УДК 004.65

Г.А. Чернышев

ОБЗОР ПОДХОДОВ К ОРГАНИЗАЦИИ ФИЗИЧЕСКОГО УРОВНЯ В СУБД

Чернышев Г.А. **Обзор подходов к организации физического уровня в СУБД.**
Аннотация. В данной работе мы рассмотрели различные методы организации физического уровня СУБД: вертикальное и горизонтальное фрагментирование, а также вкратце нами затронут вопрос репликации. Указанные методы были рассмотрены не только для локальных, но и для распределенных СУБД. Последним было уделено повышенное внимание: были рассмотрены методы размещения данных на узлах распределенной системы. Кроме теоретических работ, приведены работы практического характера, в которых освещены вопросы применения вышеуказанных методов в современных коммерческих СУБД. Они были рассмотрены как с позиции пользователя, так и с позиций архитектора и программиста СУБД.
Ключевые слова: базы данных, физический уровень СУБД, настройка СУБД, вертикальное фрагментирование, горизонтальное фрагментирование, размещение данных.

Chernishev G.A. **A survey of DBMS physical design approaches.**
Abstract. In this paper we survey various DBMS physical design options. We will consider both vertical and horizontal partitioning, and briefly cover replication. This survey is not limited only to local systems, but also includes distributed ones. The latter adds a new interesting question — how to actually distribute data among several processing nodes. Aside from theoretical approaches we consider the practical ones, implemented in any contemporary DBMS. We cover these aspects not only from user, but also architect and programmer perspectives.
Keywords: databases, physical database design, database tuning, vertical partitioning, horizontal partitioning, allocation.

1. Введение. Современные СУБД предоставляют богатый выбор вариантов организации обрабатываемых данных [7, 11–13, 29, 55, 56, 58, 59, 76, 124, 135, 143, 159, 163]. Это и вертикальное и горизонтальное фрагментирование отношений, выбор узлов хранения отношений в распределенной системе, выбор индексов и материализованных представлений, а также репликация отношений. Оптимальность выбора структур физического уровня играет немаловажную роль в обеспечении высокой производительности СУБД.

Исследователи занимались вопросами устройства физического уровня практически со времен становления баз данных как самостоятельной области информационно-компьютерных технологий. В литературе данная область называется «автоматическая на-

стройка СУБД» («automated database tuning», «self-management technology in databases») [39] и состоит из нескольких разделов, посвященных следующим вопросам:

1. применение фрагментирования данных (partitioning, fragmentation) — какие отношения и как следует фрагментировать. В свою очередь он разбивается на множество подвопросов: использование горизонтального (horizontal partitioning) и вертикального фрагментирования (vertical partitioning) [128], а также их комбинаций (называемых в литературе гибридным или смешанным [122] фрагментированием);

2. Размещение данных по узлам в случае распределенной СУБД (data allocation problem). Для размещения данных следует выбрать отношения и наборы узлов, определить атрибуты отношений, по которым будет производиться декластеризация (размещение с учетом возможной сборки [116]), и, наконец, выбрать метод декластеризации;

3. Использование дополнительных структур. Какие дополнительные структуры — индексы и материализованные представления (index selection problem, materialized view selection problem) — следует использовать для повышения производительности исполнителя запросов. В настоящей работе мы не рассматриваем эти вопросы, хороший обзор представлен в [113, 174];

4. Репликация данных (replication). Репликация данных определяется как процесс создания копии данных [175]. Она включает в себя следующие подвопросы: когда целесообразно реплицировать данные, как создавать и поддерживать реплицированные данные, как выбирать данные для репликации и пр. Работы [88,103,128,161,175] содержат классификации подходов реплицирования данных, они могут служить хорошей отправной точкой для изучения данных вопросов. В нашем исследовании мы не разбираем эти вопросы систематически, а рассматриваем только отдельные работы, которые решают задачу репликации совместно с задачей фрагментирования или размещения.

Методы решения этих вопросов могут являться чрезвычайно мощными [1,167] средствами повышения производительности системы,

поэтому им следует уделять повышенное внимание при построении физического уровня базы данных. Выбор конкретного метода (и его применение к схеме) может производиться как вручную, усилиями администратора базы данных, так и в автоматическом или полуавтоматическом (иначе говоря, автоматизированном) режиме. В последнем случае администратору будут предлагаться альтернативы, возможно каким-либо образом аннотированные и ранжированные в соответствии с потенциальной выгодой и затратами, из которых он сможет выбрать подходящую. Необходимо заметить, что в настоящее время преобладают тенденции полного исключения вмешательства человека в процесс настройки СУБД, то есть стремление сделать его автоматическим [39, 76, 163]. Почти все современные СУБД обладают в той или иной степени функционалом автоматического принятия решений по каждому из представленных вопросов (подробно представлены в разделе 3.3). Кроме того, существует масса сторонних инструментов [59]. Успешное решение этих вопросов может позволить:

1. Увеличить допустимый объем загружаемых и обрабатываемых данных за счет внесения в систему новых узлов и устройств хранения;

2. уменьшить время выполнения отдельного запроса посредством использования специальных структур данных и методов их обработки, а также с помощью использования внутризапросного параллелизма;

3. уменьшить время выполнения набора запросов с помощью использования межзапросного параллелизма (повышение степени параллелизма системы);

4. уменьшить затраты на обслуживание системы и тем самым понизить совокупную стоимость владения (материальные расходы) [39, 83].

Согласно классификации [83], алгоритмические решения поставленных вопросов могут быть двух типов — статическими (static) и динамическими (dynamic). Первые из них не реагируют на изменение схемы данных или нагрузки, используя выбранные значения параметров в течение всего времени работы. Вторые могут принимать решения об изменении установок в процессе работы.

Например, в случае изменения доминирующего запроса, для увеличения производительности системы может быть целесообразно кластеризовать таблицу по другому ключу. Подробно эти классы алгоритмов и актуальные вопросы построения и использования динамических алгоритмов обсуждаются в [83].

Также необходимо отметить, что существующие решения не всегда учитывают только один избранный аспект в отдельности, но учитывают сразу несколько в совокупности. Например, задача может формулироваться так: как фрагментировать и разместить данные по узлам распределенной СУБД с возможностью репликации. Попытка классифицировать работы согласно решаемым задачам предпринята в [122].

Историю развития области настройки СУБД принято подразделять на три этапа [124]:

1. Разработка моделей, описывающих поведение системы через стоимостную функцию. Данный подход зародился в конце 60-х и был первой попыткой решить задачу о выборе структур физического уровня. Эти работы можно охарактеризовать как формулирование и решение экстремальной задачи. Нам важно ознакомиться с ними, так как их результаты составляют фундаментальные основы современных автоматизированных систем. Кроме того, эти подходы используются до сих пор, наряду с современными (например [58, 132]). Мы подробно рассмотрим эти модели в разделах «Горизонтальное фрагментирование и размещение» и «Вертикальное фрагментирование»;

2. Работы, использующие оптимизатор запросов СУБД для оценки потенциальных решений-кандидатов («what-if» оптимизация). Этот тип работ появился в конце 80-х и в настоящее время доминирует. Пример такой работы будет рассмотрен далее, во втором разделе;

3. Глубокая интеграция с оптимизатором запросов. Это относительно недавнее направление, которое предлагает выполнять поиск ответа внутри оптимизатора запросов. Обзор современных работ данного и предыдущего типа будет произведен в конце раздела «Фрагментирование и размещение данных: избранные работы».

Цель настоящей работы — на основе сравнительного анализа источников представить обзор возможных подходов к решению задач, связанных с выбором структур физического уровня СУБД. Существующие в настоящий момент обзоры недостаточны по нескольким причинам: покрывают только один, отдельно взятый аспект выбора физического представления [113, 174], при этом не являющийся основным фокусом настоящей работы, или не обладают достаточной полнотой и широтой охвата [83, 110], либо, наконец, устарели [63].

Основная часть работы состоит из трех разделов. В следующем разделе мы ознакомимся с формализованной постановкой задачи настройки СУБД, методами ее решения, а также разберем в качестве примера одну из современных работ [7]. Затем, в третьем разделе рассмотрим вопросы горизонтального фрагментирования и размещения данных в распределенной СУБД. Мы обстоятельно изучим эти вопросы и начнем с рассмотрения способов фрагментирования как инструментов, предоставляемых разработчикам баз данных. После этого изучим стоимостные модели фрагментирования и размещения. Начав с самых ранних, мы проследим их эволюцию до настоящего времени. В конце этого раздела разберем работы, описывающие автоматизированные рекомендательные системы в современных коммерческих СУБД. Аналогичным способом в четвертом разделе будет представлено вертикальное фрагментирование.

2. Постановка задачи и методы решения.

2.1. Постановка задачи о настройке СУБД. Рассмотрим в общем виде задачу о настройке СУБД [39]. Работу СУБД можно описать как абстрактную функцию:

$$f : C \times W \to P,$$

где C обозначает конфигурацию системы, W — нагрузку, а P — производительность. Конфигурация — это описательная характеристика системы, включающая в себя:

1. аппаратную конфигурацию (количество и характеристики процессоров, характеристики оперативной памяти, характеристики дисковых устройств, характеристики используемой сети и т.д.);

2. программную конфигурацию — настраиваемые параметры СУБД, вступающие в силу после загрузки (например, интервал времени до повышения гранулярности замков);

3. конфигурацию физического уровня СУБД (использование индексов, материализованных представлений и др.);

4. процедуры резервного копирования и репликации;

5. стратегии адаптации к изменяющимся условиям;

6. стратегии обработки исключительных ситуаций.

Нагрузка — это предполагаемый набор запросов и их характеристики — затрагиваемые атрибуты, типы запросов, интенсивность поступления в заданные периоды (рабочие или пиковые часы), распределение параметров этих запросов и др.

Производительность P — это некоторый численный показатель, характеризующий работу системы. В качестве производительности могут браться:

1. пропускная способность системы, измеряемая в количестве обработанных запросов в единицу времени (throughput);

2. время ответа на запрос (response time);

3. различные метрики измерения функциональной надежности (dependability) — доступность (availability), надежность (reliability) и производительность в условиях ослабленных конфигураций (performability).

В этих терминах задача формулируется следующим образом [39]: для известной нагрузки необходимо найти конфигурацию, позволяющую достичь заданного уровня производительности.

Данное определение — общее, в этой же работе нас интересует весьма узкая задача — выбор физического уровня СУБД. Поэтому потребуются некоторые уточнения.

1. Конфигурация определяется как конфигурация физического уровня СУБД. Все остальные компоненты конфигурации (чаще всего аппаратная или программная конфигурация) фиксированы и выступают в роли условий. Иногда [55] часть конфигурации известна заранее;

2. Информация о нагрузке дополнена достаточно подробной информацией о данных, к которым обращаются запросы (например, количество и характеристики атрибутов таблиц);

3. Возможно наличие пользовательских ограничений, не связанных напрямую с системой (могут считаться частью нагрузки). Например, передача данных между узлами в распределенной компьютерной сети осуществляется за плату, и для решения задачи выделен определенный бюджет.

В итоге, уточненная формулировка выглядит следующим образом: найти конфигурацию физического уровня, обладающую заданной производительностью. Однако чаще всего требуемый уровень производительности заранее не известен, и требуется найти максимально достижимый. Таким образом, мы получаем экстремальную задачу, где в качестве максимизируемой функции выступает производительность P, ограничения описываются набором уравнений и неравенств, а искомая конфигурация представлена набором параметров.

Такая постановка достаточно точно описывает задачи фрагментирования и размещения, которым посвящена настоящая работа.В следующих разделах эти формулировки будут уточнены и дополнены деталями, характерными для каждой задачи в отдельности.

2.2. О решении. С учетом сказанного в предыдущем разделе и рассмотренной совокупности работ, точным решением мы будем называть набор структур физического уровня СУБД, обеспечивающий наибольшую производительность для указанной нагрузки.

Для нахождения решения необходимо иметь возможность оценивать конфигурации. В качестве одного из возможных способов часто предлагают стоимостную модель, учитывающую характеристики аппаратной конфигурации, характеристики нагрузки и, возможно, другие пользовательские ограничения.

Самый очевидный метод отыскания точного решения — используя эту модель, оценить каждую конфигурацию и выбрать наилучшую среди всех возможных. Однако при этом возникают сложности с генерацией множества допустимых конфигураций, а также с их оценкой. Количество и специфика параметров ведет к комбинаторному взрыву (combinatorial explosion) числа конфигураций даже в простых случаях. Например, число вариантов вертикального фрагментирования отношения равняется числу Белла [83, 172]. В

общем случае нахождение оптимального решения для подобной задачи является NP-трудным [16,25,128], и таким образом, не подходит для практических применений. Поэтому, для решения поставленной задачи применяются следующие два типа методов поиска приближенного решения — процедурный и стоимостной.

Первый подход предлагает некоторую процедуру, результат которой и будет приближенным решением. При этом процедура не работает с моделью, описывающей поведение системы, и не обращается к функции стоимости выполнения запросов для оценки конкретной конфигурации. Вместо этого она последовательно формирует ответ, исходя, часто из неочевидных, посылок о том, каким должно быть решение. Данный метод обычно сопровождается доказательством или обоснованием, почему он дает хорошее решение. Примером может служить работа [123], где для выработки вертикальных фрагментов используется поиск специальных структур (affinity cycle) в графе. Методы данного подхода особенно популярны при решении задачи вертикального фрагментирования; они будут широко представлены в соответствующем разделе. Кроме того, к данному типу можно отнести и работы, предлагающие различные методологии выбора структур физического уровня.

Стоимостные подходы, в свою очередь, также можно поделить на два типа. Как мы уже отмечали в предыдущем разделе, в общем виде задача выбора структур физического уровня в СУБД может трактоваться как экстремальная задача. Разница между этими двумя типами состоит в способе решения данной задачи.

- Стоимостной подход, использующий для поиска решения математические методы целочисленного программирования (integer programming). Это самый ранний тип работ, первые исследования можно отнести к концу 60-х годов прошлого века. Первые работы в значительной степени фокусировались именно на математических методах решения (использование лагранжевой релаксации или вопросы линеаризации уравнений) [44, 46, 60] и вопросах их решения на ЭВМ. Позднее, фокус сместился непосредственно к моделям, а для решения систем уравнений стали применяться различные математические пакеты [52,117,132]. Данный подход наиболее популярен в работах, посвященных размещению данных в распределенных СУБД.

