УДК 004.9

Ю.А. Григорьев, Е.Ю. Ермаков

СРАВНЕНИЕ ПРОЦЕССОВ ОБРАБОТКИ ЗАПРОСА К ОДНОЙ ТАБЛИЦЕ В ПАРАЛЛЕЛЬНОЙ СТРОЧНОЙ И КОЛОНОЧНОЙ СИСТЕМЕ БАЗ ДАННЫХ

Приведены результаты сравнения процессов обработки запросов в строчной и колоночной СУБД. Показано преобразование Лапласа — Стилтьеса (ПЛС) времени обработки запроса с планом $\pi_A(\sigma_F(R))$ в этих СУБД. Приведены также результаты сравнения среднего времени выполнения запроса с указанным планом.

E-mail: iu5vmch@rambler.ru

***Ключевые слова*:** *параллельные строчные и колоночные базы данных, преобразование Лапласа — Стилтьеса, сравнение строчных и колоночных систем баз данных*

Введение. К настоящему времени во многих организациях накоплены колоссальные объемы данных, на основе которых можно решать самые разнообразные аналитические и управленческие задачи в любой сфере деятельности. Проблемы хранения и обработки аналитической информации становятся все более актуальными и привлекают внимание специалистов, работающих в области информационных технологий. Именно на решение этих задач направлены технологии, объединяющиеся под общим названием хранилища данных и бизнес-анализа. По оценке Gartner, хранилища в ближайшей перспективе останутся одними из ключевых компонентов автоматизированных информационных систем предприятий [1].

Несмотря на то, что классические реляционные хранилища обеспечивают наилучшее сочетание простоты, устойчивости, гибкости, производительности, масштабируемости и совместимости, их показатели по каждому из этих пунктов не обязательно выше, чем у аналогичных систем, ориентированных на какую-то одну особенность. Согласно Майклу Стоунбрейкеру, пионеру исследований в области больших баз данных [2], такая идея «безразмерности», когда традиционная архитектура СУБД, изначально разработанная и оптимизированная для обработки бизнес-данных, используется для поддержки приложений, требующих обработки больших объемов данных, больше не применима к рынку баз данных. Мир коммерческих СУБД будет дробиться на набор независимых, специализированных средств управления базами данных [3].

Одним из основных и самых перспективных архитектурных решений для специализированных СУБД в области хранилищ данных является колоночное хранение данных: большой потенциал колоночных систем подтверждают аналитические исследования и прогнозы аналитиков [1, 3—5]. Например, в работе [5] показано 200-кратное

сокращение объема ввода—вывода по сравнению с аналогичной реляционной СУБД (РСУБД). Это достигается за счет того, что из базы данных читаются только те атрибуты, которые участвуют в запросе, а также применяются эффективные методы сжатия столбцов.

Таким образом, перед проектировщиком системы обработки данных возникает непростая задача выбора между традиционными (строчными — Oracle, MS SQL Server и др.) и специализированными СУБД (колоночными — Vertica, ParAccel и др.). Для принятия обоснованного технического решения по выбору типа СУБД необходимо использовать средства моделирования. Для традиционных РСУБД такие методы уже существуют [6]. Для параллельных СУБД подобные исследования ведутся [7—10].

В работе приведены результаты сравнения процессов обработки запросов в параллельной строчной и колоночной системе баз данных, а также времени выполнения запроса к одной таблице на основе математических методов, предложенных авторами в работах [11, 12] и учитывающих особенности выполнения запросов к колоночным и строчным базам данных.

Организация работы строчной и колоночной системы баз данных. Под строчным хранением данных обычно понимается физическое хранение кортежа любого отношения в виде одной записи, в котором значения атрибута идут последовательно одно за другим, а за последним атрибутом кортежа в общем случае следует новый кортеж отношения.

Таким образом, на физическом носителе отношение R представлено в следующем виде:

$$\left[\dot{a}_{11}, \dot{a}_{21}, ..., \dot{a}_{n1}\right]_1 \left[\dot{a}_{12}, \dot{a}_{22}, ..., \dot{a}_{n2}\right]_2 \left[\dot{a}_{13}, \dot{a}_{23}, ..., \dot{a}_{n3}\right]_3 ... \left[\dot{a}_{1m}, \dot{a}_{2m}, ..., \dot{a}_{nm}\right]_m,$$

где $\dot{a}_{ij}$ — значение атрибута a_i в j-м кортеже отношения R; $\left[\dot{a}_{1j}, \dot{a}_{2j}, ..., \dot{a}_{nj}\right]_j$ — j-й кортеж отношения R; n — число атрибутов отношения R; $m = T(R)$ — число кортежей отношения R.

В колоночных СУБД значения одного атрибута хранятся последовательно друг за другом [13], т. е. на физическом носителе отношение R примет следующий вид:

$$\dot{a}_{11}, \dot{a}_{12}, \dot{a}_{13}, ..., \dot{a}_{1m} \;\; \dot{a}_{21}, \dot{a}_{22}, \dot{a}_{23}, ..., \dot{a}_{2m} ... \dot{a}_{n1}, \dot{a}_{n2}, \dot{a}_{n3}, ..., \dot{a}_{nm},$$

где $\dot{a}_{ij}$ — значение атрибута a_i в j-м кортеже отношения R; $\dot{a}_{i1}, \dot{a}_{i2}, \dot{a}_{i3}, ..., \dot{a}_{im}$ — i-й столбец (атрибут) отношения R.

