Survey Paper on Creation Of dynamic query Form for mining highly optimized transactional databases

Jayashri M. Jambukar,
PG student, Amrutvahini college of Engineering, Sangamner.

ABSTRACT

In New scientific databases and web databases maintain huge and heterogeneous data. These concrete world databases include over so many relations and attributes. Historic predefined query forms are not able to answer different ad -hoc queries from users on those databases. 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, sup port 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 w ith 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: *Form creation, Query Form, User Interaction.*

I. INTRODUCTION:

Query form is one of the most extensively used user interfaces for querying databases to access information. Historic query forms are configured and predefined by developers or Database Administrator in different information management systems. With the fast development of web information and scientific databases, new databases become very huge and difficult. In natural sciences, like genomics and diseases, the databases have number of entities for chemical and/or biological data resources. Different types of web databases, like Freebase and DBPedia, have thousands of structured web entities. Therefore, it is difficult to design a set of static query forms to answer various ad-hoc database queries on those difficult and complex databases.

Many existing database management and development tools, like EasyQuery, Cold Fusion, SAP and Microsoft Access, provide various mechanisms to let users generate customized queries on databases. But, the customized queries generation totally depends on users' manual editing's. If a user is not familiar with the database schema in advance, those hundreds or thousands of data attributes will confuse him or her.

1.1 INTERACTION BETWEEN USERS & DQF

A. Query Form Enrichment

1) Dynamic Query Form (DQF) recommends a ranked list of query form components to the user.

2) The user has to select the desired form components into the current query form. B. Query Execution

1) The user fills out the current query form and submits a query.

2) DQF will execute the query and the results are shown.

3) The feedback about the query results is provided by user.

II. SYSTEM ARCHITECTURE

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

A. Query Form Enhancement

B. Query Execution

C. Customized Query Form

D. Database Query Recommendation

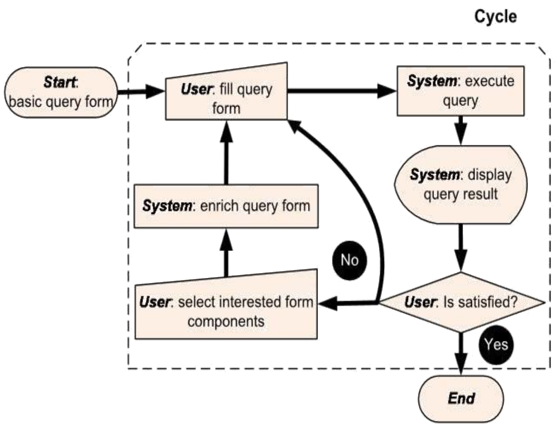


Fig 1: System Architecture

A. Query Form Enhancement

1) Dynamic Query Form endorses a ranked list of query form components for the user.

2) The user has to select the preferred form components into the current query form.

B. Query execution

1) The user has to fills out the current query form and submits the query.

2) DQF performs the query and displays the results.

3) The user offers the feedback on the query results.

C. Customized Query Form

These provide visual interfaces for developers to generate or customize query forms. The issue of those tools is that, they are for the professional programmer who is aware with their databases, but not for the end-users. It suggests a system which permits end-users to customize the existing query form at run time. But, the end-user may not be familiar with the database. If the database schema is very huge, it is hard for them to search specific database entities and attributes and to generate desired query forms.

D. Database Query Recommendation

Current studies introduce shared method to recommend database query components for database research. They consider SQL queries as elements in the collaborative filtering strategy, and proposes similar queries to relevant users.

III. ALGORITHM

Below algorithm shows the algorithm of the One-Query’s query creation. The function createQuery is to create the database query centered on the given group of projection attributes *Aone* with selection expression *σone*.

Data: $Q = \{Q1, Q2, ...,\}$ *This* is the set of earlier queries executed on *Fi*.

Result: *Qone* is the query of One-Query begin

```
σone <-- 0 for Q € Q do
σone.<-- σ one V σQ
Aone <-- AFi U Ar (Fi)
Qone <-- CreateQuery(Aone, σone)
```

When the system gets the result of the query *Qone* from database engine, it requests the second algorithm of One-Query to search[2] best query condition.

