ISSN:0975-9646

QueRIE: System for Personalized Query Recommendation

Minakshi N. Patil[1], Prof. A.K. Gulve[2]

[1]*PG student (CSE),* [2]*Associate Professor (MCA)*
[1]*Department of computer science and Engineering,* [2]*MCA Department*
[1,2]*Government College of Engineering*
Aurangabad, India.

Abstract: **Personalized query recommendation for database is a major task in information mining. Query recommendation is basically meant for the users who are lacking in SQL expertise and face many problems in handling database schema, joins, primary key foreign key relations, views, clusters etc. We developed a personalized query recommendation system to streamline task for these users. QueRIE continuously monitor task performed by active user and finds a matching pattern with previous user from query log and identifies similar information needs.**
Recorded query fragments are used to match similar query fragments of recorded sessions of previous users which in turn provide potentially interesting queries for active user. Proposed system generates recommendation for real time computations on huge database.

Keywords**: SQL, Tuple based, Fragment based, Interactive exploration, QueRIE, personalize, Query Log, Query Recommendation, Relational Database.**

I. INTRODUCTION

Database management system (DBMS) provides critical approach to access, analyze and manage huge amount of data. Like warehouses which supports business intelligence, data analysis, business exploration, scientific data exploration etc. Despite availability of query processing applications large database new users often face difficulties in understanding database schema, relational database and formulation of queries.

For instance the data warehouse platform used in social networking sites like Facebook, Twitter, linked in, result of heavy usage frontrunners to lot of tables generated in the pool of warehouse and this in turn generate the need of data discovery application, especially for new user [1]. Even for expert users, who are able to handle complex queries, the task of knowledge discovery remain a big challenge as users may unable to understand database schema, relation between databases or may not have required expertise to formulate specific queries. Moreover due to continuous increase in the size of data, exploration of databases is infeasible. The goal of a QueRIE system is to assist users with interactive exploration of a large database [2].

The proposed system generates recommended queries as per required information by active user instead of composing new one, although this recommendation has few technical challenges. QueRIE is inspired by recommender system: If two users A and B posed the same query, later if user A is interested in user B's query and vice versa to explore their databases, collaborative filtering is used to propose this idea, a well known technique used in recommender system [3].

Allocation of this approach into the database context causes several challenges. First, SQL is a declarative programming language and therefore syntactically different query may result same information. We cannot simply compare SQL queries hence we have to resolve difficulties in query-equivalence problem. The second challenge is, regarding knowing which queries are important in the computation of user similarity. Finally recommended queries need to be updated by the user as per their requirement. Closed loop approach is employed to address these challenges [4].

II. THE QUERIE FRAMEWORK

Queries of active user are sent to the DBMS and the recommendation engine as shown in fig 1. DBMS processes each query and returns required information. At the same time each session is stored in a query log. The recommendation engine combines the active user requirements and information gathered from database interaction of previous users as per the recorded sessions in query log and generate query recommendations for the active user by finding maximum matching pattern using Re-ranking algorithm [5].

The QueRIE framework workflow is as shown in figure 1. The queries of active user are forwarded to the DBMS and recommendation engine too. The DBMS processes each query and returns required information. At the same time each query is transferred to query log [6].

The Recommendation Engine combines the active user input with the information stored earlier in the database by previous user, since recorded in the query log, and provide set of recommendations to the active user. The goal of this exploration is to find not only gripping information but also verify the particular hypothesis. The queries are generated based on this goal and reached the active user information needs. As a result, queries reported by the user during one visit to the database are verified; the user developed the new query in the sequence after checking the result of previous queries [7].