Каждая колонка, хранимая на диске, разделена на блоки определенного размера ($S_б$). Блок состоит из заголовка, размер которого пренебрежимо мал по сравнению с размером блока и непосредственно данных. При одном запросе к диску происходит чтение нескольких блоков, количество которых определяется параметром.

Каждой записи в столбце сопоставляется ее позиция (номер строки). В большинстве современных колоночных баз данных значения столбца упорядочиваются по их позициям [14].

На логическом уровне колоночные и строчные СУБД идентичны, т. е. способны обрабатывать одни и те же SQL-запросы. Но отличия в физической организации хранения данных существенно влияют на реализацию процессов, протекающих при формировании плана выполнения запроса и его реализации.

В строчных СУБД план запроса представляет собой дерево, у каждого узла которого имеется один родитель и один (или два в случае пересечения) дочерних узла. Реализация исполнителя планов базируется на следующих трех базовых парадигмах [15]:

— синхронный конвейер;

— итераторная модель;

— скобочный шаблон.

Синхронный конвейер. Суть данного метода состоит в том, что, как только операция получает очередной кортеж своего результирующего отношения, она передает его по конвейеру выше стоящей операции (ОП1—ОП6) для обработки. Например, узел, читающий записи из исходной таблицы, передает их узлу, выполняющему соединение записей разных таблиц.

Итераторная модель. Эта модель является общепринятым методом, используемым в СУБД для эффективной реализации синхронного конвейера. В соответствие с итераторной моделью с каждым узлом дерева плана запроса связывается специальная структура управления, называемая итератором. Интерфейс итератора представлен двумя стандартными операциями с предопределенной семантикой:

— reset — установка итератора в состояние «перед первым кортежем»;

— next — выдать очередной кортеж результирующего отношения.

Алгоритм выполнения плана запроса на базе итераторной модели изображен на рис. 1. На первом шаге выполняется метод reset применительно к корневому узлу. Затем в цикле выполняется метод next для этого же узла. Он каждый раз возвращает указатель на очередной кортеж результирующего отношения. В приведенном примере эти кортежи просто выводятся на экран. Цикл завершается, когда метод next выдает указатель на специальный кортеж, обозначающий конец файла — EOF (End Of File). Методы reset и next родителя прямо или косвенно могут вызывать соответствующие методы дочерних узлов. Эти вызовы изображены на рисунке пунктирными стрелками. Реализация итератора базируется на скобочном шаблоне, который рассмотрен ниже.

Скобочный шаблон. Основными атрибутами скобочного шаблона являются:

— выходной буфер, в который помещается очередной кортеж результата (он может быть прочитан путем вызова родителем метода next соответствующего дочернего узла);

— КОП — код реляционной операции, реализуемой данным узлом;

— указатель на скобочный шаблон левого дочернего узла;

— указатель на скобочный шаблон правого дочернего узла («пусто» для унарных операций).

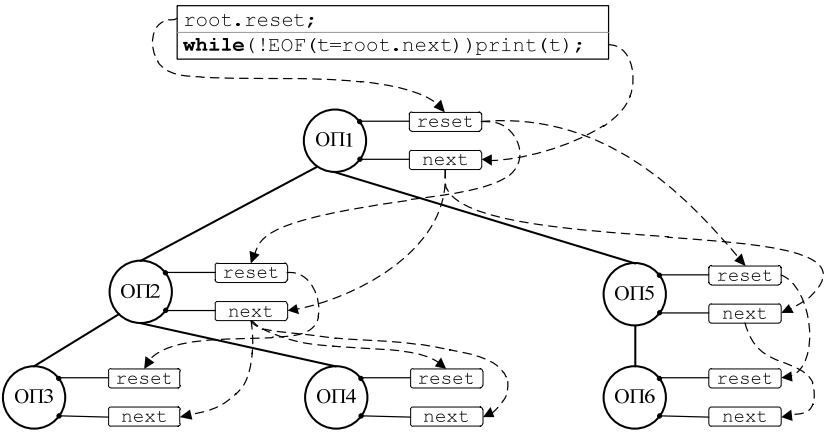

Рис. 1. Алгоритм выполнения плана запроса на базе итераторной модели

Сам по себе скобочный шаблон не содержит конкретной реализации реляционной операции. Однако после оптимизации запроса СУБД «вставляет» в каждый скобочный шаблон ту или иную реализацию соответствующей реляционной операции. Например, для операции соединения СУБД может выбрать один из следующих кодов: «соединение вложенными циклами», «соединение сортировкой и слиянием», «соединение хешированием». При выполнении этой операции узел обращается к скобочным шаблонам своих дочерних узлов.

Рассмотрим, какие изменения вносят колоночные базы данных в каждый из рассмотренных принципов.

Синхронный конвейер. В колоночных СУБД при реализации конвейера учитываются следующие особенности:

— в связи с фундаментальными отличиями в организации хранения информации на физическом носителе операции выполняются не над кортежами отношения, а над блоками атрибутов отношения;

— существует возможность проводить операции не над данными (блоками), а над позициями значений в этих блоках;

— между узлами конвейера могут передаваться как позиции, так и указатели на блоки данных.

