УДК 004.652 DOI: 10.14529/cmse160108

СРАВНЕНИЕ ОБЛАСТЕЙ ИСТИННОСТИ ЗАПРОСОВ К РЕЛЯЦИОННОЙ БАЗЕ ДАННЫХ

С.В. Мосин

В данной статье предлагается описание подходов аналитического сравнения пользовательских запросов к реляционной базе данных. Такое сравнение имеет целью установление возможности частичного или полного использования закэшированных на компьютере пользователя запросов к СУБД и основано на применении аппарата логики предикатов, где в качестве формул выступают логические ограничения SQL, а предикатами служат элементарные операции SQL. В случае, если результат выполнения пользовательского запроса полностью содержится в кэше, то данные можно взять оттуда, минуя запрос к удаленному серверу. Описанные подходы выражены в алгоритме использования кэшированных данных. Предложенный алгоритм также может быть использован для определения недостающих в кэше данных и последующего запроса только на эти данные. Для этого также используются аналитические вычисления, что экономит сетевой трафик и время на выполнение запросов и является принципиальным отличием данной технологии от существующих аналогов.

Ключевые слова: реляционная база данных, кэш, область истинности.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Мосин С.В. Сравнение областей истинности запросов к реляционной базе данных // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 1. С. 85–99. DOI: 10.14529/cmse160108.

Введение

В данной работе рассматривается проблема использования результатов выполнения запросов, закэшированных на компьютере пользователя. Такие данные будем называть представлениями (иначе сохраненные представления, промежуточные представления). Мы предполагаем наличие централизованного сервера и множество удаленных клиентов с расположенной на сервере реляционной базой данных. В кэше сохраняются результаты выполнения запросов для того, чтобы обеспечить максимальное использование сохраненных данных при выполнении последующих запросов.

Поставленная проблема связана с областью оптимизации запросов, поскольку нацелена на сокращение объема передаваемых данных с сервера базы данных. Зарезервированные данные активно используется в системах управления базами данных (СУБД). Но в большинстве случаев это касается повторного использования данных, записанных в кэш, без предварительного анализа содержимого на предмет возможности частичного или комбинированного использования. Работа СУБД ограничивается тем, что при выполнении очередного запроса блоки данных не запрашиваются с внешних устройств, если они есть в кэше, т.е. анализируются номера блоков, а не их содержимое.

Целью данной работы является разработка методов аналитического сравнения логических ограничений запросов, а также построение алгоритма использования кэшированных данных. Проблема актуализации данных не затрагивается в данной статье.

Раздел 1 посвящен обзору публикаций на схожие темы: использование кэша при работе с различными видами баз и хранилищ данных, логика предикатов и ее применение в сфере баз данных. В разделе 2 описывается формализованная постановка задачи. Раздел 3 представляет подходы к аналитическому сравнению простейших операций языка

SQL, из которых строятся все логические формулы. В следующем разделе, 4, выводятся основные эквивалентности, используемые при определении возможности использования кэша. Заключительный раздел 5 объединяет полученные результаты в виде схемы алгоритма использования кэша в ходе исполнения запросов к реляционной базе данных. В заключении сделано обобщение полученных результатов и приведены дальнейшие направления исследований.

1. Обзор существующих работ

В области теоретических исследований ситуация складывается следующим образом. Огромное число публикаций посвящено проблеме построения оптимального плана запроса на основе формальных правил, в которых не используются области определения предикатов в SQL-операторах (логическая оптимизация) либо эти области учитываются при вычислении статистических оценок для оптимизации физического доступа к базе данных. Близкими по методам решения являются задачи выполнения запросов на потоках данных [5, 13], однако различные, по сравнению с настоящей работой, цели приводят к различным результатам.

Наиболее близка к рассматриваемой проблеме работа [1]. В ней рассматриваются конъюнктивные запросы над доменами данных с предикатами в виде арифметических сравнений, и представлены алгоритмы вычисления запросов с использованием представлений. В настоящей работе рассматривается специальный вид универсального реляционного запроса над отношениями базы данных, а не над отдельными доменами. Хотя цели в обеих работах совпадают, результаты различны по указанной причине. В частности, в настоящей работе нет необходимости разрабатывать алгоритмы выборки данных из промежуточных представлений, так как их замещает реляционная алгебра.

Существует ряд работ, в которых рассматривается проблема оптимизации запросов к многомерным хранилищам данных с использованием сохраненных представлений. При этом существуют два основных подхода для резервирования результатов запросов: статический [2, 6, 7] и динамический [9, 14–16]. Первый базируется на использовании набора фиксированных запросов, во втором же предполагается динамический выбор результатов запросов для резервирования на основе статистики появления а также вычислительной стоимости выполнения запросов. В качестве источника данных используются хранилища данных, либо реляционная база данных, преобразованная к иерархическому виду. При этом, рассматриваются специальные запросы манипулирования многомерными данными. В данной статье рассматривается технология раздельного формирования размерностей представления данных. Поэтому интерес представляют стандартные SQL запросы к базе данных и промежуточным представлениям. Результирующее представление может быть затем обработано специальными запросами манипулирования многомерными данными.

В работе [10] решается задача анализа содержимого кэша, при этом промежуточным представлениям в кэше ставятся в соответствие предикаты. Проблема использования представлений решается за счет выводимости предикатов. В данной работе проблему использования промежуточных представлений предлагается решать посредством вычисления их областей истинности. Это позволяет, во-первых, аналитически определить возможность использования кэша, а во-вторых, сформировать SQL команды, которые позволят загрузить недостающие данные с сервера базы данных.

Данная работа основывается на теоретических результатах, описанных в статье [12], где предлагается технология, в основе которой лежит понятие области истинности логического ограничения, накладываемого на пользовательский запрос к базе данных. Фактически, данная статья является логическим продолжением указанной работы, определяя аналитические операции над областями истинности пользовательских запросов.

