DATABASE QUERY OPTIMIZATION: NEW APPROACH TO GENETIC PROGRAMMING

Lela Tsitashvili[1], Badri Meparishvili[2], Gulnara Janelidze[2]
1. Akhaltsikhe State Educational University, Georgia
2. Georgian Technical University

Summary

This paper discusses one of the actual problem of database management which is called Query Optimization. In the process of implementation every possible plan of binary tree makes the so-called situation space that is determined by dimension appropriate to *n!* Factorial in case of *n* number of table. Contamperary database management systems, for instance SQL Server may be consisted of about 32 tables in one query. In this case the variation if the query are equal of about 2.6 *10^{35} plans and selecting the optimal one causes time problem, even with superfast computers. The major function of the Query Optimizer existing in the database management system is searching the best plan that is a quit difficult task proceeded from the above-mentioned dimension. Thus, the needed to develop new and more effective methods for the Query optimization becomes evident. The paper discusses one modified algorithm of Genetic Programming, which carries out the selection of a combination of Relational Algebra operations and finds the optimal solutions very fast.

Keywords*:* Query processing. Query optimization. Evolutionary algorithms. Genetic programming.

1. Introduction

An SQL query is first translated into an equivalent extended relational algebra expression-represented as a query tree data structure-that is then optimized. In relational DBMS query optimization is based on formulation of a query and convert it into an algebraic query evaluation tree. Query tree is a tree data that corresponds to a relational algebra expression. It represents the input relations of the query as *leaf nodes* of the tree, and represents the relational algebra operations as *internal nodes*. The transformation of query purports the multiple enumeration of possibilities of free graph-based query structure (figure 1).

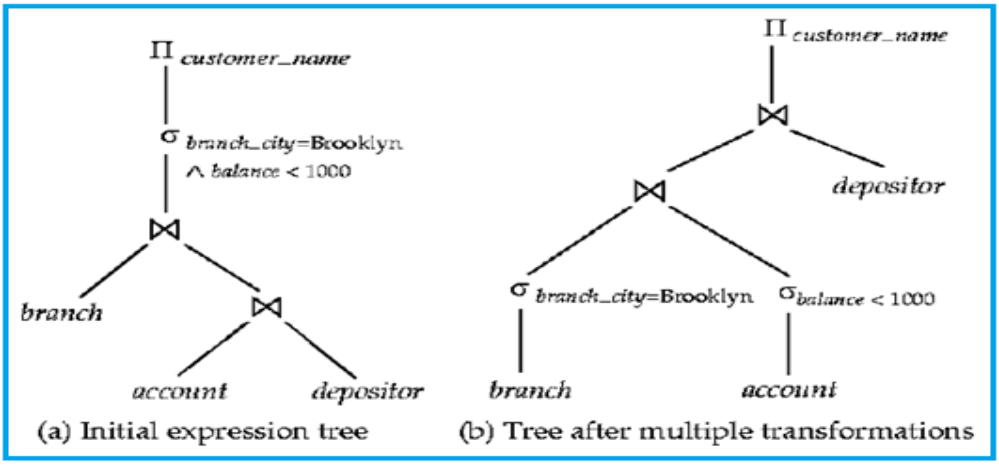

Fig.1

An execution of the query tree consists of executing an internal node operation whenever its operands are available and then replacing that internal node by the relation that results from executing the operation: Select : σ, Project : Π, Join : ⋈, Cartesian Product : x, A comparison operator: θ.

As regards query graph is a graph data structure that corresponds to a relational calculus expression. It does not indicate an order on which operations to perform first. There is only a single graph corresponding to each query. Operator graphs describe an operator-controlled data flow by representing operators as nodes that are connected by edges indicating the direction of data movement. In addition operator graphs can be used for the representation of algebra expressions [1].

From the viewpoint of query optimization, query processing is a set of activities which includes parsing the queries and translate them into expressions that can be implemented at the physical level of the file system, optimizing the query of internal form to get a suitable execution strategies for processing and then doing the actual execution of queries to get the results.

