Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/070515

Rule Evaluation Algorithm for Semantic Query Optimisation

Ayla Sayli[*] *and Armagan Elibol*

Department of Mathematical Engineering, Yildiz Technical University, Davutpasa Campus,34210, Istanbul, Turkiye

Received: 8 Jan. 2013, Revised: 11 May. 2013, Accepted: 12 May. 2013
Published online: 1 Sep. 2013

Abstract: Semantic Query Optimisation (SQO) in Relational Database Management Systems (RDMSs) is a query optimisation approach which uses rules learned from past queries in order to execute new queries more intelligently without accessing database, whenever possible. The approach is composed of several components: Query Representation, Query Optimisation, Automatic Rule Derivation and Rule Maintenance. This paper focused on the query optimisation component. In RDMSs, during the traditional SQO, different alternative queries of a given query can be constructed using matching rule(s) from the rule set, and then its optimiser selects one of the alternatives as an optimum query which will give the same answer set but it can be executed faster than the original query. One of the main problems occurs during this process is to have many matched rules e.g., if the number of the rules is N, the number of the alternative queries is $2^N - 1$. The construction and the optimisation of these alternatives also take time in addition to the execution of the query. In order to overcome this problem, in this paper we propose a new Rule Evaluation Algorithm. The main goal of the algorithm is to evaluate matching rule(s) and select useful/promising rules. And then use selected rules to construct an optimum query. The algorithm can answer the question of the utility of rules in the query optimisation. The system of the approach based on the algorithm has been implemented and its computational results are given. The experimental results show that the algorithm can trim the number of the rules significantly.

Keywords: Rule Evaluation Algorithm, Cost Estimation, Semantic Query Optimisation, Relational Database Management Systems.

1. Introduction

Semantic Query Optimisation (SQO) can be seen as an approach that learns and uses the rules during the query optimisation in database systems. Although the approach was researched as the semantic reasoning by [1], it has been announced by [2] and the complete system of the approach was developed mainly in the late eighties [3–5]. The main advantage of SQO is to derive rules and then use them to reduce the execution time of future queries. The use of rules makes the optimiser more intelligent. The approach has several components: Query Representation, Automatic Rule Derivation, Query Optimisation and Rule Maintenance.

The first component of the approach is to represent a query in a query language in order to express it in a way that can be understood by the chosen database environment. Many Database Systems use Structural Query Language for this purpose.

The second component is the automatic rule derivation to learn new rules based on the conditions of the given query and the query answer from the database. The structure of rules needs to be explained first. The majority of researchers worked on the approach use a Simple Rule form that has a condition on the left-hand-side (called 'Antecedent') and a condition on the right-hand-side (called 'Consequent'). This type of rule cannot contain more than one condition on either side. Other rules can be seen as complex rules which can have more conditions on one side using AND/OR. Somehow the complex rules can be expressed by [6] but using them in the query optimisation becomes too difficult, also the rule maintenance becomes problematical [2, 4, 5, 7]. The condition(s) of the query is taken as candidate antecedent(s) of new rules. For each candidate antecedent, the answer set of the query is searched to find out a consequent if possible. A rule could be formed as $x \oplus x_1 \rightarrow y \otimes y_1$ where x_1 and y_1 are constant values of x and y attributes of a relation; $\oplus$ and $\otimes$ are one of comparison operators such as $<$, $<=$, $>$, $>=$, $=$, $!=$. When the antecedent is satisfied, then the consequent has to be true. This representation of the rule can be named as a

[*] Corresponding author e-mail: sayli@yildiz.edu.tr

Simple Rule that is used by the well-known techniques for automatic rule derivation [1–3, 5, 8, 9]. Despite the advantages of the SQO approach, the automatic rule derivation takes a considerably long time and the time increases according to the database size. Another problem is that the total number of rules in the rule set may increase rapidly. For the derivation, several researchers in the database systems have also suggested the use of Statistics and Probability in databases, especially on the functional dependencies but not data dependencies. On data dependencies, several researches on the data-driven systems [2,3,10–16] had mentioned it as well but how and what to do in database-based systems remain to be seen. Although the rule derivation is one of the most important components of the SQO approach, it is not within the scope of this paper.

The third component is to optimise a given query. The query optimiser constructs alternative queries using the rule set and then selects the best alternative as an optimum query. However, several queries might be given in their best forms that their constructed optimum queries could not give the better execution in terms of time. This means that there is not always time saving in the query optimisation and processing. In most situations, the execution time of queries could be reduced, especially when the consequent of a rule contains the primary key and indexes [2,5,8,10,11]. Assuming that N is the number of matching rules of the given query, the number of alternative queries can be found from $(2^N - 1)$. The significant work has been done in the EXODUS system by [3] in where cumulative arithmetic average method and cumulative geometric average method were used. Although several researches worked on the limitation of the rule sets used for the query processing to improve the quality of the rules in the rule set [9,12,13,17], it is still not completely solved problem..

