УДК 004.65

Г.А. Чернышев

ОРГАНИЗАЦИЯ ФИЗИЧЕСКОГО УРОВНЯ КОЛОНОЧНЫХ СУБД

Чернышев Г.А. **Организация физического уровня колоночных СУБД**
Аннотация. В данной работе мы рассмотрели колоночные СУБД и их устройство. Было дано определение колоночной СУБД, представлены отличия от классических СУБД с построчным хранением. Были представлены сильные и слабые стороны колоночного подхода, причины его появления и успеха. Кроме того, в работе описана ниша, занимаемая современными коммерческими колоночными СУБД. Затем нами был рассмотрен набор технологических решений, используемых в данных СУБД. Наконец, в данной работе были рассмотрены вопросы выбора структур физического уровня для колоночных СУБД. В начале представлено краткое введение в проблему выбора данных структур. Затем, на основе произведенного обзора анализируется применимость подходов к устройству физического уровня классических СУБД (фрагментирование, размещение и пр.) к колоночным СУБД. Кроме того, рассматриваются способы, характерные именно для колоночных систем (различные порядки колонок, индексы соединения).
Ключевые слова: базы данных, колоночные СУБД, физический уровень СУБД, настройка СУБД, вертикальное фрагментирование, горизонтальное фрагментирование, размещение данных.

Chernishev G.A. **Physical design approaches for column-stores.**
Abstract. In this paper we survey column-store database technology. We describe column-store database management system and point out the distinctions from the classical row-store DBMS. We present strong and weak points of the column-stores, the reasons of their appearance and success. We survey a variety of technical solutions, which provide these systems with the advantage. We also describe workloads which benefit this approach the most and thus constitute the area of application of commercial column-store DBMS systems. Then, we present a brief introduction into the problem of DBMS physical design. We analyze the perspectives of various classical approaches (allocation, partitioning etc), as well as column-store specific ones (column sort orders and join index application).
Keywords: databases, physical database design, database tuning, column-stores, vertical partitioning, horizontal partitioning, allocation.

1. Введение. Данная работа является продолжением работы [3], в которой мы рассматривали вопросы выбора структур физического уровня для классических систем с построчным хранением (row-store DBMS). В настоящей работе мы рассмотрим колоночные СУБД (column-store DBMS), разберем их устройство и рассмотрим вопрос выбора структур физического уровня для систем данного типа.

Приведем вкратце основные тезисы [3]:

1. Современные СУБД предоставляют богатый выбор вариантов организации обрабатываемых данных [7, 17, 18, 20, 38, 41, 45]. Оптимальность выбора структур физического уровня играет немаловажную роль в обеспечении высокой производительности СУБД.

2. Эффективная организация физического уровня СУБД может позволить:

   (a) Увеличить допустимый объем загружаемых и обрабатываемых данных за счет внесения в систему новых узлов и устройств хранения;

   (b) Уменьшить время выполнения отдельного запроса посредством использования специальных структур данных и методов их обработки, а также с помощью использования внутризапросного параллелизма;

   (c) Уменьшить время выполнения набора запросов с помощью использования межзапросного параллелизма (повышение степени параллелизма системы);

   (d) Уменьшить затраты на обслуживание системы и тем самым понизить совокупную стоимость владения (материальные расходы).

3. Возможны следующие варианты устройства физического уровня:

   (a) Фрагментирование данных (partitioning, fragmentation). Подразделяется на: горизонтальное (horizontal partitioning), вертикальное фрагментирование (vertical partitioning), и их комбинации (гибридное или смешанное фрагментирование). В свою очередь, горизонтальное фрагментирование подразделяется на основное (primary fragmentation) и производное (derived fragmentation);

   (b) Размещение данных по узлам в случае распределенной СУБД (data allocation problem);

   (c) Использование дополнительных структур — индексов и материализованные представлений (index selection problem, materialized view selection problem);

(d) Репликация данных (replication).

4. Эти варианты устройства могут использоваться не только по отдельности, но и совместно, в различных комбинациях. Кроме того, методы выбора структур физического уровня могут быть статическими (static) или динамическими (dynamic). В последнем случае в процессе работы учитывается изменение параметров нагрузки.

5. Задача оптимального выбора структур физического уровня NP-трудна. Поэтому на практике применяются алгоритмы, отыскивающие приближенное решение.

6. В настоящее время курс как индустрии баз данных, так и исследовательского сообщества, нацелен на полное исключение вмешательства человека в процесс выбора структур физического уровня, то есть на полную автоматизацию процесса.

Задачей выбора структур физического уровня в контексте классических СУБД исследователи занимаются чрезвычайно долгое время. Накоплен огромный опыт, который подробно анализируется в нашей предыдущей работе, рассматриваются более 150 научных работ [3].

Около десяти лет назад приобрел популярность новый тип СУБД - колоночная СУБД (column-store DBMS) [9, 10]. В современных работах он противопоставляется [6] классическим СУБД с хранением по строкам (row-store DBMS).

