyed out.
Note that if the Toolbar is guessing in this way, the Actual PR of the page is 0 - though its PR will be calculated shortly after the Google spider first sees it.
PageRank says nothing about the content or size of a page, the language it's written in, or the text used in the anchor of a link!
They proposed a PageRank-like algorithm to compute the importance of an attribute in the schema. In this paper, we utilize the schema graph to compute the relevance of two attributes. A database schema graph is denoted by G= (R,FK, ξ,A), in which R is the set of nodes representing the relations, A is the set of attributes, FK is the set of edges representing the foreign keys, and ξ:A-Ris an attribute labeling function to indicate which relation contains the attribute. Based on the database schema graph.

3. EXISTING SYSTEM

Recently proposed automatic approaches to generate the database query forms without user participation presented a data-driven method. It first finds a set of data attributes, which are most likely queried based on the database schema and data instances. [3]Then, the query forms are generated based on the selected attributes.
One problem of the aforementioned approaches is that, if the database schema is large and complex, user queries could be quite diverse. In that case, even if we generate lots of query forms in advance, there are still user queries that cannot be satisfied by any one of query forms. Another problem is that, when we generate a large number of query forms, how to let users find an appropriate and desired query form would be challenging. A solution that combines keyword search with query form generation is proposed. It automatically generates a lot of query forms in advance. The user inputs several keywords to find relevant query forms from a large number of pre-generated query forms. It works well in the databases which have rich textual information in data tuples and schemas. However, it

is not appropriate when the user does not have concrete keywords to describe the queries at the beginning, especially for the numeric attributes.

Disadvantages of Existing System:

However, the creation of customized queries totally depends on users' manual editing. If a user is not familiar with the database schema in advance, those hundreds or thousands of data attributes would confuse him/her.

4. PROPOSED SYSTEM

We propose a Dynamic Query Form system: DQF, a query interface which is capable of dynamically generating query forms for users. Different from traditional document retrieval, users in database retrieval are often willing to perform many rounds of actions (i.e., refining query conditions) before identifying the final candidates. The essence of DQF is to capture user interests during user interactions and to adapt the query form iteratively. Each iteration consists of two types of user interactions: Query Form Enrichment and Query Execution [5].

The following figure shows the work-flow of DQF. It starts with a basic query form which contains very few primary attributes of the database. The basic query form is then enriched iteratively via the interactions between the user and our system until the user is satisfied with the query results.

Advantages of Proposed System:

A dynamic query form system which generates the query forms according to the user's desire at run time. The system provides a solution for the query interface in large and complex databases. The goodness of a query form is determined by the query results generated from the query form. Based on this, we rank and recommend the potential query form components so that users can refine the query form easily. [4] We develop efficient algorithms to estimate the goodness of the projection and selection form components. Here efficiency is important because DQF is an online system where users often expect quick response.

Proposed System architecture

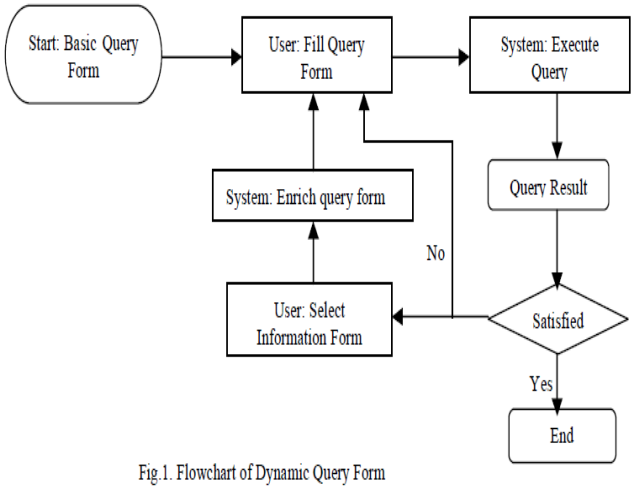

Fig.1. Flowchart of Dynamic Query Form

1. Query result

To decide whether a query form is desired or not, a user does not have time to go over every data instance in the query results. In addition, many database queries output a huge amount of data instances. In order to avoid this "Many-Answer" problem , we only output a compressed result table to show a high level view of the query results first. Each instance in the compressed table represents a cluster of actual data instances. Then, the user can click through interested clusters to view the detailed data instances. The compressed high-level view of query results is proposed in . There are many one-pass clustering algorithms for generating the compressed view efficiently .In our implementation, we choose the incremental data clustering framework because of the efficiency issue. Certainly, different data clustering methods would have different compressed views for the users.[5] Also, different clustering methods are preferable to different data types. In this paper, clustering is just to provide a better view of the query results for the user. The system developers can select a different clustering algorithm if needed.