Пример конвейера колоночного хранилища для простого запроса по двум атрибутам таблицы, использующего перечисленные выше возможности, представлен на рис. 2. Результатами операций 5 и 6 являются позиции значений в атрибутах. Списки полученных позиций передаются и обрабатываются (соединяются, пересекаются и др.) в операторе 4, результат действия которого попадает в операторы 2 и 3, считывающие указанные в полученном наборе позиций значения.

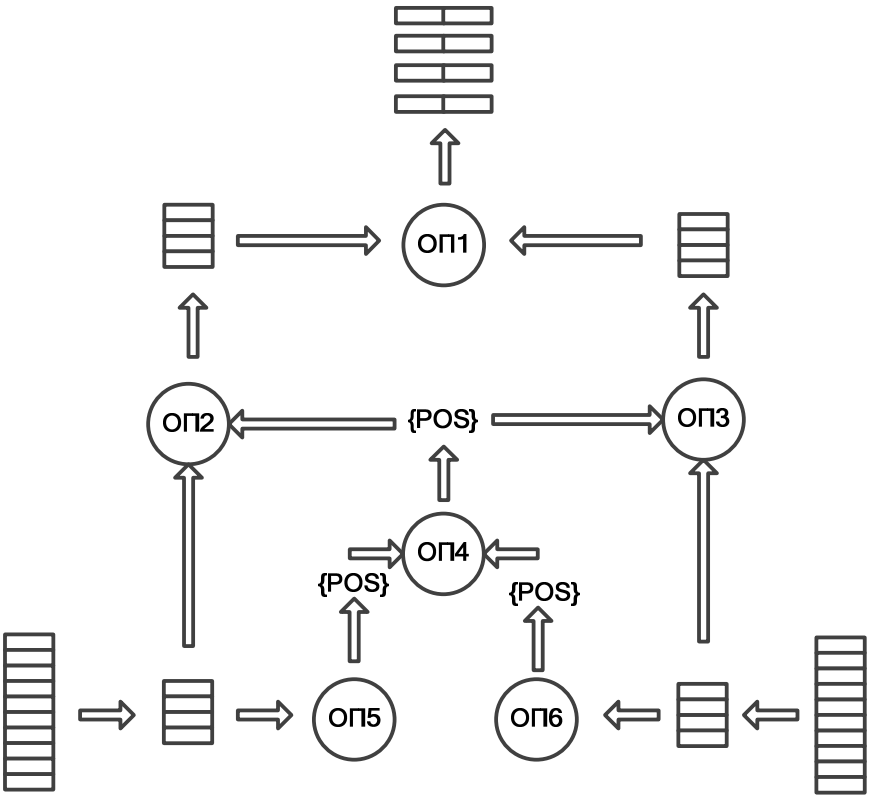

Рис. 2. Пример синхронного конвейера колоночного хранилища

Итераторная модель. Для колоночных хранилищ характерны следующие изменения:

— возможно наличие нескольких родительских узлов, т. е. результаты операции передаются не единственному следующему оператору;

— используются как итераторы по кортежам, так и итераторы по блокам;

— вводится операция материализации кортежа: получение исходного или необходимого на данном этапе кортежа на основе передаваемых блоков значений атрибута.

Наличие нескольких родительских узлов ставит следующую проблему: если один родительский элемент обрабатывает входные данные значительно быстрее, чем другой, данные могут быть потеряны и не переданы более медленному родительскому узлу. В работе [14] данную проблему решают следующим образом: вершиной у графа всегда является единственный элемент, который гарантирует одинаковую частоту запросов к нижестоящим узлам для всего графа.

Скобочный шаблон. Для колоночных систем в скобочный шаблон не вносится значительных изменений. Меняется формат выходных данных: это могут быть как кортежи, так и позиции элементов и указатели на блоки данных.

Основной формой параллельной обработки запросов в строчных и колоночных СУБД является фрагментный параллелизм. Подробно данный процесс рассмотрен в работах [7—10, 15]. В соответствие с этой схемой запрос на языке SQL преобразуется в некоторый последовательный план, который преобразуется в параллельный, представляющий собой совокупность *n* идентичных параллельных агентов, которые реализуют те же операции, что и последовательный план. Здесь *n* обозначает число процессорных узлов. Это достигается путем вставки оператора обмена exchange в соответствующие места дерева плана запроса. На завершающем этапе агенты рассылаются на соответствующие процессорные узлы, где интерпретируются исполнителем запросов. Результаты выполнения агентов объединяются корневым оператором exchange на нулевом процессорном модуле.

Наиболее распространенной системой классификации параллельных систем баз данных является система, предложенная Майклом Стоунбрейкером [15]:

1. SE (Shared-Everything) — архитектура с разделяемыми памятью и дисками.
2. SD (Shared-Disks) — архитектура с разделяемыми дисками.
3. SN (Shared-Nothing) — архитектура без совместного использования ресурсов.

Особенности обработки запросов в колоночных СУБД. Одним из процессов при формировании ответа на запрос в колоночных базах данных является материализация кортежей — процесс воссоздания кортежа на основе столбцов-атрибутов. В зависимости от момента применения данной операции в плане запроса в работе [16] предлагаются следующие варианты материализации.

Ранняя материализация. Данный вариант аналогичен «естественной» материализации, применяемой в строчных СУБД: каждый раз, когда осуществляется доступ к новому атрибуту, он добавляется к кортежу.