The cost of processing of query is determined by the several possible strategies for processing exist, especially when query is complex. The difference between a good strategies and a bad one may be several order of magnitude. In order to visualize what the main components of a database query optimizer are and how these components interact in order to produce a query plan that is ready for evaluation, it may be helpful to consider the figure 2 [2].

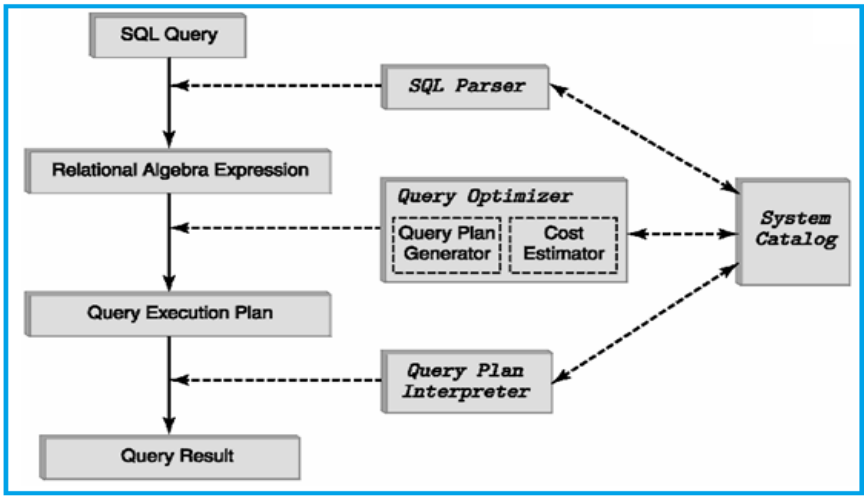

Fig.2

This article discusses the general optimization technique for query execution plan using the relational calculus representation of queries. Traditional query processing is dominated by statistics collection and probability theory, but analyzing and estimating query plans in the probabilistic manner described above has several inherent flaws.

At compilation-time, the statistics necessary to compute an optimal query plan may be unavailable or of poor quality. Additionally, the performance of any given plan may differ as available memory fluctuates, producing suboptimal results at different times in different situations depending on the server's load.

In distributed queries, network delays, node inefficiencies, and the potentially heterogeneous nature of the data make for suboptimal execution plans. Given the inherent and extreme complexity of database query optimization, it is reasonable to assume that query optimization exhibits a highly multimodal search space, and as such, precludes the use of either direct or indirect calculus-based search methods. Fortunately, there do exist a number of clever, robust search methods that work especially well for complex multimodal search spaces; one such method is known as a genetic algorithms.

The query processor applies rules to the internal data structures of the query to transform these structures into equivalent, but more efficient representations. The rules can be based upon mathematical models of the relational algebra expression and tree (heuristics), upon cost estimates of different algorithms applied to operations or upon the semantics within the query and the relations it involves. Selecting the proper rules to apply, when to apply them and how they are applied is the function of the query optimization engine, which can design to process particular relational operation and access path combinations using genetic programming.

2. GP Based Query Optimization

Genetic Programming (GP) as a specialization of genetic algorithms is based on biological evolution principle to find an optimum of the entire function. It is a machine learning technique used to optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a given computational task. Genetic programming is modification of genetic algorithms with one major difference. The population consists of individuals represented by specific data structure - trees. Inner nodes of the trees can represent functions (e.g. arithmetic operators, conditional operators or problem specific functions) and leaves would be terminals – external inputs, constants, zero argument functions. GP evolves computer programs, traditionally represented in memory as a tree structure, which can be easily evaluated in a recursive manner. Every tree node has an operator function and every terminal node has an operand, making mathematical expressions easy to evolve and evaluate. The main operators used in genetic algorithms such as GP are crossover and mutation [3].

