УДК 004.652

Алгоритм использования кэша запросов к реляционной базе данных

С. В. Мосин*

В данной статье предлагается подход для кэширования запросов к реляционной базе данных, основанный на анализе логических формул запросов, и представлен алгоритм, определяющий для заданного запроса, может ли он быть выполнен с использованием кэша. В случае если данных не хватает, они могут быть получены с сервера. При этом реализована проверка того, не превосходит ли объем недостающих данных объема запроса, так как в таком случае использовать кэш нецелесообразно. Упомянутая проверка основывается на использовании результатов по оценке мощности операции естественного соединения. В данной работе получена улучшенная оценка для естественных соединений с логическими ограничениями.

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

1. 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 множества 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 используются 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 кеше, т.е. анализируются номера блоков, а не их содержимое.

Целью данной работы является построение алгоритма получения искомого представления данных с использованием кэша. Проблема актуализации данных не затрагивается в данной статье.

Раздел 2 посвящен обзору публикаций на схожие темы: использование кэша при работе с различными видами баз и хранилищ данных. В разделе 3 описывается формализованная постановка задачи. Раздел 4 излагает основные результаты данной работы. Здесь представлена исходная версия алгоритма, позволяющего получать данные, запрашиваемые пользователем, из закэшированных представлений данных. В следующем разделе, 5, приведена оценка мощности запросов с учетом некоторых предположений о данных, а также представлена оптимизированная версия алгоритма. В заключении сделано обобщение полученных результатов и приведены дальнейшие направления исследований.

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

2. 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 [15, 7], однако различные по сравнению с настоящей работой цели приводят к различным результатам.

Наиболее близка к рассматриваемой проблеме работа [2]. 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 существует два основных подхода для резервирования результатов запросов: статический [3, 9, 10] и динамический [17, 18, 12, 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 [13] 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 [14], 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.

3. 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:

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

$$K_i = T_1^i \,\&\, T_2^i \ldots \&\, T_p^i(i), i = 1, \ldots, l, \tag{2}$$

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

Возможные значения T_j^i:

- 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 $\langle F \rangle$.

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

k – количество атрибутов, входящих в формулу

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 [14] 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 [14].

В соответствии с данными определениями легко понять, как будут устроены операции над областями истинности. $M(F)$ для некоторой формулы F, заданной своей ДНФ, 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 $TRUE$.

Утверждение 1 (Свойство включения). $\forall X \subseteq \langle F \rangle \quad M(F) \subseteq M(F[X])$.

Содержимое кэша обозначим $P = \{P_1, P_2, \ldots, P_n\}$. Его элементы будем называть *промежуточными представлениями данных* (или просто *промежуточными представлениями*):

$P_v = \pi_{X_v}(\sigma_{F_v}(R_1^v \bowtie R_2^v \bowtie \cdots \bowtie R_{s(v)}^v))$, $s(v)$ – количество отношений в базе данных, использованных при формировании представления P_v, π_{X_v} – операция проекции по множеству атрибутов X_v, σ_{F_v} – операция селекции с логическим ограничением на кортежи F_v. Целевое выражение, которое надо будет получить из представлений P, запишем в виде:

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

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

Доказательство приводится в статье [14].

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^*. Эти кортежи могут быть 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) $\langle F^* \rangle \subseteq X$.

Доказательство также приведено в статье [14].

Утверждение 2. Предположим, $\Re_1 = R_1 \bowtie \cdots \bowtie R_k$ – естественное соединение некоторых k отношений. Также предположим, что $\Re_2 = R_1 \bowtie \cdots \bowtie R_k \bowtie R_{k+1} \bowtie \cdots \bowtie R_n$. Тогда $\Re_2[\langle \bowtie_{i=1}^{k} R_i \rangle] \subseteq \Re_1$

Доказательство утверждения также можно найти в статье [14].

Теперь мы можем четко сформулировать цель данной работы. Она заключается в алгоритмическом решении проблемы использования кэша в условиях выполнения условий теоремы 1. Этому будут посвящены следующие разделы.

