Методика повышения производительности крупных информационных систем за счет реструктуризации данных на основе кластерного анализа статистики запросов

И. В. Бельченко

Кубанский государственный технологический университет
350072, Краснодар, ул. Московская, 2

e-mail: ilur@mail.ru

Аннотация. Большинство крупных информационных систем и программных комплексов используют реляционные базы данных для хранения информации. Разнообразие запросов на чтение информации от клиентских приложений делает задачу выбора структуры базы данных весьма трудоемкой и требующей применения формальных методов анализа. Статья посвящена разработке методики реструктуризации табличных структур данных крупных информационных систем на основе кластерного анализа статистики запросов. Для формализации предметной области выделены параметры и множества, влияющие на скорость обработки запросов, на чтение информации к исследуемой таблице базы данных. Рассмотрены существующие подходы к повышению производительности баз данных. Сформулирована задача оптимизации количества блоков данных, необходимых для обработки группы запросов на чтение информации. Предложена методика поиска субоптимального разбиения исследуемой таблицы на основе кластерного анализа статистики запросов. Проведена экспериментальная апробация методики. Полученные результаты могут быть использованы при проектировании СУБД.
Ключевые слова: система поддержки принятия решений, оптимизация, кластерный анализ, нечеткая логика, структуры данных, базы данных, системный анализ.

1. Введение

Задача повышения производительности крупных информационных систем особенно актуальна в связи с общей тенденцией к глобализации и централизации в сфере информационных систем и технологий. Существуют несколько общепринятых подходов к решению данной задачи: улучшение характеристик аппаратной составляющей информационных систем, повышение качества программных комплексов и компонент, оптимизация хранилищ и баз данных.

Наращивание мощностей аппаратной составляющей информационных систем является наименее трудоемким и наукоемким процессом и чаще всего заключается

в масштабировании вычислительных комплексов, повышении пропускной способности вычислительных сетей и т. д. При решении задачи выбора новой конфигурации аппаратного обеспечения чаще всего учитываются стоимость реконфигурации, а также возрастающая потребность в энергоресурсах [1]. Повышение качества программных комплексов и компонент — ресурсоемкий процесс, требующий временных и финансовых затрат. При выборе этого подхода как решения проблемы повышения производительности крупных информационных систем лицо, принимающее решение (ЛПР), сталкивается с множеством слабоструктурированных задач, начинающихся с анализа требований к информационной системе и заканчивающихся подбором конечных исполнителей — разработчиков.

Данное исследование посвящено подходу, основанному на оптимизации хранилищ и баз данных. Так как большинство WEB-систем, мобильных приложений, настольного программного обеспечения используют для хранения данных реляционные базы данных (БД), то задача уменьшения среднего времени выполнения запросов на чтение информации является актуальной и значимой.

Существуют несколько подходов к уменьшению среднего времени выполнения запросов на чтение информации к БД:

- подход, основанный на статистическом анализе группы запросов на чтение информации и получении оптимального набора индексов таблиц для минимизации времени выполнения запроса [2];
- денормализация БД — приведение структуры базы данных в состояние, не соответствующее критериям нормализации, проводимое с целью ускорения операций чтения из базы за счет добавления избыточных данных [3];
- секционирование данных — основной смысл состоит в физическом размещении частей одной таблицы по разным файлам, взаимодействие с которыми на уровне файловой системы и дискового массива осуществляется параллельно [4];
- рефакторинг табличных структур, который основан на их вертикальном разделении [5].

Вопрос рефакторинга табличных структур и его целесообразности в научном сообществе поднимается довольно часто, особенно для систем оперативной обработки транзакций (OLTP). Но все методики сводятся к рекомендациям для проектировщиков БД и являются умозрительными, четкой методики разделения исследуемой таблицы на дочерние нет.

В статье рассмотрена методика повышения производительности информационной системы за счет уменьшения среднего времени выполнения группы запросов на чтение информации базы данных. От структуры данных, способах ее физическо-

го размещения на жестких дисках зависит количество обращений к дисковым накопителям, которые сопровождаются соответствующими прерываниями и задержками по времени. На скорость поиска информации в БД влияют: объем блока в байтах, объем файла, количество записей в блоке файла, количество записей в блоке индекса, количество блоков в файле, доля резервной части блока, число полей в записи, размер записи в байтах [6].

