Optimizing Complex Queries Using Case-Based Reasoning with Dynamicity Management

Pragya Shukla[*], Sakshi Mathur[**]

[*]Computer Science, IET-DAVV Indore
[**]Information Technology, IET-DAVV Indore

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 an optimization technique for complex queries using case based reasoning (machine learning technique) in 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 cases and replace them with new case in order to maintain size of case base.

Index Terms- Case-based reasoning, Dynamicity management, Metadata, Traditional query optimization techniques, Ubiquitous computing environment.

I. INTRODUCTION

Query optimization is vital to the performance of a database management system (DBMS). The main task of a query optimizer in a DBMS is to seek an efficient query execution plan for a given user query [5]. 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 [8]. Metadata for query processing attainment and preservation is not feasible in ubiquitous environment [4].

However, as database applications become more and more complex and database sizes become larger and larger, queries with a large number of lines occur more and more often in the real world. Existing techniques can no longer be used to optimize such queries. For an algorithm with an exponential complexity, it may take months or years to optimize a complex query [5].

Query optimization is essential to expand the performance of query processing through which information can be accessed from any location and at any time. Antagonizes 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] [8]. 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 ma-chine 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 technique. Section 5 presents the conclusion of our work.

II. TRADITIONAL QUERY OPTIMIZATION TECHNIQUE

We provide an abstraction of traditional query optimization process. The modules that participate in the Classical query evaluation processes 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 per-formed by the code generator, to be executed by the last module, 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 exe-cutes 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.

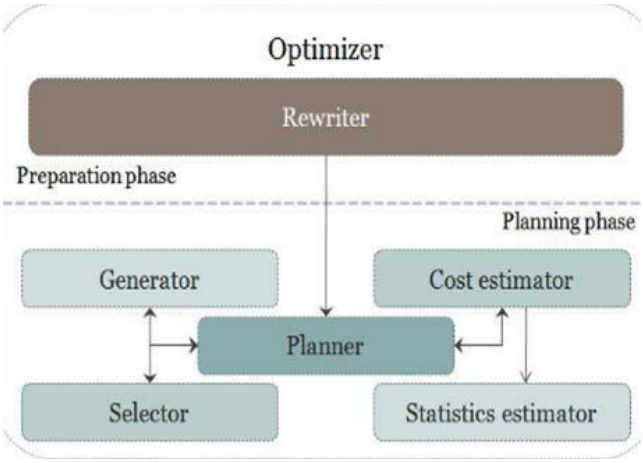

Figure 1: Query Optimization Architecture.

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.

As database technology is applied to more and more application domains, user queries become more and more complex. Query optimization for such queries becomes very challenging. Existing query optimization techniques either take too much time (e.g. dynamic programming) or yield a poor execution plan (e.g. simple heuristics). Although some randomization-based techniques can deal with this problem to a certain degree, the quality of the execution plan generated for a given query is still unsatisfactory because these techniques do not take the special characteristics of a complex query into consideration [5].

In this paper, we propose a new technique for optimizing complex queries based on exploiting similarity between them.

III. CBR TECHNIQUE FOR OPTIMIZING COMPLEX QUERIES

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 [13]. 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 elaborated 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

• *Readapt:* Readapting 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.

• *Retain:* 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 [7].

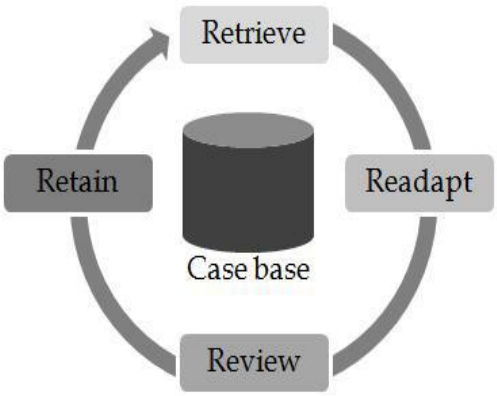

Figure 2 Case-Based Reasoning Cycle

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 re-sources, and finally, the optimization objective that can be a single resource. Illustrating case based technique with example.

