International Journal of Engineering Research and General Science Volume 2, Issue 6, October-November, 2014
ISSN 2091-2730

Espionage on Search Optimization using Dynamic Query Form

Prajkta Dagade[1], Mansi Bhonsle[2]

Computer science and Engineering G.H.R.C.E.M, Ahmednagar, India[1]

Computer science and Engineering G.H.R.C.E.M, Wagholi, Pune, India[2]

prajkta.dagade@gmail.com[1]

mansi.bhonsle@gmail.com[2]

Abstract— Traditional data mining technologies cannot work with huge, heterogeneous, unstructured data. Modern web database as well as scientific database maintains tremendous and heterogeneous data. These real word databases may contain hundreds or even thousands of relations and attributes. Query form is one of the most widely used interfaces for querying database. Traditional query forms are designed and pre-defined by developer or DBA in various information management systems. But extracting the useful information with this traditional query form from large dataset and streams of the data is not possible and it is difficult to design set of static query forms to satisfy numerous ad- hoc database queries on those complex databases. In this paper, we propose a Search optimization using dynamic query form system SODQF. SODQF is a query interface which is able to dynamically generating query forms for user. Unlike to tradition document retrieval, users in database retrieval are often willing to perform many rounds of action before identifying final results. Dynamic query form captures user interest during the user interaction and to adapt the query form interactively. Each iteration consists of three types of user interactions, Selection from an assortment of the forms, Query form Renovation and Query Execution. The Query from is enriched repeatedly until the user is satisfied with the query results. In this paper we are mainly focusing dynamic generation if query forms and ranking of query form components.

Keywords— Query Form, Query Renovation, Information Extraction, Dynamic Approach, User interaction, Keyword Search

.

INTRODUCTION

Traditional database systems require the user to construct the database query from language primitives [1]. Such systems are powerful and expensive but not easy to use, especially when user is not familiar with the database scheme. The continuous advancement of wide-area network technology has resulted in rapid growth in number of data sources available online and the demand for information access over the internet from a diversity of clients. Query form is one of the most important user interfaces used by users for querying databases. With the fast development of web information and scientific databases, modern databases became tremendous and complex. Many databases such as Freebase, DBPedia have thousands of structured web entities[2] [4]. Hence it is difficult to design set of static query forms to satisfy different ad-hoc database queries on those large databases.

Many database management and development tools, such as Easy Query [5], Cold Fusion [3], SAP and Microsoft Access, provides several mechanisms to let users create customized queries on databases. This customized query totally depends on user's manual editing [6]. It will be critical for user if user is not familiar with the database schema, in short those thousands of data attributes confuse user.

EXISTING APPROACHES

In the analysis of the query form which is one of the most useful user interfaces for users for querying database, some different approaches have been proposed for prompt response to user. Lots of research works focus on database interfaces which assist users to query the relational database without structured query language. Query by example and Query form are two most widely used database querying interfaces. Query Form has been utilized in most real-time business or scientific information systems. Main goal in this paper is to focus how to generate Query forms and to discover techniques that assist users who are unfamiliar to and do not want to use Structured Query language in posing ad-hoc structured queries over relational databases. However, the creation of customized queries totally depends on user's manual editing [6].

Dynamic Faceted Search is a type of search engines where relevant facets are presented for the users according to their navigation paths [7] [8]. Dynamic faceted search engines are similar to dynamic query forms if we only consider *selection* components in a query. However besides *Selections*, a database query firm has other important components such as *Projection* components. Projection components control the output of the query form and needs to be considered. Moreover, designs of *Selection* and *Projection* have inherent influences to each other.

Autocompletion for Database Queries, in [9] ,[10], novel user interfaces have been developed to assist the user to type database queries based on query workload, the data distribution and the database schema. Queries in their work are in the forms of SQL and keywords.

An Efficient Sql-Based Rdf Querying Scheme is a devising scheme for efficient and scalable querying of Resource Description Framework (RDF) data has been an active area of current research. However, most approaches define new languages for querying RDF data, which has the following shortcomings:

1) They incur inefficiency as data has to be transformed from SQL to the corresponding language data format, and

2) They are difficult to integrate with SQL queries used in database applications.

This paper proposes the SQL based scheme that avoids these problems. Specifically, it introduces a SQL table function RDF_MATCH to query RDF data. The results of RDF_MATCH table function can be further processed by SQL's rich querying capabilities and seamlessly combined with queries on traditional relational data. Furthermore, the RDF_MATCH table function invocation is rewritten as a SQL query, thereby avoiding run-time table function procedural overheads [11].