Основным инструментом работы с логическими формулами в статье является аппарат математической логики первого порядка, или логики предикатов [3, 8, 11]. Данная работа не ставит целью получение новых результатов в области математической логики, а лишь использует имеющиеся результаты в применении к логическим ограничениям языка SQL, расширяя таким образом возможности анализа пользовательских запросов по сравнению с классическими подходами СУБД, где нет возможности частичного или комбинированного использования закэшированных запросов.

В книге [4] подробно описано то, как логика предикатов используется в языке SQL в виде реляционного исчисления (relational calculus), а также приведены основные эквивалентности логических предикатов, используемые в данной статье.

2. Постановка задачи

Будем рассматривать реляционную базу данных со схемой $R = \{R_1, R_2, \ldots, R_N\}$.

Определение 1. *Универсальным реляционным запросом называется запрос следующего вида:*

$$P = \pi_X(\sigma_F(R_1 \bowtie \cdots \bowtie R_m)), m \in [1, \ldots, N]$$

Логические формулы, по которым проводится селекция, будем рассматривать в дизъюнктивной нормальной форме. В общем случае формула F имеет вид

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

$$K_i = T_1 \ \& \ T_2 \ldots \& \ T_s, i = 1, \ldots, l, \tag{2}$$

где $T_j, j = 1, \ldots, s$ – предикаты, в которых явным образом специфицированы расширенные имена атрибутов $R_i.A_j$ (атрибут A_j в отношении R_i):

Возможные значения T_j:

- операция сравнения $Expr_1\ \theta\ Expr_2$, θ – операция сравнения ($\theta \in \{=, \neq, >, <, \leq, \geq\}$), $Expr_i$ – согласованные по типам допустимые выражения, определенные на множестве расширенных имен атрибутов и констант;
- операция $Expr_1$ $[NOT]$ $BETWEEN$ $Expr_2$ AND $Expr_3$ (содержимое в прямоугольных скобках $[*]$ для предиката не является обязательным при написании);
- операция $Expr$ $[NOT]$ IN S, где S – список значений либо подзапрос, результатом которого является столбец атрибута A_j в отношении R_i;
- операция Str_1 $[NOT]$ $LIKE$ Str_2, где Str_i – строки;
- операция $Expr\ \theta\ ALL/ANY\ S$.

Обозначение. Множество атрибутов, входящих в формулу, выражает размерность формулы и обозначается $\langle F \rangle$.

$$\langle F \rangle = \{R_1^F.A_1^F, \ldots, R_k^F.A_k^F\}, k - - \text{количество атрибутов, входящих в формулу} \tag{3}$$

Перечисленные варианты операций используют не все возможности языка SQL. Например, предикат *EXISTS* не используется, поскольку в нем явно не специфицированы расширенные имена атрибутов, предикат *NULL* используется в данной работе для другой цели.

В статье [12] приведен алгоритм преобразования логических формул, который позволяет избежать получения значения *UNKNOWN* в результате вычисления значений формул. Далее будем предполагать, что все формулы F являются преобразованными.

Введем в рассмотрение множество $\mathcal{A} = \{(a_1, \ldots, a_L) \mid a_i \in Dom(A_i), i = 1, \ldots, L\}$, где $Dom(A_i)$ – множество всех допустимых значений атрибута A_i. Декартово произведение $Dom(A_1) \times Dom(A_2) \times \cdots \times Dom(A_L)$ – L-мерное пространство значений всех атрибутов базы данных.

Определение 2. *Областью истинности логической формулы F, заданной (1), (2), (3), является множество, определяемое по следующему правилу: $M(F) = \{a \in \mathcal{A} \mid F(a) = TRUE\}$.*

Данное определение изначально введено в статье [12].

В соответствии с данными определениями легко понять, как будут устроены операции над областями истинности. $M(F)$ для некоторой формулы F, заданной своей дизъюнктивной нормальной формой, является объединением областей истинности, представленных отдельными конъюнктами формулы. Область истинности каждого конъюнкта определяется как пересечение областей истинности предикатов, входящих в него.

Далее введем определения, касающиеся модификации вхождения атрибутов в логические формулы.

Определение 3. *Проекцией логической формулы F, заданной (1), (2), (3), на множество атрибутов X называется логическая формула $F[X], \langle F[X] \rangle = X$, в которой все термы, содержащие атрибуты $R_i^F.A_i^F \notin X$, заменены на тривиальный терм TRUE.*

Основные теоретические результаты, позволяющие судить о возможности использования кэша при выполнении запросов:

Теорема 1. $P^* \subseteq \pi_{X^*}(\sigma_{F^*[X]}(P_1 \bowtie \cdots \bowtie P_n))$, где $X = \bigcup\limits_{v=1}^{n} X_v$ если:

а) $X^* \subseteq X$

б) $\bigcup\limits_{v=1}^{n} \{R_1^v, \ldots, R_{s(v)}^v\} = \{R'_1, \ldots, R'_{s'}\} \subseteq \{R_1^*, \ldots, R_l^*\}$

в) $M(F^*) \subseteq M(F_v), v = 1, \ldots, n.$

Предложенные в теореме условия гарантируют, что данные, необходимые для формирования целевого запроса P^*, содержатся в промежуточном представлении P_v. Однако в нем могут быть лишние кортежи, которые дают значение *TRUE* при подстановке в формулу F^*. Дело в том, что эти кортежи будут удалены при выполнении операции естественного соединения с отношениями, которых не хватает в множестве $R_1^v \bowtie R_2^v \bowtie \cdots \bowtie R_{s(v)}^v$ для совпадения с множеством $R_1^* \bowtie R_2^* \bowtie \cdots \bowtie R_l^*$.

Следующая теорема соответствует частному случаю, где проблема лишних кортежей не возникает.

