A SURVEY ON RANKING SQL QUERIES USING SKYLINE AND USER PREFERENCES

Ritu Maheshwari[1] J. Raja Sekar[2]

[1]PG Scholar, Computer Science and Engineering, Mepco Schlenk Engineering College, Tamil Nadu, India
[2]Assistant Professor, Computer Science and Engineering, Mepco Schlenk Engineering College, Tamil Nadu, India

Abstract

Databases consist of large sets of data, finding relevant data among it is a tedious task. However when a user enters the query in a user interface, it results in much of the extraneous data. This extraneous data is not relevant to user. The solution to this problem is to obtain a set of non dominated records called skyline records and rank the records according to user preferences. The purpose of this survey paper is to illustrate how skyline operation is implemented and how to capture user preferences. Various techniques of user preferences and algorithms for skyline have been elaborated in this paper. To navigate the user according to their preferences clustering techniques are also used. Clustering can be done by group by operator, by clustering techniques, and by automatic categorization methods. Navigation tree is obtained on the basis of the clusters. A study of capturing users' preferences and navigating results to user is described in this paper.

Keywords: *Navigational behavior, ranking, skyline, users' preferences.*

----------***----------

1. INTRODUCTION

Databases are a collection of data. User searches their required data in the database by entering the query in the form based interface. All the results obtained from the query are not relevant to user. Thus again user will refine the results according to their preferences until they obtain their required outcomes. So, this refining process is time consuming and many users will not be interested to reach the final step.

The solution to this problem is first, to fetch the skyline records. Skyline is defined as the tuples which are not dominated by other tuples in the database. For example, the hotel database consists of attributes *price* and *distance*. The query with a hotel price Rs 4000 and distance 2 miles dominates the query with a hotel price Rs 8000 and distance 3 miles. Skyline operation can be useful for the travel agency. According to example, skyline eliminates all the hotels in which users or customers are not interested. Second, user preferences are captured by user navigational behaviour and query history. Navigational behaviour allows the resulted query to be navigated to various options. The options are the sets of small group, if the condition matches the query; it falls under one of the group. To form the group, clustering technique is used. Moreover, automatic categorization and group by operator can also be used for forming groups. Automatic categorization forms a tree-like structure to organise the results. Group by operator specifies the groups on the basis of the major attribute in a database table. Query history captures user's previous or past preferences, but the problem is that it cannot capture user's current preferences. Also, user preferences are dynamic it changes over time and situation. If a new user comes in then the query history alone is not sufficient to display the desired results because the history is nil for the new user. Thus query history alone is not sufficient to capture user preferences and so navigational behaviour comes into picture.

Example 1: Consider a test car dataset which consist of 4000 records. Navigational tree is shown in **figure 1**. The nodes of tree represent the attribute of the database. Every tuple will be navigated on the basis of the condition specified on the attribute. The parenthesis near the root node represents the number of records in the database. Initially 4000 records are present, if user specifies that *modelyear*=2014 then he should examine only 800 records instead of 4000.

It is difficult to obtain user preferences because extra effort has to be made by user to specify his/her preferences. There exist two challenges addressing the issues of user preferences. First, summarize the user preferences of existing user in the system. Second, deciding the subset of user preferences associated with the particular user.
Root

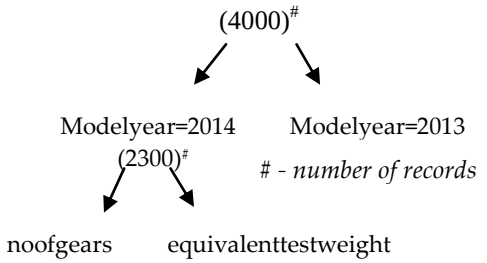

Fig 1: User navigational behaviour

Finally, ranking process has to be applied on the above generated results. Ranking the queries reduces the time of user and provides the most updated queries to the user.

Ranking the records on the basis of centroid is the basic idea. But it is inappropriate because user want to find the optimized records example, the no. of gears in the car, axle ratio of the car rather than the records which are nearby centroid example the average axle ratio. Rank support vector machine is the upcoming ranking method in the information retrieval community. But still there exists various disadvantages in it. Some of the disadvantages are that it cannot handle relational database, expensive etc.

Chen[1] used an approach that aims at relational data. It uses training set examples inferred from user navigational behaviour and skyline. In this approach the ranking functions are adjusted dynamically for each cluster. Thus it captures the user interest. Moreover it does not provide the global ranking approach and does not require ordered set.

The general architecture is shown in figure 2.

