International Conference on Modern Trends in Engineering Science and Technology
© 2019 IJSRSET | Volume 5 | Issue 5 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section : Engineering and Technology

Implementation of Dynamic Query Form Generation for Database Queries

Alisha Neware, Aarti Shukla, Prajakta Kawale, Roshani Wanjari
BE Department of Computer Science and Engineering, Shrimati Rajshree Mulak College of Engineering, Nagpur, Maharashtra, India

ABSTRACT

With the fast advancement of databases what's more web information databases are getting to be exceptionally tremendous in size and critical in nature. These databases look after substantial furthermore heterogeneous information, with extensive number of relations and attributes. So it is extremely hard to plan a set of static query forms to answer different specially appointed database queries on modern databases. Accordingly there is need of such framework which creates Query Forms dynamically as indicated by the client's need at run time. The proposed framework Dynamic Query Form i.e.DQF framework going to give an answer by the query interface in huge and complex databases by using cache memory concept. In proposed framework, the center idea is to catch client interest all through client communications and to adjust the query type iteratively. Each emphasis comprises of two sorts of client interaction: Enriching Query Form and Query Execution. In first type, DQF would prescribe a list of ranked query form part to client so he/she can choose desired form parts into current query form. In Query Execution client fills current query form and submit query, DQF going to show result and take feedback from client on gave query results. A client would have another thing to do, to fill the query form and submit questions to view the query result at every emphasis. So that a query form could be dynamically refined till the client fulfills with the query results. The results are then stores in cache memory. Next time when client wants results which are already fetched previously, then it can be fetched directly from cache memory. So, that retrieval result can be fetched as faster. This can be improved efficiency of system.

Keywords : Dynamic Query Form DQF, Query Refinement, Ranking, Interaction, Query Form Generation.

I. INTRODUCTION

A database is just as functional as query interface permits it to be. In the meantime, a client is not ready to communicate with the database, even the rich information store gives petite or with no worth. Composing well structured queries, for example, SQL and XQuery can be trying because of various reasons, including the client's absence of nature with the query language and the client's lack of awareness of the basic schema. A form based query interface, which just obliges filling spaces to recognize query parameters, is valuable since it helps make information clients with no learning of authority query language or the database schema. By and by, form-based interfaces are utilized normally, however normally each one form is outlined in an adhoc way and its relevance is confined to a little set of fixed questions. Query form is one of the greater part utilized client interfaces for querying databases. Conventional query forms are outlined and predefined by engineers or DBA in different information administration frameworks. DQF framework, query interface that is equipped for dynamically producing query forms for clients. Traditional document retrieval structure, clients in information retrieval domain, typically ready to perform a few rounds of activities before unique the last applicants. The DQF is to catch client interest all

through client connections and to adjust the query sort.

Query language were produced to point out to a database engine how a query ought to be assessed and not to offer assistance clients knows the semantics of the query and choose on the off chance that it matches the query in their mind. Besides, given a query in a definitive language, it is not generally evident how to make a relating form that is understandable to the client and catches the information needed. Despite the fact that the requirement predicates and return attributes are steady and directed by the query, it doesn't straightforwardly detail the structural connections between the attributes included, nor does it recommend how they can be displayed to a client in a significant way. It is simple for an individual with the query also the hidden information structure to plan a form for it, also map client information fields to the suitable query predicates. While such a form would do the employment pleasantly for the query close by, it is normally not extensible, and brings to bear outer human information. As questions and schema get to be more critical, manual form era is no simpler, and concealed suppositions become possibly the most important factor substantially more every now and again. Then again, making forms consequently is a long way from trivial on the grounds that it is hard to fulfill all the while some properties wanted of any form-based interface. In this paper, we show an automated system to produce a form based query interface that augments efficient qualities [1].