Несмотря на кажущуюся простоту колоночного подхода, он требует новых методов при выборе структур физического уровня. Это происходит вследствие иного процесса оптимизации запроса и возможности применения иных структур физического уровня.

Цель настоящей работы — на основе анализа существующих подходов к построению колоночных СУБД выработать указания для выбора структур физического уровня. Данные указания позволят создать автоматизированную систему выбора таких структур, наподобие существующих в настоящее время для коммерческих систем с построчным хранением.

Основная часть настоящей работы состоит из трех разделов. В следующем разделе мы ознакомимся с существующими подходами к построению колоночных СУБД, рассмотрим отличия от класси-

ческих. В третьем разделе будет приведено обоснование преимущества колоночных СУБД. Мы опишем основные классы технических решений применяемые в СУБД данного типа. Кроме того, в этом разделе будут описаны предпосылки для появления и успеха этих систем. Также будут описаны два класса нагрузок СУБД и описана ниша применения современных коммерческих колоночных систем. Наконец, в четвертом разделе, будут приведены соображения касательно выбора структур физического уровня в колоночной системе. Будут рассмотрены возможности переиспользования структур физического уровня, существующие в системах с построчным хранением. Кроме того, будут рассмотрены и структуры, характерные только для колоночных систем.

2. Обзор технологий колоночных СУБД. Колоночной СУБД называется специализированная система, ориентированная на эффективную работу с отдельными атрибутами. Термин «колоночная СУБД» приобрел широкую популярность в конце 90х годов. Однако нельзя сказать, что это была полностью новая идея. Еще в 80х годах было целое направление, занимавшееся СУБД, нацеленными на обработку статистической информации [42]. Некоторые из систем того времени были построены на похожих принципах [14, 33].

В классических системах с построчным хранением используется так называемая слотированная страница (slotted page) [47]. Она представляет собой область памяти, заполненную записями последовательно с начала страницы. Указатели на сами записи (смещения от начала страницы) записываются в обратном порядке, начиная с конца страницы. И самое важное — каждая запись хранит все атрибуты таблицы. Кроме этого, страница хранит системную информацию, специфичную для каждой СУБД. Описанный подход предоставлял возможность хранения на страницах записей переменной длины и обеспечивал достаточно хорошие характеристики чтения и записи данных. Эта модель хранения является классической и применяется не один десяток лет. Согласно [47], как минимум до начала нулевых эта модель повсеместно использовалась в коммерческих системах.

Основную идею колоночной системы можно описать как «хранить каждый атрибут отдельно от других». Это решение позволит отойти от модели слотированной страницы и хранить данные просто в виде последовательности значений атрибута. Кроме того,

отпадет необходимость хранить смещения.

Перечислим основные отличия от классических систем с хранением по строкам.

1. Присутствует необходимость проводить «сборку» записи из набора колонок, в литературе называемая материализацией. Это весьма дорогостоящий процесс, который при неправильном проведении может свести на нет все плюсы колоночного подхода [34]. Различают две стратегии материализации, каждая из которых обладает своими достоинствами и недостатками [34]. Материализацию как процесс сборки записи из набора колонок не следует путать ни с материализацией в смысле накопления полного результата промежуточной операции в итераторной модели [22], ни с материализацией в смысле представлений записанных на диск (materialized view).

2. Подход к представлению данных на диске требует продуманности и аккуратности реализации. В работе [6] приводится факт, что при проведении экспериментов колоночное представление занимало практически в три раза больше дискового пространства. Это было вызвано тем, что колоночная СУБД дополнительно хранила идентификаторы каждой записи. Для решения был использован целый набор методов сжатия данных. Альтернативой могут выступать реализации, не хранящие в явном виде данные идентификаторы [9]. Кроме того, при повышении доли вычитываемых колонок становится выгоднее использовать классический подход [39].

3. Индексы в классическом смысле слабо применимы или не применимы вовсе (детали зависят от выбранных методов реализации СУБД). Вместо них был предложен целый ряд методов индексирования данных для колоночных систем [23,30, 31] (адаптивное индексирование, adaptive indexing).

4. План зачастую не является деревом, необходимо каким-то образом решать проблему повторного считывания отношения с диска [34].

Самыми известными исследовательскими СУБД являются MonetDB [9, 30] и C-Store [10]. А с середины нулевых стали появляться коммерческие реализации [2].

3. Преимущества колоночных СУБД и причины успеха. Обоснование эффективности колоночного подхода было сделано в работе [6]. Эта работа дает общую картину производительности колоночной системы в сравнении с неназванной коммерческой СУБД с хранением по строкам. Работа целиком и полностью посвящена экспериментам с колоночной СУБД C-Store, которые проводятся на эталонном наборе Star Schema Benchmark [37].

