International Journal of Innovative Research and Advanced Studies (IJIRAS)
Volume 3 Issue 5, May 2016

ISSN: 2394-4404

Design Enrichment of Query Forms For Database Query

Dhananjay

Professor, Department of Information Technology,
SVIT College of Engineering, Chinchili, Nashik,
Maharashtra, India

Kumbhakarna

Ashvini Awhad

Kirti Shinde

Dipalee Tidke

UG Student, Department of Information Technology,
SVIT college of Engineering, Chinchili,
Nashik, India

Abstract: ***The scientific databases & web databases maintain huge and large amount of data. The real-world databases contain over thousands of relations & attributes. predefined database query forms are not able to satisfy various queries from users on those databases. The review of DQF is to capture a user's preference and rating query form components, assisting to take decisions. The creation of a query form is an faster process and is given by the user. A user can also create the query form and submit queries to view the query output at each iteration. This way, a query form could be dynamically created till the user satisfies with the query forms. The important F-measure for measuring the goodness of a query form. A model is developed for estimating the goodness of a query form in DQF. Experimental evaluation and user study demonstrate the accuracy and performance of the system. 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 output at each step. This type a query form could be dynamically refined till the user satisfies with the query results.***

Keywords: Query Form, User Interaction, Query Form Generation

I. INTRODUCTION

Query form is one of the most widely 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. In many natural studies, such as genomics and diseases, the databases have over hundreds of entities for chemical and biological data resources. Many web databases, such as BigData and MongoDB, approximately have thousands of structured web entities. Therefore, it is hard to design a set of static query forms to satisfy various ad-hoc database queries on those complex databases.

Many old database management and development tools, like Easy Query, Cold Fusion, SAP and Microsoft word, provide several mechanisms to let users create customized queries on databases. The creation of customized queries totally depends on users' manual editing. If a user is not friendly with the database in advance, those thousands of data attributes would confuse and give the error.

II. EXISTING SYSTEM

Recently system is automatic approaches to create the database query forms without user interaction presented a data-driven method. It first finds a set of data relations, which are most likely queried based on the database schema and data instances. 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 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 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. It works well in the databases which have rich textual information in data tuples and schemas. It is not appropriate when the user does not have concrete keywords to describe the queries at the beginning, especially for the numeric attributes.

III. PROPOSED SYSTEM

A Dynamic Query Form system (DQF) , a query process 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., fetching query conditions) before identifying the last candidates. The essence of DQF is to capture user interests during user interactions and to adapt the query form iteratively. Each step consists of two types of user interactions: Query Form design and Query Execution. 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 outputs.

A. SYSTEM APPROACH

To propose a Dynamic Query Form system: DQF, a query interface which is capable of automatically 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 outputs. The essence of DQF is to capture user interests during user interactions and to adapt the query form iteratively. Each time consists of two types of user interactions: Query Form Enrichment and Query Execution (see Table 1). Figure 1 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 desing iteratively via the interactions between the user and our system until the user is satisfied with the query results. Mainly study the ranking of query form components and the dynamic generation of query forms.

Query Form Design	✓ DQF recommends a ranked list of query form components to the user. ✓ The user selects the desired forms components into the current query form.
Query Evalution	✓ The user fills the current query forms and submit the query. ✓ DQF evalutes the query and shows the results. ✓ The user provides the feedback about the query output.

Table 1: Interactions Between Users and DQF

B. MODULES

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

- ✓ Query Form Enrichment
- ✓ Query Execution
- ✓ Customized Query Form
- ✓ Database Query Recommendation

QUERY FORM ENRICHMENT

- ✓ DQF recommends a ranked list of query form components to the user.
- ✓ The user selects the form components into the current query form.

QUERY EXECUTION

- ✓ The user adds out the current query form and submit a query.
- ✓ DQF evaluate the query and shows the outputs.
- ✓ The user provides the feedback about the query outputs.

CUSTOMIZED QUERY FORM

The providing visual interfaces for developers to create or customize query forms. The error of those tools is that, they are provided for the professional developers who are familiar with their databases, not for last users. If proposed a system which allows last users to customize the existing query form at run time. An last user may not be friendly with the database.

DATABASE QUERY RECOMMENDATION

Recent studies introduce collaborative approaches to recommend database query components for database exploration.