Поздняя материализация. Специфика колоночных СУБД позволяет отложить процесс материализации до определенного момента, используя в процессе выполнения запроса позиции значений в колонках вместо самих значений атрибутов. К преимуществам данного метода можно отнести более высокую скорость работы с позициями значений по сравнению со всем кортежем. Слабым местом такого подхода является необходимость двойного чтения данных из столбца — в первом случае для получения позиций, во втором, уже после анализа и преобразования номеров, для получения значений.

Следовательно, операцию материализации можно рассматривать в качестве момента, после которого исполнитель запросов начинает применять классические покортежные операции. В работе [16] предлагаются два варианта обработки столбцов — параллельный и последовательный. В первом случае все колонки считываются независимо друг от друга. При поздней материализации после считывания полученные битовые маски пересекаются, и на основе результирующей

битовой маски происходит чтение остальных атрибутов, участвующих в операции проекции. При последовательной обработке, в отличие от параллельной, атрибуты, участвующие в запросе, упорядочиваются по убыванию селективности и считываются по очереди. В каждую следующую операцию чтения атрибута передается битовая маска предыдущего чтения (рис. 3).

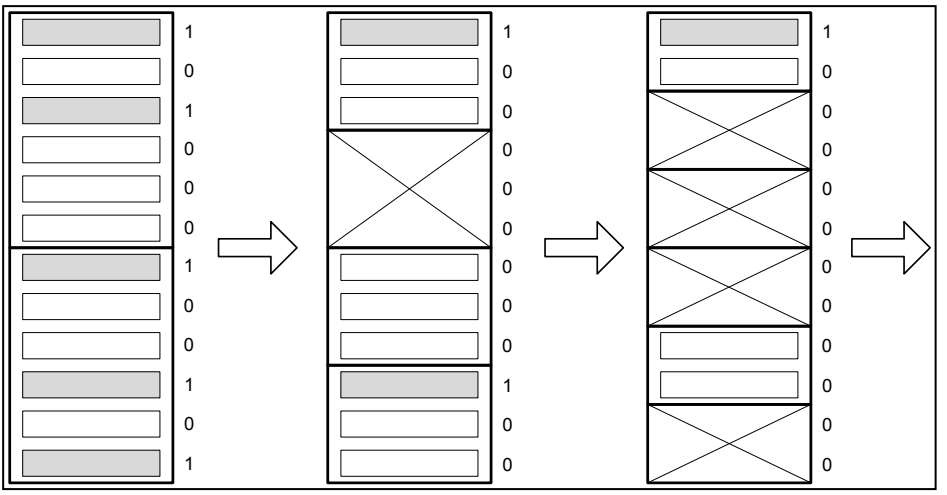

Рис. 3. Иллюстрация особенности последовательной обработки атрибутов

Подобная организация процесса чтения атрибутов позволяет уменьшить число операций чтения блоков данных с диска. На рис. 3 показаны колонки атрибутов. Перечеркнуты те блоки, которые нет необходимости считывать, так как соответствующие записи не удовлетворяют условию поиска по предыдущим атрибутам.

В современных СУБД широко используется ***сжатие данных***. Это позволяет повысить производительность за счет уменьшения числа дисковых операций ввода—вывода и объема передаваемых по сети данных. Колоночное хранение отношений позволяет улучшить этот показатель по сравнению со строчными СУБД. Это достигается за счет использования коэффициентов повторяемости значений атрибутов и возможности оперировать сжатыми данными (т. е. отсутствия затрат на декомпрессию). Ниже приведены описания некоторых алгоритмов сжатия, применяемых в колоночных базах данных [17]:

1. *RLE (кодирование длин серий)*. Последовательная серия одинаковых элементов данных заменяется на два символа: элемент и число его повторений.

2. *Словарный метод.* Используется словарь, состоящий из последовательностей данных или слов. При сжатии эти слова заменяются на их коды из словаря. В наиболее распространенном варианте реализации в качестве словаря выступает сам исходный блок данных.

3. *Векторное кодирование.* С каждым значением сопоставляется битовая строка, где значение 1 обозначает позицию с данной величиной.

4. *Алгоритм Лемпеля — Зива — Велча.* Данный алгоритм при сжатии (кодировании) динамически создает таблицу преобразования строк: определенным последовательностям символов (словам) ставятся в соответствие группы бит фиксированной длины (обычно 12-битные).

В работе [17] предлагается полученный эмпирическим путем алгоритм выбора типа компрессии данных в столбцах.

Преобразования Лапласа — Стилтьеса (ПЛС) времени выполнения запроса к одной таблице в строчной и колоночной системе баз данных. В работе [8] приведено ПЛС времени выполнения запроса к строчной базе данных с планом $\pi_{\text{A}}(\sigma_{\text{F}}(R))$, π — операция проекции, σ — операция селекции:

$$\varphi(s) = G(\varphi_D^{1/L}(s)\,\varphi_M^2(s)(1 - P_F(1 - \varphi_N(s)))\,\varphi_P(s)), \qquad (1)$$

где $G = z^{V/n}$ — производящая функция числа записей фрагментной таблицы R, обрабатываемых на одном процессоре; V — общее число записей в таблице R; n — число процессоров (или персональных компьютеров в кластере); $\varphi_D(s)$ — ПЛС времени чтения блока БД фрагментированной таблицы с диска (с учетом очереди к дисковому массиву); L — число записей таблицы в блоке БД; $\varphi_M(s)$ — ПЛС времени чтения/сохранения записи фрагментированной таблицы в оперативной памяти (с учетом очереди к шине памяти); $\varphi_N(s)$ – ПЛС времени межпроцессорного обмена при передаче результирующей записи по сети N; $\varphi_P(s)$ — ПЛС времени обработки записи в процессоре, который является неразделяемым ресурсом; P_F — вероятность, что запись удовлетворяет условию поиска F (эту вероятность рассчитывают по формулам [6]).