4. Получение требуемых данных из промежуточных представлений

Результаты, полученные в теореме 1, нельзя сразу использовать на практике. Отношение P^* содержится в сохраненных представлениях, однако неизвестно, какие именно кортежи являются частью P^*. Лишние кортежи могут возникнуть по следующим причинам:

1. Из-за невыполнения требования свойства операции проекции на каких-либо кортежах из множеств X_v.
2. В силу утверждения 1, так как правая часть содержит проекцию исходной формулы F^*

на некоторое множество атрибутов.

3. В силу утверждения 2, так как в промежуточном представлении отношений не больше, чем в искомом P^*.

Эти причины независимы. Это значит, что лишние кортежи, возникшие по одной из них, вообще говоря, отличаются от кортежей, возникших по другой.

Решение проблемы возникновения лишних кортежей предлагается сделать сведением к теореме 2 путем получения недостающих данных с сервера.

Введем следующие обозначения:

$\mathbf{P} = \{P_1, \ldots, P_n\}$,

$\mathbf{R}_v = \{R_1^v, \ldots, R_{s(v)}^v\}, v = 1, \ldots, n$,

$\mathbf{R}' = \bigcup\limits_{v=1}^{n} \mathbf{R}_v$,

$\mathbf{R}^* = \{R_1^*, \ldots, R_l^*\}$,

$\overline{\mathbf{R}'} = \mathbf{R}^* \setminus \mathbf{R}'$ – множество отношений, отсутствующих в промежуточных представлениях $\mathbf{P}$, но присутствующих в пользовательском запросе.

$Y = \langle \mathbf{R}' \rangle \cap \langle \overline{\mathbf{R}'} \rangle$ – множество атрибутов, по которым отношения из $\mathbf{P}$ пересекаются с отсутствующими отношениями. По всем этим атрибутам должна быть осуществлена операция естественного соединения, чтобы гарантировать избавление от всех избыточных данных.

$query$ – функция, осуществляющая запрос к серверу БД на получение данных пользовательского запроса.

Алгоритм 1. Получение искомого представления данных с использованием кэша.

Вход: $\mathbf{P}$, $\mathbf{R}^*$, F^*, $X^* : X^* \subseteq X$, $\mathbf{R}' \subseteq \mathbf{R}^*$, $M(F^*) \subseteq M(F_v), v = 1, \ldots, n$

Выход: $R = P^*$

for $v = 1$ **to** n:

$X' = X_v$

if $\langle \mathbf{R}_v \rangle \cap (\bigcup\limits_{\substack{w=1 \\ w \neq v}}^{n} \langle \mathbf{R}_w \rangle) \not\subseteq X_v$**:** $\quad X' = X' \cup \langle \mathbf{R}_v \rangle \cap (\bigcup\limits_{\substack{w=1 \\ w \neq v}}^{n} \langle \mathbf{R}_w \rangle)$

if $Y \cap \langle \mathbf{R}_v \rangle \not\subseteq X_v$**:** $\quad X' = X' \cup Y \cap \langle \mathbf{R}_v \rangle$

if $\langle F^* \rangle \cap \langle \mathbf{R}_v \rangle \not\subseteq X_v$**:** $\quad X' = X' \cup \langle F^* \rangle \cap \langle \mathbf{R}_v \rangle$

if $X' \neq X_v$**:** $\quad P_v = query(\pi_{X'}(\sigma_{F_v}(\underset{R \in \mathbf{R}_v}{\bowtie} R)))$

$P_{n+1} = query(\pi_{Y \cup \langle F^* \rangle \cap \langle \overline{\mathbf{R}'} \rangle}(\sigma_{F^*[\langle \overline{\mathbf{R}'} \rangle]}(\underset{R \in \overline{\mathbf{R}'}}{\bowtie} R)))$

$\mathbf{R} = \pi_{X^*}(\sigma_{F^*}(P_1 \bowtie \cdots \bowtie P_n \bowtie P_{n+1}))$

Поясним шаг алгоритма 1.

Для каждого сохраненного представления данных выполняется набор проверок.

