DOI: 10.18721/JCSTCS.12205
УДК 004.652.42, 004.657, 004.655.3:004.652.4

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ДЕКОМПОЗИЦИИ И ВЫПОЛНЕНИЯ ЗАПРОСОВ НА ВЫБОРКУ В ГЕТЕРОГЕННЫХ СИСТЕМАХ УПРАВЛЕНИЯ РЕЛЯЦИОННЫМИ БАЗАМИ ДАННЫХ

С.Г. Попов, А.А. Пурий

Санкт-Петербургский политехнический университет Петра Великого,
Санкт-Петербург, Российская Федерация

При работе с независимыми базами данных появляется проблема кусочной автоматизации, состоящая в невозможности получать данные из независимых баз с заранее неизвестной схемой. У проблемы два аспекта: отсутствие семантической связи между независимыми базами и динамика семантики схем. Разработаны и исследованы алгоритмы построения интегрированных запросов на выборку данных к гетерогенным распределенным системам управления базами данных. Вычислены алгоритмы декомпозиции и выполнения запроса на выборку. Для проверки работоспособности алгоритмов разработано программное обеспечение и база данных, обеспечивающие обращение к независимым базам, как если бы они находились в единой СУБД, но без полной репликации объединяемых баз данных.

Ключевые слова: гетерогенные СУБД, метаданные, распределенные запросы, декомпозиция запросов, распределенные СУБД, интеграция баз данных, обработка запросов, СУБД, базы данных.

Ссылка при цитировании: Попов С.Г., Пурий А.А. Исследование алгоритмов декомпозиции и выполнения запросов на выборку в гетерогенных системах управления реляционными базами данных // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. 2019. Т. 12. № 2. С. 50–67. DOI: 10.18721/JCSTCS.12205

STUDY DECOMPOSING AND EXECUTING ALGORITHMS OF DATA EXTRACTION QUERIES IN HETEROGENEOUS RELATIONSHIP DATABASE MANAGEMENT SYSTEMS

S.G. Popov, A.A. Purii

Peter the Great St. Petersburg Polytechnic University,
St. Petersburg, Russian Federation

When working with independent databases, the problem of partial automation arises, which consists in the impossibility of obtaining data from independent databases with a previously unknown scheme. The problem consists of two aspects: the lack of semantic connection between independent databases, and the dynamics of the semantics of schemes. The goal is the development and study of algorithms

for constructing integrated queries for data sampling to heterogeneous distributed database management systems. To test the functionality of the algorithms, software and database of databases, which provides the ability to access independent databases, as if they were a single DBMS, but without full replication of the merging databases.

Keywords: heterogeneous DBMS, metadata, distributed queries, queries decomposition, distributed DBMS, databases integration, queries processing, DBMS, databases.

Citation: Popov S.G., Purii A.A. Study decomposing and executing algorithms of data extraction queries in heterogeneous relationship database management systems. St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunications and Control Systems, 2019, Vol. 12, No. 2, Pp. 50–67. DOI: 10.18721/JCSTCS.12205

Проблема интеграции гетерогенных систем управления базами данных

Независимые базы данных содержат данные об одних и тех же объектах реального мира, но для получения полного набора данных об одном и том же объекте необходимо последовательно извлекать данные из этих баз, а затем осуществлять их дополнительную обработку в специализированном программном обеспечении. Эта проблема вызвана кусочной автоматизацией, состоящей во фрагментации частей единой информационной среды, и, как следствие, невозможности получать данные из независимых баз с заранее неизвестной схемой.

Проблема состоит из двух аспектов. Первый: независимые базы не связаны семантически, соединение ранее независимых баз требует дополнительных усилий администратора баз данных, получив при этом семантически корректный результат [11, 12]. Второй: динамика семантики схем, состоящая в том, что набор и схемы баз данных для выполнения к ним запросов постоянно меняется. Проблема отсутствия семантических связей разрешается с помощью автоматизированных предметно-ориентированных алгоритмов [13, 15, 16], в том числе с использованием метаданных [14, 17, 20].

Второй аспект проблемы может быть решен с помощью нескольких подходов, состоящих в учете динамики семантической схемы. На выбор решения влияют два фактора: динамика семантической схемы [18] и динамика семантики запросов [19]. Современные СУБД рассчитаны на решение задач с большим числом запросов и низкой динамикой схемы данных. В этом случае все базы можно объединить в одной СУБД. Такой выбор не подходит для решения задач с высокой динамикой схем и любой интенсивностью запросов, поскольку при высокой динамике придется включать схемы, которые используются, например, единожды. Кроме того, такое решение требует согласия владельцев на репликацию их баз в единую СУБД. Построение запросов к таким системам может быть решено при помощи класса систем интеграции гетерогенных СУБД.

В этом случае под системой интеграции распределенных гетерогенных СУБД подразумевается системное программное обеспечение и база данных баз данных, которая предоставляет возможность обращаться к независимым базам, как если бы они находились в единой СУБД, но без полной репликации объединяемых баз данных.

Описание методов и технологий реализации распределенных запросов

Построение оптимизированных планов выполнения запросов являлось одной из важнейших задач для систем управления базами данных. Исследования в этой области, проводимые в последние годы, не

только решают задачи построения запросов с использованием новых методов, таких как глубокое обучение с подкреплением, но и задачи построения запросов в более сложных, распределенных системах.

Для оптимизации запросов с помощью точных оценок может использоваться метод оптимизации запросов с помощью выполнения уточняющего подзапроса [1]. Результат такого подзапроса позволяет оценить количество данных, участвующих в запросе, и на этом основании построить оптимальный запрос. Подобный подход позволяет избегать накопления ошибок, возникающих из-за устаревших статистических данных о количестве записей в базе, при построении плана, и основывается только на точных оценках. Данный подход разработан для баз данных, расположенных в ОЗУ, а обработка запросов производилась массивно параллельно на GPU. Негативный аспект подобного подхода в неэффективной работе с базами данных, расположенными в постоянной памяти.

Альтернативным подходом для построения запросов с помощью точных оценок может быть реоптимизация динамических запросов с использованием временного планирования [7]. Временное планирование заключается в выполнении набора заранее определенных запросов для сбора метрик о состоянии системы, например, загрузка CPU, количество свободной памяти, скорость передачи данных по сети. Если состояние системы изменилось таким образом, что текущий план выполнения не является оптимальным, то происходит реоптимизация, и план запроса перестраивается. Подобный подход позволяет создавать более устойчивые к изменениям окружения динамические планы выполнения запросов.

Другим способом оптимизации запросов является использование методов глубокого обучения с подкреплением [2, 3]. Оптимизация заключается не в полной оптимизации запроса, а в составлении оптимального порядка соединений таблиц, передающихся в оптимизатор запроса. Вначале строится дерево соединений, и на нём осуществляется итеративно выбор порядка соединений. На каждой итерации две таблицы или поддерева объединяются в поддерево или результирующее дерево соединений. Нейронная сеть при обучении пытается достичь результата стоимости и времени выполнения запроса меньше, чем результат стандартного оптимизатора соединений в базе данных. Таким образом, она должна обучаться отдельно для каждой реализации системы управления базы данных.

Для обработки запросов в распределенной базе данных в среде беспроводных сенсоров [4] могут использоваться распределенный и централизованный методы оптимизации и обработки [9]. Централизованный алгоритм для проведения операции соединения собирает данные всех таблиц, входящих в запрос, на центральный сервер, на котором и происходит обработка, что повышает нагрузку на сеть. Алгоритм распределенной циклически-вложенной обработки соединений строит дерево, узлами которого являются регионы, вычисляющие запросы соединения на подчиненных им таблицах или регионах. В результате вычислений в корне дерева будет содержаться ответ на запрос. Исследование показало, что время выполнения запроса распределенным алгоритмом было ниже, чем время выполнения централизованным алгоритмом.