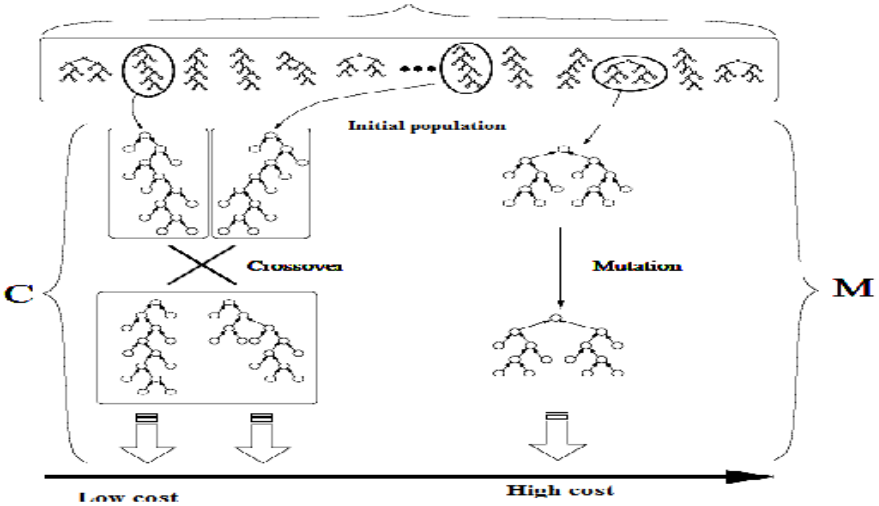

Fig.3

GP based the query optimizer module examines all algebraic expressions that are equivalent to the given query and chooses the one that is estimated to be the cheapest. A GP algorithm works on a population of individuals, each of which represent a potential solution to a treelike expression composed of relational algebra operations. It is assumed that by recombining relevant sub-trees, it is possible to produce new expressions that provide fitter solutions. In order to provide population diversity and allow the exploration of areas of the solution space not represented in the initial population, a mutation operator may also be used. Mutation merely consists of randomly changing a function, input or constant in one of the mathematical expressions making up the present population. The randomly selection of genetic operation (crossover or mutation) defines the branch of GP algorithm [4].

In the generation of reproduction loop the probability of each selected expression is proportional to fitness. The innovation of this method - difference between the Genetic Programming and algorithms starts in iteration process from the block of reproduction method selection. Several approaches are available: only the hybridization or mutation operators, hybridization or mutation operators randomized selection, hybridization or mutation operators to perform certain proportions, hybridization or mutation operators adaptive performance, fitness if the population. The dynamics of the function itself will be determining factor in the selection of genetic operators. Particularly, in case of fitness growing role dynamic hybridization is given the priority, closer to the optimum phase the mutation is given priority.

After the initial fitness population, fitness function of each tree is defined and population trees are sorted by means of ascending value or descending fitness function. Depending on the operator: mutation or breeding reproduction is carried out and the two “parents” or only one of the best is selected in the list. The remaining trees will no longer be considered. Then, according to the algorithm “weight coefficients” are calculated for the best tress and the “most hard” blocks or sub tree are showed and choosing (hybridization and mutation) points are defined.

A flowchart of the GP algorithm is shown in Figure 4.

In the case of the operation of hybridization “weight coefficients” are calculated for the second “parent” and the most “light” blocks is showed and chamfered and it is followed by the hybridization of the operation itself. In case mutation “weight coefficients” are calculated or the most “hard” blocks are showed and randomly generated “light” is replaced [5].

The algorithm are performed in two phases: ascending - are calculated according to the levels, sequentially, for each node, according to his relation operator tuple quantity. The second phase in the descending order according to the levels of tree nodes in the left or right subtree values of “weight coefficients. Determining the most “hard” or “light” blocks and crossing point for the hybridization and mutation operations. The result of calculation showed that the number of calculations and the time the fulfillment of the requirements are reduced by using the developed methods. Growing of the system is getting faster and more effective. So this is a new approach in the sphere of Query Optimization. This result will be successfully implemented in the Database management system. The main goal will be reached. The number of calculation will be reduced and the time will be saved. This is the major value of the database.

