Модел. и анализ информ. систем. Т. **22**, № **2** (2015) 248–258

UDC 519.987

Truth Space Method for Caching Database Queries

Mosin S. V., Zykin S. V.

Sobolev Institute of Mathematics of the Siberian Branch of the Russian Academy of Sciences, Pevtsova str., 13, Omsk, 644043, Russia

e-mail: svmosin@gmail.com, szykin@mail.ru

received March 1, 2015

Keywords: relational databases, caching, truth space

We propose a new method of client-side data caching for relational databases with a central server and distant clients. Data are loaded into the client cache based on queries executed on the server. Every query has the corresponding DB table – the result of the query execution. These queries have a special form called "universal relational query" based on three fundamental Relational Algebra operations: selection, projection and natural join. We have to mention that such a form is the closest one to the natural language and the majority of database search queries can be expressed in this way. Besides, this form allows us to analyze query correctness by checking lossless join property. A subsequent query may be executed in a client's local cache if we can determine that the query result is entirely contained in the cache. For this we compare truth spaces of the logical restrictions in a new user's query and the results of the queries execution in the cache. Such a comparison can be performed analytically , without need in additional Database queries. This method may be used to define lacking data in the cache and execute the query on the server only for these data. To do this the analytical approach is also used, what distinguishes our paper from the existing technologies. We propose four theorems for testing the required conditions. The first and the third theorems conditions allow us to define the existence of required data in cache. The second and the fourth theorems state conditions to execute queries with cache only. The problem of cache data actualizations is not discussed in this paper. However, it can be solved by cataloging queries on the server and their serving by triggers in background mode.

The article is published in the author's wording.

Introduction

In this paper we discuss the use of data views that are cached on the client's computer. We assume client-server environment with server based Relational Database (RDB). The cache stores query results in order to provide maximum usage of saved data in subsequent queries.

The purpose of this paper is to study the problem of building and using data views on a client's computer. This problem is similar to query optimization because it aims to decrease the data transfer from a Database server. Cached data is actively used in Database Management Systems (DBMS), but mostly it is only repeating use of data written in cache, without any prior data analysis aimed to define any partial or combined use. The DBMS can only avoid requesting blocks of data from external devices, while serving the query, if they are present in the cache. So, only block numbers are analyzed, not their information.

This paper is based on the results, obtained in [1]. We have removed a constraint that limits the choice of attributes for Intermediate Data View (IDV) and also made some generalizations on multiple IDVs case. The second section provides an overview of publications on the subject of this paper. We have considered only those closest to the problem at hand. In the end of this section we present an example to explain our approach for solving the problem.

In the third section we provide a formalization of the problem. We also present the approach for removing the uncertainties in database queries and present auxiliary properties and definitions that are used later.

The main results are presented in Section 4. We research the possibility of using cached results of previous queries when performing a new one. The results are presented in the form of theorems.

One of the directions of further researches is combining this technology with efficient data manipulations approaches. Those can be performed with a help of Graphical Processors. GPUs gain serious attention for big data processing [2, 3, 4, 5]. Such processor architectures have big potential in multitasking and can reach enormous speeds compared to CPUs, especialy if the operations are performed similarly on different parts of data.

1. Existing solutions overview and comparison

One of the main research problems of queries optimization to database is the construction of an optimal query plan. Queries are transformed without content analysis of the database content and cache. In other cases, this information is taken into account for calculation statistical estimations to improve physical access to data. These problems are the subject of many studies, but they are outside of our approach.

A lot of publications are devoted to the problem of cache content management. For example, the heuristic algorithms update the cache with regards to user data [6], storage of important user queries in a cache [7], storage of the cached data on multiple servers [8]. This paper solves the problem of the best use of a cache for execution of the user's queries. Cache content management and the best use of the cache are two complementary problem.

The most similar to our work is paper [9]. The authors analyze conjunctive queries on data domains with predicates in the form of arithmetical comparisons, and present query computation algorithms using IDVs. In our paper, the special case of universal relational query is considered. It is a query on the Database relations, not particular domains. Although we had similar aims, the results obtained differ because of stated factors. In particular, there is no need to create any algorithms of data selection in our work, as they are replaced by Relational Algebra.