Для оптимизации запросов к распределенным базам данных можно использовать улучшенный генетический алгоритм. Оптимизация выполняется только для JOIN. Соединения в виде хромосом задаются закодированными деревьями – иерархией соединений таблиц. Фитнесс-функция – время, которое необходимо затратить на соединение, с учетом затрат для передачи данных. При осуществлении кроссовера используются различные вероятности мутации для разных рангов: чем выше ранг, тем ниже вероятность. Популяция кластеризуется на ранги алгоритмом нечёткой кластеризации. При мутации происходит взаим-

ное перемещение двух таблиц в иерархии, вероятность мутации также зависит от ранга. Критериями завершения алгоритма являются достижение заданного числа эпох или достижение заданной точности.

Обработка интегрированных запросов к гетерогенным базам данных реализована в промышленных программных продуктах, представителями которых являются Apache Calcite и Presto. Пакет Calcite – фреймворк с открытым исходным кодом для оптимизированной обработки запросов для гетерогенных источников данных, – позволяет использовать кроссплатформенную оптимизацию между несколькими системами управления данными, путем использования стандартного интерфейса [8]. Также Calcite требует ручного построения плагинов и проводников для извлечения метаданных и правил для оптимизации запросов. Calcite имеет возможность обрабатывать только не интегрированные SQL-запросы.

Аналогом Calcite является движок с открытым исходным кодом для обработки распределенных SQL-запросов Presto [10]. Presto имеет встроенный оптимизатор, использующий статистическую информацию о таблицах в удаленных базах данных. Основное направление оптимизации – упорядочивание JOIN и распределение их выполнения между узлами. Для упорядочивания JOIN Presto имеет три режима: автоматический, удаление избыточных декартовых произведений (CROSS JOIN) и без оптимизации. В автоматическом режиме оптимизатор упорядочивает JOIN на основе статистических данных, если статистические данные получить невозможно, то оптимизация производится в режиме удаления избыточных декартовых соединений. Исполнение JOIN может происходить в двух режимах: широковещательном и раздельном. В широковещательном режиме каждый узел, участвующий в запросе, использует все данные, они реплицируются на каждый узел. В раздельном режиме каждый узел использует только часть данных. Оптимизатор может выбирать подходящие режимы исполнения автоматически или использовать какой-то определенный в зависимости от настройки. Важной функцией является механизм анализа плана запроса. Он показывает, какие действия будут проведены при выполнении запроса, оценочные показатели количества обрабатываемых строк, количества использования CPU, памяти и сетевого трафика. Такой анализ плана помогает при отладке запросов.

Постановка задачи

Наша задача – разработка и исследование алгоритмов декомпозиции и выполнения интегрированных запросов на выборку данных к гетерогенным СУБД. Для проверки работоспособности и эффективности разрабатываемых алгоритмов предложим архитектуру системы выполнения запросов к гетерогенным СУБД. На основе архитектуры реализуем макет программного обеспечения промежуточного слоя, обеспечивающий хранение метаинформации объединяемых баз данных и реализующий интерпретатор запросов с возможностью декомпозиции и выполнения запросов. Исследуем предложенные алгоритмы, определив границы их применимости.

Алгоритмы выполнения распределенных запросов

Технология централизованного выполнения запросов. При выполнении запросов на выборку данных к гетерогенным СУБД наиболее важной частью является гомогенизация источников, поскольку запросы, быстро выполняющиеся на одном типе гетерогенной СУБД, могут выполняться медленно на другом. Поэтому возникает потребность в переносе данных из гетерогенных источников данных в некоторый источник определенного типа, чтобы избавиться от неоднозначности при построении запросов, а также при получении оценок во время построения этих запросов. Один из самых действенных способов получения данных из гетерогенных СУБД – выполнение к каждой СУБД запросов на выборку данных, при этом минимизировав объем передаваемых данных, выполняя

выборку по определенному условию и с проекцией только на используемые в основном запросе атрибуты. Пример декомпозиции исходного запроса на подзапросы к гетерогенным СУБД приведен на рис. 1. Результаты таких запросов необходимо поместить в некоторую локальную СУБД, к которой имеется полный доступ, и особенности ее функционирования известны. И только после этих операций можно производить соединение локализованных таблиц.

Алгоритмы декомпозиции запросов обеспечивают взаимодействие между различными СУБД, расположенными на разных удаленных серверах. Для осуществления взаимодействия применяется централизованный подход к обработке запросов. При обработке исходного запроса для построения подзапросов, а также динамической проверки корректности оригинального запроса, используются метаданные удаленных СУБД. Для их получения применяется локальная база данных метаданных, в которой хранятся метаданные всех серверов, входящих в систему.

Выполнение запроса осуществляется в три этапа и реализуется четырьмя алгоритмами: алгоритмом проверки корректности запроса, двумя алгоритмами декомпозиции и одним алгоритмом выполнения плана. Алгоритм проверки корректности запроса обеспечивает динамическую проверку синтаксиса запроса к нескольким СУБД. Алгоритмы декомпозиции используют левый вывод для разделения исходного запроса на подзапросы к удаленным СУБД. Алгоритм выполнения плана объединяет промежуточные результаты в окончательный результат запроса. Для получения данных из удаленных СУБД алгоритм выполнения плана использует подход создания локальных копий во временной базе данных на локальном сервере. Схема взаимодействия алгоритмов в ходе централизованного выполнения запроса приведена на рис. 2.

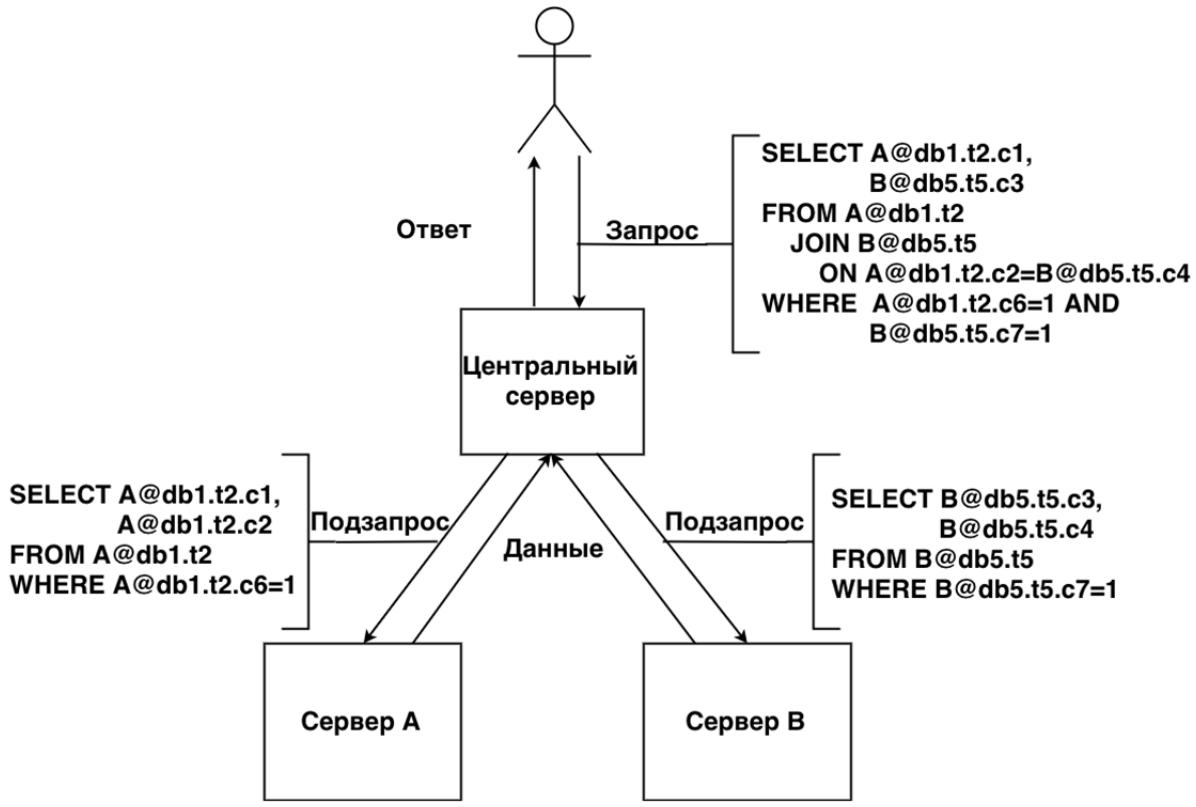

Рис. 1. Пример декомпозиции запроса
Fig. 1. Query decomposition example

Рис. 2. Взаимодействие алгоритмов централизованного метода выполнения запросов

Fig. 2. Centralized query execution method algorithms interaction

Метод выполнения запросов на выборку заключается в использовании одного сервера или центрального сервера в качестве коммутатора для всех серверов СУБД, входящих в систему. Основной задачей данного коммутатора является управление потоками данных из удаленных СУБД и их локальное соединение. План действий с промежуточными результатами, исполняемый алгоритмом выполнения плана, строится при обработке исходного запроса к системе алгоритмами декомпозиции, а результатом выполнения такого плана должен быть результат выполнения исходного запроса.