- Стоимостной подход, в котором для нахождения решения используются классические методы комбинаторной оптимизации наподобие метода ветвей и границ [39]. Сюда же отнесем и работы, использующие и эвристические алгоритмы — генетические алгоритмы, алгоритмы поиска локального минимума, метод отжига и др. Эти работы появились несколько позже, однако сейчас они наиболее популярны в применении к рассматриваемым задачам.

2.3. Пример. В качестве примера стоимостного подхода рассмотрим работу Агравала и др. [7]. Согласно условию, необходимо выбрать атрибуты для построения индексов, а также произвести вертикальное и горизонтальное фрагментирование. Для локальной СУБД необходимо отыскать указанные структуры, дающие наибольший прирост производительности для известного заранее набора запросов.

Метод горизонтального фрагментирования предполагает разбиение отношения на несколько частей, содержащих записи с исходным набором атрибутов. Неформально говоря, производится «разрезание» отношения поперек атрибутов. Формальное определение горизонтального фрагментирования, свойства, а также обзор работ приведен в следующем разделе.

В разбираемой работе используются два способа горизонтального фрагментирования — интервальный и с использованием хеш-функции. Фрагментирование производится на основании значения (значений) выбранного атрибута или атрибутов. Первый способ использует набор граничных значений для задания результирующих фрагментов (предполагается, что они не пересекаются). Во втором способе количество фрагментов и принадлежность записи к фрагменту определяются с помощью хеш-функции. Разница между этими способами заключается в их различном влиянии на выполнение каждого конкретного запроса. Например, в случае запроса, выбирающего записи, превышающие какое-то значение, целесообразно фрагментировать отношение интервальным способом по задействованному атрибуту. Данный выбор и выбор стратегии фрагментирования в целом, во многом обуславливается механизмом выполнения запросов. Фрагментирование и исполнение запросов зависят друг от друга [69] и должны рассматриваться совместно; однако этот вопрос выходит за рамки настоящей рабо-

ты. В качестве введения в исполнение и оптимизацию запросов в современной СУБД мы рекомендуем [89].

Идея вертикального фрагментирования состоит в «разрезании» отношения вдоль, так чтобы разделить атрибуты на несколько групп. При этом каждый атрибут, помимо ключевых, должен принадлежать только одной группе. Для обеспечения восстановимости исходной записи, ключевые атрибуты должны присутствовать во всех группах. Распространенной практикой является выделение в отдельный фрагмент совместно запрашиваемых атрибутов.

В работе [7] вводится понятие структуры физического уровня — тройки, состоящей из объекта, метода горизонтального фрагментирования и набора колонок. Эта тройка — средство формального описания возможных способов фрагментирования одного отношения. В данном определении объект включает в себя отношение и метод доступа к нему (access path), а именно: кластеризованный индекс, некластеризованный индекс или же материализованное представление. Метод горизонтального фрагментирования — один из двух упомянутых методов, применяющийся к указанному набору колонок. Вертикальное фрагментирование в этой структуре тоже учитывается — в качестве объекта может браться как исходное отношение, так и вертикальный фрагмент, к которому добавлен метод доступа (например, построенный индекс на основе фрагмента).

Конфигурация определяется как набор допустимых структур физического уровня. Допустимая структура — та, которую можно реализовать в данной схеме базы данных. Например, невозможно иметь два кластеризованных индекса для одной таблицы.

В условиях вышесказанного, задача формулируется таким образом: найти такую конфигурацию, состоящую из структур физического уровня указанных типов, которая:

1. Минимизирует суммарную стоимость исполнения известного набора запросов;
2. Укладывается в указанное ограничение на объем данных;
3. Выполняет другие ограничения на структуры физического уровня.

В рассматриваемой работе не предлагается никаких специальных моделей для оценки конфигурации. Вместо этого используется

оптимизатор в режиме «what-if» («что если»). Данный механизм позволяет оптимизатору оценить время выполнения запроса для указанных структур представления данных. При этом для оценки используется внутренняя модель, то есть оценка производится без создания этих структур и без выполнения запросов.

Итак, можно сказать, что у нас имеются в наличии все компоненты, перечисленные в разделе 2.1:

1. Конфигурация системы C — набор структур физического уровня, о которых речь шла выше, ограничения на них, а также ограничение на объем занимаемой этими структурами памяти;

2. Нагрузка W представлена в виде набора запросов;

3. Производительность P, вычисляемая как суммарное время выполнения всех запросов.

Функция f, связывающая эти компоненты, представлена оптимизатором. Решение данной задачи было разбито на следующие четыре этапа:

1. Отсев вертикальных фрагментов. Цель данного этапа — уменьшить количество вертикальных фрагментов, учитываемых в дальнейшем. Из рассмотрения исключаются те, что не несут потенциальной выгоды, и, скорее всего, не будут присутствовать в решении. Оценка выгоды производится на основании модели, учитывающей интенсивность использования атрибутов;

2. Выбор базовых структур. На данном шаге выбираются наиболее выгодные физические структуры для каждого запроса в отдельности. Для оценки применяется вышеупомянутый оптимизатор;

3. Этап слияния. Множество структур, полученных на предыдущем шаге, дополняется структурами, полученными в результате их смешивания. Этот шаг используется, например, для борьбы с чрезмерной специализацией структур;

4. Поиск итогового решения среди структур, полученных на предыдущем шаге. Авторы работы разработали алгоритм

$Greedy(m, k)$, гарантирующий оптимальный ответ при выборе m структур из k, а для поиска остальных использующий жадный алгоритм.

3. Горизонтальное фрагментирование и размещение данных. В этом разделе мы рассмотрим избранные работы на темы горизонтального фрагментирования и распределения данных. Мы рассмотрим эти вопросы в одном разделе, так как они взаимосвязаны – результаты фрагментирования используются при распределении. Однако, большинство работ рассматривают эти аспекты независимо [161].

3.1. Горизонтальное фрагментирование. Горизонтальное фрагментирование разбивает отношение на набор фрагментов, каждый из которых будет хранить часть записей исходного отношения. В [128] приведены формальные критерии корректности фрагментирования: восстановимость исходного отношения, полнота и попарное непересечение фрагментов. Там же отмечена схожесть данных правил с правилами нормализации СУБД в смысле необязательности исполнения. Большинство работ по горизонтальному фрагментированию следуют этим трем правилам, хотя иногда последнее правило игнорируется, и тогда можно считать, что производится репликация.

Горизонтальное фрагментирование повышает производительность СУБД за счет уменьшения объемов данных, которые потребуется обработать при выполнении запроса. Данные стремятся локализовать в одном или нескольких фрагментах суммарно меньших объемов, чем исходное отношение. Фрагментирование — инструмент администратора базы данных, поддерживающийся практически в каждом диалекте SQL DDL.

В настоящее время распространены несколько методов горизонтального фрагментирования [90]. Их можно разбить на два класса, первый из которых использует значения атрибутов для размещения записей по фрагментам, а второй не зависит от них. Среди методов первого класса наиболее популярны следующие:

1. Интервальное фрагментирование (range data partitioning). Домены атрибутов фрагментирования делятся на несколько непересекающихся интервалов. Запись относится к выбранному фрагменту, если значение атрибута (атрибутов) принадлежит к характеризующему интервалу;

2. Фрагментирование с помощью хеш-функции (hash data partitioning). Результат применения хеш-функции к значениям выбранных атрибутов определяет принадлежность записи к конкретному фрагменту;

3. Списковое фрагментирование (list partitioning) [23], при котором каждый фрагмент задается списком допустимых значений;

4. Колоночное фрагментирование (column partitioning) [23]. В данном случае в отношение добавляется виртуальный атрибут, значение которого задается формулой от набора других атрибутов. Затем, для этой колонки применяются вышеупомянутые методы фрагментирования.

Также возможно применять фрагментирование, не зависящее от значений атрибутов. Большой плюс этих методов — отсутствие неравномерного распределения нагрузки по узлам (skew). Вот некоторые из них:

1. Случайное равномерное фрагментирование (random-equal data partitioning). Записи распределяются между фрагментами случайным образом;

2. Круговое фрагментирование (round-robin data partitioning). Является наиболее популярной реализацией этого класса методов фрагментирования. Записи относятся к фрагментам последовательно, по кругу, в зависимости от номера.

Кроме вышеперечисленных, в ряде современных СУБД поддерживается и составное (composite) фрагментирование [23]. Оно заключается в последовательном применении двух методов фрагментирования, к возможно различным атрибутам. Например, range-hash метод сначала фрагментирует интервальным методом, затем с помощью хеш-функции.

Дополнительно, в [83] выделяется направление фрагментирования, называемое «кластеризация записей» (record clustering). Идея данного подхода состоит в том, чтобы создавать фрагменты в результате исполнения запросов, так чтобы совместно запрашиваемые данные находились в одном фрагменте. Для достижения этого предлагается, в частности, использовать различ-

ные алгоритмы кластеризации. К данным работам можно отнести [5,10,17,98,118,127]. Этот подход напоминает современное направление адаптивного индексирования [97].

Существуют и другие способы формирования фрагментов. В [33,34] предлагается формировать горизонтальные фрагменты, используя извлеченные из набора запросов предикаты. Аналогичный подход используется в [42,105,122,125,128,180]. В работе [151] используются методы логического вывода. Внешняя по отношению к системе информация, предоставляемая пользователем, используется для получения фрагмнтов в [88].

Работа [33] делит методы фрагментирования на основные (primary fragmentation) и производные (derived fragmentation). Основное фрагментирование отношения определяется только значениями собственных атрибутов, в то время как производное дополнительно зависит и от фрагментов другого отношения [180]. Производное фрагментирование применяется при фрагментировании отношения, связанного посредством внешнего ключа с некоторым другим. Идея состоит в формировании фрагментов таким образом, чтобы каждому фрагменту первого отношения сопоставлялся фрагмент второго. Этот метод позволяют получать цепочки фрагментирования наборов отношений, связанных ключами. В современных работах [23] эти два типа фрагментирования называются моно- и табличнозависимое фрагментирования (mono и table dependent соответственно). Иногда [159] табличнозависимое фрагментирование называют ссылочным (referential).

Необходимо заметить, что кластеризация записей и пара основное-производное фрагментирование получили широкое распространение в задачах фрагментирования объектных СУБД [15,22,115,150, 166].

Отдельно можно отметить ряд работ, посвященных фрагментированию данных в OLAP системах [18–21,23,24,27,53,74,81,106,107, 125,137,147,156–158,169]. Характерной чертой этих работ является учет схемы данных — наличия одной одной или нескольких таблиц фактов (fact table) и таблиц измерений (dimension table). В этих работах популярен следующий подход: горизонтально фрагментировать таблицу фактов (обычно большого объема), разместить ее на узлах распределенной СУБД и реплицировать таблицы измерений.

Дополнительный перечень и разбор современных работ по го-

ризонтальному фрагментированию не только для реляционных, но и для объектных систем, может быть найден в работах [88] (повышенное внимание уделено вопросам репликации), [83], а также в диссертации [110] (освещаются, в том числе, объектные системы).

3.2. Задача размещения данных на узлах распределенной СУБД. Горизонтальное фрагментирование часто рассматривается в контексте распределенной СУБД (заметим, что оно применяется и для локальных СУБД [7, 20, 34]). В таком случае задача выбора фрагментов решается совместно с задачей размещения данных по узлам. Одной из целей такого совместного рассмотрения является повышение производительности или надежности распределенной СУБД. Альтернативную цель — балансировку нагрузки (load balancing) — в данной работе мы подробно рассматривать не будем, хотя она и будет вкратце затронута в конце данного раздела.

Постановка задачи размещения данных описывается схемой из 2.2, однако требуются некоторые уточнения.

1. Аппаратная конфигурация включает в себя число узлов и их характеристики. Также описываются параметры сети;

2. Искомая конфигурация физического уровня должна сопоставлять каждому фрагменту узел и при этом не нарушать ограничения (например, объем жесткого диска узла не должен быть превышен);

3. В качестве метрики P дополнительно может выступать суммарный объем переданных по сети данных [9, 33, 49, 85, 94, 99, 101, 109, 149].

Один из острейших вопросов данной области исследований — целесообразно ли разрабатывать метод распределения данных в отрыве от фрагментирования (итеративный дизайн [18, 21]) или же следует решать эти два аспекта совместно (комбинированный или интегрированный [59] дизайн). Сторонники раздельного решения утверждают, что совместное решение не обладает должной гибкостью [69]. В то же время алгоритм фрагментирования, работающий с произвольным набором фрагментов, может давать в целом худший результат [18, 161]. Последнее верно в силу того, что теряется часть семантики разбиения отношения на фрагменты и

ее невозможно использовать при размещении данных. Если же сохранять и передавать эту семантику, то теряется универсальность алгоритма размещения. В итоге, данная проблема привела к появлению двух независимых сообществ исследователей [18].

Движущий принцип как размещения, так и фрагментирования — «локальность ссылки» (locality of reference) [71, 149, 161]. Он состоит в совместном хранении данных, которые будут совместно запрашиваться, будь то записи внутри фрагмента или же фрагменты внутри узла. Во многих случаях это позволяет снизить стоимость обработки запроса. В работе [152] утверждается, что согласно [35] в хорошо спроектированной базе данных, 90% обращений к данным должны быть локальными. Например, следуя этому принципу, можно было бы фрагментировать данные и расположить фрагменты на узлах так, чтобы при исполнении заранее известного запроса, содержащего соединения, можно было бы целиком исключить передачу данных между узлами. Значительное количество работ опирается на данную идею.

Задача размещения бывает поставлена в двух вариантах [35]:

1. Неизбыточное размещение, при котором каждый фрагмент или отношение находится ровно на одном узле;

2. Избыточное размещение, то есть репликация, может применяться для повышения надежности или производительности СУБД.

Исторически, задача размещения данных «выросла» из задачи о размещении файлов на узлах распределенной компьютерной сети (FAP, file allocation problem) [69]. Эта задача, по сути, представляет собой упрощенный вариант исходной и определяется аналогичным образом, но в качестве запросов рассматривает простые операции с файлом или набором файлов [176].

