UDC 004.65
DOI: 10.15587/1729-4061.2019.170071

CONSTRUCTION OF A PARAMETRIC MODEL OF COMPETITIVE ACCESS IN RELATIONAL DATABASES BY USING A RANDOM FOREST METHOD

D. Gromey*
E-mail: gromeydd@outlook.com
E. Lebedenko
PhD, Associate Professor*
E-mail: lebedenko_eugene@mail.ru
D. Nikolaev*
E-mail: mriddi@bk.ru
T. Rozhkova*
E-mail: tancha2302@yandex.ru
*Department of Informatics and Computer Science
Federal State Government Educational Institution of Higher Education
"The Academy of Federal Security Guard Service of the Russian Federation"
Priborostroitel'naya str., 35,
Orel, Russia, 302015

Розглянуто проблему моделювання часу виконання запитів в автономних реляційних базах даних з конкурентними запитами. Відзначено недоліки існуючих підходів, які ігнорують витрат на частку послідовних операцій при кооперативному доступі до даних в ієрархії пам'яті. Розглянуто питання застосування умовної вартості виконання складових операцій плану запиту, замість розрахунку передбачуваного часу обчислень.

Запропоновано спосіб формального формування прецедентів навчальної вибірки, а також підхід до побудови регресійної моделі. Розроблена модифікація методу машинного навчання випадковий ліс застосовується для розрахунку часів виконання запитів за їхніми текстами та тимчасовими позначками початку, тривалості виконання.

Розроблена параметрична модель конкурентного доступу до даних необхідна для отримання точних оцінок часу виконання запитів при використанні паралельних обчислень. Моделі з подібними характеристиками потрібні в рішенні задач автоматизації управління фізичною схемою даних, створення СУБД, що самовизначаються. Ключовими відмінностями від існуючих підходів є використання часу виконання запитів в якості цільового значення, обліку значень предикатів і взаємовпливу паралельно виконуваних запитів.

Для підтвердження отриманих результатів була використана імітаційна модель на базі широко відомого тесту TPC-C. В якості функції втрат, з урахуванням регресійної природи моделі, було використано відношення суми модулів різниці фактичних і одержуваних часів до фактичних часів. Сама перевірка проводилася за контрольною вибіркою, що сформована за зростаючою довжиною навчання на відкладених даних. В ході проведених досліджень була доведена можливість застосування методу машинного навчання випадковий ліс для обробки статистичних даних виконання SQL запитів. Отриманий результат свідчить про перспективність такого підходу і дозволяє отримувати параметричні моделі конкурентної обробки запитів

Ключові слова: автономні систем управління базами даних, бази даних, що самовизначаються, випадковий ліс, конкурентний доступ, паралельні обчислення в реляційних систем управління базами даних

1. Introduction

Recent decades have seen a tangible improvement in the informatization of society, increasing the degree of integration of information technology in all spheres of human life. Their contribution to scientific and technical progress has been growing, with an explosive increase in the volumes of stored and processed data. More and more new directions are covered by the implementation of information systems and underlying most of them are database management systems as a set of tools to manage information represented in objective form.

Since the establishment of the earliest general-purpose DBMS, and up to now, the main requirement to them has been their operation under different conditions. A database software must ensure maintaining the operational performance and execution of its functions under conditions of uncertainty for the resulting working conditions with an adequate response to changes in them. In this case, DBMS should work optimally at different characteristics of processed data and hardware and software tools. Specifically, the amount of data stored may vary from hundreds to tens of kilobytes of exabytes and the number of logical cores from one to several hundreds. In this case, the most important factor that has remained unknown is the information whose structure and statistical characteristics is to be stored in a database. It is the physical structure of data that is decisive in the choice of an optimal way to transmit the requests that enter the system into a final set of operations over the stored data. And the greatest progress has been made in this field by relational databases with competitive access, which are the most commonly used.

Alas, up to today, a key role in adapting the operation of DBMS to a specific database has belonged to a human, a database administrator. However, the number of operated databases has significantly exceeded the number of administrators, and their complexity is such that even a professional and experienced specialist must resort to assistive technology. People have to use not the experience or strict mathematical and formalized representations, but rather an experiment involving a particular database. The issue on constructing autonomous databases or, in other words, self-identifiable DBMS, is an increasingly apparent challenge towards further development. Building self-identifiable DBMS requires the parametric models that would be suitable for estimating the time of parallel query execution. Such estimations are needed in the automation tasks on forming the distribution of data in a PC memory. Importantly, in contrast to existing ones, they should take into consideration competition between requests, their predicates, as well as possess greater accuracy. Known methods for assessing the cost of computing are used at the stage of choosing a query execution plan – the time spent on them is part of the time costs for query execution. Existing industrial implementations operate in soft real-time and the final result is considered to be any intermediate result achieved at exhausting the reserve of time for choosing a quasi-optimal query plan. In turn, autonomous DBMS are designed to perform operations that change the distribution of data under a background mode, in parallel to other computations. The arising difference in the application technique removes constraints for data processing time. It becomes possible to construct new methods by using the previously unavailable approaches, and to improve the accuracy of calculated estimates due to a higher computational complexity. There is a need to devise new models for competitive access to data in relational databases.

2. Literature review and problem statement

Underlying the work of an autonomous, self-identifiable database is a change in the current state of internal storage structures changing the distribution of data. Such a change must be based on an estimation of the existing distribution optimality in relation to the diversity of its alternatives [1]. These requirements are essentially close enough to earlier works in the field of construction of DBMS – the need to estimate possible query execution plans by a DBMS optimizer [2]. Therefore, there are quite a lot of related developments in this subject area, which, under different constraints, resolve a targeted task on assessing the cost of query execution.

Modern DBMS employ approaches described in [2–4] for estimating the selectivity of different query execution plans. The generalized idea of such papers comes down to the idea of uneven distribution of values over a data set when the preliminary estimation of query predicates makes it possible to establish its correspondence to a meaningful or negligibly small part of rows in the target table. For multi-version DBMS, such an estimation is typically the primary criterion to the use of auxiliary data structures. For example, one can select one of the indexes or, in the opposite case, there may happen that they are abandoned in favor of a full scan of all rows in the table. In the latter case, such an estimation is used to minimize the costs of indirect access to the data. For locking-based DBMS, this estimation is also used for solving a task on the pre-emptive selection of the lock escalation level. Key shortcomings of this direction are at the same time the purpose of its improvement [3, 4] – the impossibility to compare the previously calculated statistics to the whole variety of possible conditions for incoming queries, which limits its applicability in principle. The case of parallel computing provokes the same fundamental issue on the inability to take into consideration the cooperation of shared data access among concurrent requests. The queries selectivity estimation approaches based on the pre-calculated statistics are certainly an important part of modern DBMS. However, given their constraints, such numerical estimates are used mostly as an auxiliary tool.