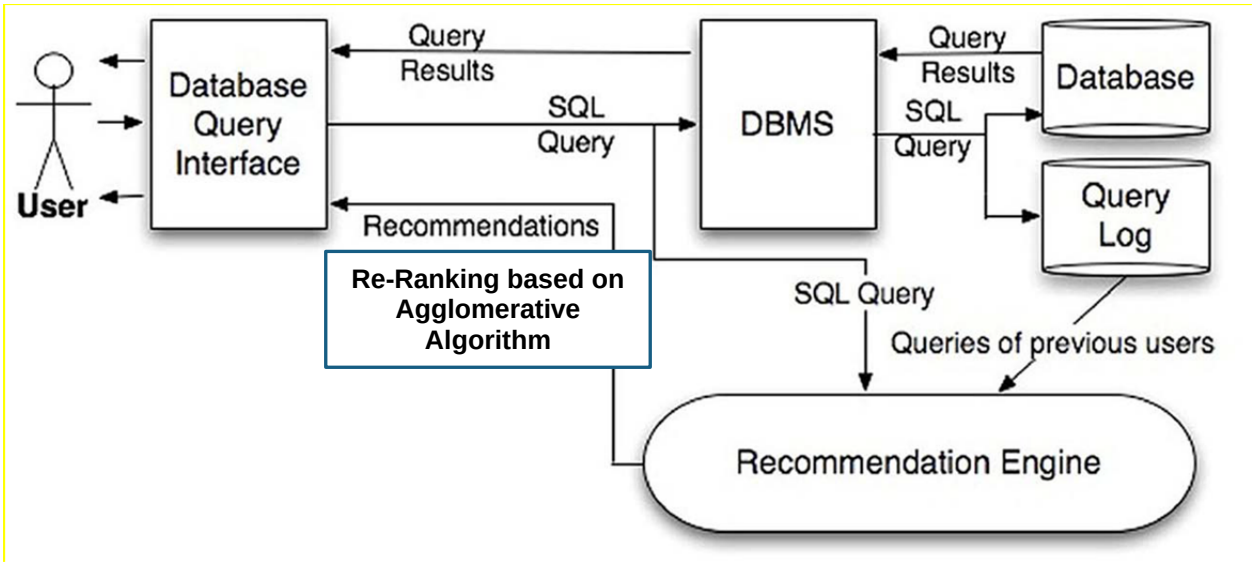


Fig 1: QueRIE framework

Three methodologies are used to generate query recommendations:

a) Dictionary Mapping
b) Tuple based query recommendation
c) Fragment History

Let S_i represent session summary for user i. i=0 represent active user for whom recommendation will generate. i=1….n represent past user of the system. To generate recommendation for current user S_0, the query framework first compute predicted summary S^{pred}. This summary captures the interest of S_0, then S^{pred} used as 'seed' for the generation of recommendations [8]. Predicate summary can be define as

$$S^{pred}=f(\alpha, S_0, S_1, S_2 \ldots\ldots S_n)$$

Where f is a function that combines all information of the active user S_0 and previous users S_1 …..S_n. The mixing factor α is very important, if α=1, S^{pred} takes into account only the queries in S_0 where as if α=0, S^{pred} queries of previous users effect on the recommendation.

Using S^{pred}, the framework generate queries with highest prediction that covers the subset of the database. It is important to recommend meaningful and intuitive queries which should have non empty result sets [9].

III. Implementation

A. Mathematical Model:

Let, I is a set of input i.e. Query which is submitted. F is the set of functions used for the implementation. O is the output.

S= (I, F, O)
I: Input Query
F: Set of Functions
O: Set of Output
F=F1, F2, F3, F4, F5, F6, F7, F8.

F1: Enter keyword based on SQL programming.
F2: Choose Log Filter.
F3: Choose Schema Filter.
F4: Execute the query.
F5: Maintaining of query log.
F6: Extraction of session summary.
F7: Generation of target tuples.
F8: Re-ranking based on clarity score.

F1: Enter keyword based on SQL programming.
X: Enter the particular keyword based on SQL programming to get the recommendations which is desirable, a list of recommended queries will be obtained from which select the one which is needed.
F(X): Based on the previous user log the recommendations will be displayed.

F2: Choose the Log Filter.
X: Log Filter can be selected based on either Individual Log or Collaborative Log.
F(X): Individual log gives recommendation based on current users querying behaviour, while that of the Collaborative Log gives recommendations based on current as well as past user querying behaviour.

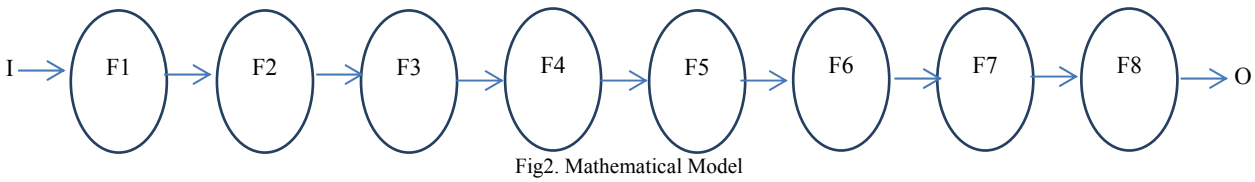


Fig2. Mathematical Model

F3: Choose the Schema Filter.
X: Schema Filter can be selected based on either Tuple based or Fragment based approach.
F(X): The tuple-based approach captures the users querying behavior at a very fine level of detail the individual witnesses to the user queries. While that in the Fragment-based the coordinates of the session summaries correspond to fragments of queries instead of witnesses.