2. Постановка задачи рефакторинга табличных структур данных

Процесс построения оптимальной модели данных информационной системы включает оптимальное вертикальное распределение таблиц базы данных по блокам на дисковом накопителе. Основным критерием оптимизации модели данных информационной системы является минимальный размер строки таблицы реляционной базы данных, позволяющий в одном блоке хранить больше данных, и как следствие, минимизировать количество операций чтения блоков данных с жесткого диска при выполнении запросов к базе данных. Это достигается за счет уменьшения объема данных, побочно участвующих в запросе [7].

В рамках методики предлагается разделить таблицы базы данных на несколько сущностей, связанных отношением один к одному. В соответствии с принципами блочного хранения данных в СУБД каждая таблица будет храниться в отдельном наборе блоков. При выполнении запроса на чтение информации СУБД считывает блоки данных с жесткого диска в оперативную память каждой таблицы, атрибуты которой участвуют в запросе.

Задача повышения производительности информационной системы сводится к поиску оптимального разделения табличных структур базы данных с учетом конкретной группы запросов на чтение информации, выявленной статистически в рамках жизненного цикла БД.

3. Оптимизация табличных структур данных информационной системы

Для формализации задачи рассмотрим множества и параметры, влияющие на скорость обработки запросов на чтение информации к исследуемой таблице базы данных.

1. Целочисленный параметр TS, равный количеству атрибутов в исследуемой таблице.

2. Вектор типов данных $DBT = \{dbt_{idbt} \mid idbt = \overline{1, ndbt}\}$, которые поддерживаются конкретной выбранной СУБД. Элемент вектора — занимаемый элементом типа размер данных в байтах памяти.

3. Набор атрибутов (столбцов таблицы) TA, который задан бинарной матрицей, элемент которой $ta_{ita,\,jta}$ равен единице, если столбец ita таблицы имеет тип $jta, ita = 1, ..., TS,\ jta = 1, ..., ndbt.$

4. Множество, представляющее группу запросов $Q = \{q_{iq} \mid iq = \overline{1, nq}\}$ на чтение информации из таблицы базы данных, элемент множества кортеж из двух элементов $q_{iq} = \{SFQ_{iq}, QA_{iq}\}$, где SFQ_{iq} — числовой параметр, равный частоте появления запроса за выбранный период времени, $QA_{iq} = \{qa_{iqa} \mid iqa = \overline{1, TS}\}$ — бинарный вектор, размерность которого равна количеству атрибутов таблицы TS; $qa_{iqa} = 1$, если атрибут таблицы TA участвует в запросе, и 0, в противном случае; nq — количество запросов в статистической выборке, выявленной в рамках жизненного цикла БД.

Множество индексов, характеризующихся набором полей таблицы, по которым построен индекс $IN = \{in_{iin} \mid iin = \overline{1, nin}\}.$ Элемент множества $in_{iin} = in_{iin,\,jin} \mid jin = \overline{1, TS}\}$ — бинарный вектор, размерность которого равна количеству атрибутов таблицы TS, $in_{iin,\,jin} = 1$, TS, если атрибут jin таблицы TA участвует в индексе in_{iin} и 0 — в противном случае.

Хранимые процедуры и функции $PF = \{pf_{ipf} \mid ipf = \overline{1, npf}\}$, характеризующиеся набором полей, используемых в теле хранимой процедуры или функции. Элемент множества $pf_{ipf} = \{pf_{ipf,\,jpf} \mid jpf = \overline{1, TS}\}$ — бинарный вектор, размерность которого равна количеству атрибутов таблицы TS, $pf_{ipf,\,jpf} = 1$, если атрибут jpf таблицы TA участвует в теле хранимой процедуры или функции pf_{ipf} и 0 — в противном случае.

