SOME ASPECTS OF DATABASE QUERY OPTIMIZATION

Lela Tsitashvili[1], Badri Meparishvili[2]

1. Akhaltsikhe State Educational University, Georgia
2. Georgian Technical University

Summary

The ability to obtain to requests for desired information from the user i.e. query processing is one of the fundamental applications of database management (relational, distributed or others) and data warehouse systems. The principal mechanism through which a database maintains an optimal level of performances is known as the database query optimization, which is inherently essential matter for performing a complex search for the best possible plan among the set of semantically equivalent plans that can be generated for any given query. This article discusses the general query evaluation procedure using the relational calculus representation of queries. In relational DBMS, query optimization is based on formulation of a query and their reflection ontoan algebraic query evaluation tree. Because of the optimization's paramount importance to the robustness and flexibility of a database, it is worthwhile to explore the fundamental concepts behind the two topics at hand - databases querying and evolutionary algorithms.

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

1. Introduction

Every organization requires the making of decisions, the coordinating of activities, the handling of people, and the evaluation of performance directed toward group objectives. This article discusses the management of information systems in support of businesses. One way to view the process of management is to identify the basic functions, which together make up the process. Business intelligence includes tools in various categories, including some of the following:

- Data mining (DM) and Data warehouses;
- Decision Support Systems (DSS) and Forecasting;
- Knowledge Management;
- User/End-user Query and Reporting.

The term "Knowledge discovery in databases" is defined as the process of identifying useful and novel structure (model) in data. KDD is a broad area that integrates methods from several fields including statistics, databases, AI, machine learning, pattern recognition, machine discovery, uncertainty modeling, data visualization, high performance computing, optimization, management information systems (MIS), and knowledge-based systems [1]. In such information technologies and products as Relational Database Systems, Data Warehouse, Data Mining and Knowledge Discovery, Distributed Database or Knowledge system, and any Decision Support Systems there are some problem of data or semantic query processing [2,3,4].

Within literature, many different approaches of query processing can be found. In this contribution, we want to give a brief overview of some important definitions, because these problems are vital for our further discussions.

2. Optimization Objectives and Query Representation

Queries are used in several settings. The most obvious application is that of direct requests by end users who need information about the structure or content of the database. Queries are used in describes data to be retrieved from a database. 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*. Consider the relational schema of a database that describes the sample *customer* and *depositor* relations [5]:

customer (customer_ name, customer_ street, customer_ city)
depositor(customer_name, account_number)

Query example can be described by the expression:

Π*customer_name*((σ*branch_city*= "Brooklyn"(*branch)) (*account depositor)),

- Find the names of all customers with an account at a Brooklyn branch whose account balance is over $1000:

Π*customer_name*((σ*branch_city* = "Brooklyn"∧*balance* > 1000(*branch (account depositor*))),

- Transformation using join associatively:

Π*customer_name*((σ*branch_city*= "Brooklyn"∧*balance* > 1000(*branch account*)) *depositor*),

- Second form provides an opportunity to apply the "perform selections early" rule, resulting in the sub expression:

σ*branch_city*= "Brooklyn"(*branch*) σ*balance* > 1000(*account*)

- When we compute:

(σbranch_city= "Brooklyn"(*branch*) *account*),

we obtain a relation whose schema is (*branch_name, branch_city, assets, account_number, balance),*

- Push projections and eliminate unneeded attributes from intermediate results to get:

Π*customer_name*((Π*account_number*((σbranch_city= "Brooklyn"(*branch*) *account*)) *depositor*).

The transformation of query purports the multiple enumeration of possibilities of free graph-based query structure (figure 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. 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.

