IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 7, July 2015.
www.ijiset.com
ISSN 2348 – 7968

Data Retrieving Using Dynamic Query Forms Approach

#1 Ku. Kalyani Matey

M.Tech Semester-IV student, Department of CSE, RTM Nagpur University Priyadarshini Bhagwati College of Engineering, Nagpur (M.S),India.

#2 Ms. Archana Nikose

2Assitant Professor, Department of CSE, Priyadarshini Bhagwati College of Engineering, Nagpur (M.S.),India

***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. With the 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. In this paper, 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 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. 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.**

Index Terms - Query Forms, User Interaction, Query Form Generation, DQF.

I. INTRODUCTION

A database is only as functional as its query interface allows it to be. If a user is not capable to communicate to the database what he or she wishes from it, even the richest data store provides petite or no value. Writing well-structured queries, in languages such as SQL and X-Query, can be challenging due to a number of reasons, including the user's lack familiarity with the query language and the user's ignorance of the underlying schema. A form-based query interface, which only requires filling blanks to identify query parameters, is precious since it helps make data users with no knowledge of official query languages or the database schema. In practice, form-based interfaces are used frequently, but usually each form is designed in an ad-hoc way and its applicability is restricted to a small set of fixed queries. Query form is one of the majority used user interfaces for querying databases. Traditional query forms are designed and predefined by developers or DBA in various information management systems. With the rapid development of web information and scientific databases, modern databases become very large and complex. Dynamic Query Form framework (DQF), an inquiry outskirt which is fit for eagerly creating Query forms for clients. Not quite the same as routine report rescue, clients in database recuperation are as often as possible eager to do part of rounds of activities (that is refining nature's domain) before distinguishing a definitive competitors. The heart of DQF is to bind client profits amid client correspondences and to adjust the Query form over and over.

The important feature of DQF is:
1) During the user interactions, capture the user interest.
2) Iteratively adapt the query forms.

Each of this iteration is made up of two types of user interactions. They are:
1) Query Execution, and
2) Query Form Enrichment.

Interaction between Users and DQF

Query form Enrichment :- DQF recommends a ranked list of query form components to the user. The user selects the desired form components into current query form.

Query Execution :- The user fills out the current query form and submit a query DQF executes the query and shows the results. The user provides the feedback about the query results.

The work flow of the DQF starts with a basic query form containing very few primary attributes of the database. It is then enriched iteratively by the means of interactions between the user and the system, unless the user is pleased with the query outcome. Mainly the study of query form components and the dynamic production of query forms are done in this paper. This paper proposes a dynamic query form system, generating the query forms in accordance with the run time desire of the user. The system presents an elucidation for the query interface in large and composite databases. The technique applies F-measure for approximation of the goodness of a query form [7]. Using this, we can rank and recommend the probable query form components, so that the users can filter the query form effortlessly. By using the proposed metric, to approximate the goodness of the protrusion and assortment of form components, have developed an efficient algorithms. As the DQF is an online tool and users usually expect quick response, Efficiency is very important. Here, this paper proposes a dynamic query form generation technique which will assist users to dynamically generate query forms. The key idea is based on user preferences, to use probabilistic model to rank form

components. By using both, runtime feedback and historical queries, system captures the user preferences

II LITERATURE SURVEY

A lot of research works focus on database interfaces which assist users to query the relational database without SQL. QBE (Query-By-Example)[6] and Query Form are two most widely used database querying interfaces. Current studies and works mainly focus on how to generate the query forms.

Modified Query Form:
The tools provided by the database clients make great efforts to help developers generate the query forms, such as Easy Query [2], Cold Fusion [1] and so on. They visual interfaces for developers to create or customize query forms. The problem of those tools is that, they are provided for the professional developers [3].H.V. Jagadish proposed a system which allows end-users to customize the existing query form at run time [7]. If the database schema is very large, it is difficult for end user to find appropriate database entities and attributes.

Automated Creation of Forms:
M. Jayapandian presented a data-driven method [3]. It first finds a set of data attributes, which are most likely queried based on the database schema and data instances, the query forms are generated based on the selected attributes.