Множество триггеров базы данных $TG = \{tg_{itg} \mid itg = 1, ntg\}$, характеризующихся набором полей таблицы, используемых в теле триггера. Элемент множества $tg_{itg} = \{tg_{itg,\,jtg} \mid jtg = \overline{1, TS}\}$ — бинарный вектор, размерность которого равна количеству атрибутов таблицы TS, $tg_{itg,\,jtg} = 1$, если атрибут jtg таблицы TA участвует в теле триггера tg_{itg} и 0 — в противном случае.

4. Анализ влияния количества физических блоков данных, используемых таблицей базы данных, на общее время выполнения группы запросов к ней

Множество запросов Q к рассматриваемой таблице обрабатывается СУБД за время $T(Q, TA, DBT)$. Временные затраты $T(Q, TA, DBT)$ можно представить в виде суммы временных затрат на чтение блоков данных таблиц $T_h(Q, TA, DBT)$, участвующих в запросах Q, и остальных временных затрат $T_O(Q, TA, DBT)$, к которым относятся временные затраты на выполнение плана обработки запроса, на передачу информации и т. д.

$$T(Q, TA, DBT) = T_h(Q, TA, DBT) + T_O(Q, TA, DBT).$$

В рамках методики предлагается уменьшить слагаемое, влияющее на общее время выполнения запроса $T_h(Q, TA, DBT)$. Временные затраты $T_h(Q, TA, DBT)$ в общем виде зависят от количества операций чтения блоков данных таблиц с жесткого диска. Пусть временная задержка, связанная с считыванием одного блока данных, равна T_b, тогда

$$T_h(Q) = \left(\sum_{iq}^{nq} B(q_{iq}, TA, DBT) \right) \cdot T_b,$$

где $B(q_{iq}, TA, DBT)$ — число информационных блоков, которые необходимо считать с жесткого диска в кэш СУБД для дальнейшего выполнения запроса q_{iq} к таблице, заданной бинарной матрицей TA. Кэш СУБД находится в оперативной памяти вычислительного устройства; T_b — временная задержка, связанная со считыванием одного блока данных.

Функция $B(q_{iq}, TA, DBT)$ вычисляется как

$$B(q_{iq}, TA, DBT) = RC \cdot RS(q_{iq}, TA, DBT) / DB,$$

где RC — количество строк в рассматриваемой таблице.

$$RS(q_{iq}, TA, DBT) = RSS(q_{iq}, TA, DBT) + RST(q_{iq}, TA, DBT);$$

$RS(q_{iq}, TA, DBT)$ — величина, характеризующая дисковое пространство, занимаемое одной строкой таблицы в байтах; $RSS(q_{iq}, TA, DBT)$ — количество памяти, занимаемое служебными отметками СУБД для строки, считываемое при выполнении запроса q_{iq}; $RST(q_{iq}, TA, DBT)$ — количество памяти, занимаемое атрибутами таблицы в строке, считываемое при выполнении запроса q_{iq}. DB; DB — фиксиро-

ванный размер блока данных выбранной СУБД. В большинстве СУБД он равен 8Кб. Параметры RC и DB остаются неизменными.

Так как временную задержку T_b, связанную со считыванием одного блока данных, допускается считать постоянной величиной, на сумму временных затрат на чтение блоков данных таблицы TA — $T_h(Q, TA, DBT)$ влияет количество блоков, необходимое для считывания, которое вычисляется как функция $F(Q, TA, DBT)$:

$$F(Q, TA, DBT) = \sum_{iq}^{nq} B(q_{iq}, TA, DBT).$$

Подставим в формулу $F(Q, TA, DBT)$ формулу функции $B(q_{iq}, TA, DBT)$. Функция, определяющая количество блоков, необходимых для считывания с жесткого диска в оперативную память при выполнении множества запросов Q к рассматриваемой таблице TA:

$$F(Q, TA, DBT) = \sum_{iq}^{nq} RC \cdot RS(q_{iq}, TA, DBT) / DB.$$

5. Формулировка целевой функции и структурных ограничений

В рамках методики предлагается разделить рассматриваемую таблицу на $NB \in [1, TS]$ дочерних таблиц, связанных с родительской отношением один к одному, 1:1.

