Literature Survey on Preference Database

Divyavani .P , Meenakshi .R , Monica .K.V

B.E Computer Science and Engineering, R.M.K Engineering College (Anna University)

R.S.M Nagar, Kavaraipettai, Chennai-601 206, Tamilnadu, India.

ABSTRACT

Preference data base (Pref DB) used to provide appropriate recommendations to the users on their relevant search and preferences. The design is made such that user interested data is obtained from the database. PrefDB uses object oriented query technique to retrieve the data from the database. A very good example we are dealing in this paper is getting the favorite movie of the user (movie recommendation) based on his interest. The main aim is to retrieve the user interested movie as the interests of the user changes with respect to time.

Keywords: *PrefDB, Query Parsing, Recommendation, Query Optimization, Query Execution, Relational Database and Joins.*

1. INTRODUCTION

PrefDB takes user profile along with preferences and stores in the database. Already there exists two traditional approaches plug-in and native approach. In plug-in approach preferences are translated into complex queries. In native approach operators are injected into the query engine to execute the queries along with the preferences. The disadvantage of plug-in is, it operates above the query engine. Therefore it is hardwired. The disadvantage in native approach is the entire database core must be changed. *PrefDB* is the use of an extended relational data model and algebra that allow expressing different flavors of preferential queries [4].

In prefDB once the user has logged in, the user has to provide his profile information with his interests. For example in movie rental application the user provides his interests in terms of favorite movies, actors, directors, genres. The movie will be recommended by filtering the recommendation in terms of high, medium and low level as per the user's choice. In high level recommendation there will be more number of constraints ,which results in retrieving closest match to the users interest and preferences that is in a movie rental application you will get the favorite movie .Whereas, in medium and low level recommendation the number of results will be more, if the level is medium then the results will be based on favorite movie and director and if the level is low the results will be based on all the four categories , but the movies which satisfy any one constraint will be selected .

Similar to prefDB there is careDB which is used in directing the users to the interested restaurants based on his cuisine interests .Here, the external factors like traffic, climate, waiting in restaurants, etc are also included. The careDB along mapping software, location detection software (GPRS, Antenna)

processes the query and provides the user with the best result of the preferred restaurant of the user in his handheld device. The result will be based on the score and confidence will vary the result based on the previous choice of restaurants chosen by the customer. The constraints for the query formation will include the environmental factors like climatic changes, road block, traffic etc... And other external factors like long waiting time in restaurants, restaurants closed preferred cuisine unavailable etc... The users get the query result in their handheld device with a map layout for showing the direction.

2. APPROACH TO QUERY PROCESSING IN PREF DB

PrefDB is mainly based on getting the user interest and parsing the query and then the query is extended and sent to a query optimizer and then to the execution engine for executing the query.

2.1 QUERY PARSING:

The query parser along with the user input and preferences keeps the order of the operators to construct a baseline extended query. It also creates additional joins. The query parser along with the joins adds the prefer operator too for each preference. In the conditional or scoring part it checks whether it involves an attribute for each preference. Through this an extended query plan is formed.

2.2 QUERY OPTIMIZATION:

Optimization which intends to rewrite the initial query such that we can expect an improved efficiency during the evaluation of the query [2]. The context-aware query optimizer should be able to decide on executing only parts of the query to prune the search before executing more expensive modules [8]. PrefDB's hybrid implementation enables operator-level query optimizations without being obtrusive to the database engine [4].

There are two tables where, one is the base table and the other is the score table. Non-preference query part is the base table tuples and the preference evaluation is done on score table. There are two costs involved because of these two tables.

The joins and filters which are performed in the tuples are not affected by the prefer operator or the extended relational operator. Thus, the non preference related cost is not affected. The main aim of the query optimizer is to reduce the cost related with query optimization. For this, we have to have the join order and must consider the non preference related cost as fixed. The disk I/O's is the critical parameter that modifies the processing cost of query evaluation.