Теорема 2. $P^* = \pi_{X^*}(\sigma_{F^*}(P_1 \bowtie \cdots \bowtie P_n))$, где $X = \bigcup\limits_{v=1}^{n} X_v$ если:

а) $X^* \subseteq X$, $X_v \supseteq \langle \bowtie_{i=1}^{s(v)} R_i^v \rangle \cap (\bigcup\limits_{\substack{w=1 \\ w \neq v}}^{n} \langle \bowtie_{i=1}^{s(w)} R_i^w \rangle), v = 1, \ldots, n$

б) $\bigcup\limits_{v=1}^{n}\{R_1^v,\ldots,R_{s(v)}^v\}=\{R_1',\ldots,R_{s'}'\}=\{R_1^*,\ldots,R_l^*\}$
в) $M(F^*)\subseteq M(F_v), v=1,\ldots,n$
г) $\langle F^*\rangle\subseteq X$.

Доказательства теорем приведены в статье [12].

Для проверки условий в) обеих теорем, очевидно, требуется уметь аналитически сравнивать области истинности логических формул. Этому будет посвящен следующий раздел. Далее на основе полученных результатов будет построен алгоритм, позволяющий для данного набора закэшированных представлений и некоторого целевого представления определить, можно ли воспользоваться кэшем для получения искомых данных и, если нет, получить недостающие данные.

3. Логические операции над предикатами

Рассмотрим элементарные логические предикаты, из которых состоит логическая формула, заданная (1), (2), (3). Для этого нам понадобится вычислять значение логических операций $\vee,\&,\neg$ над предикатами $T_j, j=1,\ldots,s$. Определение значения логической формулы аналитически возможно в случае, если формула не зависит от значений атрибутов базы данных (БД). Приведение к такому виду возможно в ряде изложенных далее случаев.

3.1. Упрощение конъюнкции и дизъюнкции простых предикатов

1. $Expr_1\ \theta\ Expr_2$, $(\theta\in\{=,\neq,>,<,\leq,\geq\})$
   Данный тип операций определяет простейшие арифметические операции равенства и неравенства. Сравнение таких предикатов сводится к применению элементарных правил решения неравенств.
   Пусть даны следующие предикаты:
   - $T_1=(R_i.A_j\ \theta_1\ a),\ a\in Dom(R_i.A_j)$
   - $T_2=(R_i.A_j\ \theta_2\ b),\ b\in Dom(R_i.A_j)$

   Рассмотрим всевозможные комбинации конъюнкции и дизъюнкции данных типов предикатов, при которых в результате получается тривиальное значение $TRUE$ либо $FALSE$.
   Заметим, что операции $\leq,\geq$ сводятся к операциям $<,>,=,\neq$: $x_1\leq x_2\sim x_1<x_2\vee x_1=x_2$, поэтому мы исключаем их из дальнейшего рассмотрения.
   Также обратим внимание, что если $\theta_1=\theta_2$, то $T_1\ \&\ T_2\equiv TRUE\iff a=b$. Далее рассматриваем только различные операции θ_1 и θ_2.

   (a) $\theta_1=$ "$=$"
   i. $\theta_2=$ "$\neq$"

   $$T_1\ \&\ T_2\equiv TRUE\iff a=b$$

   $$T_1\ \&\ T_2\equiv FALSE\iff a\neq b$$

   ii. $\theta_2=$ "$<$"

   $$T_1\ \&\ T_2\equiv FALSE\iff a>b$$

   iii. $\theta_2=$ "$>$"

   $$T_1\ \&\ T_2\equiv FALSE\iff a<b$$

(b) $\theta_1 =$ "$\neq$"
Никакие прочие операции не дают в результате тождественной истины или лжи при проведении конъюнкции или дизъюнкции с T_1 в данном случае.

(c) $\theta_1 =$ "$<$"
Упрощение возможно только в случае $\theta_2 =$ "$>$". Получаем:

$$T_1 \,\&\, T_2 \equiv FALSE \iff a < b$$

$$T_1 \vee T_2 \equiv TRUE \iff a \geq b$$

2. $Expr_1$ $[NOT]$ $BETWEEN$ $Expr_2$ AND $Expr_3$
Эта и дальнейшие операции (кроме $LIKE$) сводятся к операциям первого типа:
   (a) $Expr_1$ $BETWEEN$ $Expr_2$ AND $Expr_3$
   $T = (R_i.A_j$ $BETWEEN$ a AND $b)$, $a, b \in Dom(R_i.A_j)$ Эквивалентное представление данной операции: $T = (a \leq R_i.A_j \leq b)$
   (b) $Expr_1$ NOT $BETWEEN$ $Expr_2$ AND $Expr_3$
   $T = (R_i.A_j$ NOT $BETWEEN$ a AND $b)$, $a, b \in Dom(R_i.A_j)$ Эквивалентное представление данной операции: $T = (R_i.A_j \leq a \vee R_i.A_j \geq b)$
3. $Expr$ $[NOT]$ $LIKE$ Str,
$Expr$ — атрибут строкового типа данных, Str — строка особого типа, содержащая управляющие символы языка SQL. К ним относятся:
   - '%' Данный символ совпадает при сравнении с любым количеством любых символов строки
   - '_' Данный символ совпадает при сравнении с одним любым символом строки

   Различные диалекты языка SQL от разных производителей СУБД дополняют этот список другими управляющими конструкциями, так что он может отличаться в зависимости от реализации. Но так или иначе сравнение предикатов такого вида имеет схожую схему. По сути данная операция представляет из себя упрощенное регулярное выражение. Проведение таких операций, как дизъюнкция, конъюнкция и отрицание, осуществляется точно также как для регулярных выражений.
4. $Expr$ $[NOT]$ IN S, где S – список значений или подзапрос, результат которого – это множество значений атрибута $R_i.A_j$.
   (a) $Expr$ IN S
   Если S – это список заранее заданных значений:
   $T = (R_i.A_j$ IN $\{s_1, s_2, \ldots, s_l\})$, $s_k \in Dom(R_i.A_j), k = 1, \ldots, l$
   Эквивалентное представление: $T = (\bigvee\limits_{k=1}^{l} R_i.A_j = s_k)$
   Если S – это подзапрос:
   $T = (R_i.A_j$ IN $\pi_{R_i.A_j}(\sigma_F(R_1 \bowtie \cdots \bowtie R_n)))$