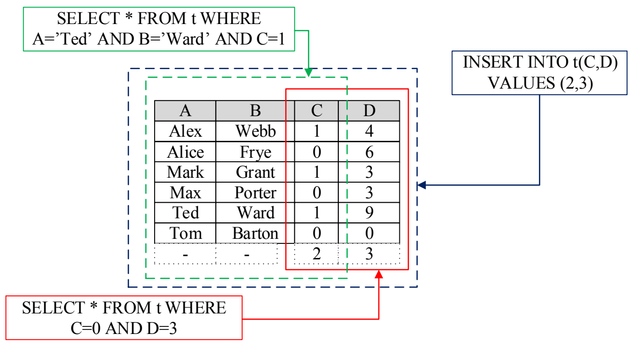
Most modern DBMS employ approaches based on the algorithms by Graefe [6], Bassou [7] and their ideological equivalents. The general idea of this field of research is the introduction of cost functions that conditionally estimate the cost of any query execution plan in certain abstract units. Every object involved in the course of executing a query plan is assigned with the sets of access techniques, and their combinations, in turn, are assigned with some functions for estimating the cost of choosing any path of execution. One of the first implementations, adopted commonly in manufacturing, was the implementation by IBM DB2 [8]. A distinctive feature of the proposed implementation was a high adaptability of estimating read operations or the modification of data scheme objects, depending on the locking captured over the structure of a database. The IBM-proposed implementation made it possible to take competitiveness into consideration and to correct the choice of a query execution plan depending on the queries already addressed.

The main problem with the methods based on estimates of the cost of implementation is the measurement units used. The ultimate magnitude that evaluates the cost of executing multiple requests is the time required to execute each of them. The use of surrogate measurement units was due to the high complexity of forecasting time, leading to a loss of accuracy [1]. Unfortunately, this cost has a nonlinear relationship directly to a run time, and does not make it possible to effectively estimate it. Moreover, there may be situations when a request with a lower value would take longer due to the linear nature of the final cost calculations, at the non-linear character of operations [4, 8]. Cost approaches, certainly a good solution to problems on searching for a query execution plan, given the stringent time limits and the high frequency of solving this task, for each request. However, low accuracy makes their application inappropriate for tasks at self-identifiable databases.

A series of papers [9, 10] describe methods for estimating the operation of a database based on imitational simulation. Such approaches have not become practically applied at industrial systems due both to the high cost and technical complexity for constructing adequate models, as well as due to the limited character of results derived. Most of the proposed simulation models do not take into consideration the predicates of handled queries, or offer an estimation for the cumulative characteristics of the simulated system. Due to these constraints, simulation methods are considered inappropriate for use in promising autonomous DBMS.

A quite separate direction is the class of methods for estimating the query execution costs as part of parallel computing. The approaches, described in a series of papers, are mainly based on the set-theoretic models of cooperative access to a linearly represented shared memory [11] or to resources as the hierarchically organized and logically separate pools of memory domains [12]. An example of competitive access to data is shown in Fig. 1.

However, such studies are mostly theoretical, the main constraint implied is the linear homogeneity of memory as a PC resource. At the same time, existing processor architectures imply the inverse memory features. A nonlinear memory access occurs, as a superposition of interaction between different levels of caches, the distribution of arithmetically-logical devices and cache time among the concurrent conveyors of commands. The access time is influenced by the types of memory – buffered, non-buffered, or register. In the same way there arises the reciprocal effect of the sequence of the processed memory access operations. The proposed methods are acceptable for the case of estimating the costs of cooperative query execution at their equivalent time of calculation for sequential execution. At the same time, such models face the constraint of an issue on processing heterogeneous queries. In existing information and computing systems, this situation is rare. As a result, the derived models for queries computation are either of little value or require substantial rework for industrial application.

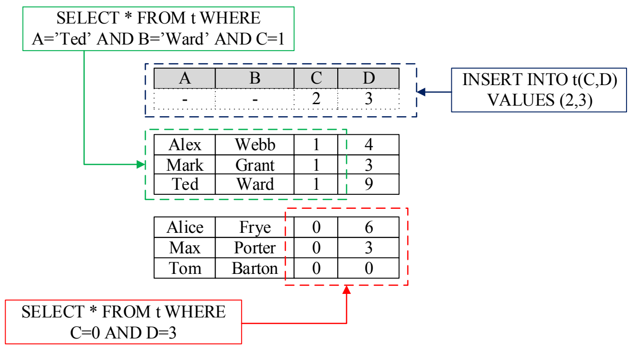

SELECT * FROM t WHERE A='Ted' AND B='Ward' AND C=1
INSERT INTO t(C,D) VALUES (2,3)
SELECT * FROM t WHERE C=0 AND D=3

A	B	C	D
Alex	Webb	1	4
Alice	Frye	0	6
Mark	Grant	1	3
Max	Porter	0	3
Ted	Ward	1	9
Tom	Barton	0	0
-	-	2	3

Fig. 1. Example of competitive access to data in a relational DBMS

Papers [13, 14] focus on the application of machine learning theory approaches to construct autonomous DBMS. The papers consider the statistical characteristics of a set of queries that entered the system over a certain time interval. For any query, predicates are extracted and summarized into a variety of patterns with a less cardinal number, based on the exclusion of specific values from conditions. The obtained sets are clustered and sent to the input of a recurrent neural network with LSTM units of long-term memory that makes it possible to generalize the frequency and occurrence of requests trends. The resulting estimates are used for the automatic selection of indexes in paper [13] and the physical representation of data [14]. The papers first of all prove the fundamental applicability of methods from a machine learning theory to construct autonomous DBMS. The proposed models lack the estimation of costs related to competitive access to data; in fact, the prototype proposed by the authors is mostly oriented towards a sequential processing of queries.