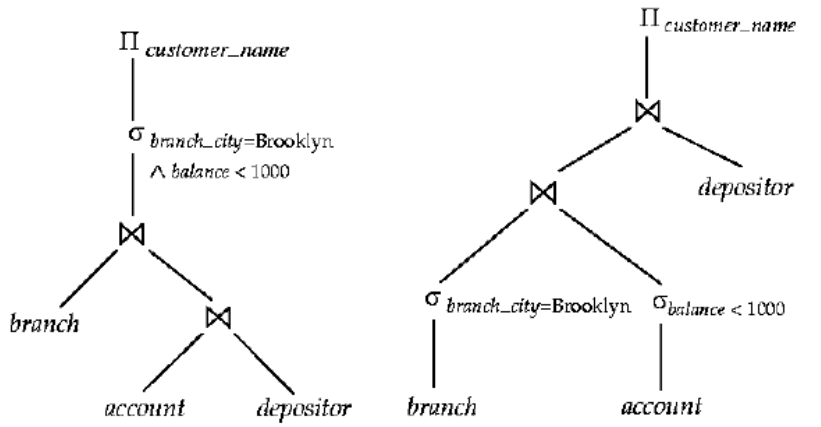

Figure 1

Within literature, the term "query optimization" is defined as the process of choosing a suitable execution strategy for processing a query. Query optimization research in the literature can be divided in two classes, which can be described as bottom up and top down [6].

Generally queries can be represented in a number of forms. In the context of query optimization, an appropriate query representation form must fulfill the following requirements: It should be powerful enough to express a large class of queries, and it should provide a well-defined basis for query transformation. In this section we present one of the different query representation forms, each of which has been used in a number of approaches to query optimization.

Graphs have been used for the visual representation of structured objects in a number of areas. Two well-known examples are the use of syntax trees in compiler construction and the use of AND/OR graphs artificial intelligence applications. graphs are used in query optimization for the representation of queries or query evaluation strategies.

Two classes of graphs can be distinguished: object graphs and operator graphs. Nodes in object graphs represent objects such as (relation) variables and constants. Edges describe predicates that these objects are to fulfill. Object graphs contain the properties of the query result and are therefore closely related to the relational calculus. 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 [7].

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 [8].

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.

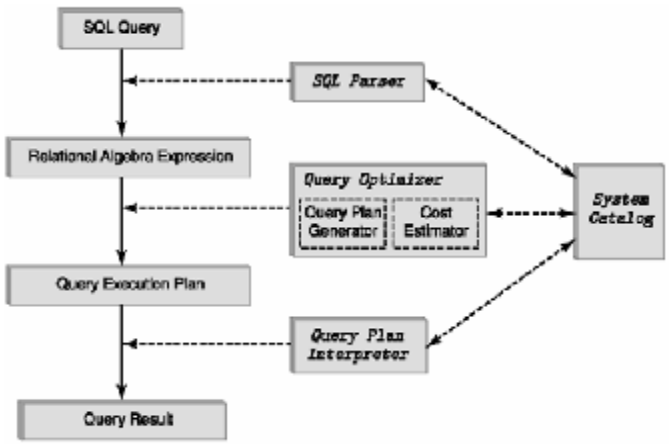

Figure 2

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.

3. Genetic Programming – Brief Overview

Genetic Programming (GP) as a specialization of genetic algorithms is based on biological evolution pronciple 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 [9].

Crossover is applied on an individual by simply switching one of its nodes with another node from another individual in the population. With a tree-based representation, replacing a node means replacing the whole branch. This adds greater effectiveness to the crossover operator. The expressions resulting from crossover are very much different from their initial parents. Figure 3 illustrates standard crossover on trees, which is the most common method of recombination used in GP.

Mutation is realized by choosing a random node in the tree. The sub trees in the chosen nodes are recombined, i.e. swapped between two parents to the corresponding place. To maintain integrity, operations must be fail-safe or the type of information the node holds must be taken into account. For example, mutation must be aware of binary operation nodes, or the operator must be able to handle missing values.

