www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 4 Issue 2 February 2015, Page No. 10450-10450

Query Recommendations for Interactive Database Exploration

Sruthi T. K

Dept. of Computer Science and Engineering, M.Tech Student
M.G University, Mount Zion College of Engineering,
Kadammanitta, Pathanamthitta, Kerala, India
sruthu777@gmail.com

Abstract: ***Query Recommendation for Interactive Database Exploration (QueRIE) is a recommendation system that supports interactive database exploration. The users who are not familiar with the database schema may face great difficulty in performing this job. This system aims at assisting non expert users of relational databases by generating personalized query recommendations. QueRIE tracks the querying behavior of previous users and identifies similar patterns. These similar query patterns are used to generate recommendations. There are three approaches used in this work for generating query recommendations viz., suggested query by dictionary mapping, tuple based and fragment history. And also the performance is analyzed among the three approaches.***

Keywords: Query recommendation, dictionary mapping, tuple based, fragmentation, Interactive database exploration, personalization.

1. Introduction

The database systems store large amount of data. When the number of users who use the database increases, it is critical to access large volume of data from the database systems. The users who lack familiarity with the database schema often have difficulties in retrieving relevant data. First-time users may not have the necessary knowledge to know where to start their exploration. Other times, users may simply overlook queries that retrieve important information.

This work describes a framework to assist non-expert users by providing personalized QueRIE recommendations. The recommended queries are relevant to the user's information needs and can be submitted directly or be further refined. In other words, the user can use them as "templates" for query formulation instead of having to compose new ones.

QueRIE is built on a simple premise that is inspired by Web recommender systems: If users A and B have similar queries, then the other queries of B may be of interest to user A and vice versa. In other words, we can recommend the queries of user B in order to help user A in their exploration of the database. But, the transfer of this approach to the database context introduces several technical challenges. First, SQL is a declarative language, and hence syntactically different queries may reflect the same information need. A second important challenge is that how do we know which queries are important in the computation of user similarity? Finally, the recommended queries need to be intuitive so that the user can understand and refine if necessary.

QueRIE addresses these challenges by decomposing each query into basic elements. These elements are used to compute similarities between users. Recommendations are generated by mining queries from the system log that match well with the users query fragments. Hence, the user is presented with queries that match her querying behavior. In this, QueRIE continuously monitors the user's querying behavior and finds matching patterns in the system's query log, in an attempt to identify previous users with similar information needs. Subsequently, QueRIE uses these "similar" users and their queries to recommend queries that the current user may find interesting.

The remaining of this paper is organized as follows: The remaining of this paper is organized as follows: Section 2 describes the literature review and section 4 gives the architecture. Section 5 provides a brief methodology, and the modules. Section 6 discusses the performance evaluation. Section 7 and conclude the paper with the plans for future work.

2. Review of Literature

2.1 Hive - A petabyte scale data warehouse using HADOOP *(A. Thusoo et al.)*

This paper says that HADOOP is a popular open-source map-reduce implementation which is being used in companies like Yahoo, Facebook etc. to store and process extremely large data sets on commodity hardware. However, the map-reduce programming model is very low level and requires developers to write custom programs which are hard to maintain and reuse. Hive, an open-source data warehousing solution built on top of HADOOP. Hive supports queries expressed in a SQL-like declarative language - HiveQL, which are compiled into map-reduce jobs that are executed using HADOOP.

2.2 QueRIE: A recommender system supporting interactive database exploration *(S. Mittal, J. S. V. Varman)*

This paper mentioned that the demonstration presents QueRIE, a recommender system that supports interactive database exploration. This system aims at assisting non-expert users of scientific databases by generating personalized query recommendations. Drawing inspiration from Web recommender systems, QueRIE tracks the querying behavior of each user and identifies potentially "interesting" parts of the database related to the corresponding data analysis task by locating those database parts that were accessed by similar users in the past. It then generates and recommends the queries that cover those parts to the user.

2.3 Amazon.com recommendations: Item-to-item collaborative filtering (G. Linden, B. Smith, and J. York)

This paper wrote that recommendation algorithms are used to personalize the online store for each customer. The store radically changes based on customer interests, showing programming titles to a software engineer and baby toys to a new mother. There are three common approaches to solving the recommendation problem: traditional collaborative filtering, cluster models, and search-based methods. Here, it compares these methods with our algorithm, which is called item-to-item collaborative filtering. The algorithm produces recommendations in real-time, scales to massive data sets, and generates high quality recommendations.

2.4 Recommending Multidimensional Queries (A. Giacometti, P. Marcel, and E. Negre)

In this work, the authors propose a framework for generating OLAP query recommendations for the users of a data warehouse. The techniques and the algorithms employed in the multidimensional scenario (for example, the similarity metrics and the ranking algorithms) are very different to the one that proposed. Recommendations can be computed on the fly efficiently and that our system can be tuned to obtain objectively good recommendations.