The model built on the basis of a neural network estimates only the data schema objects specified in query predicates, thereby ignoring the statistical characteristics of data reported in papers [2, 8, 9]. Query execution time is ignored, which provokes blunders in the conditions for processing heterogeneous queries. For example, suppose that a table contains two fields – on this case, a set of query texts mentioning the first field has a greater cardinality, and the second – the larger sum of execution time. The proposed method would select the first field in the first place to automatically generate the index, at the same time, all other things being equal, it is the index of the second field that has a larger impact on the target parameter – a query execution time. The proposed approach is not applicable for the operation of data segmentation, as it does not make it possible to select a segmentation key. Fig. 2 shows an example of table segmentation for field C for parallel execution of multiple queries.


SELECT * FROM t WHERE A='Ted' AND B='Ward' AND C=1
INSERT INTO t(C,D) VALUES (2,3)
SELECT * FROM t WHERE C=0 AND D=3

A	B	C	D
-	-	2	3

Alex	Webb	1	4
Mark	Grant	1	3
Ted	Ward	1	9

Alice	Frye	0	6
Max	Porter	0	3
Tom	Barton	0	0

Fig. 2. Example of table segmentation for parallel computing

It should be noted that the current situation with a limited use of existing approaches is due to historical reasons, stages in the development of technological base and the course of processes related to informatization of the society. When DBMS were first constructed, they existed in tens and hundreds, were often experimental and with a high price. Relative cost of human labor involved in solving manually the tasks on database optimization was negligible compared to the cost of the employed technical means. And the direct complexity of work rendered automation impractical. Decades of growth in the number of database copies and their diversity, the increasing volume of data stored, as well as cheaper hardware, have led to today's situation, quite opposite to the days of yore. And it is the absence of self-identifiable, autonomous databases that contributes to stagnation process when DBMS often operate slower by tens or hundreds of times than it could have been achieved at the same hardware resources.

3. The aim and objectives of the study

The aim of this study is to construct a parametric model of competitive access in relational databases, which would make it possible to take into consideration the predicates and mutual influence of queries executed in parallel by methods of machine learning. This would enable the application of operational statistics of a relational DBMS in order to identify the tuples of logical objects and the conditions for access assigned by queries, to rank them based on the greatest contribution during parallel data processing. The ordered sets of objects, derived in this way, could be used in algorithms that control the physical scheme of data storage, adjusting it for the load characteristic of a specific database.

To accomplish the set aim, the following particular tasks must be solved:

– to devise a set-theoretic model of control over the process of competitive access to queries in relational databases;

– to construct a method for calculating the parametric model of data access in a relational DBMS by using machine learning;

– to perform computational experiments that would confirm the effectiveness of the constructed model and method, using a simulation model of the competitive processing of queries in a relational database.

4. Materials and methods to study the process of competitive access in relational DBMS

4. 1. A set-theoretic model of the process of control over competitive access to queries in DBMS of the relational type

In a general case, query processing in a relational DBMS represents a certain mapping

$$DBMS : Q \times S \xrightarrow{CPU} (\Re, T, S'),$$

within some computer system *CPU*. The defining domain is the Cartesian product over the set of valid queries *Q* and possible internal database states *S*. A range of values is assigned by a set of tuples from the result of computing the query $\Re$, on the time required for its execution *T*, as well as a new internal state of the database *S′*.

In turn, each query

$$Q = (Q^{SQL}, \{Q^{PRD}\})$$

is a tuple of the direct text of query Q^{SQL} in the SQL language, and the set of related predicates Q^{PRD}, corresponding to the query bind variables. The set S is represented by its Cartesian product

$$S \subseteq S_{PD} \times S_{UD} \times S_{AD} \times S_{SD}$$

of particular states:

– S_{PD} – set of objects assigned by a developer to represent information in accordance with the rules of relational algebra, in particular, a table, constraint for integrity, or relation;

– S_{UD} – representation of the information assigned by users, directly stored data;

– S_{AD} – auxiliary structures that define the arrangement of data in memory and the structures of a database assigned by a DBMS administrator or mechanisms for self-determination, including indexes or segments;

– S_{SD} – internal structures of a DBMS core, used in the processing of queries, including all types of caches.

When solving optimization problems for DBMS, a target parameter is the time *T* that represents the sum of the times of different stages in the processing of the original query. And, in accordance with the above expressions, the problem can be solved in several ways:

– modernization of hardware *CPU*;

– change of queries to system *Q*;

– improvement of DBMS software implementation by changing S_{PD} and S_{SD};

– reduction in the volume of stored data and improvement of their model by changing S_{UD};

– change in the parameter S_{AD}, generated by a DBMS administrator.

In their practical work, database administrators operate the already developed DBMS software, designed by information systems that generate queries *Q*. As a result, the administrators must ensure the optimal use of *CPU* resources for the assigned data S_{UD} – the parameters are, accordingly, the constants for solving an optimization problem: $(S_{PD}, S_{UD}, S_{SD}) = \text{const}$. Thus, an administrator, all other parameters unchanged, should find such a value for S_{AD}, at which $T \to \min$. In turn, the goal of an autonomous DBMS is the iterative change in S_{AD}, at which $\sum T \to \min$ for the set of transformations $DBMS$, over a certain interval of DBMS operation.

Solving such an optimization problem requires using the following functional dependence

$$DBS : Q \times S \xrightarrow{CPU} T.$$

In this case, any modern DBMS is based on parallel computing, so that at any point in time the system can process more than one query, which in turn defines the operating mode by the Amdahl's law. An additional constraint is the ACID requirements, in particular the need for the isolation of queries over a dataset S and the atomicity of transitions $S \to S'$. Similar transformations form a share of the serial computation of the law and lead to strong mutual dependences of representations of DBM parallel queries. In a system with parallel computation the expression acquires the following meaning

$$DBM : Q_t \times S_t \xrightarrow{CPU} T,$$

where Q_t is the query received at time t, and S_t is the internal state of a database.

Constructing autonomous DBMS requires the representation of DBM mapping at an accuracy sufficient to reduce the target parameter $\sum T \to \min$. However, the complexity of accurate representation of the hardware and software tools in an analytical form simply rules out such an approach even for systems with sequential computation [9, 12]. The resulting set-theoretical model makes it possible to formally relate the target parameter, the sum of time for processing a set of queries, to the basic characteristics of the examined system.

