

РАЗРАБОТКА ПАРАМЕТРИЧЕСКОЙ МОДЕЛИ КОНКУРЕНТНОГО ДОСТУПА В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ МЕТОДОМ СЛУЧАЙНОГО ЛЕСА

Д. Д. Громей, Е. В. Лебеденко, Д. А. Николаев, Т. С. Рожкова

Розглянуто проблему моделювання часу виконання запитів в автономних реляційних базах даних з конкурентними запитами. Відзначено недоліки існуючих підходів, які ігнорують витрат на частку послідовних операцій при кооперативному доступі до даних в ієрархії пам'яті. Розглянуто питання застосування умовної вартості виконання складових операцій плану запиту, замість розрахунку передбачуваного часу обчислень.

Запропоновано спосіб формального формування прецедентів навчальної вибірки, а також підхід до побудови регресійної моделі. Розроблена модифікація методу машинного навчання випадковий ліс застосовується для розрахунку часів виконання запитів за їхніми текстами та тимчасовими позначками початку, тривалості виконання.

Розроблена параметрична модель конкурентного доступу до даних необхідна для отримання точних оцінок часу виконання запитів при використанні паралельних обчислень. Моделі з подібними характеристиками потрібні в рішенні задач автоматизації управління фізичною схемою даних, створення СУБД, що самовизначаються. Ключовими відмінностями від існуючих підходів стали використання часу виконання запитів в якості цільового значення, обліку значень предикатів і взаємовпливу паралельно виконуваних запитів.

Для підтвердження отриманих результатів була використана імітаційна модель на базі широко відомого тесту TPC-C. В якості функції втрат, з урахуванням регресійної природи моделі, було використано відношення суми модулів різниці фактичних і одержуваних часів до фактичних часів. Сама перевірка проводилася за контрольною вибіркою, що сформована за зростаючою довжиною навчання на відкладених даних. В ході проведених досліджень була доведена можливість застосування методу машинного навчання випадковий ліс для обробки статистичних даних виконання SQL запитів. Отриманий результат свідчить про перспективність такого підходу і дозволяє отримувати параметричні моделі конкурентної обробки запитів

Ключові слова: автономні систем управління базами даних, бази даних, що самовизначаються, випадковий ліс, конкурентний доступ, паралельні обчислення в реляційних систем управління базами даних

1. Введение

Последние десятилетия осязаемо нарастает информатизация общества, увеличивается степень интеграции информационных технологий во всех сферах человеческой жизни. Увеличивается их вклад в научно-техническом про-

грессе, взрывообразно растут объемы хранимых и обрабатываемых данных. Всё новые и новые направления охватываются внедрением информационных систем и основой большинства из них выступают системы управления базами данных, как совокупность средств управления представленной в объективной форме информации.

С момента создания самых первых СУБД общего назначения и по сегодняшний день основное требование к ним – работы в разных условиях. Программное обеспечение базы данных должно обеспечивать сохранение работоспособности и выполнения своих функций в условиях неопределенности конечных условий работы с адекватным ответом на их изменения. При этом СУБД должны работать оптимальным образом при различных характеристиках обрабатываемых данных и аппаратно-программных средств. В частности, объем хранимых данных может колебаться от сотен килобайт до десятков экзбайт, а число логических ядер от одного до нескольких сотен. При этом остаётся неизвестным самый важный фактор – информация, с какой структурой и статистическими характеристиками будет находиться в базе данных. Именно физическая структура данных является определяющей в выборе оптимального способа транслировать поступающие в систему запросы в конечный набор операций над хранимыми данными. И наибольший прогресс в этом направлении достигнут с реляционными базами данных с конкурентным доступом, которые одновременно является и одними из самых распространённых.

Но и по сей день ключевая роль в адаптации работы СУБД к конкретной базе данных возлагается на человека – администратора базы данных. Вместе с тем, число эксплуатируемых баз данных существенно превысило число администраторов, а их сложность такова, что даже профессиональный и опытные специалист вынужден пользоваться вспомогательными средствами. Людям требуется использовать не опыт или строгие математические и формализованные представления, а эксперимент над конкретной базой данных. Вопрос создания автономных баз данных или иначе – самоопределяемых СУБД – становится всё более явным вызовом на пути дальнейшего развития. Создание самоопределяемых СУБД требует параметрических моделей, пригодных для оценки времён выполнения параллельных запросов. Подобные оценки необходимы в задачах автоматизации формирования распределения данных в памяти ЭВМ. И в отличие от существующих должны учитывать конкуренцию между запросами, их предикаты, а также иметь большую точность. Известные методы оценки стоимости вычислений используются на этапе выбора плана выполнения запроса – затраченное на них время является частью временных затрат выполнения запроса. Существующие промышленные реализации работают в режиме мягкого реального времени и конечным считается любой промежуточный результат, достигнутый при исчерпании резерва времени на выбор квазиоптимального плана запроса. В свою очередь, автономные СУБД рассчитаны на выполнение операций изменения распределения данных в фоновом режиме, параллельно иным вычислениям. Возникающее различие в способе применения снимает ограничения на время обработки данных. Становится возможным разрабатывать новые методы, используя ранее недоступные подходы, и наращивать точ-

ность рассчитываемых оценок за счет более высокой вычислительной сложности. Возникает необходимость в разработке новых моделей конкурентного доступа к данным в реляционных базах данных.

2. Анализ литературных данных и постановка проблемы

В основе работы автономной, самоопределяемой базы данных лежит изменение текущего состояния внутренних структур хранения – смены распределения данных. Подобное изменение должно быть основано на оценке оптимальности существующего распределения, по отношению ко всему многообразию его альтернатив [1]. Эти требования по своей сути достаточно близки к более ранним работам в области разработки СУБД – необходимости оценки возможных планов выполнения запросов оптимизатором СУБД [2]. Поэтому в предметной области достаточно много смежных работ, с теми или иными ограничениями, решающими целевую проблему оценки стоимости выполнения запросов.

Для современных СУБД применяются подходы, описанные [2–4], по оценке селективности разных планов выполнения запроса. Обобщенная мысль таких работ сводится к идее неравномерного распределения значений на множестве данных, когда предварительная оценка предикатов запроса позволяет установить его соответствии значимой или пренебрежительно малой части строк целевой таблицы. Для мультиверсионных СУБД подобная оценка, как правило, является основным критерием к применению вспомогательных структур данных. К примеру, может быть выбран один из индексов или наоборот произойти отказ от них в пользу полного сканирования всех строк таблицы. В последнем случае подобная оценка используется для минимизации издержек косвенного доступа к данным. В СУБД, основанных на блокировках, эта оценка также применяется для решения задачи упреждающего выбора уровня эскалации блокировки. Ключевые недостатки этого направления являются и целью его совершенствования [3, 4] – невозможность сопоставления предварительно рассчитанной статистики всему многообразию возможных условий поступающих запросов, что ограничивает её применимость в принципе. В случае параллельных вычислений так же провоцируется фундаментальная проблема невозможности учитывать кооперацию совместного доступа к данным между одновременно выполняемыми запросами. Подходы оценки селективности запросов по предварительно рассчитанным статистическим данным, безусловно, важная часть современных СУБД. Однако в силу своих ограничений подобные числовые оценки используются в большей степени как вспомогательное средство.