В работе [8] приведены формулы для $\varphi_D(s)$, $\varphi_M(s)$, $\varphi_N(s)$ (т. е. для ПЛС времени обработки кортежей в ресурсах) для различных режимов функционирования системы баз данных и различных архитектурных решений.

Используя подход, предложенный в работе [4], в работе [12] авторами получено ПЛС времени выполнения запроса к колоночной базе данных с планом $\pi_{\text{A}}(\sigma_F(\text{R}))$ и условием

$$F = f_0 \cap \bigcap_{i=1}^{|K_F|} f_i,$$

где f_i — элементарное условие поиска по i-му атрибуту таблицы R, например $a_i > 5$; K_F — множество таких атрибутов таблицы R, на которые накладываются элементарные условия поиска; f_0 — условие поиска, которое включает сравнение разных атрибутов таблицы R, например $a_i > a_j$.

Для ПЛС времени выполнения запроса на одном процессоре кластера (или машины) имеем:

$$\varphi(s) = G(\chi_1(s, r_1, m_1)\Psi_1(s, z)),$$

$$z = \Omega(P_T, \Psi_\pi(s) \cdot \varphi_M^{w \cdot v}(s)\varphi_N^{w \cdot v}(s), \qquad (2)$$

$$\Omega(P, z) = 1 - P(1 - z),$$

где $G = z^{V/n}$ — производящая функция (ПФ) числа позиций (записей) таблицы (отношения) R, обрабатываемых на одном процессоре; V — общее число записей в таблице R; n — число процессоров в кластере (или в машине).

При хорошей хеш-функции таблица равномерно фрагментируется по ключевому (уникальному) атрибуту с точностью до 0,5 %.

Функция $\psi_1(s, z)$ учитывает, что читаются кортежи колонок по позициям, которые удовлетворяют условиям поиска по предыдущим атрибутам. Эта функция рекуррентно определяется следующим образом:

$$\Psi_1(s, z) = \Omega\left(P_{f1}, \chi_2(s, r_2, m_2)\Psi_2(s, z)\right), \qquad (3)$$

...

$$\Psi_i(s, z) = \Omega\left(P_{fi}, \chi_{i+1}(s, r_{i+1}, m_{i+1})\Psi_{(i+1)}(s, z)\right),$$

...

$$\Psi_b(s, z) = \Omega\left(P_{fb},\ \varphi_P^u(s)z)\right),$$

$b = |K_F|$ — мощность подмножества атрибутов отношения, по которым происходит фильтрация кортежей по условию $\bigcap_{i=1}^{|K_F|} f_i$;

$\Psi_\pi(s) = \prod_{j=|K_F|+1}^{|K_F|+|K_\pi|} \chi_j(s, 1, m)$; $\chi_i(s, r, m) = \varphi_{Di}(s)\varphi_M^{m\nu_i}(s)\varphi_P^r(s)$; $\varphi_{Di}(s)$ — ПЛС времени чтения кортежа i-го столбца с диска; $\varphi_M^{m\nu_i}(s)$ — ПЛС времени сохранения атрибута в ОП и его чтения в кэш процессора; ν_i — размер атрибута; $m\nu_i$ — число операций чтения/записи в оперативную память, необходимых для проверки условия по соответствующему атрибуту (для i-го столбца вводится аргумент m_i — см. (2) и (3)).

Считается, что кэш процессора — это «черная дыра», в которой сохраняются данные процессора, необходимые для вычислений. Из кэша в ОП перемещаются только результирующие материализованные записи, передаваемые по шине (см. далее $\varphi_N(s)$) процессору, где выполняется сборка.

$\varphi_P^r(s)$ — ПЛС времени обработки кортежа столбца в процессоре; r — число логических операций, необходимых для проверки условия по соответствующему атрибуту (для i-го столбца вводится аргумент r_i — см. (2) и (3));

$|K_\pi|$ — мощность подмножества атрибутов отношения, которые участвуют в операции проекции и не присутствуют в множестве K_F;

$\chi_j(s, 1, m_j)$ — ПЛС времени чтения кортежа j-го столбца проекции с диска в кэш процессора и обработки в нем; 1 означает, что в процессоре проверяется только значение битовой маски в позиции, указанной в кортеже;

$\varphi_P^u(s)$ — учитывает, что для проверки условия f_0 для материализованных записей потребуется «u» логических операций процессора;

P_{fi} — вероятность, что кортеж колонки i-го атрибута удовлетворяет условию f_i;

P_T — вероятность, что сформированный кортеж удовлетворяет условию f_0;

$\varphi_M^{w\nu}(s)$ — учитывает перемещение записей таблицы R, удовлетворяющих условию поиска F, из кэша процессора в ОП, а затем из ОП в буфер межпроцессорной шины (см. (2)); ν — размер сформированного кортежа $\left(\sum_{i=1}^{|K_\pi|} \nu_i\right)$, $w\nu$ — количество операций чтения/записи, которое необходимо для перемещения сформированных записей;

$\varphi_N^{w\nu}(s)$ – учитывает передачу записей таблицы R, удовлетворяющих условию поиска F, по шине процессору, выполняющему сборку.