Этот эталонный набор построен на основе более известного набора TPC-H [46] и описывает схему хранилища данных магазина запчастей. Он моделирует популярную схему «звезда». Используя данный тестовый набор, авторы проводили сравнение прототипа колоночной системы C-Store и коммерческой системы с хранением по строкам. Интересно, что они выполнили не просто сравнение, а попытались проэмулировать колоночную систему с помощью ряда методик. В итоге, эксперименты показали, что превосходство колоночной системы над классической составляет от 4 до 10 раз в зависимости от запроса (в среднем по всем запросам в 5 раз). Имитируемая колоночная система все равно проигрывает C-Store, в среднем в 2.5 раза. Если же сымитировать внутри колоночной системы классический подход, то он покажет худшие результаты, чем рассматриваемая коммерческая система.

Перечислим теперь те сильные стороны колоночных систем, которые позволяют добиться высокой производительности.

1. Эффективность работы с диском [39] — нет необходимости считывать атрибуты, не участвующие в исполнении запроса. При обработке больших объемов данных зачастую именно считывание с диска становилось «узким» местом, ограничивавшим производительность всей системы в целом. Кроме того, такое представление дает ряд дополнительных возможностей, аналогичных предоставляемыми индексами [19].

2. Возможность эффективного использования различных методик сжатия [4, 14, 26]. Исследователи уже применяли сжатие в СУБД, однако нельзя сказать, что были достигнуты большие успехи. В колоночных системах ситуация гораздо лучше по следующей причине — данные лучше поддаются сжатию вследствие гомогенности. Таким образом, удалось повысить коэффициент сжатия до десяти [5, 40] (для сравнения классические системы имеют коэффициент 2-3 [5]).

3. Возможность оперирования непосредственно над сжатыми данными [4, 6]. Были созданы Volcano-подобные операторы, способные работать с запакованными данными. Система, построенная на данном принципе, производит распаковку только перед передачей результата клиенту. Соответственно, довольно дорогостоящие операции распаковки задействуются только для части данных. Например, если в запросе производится фильтрация по предикату, распакованы будут только данные, которые удовлетворяют ему.

4. Ориентированность на эффективное использование процессора и памяти. Данный аспект был серьезно проработан в контексте СУБД в оперативной памяти (main-memory DBMS). Он включал в себя: минимизацию промахов в кэш-памяти [24,29,47], векторизацию операций (использование SIMD, позволяющее за один такт процессора обработать несколько значений атрибута) [5], операторы реляционной алгебры, оптимизированные для работы с кэш-памятью процессора [9].

5. Специализированные операторы. К ним относятся не только операторы, оптимизированные для работы с кэш-памятью, но и сконструированные для решения специфических задач, например оператор «невидимое соединение» [6] или специализированные операторы из [32].

6. Возможность эффективного совместного использования СУБД и подходов MapReduce [1,2]. Колоночное устройство дает возможность эффективно использовать MapReduce для аналитических задач в распределенной среде.

Теперь рассмотрим предпосылки для популяризации колоночных систем. Их можно описать как «новое аппаратное обеспечение» и «новые задачи». Именно изменения в аппаратном обеспечении позволили получить вышеописанные преимущества [5]. Дело в том, что за прошедшие десятилетия произошли как качественные, например, в архитектуре процессоров, так и количественные изменения. В качестве примера последних можно назвать возросшие объемы оперативной памяти и снижение задержек при доступе к ней. Авторы [5] утверждают, что эти изменения требуют пересмотра устоявшихся практик реализации основных компонентов

СУБД, а колоночный подход является результатом этого пересмотра.

Другой предпосылкой для популяризации колоночного подхода было появление новых задач, решаемых с помощью СУБД. Еще в работах 80х наметилось деление систем по целям на два класса [16]. К первому классу относились системы, в которых лучше исполнялись запросы на изменение данных, ко второму — те, где были эффективны так называемые аналитические запросы (OLAP, также называемые запросами поддержки принятия решений — Decision Support, DSS). Разница [12, 21, 36] между этими двумя типами запросов приведена в таблице.

Разница в обрабатываемых запросах обусловлена разницей в целях этих систем. Цель первой — иметь высокую степень параллелизма, обслуживать большое количество клиентов, работая с наиболее свежими данными. Цель систем второго типа — обеспечить высокую производительность исполнения запросов в условиях пониженных требований на степень параллелизма и свежесть данных.

Системы первого типа называются системами оперативной обработки транзакций (OLTP, OnLine Transaction Processing), а системы второго — системами оперативной аналитической обработки данных (OLAP, OnLine Analytical Processing) [36]. Примером системы первого типа может являться банковская система, управляющая счетами своих клиентов. Примером же системы второго типа является база данных продаж за большой период времени — месяц, год.

Данное деление на классы произошло по причине конфликта целей этих систем: практически невозможно построить СУБД, которая могла бы эффективно выполнять задачи обоих типов. Уже в начале 90х это стало повсеместно принятой точкой зрения, а в научной среде данный вопрос получил название «one size fits all» [21]. Индустрия также пошла по этому пути, появились коммерческие продукты, которые специализировались на аналитических запросах и не работали с обновлениями реального времени.