   (b) $Expr$ NOT IN S
   Если S – это список заранее заданных значений:
   $T = (R_i.A_j$ NOT IN $\{s_1, s_2, \ldots, s_l\})$, $s_k \in Dom(R_i.A_j), k = 1, \ldots, l$
   Эквивалентное представление: $T = (\bigwedge\limits_{k=1}^{l} R_i.A_j \neq s_k)$
   Если S – это подзапрос:

$T = (R_i.A_j\ IN\ \pi_{R_i.A_j}(\sigma_F(R_1 \bowtie \cdots \bowtie R_n)))$

5. *Expr θ ALL/ANY S.*
   Данная операция является обобщением операций первого типа и сводится к нему в случае простого списка значений.
   (a) *Expr θ ALL S*
   Если S – это список заранее заданных значений:
   $T = (R_i.A_j\ \theta\ ALL\ \{s_1, s_2, \ldots, s_l\}),\ s_k \in Dom(R_i.A_j), k = 1, \ldots, l$
   Эквивалентное представление:
   $T = (R_i.A_j\ \theta\ s_1\ \&\ R_i.A_j\ \theta\ s_2\ \&\ \ldots \&\ R_i.A_j\ \theta\ s_k)$
   Если S – это подзапрос:
   $T = (R_i.A_j\ \theta\ ALL\ \pi_{R_i.A_j}(\sigma_F(R_1 \bowtie \cdots \bowtie R_n)))$

   (b) *Expr θ ANY S*
   Если S – это список заранее заданных значений:
   $T = (R_i.A_j\ \theta\ ANY\ \{s_1, s_2, \ldots, s_l\}),\ s_k \in Dom(R_i.A_j), k = 1, \ldots, l$
   Эквивалентное представление:
   $T = (R_i.A_j\ \theta\ s_1\ \vee\ R_i.A_j\ \theta\ s_2\ \vee\ \cdots \vee\ R_i.A_j\ \theta\ s_k)$
   Если S – это подзапрос:
   $T = (R_i.A_j\ \theta\ ANY\ \pi_{R_i.A_j}(\sigma_F(R_1 \bowtie \cdots \bowtie R_n)))$

3.2. Упрощение конъюнкции и дизъюнкции предикатов, содержащих подзапросы

Если в операциях 4 и 5 типа встречается подзапрос, аналитическое сравнение таких предикатов возможно только в некоторых частных случаях. Дело в том, что нельзя заранее определить область истинности таких предикатов, так как она зависит от реализации базы данных. Использование же информации о БД противоречит аналитическому подходу. Рассмотрим условия, при которых возможно их сравнивать аналитически.

Пусть подзапросы имеют следующий вид:

$$S_1 = \pi_{R_i.A_j}(\sigma_{F_1}(R_1^1 \bowtie \cdots \bowtie R_l^1))$$

$$S_2 = \pi_{R_i.A_j}(\sigma_{F_2}(R_1^2 \bowtie \cdots \bowtie R_k^2))$$

S_1 и S_2 являются множествами и, следовательно, могут быть следующие ситуации:

1. $S_1 \cap S_2 = \emptyset$
   Определить, что подзапросы не пересекаются, аналитически можно, если $M(F_1) \cap M(F_2) = \emptyset$. В данном случае наблюдаются следующие эквивалентности:
   (a) $(R_i.A_j\ IN\ S_1)\ \&\ (R_i.A_j\ IN\ S_2) \equiv FALSE$
   (b) $(R_i.A_j\ =ALL\ S_1)\ \&\ (R_i.A_j\ =ALL\ S_2) \equiv FALSE$
2. $S_1 \cap S_2 \neq \emptyset$
   В общем случае нет возможности установить упрощения к виду тождественной истины/лжи.

3.3. Операция отрицания для всех типов предикатов

Логическое отрицание ставит в соответствие противоположный предикат в каждом из случаев.

1. $Expr_1\ \theta\ Expr_2$, $(\theta \in \{=, \neq, >, <, \leq, \geq\})$
   $T = (R_i.A_j\ \theta\ a) \ \rightarrow \neg T = (R_i.A_j\ \neg\theta\ a), a \in Dom(R_i.A_j)$
   - $T = (R_i.A_j = a) \rightarrow \neg T = (R_i.A_j\ \neq a)$
   - $T = (R_i.A_j > a) \rightarrow \neg T = (R_i.A_j\ < a)$
   - $T = (R_i.A_j \geq a) \rightarrow \neg T = (R_i.A_j\ \leq a)$
2. $Expr_1\ [NOT]\ BETWEEN\ Expr_2\ AND\ Expr_3$
   $T = (R_i.A_j\ BETWEEN\ a\ AND\ b),\ a, b \in Dom(R_i.A_j) \rightarrow$
   $\rightarrow \neg T = (R_i.A_j\ NOT\ BETWEEN\ a\ AND\ b)$
3. $Expr\ [NOT]\ LIKE\ Str$
   $T = (R_i.A_j\ LIKE\ Str) \rightarrow \neg T = (R_i.A_j\ NOT\ LIKE\ Str)$
4. $Expr\ [NOT]\ IN\ S$
   $T = (R_i.A_j\ IN\ S) \ \rightarrow \neg T = (R_i.A_j\ NOT\ IN\ S)$
5. $Expr\ \theta\ ALL/ANY\ S$
   $T = (R_i.A_j\ \theta\ ALL\ S) \ \rightarrow \neg T = (R_i.A_j\ \neg\theta\ ANY\ S)$

4. Операции над областями истинности логических формул

Здесь и далее будем рассматривать только случай нескольких промежуточных представлений как более общий.

Проверка на пустоту пересечения областей истинности логических формул осуществляется следующим образом: $M(F^*) \cap (M(F_1) \cup M(F_2) \cup \cdots \cup M(F_n)) = \emptyset \iff M(F^*\ \&\ (F_1 \vee F_2 \vee \cdots \vee F_n)) = \emptyset \iff F^*\ \&\ (F_1 \vee F_2 \vee \cdots \vee F_n) \equiv FALSE$.