В большинстве современных СУБД используются подходы, основанные на алгоритмах Грэфа [6], Басу [7] и их идейных эквивалентах. Общая мысль этого направления разработок заключается во введении стоимостных функций, выполняющих условную оценку стоимости выполнения того или иного плана выполнения запроса в некоторых абстрактных единицах. Всякому объекту, участвующему в ходу выполнения плана запроса, сопоставляются множества способов доступа, а их комбинациям, в свою очередь, некоторые функции оценки стоимости выбора того или иного пути исполнения. Одной из первых реализаций, получивших распространение в производстве, стала реализация из IBM

DB2 [8]. Отличительной чертой предложенной реализации стали высокая адаптивность оценивания операций чтения или модификации объектов схемы данных в зависимости от захваченных над структурами базы данных блокировок. Предложенная IBM реализация позволила учитывать конкурентность и корректировать выбор плана выполнения запроса в зависимости от уже выполняемых запросов.

Основная проблема методов, основанных на оценках стоимости выполнения, – используемые единицы измерения. Конечная величина, оценивающая стоимость выполнения множества запросов, – время выполнения каждого. Применение суррогатных единиц измерения обусловлено высокой сложностью прогнозирования времени и приводит к утрате точности [1]. К сожалению, эта стоимость имеет нелинейную взаимосвязь непосредственно со временем выполнения, и не позволяет эффективно его оценивать. Более того, возможны ситуации, когда запрос с меньшей стоимостью будет выполняться дольше из-за линейного характера расчетов конечной стоимости, при нелинейном характере выполняемых операций [4, 8]. Стоимостные подходы, конечно хорошее решение в задачах поиска плана выполнения запроса, ввиду жестких ограничений по времени и высокой частоты решения этой задачи – для каждого запроса. Но низкая точность делает малопригодными их применение в задачах самоопределения баз данных.

Ряд работ [9, 10] описывает методы оценивания работы базы данных на основе имитационного моделирования. Такие подходы не нашли практического применения в промышленных системах ввиду как высокой стоимости, так и технической сложности создания адекватных моделей, а также в силу ограниченности получаемых результатов. Большинство предлагаемых имитационных моделей не учитывают предикаты обрабатываемых запросов или предлагают оценку совокупных характеристик моделируемой системы. Из-за подобных ограничений методы имитационного моделирования полагаются малопригодными для применения в перспективных автономных СУБД.

Достаточно обособленно стоит класс методов оценки стоимости выполнения запросов как части параллельных вычислений. Описываемые в ряде работ подходы основываются преимущественно на теоретико-множественных моделях кооперативного доступа к линейно представленной общей памяти [11] или же ресурсам, как иерархически организованным и логически обособленным совокупностям областей памяти [12]. Пример конкурентного доступа к данным представлен на рис. 1.

Однако подобные работы носят в большей степени теоретический характер, основное предполагаемое в них ограничение – линейная однородность памяти как ресурса ЭВМ. В то же время, существующие процессорные архитектуры предполагают обратные характеристики памяти. Возникает нелинейность доступа к памяти, как суперпозиция взаимодействия кэшей разных уровней, распределения арифметически-логических устройств и времени шин между параллельно работающими конвейерами команд. На время доступа влияют сами виды памяти – буферизированная, небуферизированная или регистровая. Так же возникает взаимное влияние последовательности обрабатываемых операций

доступа к памяти. Предлагаемые методы допустимы в случае оценки издержек кооперативного выполнения запросов при их эквивалентном времени исчисления для последовательного выполнения. В то же время такие модели выносят в ограничения вопрос обработки разнородных запросов. В существующих информационно-вычислительных системах подобная ситуация редка. Как итог – получаемые модели вычисления запросов либо малопригодны, либо требуют существенной доработки для промышленного применения.

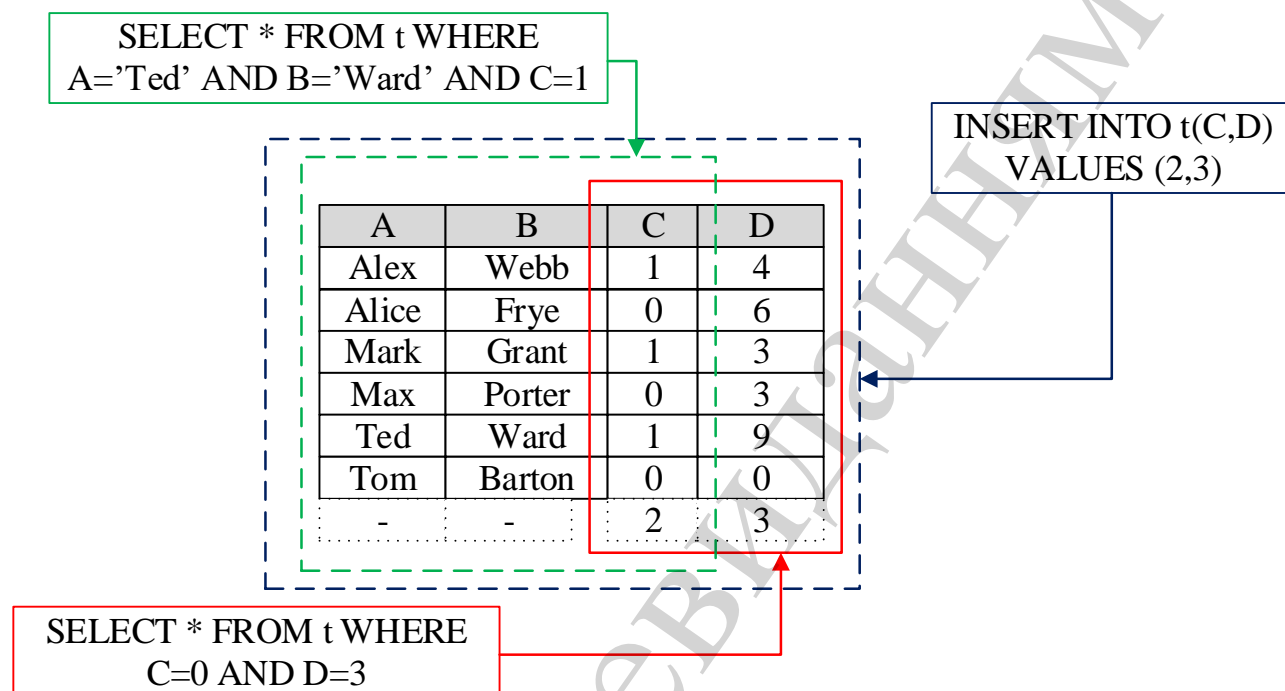


Рис. 1. Пример конкурентного доступа к данным в реляционной СУБД

Работы [13, 14] посвящены прикладному применению подходов из теории обучения машин для разработки автономных СУБД. Работы рассматривают статистические характеристики множества поступивших в систему запросов за некоторый временной интервал. Для всякого запроса извлекаются предикаты и обобщаются во множество шаблонов с меньшим кардинальным числом, по исключению из условий конкретных значений. Полученные множества кластеризуются и подаются на вход рекуррентной нейронной сети с LSTM блоками долговременной памяти, что позволяет обобщить периодичность и повторяемость трендов запросов. Итоговые оценки применяются для автоматического выбора индексов в работе [13] и физического представления данных [14]. Работы в первую очередь доказывают принципиальную применимость методов из теории обучения машин для построения автономных СУБД. В предлагаемых моделях отсутствует оценка издержек конкурентного доступа к данным, фактически предлагаемый авторами прототип в большей степени ориентирован на последовательную обработку запросов.