An SQL-ish query language for RDF provides consistent, human-understandable, access to repositories of semantic data, whether stored files or large databases, enabling application programmers to create semantic web applications quickly. This paper describes the conceptual framework for the query language: this is closely tied to RDF graph, providing a base level of query of RDF data. There are number of other query languages of RDF available, one of the earliest was rdfDB [12] and this is the basic for SquishQL syntax. It is simple graph query languages designed for use in the rdfDB database system. It differs from SquishQL and does not contain the constraints on the variables used by SquishQL. It returns result as a table. This paper describes a refined framework for the querying of RDF data and have implemented SquishQL in tree systems: 1) Inkling, stores RDF data in relational database or in external XML files, 2) The second implementation, RDQL, is part of the Jena RDF toolkit and combines query with manipulation of the RDF graph at a fine-grained level through the jena RDF API and, 3) The third, in RDFStore, is close coupling of RDF and Perl data access styles [13].

SeeDb is a DBMS that practically automates the especially laborious aspects of the search for useful data insights or a query. In a nutshell, given an input query Q, the new DBMS optimizer will explore not only the space of physical plans for Q, bus also the space of possible visualization of potentially "interesting" or "useful" visualizations, where each visualization is coupled with a suitable query execution plan. SeeDb provides analysts with visualization highlighting interesting aspects of the query results. There are several concrete problems in architecting SeeDb, relating to areas ranging from multi-query optimization and approximation to multi-criteria optimization. As per this work the general area of bringing visualization closer to the DBMS is challenging, yet, important direction for database research in the future [14].

Automated Technique for forms-based interface seeks to maximize the ability of a forms-based interface to support queries a user may ask, while bounding both the number of forms and the complexity of any one form. This automated technique generates a reasonable set of forms, one that can express 60-90% of user queries without any input from the database administrator. This technique does not consider any actual query log at hand, solely based on the schema and data content of a database. This technique break the forms interface design problem down into three challenges:

1) First, Determining the schema fragments(s) most likely to be of interest to a querying user.

2) Second, Partition the filtered collection of schema elements into groups (not necessarily distinct).

3) Convert each of these groups of schema elements into a form that a user can employ to express a desired query [6].

Algae are another early query language for RDF. It uses S-expression syntax to do graph matching. It is used to power the W3C's Annotea annotations system [15], and other software at the W3C. It is written in pearl, and can be used with an SQL database. It returns a set of triples in support of each result.

RQL [16, 17] is a combined RDF store and query system. It also provides a schema validating parser and has a syntax targeted at RDFS [18] applications. It can perform similar queries to SquishQL, with the added power of support for transitive closure on RDFS subclass and sub property. This is also the query language used by Seasame [19].

Skyline Query can be used for addressing multi criteria decision making. This will enhance the performance of Dynamic Query Forms, which is a query interface capable for generating query forms dynamically for users. As the success of Internet search engines makes abundantly clear, when faced with discovering documents of interest, 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. So the concept of multi criteria decision making can be in cooperated and for addressing multi criteria decision making the concept of skyline query has been used [20].

QueryScope, is a prototype query visualization system. It is a novel query workload visualization and exploration system. It uses compact visual semantics to capture key elements of queries. Enhanced with common query pattern mining and similarity search, it enables database consultants to look up queries captured in related data warehouse projects and examine opportunities for performance tuning. Main Goals of this system are:

1) To communicate the essence of a query (or a collection of queries) pictorially through a controlled visual semantics, 2) To provide a variety of visualization options so that a user can focus on the aspects of queries of relevance,

3) To visualize queries in the context of a physical schema, 4) To facilitate searches for similar queries,

5) To make the tuning process productive and repeatable.

QueryScope was successfully used in some of engagements, a thorough assessment of the value of QueryScope would require putting it in the hands of many practitioners for use in real customer projects to judge tuning knowledge accumulation and repeatability of the tuning advice. Ling hue [21] intends to pursue this revenue in the future and release the tool for public trial [21].

