IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. III (Sep – Oct. 2014), PP 42-49
www.iosrjournals.org

Dynamic Query Processing and Optimization in Relational Complex Databases with Keyword Search

Remya Raj[1], Rosna P. Haroon[2]
[1]*Dept. of CSE, Ilahia College Of Engineering and Technology, Kerala,India*
[2]*Dept. of CSE, Assistant Professor, Ilahia College Of Engineering and Technology, Kerala, India*

Abstract: *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 while continuously viewing the changing results. The main drawback of a form-based query interface is that it is restrictive. Modern scientific databases and web databases maintain large and heterogeneous data. These real-world databases contain over hundreds or even thousands of relations and attributes. Traditional predefined query forms are not able to satisfy various ad-hoc queries from users on those databases. So a Dynamic Query Form (DQF) system was proposed. Dynamic queries interfaces suggest that they offer a dramatic change from existing methods for querying databases. DQF, a query interface which is capable of dynamically generating query forms for users. Different from traditional document retrieval users in database retrieval are capable of performing different round of actions iteratively by filling in forms and retrieving results based on users desire and conditions at run time. Each iteration consists of two types of user interactions: Query Form Endowment and Query Execution. In Query Form Endowment DQF recommends the user selects the desired form components into the current query form and the form components selected by the user will be in ranked order. In Query Execution the user fills out the current query form and submits a query, DQF executes the query and shows the results and the user provides the feedback about the query results. F-measure value can be used for measuring the goodness of a query form. A comparison between the Customized query form and Dynamic query form can be generated. As an enhancement, the database querying can be formalized by combining the keyword search and forms. Here at query time, a user with a question to be answered issues standard keyword search queries; but instead of returning tuples, the system returns forms relevant to the question. This paper lacks the addressing of multi-criteria decision making. So skyline query concept is used for satisfying the multi-criteria decision making.*

Key words: *Dynamic Query, Skyline Queries, Query Forms*

I. Introduction

A form-based query interface is usually the preferred means to provide an unsophisticated user access to a database. Not only is such an interface easy to use, requiring no technical training, but it also requires little or no knowledge of how the data is structured in the database. One of the simplest ways to query a database is through a form, where a user can fill in relevant information and obtain desired results by submitting the form. However, a typical form is static and can express only a very limited set of queries. Without room for change, query specification is limited by the expertise and vision of the interface developer at the time the form was created. If an available form cannot express a desired query, the user is stuck. Filling forms is easy, the user needs only little bit training to learn how to fill a form correctly. In contrast, traditional database querying requires users to be able to write code in SQL (or XQuery) and to also know the database schema. It is no wonder that form-based query interfaces are so widely used by databases today. The main drawback of a form-based query interface is that it is restrictive.

Form design has two conflicting goals: forms should be simple to understand, and at the same time must provide the broadest possible querying capability to the user. 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. Therefore in this paper a dynamic query form approach is proposed. 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. Dynamic queries interfaces suggest that they offer a dramatic change from existing methods for querying databases. While languages such as SQL have become standardized and form fill-in interfaces have become widespread, now dynamic queries empower users to perform far more complex searches. Dynamic queries are an application

of the direct manipulation principles in the database environment. Dynamic queries provide a visual alternative for querying database depending on datatypes and the values decides the input control for search.

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 capable of performing different round of actions iteratively by filling in forms and retrieving results based on users desire and conditions at run time.

As the success of Internet search engines makes abundantly clear, when faced with discovering documents of interest, the general public is successful at using keyword search to accomplish the task. So as an enhancement, the database querying can be formalized by combining the keyword search and forms. Here at query time, a user with a question to be answered issues standard keyword search queries; but instead of returning tuples, the system returns forms relevant to the question.

The concept of multicriterian desicion making can be in cooperated here. So skyline query concept can be used for addressing multicriterian decision making.

II. Related Works

Existing database clients and tools make great efforts to help developers design and generate the query forms [2].They provide visual interfaces for developers to create or customize query forms. This system allows end-users to customize the existing query form at runtime. It provides a mechanism to let a user existing form to express the desired query. The modifications are specified through filling forms to create an expression. But an end-user may not be familiar with the database. If the database schema is very large, it is difficult for them to find appropriate database entities and attributes and to create desired query forms.