Another set of works describes processing scheme for event-driven continuous queries [10, 11]. In the proposed approach, query result caching is introduced to achieve a flexible way to share common operators among queries activated by unpredictable events. When a query is activated, an intermediate result generated for the query is stored into the cache area if it is expected to be reused by other queries. When other queries including the same operator are activated, they reuse the cached result if the cache includes reusable data.

The work [12] proposes methodology for functionally decomposing complex queries in terms of primitives so that multiple reuse sites are exposed to the query optimizer, to increase the amount of reuse.

In [13, 14] queries are separated into subqueries that can yield results from cached nodes. Then query result can be combined from several subqueries.

Using cloud storages one can afford a lot of computational resources for a short time to execute complicated queries effectively on large data with a help of virtual machine clusters [15, 16, 17, 18]. A set of reliable heuristic algorithms is used. Finally the authors carry out a series of experiments that show that their optimizations speed up the federated query evaluation process.

In papers [19], [20] similar to our problem is discussed. In this paper we make correspondence between cache contents and predicates. Cached data usage problem is resolved in terms of truth spaces. We compute the truth spaces of query results in the cache. It allows us to define records in IDV that can be used to form a new view and new SQL queries that will let us load missing data from Database server. The following example demonstrates suggested approach.

Example 1. Let's assume the following database schema fragment, which represents the University Study plan:
$R_1 = \mathit{Students}\ (\mathbf{Stud\_ID}, \mathrm{Stud\_Name}, \mathrm{Group})$
$R_2 = \mathit{Schedule}\ (\mathbf{Group}, \mathbf{Room\_ID}, \mathrm{Course})$
$R_3 = \mathit{Progress}\ (\mathbf{Stud\_ID}, \mathbf{Course}, \mathrm{Score})$,
relation names are italic, Primary Key attributes are in bold. Assume that on he user's computer the following queries are cached:

Query 1: List of students studying physics, whose ID is bigger than 210:

$$P_1 = \pi_{X_1}(\sigma_{F_1}(R_1 \bowtie R_2)),$$

where π_{X_1} – projection operation over the set of attributes $X_1 = \{\mathrm{Stud\_ID}, \mathrm{Stud\_Name}, \mathrm{Group}, \mathrm{Course}\}$, σ – selecting operation, F_1 – logical formula: $(\mathrm{Stud\_ID} > 210\ \&\ \mathrm{Course} = \mathrm{Physics})$, $\bowtie$ –natural join.

Query 2: Examination sheets of the group M10:

$$P_2 = \pi_{X_2}(\sigma_{F_2}(R_1 \bowtie R_3)),$$

where $X_2 = \{\mathrm{Stud\_ID}, \mathrm{Group}, \mathrm{Course}, \mathrm{Score}\}$, F_2 – logical condition: $(\mathrm{Group} = \mathrm{M10})$.

Let us assume that the user has requested information formalized with the following query:

$$P^* = \pi_{X^*}(\sigma_{F^*}(R_1 \bowtie R_2 \bowtie R_3)),$$

where $X^* = \{\mathrm{Stud\_ID}, \mathrm{Group}, \mathrm{Score}\}$, F^* – logical condition: $(\mathrm{Stud\_ID} > 300\ \&\ \mathrm{Group} = \mathrm{M10}\ \&\ \mathrm{Course} = \mathrm{Physics})$.

Using the calculation and comparison of the truth domains P_1, P_2 and P^*, we obtain that the query can be executed in the cache: $P^* = \pi_{X^*}(\sigma_{F_3}(P_1 \bowtie P_2))$, where F_3 – logical condition: $(\mathrm{Stud\_ID} > 300)$. Requesting the server in this case is not required.

2. Logical constraint definition and properties

To simplify domain computations we will consider logical formulas in Disjunctive Normal Form (DNF). In general case formula F has the following form:

$$F = K_1 \vee K_2 \vee \cdots \vee K_m, \tag{1}$$

$$K_i = T_1 \& \ T_2 \ldots \& \ T_n, i = 1, \ldots, m, \tag{2}$$

here $T_j, j = 1, \ldots, n$ – predicates, where expanded attribute names are specified. $R_i.A_j$ means attribute A_j in relation R_i. Those predicates can be:

- comparison operation $\mathit{Expr}_1 \ \theta \ \mathit{Expr}_2$, θ – comparison operator ($\theta \in \{=, \neq, >, <, \leq, \geq\}$), Expr_i – type conformant expressions, defined in a space of expanded attribute names and constants;
- operation Expr_1 [NOT] $\mathit{BETWEEN}$ Expr_2 AND Expr_3 (symbols inside square brackets [$*$] are optional);
- operation Expr [NOT] IN S, here S - values list or subquery, having a set of attribute $R_i.A_j$ values as a result;
- operation Str_1 [NOT] LIKE Str_2, here Str_i – strings;
- operation $\mathit{Expr} \ \theta \ \mathit{ALL/ANY} \ S$.