В работе [12] приведены формулы для $\varphi_D(s)$, $\varphi_M(s)$, $\varphi_N(s)$ (т. е. для ПЛС времени обработки кортежей в ресурсах) для различных режимов функционирования системы баз данных и различных архитектурных решений.

При выводе (2) учитывались следующие особенности выполнения запроса в колоночной СУБД [11]:

— каждая колонка хранится на диске в своих блоках, где отдельная колонка представляет собой таблицу с кортежем (значение атрибута, позиция);

— последовательная и параллельная обработка запросов с поздней материализацией кортежей;

— наличие компрессии данных (метод RLE);

— получение времени работы обслуживающих устройств на основе измеряемых с помощью синтетических тестов показателей.

При этом рассматривались два режима работы [12]:

1. *Пакетный режим* (offline, система рассматривается как замкнутая). При данном режиме работы в колоночной системе баз данных обрабатываются пакеты запросов. В каждом пакете SQL-запросы

выполняются последовательно (предполагается, что они связаны по данным: выходные данные одного запроса являются входными данными другого). Но запросы разных пакетов (по одному из каждого пакета) могут обрабатываться параллельно. Предполагается, что «узкое место» в данном режиме — дисковая подсистема.

2. *Режим запрос—ответ* (online, система рассматривается как разомкнутая). При таком режиме работы предполагается, что *i*-я рабочая станция обращается к *j*-му запросу с некоторой интенсивностью. При условии, что эти входные потоки заявок являются пуассоновскими, время обслуживания в ресурсах распределено по экспоненциальному закону, а переход от ресурса к ресурсу выполняется по вероятности, модель обработки запросов можно представить в виде сети массового обслуживания. В этой сети обработку в узлах ресурсов можно представить в виде совокупности независимых СМО M/M/1 (это доказывается в теории массового обслуживания в виде теоремы разложения Джексона).

Сравнение среднего времени обработки запроса к одной таблице в строчной и колоночной базах данных. Оценку среднего времени выполнения запроса можно получить, дифференцируя выражения (1) и (2) в нуле (в работе [12] были использованы численные методы). Ниже приведены результаты расчета отношения среднего времени обработки простого запроса $\pi_A(\sigma_F(R))$ в строчной СУБД к среднему времени выполнения этого запроса в колоночной СУБД в зависимости от отношения количества атрибутов, участвующих в запросе, к общему количеству атрибутов в таблице.

Для упрощения расчетов будем считать, что $K_A = K_F = K$, т. е. количество атрибутов, участвующих в операции фильтрации, равно количеству атрибутов, использующихся в операции проекции. Также примем, что таблица состоит из $H = 100$ одинаковых по размеру и типу атрибутов. Расчеты были выполнены при следующих значениях характеристик ресурсов.

1. Процессор — Intel Xeon 5160. Для выбранного процессора измеренное значение числа процессорных циклов, выполняемых в секунду — $\mu_P = 1{,}5 \cdot 10^9$ 1/с.

2. Внешняя память — $N = 50$ дисков 3.5" Seagate Cheetah 15K.6 ST3146356FC; размер блока чередования (stripe size) — $Q_{\text{б.ч}} = 64$ кбайт; среднее время поиска и чтения блока чередования с диска — $t_{\text{б.ч}} = t_{\text{п}} + t_{\text{в}}/2 + Q_{\text{б.ч}}/v_{\text{ч}} = 4 + 4/2 + 64/200 = 6{,}3$ мс, где $t_{\text{п}}$, $t_{\text{в}}$ — время подвода и вращения диска соответственно; $v_{\text{ч}}$ — скорость чтения данных с диска. Поэтому интенсивность чтения блоков с диска равна $\mu_D = 1000/6{,}3 = 160$ 1/с.

3. Оперативная память — DDR3-1600 PC3-12800. Расчеты показывают, что интенсивность чтения записей базы данных из ОП равна $\mu_M = 10{,}4 \cdot 10^6$ 1/с.

4. Остальные параметры для расчетов следующие: $V = 10^6$; $P_1 = 0{,}01$; $P_2 = 0{,}007$; $r_i = 20$; $u = 50$; $p_D = 0{,}9$; $L = 100$; $H = 100$;

$L_i = LHk_c$, где L — среднее число записей таблицы R в блоке чередования для строчной СУБД; k_c — среднее число позиций, покрываемых одним кортежем столбца колоночной базы данных, если нет сжатия $k_c = 1$.

Зависимости отношения времени выполнения запроса в строчной СУБД к времени выполнения запроса в колоночной СУБД (Y) от отношения числа атрибутов, участвующих в запросе, к общему числу атрибутов в таблице ($X = 200\ K/H$) для различного среднего числа позиций, покрываемых одним кортежем (коэффициент k_c учитывает сжатие), представлены на рис. 4. Зависимости построены для числа процессоров $n = 2$. В таблице приведены значения Y для некоторых значений X при различных значениях k_c.