Увеличение объемов дисковой и оперативной памяти, их удешевление, давшее возможность хранить много данных, положительно сказалось на росте рынка хранилищ данных. В [36] отмечается, что рынок OLAP систем составлял в 2006 году около 6 миллиардов долларов, тогда как в 98 году он составлял всего 2 мил-

Таблица 1. Два типа нагрузок в СУБД

OLAP	OLTP
Запросы заранее неизвестны	Шаблоны запросов известны заранее, например транзакция по оплате услуги или по добавлению пользователя
Запросы могут затрагивать десятки таблиц	Запросы затрагивают малое количество таблиц
Пакетное обновление или полное отсутствие обновления данных	Запросы обновляют данные, при этом делают это на лету
Сложные запросы, требующие оптимизации	Запросы простые — не содержат сложных операций, таких как соединение и агрегирование, а значит и не сильно зависят от оптимизации
Большой результат запроса — тысячи и миллионы записей	Результат запроса мал — десятки или сотни записей
Низкоселективные запросы	Высокоселективные запросы — результатом будет малая часть исходного отношения
Запрос затрагивает отдельные атрибуты из сотен или даже тысяч атрибутов таблицы	Запрос затрагивает все атрибуты таблицы
Малое количество клиентов — десятки или сотни, меньшая важность параллелизма	Запросы поступают параллельно, от большого числа клиентов

лиарда. По информации TWDI (The Data Warehousing Institute) за прошедшее десятилетие на порядок выросли и объемы обрабатываемых данных [8]. Дополнительно можно отметить появление собственно термина OLAP (1993), создание OLAP Council (1995). То есть, несмотря на то, что подобные применения баз данных существовали с начала 80х — подъем начался в 90е, а в настоящее время данная область переживает расцвет.

Колоночные системы заняли нишу OLAP систем, то есть они являются довольно узкоспециализированными. Исполнение OLTP запросов в этих системах не дает такого большого выигрыша в производительности по сравнению с классическими СУБД. Например, если система считывает достаточно большой процент колонок, то выигрыш может быть «съеден» затратами на реконструкцию записи. Отказ практически от любой перечисленной в таблице характеристики может привести к серьезному падению производительности. При этом надо заметить, что попытки построить эффективную колоночную OLTP систему продолжаются [28, 29, 40].

4. Актуальные вопросы выбора физических структур для колоночных СУБД. Разберем теперь непосредственно вопрос о фрагментировании в колоночных системах. Несмотря на существующий интерес к распределенным колоночным СУБД и обилие коммерческих реализаций, вопрос о горизонтальном и вертикальном фрагментировании научным сообществом практически не освещается. При этом, фрагментирование в классических системах, в том числе и для OLAP нагрузок, широко изучалось (мы уже рассматривали этот вопрос в работе [3]).

Например, в работе [35] отмечается, что колоночная система Vertica обладает собственным инструментом автоматизации разработки схемы фрагментирования данных. Этот инструмент называется Vertica Database Designer [44], и он решает задачу выработки подходящей схемы фрагментирования (горизонтального и вертикального), исходя из нагрузки для распределенной колоночной СУБД. Однако вследствие того, что эта система является коммерческой, детали ее устройства не раскрываются [35]. Также известно [29] что и многие классические коммерческие СУБД предложили аналогичные средства для колоночного хранения. Про их устройство также практически ничего не известно.

Однако вертикальное и горизонтальное фрагментирование необходимо для построения высокопроизводительной распределенной

колоночной СУБД. Существующие модели не подходят для этих систем, так как не учитывают специфику их функционирования и используемые структуры физического уровня. Поэтому следует создать модель, учитывающую особенности именно колоночного подхода: отличную от классической природу обработки запросов и открывшиеся возможности хранения данных. Ниже разберем эти особенности более подробно.

1. В настоящее время большой интерес представляют распределенные системы без совместного использования ресурсов [25]. Поэтому для построения распределенной колоночной СУБД потребуется рассматривать и вертикальное, и горизонтальное фрагментирование данных. Также необходимо будет решать вопрос о размещении колонок на узлах. Для максимизации производительности желателен интегрированный подход [43] и учет производного фрагментирования. Миграция данных, скорее всего, не потребуется вследствие OLAP нагрузок. То есть алгоритм может быть статическим. Также модель не потребует учета транзакционной составляющей, зато потребуется глубокая проработка стоимостей операторов;

2. Необходим учет семейств колонок и их взаимодействий. Семейство колонок — совместно хранящийся набор колонок. В распределенной колоночной СУБД Vertica такие семейства присутствуют и называются суперпроекцией (superprojection) [10];

3. Несколько возможных порядков сортировки каждого семейства при физическом расположении данных на диске [10,13]. Например, колонка, состоящая из одного атрибута, может быть кластеризована на диске по значению атрибута, а может и по идентификатору записи. В первом случае будет возможно эффективно производить поиск по предикату, во втором — восстановить запись. Наличие нескольких копий данных, различным образом упорядоченных, положительно влияет на качество планов генерируемых оптимизатором запросов, а значит и на производительность системы в целом. Этот фактор никоим образом не учитывался в предыдущих работах по фрагментированию;