Модель, построенная на базе нейронной сети, оценивает только объекты схемы данных, указываемые в предикатах запросов, игнорируются статистиче-

ские характеристики данных из работ [2, 8, 9]. Игнорируется время выполнения запросов, что провоцирует грубые ошибки в условиях обработки неоднородных запросов. К примеру, предположим, что у таблицы имеется два поля – при этом множество текстов запросов с упоминанием первого поля имеет большее кардинальное число, а второе большую сумму времен выполнения. Предлагаемый метод выберет первое поле в первую очередь для автоматического формирования индекса, в тоже время, при прочих равных, именно индекс второго поля в большей степени влияет на целевой параметр – время выполнения запросов. Также предлагаемый подход не применим для операций сегментации данных, поскольку не позволяет выполнить выбор ключа сегментации. На рис. 2 представлен пример сегментирования таблицы по полю C для параллельного выполнения нескольких запросов.

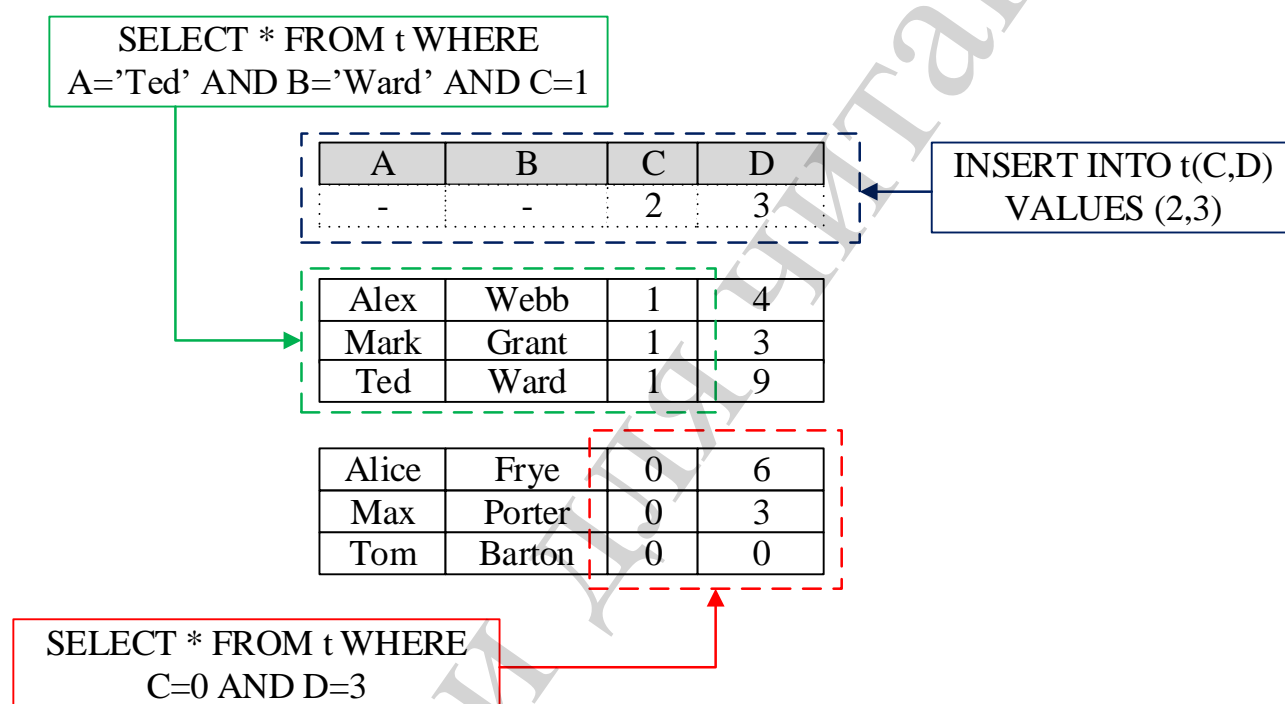


Рис. 2. Пример сегментации таблицы для параллельных вычислений

Следует отметить, что текущее положение, с ограниченностью применения существующих подходов обуславливается историческими причинами, этапностью развития технологической базы и протеканием процессов информатизации общества. Во времена появления СУБД существовали в числе десятков и сотен, были зачастую экспериментальными образцами и дорого стоили. Удельная стоимость человеческого труда, по ручному решению задач оптимизации баз данных, была пренебрежительна по сравнению со стоимостью задействованных технических средств. А непосредственная сложность работы делала бессмысленной автоматизацию. Десятилетия роста числа экземпляров баз данных и их разнообразия, увеличение объемов хранимых данных и удешевления аппаратных средств привели к сегодняшнему, уже обратному положению. И отсутствие самоопределяемых, автономных баз данных, провоцирует процесс

стагнации, когда СУБД работают зачастую в десятки и сотни раз медленней, чем можно было бы достичь на тех же аппаратных ресурсах.

3. Цель и задачи исследования

Целью исследования является разработка параметрической модели конкурентного доступа в реляционных базах данных, позволяющего учитывать предикаты и взаимовлияние параллельно выполняемых запросов методами машинного обучения. Это позволит применять статистику работы реляционной СУБД для определения кортежей логических объектов и задаваемых запросами условий доступа, ранжировать их по наибольшему вкладу во время параллельной обработки данных. Получаемые подобным образом упорядоченные множества объектов могут применяться в алгоритмах управления физической схемой хранения данных, адаптирующих её под характерную для конкретной базы данных нагрузку.

Для достижения поставленной цели необходимо решить следующие частные задачи:

- разработать теоретико-множественную модель процесса управления конкурентным доступом запросов в реляционных базах данных;
- разработать метод расчета параметрической модели доступа к данным в реляционной СУБД с использованием машинного обучения;
- провести вычислительные эксперименты, подтверждающие результативность разработанной модели и метода, с использованием имитационной модели процесса конкурентной обработки запросов в реляционной базе данных.

4. Материалы и методы исследования процесса конкурентного доступа в реляционных СУБД

4.1. Теоретико-множественная модель процесса управления конкурентным доступом запросов в СУБД реляционного типа

В общем случае обработка запроса в реляционной СУБД представляет собой некоторое отображение

$$DBMS : Q \times S \xrightarrow{CPU} (\mathfrak{R}, T, S'),$$

в некоторой вычислительной системе CPU . Область определения – декартово произведение над множествами допустимых запросов Q и возможных внутренних состояний базы данных S . Область значений задаётся множеством кортежей из результата исчисления запроса \mathfrak{R} , времени, затраченного на его выполнение T , а также нового внутреннего состояния базы данных S' .