Значения Y для некоторых значений X при различных значениях k_c

X, %	Значение Y при k_c		
	1	5	10
2 ($K_A = K_F = 1$)	31	70	99
65	1,0	2,5	3,8
100	0,66	1,65	2,5

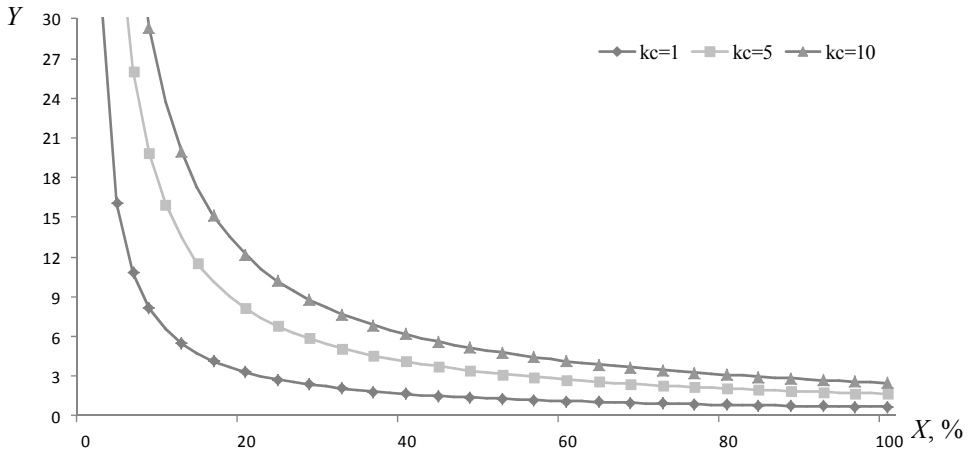

Рис. 4. Зависимости отношения времени выполнения запроса в строчной СУБД к времени выполнения запроса в колоночной СУБД (Y) от отношения числа атрибутов, участвующих в запросе, к общему числу атрибутов (X, %) при различных значениях k_c

Из зависимостей следует, что при использовании менее 20 % атрибутов время выполнения запроса в колоночной СУБД в разы меньше по сравнению с временем выполнения в строчной СУБД, при большем числе атрибутов время выполнения запроса растет практи-

чески пропорционально числу используемых в запросе атрибутов. Для k_c =1 (нет сжатия данных) среднее время выполнения запроса в строчной и колоночной СУБД становится равным (Y = 1) при использовании 65 % атрибутов. Увеличение времени выполнения запроса в колоночной СУБД при большем числе используемых атрибутов можно объяснить ростом числа читаемых с диска столбцов таблицы (для строчных СУБД время не изменяется, так как с диска записи читаются целиком). При X = 100 % и k_c = 1 среднее время выполнения запроса в строчной СУБД в 1,5 (1/0,66) раза меньше, чем в колоночной СУБД. При достаточно хорошем сжатии столбцов таблицы картина меняется: колоночная СУБД лучше строчной даже при использовании в запросе 100 % атрибутов.

На рис. 5 представлены зависимости среднего времени выполнения запроса в колоночной СУБД от числа процессоров для различного соотношения используемых в запросе атрибутов (10, 50 и 100 %), а также время выполнения запроса в строчной СУБД. Видно, что для строчной СУБД пятнадцатисекундная отметка среднего времени обработки запроса достигается при числе процессоров n = 10. Для колоночной СУБД эта отметка достигается при соотношении используемых в запросе атрибутов 10 % (10 атрибутов из 100) уже при n = 2 (и это при отсутствии сжатия столбцов, k_c =1). Экономия вычислительных ресурсов налицо.

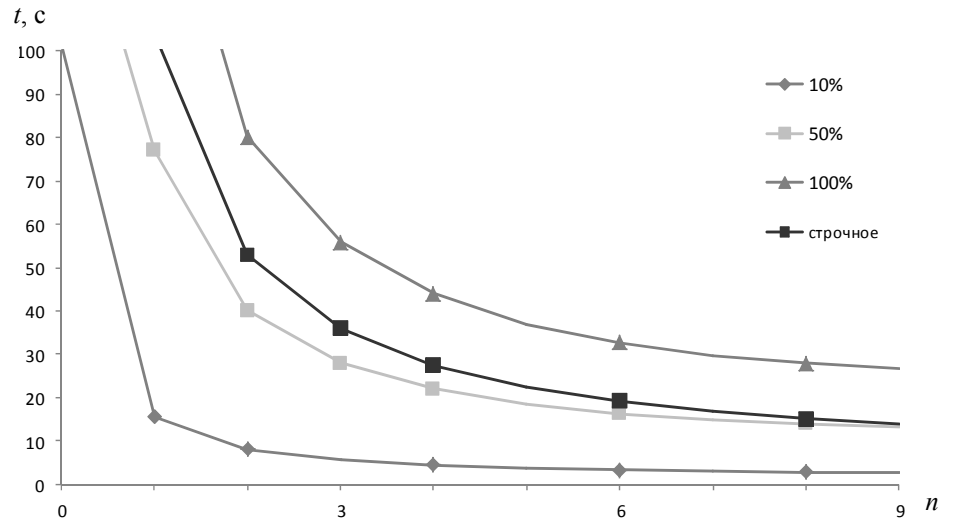

Рис. 5. Зависимости среднего времени выполнения запроса в колоночной СУБД от числа процессоров для различного отношения используемых в запросе атрибутов и времени выполнения запроса в строчной СУБД (k_c =1)

Заключение. Проанализированы процессы выполнения запросов в строчной и колоночной системе баз данных. Рассмотрены измене-

ния, вносимые в синхронный конвейер, итераторную модель и скобочный шаблон, а также операции материализации и компрессии данных для колоночных СУБД.