Automating the design and construction of query forms:
H.V. Jagadish presented a workload-driven method [8].It clustering algorithm on historical queries to find the representative queries. The query forms are then generated based on those representative queries. One problem of the aforementioned approaches [3],[8] is that, 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 query form would be challenging.

Combining keyword search and forms:
A solution for aforementioned approaches [3], [8] is proposed in [9].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 but it is not appropriate when the user does not have concrete keywords to describe the queries

USHER: Improving Data Quality with Dynamic Forms:
K.Chen proposed the system which shows the probabilistic approaches can be used to design intelligent data entry forms that promote high data quality. USHER leverages data driven insights to automate multiple steps in the data entry pipeline. Before entry, we find an ordering of form fields that promotes rapid information capture, driven by a greedy information gain principle. During entry, we use the same principle to dynamically adapt the form based on entered values. After entry, we automatically identify possibly erroneous inputs, guided by contextualized error likelihood, and re-ask those questions to verify their correctness. Our empirical evaluations demonstrate the data quality benefits of each of these components: question ordering allows better prediction accuracy and the re-asking model identifies erroneous responses effectively. There are a variety of ways in which this work can be extended. Our intention to answer this problem in greater depth with user studies and field deployments of our system. On the modelling side, our current probabilistic approach assumes that every question is discrete and takes on a series of unrelated values. Relaxing these assumptions would make for a richer and potentially more accurate predictive model for many domains. Additionally, we want to consider models that reflect temporal changes in the underlying data. Our present error model makes strong assumptions both about how errors are distributed, and what errors look like. On that front, an interesting line of future work would be to learn a of data entry errors and adapt our system to catch them. Finally, we plan to measure the practical impact of our system, by piloting USHER with our field partners, the United Nations Development Program's Millennium Villages Project in Uganda, and a community health care program in Tanzania. These organizations data quality concerns were the original motivation for this work, and thus serve as an important litmus test for our system.

III METHODOLOGY

In the proposed architecture the dynamic query form can be implemented as a web based system. Here a dynamic web interface for the query forms is implemented. The Generate Query algorithm executes a query form. A query form F is defined as a tuple (AF, RF, σF), which represents a database query template as follows:

F= (SELECT A1,A2,.....Ak FROM RF WHERE σF), Where, AF = {A1,A2,.....Ak} are k attributes for projection. AFis the set of columns of the result table. σFis the set of input components for users to fill or is a conjunction of expressions for selections on relations in RF. And RFis the relation. The database query is generated from the query form by using the Generate Query algorithm, based on the given set of projection attributes Au with selection expression σu. The query construction algorithm is used to get the attributes and conditions given by user each time and is given to Generate Query algorithm for query generation and execution. And the result is displayed to the user.

ISSN 2348 – 7968

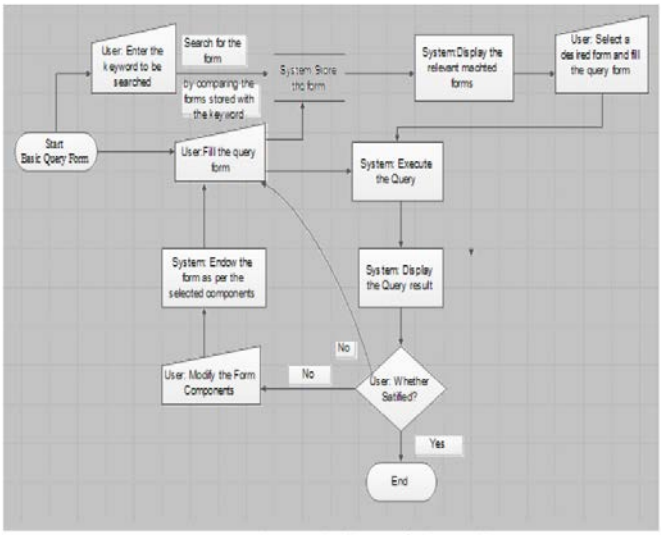


Fig.1: the work-flow of the architecture

Algorithm:

Data: $Q = \{Q1, Q2, \ldots, \}$ is the set of previous queries executed on Fi.
Result: $Qone$ is the query of One-Query
begin
σone - 0
for Q Q do
σone - $\sigma one \sigma Q$
$Aone$ - AFi
$Ar(Fi)$
$Qone$ - GenerateQuery($Aone, \sigma one$)

In this system the user can give the necessary "where" conditions and can make changes iteratively required to get the desired result. The "where" clauses given by each user is stored in the history table. So that when a user selects a table the history of "where" clauses used in that table will be displayed in the sorted form. And the user can select from this "where" clauses if needed. The where clauses displayed will be in the sorted form by taking the rank of each condition.