В свою очередь, каждый запрос $Q = (Q^{SQL}, \{Q^{PRD}\})$ представляет кортеж из непосредственно текста запроса Q^{SQL} на языке SQL, и множества связанных предикатов Q^{PRD} , соответствующих связанным переменным запроса («bind variables»). Множество S представимо своим декартовым произведением

$$S \subseteq S_{PD} \times S_{UD} \times S_{AD} \times S_{SD}$$

частных состояний:

- S_{PD} – совокупность объектов, заданных разработчиком для отображения информации в соответствии с правилами реляционной алгебры, в частности таблицы, ограничения целостности или связи;
- S_{UD} – отображение заданной пользователями информации, непосредственно хранимые данные;
- S_{AD} – вспомогательные структуры, определяющие размещение данных в памяти и задаваемые администратором СУБД или механизмами самоопределения структуры базы данных, в том числе индексы или сегменты;
- S_{SD} – внутренние структуры ядра СУБД, используемые при обработке запросов, включая всё виды кэшей.

В решении оптимизационных задач для СУБД целевым параметром считается время T , представляющее собой сумму времен разных этапов обработки исходного запроса. И в соответствии с приведенными выражениями задача может решаться несколькими путями:

- модернизации аппаратных средств CPU ;
- изменением запросов к системе Q ;
- совершенствованием программной реализации СУБД изменением S_{PD} и S_{SD} ;
- уменьшением объема хранимых данных и совершенствованием их модели изменяя S_{UD} ;
- посредством изменения параметра S_{AD} , формируемого администратором СУБД.

В практической работе администраторы баз данных работают с уже созданными программными средствами СУБД, разработанными информационными системами, порождающими запросы Q . Как следствие, администраторы должны обеспечить оптимальное использование ресурсов CPU для заданных данных S_{UD} – соответственно параметры являются константами для решения оптимизационной задачи: $(S_{PD}, S_{UD}, S_{SD}) = \text{const}$. Таким образом, администратор, при неизменности иных параметров, должен найти такое значение S_{AD} , при котором $T \rightarrow \min$. В свою очередь цель автономной СУБД – итеративное изменение S_{AD} , при котором $\sum T \rightarrow \min$ для множества преобразований $DBMS$, за некоторый интервал работы СУБД.

Решение подобной оптимизационной задачи требует работы с функциональной зависимостью

$$DBS : Q \times S \xrightarrow{CPU} T.$$

При этом, любая современная СУБД основана на параллельных вычислениях, таким образом, что в любой момент времени в системе может обрабаты-

ваться более одного запроса, что в свою очередь определяет работу закона Амбдала-Уэра. Дополнительным ограничением представляются требования ACID, в частности необходимость изолированности запросов над набором данных S и атомарности переходов $S \rightarrow S'$. Подобные преобразования формирует долю последовательных вычислений закона и приводит к сильным взаимным зависимостям отображений DBM параллельных запросов. В системе с параллельными вычислениями выражение приобретет смысл

$$DBM : Q_t \times S_t \xrightarrow{CPU} T,$$

где Q_t – запрос, поступивший в момент времени t , а S_t – внутреннее состояние базы данных.

Создание автономных СУБД требует представления отображения DBM с точностью, достаточной для уменьшения целевого параметра $\sum T \rightarrow \min$. Однако сложность точного представления аппаратных и программных средств в аналитической форме просто исключает подобный подход даже для систем с последовательными вычислениями [9, 12]. Полученная теоретико-множественная модель позволяет формально связать целевой параметр – сумму времен обработки множества запросов с основными характеристиками рассматриваемой системы.

4. 2. Подход к формированию признакового описания запросов в реляционных СУБД

Предположим, что за некоторое время наблюдения счетного множества TN в систему с начальным состоянием S^N поступило множество запросов Q^N , каждое в свой момент дискретного времени:

$$Q^N \subseteq \{Q_t^N\}, t \in TN.$$

Соответственно, исчисление DBM привело к каскаду преобразований $S_{t0}^N \rightarrow S_{t1}^N \rightarrow \dots \rightarrow S_{\max(t)}^N$ и отождествляемому множеству T^N области значений возможного времени исполнения. Зафиксируем константы преобразования как ограничения модели: $CPU = \text{const}$, $S_{SD} = \text{const}$. Значения S_{PD} , S_{UD} могут меняться в каскаде преобразований, но не могут быть изменены при решении задачи. По своему определению, запросы только выборки данных не порождает фактической смены состояния

$$(\forall Q : Q^{SQL} \propto SELECT) S = S'.$$

При этом, согласно ряду исследований [2, 8], отношение измененной информации при иных запросах как правило много меньше её общей величины $I(S / S') \ll I(S)$. Таким образом, переход состояний происходит небольшими

итерациями и вектор расстояния между точками пространства существенно меньше максимальной возможной величины метрики в этом пространстве. Достаточно редки ситуации, когда запросы, изменяющие множество строк таблиц, преобладают по своему числу или времени выполнения над запросами изменения отдельных строк. Поэтому в общем случае можно полагать верным утверждение:

$$|DBM(CPU, Q_t, S) - DBM(CPU, Q_t, S')| \ll \left| \frac{DBM(CPU, Q_t, S) - DBM(CPU, Q_t, S^R)}{-DBM(CPU, Q_t, S^R)} \right|.$$

В выражении S^R – случайная точка пространства состояний. Таким образом, что также подтверждается рядом исследований [8, 10], за время наблюдения отображение DBM при произвольных S^N может быть представлена как

$$DBM \approx DBM^\Delta : Q_t^N \rightarrow T_t^N + \Delta N,$$

где ΔN – погрешность на интервале наблюдения, вызываемая каскадом преобразований S^N . Это важное выражение для автономной базы данных, поскольку позволяет прийти к выражению:

$$\begin{aligned} \sum_{TN} DBM(CPU, Q_t^N, S_t^N) &= T^N \approx \sum_{TN} DBM^\Delta(Q_t^N) = \\ &= T^N + \Delta N \approx \sum_{TN'} DBM(CPU, Q_t^{N'}, S_t^{N'}) = T^{N'}, \end{aligned}$$

определяющему признаковое описание запроса. Для автономной базы данных, помимо периода наблюдения TN , существует также период TN' работы после перехода $S_{AD} \rightarrow S_{AD}'$. Следовательно, $S_{t0}^{N''} = S_{t0}^N$; $Q^N = Q^{N'} \Rightarrow T^N = T^{N'}$ и при значимом числе периодов TN зависимость $Q^N \rightarrow T^N$ подчиняется центральной предельной теореме, таким образом, что изменение пренебрежительно малого подмножества элементов множества Q^N изменяет сумму времен выполнения запросов пренебрежительно малым образом. Рассмотрим временные интервалы наблюдения и последующий за ним, модуль разницы сумм отображения DBM^Δ множеств запросов Q_t^N и $Q_t^{N'}$. Можно утверждать, что отображение DBM тех же подмножеств меньше, чем аналогичные модули для пар (Q_t^N, Q_t^R) и $(Q_t^{N'}, Q_t^R)$, где Q_t^R – случайное подмножество Q , не менее чем в половине случаев. Это аксиоматическое выражение определяет естественное ограничение в работе автономных баз данных, и является для них ключевой аксиомой. В соответствии с этим выражением и определяемым им допущением, здесь и далее возможно рассматривать DBM^Δ , в качестве приближения DBM . Имеющейся между ними погрешностью можно пренебречь, поскольку величина

на этой погрешности сопоставима с величиной погрешности, возникающей вследствие несоответствия множеств Q_i^N и $Q_i^{N'}$. Такой переход важен, поскольку частное состояние S_i^N , представляемое в виде двоичного кода, может требовать десятки гигабайт или терабайт для своей записи, а работа с переменными подобных размерностей лишена практичности.

