Query Optimization Using Case Base Reasoning With Replacement Policy

Pragya Shukla, Rohini Upadhyay

Abstract— A query optimizer is a core component of any Database Management System. Multiple approaches have been suggested which were based on framework of classical query evaluation procedures that heavily dependent on metadata. There are computational environments where metadata acquisition and support is very expensive. In this paper query optimization technique using case based reasoning (machine learning technique) for ubiquitous computing environment is deliberated. In this technique a new problem is solved by finding a similar past case, and reusing it in the new problem situation. As CBR is an approach to incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it immediately available for future problems which in return may create a bulky case base. Thus we were proposing a technique of dynamic deletion of irrelevant cases from case base. Through which system can detect the inappropriate case and replace them with new case in order to maintain size of case base.

Index Terms— Case-based reasoning, dynamicity, machine learning, metadata, similarity level, traditional query optimization techniques, ubiquitous computing environment.

1 INTRODUCTION

QUERY optimization is the process of selecting the most efficient query evaluation plan from among the option available for processing a given query. For appropriate performance of query processing it is excepted from system to construct a query evaluation plan that minimizes the cost of query evaluation which can done through query optimization. Query optimizers examine all expression which is equivalent to the new query and select the suitable cheapest plan. The area of query optimization in database field is very vast. It has been studied in a great variety of contexts and from many different angles which provides many miscellaneous solutions. Most of this approaches were based on classical query optimization were dependency on metadata is very high. Due to this dependency it may not work effectively in all type of computational environment. Ubiquitous environment is one of the appropriate examples where information technology become pervasive, embedded in environment, heterogeneous, sovereign and invisible to users. According to this vision network will be saturated by computation and wireless communication capacity which will be gracefully integrated with user.

Metadata for query processing attainment and preservation is not feasible in ubiquitous environment. Thus environment must provide a set of procedures to retrieve information from minimized resources. Additionally, resources used in this environment have physical limitations that restrict their suitable operations like distributed in different locations, limited storage and processing capability, power supply etc[6].

Query optimization is essential to expand the performance of query processing through which information can be accessed from any location and at any time. Antagonises in ubiquitous environment for query optimization are conferred in this paper. Here we propose a query optimization approach which will deal with these challenges and work effectively in lack of metadata [3],[6]. Also we address dynamicity management in proposed approach through replacement policy which provides a better framework for the resources operates in ubiquitous environment. Here we were trying to use machine learning techniques for optimization.

The remaining of this paper is organized as follows. Section 2 contains traditional query optimization techniques which were used up till now. Section 3 contains our optimization technique based on case based reasoning a machine learning technique. Section 4 contains dynamicity management techquie. Section 5 presents the conclusion of our work.

2 TRADITIONAL QUERY OPTIMIZATION TECHNIQUE

We provide an abstraction of traditional query optimization process. The modules that participate in the classical query evaluation process are the query parser, query optimizer, code generator and the query executor. The query parser is in charge to verify if the query is syntactically (well formed) and semantically correct. The output of this module is a tentative algebraic query tree. It is a sequence of algebraic operations (e.g. selection, projection and joins) that indicate the operations that must be performed on the data for solving the query. Then, a valid query must to be optimized; this is carried out in by the Query optimizer module. That estimates the best order to perform the operations included by the algebraic query tree and assigns to each algebraic operator an algorithm to execute it. The result is the execution plan. When the query is optimized, a codification of the execution tree must be performed by the code generator, to be executed by the last mod-

- *Pragya Shukla is withInstitute of Engineering and Technology – DAVV, Indore, India. E-mail:pragyashukla_iet@yahoo.com*
- *Rohini Upadhyay is student with Institute of Engineering and Technology – DAVV, Indore, India. E-mail:rohini1dec@gmail.com*

ule, the query executor. Finally, the data that solve the query is obtained.

A query optimizer is the component of a database system responsible to determine the optimum plan for a query execution. Fig.1. Illustrate the general query optimizer architecture. Modules that compose this architecture are grouped according to query optimization phases. Rewriter is the module that executes the rewriting phase. Generator, Selector, Cost estimator, Statistics estimator and Planner are the modules that execute the planning phase [2].