Первая работа [46] формулировала задачу в терминах целочисленного программирования (integer programming). В ней, как и во многих последующих, рассматривался вопрос решения экстремальной задачи с использованием математических методов. Позднее появились и работы, где предлагались эвристические методы поиска решения [31–33, 114, 119, 176]. Одна из причин отказа от решения методами целочисленного программирования — слишком большое количество неизвестных [33, 99, 112].

Работа [63] — старый, но обладающий достаточной полнотой, обзор работ данной тематики. В нем представлено более десяти типов моделей для задачи о размещении файлов.

Однако заимствованные подходы оказались слабо приспособленными. Это объясняется тем, что модели не учитывали специфику исполнения запросов в СУБД (например, необходимость узлов обмениваться информацией друг с другом при выполнении реляционной операции соединения) [9, 149]. Дополнительная критика ранних моделей приведена в [160].

Поэтому позднее появились работы [25, 33, 36, 57, 61, 75, 177] в которых предлагались специализированные модели. Появившиеся модели могли включать в себя стоимость передачи данных между узлами, стоимость размещения данных на узлах, а так же стоимость выполнения запросов характерных именно для СУБД. Важно отметить, что модели разделяют запросы на обновления и запросы на выборки, учитывая их различными способами. Кроме перечисленных компонентов, часто присутствуют и другие, характерные для каждой постановки задачи в отдельности.

Среди необычных типов моделей можно назвать:

1. Модели, учитывающие проведение распределенных транзакций. В качестве параметров в них дополнительно может фигурировать стоимость запроса замка или количество сообщений, которые необходимо передать [36, 140, 141, 161];

2. Модели, требующие обеспечения определенного уровня надежности [44, 112, 120, 136, 160];

3. Модели, учитывающие характер и стоимости отдельных реляционных операторов (например, оператора соединения), составляющих план запроса [51, 52, 61, 99, 101, 114, 117, 149].

Оптимальное решение как задачи размещения файла, так и задачи размещения данных в СУБД является NP-трудным [16, 32, 68, 128, 148]. Поэтому в подавляющем большинстве работ ищется приближенное решение с использованием эвристических методов, упомянутых в 2.2. Кроме того, можно отметить применение генетических алгоритмов для решения задачи как размещения, так и фрагментирования [2, 3, 18, 49, 69, 73, 114], методов машинного обучения [37] и методов муравьиной оптимизации (ant colony optimization) [100].

Приближенные процедурные (эмпирические) методы так же предлагались для решения данных задач [85, 161]. Большой список подобных работ и разбор [161] будет представлен в следующем разделе.

Первые работы по размещению данных рассматривали задачу в статическом контексте — когда ни запросы, ни данные не изменяются. Однако со временем нагрузка может меняться или уточняться, может поменяться топология сети [144] и т.д. Всё это может негативно сказаться на производительности системы. Поэтому позже появились работы, изучавшие и миграцию (migration) данных [30,37,54,66,88,93,99,102,144,152,168] в условиях меняющейся нагрузки.

В результате миграции данных обеспечивается выравнивание загрузки узлов при исполнении запросов. То есть можно сказать, что в данных работах решается задача балансировки нагрузки. Однако этот вопрос гораздо более объемен и не укладывается в тематику настоящей работы, поэтому рассматривать его целиком мы не станем (дополнительная причина будет приведена в 5.3), ограничившись лишь общей схемой, применявшейся в вышеперечисленных статьях.

1. Собирается статистика обращения к фрагментам (например, с помощью счетчиков);

2. Через определенный промежуток времени или по выполнению определенных условий (например, появлению новых запросов) производится анализ существующей конфигурации;

3. В зависимости от пороговых значений счетчиков или потенциальной стоимости новой конфигурации принимается решение об изменениях — миграции фрагмента на другой узел или его изменении (расщеплении или слиянии с другим).

Важно отметить, что работы стоимостного характера расширяют множество характеристик P. Дополнительно могут учитываться время завершения миграции (makespan, completion time), количество задействованных узлов и т.д. [144]. Исследователи стремятся минимизировать данные характеристики, так как реорганизация замедляет или приостанавливает штатную работу СУБД.

Другой важный вопрос — каким узлом принимается решение о проведении реорганизации [88]. Рассматриваемые подходы мо-

гут быть централизованными [7, 9, 69] или децентрализованными [30, 168]. В первом случае выделяется узел-координатор, который собирает информацию со всех узлов и принимает решение для каждого узла. При децентрализованном подходе каждый узел принимает решение самостоятельно. Также есть и решения, в которых присутствуют несколько координаторов, каждый из которых обслуживает свою группу узлов [88]. Для ознакомления со списком работ по вопросам динамического размещения с уклоном в репликацию, а также с представленной таблицей классификации мы рекомендуем [88]. Эта работа содержит глубокий и профессионально выполненный обзор, включает в себя обширный список работ и их классификацию, хотя и не полностью покрывает работы, упомянутые в настоящем исследовании.

Сейчас же мы рассмотрим отдельные работы, где решается вопрос, каким образом следует эффективно применять как фрагментирование, так и размещение, как следует выбирать метод и атрибуты для применения, какую архитектуру системы использовать, как эффективно проводить поиск решения и т.д.

3.3. Фрагментирование и размещение данных: избранные работы. Одними из первых работ, совмещавших горизонтальное фрагментирование и распределение данных были [9, 33, 148]. В них ставился вопрос о фрагментировании набора отношений и размещении их на узлах распределенной СУБД. В качестве исходных данных выступал заранее заданный набор транзакций и характер доступа к данным. Разработанная в [148] модель учитывает несколько ограничений (выраженных в виде стоимостей): ограничение на объем переданных данных между любой парой узлов, ограничения объема считанных с диска данных и ограничение затраченного процессорного времени. Последние два заданы для каждого узла в отдельности. В условиях этих ограничений необходимо найти такую конфигурацию фрагментов, которая бы минимизировала суммы всех вышеперечисленных стоимостей. В работе было произведено доказательство NP-трудности данной задачи, предложен приближенный, в данном случае модифицированный жадный алгоритм решения.

В работе [9] задача ставилась похожим образом, но с другими требованиями к фрагментам. Выполняя набор запросов, необходимо минимизировать объем переданных данных между узлами. Этот вопрос решается не только относительно горизонтальных, но

и вертикальных фрагментов, а также рассматривается вопрос выработки фрагментов. В работе минимизируются стоимости передачи данных между фрагментами и стоимости передачи данных до узла, инициировавшего запрос [25]. В остальном, задача и ее решение достаточно точно следуют стоимостной схеме, описанной в пункте 2.2.

Вопросы реализации составного фрагментирования совместно с размещением разбираются в [54, 77–79].

Составное фрагментирование одного атрибута, комбинирующее интервальное и круговое фрагментирования, было представлено в [77]. Составное многоатрибутное фрагментирование рассматривалось в [54, 78, 79]. В [78] для реализации составного фрагментирования вида «интервал-интервал» использовался двухмерный гридфайл. В [54] была предложена схема, где для нахождения записи по условию на атрибут вторичного фрагментирования необходимо было бы опросить максимум два фрагмента. Подробный разбор этих методов представлен в [90].

В [69] сравниваются эволюционные методы решения задачи размещения фрагментов по узлам в распределенной СУБД. Модель, предложенная в этой работе, основана на учете зависимостей по данным между фрагментами. В результате была получена функция, выражающая общую стоимость выполнения запроса в объеме переданных данных, которую и необходимо минимизировать. В экспериментальном сравнении с оптимальным решением наилучшие результаты показали подходы на основе генетических алгоритмов; подходы на основе метода поиска локального минимума и нейронных сетей с использованием метода отжига показали худший результат. К недостаткам этой работы можно отнести отсутствие экспериментов на эталонном тесте. Другой недостаток — рассмотрение задачи разложения фрагментов по узлам без учета метода их получения. Последнее замечание существенно, ибо данный подход не предоставляет возможность использовать семантику взаимосвязей между фрагментами.

Задача горизонтального фрагментирования и размещения в OLAP системе решалась в цикле работ [18,19,169]. Для фрагментирования использовался генетический алгоритм, а для размещения использовалась идея близости атрибутов и поиска цикла [123] (подробно рассмотрим в разделе о вертикальном фрагментировании).

Теперь подробнее рассмотрим вопрос о производном фрагмен-

тировании. Определение производного фрагментирования впервые было представлено в [33], однако эта работа была в значительной степени теоретической, изучала вопрос только с использованием моделей и рассматривалась как задача целочисленного линейного программирования. Были предложены модель и метод решения, использующий декомпозицию, отдельным шагом учитывалась репликация.

Одной из первых работ, использовавших данный подход на практике, была [62]. В ней решается задача горизонтального фрагментирования, размещения и репликации в распределенной СУБД. Суть работы состоит в том, чтобы для обеспечения фрагментирования и размещения использовать предикаты совместно с триггерами. Предикаты задают принадлежность каждой записи фрагменту, а фрагмента — множеству узлов. Триггеры, в свою очередь, используются для обеспечения корректности при выполнении распределенных транзакций и распространении изменений. Авторы работы исследовали эти вопросы в контексте децентрализации существующей базы данных коммерческой организации. Был получен интересный результат — в их системе производное фрагментирование не формирует цепочек длины более чем три отношения [62].

Способ совместного фрагментирования пары таблиц аналогичный [33,62] предлагается в [159]. В работе было предложено расширение SQL DDL, позволяющее распространить фрагментирование одной таблицы на другую, если та связана с первой посредством внешнего ключа. Этот подход, называемый ссылочным фрагментированием, существенно упрощает управление фрагментами, позволяет использовать каскадные операции при изменении фрагментов и предоставляет дополнительные возможности оптимизатору запросов. Описанная функциональность была разработана и внедрена в СУБД Oracle.

Задача автоматизации выбора таблицы-измерения (dimension table) для проведения ссылочного фрагментирования была решена в [23].

Каким образом следует производить горизонтальное фрагментирование в распределенной СУБД для достижения высокой степени параллелизма представлено в [179]. В работе предложена концепция локальной достаточности — способности узла выполнить запрос без обращения по сети к другим узлам. Под нее бы-

ло подведено теоретическое обоснование в виде графа фрагментирования, строящегося на основании связей «первичный-внешний» ключ. В результате была разработана процедура фрагментирования, использующая информацию о модели данных и не зависящая от стоимостной функции.

Работа [161] предлагает методологию для проведения фрагментирования и размещения данных с возможностью репликации. Авторами была разработана процедура, где каждый шаг посвящен отдельному вопросу, связанному с необходимостью выбора структур физического уровня. Например, после первичного размещения данных, предлагается обратить внимание на транзакции, которые должны иметь малое время ответа. Также были предложены семь параметров, на основании которых выводится рекомендации по вопросам фрагментирования и размещения. Работа подкреплена рассмотрением конкретного ситуационного примера (case study). Можно отметить и другие работы, предлагавшие методологии фрагментирования и размещения данных [21, 35, 121, 162].

В работе [156] разбирается вопрос фрагментирования и размещения данных для OLAP систем, работающих со схемой «звезда» [129]. Предлагается механизм горизонтального иерархического фрагментирования с учетом выбора битовых (Bitmap) индексов. Эксперименты осуществлялись с использованием тестовго набора APB-1 [8].

Вопрос о фрагментировании и размещении рассматривался и с позиций практики. Полной декластеризацией (full declustering) [116] называется специальный метод горизонтального фрагментирования и размещения отношения. Она состоит в следующем: сначала отношение горизонтально фрагментируется на количество частей, равное числу узлов в системе, а затем, они размещаются так, чтобы на одном узле находился один фрагмент. Частичная декластеризация (partial declustering) выполняется аналогично, за исключением того, что рассматривается строгое подмножество всех узлов системы.

Сравнение обоих методов декластеризации проводятся в работах [54, 116, 139, 182]. Это в основном экспериментальные исследования — изучается либо работающая система, либо модель на симуляторе. Кроме сравнения методов, в этих работах изучался и вопрос о выборе степени декластеризации в различных условиях, например, в случае неравномерного распределения нагрузки по уз-

лам. Работа [116] содержит очень детальный анализ подобных вопросов, а также обширную библиографию на тему использования частичной декластеризации. В [139] изучалось влияние четырех стратегий балансировки нагрузки в случае частичной или полной декластеризации. Декластеризация набора отношений и метод выбора ключа декластеризации рассматривается в [182]. Более подробно этот вопрос разбирается в диссертации [181].

Рассмотрим теперь современные работы, которые на практике реализуют представленные выше модели и методы. В настоящее время широко применяется «what-if» подход [29] — метод оценки конфигурации без затрат на ее создание и выполнение над ней запросов. Вместо этого, привлекательность конфигурации (например время исполнения запросов или объем структур данных) оценивается исходя из статистической информации и метаданных. Данная функция впервые [135] появилась в [70], а ныне она встроена в практически все современные СУБД.

Можно утверждать, что большинство современных работ [6, 7, 11, 13, 38, 55, 58, 59, 124] следуют единому процессу решения задачи о выборе структур физического уровня [29].

1. Поиск структур-кандидатов. На этом этапе из набора запросов выделяются потенциально выгодные структуры;

2. Различные процедуры расширения списка кандидатов, такие как смешивание различных составляющих этих структур;

3. Некоторая процедура отбора. Это, зачастую, итеративный процесс, конструирующий решение снизу вверх путем добавления одной структуры на каждом шаге и использующий «what-if» метод.

Рассмотрим подробно отдельные работы, в которых исследуется вопрос фрагментирования или размещения.

Исследование [13] решает задачу фрагментирования отношений в условиях распределенной СУБД без совместного использования ресурсов (shared-nothing). Кроме фрагментов, дополнительно учитывались и материализованные представления. Для решения этой задачи авторы предлагают интегрировать модуль поиска фрагментов и оптимизатор запросов. Предыдущие работы [13]:

1. Не встраивали поиск альтернатив внутрь оптимизатора;

2. Часто использовали свои стоимостные модели, не связанные с моделями оптимизатора, что может привести к выбору неоптимального решения и, как следствие, худшей производительности.