1. Проверка условия $\langle \mathbf{R}_v \rangle \cap (\bigcup\limits_{\substack{w=1 \\ w \neq v}}^{n} \langle \mathbf{R}_w \rangle) \not\subseteq X_v$. Данное условие соответствует невыполнению условия а) теоремы 1 и призвано устранить первую причину возникновения лишних кортежей. Если оно выполнено, помещаем в X' недостающие атрибуты.
2. Проверка условия $Y \cap \langle \mathbf{R}_v \rangle \not\subseteq X_v$. Благодаря этому условию мы можем присоединить недостающие атрибуты к промежуточному представлению, что позволит провести естественное соединение с недостающими отношениями, которые будут получены далее.
3. Проверка условия $\langle F^* \rangle \cap \langle \mathbf{R}_v \rangle \not\subseteq X_v$. Оно позволяет определить, содержатся ли в X_v все атрибуты, необходимые для проведения селекции по формуле F^*. Это условие призвано устранить вторую причину возникновения лишних кортежей.
4. Если хотя бы одна из проверок прошла, то изначальное множество атрибутов X' будет пополнено некоторым множеством атрибутов. В этом случае выполняется запрос к

БД на получение представления, отличающегося от текущего расширенным множеством атрибутов.

После выполнения проверки всех промежуточных представлений выполняется запрос на получение недостающих отношений из множества $\overline{\mathbf{R}'}$.

Легко видеть, что данный алгоритм может осуществлять довольно большое количество запросов к базе данных с целью получения недостающих данных. Если объем данных, запрошенных с сервера, будет сравним с объемом данных, требуемых для выполнения пользовательского запроса, использование кэша станет нецелесообразным. В связи с этим необходимо добавить в алгоритм сравнение объема недостающих данных и данных пользовательского запроса. В случае если первый становится равен или близок по значению ко второму, стоит просто получить требуемые данные с сервера напрямую, так как кэш в данном случае бесполезен.

5. Оценка мощности запросов

5.1. Обзор существующих методов

В публикациях рассматриваются различные подходы к получению оценки мощности результата операции естественного соединения. В работе [1] используется предположение о равномерном распределении атрибутов при оценке мощности соединения отношений (u-метод). В работе [8] предложен метод получения наихудшей (пессимистической) оценки для соединения двух отношений. В работе [6] рассмотрен метод полного знания информации (pk-метод). В работах [4, 5, 8, 1] рассматриваются различные варианты кусочно-равномерных методов (pu-методы) и алгоритмы для оценки мощности соединения двух и более отношений. Последней работой по данной тематике является статья [11], в которой сравниваются различные варианты метода, основанного на случайной выборке кортежей по распределению Бернулли и последующей оценке размера соединения для отобранных кортежей.

В нашем случае обращение к БД является нежелательным, так как это, очевидно, приведет к снижению эффективности кэша (основная цель которого – избежать лишних обращений к базе данных). Поэтому подходы, аналогичные pk- и pu-методам, неприемлемы для получения статической оценки мощности соединения. Случайная выборка кортежей требует наличия полной версии таблицы, но для этого необходимо обеспечить в кэше полное хранение копии всей базы данных, что практически всегда невозможно из-за больших объемов данных.

Из сказанного следует, что в общем случае наиболее целесообразно использование предположения о равномерном распределении значений атрибутов на этапе проектирования БД. Наилучшей оценкой для значений любой случайной величины является математическое ожидание, в нашем случае – среднее. Кроме того, как показано в экспериментальной оценке в работе [19], в предположении равномерности наиболее вероятны значения мощности соединения, близкие к среднему, а пессимистическая оценка маловероятна. При получении среднего значения будем пользоваться предположением о равномерности – равной вероятности появления любого возможного значения атрибута. Кроме того, потребуются предположения о статистической независимости значений атрибутов в каждом отношении и независимости значений общих атрибутов в различных отношениях.

Приведенные далее оценки получены и обоснованы в работе [19].