A form is a basic and instinctive query interface very often times utilized to give simple database access. It obliges no learning, on the piece of the client, of how the information is sorted out away and no skill in query language. Consequently, forms are a widespread decision for the vast majority of today's databases. Moreover, while simple to utilize, forms give the client a constrained perspective of the fundamental information. In the event that a client requires some information that is show in the database not accessible through the accessible forms, he or she is vulnerable without a querying option. While at times certain query sorts are purposefully prohibited for security, performance or different reasons, it is regularly the case that a query isn't backed basically on the grounds that the interest for it wasn't expected by the interface engineer. On the other hand, it is not practical to help all conceivable questions, especially in the schema that the mapping of the database is unpredictable: the interface would require extremely numerous forms and each one form would be excessively critical, overpowering clients and nullifying the profits of a forms-based interface. Consequently trade-off necessities to be made between expressivity what's more many-sided quality while planning forms. This trade-off is basic to interface ease of use and is non-trivial because of the conceivably wide scope of querying needs of planned clients [2].

This paper is composed further as: Section II talks about related work studied till now. Section III presents implementation details, algorithm used and mathematical model. It also includes experimental setup. Results and discussions talked in section IV. Section V ends with the conclusions and presents future work.

II. RELATED WORK

In [3] had considered the subject of how best to expand results about the region of uncertain inquiries. This is an issue that most web crawlers go up against as customers routinely underspecified their genuine data needs. They took over both the criticalness of the records and the distinctions of rundown things and showed an objective that explicitly redesigns for the two. They had given a ravenous calculation to the objective with extraordinary gauge guarantees. To evaluate the sufficiency of this strategy, they had proposed theories of adequately analyzed estimations to make into note of the points of the customers. The

objective in enhance can be viewed as a moderate metric that way to extend the probability that the typical customer will find some significant data among the rundown things.

In the paper [4] have shown a question proposition framework supporting the insightful examination of social databases and an instantiate of this structure in light of customer based community filtration. They considered the tension this is a first-cut response for the incredibly interesting issue of redid question recommendations. There are various open issues that should be tended to. For example, a fascinating issue is that of recognizing "practically identical" questions with respect to their structure and not the tuples they recoup. Two questions might be semantically similar anyway recuperate particular outcomes as a result of some isolating conditions. Such questions should be considered in the recommendation procedure.

In paper [5] exhibited that probabilistic procedures can be used to plot brilliant data section shapes that advance high data quality. USHER impacts data driven encounters to motorize distinctive strides in the data section pipeline. Prior to section, an asking for of structure handle that propels quick data get, controlled by an insatiable data get standard. After passage, we use a similar standard to progressively modify the structure in perspective on entered characteristics. After the passage, normally recognize conceivably mistaken data sources, guided by contextualized blunder likelihood, and re-make those inquiries to affirm their rightness. The observational evaluations display the data quality preferences of every one of these parts: inquiry asking for grants better conjecture precision and the re-asking model separates wrong responses effectively.

In [6] inspected the system of using look catchphrase to lead customers to shapes for uncommonly named questioning of databases. They considered different issues that develop in the use for this technique: sketching out and creating shapes in an effective way, dealing with watchword addresses that are a blend of data terms and layout terms, sifting through structures that would convey no results in regards to a customer's question, and situating and appearing in a way that assistance customers find accommodating structures even more quickly. Their experience prescribes a couple of ends. One is that an inquiry change by mapping data characteristics to design characteristics amid hunt catchphrase, combined with isolating structures that would incite empty outcomes, is an alluring philosophy. A substitute end is that simply demonstrating the returned structures as a dimension rundown may not be appealing, by one way or another of collection and showing relative structures to customers is essential.

In [7] have proposed a specific likeness measure between various pieces of a 0/1 connection moreover portrayed measures between qualities, between lines, between sub-relations of the database. The measures depend on the setting of the individual data. This subsequently, reveals the simple relations between parts, something which is missing from the more clear measures. Therefore, displayed an iterative calculation which, when given the subjective starting characteristics, meets quickly to separate which is steady in useful way.Author in [8] taken together, the two examinations demonstrate how using comprehended input and AI can convey significantly specific web files. While inclinations make inferred analysis data difficult to decipher, techniques are open for keeping up a key separation from these tendencies, and the resulting pairwise tendency clarifications can be used for suitable learning. In any case, much remains to be done, running from keeping an eye on security issues and the effect of new types of spam, to the framework of natural examinations and dynamic learning techniques.