A mechanism to overcome the challenges that limit the usefulness of forms, namely their restrictive nature and the tedious manual effort required to construct them is proposed. An algorithm to generate a set of forms automatically given the expected query workload is introduced [3]. This is workload-driven method. Helps to bring novice users closer to the rich database resources they need to use, and maximize their efficiency with a sizeably reduced learning curve. But limitation of automated form generation with minimal human input. If the database schema is large and complex, user queries could be quite diverse. Finding an appropriate query form will be a challenging task.

SnipSuggest is a context-aware, SQL autocomplete system. SnipSuggest is motivated by the growing population of non-expert database users, who need to perform complex analysis on their large-scale datasets [4]. SnipSuggest aims to ease query composition by suggesting relevant SQL snippets, based on what the user has typed so far. SnipSuggest produces context-aware suggestions. Provides a user interface that assists the user to type the database queries based on the query workload, the data distribution, and the database schema [4]. But the queries mentioned here are in the forms of SQL and keywords. Difficult for a novice user who need to access a complex database.

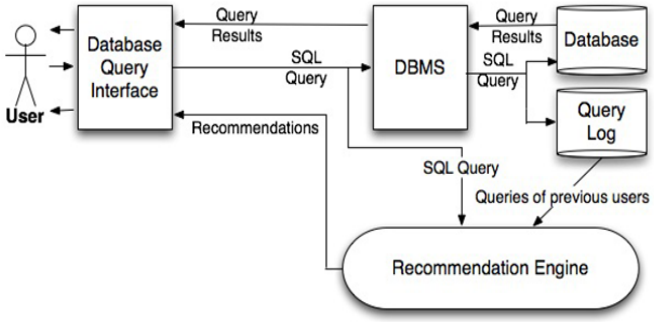
Keyword Search for Querying of Database generates a lot of query forms in advance. The user inputs several keywords to find relevant query forms from a large number of pregenerated query forms. The system proposes to take as input a target database and then generate and index a set of query forms offline [5]. At query time, a user with a question to be answered issues standard keyword search queries; but instead of returning tuples, the system returns forms relevant to the question. The user may then build a structured query with one of these forms and submit it back to the system for evaluation. It works well in the databases which have rich textual information in data tuples and schemas. But it is not appropriate when the user does not have concrete keywords to describe the queries at the beginning. Sometimes the result may not be appropriate after using all of the existing query forms.

A novel data-driven approach, called Query By Output (QBO) is presented here, which can enhance the usability of database systems. The central goal of QBO is as follows: Given the output of some query Q on a database D, denoted by Q(D), we wish to construct an alternative query Q0 such that Q(D) and Q0(D) are instance-equivalent.QBO targeted at improving the usability of database management systems[6]. QBO proposes a method to recommend an alternative database query based on results of a query. Does not require the knowledge of the query. Can be used in a simple database, where the user knows what he wants. But in case of a large database predicting an output becomes difficult, hence the method won't work.

Domain independent system that provides effective minimum-effort based faceted search solution over enterprise db. Present relevant facetes for the users according to their navigation path. Dynacet is a middleware system that sit between user and the db & dynamically suggest facetes for drilling db[7]. Dynamic faceted search engine is similar to our dynamic query form if we only consider selection component in a query. Besides selection, db has other important component called projection component. Projection component control the output of the query form. Faster access to information stored in databases while taking into consideration the variance in user knowledge and preferences. But Dynacet considers only selection component of a query.

Data quality is a critical problem in modern db Here present USHER, an end to end system for form design, entry & data quality assurance. It's a model for detecting and mitigating errors[8]. Before entry, it induces a form layout that captures the most important data values of a form instance as quickly as possible. During entry, it dynamically adapts the form to the values being entered, and enables real-time feedback to guide the data enterer towards their intended values. USHER's approach provides a framework for reasoning about and organizing feedback mechanisms for data-entry user interface. But USHER is a dynamic data entry form that deals with only data-entry form instead of database query forms.

A collaborative approach to recommend db query form for db exploration is introduced here. Users employ a query interface (typically, a web-based client) to issue a series of SQL queries that aim to analyze the data and mine it for interesting information. Treat SQL as items , recommend similar queries to related users[9]. The idea is: to track the querying behavior of each user identify which parts of the database may be of interest for the corresponding data analysis task and recommend queries that retrieve relevant data. A query recommendation framework supporting the interactive exploration of relational databases are presented. The information flow of the QueRIE framework :



Treat SQL queries as items, and recommend similar queries to related users. But Query Recommendations for Interactive Database Exploration do not consider the goodness of the query.

