Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 1, January 2015, pg.222 – 228

REVIEW ARTICLE

A Review Report on Dynamic Query Forms for Relational Databases Queries

Siddharth Tayade[1], Prof. Sonali Bodkhe[2]

[1]Student (M.Tech) CSE, G.H.Raisoni College Academy of Engineering and technology Nagpur, (INDIA)

[2]H.O.D (M.Tech) CSE, G.H.Raisoni College Academy of Engineering and technology Nagpur, (INDIA)

[1] siddhutayade9@hotmail.com; [2] sonali.mahure@gmail.com

Abstract-- Query form is one of the most widely used user interfaces for querying databases. Traditional query forms are designed and pre-defined by developers or DBA in various information management systems. By this rapid development of web information and scientific databases, modern databases become very large and complex. Therefore, it is difficult to design a set of static query forms to satisfy various ad-hoc database queries on those complex databases. Dynamic queries are a novel approach to information seeking that may enable users to cope with information overload. They allow users to see an overview of the database, rapidly explore and conveniently filter out unwanted information. Users fly through information spaces by incrementally adjusting a query (with sliders, buttons, and other filters) while continuously viewing the changing results. This paper proposes Dynamic Query form, a curious database query form interface, which is able to dynamically create query forms. The significance of DQF is to capture a user's choice and classify query form components, support him/her to make conclusion. The creation of query form is a repetitive process and is conducted by the users. In each repetition, the system automatically creates classification lists of form components and the user then adds the desired form components into the query form. The classification of form components is based on the captured user choice. A user may al so fill up the query form and deliver queries to view the query output at each step. Thus, a query form could be dynamically refined till the user answer with the query output. A probabilistic model is developed for estimating the excellence of a query form in DQF. I have studied evaluation and user study certify the effectiveness and efficiency of the system.

Keywords-- Query Form, User Interaction, Query Form Generation, *Iterative process, DQF*

I. INTRODUCTION

A database is only as useful as its query interface allows it to be. If a user is unable to convey to the database what he or she wants from it, even the richest data store provides little or no value. Writing well-structured queries, in languages such as SQL and XQuery, can be challenging due to a number of reasons, including the user's lack of familiarity with the query language and the user's ignorance of the underlying schema. A form is a simple and intuitive query interface frequently used to provide easy database access. It requires no knowledge, on the part of the user, of how the data is organized in storage and no expertise in query languages. For these reasons, forms are a popular choice for most of today's databases. Creating a forms-based interface for an existing database requires careful analysis of its data content and user requirements. Many existing database management and development tools, such as Easy Query [4], Cold Fusion [2], SAP and Microsoft Access, provide several mechanisms to let users create customized queries on databases. However, the creation of customized queries totally depends on users' manual editing [3]. If a user is not familiar with the database schema in advance, those hundreds or thousands of data attributes would confuse him/her.

II. RELATED WORK

In Assisted querying using instant response interfaces, A.Nandi and H.V. Jagdish [6], this proposed auto completion approaches for database queries. Here a novel user interface have been developed to assist the user to type the database queries based on query workload, the data distribution and the database schema.

In automated creation of a form-based database query interface and In expressive query specification through form customization, M. Jayapandian and H.V. Jagadish [7] [8], Here proposed a customized approach to provide visual interface for developers to create or customize query form. EasyQuery, ColdFusion, SAP are the main tools to help developers design and generate the query form. The problem of these tool is that, they are provided for the professional developers who are familiar with their database not for end users. Another problem is that, if the database schema is very large, it is difficult for them to find appropriate database entities, attributes and to create desired query form.

In Combining keyword search and forms for adhoc querying of database, E.Chu, A.Baid X. Chai, Doan and J.F. Naughton[10], It automatically generate a lot of query form in advance. Here user input several keywords to find relevant query form. 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, M.Jayapandian and H.V. Jagadish [9], here propose automatic approaches to generate the database query without user participation. It is a work-load driven model. It applies clustering algorithm to find representative queries. One of the disadvantage 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, M. Eirinaki and N. Polyzotis [11], here introduce a collaborative approach to recommend database query components for database exploration. They treat SQL queries as item in the collaborative filtering approach and recommend similar queries to related users. One of the problem is that they do not consider the goodness of query result. In proposed system recommendation is a query component for each iteration.

In Building dynamic faceted search systems over database, S.B. Roy, H. Nambiar, G. Das and M.K. Mohania [12], a domain independent system that provides effective minimum- effort based dynamic faceted search solution over enterprise database.

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

In Usher: Improving Data Quality with dynamic forms, K.Chen, H.Chen, N.Conway, J.M. Hellerstein and T.S. Parikh [13], Data quality is a critical problem in modern database. USHER, an end to end system for form design, entry and data quality assurance. In DQF, deals with database query forms instead of data entry forms.