The last component is used to keep the rules up to date for changes in the databases because these rules represent a time-dependent property of a database at a particular database state. When updates are made on the database, these rules may become invalid. Therefore the derived rules need to be maintained to be consistent with the current database state [18].

In this paper, we propose a new Rule Evaluation Algorithm (REA) to solve the problem of the utilising rules in the database query processing, mentioned in the third component above. The algorithm checks the consequent(s) of the matching rules if it is promising by the use of the rule cost estimation, it is used to construct the optimum query. Otherwise it is avoided. This limits the number of the matching rules in the query processing of the SQO approach.

The system of the traditional SQO approach is implemented and it is explained in Section 2, the concept of Semantically Equivalent Queries is especially described in Section 2.1. The proposed SQO system with the REA is introduced in Section 3. The computational results are given in Section 4. Finally conclusions can be found in Section 5.

2. Traditional SQO Approach

As it is mentioned before that the traditional SQO approach has several components. Figure 1 shows how these components are processed and in which order.

In the first step, the original query is entered into the SQO approach. In the second step, the query is represented in SQL. In the third step, a check is done to examine if the query is a SELECT statement in the View Definition Language (VDL). If it is, then the statement is taken into the process of the query optimisation in the fifth step. If the statement does not contain the VDL statement, in the fourth step, another check is done to see if the query is an INSERT/UPDATE/DELETE statement in Data Manipulation Language (DML). If it is, then the statement is taken into the process of the Rule Maintenance Manager. It is important to realise that a database is dynamic. This causes a serious situation that each DML statement changes the database, so the rule set has to be updated either before or after applying these statements in order to ensure the accuracy of the rule set. Moreover, this component can be seen as two parts; Rule Detection and Rule Maintenance. Although this component is very important, it is not within the scope of this paper. If the statement is also not the DML statement, the process goes back to the user to get a new request.

In the fifth step, the process of the component of the query optimisation (represented with the cut-line bolder in Figure 1) starts, and in which the condition(s) of the query are taken into account to determine if the rule set has a matching rule or not. If the matching rule(s) are found, then the query transformation and the query optimisation are triggered to construct all alternative queries. After the construction, the optimiser then selects one query among the alternatives as an optimum query that can be executed faster than the others while yielding the same answer set (The process of this component is especially described in Figure 2). Finally the answer set of the optimum query is given to the user.

After the query is executed by the optimiser, the sixth step begins for the automatic rule derivation component which takes the unmatched condition(s) of the original query to derive possible and acceptable new rules using the answer set of the query. If a new rule can be derived, it is added to the rule set automatically.

In addition to the execution of the optimum query in the traditional way of the SQO approach, three special circumstances may occur: Query Refutation, Query Answering and Construction of Optimum Query using Matching Rules. They can be seen in Figure 2. In the case of Query Refutation, the conditions of the original query conflict with the consequent(s) of matching rules in the rule set. In case of Query Answering, the answer of a query can be found from the consequent of the matching rules. These special cases are the most profitable aspects of the SQO approach, with substantial time savings (%99) reported in [17].

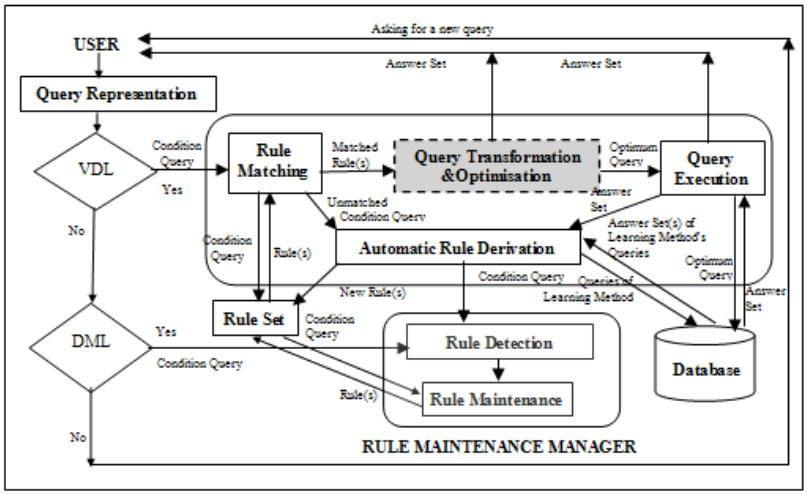

Figure 1 Traditional SQO System.

