Iteration of database relations for queries of special form

Sergey Zykin, Andrey Poluyanov
Sobolev Institute of Mathematics, Novosibirsk, Russia

Abstract. In this paper, the material for compiling a plan of performing "join-selection-projection" queries to relational databases is systematized. This is necessary for counting the number of iterations in the query execution algorithms. The presented topic is relevant for a considerable period of time. However, every time there are new requirements for queries, new methods of data presentation, etc. This leads to the necessity of resolving old problems. Many papers were devoted to various aspects of the optimization of queries to relational databases. Thus, it is possible to define two main directions. They are logical optimization of the queries and physical optimization of the queries execution. The basis of logical optimization is heuristics about the order of relational operations. However, more effective solution to the optimization problem is minimization of input-output operation quantity during the query execution. The approach to the development of the query execution plan on the basis of estimators for input-output operations for conventional and specialized methods of data storage is offered in this paper. The expediency of transformations, use of indexes etc. are determined by estimators.

Keywords: database, joint storage, query, estimator

1. Introduction
The single iteration usually used as a unit of measurement of algorithm difficulty. However, the difficulty of the iteration itself usually is not considered. In this paper the iteration scheme of relations with using results obtained earlier is offered: the basic properties of queries logical optimization [1]; methods for transforming relational queries [2]; optimization and design infrastructure for efficient and correct SQL transformations [3]. The physical storage design methods of data and the calculation of the corresponding estimates are presented in the following works: the basics of query optimization in relational system [4]; development of database structures [5]; creating an optimal query execution plan [6]; the use of statistical estimates when performing queries [7]. (the references relate only to the summarized and systematized publications).

The necessity to study the problem of optimizing database queries periodically arises in the analysis of ad hoc queries, or special ways of storing data as well as in solving unusual application problems. Let us consider only some of them, which are the closest to the subject of this paper. An area of query optimization such as e-commerce is considered in [8]. In this work at researching the message system in the instant messaging system based on the multi-language e-commerce platform in order to design the instant messaging system in multi-language environment and exhibit the national characteristics based information as well as applying national languages to e-commerce. Oracle memory database is adopted as user message buffer, so the messages transmitted and accepted in user communication process can be directly read from the memory and there is no need to read and write in the magnetic disk for each time, thus improving system throughput capacity and user concurrency.

Currently considerable interest is taken in the methods of using the cache [9, 10] to reduce the execution time of the query. Consideration of the specific problem in this case allows more efficient use of the cache contents. In work [9] the use of cache in a distributed database is analyzed. The problem of the time required to receive information about the cache contents from multiple sources is
investigated in this paper. The model of distributed query optimization with a cache location between
the local optimizer and the local database is proposed. Whenever a request is given to the local
optimizer, the optimizer first checks the local cache. The proposed model to optimize the cost of work
is based on four different factors: the distance to the server, server capacity, server load and the current
length of the queue at the node where the query should be executed. In our work on the basis of these
results we propose the method of using estimators in a distributed system.

In paper [10] a new method of client-side data caching for relational databases with a central server
distant clients is proposed. Data are downloaded to the client cache, in which the results of
previous queries are stored. The query result is a special form called a "universal relational query". The
basis of such queries is three main operations of relational algebra: selection, projection, and
natural join. A subsequent query may be executed in a client’s local cache. For this truth spaces of the
logical restrictions in a new user’s query and the results of the queries execution in the cache are
compared. Such a comparison can be performed analytically without the need of additional queries on
a server. This method may be used to define the lack of data in the cache and execute the query on the
server only for these data. The solution to this problem with the use of analytical methods which
distinguishes the research results shown in this paper from those ones obtained before. Our research is
complemented by estimators of cache update if necessary: all contents of the table or its part will be
updated in cache.

The problem of managing complex database systems will not lose its relevance in the foreseeable
future. Such systems include a set of different hardware and software resources of a large number of
configuration options that require a lot of experience, and it is time consuming. In work [11] a
multilevel control system "MAG" is presented. This system can be used to monitor, analyze, predict
configuration parameters, and it allows adjusting these parameters of database management. This
system has the following features: a prediction of system performance depending on the set
parameters; analysis and identification of the source of the problem. The method of estimation
proposed in this paper can be embedded in "MAG" control system. In addition, there is a necessity for
peculiarities of the storage structure in the system to be taken into account.