C. AIM

Our contributions can be summarized as follows:

- ✓ Propose a dynamic query form system which creates the query forms according to the user's desire at run time. The system provides a answer for the query interface in large and complex databases.
- ✓ Apply F-measure to estimate the greatness of a query form. F-measure is a typical metric to evaluate query outputs. This metric is also accurate for query forms because query forms are designed to help users query the database. The greatness of a query form is determined by the query outputs created from the query form. Based on this, rating and recommend the potential query form components so that users can define the query form easily.

✓ Based on the proposed metric, develop efficient algorithms to estimate the greatness of the projection and selection form components. Here accuracy is important because DQF is an online system where users often expect quick response.

IV. SYSTEM ARCHITECTURE

This topic are trying to develop multiple methods to capture the users interest for the queries besides the click feedback. Adding a text-box for users to input some keywords queries. The relevance score between the keywords and the query form can be incorporated into the ranking of form components at each step.

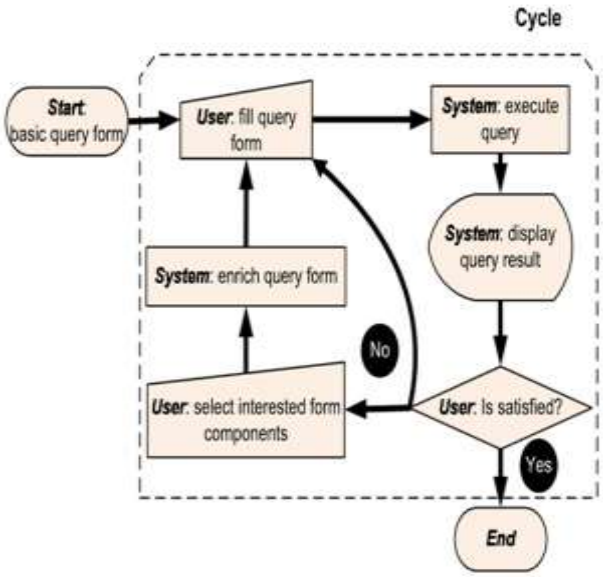

Figure 1: System Architecture

V. QUERY FORM INTERFACE

A. QUERY FORM

This part formally define the query form. Each query form corresponds to an SQL query template.

Definition 1: A query form F is defined as a tuple (AF, RF, σF, ◁▷ (RF)), which represents a database query template as follows:

F = (SELECT A1, A2, ..., Ak
FROM ◁▷ (RF) WHERE σF),
where AF = fA1, A2, ...,

A_kg are k attributes for projection, k > 0. RF = fR1, R2, ..., R_ng is the set of n relations (or entities) involved in this query, n > 0. Each attribute in AF belongs to one relation in RF. σF is a conjunction of expressions for selections (or conditions) on relations in RF. ◁▷ (RF) is a join function to create a conjunction of expressions for joining relations of RF.

The user interface of a query form F, AF is the set of columns of the output table. σF is the set of input components for users to fill. Query forms allow users to create parameters to generate different queries. RF and ◁▷ (RF) are not visible in the user 3 interface, which are usually created by the system according to the database schema. For a query form F, ◁▷ (RF) is automatically constructed according to the primary keys among relations in RF. Meanwhile, RF is determined by AF and σF. RF is the union set of relations which contains at least one attribute of AF or σF. Hence, the components of query form F are actually determined by AF and σF. As we mentioned, only AF and σF are visible to the user in the user interface. Focus on the projection and selection components of a query form. Ad-hoc join is not handled by our automatic query form because join is not a part of the query form and is invisible for users. As for ”Aggregation” and ”Order by” in SQL, there are limited options for users. For example, ”Aggregation” can only be MAX(maximum), MIN(minimum), AVG(average), and so on; and ”Order by” can only be ”increasing order” and ”decreasing order”. Our dynamic query form can be easily extended to include those options by implementing them as dropdown boxes in the user interface of the query form.

B. QUERY OUTPUTS

To decide whether a query forms is right or not, a user does not have time to go over each data step in the query outputs. In many database queries output a large amount of data instances. In series to avoid this “Multiple-Answer” problem, we only output a compressed result table to show a higher level view of the query outputs first. Each instance in the compressed table represents a cluster of actual data instances. The user can check through interested clusters to show the detailed data instances. Figure 2 shows the flow of user actions. The compact upper-level view of query outputs is proposed in. There are many one-pass clustering algorithms for generating the compressed view efficiently Certainly, different data clustering methods would have different compressed views for the users. Different clustering methods are preferable to different data types. Clustering is just to provide a goodness view of the query outputs for the user. The system developers can select a different clustering algorithm if needed.