Table 1 Two example rules in Department table.

$Rule_1$:	Lecturers in Mathematical Engineering Department earn greater than 30K
	DNAME='Mathematical Engineering' → salary >30K
$Rule_2$:	Department code of Mathematical Engineering Department is 'ME'
	DNAME='Mathematical Engineering' → DCode ='ME'

If the query answer could not be found, the database query processing continues to construct alternative queries of the original query in the third case. The number of the alternative queries can be estimated from $(2^N - 1)$ where N is the number of the distinct consequents of matching rules. According to the semantic query transformation in Section 2.1, alternative queries can be structured. The next stage is to compare these alternative queries to select which can be the optimum query. This selection is suggested to be made according to the costs of alternatives. These costs can be estimated differently. For example, Siegel et al. [5] suggested using statistics and probability to establish a cost model and Graefe and Dewitt [3] proposed that these costs could be estimated from past experiments. After determining the optimum query, it is executed to find the answer set, instead of executing the original query.

Moreover, when the number of the matching rules increases, the problem of the processing in the construction of the alternative queries gets difficult to choose an optimum query between $(2^N - 1)$ alternative queries.

2.1. Semantic Query Transformation

Another important concept of the SQO approach is the semantic query transformation which provides "Semantically Equivalent Queries". This transformation was described as the heart of the SQO approach in [2]. The main idea of the transformation is to use rules to transform a given query into alternative queries that have the same answer set as the given query but constructed semantically. For example, assume that two rules exist in Department table such as given in Table 1. If the names of all lecturers in the Department of Mathematical Engineering, the original query can be represented in SQL as in Table 2. Using $Rule_1$ and $Rule_2$, alternative queries can be built as $Query_1$, $Query_2$, and $Query_3$ that are semantically equal to the original:

Table 2 Original and semantically equivalent queries.

Query:	SELECT name FROM dept WHERE DName = 'Mathematical Engineering';
$Query_1$:	SELECT name FROM dept WHERE DName='Mathematical Engineering' and Salary > 30K;
$Query_2$:	SELECT name FROM dept WHERE DNAME='Mathematical Engineering' and DCode='ME';
$Query_3$:	SELECT name FROM dept WHERE DNAME='Mathematical Engineering' and Salary > 30K and DCode='ME';

The next problem is how to select one of these alternative queries as the optimum query to execute it instead of the original query. The selection of the optimum query is suggested to be made according to the costs of alternatives.

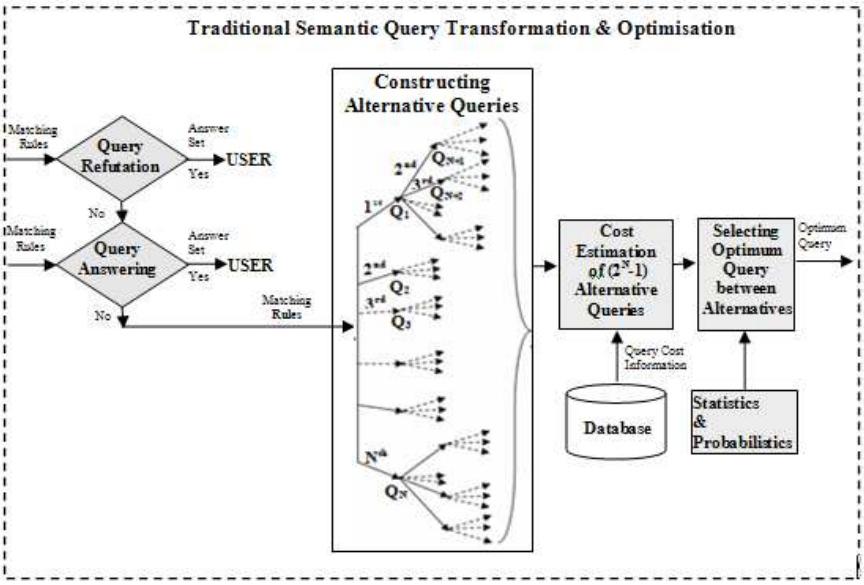

Figure 2 Traditional Semantic Query Transformation & Optimisation of the SQO approach.

These costs can be estimated differently which is given in the next section.

