www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume – 5 Issue -02 February, 2016 Page No. 15744-15748

Adaptive Query Interface for Database Search

Mr.Prashant S. Chavan, Prof. Dr.B.D.Phulpagar
Postgraduate Research Scholar
PESMCOE, Pune, India
Department of Computer engineering
PESMCOE, Pune, India

Abstract- Modern knowledge bases contain huge data with complex relations between the attributes. From these sort of databases obtaining satisfactory results is troublesome task. Use of traditional predefined query interfaces during this sort of databases doesn't provide satisfactory results. projected system generates query interface forms with user participation. User will provide feedback by click through therefore capturing user's preference. Query form is adaptive since it dynamically refined until user gets satisfactory results.

Index Terms- information schema, query form, foreign key

1. INTRODUCTION

Information is merely as helpful as its query interface permits it to be. If a user is unable to convey to the information what he or she desires from it, even the richest knowledge store provides very little or no worth. Static query forms or predefined query forms area unit utilized by the DBA to retrieve the knowledge from the information. however current used databases contain variety of attributes and relations. So, retrieving info with static query forms is troublesome. Conjointly it's impractical to style static query kind with too several attributes to handle. Several direction tools give mechanisms to style predefined query forms. The method is advanced as a result of user should manually edit to style predefined query forms. If a user is unaware of the information schema then handling attributes within the method of coming up with predefined query forms becomes too advanced to handle.

Adaptive query Interface is a approach that generates program dynamically. In static query forms obtaining desired result's one step method however if the information is big, user find obtaining too several instances of results and therefore desired info is inadequate. Projected approach uses several rounds of actions as inputted by the user to get adaptive query forms dynamically. Since filtration of results is predicated on user actions. Methods are often extended until satisfactory result or satisfactory varieties of filtered results are often found. Projected approach is additionally helpful to the non skilled user. It starts with a basic query type that contains only a few primary attributes of the information. The essential query type is then enriched iteratively via the interactions between the user and our system till the user is satisfied with the query results. In this paper, we have a tendency to chiefly study the ranking of query type elements and also the dynamic generation of query forms. Query forms generated once more are often refined in line with user feedback and dynamically be modified therefore name given as Adaptative Query Interface. Figure one shows the flow diagram of the method

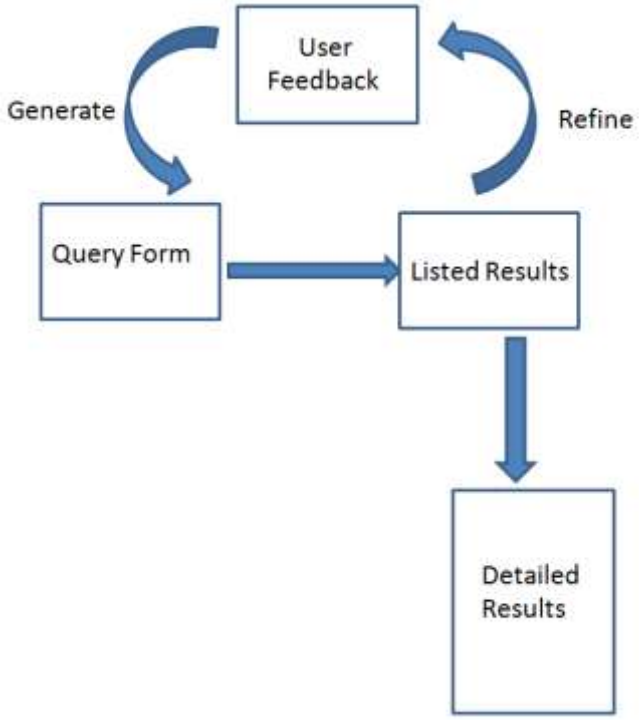

Fig.1. User Participation

2. LITERATURE SURVEY

Current challenge in retrieving info from large fashionable information is to let non skilled user create use of relative databases. therefore during this space, chiefly work is concentrated on a way to generate query forms so while not knowing the fields of information schema non skilled user conjointly can able to fetch info. Presently, query forms area unit accustomed meet this want up to some extent. To spotlight a 1 query by Example is one sort of information querying interface. Existing information purchasers like Microsoft access can also be accustomed give interface to developers to form customise query forms. However to use this tool one have to be compelled to understand the information schema therefore it's helpful to developer and to not user. Paper [3] proposes a system that is automatic