5.2. Оценка мощности естественных соединений

Пусть $|R_i| = N_i, i = 1, \ldots, M$ – мощности отношений, $A(C(m)) = \{A_1, A_2, \ldots, A_p\}$ – множество атрибутов, принадлежащих, по крайней мере, одному пересечению $R_i \cap R_j$, где

$i, j \in C(m)$. $C(m) = \{j_1, j_2, ..., j_m\}$ – совокупность номеров отношений, являющаяся сочетанием без повторений по m элементов из множества $I = \{1, 2, ..., M\}$, $|S(C(m))||$ – мощность естественного соединения отношений $R_i, i \in C(m)$. $k(A_j, C(m))$ – количество различных значений атрибута A_j, являющихся общими для всех отношений R_i, $i \in C(m)$, которые содержат атрибут A_j. В частности, $k(A_j, i)$ – количество различных значений атрибута A_j только в реализации R_i. $R_i \cap A(C(m)) = X_i(C(m))$. Тогда оценка мощности соединения отношений с индексами $C(m)$ принимает следующий вид:

$$|S(C(m))| = \frac{\prod\limits_{j=1}^{p} k(A_j, C(m)) \prod\limits_{i \in C(m)} N_i}{\prod\limits_{i \in C(m)} \prod\limits_{A_j \in X_i(C(m))} k(A_j, i)}. \quad (4)$$

Оценка (4) является обобщением оценки в [1] для мощности результата операции естественного соединения.

Из формулы видно, что для получения оценки необходимы лишь размеры таблиц, участвующих в соединении, и мощности общих атрибутов. Эти данные легко получить и можно также закэшировать и обновлять периодически, так как зачастую они меняются незначительно (при больших объемах отношений и/или нечастых операциях записи).

5.3. Корректировка оценки при выполнении селекции

Для оценки реальных запросов к БД необходимо учитывать логические ограничения на естественное соединение, или условия $WHERE$ языка SQL. В силу свойства операций селекции и проекции логической формулы имеем:

$$\sigma_F(R_1 \bowtie \cdots \bowtie R_M) \subseteq \sigma_{F[\langle R_1 \rangle]}(R_1) \bowtie \cdots \bowtie \sigma_{F[\langle R_M \rangle]}(R_M),$$

где формула F задана (1), (2). Мощность $\sigma_{F[\langle R_1 \rangle]}(R_1) \bowtie \cdots \bowtie \sigma_{F[\langle R_m \rangle]}(R_m)$ можно оценить по формуле 4, где в качестве отношений $R_i, i = 1, \ldots, M$ выступают $\sigma_{F[\langle R_i \rangle]}(R_i), i = 1, \ldots, M$.

Для оценки нам потребуются дополнительные сведения о базе данных – списки всех значений атрибутов. Обозначим $V(R_i.A_j) = \{v_1, \ldots, v_{k(A_j,i)}\}$ – множество значений атрибута A_j в таблице R_i.

Рассмотрим, как изменятся величины, входящие в (4). Оценить $k(A_j, C(m))$ не представляется возможным, за исключением случая, когда какой-либо из атрибутов A_j участвует в формуле F, так как у нас нет данных о взаимном распределении атрибутов в отношениях. В случае же $A_j \in \langle F \rangle$ данные $V(R_i.A_j), i \in C(m)$ позволяют пересчитать значения $k(A_j, C(m))$. Обозначим $R'_i = \sigma_{F[\langle R_i \rangle]}(R_i)$. Тогда $V(R'_i.A_j) = \{v \in V(R_i.A_j) \mid F[\langle R_i \rangle](v) = TRUE\}$.

Оценка $k(A_j, C(m))$ принимает вид:

$$k'(A_j, C(m)) = \begin{cases} |\bigcap\limits_{i \in C(m)} V(R'_i.A_j)|, & \text{если } A_j \in \langle F \rangle \\ k(A_j, C(m)), & \text{иначе} \end{cases}. \quad (5)$$

Для атрибутов, которые не входят явным образом в формулу F, данные величины остаются в оценке без изменений.

Далее рассмотрим, как изменятся мощности отношений. Для оценки этих значений воспользуемся информацией о виде логической формулы F и теми же списками значений атрибутов $V(R_i.A_j)$.