Приведено преобразование Лапласа — Стилтьеса времени выполнения запроса, имеющего план $\pi_A(\sigma_F(R))$, в параллельной строчной и колоночной СУБД.

Представлены результаты сравнения среднего времени выполнения запроса с планом $\pi_A(\sigma_F(R))$ в строчной и колоночной СУБД. Приведен пример расчета отношения среднего времени выполнения запроса в строчной СУБД к среднему времени выполнения запроса в колоночной СУБД в зависимости от отношения количества атрибутов, участвующих в запросе, к общему количеству атрибутов в таблице. На его основании можно сделать вывод о том, что при хорошем сжатии столбцов (k_c) время выполнения запроса в колоночной СУБД меньше, чем в строчной СУБД даже при использовании в запросе 100 % атрибутов (см. таблицу).

Для колоночной СУБД десятисекундная отметка среднего времени выполнения запроса при отношении используемых в запросе атрибутов 10 % достигается при меньшем числе процессоров (n = 2), чем для строчных СУБД (n = 15). Это свидетельствует об экономии вычислительных ресурсов при использовании колоночных СУБД.

Предполагается продолжить исследования и получить оценки времени выполнения запросов с более сложными планами реализации (например, для плана выполнения запроса к хранилищу данных типа «звезда»).

СПИСОК ЛИТЕРАТУРЫ

1. Арсентьев А. Хранилища данных становятся инфраструктурным компонентом №1. CNews аналитика. 2010. [Электронный ресурс]. [http://retail.cnews.ru/reviews/free/BI2010/articles/articles6.shtml]. Проверено 27.06. 2011.
2. Michael Stonebraker Biography, 2008. [Электронный ресурс] [http://www.csail.mit.edu/user/1547]. Проверено 28.06.2012.
3. Stonebraker M., Çetintemel U. One Size Fits All: An Idea Whose Time Has Come and Gone / Перевод С. Кузнецова, 2007. [Электронный ресурс]. [http://citforum.ru/database/articles/one_size_fits_all/]. Проверено 27.06.2011.
4. One Size Fits All / Stonebraker M., Bear C., Çetintemel U. et al. P. 2: Benchmarking Results // 3rd Biennial Conf. on Innovative Data Systems Research (CIDR), January 7–10, 2007. Asilomar, California, USA / Перевод Сергея Кузнецова, 2007. [Электронный ресурс]. [http://citforum.ru/database/articles/one_size_fits_all_2/]. Проверено 27.06.2011.
5. Stonebraker M. My Top 10 Assertions About Data Warehouses / Перевод Сергея Кузнецова, 2010. [Электронный ресурс]. [http://citforum.ru/gazeta/166/]. Проверено 27.06.2011.
6. Григорьев Ю.А., Плутенко А.Д. Теоретические основы анализа процессов доступа к распределенным базам данных. — Новосибирск: Наука, 2002. — 222 с.

7. Григорьев Ю.А., Плужников В.Л. Оценка времени соединения таблиц в параллельной системе баз данных // Информатика и системы управления. 2011. № 1. — С. 3—16.
8. Григорьев Ю.А., Плужников В.Л. Модель обработки запросов в параллельной системе баз данных // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2010. № 4. — С. 78–90.
9. Григорьев Ю.А., Плужников В.Л. Оценка времени соединения таблиц в параллельной системе баз данных // Информатика и системы управления. — 2011. № 1. — С. 3–16.
10. Григорьев Ю.А., Плужников В.Л. Анализ времени обработки запросов к хранилищу данных в параллельной системе баз данных // Информатика и системы управления. 2011. № 2. — С. 94–106.
11. Григорьев Ю.А., Ермаков Е.Ю. Модель обработки запросов в параллельной колоночной системе баз данных // Информатика и системы управления. 2012. № 1. — С. 3–15.
12. Григорьев Ю.А., Ермаков Е.Ю. Модель обработки запроса к одной таблице в параллельной колоночной системе баз данных и анализ ее адеватности // Информатика и системы управления. 2012. № 2. — С. 170—179.
13. C-Store: A Column-Oriented DBMS / M. Stonebraker, D. J. Abadi, A. Batkin et al. [Электронный ресурс]. [http://www.cs.yale.edu/homes/dna/pubs/displaypubs.cgi/]. Проверено22.10.2011.
14. Abadi D.J. Query Execution in Column-Oriented Database Systems. [Электронный ресурс]. [http://www.cs.yale.edu/homes/dna/papers/abadiphd.pdf]. Проверено 25.12.2011.
15. Соколинский Л. Б., Цымблер М. Л. Лекции по курсу «Параллельные системы баз данных»: [Электронный ресурс]. [http://pdbs.susu.ru/CourseManual.html]. Проверено 22.10.2011.
16. Materialization Strategies in a Column-Oriented DBMS / D. J. Abadi, D.S. Myers, D. J. DeWitt et al. // In Proceedings of ICDE, 2007. [Электронный ресурс]. [http://db.lcs.mit.edu/projects/cstore/abadiicde2007.pdf]. Проверено 25.12.2011.
17. Abadi D. J., Madden S. R., Ferreira M.C. Integrating Compression and Execution in Column-Oriented Database Systems // In Proceedings of ICDE, 2006. [Электронный ресурс]. [http://db.lcs.mit.edu/projects/cstore/abadisigmod06.pdf]. Проверено 25.12.2011.

Статья поступила в редакцию 4.07.2012