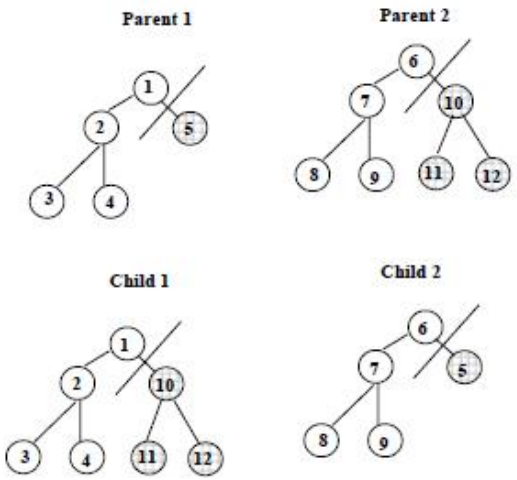

Figure 3

4. GP Based Query Optimization

In practice, SQL is the query language that is used in most commercial RDBMSs. An SQL

query is first translated into an equivalent extended relational algebra expression-represented as a query tree data structure-that is then optimized. 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. In the generation of reproduction loop the probability of each selected expression is proportional to fitness. A flowchart of the GP algorithm is shown in Figure 4.

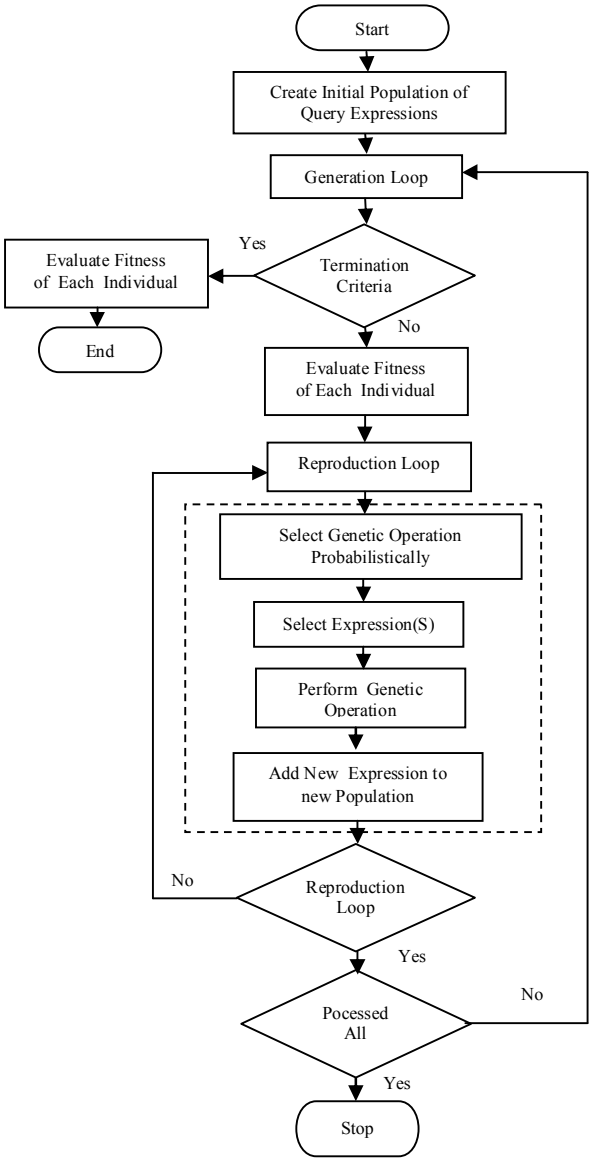

Figure 4

5. 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 chooses the one that is estimated to be the cheapest.

References:

1. Markovic, I., and Pereira, A.C. (2007). Towards a Formal Framework for Reuse in Business Process Modeling. In *Workshop on Advances in Semantics for Web services(semantics4ws), in conjunction with BPM '07*, Brisbane, Australia.

2. Meparishvili, B. Data and Knowledge Management in Information Systems: An Overview. The International Scientific Conference Devoted to the 80-th anniversary of Academician I.V. Prangishvili *"Information and Computer Technologies, modeling, Control"*. Tbilisi, Georgia, November 1-4, 2010.