3. A Rule Evaluation Algorithm (REA) for SQO Approach

The total number of rules in the rule set increases very fast related to the rule derivation method where new rules would be learnt for each query. This raises an issue of "Whether a matching rule of a given query is worth being used in the optimisation or not". For this purpose, the REA is implemented to select the rules which are useful for the query optimisation. The REA eliminates the kind of rules that can be less useful. Thus, it reduces all possible alternative queries. For cost estimation in Figure 2, the algorithm in Table 3 is proposed. This REA is placed into the query transformation and optimisation, just before constructing the alternative queries. It is shown in Figure 3 where the implemented system of the Semantic Query Transformation and Optimisation with the REA starts after the matching rules are found. Firstly, the query refutation is controlled. If the conditions of the original query conflict with the consequent(s) of matching rules in the rule set, the answer of the query is given as NULL to the user, without any access to the database. Otherwise, secondly the query answering is controlled. If the answer of the query can be found from the consequent of matching rules in the rule set, the answer of the query is given to the user, without any access to the database.

If the answer of the given query could not find from the query refutation and the query answering, thirdly the cost estimation based on the numbers of instances identified by the antecedent and the consequent is used to calculate the costs of the matching rules. Moreover, the number of instances can be saved when the rule was initially derived in the automatic rule derivation process. It then continues to estimate the cost ratios of the rules. The main elements of the REA are the cost ratio and the idea of that if the cost of the consequent of the rule is cheaper to execute in comparison with the antecedent, this rule should be selected to construct the optimum query.

Cost estimation model of antecedents and consequents is given in Section 3.1 and the determination of cost ratio(s) of matching rule(s) is described in Section 3.2.

3.1. Cost Estimation Model

Assuming that the number of records of an antecedent or a consequent is R and the total number of disk blocks is B, the probability of a record being in a given block is $\frac{1}{B}$. The probability of a record not being there is $(1-\frac{1}{B})^R$. From this, it can be seen that the probability of a retrieved block is $(1-(1-\frac{1}{B})^R)$. The approximate number of disk blocks retrieved for the R records, A, can be found by multiplying the last expression by the total number of B blocks which is formulated in Function 1:

$$A = B \times (1-(1-\frac{1}{B})^R) \quad (1)$$

This assumes a random distribution of records across the relation space and 100% block packing density. If the condition contains an index attribute, the optimiser only searches for the number of the records located in A blocks. Our cost approximation model may then be extended to

Table 3 Cost Estimation Algorithm.

RULE EVALUATION ALGORITHM

Input :	Matching Rules $(P \rightarrow Q)$, R is the number of total records of an antecedent or a consequent, B is the total number of disk blocks, N is the number of records per block L is the length of the attribute of R in bytes
Output:	Optimum Query
Step 1 :	Check the Query Refutation exists, the answer of the query is given as NULL to the user and then go to Step 6. Otherwise, go to Step 2.
Step 2 :	Check the Query Answering exists, the answer of the query is given as the consequent value to the user, and then go to Step 6. Otherwise, go to Step 3.
Step 3 :	Take each matching rule: Estimate the cost of antecedent, CostP, using Function 1, 2 and 3 (described in Section 3.1), Estimate the cost of consequent, CostQ, using Function 1, 2 and 3 (described in Section 3.1), Determine a Cost Ratio of the rule, using Function 4 (described in Section 3.2), If the cost ratio of a rule is less then or equal to zero, this rule is ignored. Otherwise, add to an Evaluated-Rule Set (ERS) of the matching rules.
Step 4 :	Construct the Optimum Query using the consequents of rules from the ERS
Step 5 :	Send the Optimum Query to execute instead of the original query
Step 6 :	END

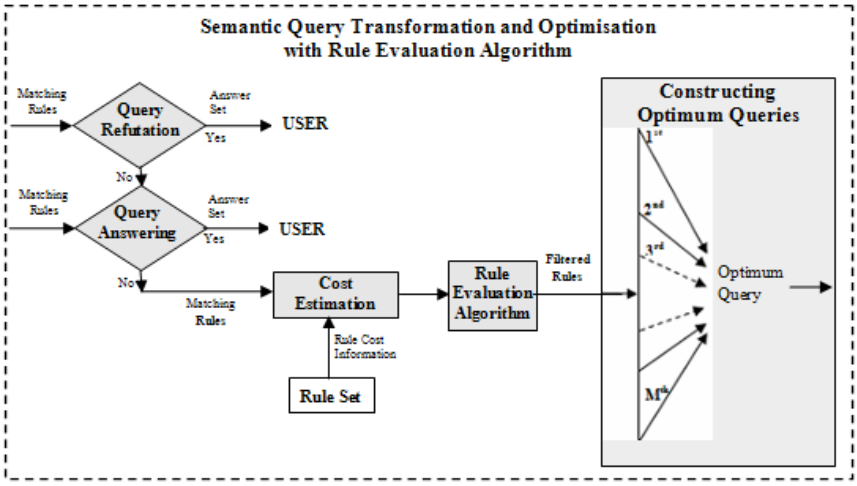

Figure 3 New Semantic Query Transformation & Optimisation with Rule Evaluation Algorithm.

