International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 7 4348 - 4352

Query Formulation and Recommendation for Relational Databases Using User Sessions and Collaborative Filtering

Mr. S. D. Chopade
Department Of Computer Engineering
DGOI, FOE, Daund
Savitribai Phule Pune University, Pune
Duand, Pune, India
sagarchopade30@gmail.com

Prof. S. S. Bere
Department Of Computer Engineering
DGOI, FOE, Daund
Savitribai Phule Pune University, Pune
Duand, Pune, India
sachinbere@gmail.com

Abstract- Structured Query Language (SQL) has a uniform structure over different programming languages. The queries fired on Database Management System (DBMS) contain textual information along with selected segments of data parsed by data base management system to fire it as a structured query. Currently DBA needs to execute complex queries on large databases. Many times user or DBA fires similar queries on database server to get useful information. The queries which are similar to each other can then be categorized into two types a) the tuples retrieved by SQL queries are similar b) the fragment of the queries are similar. System gives recommendation to those similar queries so that it saves the time of DBA to construct it again and again. Query suggestions given to DBA or users are known as Query Recommendation. To develop a Query Recommendation system many authors suggested the use of Query Log. Query suggestions are divided into two areas mainly Collaborative Recommendations and Single Log Recommendations. This system is designed by single or collaborative log using parameter known as mixing factor. In this paper we analyzed Sql query Recommendation concepts and their uses.
There are basically two types of similarity measure for Query Recommendation considered in [1] such as 1) Fragment Based 2) Tuple Based. Here in this research paper we are motivated towards generating recommendations for nested SQL queries. We adopt hierarchical classification on query log to create classes of similar queries and further to generate recommendations for SQL Query we proceed with finding matching class from which the recommendations can be modeled.

Keywords- *Recommendation Systems, SQL Query, SQL Query Log, DBMS, Mixing factor, Query Fragments.*

*****

I. INTRODUCTION

Database systems are largely recommended in scientific community. These databases usually employ a web-based interface that allows users to submit SQL queries and retrieve the results. Relational database systems executes complex queries on large data sets, the discovery of useful information remains a big challenge. As an example, users who are not familiar with the database may overlook all queries data that are not useful, or user may don't know data provided by database contains relevant information or not. Many times users of database system don't have relevant knowledge of sql queries that would allow them to run complex queries and retrieve results [1].
To address this important problem of assisting users when exploring a database, we designed the QueRIE (Query Recommendations for Interactive data Exploration) system using two methods- 1) Tuple based recommendation which maintains session of particular user. 2) Fragment based recommendation which fragment the quire for comparison.
QueRIE system continuously monitors the users querying behavior and finds matching patterns in the systems query log, in an attempt to identify previous users with similar information needs and uses these similar users and their queries to recommend queries that the current user may find interest-ing. In this we describe an instantiation of the framework, where the active users 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 SQL database and propose a scalable design that enables the incremental update of similarities, making real-time computations on large amounts of data feasible [2].

II. RELATED WORK

In Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang and Raghotham Murthy, Hive- A Petabyte Scale Data Warehouse Using Haddop, Hadoop is a popular open-source map-reduce implementation which is being used in companies like Yahoo, Facebook etc. to save and execute every big data sets on commodity hardware. Map-reduce is programming model for processing large data set with a parallel distributed algorithm on a cluster. Hive also includes a system catalog megastore that contains schemas and statistics, which are useful in data exploration, query optimization and query compilation. In facebook, the Hive warehouse contains tens of thousands of tables and stores over 700TB of data and is being used extensively for

both reporting and ad-hoc analyses by more than 200 users per month. The entire data processing infrastructure in facebook prior to 2008 was built around a data warehouse built using a commercial RDBMS. The data that we were generating was growing very fast - as an example we grew from a 15TB data set in 2007 to a 700TB data set today. In current world Hadoop is a technology that addresses our scaling needs. Hadoop was already an open source project that was being used at megabyte scale and provided scalability using commodity hardware was a very compelling proposition for us. The same jobs that had taken more than a day to complete could now be completed within a few hours using Hadoop. Currently author considers only a subset of SQL as valid queries. Authors are working towards making HiveQL subsume SQL syntax. Hive currently has a naive rule-based optimizer with a small number of simple rules [5].

In Badrul Sarwar , George Karypis , Joseph Konstan and John Riedl, Item Based Collaborative Filtering Recommendation Algorithms, Science and Engineering University of Minnesota, Minneapolis, MN 55455 Recommender systems apply knowledge discovery techniques to the problem of making personalized recommendations for information, products or services during a live interaction. These systems, especially the k-nearest neighbor collaborative filtering based ones, are achieving widespread success on the Web. The tremendous growth in the amount of available information and the number of visitors to Web sites in recent years poses some key challenges for recommender systems. These are: producing high quality recommendations, performing many recommendations per second for millions of users and items and achieving high coverage in the face of data sparsity. In traditional collaborative filtering systems the amount of work increases with the number of participants in the system. New recommender system technologies are needed that can quickly produce high quality recommendations, even for very large-scale problems [11].