Note 1. *We assume logical formulas have no trivial conditions on attributes, for example, $\mathit{Expr}_1 = \mathit{Expr}_1$ and those reduced to such form. In general, we assume that $R_i.A_l$ domain is not fully contained in $T_j(\ldots, R_i.A_l, \ldots)$ predicate truth space. Such conditions can be removed from the formula without a change in the truth space (we will define it later).*

Definition 1. *Space of attributes contained in a formula show the dimension of the formula and is denoted as $\langle F \rangle$.*

$$\langle F \rangle = \{R_1^F.A_1^F, \ldots, R_k^F.A_k^F\}. \tag{3}$$

Listed variants of operations don't use every single SQL capability. For example, we don't use the *EXISTS* predicate, because it has no expanded attribute names in it. The *NULL* predicate is used for another purpose in our paper.

During logical formula domain calculation, if we have some attribute having $NULL$ value on a tuple t, we then get *UNKNOWN* value for the whole formula, because SQL-query results follow the Three-valued logic. It leads us to ambiguous interpretations of results of both usual users and experienced programmers. To solve this problem we suggest the following constraint: every attribute in F^* is supplied with a property "Use of undefined value" with two mutually exclusive values: "Yes" or "No". The reasoning behind this property is the following: if it is assigned "Yes", then we leave tuples with $NULL$ value for further consideration. Otherwise, having "No" in that property guarantees us removing all such tuples.

Let's write expression (1) for F in the following form: $F(\ldots, T_j, \ldots)$, here T_j – predicates of expression (2). After the modification it will be the following:

$F(\dots, T'_j, \dots) \wedge_{i,j} (R_i.A_j \neq NULL)$, here $\wedge_{i,j}(R_i.A_j \neq NULL)$ – conjunction of all F attributes,for which $NULL$ is not allowed, and $T'_j = (T_j \vee_{i,j} (R_i.A_j = NULL))$, here $\vee_{i,j}(R_i.A_j = NULL)$ – disjunction of all F attributes, for which $NULL$ is allowed. Outer brackets for T'_j predicate define operation priority. We can see that this logical formula can only have $TRUE$ and $FALSE$ values when considering it in Three-valued logic. We can also note that if tuples don't have undefined values, the initial formula F will be equivalent to the transformed one, so semantics of view P almost undistorted. For the disclosure of the term "almost"we consider example. Assume $F = R_1.A_2 > 3 \vee R_3.A_4 < 4$. Let tuple t for $R_1.A_2$ has value $NULL$, with a property “Use of undefined value” equal to “Yes”, and $R_3.A_4$ value equal to 5, so value of $R_3.A_4 < 4$ is $FALSE$. Then transform of formula F on tuple t will be equal $TRUE$, which is not obvious.

Hereafter we will assume all the F formulas to be transformed.

We consider set $\mathcal{A} = \{(a_1, \dots, a_n) \mid a_i \in Dom(A_i), i = 1, \dots, n\}$, here $Dom(A_i)$ – is a domain of A_i. Cartesian product $Dom(A_1) \times Dom(A_2) \times \dots \times Dom(A_n)$ is an n-dimensional space of all values for all Database attributes. Data constraints bound this space to some set of points that represents a set of available Database states.

In our example with University Study plan, the $\mathcal{A}$ can be the following:

$$\mathcal{A} = Dom(\text{Stud\_ID}) \times Dom(\text{Stud\_Name}) \times Dom(\text{Group}) \times Dom(\text{Room\_ID}) \times Dom(\text{Course}) \times Dom(\text{Score}) = \{205, 315, 461\} \times \{\text{Rachel Davis, Noam Angrist, Cameron McCord}\} \times \{\text{M10, M11}\} \times \{100, 101, 102, 103\} \times \{\text{Physics, Chemistry}\} \times \{70, 80, 90\}$$

Obviously, without constraints this set represents more states than DB can have. For example, each student has his/her own ID, whereas there are 3 corresponding IDs for each student in $\mathcal{A}$: $\{205, 315, 461\} \times \{$Rachel Davis,
Noam Angrist, Cameron McCord$\}$.