In work [12] the data integration for distributed query execution is considered. The main focus is on
the algorithms for processing heterogeneous and distributed data. The conclusion is that the well-
known universal algorithms are ineffective in such environment. This work considers the possibility of
joint storage of data in the form of clusters. In our paper we consider a generalization of the clustered
storage environment which allows using the projection operations instead of the operations of natural
join of an arbitrary set of database relations being stored together.

As the initial form of the query representation we shall consider a "universal" relational query:

$$\pi_X (\sigma_C (R_1 \bowtie R_2 \bowtie \ldots \bowtie R_k))$$

(1)

where $\pi$, $\sigma$, $\bowtie$ are the projection, selection and join operators; $X$ is the set of attributes in the resulting
relation; $C$ is the logic expression. The attribute names in (1) have an extended format: <relation name>.<attribute name>. This allows an unambiguous determination of attribute affiliation in $X$ and in
$C$, and the natural join operation is converted into Cartesian product. Expression (1) covers a wide
class of relational queries, but not all possible queries.

In [1, 2] the graph decomposition algorithm requiring transformation of expression $C$ to the
conjunctive normal form is considered. The heuristics proposed in [2] leads to the need of iterative
procedures over two not small relations that requires more input-output operations than a pair-by-pair
iterative procedure. Besides, opportunities of physical optimization including the use of indexes are
not taken into account. In [13, 14] queries are optimized with the use of the index, but the lack of rules
for determining the usefulness of the index could lead to increased costs. The formulated problems can
be effectively resolved with the use of estimators.

First of all we shall consider the environment of storage and data processing. It is supposed that there
is a distributed database on several servers connected by reasonably high-speed channels. The offered
technique can be used without changes for one database server and one workstation.

2. Algorithmic basis of the method
For further treatment it is necessary to consider the basic algorithm of iterating (BAI). Let the sequence of the relations \( R_1, R_2, \ldots, R_k \) is fixed by some rules; \( D_i \) is the viewing domain for \( R_i \) in compliance with the restrictions \( C \), and current restrictions from the relations \( R_j, j<i \). \( D_i \) contains all \( R_i \) tuples in the worst case, however, the use of indexes and links between relations or clustering can essentially reduce \( D_i \). The domain \( D_i \) for a single restriction and for one index is defined by the set of initial and final interval values. The number of intervals is determined by the rating: the cost of the search of the next interval by the index should not exceed the cost of viewing the relation before the start of the interval. The domain \( D_i \) for the set of restrictions with the use of inverted file is the set of references on database records. In addition to the need of splitting on the intervals there is a need to determine the feasibility of using the indexes of fields. The calculation algorithms of \( D_i \) in BAI on the expression \( C \) and the data access method (indexations, links, clustering etc.) are assumed to be given.

Let us consider the basic algorithm of iteration:

\[
\text{<calculation } D_i \text{>}
\]

\[
<\text{search of the first tuple } t[R_i] \in D_i>
\]

\[
i = 1
\]

\[
r = \emptyset
\]

\[
<\text{cycle}>
\]

\[
\text{if } t[R_i] \in D_i \text{ then}
\]

\[
\text{if } i = k \text{ then}
\]

\[
\text{if } C(t) = \text{TRUE} \text{ then}
\]

\[
r = r + t
\]

\[
\text{skip } D_i
\]

\[
\text{end if}
\]

\[
i = i + 1
\]

\[
<\text{calculation } D_i \text{>}
\]

\[
<\text{search of the first tuple } t[R_i] \in D_i>
\]

\[
\text{end if}
\]

\[
\text{else}
\]

\[
\text{if } i = 1 \text{ then}
\]

\[
<\text{exit from cycle}>
\]

\[
\text{else}
\]

\[
i = i - 1
\]

\[
\text{skip } D_i
\]

\[
\text{end if}
\]

\[
<\text{the end of a cycle}>
\]

\[
<\text{the end of the algorithm}>,
\]

where \( r \) is the resulting relation; \( t \) is the tuple that is defined on attributes \( X (1) \) and on attributes \( C \); skip \( D_i \) is the transition from the current tuple to the next tuple.