4. 2. Approach to forming an attributive description of queries in relational DBMS

Suppose that over a certain time of observing the computing set TN the system with initial state S^N received a set of queries Q^N, each at its own point of discrete time:

$$Q^N \subseteq \{Q_t^N\}, t \in TN.$$

Accordingly, computing led DBM has led to a cascade of transformations

$$S_{t0}^N \to S_{t1}^N \to \ldots \to S_{\max(t)}^N$$

and the identifying set T^N in the domain of values for a possible execution time. Fix the conversion constants as the constraints to the model: $CPU = \text{const}$, $S_{SD} = \text{const}$. Values for S_{PD}, S_{UD} can change in the cascade of transformations, but cannot be changed when solving a problem. By definition, the queries from a data sample do not generate an actual change in the state

$$(\forall Q : Q^{SQL} \propto SELECT) S = S'.$$

In this case, according to several studies [2, 8], the ratio of the changed information during other queries tends to be much less than its total magnitude $I(S / S') << I(S)$. Thus, a state transition occurs at small iterations, and the vector of distance between the points in space is significantly less than the maximally possible magnitude for the metric in this space. Quite rare are the situations when the queries that change the set of rows in tables dominate, in terms of their number or execution time, the queries that modify individual rows. Therefore, one can generally assume true statement:

$$\left|DBM(CPU,Q_t,S)-DBM(CPU,Q_t,S')\right|<<$$
$$<<\left|DBM(CPU,Q_t,S)-DBM(CPU,Q_t,S^R)\right|.$$

In the expression, S^R is a random point in a state space. Thus, as confirmed by a series of studies [8, 10], over the period of observation the mapping of DBM at arbitrary S^N can be represented as

$$DBM\approx DBM^{\Delta}:Q_t^N\to T_t^N+\Delta N,$$

where ΔN is the error over the observation interval, called a cascade of transformations S^N. This is an important expression for an autonomous database, because it allows one to arrive at expression:

$$\sum_{TN}DBM(CPU,Q_t^N,S_t^N)=T^N\approx\sum_{TN}DBM^{\Delta}(Q_t^N)=$$
$$=T^N+\Delta N\approx\sum_{TN'}DBM(CPU,Q_t^{N'},S_t^{N'})=T^{N'},$$

defining the attributive description of a query. For an autonomous database, in addition to an observation period TN, there is also a period TN' of operation after transition

$$S_{AD}\to S_{AD}{}'.$$

Therefore,

$$S_{t0}^{N''}=S_{t0}^N;$$

$$Q^N=Q^{N'}\Rightarrow T^N=T^{N'}$$

and at a significant number of periods TN the dependence $Q^N\to T^N$ obeys the central limit theorem, so that changing a negligibly small subset of elements from set Q^N changes the total query execution times in negligible manner. Consider the temporal observation intervals followed by the module of difference between the sum of representation DBM^{Δ} for the sets of queries Q_t^N and $Q_t^{N'}$. It can be argued that the representation DBM^{Δ} of the same subsets is less than similar modules for pairs (Q_t^N,Q_t^R) and $(Q_t^{N'},Q_t^R)$, where Q_t^R is a random subset Q, not less than in half of the cases. This axiomatic expression defines the natural constraint for operation of autonomous databases, and is a key axiom for them. In accordance with this expression and the assumption defined by it, hereafter one may consider DBM^{Δ}, as an approximation of DBM. The error between them can be neglected, because the magnitude for this error is comparable to the magnitude of an error arising from the mismatch between sets Q_t^N and $Q_t^{N'}$. Such a transition is important, since a particular state S_t^N, represented in the form of a binary code, could require tens of gigabytes or terabytes for its record while operating with variables of such dimensionalities is impractical.

4. 3. A method for calculating the parameters of a set-theoretic model of the process of control over a competitive access of queries to a DBMS of the relational type, based on a random forest algorithm

To represent the desired functional dependence, in this paper we suggest using approaches from a machine learning theory. We shall consider the set of pairs (Q_t^N,T_t^N), related through the representation DBM^{Δ}, to be a certain training sample. Since T_t^N is the time of query execution, the problem belongs to the class of problems on the reconstruction of a numerical regression. It is often the case in practical tasks on machine learning that the main regularities of restored dependences are not known. Selecting a subset of used attributes and applied algorithms depends on examining the basic statistical characteristics and searching for attributes' correlations among each other. In a general case, the basic patterns in the restored dependences are not known, so finding a reasonable subset of attributes and a particular machine learning method is performed experimentally, based on studying statistical characteristics and numerical experiments. However, such an approach is not applicable to the problem being solved because (Q_t^N,T_t^N), CPU, S would vary depending on a specific DBMS, changing in an undefined way the results from numerical experiments. However, at the same time, for the examined problem, the key patterns are known, those that link individual characteristics for variable Q_t^R, S_t^R, T_t^R to the successive queries, as well as the general character of change is the dependences on variable CPU and queries that are executed in parallel. We shall prove this assertion by introducing the determination of a query execution time:

$$T=T_{net}+T_{ast}+T_{opt}+T_{eq}+T_{da},$$

where T_{net} is the time of query transfer from an information system to a DBMS core; T_{ast} is the time of operation of the syntactic and lexical analyzer parsing a query; T_{opt} is the time of optimizer's work required to choose, among existing, or to find a new query execution plan; T_{eq} is the time to perform computing operations in an isolated workflow (operations on sorting, thinning, or converting an extracted dataset); T_{da} is the time to access the internal state of a database S_{UD}.