For a declarative query, to design a form, we must first the required results. Then we use information gathered from this analysis, as well as from the schema of the database, to create the necessary set of form-elements analyse it and identify its constraints and. Finally, we arrange these elements in groups, label them suitably, and lay them out in a meaningful way on the form. Thus our challenge is to design a good set of forms without having an actual query log at hand. In most cases the schema complexity is simply due to the richness of the data. This complexity is reflected in the queries to the database, many with more than one entity of interest. 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. Figure 1 shows the work-flow of Dynamic Query Form. 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. After submitting query form by user Dynamic Query Form System perform different task iteratively. First It create the query in accordance with the user query form generation after that system will execute the query and display the result. If user satisfied with the result then it end the process otherwise user select the interested form components from the displaying result and as per selected components by user system enrich the form and process again going to be work from initial fill query form stage in iteratively manner.

Modules :

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

1. Query Form Generation
2. Query Creation
3. Query Execution
4. Query Form Enrichment

1.Query Form Generation

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. In this way, a query form could be dynamically refined till the user satisfies with the query results. The form components here refer to the selection and projection components. DQF provides a two-level ranked list for projection components. The first level is the ranked list of entities. The second level is the ranked list of attributes in the same entity. The selection 0attributes must be relevant to the current projected entities; otherwise that selection would be meaningless. Therefore, the system should first find out the relevant attributes for creating the selection components. Here first describe how to select relevant attributes and then describe a naive method and a more efficient one-query method to rank selection components.

2.Query Creation

As the user give the required query form to the system then system works is to execute the query as per the user

ISSN 2348 – 7968

requirement. System will capture the similar types of components dynamically and generate the query forms. The creation of a query is to be estimates the goodness of the projection and selection form of the components. Here efficiency is important because Dynamic Query Form is an online system where user often expect quick response.

3.Query Execution

In the Dynamic Query Form Approach after creation of the appropriate query creation the system will be able to execute for displaying the appropriate result for the query generation. The user fills out the current query form and submits a query. Dynamic Query Form executes the query and shows the result. The user provides the feedback about the query result. Many database queries output a huge amount of data instances. In order to avoid this 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. Fig. 2 shows the flow of user actions.

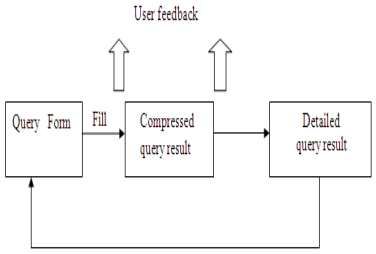


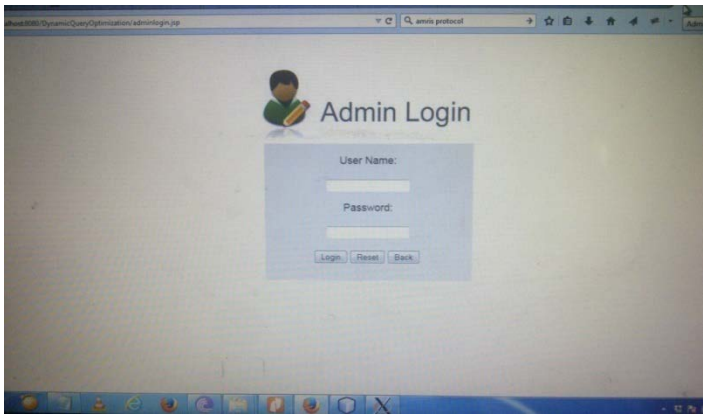
Figure 2: User Actions

4.Query Form Enrichment

As the query to be executed then the result to be displayed over the appropriate query. After seen the result if user will satisfied about the query result then it will end the procedure otherwise it will follow the procedure of query form enrichment. In the query form enrichment procedure dynamic query forms recommends a ranked list of query forms components to the user. The user selects the desired form components into the current query form and then selected component again to be followed over to the initial stage then again the forms to be generated according to the selected form components. Query forms are designed to return the user's desired result. There are two traditional measures to evaluate the quality of the query results: precision and recall. Expected precision is the expected proportion of the query results which are interested by the current user. Expected recall is the expected proportion of user interested data instances which are returned by the current query form. The user interest is estimated based on the user's click-through on query results.