Этап проверки корректности запроса. На данном этапе выполняется алгоритм динамической проверки корректности. В его задачи входит проверить: все ли имена серверов, баз данных, таблиц и атрибутов, входящие в запрос, существуют; верно ли расставлены все идентификаторы в запросе, например, все ли идентификаторы в «from» и «join» частях запроса являются таблицами, а все идентификаторы в «on» части «join» являются атрибутами. Для таких проверок необходимо знать только имена соответствующих частей идентификаторов.

Алгоритм динамической проверки относится к классу линейной сложности, его оценка – $O(n)$, где n – число идентификаторов в запросе.

Этап построения плана запроса. На этом этапе выполняется выбор последовательности локализации и соединений таблиц. Соединения выполняются двумя алгоритмами: на основе стратегий левого вывода и переупорядочиванием.

Алгоритм декомпозиции левым выводом составляет план выборки и соединения таблиц согласно их порядку во «from» части запроса. Согласно технологии централизованной обработки, данные внеш-

них серверов должны быть локализованы, только после этого можно производить их соединение. В случае с левым выводом алгоритм обрабатывает таблицы попарно. Сначала берутся первые две таблицы, локализуются, затем соединяются. Поскольку результат их соединения находится на центральном сервере и является левой таблицей в следующем соединении, то при последующих соединениях необходимо локализовать только правую таблицу. Для снижения объемов передаваемых данных локализуется не вся удаленная таблица, а только ее проекция с используемыми в запросе атрибутами.

Алгоритмическая сложность декомпозиции запроса левым выводом линейная – $O(n)$, где n – число таблиц в части запроса «from». Учитывая, что число таблиц в части «from» запроса редко превышает 10, использование такого алгоритма для построения планов выполнения запросов является целесообразным.

Алгоритм с переупорядочиванием дополняет алгоритм декомпозиции левым выводом переупорядочиванием соединений в «from» перед началом построения дерева соединений. Для переупорядочивания производится перебор допустимых вариантов порядка соединений. Каждое из них статически оценивается числом строк в результирующей таблице после соединения всех таблиц в указанном порядке.

Для расчета оценки используются кардинальные числа таблиц и коэффициент уникальности атрибута. Оценки объёмов данных в процессе выполнения соединений приведены в табл. 1. В таблице $u(t)$ обозначен коэффициент уникальности для атрибута в таблице t, по которому происходит соединение. Для соединения типа «декартово произведение» верхняя оценка является точной и равна $T_1 * T_2$.

Наилучшим считается то соединение, число строк в результирующей таблице которого минимально. При равной оценке у нескольких порядков соединений приоритет отдается порядку, наиболее эффективно выполняемому используемыми программными средствами реализации.

Сложность алгоритма декомпозиции равна сумме сложностей для рекуррентной процедуры перебора и оценки возможных соединений, и сложности алгоритма декомпозиции левым выводом. Сложность алгоритма декомпозиции левым выводом линейна и зависит от числа таблиц во «from» части запроса. Поскольку перебор осуществляется путем обхода дерева возможных соединений в глубину, то его сложность будет равна $O(E)$, где E – число ребер в дереве, оно зависит от типа соединений таблиц и числа таблиц в запросе. Таким образом, сложность алгоритма с перестановками равна $O(E + n)$.

Этап выполнения запроса. На данном этапе реализуется алгоритм выполнения плана. Алгоритм выполняет пошаговое выполнение плана, составленного алгоритмами декомпозиции. Алгоритм обеспечивает ленивое выполнение плана путем прекращения выполнения запроса, если какая-либо из локализуемых таблиц или результат соединения пустой.

Таблица 1

Статические оценки для соединения двух таблиц

Table 1

Statistic estimates for two tables join

T_1 \ T_2	$u(T_2) = 1$	$u(T_2) < 1$
$u(T_1) = 1$	$\min(T_1, T_2)$	$\max(T_1, T_2)$
$u(T_1) < 1$	$\max(T_1, T_2)$	$(1 - u(T_1)) * T_1 * (1 - u(T_2))T_2$

Алгоритм выполнения плана реализует две процедуры: локализации таблиц и алгоритм соединения таблиц. Процедуры могут работать независимо, но для обеспечения ленивого выполнения порядок их вызова строго описан в плане. Сложность алгоритма выполнения плана линейна и зависит только от числа пунктов в плане – $O(n)$, где n – число пунктов в плане.

Реализация прототипа системы управления выполнением запроса к гетерогенным СУБД

Проверка эффективности алгоритмов выполняется в прототипе системы для обработки запросов к гетерогенным СУБД. Система включает в себя несколько модулей, отвечающих за работу определенной подсистемы. Одним самых важных модулей является модуль оптимизации. Этот модуль содержит реализацию алгоритмов декомпозиции. Помимо внутреннего взаимодействия система обменивается данными с различными СУБД. Для возможности функционирования системы необходимо наличие двух технологических баз: базы данных метаданных и временной базы данных.

База данных метаданных содержит метаданные всех СУБД, включенных в систему. Метаданные используются на всех этапах обработки и выполнения запроса. Для сохранения промежуточных результатов при выполнении запроса используется временная база данных. Временные базы создаются по одной на выполняемый запрос и удаляются при завершении выполнения запроса.

Система управления запросами реализована на основе модульного принципа модулей. Структурная схема модулей изображена на рис. 3.

На рис. 4 изображена функциональная схема, демонстрирующая, в каких конфигурациях среды возможно использование системы. Поскольку система взаимодействует со всеми СУБД через глобальную или локальную сеть, то возможно два типа размещения базы данных метаданных и временной базы данных. Первый – использование баз, размещенных в глобальной сети Интернет. Преимущество такого подхода в том, что он позволяет сразу множеству копий системы использовать базу метаданных как общий ресурс. Очевидный минус – в том, что при обращении к базе метаданных объем передаваемых данных невелик, но возникающие задержки могут уменьшить производительность.

Второй тип размещения – развертывание локальной СУБД. Преимуществом при таком размещении будет ускорение работы программы за счет отсутствия задержек и высокой скорости при доступе локальной СУБД. При подобном подходе можно использовать СУБД, располагающиеся в оперативной памяти. К недостаткам можно отнести повышение требований к используемому аппаратному обеспечению и необходимость установки и настройки СУБД.

Система реализована на языке программирования Python 3, технологические СУБД – MySQL, внешние базы данных – MySQL и PostgreSQL.

Исследование времени выполнения алгоритмов декомпозиции запросов

Для системы управления запросами к гетерогенным СУБД общее время выполнения запроса складывается из времени, затраченного на построение плана выполнения запроса и на выполнение плана. При выполнении плана основные затраты времени приходятся на локализацию, состоящую в получении данных из удаленных СУБД и в помещении их во временную базу, а также операции соединения, проводимые во временной базе.

Постановка эксперимента. При составлении запросов с соединениями типа «декартово произведение» применялись таблицы-словари небольшого размера. Поскольку размер итоговой таблицы в строках при «декартовом произведении» равен произведению числа строк всех входящих в него таблиц, при использовании больших таблиц время вычисления результата возрастало бы многократно, и поэтому на практике такие соединения не используются.