4. Использование индексов соединения (join index) [10]. Данную

структуру можно упрощенно представить как таблицу, построенную для двух семейств. В ней значения первого атрибута (по которому она и кластеризована на диске) содержит номер записи из первого семейства, а значения второго атрибута — номер записи во втором семействе. Она необходима для ускорения работы при восстановлении записи из нескольких семейств, кластеризованных по различным атрибутам. Таким образом, вместе с рассмотрением различных порядков сортировки необходимо учитывать индексы соединения. Этот фактор так же не учитывался раннее;

5. Репликация данных, которая может проистекать как из перекрывающихся семейств колонок, так и репликация, которая может происходить из колонок, хранящихся в отсортированном виде по различным атрибутам, а также из прямого реплицирования данных;

6. Требуется учет специфики операторов, например, учет необходимости материализации результата (материализация как процесс сборки записи из набора колонок). Также, колоночные операторы сильно отличаются от стандартных, и это, безусловно, необходимо учитывать;

7. Интерес представляют и модели, учитывающие селективности предикатов для выбора подходящей конфигурации. Как следует из обзора [3], эта идея не пользовалась популярностью в литературе (лишь малая часть работ использовала подобную идею, например [11, 15, 27]). Однако запросы, ориентированные на чтение, и их характер делают ее привлекательной для рассмотрения. Например, селективности предикатов могут дать дополнительные указания на то, какие семейства следует стремиться сформировать.

5. Заключение. В настоящей работе мы рассмотрели вопрос организации физического уровня для колоночных СУБД:

1. Во введении были представлены подходы к организации физического уровня. Вкратце описаны возможные варианты устройства: вертикальное и горизонтальное фрагментирование, размещение;

2. Описано, в общих чертах, что такое колоночная СУБД и её отличия от классической СУБД с построчным хранением. Эти СУБД описаны в исторической перспективе: начиная от самых ранних и кончая современными. Приведены предпосылки для их появления и успеха;

3. Произведен обзор технологических решений, используемых при построении колоночных СУБД. Выделены важные для достижения высокой производительности компоненты;

4. Были описаны сильные и слабые стороны колоночных СУБД. Описаны два типа стандартных нагрузок для СУБД (OLAP и OLTP), приведены их свойства. Указаны работы, пытающиеся расширить нишу применения колоночных систем на OLTP нагрузки;

5. Охарактеризованы два наиболее известных исследовательских прототипа колоночной СУБД: ориентированный на работу с диском C-Store и ориентированный на работу в оперативной памяти MonetDB;

6. Наконец, был представлен анализ нужд физического уровня колоночной системы, рассмотрены отличия от классических систем. Были выработаны требования к функционалу рекомендательной системы структур физического уровня для колоночной СУБД.

Список литературы

[1] Кузнецов С.Д. Год эпохи перемен в технологии баз данных // Труды Института системного программирования РАН. Vol. 19. с. 9–34. 2010.

[2] Кузнецов С.Д. Mapreduce: внутри, снаружи или сбоку от параллельных субд? // Труды Института системного программирования РАН. Vol. 19. с. 35–70. 2010.

[3] Чернышев Г.А. Обзор подходов к организации физического уровня в субд // Тр. СПИИРАН. Vol. 1(24). с. 222–277. 2013.

[4] Abadi D., Madden S., Ferreira M. Integrating compression and execution in column-oriented database systems // Proceedings of the 2006 ACM SIGMOD international conference on Management of data. SIGMOD '06. New York, NY, USA : ACM, 2006. P. 671–682. URL: http://doi.acm.org/10.1145/1142473.1142548.

[5] Abadi D. J., Boncz P. A., Harizopoulos S. Column-oriented database systems // Proc. VLDB Endow. 2009. Vol. 2, no. 2. P. 1664–1665. URL: http://dl.acm.org/citation.cfm?id=1687553.1687625.

[6] Abadi D. J., Madden S. R., Hachem N. Column-stores vs. row-stores: how different are they really? // Proceedings of the 2008 ACM SIGMOD international conference on Management of data. SIGMOD '08. New York, NY, USA : ACM, 2008. P. 967–980. URL: http://doi.acm.org/10.1145/1376616.1376712.

[7] Automatic SQL tuning in oracle 10g / Benoit Dageville, Dinesh Das, Karl Dias et al. // Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. VLDB '04. VLDB Endowment, 2004. P. 1098–1109. URL: http://dl.acm.org/citation.cfm?id=1316689.1316784.

[8] Bellatreche L., Benkrid S. A joint design approach of partitioning and allocation in parallel data warehouses // Data Warehousing and Knowledge Discovery / Ed. by Torben Pedersen, Mukesh Mohania, A Tjoa. Springer Berlin / Heidelberg, 2009. Vol. 5691 of Lecture Notes in Computer Science. P. 99–110. 10.1007/978-3-642-03730-6_9. URL: http://dx.doi.org/10.1007/978-3-642-03730-6_9.