predict the number of byte comparisons as shown in Function 2 below:

$$Cost = A \times N \times L \tag{2}$$

Where N is the number of records per block and L is the length of the attribute of the antecedent or the consequent in bytes. If the attribute of the antecedent or the consequent does not contain an index attribute, the optimiser has no information about the locations of A blocks. Therefore, it searches a number of blocks which should be between A and B. The cost approximation model may then be extended to predict the number of byte comparisons as shown in Function 3 below:

$$Cost = \left[\frac{A \times (B+1)}{A+1}\right] \times N \times L \tag{3}$$

Function 2 or 3 can be used to evaluate the cost of an antecedent or a consequent of a rule. The next step is to define the cost ratio of a matched rule for the Algorithm.

3.2. Cost Ratio Determination of a Rule

For each matching rule, firstly the cost of the antecedent and the consequent of the rule must be estimated. After the cost estimation of the antecedent and the consequent of the rule, they are then used to calculate a 'Cost Ratio' of this rule. Assuming this rule is represented as $P \rightarrow Q$, CostP is the comparison cost to find the records identified by the antecedent, and CostQ is the comparison cost to find the records identified by the consequent. These costs can be found by Function 2 or 3. A cost ratio of this rule is can be formalised below:

$$CostRatio = \frac{CostP - CostQ}{CostP} \quad (4)$$

This cost ratio of the Algorithm is used to compare the use of Q against P in order to determine the effectiveness of a rule. It should be noted that the cost ratio can be negative as well as positive. If the cost ratio of a rule is less then or equal to zero, this rule is ignored to construct the optimum query by the REA. Otherwise, this rule is used for the optimum query. The selected rule(s) by the algorithm can be seeing as promising rules for the optimisation in order to construct the optimum query. This reduces the number of the matching rules and the number of the alternative queries.

Example: Assuming that the record length is 40 bytes, a disk block can hold 12 records and the number of disk blocks B is 20. The sample relation is the DEPARTMENT relation which has 240 records and 5 different attributes (Dcode# char(4), Dname char(12), Project integer, Lecturer char(4), Location char(15)), 'Project' is an indexed attribute of the relation and initially the system has 2 rules in the rules set after the first rule derivation process. These rules are shown as following:

$Rule_1$: DCode='MATH'→Dname='Mathematics'

$Rule_2$: DCode='MATH'→Lecturer='AE' Firstly it is necessary to estimate the cost ratios of the rules. Table 4 shows the results of the cost estimation for each rule.

Table 4 Cost Computation Results

Rule P→Q	P Antecedent				Q Consequent			
	R	**L**	**A**	**CostP**	**R**	**L**	**A**	**CostQ**
$Rule_1$	30	4	15.7	947.66	40	10	17.42	2383.26
$Rule_2$	30	4	15.7	947.66	80	2	19.65	479.6

It can be seen that the antecedent and the consequent of $Rule_1$ are not indexed and the REA proceeds to the cost ratios as follows: For $Rule_1$, the Cost Ratio can be calculated from Function 4:

$$CostRatio = \frac{CostP - CostQ}{CostP} = -1.51 < 0 \quad (5)$$

This rule is ignored because the cost of its antecedent is less than the cost of the consequent. In other words, the execution of the consequent cannot be faster than the execution of its antecedent.

For $Rule_2$, the Cost Ratio can be calculated from Function 4:

$$CostRatio = \frac{CostP - CostQ}{CostP} = 0.49 > 0 \quad (6)$$

This rule is used to construct the optimum query because the cost of its antecedent is higher than the cost of its consequent. For $Rule_2$, it can be said that it is promising to reduce the execution time of future queries. This kind of the algorithms is necessary to have in the query processing to identify whether a rule is 'Useful' or not in terms of the contribution of its consequent.

4. Experimental Results

Computational results are presented in order to show the different aspects of the SQO approach. 'EPISODE' is used as a sample database that its schema is given in Table 5. This relation in our database has 41046 instances. It was created by The Department of Health (England), Hospital Episode Statistics - 1998/2003:

Table 5 Schema of 'EPISODE' relation

EPISODE		(id#, startyear, finishyear, epcode, episodes, admissions, sex, emergency, waiting_list, mean_waiting, median_waiting, mean_length, mean_age, age_0_14, age_15_59, age_60_74, age_75, day_case, bed_days)
Index Attribute(s)	:	'startyear', 'finishyear', 'epcode'

The main reason of using this database is to have difficulties to find a real database due to the confidentiality of data security. EPISODE is free and available on internet [1]. If other databases are provided, the implemented system with the algorithm can be run easily.