Note 2. *Dimension of formula F can be smaller than dimension of $\mathcal{A}$. In this case, we consider the equivalent form of the formula that has all other attributes taking any values in their domains.*

Looking back to our Example 1, we see that the formula in Query 1 contains only two attributes: $F_1 = (\text{Stud\_ID} > 210 \ \& \ \text{Course} = \text{Physics})$, in other words $\langle F \rangle = 2$. The equivalent representation that we will use is the following:

$$F_1 = (\text{Stud\_ID} > 210 \ \& \ \text{Course} = \text{Physics} \ \& \ \text{Stud\_Name} \ \mathit{IN} \ \{\text{Rachel Davis, Noam Angrist, Cameron McCord}\} \ \& \ \text{Group} \ \mathit{IN} \ \{\text{M10, M11}\} \ \& \ \text{Room\_ID} \ \mathit{IN} \ \{100, 101, 102, 103\} \ \& \ \text{Score} \ \mathit{IN} \ \{70, 80, 90\})$$

The motivation behind this transformation is to allow us comparing truth spaces of formulas defined on different attributes.

Definition 2. *Truth space of logical formula F, defined by (1), (2), (3), is a set $M(F) = \{a \in \mathcal{A} \mid F(a) = TRUE\}$.*

So, the truth space of our formula F_1 will be the following subset of $\mathcal{A}$: $M(F) = \{$(315, Rachel Davis, M10, 100, Physics, 70), (315, Noam Angrist, M10, 100, Physics, 70), ...(461, Cameron McCord, M10, 100, Physics, 90)$\}$

$M(F)$ for a given formula F, written in DNF, is nothing but a union of Truth spaces of distinct conjunctive clauses. Truth space of a single conjunctive clause is an intersection of Truth spaces of it's predicates.

Note 3. *The complexity of Expr, S and Str predicates is defined by the software's ability to compute Truth spaces of formulas, as it is shown in example 1.*

Definition 3. *Given a logical formula F, defined by (1), (2), (3), the projection of F on the X attribute set is a logical formula $F[X], \langle F[X]\rangle = X$, that has all its predicates with $R_i^F.A_i^F \notin X$ replaced with trivial predicate $TRUE$.*

So, the projection of F_1 on the attribute set $X = \{\text{Stud\_ID}\}$ will be $F_1[\text{Stud\_ID}] = (\text{Stud\_ID} > 210 \text{ \& TRUE}) = (\text{Stud\_ID} > 210)$

Statement 1 (Inclusion property). $\forall X \subseteq \langle F\rangle \quad M(F) \subseteq M(F[X])$

Proof. If $X = \langle F\rangle$, then $F = F[X]$ and $M(F) = M(F[X])$. Assume $X \subset \langle F\rangle$. Consider arbitrary point $a \in M(F)$, i.e. $F(a) = TRUE$. F to $F[X]$ transformation is made by replacing T_j predicates that contain $A_j \in \langle F\rangle$ attribute, such that $A_j \notin X$ with $TRUE$ value. We get $F([X]) = TRUE$ according to (1) and (2). Therefore, $a \in M(F[X])$. □

This property of logical formulas is used while building new data view from given IDVs.

3. Intermediate Data View properties research

We denote saved IDVs as $P = \{P_1, P_2, \ldots, P_m\}$, here P_v is an IDV, $P_v = \pi_{X_v}(\sigma_{F_v}(R_1^v \bowtie R_2^v \bowtie \cdots \bowtie R_{s(v)}^v))$, $s(v)$ – amount of relations in Database that were used while building P_v, π_{X_v} – projection on X_v attributes, σ_{F_v} – selection with F_v logical formula. Eventually we need to get the following view:

$$P^* = \pi_{X^*}(\sigma_{F^*}(R_1^* \bowtie R_2^* \bowtie \cdots \bowtie R_l^*))$$

Let's study the problem of building data view P^* using existing P_v IDVs.