[9] Boncz P. A., Kersten M. L., Manegold S. Breaking the memory wall in monetdb // Commun. ACM. 2008. Vol. 51, no. 12. P. 77–85. URL: http://doi.acm.org/10.1145/1409360.1409380.

[10] C-store: a column-oriented dbms / Mike Stonebraker, Daniel J. Abadi, Adam Batkin et al. // Proceedings of the 31st international conference on Very large data bases. VLDB '05. VLDB Endowment, 2005. P. 553–564. URL: http://dl.acm.org/citation.cfm?id=1083592.1083658.

[11] Chu W., Ieong I. A transaction-based approach to vertical partitioning for relational database systems // Software Engineering, IEEE Transactions on. 1993. Vol. 19, no. 8. P. 804–812.

[12] Clotho: decoupling memory page layout from storage organization / Minglong Shao, Jiri Schindler, Steven W. Schlosser et al. // Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. VLDB '04. VLDB Endowment, 2004. P. 696–707. URL: http://dl.acm.org/citation.cfm?id=1316689.1316750.

[13] Cooperative scans: dynamic bandwidth sharing in a dbms / Marcin Zukowski, Sándor Héman, Niels Nes, Peter Boncz // Proceedings of the 33rd international conference on Very large data bases. VLDB '07. VLDB Endowment, 2007. P. 723–734. URL: http://portal.acm.org/citation.cfm?id=1325851.1325934.

[14] Copeland G. P., Khoshafian S. N. A decomposition storage model // SIGMOD Rec. 1985. Vol. 14, no. 4. P. 268–279. URL: http://doi.acm.org/10.1145/971699.318923.

[15] Cornell D., Yu P. An effective approach to vertical partitioning for physical design of relational databases // Software Engineering, IEEE Transactions on. 1990. Vol. 16, no. 2. P. 248–258.

[16] Das S., Agrawal D., El Abbadi A. G-store: a scalable data store for transactional multi key access in the cloud // Proceedings of the 1st ACM symposium on Cloud computing. SoCC '10. New York, NY, USA : ACM, 2010. P. 163–174. URL: http://doi.acm.org/10.1145/1807128.1807157.

[17] Database tuning advisor for Microsoft SQL Server 2005 / Sanjay Agrawal, Surajit Chaudhuri, Lubor Kollar et al. // Proceedings of VLDB. 2004. P. 1110–1121.

[18] Database tuning advisor for Microsoft SQL server 2005: demo / Sanjay Agrawal, Surajit Chaudhuri, Lubor Kollar et al. // Proceedings of the 2005 ACM SIGMOD international conference on Management of data. SIGMOD '05. New York, NY, USA : ACM, 2005. P. 930–932. URL: http://doi.acm.org/10.1145/1066157.1066292.

[19] DB2 advisor: an optimizer smart enough to recommend its own indexes / G. Valentin, M. Zuliani, D.C. Zilio et al. // Data Engineering, 2000. Proceedings. 16th International Conference on. 2000. P. 101–110.

[20] DB2 design advisor: integrated automatic physical database design / Daniel C. Zilio, Jun Rao, Sam Lightstone et al. // Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. VLDB '04. VLDB Endowment, 2004. P. 1087–1097. URL: http://dl.acm.org/citation.cfm?id=1316689.1316783.

[21] French C. D. "One size fits all" database architectures do not work for DSS // SIGMOD Rec. 1995. Vol. 24. P. 449–450. URL: http://doi.acm.org/10.1145/568271.223871.

[22] Graefe G. Query evaluation techniques for large databases // ACM Comput. Surv. 1993. jun. Vol. 25, no. 2. P. 73–169. URL: http://doi.acm.org/10.1145/152610.152611.

[23] Graefe G., Kuno H. Self-selecting, self-tuning, incrementally optimized indexes // Proceedings of the 13th International Conference on Extending Database Technology. EDBT '10. New York, NY, USA : ACM, 2010. P. 371–381. URL: http://doi.acm.org/10.1145/1739041.1739087.

[24] Hankins R. A., Patel J. M. Data morphing: an adaptive, cache-conscious storage technique // Proceedings of the 29th international conference on Very large data bases - Volume 29. VLDB '2003. VLDB Endowment, 2003. P. 417–428. URL: http://dl.acm.org/citation.cfm?id=1315451.1315488.

[25] Hellerstein J. M., Stonebraker M., Hamilton J. Architecture of a database system // Found. Trends databases. 2007. Vol. 1, no. 2. P. 141–259. URL: http://dx.doi.org/10.1561/1900000002.

[26] How to barter bits for chronons: compression and bandwidth trade offs for database scans / Allison L. Holloway, Vijayshankar Raman, Garret Swart, David J. DeWitt // Proceedings of the 2007 ACM SIGMOD international conference on Management of data. SIGMOD '07. New York, NY, USA : ACM, 2007. P. 389–400. URL: http://doi.acm.org/10.1145/1247480.1247525.