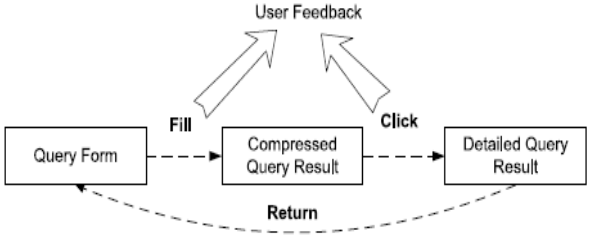

Fig. 2. Query Result

Another important usage of the compressed view is to collect the user feedback. Using the collected feedback, we can estimate the goodness of a query form so that we could recommend appropriate query form components. In real world, end-users are reluctant to provide explicit feedback. The click-through on the compressed view table is an implicit feedback to tell our system which cluster (or subset) of data instances is desired by the user. In some recommendation systems and search engines, the end-users are also allowed to provide the negative feedback. The negative feedback is a collection of the data instances that are not desired by the users. In the query form results, we assume most of the queried data instances are not desired by the users because if they are already desired, then the query form generation is almost done. [6]Therefore, the positive feedback is more informative than the negative feedback in the query form generation. Our proposed model can be easily extended for incorporating the negative feedback.

5. SYSTEM ARCHITECHTURE

The architecture shows the dynamic query form interface to the user. The query form contains many fields to the user for filling the attributes those he or she want to view. [6] In addition with the attribute the user is provided a constraint specification field by using that capability the user can easily make conditions. Every entry to the form is taken as input to the database.

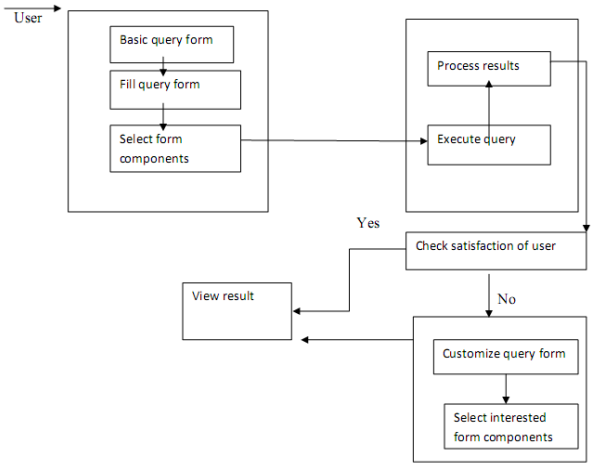

Fig. 3. System Architecture

6. MODULES

The system is proposed to have the following modules along with functional requirements.

i) Query Form Enrichment
ii) Query Execution
iii) Customized Query Form
iv) Database Query Recommendation

i) Query Form Enrichment

1)DQF recommends a ranked list of query form components to the user.
2) The user selects the desired form components into the current query form.

ii) Query execution

1) The user fills out the current query form and submit a query.
2) DQF executes the query and shows the results.
3) The user provides the feedback about the query results.

iii) Customized Query Form

They provide visual interfaces for developers to create or customize query forms. The problem of those tools is that, they are provided for the professional developers who are familiar with their databases, not for end-users. If proposed a system which allows end-users to customize the existing query form at run time. However, an end-user may not be familiar with the database. [7]If the database schema is very large, it is difficult for them to find appropriate database entities and attributes and to create desired query forms.

iv) Database Query Recommendation

Recent studies introduce collaborative approaches to recommend database query components for database exploration. They treat SQL queries as items in the collaborative filtering approach, and recommend similar queries to related users.

7. CONCLUSION

The paper gives dynamically query form generation for the database queries. Here the main idea is the rank the query form components based on user interest. To capture the user interests based on historical queries and click through feedback. The experimental results show that dynamic query form leads higher success than static query forms. The ranking is also makes easier for user selection in the query form attributes.
In this paper we propose a dynamic query form generation approach which helps users dynamically generate query forms. The key idea is to use a probabilistic model to rank form components based on user preferences. We capture user preference using both historical queries and run-time feedback such as clickthrough. Experimental results show that the dynamic approach often leads to higher success

rate and simpler query forms compared with a static approach. The ranking of form components also makes it easier for users to customize query forms. As future work, we will study how our approach can be extended to non relational data.

REFERENCES

[1] http://www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm
[2] Liang Tang, Tao Li, Yexi Jiang, and Zhiyuan Chen "Dynamic Query Forms for Database Queries"
[3] Freebase. http://www.freebase.com.
[4] Cold Fusion. http://www.adobe.com/products/coldfusion/.
[5] DBPedia. http://DBPedia.org.
[6] S. Cohen-Boulakia, O. Biton, S. Davidson, and C. Froidevaux. Bioguidesrs: querying multiple sources with a user-centric perspective. Bioinformatics, 23(10):1301–1303, 2007.
[7] G. Das and H. Mannila. Context-based similarity measures for categorical databases. In Proceedings of PKDD 2000, pages 201–210, Lyon, France, September 2000.
[8] W. B. Frakes and R. A. Baeza-Yates. Information Retrieval: Data Structures and Algorithms. Prentice-Hall, 1992
[9] M. Jayapandian and H. V. Jagadish. Automated creation of a forms-based database query interface. In Proceedings of the VLDB Endowment, pages 695–709, August 2008.