Введем следующую переменную:

$$x_{ij} = \begin{cases} 1, \text{ если } j \text{ – атрибут, необходимо выделить в } i\text{-ю таблицу,} \\ 0, \text{ в противном случае.} \end{cases}$$

Переменная представляет собой бинарную матрицу для таблицы реляционной базы данных размерностью $TS \times TS$, где TS — количество атрибутов таблицы. Строки матрицы соответствуют таблицам, на которые разбивается родительская таблица, а столбцы соответствуют их атрибутам.

Количество блоков $BM(Q, RC, DB, DBT, TA, X)$, которое необходимо считать с жесткого диска для выполнения множества запросов Q к таблице TA, вычисляется как функция, равная сумме блоков, которые необходимо считать с жесткого диска, для выполнения множества запросов Q к каждой из дочерних таблиц. Максимальное количество дочерних таблиц равно числу атрибутов родительской таблицы TA и равно NB.

$$BM(Q, RC, DB, DBT, TA, X) =$$
$$= \sum_{iq=1}^{nq}\left[RC \cdot RSM(DBT, TA, X_1) \cdot FQ(q_{iq}, X_{1)}/DB\right] + ...$$
$$+ \sum_{iq=1}^{nq}\left[RC \cdot RSM(DBT, TA, X_{irs}) \cdot FQ(q_{iq}, X_{irs)}/DB\right] + ...$$
$$+ \sum_{iq=1}^{nq}\left[RC \cdot RSM(DBT, TA, X_{nb}) \cdot FQ(q_{iq}, X_{nb)}/DB\right],$$

где $SM(DBT, TA, X_{irs}) = \left(\sum_{j}^{TS}\left[x_{irs,j} \cdot \left(\sum_{idbt}^{ndbt} DBT_{idbt} \cdot TA_j\right)\right] + RDS(DBT, TA, X_{irs})\right).$

$$F(q_{iq}, X_{irs}) = \begin{cases} q_{iq}(SFQ), \text{ если } \sum_{u}^{TS}(X_{irs})_u \cdot q_{iq}(QA))_u > 0, \\ 0, \text{ в противном случае,} \end{cases}$$

$$irs = 1, ..., nb,\ j = 1, ..., TS,\ idbt = 1, ..., ndbt.$$

Параметры RC и DB являются постоянными, RCM — функция, характеризующая количество байт информации, занимаемое одной строкой дочерней таблицы irs, $RDS(DBT, TA, X_{irs})$ — функция, характеризующая дисковое пространство, занимаемое служебными отметками СУБД в строке дочерней таблицы irs в байтах.

Следовательно, задача повышения производительности системы сводится к поиску такого разделения таблицы на дочерние, при котором сумма блоков, которые необходимо считать в КЭШ СУБД для выполнения множества запросов Q, минимальнf [7].

Целевая функция:

$$\min_{x_{irs,j}}\left[\sum_{iq,irs}\frac{1}{DB} \cdot \sum_{j}^{TS}\left[x_{irs,j} \cdot \left(\sum_{idbt}^{ndbt} dbt_{idbt} \cdot ta_j\right)\right] + RDS(DBT, TA, x_{irs}))) \cdot RC \cdot FQ(q_{iq}, x_{irs})\right],$$

где

$$F(q_{iq}, x_{irs}) = \begin{cases} 1, \text{ если } \sum_{u}^{TS}(x_{irs})_u \cdot (q_{iq}(QA))_u > 0, \\ 0, \text{ в противном случае,} \end{cases}$$

$$irs = 1, ..., nb;\ \ j = 1, ..., TS;\ idbt = 1, ..., ndbt.$$

При следующихструктурных ограничениях.