[27] Huang Y.-F., Chen J.-H. Fragment allocation in distributed database design // Journal of Information Science and Engineering. 2001. Vol. 17. P. 491–506.

[28] A hybrid row-column oltp database architecture for operational reporting / Jan Schaffner, Anja Bog, Jens Krüger, Alexander Zeier // BIRTE (Informal Proceedings). 2008.

[29] HYRISE: a main memory hybrid storage engine / Martin Grund, Jens Krüger, Hasso Plattner et al. // Proc. VLDB Endow. 2010. Vol. 4, no. 2. P. 105–116. URL: http://dl.acm.org/citation.cfm?id=1921071.1921077.

[30] Idreos S., Kersten M. L., Manegold S. Database cracking // CIDR. www.cidrdb.org, 2007. P. 68–78.

[31] Ivanova M., Kersten M. L., Nes N. Self-organizing strategies for a column-store database // Proceedings of the 11th international conference on Extending database technology: Advances in database technology. EDBT '08. New York, NY, USA : ACM, 2008. P. 157–168. URL: http://doi.acm.org/10.1145/1353343.1353366.

[32] Jaecksch B., Lehner W., Faerber F. A plan for olap // Proceedings of the 13th International Conference on Extending Database Technology. EDBT '10. New York, NY, USA : ACM, 2010. P. 681–686. URL: http://doi.acm.org/10.1145/1739041.1739126.

[33] Karasalo I., Svensson P. The design of cantor: a new system for data analysis // Proceedings of the 3rd international workshop on Statistical and scientific database management. Berkeley, CA, US : Lawrence Berkeley Laboratory, 1986. P. 224–244. URL: http://dl.acm.org/citation.cfm?id=1117037.1117068.

[34] Materialization strategies in a column-oriented dbms / Daniel J. Abadi, Daniel S. Myers, David J. DeWitt, Samuel Madden // ICDE / Ed. by Rada Chirkova, Asuman Dogac, M. Tamer Özsu, Timos K. Sellis. IEEE, 2007. P. 466–475.

[35] Nehme R., Bruno N. Automated partitioning design in parallel database systems // Proceedings of the 2011 international conference on Management of data. SIGMOD '11. New York, NY, USA : ACM, 2011. P. 1137–1148. URL: http://doi.acm.org/10.1145/1989323.1989444.

[36] Olap // Encyclopedia of Database Systems / Ed. by Ling Liu, M. Tamer Özsu. Springer US, 2009. P. 1947–1947. 10.1007/978-0-387-39940-9_3191. URL: http://dx.doi.org/10.1007/978-0-387-39940-9_3191.

[37] P. E. O'Neil, E. J. O'Neil and X. Chen. The Star Schema Benchmark (SSB). http://www.cs.umb.edu/ poneil/StarSchemaB.PDF. Дата просмотра: 30/07/2012.

[38] PARINDA: an interactive physical designer for PostgreSQL / Cristina Maier, Debabrata Dash, Ioannis Alagiannis et al. // Proceedings of the 13th International Conference on Extending Database Technology. EDBT '10. New York, NY, USA : ACM, 2010. P. 701–704. URL: http://doi.acm.org/10.1145/1739041.1739131.

[39] Performance tradeoffs in read-optimized databases / Stavros Harizopoulos, Velen Liang, Daniel J. Abadi, Samuel Madden // Proceedings of the 32nd international conference on Very large data bases. VLDB '06. VLDB Endowment, 2006. P. 487–498. URL: http://portal.acm.org/citation.cfm?id=1182635.1164170.

[40] Plattner H. A common database approach for oltp and olap using an in-memory column database // Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. SIGMOD '09. New York, NY, USA : ACM, 2009. P. 1–2. URL: http://doi.acm.org/10.1145/1559845.1559846.

[41] Supporting table partitioning by reference in oracle / George Eadon, Eugene Inseok Chong, Shrikanth Shankar et al. // Proceedings of the 2008 ACM SIGMOD international conference on Management of data. SIGMOD '08. New York, NY, USA : ACM, 2008. P. 1111–1122. URL: http://doi.acm.org/10.1145/1376616.1376727.

[42] Svensson P. The evolution of vertical database architectures — a historical review (keynote talk) // Proceedings of the 20th international conference on Scientific and Statistical Database Management. SSDBM '08. Berlin, Heidelberg : Springer-Verlag, 2008. P. 3–5. URL: http://dx.doi.org/10.1007/978-3-540-69497-7_3.

[43] Tamhankar A. M., Ram S. Database fragmentation and allocation: an integrated methodology and case study // Trans. Sys. Man Cyber. Part A. 1998. Vol. 28, no. 3. P. 288–305. URL: http://dx.doi.org/10.1109/3468.668961.

[44] The Vertica® Analytic Database Technical Overview White Paper. http://www.vertica.com/wp-content/uploads/2011/01/VerticaArchitectureWhitePaper.pdf. Дата просмотра: 30/07/2012.