In Gloria Chatzopoulou, Magdalini Eirinaki and Neoklis Polyzotis, Query Recommendations for Interactive Database Exploration. Users employ a query interface to issue a series of SQL queries that aim to analyze the data and mine it for interesting information. In this paper, authors present a query recommendation framework supporting the interactive exploration of relational databases and an instantiation of this framework based on user-based collaborative filtering. Such queries need to be considered in the recommendation process. First-time users, however, may not have the necessary knowledge to know where to start their exploration. Other times, users may simply overlook queries that retrieve important information. The experimental evaluation demonstrates the potential of the proposed approach. The authors should stress that this is a first-cut solution to the very interesting problem of personalized query recommendations. There are many open issues that need to be addressed. For instance, an interesting problem is that of identifying similar queries in terms of their structure and not the tuples they retrieve. Two queries might be semantically similar but retrieve different results due to some filtering conditions [9].

The Javad Akbarnejad , Gloria Chatzopoulou , Magdalini Eirinaki, Suju Koshy, Sarika Mittal, Duc On, Neoklis Polyzotis and Jothi S. Vindhiya Varman, SQL QueRIE Recommendations, This system aims at assisting non-expert users of scientific databases by tracking their querying behavior and generating personalized query recommendations. The system is supported by two recommendation engines and the underlying recommendation algorithms. The first 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. The second identifies structurally similar queries to the ones posted by the current user. Both approaches result in a recommendation set of SQL queries that is provided to the user to modify, or directly post to the database. The demonstrated system will enable users to query and get real-time recommendations from the SkyServer database, using user traces collected from the SkyServer query log. QueRIE does not require an explicit user profile or keyword-based queries. On the contrary, it closes the loop by accepting SQL queries as input, decomposing them in order to identify interesting database areas for each user, and re-transforms them in SQL queries that are presented as recommendations.

III. PROPOSED SYSTEM

The Design architecture of the system as depicted Fig.1. The queries that are relevant are passed to both the DBMS and the Recommendation Engine. The data base management system executes every query and gives a set of results. At the same time, the query is stored in the Query Log. The Recommendation Engine combines the current users input with information gathered from the database interactions of past users, as recorded in the Query Log, and generates a set of query recommendations that are returned to the user.

A. Preliminaries

Tuple-Based Query Recommendation: In this instantiation of the QueRIE framework, the session summary Si is represented as a weighted vector, where every coordinate corresponds to a distinct database tuple. We assume that the total number of tuples in the database, and as a consequence the length of the vector, is T. The weight $S_i[]$ represents the importance of a given tuple 2 T in session S_i, and is non-zero only if is a witness for at least one query in the session.

 ISSN: 2321-8169
4348 - 4352

The intuition is that Si captures the tuples in the base tables that are touched by the queries in the user's session. So, session having similar queries will map to the equivalent summary [3][4].

We assume that the vector SQ represents a single query Q. The value of each element SQ[] signifies the importance of the tuple as the witness for Q. [6]

Fragment-Based Query Recommendations: The fragment-based instantiation of the QueRIE framework works in a similar manner to the tuple-based one. The two main differences lie in the representation of the session summaries and the formulation of similarities. More specifically, the coordinates of the session summaries correspond to fragments of queries instead of witnesses. We identify as fragments the following syntactical features of the queries in the session: attribute references, table's references, join and selection predicates. At a high level, the idea behind this approach is to recommend queries whose syntactical features match the queries of the current user.

User-based collaborative filtering main disadvantage is that it inherently requires real-time similarity calculations, as the active users profile gets updated. This significantly slows the real-time generation of recommendations, making such a choice inappropriate for large-scale systems. On the other hand, item-based collaborative filtering performs all similarity calculations during the training process, and thus has much smaller overhead during the recommendations generation phase. This is the reason why we decided to follow a methodology similar to the item-based collaborative filtering. Our objective is to identify fragments that co-appear in several queries posed by different users, and use them in the recommendation process. These fragments may, or may not include the ones in the users active session S0 depending on the value of the mixing factor . Thus, QueRIE first calculates (offline) the pair-wise similarities of all query fragments recorded in the query logs. These similarities are subsequently used to predict, in real time, the rank (i.e. importance) of each fragment with regards to the current user session. In turn, the highest ranked query fragments are the query characteristics used to mine the query logs and select the most relevant queries that are used as recommendations [4][6].