2.5 QueRIE: Collaborative Database Exploration (Magdalini Eirinaki, Suju Abraham, Neoklis Polyzotis, Naushin Shaikh)

This work describes an instantiation of the QueRIE framework, where the active user's session is represented by a set of query fragments. The recorded fragments are used to identify similar query fragments in the previously recorded sessions, which are in turn assembled in potentially interesting queries for the active user. We show through experimentation that the proposed method generates meaningful recommendations on real-life traces from the SkyServer database and propose a scalable design that enables the incremental update of similarities, making real-time computations on large amounts of data feasible.

3. QUERIE Architecture

The framework for generating query recommendations is given in Figure 1. The input query is accepted from the user. The user's query is forwarded to both the database system and the recommendation engine. The DBMS processes the input query and return the result to the database query interface. At the same time, the query is stored in the query log. The recommendation engine finds matching patterns in the system's query log by using previous user's querying behavior. And generate a set of query recommendation that is returned to the user through the query interface. Three approaches are used to generate the query recommendations: (1) suggested query by dictionary mapping (2) suggested query by tuple based query recommendations (3) suggested query by fragment history.

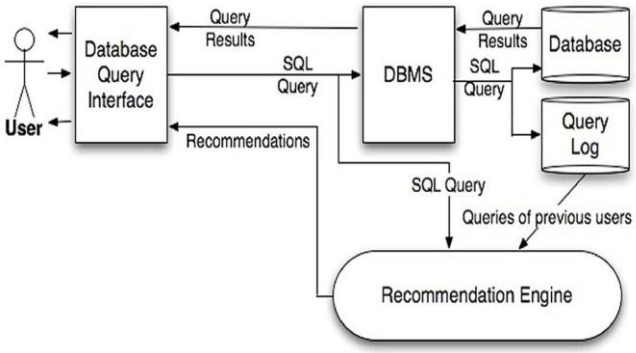

Figure 1: Architecture of QueRIE framework

4. Methodology

The active users query is decomposed into basic elements by the Querie framework paper Querie framework that captures the essence of the query's logic. These elements are used to compute similarities between users, as well as signature of the user's querying behavior. Recommendations are generated by mining queries from the system log that match well with the signature. The proposed system consists of the following three modules:

- Database Monitoring
- Query Analyzer
- Query Recommendation

When a user logs in the system the user have to submit an SQL query to the query interface. The SQL query is sent to the database as well as the recommendation engine. Also the query is fragmented and stored in the system log. The query is processed by the query analyzer and returns the result. At the same time recommendation engine will generate the recommendation that match well with the query.

4.1 Database Monitoring

Each time when a user logs in the system he/she have to submit an SQL query through the database query interface.
SQL query which is provided by the user is sent to syntax checker to check SQL syntax then the SQL query is fragmented and then transmitted to both the DBMS and recommendation engine. The input query is fragmented and stored in the system query log. This will make the comparison easier when finding the highly similar patterns from the system query log. Query fragmentation also makes storing the query in the database in an easier way.

4.2 Query Analyzer

The DBMS requested to the database and return the results to the user and then query is fragmented, creating an implicit query profile which consists of fragments. The given query is decomposed into fragments with respect to the keywords such as select, from, where, group by, having, order by. Names are given for the fragmented queries. The fragmented query

attributes are stored in the fragment table with respect to the fragment name.

4.3 Query Recommendation

Recommendation engine combine the active user query and query log and generate the set of queries, which are the recommendations. These recommendations are generated by the methods such as suggested query by dictionary mapping, suggested query by tuple based query recommendations, suggested query by fragment history.

The data dictionary will store the column name and the synonyms. This will map the column name given by the user to the synonyms given in the dictionary when the column name given by the user is invalid. Using the dictionary mapping the query analyzer will generate the result.

In the tuple based instantiation user's querying behavior is used to generate result. This will captures the individual witness of the tuple. The similarity between the current user session and that of the previous users should be calculated every time when the user submits a new query.

The fragment based method will compare the query fragments of the active user with the fragments stored in the query log. The idea behind this approach is to recommend queries whose syntactical features match the queries of the current user.

5. Performance Evaluation

The evaluation of the proposed Querie framework is done by using the MySQL Server. The evaluation is based on the following SQL query

SELECT * FROM FACEDICTIONARY

This is given to the Database Query interface. It will generate a set of result which is returned to the user. And also a set of recommendations are generated by using the three methods viz., suggested query by dictionary mapping, suggested query by tuple based query recommendations, suggested query by fragment history. At all times, the active user is able to: (a) formulate a query from scratch, (b) select a recommended query and submit it as it is, or (c) select a recommended query and edit it before submitting it to the database. Moreover, the interface allows the user to browse the database schema, analyze and re-submit queries that were posed during her recent history. A snapshot of the QueRIE prototype is shown in Figure 2. The recommendations are generated in the Figure 3. The recommendations are generated in three different ways i.e., suggested query by dictionary mapping, suggested query by tuple based query recommendations, suggested query by fragment history. From this chart it is understood that the fragment based method is efficient than the other two methods. Because it takes only 0.073208363 second. But the other two methods namely dictionary mapping and tuple based query recommendations takes 0.383681081second and 0.703118613 second respectively.