[45] Thiem A., Sattler K.-U. An integrated approach to performance monitoring for autonomous tuning // Proceedings of the 2009 IEEE International Conference on Data Engineering. ICDE '09. Washington, DC, USA : IEEE Computer Society, 2009. P. 1671–1678. URL: http://dx.doi.org/10.1109/ICDE.2009.142.

[46] TPC Benchmark H. Decision Support. http://www.tpc.org/tpch. Дата просмотра: 30/07/2012.

[47] Weaving relations for cache performance / Anastassia Ailamaki, David J. DeWitt, Mark D. Hill, Marios Skounakis // Proceedings of the 27th International Conference on Very Large Data Bases. VLDB '01. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2001. P. 169–180. URL: http://dl.acm.org/citation.cfm?id=645927.672367.

Чернышев Георгий Алексеевич — ассистент кафедры информатики математико-механического факультета СПбГУ. Область научных интересов: распределенные базы данных. Число научных публикаций — 10. chernishev@gmail.com; телефон: +7(905)258-0279.

Chernishev George A. — assistant professor of Computer Science Department, SPbSU. Research interests: distributed databases. The number of publications - 10. chernishev@gmail.com; phone +7(905)258-0279.

Поддержка исследований. Работа выполнена при финансовой поддержке РФФИ, грант №12-07-31050.

Рекомендовано СПИИРАН, лаборатория ТиМПИ, заведующий лабораторией А.Л. Тулупьев, д.ф.-м.н., доц.

Статья поступила в редакцию 22.08.2013.

РЕФЕРАТ

Чернышев Г.А. **Организация физического уровня колоночных СУБД**

В статье рассматривался вопрос устройства физического уровня колоночных СУБД. Вначале были приведены основные моменты предыдущей статьи — обзора подходов к организации физического уровня в классических СУБД. Затем мы дали определение колоночной СУБД, охарактеризовали их и привели основные отличия от классических, важные как для пользователя, так и для разработчика. После этого нами были перечислены существующие исследовательские реализации и коммерческие системы.

В третьем разделе мы привели описание экспериментального сравнения прототипа колоночной СУБД и неназванной коммерческой системы. Была вкратце описана эталонная нагрузка Star Schema Benchmark. Были описаны результаты тестирования колоночной системы C-Store на данной нагрузке. После подведения итогов тестирования, было произведено перечисление сильных сторон колоночных СУБД. Мы рассмотрели работу с диском, сжатие, оперирование над сжатыми данными, эффективность при работе с аппаратным обеспечением, специализированные операторы и перспективы применения MapReduceв распределенном случае.

Кроме того, нами были рассмотрены две предпосылки для появления и популяризации колоночных систем: «новое аппаратное обеспечение» и «новые задачи». Было описано разделение СУБД по нагрузкам на два класса — класс аналитических (OLAP) и класс транзакционных (OLTP). Эти два класса охарактеризованы, приведены восемь основных различий. В конце данного раздела вкратце приведена история популяризации области OLAP приложений баз данных.

Наконец, в четвертом разделе мы обсудили вопросы устройства физического уровня в колоночной СУБД. Были рассмотрены перспективы использования популярных классических методов организации физического уровня: вертикального и горизонтального фрагментирования, вопросов размещения и миграции данных. Кроме них были рассмотрены и методы, характерные именно для колоночных СУБД: порядок сортировки колонки на диске и использование индексов соединения. Дополнительно мы рассмотрели менее популярные классические подходы: учет специфики операторов и учет селективностей предикатов. В результате нами были получены рекомендации для построения автоматической системы выбора конфигурации физического уровня для колоночных СУБД.

SUMMARY

Chernishev G.A. **Physical design approaches for column-stores**

In this paper we consider various physical design issues for column-store DBMS. In the introduction section we recollect the most important points of our previous paper. In that paper we surveyed physical design issues in the classical row-store DBMS. Then we define column-store database systems and describe them in brief. We give basic differences from the classical ones which are important both for users and system developers. Then we enumerate major research prototypes and provide a reference to commercial implementations.

In the third section we describe experimental evaluation which was carried out by the creators of C-Store. This evaluation involved undisclosed commercial system and Star Schema Benchmark. We describe this benchmark in brief and provide the results of evaluation. Then, we discuss strong points of column-stores, namely: effective disc usage, compression, hardware-friendliness (cache and CPU usage), specialized operators and MapReduce perspectives for distributed column-store.

Also, we consider two premises for the raise of interest for column-store databases: "new hardware"and "new tasks". We describe the division of database systems into two categories — analytical and transactional, depending on workload. These two categories are characterized and eight main differences are provided. In the end of this section we provide a brief history of popularization of OLAP databases.

Then, in the fourth section we discuss physical design aspects for column-store DBMS. We had considered perspectives of reuse of classical DBMS approaches: horizontal and vertical partitioning, allocation and migration of data. We also consider column-store specific approaches: column sort order and join index usage. Then we evaluate less popular classic approaches: taking into account operator specifics and selectivity. The result of this section is the recommendations for the construction of automatic system for the physical design of column-store DBMS.