Theorem 1. $P^* \subseteq \pi_{X^*}(\sigma_{F^*[X_v]}(P_v))$, *if:*
a) $X^* \subseteq X_v$
b) $\{R_1^v, \ldots, R_{s(v)}^v\} \subseteq \{R_1^*, \ldots, R_l^*\}$
c) $M(F^*) \subseteq M(F_v)$.

Proof. Let t^* be any tuple in P^*. We need to show that $t^* \in \pi_{X^*}(P_v)$. From condition $t^* \in P^*$, there is the tuple $t' \in R_1^* \bowtie R_2^* \bowtie \ldots \bowtie R_l^*$ and $t^* = t'[X^*]$. There may be several such tuples. We will choose the one that satisfies $F^*(t') = TRUE$. This tuple exists indeed, because otherwise t^* would not be in P^*. Thus, there are tuples $t_i^* \in R_i^*$:

$$t_i^* = t'[\langle R_i^*\rangle], i = 1, \ldots, l, \tag{4}$$

and for any pair i and j, such that $\langle R_i^* \rangle \cap \langle R_j^* \rangle \neq \emptyset$, equality is held:

$$t_i^*[\langle R_i^* \rangle \cap \langle R_j^* \rangle] = t_i'[\langle R_i^* \rangle \cap \langle R_j^* \rangle], i, j = 1, \dots, l, \tag{5}$$

Whereas conditions (4) and (5) are held for the whole set of relations $\{R_1^*, \dots, R_l^*\}$, they are true for any its subset also, including $\{R_1^v, \dots, R_{s(v)}^v\}$. Hence, after joining tuples t_i^* from relations R_j^v we get a tuple t'', such that $t'' = t'[\langle \bowtie_{i=1}^{s(v)} R_i^v \rangle]$. Equality $t''[X^*] = t'[X^*] = t^*$ follows from a) and $X_v \subseteq \langle \bowtie_{i=1}^{s(v)} R_i^v \rangle$.

Condition $F^*(t') = \mathit{TRUE}$ and statement 1 imply the truth of the projection $F^*[X_v]$ on a tuple t', and consequently on t'', as the formula is defined at common attributes of these tuples. Furthermore, according to condition c), we have $F_v(t') = \mathit{TRUE} \Rightarrow F_v(t'') = \mathit{TRUE}$. It means $t^* = t''[X^*] \in \pi_{X^*}(P_v)$. The theorem has been proven. □

The conditions given in the aforementioned theorem guarantee the data to build P^* to be contained in P_v IDV. However, there can be excess tuples that make F^* take *TRUE* value. They appear because there are some relations in $R_1^* \bowtie R_2^* \bowtie \cdots \bowtie R_l^*$ that aren't presented in $R_1^v \bowtie R_2^v \bowtie \cdots \bowtie R_{s(v)}^v$ and will be deleted if we join those missing relations. Using Truth spaces of logical formulas we can query DBMS to get a minimal required data set to define excess tuples.

In the next theorem we describe the case of coinciding relation sets.

Theorem 2. *$P^* = \pi_{X^*}(\sigma_{F^*}(P_v))$, if:*
a) $X^* \subseteq X_v$
b) $\{R_1^v, \dots, R_{s(v)}^v\} = \{R_1^*, \dots, R_l^*\}$
c) $M(F^*) \subseteq M(F_v)$
d) $\langle F^* \rangle \subseteq X_v$.

Proof. Conditions of the theorem are a special case of theorem 1, so inclusion $P^* \subseteq \pi_{X^*}(\sigma_{F^*}(P_v))$ is considered to be proven. It is necessary to show that $\pi_{X^*}(\sigma_{F^*}(P_v)) \subseteq P^*$. Assume tuple $t_v \in \pi_{X^*}(\sigma_{F^*}(P_v))$. Let's show that $t_v \in P^*$. From definition of tuple t_v, it follows that there is tuple $t' \in R_1^v \bowtie R_2^v \bowtie \dots \bowtie R_{s(v)}^v$ and $t_v = t'[X^*]$, $F_v(t') = \mathit{TRUE}$, $F^*(t') = \mathit{TRUE}$. According to the commutativity of natural join, $R_1^v \bowtie R_2^v \bowtie \dots \bowtie R_{s(v)}^v = R_1^* \bowtie R_2^* \bowtie \dots \bowtie R_l^*$. Therefore, $t' \in R_1^* \bowtie R_2^* \bowtie \dots \bowtie R_l^*$. Hence, $t_v \in P^*$. The theorem has been proven. □

It is possible to use several IDVs to build a resulting representation. First, let's consider the following simple property of natural join.