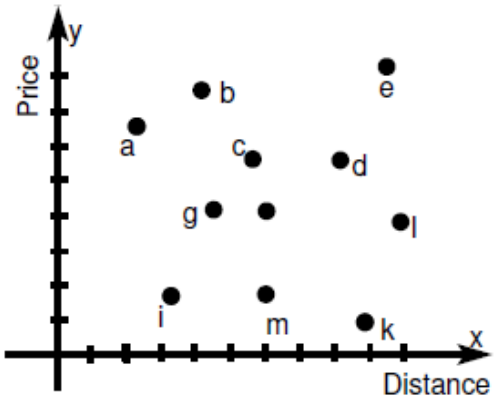
Helpdesk databases are used to store past interactions between customers and companies to improve customer service quality. Assuming that the helpdesk databases are organized into a number of cases, where each case contains: The interactions between a customer and the service team about a particular problem [10]. The features extracted from these interactions (e.g., initial information provided by the customer, clarification questions asked by the service team and answers from the customer, etc.), and The final recommendations by the service team. Develops the active featuring probing technique for automatically generating clarification questions to provide appropriate recommendations to users in database search. Different from this work which focuses on finding the appropriate questions to ask the user, DQF aims to select appropriate query components. Different from this work which focuses on finding the appropriate questions to ask the user, DQF aims to select appropriate query components.

Incompleteness due to missing attribute values (aka "null values") is very common in autonomous web databases. A novel query rewriting and optimization framework QPIAD is used [11]. The technique involves reformulating the user query based on correlations among the database attributes. The reformulated queries are aimed at retrieving the relevant possible answers in addition to the certain answers. QPIAD helps in reducing the costs of database query processing.

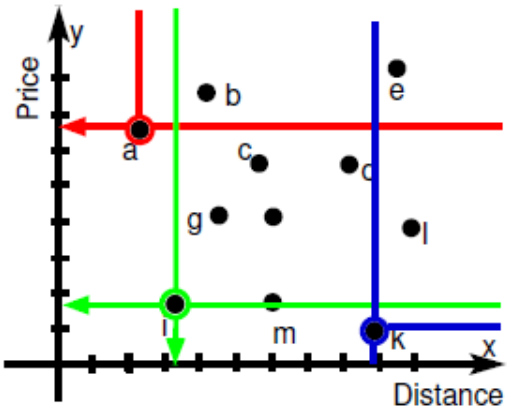
The skyline query concept can be used for satisfying the multi-criteria decision making. In a database, a Skyline is a set of tuples of information (points) which stand out among the others because are of special interest to us. A Skyline is defined as those points which are not dominated by any other point [12]. A point dominates another point if it is as good or better in all dimensions and better in at least one dimension.

Definition: Given a set of d-dimensional points, a skyline query returns a subset of points that are interesting in that they are not dominated by other points. Point p1 dominates p2 if p1 is better than p2 in at least one dimension and no worse than p2 in any other dimension.

Example: A dataset containing information about hotels; the distance to the beach and the price for each data point is recorded. Consider a two dimensional plot of the dataset, where the distance and price are assigned to the X,Y ax is of the plot.



The goal of the search is to find a hotel whose distance to the beach and the price are both minimum (not restricted to minimum, any other function max, join, group-by clause could be used.) the preference function in our example is "minimum price and minimum distance". The dataset may not have one single data point that satisfies both these desirable properties the user is presented with a set of interesting points that partly satisfy the imposed constraints. A dataset containing information about hotels; the distance to the beach and the price for each data point is recorded. Consider a two dimensional plot of the dataset, where the distance and price are assigned to the X,Y axis of the plot. The interesting data-points in the dataset are {a, i, k}. a has the least distance from the beach, k has the lowest price, i has a lesser distance value than k and a lower price value than a. i is hence not dominated by either one of them. All other points in the dataset are dominated by the set of points {a, i, k} i.e, both the distance and price values are greater than one or more skyline points.



III. Proposed System

The proposed architecture consists of a Dynamic Query Form (DQF) system for the query processing, optimization of a relational complex database and also allows the keyword based search of the dynamically generated query forms. The Fig.1 shows the work-flow of the architecture.

The work-flow 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. The user first fills the basic query form and submits the query form. At the time of submission the query relevant to the filled query form will be generated and at the same time this filled query form will be stored in the database and then the system will execute the query. The system then displays the query results. If the user is satisfied with the result then the user can end the process. If the user is not satisfied with the result the user can move through any of the 2 processes. First is, modify the above submitted form by adding new selection and projection components into the form. The system then endows the form as per the selected components and can fill the form again and submit to get the desired result.