3.1 Query Form Interface

a) Query Form-

Every query form resembles to an SQL query template. Definition 1: A query form F is defined as a tuple(AF , RF , σF , ◃▹ (R)), this signifies a database query template like in below:

F = (SELECT A1,A2, ...,Ak
FROM ▹◃ (RF) WHERE σF),

where AF = {A1,A2, ...,Ak} are k attributes of projection, k > 0. RF = {R1,R2, ..., Rn }

which is the group of n relations included in this query, n > 0.Every attribute in AF will belong to one relation in RF. σF is the conjunction of expressions for selections on relations in RF. ▹◃(RF) is a join function to create a conjunction of expressions for joining relations of the RF .In user interface of a query form F, AF is the group of columns of result table. σF is group of input

components to fill for users. Query forms permit users to fill parameters to create various queries. RF and ▹◃ (RF) are not visible in end user interfaces, which are generally created by system as per the database schema. For query form *F*, ▹◃ (*RF*) is automatically constructed as per foreign keys among relations in *RF* .

In the meantime, *RF* is determined by *AF* and *σF* . *RF* is union group of relations which has at least one attribute of *AF* or *σF* So as, the components of query form *F* are in actual determined by *AF* and *σF* . As mentioned, only *AF* and *σF* are visible to user in user interfaces. We focus on projection & Selection components of a query form. Ad-hoc join is not touched by our dynamic query form because join is not a part of query formand is not visible for the users. As for"Aggregation" and"Order by" in SQL, there are limited choices for users. Like ,"Aggregation" can be MIN,MAX,AVG, and so on; and"Order by" can be "increasing order" & "decreasing order". Our dynamic query formcan be easily enhanced to include those options by implementing them as dropdown boxes in user interface of query form.

b) Query results-

To conclude if a query form is required or not, a user doesn't have time to go over each data instance in query results. Also, many database queries results a large amount of data instances. We only output a compressed output table to display a highlevel view of the query results. Every instance in compressed table signifies a group of actual data instances. Next, user can click through desired clusters to view detailed data instances. Below figure shows user action flow. The compressed view of query results will be proposed. There are many clustering algorithms for creating compressed view efficiently. For our implementation, we select incremental data clustering framework because of efficiency issue. Different clustering methods are preferable to different data types. Here, clustering is just to give a better view of query results for users. The system programmer can choose a various clustering algorithm if required.

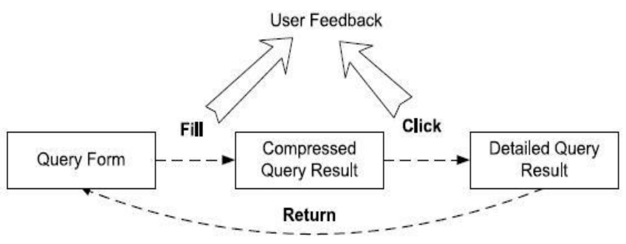


Fig 2: User Actions

c) Ranking Metrices

Query forms are developed to return user's anticipated result. There are two traditional measures to estimate quality of the query outputs: ***precision*** and ***recall.*** Query forms are able to generate different queries by various inputs, and various queries can output different query results and obtain different ***precisions*** and ***recalls***, so we are using *desired precision* and *expected recall* to calculate the expected performance of the query form. E*xpected precision* is the expected proportion of query results which are interested by user. *Expected recall* is expected proportion of user interested data instances which are returned by current query form. User interest is anticipated based on user's click through on query results showed by the query form.
Like, in case some data instances are clicked by user, those data instances should have vital user interests. So, query form components which can capture those data instances should be ranked at high than remaining components. Afterwards we introduce some notations and then define desired precision and recall.

ID	C1	C2	C3	C4	C5
I1	A1	B1	C1	30	1
I2	A2	B2	C2	30	100
I3	A3	B2	C3	40	99
I4	A4	B1	C4	30	1
I5	A5	B3	C4	20	2

Table 1:- Example of data table

Like take a query form with one relational data table as shown in the Table. There are 5 data instances in the table, *D* = *{I*1*, I*2*, ..., I*5} which has 5 data attributes *A* = *{C*1*,C*2*,C*3*,C*4*,C*5}= 5.