approach to get query forms. Here user participation isn't necessary, system initial finds knowledge attributes in schema and consequently generates query forms. though this method having advantage of automatic generation. it's not appropriate for the information schemas having thousands of attributes. If variety of attributes area unit quite kinds that area unit generated area unit too several in numbers and therefore it's confusing for user that form is to be used. Therefore, during this approach end product isn't satisfactory. Paper [5] can also be taken on similar lines as explained.

To overcome downside of said approach paper [1] proposes a system which might be aforementioned as extension of labor [3] and [5]. during this paper they enclosed feature of keyword looking within the generated kinds therefore user currently will realize that form are often used for looking. Therefore system generates ton of query forms beforehand and user then searches the forms with keywords. This method although takes downside from the higher than system its best fitted for information schemas that involves concrete keywords for attributes. However during this system it should be noted that this comes with the disadvantage of knowing the schema beforehand. It means that user should understand the information schema to look desired forms.

3. IMPLEMENTATION DETAILS

3.1 Design

Definition 1: A query form F is defined as a tuple (A_F , R_F , σ_F, $\bowtie$ (R_F)), which represents a database query template as follows:

F = (SELECT A_1,A_2, ..., A_k From $\bowtie$ (R_F) WHERE σ_F),
where A_F = {A_1,A_2, ..., A_k} are k attributes for projection and k > 0.

R_F = {R_1,R_2, ...,R_n} is the set of n relations (or entities) involved in this query, n > 0.

Each attribute in AF belongs to one relation in RF .σ_F is a conjunction of expressions for selections (or conditions) on relations in R_F. $\bowtie$ (R_F) is a join function to generate a conjunction of expressions for joining relations of R_F. In the user interface of a query form F, A_F is the set of columns of the result table. σ_F is the set of input components for users to fill. Query forms allow users to fill parameters to generate different queries. R_F and $\bowtie$ (R_F) are not visible in the user interface, which are usually generated by the system according to the database schema.

For a query form F, $\bowtie$(R_F) is automatically constructed according to the foreign keys among relations in R_F . Meanwhile, R_F is determined by A_F and σ_F . R_F is the union set of relations which contains at least one attribute of A_F or σ_F . Hence, the components of query form F are actually determined by A_F and σ_F. As we mentioned, only A_F and σ_F are visible to the user in the user interface. 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. As for "Aggregation" and "Order by" in SQL, there are limited options for users. For example, "Aggregation" can only be MAX, MIN, AVG, and so on; and "Order by" can only be "increasing order".

To decide whether generated query interface is desired or not, it is difficult to decide by checking every instance of the result. This give rise to many answer problem. To address this, only compressed results can be shown with higher level view. Furthermore to get accuracy, user can participate and get results from required category. In first pass of the results since we are targeting to view results in sets, where set means results having same type of results. These results can be clustered by using clustering algorithm [4]. These clustered results can be explored according to the user click through. Figure 2 shows the flowchart of the process.