4. 3. Метод расчета параметров теоретико-множественной модели процесса управления конкурентным доступом запросов в СУБД реляционного типа на основе алгоритма случайного леса

Для представления искомой функциональной зависимости в данной работе предлагается использовать подходы из теории обучения машин. Будем считать множество пар (Q_i^N, T_i^N) , связанных отображением DBM^A , некоторой обучающей выборкой. Поскольку T_i^N – время выполнения запроса, задача относится к классу задач восстановления числовой регрессии. Зачастую, в практических задачах машинного обучения основные закономерности восстанавливаемых зависимостей не известны. Выбор подмножества используемых признаков и применяемых алгоритмов обуславливается изучением основных статистических характеристик и поиском корреляций признаков друг с другом. В общем случае, основные закономерности восстанавливаемых зависимостей не известны, поэтому поиск разумного подмножества признаков и конкретного метода машинного обучения выполняется экспериментально, основываясь на изучении статистических характеристик и численных экспериментах. Но подобный подход не приемлем к решаемой задаче, поскольку (Q_i^N, T_i^N) , CPU , S – будут изменяться в зависимости от конкретной СУБД, неопределенным образом меняя результаты численных экспериментов. Но в то же время, в рассматриваемой задаче известны ключевые закономерности, связывающие отдельные характеристики переменных Q_i^R , S_i^R , T_i^R для последовательных запросов, а также общий характер изменения – это зависимости от переменной CPU и параллельно выполняемых запросов. Докажем это утверждение, введём определение времени выполнения запроса:

$$T = T_{net} + T_{ast} + T_{opt} + T_{eq} + T_{da},$$

где T_{net} – передачи запроса от информационной системы к ядру СУБД; T_{ast} – работы синтаксического и лексического анализатора по разбору запроса; T_{opt} – работы оптимизатора на выборку из имеющихся или поиск нового плана выполнения запроса; T_{eq} – выполнения вычислительных операций в изолированном рабочем потоке (операции сортировки, прореживания или преобразования извлеченного набора данных); T_{da} – на операции доступа к внутреннему состоянию базы данных S_{UD} .

По определению, переменные T_{ast} , T_{opt} связаны с состояниями S_{PD} , S_{SD} и неизменны. В практических задачах временем работы парсера и оптимизатора пренебрегают [8, 10] ввиду малого вклада в конечное время выполнения запроса. Аналогично, время T_{net} будем полагать константой, поскольку зависит от размерностей (Q, \mathfrak{R}) , а также каналов связи между СУБД и информационной системой. Время T_{eq} по определению зависит только от (CPU, \mathfrak{R}) и так же константа для нашей задачи. Таким образом, исследуемое выражение можно представить в виде $Q_t \times S_t \xrightarrow{CPU} T_{eq} + const$. Представим S_t – произвольной точкой многомерного пространства, которая может быть представлена некоторым вектором. Таким образом, и

$$S_{UD} = (a_0, a_1, \dots, a_n); a \in A,$$

где A – множество атомарных данных в памяти СУБД. При этом, из определения $Q = (Q^{SQL}, \{Q^{PRD}\})$ следует, что текст запроса на языке SQL вместе со схемой данных формируют множество предикатов $Q^{SQL} \times S_{PD} = \{Q_{SQL}^{PRD}\}$, которое в свою очередь объединяется со множеством исходных предикатов запроса для исчисления

$$DBMS' : (\{Q_{SQL}^{PRD}\} \cup \{Q^{PRD}\}) \times S_{UD} \xrightarrow{CPU} (\mathfrak{R}, T_{da}).$$

Значит в используемой формализации каждый предикат запроса устанавливает отображение

$$Q^P : A \rightarrow A^{PRD}, |A^{PRD}| \leq |A|, Q^P = \{Q_{SQL}^{PRD}\} \cup \{Q^{PRD}\},$$

а объединение или множество предикатов формирует соответственно пересечение областей их значений.

Отдельно следует упомянуть связь определяемых предикатов Q^{PRD} с фактическим содержанием SQL запроса, где возможны операции инверсии NOT, объединения OR, пересечения AND условий, а также формирования их иерархий, практически любой сложности. Все эти операции формируются в терминах алгебры логики и, как следствие, могут быть нормализованы в конъюнктивной форме, как следствие, соответствуя терминологии статьи. Таким образом, текст запроса и его предикаты формируют ограничение над внутренним состоянием S_{UD} , такое, что запрос исчисляется над некоторой частью этого состояния $A^Q \subseteq A$, сформированной пересечением множеств атомарных значений. При этом всякое $a \in A$ – двоичная последовательность в памяти вычислительной машины, таким образом мощность множества A^Q определяет число операций на доступ к данным в время T_{da} . При этом разные запросы, имея от-

личные множества предикатов, будут формировать отличные подмножества над A , которые при этом могут пересекаться. При этом совместный доступ к пересекающимся подмножествам атомарных элементов будет порождать конкуренции и дополнительные затраты времени вычислительной системы [1] и долю последовательных операций закона Амбдала-Уэра, формируя зависимость времени выполнения от параллельно выполняемых запросов.

Полученная зависимость позволяет перейти от исходного вида зависимости $DBM^{\Delta} : Q_i^N \rightarrow T_i^N$ к форме

$$\begin{aligned} T_i^N &= T_{DA} + \text{const} = \\ &= X_0(Q_i^P) + \sum_{i \in I} k_i \cdot X_i(Q_i^P \cup Q_i^P) + \text{const}, \end{aligned}$$

где $Q_i^P = \{q^P \in Q \mid A^{q^P} \cap A^{Q_i^P} \neq \emptyset\}$ соответствует множеству предикатов параллельно выполняющихся запросов, которые определяют ограничения на S_{UD} , образующие непустые множества атомарных элементов в пересечении с множеством атомарных элементов, сформированном совокупностью предикатов, определяемых исследуемым запросом Q_i . В свою очередь, коэффициенты k_i задают отношение времен выполнения исследуемого и классов I параллельно выполняемых запросов, где каждый класс задаётся совокупностью предикатов Q_x^P . Интересной особенностью полученного выражения является отсутствие в нём S_{UD} , вследствие ранее доказанного приближения

$$DBM \approx DBM^{\Delta} : Q_i^N \rightarrow T_i^N + \Delta N.$$

Также для простоты всех выражений полагаем, что возникновение двух запросов Q в один момент времени t не представляется возможным вследствие столь малой величины дискретизации, которая только для этого необходима.