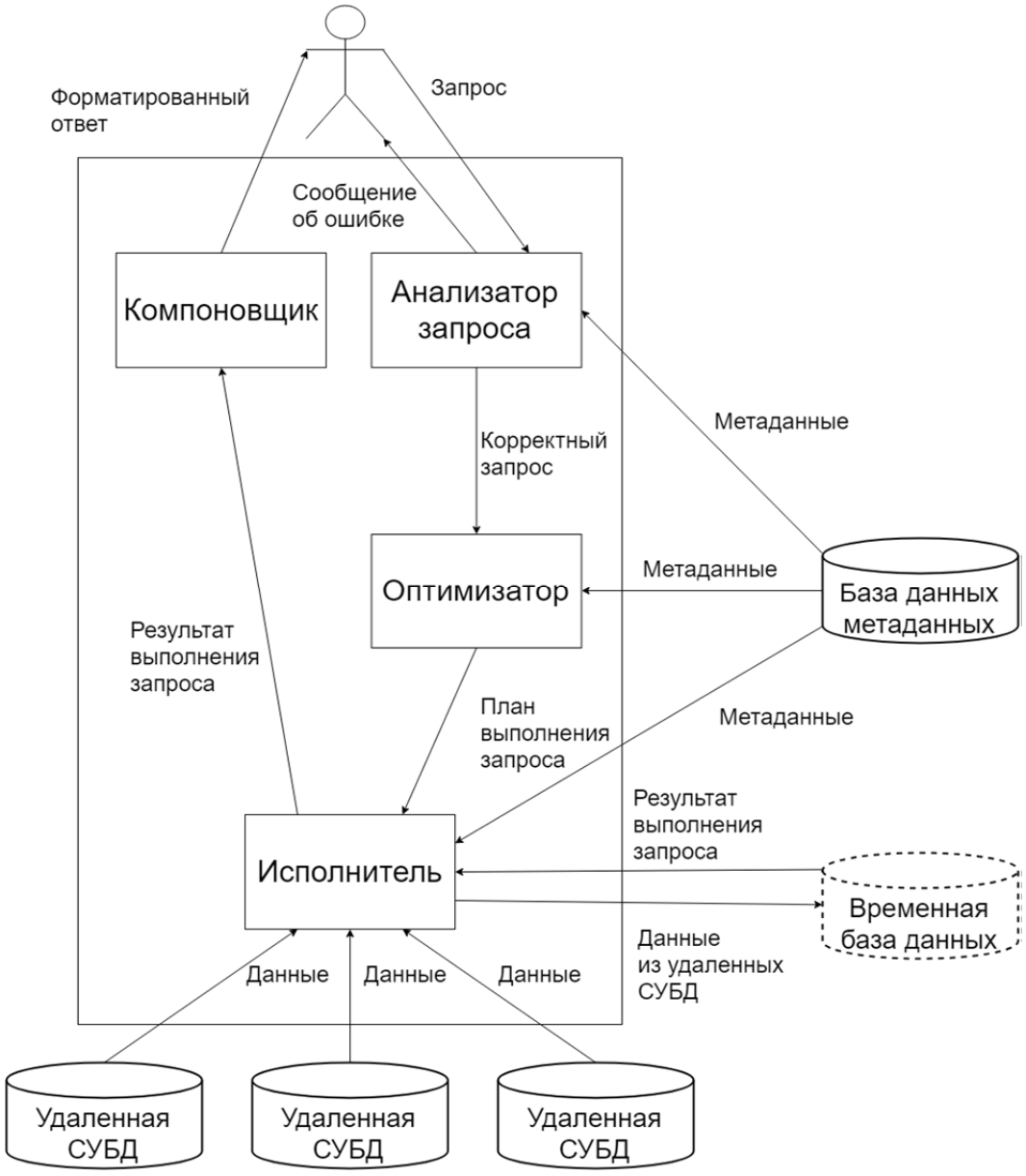

Рис. 3. Структурная схема системы управления выполнением запросов

Fig. 3. Structural diagram of query execution control system

При составлении запросов с соединениями по правилу использовались запросы с правилами соединения типа «первичный ключ – внешний ключ». Такое соединение уже позволяло включать в запрос таблицы больших размеров порядка 10^5 строк.

Составление запросов для смешанного типа соединений производилось путем удаления правил соединения из запросов типа соединение с условием. Для дополнительной проверки времени выполнения системы с большими и малыми объемами записей запросы для последних двух типов соединений были повторно выполнены, но при сниженном объеме входящих в них таблиц.

Для проверки влияния скорости передачи данных на время выполнения запроса все проверки были выполнены как с доступом к СУБД через глобальную сеть Интернет, так и с доступом через локальную сеть. Параметры эксперимента приведены в табл. 2.

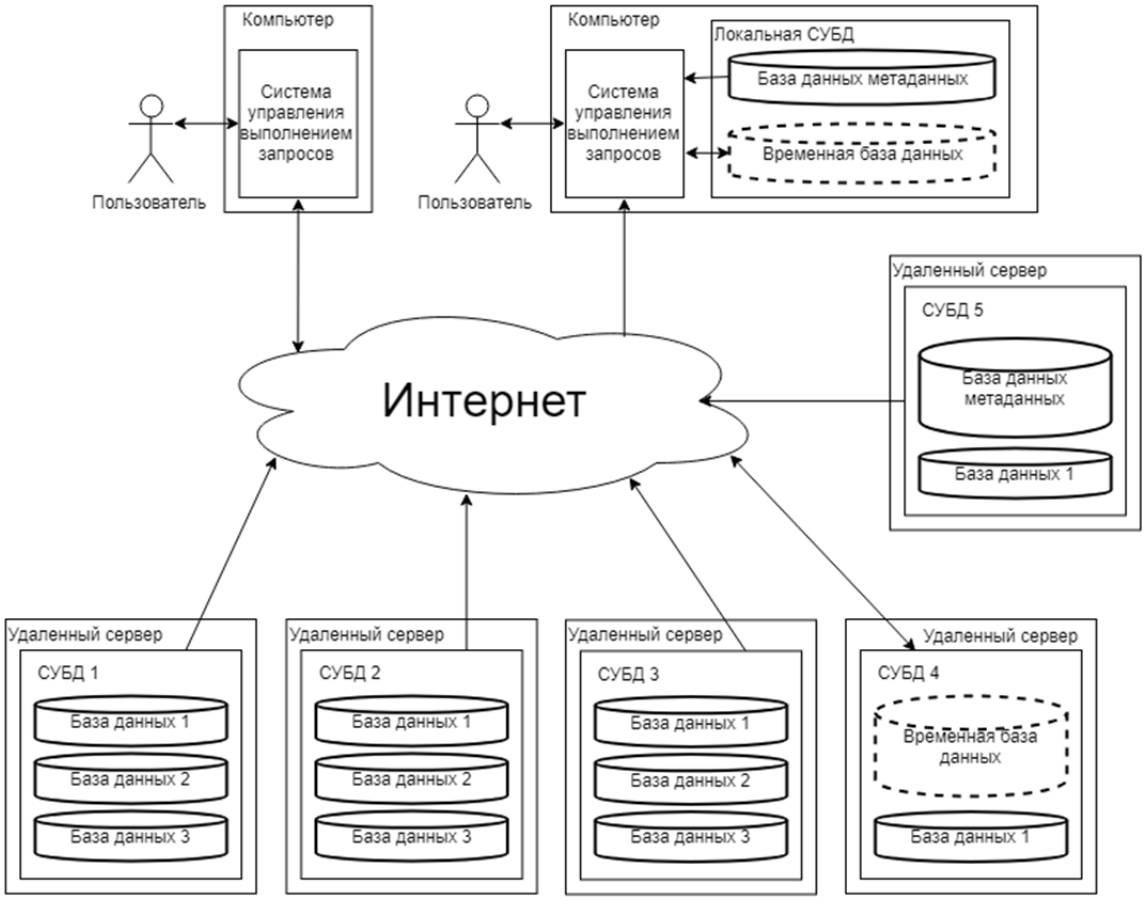

Рис. 4. Функциональная схема системы управления выполнением запросов

Fig. 4. Functional diagram of query execution control system

Таблица 2

Параметры и константы эксперимента

Table 2

Experiment parameters and constants

Параметр или константа	Значение
Параметры данных	
Типы СУБД	MySQL, PostgreSQL
Число внешних серверов БД, шт.	4
Типы сетей	Локальная сеть, Интернет
Объем данных, строки	70000, 5000
Типы соединений	«Декартово произведение», «соединение по правилу», «смешанные соединения»