Second is, the user can directly use the basic form itself without adding any new projection and selection components by just refilling it with other values. This is the basic part of the proposed architecture. Next part is the keyword based search. Here, at the start of the work-flow of the system the user is given with a

textbox area where the user can enter the keyword for formalizing the database querying by combining the keyword search and forms. Here at query time, a user with a question to be answered issues standard keyword search queries; but instead of returning tuples, the system returns forms relevant to the key word entered by the user. First, the user has to enter the keyword to be searched. The keyword will be compared with the stored form and relevant form based on the match will be retrieved by the system. The user can then select the desired form, fill in the form and can submit the form for further processing.

IV. Solution 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 GenerateQuery algorithm executes a query form. A query form F is defined as a tuple (A_F, R_F, σ_F), which represents a database query template as follows:
F= (SELECT A_1,A_2,.....A_k FROM RF WHERE σ_F),
Where, A_F = {A_1,A_2,.....A_k } are k attributes for projection. A_F is the set of columns of the result table. σ_F is the set of input components for users to fill or is a conjunction of expressions for selections on relations in R_F. And R_F is the relation.

The database query is generated from the query form by using the GenerateQuery algorithm, based on the given set of projection attributes A_u with selection expression σ_u. The query construction algorithm is used to get the attributes and conditions given by user each time and is given to GenerateQuery algorithm for query generation and execution. And the result is displayed to the user.
Algorithm: Query Construction

Data: Q= { Q_1,Q_2,Q_3,.....} is the set of previous queries executed on F_i.
Result: Q_u is the query
Begin
$\sigma_u \leftarrow 0$
for Q ϵ Q do
$\sigma_u \leftarrow \sigma_u$ V σ_Q
$A_u \leftarrow A_{Fi}$ U A_r (F_i)
$Q_u \leftarrow$GenerateQuery(A_u, σ_u.)

σ_Q : set of selected database query
A_{Fi} : set of projection attributes
$A_r(F_i)$: set of relevant attribute

In this system the user can give the necessary "where" conditions and can make changes iteratively required

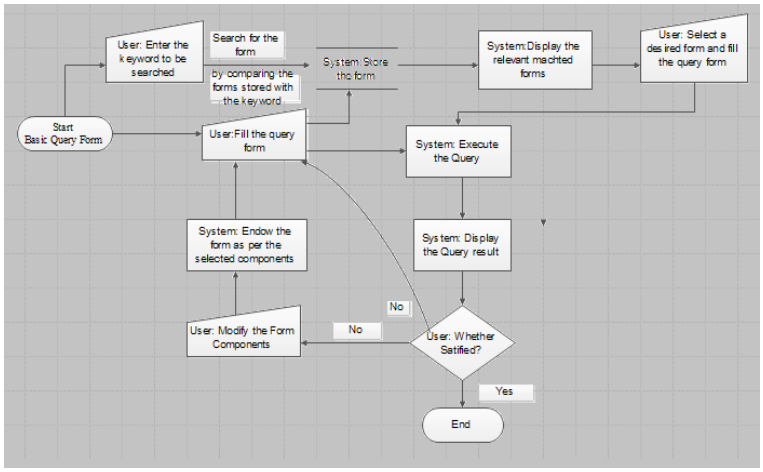


Fig.1: the work-flow of the architecture

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.

Tables 1, 2, 3 shows the details of the where clauses. Table 1 contains the columns like Transaction ID, Related Tables, and Transaction Weight. Related Tables contains the tables selected by the user during each transaction. Transaction ID contains the ID of each transaction. Transaction Weight contains the number of related tables included in each transaction. Table 2 shows the Rank Matrix calculation. Rank Matrix contains

the "where" clauses given by the user and the Transaction Ids. If any condition is used in a transaction its value will be 1.0 otherwise the value will be 0.0, and likewise the summation of each "where" clause is calculated and it will be the rank value. Table-3 contains the sorted order of the where clauses based on the increasing order of their rank value.

The multi criterion decision making can be addressed with the help of skyline queries. To implement this user have to mention the two skyline points (2 column names) in the skyline area given in the form along with the limit. Store the values of 2 points in the array of arrays.