**Remark 1.** The relation \( R_i \) in the algorithm refers to as the leading for the relations \( R_j, j>i \).

**Remark 2.** The submitted algorithm is reasonably simple and can be transformed into recursive algorithm. However, this will bring to an increase of operative memory costs by the system software. Let \( \phi(1, 2, \ldots, k; l) \) is the iteration cost of relations \( R_1, R_2, \ldots, R_k \) sequence on server with number \( l \), \( \psi(1, 2, \ldots, k; l) \) is the cost of result formation of iteration on server \( l \). The general costs of iteration procedure are defined by the sum: \( \phi + \psi \), where \( \phi \) includes the quantity of input operations and costs of initial data transfer on server \( l \), \( \psi \) is the quantity of output operations on server \( l \).

Let us consider the common schema of construction the quazyoptimal query realization plan:

1. Pair of relations \( R_i, R_j \) and server \( l \), for which:

\[
\phi(i, j; l) + \psi(i, j; l) \rightarrow \min\{(i, j; l)\}
\]

is chosen.

1.1. For each pair the leading relation is defined. If \( \phi(i, j; l) \leq \phi(j, i; l) \), the leading relation is \( R_i \); otherwise the leading one is \( R_j \).
1.1.1. For the driven relation \( R_i \) a preliminary iteration is necessary if:
\[
\varphi(j; l) + \psi(j; l) + \varphi(i, m; l) < \varphi(i, j; l),
\]
where \( R_m \) is the resulting relation after the \( R_i \) iteration.

1.2. If there has been non-

\[
\mathcal{C} = \bigvee \bigwedge_{j=1}^{k_i} \mathcal{C}_{i_j},
\]
where \( \mathcal{C}_{i_j} = \langle \text{attribute} \rangle \langle \text{operation} \rangle \langle \text{attribute} \rangle \) or \( \mathcal{C}_{i_j} = \langle \text{attribute} \rangle \langle \text{operation} \rangle \langle \text{constant} \rangle \).

\( \mathcal{C}_{i_j} \) can contain expressions on attributes, if they allow the normalization [3], and the functions \( \varphi \) and \( \psi \) can be calculated for them.

Remark. More optimum sequences of the query execution (1) are possible in some cases. The choice of the submitted schema is stipulated by the following:

- The complete formation of optimum schema is an NP–complex problem.
- Estimator calculations for the completely optimum schema by the estimator \( \psi \) are executed. This estimator contains errors that will cause an increase of calculation errors.
- The costs of the quazyoptimal schema search have the second orders which are quite acceptable in comparison of the processor speed and the data transfer speed.
It is necessary to expand expression $C$ by constraints before its transformation to DNF. Expression $C$ must be expanded by the equivalent conditions, for example:  
\[ A_1 > A_2 \land A_2 = A_3 \equiv A_1 > A_2 \land A_2 = A_1 \land A_1 > A_3. \]
These operations will allow essential reduction of the intermediate computational complexity.

Let us consider the realization schema of restrictions $C$ in the iteration process of the relations.

Restrictions $C_{ij}$ in (3) are active under the iteration procedure, if all their attributes belong to the considered relations. All other $C_{ij}$ are replaced by the value $TRUE$. The active restrictions are used for the estimators $\varphi$ and $\psi$ computation and for tuples filtration in the resulting relation. When the current iteration terminated, the active restrictions $C_{ij}$ from $C$ should be deleted, if they have the $TRUE$ value on all tuples of the iteration result. Other $C_{ij}$ will participate in the next iterations as active.

**Remark.** If all $C_{ij}$ in any conjunctive group are deleted, then the expression $C$ will be equivalent to the $TRUE$ value and on the next iterations the Cartesian products execution will be needed. If the $C_{ij}$ restriction is deleted, then the matching attributes can be deleted, if the set $X$ does not contain these attributes.

The iteration result for query (1) will be specified by $r, r_2 = R_1 \times R_2 \times \ldots \times R_n, r_1 = \sigma_C(r_2), r_0 = \pi_X(r_1)$.