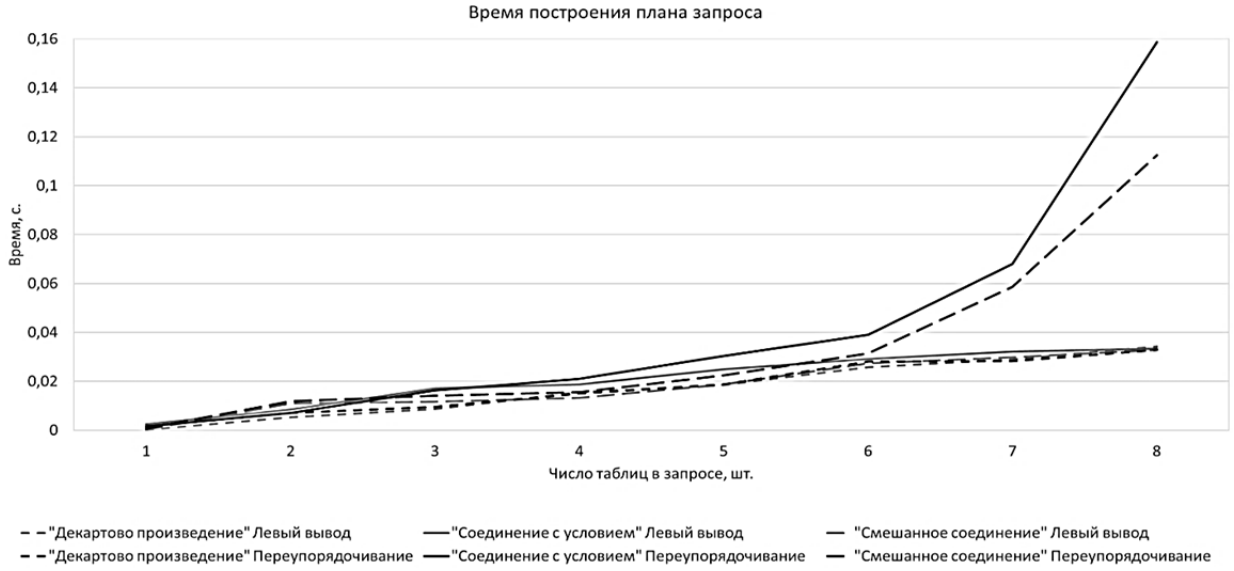

Рис. 5. Зависимость времени построения плана от числа таблиц в запросе

Fig. 5. Dependence of the plan building time on the tables count in the query

СУБД MySQL использовалась как для создания удаленных баз, так и для развертывания локального репозитория для размещения в нем базы данных метаданных и временной базы данных. СУБД PostgreSQL использовалась только для создания удаленных баз.

Для проведения экспериментов написано 40 запросов, по 8 запросов для каждого типа соединения с соответствующим возрастанием числа соединений, и по 8 запросов к СУБД MySQL и к СУБД PostgreSQL. Каждый эксперимент повторялся 20 раз.

Эксперимент 1. Выявление различий во времени построения плана для двух алгоритмов декомпозиции «левого вывода» и «перестановками» для построения плана для трех типов запросов на выборку.

Постановка эксперимента 1. Для каждого из двух алгоритмов и каждого из трех типов запросов «декартово произведение», «соединение по правилу», «смешанные соединения» последовательно выполнить запросы на выборку, содержащие от одной до пяти соединяемых таблиц.

На рис. 5 приведены результаты сравнения времени выполнения для запросов только с соединениями по условию, для запросов только с соединениями типа «декартово произведение» и для смешанного типа запросов. Данные, полученные в результате эксперимента, показывают, что время построения плана алгоритма декомпозиции левым выводом и алгоритма с переупорядочиванием одного порядка.

Для запросов типа «декартово произведение» время выполнения совпадает за счет эвристики соединения таблиц в порядке возрастания числа строк в таблицах.

Время выполнения переупорядочивания зависит только от типа и количества соединений, поэтому для сравнения достаточно привести результаты только одной конфигурации расположения серверов и объемов, обрабатываемых данных. Наихудшим вариантом должно быть наличие в запросе только «декартовых произведений», поскольку в этом случае происходит полный обход дерева всех возможных комбинаций в глубину. Однако в алгоритме переупорядочивания учтены свойства оценок запросов только с соединениями типа «декартово произведение»: для всех возможных вариантов соединения верхняя оценка будет одинаковой и точной. Соответственно, нет необходимости в переборе всех возможных соединений, достаточно выбрать одно – соединение по возрастанию размеров таблиц в строках, поскольку оно обеспечивает наименьшее число операций соединения кортежей. Наилучший вариант – это наличие в запросе только соединений с условием. В данном случае также требуется полный обход дерева в глубину, но дерево состоит не из всех возможных соединений, а только из соединений, удо-

влетворяющих условиям. Соответственно, при смешении двух типов соединений результат должен находиться между наихудшим и наилучшим вариантами.

Наибольшее расхождение между временем построения наблюдается для запросов типа «соединение с условием», причем расхождение увеличивается при увеличении числа таблиц в соединении. Причина такого расхождения в увеличении размеров дерева возможных соединений, которое влечет за собой дополнительный перебор вариантов соединений, а значит, и наиболее затратную с точки зрения вычислительных ресурсов операцию – проверку допустимости соединения. В точке максимального расхождения разница во времени составляет порядка 0,1 с.

Для уменьшения числа рассматриваемых поддеревьев и, соответственно, уменьшения времени построения плана, используется отсечение заведомо неподходящих ветвей. Отсечение ветвей происходит при условии, что оценка в текущем узле дерева больше или равна текущей минимальной оценке. Поскольку поддеревья, которые начинаются с соединений типа «декартово произведение», быстро отсекаются, общее время построения уменьшается. Поэтому разница между двумя планами увеличивается тем больше, чем больше в запросе соединений с условием, и поэтому время построения плана для смешанных соединений меньше, чем для соединений с условием.

Эксперимент 2. Выявление различий времени выполнения планов запросов, построенных двумя алгоритмами.

Постановка эксперимента 2. Для баз данных больших и малых объемов данных, расположенных в локальной и глобальной сетях, и трех типов запросов («декартово произведение», «соединение по правилу», «смешанные соединения») выполнить план соединения от одной до пяти таблиц.

Время выполнения запроса складывается из времени передачи данных по сети и времени соединения этих данных во временной базе данных. При такой конфигурации результаты для локальной сети в большей степени будут демонстрировать непосредственно время соединения локализованных таблиц во временной базе, поскольку в локальной сети влияние задержек минимально, а скорость передачи стабильная и высокая. Для глобальной сети на результаты будет оказываться влияние больших задержек и нестабильной скорости передачи данных.

Цель данного эксперимента – выяснить, существует ли преимущество во времени выполнения плана запроса, построенного алгоритмом с переупорядочиванием, и каков вклад времени передачи данных между таблицами в общее время выполнения запроса.