По свойству операции селекции

$$\sigma_F(R) = \sigma_{T_1^1}(\ldots(\sigma_{T^1_{p(1)}}(R))) \cup \cdots \cup \sigma_{T_1^l}(\ldots(\sigma_{T^l_{p(l)}}(R))),$$

где F задана (1), (2). Очевидно, что

$$|\sigma_{T_1^1}(\ldots(\sigma_{T_{p(1)}^1}(R)))| \le \min(|\sigma_{T_1^1}(R)|,\ldots,|\sigma_{T_{p(1)}^1}(R)|).$$

Тогда:

$$|\sigma_F(R)| = |\sigma_{T_1^1}(\ldots(\sigma_{T_{p(1)}^1}(R)))| + \cdots + |\sigma_{T_1^l}(\ldots(\sigma_{T_{p(l)}^l}(R)))| \le$$
$$\le \min(|\sigma_{T_1^1}(R)|,\ldots,|\sigma_{T_{p(1)}^1}(R)|) + \cdots + \min(|\sigma_{T_1^l}(R)|,\ldots,|\sigma_{T_{p(l)}^l}(R)|). \quad (6)$$

Задача свелась к оценке значений мощностей отношений, ограниченных элементарным предикатом: $|\sigma_{T_k^j}(R_i)|, i \in C(m); j = 1,\ldots,l; k = 1,\ldots,p(j)$. Предикаты $T_j^i, i = 1,\ldots,l, j = 1,\ldots,p(i)$ могут принимать значения, описанные в разд. 3. Во всех приведенных в списке предикатах языка SQL значение атрибута может сравниваться либо с заданной константой, либо с другим атрибутом. В последнем случае мы не сможем провести оценку по тем же причинам, что и при оценке $k(A_j, C(m))$, и будем оставлять мощности отношений без изменений. В случае же с константными ограничениями оценки элементарно получаются, основываясь на данных $V(R_i.A_j)$, а также на данных о количестве всех значений, принимаемых всеми атрибутами всех отношений: $H_{ij} : V(R_i.A_j) \to \mathbb{Z}^*$.

Пример 5.1. Пусть $T = (R_1.A_5 > 16)$. Тогда $|\sigma_T(R_1)| = \sum\limits_{\substack{v \in V(R_1.A_5) \\ v > 16}} H_{15}(v)$.

Обозначим через $T(i)_k^j$ предикаты, составляющие формулы $F[\langle R_i\rangle], i \in C(m)$. Тогда в силу (6):

$$|\sigma_{F[\langle R_i\rangle]}(R_i)| \le \sum_j \min_k(|\sigma_{T(i)_k^j}(R_i)|) = N_i', i \in C(m). \quad (7)$$

Подставляя оценки (5) и (7) в (4), получаем:

$$|\sigma_F(S(C(m)))| \le \frac{\prod\limits_{j=1}^{p} k'(A_j, C(m)) \prod\limits_{i \in C(m)} N_i'}{\prod\limits_{i \in C(m)} \prod\limits_{A_j \in X_i(C(m))} k'(A_j, i)}. \quad (8)$$

5.4. Улучшенная версия алгоритма

Далее приводится улучшенный алгоритм, учитывающий требуемый объем дополнительных данных при определении возможности использования кэша.

Как и прежде, будем обозначать:

$\mathbf{P} = \{P_1,\ldots,P_n\}$,

$\mathbf{R}_v = \{R_1^v,\ldots,R_{s(v)}^v\}, v = 1,\ldots,n$,

$\mathbf{R}' = \bigcup\limits_{v=1}^{n} \mathbf{R}_v$,

$\mathbf{R}^* = \{R_1^*,\ldots,R_l^*\}$,

$\overline{\mathbf{R}'} = \mathbf{R}^* \setminus \mathbf{R}'$ – множество отношений, отсутствующих в промежуточных представлениях $\mathbf{P}$, но присутствующих в пользовательском запросе.

$Y = \langle\mathbf{R}'\rangle \cap \langle\overline{\mathbf{R}'}\rangle$ – множество атрибутов, по которым отношения из $\mathbf{P}$ пересекаются с отсутствующими отношениями. По всем этим атрибутам должна быть осуществлена операция естественного соединения, чтобы гарантировать избавление от всех избыточных данных.