**Theorem 1.** The iterative algorithm and the correct realization of the restriction schema form the resulting relation: $r = r_0$.

**Proof.** 1. Let a tuple $t \in r$ and $t \neq r_0$ are exist. The tuples $\{t[R_i]\}$ in each relation are exist, and they are used for tuple $t \in r$ generation. Let $t_1 \in \{t[R_1]\} \times \{t[R_2]\} \times \ldots \times \{t[R_k]\}$, hence $t = \pi_X(t_1)$. Under the assumption of $C(t_1) = FALSE$. Hence, in each conjunctive group $C_{g(i)}(t_1) = FALSE$ exists, for each $1 \leq i \leq n, j(i)$ is the component number in the group. The restrictions $C_{g(i)}$ under the condition after each iteration cannot be deleted. Hence, the tuple $t_1$ at least on the last iteration will be deleted, and this contradicts the initial assumption.

2. Let a tuple $t \notin r$ and $t \in r_0$ are exist. Let $C_j$ is the restriction of $j$-s iteration, where the unused restrictions are replaced by the value $TRUE$. Obviously, that is executed by the following identity:  
\[ C(u) \rightarrow C(u) \equiv TRUE \]
for arbitrary iteration $j$ and arbitrary tuple $u$, including tuple $t$. Let us assume that $C(t) = TRUE$. Hence, there are no iterations on which the tuples join $\{t[R_i]\}, i = 1, 2, \ldots, k$ would be removed, and this contradicts the initial assumption. This completes the proof.

### 4. Model of data representation

As the implementation of the context we use data representation in the form of the Table of Joins, which is a modification of relation data representation [15].

Let us consider the transformation of relational database: $R = \{R_1, R_2, \ldots, R_k\}$ to the Table of Joins ($TJ$) has a scheme $(S, g)$, where $S$ is a scheme that defined on the set of attributes $U = \{A_1, A_2, \ldots, A_n\}$, $g$ is a vector length $k$. We consider the method of forming tuples $t \in s$, where $s$ is the implementation of the $TJ$ table. Consider all possible combinations without repetition of relations $R_1, R_2, \ldots, R_k$ satisfying lossless join property [1]. This set of relations will be called a "context". Let the set of relations $C = \{R_{m(1)}, R_{m(2)}, \ldots, R_{m(p)}\}$ be the context, where $m$ is an integer array of $p$ numbers of relations of the current combination and $c'$ is the implementation of context constrained by selection $\sigma_f$ with logical formula $F$:

\[ c' = \sigma_f(R_{m(1)})[W_{m(1)}] \times R_{m(2)}[W_{m(2)}] \times \ldots \times R_{m(p)}[W_{m(p)}] \]

(4)

For each tuple $u \in c'$ we form a tuple $t$ by the following rules: if the attribute $A_1$ belongs to at least one relation of current join, then $t[A_1] = u[A_1]$, otherwise $t[A_1] = emp$, where $emp$ is an empty value. To each tuple we assign a bit vector $g(t) = (g_1(t), g_2(t), \ldots, g_k(t))$. If the relation $R_j$ is involved in the current join, then $g_j(t) = 1$, otherwise $g_j(t) = 0$.

We define a partial order on the tuples $t \in s$. 
Definition 1. A tuple \( t \in s \) is less definite or equal to a tuple \( t' \in s \) if for any attribute \( A \), the following condition holds: if \( t[A] \neq t'[A] \), then \( t[A] = \text{null} \) and \( g(t) \geq g(t') \), \( j = 1, \ldots, k \). In this case we write \( t \prec t' \), the tuple \( t \) is said to be subordinate to the tuple \( t' \) and both of these tuples are called comparable.

Note. If we assume that \( A_i \) is \text{null} in both tuples, then \( t[A_i] = t'[A_i] \).

In the representation \( s \), it is sufficient to store only the tuple \( t' \), because \( t' \) contains all subordinate tuples. On the next step we remove all subordinate tuples. Equality of \text{null} values in definition 1 gets rid of the subordinate tuples \( t \), which are derived from the same database tuples that \( t' \). The difference between \text{null} and \text{null} is that the first refers to indefinite attribute value, and the second refers to the absence of a corresponding tuple in the current join. It is obvious that the comparison operation \( < \) is transitive.