Для выполнения условий теорем 1, 2 требуется установить включение области истинности формулы пользовательского запроса $M(F^*)$ в область истинности формул промежуточных представлений $M(F_1), M(F_2), \ldots, M(F_n)$, или $M(F^*) \subseteq M(F_v), v = 1, \ldots, n$, то есть должно быть выполнено $M(F^*) \subseteq M(F_1)\ \&\ M(F^*) \subseteq M(F_2)\ \&\ \ldots\ \&\ M(F^*) \subseteq M(F_n) \equiv TRUE \iff M(F^*) \subseteq M(F_1\ \&\ F_2\ \&\ \ldots\ \&\ F_n) \equiv TRUE \iff F^* \rightarrow F_1\ \&\ F_2\ \&\ \ldots\ \&\ F_n \equiv TRUE \iff F_1\ \&\ F_2\ \&\ \ldots \&\ F_n \vee \neg F^* \equiv TRUE$.

Практическая проверка данных условий аналитическим способом возможна путем упрощения формул при помощи правил, изложенных в предыдущем пункте.

Для получения недостающих в кэше данных необходимо уметь строить формулу, область истинности которой равна разности областей истинности исходных формул: $M(F^*) \setminus M(F_1\ \&\ F_2\ \&\ \ldots\ \&\ F_n) = M(F_{n+1})$. Такая формула F_{n+1}, очевидно, выражается следующим образом: $F_{n+1} = F^*\ \&\ \neg(F_1\ \&\ F_2\ \&\ \ldots\ \&\ F_n)$.

5. Алгоритм использования кэшированных данных

Далее приводится схема алгоритма аналитической проверки логических ограничений и определения недостающих в кэше данных. Затем рассматривается 2 примера, демонстрирующие его применение.

5.1. Схема алгоритма

Пусть сохраненные на пользовательской машине данные определены $P = \{P_1, \ldots, P_n\}$, целевой запрос обозначим P^*. Пусть также выполнены условия а) и б) Теоремы 1.

Алгоритм получения искомого представления данных с использованием кэша.
Функция *get_ data_ from_ server()* запрашивает требуемые данные с сервера БД, минуя кэш. Функция *get_ data_ from_ cache()* получает данные из кэша. Алгоритм, лежащий в основе, базируется на результатах теорем 1, 2. Подробное его описание выходит за рамки данной статьи и будет представлено отдельно в будущем.

Require: $P = \{P_1, \ldots, P_n\}$;
$R^* = \{R_1^*, \ldots, R_l^*\}$;
F^*;
$F_v, v = 1, \ldots, n$;
X^*;
$X_v, v = 1, \ldots, n$;
$\bigcup\limits_{v=1}^{n} \{R_1^v, \ldots, R_{s(v)}^v\} = \{R'_1, \ldots, R'_{s'}\} \subseteq \{R_1^*, \ldots, R_l^*\}$;
$M(F^*) \subseteq M(F_v), v = 1, \ldots, n.$
if $M(F^*) \cap (M(F_1) \cup M(F_2) \cup \cdots \cup M(F_n)) = \emptyset$ **then**
  return *get_ data_ from_ server()*
end if
if $M(F^*) \subseteq (M(F_1) \cap M(F_2) \cap \cdots \cap M(F_n))$ **then**
  return *get_ data_ from_ cache()*
end if
$F_{n+1} \leftarrow F^* \ \&\ \neg(F_1 \ \&\ F_2 \ \&\ \ldots \ \&\ F_n)$
$P_{n+1} \leftarrow \pi_{X^*}(\sigma_{F_{n+1}}(R_1 \bowtie \cdots \bowtie R_l))$
return *get_ data_ from_ cache()*

Пояснение шагов алгоритма:

1. Проверка условия $M(F^*) \cap (M(F_1) \cup M(F_2) \cup \cdots \cup M(F_n)) = \emptyset$. Если выполнено, делаем запрос к БД, кэш в данном случае бесполезен.
2. Проверка условия $M(F^*) \subseteq (M(F_1) \cap M(F_2) \cap \cdots \cap M(F_n))$. Если выполнено, получаем данные из кэша.
3. Вычисляем недостающие данные: $P_{n+1} = \pi_{X^*}(\sigma_{F_{n+1}}(R_1 \bowtie \cdots \bowtie R_l))$, $M(F_{n+1}) = M(F^*) \setminus M(F_1 \ \&\ F_2 \ \&\ \ldots \ \&\ F_n)$. Делаем запрос к БД только на эти данные, помещаем их в кэш. По построению условие $M(F^*) \subseteq (M(F_1) \cap M(F_2) \cap \cdots \cap M(F_n) \cap M(F_{n+1}))$ теперь выполнено. $P^* \subseteq \pi_{X^*}(\sigma_{F^*[X]}(P_1 \bowtie \cdots \bowtie P_n)) \cup P_{n+1}$, и мы можем получить данные из кэша.

5.2. Примеры

Рассмотрим фрагмент схемы БД, представляющий учебный план в университете:

$$R_1 = \textit{Студенты}\ (\textbf{№ студента}, \text{ФИО}, \text{Группа})$$

$$R_2 = \textit{Группы}\ (\textbf{Группа}, \text{Факультет})$$

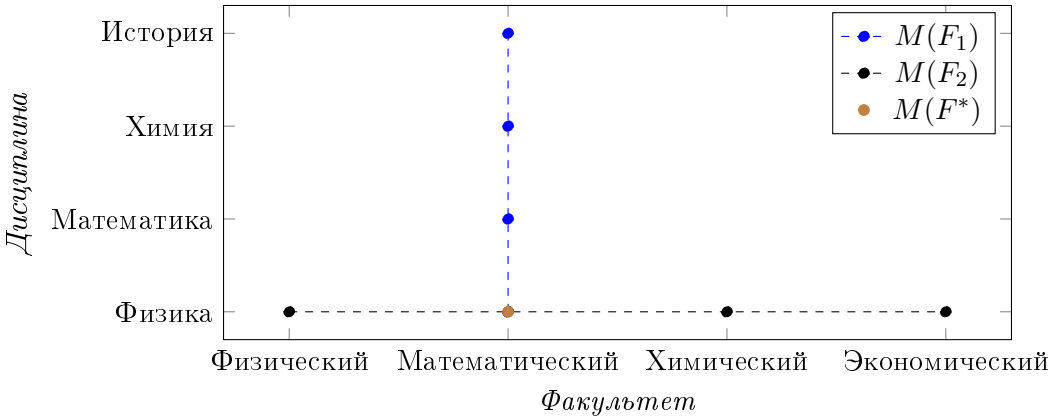

Рис. 1. Области истинности логических формул

$$R_3 = \textit{Успеваемость}\ (\textbf{№ студента}, \textbf{Дисциплина}, \text{Оценка})$$

Имена отношений выделены курсивом, их первичные ключи – жирным шрифтом.

Пример 1.

Предположим, что на компьютере пользователя сохранены результаты выполнения следующих запросов:

1. Список студентов математического факультета:

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

где π_{X_1} — проекция по множеству атрибутов X_1. $X_1 = \{\text{ФИО}, \text{Группа}\}$, σ — операция селекции, F_1 — логическая формула: $F_1 = (\text{Факультет} = \text{"Математический"})$.

В результате данного запроса будут получены те кортежи БД, которые удовлетворяют формуле F_1, причем с сервера будут запрошены только атрибуты ФИО, Группа.

2. Ведомости по физике:

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

где $X_2 = \{\text{ФИО}, \text{Группа}, \text{Оценка}\}$, F_2 — логическая формула: $F_2 = (\text{Дисциплина} = \text{"Физика"})$.

3. Отчет успеваемости по физике математического факультета:

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

где $X^* = \{\text{ФИО}, \text{Группа}, \text{Оценка}\}$, F^* — логическая формула: $F^* = (\text{Факультет} = \text{"Математический"}\ \&\ \text{Дисциплина} = \text{"Физика"})$.

Проверим возможность применения алгоритма. В нашем случае закэшированные данные - это два первых запроса: $P = \{P_1, P_2\}$. Целевой запрос – P^*. $X^* = \{\text{ФИО, Группа, Оценка}\}, X = X_1 \cup X_2 = \{\text{ФИО, Группа, Оценка}\} \Rightarrow X^* = X$. Таким образом, условие а) выполнено. $\{R_1, R_2\} \cup \{R_1, R_3\} = \{R_1, R_2, R_3\} \subseteq \{R_1, R_2, R_3\}$. Следовательно, условие б) также выполняется.

Выполняем пункт первый алгоритма. Необходимо понять, есть ли вообще что-то общее у закэшированных запросов и целевого запроса, т.е. проверить условие: $M(F^*) \cap (M(F_1) \cup M(F_2)) = \emptyset$. Как показано в пункте 4, данное условие эквивалентно $F^*\ \&\ (F_1 \vee F_2) \equiv \mathit{FALSE}$. Подставляя формулы запросов, получаем: Факультет = "Математический" & Дисциплина = "Физика" & (Факультет = "Математический" $\vee$

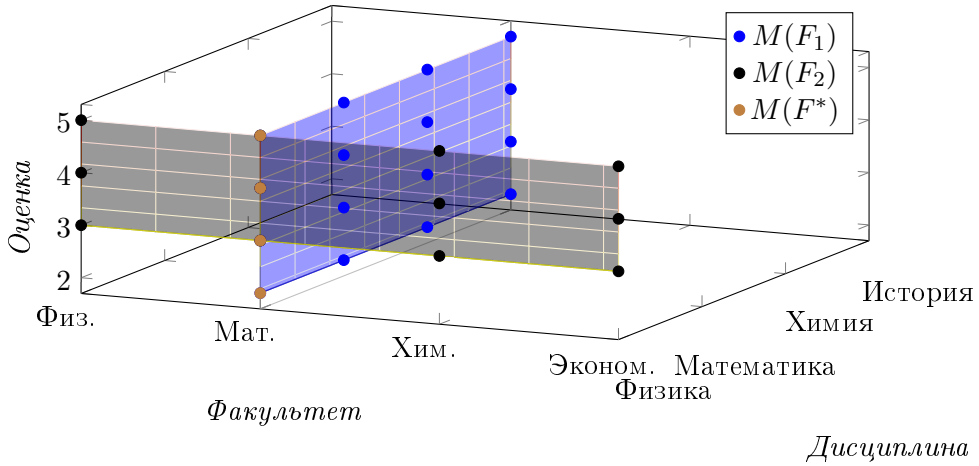

Рис. 2. Области истинности логических формул

Дисциплина = "Физика") $\equiv$ *FALSE*. Применим свойство дистрибутивности: Факультет = "Математический" & Дисциплина = "Физика" $\equiv$ *FALSE*. Это, очевидно, неверно.

Выполняем второй шаг алгоритма. Необходимо проверить условие $M(F^*) \subseteq (M(F_1) \cap M(F_2))$, что, как опять же показано в 4, равносильно F_1 & $F_2 \vee \neg F^* \equiv TRUE$. Подставляя данные, получаем: Факультет = "Математический" & Дисциплина = "Физика" $\vee$ $\neg$(Факультет = "Математический" & Дисциплина = "Физика") $\equiv$ *TRUE*. Данное выражение эквивалентно формуле $A \vee \neg A$ и является тождественно истинным при любом значении A. Условие выполнено, алгоритм завершен.

На рис. 1 изображены области истинности формул F_1, F_2, F^* спроецированные на плоскость (Факультет, Дисциплина). Из рисунка видно, что область истинности $M(F^*)$, представляющая из себя точку в данном пространстве, включается в пересечение областей истинности $M(F_1)$, $M(F_2)$.

Пример 2.