Rewriter: Rewrites from the original query, a set of equivalent queries by applying a set of transformations. Transformations depend on declarative characteristics of the original query. If the rewriting is beneficial then the original query is discarded.

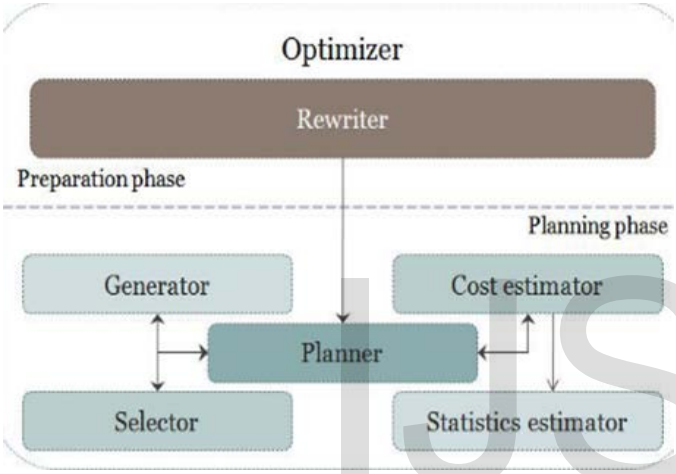

Fig.1. Query optimization architecture

Planner: Planners examine all the possible optional plans for query produced by rewriter and select the cheapest one to be generating the answer for the query. Planning phase consists of following sub modules:

- Generator: Generates a set of query execution plans to process a query. This execution plans are represented by algebraic expressions.
- Selector: Select a set of algorithms to execute each operator that is included in algebraic expressions related to a query execution plan proposed by the Generator module.
- Cost estimator: It specify calculus functions (in accordance a cost model) to evaluate execution plans to estimate their cost.
- Statistical estimator: It specifies a set of functions to estimate relations size, indexes and results, as well as, the distribution feculence of the values associated to an attribute include by a relation.
- Planner: It employs a search strategy to examine the execution plans space (with an associate cost) and select the less expensive in order to process a query and generate the result.

3 CBR TECHNIQUE FOR QUERY OPTIMIZATION

The query optimization technique that we propose is an adaptation of the general case-based reasoning process. This technique aims to solve the problem of lack of metadata, thus it is feasible to be applied in different execution environments which can't afford expensive acquisition and maintenance of metadata [11]. A ubiquitous computing environment is an appropriate example for it. Since case and problem are the main units of knowledge in this learning approach, we select useful knowledge for query optimization in order to instantiate both concepts. According to our approach, a case represents the knowledge related to the experience gained from the optimization and evaluation of a query. A problem represents a new query, that we call query problem, which is submitted in some application pertaining to the ubiquitous environment. The reasoning process that must be accomplished to optimize a query problem is elaborate in the following

- Retrieval: This step is based on a similarity function in order to perform a smart search to retrieve the most relevant cases to solve the query problem. Among these relevant cases, the one that minimizes the cost function of the problem is selected

- Reuse: Reusing step is related to the adaptation process of the execution plan involved by the case that resulted relevant to solve the new query.

- Review: Reviewing step consists in verify the query by means of its execution. During this step measures about performance as well as computational resources consumption are taken.

- Retention: Finally, in the retaining step, the problem and its solution are stored in the case base in form of a new case.

Since this approach is based in a try and learn principle, when a relevant case to solve a query problem is not founded in the case base, is necessary to propose a new solution [5].