.

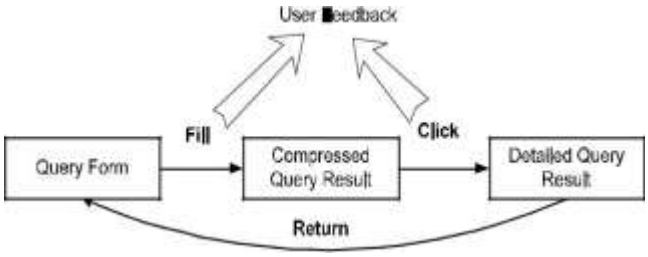

Figure 2: User Actions

VI. RANKING TABLE

Query forms are designed to return the user’s require output. There are two traditional measures to maintaining the quality of the query outputs: *precision* and *recall*. Query forms

are able to generate different queries by different inputs, and different queries can output different query outputs and achieve different *precisions* and *recalls*, so we use *expected precision* and *expected recall* to evaluate the expected performance of the query form. Intuitively, *expected precision* is the expected proportion of the query outputs 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 outputs displayed by the query form. For example, if some data instances are clicked by the user, these data instances must have high user interests. The query form components which can capture these data instances should be rating higher than other components. Next introduce some notations and then define expected precision and recall.

NOTATIONS

Lists the symbols used in this topic. Let F be a query form with selection condition σ_F and projection attribute set A_F. Let D be the collection of instances in $\triangleleft\triangleright(R_F)$. N is the number of data instances in D. Let d be an instance in D with a set of attributes $A = fA_1, A_2, ..., A_ng$, where $n = jAj$. We use d_{AF} to denote the projection of instance d on attribute set A_F and call it a projected instance. $P(d)$ is the occurrence probability of d in D. $P(\sigma_F\, jd)$ is the probability of d satisfies σ_F. $P(\sigma_F\, jd)$ 2 $f0, 1g$.

F	query form
RF	set of relations involved in F
A	set of all attributes in $\triangleleft\triangleright(R_F)$
AF	set of projection attributes of query form F
$A_r(F)$	set of relevant attributes of query form F
σF	set of selection expressions of query form F
OP	set of relational operators in selection
D	data instance in $\triangleleft\triangleright(R_F)$
D	the collection of data instances in $\triangleleft\triangleright(R_F)$
N	number of data instances in D
dA_1	data instance d projected on attribute set A_1
DA_1	set of unique values D projected on attribute set A1
Q	database query
DQ	results of Q
Duf	user feedback as clicked instances in D_Q
A	fraction of instances desired by users

Table 2: Symbols and Notations

P (σF jd) = 1 if d is returned by F and P (σF jd) = 0 otherwise.

Since query form F projects instances to attribute set A_F, we have D_{AF} as a projected database and $P(d_{AF})$ as the probability of projected instance d_{AF} in the projected database.

PROBLEM DEFINITION: In this topic, provide a rating list of query form components for the user. Problem 1 is the formal statement of the rating problem.

Problem 1: Let the current query form be F_i and the next query form be F_{i+1}, construct a rating of all candidate form components, in descending order of $F\ Score_E(F_{i+1})$, where F_{i+1} is the query form of F_i designed by the corresponding form component.

$F\ Score_E(F_{i+1})$ is the estimated greatness of the next query form F_{i+1}. Aim of the topic to maximize the greatness of the next query form, the form components are rating in descending order of $F\ Score_E(F_{i+1})$. In the next topic, discuss how to compute the $F\ Score_E(F_{i+1})$ for a specific form component.

VII. ESTIMATION OF RATING SCORE

A. RATING PROJECTION FORM COMPONENTS

DQF provides a 2nd level rating list for projection components. The 1st level is the rating list of entities. The 2nd level is the rating list of attributes in the same entity. 1st describe how to rank each entity's attributes locally, and then describe how to rank entities.

Algorithm 1 give detail in for the algorithm of the One-Query's query construction. The function Generate Query is to generate the database query based on the given set of projection attributes A one with selection expression σone.