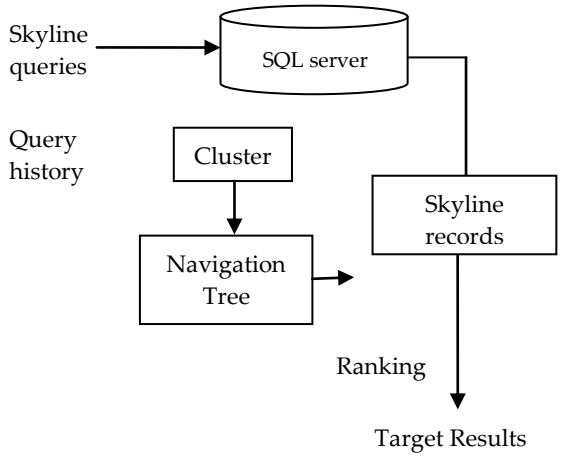

Fig 2: General architecture of the system

The general architecture describes that the process is conducted in two steps. One, to obtain skyline records by writing an appropriate skyline query. Second, to fetch the query history of the users and on the basis of that clusters are formed. These clusters form navigational tree. The ranking is applied in the skyline query and navigational tree. Finally the targeted results are obtained.

The rest of the paper is focused on the following section: Section 2 on skyline implementation. Section 3 specifies on how to capture user preferences. Section 4 discusses the methods of ranking. Finally, Section 5 describes the conclusion portion.

2. SKYLINE

Skyline queries are the most preferred queries. The term skyline refers to collection of tuples which are not dominated by any other tuple in the database. Dominated tuple *p* is defined as a tuple which dominates all the other tuples in the database in every aspect. The tuple can be dominated on basis of smaller value or larger value depending on the value of the attribute specified. The benefit of using skyline query is that it does not need any value to be specified by user, comparison is made among the attribute in the database. Skyline query also has a disadvantage that the output is not bounded in size so in that case all tuples acts as a skyline results. Some of the algorithms [2] are discussed below.

2.1 Translating a Skyline Query into Nested SQL Query

This section shows how skyline queries can be implemented on relational database system. The test car dataset is used for writing skyline query. It focuses on modelyear and axleratio. The query can be written as

```
SELECT * FROM maincar c WHERE
c.modelyear = 2014 AND NOT EXISTS
        (SELECT * FROM maincar c1 WHERE
c1.modelyear =2014 AND
c1.axleratio < c.axleratio)
```

This query has poor performance because of the following reasons:

1. Nested loop
2. Expensive when merged with other SQL operators.
3. More time consumption

2.2 Basic Block Nested Loop Algorithm

Basic block nested loop algorithm continuously reads the set of tuples. The idea behind this algorithm is to keep a window. Window is the set of tuples, but initially window is empty. When a tuple *p* acts as the input then *p* is compared with the set of tuples in the window and on the basis of this comparison, two cases are possible. First, if *p* is dominated by the other tuples in the window then p is eliminated and is not considered for further iterations. Second, if *p* is not dominated by the other tuples in the window then *p* is inserted in the

window and other dominated tuples are removed from window. Thus finally, window at the end consist of non dominated tuples called as skyline records. But the flaw with this algorithm is that it works well with small datasets.

3. USERS' PREFERENCES

Users' preferences are difficult to capture for dynamic users. Query history can be retrieved for the user who have static preferences and those who are not new to database.

One solution is to obtain clusters and form navigational tree. For example, if mutual fund dataset is considered which consist of the attribute high return and low return, then three clusters can be obtained from that. One for high return, second for low return and the third for the funds in which no user is interested. Navigational tree can be obtained when the user enters the query. This query will fall into any one of the clusters generated above and then tree is constructed automatically over these intersected clusters. Thus finally this tree is displayed to the user. Accordingly, user can browse through the cluster; can perform ranking or categorization according to his/her needs. The next sub section 3.1 and 3.2 describes how to cluster on the basis of references and then how to generate navigational tree respectively.

3.1 Preference Based Clustering

It describes how to cluster data on the query history basis. Two methods are described for that, Query pruning and Query merging.

The query pruning process is based on two heuristics. First queries with empty answers are not useful. Second, initially user will give general query and obtains multiple results. User iteratively does this process to refine the results and obtain the desired answers. Thus only the last refined result is useful to user. Consider two queries Q_i and Q_j. A general way to verify the refinement result is to execute a SQL statement such that Q_i minus Q_j results in empty set. This allows some queries to be pruned and it works more efficiently rather than executing the queries.

Query merging process is based on the semantic similarity. Similarity can be obtained when two different quires result has some relationship in between them. Query merging can be obtained by greedy algorithm. Greedy algorithm iteratively merges the pair of query clusters until no pairs can be merged. Calculate the average distance between the pair of queries. Merging can be done on the basis of smallest average distance.