Existing database clients and tools make great efforts to help developers design and generate the query forms, such as Easy Query [4], Cold Fusion [2], SAP, Microsoft Access and so on 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.

III.PROPOSED SYSTEM ARCHITECTURE

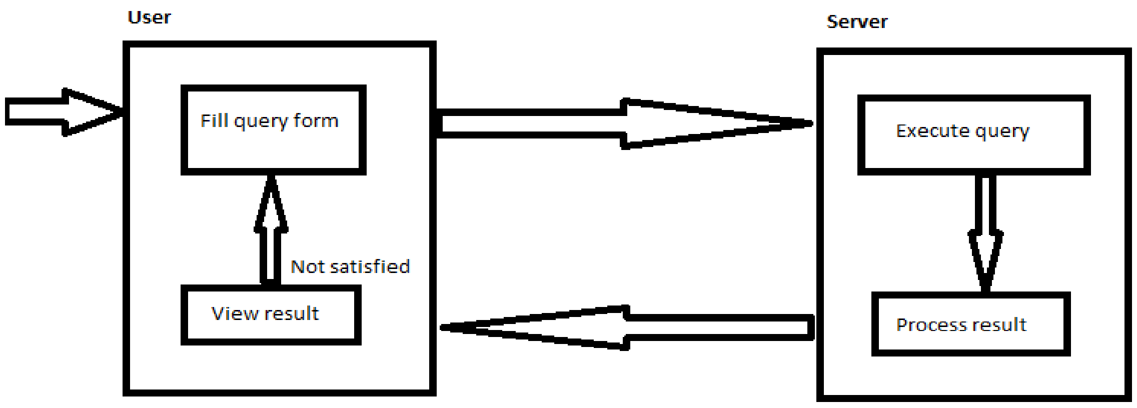

Fig1: Flow chart of dynamic query form

Figure shows flow chart of dynamic query form. A dynamic query form system which generates query form according to the user’s desire at run time. The system provides a solution for query interface in large and complex database. Apply F measure to estimate the goodness of a query form. F-measure is a typical metric to evaluate query result. The metric is also appropriate for query form because query forms are designed to help users query the database. The goodness of a query form is determined by the query results generated from the query form. Based on this, we can rank and recommend the potential query form components. Here efficiency is important because dynamic query form is an online system where users often expect quick response.

Each query form corresponds to SQL query template. Query forms allow users to fill parameters to generate different queries. In this paper, we focus on the projection and selection components of a query form. Ad-hoc join is not handled by our dynamic query form because join is not a part of the query form and is invisible for users. 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 result. In addition, many database queries output a huge amount of data instances. In order to avoid this many-answer problem, only output compressed result table to show a high level view of the query result 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.

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 users. Also, different clustering methods are preferable to different data types. Here, clustering is just to provide a better view of the query result for user. The system developers can select a different clustering algorithm if needed. 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. Figure below shows the user action.

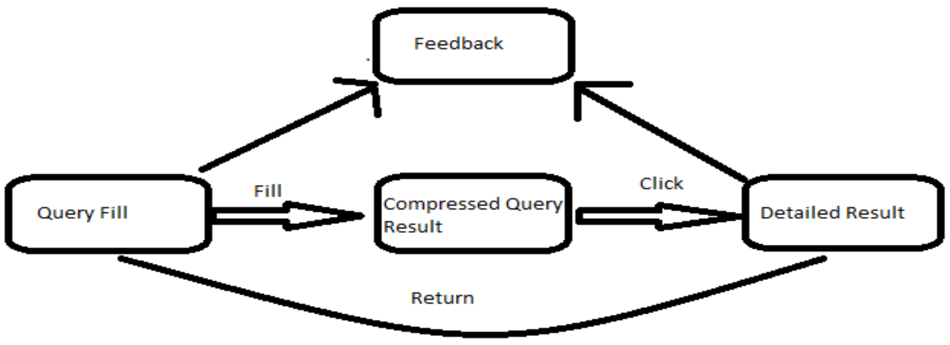

Fig.2 User action

IV. ALGORITHM

<u>Algorithm 1</u>: Generate Forms.

```
**Input**: A query $Q$ (as an Evaluation Plan)
**Output**: A form $F$
// Element Construction and Grouping
Create a new form-group $g$ and add it to the form-tree $T$;
For each operation $o \in Q$ when traversed top-down do
```

```
case o is a "selection"
Create a constraint-element using the selection predicate;
Put this constraint-element in g;
case o is a "projection"
Create a result-element using each projected attribute;
Put these result-elements in g;
case o is a "join"
Create a join-element using the two (left and right)
attributes of the join condition;
Put this join-element in g;
case o is an "aggregate function"
Create an aggregate-element using the the group-by attribute, the grouping-basis and the aggregate function;
Put this aggregate-element in g;
case o is an "order function"
Create an order-element using the the order-by attribute, the ordering-basis and the order function;
Put this order-element in g;
Create a new group g' as a child of g in T;
Set $g \leftarrow g$;
   End
// Element and Group Labeling
foreach form-group $g \in T$ do
Label g relative to its parent group (use absolute path if
g is the root);
foreach form-element $e \in g$ do
Label e relative to g;
end
end.
```