B. The Framework

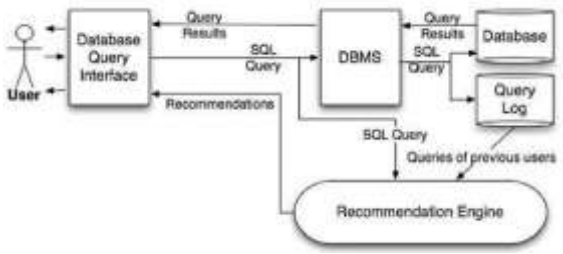

Figure 1: System Architecture

Algorithm:

Input:

Set Of SQL Queries Q = {Q_0, Q_1, Q_2……..Q_{n-1}}

Threshold thr=0.5

Output:

Set Of Clusters C= {C_0, C_1, C_2……C_{m-1}}

Steps:

Step 1: Create first cluster C_0, assign first Query Q_0 to C_0

C_0<- Q_0

C= C U C_0

Step 2: for i=1 to n-1

flag=0

mostmatching=-1

Do

For j=0 to m-1

2.1 CalculateCosineSimilarity, sim=(Q_i, C_j)

If(sim>= thr and sim>= flag)

flag=sim

mostmatching=j

2.2 if(mostmatching==-1)

Create new cluster Cm

C_m<- Q_i

C=C U C_m

Else

$C_{mostmatching}$<- Qi

End

IV. DATASET AND RESULT ANALYSIS

The general query log is a general record of what mysqld is doing. The server writes information to this log when clients connect or disconnect, and it logs each SQL statement received from clients. The general query log can be very useful when you suspect an error in a client and want to know exactly what the client sent to mysqld [1]. We extensively tested the proposed system on a sample SQLEXPRESS database containing 7 relational tables. The log contains 60 simple SQL queries and 40 nested SQL queries. Further we fired some queries over database along with Fragment based query recommendation, tuple based recommendation and hierarchical classification based nested query recommendation. Then we calculated accuracy in terms of precision and availability in terms of recall to find efficiency of each technique.

 ISSN: 2321-8169
4348 - 4352

Sr. No	#Relevant Queries in log	#Recommended Queries	#Correctly recommended Queries	Recall	Precision
1	35	8	3	0.23	0.38
2	67	21	10	0.31	0.48
3	31	17	12	0.54	0.70
4	16	13	10	0.76	0.76
5	10	9	7	0.90	0.77

Table 1: Results Analysis for Tuple based Recommendations

Sr. No	#Relevant Queries in log	#Recommended Queries	#Correctly recommended Queries	Recall	Precision
1	31	10	6	0.32	0.60
2	32	15	11	0.47	0.73
3	25	18	15	0.72	0.83
4	15	14	12	0.93	0.86
5	9	9	8	1.00	0.89

Table 2: Results Analysis for Fragment based Recommendations

Sr. No	#Relevant Queries in log	#Recommended Queries	#Correctly recommended Queries	Recall	Precision
1	51	16	10	0.31	0.63
2	26	13	11	0.50	0.85
3	19	14	13	0.74	0.93
4	27	25	24	0.92	0.96
5	9	9	9	1.00	1.00

Table 3: Results Analysis for Classification based nested Query Recommendations

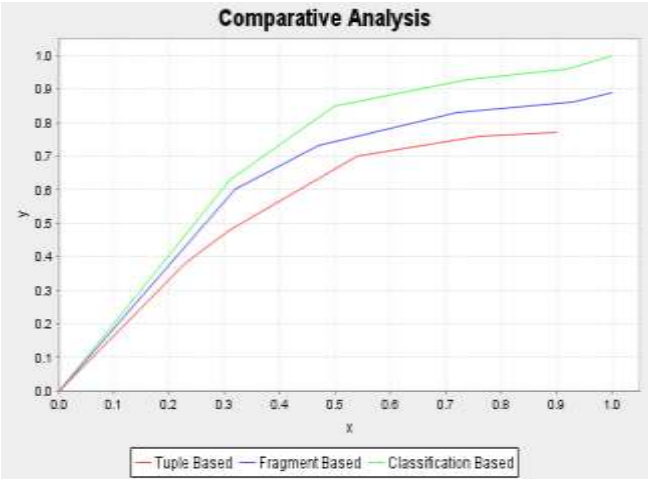

Figure 2: Precision Vs Recall graph

As shown in Figure2 the three techniques and their respective Presion Vs Recall graphs are plotted. For Tuple based recommendation we found 0.55 (55%) availability of the reccommendations and corresponding accuracy 0.62 (62%). Further for the Fragment based recommendation we found 0.67 (67%) availability of the reccommendations and corresponding accuracy 0.78 (78%). Then we tested hirarchical classification based recommendation and found availability as 0.69 (69%) while precision as 0.87 (87%). The extensive analysis shows that classification based recommendation has comparatively high precision and recall values than Tuple based and Fragment based recommendation techniques.