Statement 2. *Assume $\Re_1 = R_1 \bowtie \cdots \bowtie R_k$ - natural join of some k relations. Also assume $\Re_2 = R_1 \bowtie \cdots \bowtie R_k \bowtie R_{k+1} \bowtie \cdots \bowtie R_n$. Then $\Re_2[\langle \bowtie_{i=1}^{k} R_i \rangle] \subseteq \Re_1$*

Proof. It is so indeed, because joining additional relations to $\Re_1$ can only remove some tuples that were presented there. After the projection, we will get the same relation $\Re_1$ if no tuples were deleted or some subset otherwise. The order of joins doesn't matter because of commutativity of this operation. □

Theorem 3. *$P^* \subseteq \pi_{X^*}(\sigma_{F^*[X]}(P_1 \bowtie \cdots \bowtie P_n))$, where $X = \bigcup_{v=1}^{n} X_v$ if:*

a) $X^* \subseteq X$

b) $\bigcup_{v=1}^{n}\{R_1^v,\dots,R_{s(v)}^v\}=\{R'_1,\dots,R'_{s'}\}\subseteq\{R_1^*,\dots,R_l^*\}$
c) $M(F^*)\subseteq M(F_v), v=1,\dots,n.$

Proof. Again, we will choose an arbitrary tuple $t^*\in P^*$ and show that $t^*\in\pi_{X^*}(\sigma_{F^*[X]}(\bowtie_{v=1}^{n} P_v))$. By analogy to theorem 1, the tuples $t', t''=t'[\langle\bowtie_{i=1}^{s'}R'_i\rangle]$ are built. Note that $\bigcup_{v=1}^{n}X_v\subseteq\bowtie_{i=1}^{s'}\langle R'_i\rangle$, hence $t''[X^*]=t'[X^*]=t^*$. The further reasonings also apply to all IDVs. We get $F^*[X](t'')=\mathit{TRUE}$ and $F_v(t'')=\mathit{TRUE}, v=1,\dots,n$. Considering statement 2, we have $t''[\langle\bowtie_{i=1}^{s(v)}R_i^v\rangle]\in R_1^v\bowtie\cdots\bowtie R_{s(v)}^v, v=1,\dots,n\Rightarrow t''[X_v]\in P_v, v=1,\dots,n\Rightarrow t''[\bigcup_{v=1}^{n}X_v]\in P_1\bowtie\cdots\bowtie P_n$. It means that $t^*=t''[X^*]\in\pi_{X^*}(\sigma_{F^*[X]}(P_1\bowtie\cdots\bowtie P_n))$. The theorem has been proven. □

Just like in the previous case, there are some additional conditions that allow us to get precise data presentation from IDVs.

Theorem 4. *$P^*=\pi_{X^*}(\sigma_{F^*}(P_1\bowtie\cdots\bowtie P_n))$, where $X=\bigcup_{v=1}^{n}X_v$ if:*

a) $X^*\subseteq X,\ X_v\supseteq\langle\bowtie_{i=1}^{s(v)}R_i^v\rangle\cap(\bigcup_{\substack{w=1\\w\neq v}}^{n}\langle\bowtie_{i=1}^{s(w)}R_i^w\rangle), v=1,\dots,n$
b) $\bigcup_{v=1}^{n}\{R_1^v,\dots,R_{s(v)}^v\}=\{R'_1,\dots,R'_{s'}\}=\{R_1^*,\dots,R_l^*\}$
c) $M(F^*)\subseteq M(F_v), v=1,\dots,n$
d) $\langle F^*\rangle\subseteq X.$