F4: Execute the Query.
X: Query is executed.
F(X): Query is executed and result is displayed to the user.

F5: Maintaining the Query Log.
X: Query Log is maintained.
F(X): Based on the users querying behaviour the query log is maintained.

F6: Extraction of session summary.
X: Session summary is extracted.
F(X): Any Current or Active user whose query matches with the previous user query then that summary can be extracted by the current user.

F7: Generation of Target tuples.
X: Based on session summary generate target tuple.
F(X): Providing only the necessary tuple to the active user.

F8: Re-ranking based on Clarity score.
X: Re-ranking of query is done based on clarity score.
F(X): Target tuples are re-ranked and displayed to the user.

B. Agglomerative Clustering Algorithm

The algorithm forms clusters in a bottom up manner, as given below:
I. Initially, put each article in its own cluster.
II. Among all current clusters, pick the two clusters which are at smallest distance.
III. Replace these two clusters with a new cluster.
IV. Repeat the above two steps until there is only one cluster remain in the pool. Thus agglomerative clustering algorithm will result in a binary cluster tree.

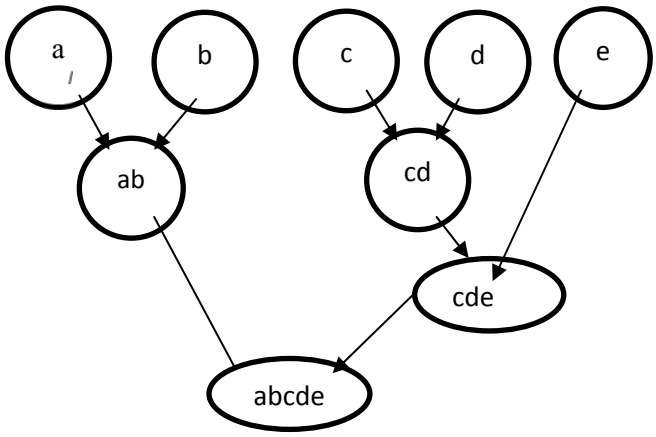


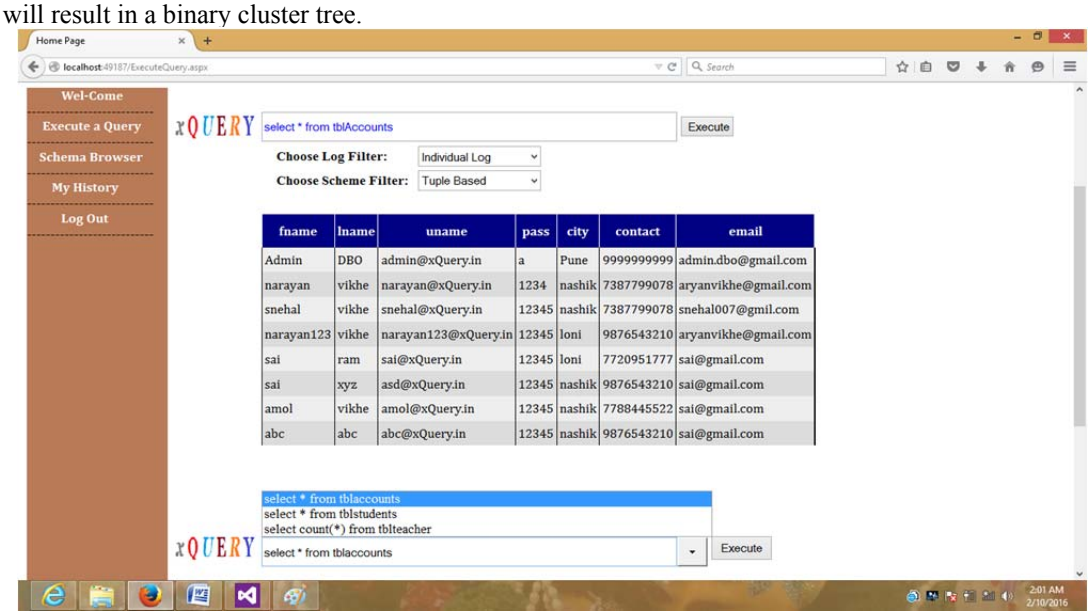
Fig3. Workflow of Agglomerative algorithm

C. Re-ranking:

The ranked retrieval mode has rapidly become the effective way to search exact matching pattern. There are two critical requirements for re-ranking. First, the recommendations precisely follow the specified ranking function, i.e., there is no loss of accuracy and the recommendation service is transparent to the active user as far as query concerned. Second, the query re- ranking service must minimize the number of new queries to be created. This requirement is crucial for two reasons: First is to ensure a fast response time to the user query. Second is to reduce the burden on the active user in order to get query recommendations by previous users [10].