В статье [13] было предложено расширить функционал оптимизатора двумя режимами. В первом из них, называемом «рекомендация», оптимизатор накапливает потенциальные решения задачи фрагментирования для каждого запроса в отдельности. Затем, для каждого из них находит оптимальный план и варианты фрагментирования. В режиме «вычисление» оптимизатор подставляет найденную на предыдущем шаге конфигурацию и использует ее при оптимизации запросов. Заметим, что результатом работы в этих двух режимах является сгенерированная структура (план выполнения запроса), никакие дополнительные дорогостоящие действия не предпринимаются, выполнения запроса не происходит. Предложенный подход был экспериментально оценен с использованием эталонного набора запросов TPC-H [165]. Практическим результатом данной работы являлось средство вычисления подходящего набора фрагментов, внедренное в коммерческую распределенную СУБД IBM DB2.

Работа [59] является продолжением предыдущей. В ней рассматривается вопрос построения системы, которая бы учитывала несколько различных типов физических структур. Кроме фрагментирования предлагается учитывать индексы, материализованные представления и структуры многомерной кластеризации. В деталях про каждый отдельный компонент рассказывается в [13, 106, 143].

Авторами рассматривается вопрос (уже освещавшийся в разделе 3.2) о выборе подхода к реализации многокомпонентных рекомендательных систем. С позиций реализации обсуждаются следующие альтернативы:

- Итеративный (iterative) подход — решающий задачу нахождения структур каждого типа последовательно, для каждой компоненты в отдельности. Отдельные алгоритмы рассматриваются как «черный ящик»;
- Интегрированный (комбинированный) (integrated) подход, при котором ищется решение для компонентов всех типов одновременно, внутри одного алгоритма.

Однако эти подходы имеют существенные недостатки [59]. Первый метод не учитывает зависимости между выбором различных компонент. Например, индексы и материализованные представления зависят друг от друга. Наличие материализованного представления может уменьшить выгоду от какого-либо индекса и наоборот. Второй подход этого недостатка лишен, однако он не обладает расширяемостью — процесс добавления компонента нового типа достаточно сложен.

В результате, в качестве альтернативы был предложен гибридный метод. Он состоит в оценивании степени влияния компонент одного типа на другой с целью выявления зависимостей и благоприятного порядка обработки. Предложенный алгоритм по природе итеративен, однако выбирать все компоненты он может несколько раз, в цикле.

Работа [124] развивает идею глубокой интеграции рекомендательной подсистемы с оптимизатором. Для обоснования своего подхода авторами приводится аргументация против распространенной практики использования «what-if» методов.

1. Слабая масштабируемость данного подхода. При увеличении числа таблиц, число возможных конфигураций растет экспоненциально и увеличивает количество необходимых вызовов оптимизатора;

2. Неэффективность. Каждый вызов оптимизатора тратит много ресурсов на подготовку к работе. Дополнительные затраты — синтаксический анализ запроса, проверка корректности, проверка ограничений безопасности и вообще любая сопутствующая деятельность, не связанная с фрагментированием. В итоге, на работу с конфигурациями будет тратиться лишь малая доля общего времени.

Для решения этих проблем авторы предлагают встраивание компонента рекомендации фрагментов в оптимизатор и параллелизацию поиска оптимальной конфигурации.

Данная работа посвящена вопросам реализации такой системы в продукте Microsoft SQL Server 2008 Parallel Data Warehouse, распределенной СУБД, предназначенной для выполнения OLAP запросов. Задача сформулирована в классической постановке с учетом репликации. Авторы сравнивали метод из [13], генетиче-

ский алгоритм и свой подход, основанный на методе ветвей и границ. Эксперименты проводились на стандартных тестовых наборах [165].

В работе [133] также рассматривается вопрос ускорения поиска оптимальной конфигурации за счет снижения нагрузки на оптимизатор запросов. Для этого предлагается частично кэшировать планы запросов, отделив операторы непосредственного доступа к данным. Несмотря на то, что основной целью работы была автоматизация выбора индексов, эти наработки могут быть применены и для выбора фрагментов [133].

Средство выбора индексов и фрагментов для СУБД PostgreSQL представлено в [11,135]. Компонент рекомендации фрагментов взят из средства AutoPart [131], которое будет рассмотрено в следующем разделе.

В качестве промежуточного итога можно сказать, что область горизонтального фрагментирования зрелая, насчитывает более сорока лет исследований. Как уже отмечалось во введении, в настоящее время практически все современные СУБД обладают или работают над функционалом рекомендации структур физического уровня, и в частности фрагментирования [12, 159](Oracle), [13, 58, 59, 143] (IBM DB2), [7, 29, 55, 56] (Microsoft SQL Server), [124] (Microsoft SQL Server 2008 Parallel Data Warehouse), [11, 76, 135] (PostgreSQL), [163] (Ingres). Несмотря на это область является активным полем исследований и ее развитие продолжается [14].

1. Повышение степени интеграции рекомендательной системы и оптимизатора запросов. Сюда относятся новые методы поиска решений [29] — использование параллелизма при поиске конфигураций [124], кеширование планов при поиске [133] и др;

2. Практические аспекты реализации методов фрагментирования (разработка и реализация новых схем фрагментирования, языковая поддержка этих средств) [159];

3. Виртуальное фрагментирование [134] (фрагментирование в условиях распределенных СУБД, построенных с использованием связующего программного обеспечения — middleware);

4. Разработка моделей, более точно описывающих поведение оп-

тимизатора и предметную область, с объектами которой работает СУБД [76, 131];

5. Вопросы сравнения и оценки систем рекомендации структур физического уровня [28, 80].

4. Вертикальное фрагментирование. Вертикальное фрагментирование или кластеризация атрибутов (attribute clustering) [83]. Эта процедура состоит в разбиении исходного отношения на фрагменты, каждый из которых содержит подмножество атрибутов исходного отношения и первичные ключи [128]. Ключи необходимы для выполнения требования о восстановимости исходного отношения, о котором мы уже писали ранее. В работе [172] говорится о двух возможных способах удовлетворения данного требования — использование первичных ключей исходного отношения, либо использование внутреннего идентификатора записи.

Вертикальное фрагментирование также является инструментом повышения производительности базы данных. Однако в отличии от горизонтального фрагментирования, администратор базы данных менее свободен в выборе способа вертикального фрагментирования, особенно, если речь касается базы данных, существующей длительное время. При проектировании с нуля также появляются сложности — кроме вопросов фрагментирования необходимо учитывать внешние требования к схеме данных, например, требование нормализации отношений.

В [131] отмечается, что хотя вертикальное фрагментирование несет выгоду, потенциально оно может быть затратным ввиду необходимости проведения дополнительных операций соединения, в случае если запросы в наборе плохо «отделимы» друг от друга.

Вертикальное фрагментирование бывает двух типов — с перекрытием атрибутов или без него. В первом случае один и тот же атрибут может входить в несколько отношений, то есть первый вариант подразумевает репликацию поля (field replication) [167]. Данный случай редко встречается в литературе, это отчасти обусловлено трудностями, возникающими при работе с обновлениями. Используя такой подход, будет трудно поддерживать все фрагменты в актуальном состоянии. Другая причина — фазу обработки репликации часто рассматривают отдельно и добавляют к существующему алгоритму. Тем не менее, использование репликации поля может быть очень важно [167]. В этом исследовании, на конкрет-

ном примере, показано, что даже в условиях наличия обновлений можно добиться увеличения производительности в разы. Конечным результатом данной работы была схема описания структур физического уровня. Кроме того, был предложен механизм ее интеграции с исполнителем запросов. К сожалению, метод автоматического поиска выгодных для репликации полей не был рассмотрен, так же как и в большинстве работ, перечисленных ниже.

Кроме того, репликация поля может быть целесообразна, если какой-либо атрибут сложно однозначно отнести к одному из полученных фрагментов [42].

Работы [110, 146] предлагают делить алгоритмы вертикального фрагментирования на две категории.

1. Подход на основе близости атрибутов (attribute affinity). В работах данной группы фрагменты выделяются процедурно, в зависимости от количества совместных появлений каждой пары атрибутов в наборе запросов. В [71, 146] приводится серьезный довод против данного подхода — невозможность учета близости группы из нескольких атрибутов. Также возможна необоснованная группировка [65]. В предлагаемой в [64] классификации отмечается, что в работах данной группы используется эмпирическая функция в противовес стоимостной. Эти функции (отсюда и название) основываются на эмпирических предположениях о том, какое фрагментирование наиболее выгодно. Они менее точны, чем стоимостные, зато могут быть полезны, когда нет подробной информации о физических параметрах;

2. Стоимостной подход, также называемый оптимизационным. В этом случае анализируется характер доступа к данным, строится модель системы, ищется и минимизируется функция стоимости исполнения запросов. Мы уже рассматривали общую схему данного подхода в разделе 2.2.

Разберем некоторые работы и начнем с алгоритмов фрагментирования на основе близости атрибутов [40–43, 65, 67, 92, 108, 122, 123, 123, 172].

Первые работы данного типа появились в 70-80х годах прошлого века [92, 123, 172]. В этих работах было введено понятие матрицы близости атрибутов (attribute affinity matrix), симметричной квад-

ратной матрицы, показывающей степень связанности пары атрибутов. Она получается из матрицы использования атрибутов (attribute usage matrix) с помощью преобразования, учитывающего частоты доступа к атрибуту в рамках транзакции и частоту появления этой транзакции в общем наборе. Матрицу использования атрибутов получают, ставя на позицию (i, j) единицу, если транзакция j использует атрибут с номером i, и ноль в противном случае. Подобные методы представления и обработки характеристик запросов используются и в современных работах. В том числе и в работах стоимостного подхода [7, 18, 41, 42, 67, 71, 84, 108, 111, 180].

Одной из самых первых работ была [92]. Авторы рассматривали задачу в классической постановке — найти набор вертикальных фрагментов по известным запросам и характеристикам атрибутов. Строилась матрица близости атрибутов, к которой затем применялся BEA (bond energy algorithm) —алгоритм перестановки строк и столбцов, максимизирующий функцию связанности соседствующих атрибутов.

Работы [40, 41, 43] развивают идеи [92] — в данных исследованиях также рассматривается вопрос кластеризации матрицы близости атрибутов. Для этого используется метод ветвей и границ, а также применяются алгоритмы нахождения кластеров в матрицах (Cluster Identification).

В [172] рассматривается вопрос о вертикальном фрагментировании отношения в набор возможно пересекающихся фрагментов (то есть с репликацией) в условиях:

1. Одного узла с одним уровнем иерархии памяти;
2. Одного узла с несколькими уровнями иерархии памяти;
3. Нескольких узлов с одним уровнем иерархии памяти (случай распределенной СУБД) в вариантах с репликацией и без.

Второй случай подразумевает, что на узле присутствует несколько уровней памяти, имеющих различные технические характеристики и стоимость. Такая формулировка может быть важна, если один набор атрибутов затрагивается транзакциями чаще другого. Тогда можно увеличить производительность системы, поместив чаще употребляющийся набор в быструю память.

Решение перечисленных задач осуществляется в два этапа. Сначала производится кластеризация матрицы близости атрибутов,

а затем применяются специфические для каждого случая алгоритмы. Под кластеризацией в данных работах понималась перестановка строк и столбцов матрицы для приведения ее в блочно-диагональную форму. Упомянутые же алгоритмы с помощью стоимостных моделей, учитывающих, в том числе и размер атрибута в байтах, генерируют искомые разбиения. То есть первая фаза обоснована только интуицией и в ней идея стоимости не используется [180]. Таким образом, данную работу можно отнести как к первому типу, так и ко второму.

Необходимо заметить, что предложенное решение относится к бинарному фрагментированию (binary partitioning). Данный тип алгоритмов всегда разбивает отношение на два фрагмента. Соответственно, для получения большего количества фрагментов его приходится применять несколько раз, что и проделывается в данной работе.

В [42] задача поиска набора вертикальных фрагментов также формулируется в виде задачи кластеризации атрибутов на основе их попарной близости. Она сводится к задаче о бродячем торговце, которую решают с помощью генетических алгоритмов. Затем найденый путь разрезается на кластеры с использованием функции расстояния, предложенной авторами.

Работы [42, 180] — попытки применить подобные методы для выработки горизонтального фрагментирования. В указанных работах матрица близости строится для предикатов входящих в запросы.

В работе [123] для решения задачи вертикального фрагментирования применяется несколько иной подход. Полученная матрица близости атрибутов трактуется как матрица смежности, задающая граф связей между атрибутами. Вершинами этого неориентированного графа являются атрибуты, а вес ребер показывает степень близости. В таком представлении постановка задачи переформулируется: необходимо найти набор кластеров атрибутов, таких, что каждый атрибут участвует только в одном кластере и вершины одного кластера «сильно связаны» друг с другом. «Сильно связаны» означает, что веса ребер между атрибутами, входящими в кластер, больше, чем веса ребер между ними и атрибутами, не входящими в кластер. Задача решается с помощью обхода графа и специальной процедуры добавления атрибута в кластер. Работа содержит описание и доказательство корректности алгоритма по-

строения кластера атрибутов с заданными свойствами. Также была найдена вычислительная сложность алгоритма, которая оказалась ниже, чем у других известных в то время алгоритмов. Однако качество полученных разбиений в применении к набору запросов в данной работе не измеряется, практические эксперименты полностью отсутствуют.

Данная идея получила свое развитие в работах [108, 122], так же подобный метод был применен и для генерации горизонтальных фрагментов в объектных СУБД [15], и для размещения данных в OLAP системах [18].

Упростить алгоритм из [123] предлагалось в [67]. Для этого были введены дополнительные параметры, управляющие ходом формирования групп. Тесты показали хорошие результаты, однако формальное доказательство корректности разработанного алгоритма представлено не было.

Графовый подход применяют и в [153]. В этой работе на основании матрицы близости атрибутов строится нечеткий граф (fuzzy graph), и решается задача его разрезания. К достоинствам данного подхода можно отнести возможность проведения N-арного фрагментирования, то есть возможность заранее задать количество необходимых фрагментов. Однако в этой работе задача рассматривается без учета репликации.

Наконец, в [65] приводится обстоятельная критика подхода, использующего близость атрибутов. Например, приводятся примеры необоснованной группировки, которую производят алгоритмы данного класса. Рассматривается сложный случай, когда несколько атрибутов не обладают достаточным значением близости по отношению к каким-либо другим. Для решения вышеописанных трудностей, а также некоторых других, авторы предлагают новую меру близости и новый алгоритм.