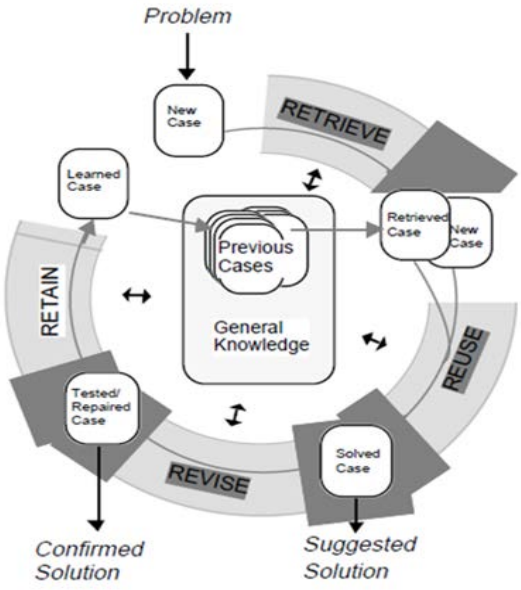

Fig.2. Case based reasoning cycle

A problem is composed by a query problem, the specification of the execution context by means of a set of measures that express the availability of different computational resources, and finally, the optimization objective that can be a single resource. Illustrating case based technique with example

Query 1

SELECT boarding_up, aligting_up **FROM** unique_name **WHERE** route_no between 1 and 10.

The query Q contain three clauses select from and where. As it will be the new query it will be store in case base in form of new learn case. Mainclass for this query is unique_name (from clause) and subclasses will be created on basis of where clause as shown in table below:

TABLE.1
Example Query 1

Category	Clause	Values
Main class	From	Unique_name
Subclass	Where route_no	between 1 and 10
Querytype	Select	boarding_up, aligting_up

As it store in case base as learned case now for any new query which is similar to this query we can reuse it from case base and readapt the values according to new query. Let's take example of new query

Query 2

SELECT boarding _dwn , aligting _dwn **FROM** unique_name **WHERE** route_no between 1 and 10.

The query Q' contain three clauses select, from and where. Now here we can apply case based reasoning technique and instead of generating a new execution plan for this query we will retrieve similar case from case base. First CBR technique start with retrieval process where new query Q' will be compared with already stored query Q and similarity will be checked.

TABLE. 2
Example Query 2

Category	Clause	Values
Main class	From	Unique_name
Subclass	Where route_no	route_no between 1 and 10
Querytype	Select	boarding_dwn, aligting_dwn

We were using a similarity function that is performed in following steps. First we find the membership of a query through an inter-class similarity function. Then, the most relevant case within the class must be retrieved by means of an intra-class similarity function [7],[9]. After finding the most relevant case, a detailed comparison between the clauses of the new query and the relevant query (the query included by the relevant case) is carried out. This determines a similarity level between the two queries [3],[8].

In Q and Q' first comparison will be done through inter class similarity function which will be perform on the basis of mainclass category. Here in this example we can see that Q and Q' were belonging to the same mainclass thus second step of similarity check will be perform i.e. is intra class similarity check to find out the most relevant case. Now comparison will be done on basis of sub classes based on where clause.

WHERE clause of Q = route_no between 1 and 10

WHERE clause of Q' = route_no between 1 and 10

As both queries containing similar where clause condition hence Q can be considered as most relevant case for Q'After completion of retrieval process next step comes is reuse where we have to adapt our retrieval case according to new query problem using similarity level. The similarity level between two queries indicates which clauses of the relevant query must be adapted. This adaptation can be performed only on Select and Where clauses. Reason behind this is that for Select clause, interesting attributes to be projected can vary and for Where clause, comparison operators or some values related to the variables can be modified. On the other hand, the From clause cannot be changed because the tables to be queried cannot be changed. Table 3 illustrates the diverse similarity levels

TABLE. 3
Similarity level between two queries

Similarity level	Similar clause	Different clause
4	Select,from,where	--------
3	From ,where	Select
2	From, select	Where
1	From	Select, where
0	----------	Select from where

In our example similarity level between both the queries Q and Q′ is 3, modification will performed only in select clause value which is

SELECT clause of Q = boarding_up, aligting_up

SELECT clause of Q′ = boarding_dwn, aligting_dwn.

Next is review step where proposed solution is verified through execution. Finally in the end retention of newly learned case is performed for future use. at the retaining step, a case is stored in case base, according to a defined classification. It is possible to know to which class pertains a case determining the similarity between the class of the query problem and the class in the case base.