IV. Result Analysis

Evaluation of proposed system is done by using the MySQL server. Simple SQL queries as well as nested queries (sub queries) supported by proposed system. Evaluation is done on the following SQL query.
Select * from tblAccounts



This input is given to the database query interface. It will generate information needed by active user as well as recommendations by using three methods viz, Dictionary Mapping, Tuple Based Fragmentation and Fragment History. The goal of proposed system is to generate recommendations rapidly with maximum matching patterns. To achieve this re-ranking based on agglomerative technique is used, as it is an efficient way to retrieve exact matching patterns rapidly. At any time active user able to: (a) Formulate new query, (b) Select recommended query and submit it as it is or edit it before submitting it to the database. The interface allows the user to browse the schema of database, analyze and re-submit queries. A snapshot of the query model is as shown in figure 4.We can conclude that fragment takes less computation time as compare to other two systems. Fragment based approach grows in scalable system.

V. CONCLUSION

In this paper we design QueRIE framework which is used to generate SQL query recommendations for the current user or active user. Fragment based approach is efficient among all three but it representing coarser level of details. Most important, fragment to fragment information can be stored offline and stored for fast retrieval when recommendation needs to be generate. The experimental result showed that non expert users, who are lack in SQL expertise they able to access required information through recommendation instead, basic knowledge of query language is mandatory for the user.

There are many interesting zones we would like to explore in the future. We would like to measure impact of relaxation process over the recommendations. Exploring sequence based approach is another area of interest for the future work. To find similarities in query sessions pure sequence information is not sufficient. Instead, we might have to work on other strategies for e.g. selection of predicate is more efficient in advance query, in order to properly detect matching pattern. I also plan to focus on relational databases which supports form based interface. Finally, as my aim to develop more generic and scalable system, I am currently working on alternative technique for generating recommendations.

ACKNOWLEDGEMENT:

I wish to express my sincere gratitude to the Head Of Department Prof. V.P. Kshirsagar for providing me an opportunity for presenting a project on the topic ”QueRIE :System for Personalized Query Recommendation”. I sincerely thank my project guide Prof. A. K. Gulve for his guidance and encouragement in the partial stage completion of my project work. I also wish to express my gratitude to the officials and other staff members who rendered their help during the period of my project work. Last but not least I wish to avail myself of this opportunity, to express a sense of gratitude and love to my friends and my parents for their manual support, strength, and help.

REFERENCES

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

[2] ”database, n”. OED Online. Oxford University Press. June 2013. Retrieved July 12, 2013.

[3] G. Linden, B. Smith, and J. York, Amazon.com recommendations: Item-to-item collaborative filtering, IEEE Internet Comput., vol. 7, no. 1, pp. 7680, Jan./Feb. 2003.

[4] Magdalini Eirinaki, Suju Abraham, Neoklis Polyzotis, and Naushin Shaikh. JULY 2014. ”QueRIE: Collaborative Database Exploration”. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 26, NO. 7.

[5] Proctor, Seth (2013). ”Exploring the Architecture of the NuoDB Database, Part 1”. Retrieved 2013-07-12.

[6] E. Cohen, “Size-estimation framework with applications to transitive closure and reachability,” J. Comput. Syst. Sci., vol. 55, no. 3, pp. 441–453, 1997.

[7] J. Akbarnejad et al., “SQL QueRIE recommendations,” PVLDB, vol. 3, no. 2, pp. 1597–1600, 2010.

[8] N. Koudas, C. Li, A. K. H. Tung, and R. Vernica, “Relaxing join and selection queries,” in Proc. 33nd Int. Conf. VLDB, Seoul, Korea, 2006, pp. 199–210.

[9] B. M. X. Jin and Y. Zhou, “Task-oriented web user modeling for recommendation,” in Proc. User Modeling, Edinburgh, U.K., 2005.

[10] Xiaogang Wang, Shi Qiu, Ke Liu, and Xiaoou Tang, “Web Image Re-ranking Using Query-Specific Semantic Signatures”2013