ALGORITHM 1: Query Generation

DATA: $Q = fQ_1, Q_2, ..., g$ is the set of previous queries executed on F_i.
RESULT: Q_{one} is the query of One-Query
BEGIN
    σ_{one} 0
    for Q 2 Q **do**
        $^{\sigma}one\ ^{\sigma}one - ^{\sigma}Q$
        ^{A}o
    ne $A_{Fi}\ [\ A_r(F_i)$
    ^{Q}
one GenerateQuery(A_{one},σ_{one})

When the system receives the output of the query Qone from the database engine, it calls the second algorithm of One-Query to find the best query condition. 1st discuss the condition. The basic idea of this algorithm is based on a simple property. For a specific attribute As with a data instance d, given two conditions:

s1: As a1, s2: As a2, and a1 a2, if s1 is satisfied, then s2 must be satisfied. Based on this property, user could incrementally compute the F Score of each query condition by scanning one pass of data instances.

ALGORITHM 2: FindBestLessEqCondition

DATA: α is the fraction of instances desired by user,
D_{Qone} is the query result of Q_{one}, A_s is the selection attribute.
RESULT: s is the best query condition of A_s.
BEGIN
    // sort by A_s into an ordered set D_{sorted}
    Dsorted ←− Sort(DQ_{one} , *As*)

$s \leftarrow -\ \emptyset$, $f\ score \leftarrow -$
$0\ n \leftarrow - 0$, $d \leftarrow - \alpha\beta^2$
for $i \leftarrow 1$ **to** $|D_{sorted}|$ **do**
$d \leftarrow - D_{sorted}[i]$
$s \leftarrow -$ "$A_s \leq d_{As}$"
// compute fscore of "$A_s \leq d_{As}$"
$n \leftarrow - n + Pu(dA_{Fi})P\ (dA_{Fi})P\ (\sigma F_i\ |d)P\ (s|d)\ d \leftarrow - d + P\ (d_{AFi})P\ (\sigma_{Fi}\ |d)P\ (s|d)$
$f\ score \leftarrow - (1 + \beta^2) \cdot n/d$
if $f\ score \geq f\ score$
then $s \leftarrow - s$
$f\ score \leftarrow - f\ score$

COMPLEXITY: As for other query conditions, such as "=", " ", user can also find similar incremental approaches to compute their FScore. User can also share the sorting output in the 1st step. And for the 2nd step, all incremental computations can be merged into one pass of scanning D_{Qone} . The time complexity of finding the best query condition for an attribute is $O(jD_{Qone}\ j\ jA_{Fi}\ j)$. Ranking every attribute's selection component is $O(jD_{Qone}\ j\ jA_{Fi}\ j\ jA_r(F_i)j)$.

VIII. EVALUATION

The goal of our implementation is to check the following hypotheses:

H1: Is DQF more usable than older approaches such as static query form and customized query form?

H2: Is DQF more effective to rate projection and selection components than the baseline method and the random method?

H3: Is DQF efficient to rate the suggested query form components in an online user interface?

IX. CONCLUSION

Thus system is proposed that a dynamic query form creation approach which helps users dynamically create query forms. The key idea is to use a probabilistic model to rate form components based on user preferences. It captures user preference using both historical queries and run-time feedback such as click-through. Experimental outputs show that the dynamic approach often leads to greater success rate and simpler query forms compared with a static approach. The rating of form components also makes it simple for users to customize query forms. As future work, user will study how the approach can be extended to non relational data.

REFERENCES

[1] Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," *In Procceding VLDB*, vol. 54, no. 4, pp. 81–92, September 2003.

[2] R. Aggarwal, S. Gollapudi, A. Halverson, and S. Leong, "Diversifying search result," *In Procceding VLDB, Barcelona, Spain*, vol. 54, no. 4, pp. 5–14, February 2009.

[3] S. Aggarwal, S. Choudhari, G. Das, and A. Gionis, "Automated ranking of database query results," *In CIDR*, vol. 46, no. 4, pp. 14–21, February 2003.

[4] S. Boriah, V. Chandola, and V. Kumar, "Similarity measures for categorical data: A comparative evaluation," *In Proceedings of SIAM International Conference on Data Mining (SDM 2008)*, pp. 243–253, April 2008.

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