$query$ – функция, осуществляющая запрос к серверу БД на получение данных пользовательского запроса.

Введем новые обозначения:

size – функция получения размера запроса. Эта функция применяется только к запросам, сохраненным в кэше, так что дополнительных запросов на сервер не требуется.

estimate – функция оценки размера запроса по формуле (8). На вход она получает массив отношений, на выход дает числовую оценку.

$S = \mathit{estimate}(P^*)$ – оценка мощности целевого запроса.

$\sigma \in [0, 1]$ – коэффициент, задающий предел использования кэша. Если $\sigma = 1$, кэш будет использован вплоть до ситуации, когда объем требуемых дополнительных данных равен оцениваемому объему данных пользовательского запроса.

Алгоритм 2. Получение искомого представления данных с использованием кэша

Вход: $\mathbf{P}$, $\mathbf{R}^*$, F^*, X^*, S, σ
Выход: $R = P^*$
$\mathbf{X} = [...]$
$s = 0$
for $v = 1$ **to** n:
    $X' = X_v$
    if $\langle \mathbf{R}_v \rangle \cap (\bigcup\limits_{\substack{w=1 \\ w \neq v}}^{n} \langle \mathbf{R}_w \rangle) \not\subseteq X_v$**:** $\quad X' = X' \cup \langle \mathbf{R}_v \rangle \cap (\bigcup\limits_{\substack{w=1 \\ w \neq v}}^{n} \langle \mathbf{R}_w \rangle)$
    if $Y \cap \langle \mathbf{R}_v \rangle \not\subseteq X_v$**:** $\quad X' = X' \cup Y \cap \langle \mathbf{R}_v \rangle$
    if $\langle F^* \rangle \cap \langle \mathbf{R}_v \rangle \not\subseteq X_v$**:** $\quad X' = X' \cup \langle F^* \rangle \cap \langle \mathbf{R}_v \rangle$
    if $X' \neq X_v$**:**
        $s = s + \mathit{size}(P_v)$
        $\mathbf{X}[v] = X'$
if $s > \delta S$**:** $\quad R = \mathit{query}(P^*)$
else:
    for $i = 0$ **to** N:
        if $\mathbf{X}[i] \neq \emptyset$**:** $\quad P_i = \mathit{query}(\pi_{X'}(\sigma_{F_i}(\underset{R \in \mathbf{R}_i}{\bowtie} R)))$
$P_{n+1} = \mathit{query}(\pi_{Y \cup \langle F^* \rangle \cap \langle \overline{\mathbf{R}'} \rangle}(\sigma_{F^*[\langle \overline{\mathbf{R}'} \rangle]}(\underset{R \in \overline{\mathbf{R}'}}{\bowtie} R)))$
$\mathbf{R} = \pi_{X^*}(\sigma_{F^*}(P_1 \bowtie \cdots \bowtie P_n \bowtie P_{n+1}))$

Алгоритм 2 имеет ту же вычислительную сложность, что и алгоритм 1. В ходе первого цикла запоминаются все размеры соединений, которые будет необходимо получить с сервера. В случае если итоговый объем требуемых данных не превышает некоторого процента от размера пользовательского запроса (σ), то эти данные получаются с сервера, в остальном используются закэшированные данные. В противном случае кэш не используется.

Заключение

В данной статье рассмотрена проблема использования кэшированных данных и её алгоритмическое решение. Предложен алгоритм, позволяющий получить недостающие данные, чтобы точно определить, какие закэшированные кортежи являются частью пользовательского запроса. Далее предложена оптимизированная версия данного алгоритма, использующая оценку мощности естественного соединения для определения целесообразности использования кэша.

Оценку мощности запросов к серверу можно улучшить, если брать в расчет логические ограничения запросов, так как они могут очень сильно влиять на объем запрашиваемых данных.

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

В дальнейшем планируется применить алгоритм, изложенный 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.