The final stage of constructing a representation \( s \) is to remove tuples for which \( F(t) = \text{false} \). In this case, the tuples \( t' \), for which \( t' \prec t \) and \( F(t') = \text{true} \) are excessive, because indefinite data in \( t' \) are defined in \( t \) and the user rejects them.

Suppose that \( R(L) = (R_{m_1}, R_{m_2}, \ldots, R_{m_p}) \), where \( L = (m(1), m(2), \ldots, m(p)) \), and \( m \) is an integer array of numbers of \( p \) relations. Let us define the operation of the projection on the \( s \).

Definition 2. Projection \( \pi_{R(L)}(s) \) is a set of tuples \( u[R(L)] \), defined on the set of all attributes of relations \( R(L) \), where for each \( u[R(L)] \) there is a tuple \( t \in s \) such that \( u[R(L)] = u[R(L)] \) and \( g_{m_i}(t) = 1, i = 1, 2, \ldots, p \). Logical constraint \( F(t) \) for each dimension is represented as a disjunctive normal form:

\[
F = F_1 \lor F_2 \lor \ldots \lor F_m,
\]

where each formula \( F_i \) is a conjunction:

\[
F_i = F_{i1} \land F_{i2} \land \ldots \land F_{ip},
\]

where \( F_{ij} \) is predicate comparisons of SQL. If any predicate \( F_{ij} \) is not defined on a tuple \( t \), then it is replaced with the value \text{true} in case this conjunct still has some predicates, otherwise it is replaced with \text{false}. This substitution allows us to remove from \( s \) the tuples for which some attributes are not defined yet or there are no tuples with coinciding values in other relations. This situation is also the subject of data analysis. After such substitution formula \( F \) can be either \text{true} or \text{false}.

Implementation of this property requires specific way of building \( s \). If it is generated starting from all relations of the context, continued by relation subsets and so on up to a single relation with tuples filtering by using formula \( F \), then in \( s \) there are tuples subordinate to those deleted in previous iterations. In addition, different sequences of deleting subordinate tuples may yield different results, as far as a specified tuple order is a partial order, not a strict linear one. Consequently, the shaping of Table of Join should be executed on the considered rule.

5. Scheme of queries realization

We shall consider a situation, when any \( R_i \) in one Table of Joins or one cluster [13] is represented.

Probably, the same relation is represented in (1) for several times under various extended names. Let us name such relations by the joint stored.

The one pass from Table of Joins or cluster or duplicated relation in this case is more advantageous. The BAI modification is necessary so that resulting relations in BAI could be formed.

Let \( R_j \) is the next relation when generating scheme of iterations. The relations are stored jointly with it

\[
R_{m_1}, R_{m_2}, \ldots, R_{m_p}
\]

which has not been iterated yet. For each \( R_{m(q)} \), \( q = 1, 2, \ldots, p \), we realize checking. If \( R_j \) and \( R_{m(q)} \) in the Table of Joins or the cluster are jointly stored and natural join conditions are specified for them, then \( R_{m(q)} \) should be added to iterative schema without generating a new resulting relation.

Otherwise, generation of a new resulting relation for \( R_{m(q)} \) is necessary, if:

\[
\varphi(n; l) < \varphi(m(q); l)
\]

where \( R_n \) is the iteration result of \( R_{m(q)} \). Further \( R_j \) instead of \( R_{m(q)} \) in (1) is used.

Remark 1. The Table of Joins, unlike the cluster, can contain any relations which have different common attributes. Hence, the relation number in one iteration can be larger for the Table of Joins.