Table-1

SINO	TransactionID	Related Tables	Transaction Weight
1	8	cars	1
2	9	cars	1
3	10	cars	1
4	11	cars	1
5	12	cars	1
6	13	cars	1
7	14	cars	1
8	15	cars	1
9	16	cars	1
10	17	cars	1
11	18	cars	1
12	19	cars	1
13	20	cars	1
14	21	cars	1
15	22	cars	1
16	23	cars	1
17	24	cars	1
18	25	cars	1
19	26	cars	1
20	27	cars	1
21	28	cars	1
22	29	cars	1
23	30	cars	1

Table-2

RANK MATRIX																									
Where Clause	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Rank
cars.Cyl<7	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
cars.Air_Pollution_Score<7	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
cars.Fuel<6	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
cars.Model<12	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0
cars.Model>12	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
cars.Model<34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
cars.Displ<45	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
cars.Cyl=6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
cars.Cyl>4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0
cars.Model=ACURA RL	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
cars.Air_Pollution_Score>6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
cars.Veh_Class=suv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
cars.Veh_Class=SUV	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	2.0
cars.Veh_Class="SUV"	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
cars.Model="ACURA MDX"	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0

Table-3

Rank	Where Caluse	Weight
1	cars.Cyl>4	5.0
2	cars.Model<12	3.0
3	cars.Air_Pollution_Score<7	2.0
4	cars.Model>12	2.0
5	cars.Air_Pollution_Score>6	2.0
6	cars.Veh_Class=SUV	2.0
7	cars.Cyl<7	1.0
8	cars.Fuel<6	1.0
9	cars.Model<34	1.0
10	cars.Displ<45	1.0
11	cars.Cyl=6	1.0
12	cars.Model=ACURA RL	1.0
13	cars.Veh_Class=suv	1.0
14	cars.Veh_Class="SUV"	1.0
15	cars.Model="ACURA MDX"	1.0

The values are sorted i.e., the rows of the 2 skyline points are sorted. Then the rows are numbered and likewise a table is created.

Then another table is created where the rows are separated based on the skyline points. Each row contains both the skyline points. Graph Plotting: Create a graph data structure. The numbers are represented in the graph. Take a point and draw lines to x-axis and y-axis from that point and a rectangle is formed below that point and there must be no point inside that rectangle. Likewise a set of layers are formed. Suppose if there are 4 points in the graph of layer-1 then draw another graph excluding the first 4 points and that becomes layer-2. Then plot all the points as different layers. After the plotting select the points starting from the first layer upto the limit given by the user.

The forms filled by the user can be stored for searching purpose. For the keyword based search when the user selects table the basic form related to that table and and a form searching area will be displayed. If the user want to create a new form then the user can opt the basic query form. Or the user can use the keyword search area. Here the user can enter the keyword to be searched. The links of the matching forms based on the keyword will be displayed and the user can click on the link and can select the desired form.

V. Conclusion And Future Work

In this paper we are proposing an enhanced dynamic query form system which helps the users to dynamically generate the query forms and to get the desired result iteratively at runtime. Thereby an existing customized query form is enriched by the iteration concept. Dynamic Query Form (DQF) system allows query processing and optimization of a relational complex database and also allows the keyword based search of the dynamically generated query forms. The concept of multicriterian decision making is also cooperated here. So skyline query concept can be used for addressing multicriterian decision making. As a future work natural language processing can be incorporated in this dynamic query form system and also this approach can be extended to non relational data.

Acknowledgment

The authors wish to thank the Management and Principal and Head of the Department(CSE) of Ilahia College of Engineering and Technology for their support and help in completing this work.

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 April 2013

[2]. 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.

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

[4]. N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu. "Snipsuggest: Context-aware autocompletion for sql". PVLDB, 4(1):22–33, 2010..

[5]. 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, pages349–360, Providence, Rhode Island, USA, June 2009.

[6]. 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.
[7]. S. B. Roy, H. Wang, U. Nambiar, G. Das, and M. K. Mohania.”Dynacet: Building dynamic faceted search systems over Databases”. In Proceedings of ICDE, pages 1463–1466, Shanghai, China, March 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]. G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. “Query recommendations for interactive database Exploration”. In Proceedings of SSDBM, pages 318, New Orleans, LA, USA, June 2009
[10]. S. Zhu, T. Li, Z. Chen, D. Wang, and Y. Gong.“Dynamic active probing of helpdesk databases”. Proc. VLDB Endow., 1(1):748–760, Aug. 2008
[11]. 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.
[12]. Hua Lu, Member, IEEE, Christian S. Jensen, Fellow, IEEE Zhenjie Zhang Student Member, IEEE ”Flexible and Efficient Resolution of Skyline Query Size Constraints”.