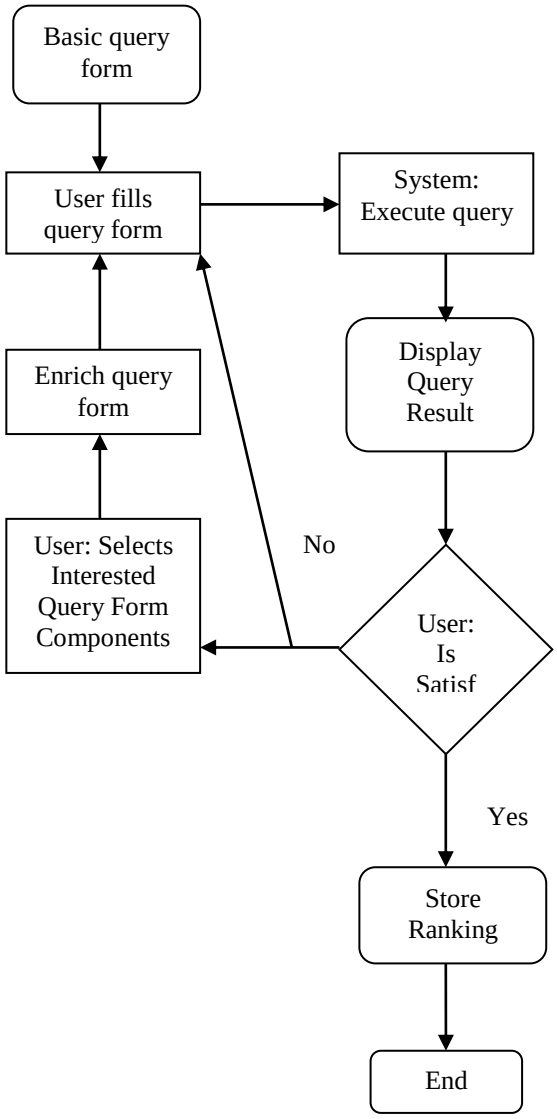

Fig. 2. Flowchart of Adaptive Query Interface

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 [15]. 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. It can help our system generate recommended form components that help users discover more desired data instances. 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. 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.

4. RANKING METRIC

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. Query forms are able to produce different queries by different inputs, and different queries can output different query results 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 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 displayed by the query form. For example, if some data instances are clicked by the user, these data instances must have high user interests. Then, the query form components which can capture these data instances should be ranked higher than other components. Next we introduce some notations and then define expected precision and recall.

Let F be a query form with selection condition σF and projection attribute set A_F. Let D be the collection of instances in $\triangleright\triangleleft$ (R_F). N is the number of data instances in D. Let d be an instance in D with a set of attributes A = {A_1,A_2, ...,A_n}, where n = |A|.

F	query form
R_F	set of relations involved in F
A	set of all attributes in $\triangleright\triangleleft$ (RF)
A_F	set of projection attributes of query form F
A_r(F)	set of relevant attributes of query form F
σ_F	set of selection expressions of query form F
O_P	set of relational operators in selection
d	data instance in $\triangleright\triangleleft$ (RF)
D	the collection of data instances in $\triangleright\triangleleft$ (RF)
N	number of data instances in D
d_{A1}	data instance d projected on attribute set A_1
D_{A1}	set of unique values D projected on attribute set A_1
Q	database query
D_Q	results of Q
D_{uf}	user feedback as clicked instances in DQ
α	fraction of instances desired by users

Table 1: Symbols and notations

We use d_{AF} to denote the projection of instance d on attribute set A_F and we call it a projected instance. P(d) is the occurrence probability of d in D. P(σ |d) P(σ_F |d) = 1 if d is returned by F and P(σ_F |d) = 0 otherwise. Since query form F projects instances to attribute A_F, we have D_{AF} as a projected database and P(d_{AF}) as the probability of projected instance d_{AF} n the projected database. Since there are often dupliated projected instances, P(d_{AF}) may be greater than N. Let P_u(d) be the probability of d being desired y the user and Pu(d_{AF}) be the probability of the user being interested in a projected instance. We give an example below to illustrate those notations.