The rule derivation methods are not within the scope of the paper but it is necessary to learn rules for the REA in the query optimisation. Therefore, the rule derivation method of Data Driven Based Learning in [7] was used and 9115 rules were learnt in total.

Although the following experiments were conducted for several thousand queries on the relation and were based on many tests in order to analyse the time saving using our system of the SQO Approach with/without the REA, we selected the 855 queries as a prototype set according to their special features.

For the query refutation and the query answering, as mentioned before that these special cases are the most profitable aspects of the SQO approach, with substantial time savings (99%) which also were measured in our tests. The main reason of this high saving was that there was no need to access the database, only the rule set was searched linearly.

[1] http://www.dh.gov.uk/en/publicationsandstatistics/statistics/hospitalepisodestatistics/index.htm

The other experiments were done especially in two cases which are listed as follows:

1.Execution Times of Original Queries & Optimum Queries with the REA
2.Execution Times of Original Queries & Optimum Queries without the REA

For each case, the 855 queries were executed 10 different times, on the EPISODE relation and the collected times were averaged. Computational results of the cases are given in Section 4.1 and Section 4.2.

4.1. Execution Times of Original and Optimum Queries with REA

This experiment was done for the execution times of original queries and optimum queries constructed by the Algorithm in the SQO approach, and the computational results are given in Figure 4 where "Original Query" represents each given query on a commercial database and was executed without the SQO approach (in other words, without the use of rules) and "Optimum Query 1" represents each optimum query constructed by the Algorithm in the SQO approach.

It can be seen from Figure 4 that the execution times of original queries are higher than the execution times of optimum queries in which the evaluated-rules were used. For the executions of 855 queries, the time taken by the SQO approach with the Algorithm was 83 seconds. The percentage of saving time of the optimum query execution compared to the original query execution was found as 25.62% on average.

There are different behaviours of optimum queries that can be noticed from the graph. These behaviours are listed below:

i) Some of the optimum query execution approximately was highly lower than their originals (roughly like from Query 5 to Query 90): There are specified reasons for the kind of optimum queries which were
   (a)The consequent(s) of the evaluated-rules for these queries could have a better selectivity. For example, consequent attribute(s) could contain the index/key structure or consequent condition(s) could be based on the comparison operation as $=$
   (b)The cost ratios of the rule(s) could select useful rules that the optimum queries could give the same answer set but faster than the originals.
   (c)The original query could be given in its worst form.
ii) Some of the optimum query execution approximately was closer to their originals (roughly from Query 370 to 490): There are three specified reasons for these optimum queries which were:
   (a)The number of evaluated-rules could be few.
   (b)The consequents of the evaluated-rules for these queries could not have a better selectivity. For example, consequent attribute(s) could not contain an index structure, or consequent condition(s) could be based on the comparison operation as $<$, $<=$, $>$, $>=$ or $!=$).
   (c)The original query could be given in its best form which could be nearby the optimum form.

Another point is that while the size of the database and the number of user queries in real world applications increase rapidly (especially internet applications based on the database management systems by multi-users), this causes the enlargement of the number of rules in the rule set and therefore the significance of the Algorithm is well-proportioned. It can be finalised to say that the SQO approach with the Algorithm is sufficient and effective.

4.2. Execution Times of Original and Optimum Queries without REA

This experiment was done for the execution times of original queries and optimum queries with the use of all rules in the SQO approach (without the Algorithm), and the computational results are given in Figure 5 where "Original Query" represents each given query on a commercial database and was executed without the SQO approach (in other words, without the use of rules) and "Optimum Query 2" represents each optimum query with the use of all rules in the SQO approach (without the REA).

It can be seen from Figure 5 that the execution times of original queries are higher than the execution times of optimum queries. From the executions of 855 queries, the time taken by the SQO approach without the REA was 89 seconds. The percentage of saving time compared to the original query execution was 20.82% on average. It is also possible to make similar interpretations as Figure 4, especially about the profit of the use of rules.

However it can be noticed from the Figure 5 that some of original queries were executed faster than the optimum queries. Although the given reasons of (ii) in Section 4.1, these queries could not be executed better than their optimums in Figure 4. In other words, this also proves that all matching rule(s) from the 9115 rules were not useful for their optimum queries.

When Figure 4 and 5 are compared, the difference between the percentages of saving times is 4.8% and this makes the SQO approach with the Algorithm worthwhile, especially it is mentioned before that the number of rules in the rule set increases rapidly for new queries and these rules has to be limited in order to have an optimum query execution process.

In general, it is necessary to have a mechanism to select the promising rules during the query transformation and optimisation component of the SQO approach and it is shown that having many matching rules for a given query does not mean that all of the matching rules are useful. The algorithm can be a way to evaluate the rules efficiently and it is easily structured in the query transformation and optimisation process.