В диссертации [110] подробно разбираются некоторые из вышеупомянутых работ, а также приводится еще несколько современных исследований данного класса.

Рассмотрим теперь стоимостной подход. К нему можно отнести целый ряд работ [7, 45, 50, 60, 64, 71, 86, 91, 95, 96, 111, 126, 130, 146, 154, 170, 171, 173].

Судя по всему, одними из первых работ такого рода были [86] и [91]. В [86] задача поиска фрагментов решалась с помощью стоимостной функции и метода поиска с восхождением к вершине

(hill-climbing). Была предложена эвристика, уменьшившая алгоритмическую сложность приближенного решения с кубической до квадратичной. Согласно [67], в [91] была создана модель, которая учитывала стоимости хранения, поиска и обновления. Требовалось минимизировать их линейную комбинацию. Для решения поставленной задачи применялось целочисленное программирование.

Исследование [64] приводит [45] и [50] как примеры работ, в которых требование минимизации доступа к диску диктует необходимость использования стоимостного бинарного вертикального фрагментирования. Предлагаемый подход базируется на идее использования транзакций в противовес [64] доминировавшему в то время подходу учета отдельных атрибутов. Работа [50] использует методы целочисленного программирования для выработки необходимого набора фрагментов (осуществляется бинарное фрагментирование). Предложена весьма детальная модель, учитывающая селективности предикатов, различные типы доступа к данным на диске и наличие операций соединения в запросах. В [45] предлагается теория оптимального разрезания отношений, на основе которой были предложены оптимальный и приближенный алгоритмы фрагментирования. Самое главное в этой работе — оценка сложности этого алгоритма зависит не от количества атрибутов, как в большинстве других работ, но от количества транзакций. Это достигается за счёт того, что за единицу распределения по фрагментам берут не атрибут, а множество атрибутов которые затрагивает транзакция. Предложенный алгоритм может быть чрезвычайно полезен в случае, когда транзакций меньше чем атрибутов. Кроме того, количество рассматриваемых транзакций можно еще уменьшить, воспользовавшись правилом 80/20 [45].

Однако в [64] утверждается, что описываемые подходы выработаны для случая одного узла и не применимы в распределенном случае. В [104] предлагается расширение [45] на случай горизонтального фрагментирования и реплицирования. В качестве атрибутов выступают предикаты.

Аналогично [45] зависел от количества транзакций и подход [130]. В этой работе были предложены две характеристики фрагмента — достаточность (sufficient) и поддержка (support). Фрагмент обладает достаточностью, если содержит все атрибуты, необходимые для выполнения транзакции. Поддержка группы транзакций означает, что транзакции достаточны на данном фрагменте и

фрагмент не обладает лишними атрибутами. Основываясь на этих характеристиках, была построена стоимостная функция, которую максимизируют с помощью предложенного алгоритма.

В [71,126] строится функция для оценки конфигураций, задающих вертикальное фрагментирование (Partition Evaluator). Функция состоит из двух компонентов — вклада от успешного использования фрагментирования (все необходимые для выполнения запроса данные содержатся во фрагменте) и стоимости вынужденного обращения к другим фрагментам. Предполагается, что при увеличении количества фрагментов стоимость первого компонента будет падать, а второго — расти.

В работах [146] и [154] рассматривается случай вертикального фрагментирования в распределенной СУБД. В [154] утверждается, что не всегда целесообразно искать наиболее подходящую с точки зрения стоимости конфигурацию (best fit vertical partitioning), но иногда необходимо ограничить количество фрагментов (n-way vertical partitioning). В качестве иллюстрирующего примера авторы предлагают случай распределенной СУБД с K узлами. Утверждается, что имеет смысл выбрать ровно K фрагментов для обеспечения равной загрузки узлов. Бинарное фрагментирование из [172] не подходит для этой цели, так как не задает правила выбора лучшего набора из нескольких возможных. В качестве решения этой задачи авторы предложили процедуру построения решения снизу вверх, где в качестве исходных фрагментов рассматриваются отдельные колонки. Предложенная стоимостная функция учитывает как положительный вклад от совместного хранения колонок, так и отрицательное влияние, вызванное конкуренцией запросов за доступ к фрагменту.

Работа [146] расширяет [154], включая в рассмотрение размер атрибутов в байтах, с целью повышения производительности распределенных мультимедиа СУБД. В такого рода системах встречаются как атрибуты привычных типов, имеющие обычный размер, так и атрибуты, являющиеся двоичными данными, например изображения или видео. В настоящее время в распределенных СУБД передача данных по сети является самым узким местом. Поэтому вопрос минимизации объема передаваемых данных становится актуальным для задачи фрагментирования в данной постановке.

Подобный подход был и в [111]. В работе была представлена стоимостная модель для решения задачи вертикального фрагмен-

тирования и размещения в распределенной СУБД. Она была аналогична [146], также учитывала размер атрибута, однако был использован иной метод решения. В [146] решение строилось снизу вверх, подсчитывая на каждом шаге выгоду от хранения пары атрибутов. Используется идея о стоимости как о сумме доступов двух типов [71, 126]. В [111] атрибуты последовательно распределялись на узлы с минимальной стоимостью размещения атрибута.

В обоих работах [111, 146] не рассматривалась репликация, и не было ограничения на объем дискового пространства узла.

Генетические алгоритмы применялись для нахождения вертикальных фрагментов [10, 42, 64, 155]. Необходимо заметить, что в большинстве работ данного рода проводятся экспериментальные сравнения с другими генетическими алгоритмами. Также проводятся сравнения с решением, полученным полным перебором или случайным поиском. Автору настоящей работы неизвестны эксперименты, в которых было бы произведено сравнение качества разбиений, построенных при помощи генетических алгоритмов, с качеством разбиений, полученных с использованием других стоимостных подходов. Исключением является [155], где использовалась модель из [50].

Дополнительно к минусам этого рода работ можно отнести и тот факт, что существенная часть экспериментов с генетическими алгоритмами использовала эталонную нагрузку, состоящую из матрицы в 20 атрибутов и 15 транзакций, впервые упомянутую в [172] или подобную. За время, прошедшее с 1984 года, размеры СУБД выросли, появилось деление нагрузок по их характеристикам [72]. В настоящее время существуют стандартные эталонные нагрузки, более соответствующие современным задачам [8, 28, 80, 164, 165].

Методы интеллектуального анализа данных (Data Mining) также применялись для решения этой задачи [26, 82, 84, 145]. В [84] для уменьшения пространства поиска сначала применялись алгоритмы кластеризации атрибутов на основании частот появления в запросах, а затем среди кандидатов выбирался наилучший на основании минимальной стоимости. Поиск фрагментов с использованием ассоциативных правил применялся в [82, 145].

Рассмотрим теперь современные работы. Вопрос интеграции вертикальных и горизонтальных фрагментов, а также индексов разбирается в [7, 55]. В алгоритме, представленном в данных ра-

ботах, нам важен аспект совместного использования вертикального и горизонтального фрагментирования. Эта работа полностью описывается общей схемой стоимостных подходов. Среди интересных моментов можно отметить шаг слияния. В нем множество отобранных структур-кандидатов расширяется за счет добавления структур, полученных в результате смешивания. Этот шаг позволяет учесть структуры, которые могут положительно повлиять на скорость исполнения сразу нескольких запросов. В некотором роде этот шаг аналогичен применению оператора скрещивания (crossover), использующегося в генетических алгоритмах, приведенных выше.

В данной работе авторы хотели показать, что необходим интегрированный подход для выбора структур физического уровня. Для этого был предложен набор алгоритмов для их оценки и эффективной генерации. Разработанные алгоритмы были внедрены в СУБД Microsoft SQL Server, а эксперименты были проведены с использованием современной эталонной нагрузки TPC-H [165]. Результаты показали состоятельность выбранного подхода и полностью подтвердили высказанные гипотезы.

Гибридное фрагментирование с учетом реплицирования атрибута изучается в [131]. Предложенный метод решает задачу эффективной обработки научных данных большого объема, где все данные представлены одной таблицей.

Вопросами вертикального фрагментирования и его глубокой интеграции с исполнением запросов занимались в [48, 138, 142]. В них рассматривается DSM (decomposition storage model), — модель представления данных на диске, в которой каждый атрибут храниться отдельно от остальных. В рамках данной модели возможен альтернативный подход к выполнению запросов в СУБД.

Кроме того, в последнее время произошел всплеск интереса к вертикальному фрагментированию отношений в оперативной памяти [47, 87, 96, 178]. В этих работах изучались способы повышения скорости обработки данных за счет уменьшение простоя процессора. Для этого предлагались различные способы формирования страниц данных, позволяющие эффективно использовать кеш-память процессора. Были созданы модели для формирования наборов атрибутов хранящихся совместно. Последние две группы работ считаются [4] предшественниками современных колоночных СУБД.

5. Заключение. В настоящей работе мы рассмотрели подходы к организации физического уровня в СУБД с хранением по строкам, а именно:

1. Горизонтальное фрагментирование:
   - горизонтальное фрагментирование как инструмент администратора базы данных: классифицированы и описаны три типа способов горизонтального фрагментирования, представлены методологии проведения фрагментирования, были приведены примеры фрагментирования на конкретных случаях (case study);
   - представлены модели для проведения горизонтального фрагментирования, как основного, так и производного, рассмотрено фрагментирование в OLAP системах;
   - вкратце освещен вопрос использования кластеризации записей для горизонтального фрагментирования.
2. Вертикальное фрагментирование:
   - модели, использующие близость атрибутов для проведения вертикального фрагментирования;
   - стоимостные модели для проведения вертикального фрагментирования;
   - вкратце рассмотрены современные подходы к фрагментированию данных в оперативной памяти.
3. Различные вопросы размещения данных в распределенной СУБД:
   - рассмотрены различные типы моделей направленных на достижение: высокого уровня надежности, высокой производительности в условиях выполнения распределенных транзакций, высокой производительности на нагрузках, не требующих выполнения транзакций;
   - освещен взгляд с практической стороны — рассмотрены стратегии фрагментирования и размещения данных на узлах (стратегии полной и частичной декластеризации);
   - вкратце затронуты вопросы репликации и миграции данных.

Кроме этого, нами были рассмотрены системы рекомендации структур физического уровня, работающие с вышеуказанными компонентами в современных коммерческих СУБД. Был перечислен ряд наиболее активных в настоящее время направлений исследований.

6. Благодарности. Автор данной работы хотел бы выразить благодарность за помощь в подготовке текста Новикову Б.А., Смирнову К.К. и Федотовскому П.В.

Список литературы

[1] Д.П. . Требования к системе реструктурирования базы данных // Тр. СПИИРАН. 2004. Т. 2, № 1. С. 312–315. URL: http://mi.mathnet.ru/trspy164.

[2] В.Б. . Метод автоматизации проектирования распределенной реляционной базы данных // Программные продукты и системы. 2008. № 3. С. 45–48. URL: http://swsys.ru/index.php?page=article&id=1578.

[3] Е.И. . ПРОЕКТИРОВАНИЕ РАСПРЕДЕЛЕННОЙ БАЗЫ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ГЕНЕТИЧЕСКОГО АЛГОРИТМА // «Труды интернет-конференции «Актуальные проблемы аппаратно-программного и информационного обеспечения науки, образования, культуры и бизнеса». МГУПИ, 2010. URL: http://it4-mgupi.ru/files/internet/2010/Cybankov.pdf.

[4] 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.

[5] Adaptive record clustering / C. T. Yu, Cheing-mei Suen, K. Lam, M. K. Siu // ACM Trans. Database Syst. 1985. Vol. 10. P. 180–204. URL: http://doi.acm.org/10.1145/3857.3861.

[6] Agrawal S., Chu E., Narasayya V. Automatic physical design tuning: workload as a sequence // Proceedings of the 2006 ACM SIGMOD international conference on Management of data. SIGMOD '06. New York, NY, USA : ACM, 2006. P. 683–694. URL: http://doi.acm.org/10.1145/1142473.1142549.

[7] Agrawal S., Narasayya V., Yang B. Integrating vertical and horizontal partitioning into automated physical database design // Proceedings of the 2004 ACM SIGMOD international conference on Management of data. SIGMOD '04. New York, NY, USA : ACM, 2004. P. 359–370. URL: http://doi.acm.org/10.1145/1007568.1007609.

[8] APB-1 OLAP Benchmark, Release II. OLAP Council, Nov. 1998. www.olapcouncil.org/research/bmarkly.htm. Дата просмотра: 30/07/2012.

[9] Apers P. M. G. Data allocation in distributed database systems // ACM Trans. Database Syst. 1988. Vol. 13. P. 263–304. URL: http://doi.acm.org/10.1145/44498.45063.

[10] Applying genetic algorithms in database partitioning / Vincent Ng, Dik Man Law, Narasimhaiah Gorla, Chi Kong Chan // Proceedings of the 2003 ACM symposium on Applied computing. SAC '03. New York, NY, USA : ACM, 2003. P. 544–549. URL: http://doi.acm.org/10.1145/952532.952639.

[11] An automated, yet interactive and portable DB designer / Ioannis Alagiannis, Debabrata Dash, Karl Schnaitter et al. // Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. SIGMOD '10. New York, NY, USA : ACM, 2010. P. 1183–1186. URL: http://doi.acm.org/10.1145/1807167.1807314.

[12] 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.

[13] Automating physical database design in a parallel database / Jun Rao, Chun Zhang, Nimrod Megiddo, Guy Lohman // Proceedings of the 2002 ACM SIGMOD international conference on Management of data. SIGMOD '02. New York, NY, USA : ACM, 2002. P. 558–569. URL: http://doi.acm.org/10.1145/564691.564757.

[14] Badia A., Lemire D. A call to arms: revisiting database design // SIGMOD Rec. 2011. Vol. 40, no. 3. P. 61–69. URL: http://doi.acm.org/10.1145/2070736.2070750.

[15] Baião F., Mattoso M., Zaverucha G. A distribution design methodology for object DBMS // Distributed and Parallel Databases. 2004. Vol. 16. P. 45–90. 10.1023/B:DAPD.0000026268.04288.b9. URL: http://dx.doi.org/10.1023/B:DAPD.0000026268.04288.b9.