By definition, variables T_{ast}, T_{opt} are associated with states S_{PD}, S_{SD} and are constant. In practical tasks, the operation time of the parser and optimizer is neglected [8, 10] due to its small contribution to the final query execution time. Similarly, time T_{net} is assumed to be a constant, because it depends on dimensionalities $(Q,\Re)$, as well as channels of communication between a DBMS and an information system. Time T_{eq}, by definition, depends only on $(CPU,\Re)$ and it is a constant for our problem. Thus, the examined expression can be represented in the form

$$Q_t\times S_t\xrightarrow{CPU}T_{eq}+\text{const}.$$

Imagine S_t to be an arbitrary point in the multidimensional space that may be represented by a certain vector. Thus,

$$S_{UD}=(a_0,a_1,...,a_n);\;\; a\in A,$$

where A is the set of atomic data in a DBMS memory. In this case, it follows from determining $Q=(Q^{SQL},\{Q^{PRD}\})$ that a query text in the SQL language along with a data scheme form a set of predicates $Q^{SQL}\times S_{PD}=\{Q_{SQL}^{PRD}\}$, which in turn is merged with the set of original query predicates in order to compute

$$DBMS':(\{Q_{SQL}^{PRD}\}\cup\{Q^{PRD}\})\times S_{UD}\xrightarrow{CPU}(\Re,T_{da}).$$

It means that in the applied formalization each predicate of a query establishes the following mapping

$$Q^P : A \to A^{PRD}, \ \left|A^{PRD}\right| \le |A|, \ Q^P = \{Q_{SQL}^{PRD}\} \cup \{Q^{PRD}\},$$

while pooling or a set of predicates forms, respectively, the intersection of their values' domains.

One should separately note the relationship between the determined predicates Q^{PRD} and the actual content of an SQL query where there are possible inversion operations NOT, merging OR, intersection AND of conditions, as well as the formation of their hierarchies, of almost any complexity. All these operations are formed in the terms of the algebra of logic and, as a consequence, can be normalized in a conjunctive form, thereby corresponding to the terminology of the article. Thus, the text of a query and its predicates form a constraint for an internal state S_{UD}, such that the query is calculated over a certain part of this state $A^Q \subseteq A$, formed by the intersection of sets of atomic values. In this case, any $a \in A$ is a binary sequence in the memory of a computer, so the power of set A^Q defines the number of operations on data access at time T_{da}. In this case, various queries, while having different sets of predicates, would form different subsets over A that may overlap in this case. In this case, the shared access to the overlapping subsets of atomic elements would generate competition and additional time expenses for a computing system [1], as well as the share of successive operations on line with the Amdahl's law, forming a dependence of execution time on the concurrent queries.

The resulting dependence makes it possible to proceed from the initial form of dependence

$$DBM^{\Delta} : Q_t^N \to T_t^N$$

to the following form

$$T_t^N = T_{DA} + \text{const} = \\ = X_0(Q_t^P) + \sum_{i \in I} k_i \cdot X_i(Q_t^P \cup Q_i^P) + \text{const},$$

where

$$Q_i^P = \{q^P \in Q \mid A^{q^P} \cap A^{Q_i^P} \ne \varnothing\}$$

corresponds to the set of predicates in parallel running queries that specify constraints for S_{UD}, forming the non-empty sets of atomic elements in the intersection with the set of atomic elements, formed by a combination of predicates defined by the examined query Q_t. In turn, coefficients k_i assign the ratio of execution times for the investigated, as well as I, classes of concurrent queries, where each class is assigned by the combination of predicates Q_x^P. An interesting feature of the derived expression is the absence of S_{UD}, as a consequence of the previously proven approximation

$$DBM \approx DBM^{\Delta} : Q_t^N \to T_t^N + \Delta N.$$

For the simplicity in all expressions, we also assume that the emergence of two queries Q at a single point in time t is not possible because of such a small magnitude of sampling that could only be required.

By using the introduced constraints, a training sample can ve represented by a set

$$Q_t^P \cup \bigcup_{i \in I} (i, k_i, Q_i^P).$$

Such a feature space corresponds to those dependences that we derive by examining the simulated system. In this case, the resulting space has several disadvantages. First, this is the complexity of representing the predicates in a binary form while preserving their sufficient semantics. For the method of support vectors to apply, we must select a conversion kernel – a kernel function. The selected representation must retain in the resulting multidimensional space the distance metrics corresponding to the relationships within predicates that are hidden from the observer. For the case of complex, composite attributes, such a choice is a very difficult task [15] and makes it possible to eliminate SVM in order to work with such data. Second, since the execution of all classes of queries at a single moment of time is generally rather unlikely, then, based on the vast experience of practical systems [2, 5, 8], partial attributes would prevail as the initial data. That, together with the first feature, increases by orders of magnitude the requirements to training samples for methods based on neural networks, as well as the overall computational complexity of their application [15]. The methods based on gradient boosting are also hardly applicable as a result of a sufficiently large number of elements in the set I, as a result of high requirements to the number of involved basic algorithms or their complexity [16]. Thus, among the well-studied and generic methods of machine learning there remains a Random Forest [15].

Random forest, as a machine learning ensemble method, in the context of the current work has a series of important characteristics: high resistance to partial and incomplete attributes, the simplicity of using predicates as decisive rules for building trees. Nevertheless, despite its merits, the method cannot be applied to the set of tuples in the form (i, k_i, Q_i^P) without some adaptation. The queries from one class or the queries that differ little with respect to time of cooperative implementation would be considered as completely different attributes. Consequently, without additional transformations, the logically interrelated attributes would be considered independent of each other. Such a simplification would reduce the generalizing capacity of the algorithm, as well as increase the costs of keeping a forest and the complexity of its construction [17]. To resolve this task, the modifications to the original method are proposed. The selected criterion of branching is the Gini importance [18]

$$\Gamma(\vec{p}) = \sum_{t=1}^{m} p_i(1 - p_i),$$

where vector p is composed of m probabilities, occurring in a subset of the learning set. Thus, a set of values k_i for any query under consideration is considered to be equivalent to the coefficient before an index, explicitly laying a relationship between the competition of queries and the choice of a branching condition in the synthesis of a decisive tree. It is the first modification of the classical method of random forest. The use of an additional weighting factor makes it possible to give priority to those attributes, which are an integral part of concurrently executing expressions.