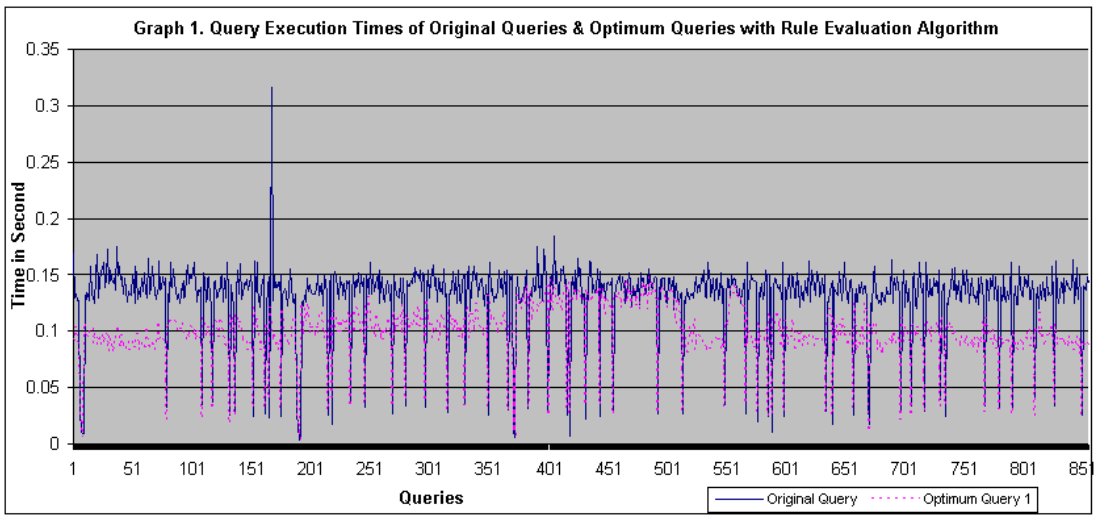

Figure 4 Query Execution Times with REA.

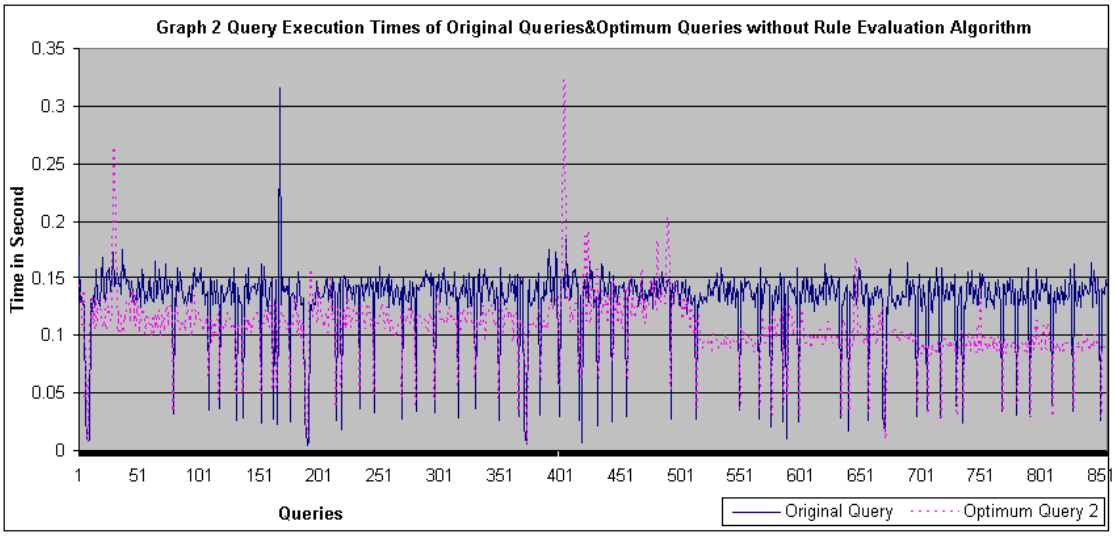

Figure 5 Query Execution Times without REA.

5. Conclusions

In this paper, the system of the SQO approach with the REA of selecting useful rules is given. Its algorithm, REA is introduced and it is shown that it can be easily run for any given database. The REA is dynamic and flexible. Computational results of the system of the query transformation and the optimisation with and without the REA are given to explain how it can be used to select the useful rules quickly and effectively.

In the future, our research would be extended to the components of the SQO approach such as the query optimisation [19], the automatic rule derivation [20] and the rule maintenance, especially for the methods of Statistics and Knowledge Discovery & Design.

References

[1] M. Hammer and D. J. Mcleod, Semantic Integrity in A Relational Data Base System, In Proceeding of the 1st International Conference on VLDB, 25–47 (1975).