На рис. 6, 7 приведены результаты выполнения планов, построенных алгоритмами декомпозиции для запросов типа «декартово произведение», «соединение с условием» и «смешанное соединение», для больших и малых объемов данных при соединении через локальную сеть. Время выполнения плана, построенного алгоритмом с переупорядочиванием, не во всех случаях показало уменьшение времени выполнения. Так, для малых объемов данных не выявлено снижение времени выполнения. Для больших объемов данных выполнение плана, построенного алгоритмом декомпозиции с переупорядочиванием, занимало меньше времени, чем выполнение плана, построенного алгоритмом декомпозиции левым выводом. При условии, что максимальное увеличение времени построения плана алгоритмом с перестановками по сравнению с левым выводом порядка 0,1 с, а максимальное уменьшение времени выполнения порядка 53 с. При таком значительном выигрыше, с учетом малых затрат для получения этого выигрыша, алгоритм декомпозиции с перестановками является более предпочтительным для построения планов выполнения запросов к гетерогенным СУБД больших объемов, чем алгоритм декомпозиции левым выводом. Для малых объемов может наблюдаться незначительное увеличение времени выполнения.

На рис. 8, 9 приведены результаты выполнения планов, построенных алгоритмами декомпозиции для запросов типа «декартово произведение», «соединение с условием» и «смешанное соединение», для больших и малых объемов данных при соединении через глобальную сеть. Нестабильная скорость соединения и задержки не оказывают значимого воздействия на разницу во времени выполнения.

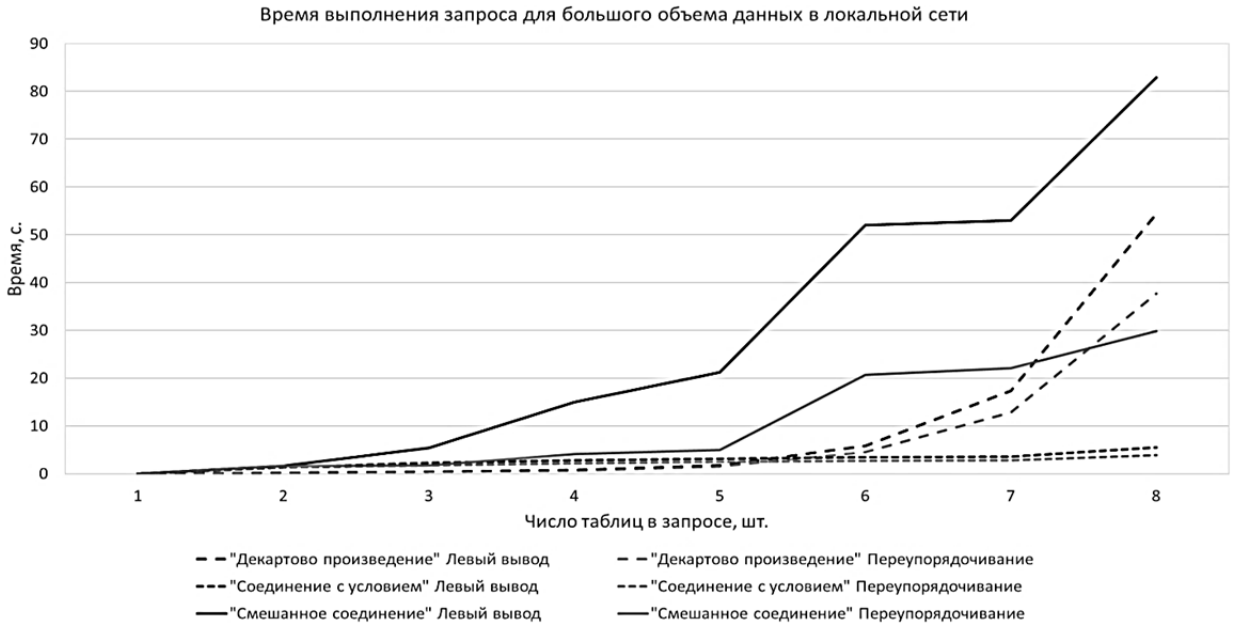

Рис. 6. Зависимость времени выполнения плана от числа таблиц в запросе для большого объема данных для запросов типа «декартово произведение», «соединение с условием» и «смешанное соединение» в локальной сети

Fig. 6. Dependence of the plan execution time on the tables count in the query for a large amount of data for queries such as «Cartesian product», «conditional join» and «mixed join» in the local area network

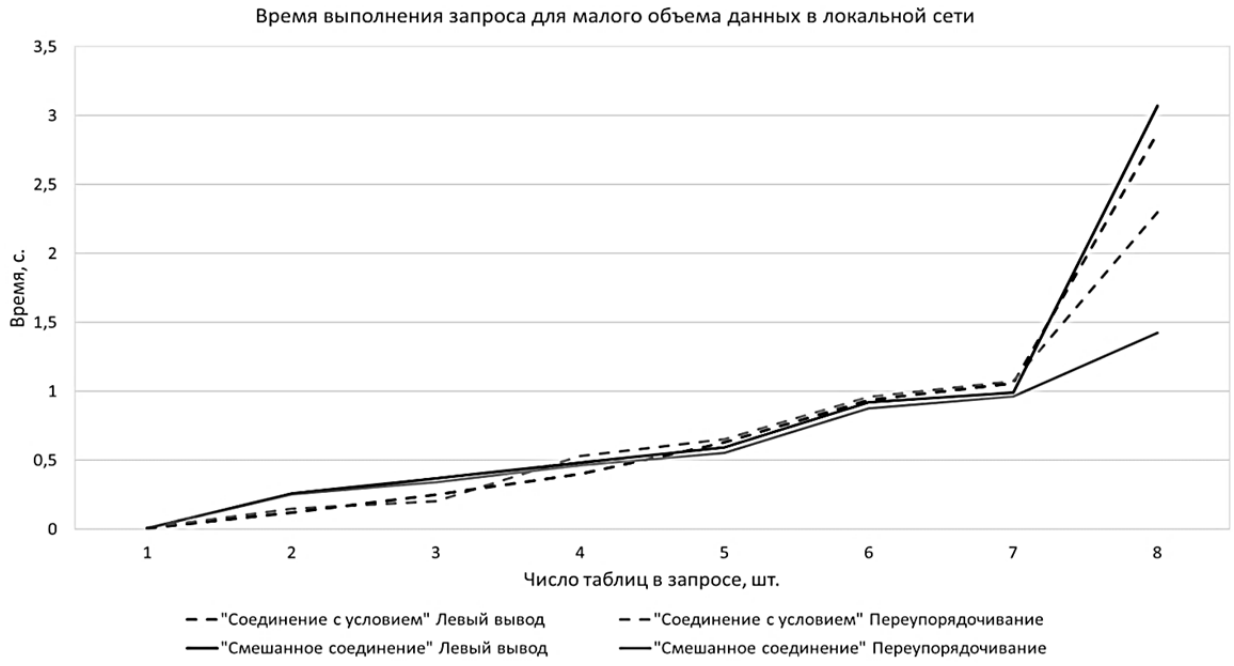

Рис. 7. Зависимость времени выполнения плана от числа таблиц в запросе для малого объема данных для запросов типа «декартово произведение», «соединение с условием» и «смешанное соединение» в локальной сети

Fig. 7. Dependence of the plan execution time on the tables count in the query for a small amount of data for queries such as «Cartesian product», «conditional join» and «mixed join» in the local area network

Рис. 8. Зависимость времени выполнения плана от числа таблиц в запросе для большого объема данных для запросов типа «декартово произведение», «соединение с условием» и «смешанное соединение» в глобальной сети

Fig. 8. Dependence of the plan execution time on the tables count in the query for a large amount of data for queries such as «Cartesian product», «conditional join» and «mixed join» in the wide area network

Рис. 9. Зависимость времени выполнения плана от числа таблиц в запросе для малого объема данных для запросов типа «декартово произведение», «соединение с условием» и «смешанное соединение» в глобальной сети

Fig. 9. Dependence of the plan execution time on the tables count in the query for a small amount of data for queries such as «Cartesian product», «conditional join» and «mixed join» in the wide area network

Рис. 10. Зависимость времени соединения таблиц от числа таблиц для большого объема данных для запросов типа «смешанное соединение» для СУБД MySQL и СУБД PosgreSQL

Fig. 10. Dependence of the tables join time on the tables count in the query for a large amount of data for a «mixed join» queries using PostgreSQL and MySQL DBMS

Эксперимент 3. Исследовать время соединения таблиц больших объемов для запросов из 1–5 таблиц на выборку, из систем управления базами данных MySQL, PostgreSQL, расположенных в локальной сети, для трех типов запросов: «декартово произведение», «соединение по правилу», «смешанные соединения».