ID	C1	C2	C3	C4	C5
I1	a1	b1	c1	20	1
I2	a1	b2	c2	20	100
I3	a1	b2	c3	30	99
I4	a1	b1	c4	20	1
I5	a1	b3	c4	10	2

Table 2: Data Table

Example 1: Consider a query form F_i with one relational data table shown in Table 3. There are 5 data instances in this table, D = {I_1, I_2, ..., I_5}, with 5 data attributes A = {C_1,C_2,C_3,C_4,C_5}, N = 5. Query form F_i executes a query Q as "SELECT C_2, C_5 FROM D WHERE C_2 = b1 OR C_2 = b_2". The query result is DQ = {I_1, I_2, I_3, I_4} with projected on C_2 and C_5. Thus P(σFi |d) is 1 for I_1 to I_4 and is zero for I_5. Instance I_1 and I_4 have the same projected values so we can use I_1 to represent both of them and P(I) = 2/5.

We now describe the two measures expected precision and expected recall for query forms.

Definition 2: Given a set of projection attributes A and a universe of selection expressions σ, the expected precision and expected recall of a query form F=(A_F , R_F , σ_F , $\triangleleft\triangleright$ (R_F)) are $Precision_E$(F) and $Recall_E$(F) respectively,

$$Precision_E(F) = \frac{\sum_{d\in DAF} P_u(d_{AF})P(d_{AF})P(\sigma_F|d)N}{\sum_{d\in D} P(d_{AF})P(\sigma F|d)N} \quad \text{.....Eq.(1)}$$

$$Recall_E(F) = \frac{\sum_{d\in DAF} P_u(d_{AF})P(d_{AF})P(\sigma_F|d)N}{\sum_{d\in D} P(d_{AF})P(\sigma F|d)N} \quad \text{.....Eq.(2)}$$

where $A_F \subseteq A$, $\sigma F \in \sigma$, and α is the fraction of instances desired by the user, i.e., $\alpha = \sum_{d\in D} P_u(d)P(d)$. The numerators of both equations represent the expected number of data instances in the query result that are desired by the user. In the query result, each data instance is projected to attributes in A_F. So $P_u(d_{AF})$ represents the user interest on instance d in the query result. $P(d_{AF})N$ is the expected number of rows in D that the projected instance d_{AF} represents. Further, given a data instance d ∈ D, d being desired by the user and d satisfying σF are independent. Therefore, the product of $P_u(d_{AF})$ and P(σ_F |d) can be interpreted as the probability of d being desired by the user and meanwhile d being returned in the query result. Summing up over all data instances gives the expected number of data instance in the query result being desired by the user.

Considering both expected precision and expected recall, we derive the overall performance measure, expected F-Measure as shown in above equation. Note that β is a constant

parameter to control the preference on expected precision or expected recall.

The system implementation we have decided to use Oracle SQL as back end and ASP.net or JSP as front end. Since it can be deployed as web interface, the system is platform independent.

5. DATASET

Implemented system uses student database as input to the system. We have taken all the information related to academics for the students. It contains personal as well as academic information. Whole information is organised in 10 tables comprising 116 attributes. According to the search, instances are projected by combining all the selected attributes.

6. ALGORITHMIC APPROACH

For projection of query attributes are selected form various tables. This attributes are refined according to the selection of the attributes by the user. The said approach is taken care of by the query construction

Query Construction:

Data: $Q_f \leftarrow A_1 \cup A_2 \cup A_3 \cup .... \cup A_i$

Result: Q_f is the final query

Begin:

$\sigma_{(one)} \leftarrow \emptyset$

for $Q \in Q$ do

$\sigma_{(one)} \leftarrow \sigma_{(one)} \cup \sigma Q_f$

$A_{(one)} \leftarrow A_F \cup Ar_F$

$Q_{(one)} \leftarrow GenerateQuery(A_{(one)}, \sigma_{(one)})$

$F \leftarrow Project\ (A_{(one)}, \sigma_{(one)})$

Where, Q_f is query form

A_1-A_i: Attributes

$Q_{(one)}$: Representative query

Thus, as above algorithm suggests query selection can be repetitively refined till we get satisfactory results.