Proof. Conditions of the theorem are a special case of theorem 3, so inclusion $P^*\subseteq\pi_{X^*}(\bowtie_{v=1}^{n}P_v)$ is considered to be proved. It is necessary to show, that $\pi_{X^*}(\sigma_{F^*}(\bowtie_{v=1}^{n}P_v))\subseteq P^*$. Assume tuple $t\in\pi_{X^*}(\sigma_{F^*}(\bowtie_{v=1}^{n}P_v))$. Let's show, that $t\in P^*$. Denote $\sigma_{F^*}(\bowtie_{i=1}^{s(v)}R_i^v)$ as P'_v, so $P_v=\pi_{X_v}(P'_v)$. Using the property of the projection operation and condition a), we have $P_1\bowtie\cdots\bowtie P_n=\pi_{X_1}(P'_1)\bowtie\cdots\bowtie\pi_{X_n}(P'_n)=\pi_X(P'_1\bowtie\cdots\bowtie P'_n)$. Thus, according to the definition of tuple t, there is a tuple $t'\in R'_1\bowtie\ldots\bowtie R'_{s'}$, such that $t=t'[X^*]$, $F^*(t')=\mathit{TRUE}$. According to the commutativity of natural join, $R'_1\bowtie\ldots\bowtie R'_s=R_1^*\bowtie\ldots\bowtie R_l^*$, hence, $t'\in R_1^*\bowtie\ldots\bowtie R_l^*$. Therefore $t\in P^*$. The theorem has been proven. □

The conditions listed above aren't exceptional in any case. Many applications require repeat of data entry with slight modifications. For example, applications that work with multidimensional data. Hypercube dimensions do not change often, so that we can create new hypercubes from IDVs without frequent interaction with the database.

4. Conclusion

The approach suggested in this paper is a theoretical base for a system interacting with Relational Database. Such system will be capable of defining cached data availability when executing consecutive queries. This solution is fresh and wasn't used before. The conditions stated in the theorems don't require hitting Database to be checked. Results obtained allow us to analitycally define data missing in cache and query only this data.

The proposed solution will be used in dynamic building of multidimensional data. Multidimensional data building systems that use redundant data often face the problem of updating it. It is sometimes solved by the periodic updating of the hypercube contents. A similar method can be used to update the views P_v. To reduce the update time it might be a good practice to use the change log on the database server and update only those views that have original data changed. However, those dimensions don't change often.

References

[1] Zykin S., Poluyanov A., "Multidimensional data building using intermediate representations", *Administration problems*, **5** (2013), 54–59.

[2] Owens J. D. et al., "A survey of general-purpose computation on graphics hardware", *Computer Graphics Forum*, **26**, 2007, 80–113, http://www.blackwell-synergy.com/doi/pdf/10.1111/j.1467-8659.2007.01012.x.

[3] Bakkum P., Skadron K., "Accelerating sql database operations on a gpu with cuda", GPGPU, *ACM*, **425** (2010), 94–103, http://dblp.uni-trier.de/db/conf/asplos/gpgpu2010.html.

[4] Govindaraju N. K. et al., "Fast computation of database operations using graphics processors", SIGMOD Conference, *ACM*, 2004, 215–226, http://dblp.uni-trier.de/db/conf/sigmod/sigmod2004.html.

[5] He B. et al., "Relational joins on graphics processors", ACM, 2008, 511–524, http://dblp.uni-trier.de/db/conf/sigmod/sigmod2008.html.

[6] Park C.-S., Kim M.-H., Lee Y.-J., "Usability-based caching of query results in olap systems", *Journal of Systems and Software*, **68**:2 (2003), 103–119.

[7] Baralis E., Paraboschi S., Teniente E., "Materialized views selection in a multidimensional database", *Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB '97*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, 156–165, http://dl.acm.org/citation.cfm?id=645923.671019.

[8] Kalnis P., Papadias D., "Proxy-server architectures for olap.", ACM, 2001, 367–378.

[9] Afrati F. N., Li C., Mitra P., "Rewriting queries using views in the presence of arithmetic comparisons", *Theor. Comput. Sci.*, **368**:1–2 (2006), 88–123.

[10] Watanabe Y., Kitagawa H., "Query result caching for multiple event-driven continuous queries", *Inf. Syst.*, **35**:1 (2010), 94–110, http://dblp.uni-trier.de/db/journals/is/is35.html#WatanabeK10.

[11] Yates D. J. et al., "Data quality and query cost in pervasive sensing systems", IEEE Computer Society, 2008, 195–205, http://dblp.uni-trier.de/db/conf/percom/percom2008.html#YatesNKS08.

[12] Andrade H. et al., "Active semantic caching to optimize multidimensional data analysis in parallel and distributed environments", *Parallel Computing*, **33**:7–8 (2007), 497–520, http://dblp.uni-trier.de/db/journals/pc/pc33.html#AndradeKSS07.