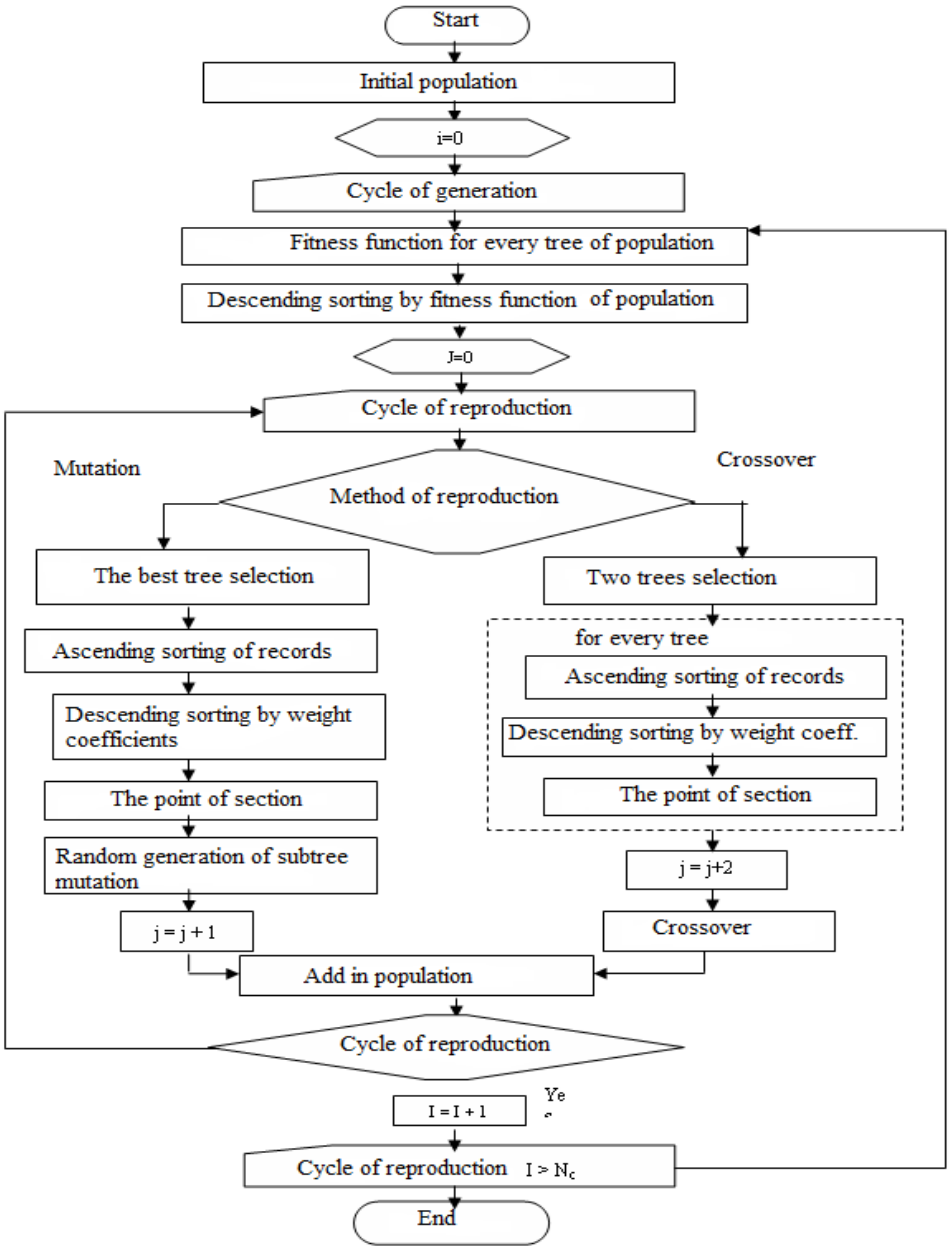

Fig.4

3. Conclusions

This article discusses the database querying process, which is probably the most studied data mining task. The query processor applies rules which can be based upon mathematical models of the relational algebra expression and tree (heuristics), upon cost estimates of different algorithms applied to operations or upon the semantics within the query and the relations it involves. Selecting the proper rules to apply are the function of the query optimization engine, which can design to process particular relational operation and access path combinations using *genetic programming*, where by recombining relevant sub-trees, it is possible to produce new expressions and to choose the one that is estimated to be the cheapest.