Algorithm 2: FindBestLessEqCondition.

```
Data: $\alpha$ is the fraction of instances desired by user, DQone is the query result of Qone, As is the selection attribute.
Result: $s^*$is the best query condition of As.
Begin
// sort by As into an ordered set Dsorted
Dsorted $\leftarrow$Sort (DQone, As)
      $s \leftarrow \emptyset$, $fscore^* \leftarrow 0$
       $n \leftarrow 0$, $d \leftarrow \alpha\beta 2$
      for $i \leftarrow 1$ to $|Dsorted|$ do
```

```
$d \leftarrow Dsorted[i]$
$s \leftarrow$ "$As \leq dAs$" // compute fscore of "$As \leq dAs$"
$n \leftarrow n + Pu\ (dAFi)\ P\ (dAFi)\ P\ (\sigma Fi\ |d)\ P\ (s|d)$
$d \leftarrow d + P\ (dAFi)\ P\ (\sigma Fi\ |d)\ P\ (s|d)$
$fscore \leftarrow (1 + \beta 2) \cdot n/d$
if $fscore \geq fscore$ then
$s^* \leftarrow s$
$fscore^* \leftarrow fscore$
```

V. STATIC VS DYNAMIC QUERY FORMS

When a query task is covered by one historical queries, then SQF built on those historical queries can be used to fill that query task 3But the costs of using SQF and DQF to fulfil those task are different. *Form- Complexity* was proposed in to estimate cost of using a query form. That is sum of the number of selection components, projection components, and Relations.

VI. CONCLUSIONS

I studied 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 click through. Experimental results show that the dynamic approach often leads to the higher success rate and simpler query forms compared with a static approach. Ranking of form components also makes it easier for users to customize query form.

As future work, we will study how our approach can be extended to non-relational data. As for the future work, we plan to develop multiple methods to capture the user's interest for the queries besides the click feedback. For instance, we can add a text-box for users to input some keywords queries.

REFERENCES

[1] Liang Tang, Tao Li, Yexi Jiang, and Zhiyuan Chen." Dynamic Query Forms for Database Queries". IEEE transactions on knowledge and data engineering vol: pp no: 99 year 2013
[2] Cold Fusion. http://www.adobe.com/products/coldfusion/.
[3] DBPedia. http://DBPedia.org
[4] EasyQuery. http://devtools.korzh.com/eq/dotnet/.
[5] Freebase. http://www.freebase.com
[6] A.Nandi and H.V. Jagdish. Assisted querying using instant response interfaces. In proceedings of ACMSIGMOD, pages 1156-1158, 2007.
[7] 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.
[8] M. Jayapandian and H. V. Jagadish. Expressive query specification through form customization. In proceedings of International Conference on Extending Database Technology (EDBT), pages 416-427, Nantes, France, March 2008.

[9] M. Jayapandian and H. V. Jagadish. Automating the design and construction of query forms. IEEE TKDE, 21(10): 1389-1402, 2009.
[10] E.Chu, A. Baid, X. Chai, A. Doan and J. F. Naughton. Combining keyword search and form for ad hoc querying of databases. In proceedings of ACM SIGMOD Conference, pages 349-360, providence, Rhode Island, USA, and June 2009.
[11] 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.
[12] 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.
[13] K. Chen, H. Chen, N. Conway, J. M. Hellerstein and T. S. Parikh. Usher: Improving data quality with dynamic forms. In proceedings of ICDE Conference, pages 321-332, Long Beach, California, USA, March 2010.
[14] W. B. Frakes and R. A. Baeza-Yates. "*Information Retrieval: Data Structures and Algorithms"*. Prentice-Hall, 1992.
[15] T. Joachims and F. Radlinski. "Search engines that learn from implicit feedback". *IEEE Computer (COMPUTER),* 40(8):34–40, 2007.
[16] 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.
[17] Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. "Query by output". In *Proceedings of SIGMOD*, pages 535–548, Providence, Rhode Island, USA, September 2009.
[18] Jayashri M. Jambukar." Survey Paper on Creation Of dynamic query Form for mining highly Optimized transactional databases". International Journal of Computational Engineering Research||Vol, 04||Issue, 3||, March 2014.