The main aim is to,

1. Reduce the number of tuples which are given as input to the prefer operator

2. Limit the search space

There are two plan enumeration algorithm proposed rule-based query optimization and cost based query optimization. These algorithms are used in examining alternative position of prefer operator, to estimate cost and to select the plan whichever has the cheapest cost.

Query Modification is one of the phases of Query

2.2.1 RULE-BASED QUERY OPTIMIZATION

The query optimizer gets the plan to be executed but that plan does not include preferences. It actually retrieves that plan from a native query optimizer. Then it makes the necessary changes

which follows the join order. The main aim is to reduce the number of tuples that flow through the operators in a query plan.

1. Selecting the attributes other than score and confidence and pushed down the query plan [1].If the selection as many relations, it can be split based on conditions and each part is pushed down separately.

2. Projections are pushed down the query plan as far as possible ensuring that required attributes will still be available for subsequent operators [1].

3. If a prefer operator defined on a binary operator involves attributes only on one part of the operator, if it is pushed down to this part [1].

4. All selection conditions that involve attributes other than score and confidence are added to the conditional part of the prefer operator, if both prefer and select operator are defined on the same relation.

The above four are the heuristics that our optimizer applies. For restricting the number and size of the tuples heuristics 1 and 2 are used. To set prefer operator to the deepest efficient position in the plan we use heuristic 3. For query execution algorithm we use the heuristic 4.

2.2.2 COST-BASED QUERY OPTIMIZATION

Cost-based query optimizers evaluate the resource footprint of various query plans and use this as the basis for plan selection [3]. Costs are used to estimate the runtime cost of evaluating the query, in terms of the number of I/O operations required, CPU Path Length, amount of disk buffer space, disk storage service time, and interconnect usage between units of parallelism, and other factors determined from the data dictionary [3].

Cost model is the most efficient plan to estimate the total cost of query plan based on prefer operator position inside the plan. The cost of the prefer operator which is used in our plan is said to be proportional to the tuples which are affected by the operator.

The expected size of the base table is used for estimating the cost of unions, inner joins and intersections. The values inside the base tables are dependent on the preferences. The cost related with score relations and the non-preference related cost includes the join order. If the non-preference related cost is kept fixed then the preference evaluation cost is given by,

$$cost(planq) \cong' \sum p\ op\ 2\ planq\ cost(p\_op) \quad [1]$$

2.2.2.1 ENUMERATING PLANS:

An enumeration plan pick an inexpensive execution plan for a given query by exploring the search space [5].Our main optimization goal is to reduce the cost plan where the prefer operator differ in the position of the plan. Let h be the height of a query plan and P be the number of prefer operators, then an exhaustive examination of all alternative plans would require $O(h^p)$ cost estimations, which is prohibitively expensive even for small values of h or P[1].

There are two approximate enumerating plan algorithms to derive an efficiency query plan. The first algorithm is the dynamic programming technique. Limited search of dynamic programming methods leads to an exponential number of calculations when large number of joins or preferences (complex queries) is given. The second algorithm is the Greedy algorithm. The Greedy algorithm is used to iteratively determine the position

of each operator in the query plan. Both the algorithm can produce an optimal plan.

2.2.2.2 DYNAMIC PROGRAMMING ALGORITHM

This technique borrows ideas from the plan enumeration method used in traditional query optimization where the goal is to produce an efficient join order [6]. The cost not only depends on the order of operators but also on the specific position of operators in the plan. This is because of two reasons.

The first reason is that the prefer operator which was applied on base table could be affected by the position, which constitute certain number of tuples. The second reason is that we have to include the score aggregation cost of the binary operators when calculating the total cost. Therefore it requires a complexity of O (h) calculations. By calculating each plan cost and the cost of prefer operator, we can maintain the position that minimizes the estimated cost.