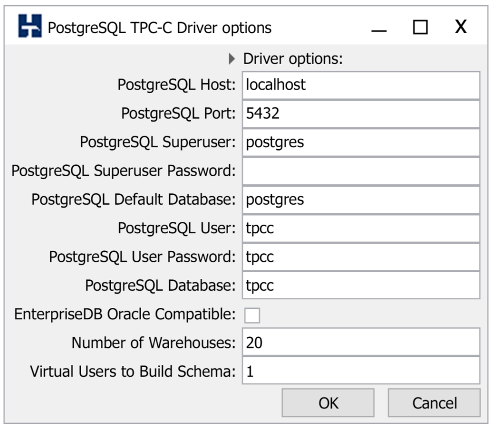
As a result, there is a decrease in the impact of attributes from the expressions, which assume a smaller share of competition, all other conditions being equal. In a general case, such an approach should make it possible to improve resistance to retraining. However, it does not limit the sensitivity of the model in terms of the theory of machine learning. For the formation of a forest, one would select the decisive rules from a subset of attributes of low competitive, but frequently performed, parallel queries. However, such rules will be clipped during the execution of a CART procedure, owing to the small magnitude for the base value of the Gini coefficient [18]. Consequently, the synthesis of a single decisive tree will not be dominated by the choice of logical rules, correspondent to queries with the highest share of sequential calculations, or the most frequent identical queries. Thus, one solves the task on the normalization of original data that increases the accuracy of the resulting ensemble [15].

The method of random subspaces implies a random selection of a subset of attributes for the synthesis of a single tree. This is a rather reasonable strategy for cases when there are no *a priori* data about the structure of a feature space, their relationships, and it is not possible to generate an intermediate representation of attributes while segregating the redundant attributes. However, for the case of a constructed tuple (i,k_i,Q_i^P), it is obvious that there are links between the subsets Q^P from the same class i as between the components of a holistic, semantically and syntactically bound expression in the SQL language. Thus, an arbitrary choice at explicitly expressed subclasses of attributes would increase the computational complexity due to the larger number of steps at the stages of sorting the subspaces and clipping the retrained trees from the overall ensemble. In the context of the proposed representation of precedents, a second modification of the random forest method implies the application of two iterations of a random subspace method. The original method assumes the homogeneity of a set of attributes. Generalizing the texts of SQL queries into a subset of classes $\{i\}$ generates an additional *a priori* information about each attribute. And the proposed modification is to use two stages of sampling when generating a subspace of attributes of an individual tree. The first stage involves the construction of a subset of all subclasses of competitive running queries i. At the second stage, one directly selects the attributes, limited by subclasses from the first stage and the current query. Using two iterations of a random subspace method reduces the probability of forming the retrained trees. There are reduced requirements to the volume of a training sample and to the computational complexity in assembling the trees obtained. In some cases, in general, this makes it possible to calculate the model of a forest, which cannot be obtained due to the significant number of invariants in the absence of the dedicated query classes.

The chapter presents the constructed method for the calculation of parameters for a set-theoretic model of the process of control over competitive access. The proposed method employs statistical data for the period of work of a relational DBMS to calculate its numerical model by machine learning methods. In contrast to existing analogs, target parameter serves time, not its surrogate representation, any terms and conditions in the text of the query and mutual competitiveness.

5. Results of numerical experiments on estimating the effectiveness of the constructed method for calculating the parameters for a set-theoretical model of the process of control over competitive access of queries in a DBMS of the relational type

A machine learning theory is largely an applied discipline and requires the computational validation of predicted results when running an experiment. To construct an experimental sample, we used the PostgreSQL DBMS and a standard TPC-C test with an OLTP load based on the software for imitational simulation and load testing with the open-source code HammerDB. Fig. 3 and 4 show source data for the organization of a layout, as well as the simulation model parameters, used in a computing experiment.

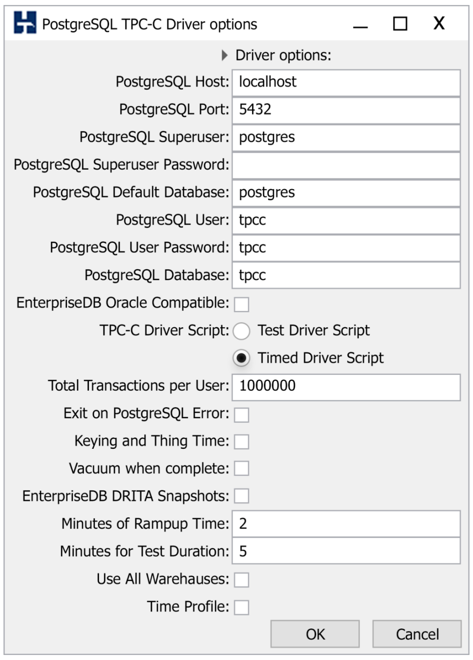


PostgreSQL TPC-C Driver options

Driver options:

PostgreSQL Host:	localhost
PostgreSQL Port:	5432
PostgreSQL Superuser:	postgres
PostgreSQL Superuser Password:	
PostgreSQL Default Database:	postgres
PostgreSQL User:	tpcc
PostgreSQL User Password:	tpcc
PostgreSQL Database:	tpcc
EnterpriseDB Oracle Compatible:	☐
Number of Warehouses:	20
Virtual Users to Build Schema:	1

Fig. 3. Source data for the organization of a database layout



PostgreSQL TPC-C Driver options

Driver options:

PostgreSQL Host:	localhost
PostgreSQL Port:	5432
PostgreSQL Superuser:	postgres
PostgreSQL Superuser Password:	
PostgreSQL Default Database:	postgres
PostgreSQL User:	tpcc
PostgreSQL User Password:	tpcc
PostgreSQL Database:	tpcc
EnterpriseDB Oracle Compatible:	☐
TPC-C Driver Script:	○ Test Driver Script
	● Timed Driver Script
Total Transactions per User:	1000000
Exit on PostgreSQL Error:	☐
Keying and Thing Time:	☐
Vacuum when complete:	☐
EnterpriseDB DRITA Snapshots:	☐
Minutes of Rampup Time:	2
Minutes for Test Duration:	5
Use All Warehouses:	☐
Time Profile:	☐

Fig. 4. Source data for a model of DBMS competitive operation

Ensuring the logging of incoming requests was performed by means of DBMS. The content of the configuration file in the installed copy of software was changed in accordance with Table 1.

Table 1

Installed parameters for DBMS configuration

Parameter title	Value
log_destination	'csvlog'
log_min_duration_statement	0
log_line_prefix	'%m [%p]: [%l-1] user=%u,db=%d,app=%a, client=%h %c: %l '
log_duration	on
log_statement	'ddl'

To construct a training set, to build a model and to rank the parameters, we developed a software implementation of the solutions, proposed in the current work, in the programming language C# using the Accord.NET library. To exclude the influence of logging to the basic operation of the system on processing the incoming queries, the work logs were sent to a separate disk in a PC RAM. To accomplish this task, we have used the most recent freely-available published version of the software SoftPerfect RAM Disk.