3.2 Navigational Tree Construction

Navigational tree construction is analogous to a decision tree. The aim is to minimize the cost of navigation. Cost of navigation includes the cost of the cost of visiting the intermediate node and cost of visiting the leaf nodes. Obtaining the tree with minimal cost is a NP-hard problem. The algorithm includes those attributes which have been queried by the user so that when clusters are formed on the basis of these attributes. Cluster formation depends on the attribute retrieved from the query history. The algorithm scans the tree from root to leaf. In clustering, every record is assigned a class label. If all the records have the same class label the algorithm stops or else algorithm splits that node to expand the tree.

The attributes are classified on the basis of categorical and numerical. Categorical attribute generates a new subtree for each value of the attribute whereas numerical attribute can form binary tree or multi way tree. Binary split considers all possible locations. Moreover, it selects the best partition among all the possibilities.

4. RANKING

Ranking of results is done in order to reduce the time of user. Moreover, the desired results are placed on the top of the results obtained from the database. There are various ranking approaches and in the recent scenario there exists a ranking function which dynamically adjusts the ranking function for each cluster. The ranking function is defined as a function f_p for a group g_p can be expressed as

$$F_p(r, Q) = \sum_{j=1}^{l} w_{pj}\, s_j(r)$$

Where s_j is a ranking term obtained by skyline operator, w_{pj} is the weight of that term and l is the number of rules in skyline operator.

The technique for ranking is support vector machine (SVM). Ranking SVM is pair-wise ranking method. It is used to sort the results obtained from the specific query on the relevance characteristic. A mapping function is required to define each data pair onto a feature space. These features combined with user's click-through data can be considered as the training data for machine learning algorithms. Generally, Ranking SVM includes three steps in the training period:

1. It maps the similarities between queries and the clicked pages onto certain feature space.
2. It calculates the distances between any two of the vectors obtained in step 1.
3. It forms optimization problem which is similar to SVM classification and solve such problem with the regular SVM solver.

5. CONCLUSIONS

The main aim of this survey is to capture user preferences and perform ranking on user preference results. The skyline operation is used to eliminate all the results in which users are not interested. Moreover, users query history and preferences

are also captured. Many techniques have been listed to implement skyline and user preferences. Finally ranking is performed to obtain the desired result in much lesser time.

ACKNOWLEDGEMENTS

I am thankful to my guide J. Raja Sekar and HOD Dr.K.Muneeswaran who gave me the golden opportunity to do a survey on this wonderful title named *Ranking SQL queries using skyline and user preferences.* This survey not only helped me in doing a research but also I came to know about so many new things.

REFERENCES

[1] Zhiyuan Chen, Tao Li, and Yanan Sun, "A Learning Approach to SQL Query Results Ranking Using Skyline and Users' Current Navigational Behaviour ", IEEE transaction on knowledge and data engineering, Vol.25,No. 12, pp. 1-11, December 2013.

[2] Stephan Borzsonyi, Donald Kossmann, Konrad Stocker, "The Skyline Operator", IEEE transaction on knowledge and data engineering, pp. 1-10, 2011.

[3] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis, "Automated Ranking of Database Query Results," Proc. Conf. Innovative Data Systems Research (CIDR), 2003.

[4] O. Chapelle and S.S. Keerthi, "Efficient Algorithms for Ranking with SVMs", Information Retrieval, vol. 13, pp. 201-215, June 2010.

[5] Z. Chen and T. Li, "Addressing Diverse User Preferences in SQL Query-Result Navigation," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 641-652, 2007.

[6] C. Li, M. Wang, L. Lim, H. Wang, and K.C.-C. Chang, "Supporting Ranking and Clustering as Generalized Order-by and Group-By," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 127-138, 2007.

[7] K. Chakrabarti, V. Ganti, J. Han, and D. Xin, "Ranking Objects Based on Relationships," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 371-382, 2006.

[8] C. Li, M. Wang, L. Lim, H. Wang, and K.C.-C. Chang,"Supporting Ranking and Clustering as Generalized Order-by and Group-By," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 127-138, 2007.

BIOGRAPHIES

Ritu Maheshwari is pursuing ME in computer science and engineering from Mepco Schlenk Engineering College, Sivakasi. She has completed her BE in computer science and Engineering in RKDF IST Bhopal. Her area of interest is data mining and M2M technology.

J. Raja Sekar is pursuing PhD in computer science and engineering from Anna University, Trichy. He has completed his ME (CSE) from Anna University, Guindy and BE (CSE) from Mepco Schlenk Engineering College. His area of interest is image processing and internet programming.