Попробуем немного усложнить задачу. Изменим логическую формулу запроса 2: F_2 = (Дисциплина = "Физика" & Оценка $\geq$ 3). Теперь запрос имеет смысл «список студентов, сдавших физику».

Условие первого пункта слегка меняется, но результат, очевидно, остается прежним – области истинности по-прежнему пересекаются. Факультет = "Математический" & Дисциплина = "Физика" & (Факультет = "Математический" $\vee$ Дисциплина = "Физика" & Оценка $\geq$ 3) $\equiv$ *FALSE*. Применим свойство дистрибутивности: Факультет = "Математический" & Дисциплина = "Физика" $\vee$ Факультет = "Математический" & Дисциплина = "Физика" & Оценка $\geq 3 \equiv$ *FALSE*. Это неверно, так как дизъюнкция 2 выражений может быть тождественно ложной лишь в случае, если оба выражения тождественно ложны.

На втором шаге встречаем различия с прошлым примером. Новые условия формулы F_2 сужают область истинности, и включения не наблюдается. Факультет = "Математический" & Дисциплина = "Физика" & Оценка $\geq$ 3 $\vee$ (Факультет $\neq$ "Математический" $\vee$ Дисциплина $\neq$ "Физика") $\equiv$ *TRUE*. Как уже было сказано в прошлом примере, такое выражение может быть тождественно истинным лишь в случае противоположных выражений. Теперь это условие нарушено. Условие не выполнено, переходим на шаг 3 алгоритма.

Третий шаг. Для того, чтобы было возможно воспользоваться кэшем, необходимо сделать запрос на недостающие данные. По сути эти данные удовлетворяют требованию «кортежи, которые есть в целевом запросе, но отсутствующие в кэше». Формализация этого требования выглядит, как это было описано в схеме алгоритма, следующим образом: $P_3 = \pi_{X^*}(\sigma_{F_3}(R_1 \bowtie R_2 \bowtie R_3))$, $F_3 = F^* \ \& \ \neg(F_1 \ \& \ F_2)$. Подставляя данные, получаем: F_3 = Факультет = "Математический" & Дисциплина = "Физика" & $\neg$(Факультет = "Математический" & (Дисциплина = "Физика" & Оценка $\geq$ 3)) $\sim$ Факультет = "Математический" & Дисциплина = "Физика" & (Факультет $\neq$ "Математический" $\vee$ Дисциплина $\neq$ "Физика" & Оценка = 2) $\sim$ Факультет = "Математический" & Дисциплина = "Физика" & Оценка = 2. Таким образом, для того, чтобы воспользоваться кэшем, необходимо получить данные о студентах математического факультета, не сдавших физику. В других данных необходимости нет, они уже содержатся в хранилище.

На рис. 2 теперь показана проекция областей истинности на трехмерное пространство (Факультет, Дисциплина, Оценка). Наглядно видна область пересечения областей истинности формул F_1 и F_2. Она не включает точку (Мат., Физика, 2), соответствующую данным, которые мы и получим, выполнив шаг 3 алгоритма.

Заключение

В данной статье рассмотрена проблема разработки методов аналитического сравнения логических ограничений пользовательских запросов к реляционной базе данных. Предложен подход, основанный на сравнении областей истинности запросов с использованием аппарата реляционного исчисления. В результате разработан алгоритм использования кэша, позволяющий получать запрашиваемые данные, не прибегая к запросам на сервер или совершать запросы лишь на часть требуемых данных, экономя сетевой трафик и время на выполнение запроса. Данные результаты решают 2 важных задачи использования технологии, описанной в статье [12]:

1. Аналитическая проверка возможности использования кэша
2. Аналитическое определение недостающих данных

Данные пункты отражены в схеме алгоритма в шагах 1 и 3 соответственно.

Проблема актуализации данных не затрагивается в этой работе. Однако она может быть решена путем учета запросов на сервере и обновлении данных при помощи триггеров.

В дальнейшем планируется применить методы и алгоритмы, изложенные в данной статье для разработки ПО, являющегося прослойкой над СУБД и управляющего использованием кэша, снижая тем самым количество передаваемых данных и общее время работы системы.

Предложенная технология будет использована при динамическом построении многомерных данных. Промежуточные представления имеют ту же структуру данных, что и таблицы соединений, используемые для построения гиперкубов. Сохраненные представления данных могут храниться на компьютере пользователя-аналитика и существенно сократить время на формирование данных, необходимых для принятия решений.

Работа выполнена по проекту ОМН РАН № II.2.П/I.1-7.

Литература

1. Afrati F.N., Li C., Mitra P. Rewriting queries using views in the presence of arithmetic comparisons // Theor. Comput. Sci. 2006. Vol. 368, No. 1-2. P. 88–123. DOI: 10.1016/j.tcs.2006.08.020.

2. 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. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1997. P. 156–165.

3. Church A. Introduction to Mathematical Logic. [S. l.] : Princeton University Press, 1996. 378 p.

4. Date C.J. SQL and Relational Theory. [S. l.] : O'Reilly, 2009. 448 p.

5. Denny M., Franklin M.J. Predicate Result Range Caching for Continuous Queries // Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. SIGMOD '05. New York, NY, USA : ACM, 2005. P. 646–657. DOI: 10.1145/1066157.1066231.

6. Gupta H. Selection of Views to Materialize in a Data Warehouse // Proceedings of the 6th International Conference on Database Theory. ICDT '97. London, UK, UK : Springer-Verlag, 1997. P. 98–112.

7. Gupta H., Mumick I.S. Selection of Views to Materialize Under a Maintenance Cost Constraint // Proceedings of the 7th International Conference on Database Theory. ICDT '99. London, UK, UK : Springer-Verlag, 1999. P. 453–470.

8. Hilbert D., Ackermann W. Principles of Mathematical Logic. [S. l.] : AMS Chelsea Publishing, 1950. 172 p.

9. Kalnis P., Papadias D. Proxy-Server Architectures for OLAP // SIGMOD Conference / Ed. by Sharad Mehrotra, Timos K. Sellis. [S. l.] : ACM, 2001. P. 367–378. DOI: 10.1145/375663.375712.