Experimental Result And Analysis

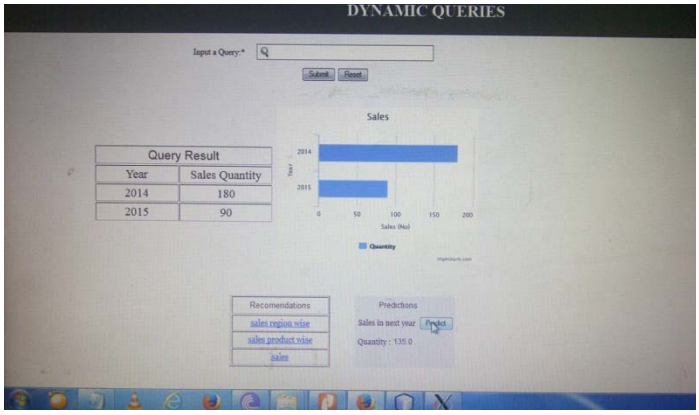
Here is the proposed system in that the working is to be done on netbeans platform. As the basic concept of the DQF to show the appropriate result to the user by using query techniques. So from the reference of it here we developed the web based application which showing the information related to the electronics products sales and management. So for the there is an admin login over there.



After login there were many activities admin can perform. But as considering to the basic concept of our dynamic query form approach the queries will show in the reports form.

As when admin insert any kind of query such as he wish to view the sales of all electronics products them he fires a query like sales then query get run and it retrieve the appropriate data from the related database and showing result to the user. Also as per the query i.e. sales so system shows the recommendations so that user will not satisfied with the query result then with the help of the recommendations he will able to search correct result.

ISSN 2348 – 7968



CONCLUSION

In this project 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 click-through. 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. As for the future work, our aim is to how this approach can be extended to non-relational data. 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. The relevance score between the keywords and the query form combining keyword search and forms for ad-hoc querying of databases can be incorporated into the ranking of form components at each step.

REFERENCES

[1] M. M. Zloof. Query-by-example: the invocation and definition of tables and forms. In *Proceedings of VLDB*, pages 1–14, Framingham, Massachusetts, USA, September 1975.

[2] EasyQuery. http://devtools.korzh.com/eq/dotnet/.

[3] Cold Fusion. http://www.adobe.com/products/coldfusion/.

[4] 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.

[5] 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.

[6] M. Jayapandian and H. V. Jagadish. Automating the design and construction of query forms. *IEEE TKDE*, 21(10):1389– 1402, 2009.

[7] E. Chu, A. Baid, X. Chai, A. Doan, and J. F. Naughton. Combining keyword search and forms for ad hoc querying of databases. In *Proceedings of ACM SIGMOD Conference*, pages 349–360, Providence, Rhode Island, USA, June 2009.

[8] 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.

[9] 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.

[10] G. Das and H. Mannila. Context-based similarity measures for categorical databases. In *Proceedings of PKDD 2000*, pages 201–210, Lyon, France, September 2000.

[11] W. B. Frakes and R. A. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, 1992.

[12] A. Seffah, M. Donyaee, R. B. Kline, and H. K. Padda. Usability measurement and metrics: A consolidated model. *Software Quality Journal*, 14(2):159–178, 2006.

[13] Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. Static Checking of Dynamically Generated Queries in Database Applications, Rhode Island, USA, September 2009.

[14] G. Wolf, H. Khatri, B. Chokshi, J. Fan, Y. Chen, and S. Kambhampati. Query processing over incomplete autonomous databases. In *Proceedings of VLDB*, pages 651–662, 2007.