The library Accord.NET is often used in research into the application of machine learning algorithms [15, 16], including the studies into similar topics [17, 19]. The closest analogs to this library are the commercially-available suite MATLAB, the R language implementation, as well as the free library scikit-learn. A key feature of the Accord.NET library, compared to those systems, is the affordability of application of reflection in the .NET environment. A possibility to substitute individual methods within the used libraries makes it possible to easily make changes to the work of the algorithms that were already implemented in the library. There is no need to directly borrow a source code or to reconstruct existing algorithms independently. In addition, compared to the system MATLAB, the library of Accord.NET is free for use. In comparison with the library scikit-learn, the most common in the applied tasks on machine learning, it has a several-time greater productivity [15]. Among other things, the resulting software module can be simply deployed in the industrial environment, because it is a standard program, not dependent on third party software [19].

The figures from the experiments reported here correspond to running a test at a PC with the processor AMD FX-4330, 4 GHz, 16 GB RAM, and the Intel SSD drive SC2KW24. The test was directly run by the simulated 4, 8 and 12 competing users. The selected number of users is based on the results from available experimental studies that estimated the impact of multithreading on a query execution time [12, 13, 14]. The experiments at other hardware-computing platforms did not reveal any significant changes in the frequency of errors in a control sample.

The main criterion of efficiency of the models obtained by using the methods of machine learning, is the average error rate at control [18]. For autonomous DBMS, not important is an individual query execution time, what matters is a possibility to estiamte the total time required to execute many queries [12, 14, 19]. This feature has simplified the study of the results obtained. Thus, we selected, as a loss function, the module of difference in the total query execution times from the precedents in a control sample and the values for regression with respect to the amount of times in the control sample. Dividing the precedents into the training and control samples was performed in a pseudorandom fashion, based on the Vortex Mersenne MT19937. To assess the degree of retraining, we executed control at increasing length of learning at hold-out CV. The results obtained are given in Table 2.

Table 2

Results of computational experiment

Number of users	Training sample size	Error frequency at control
4	10,000	0.0012545776
4	50,000	0.0011904734
4	500,000	0.0011912891
8	10,000	0.0012336765
8	50,000	0.0012248468
8	500,000	0.0011715882
12	10,000	0.0011814117
12	50,000	0.0011349580
12	500,000	0.0011483744
4*	500,000*	0.8124643188*
8*	500,000*	0.8315736365*
12*	500,000*	0.8925198401*

The results marked in the table with symbol "*" were obtained by using the non-modified implementation of a random forest method from the Accord.NET library, which makes it possible to compare it with the devised approach. For simplicity, all tests employed the fixed test sample the size of three million queries; in this case, we discarded the very first and the last queries from the total number, coinciding with the beginning and end of test runs. For the non-modified method, we used the maximum size of the training sample, because decreasing it did not have any significant impact on the frequency of errors that would indicate the model's retraining.

The reported results from computational experiments suggest several main conclusions. First, the proposed approach to the formation of an attributive description makes it possible to formalize statistical data on SQL queries in the format suitable for calculating numerical models by machine learning methods. Second, the experiment confirms the effectiveness of the constructed set-theoretical model and the parameter calculation method for the model of control over competitive access.

6. Discussion of results from modeling and conducting a computational experiment

The magnitudes for errors on control samples, obtained in the course of a computing experiment, testify to the suitability of the proposed model, at least for the application to relational DBMS with an OLTP workload, similar to that simulated by a TPC-C test. The changes introduced to the original method could significantly reduce the frequency of errors even at smaller sizes of training samples and avoid both undertraining and retraining.

Estimating the results from a computational experiment suggests:

– increasing the sample size for the proposed algorithm does not lead to a compromised accuracy, that is no retraining was observed in the experiment;

– the magnitude of an error is significantly less than one second, which makes it possible to improve the accuracy by incorporating the derived model into other ensembles as a base algorithm;

– the proposed modifications to the original variant of the random forest method improve accuracy of the obtained parametric model.

The results from a computational experiment on a simulation models make it possible to draw a conclusion on the feasibility of the application of the devised approach to modeling the time of query execution in relational DBMS with competitive access. The use of time as the target parameter makes it possible to exclude artificial intermediate representations and enhances the accuracy of the resulting estimates.

This work describes a new approach to representing the process of control over competitive access in relational databases, based on the set-theoretical representation of attributive descriptions of queries. Based on the introduced relationships, we have derived and proven the functional dependence between an execution time and the texts of queries that are calculated in parallel by SQL, atomic data in the memory of DBMS.

The shortcomings of the proposed approach include specific requirements to a set of queries received by the system. First, constructing a model by using machine learning methods axiomatically limits the minimum size of a training set, depending on the complexity and diversity of queries processed. In general, this requirement stems from and reflects the applied aspects in the application of the Vapnik-Chervonenkis theory. In practical terms, this means the inability to predict in advance at which size of the training sample the error value in cross-validation would reach the magnitudes that are acceptable in the context of the ultimate task. Accordingly, unknown are the time intervals over which one must monitor the system. Second, the system must not significantly change the characteristics of its operation over timeframes within which the attributes are acquired. Random Forest, which is one of the strongest methods of machine learning [15], possesses a good generalizing capacity. Because the base primitive is the ensemble of piecewise-constant functions, the derived parametric models could approximate arbitrary functional dependences. However, changing the characteristics for the operation of DBMS in time would lead to noise in the source data and provoke a rise in the bias of responses from the final algorithm. As a result, the accuracy of the obtained model tends to decrease towards magnitudes that rule out its practical application.

The promising directions to advance the current study include:

– construction of similar models for use in non-relational databases;

– design of an attributive query description so that it includes information about durable and composite transactions;

– experimental estimation of the applicability of other methods of machine learning, a possibility to improve the ensemble algorithms and the algorithms for building decisive trees.

7. Conclusions

1. The current work proposes a model for the process of control over competitive access of queries in relational databases based on the set-theoretical representation of an attributive description of a query. In contrast to existing ones, the suggested model applies conditions for access to data from the texts of queries in order to calculate the share of sequential computations. The proposed approach focuses on the direct estimation of a query execution time in a competitive environment, without the use of intermediate units for the cost of individual operations.