7. DESIGN AND IMPLEMENTATION

In system, two login types are maintained. First one is admin login where we can add and delete users who are expected users. Second one is for normal users for database searching. As shown in the following form left side pane maintains the list of attributes and the result instances are projected according to the selection. As user selects attributes, in each iteration attributes are ranked according to the user preference.

Fig. 3 Query Interface of Searching

After getting result we can getcollective information of single instance by clicking on it. As shown in figure below it includes all the information of single student.

Fig. 4 Detailed information of single instance

8. RESULTS

Result shows adaptive query interface is more effective than that of static query interface. As we can see, as we iterate the searching, accuracy and efficiency of the searching increases.

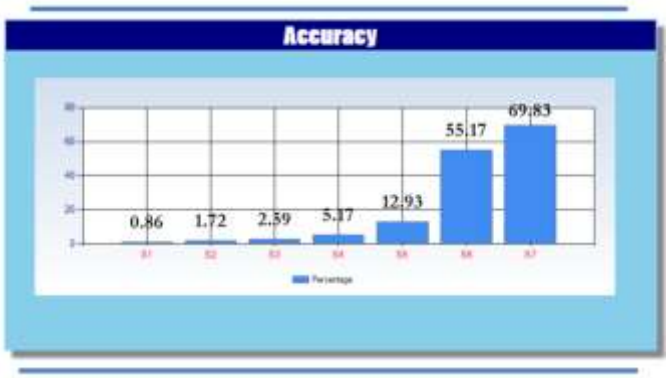

Fig. 5 Incremental improvement in accuracy

9. CONCLUSION AND FUTURE SCOPE

In this paper, idea of adaptive query interface is proposed. This system generates query forms dynamically. To capture user feedback run time click through process is used which helps in filtering of the results. From the related concepts studied we can conclude that success rate in this approach will be higher as compared to static approach.

For future scope, as we have used student information as input various algorithm can be applied to find patterns in student performance. Likewise, we may also predict future dropout and failures in student academics.

10. REFERENCES

[1] Eirinaki M., Abraham S., Polyzotis N., Shaikh N. QueRIE: Collaborative Database Exploration in IEEE Transactions on Knowledge and Data Engineering, (Volume:PP , Issue: 99), May 2013.

[2] 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.

[3] 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.

[4] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In Proceedings of VLDB, pages 81–92, Berlin, Germany, September 2003.

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

[6] Liang Tang, Tao Li, Yexi Jiang, and Zhiyuan Chen. Dynamic Query Forms for Database Queries,IEEE transaction on knowledge and data engineering, March'2013.

[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] 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.

[9] XMark: An XML Benchmark Project: http://www.xml-benchmark.org/.

[10] Magesh Jayapandian, Adriane Chapman, et al. Michigan Molecular Interactions (MiMI): Putting the Jigsaw Puzzle Together. Nucleic Acids Research (Database Issue), 35, 2007.

[11] Daniele Braga, Alessandro Campi, and Stefano Ceri. XQBE (XQuery By Example): A Visual Interface to the Standard XML Query Language. ACM TODS, 30(2), 2005.

[12] Shishir Gundavaram. CGI Programming on the World Wide Web. O'Reilly, 1996.

[13] D. J. Helm and B. W. Thompson. An Approach for Totally Dynamic Forms Processing in Web-Based Applications. In ICEIS (2), 2001.

[14] Sergey Melnik. Generic Model Management: Concepts and Algorithms. Chapter 7. PhD thesis, University of Leipzig, 2004.

[15] Anders Tornqvist, Chris Nelson, and Mats Johnsson. XML and Objects-The Future for E-Forms on the Web. In WETICE. IEEE Computer Society, 1999.

[16] T. Joachims and F. Radlinski. Search engines that learn from implicit feedback. IEEE Computer (COMPUTER), 40(8):34–40, 2007.

[17] 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.