Query form executes a query *Q* as "SELECT *C*2, *C*5 FROM *D* WHERE *C*2 = B1 OR *C*2 = B2".
The query result is DQ = {I1, I2, I3 ,I4}which are projected on C2 and C5.
|
Thus P(σFi d) has 1 for I1 to I4 and has zero foe I3.Instance I1 and I4 has same Projected values.

So we use *I*1 to represent both of them and *P*(*I*1*C*2*;C*5) = 2*/*5.

IV. 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 [3]But the costs of using SQF and DQF to fulfill 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.

V. USABILITY METRICS

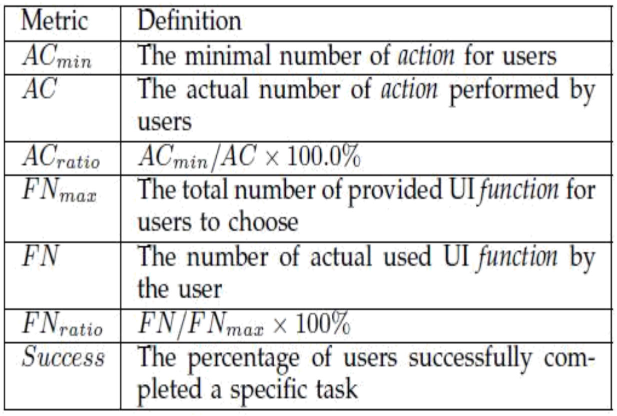
For database query forms, one *action* means a mouse click or a keyboard input of a textbox. *ACmin* is a minimal number of *actions* for a specific querying task. One *function* signifies a provided option for user to use, like a query form or a form component. In case of web page based system, *FNmax* is total number of UI components in web pages explored by users.

Here each page at most contains 5 user interface components. The smaller *ACmin*, *AC*, *FNmax*, and *FN*, the better will be the usability. And higher the *ACratio*, *FNratio*, and *Success*, the better will be the usability. There is a trade-off between *ACmin* and *FNmax*. The extreme case will be when, we create all possible query forms in one web page,and user only needs to select one query form to complete their query task, so *AC min* is 1. However, *FNmax* should be number of all possible query forms with their components, which can be a large number. On other side, when users have to interact a lot with a system, that system should know better about user's anticpation. In such case, the system would cut down many unwanted functions, so that *FNmax* will be smaller. But *ACmin* will be high since there are many of user interactions.

Effectivness-

Here we compare ranking function of DQF with other two ranking methods: baseline method and other is random method. Baseline method ranks projection and selection attributes in ascending order of their schema distance to current query form. In case of the query condition, it selects the most frequently used condition in training set for that particular attribute. Random method randomly proposes one query form component. Final truth of the query form component ranking is obtained from the query workloads.

Here we use some widely used metrics in Human-Computer Interaction and Software Quality for measuring the usability of a system. These metrics are listed in below Table:



Metric	Definition
AC_{min}	The minimal number of *action* for users
AC	The actual number of *action* performed by users
AC_{ratio}	$AC_{min}/AC \times 100.0\%$
FN_{max}	The total number of provided UI *function* for users to choose
FN	The number of actual used UI *function* by the user
FN_{ratio}	$FN/FN_{max} \times 100\%$
$Success$	The percentage of users successfully completed a specific task

Table 2:- Usability Metric

VI. CONCLUSION

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.

REFERENCES

[1] DBPedia. ht tp :// D BPe dia .or g.

[2] Easy Query. h ttp:/ /devt ool s.k or zh .com/eq /d otne t /

[3] F reeb ase . Http: // w w w .freeb ase .c om.

[4] C.C.Aggarwal,J Han, J. Wang,& P. S. Yu. A frameworkfor clustering evolving data streams. In *Proceedings of VLDB*, pages 81–92, Berlin, Germany, Sept 2003.

[5] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *CIDR*, 2003.

[6] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic information retrieval approach for ranking of database query results. *ACM Trans. Database Syst. (TODS)*, 31(3):1134–1168, 2006.

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