1. Каждый атрибут родительской таблицы может присутствовать только в одной дочерней таблице.

$$\sum_{m_1} x_{m_1,i_1} = 1,\ m_1 = 1, ..., TS,\ i_1 = 1, ..., TS.$$

2. Атрибуты таблицы, используемые при построении индексов, должны принадлежать хотя бы одной дочерней таблице.

$$\forall ix,\ ix = 1, ..., |IN|: \prod_{m_2}^{TS}\left[\sum_{i_2}^{TS}(x_{m_2,i_2}\cdot in_{ix,i_2} - in_{ix,i_2})\right] = 0,$$

$$m_2 = 1, ..., TS,\ i_2 = 1, ..., TS.$$

3. Атрибуты таблицы, используемые в теле хранимых процедур или функций, должны принадлежать хотя бы одной дочерней таблице.

$$\forall px,\ ix = 1, ..., |PF|: \prod_{m_2}^{TS}\left[\sum_{i_3}^{TS}(x_{m_3,i_3}\times pf_{px,i_3} - pf_{px,i_3})\right] = 0,$$

$$m_3 = 1, ..., TS,\ i_3 = 1, ..., TS.$$

4. Атрибуты таблицы, используемые в работе триггеров исследуемой таблицы, должны принадлежать хотя бы одной дочерней таблице.

$$\forall tx,\ tx = 1, ..., |TG|: \prod_{m_4}^{TS}\left[\sum_{i_4}^{TS}(x_{m_4,i_4}\times tg_{tx,i_4} - g_{tx,i_4})\right] = 0,$$

$$m_4 = 1, ..., TS,\ i_4 = 1, ..., TS.$$

5. Отношение количества физических блоков данных, необходимого для хранения данных рассматриваемой таблицы до применения к количеству блоков, необходимого для хранения данных в полученных после применения методики дочерних таблицах, не должно превышать заданного параметра $TSIZE$, $TSIZE \in (0;1]$.

$$TSIZE = \frac{\sum_{iq}^{nq}(RC\cdot RS(q_{iq}, TA, DBT)/DB)}{\left(\sum_{j}^{TS}\left(\left[x_{irs,j}\cdot\sum_{idbt}^{ndbt}(dbt_{idbt}\times ta_j)\right] + RDS(DBT, TA, x_{irs})\right)\cdot RC\times FQ(q_{iq}, x_{irs})\right)/DB}.$$

Методика нахождения оптимального разделения таблицы на дочерние для выполнения группы запросов на основе кластерного анализа статистики запросов.

Целевая функция не линейна, а также не линейны ограничения. Переменная X — бинарная матрица размерностью $TS\times TS$. Представим переменную в виде машинного слова длиной $TS\times TS$. Следовательно, количество возможных комбинаций переменной определяется как $2^{TS\times TS}$. Исходя из этого, задача обладает экспоненциальной сложностью и является NP-трудной.

Для решения задачи разработана методика, основанная на кластерном анализе группы запросов к БД.

Эвристический алгоритм состоит из следующих этапов.

Этап 1. Рассмотрим бинарное нечеткое отношение R на множестве $ATR = \{atr_1, atr_2, ..., atr_{TS}\}$, заданное в виде функции принадлежности

$$R = \{\mu_R(atr_{a_1}, atr_{a_2}) \mid atr_{a_1}, atr_{a_2} \in ATR\},$$

где

$$\forall \mu_R(atr_{a_1}, atr_{a_2}) = \sum_{iq}((q_{iq}(QA))_{a_1} \cdot (q_{iq}(QA))_{a_2} \cdot (q_{iq}(SFQ));$$

$iq = 1, ..., nq,\ a_1, a_2 \in [1, TS].$

Функция принадлежности определяет во скольких запросах атрибуты atr_{a_1}, atr_{a_2} встречаются вместе. Отношение R обладает следующими свойствами:

1. Рефлексивно:

$$\mu_R(atr_{a_1}, atr_{a_1}) = 1,\ \forall atr_{a_1} \in ATR.$$

2. Симметрично:

$$\mu_R(atr_{a_1}, atr_{a_2}) = \mu_R(atr_{a_2}, atr_{a_1}),\ \forall atr_{a_1}, atr_{a_2} \in ATR.$$