One more system for Query, Analysis, and Visualization of Multidimensional Relational Databases is Polaris. Polaris is an interface for exploration of multidimensional databases that extends the Pivot table interface to directly generate a rich, expressive set of graphical displays. Polaris builds table using an algebraic formalism involving the fields of the database. The use of tables to organize multiple graphs on a display is a technique often used by statisticians in their analysis of data [22], [23], and [24]. The Polaris interface is simple and expressive because it is built upon formalism for constructing graphs and building data transformations. Polaris can be extended as future work; one area of future work is exploring database performance issues. A related area is expanding Polaris to expose the hierarchical structure of data cubes. Another area of future work is to leverage the direct correspondence of graphical marks in Polaris to tuples in the relational databases in order to generate database tables from a selected set of graphical marks [25].

DIOM, dynamic query processing framework and the strategies used for improving query responsiveness. There are some features that distinguish the dynamic query processing in DIOM from other approaches, such as Carnot, Garlic[26], TSIMMIS [27] are:

1) Allow users to pose queries on the fly, without relying on a predefined view that integrates all available information sources,

2) It identify the importance of query routing step in building efficient query scheduling framework for distributed open environment,

3)It is three tier approach i.e. Query routing, Heuristic-based optimization, and cost-based planning to eliminate the worst schedules as early as possible.

In future it could be possible to adapt and incorporate the state of art research results in improving query responsiveness with the DIOM system, such as the query scrambling approach for dynamic query plan modification [28] and the online aggregation for fast delivery of aggregate queries by continuous sampling.

Usher is the probabilistic approach that can be used to design intelligent data entry forms that promote high data quality. Usher basically learns the probabilistic model over the questions of the form. Usher then applies this model at every step of the data entry process to improve data quality. This system focuses on:

1) Data-driven approach 2) Learning a model for data Entry, i.e. it uses probabilistic model of the data, represented as a Bayesian network over form questions, 3) Question Ordering, 4) Question re-asking, 5) Evaluation of the benefits of Usher and on the quality of this model.

USHER leverages data-driven insights to automate multiple steps in the data entry pipeline. Before entry, there is an ordering of form fields that promotes rapid information capture, driven by a greedy information principle. This demonstrates the data quality benefits, question ordering allows better prediction accuracy and the re-asking model identifies erroneous responses effectively. This model makes assumptions both about how errors are distributed, and what errors look like. Based on this, future work would be to learn a model of data entry errors and adapt our system to catch them [29].

OUR APPROACH

In this paper we propose search optimization using Dynamic Query form which is capable of dynamically generating query forms for users. Dynamic Queries let users "fly through" database and it captures user interests during user interactions and adapt the query form repeatedly. Each iteration consists of three types of interaction:

Selection from an assortment of form components	**Recommends a ranked list of query form components to user**
Query Form Renovation	**User selects the desired form components from current query form** **User fills out the current query form and submit a query**
Query Execution	**OQF executes the query and displays the desired results** **The user gives the feedback about the query results**

Selection from an assortment of forms, Query Form Renovation, Query Execution. We mainly focus on developing the methods to capture user's interest besides the click through feedback. For instance we can add a text box for keyword queries as an input from users, ranking of query form components and dynamic generation of query forms.

ACKNOWLEDGMENT

This Project is by far the most significant accomplishment in my PG and it would be impossible without people (especially my family) who supported me and believed in me.

I am thankful to **Prof. Mansi Bhonsle**, ME Coordinator of department of Computer Engineering, Raisoni College, Pune for giving me the opportunity to work under her and lending every support at every stage of this work. I truly appreciate and value her esteemed guidance and encouragement from the beginning to the end of this work. I am indebted to her for having helped me shape the problem and providing insights towards the solution. Her trust and great support inspired me in the most important moments of making right decisions and I am very glad to work with her.

CONCLUSION

We discussed different Form Generation Approaches; from them we can compare three basic approaches to generate query forms:

1) DQF: The dynamic query form system mainly focused in this paper,

2) SQF: The static query form generation approach proposed in [30]. It also uses query workload. Queries in the workload are first divided into clusters. Each cluster is converted into query form,

3) CQF: the customized query form generation used by many existing database clients, such as Microsoft Access, EasyQuery, and ActiveQueryBuilder. In this paper we focused on an innovative form based approach called Dynamic Query Form for search optimization. This form generation approach can indeed produce forms, of manageable number and complexity, which are capable of posing a majority of user queries to a given database. We consider number of issues that arise in the implementation for this approach such as: designing and generating forms in a systematic fashion, handling keyword queries, filtering out forms that that would produce no results with respect to a user's query, and ranking and displaying the forms in a way that help users find useful forms more promptly.