3. Meparishvili, B. Ediberidze, A. Janelidze, G. New Approaches to a Modeling of Knowledge. IFAC 9 th Workshop on Intelligent Manufacturing Systems (IMS'08), Szczecin, Poland, October 9-10, 2008. 99-103 pp.

4. Meparishvili, B. Kervalishvili, P. Janelidze, G. Information Exchanging Processes: Some Views. ERA-5, SynEnergy Forum.The conference for International Synergy in Energy, Environement, Tourism and Information Technology. T.E.I. Piraeus, Greece, 15-18 September, 2010.

5. Database System Concepts Sixth Edition Avi Silberschatz Henry F. Korth S. Sudarshan McGraw-Hill ISBN 0-07-352332-1.

6. Matthias Jarke, Jurgen Koch. (1984). Query Optimization in Database Systems. Computing Surveys, Vol. 16, No. 2,

7. Ioannidis, Y. E. (1996). Query optimization, *ACM Computing Surveys*, vol.28, no.1, pp. 121–123.

8. Koza, J., (1992), Genetic programming: On the programming of computers by means of natural selection, The MIT Press, USA.

9. Koza J.R. (2007). Introduction to Genetic Programming/Tutorial**.** GECCO-LONDON.

მონაცემთა ბაზებში მოთხოვნების ოპტიმიზაციის ზოგიერთი ასპექტი

ლელა წითაშვილი[1], ბადრი მეფარიშვილი[2]

1-ახალციხის სახელმწიფო უნივერსიტეტი, საქართველო
2-საქართველოს ტექნიკური უნივერსიტეტი

რეზიუმე

მონაცემთა ბაზების მართვის (რელაციური, განაწილებული და სხვ.) და მონაცემთა საცავების სისტემების ერთ-ერთი ფუნდამენტური დანიშნულებაა მომხმარებელმა მიიღოს მოთხოვნების შესაბამისი ინფორმაცია, ანუ მოხდეს მოთხოვნების შესრულება. ამ პროცესში მთავარი მექანიზმი, რომლის მეშვეობითაც მოთხოვნათა შესრულების გეგმა აღწევს ოპტიმალურ სტრუქტურას, ემყარება რელაციური ალგებრისა და აღრიცხვის პროცედურებს მოთხოვნების წარმოდგენისას. მბმს-ში მოთხოვნების ოპტიმიზაცია ეფუძნება მოთხოვნების დეკომპოზიციასა და მათ ასახვას მოთხოვნათა შეფასების ალგებრულ ხეზე. მოცემული სტატია განიხილავს მოთხოვნების ოპტიმიზაციის შესაძლებლობებს ევოლუციური ალგორითმების, კერძოდ გენეტიკური პროგრამირების გამოყენების საფუძველზე.

საკვანძო სიტყვები: მოთხოვნების დამუშავება. მოთხოვნების ოპტიმიზაცია. ევოლუციური ალგორითმები. გენეტიკური პროგრამირება.

НЕКОТОРЫЕ АСПЕКТЫ ОПТИМИЗАЦИИ ЗАПРОСОВ В БАЗАХ ДАННЫХ

Л. Циташвили[1] , Б. Мепаришвили[2]
1-Ахалцихский Государственный Университет
2-Грузинский Технический Университет

Резюме

Одно из фундаментальных назначений систем управления базами данных (реляционными, распределенными или др.) и систем хранилищ данных является способность получения желаемой информации по запросам пользователя, то есть обработка запросов. В этом процессе главный механизм, посредством которого план выполнения запросов принимает оптимальную структуру, базируется на представление запросов на основе реляционной алгебры и исчисления. В реляционных СУБД оптимизация запросов базируется на формализации запросов и их отражении на алгебраическом дереве оценок запросов. В данной статье обсуждается подход к оптимизации запросов с использованием эволюционных алгоритмов, в частности, генетического программирования.

Ключевые слова: обработка Запроса. Оптимизация Запроса. Эволюционные алгоритмы. Генетическое программирование.