V. CONCLUSION

In this paper we present the QueRIE framework that aims to generate useful SQL query recommendations to users of relational databases. Taking into consideration the findings of our previous work, where we developed a tuple-based instantiation of the framework using user-based similarities to generate recommendations, we decided to follow an item-based approach using query fragments to represent user sessions. On the other hand, the fragment-based approach can be implemented very efficiently; the space of fragments grows slowly, the summaries are very sparse and, most importantly, the fragment-to-fragment similarities can be computed and stored for very fast retrieval when recommendations need to be generated. The analysis showed that this trade-of between computational efficiency and accuracy is worth pursuing, since we are able to have a scalable implementation running with real, big data, with an acceptable loss in precision. In fact, when the tuple-based instantiation employs approximation techniques to enable real-time calculations, the loss in precision is much greater than that of the fragment-based one.

ACKNOWLEGEMENT

I express great many thanks to Prof. S. S. Bere and Department Staff for their great effort of supervising and leading me to accomplish this fine work. They were a great source of support and encouragement. To every person who gave me something too light along my pathway. I thanks for believing in me.

REFERENCES

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

[2] A. Thusoo et al., Hive - A petabyte scale data warehouse using hadoop, Proc. IEEE 26th ICDE, Long Beach, CA, USA, Mar. 2010, pp. 9961005.

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

[4] J. Akbarnejad et al., SQL QueRIE recommendations,PVLDB, vol. 3, no. 2, pp.15971600, 2010.

[5] N. Alon, Y. Matias, and M. Szegedy, The space complexity of approximating thefrequency moments,Proc. 28th STOC, New York, NY, USA, 1996.

[6] E. Cohen, Size-estimation framework with applications to transitive closure andreachability,J. Comput. Syst. Sci., vol. 55, no.3, pp. 441453, 1997.

[7] 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.

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

[9] V. Singh et al., "Skyserver traffic report—The first five years,"Microsoft Research, Tech. Rep. MSR TR-2006-190, 2006.

[10] B. Liu, Web Data Mining: Exploring Hyperlinks, Contents and UsageData, 2nd ed. Berlin, Germany: Springer, 2007.

[11] B. M. X. Jin and Y. Zhou, "Task-oriented web user modeling forrecommendation," in Proc. User Modeling, Edinburgh, U.K., 2005.

[12] B. Mobasher, "Data mining for personalization," in The AdaptiveWeb: Methods and Strategies of Web Personalization. Berlin,Germany: Springer, 2007, pp. 90–135, LNCS 4321.

[13] K. Stefanidis, G. Koutrika, and E. Pitoura, "A survey on representation,composition and application of preferences in databasesystems," ACM Trans. Database Syst., vol. 36, no. 4, Article 19, 2011.

[14] G. Koutrika, "Personalized DBMS: An elephant in disguise or achameleon?" IEEE Data Eng. Bull., vol. 34, no. 2, pp. 27–34, Jun.2011.

[15] S. Borzonyi, D. Kossmann, and K. Stocker, "The skyline operator,"in Proc. IEEE ICDE, Heidelberg, Germany, 2001.

[16] J. Chomicki, Preference formulas in relational queries,ACM Trans. Database Syst., vol. 28, no. 4, pp. 427466, 2003.

[17] W. Kiessling, Foundations of preferences in database systems,Proc. Int. Conf. VLDB, Hong Kong, China, 2002.

[18] W. Kiessling, M. Endres, and F. Wenzel, The preference SQL system - An overview,IEEE Data Eng. Bull., vol. 34, no. 2, pp. 1118,Jun. 2011.

[19] G. Koutrika and Y. Ioannidis, Personalized queries under a generalized preference model,Proc. 21st ICDE, Washington, DC,USA, 2005.

[20] J. Levandoski, M. Mokbel, and M. E. Khalefa, FlexPref: A framework for extensible preference evaluation in database systems,Proc. IEEE ICDE, Long Beach, CA, USA, 2010.

[21] E. Pitoura, K. Stefanidis, and P. Vassiliadis, Contextual database preferences,IEEE Data Eng. Bull., vol. 34, no. 2, pp. 1926,Jun. 2011.

[22] A. Giacometti, P. Marcel, and E. Negre, Recommending multidimensional queries,. 11th Int. Conf. DaWaK, Linz,Austria, 2009.