Алгоритм автоматической кластеризации требует, чтобы отношение R обладало свойством $(\max - \min)$ транзитивности:

$$\mu_R(atr_{a_1}, atr_{a_2}) \geq (\mu_R(atr_{a_1}, atr_k) \wedge \mu_R(atr_k, atr_{a_2})), \forall atr_{a_1}, atr_{a_2}, atr_k \in ATR.$$

$$\mu_R\left(atr_{a_1}, atr_{a_2}\right) \geq \bigvee_{atr_k \in ATR}\left(\mu_R\left(atr_{a_1}, atr_k\right) \wedge \mu_R\left(atr_k, atr_{a_2}\right)\right),$$

$$\forall\, atr_{a_1}, atr_{a_2}, atr_k \in ATR.$$

$(\max - \min)$ транзитивным замыканием бинарного нечеткого отношения R на множестве ATR, где $card(ATR) = TS$, называется бинарное нечеткое отношение $\breve{R}$ на множестве ATR, определяемое следующим образом [8]:

$$\breve{R} = R^1 \bigcup R^2 \bigcup ... \bigcup R^{TS},$$

где отношения R^{TS} определяются рекурсивно:

$$R^1 = R,\ R^{TS} = R^{TS-1} \circ R,\ TS = 2, 3, ...$$

Этап 2. После получения бинарного нечеткого транзитивного отношения $\breve{R}$, вычисляются α-уровни, $\alpha \in [0,1]$, выделяются кластеры $A^l, l \in [1, TS]$ в соответствии с правилом: если $\mu_{\breve{R}}(\alpha tr_{a_1}, \alpha tr_{a_2}) \geq \alpha$, для некоторых $\alpha tr_{a_1}, \alpha tr_{a_2} \in ATR$, то объекты $\alpha tr_{a_1}, \alpha tr_{a_2} \in ATR$ принадлежат кластеру A^l.

Этап 3. Из полученной иерархии разбиений выбирается разбиение, эффективность которого оценивается при помощи выведенной в рамках исследования целевой функции. Атрибуты, принадлежащие кластеру, выделяются в отдельную дочернюю таблицу [9].

7. Практическая апробация полученной методики

Для анализа эффективности полученной методики были выделены исходные данные и базовые множества.

1. Параметр *TS*, характеризующий количество столбцов в таблице, равен 16. Описание полей таблицы представлено в табл. 1.

Таблица 1. Описание полей таблицы

№ п/п	Наименование столбца	Тип данных СУБД
1	Id	bigint
2	Attr1	nchar(10)
3	Attr2	nchar(10)
4	Attr3	nchar(10)
5	Attr4	nchar(10)
6	Attr5	nchar(10)
7	Attr6	nchar(10)
8	Attr7	nchar(10)
9	Attr8	nchar(10)
10	Attr9	nchar(10)
11	Attr10	nchar(10)
12	Attr11	nchar(10)
13	Attr12	nchar(10)
14	Attr13	nchar(10)
15	Attr14	nchar(10)
16	KeyForSearch	nchar(10)

2. Множество типов данных DBT, которые поддерживаются конкретной выбранной СУБД MS SQL 2012. Задано вектором, характеризующим занимаемое типом данных дисковое пространство в байтах DBT={4, 8, 20}. Количество типов данных СУБД уменьшено для компактности статьи. Множество типов данных СУБД MS SQL 2012 представлено в табл. 2.

Таблица 2. Множество типов данных СУБД MS SQL 2012

№ п/п	Наименование типа данных	Занимаемое дисковое пространство в байтах
1	int	4
2	bigint	8
3	nchar(10)	20

3. Набор атрибутов (столбцов таблицы) $TA.$, показан в табл. 3.

Таблица 3. Набор атрибутов (столбцов таблицы) TA

	DBT_1	DBT_2	DBT_3
A_1	0	1	0
A_2	0	0	1
A_3	0	0	1
A_4	0	0	1
A_5	0	0	1
A_6	0	0	1
A_7	0	0	1
A_8	0	0	1
A_9	0	0	1
A_{10}	0	0	1
A_{11}	0	0	1
A_{12}	0	0	1
A_{13}	0	0	1
A_{14}	0	0	1
A_{15}	0	0	1
A_{16}	0	0	1

4. Множество, представляющее группу запросов Q на получение информации из базы данных, состоящее из 3 элементов. Характеристики запросов представлены в виде табл. 4.