[2] J. J. King, QUIST: A System For Semantic Query Optimisation in Relational Databases, In Proceeding of the 7th VLDB Conference, 510–517 (1981).

[3] G. Graefe, D. Dewitt, The EXODUS Optimiser Generator, In Proceedings of the ACM-SIGMOD Conference on Management of Data, San Francisco, USA, 160–171 (1987).
[4] U. S. Chakravarthy, J. Grant and J. Minker, Logic-based Approach to Semantic Query Optimisation, ACM Transactions on Database Systems, **15**, 162–207 (1990).
[5] M. D. Siegel, E. Sciore and S. Salvater, A Method for Automatic Rule Derivation to Support Semantic Query Optimisation, ACM Transactions on Database Systems, **17**, 563–600 (1992).
[6] D. Genest and M. Chein, A Content- Search Information Retrieval Process based on Conceptual Graph, Knowledge and Information System, **8**, 292–309 (2005).
[7] I. K. Ibrahim, W. Miniwalter and E. Weippl, Logic-based Approach to Semantic Query Transformation for Knowledge Management Applications, Proceeding of the International Conference on Knowledge Management, Berlin, Springer (2002).
[8] C. A. Knoblock, K. Lerman, S. Minton and I. Muslea, Accurately and Reliably Extracting Data from Web: a Machine Learning Approach Intelligent Exploration of the Web, Physica-Verlag GmbH Heidelberg, Germany, 275-287 (2003).
[9] S. Shekhar, B. Hamidzadeh, A. Kohli and M. F. Coyle, Learning Transformation Rules for Semantic Query Optimization: A Data-Driven Approach, IEEE Transactions on Knowledge and Data Engineering, **5**, 949–964 (1993).
[10] F. Marchi, S. Lopes and J. M. Petit, Efficient Algorithms for Mining Inclusion Dependencies, In Proceeding of International Conference on Extending Database Technology, 464–476 (2002).
[11] V. Pudi and J. R. Haritsa, Generalized Closed Item Sets for Association Rule Mining, Proceeding of 19th International Conference on Data Engineering, 714–716 (2003).
[12] K. C. Chan and A. K. C. Wong, A Statistical Technique for Extracting Classificatory Knowledge Form Databases, Knowledge Discovery in Databases, Ed. AAAI Press, 107–123 (1991).
[13] G. Piatetsky-Shapiro and C. Matheus, Measuring Data Dependencies in Large Databases, Knowledge Discovery in Databases Workshop, 162–173 (1993).
[14] I. F. Imam, R. S. Michalski and L. Kerschberg, Discovering Attribute Dependence in Database by Integrating Symbolic Learning and Statistical Analysis Techniques, Knowledge Discovery in Databases Workshop, 264–275 (1993).
[15] S. G. Lee, L. J. Chun and T. Lee, Identifying Relevant Constraints for Semantic Query Optimisation, Information and Software Technology, 899–914 (2000).
[16] J. Bakus and M. S. Kamel, Higher Order Feature Selection for Text Classification, Knowledge and Information System, **9**, 468–491 (2006).
[17] A. Sayli and B. Lowden, A Fast Transformation Method to Semantic Query Optimisation, International Database Engineering and Applications Symposium, Montreal, Canada, 319–326 (1997).
[18] N. S. Ishakbeyoglu and Z. M. Ozsoyoglu, On the Maintenance of Implication Integrity Constraints, In the Proceedings of 4th International Conference on Database and Expert System Applications, 221–232 (1993).
[19] Y. Tzitzikas and A. Analyti, Mining the Meaningful Term Conjunctions from Materialised faceted Taxonomies: Algorithms and Complexity, Knowledge and Information System, **9**, 430–467 (2006).
[20] A. Sayli and A. O. Uysal, A Dynamic Self-Learning Method for Semantic Query Optimisation, International Journal of Technology, Policy and Management, **8**, 126–147 (2008).

Ayla Sayli is an associate professor of Mathematical Engineering Department at the Yildiz Technical University, Istanbul, Turkey. She holds a PhD in Computer Science from the University of Essex, Colchester, UK. Her research interests are in the areas of database systems and data analysis as well as software engineering and algorithms.

Armagan Elibol has a PhD in Computer Engineering from the University of Girona in 2011. He was supported by the Autonomous Catalan Government during his PhD. Currently he is a full time researcher at the Department of Mathematical Engineering at Yildiz Technical University (Turkiye). His research interests are in the area of Computer Vision, Robotics and Computational Science, with a special focus on the problems arise in underwater vision and seabed mapping. He has attended several national and international conferences and has published several papers in different conferences and journals.