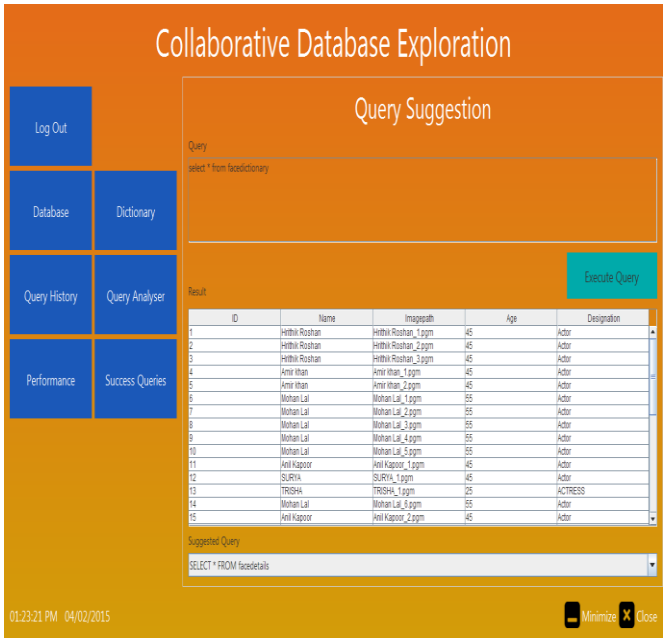

Figure 2: QueRIE interface after a query has been submitted.

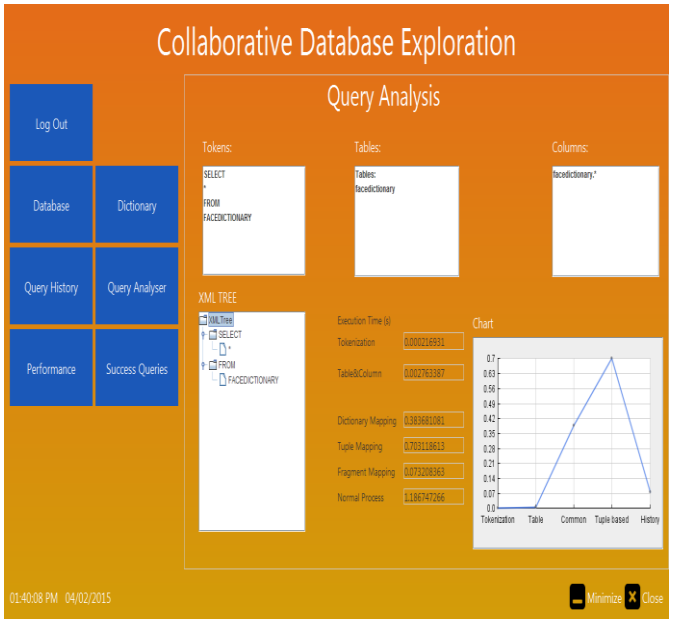

Figure 3: Performance of the various recommendation methods.

The big advantage of the fragment-based approach is that it can be implemented very efficiently; the space of fragments grows slowly allowing for a scalable system. The fragment-to-fragment similarities can be computed offline and stored for very fast retrieval when recommendations need to be generated, leveraging all the advantages of item-to-item collaborative filtering.

6. Conclusion and Future work

The Querie system supports interactive database exploration. This system aims to assist the users of the relational database by generating recommendations. The recommendations are generated by using the three methods viz., tuple based method, fragment based method and dictionary mapping method. And it is clear from the performance evaluation of the above mentioned method that the fragment based method is efficient than the other two. This system is able to generate almost all the recommendation, because it uses all the three methods in a single system.

Future work lies in several areas, one is at developing a more generic and scalable system such as by using matrix factorization methods.

References

[1] Magdalini Eirinaki, Suju Abraham, Neoklis Polyzotis and Naushin Shaikh "QueRIE: Collaborative database exploration," in *Proc. IEEE Transactions*,vol.26, no.7, july 2014.

[2] A. Thusoo *et al.*, "Hive - A petabyte scale data warehouse using hadoop," in *Proc. IEEE 26th ICDE*, Long Beach, CA, USA, Mar. 2010, pp. 996–1005.

[3] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, "Collaborative filtering for interactive database exploration," in *Proc. 21st Int. Conf. SSDBM*, New Orleans, LA, USA, 2009, pp. 3–18.

[4] S. Mittal, J. S. V. Varman, G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, "QueRIE: A recommender system supporting interactive database exploration," in *Proc. IEEE ICDM*, Sydney, NSW, Australia, 2010.

[5] J. Akbarnejad *et al.*, "SQL QueRIE recommendations," *PVLDB*, vol. 3, no. 2, pp. 1597–1600, 2010.

[6] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," in *Proc. 28th STOC*, NewYork, NY, USA, 1996.

[7] E. Cohen, "Size-estimation framework with applications to transitivenclosure and reachability," *J. Comput. Syst. Sci.*, vol. 55, no. 3, pp. 441–453, 1997.Pocket Telephone, Inc.

[8] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.

Author Profile

Sruthi T. K received the B.Tech degree in Computer Science and Engineering from M.G University, Kerala at Sree Buddha College of Engineering for Women in 2013. And now she is doing her M.Tech degree under the same university in Mount Zion College of Engineering.