Query 1
SELECT emp_name salary FROM Employee WHERE salary IN (SELECT sal FROM Dept WHERE sal>5000).

The query Q contain three clauses select from and where. And also contains a sub query having three clauses. As it will be the new query it will be store in case base in form of new learn case. Main class for this query is Employee (from clause) and subclasses will be created on basis of where clause, sub query main class is Dept and subclass is Where clause as shown in table below:

Table 1
Example query 1

Category	Clause	Values
Main class	From	Employee
Subclass	Where	Salary
Sub query main class	From	Dept
Sub query sub class	Where	Sal>5000
Query Type	Select	Emp_name, salary

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 emp_name salary FROM Employee WHERE salary IN (SELECT sal FROM Dept WHERE sal between 5000 and 10000).

The query Q' also contains same clauses. 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 com-pared with already stored query Q and similarity will be checked.

Table 2
Example query 2

Category	Clause	Values
Main class	From	Employee
Subclass	Where	Salary
Sub query main class	From	Dept
Sub query Subclass	Where	Sal between 5000 and 10000
Query Type	Select	Emp_name, salary

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], [10].

In Q and Q' first comparison will be done through inter class similarity function which will be perform on the basis of main class category. Here in this example we can see that Q and Q' were belonging to the same main class 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 and sub query classes. As Where clause contains a sub query, first sub query main class is checked and then its subclasses:

WHERE clause of Q = sal>5000
WHERE clause of Q' = sal between 5000 and 10000

As both queries containing similar where clause condition in the sub query 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 levels between two queries

Similarity level	Similar clause	Different clause
5	All clauses	----------
4	From, Where, From and where clause of Sub query	Select
3	From, Where, From clause of Sub query	Where clause of sub query, Select
2	From, Where	Select, Sub query
1	From	Where, Select, Sub query
0	----------	All clauses

In our example similarity level between both the queries Q and Q" is 4, modification will performed only in where clause value which is
WHERE clause of Q = sal>5000
WHERE clause of Q' = sal between 5000 and 10000

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.

IV. REPLACEMENT POLICY FOR DYNAMICITY MANAGEMENT

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 sys-tem 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 [11].

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, let's suppose we 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.

V. CONCLUSIONS AND FUTURE WORK

We propose a new query optimization technique that exploits case-based reasoning 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. We propose a technique that is based on the useful knowledge (resource consumption measures) obtained from previous query executions. In addition, we propose a technique that allows the configuration of the optimization objective according to the users and application requirements, even for each single query. Finally, we propose a pseudo-random query plan generator able to evaluate queries even if there is no corresponding case in the case base [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. We are also working on improving the knowledge acquisition and exploitation, for example by sharing case bases between nodes in a network or by reducing the granularity of cases to handle sub queries instead of full queries. We can also work on having and group by clause.

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 Computer Survey., 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 environments in Mexican International Conference on Computer Science, 2009.

[4] Pragya Shukla, Sakshi Mathur," A Framework To Subquery Optimization Using Case-Based Reasoning", International Journal of Computer Applications (0975 – 8887) Volume 88 – No.2, February 2014.

[5] Qiang Zhu1,Yingying Tao1, Calisto Zuzarte2, "Optimizing complex queries based on similarities of subqueries", Springer-Verlag London Ltd. © 2005 Knowledge and Information Systems (2005) 8: 350–373.

[6] 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.

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

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

[9] 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.

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

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

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

[13] 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.

AUTHORS

First Author – Pragya Shukla, Computer Science, IET-DAVV Indore, She has done her Bachelor in engineering in Computer Science, Masters in engineering in Computer engineering and Ph.d. in Computer engineering. Currently she is working as an associate professor in IET, DAVV Indore with a work experience of 16 years. She has published 20 research papers. Her areas of interests are artificial intelligence, data mining, software computing etc.

Second Author – Sakshi Mathur, Information Technology, IET-DAVV Indore, She has done her Bachelor in engineering in Information Technology. Currently she is pursuing Masters in

engineering in Information Technology in IET DAVV, Indore. She has published 1 research paper and her areas of interests are Operating system, databases, data structures etc.