In paper [9] have states that the new circumstances are creating where colossal amounts of customers need to make and run complex inquiries over a broad, bestowed data store. Cases consolidate broad

insightful databases and Web-related data. Their goal is to address these troubles, make a CQMS, and test it in an analytical database condition. SnipSuggest, a setting careful, SQLautocomplete System is presented by [10]. SnipSuggest is awakened by the creating masses of non-ace database customers, who need to perform complex examination on their extensive scale datasets, yet experience issues with SQL. SnipSuggest hopes to effortlessness question association by prescribing huge SQL scraps, considering what the customer has written along these lines. They have shown that SnipSuggest has the limit make pleasing proposals, at wise rates for two separate datasets. Creator in [11] have proposed Facetedpedia, a faceted question look structure over Wikipedia. This structure gives a dynamic and robotized faceted mission interface for customers to examine the articles that are the outcome of a watchword result question. Given the sheer size and basic thing of Wikipedia and the huge space of possible faceted interfaces, they proposed estimations for situating faceted interfaces and capable calculations for discovering them.In paper [12] creator proposed a model and calculation for widening web crawler comes to fruition, and has presented an evaluation and examination of calculation and its results. To the best of the understanding, this is the main work that relates results quality and decent variety to anticipated outcome and risk in snaps and gives a model to improve these sums. A test in any inquiry progression including their very own is deciding experiences about factors used as a piece of the model; they have shown two or three methodologies to decide these estimations in light of data and estimations that are all things considered available in web crawlers. There is space to find better estimations about explore rates and connections which can incite increasingly exact gauges and better inquiry results.

III. IMPLEMENTATION DETAILS

A. System Overview

To outline/built a form for query, we should first examine it and recognize its limitations and the results which are required. At that point we utilize information accumulated from this investigation, and from the schema of the database, to make the fundamental set of form-components. At long last, we arranged these components in gatherings, mark them suitably, and lay them out in a significant manner on the form. The proposed a dynamic query form framework which creates the query forms as indicated by the client's desire at run time. The framework gives an answer for the query interface in extensive and critical databases.

The following Figure 1 shows the proposed system architecture. This paper proposes DQF, a novel database query form interface, which has the capacity dynamically produce query forms. The generation of DQF is to catch preferences and rank query form component, helping them to make choices. The time of generation of a query form is an iterative process and is guided by the client. At every cycle, the framework naturally creates positioning arrangements of ranking parts and the client then includes the wanted form parts into the query form. The positioning of form component is in light of the caught preferences. A client can likewise fill the query form and submit query to view the query result at every cycle. Along these lines, a query form could be dynamically refined till the client fulfills with the query results. Our proposed model cache memory concept is used. When user fills query form from previous fetched query result users have option to choose either form it or to fill it. So system is time consuming. Users can directly fetched results from the cache memory.

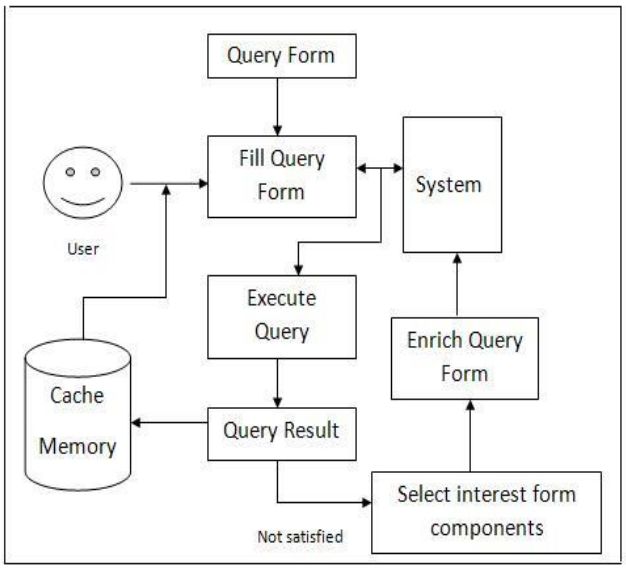

Figure 1 : System Architecture

The above proposed framework has some advantages. The proposed generation of DQF approach which helps clients dynamically produce query forms. The dynamic approach regularly prompts higher achievement rate and less difficult query forms analyzed with a static methodology. The components of ranking of form additionally make it less demanding for clients to modify query forms.