Постановка эксперимента 3. Для баз данных больших объемов, расположенных в локальной сети, выполнить план соединения запросов смешанного типа от одной до пяти таблиц только к СУБД MySQL и только к СУБД PostgreSQL.

На рис. 10 приведены результаты измерения времени непосредственно соединения таблиц. Время соединения таблиц только из СУБД PostgreSQL значительно ниже, чем время соединения таблиц из MySQL. Учитывая, что запросы содержат таблицы, присутствующие на обоих серверах и заполненные идентичными данными, время соединений должно быть равным. Единственная причина такого расхождения кроется в преобразовании типов. Поскольку временная база данных, используемая для соединения таблиц, находится на MySQL сервере, то для локализации таблиц необходимо приводить типы данных PostgreSQL к типам MySQL. В данном проекте такое преобразование выполняется при заполнении базы данных метаданных. Соответственно, при соединении во временной базе данных аналогичных таблиц только из СУБД MySQL или только из СУБД PostgreSQL, происходит соединение таблицы с идентичной структурой, но различными типами атрибутов. Такая особенность работы приводит к возникновению разницы во времени соединения.

Заключение

Рассмотрено решение проблемы кусочной автоматизации, состоящей в фрагментации частей единой информационной среды и, как следствие, невозможности получать данные из независимых баз с заранее неизвестной схемой. Решение состоит в учете динамики семантической схемы данных.

Целью являлась разработка и исследование алгоритмов построения интегрированных запросов на выборку данных к гетерогенным распределенным системам управления базами данных. Для проверки работоспособности и эффективности предложен-

ного алгоритма разработан макет программного обеспечения промежуточного слоя, обеспечивающий хранение метаинформации объединяемых баз данных и реализующий интерпретатор запросов с возможностью их декомпозиции и выполнения.

В качестве метода выполнения запросов на выборку данных к гетерогенным СУБД реализована технология централизованной обработки. Она заключается в гомогенизации данных гетерогенных источников, объединении их в единое хранилище и выполнении запроса в этом хранилище. Метод выполнения запроса содержит четыре алгоритма: алгоритм динамической проверки, алгоритм декомпозиции левым выводом, алгоритм декомпозиции с переупорядочиванием и алгоритм выполнения плана. Алгоритм динамической проверки осуществляет динамическую синтаксическую проверку корректности запроса. Алгоритм декомпозиции левым выводом производит построение плана на основе левого обхода дерева соединений запроса. Алгоритм декомпозиции с переупорядочиванием изменяет порядок соединений в запросе на порядок с наименьшей оценкой. Алгоритм выполнения плана производит ленивое выполнение плана запроса, построенного алгоритмами декомпозиции.

Для реализации технологии централизованной обработки разработана архитектура системы управления выполнением запроса, состоящая из модулей, реализующих алгоритмы. Структура системы обеспечивает варианты размещения внешних компонентов как в локальной сети, так и в глобальной сети Интернет. Для хранения метаданных используется технологическая база данных метаданных.

Для исследования эффективности поставлены три эксперимента. Первый – для выявления различий во времени построения плана для двух алгоритмов декомпозиции левым выводом и перестановками для построения плана для трех типов запросов на выборку. Второй – для выявления различий времени выполнения планов запросов, построенных двумя алгоритмами, для локальной и глобальной сетей. Третий – для выявления различий во времени соединения таблиц для систем управления базами данных MySQL и PostgreSQL.

В результате проведения экспериментов установлено, что при сравнимом времени построения плана алгоритмом декомпозиции левым выводом и алгоритмом декомпозиции с переупорядочиванием время выполнения запроса и для локальной, и для глобальной сети время выполнения плана, построенного алгоритмом декомпозиции с переупорядочиванием, всегда меньше для больших объемов и совпадает для малых объемов для исследуемых типов запросов «декартово произведение», «соединение с условием» и «смешанное соединение» при различном числе таблиц в запросах.

При анализе результатов эксперимента по сравнению СУБД MySQL и СУБД PostgreSQL показано, что время соединения во временной базе данных зависит от трансляции типов данных из удаленных гетерогенных СУБД в типы данных СУБД, где располагается временная база данных.

Работа подготовлена в ходе реализации комплексного проекта в рамках Постановления Правительства РФ от 09.04.2010 № 218 при финансовой поддержке Министерства образования и науки РФ. Договор № 03.G25.31.0259 от 28.04.2017.

СПИСОК ЛИТЕРАТУРЫ

1. **Shin J.H., Rusu F., Suhan A.** Exact selectivity computation for modern in-memory database query optimization // arXiv preprint arXiv:1901.01488. 2019.

2. **Marcus R., Papaemmanouil O.** Towards a hands-free query optimizer through deep learning // arXiv preprint arXiv:1809.10212. 2018.

3. **Krishnan S., et al.** Learning to optimize join queries with deep reinforcement learning // arXiv preprint arXiv:1808.03196. 2018.

4. **Vaidehi V., Devi D.S.** Distributed database management and join of multiple data streams in wireless sensor network using querying techniques // 2011 Internat. Conf. on Recent Trends in Information Technology. IEEE, 2011. Pp. 594–599.

5. **Chen F.X., Xie X.S.** Application on query of distributed database based on improved genetic algorithm // Applied Mechanics and Materials. Trans Tech Publications, 2014. Vol. 556. Pp. 4617–4621.

6. **Shao Hua Liu, Xing Xu** Distributed database query based on improved genetic algorithm // Proc. of the 2016 3rd Internat. Conf. on Information Science and Control Engineering. IEEE, 2016. Pp. 348–351.

7. **Gharibi W., Mousa A.** Query optimization based on time scheduling approach // East-West Design & Test Symposium 2013. IEEE, 2013. Pp. 1–7.

8. **Begoli E., et al.** Apache calcite: A foundational framework for optimized query processing over heterogeneous data sources // Proc. of the 2018 Internat. Conf. on Management of Data. ACM, 2018. Pp. 221–230.

9. **Hameurlain A., Morvan F.** Evolution of query optimization methods // Transactions on Large-Scale Data-and Knowledge-Centered Systems I. Springer, Berlin, Heidelberg, 2009. Pp. 211–242.

10. Документация высокопроизводительной системы выполнения распределенных запросов Presto // URL: https://prestosql.io/docs/current/optimizer/statistics.html (Дата обращения: 20.04.2019).

11. **Кропотин А.А., Ивашко А.Г.** Реализация метода идентификации семантических конфликтов метаданных и несоответствия интегрируемых данных исходя из их семантического описания // Вестник ТюмГУ: Физико-математическое моделирование. Нефть, газ, энергетика. 2017. Т. 3. № 2. DOI: 10.21684/2411-7978-2017-3-2-115-127

12. **Павлов С.В., Ефремова О.А.** Онтологическая модель интеграции разнородных по структуре и тематике пространственных баз данных в единую региональную базу данных // Онтология проектирования. 2017. Т. 7. № 3(25). DOI: 10.18287/2223-9537-2017-7-3-323-333

13. **Герасимов А.Н. и др.** Методы автоматизированного извлечения метаданных научных публикаций для библиографических и реферативных баз цитирования // Интернет и современное общество: Труды объединенной научной конференции. 2016. С. 41–48.

14. **Петрова Н.В. и др.** Разработка базы метаданных «Информационные ресурсы единой геофизической службы РАН» // Современные методы обработки и интерпретации сейсмологических данных. 2018. С. 199–203.

15. **Зыкин В.С., Цымблер М.Л.** Обновление многотабличных представлений на основе коммутативных преобразований базы данных // Вестник ЮУрГУ. Вычислительная математика и информатика. 2019. Т. 8. № 2. С. 92–106.