10. Keller A.M., Basu J. A Predicate-based Caching Scheme for Client-Server Database Architectures // VLDB J. 1996. Vol. 5, No. 1. P. 35–47.

11. Mendelson E. Introduction to Mathematical Logic. 5 edition. [S. l.] : CRC Press, 2009. 469 p.

12. Mosin S., Zykin S. Truth space method for caching database queries // Modeling and Analysis of Information Systems. 2015. Vol. 22, No. 2. P. 248–258.

13. Olston C., Jiang J., Widom J. Adaptive Filters for Continuous Queries over Distributed Data Streams // Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. SIGMOD '03. New York, NY, USA : ACM, 2003. P. 563–574. DOI: 10.1145/872757.872825.

14. Park C., Kim M., Lee Y. Usability-based caching of query results in OLAP systems // Journal of Systems and Software. 2003. Vol. 68, No. 2. P. 103–119. DOI: 10.1016/S0164-1212(02)00142-5.

15. Scheuermann P., Shim J., Vingralek R. WATCHMAN: A Data Warehouse Intelligent Cache Manager // Proceedings of the 22th International Conference on Very Large Data Bases. VLDB '96. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1996. P. 51–62.

16. Shim J., Scheuermann P., Vingralek R. Dynamic Caching of Query Results for Decision Support Systems. // Proceedings of 11th International Conference on Scientific and Statistical Database Management (SSDBM). [S. l. : s. n.], 1999. P. 254–263.

Мосин Сергей Владимирович, аспирант, инженер-исследователь лаборатории методов преобразования и представления информации, Институт математики им. С.Л. Соболева СО РАН (Омск, Российская Федерация), svmosin@gmail.com

Поступила в редакцию 25 августа 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2016, vol. 5, no. 1, pp. 85–99

DOI: 10.14529/cmse160108

TRUTH SPACE COMPARISON OF RELATIONAL DATABASE QUERIES

S.V. Mosin, Sobolev's Institute of Mathematics, Omsk, Russian Federation

We propose new methods and algorithms of analytical truth space comparison for Relational Database queries. Such a comparison aims to define the possibility of partial or full cache usage. The cache is stored on user's computer and Database server is supposed to be remote. In case user query's result is contained in cache we can simply take the data from there avoiding any server requests. The suggested method may also be used for defining data missing in cache and performing query to only receive that data. Analytical computations are also used to achieve this and it differs our approach from existing ones. Query comparison algorithms are based on the Predicate Logic theory. Formulas are presented by logical constraints in SQL and predicates correspond to SQL operations.

Keywords: relational database, cache, truth space.

FOR CITATION

Mosin S.V. Truth Space Comparison of Relational Database Queries. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 1. pp. 85–99. (in Russian) DOI: 10.14529/cmse160108.

References

1. Afrati F.N., Li C., Mitra P. Rewriting queries using views in the presence of arithmetic comparisons. Theor. Comput. Sci. 2006. vol. 368, no. 1-2. pp. 88–123. DOI: 10.1016/j.tcs.2006.08.020.
2. 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). San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1997. pp. 156–165.
3. Church A. Introduction to Mathematical Logic. [S. l.] : Princeton University Press, 1996. 378 p.
4. Date C.J. SQL and Relational Theory. [S. l.] : O'Reilly, 2009. 448 p.

5. Denny M., Franklin M.J. Predicate Result Range Caching for Continuous Queries. Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD '05). New York, NY, USA : ACM, 2005. pp. 646–657. DOI: 10.1145/1066157.1066231.

6. Gupta H., Mumick I. Selection of Views to Materialize in a Data Warehouse. Proceedings of the 6th International Conference on Database Theory (ICDT '97). London, UK, UK : Springer-Verlag, 1997. pp. 98–112.

7. Gupta H. Selection of Views to Materialize Under a Maintenance Cost Constraint. Proceedings of the 7th International Conference on Database Theory (ICDT '99). London, UK, UK : Springer-Verlag, 1999. pp. 453–470.

8. Hilbert D., Ackermann W. Principles of Mathematical Logic. [S. l.] : AMS Chelsea Publishing, 1950. 172 p.

9. Kalnis P., Papadias D. Proxy-Server Architectures for OLAP. Proceedings of the SIGMOD Conference / Ed. by Sharad Mehrotra, Timos K. Sellis. [S. l.] : ACM, 2001. pp. 367–378. DOI: 10.1145/375663.375712.

10. Keller A.M., Basu J. A Predicate-based Caching Scheme for Client-Server Database Architectures. The International Journal on Very Large Data Bases. 1996. vol. 5, no. 1. pp. 35–47.

11. Mendelson E. Introduction to Mathematical Logic. 5 edition. [S. l.] : CRC Press, 2009. 469 p.

12. Mosin S., Zykin S. Truth space method for caching database queries. Modeling and Analysis of Information Systems. 2015. vol. 22, no. 2. pp. 248–258.

13. Olston C., Jiang J., Widom J. Adaptive Filters for Continuous Queries over Distributed Data Streams. Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD '03). New York, NY, USA : ACM, 2003. pp. 563–574. DOI: 10.1145/872757.872825.

14. Park C., Kim M., Lee Y. Usability-based caching of query results in OLAP systems. Journal of Systems and Software. 2003. vol. 68, no. 2. pp. 103–119. DOI: 10.1016/S0164-1212(02)00142-5.

15. Scheuermann P., Shim J., Vingralek R. WATCHMAN: A Data Warehouse Intelligent Cache Manager. Proceedings of the 22th International Conference on Very Large Data Bases (VLDB '96). San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1996. pp. 51–62.

16. Shim J., Scheuermann P., Vingralek R. Dynamic Caching of Query Results for Decision Support Systems. Proceedings of 11th International Conference on Scientific and Statistical Database Management (SSDBM). [S. l. : s. n.], 1999. pp. 254–263.

Received August 25, 2015.