References:

1. Jarke M., Koch J. (1984). Query Optimization in Database Systems. Computing Surveys, Vol.16,No.2.
2. Ioannidis, Y. E. (1996). Query optimization, *ACM Computing Surveys*, vol.28, no.1, pp. 121–123.
3. Koza, J. (1992). Genetic programming: On the programming of computers by means of natural selection, The MIT Press, USA.
4. Koza J.R. (2007). Introduction to Genetic Programming/Tutorial. GECCO-LONDON.
5. wiTaSvili l., mefariSvili b., moTxovnebis optimizacia da genetikuri programirebis modificirebuli algoriTmi, Jurnali "inteleqti", №1(42), aprili, 2012 , gv. 122-126.

მონაცემთა ბაზების მოთხოვნათა ოპტიმიზაცია: გენეტიკური პროგრამირებისადმი ახალი მიდგომა

ლელა წითაშვილი[1], ბადრი მეფარიშვილი[2], გულნარა ჯანელიძე[2]

1. ახალციხის სახელმწიფო უნივერსიტეტი
2. საქართველოს ტექნიკური უნივერსიტეტი

რეზიუმე

განხილულია მონაცემთა რელაციურ ბაზებში მოთხოვნათა ოპტიმიზაციის პრობლემა. იმპლემენტაციის პროცესში მოთხოვნის ბინარული ხის ყოველი შესაძლო გეგმა ქმნის ე.წ. სიტუაციურ სივრცეს, რომელიც განისაზღვრება n! შესაბამისი განზომილებით, სადაც n ცხრილების რაოდენობაა. თანამედროვე მონაცემთა ბაზების მართვის სისტემები, მაგალითად SQL Server ერთ მოთხოვნაში შეიძლება შეიცავდეს 32-მდე ცხრილს. ასეთ შემთხვევაში მოთხოვნის შესრულების ვარიაციები 2.6 *1035 გეგმის ტოლია, ხოლო აქედან ოპტიმალური გეგმის რეალურ დროში ამორჩევა თვით ზესწრაფი კომპიუტერებისთვისაც შეუძლებელია. მონაცემთა ბაზების მართვის სისტემებში არსებული მოთხოვნის ოპტიმიზატორის მიზნობრივი ფუნქციაა ზემოხსენებული განზომილების პირობებში საუკეთესო გეგმის ამორჩევა, რისთვისაც მიზანშეწონილია ახალი, უფრო ეფექტური მეთოდების გამოყენება. სტატიაში განხილულია გენეტიკური ალგორითმების მოდიფიცირებული ვარიანტი, რომელიც მნიშვნელოვნად სწრაფად ახორციელებს რელაციური ალგებრის ოპერაციათა შესრულების ოპტიმალური თანამიმდევრობის პოვნას.

ОПТИМИЗАЦИЯ ЗАПРОСОВ БАЗ ДАННЫХ: НОВЫЙ ПОДХОД К ГЕНЕТИЧЕСКОМУ ПРОГРАММИРОВАНИЮ

Лела Циташвили[1], Бадри Мепаришвили[2], Гульнара Джанелидзе[2]

1. Ахалцихский Государственный Университет, Грузия
2. Грузинский Технический Университет

Резюме

Рассматривается одна из актуальнейших проблем управления базами данных, т.н. оптимизация запросов. В процессе имплементации бинарного дерева запроса каждый возможный план создает ситуационоое пространство, которое определяется соответственным измерением *n!* факториала, где *n* - это количество таблиц. Современные системы управления базами данных, например SQL Server в одном запросе может содержать до 32-х таблиц. В таком случае число вариации выполнения запросов равно 2.6 $*10^{35}$ планов, а выбор оптимального плана в реальном времени не под силу даже сверхбыстрым компьютерам. Целевой функцией оптимизатора запросов в системах управления базами данных является выбор найлучшего плана, для которого целесообразно использование новых, более эффективных методов. В статье рассматривается модифицированный вариант алгоритмов генетического программирования, который значительно быстро осуществляет поиск оптимальной последовательности выполнения операции реляционной алгебры.