[16] Bell D. A. Difficult data placement problems // The Computer Journal. 1984. Vol. 27, no. 4. P. 315–320. http://comjnl.oxfordjournals.org/content/27/4/315.full.pdf+html.

[17] Bell D. A. Physical record clustering in databases // Kybernetes. 1984. Vol. 13. P. 31–37.

[18] 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.

[19] Bellatreche L., Boukhalfa K., Abdalla H. I. Saga: A combination of genetic and simulated annealing algorithms for physical data warehouse design // BNCOD. 2006. P. 212–219.

[20] Bellatreche L., Boukhalfa K., Richard P. Data partitioning in data warehouses: Hardness study, heuristics and ORACLE validation // Data Warehousing and Knowledge Discovery / Ed. by Il-Yeol Song, Johann Eder, Tho Nguyen. Springer Berlin / Heidelberg, 2008. Vol. 5182 of Lecture

Notes in Computer Science. P. 87–96. 10.1007/978-3-540-85836-2_9. URL: http://dx.doi.org/10.1007/978-3-540-85836-2_9.

[21] Bellatreche L., Cuzzocrea A., Benkrid S. F&A: a methodology for effectively and efficiently designing parallel relational data warehouses on heterogenous database clusters // Proceedings of the 12th international conference on Data warehousing and knowledge discovery. DaWaK'10. Berlin, Heidelberg : Springer-Verlag, 2010. P. 89–104. URL: http://dl.acm.org/citation.cfm?id=1881923.1881934.

[22] Bellatreche L., Karlapalem K., Simonet A. Algorithms and support for horizontal class partitioning in object-oriented databases // Distributed and Parallel Databases. 2000. Vol. 8. P. 155–179. 10.1023/A:1008745624048. URL: http://dx.doi.org/10.1023/A:1008745624048.

[23] Bellatreche L., Woameno K. Y. Dimension table driven approach to referential partition relational data warehouses // Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP. DOLAP '09. New York, NY, USA : ACM, 2009. P. 9–16. URL: http://doi.acm.org/10.1145/1651291.1651294.

[24] Bernardino J., Madeira H. Experimental evaluation of a new distributed partitioning technique for data warehouses // Proceedings of the International Database Engineering & Applications Symposium. IDEAS '01. Washington, DC, USA : IEEE Computer Society, 2001. P. 312–321. URL: http://dl.acm.org/citation.cfm?id=646290.687056.

[25] Blankinship R., Hevner A. R., Yao S. B. An iterative method for distributed database design // Proceedings of the 17th International Conference on Very Large Data Bases. VLDB '91. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1991. P. 389–400. URL: http://dl.acm.org/citation.cfm?id=645917.672331.

[26] Bouakkaz M., Ouinten Y., Ziani B. Vertical fragmentation of data warehouses using the FP-Max algorithm // Innovations in Information Technology (IIT), 2012 International Conference on. 2012. march. P. 273–276.

[27] Boukhalfa K., Bellatreche L., Alimazighi Z. HP&BJI: A combined selection of data partitioning and join // New Trends in Data Warehousing and Data Analysis / Ed. by Stanislaw Kozielski, Robert Wrembel. Springer US, 2009. Vol. 3 of Annals of Information Systems. P. 1–23. 10.1007/978-0-387-87431-9_10. URL: http://dx.doi.org/10.1007/978-0-387-87431-9_10.

[28] Bruno N. A critical look at the tab benchmark for physical design tools // SIGMOD Rec. 2007. Vol. 36, no. 4. P. 7–12. URL: http://doi.acm.org/10.1145/1361348.1361349.

[29] Bruno N., Chaudhuri S. Automatic physical database tuning: a relaxation-based approach // Proceedings of the 2005 ACM SIGMOD international conference on Management of data. SIGMOD '05. New York, NY, USA : ACM, 2005. P. 227–238. URL: http://doi.acm.org/10.1145/1066157.1066184.

[30] Brunstrom A., Leutenegger S. T., Simha R. Experimental evaluation of dynamic data allocation strategies in a distributed database with changing workloads // Proceedings of the fourth international conference on Information and knowledge management. CIKM '95. New York, NY, USA :

ACM, 1995. P. 395–402. URL: http://doi.acm.org/10.1145/221270.221652.

[31] Casey R. G. Allocation of copies of a file in an information network // Proceedings of the May 16-18, 1972, spring joint computer conference. AFIPS '72 (Spring). New York, NY, USA : ACM, 1972. P. 617–625. URL: http://doi.acm.org/10.1145/1478873.1478955.

[32] Ceri S., Martella G., Pelagatti G. Optimal file allocation in a computer network: a solution method based on the knapsack problem // Computer Networks (1976). 1982. Vol. 6, no. 5. P. 345–357. URL: http://www.sciencedirect.com/science/article/pii/0376507582901040.

[33] Ceri S., Navathe S., Wiederhold G. Distribution design of logical database schemas // IEEE Transactions on Software Engineering. 1983. Vol. 9. P. 487–504.

[34] Ceri S., Negri M., Pelagatti G. Horizontal data partitioning in database design // Proceedings of the 1982 ACM SIGMOD international conference on Management of data. SIGMOD '82. New York, NY, USA : ACM, 1982. P. 128–136. URL: http://doi.acm.org/10.1145/582353.582376.

[35] Ceri S., Pernici B., Wiederhold G. Distributed database design methodologies // Proceedings of the IEEE. 1987. Vol. 75, no. 5. P. 533–546.

[36] Chang S.-K., Liu A.-C. File allocation in a distributed database // International Journal of Parallel Programming. 1982. Vol. 11. P. 325–340. 10.1007/BF01001955. URL: http://dx.doi.org/10.1007/BF01001955.

[37] Chaturvedi A., Choubey A., Roan J. Scheduling the allocation of data fragments in a distributed database environment: a machine learning approach // Engineering Management, IEEE Transactions on. 1994. Vol. 41, no. 2. P. 194–207.

[38] Chaudhuri S., Narasayya V. Microsoft index turning wizard for SQL Server 7.0 // SIGMOD Rec. 1998. Vol. 27, no. 2. P. 553–554. URL: http://doi.acm.org/10.1145/276305.276378.

[39] Chaudhuri S., Weikum G. Self-management technology in databases // Encyclopedia of Database Systems / Ed. by Ling Liu, M.Tamer Özsu. Springer US, 2009. P. 2550–2555. URL: http://dx.doi.org/10.1007/978-0-387-39940-9_334.

[40] Cheng C. Algorithms for vertical partitioning in database physical design // Omega. 1994. Vol. 22, no. 3. P. 291–303. URL: http://www.sciencedirect.com/science/article/pii/0305048394900426.

[41] Cheng C.-H. A branch and bound clustering algorithm // Systems, Man and Cybernetics, IEEE Transactions on. 1995. Vol. 25, no. 5. P. 895–898.

[42] Cheng C.-H., Lee W.-K., Wong K.-F. A genetic algorithm-based clustering approach for database partitioning // Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on. 2002. Vol. 32, no. 3. P. 215–230.

[43] Cheng C.-H., Motwani J. An examination of cluster identification-based algorithms for vertical partitions // Int. J. Bus. Inf. Syst. 2009. Vol. 4, no. 6. P. 622–638. URL: http://dx.doi.org/10.1504/IJBIS.2009.026695.

[44] Chiu G.-M., Raghavendra C. A model for optimal database allocation in distributed computing systems // INFOCOM '90. Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. 'The Multiple Facets of Integration'. Proceedings., IEEE. Vol. 3. 1990. jun. P. 827–833.

[45] 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.

[46] Chu W. W. Optimal file allocation in a multiple computer system // IEEE Trans. Comput. 1969. Vol. 18, no. 10. P. 885–889. URL: http://dx.doi.org/10.1109/T-C.1969.222542.

[47] 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.

[48] 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.

[49] Corcoran A. L., Hale J. A genetic algorithm for fragment allocation in a distributed database system // Proceedings of the 1994 ACM symposium on Applied computing. SAC '94. New York, NY, USA : ACM, 1994. P. 247–250. URL: http://doi.acm.org/10.1145/326619.326738.

[50] 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.

[51] Cornell D. W., Yu P. S. Site assignment for relations and join operations in the distributed transaction processing environment // Proceedings of the Fourth International Conference on Data Engineering. Washington, DC, USA : IEEE Computer Society, 1988. P. 100–108. URL: http://dl.acm.org/citation.cfm?id=645473.653573.

[52] Cornell D. W., Yu P. S. On optimal site assignment for relations in the distributed database environment // IEEE Trans. Softw. Eng. 1989. Vol. 15, no. 8. P. 1004–1009. URL: http://dx.doi.org/10.1109/32.31356.

[53] Costa M., Madeira H. Handling big dimensions in distributed data warehouses using the DWS technique // Proceedings of the 7th ACM international workshop on Data warehousing and OLAP. DOLAP '04. New York, NY, USA : ACM, 2004. P. 31–37. URL: http://doi.acm.org/10.1145/1031763.1031770.

[54] Data placement in Bubba / George Copeland, William Alexander, Ellen Boughter, Tom Keller // Proceedings of the 1988 ACM SIGMOD international conference on Management of data. SIGMOD '88. New York, NY, USA : ACM, 1988. P. 99–108. URL: http://doi.acm.org/10.1145/50202.50213.

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

[56] 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.

[57] Daudpota N. H. Five steps to construct a model of data allocation for distributed database systems // J. Intell. Inf. Syst. 1998. Vol. 11, no. 2. P. 153–168. URL: http://dx.doi.org/10.1023/A:1008676718656.

[58] 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.

[59] 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.

[60] De P., Park J. S., Pirkul H. An integrated model of record segmentation and access path selection for databases // Information Systems. 1988. Vol. 13, no. 1. P. 13–30. URL: http://www.sciencedirect.com/science/article/pii/0306437988900245.

[61] Designing a distributed database on a local area network: a methodology and decision support system / H. Lee, Y.-K. Park, G. Jang, S.-Y. Huh // Information and Software Technology. 2000. Vol. 42, no. 3. P. 171–184. URL: http://www.sciencedirect.com/science/article/pii/S0950584999000567.

[62] Didriksen T., Galindo-Legaria C. A., Dahle E. Database de-centralization — a practical approach // Proceedings of the 21th International Conference on Very Large Data Bases. VLDB '95. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1995. P. 654–665. URL: http://dl.acm.org/citation.cfm?id=645921.673313.

[63] Dowdy L. W., Foster D. V. Comparative models of the file assignment problem // ACM Comput. Surv. 1982. Vol. 14, no. 2. P. 287–313. URL: http://doi.acm.org/10.1145/356876.356883.

[64] Du J., Alhajj R., Barker K. Genetic algorithms based approach to database vertical partition // J. Intell. Inf. Syst. 2006. Vol. 26. P. 167–183. URL: http://dl.acm.org/citation.cfm?id=1139987.1139989.

[65] Du J., Barker K., Alhajj R. Attraction — a global affinity measure for database vertical partitioning // ICWI. IADIS, 2003. P. 538–548.

[66] Du X., Maryanski F. Data allocation in a dynamically reconfigurable environment // Data Engineering, 1988. Proceedings. Fourth International Conference on. 1988. feb. P. 74–81.

[67] An enhanced grouping algorithm for vertical partitioning problem in DDBs / F. Marir, Y. Najjar, M.Y. AlFaress, H.I. Abdalla // Computer and information sciences, 2007. iscis 2007. 22nd international symposium on. 2007. nov. P. 1–6.

[68] Eswaran K. P. Placement of records in a file and file allocation in a computer // IFIP Congress. 1974. P. 304–307.

[69] Evolutionary algorithms for allocating data in distributed database systems / Ishfaq Ahmad, Kamalakar Karlapalem, Yu-Kwong Kwok, Siu-Kai So // Distrib. Parallel Databases. 2002. Vol. 11. P. 5–32. URL: http://dl.acm.org/citation.cfm?id=509176.509177.

[70] Finkelstein S., Schkolnick M., Tiberio P. Physical database design for relational databases // ACM Trans. Database Syst. 1988. Vol. 13. P. 91–128. URL: http://doi.acm.org/10.1145/42201.42205.

[71] A formal approach to the vertical partitioning problem in distributed database design / J. Muthuraj, S. Chakravarthy, R. Varadarajan, S. B. Navathe // Proceedings of the second international conference on Parallel and distributed information systems. PDIS '93. Los Alamitos, CA, USA : IEEE Computer Society Press, 1993. P. 26–35. URL: http://dl.acm.org/citation.cfm?id=382019.382410.

[72] 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.

[73] Frieder O., Siegelmann H. Multiprocessor document allocation: a genetic algorithm approach // Knowledge and Data Engineering, IEEE Transactions on. 1997. Vol. 9, no. 4. P. 640–642.

[74] Furtado P. Experimental evidence on partitioning in parallel data warehouses // Proceedings of the 7th ACM international workshop on Data warehousing and OLAP. DOLAP '04. New York, NY, USA : ACM, 2004. P. 23–30. URL: http://doi.acm.org/10.1145/1031763.1031769.

[75] Gavish B., Pirkul H. Computer and database location in distributed computer systems // IEEE Transactions on Computers. 1986. Vol. 35. P. 583–590.

[76] Gebaly K. E., Aboulnaga A. Robustness in automatic physical database design // Proceedings of the 11th international conference on Extending database technology: Advances in database technology. EDBT '08. New York, NY, USA : ACM, 2008. P. 145–156. URL: http://doi.acm.org/10.1145/1353343.1353365.

[77] Ghandeharizadeh S., DeWitt D. J. Hybrid-range partitioning strategy: A new declustering strategy for multiprocessor database machines // Proceedings of the 16th International Conference on Very Large Data Bases. VLDB '90. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1990. P. 481–492. URL: http://dl.acm.org/citation.cfm?id=645916.671988.

[78] Ghandeharizadeh S., DeWitt D. J. Magic: A multiattribute declustering mechanism for multiprocessor database machines // IEEE Trans. Parallel Distrib. Syst. 1994. Vol. 5, no. 5. P. 509–524. URL: http://dx.doi.org/10.1109/71.282561.

[79] Ghandeharizadeh S., DeWitt D. J., Qureshi W. A performance analysis of alternative multi-attribute declustering strategies // SIGMOD Rec. 1992. Vol. 21, no. 2. P. 29–38. URL: http://doi.acm.org/10.1145/141484.130293.