Таблица 4. Набор атрибутов (столбцов таблицы) TA

№ п/п	**Количество запросов, поступивших на сервер за выбранный период времени**	**Бинарный вектор атрибутов, участвующих в запросе**
1	10	<0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1>
2	20	<1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1>
3	5	<1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1>

Для проведения эксперимента были исключены ограничения в виде индексов, триггеров и хранимых процедур, а также был исключен коэффициент дополнительного использования памяти. Это позволило продемонстрировать преимущества предлагаемой методики над традиционным подходом. В результате применения методики было предложено разделить исследуемую таблицу на 2 дочерние таблицы. Предлагаемое разделение представлено в виде табл. 5.

Таблица 5. Предлагаемое разделение исследуемой таблицы

№	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}
T_1	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0
T_2	1	1	1	1	1	1	0	0	1	1	0	1	1	1	1	1

Полученное решение было проверено статистически на всей группе запросов в ходе последовательного их выполнения в рамках 10 экспериментов для каждого набора данных исследуемой таблицы. Результаты экспериментов показаны в табл. 6.

Таблица 6. Предлагаемое разделение исследуемой таблицы

Количество строк в исследуемой таблице	Время выполнения группы запросов к исследуемой таблице, мс	Время выполнения группы запросов к таблицам после разделения на дочерние, мс
2400000	221384	184060
4800000	493866	415993
9600000	1142071	961270

Исследование влияния количества строк в таблице на эффективность применения методики позволяет судить об ее эффективности для крупных информационных систем. Оценка эффективности применения методики как зависимости времени обработки группы запросов от количества строк исследуемой таблицы представлена на графике, который изображен на рис. 1. Оценка повышения эффективности применения методики, выраженная в процентах, лается на рис. 2.

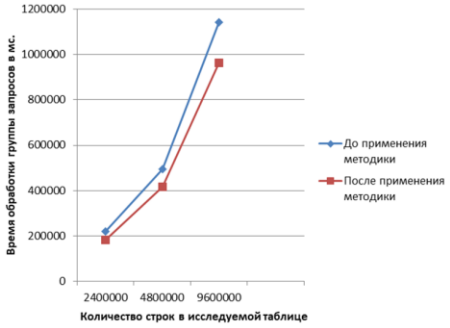

Рисунок 1. График оценки эффективности применения методики в зависимости от количества строк исследуемой таблицы

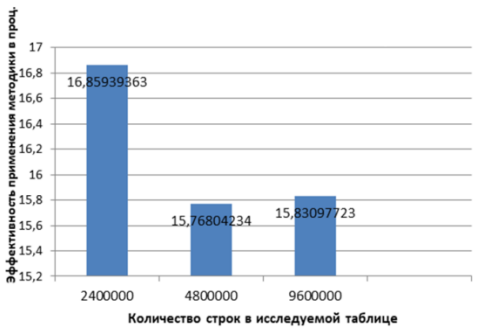

Рисунок 2. Оценка повышения эффективность от применения методики, выраженная в процентах

7. Заключение

В результате исследования была сформулирована проблема повышения производительности крупных информационных систем. Проведен анализ существующих подходов к решению данной проблемы. Среди возможных решений выделены и проанализированы методики повышения производительности СУБД и хранилищ данных. Рассмотрена возможность рефакторинга табличных структур данных. Предложен подход к нахождению субоптимального разбиения исследуемой табличной структуры на дочерние путем кластерного анализа группы запросов на чтение информации. Предложенная методика особенно актуальна для таблиц БД, которые используют крупные информационные системы, так как методика кластеризации, основанная на транзитивном замыкании нечетких бинарных отношений, более гибко учитывает корреляции между наборами атрибутов, участвующих в за-

просах, нежели методика многомодального распределения по математическим ожиданиям появления запросов.

Полученные результаты могут быть использованы при проектировании отечественных СУБД. Дальнейшие исследования в этой области связаны с разработкой программного обеспечения для внедрения методики в существующие информационные системы и комплексы.

Литература

[1] *Гребенюк В. М.* О проблемах определения возможностей масштабирования сложных систем. — М. : Вестник евразийской науки, 2013.

[2] *Boronski R., Bocewicz G.* Relational Database Index Selection Algorithm // *Communications in Computer and Information Science*. 2014. Vol. 431. P. 338–347.