B. Algorithm

We presented the form generation procedure to outline forms for a whole set of questions. Given a set of interested queries, the naive methodology would be to fabricate one form for every one of them. On the other hand, queries against a single pattern will as a general rule have same between them, which we can adventure to minimize excess. To control a form's readability, we present a complexity of form nature threshold, a measure of complexity nature that we would like no form to surpass. On the other hand, this limit might be unenforceable if set excessively low, or if a percentage of the questions included are excessively threshold. In such cases, even a one query may have many-sided quality that surpasses the threshold. In the general case, threshold is fulfilled by part a query clustered secured by a complex form into littler cluster secured by less difficult forms. To quantify how valuable a form is, we characterize its expressivity as what number of other various queries it can express.

Algorithm Generate Form
Input: A query Q (as an Evaluation Plan)
Output: A form F
// Element Construction and Grouping
Create a new form-group g and add it to the form-tree T;
for each operation o ϵQ when traversed top-down **do**
caseo is a "selection"
Create a constraint-element using the selection predicate;
Put this constraint-element in g;
caseo is a "projection"
Create a result-element using each projected attribute;
Put these result-elements in g;
caseo is an "aggregate function"
Create an aggregate-element using the group-by attribute, the grouping-basis and the aggregate function;
Put this aggregate-element in g;
caseo is a "join"
Create a join-element using the two (left and right) attributes of the join condition;
Put this join-element in g;
Create a new group g′as a child of g in T;
Set g ← g′;
end
// Element and Group Labeling
for each form-group g ϵT **do**
Label g relative to its parent group (use absolute path if
gis the root);
for each form-element e ϵg **do**
Label e relative to g;
end
end

C. Experimental Setup

The system is built using Java framework (version jdk 6)on Windows platform. The Netbeans (version 6.9) is used asa development tool. The system doesn't require any specifichardware to run, any standard machine is capable of runningthe application.

IV. RESULTS AND DISCUSSION

Dataset

Here we are planning to use Geobase databases. In this we will take 9 relations, 32 attributes and 1,329 instances.

Results

In our implementation our expected result will be like our proposed system can display query result in minimal time span than existing system. By use of cache memory system effectives may be increased.

V. CONCLUSION

Query interfaces are a basic part in deciding the helpfulness of a database. A form-based interface is broadly viewed as the most easy to use querying strategy. In this paper, we have created components to succeed the difficulties that point of confinement the helpfulness of forms, in particular their prohibitive nature. In this paper we propose an intelligent query form era approach which makes difference clients to dynamically create query forms. By use of cache memory we can improve effectiveness of a system. As future work, we will contemplate how our methodology can be stretched out to non-relational databases. With respect to the future work, we plan to create various systems to catch the client's enthusiasm for the queries other than the feedback.

VI. REFERENCES

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

[2]. 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.

[3]. R. Agawam, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In Proceedings of WSDM, pages 5-14, Barcelona, Spain, February 2009.

[4]. G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recommendationsfor interactive database exploration. In Proceedingsof SSDBM, pages 3-18, New Orleans, LA, USA, June 2009.

[5]. 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.

[6]. 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.

[7]. G. Das and H. Mannila. Context-based similarity measures for categorical databases. In Proceedings of PKDD 2000, pages201-210, Lyon, France, September 2000.

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

[9]. N. Khoussainova, M. Balazinska, W. Gatterbauer, Y. Kwon, and D. Suciu. A case for a collaborative query management system. In Proceedings of CIDR, Asilomar, CA, USA, January2009.

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

[11]. C. Li, N. Yan, S. B. Roy, L. Lisham, and G. Das. Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. In oceedings of WWW, pages 651-660, Raleigh, North Carolina, USA, April 2010.

[12]. D. Rafiei, K. Bharat, and A. Shukla. Diversifying web search results. In Proceedings of WWW, pages 781-790, Raleigh, North Carolina, USA, April 2010.

Cite this article as :

Alisha Neware, Aarti Shukla, Prajakta Kawale, Roshani Wanjari, "Implementation of Dynamic Query Form Generation for Database Queries", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), ISSN : 2456-3307, Volume 5 Issue 5, pp. 73-79, February 2019.
Journal URL : http://ijsrset.com/IJSRSET195513