Используя введенные ограничения, обучающую выборку можно представить множеством $Q_i^P \cup \bigcup_{i \in I} (i, k_i, Q_i^P)$. Такое признаковое пространство соответ-

ствует тем зависимостям, которые получаем за счет исследования моделируемой системы. При этом полученное пространство имеет ряд недостатков. Во-первых – сложность представления предикатов в двоичной форме, при сохранении их достаточной семантики. Для работы метода опорных векторов необходимо осуществить выбор ядра преобразования – ядерной функции. Выбранное отображение должно сохранить в конечном многомерном пространстве метрики расстояния соответствующие скрытым от наблюдателя взаимосвязям внутри предикатов. В случае сложных, составных признаков, такой выбор является крайне нетривиальной задачей [15] и позволяет отказаться от SVM для работы с такими данными. Во-вторых, поскольку выполнения всех классов запросов в единый момент времени представляется в целом достаточно малове-

роятным, исходя из многочисленного опыта эксплуатации практических систем [2, 5, 8], в качестве исходных данных будут преобладать частичные признаки. Что вместе с первой особенностью на порядки повышает требования к обучающим выборкам для методов, построенных на базе нейронных сетей, а также в целом вычислительную сложность их применения [15]. Методы, построенные на основе градиентного бустинга (Boosting), также оказываются малоприменимы вследствие достаточно большого числа элементов множества I , как следствие высоких требований к числу задействованных базовых алгоритмов или их сложности [16]. Таким образом, из хорошо изученных и универсальных методов машинного обучения остается случайный лес (Random Forest) [15].

Случайный лес, как метод ансамблирования теории обучения машин, в контексте этой работы имеет ряд важных особенностей: высокая устойчивость к частичным и неполным признакам, простота использования предикатов в качестве решающих правил для построения деревьев. Тем не менее, несмотря на свои достоинства, метод нельзя применить ко множеству кортежей вида (i, k_i, Q_i^P) без некоторой адаптации. Запросы одного класса или запросы, бесконечно мало отличающиеся по отношению времен кооперативного выполнения, будут считаться совершенно разными признаками. Как следствие, без дополнительных преобразований, логически взаимосвязанные признаки будут рассматриваться независимыми друг относительно друга. Такое упрощение приведёт к снижению обобщающей способности алгоритма, а также увеличению расходов на хранение леса и сложности его построения [17]. Для решения этой проблемы предлагаются модификации исходного метода. Критерием ветвления выбран индекс Джини (Gini importance) [18],

$$\Gamma(\vec{p}) = \sum_{i=1}^m p_i(1 - p_i),$$

где вектор p состоит из m вероятностей, встречающихся в подмножестве обучающегося множества. Таким образом, множество значений k_i для всякого рассматриваемого запроса полагаем эквивалентным коэффициенту перед индексом, явным образом закладывая связь между конкуренцией запросов и выбором условия ветвления при синтезе решающего дерева. Это первая модификация классического метода случайного леса. Применение дополнительного весового коэффициента позволяет, отдать приоритет для тех признаков, которые являются составной частью параллельно выполняемых выражений. Как итог, уменьшается влияние признаков из выражений, с которыми предполагается меньшая доля конкуренции, при прочих равных условиях. В общем случае подобный подход должен позволить повысить устойчивость к переобучению. Вместе с тем, не ограничивается чувствительность модели в терминах теории обучения машин. Для формирования леса будут выбираться решающие правила из подмножества признаков низко конкурентных, но часто выполняемых параллельных запросов. Тем не менее, такие правила будут отсекаются в ходе выполнения процедуры CART, вследствие малой величины базового значения ин-

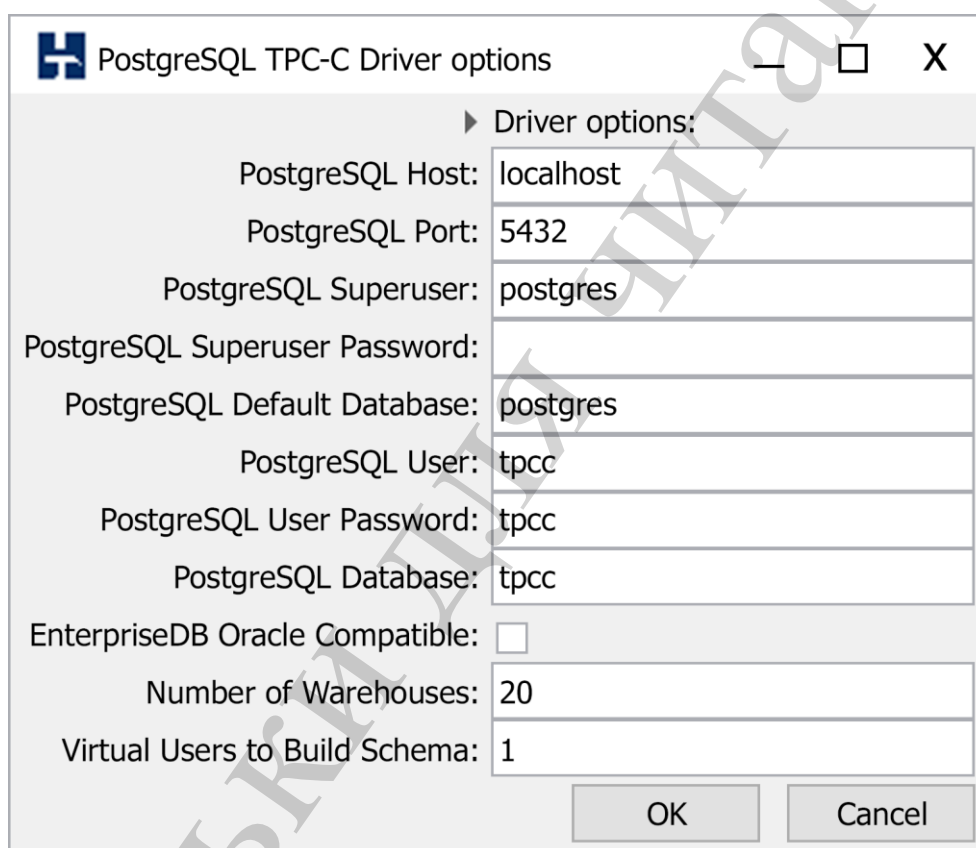
декса Джини [18]. Следовательно, при синтезе отдельного решающего дерева не будут преобладать выбор логических правил, соответствующих запросам с наибольшей долей последовательных вычислений, или наиболее частым одинаковым запросам. Таким образом, решается задача нормализации исходных данных, что увеличивает точность итогового ансамбля [15].