Литература

1. *P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin et al.* Access Path Selection in a Relational Database Management System // Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data. SIGMOD ’79. New York, NY, USA : ACM, 1979. P. 23–34.
2. *Afrati Foto N., Li Chen, Mitra Prasenjit*. Rewriting queries using views in the presence of arithmetic comparisons // Theor. Comput. Sci.. 2006. Vol. 368, № 1-2. P. 88–123.
3. *Baralis Elena, Paraboschi Stefano, Teniente Ernest*. 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.
4. *Bell David A, McClean Sally*. Pragmatic estimation of join sizes and attribute correlations // Data Engineering, 1989. Proceedings. Fifth International Conference on / IEEE. 1989. P. 76–84.
5. *Chao Tian-Jy, Egyhazy Csaba J.* Estimating Temporary Files Sizes in Distributed Relational Database Systems // Proceedings of the Second International Conference on Data Engineering. Washington, DC, USA : IEEE Computer Society, 1986. P. 4–12.
6. *Christodoulakis Stavros* Estimating Block Transfers and Join Sizes // Proceedings of the 1983 ACM SIGMOD International Conference on Management of Data. SIGMOD ’83. New York, NY, USA : ACM, 1983. P. 40–54.
7. *Denny Matthew, Franklin Michael 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.
8. *Epstein Robert S., Stonebraker Michael*. Analysis of Distributed Data Base Processing Strategies // Proceedings of the Sixth International Conference on Very Large Data Bases. VLDB ’80. VLDB Endowment, 1980. Vol. 6. P. 92–101.
9. *Gupta Himanshu*. 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.
10. *Gupta Himanshu, Mumick Inderpal Singh*. 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.
11. *David Vengerov, Andre Cavalheiro Menck, Mohamed Zait, Sunil P. Chakkappen*. Join Size Estimation Subject to Filter Conditions // Proc. VLDB Endow. 2015. Vol. 8, № 12. P. 1530–1541.
12. *Kalnis Panos, Papadias Dimitris*. Proxy-Server Architectures for OLAP // SIGMOD

Conference / Ed. by Sharad Mehrotra, Timos K. Sellis. ACM, 2001. P. 367–378.
13. *Keller Arthur M., Basu Julie*. A Predicate-based Caching Scheme for Client-Server Database Architectures // VLDB J. 1996. Vol. 5, № 1. P. 35–47.
14. *Mosin Sergey, Zykin Sergey*. Truth space method for caching database queries // Modeling and Analysis of Information Systems. 2015. Vol. 22, № 2. P. 248–258.
15. *Olston Chris, Jiang Jing, Widom Jennifer*. 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.
16. *Park Chang-Sup, Kim Myoung-Ho, Lee Yoon-Joon*. Usability-based caching of query results in OLAP systems. // Journal of Systems and Software. 2003. Vol. 68, № 2. P. 103–119.
17. *Scheuermann Peter, Shim Junho, Vingralek Radek*. 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.
18. *Shim Junho, Scheuermann Peter, Vingralek Radek*. Dynamic Caching of Query Results for Decision Support Systems // SSDBM. 1999. P. 254–263.
19. *Зыкин Сергей Владимирович*. Разработка и исследование моделей данных и средств организации взаимодействия пользователей с информационными ресурсами : Дисс... доктора наук : 06.12.05. ИМ СО РАН им. С. Л. Соболева. Новосибирск, 2005. 245 с.

Статья поступила в редакцию 15.01.2016;
переработанный вариант – 22.03.2016.

Мосин Сергей Владимирович
аспирант, инженер-исследователь лаборатории методов преобразования и представления информации ИМ СО РАН (644043, г. Омск, ул. Певцова, 13), e-mail: svmosin@gmail.ru.

Cache usage algorithm for RDB queries

S. Mosin

This paper considers the approach for caching RDB queries based on the query logical conditions. An algorithm defining whether or not this query can be performed using cache is also presented. If any data is missed it can be retrieved from the server. The data size to be requested is estimated and compared with the one required for straight query execution because if the first is greater than the latter cache usage is not worthwhile. Improved estimation for joint operations with logical constraints is also proposed in this paper and is used for checking mentioned before.

Keywords: database, cache, query.