4 REPLACEMENT POLICY FOR DYNAMICITY

As this query optimization technique is being proposed for ubiquitous type computing environment where acquisition of metadata is not feasible. in our proposal we were trying to use a machine learning technique i.e case based reasoning where query optimization is achieved thorough previously experienced situation, which has been captured and learned in a way that it can be reused in the solving of future problems. As if we were storing every new learn case in case base again there will be a possibility that size of case base will become huge and bulky. Here we were proposing a replacement policy for dynamic deletion of case from case base. So that the system should be able to detect those cases in its case base which will no longer relevant and thus delete them. This will maintain size of case base. Selection and deletion of irrelevant cases from case base performed on concept of least recently used case in which the case which has minimum references were will be deleted first.

When a new learned case is inserted into the case base, system sets the current cost case equal to zero. Thereafter, each time a user process references this case; it resets the current cost with increment. For all cases, the maximum value for the current cost is the number of time that case is referred [10].

When memory pressure exists, the system responds by removing irrelevant cases from the case base. To determine which plans to remove, the system repeatedly examines the state of each case and removes cases when their current cost is zero or minimum. A case with zero current cost is removed automatically when memory pressure exists. System repeatedly examines the cases until enough have been removed to satisfy memory requirements.

For example if lets suppose we have can accommodate only five cases at time than when a new case is arrive, than all case will examined and the case with minimum cost will be deleted.

5 CONCLUSION

We propose a query optimization technique that exploits case-based reasoning with replacement policy in order to improve query optimization in ubiquitous environments. Our approach deals with the challenge that the lack of metadata implies in this execution context. In our proposal we used a technique that is able to utilize the specific knowledge of previously experienced, concrete problem situations (cases). Case-based reasoning (CBR) is an approach to problem solving that emphasizes the role of prior experience during future problem solving (i.e., new problems are solved by reusing and if necessary adapting the solutions to similar problems that were solved in the past). CBR is memory based, thus reflecting human use of remembered problems and solutions as a starting point for new problem solving [4]. In addition, we propose a technique for detecting inappropriate cases from its case base and eliminate them so that system will achieve dynamic management as well. This will provide database of system manage able, relevant and compact.

REFERENCES

[1] Syedur Rahman1 , A. M. Ahsan Feroz2, Md. Kamruzzaman3 and Meherun Nesa Faruque4," Analyze Database Optimization Techniques", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.8, August 2010.

[2] Y. Ioannidis, "Query optimization," ACM Comput. Surv., vol. 28, no. 1, pp. 121–123, 1996.

[3] Lourdes Ang´elica Mart´ınez-Medina and Christophe Bibineau and Jose Luis Zechinelli-Martini, "Query optimization using case-based reasoning in ubiquitous environment"s in Mexican International Conference on Computer Science, 2009.

[4] L. D. Mantaras, R. McSherry, and et al, "Retrieval, reuse, revision and retention in case-based reasoning," Knowl. Eng. Rev., vol. 20, no. 3, pp. 215–240, 2005.

[5] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," AI Communications, vol. 7, no. 1, pp. 39–59, 1994.

[6] M. Franklin, "Challenges in ubiquitous data management," in Informatics - 10 Years Back. 10 Years Ahead., R. Wilhelm, Ed. Springer-Verlag, 2001, pp. 24–33.

[7] M. Gu, X. Tong, and A. Aamodt, "Comparing similarity calculation methods in conversational cbr," in In: Proceedings of the 2005 IEEE International Conference on Information Reuse and Integration, 2005, pp. 427–432.

[8] A. Tversky and I. Gati, "Studies of similarity," 1978

[9] R. Bergmann and A. Stahl, "Similarity measures for object oriented case representations," in In: Proceedings of the 4th European Workshop on Advances in Case-Based Reasoning B, B. Smyth and P. Cunningham, Eds. Springer Verlag, 1998.

[10] Mcrosoft,"Excutionplancachingandreuse,"2008,http://technet.microsoft.com/enus/library/bb545450.aspx.[Online].Available:http://technet.microsoft.com/enus/library/ms181055.aspx

[11] G. A. A. Deyand, P. Brown, and et al, "Towards a better understanding of context and context-awareness," in In: Proceedings of the 1st

international symposium on Handheld and Ubiquitous Computing (HUC 1999), September 1999.