Dynamic approach often leads to the higher success rate and simpler query compared with static approach and offer a dramatic change from existing methods for querying databases. As a future work we can extend this approach to non relational data and we can also incorporate natural language processing in this dynamic query form system. We also plan to develop multiple methods to capture user's interest for the queries besides the click feedback. For instance, we can add a text-box for user to input some keyword queries; this would be helpful for user when data which user wants is not there on the click through forms. In particular, developing automated techniques for generating better form descriptions, especially in the presence of grouping of forms, appears to be a challenging and important problem.

REFERENCES:

[1] Jeffrey D. Ullman. Principles of Database Systems. Computer Science Press, 1980.

[2] DBPedia. http://DBPedia.org.

[3] ColdFusion. http://www.adobe.com/products/coldfusion.

[4] Freebase. http://www.freebase.com.

[5] EasyQuery. http://

[6] M. Jayapandian, H. V. Jagadish. Automated Creation of a Form-based Database Query Interface. VLDB 2008.

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

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

[9] N. Khoussainova, Y. kwon, M, Balazinska, and D. Suciu. Snipsuggest: Context-aware autocompletion for sql. PVLDB, 4(1):22-23, 2010.

[10] A. Nandi and H. V. Jagadish. Assisted querying using instant response interfaces. In proceedings of ACM SIGMOD, pages 1156-1158, 2007.

[11] Eugene Inseok Chong. An Efficient SQL-based RDF Querying scheme, 2005

[12] R. V. Guha,"rdfDB : An RDF Database", web page: http://guha.com/rdfdb/

[13] Andy Seaborne, Alberto Reggiori, Libby Miller. Tree Implementation of SquishQL, a Simple RDF Query Language, April 26, 2002.

[14] Aditya Parameswaran. SeeDB: visualizing Database Queries Efficiently.

[15] J. Kahan, M. Koivunen, E. Prud'Hommeaux, R. R. Swick "Annotea: An Open RDF Infrastructure for Shared Web Annotations", http://www10.org/cdrom/papers/488/

[16] Greg Karvounarakis, "The Rdf Query Language (RQL)"

[17] G. Karvounarakis, V. Christophides, D. Plexussakis, S Alexaki, "Querying Community Web Portals", SIGMOD2000, http://www.ics.forth.gr/proj/isst/RDF/RQL/rql.html

[18] Dan Brickley, R. V. Guha (aditors), "Resouce Description Framework (RDF) Schema Specification 1.0", 27 march 2000 (W3C Candidate recommendation).

[19] Sesame, http://sesame.administrator.nl/, part of the OntoKnowledge project, http://www.ontoknowledge.org/

[20] Dr. Anil Rajput. Multicriteria data Retrieval in database using Advanced Database Operator. March 2014

[21] Ling Hu, Yuan-Chi Chang. QueryScope: Visualizing Queries for Repeatable databse Tunning

[22] J. Bertin, Graphics and Graphics Information Processing. Berlin: Walter de Gruyter, 1980.

[23] W.S. Cleveland, The Elements of Graphing Dta. Pacific Grove, Calif.: Wadsworth Advanced Books and Software, 1985

[24] E.R. Tufte, The Visual Display of Quantitative Information. Ghesire, Conn.:Graphics Press, 1983.

[25] Chris Stolte, Diane Tang. Polaris: A System For Query, Analysis, and Visualization of Multidimenal Relational Dtabases.

[26] L. Haas, D. Kossmannn, E. Wimmwers, and J. Yan. Optimizing queries across diverse data source. In The international Conference on Very large Data Bases, 1997.

[27] Y. papakonstantinous, S. Abiteboul, and H. Garcia-Molina. Object Fusion in Mediator Systems. In VLDb 96, Bombay, India, Sept. 1996.

[28] L. Amsaleg, M.J. franklin, A. Tomasic, and T. Urhan. Scrambling query plans to cope with unexpected delays. In proceeding of the International Conference on Parallel and Distributed Information System, Miami Beach, Florida, December 1996.

[29] Kaung Chen, Harr Chen, Neil Conway. USHAR: Improving Data Quality with Dynamic Forms, June 2011.

[30] Sathu G Rajan, K. Sathya Seelan. Dynamic Query recommendation for interactive database Exploration, 2014