Remark 2. If attributes of the restriction \( C_{ij} \) belong to different resulting relations, then this restriction is not activated.
6. Conclusion
We shall consider a situation, when any $R_i$ in one Table of Joins or one cluster [13] is represented. Probably, the same relation is represented in (1) for several times under various extended names. Let us name such relations by the joint stored.
The representation of $C$ in DNF permits one to unite several queries execution. The simultaneous occurrence of the several queries is usually due to preparation of one document, where the same data should be processed at various restrictions. In this case queries are united in groups, having common relations in (1). Groups of the queries with common relations are optimized and executed at first:

$$X = X_1 \cup X_2 \cup \ldots \cup X_n$$

and

$$C = C_1 \lor C_2 \lor \ldots \lor C_n,$$

where $n$ is the quantity of the queries in group, $X_i$ and $C_i$ are the attributes set and restriction of $i$'s query. Groups are separated after the exhaustion of the common relations. The process of disintegration is finished by forming the resulting relations, each of them being the response to a separate query.
The queries do not arrive simultaneously at the multiple data access. Then the system can suspend the group of queries execution and finish a new query execution at the same stage so that the total time of queries execution is reduced. In particular case:

$$C \rightarrow C_i \equiv TRUE, X_i \not\subseteq X,$$

where $C$ and $X$ are current restriction and attributes set at the stage of queries group execution, $C_i$ and $X_i$ are restriction and attributes set of a new query. Then a new query is connecting to the group and its cooperative execution is resumed.

For groups of queries their parallel execution on various servers will be realized [16]. Within the one group the parallelism is possible for various iterations, but it will cause an increase of intermediate results.

7. References
[1] Ullman J 1980 Principles of database systems (Stanford: Computer Science Press) p 379
[2] Maier D 1983 The theory of relational databases (Stanford: Computer Science Press) p 637
[3] Chaudhuri S 1998 An overview of query optimization in relational system ACM SIGACT-SIGMOD-SIGART symp. on Principles of database systems (PODS '98) pp 34–43 URL http://dx.doi.org/10.1145/275487.275492
[4] Hong J, He Z, Bell D 2010 An overview of query optimization in relational system Information Systems 35 pp 140–148 URL http://dx.doi.org/10.1016/j.is.2009.08.004
[5] Teorey T, Fry J 1982 Design of database structures (NJ: Prentice Hall, Upper Saddle River)
[6] Kuznetsov S D 1990 Optimal query plan generation in relational DBMSs Programming and Computer Software 2
[7] Chen L, Dobra A 2013 Histograms as statistical estimators for aggregate queries Information Systems 38 pp 213–230 URL https://doi.org/10.1016/j.is.2012.08.003
[8] Yang H, Chen X, Li J, Cao Y 2017 Design of Instant Messaging System of Multi-language E-commerce Platform IOP Conference Series: Materials Science and Engineering 231 pp 012023 URL http://dx.doi.org/10.1109/ICDCS.2013.21
[9] Kumar M, Batra N, Aggarwal H 2012 Cache Based Query Optimization Approach in Distributed Database International Journal of Computer Science Issues 9 pp 389–395
[10] Mosin S V, Zyklin S V 2015 Truth Space Method for Caching Database Queries Modeling and Analysis of Information Systems 22 pp 248–258
[11] Khattab A, Algergawy A, Sarhan A 2015 MAG: A performance evaluation framework for database systems Knowledge-Based Systems 85 pp 245–255 URL http://dx.doi.org/10.1016/j.knosys.2015.05.010
[12] Arebi P, Gonbadipoor N 2011 A Genetic Algorithm for Query Optimization in Database Grid by Dynamic Cost Estimation Computer Modelling and Simulation (UKSim) UKSim 13th International Conference pp 81–86 URL http://dx.doi.org/10.1109/UKSIM.2011.25
[13] Hursch J, Hursch C 1989 *Working with ORACLE version 6.0* (TAB Books) p 464
[14] Nield T 2016 *Getting Started with SQL* (O'Reilly Media) p 134
[15] Zykin S V 2010 Automation of the interface formation between multidimensional and relational representation of the data *In Book: Relational Databases and Open Source Software Development* ed Taylor J (New York: Nova Science Publishers) chapter 2 pp 43–66
[16] Dobra A, Garofalakis M, Gehrke J, Rastogi R 2009 Multi-query optimization for sketch-based estimation *Information Systems* **34** pp 209–230 URL https://doi.org/10.1016/j.is.2008.06.002

**Acknowledgments**

The work was supported by the program of fundamental scientific researches of the SB RAS № I.5.1., project № 0314-2016-0020.