[13] Mershad K. W., Artail H., "Codisc: Collaborative and distributed semantic caching for maximizing cache effectiveness in wireless networks", *J. Parallel Distrib. Comput.*, **71**:3 (2011), 495–511, http://dblp.uni-trier.de/db/journals/jpdc/jpdc71.html#MershadA11.

[14] Noor Abbani H. A., "Protecting data flow anonymity in mobile ad hoc networks that employ cooperative caching", *Ad Hoc Networks*, **26** (2015), 69–87.

[15] Tansel Dokeroglu A. C., Ali Bayir Murat, "Robust heuristic algorithms for exploiting the common tasks of relational cloud database queries", *Applied Soft Computing*.

[16] Beran P. P. et al., "A multi-staged blackboard query optimization framework for world-spanning distributed database resources", *ICCS*, **4**, Procedia Computer Science (2011), 156–165, http://dblp.uni-trier.de/db/journals/procedia/procedia4.html#BeranMSV11.

[17] Meij E. et al., “Mapping queries to the linking open data cloud: A case study using dbpedia”, *J. Web Sem.*, **9**:4 (2011), 418–433, http://dblp.uni-trier.de/db/journals/ws/ws9.html#MeijBHHR11.

[18] Aranda C. B. et al., “Federating queries in sparql 1.1: Syntax, semantics and evaluation”, *J. Web Sem.*, **18**:1 (2013), 1–17, http://dblp.uni-trier.de/db/journals/ws/ws18.html#ArandaACP13.

[19] Keller A. M., Basu J., “A predicate-based caching scheme for client-server database architectures”, *VLDB J.*, **5**:1 (1996), 35–47.

[20] Shim J., Scheuermann P., Vingralek R., “Dynamic caching of query results for decision support systems”, *SSDBM*, 1999, 254–263.

Кэширование запросов к реляционной базе данных с использованием областей истинности

Мосин С.В., Зыкин С.В.

Федеральное государственное бюджетное учреждение науки Институт математики им. С.Л. Соболева Сибирского отделения Российской академии наук (Омский филиал) 644043, Россия, г. Омск, ул. Певцова, 13,

Ключевые слова: реляционная база данных, кэш, область истинности

В данной статье предлагается новый метод кэширования запросов к реляционной базе данных для систем с центральным сервером и распределенными клиентами. Данные загружаются в клиентский кэш, основываясь на запросах, выполненных на сервере БД. Каждому запросу ставится в соответствие таблица – результат выполнения запроса. Эти запросы имеют специальный вид, называемый "универсальный реляционный запрос", основанный на трех базисных операциях реляционной алгебры: селекции, проекции, естественном соединении (natural join). Следует отметить, что такая форма запроса наиболее близка к естественному языку и большинство запросов может быть записано в этом виде. Кроме того, эта форма записи позволяет анализировать корректность запроса, проверяя свойство соединения без потери информации (СБПИ). Последовательные запросы могут исполняться на клиенте, используя кэш, если удастся определить, что результаты искомого запроса полностью содержатся в кэше. Для осуществления такой проверки анализируются области истинности логических ограничений искомого запроса и запросов, результаты которых уже содержатся в кэше. Требуемые операции могут быть проведены аналитически, без необходимости дополнительных запросов к базе банных. Предложенный метод может быть использован для определения недостающих в кэше данных и последующего запроса только на эти данные. Для этого также используются аналитические вычисления, что является принципиальным отличием данной статьи от существующих технологий. Для этой цели в статье представлено четыре теоремы. В первой и третьей теореме получены условия, позволяющие определить наличие необходимых данных, а во второй и четвертой получены условия вычисления данных только с использованием кэша. Проблема актуализации данных не затрагивается в этой статье. Однако она может быть решена путем учета запросов на сервере и обновлении данных при помощи триггеров.

Статья публикуется в авторской редакции.

Сведения об авторах:

Мосин Сергей Владимирович,

Федеральное государственне бюджетное учреждение науки Институт математики им. С.Л. Соболева Сибирского отделения Российской академии наук (Омский филиал), аспирант, ORCID 0000-0001-5459-5853

Зыкин Сергей Владимирович,

Федеральное государственне бюджетное учреждение науки Институт математики им. С.Л. Соболева Сибирского отделения Российской академии наук (Омский филиал), д-р техн. наук, профессор, ORCID 0000-0002-0576-2149