2. A method for calculating the parametric model of data access in relational DBMS has been constructed based on Random Forest. The method's feature is the use of the share of sequential computations as a weighting factor when choosing decisive rules, as well as the application of equivalence classes for similar queries in the procedure of selection of a random subspace of attributes.

3. Conducting numerical experiments under an OLTP workload indicates the model's adequacy and the applicability of the devised method for tasks on estimating a query execution time. However, the use of a machine learning method imposes a series of constraints on the applicability of the results obtained. First, over the time intervals that are used to gather statistical data and to directly estimate the time of query execution, interaction with a DBMS must be quasi-stationary. In other words, the superposition of changes in the information stored in the database and changes in the characteristics for incoming queries must be such that there is a technique to discard no more than half of all queries and to describe the system performance as a stationary process. In this case, the complexity of a formal representation of query processing in the form of a stationary process makes it more practical to experimentally estimate the applicability of the model derived for each system. Second, there is no any technique to determine *a priori* the time interval over which one must monitor and collect statistics on query execution in order to obtain an adequate model. The second constraint is directly related to the problem on assessing the dimensionality by Vapnik-Chervonenkis. There are no reasonable tools to theoretically assess the applicability of the proposed method and to calculate the minimum monitoring interval for a particular database, somebody must acquire them experimentally. In turn, changing the data stored and the queries processed would over time change these estimates in an unpredictable way. Therefore, the practical implementation of the proposed method must include an auxiliary algorithm for checking the adequacy of the calculated parametric model prior to its use in the algorithms for automating an autonomous database.

References

1. Gromey D. D., Lebedenko E. V. Automation of data distribution for the DBMS with competitive requests // Modern informatization problems in simulation and social technologies Proceedings of the XX-th International Open Science Conference. Yelm, WA, USA, 2015. P. 167–173.
2. Chaudhuri S., Narasayya V., Ramamurthy R. Exact cardinality query optimization for optimizer testing // Proceedings of the VLDB Endowment. 2009. Vol. 2, Issue 1. P. 994–1005. doi: https://doi.org/10.14778/1687627.1687739
3. Borchuk L. E. Stoimostnye optimizatory dlya SUBD: vchera i segodnya // Otkrytye sistemy. SUBD. 2016. Issue 1. P. 36–39.

4. ERMIA / Kim K., Wang T., Johnson R., Pandis I. // Proceedings of the 2016 International Conference on Management of Data – SIGMOD '16. 2016. doi: https://doi.org/10.1145/2882903.2882905
5. Neumann T., Mühlbauer T., Kemper A. Fast Serializable Multi-Version Concurrency Control for Main-Memory Database Systems // Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data – SIGMOD '15. 2015. doi: https://doi.org/10.1145/2723372.2749436
6. Rheinländer A., Leser U., Graefe G. Optimization of Complex Dataflows with User-Defined Functions // ACM Computing Surveys. 2017. Vol. 50, Issue 3. P. 1–39. doi: https://doi.org/10.1145/3078752
7. Cost-Model Oblivious Database Tuning with Reinforcement Learning / Basu D., Lin Q., Chen W., Vo H. T., Yuan Z., Senellart P., Bressan S. // Lecture Notes in Computer Science. 2015. P. 253–268. doi: https://doi.org/10.1007/978-3-319-22849-5_18
8. Automated Demand-driven Resource Scaling in Relational Database-as-a-Service / Das S., Li F., Narasayya V. R., König A. C. // Proceedings of the 2016 International Conference on Management of Data – SIGMOD '16. 2016. doi: https://doi.org/10.1145/2882903.2903733
9. Mozafari B., Niu N. A Handbook for Building an Approximate Query Engine // IEEE Data Eng. Bull. 2015. Vol. 38, Issue 3. P. 3–29.
10. Yoon D. Y., Niu N., Mozafari B. DBSherlock // Proceedings of the 2016 International Conference on Management of Data – SIGMOD '16. 2016. doi: https://doi.org/10.1145/2882903.2915218
11. Radhika G., Chhabra P., Kumari R. Consistency models in distributed shared memory systems // International Journal of Computer Science and Mobile Computing. 2014. Vol. 3, Issue 9. P. 196–201.
12. Arulraj J., Pavlo A., Dulloor S. R. Let's Talk About Storage & Recovery Methods for Non-Volatile Memory Database Systems // Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data – SIGMOD '15. 2015. doi: https://doi.org/10.1145/2723372.2749441
13. Query-based Workload Forecasting for Self-Driving Database Management Systems / Ma L., Van Aken D., Hefny A., Mezerhane G., Pavlo A., Gordon G. J. // Proceedings of the 2018 International Conference on Management of Data – SIGMOD '18. 2018. doi: https://doi.org/10.1145/3183713.3196908
14. Arulraj J., Pavlo A., Menon P. Bridging the Archipelago between Row-Stores and Column-Stores for Hybrid Workloads // Proceedings of the 2016 International Conference on Management of Data – SIGMOD '16. 2016. doi: https://doi.org/10.1145/2882903.2915231
15. Nithya B. An Analysis on Applications of Machine Learning Tools, Techniques and Practices in Health Care System // International Journal of Advanced Research in Computer Science and Software Engineering. 2016. Vol. 6, Issue 6.
16. Hogland J., Anderson N. Function Modeling Improves the Efficiency of Spatial Modeling Using Big Data from Remote Sensing // Big Data and Cognitive Computing. 2017. Vol. 1, Issue 1. P. 3. doi: https://doi.org/10.3390/bdcc1010003
17. Multi-source data analysis and evaluation of machine learning techniques for SQL injection detection / Ross K., Moh M., Moh T.-S., Yao J. // Proceedings of the ACMSE 2018 Conference on – ACMSE '18. 2018. doi: https://doi.org/10.1145/3190645.3190670
18. Janitza S., Celik E., Boulesteix A.-L. A computationally fast variable importance test for random forests for high-dimensional data // Advances in Data Analysis and Classification. 2018. Vol. 12, Issue 4. P. 885–915. doi: https://doi.org/10.1007/s11634-016-0276-4
19. Data Exploration with SQL using Machine Learning Techniques / Cumin J., Petit J., Scuturici V., Surdu S. // Open Proceedings. 2017. P. 96–107. doi: http://doi.org/10.5441/002/edbt.2017.10