Метод случайных подпространств предполагает случайное выделение подмножества признаков для синтеза отдельного дерева. Это достаточно разумная стратегия в случаях, когда отсутствуют априорные данные о структуре признаков пространства, их взаимосвязях, и возможность сформировать промежуточное представление признаков с сегрегацией избыточных неинформативных признаков. Но в случае сформированного кортежа (i, k_i, Q_i^P) достаточно очевидно наличие взаимосвязей между подмножествами Q^P из одного класса i , как между составными частями целостного, семантически и синтаксически связанного выражения на языке SQL. Таким образом, произвольный выбор явно выраженных субклассов признаков будет способствовать увеличению вычислительной сложности вследствие большего числа шагов на этапах перебора подпространств и отсека переобученных деревьев от общего ансамбля. В контексте предлагаемого представления прецедентов вторая модификация метода случайного леса состоит в применении двух итераций метода случайных подпространств. Оригинальный метод предполагает однородность множества признаков. Обобщение текстов SQL запросов в подмножество классов $\{i\}$ формирует дополнительную априорную информацию о каждом признаке. И предлагаемая модификация заключается в применении двух этапов выборки при формировании подпространства признаков отдельного дерева. На первом этапе формируется подмножество все подклассов конкурентно выполняемых запросов i . На втором этапе осуществляется непосредственный выбор признаков, ограниченных подклассами из первого этапа и текущего запроса. Применение двух итераций метода случайных подпространств снижает вероятность формирования переобученных деревьев. Уменьшаются требования к объему обучающей выборки и вычислительной сложности ансамблирования получаемых деревьев. В некоторых случаях, в целом позволяет рассчитать модель леса, которую нельзя получить из-за значительного числа инвариантов в отсутствии выделенных классов запросов.

В разделе представлен разработанный метод расчета параметров теоретико-множественной модели процесса управления конкурентным доступом. Предлагаемый метод использует статистические данные за период работы реляционной СУБД для расчета её численной модели методами машинного обучения. В отличие от существующих аналогов, целевым параметром служит время, а не его суррогатное представление, учитываются условия в текстах запросов и взаимная конкурентности.

5. Результаты вычислительных экспериментов по оцениванию результативности разработанного метода расчета параметров теоретико-множественной модели процесса управления конкурентным доступом запросов в СУБД реляционного типа

Теория обучения машин – в значительной мере прикладная дисциплина и требует вычислительной проверки предполагаемых результатов в ходе эксперимента. Для формирования экспериментальной выборки были использованы СУБД PostgreSQL и стандартный тест TPC-C с OLTP нагрузкой на базе программного обеспечения имитационного моделирования и нагрузочного тестирования с открытым исходным кодом HammerDB. На рис. 3 и 4 представлены исходные данные для организации макета, а также параметры имитационной модели, используемые для проведения вычислительного эксперимента.



PostgreSQL TPC-C Driver options

Driver options:

PostgreSQL Host:	localhost
PostgreSQL Port:	5432
PostgreSQL Superuser:	postgres
PostgreSQL Superuser Password:	
PostgreSQL Default Database:	postgres
PostgreSQL User:	tpcc
PostgreSQL User Password:	tpcc
PostgreSQL Database:	tpcc
EnterpriseDB Oracle Compatible:	<input type="checkbox"/>
Number of Warehouses:	20
Virtual Users to Build Schema:	1

OK Cancel

Рис. 3. Исходные данные для организации макета базы данных

PostgreSQL TPC-C Driver options

► Driver options:

PostgreSQL Host: localhost

PostgreSQL Port: 5432

PostgreSQL Superuser: postgres

PostgreSQL Superuser Password:

PostgreSQL Default Database: postgres

PostgreSQL User: tpcc

PostgreSQL User Password: tpcc

PostgreSQL Database: tpcc

EnterpriseDB Oracle Compatible: ☐

TPC-C Driver Script: ☐ Test Driver Script
☒ Timed Driver Script

Total Transactions per User: 1000000

Exit on PostgreSQL Error: ☐

Keying and Thing Time: ☐

Vacuum when complete: ☐

EnterpriseDB DRITA Snapshots: ☐

Minutes of Rampup Time: 2

Minutes for Test Duration: 5

Use All Warehouses: ☐

Time Profile: ☐

OK Cancel

Рис. 4. Исходные данных модели конкурентной работы СУБД

Обеспечение протоколирования поступающих запросов было выполнено средствами СУБД. В содержимое конфигурационного файла установленного экземпляра программного обеспечения были внесены изменения в соответствии с табл. 1.

Для формирования обучающей выборки, построения модели и ранжирования параметров была написана программная реализация предлагаемых в работе решений на языке программирования C# с использованием библиотеки Accord.NET. Чтобы исключить влияние средств протоколирования на основную работу системы по обработке поступающих запросов, журналы работы были выведены на отдельный диск в оперативной памяти ЭВМ. Для решения

этой задачи была использована последняя бесплатная из опубликованных версий программы SoftPerfect RAM Disk.

Таблица 1

Установленные параметры конфигурации СУБД

Наименование параметра	Значение
log_destination	'csvlog'
log_min_duration_statement	0
log_line_prefix	'%m [%p]: [%l-1] user=%u,db=%d,app=%a,client=%h %c: %l '
log_duration	on
log_statement	'ddl'

Библиотека Accord.NET часто применяется в научных исследованиях, посвященных вопросам прикладного применения алгоритмов машинного обучения [15, 16], в том числе исследованиях схожей тематики [17, 19]. Ближайшими аналогами для этой библиотеки являются коммерческая система MATLAB, реализации языка R, а также бесплатная библиотека scikit-learn. Ключевой особенностью библиотеки Accord.NET, по сравнению с этими системами, являются доступность применения рефлексии в среде исполнения .NET. Возможность подмены отдельных методов внутри используемых библиотек позволяет просто вносить изменений в работу уже реализованных в библиотеке алгоритмов. Отсутствует необходимость прямого заимствования исходного кода или самостоятельного воссоздания уже существующих алгоритмов. Также, в сравнении с системой MATLAB, библиотека Accord.NET бесплатна для использования. В сравнении с библиотекой scikit-learn, наиболее распространённой в прикладных задачах машинного обучения, имеет в несколько раз большую производительность [15]. Помимо прочего, получаемый программный модуль просто развертывать в производственной среде, поскольку это обычная программа, без зависимостей от стороннего программного обеспечения [19].

Приводимые цифры экспериментов соответствуют выполнению теста на ЭВМ с процессором AMD FX-4330, 4 ГГц, 16 Гб ОЗУ и накопителем Intel SSD SC2KW24. Непосредственно текст выполнялся с имитацией 4, 8 и 12 конкурирующих между собой пользователей. Выбранное число пользователей обусловлено результатами существующих экспериментальных исследований, в которых выполнялась оценка влияния многопоточности на время выполнения запросов [12, 13, 14]. В ходе экспериментов на других аппаратно-вычислительных платформах не было выявлено существенного изменения частоты ошибок на контрольной выборке.

Основным критерием работоспособности моделей, получаемых методами машинного обучения, служит средняя частота ошибок на контроле [18]. Для автономных СУБД важно не время выполнения каждого запроса в отдельности, а возможность оценить суммарное время выполнения множества запросов [12, 14, 19]. Подобная особенность упростила изучение получаемых результатов. Так, в качестве функции потерь был выбран модуль разницы сумм времён вы-

полнения запросов из прецедентов контрольной выборки и значений регрессии по отношению к сумме времен контрольной выборки. Разделение прецедентов на обучающую и контрольную выборки выполнялось псевдослучайно, на базе вихря Мерсенна MT19937. Для оценки степени переобучения был выполнен контроль при нарастающей длине обучения на отложенных данных (hold-out CV). Полученные результаты представлены в табл. 2.