[80] Goals and benchmarks for autonomic configuration recommenders / Mariano P. Consens, Denilson Barbosa, Adrian Teisanu, Laurent Mignet // Proceedings of the 2005 ACM SIGMOD international conference on Management of data. SIGMOD '05. New York, NY, USA : ACM, 2005. P. 239–250. URL: http://doi.acm.org/10.1145/1066157.1066185.

[81] Golfarelli M., Maio D., Rizzi S. Vertical fragmentation of views in relational data warehouses // Proc. Settimo Convegno Nazionale Sistemi Evoluti per Basi di Dati (SEBD 1999). 1999. P. 19–33.

[82] Gorla N., Yan B. P. W. Vertical fragmentation in databases using data-mining technique // Database Technologies: Concepts, Methodologies, Tools, and Applications / Ed. by John Erickson. IGI Global, 2009. P. 2543–2563.

[83] Guinepain S., Gruenwald L. Research issues in automatic database clustering // SIGMOD Rec. 2005. Vol. 34. P. 33–38. URL: http://doi.acm.org/10.1145/1058150.1058157.

[84] Guinepain S., Gruenwald L. Automatic database clustering using data mining // Proceedings of the 17th International Conference on Database and Expert Systems Applications. DEXA '06. Washington, DC, USA : IEEE Computer Society, 2006. P. 124–128. URL: http://dx.doi.org/10.1109/DEXA.2006.32.

[85] Hababeh I. O., Ramachandran M., Bowring N. A high-performance computing method for data allocation in distributed database systems // J. Supercomput. 2007. Vol. 39, no. 1. P. 3–18. URL: http://dx.doi.org/10.1007/s11227-006-0001-8.

[86] Hammer M., Niamir B. A heuristic approach to attribute partitioning // Proceedings of the 1979 ACM SIGMOD international conference on Management of data. SIGMOD '79. New York, NY, USA : ACM, 1979. P. 93–101. URL: http://doi.acm.org/10.1145/582095.582110.

[87] 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.

[88] Hauglid J. O., Ryeng N. H., Nørvåg K. Dyfram: dynamic fragmentation and replica management in distributed database systems // Distrib. Parallel Databases. 2010. Vol. 28. P. 157–185. URL: http://dx.doi.org/10.1007/s10619-010-7068-1.

[89] 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.

[90] High Performance Parallel Database Processing and Grid Databases / David Taniar, Clement H. C. Leung, Wenny Rahayu, Sushant Goel. Wiley Publishing, 2008. ISBN: 0470107626, 9780470107621.

[91] Hoffer J. A. An integer programming formulation of computer database design problems // Inf. Sci. 1976. Vol. 11. P. 29–48.

[92] Hoffer J. A., Severance D. G. The use of cluster analysis in physical data base design // Proceedings of the 1st International Conference on Very Large Data Bases. VLDB '75. New York, NY, USA : ACM, 1975. P. 69–86. URL: http://doi.acm.org/10.1145/1282480.1282486.

[93] Hua K. A., Lee C. An adaptive data placement scheme for parallel database computer systems // Proceedings of the 16th International Conference on Very Large Data Bases. VLDB '90. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1990. P. 493–506. URL: http://dl.acm.org/citation.cfm?id=645916.671965.

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

[95] Huang Y.-F., Van C.-H. Vertical partitioning in database design // Information Sciences. 1995. Vol. 86, no. 1-3. P. 19–35. URL: http://www.sciencedirect.com/science/article/pii/002002559500018K.

[96] 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.

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

[98] Jakobsson M. Reducing block accesses in inverted files by partial clustering // Information Systems. 1980. Vol. 5, no. 1. P. 1–5. URL: http://www.sciencedirect.com/science/article/pii/0306437980900630.

[99] Kamali S., Ghodsnia P., Daudjee K. Dynamic data allocation with replication in distributed systems // Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30th International. 2011. nov. P. 1–8.

[100] Karimi Adl R., Rouhani Rankoohi S. A new ant colony optimization based algorithm for data allocation problem in distributed databases // Knowledge and Information Systems. 2009. Vol. 20. P. 349–373. 10.1007/s10115-008-0182-y. URL: http://dx.doi.org/10.1007/s10115-008-0182-y.

[101] Karlapalem K., Pun N. M. Query-driven data allocation algorithms for distributed database systems // Proceedings of the 8th International Conference on Database and Expert Systems Applications. DEXA '97. London, UK, UK : Springer-Verlag, 1997. P. 347–356. URL: http://dl.acm.org/citation.cfm?id=648310.754564.

[102] Stepwise redesign of distributed relational databases : Rep. : HKUST-CS97-12 / Department of Computer Science, The Hong-Kong University of Science & Technology ; Executor: Ladan Kazerouni, Kamalakar Karlapalem : September 1997.

[103] Kemme B., Alonso G. A new approach to developing and implementing eager database replication protocols // ACM Trans. Database Syst. 2000. Vol. 25, no. 3. P. 333–379. URL: http://doi.acm.org/10.1145/363951.363955.

[104] Khalil N., Eid D., Khair M. Availability and reliability issues in distributed databases using optimal horizontal fragmentation // Database and Expert Systems Applications / Ed. by Trevor Bench-Capon, Giovanni Soda, A Tjoa. Springer Berlin / Heidelberg, 1999. Vol. 1677 of Lecture Notes in Computer Science. P. 771–780. 10.1007/3-540-48309-8_72. URL: http://dx.doi.org/10.1007/3-540-48309-8_72.

[105] Khan S., Hoque A. A new technique for database fragmentation in distributed systems // International Journal of Computer Applications. 2010. Vol. 5(9). P. 20–24.

[106] Lightstone S. S., Bhattacharjee B. Automated design of multidimensional clustering tables for relational databases // Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. VLDB '04. VLDB Endowment, 2004. P. 1170–1181. URL: http://dl.acm.org/citation.cfm?id=1316689.1316789.

[107] Lima A., Mattoso M., Valduriez P. OLAP query processing in a database cluster // Euro-Par 2004 Parallel Processing / Ed. by Marco Danelutto, Marco Vanneschi, Domenico Laforenza. Springer Berlin / Heidelberg, 2004. Vol. 3149 of Lecture Notes in Computer Science. P. 355–362. 10.1007/978-3-540-27866-5_46. URL: http://dx.doi.org/10.1007/978-3-540-27866-5_46.

[108] Lin X., Orlowska M., Zhang Y. A graph based cluster approach for vertical partitioning in database design // Data & Knowledge Engineering. 1993. Vol. 11, no. 2. P. 151–169. URL: http://www.sciencedirect.com/science/article/pii/0169023X93900038.

[109] Lin X., Orlowska M. E., Zhang Y. On data allocation with the minimum overall communication costs in distributed database design // Proceedings of the Fifth International Conference on Computing and Information. ICCI '93. Washington, DC, USA : IEEE Computer Society, 1993. P. 539–544. URL: http://dl.acm.org/citation.cfm?id=645468.654762.

[110] Ma H. Distribution Design for Complex Value Databases : Ph. D. thesis / H. Ma. Massey University, 2007.

[111] Ma H., Schewe K.-D., Kirchberg M. A heuristic approach to fragmentation incorporating query information // Proceedings of the 2007 conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DB&IS'2006. Amsterdam, The Netherlands, The Netherlands : IOS Press, 2007. P. 103–116. URL: http://dl.acm.org/citation.cfm?id=1565421.1565432.

[112] Mahmoud S., Riordon J. S. Optimal allocation of resources in distributed information networks // ACM Trans. Database Syst. 1976. Vol. 1, no. 1. P. 66–78. URL: http://doi.acm.org/10.1145/320434.320449.

[113] Mami I., Bellahsene Z. A survey of view selection methods // SIGMOD Rec. 2012. Vol. 41, no. 1. P. 20–29. URL: http://doi.acm.org/10.1145/2206869.2206874.

[114] March S. T., Rho S. Allocating data and operations to nodes in distributed database design // IEEE Trans. on Knowl. and Data Eng. 1995. Vol. 7, no. 2. P. 305–317. URL: http://dx.doi.org/10.1109/69.382299.

[115] McIver Jr. W. J., King R. Self-adaptive, on-line reclustering of complex object data // SIGMOD Rec. 1994. Vol. 23, no. 2. P. 407–418. URL: http://doi.acm.org/10.1145/191843.191924.

[116] Mehta M., DeWitt D. J. Data placement in shared-nothing parallel database systems // The VLDB Journal. 1997. Vol. 6. P. 53–72. URL: http://dx.doi.org/10.1007/s007780050033.

[117] Menon S. Allocating fragments in distributed databases // IEEE Trans. Parallel Distrib. Syst. 2005. Vol. 16, no. 7. P. 577–585. URL: http://dx.doi.org/10.1109/TPDS.2005.77.

[118] Moghrabi I., Makholian R. A new approach to clustering records in information retrieval systems // Information Retrieval. 2000. Vol. 3. P. 105–126. 10.1023/A:1009901830009. URL: http://dx.doi.org/10.1023/A:1009901830009.

[119] Morgan H. L., Levin K. D. Optimal program and data locations in computer networks // Commun. ACM. 1977. Vol. 20, no. 5. P. 315–322. URL: http://doi.acm.org/10.1145/359581.359591.

[120] Motzkin D. An optimal data allocation model for distributed databases // Mathematical and Computer Modelling. 1988. Vol. 11, no. 0. P. 920–925. URL: http://www.sciencedirect.com/science/article/pii/0895717788906280.

[121] Mukkamala R., Bruell S. C., Shultz R. K. Design of partially replicated distributed database systems: an integrated methodology // Proceedings of the 1988 ACM SIGMETRICS conference on Measurement and modeling of computer systems. SIGMETRICS '88. New York, NY, USA : ACM, 1988. P. 187–196. URL: http://doi.acm.org/10.1145/55595.55617.

[122] Navathe S. B., Karlapalem K., Ra M. A mixed fragmentation methodology for initial distributed database design // Journal of Computer and Software Engineering. 1995. Vol. 3.

[123] Navathe S. B., Ra M. Vertical partitioning for database design: a graphical algorithm // Proceedings of the 1989 ACM SIGMOD international conference on Management of data. SIGMOD '89. New York, NY, USA : ACM, 1989. P. 440–450. URL: http://doi.acm.org/10.1145/67544.66966.

[124] 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.

[125] Noaman A. Y., Barker K. A horizontal fragmentation algorithm for the fact relation in a distributed data warehouse // Proceedings of the eighth

international conference on Information and knowledge management. CIKM '99. New York, NY, USA : ACM, 1999. P. 154–161. URL: http://doi.acm.org/10.1145/319950.319972.

[126] An objective function for vertically partitioning relations in distributed databases and its analysis / Sharma Chakravarthy, Jaykumar Muthuraj, Ravi Varadarajan, Shamkant B. Navathe // Distributed and Parallel Databases. 1994. Vol. 2. P. 183–207. 10.1007/BF01267326. URL: http://dx.doi.org/10.1007/BF01267326.

[127] Omiecinski E., Scheuermann P. A parallel algorithm for record clustering // ACM Trans. Database Syst. 1990. Vol. 15. P. 599–624. URL: http://doi.acm.org/10.1145/99935.99947.

[128] Özsu M. T., Valduriez P. Principles of distributed database systems (2nd ed.). Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1999. ISBN: 0-13-659707-6.

[129] 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.

[130] Pai-Cheng C. A transaction-oriented approach to attribute partitioning // Information Systems. 1992. Vol. 17, no. 4. P. 329–342. URL: http://www.sciencedirect.com/science/article/pii/030643799290022F.

[131] Papadomanolakis S., Ailamaki A. AutoPart: automating schema design for large scientific databases using data partitioning // Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on. 2004. june. P. 383–392.

[132] Papadomanolakis S., Ailamaki A. An integer linear programming approach to database design // Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop. ICDEW '07. Washington, DC, USA : IEEE Computer Society, 2007. P. 442–449. URL: http://dx.doi.org/10.1109/ICDEW.2007.4401027.

[133] Papadomanolakis S., Dash D., Ailamaki A. Efficient use of the query optimizer for automated physical design // Proceedings of the 33rd international conference on Very large data bases. VLDB '07. VLDB Endowment, 2007. P. 1093–1104. URL: http://dl.acm.org/citation.cfm?id=1325851.1325974.

[134] Parallel OLAP query processing in database clusters with data replication / Alexandre Lima, A., Camille Furtado, Patrick Valduriez, Marta Mattoso // Distributed and Parallel Databases. 2009. Vol. 25, no. 1-2. P. 97–123. URL: http://hal.inria.fr/inria-00482183.

[135] 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.

[136] Park S.-J., Baik D.-K. A data allocation considering data availability in distributed database systems // Proceedings of the 1997 International Conference on Parallel and Distributed Systems. ICPADS '97. Washington, DC, USA : IEEE Computer Society, 1997. P. 708–713. URL: http://dl.acm.org/citation.cfm?id=646861.707304.

[137] Physical and virtual partitioning in OLAP database clusters / Camille Furtado, Alexandre A. B. Lima, Esther Pacitti et al. // Proceedings of the 17th International Symposium on Computer Architecture on High Performance Computing. SBAC-PAD '05. Washington, DC, USA : IEEE Computer Society, 2005. P. 143–150. URL: http://dx.doi.org/10.1109/CAHPC.2005.32.

[138] A query processing strategy for the decomposed storage model / Setrag Khoshafian, George P. Copeland, Thomas Jagodis et al. // Proceedings of the Third International Conference on Data Engineering. Washington, DC, USA : IEEE Computer Society, 1987. P. 636–643. URL: http://dl.acm.org/citation.cfm?id=645472.655555.

[139] Rahm E., Marek R. Analysis of dynamic load balancing strategies for parallel shared nothing database systems // Proceedings of the 19th International Conference on Very Large Data Bases. VLDB '93. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1993. P. 182–193. URL: http://portal.acm.org/citation.cfm?id=645919.672662.

[140] Ram S., Marsten R. A model for database allocation incorporating a concurrency control mechanism // Knowledge and Data Engineering, IEEE Transactions on. 1991. Vol. 3, no. 3. P. 389–395.