At the i-th step the algorithm examines all groups of prefer operators with the size I and for each combination it maintains the set of optimal positions [1].All the combinations of i+1 operators are are generated by extending i-groups with an additional prefer operator in the next step. The algorithm is terminated after estimating the cost of P operators and it outputs the optimal position which was found.

We use the Dewey decimal Coding [11], which allows for answering ancestor queries in constant time [1].

2.2.2.3 Greedy Algorithm

The main aim of the Greedy Algorithm is to place the prefer operators in the ascending order of the minimum cost. The Greedy Algorithm picks the non-fixed prefer operators and evaluates its minimum cost to estimate its position in a node and it is done for every step. After this step the subsequent positions of the other prefer operators are updated in an iterative process until the final plan is obtained.

2.3 QUERY EXECUTION

Query Execution is the processing of preferential queries using various algorithms that apply on extended query plans. In addition to the query optimization goal we focus on minimizing the intermediate materialization to execute the query plan. We rely on combining operators and using the native query engine for operator execution to minimize intermediate materialization. Two algorithms which are used in Query Execution are Baseline and GBU.

2.3.1 Baseline:

In Baseline we perform a post order traversal of the extended query plan and each operator in the plan is executed directly. This algorithm is referred to as Bottom-Up (BU). In this plan each operator is executed individually. For any internal node n_i in the execution plan, BU maintains the results of executing n_i as a pair of temporary intermediate tables T_{Bi} and $T_{Si.}$ BU terminates when the operator attach to the root node n_r as been executed, the final results are produced by joining T_{Br} with T_{Sr}[1].

2.3.2 GBU:

Group Bottom Up is a more efficient execution algorithm compared to BU which is inefficient because it has to materialize every single node. The efficiency of GBU results in (i) Saving

I/Os by grouping operators (ii) Calling the native query engine for operator execution whenever possible. Advantage of combining consecutive prefer operators results in calling the stored procedure only once by handing over the execution to the database we improve the optimizations such as efficient access paths, physical operators' implementation and operator pipelining.

Here we have a set of rules to determine when an operator can be deferred:

1. The execution of a select or project can be postponed [1].
2. Consecutive prefer operators can be executed in a single operation [1].
3. The execution of a join operator can be postponed as long as at least one input score table is empty [1]

CONCLUSION:

This project proposed preferential query optimization which involves the construction of object oriented queries. This is an advantage over the two traditional approaches , plug-in and native approach. We give importance to the user's interests rather just satisfying the search constraints. This kind of similar query optimization is also possible in location mapping which was referred to as careDB. This can also be applied to other search streams like students searching for a journal, eBooks, etc..

3. REFERENCES:

[1] Anastasios Arvanitis and Georgia Koutrika. PrefDB: Supporting Preferences as First-Class Citizens in Relational Databases. In IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING.

[2] Basic principles of query optimization [online] available: http://www.dbis.informatik.hu-berlin.de/fileadmin/research/papers/freytagifip89.pdf accessed on 21-03-2014.

[3] Query optimization [online] available http://en.wikipedia.org/wiki/Query_optimization accessed on 19/03/2014

[4] Anastasios Arvanitis and Georgia Koutrika . PrefDB: Bringing Preferences Closer to the DBMS. In SIGMOD'12

[5] An Overview of Query Optimization in Relational Systems [online] available http://research.microsoft.com/pubs/76059/pods98-tutorial.pdf accessed on 21-03-2013.

[6] P.G. Selinger, M.M. Astrahan, D.D. Chamberlin, R.A. Lorie, and T.G. Price. Access path selection in a relational database management system. In SIGMOD, pages 23-34, 1979.

[7] V. Christophides, D.Plexousakis, M.scholl, and S.Tourtounis. On labeling schemes for the semantic web. In WWW, pages 544-555, 2003.

[8] F. Mokbel Justin J. Levandoski. Toward Context and Preference Aware Location Based Services.ACMmobiDE'09.