[3] *Тарасов С. В.* СУБД для программиста. Базы данных изнутри. — М. : СОЛОН-ПРЕСС, 2015.

[4] *Королева О. Н., Мажукин А. В., Королева Т. В.* Базы данных : курс лекций. — М. : Московский гуманитарный университет, 2012.

[5] *Эмблер С. В., Садаладж П. Дж.* Рефакторинг баз данных: эволюционное проектирование : пер. с англ. — М. : ООО «И.Д. Вильямс», 2007.

[6] *Тоу Д.* Настройка SQL. Для профессионалов. — СПБ. : Питер, 2004.

[7] *Чигаркина Е. И.* Базы данных: учеб. пособие. — Самара : Издательство СГАУ, 2015.

[8] *Вятченин Д. А.* Нечеткие методы автоматической классификации : монография. — Мн. : УП «Технопринт», 2004.

[9] *Atroshchenko V. A., Belchenko V. Ye., Belchenko I. V., Dyachenko R. A.* Development and research of statistical methods and optimization algorithms of search for solutions in intelligence automated systems // *International journal of pharmacy and technology*. 2016. Vol. 8. No. 2. P. 14137–14149.

***Автор*:**
Илья Владимирович Бельченко — аспирант кафедры информатики и вычислительной техники Института компьютерных систем и информационной безопасности, Кубанский государственный технологический университет

Methods of improving the performance of large information systems by restructuring data based on cluster analysis of query statistics

I. V. Belchenko

Kuban State Technological University
2, Moskovskaya st., Krasnodar, Russia, 350072

e-mail: ilur@mail.ru

*Abstract***.** The performance of the information system is related to the data structures that are used in it. Most large information systems and software complexes use relational databases to store information. The variety of requests for reading information from client applications makes the task of choosing the optimal structure of the database very laborious and requiring the application of methods of system analysis. The aim of the work is to develop a methodology for restructuring the table structures of data of large information systems based on the cluster analysis of query statistics.
Materials and methods. To formalize the domain, the parameters and sets that affect the processing speed of requests for reading information to the database table being researched are highlighted. Existing approaches to improving database performance are considered. The problem of optimization of the number of data blocks necessary for processing a group of requests for reading information is formulated. A method is proposed for searching for a suboptimal partitioning of the investigated table on the basis of a cluster analysis of query statistics. The algorithm of the technique is described.
Conclusions. An approach is proposed for finding a suboptimal partition of the investigated table structure into its children. The proposed methodology is especially relevant for database tables of large information systems. The experimental approbation of the technique is carried out. The obtained results can be used in the design of domestic DBMS..
Key words: decision support system, optimization, cluster analysis, fuzzy logic, data structures, databases, system analysis.

References

[1] *Grebenjuk V. M.* O problemah opredelenija vozmozhnostej masshtabirovanija slozhnyh system. Moscow, Vestnik evrazijskoj nauki, 2013. [In Rus]

[2] *Boronski R.*, *Bocewicz G.* (2014) *Communications in Computer and Information Science*, **431**:338–347.

[3] *Tarasov S. V.* (2015) SUBD dlja programmista. Bazy dannyh iznutri. Moscow. [In Rus]

[4] *Koroleva O. N.*, *Mazhukin A. V.*, *Koroleva T. V.* (2012) Bazy dannyh: kurs lekcij. Moscow, Moskovskij gumanitarnyj universitet. [In Rus]

[5] *Ambler S. W., Sadalage P. J.* (2006) Refactoring Databases: Evolutionary Database Design. Addison-Wesley

[6] *Tou D.* (2004) Nastrojka SQL. Dlja professionalov. SPB.: Piter. [In Rus]

[7] *Chigarkina E. I.* (2015) Bazy dannyh: ucheb. posobie. Samara, Izdatel'stvo SGAU. [In Rus]

[8] *Vjatchenin D. A.* (2004) Nechetkie metody avtomaticheskoj klassifikacii. Mn. UP Tehnoprint, 2004. [In Rus]

[9] *Atroshchenko V. A., BelchenkoV. Ye., Belchenko I. V., Dyachenko R. A.* (2016) *International journal of pharmacy and technology*, **8**(2):14137–14149.