16. **Чубаров Д.Л., Добрецов Н.Н., Кихтенко В.А.** Отображение модели данных NETCDF в реляционную модель для работы с коллекциями данных дистанционного зондирования // Обработка пространственных данных в задачах мониторинга природных и антропогенных процессов. 2017. С. 324–328.

17. **Пестунов И.А., Федотов А.М., Жижимов О.Л.** Структура сервисов управления метаданными для разнородных информационных систем. 2015.

18. **Рублев В.С.** Отношение истории и динамика схем баз данных СУБД DIM // Моделирование и анализ информационных систем. 2015. Т. 19. № 2. С. 97–108.

19. **Иванова Е.В., Соколинский Л.Б.** Декомпозиция операций пересечения и соединения на основе доменно-интервальной фрагментации колоночных индексов // Вестник Южно-Уральского государственного университета. Вычислительная математика и информатика. 2015. Т. 4. № 1. DOI: 10.14529/cmse150104

20. **Sangkla K., Seresangtakul P.** Information integration of heterogeneous medical database systems using metadata // 21st Internat. Computer Science and Engineering Conf. Bangkok, 2017. Pp. 1–5.

Статья поступила в редакцию 04.05.2019.

REFERENCES

1. **Shin J.H., Rusu F., Suhan A.** Exact selectivity computation for modern in-memory database query optimization. *arXiv preprint arXiv:1901.01488*. 2019.

2. **Marcus R., Papaemmanouil O.** Towards a hands-free query optimizer through deep learning. *arXiv preprint arXiv:1809.10212*. 2018.

3. **Krishnan S., et al.** Learning to optimize join queries with deep reinforcement learning. *arXiv preprint arXiv:1808.03196*. 2018.

4. **Vaidehi V., Devi D.S.** Distributed database management and join of multiple data streams in wireless sensor network using querying techniques. *2011 International Conference on Recent Trends in Information Technology*. IEEE, 2011, Pp. 594-599.

5. **Chen F.X., Xie X.S.** Application on query of distributed database based on improved genetic algorithm. *Applied Mechanics and Materials. Trans Tech Publications*, 2014, Vol. 556, Pp. 4617–4621.

6. **Shao Hua Liu, Xing Xu** Distributed database query based on improved genetic algorithm. *Proceedings of the 2016 3rd International Conference on Information Science and Control Engineering*. IEEE, 2016. Pp. 348–351.

7. **Gharibi W., Mousa A.** Query optimization based on time scheduling approach. *East-West Design & Test Symposium 2013*. IEEE, 2013, Pp. 1–7.

8. **Begoli E., et al.** Apache calcite: A foundational framework for optimized query processing over heterogeneous data sources. *Proceedings of the 2018 International Conference on Management of Data*. ACM, 2018, Pp. 221–230.

9. **Hameurlain A., Morvan F.** Evolution of query optimization methods. *Transactions on Large-*

Scale Data-and Knowledge-Centered Systems I. Springer, Berlin, Heidelberg, 2009, Pp. 211–242.

10. Dokumentatsiya vysokoproizvoditelnoy sistemy vypolneniya raspredelennykh zaprosov Presto [Presto 318 Documentation]. Available: *https://prestosql.io/docs/current/optimizer/statistics.html* (Accessed: 20.04.2019). (rus)

11. **Kropotin A.A., Ivashko A.G.** Realizatsiya metoda identifikatsii semanticheskikh konfliktov metadannykh i nesootvetstviya integriruyemykh dannykh iskhodya iz ikh semanticheskogo opisaniya [Implementation of a method for identifying semantic conflicts of metadata and inconsistency of merging data based on their semantic description]. *Vestnik TyumGU: Fiziko-matematicheskoye modelirovaniye. Neft, gaz, energetika* [*Tyumen State University Herald. Physical and Mathematical Modeling. Oil, Gas, Energy*], 2017, Vol. 3, No. 2. (rus) DOI: 10.21684/2411-7978-2017-3-2-115-127

12. **Pavlov S.V., Yefremova O.A.** Ontologicheskaya model integratsii raznorodnykh po strukture i tematike prostranstvennykh baz dannykh v yedinuyu regionalnuyu bazu dannykh [Ontological model for integration of structurally heterogeneous spatial databases of various subject areas into a uniform regional database]. *Ontologiya proyektirovaniya* [*Ontology of designing*], 2017, Vol. 7, No. 3 (25). (rus) DOI: 10.18287/2223-9537-2017-7-3-323-333

13. **Gerasimov A.N., et al.** Metody avtomatizirovannogo izvlecheniya metadannykh nauchnykh publikatsiy dlya bibliograficheskikh i referativnykh baz tsitirovaniya [Automated methods of metadata extraction from scientific publications for bibliographic databases]. *Trudy obyedinennoy nauchnoy konferentsii «Internet i sovremennoye obshchestvo»* [Proceedings of the Joint Scientific Conference on *Internet and Modern Society*], 2016, Pp. 41–48. (rus)

14. **Petrova N.V., et al.** Razrabotka bazy metadannykh Informatsionnye Resursy Yedinoy Geofizicheskoy Sluzhby RAN [Development of the metadata base «Information resources of the unified geophysical service of the RAS»]. *Sovremennye Metody Obrabotki i Interpretatsii Seysmologicheskikh Dannykh* [*Proceedings at the XIII International Seismological Workshop Modern Methods and Interpretation of Seismological Data*], 2018, Pp. 199–203. (rus)

15. **Zykin V.S., Tsymbler M.L.** Obnovleniye mnogotablichnykh predstavleniy na osnove kommutativnykh preobrazovaniy bazy dannykh [Updating of multi-table views based on commutative database transformations]. *VestnikYuUrGU. Vychislitelnaya matematika i informatika* [*Bulletin of the South Ural State University. Computational Mathematics and Software Engineering*], 2019, Vol. 8, No. 2, Pp. 92–106. (rus)

16. **Chubarov D.L., Dobretsov N.N., Kikhtenko V.A.** Otobrazheniye modeli dannykh NETCDF v relyatsionnuyu model dlya raboty s kollektsiyami dannykh distantsionnogo zondirovaniya [Mapping the NETCDF data model to a relational model for working with remote sensing data collections]. *Obrabotka prostranstvennykh dannykh v zadachakh monitoringa prirodnykh i antropogennykh protsessov* [*Spatial Processing in Tasks of Monitoring Natural and Anthropogenic Processes*], 2017, Pp. 324–328. (rus)

17. **Pestunov I.A., Fedotov A.M., Zhizhimov O.L.** *Struktura servisov upravleniya metadannymi dlya raznorodnykh informatsionnykh sistem* [*The structure of metadata management services for heterogeneous information systems*]. 2015. (rus)

18. **Rublev V.S.** Otnosheniye istorii i dinamika skhem baz dannykh SUBD DIM [Evolution of DBMS DIM database schemes]. *Modelirovaniye i analiz informatsionnykh system* [*Model. Anal. Inform. Sist.*], 2015, Vol. 19, No. 2, Pp. 97–108. (rus)

19. **Ivanova Ye.V., Sokolinskiy L.B.** Dekompozitsiya operatsiy peresecheniya i soyedineniya na osnove domenno-intervalnoy fragmentatsii kolonochnykh indeksov [Decomposition of intersection and join operations based on the domain-interval fragmented column indices]. *Vestnik Yuzhno-Uralskogo gosudarstvennogo universiteta. Vychislitelnaya matematika i informatika* [*Bulletin of the South Ural State University. Computational Mathematics and Software Engineering*], 2015, Vol. 4, No. 1. (rus) DOI: 10.14529/cmse150104

20. **Sangkla K., Seresangtakul P.** Information integration of heterogeneous medical database systems using metadata. *21st International Computer Science and Engineering Conference*, Bangkok, 2017, Pp. 1–5.

Received 04.05.2019.

СВЕДЕНИЯ ОБ АВТОРАХ / THE AUTHORS

ПОПОВ Сергей Геннадьевич
POPOV Sergey G.
E-mail: popovserge@spbstu.ru

ПУРИЙ Александр Александрович
PURII Alexandr A.
E-mail: purii.alexander@gmail.com