[141] Ram S., Narasimhan S. Database allocation in a distributed environment: incorporating a concurrency control mechanism and queuing costs // Manage. Sci. 1994. Vol. 40, no. 8. P. 969–983. URL: http://dx.doi.org/10.1287/mnsc.40.8.969.

[142] Ramamurthy R., DeWitt D. J., Su Q. A case for fractured mirrors // Proceedings of the 28th international conference on Very Large Data Bases. VLDB '02. VLDB Endowment, 2002. P. 430–441. URL: http://dl.acm.org/citation.cfm?id=1287369.1287407.

[143] Recommending materialized views and indexes with the IBM DB2 design advisor / D.C. Zilio, C. Zuzarte, S. Lightstone et al. // Autonomic Computing, 2004. Proceedings. International Conference on. 2004. may. P. 180–187.

[144] Rivera-Vega P., Varadarajan R., Navathe S. Scheduling data redistribution in distributed databases // Data Engineering, 1990. Proceedings. Sixth International Conference on. 1990. feb. P. 166–173.

[145] Rodriguez L., Li X. A support-based vertical partitioning method for database design // Electrical Engineering Computing Science and Automatic Control (CCE), 2011 8th International Conference on. 2011. oct. P. 1–6.

[146] Rodriguez L., Li X. A vertical partitioning algorithm for distributed multimedia databases // Proceedings of the 22nd international conference on

Database and expert systems applications - Volume Part II. DEXA'11. Berlin, Heidelberg : Springer-Verlag, 2011. P. 544–558. URL: http://dl.acm.org/citation.cfm?id=2033546.2033607.

[147] Röhm U., Böhm K., Schek H.-J. OLAP query routing and physical design in a database cluster // Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology. EDBT '00. London, UK, UK : Springer-Verlag, 2000. P. 254–268. URL: http://dl.acm.org/citation.cfm?id=645339.650130.

[148] Sacca D., Wiederhold G. Database partitioning in a cluster of processors // ACM Trans. Database Syst. 1985. Vol. 10. P. 29–56. URL: http://doi.acm.org/10.1145/3148.3161.

[149] Sarathy R., Shetty B., Sen A. A constrained nonlinear 0-1 program for data allocation // European Journal of Operational Research. 1997. Vol. 102, no. 3. P. 626–647. URL: http://www.sciencedirect.com/science/article/pii/S0377221796002342.

[150] Savonnet M., Terrasse M., Yétongnon K. Fragtique: A methodology for distributing object oriented databases // International Conference on Computing and Information. 1998.

[151] Shin D.-G., Irani K. B. Fragmenting relations horizontally using a knowledge-based approach // IEEE Trans. Softw. Eng. 1991. Vol. 17. P. 872–883. URL: http://dl.acm.org/citation.cfm?id=126262.126266.

[152] Singh A., Kahlon K. Non-replicated dynamic data allocation in distributed database systems // IJCSNS International Journal of Computer Science and Network Security. 2009. Vol. 9, no. 9. P. 176–180. URL: http://paper.ijcsns.org/07_book/200909/20090922.pdf.

[153] Son J. H., Kim M.-H. α-partitioning algorithm: Vertical partitioning based on the fuzzy graph // Proceedings of the 12th International Conference on Database and Expert Systems Applications. DEXA '01. London, UK, UK : Springer-Verlag, 2001. P. 537–546. URL: http://dl.acm.org/citation.cfm?id=648314.755837.

[154] Son J. H., Kim M.-H. An adaptable vertical partitioning method in distributed systems // Journal of Systems and Software. 2004. Vol. 73, no. 3. P. 551–561.

[155] Song S.-K., Gorla N. A genetic algorithm for vertical fragmentation and access path selection // The Computer Journal. 2000. Vol. 43, no. 1. P. 81–93. http://comjnl.oxfordjournals.org/content/43/1/81.full.pdf+html.

[156] Stöhr T., Märtens H., Rahm E. Multi-dimensional database allocation for parallel data warehouses // Proceedings of the 26th International Conference on Very Large Data Bases. VLDB '00. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2000. P. 273–284. URL: http://dl.acm.org/citation.cfm?id=645926.671843.

[157] Stöhr T., Rahm E. WARLOCK: A data allocation tool for parallel warehouses // Proceedings of the 27th International Conference on Very Large Data Bases. VLDB '01. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2001. P. 721–722. URL: http://dl.acm.org/citation.cfm?id=645927.672208.

[158] Sun J., Grosky W. I. Dynamic maintenance of multidimensional range data partitioning for parallel data processing // Proceedings of the 1st ACM international workshop on Data warehousing and OLAP. DOLAP '98. New York, NY, USA : ACM, 1998. P. 72–79. URL: http://doi.acm.org/10.1145/294260.294275.

[159] 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.

[160] Suri R. A decentralized approach to optimal file allocation in computer networks // Decision and Control including the Symposium on Adaptive Processes, 1979 18th IEEE Conference on. Vol. 18. 1979. dec. P. 141–146.

[161] 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.

[162] Teorey T. J. Distributed database design: a practical approach and example // SIGMOD Rec. 1989. Vol. 18, no. 4. P. 23–39. URL: http://doi.acm.org/10.1145/74120.74124.

[163] 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.

[164] TPC Benchmark C. On-line transaction processing benchmark. http://www.tpc.org/tpcc. Дата просмотра: 30/07/2012.

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

[166] Tsangaris M. M., Naughton J. F. On the performance of object clustering techniques // SIGMOD Rec. 1992. Vol. 21, no. 2. P. 144–153. URL: http://doi.acm.org/10.1145/141484.130308.

[167] Tsatalos O. G., Solomon M. H., Ioannidis Y. E. The GMAP: a versatile tool for physical data independence // The VLDB Journal. 1996. Vol. 5. P. 101–118. URL: http://dx.doi.org/10.1007/s007780050018.

[168] Ulus T., Uysal M. Heuristic approach to dynamic data allocation in distributed database systems // Pakistan Journal of Information and Technology. 2003. Vol. 2. P. 231–239.

[169] Verification of partitioning and allocation techniques on teradata DBMS / Ladjel Bellatreche, Soumia Benkrid, Ahmad Ghazal et al. // Algorithms and Architectures for Parallel Processing / Ed. by Yang Xiang, Alfredo Cuzzocrea, Michael Hobbs, Wanlei Zhou. Springer Berlin / Heidelberg, 2011. Vol. 7016 of Lecture Notes in Computer Science. P. 158–169. 10.1007/978-3-642-24650-0_14. URL: http://dx.doi.org/10.1007/978-3-642-24650-0_14.

[170] Vertical fragmentation and allocation in distributed databases with site capacity restrictions using the threshold accepting algorithm / Joaquin Perez, Rodolfo Pazos, Juan Frausto et al. // MICAI 2000: Advances in Artificial Intelligence / Ed. by Osvaldo Cairo, L.Enrique Sucar, FranciscoJ. Cantu. Springer Berlin Heidelberg, 2000. Vol. 1793 of Lecture Notes in Computer Science. P. 75–81. URL: http://dx.doi.org/10.1007/10720076_7.

[171] Vertical fragmentation design of distributed databases considering the nonlinear nature of roundtrip response time / Rodolfo Pazos R., Graciela Vazquez A., Jose Martinez F., Joaquin Perez O. // Knowledge-Based and Intelligent Information and Engineering Systems / Ed. by Rossitza Setchi, Ivan Jordanov, Robert Howlett, Lakhmi Jain. Springer Berlin / Heidelberg, 2010. Vol. 6277 of Lecture Notes in Computer Science. P. 173–182. 10.1007/978-3-642-15390-7_18. URL: http://dx.doi.org/10.1007/978-3-642-15390-7_18.

[172] Vertical partitioning algorithms for database design / Shamkant Navathe, Stefano Ceri, Gio Wiederhold, Jinglie Dou // ACM Trans. Database Syst. 1984. Vol. 9. P. 680–710. URL: http://doi.acm.org/10.1145/1994.2209.

[173] Vertical partitioning algorithms in distributed databases / M. Zorrilla, E. Mora, P. Corcuera, J. Fernández // Computer Aided Systems Theory - EUROCAST'99 / Ed. by Peter Kopacek, Roberto Moreno-Díaz, Franz Pichler. Springer Berlin / Heidelberg, 2000. Vol. 1798 of Lecture Notes in Computer Science. P. 465–474. 10.1007/10720123_40. URL: http://dx.doi.org/10.1007/10720123_40.

[174] View management techniques and their application to data stream management / Christoph Quix, Xiang Li, David Kensche, Sandra Geisler // Evolving Application Domains of Data Warehousing and Mining: Trends and Solutions / Ed. by Pedro Nuno San-Banto Furtado. IGI Global, 2010. P. 83–112. URL: http://dbis.rwth-aachen.de/cms/staff/li/view-selection-survey.

[175] Wada K. Replication // Encyclopedia of Database Systems / Ed. by Ling Liu, M.Tamer Özsu. Springer US, 2009. P. 2391–2392. URL: http://dx.doi.org/10.1007/978-0-387-39940-9_1336.

[176] Wah B. File placement on distributed computer systems // Computer. 1984. Vol. 17, no. 1. P. 23–32.

[177] Wah B. W., Lien Y.-N. Design of distributed databases on local computer systems with a multiaccess network // IEEE Trans. Softw. Eng. 1985. Vol. 11, no. 7. P. 606–619. URL: http://dx.doi.org/10.1109/TSE.1985.232505.

[178] 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.

[179] Wong E., Katz R. H. Distributing a database for parallelism // SIGMOD Rec. 1983. Vol. 13, no. 4. P. 23–29. URL: http://doi.acm.org/10.1145/971695.582201.

[180] Zhang Y., Orlowska M. E. On fragmentation approaches for distributed database design // Information Sciences - Applications. 1994. Vol. 1, no. 3. P. 117–132. URL: http://www.sciencedirect.com/science/article/pii/1069011594900051.

[181] Zilio D. C. Physical database design decision algorithms and concurrent reorganization for parallel database systems : Ph. D. thesis / Daniel Costante Zilio. Toronto, Ont., Canada, Canada : University of Toronto, 1998. AAINQ35386.

[182] Partitioning key selection for a shared-nothing parallel database system : Rep. : IBM Research Report RC 19820 (87739) / IBM T.J. Watson Research Center ; Executor: Daniel C. Zilio, Anant Jhingran, Sriram Padmanabhan : 1994.

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

Chernishev A. George — assistant professor of Computer Science Department, SPbSU. Research Area: distributed databases. phone +7(905)258-0279.

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

Рекомендовано СПИИРАН, лаборатория ТиМПИ, заведующий А.Л. Тулупьев.

Статья поступила в редакцию 23.01.2013.

РЕФЕРАТ

Чернышев Г.А. **Обзор подходов к организации физического уровня в СУБД**

Цель настоящей работы — на основе сравнительного анализа источников представить обзор возможных подходов к решению задач, связанных с выбором структур физического уровня СУБД.

В статье представлена постановка задачи настройки СУБД в общем виде, приведены уточнения, с помощью которых задача приводится к задаче о выборе структур физического уровня в СУБД. Рассмотрены возможности, открывающиеся при успешном решении задачи выбора структур физического уровня СУБД. Охарактеризовано решение и классифицированы возможные методы решения. Детально изучен пример современной работы, выделены основные фазы решения. По ходу работы рассматриваются исследования методов выбора структур физического уровня СУБД с момента зарождения области и до настоящего времени. Выделено три этапа в развитии данной области.

Рассмотрено горизонтальное фрагментирование и размещение данных. Освещены вопросы, касающиеся выбора методов фрагментирования и их применения, в том числе и в OLAP системах. Рассмотрены различные типы горизонтального фрагментирования: зависимое и независимое от атрибутов. Отдельно описано производное фрагментирование и фрагментирование с использованием кластеризации записей. Установлена связь последнего подхода и современного направления адаптивного индексирования. Приведены примеры применения обоих методов в объектных системах. Освещена проблема выбора архитектуры для совместного решения задач горизонтального фрагментирования и размещения. Представлены интегрированный и итеративный подходы. Вкратце рассмотрены вопросы репликации данных в контексте выбора структур физического уровня.

Проблема размещения данных рассмотрена как в статическом, так и в динамическом варианте. Описан общий процесс поиска решения задачи в динамическом варианте. Приведены наиболее активные на момент написания статьи направления исследований.

Рассмотрено вертикальное фрагментирование, приведены два класса методов отыскания решения. Установлена связь с современными СУБД в оперативной памяти.

Кроме теоретических работ был произведен обзор применения данных методов в современных коммерческих СУБД, приведены описания подсистем рекомендации структур физического уровня. Было представлено как устройство данных подсистем, так и фрагментирование, как инструмент доступный пользователю.

SUMMARY

Chernishev G.A. **A survey of DBMS physical design approaches**

The aim of this paper is to present a survey of physical design approaches in DBMS. At first, the paper presents the general problem of database tuning, then we restrict it to the problem of selection of physical database structures. Next, we list possible benefits which can be acquired with proper selection of these structures. Then, we describe the solution in general case and classify methods to obtain one. We pick one of the contemporary research papers to study in detail in order to point out and list general solution phases. Then we study research works ranging from origins of the subject area till nowadays. The result is the classification of these works into three periods of subject area development.

In this paper we consider horizontal partitioning and data allocation. We review questions related to the choice of partitioning methods and their application and pay attention to OLAP systems. We describe two types of horizontal partitioning: attribute dependent and attribute independent. Then, we study derived fragmentation and fragmentation using record clusterization. Some examples of their application in object database systems are given. Also, we note similarity among the latter approach and adaptive indexing — a new and very active area of research. Then, we describe the architecture selection problem for the combined problem of horizontal partitioning and allocation. The alternatives — integrated and iterative approaches are presented. We briefly consider some issues of replication related to partitioning problem.

The allocation problem is considered in both static and dynamic contexts, a general scheme for solution of dynamic allocation problem is presented. Finally, we list some active research areas related to partitioning problem.

Vertical partitioning is also considered, we present two types of methods for this problem. A link between contemporary approaches for main-memory DBMS fragmentation and vertical partitioning works of the past is given.

Aside from the theoretical works, we survey the application of these methods in contemporary commercial DBMS, descriptions of the corresponding subsystems are given. Also we report user's perspective — a fragmentation as a tool approach.