Таблица 2
Результаты вычислительного эксперимента

Число пользователей	Размер обучающей выборки	Частота ошибок на контроле
4	10000	0,0012545776
4	50000	0,0011904734
4	500000	0,0011912891
8	10000	0,0012336765
8	50000	0,0012248468
8	500000	0,0011715882
12	10000	0,0011814117
12	50000	0,0011349580
12	500000	0,0011483744
4*	500000*	0,8124643188*
8*	500000*	0,8315736365*
12*	500000*	0,8925198401*

Отмеченные в таблице символом «*» результаты получены применением не модифицированной реализации метода случайной лес из библиотеки Accord.NET, что позволяет сравнить его с разработанным подходом. Для простоты во всех тестах применялась фиксированная тестовая выборка размером в три миллиона запросов, при этом из общего числа отбрасывались самые первые и самые последние запросы, приходящиеся на начало и завершение тестовых прогонов. Для не модифицированного метода использовался максимальный размер обучающей выборки, поскольку его уменьшение не оказывало существенного влияния на частоту ошибок, свидетельствующую о переобучении модели.

Представленные результаты вычислительных экспериментов позволяют сделать несколько основных выводов. Во-первых, предложенный подход к формированию признакового описания позволяет формализовать статистические данные SQL запросов в формате, пригодном для расчета численных моделей методами машинного обучения. Во-вторых, эксперимент подтверждает результативность разработанных теоретико-множественной модели и метода расчета параметров для модели управления процессом конкурентным доступом.

6. Обсуждение результатов моделирования и проведения вычислительного эксперимента

Полученные в ходе вычислительного эксперимента величины ошибок на контрольных выборках свидетельствуют о пригодности предлагаемой модели, по меньшей мере для применения к реляционным СУБД с OLTP нагрузкой, близкой к имитируемой тестом TPC-C. Внедренные в оригинальный метод изменения позволяют значительно уменьшить частоту ошибок даже при меньших размерах обучающих выборок и избежать как недообучения, так и переобучения.

Оценка результатов вычислительного эксперимента позволяет заключить:

- рост размера выборки для предлагаемого алгоритма не приводит к падению точности, т. е. в эксперименте не наблюдается переобучение;
- величина ошибки существенно менее одной второй, что позволяет ещё больше наращивать точность за счет включения получаемой модели в качестве базового алгоритма в иные ансамбли;
- предлагаемые модификации к исходному варианту метода случайный лес повышают точность получаемой параметрической модели.

Результаты вычислительного эксперимента на имитационной модели позволяют сделать вывод о возможности применения разработанного подхода к моделированию времён выполнения запросов в реляционных СУБД с конкурентным доступом. Применение времени в качестве целевого параметра позволяет исключить искусственные промежуточные представления и повышает точность результирующих оценок.

В работе представлен новый подход к представлению процесса управления конкурентным доступом в реляционных базах данных, основанный на теоретико-множественном представлении признакового описания запросов. На основе введенных отношений выведена и доказана функциональная зависимость между временем обработки и текстами параллельно вычисляемых SQL запросов, атомарными данными в памяти СУБД.

К недостаткам предлагаемого подхода можно отнести особые требования к множеству поступающих в систему запросов. Во-первых, построение модели методами машинного обучения аксиоматично ограничивает минимальный размер обучающей выборки в зависимости от сложности и многообразия обрабатываемых запросов. В целом это требование проистекает и отражает прикладные аспекты применения теории Вапника-Червоненкиса. В практическом приложении это означает невозможность заранее предсказать, на каком объеме обучающей выборки значения ошибки при кросс-валидации достигнет приемлемых в контексте конечной задачи величин. Соответственно неизвестны временные интервалы, на которых необходимо проводить наблюдение над системой. Во-вторых, система не должна существенно менять характеристики своей работы на временных интервалах, в пределах которых выполняется сбор признаков. Случайный лес, будучи одним из сильных методов машинного обучения [15], обладает хорошей обобщающей способностью. Поскольку базовым примитивом служит ансамбль кусочно-постоянных функций, получаемые параметрические модели могут аппроксимировать произвольные функциональные зависимости. Однако изменение характеристик работы СУБД во времени

будет приводить к зашумлению исходных данных и провоцировать рост смещения (bias) ответов финального алгоритма. Как следствие, точность получаемой модели начинает падать к величинам, исключающим её практическое применение.

К перспективным направлениям развития этого исследования можно отнести:

- разработку аналогичных моделей для применения в не реляционных базах данных;
- разработку признакового описания запроса, для включения в него сведений о долговременных и составных транзакциях;
- проведение экспериментальной оценки применимости иных методов машинного обучения, возможности совершенствования алгоритмов ансамблирования и построения решающих деревьев.

7. Выводы

1. В работе предложена модель процесса управления конкурентным доступом запросов в реляционных базах данных на основе теоретико-множественного представления признаковое описания запроса. В отличие от существующих, представленная модель использует условия доступа к данным из текстов запросов для расчёта доли последовательных вычислений. Предложенный подход ориентирован на оценку непосредственно времени выполнения запроса в конкурентной среде, без использования промежуточных единиц стоимости отдельных операций.

2. Разработан метод расчета параметрической модели доступа к данным в реляционных СУБД построенный на основе случайного леса. Метод отличается применением учетом доли последовательных вычислений, в качестве весового коэффициента при выборе решающих правил и применением классов эквивалентности для схожих запросов в процедуре выбора случайного подпространства признаков.

3. Проведение вычислительных экспериментов с OLTP нагрузкой свидетельствует об адекватности модели и применимости разработанного метода для задач оценки времени выполнения запроса. Однако использование метода машинного обучения накладывает ряд ограничений на применимость полученных результатов. Во-первых, на временных интервалах, которые используются для сбора статистических данных и непосредственно оценки времени выполнения запросов, взаимодействие с СУБД должно быть квазистационарным. Т. е. суперпозиция изменений хранимой в базе данных информации и изменения характеристик поступающих запросов должна быть такой, чтобы существовал способ отбросить не более половины от всех запросов и описать работу системы как стационарный процесс. При этом сложность формального представления обработки запросов в виде стационарного процесса делает более практичной экспериментальную оценку применимости полученной модели для каждой системы. Во-вторых, отсутствует способ априори определить временной интервал, на котором необходимо вести наблюдение и сбор статистических данных об обработке запросов для получения адекватной модели. Второе ограничения

напрямую связано с проблемой оценки размерности Вапника-Червоненкиса. Отсутствуют разумные способы теоретически оценить применимость предложенного метода и рассчитать минимальный интервал наблюдения для конкретной базы данных, необходимо получать их экспериментально. В свою очередь, изменение хранимых данных и обрабатываемых запросов будут с течением времени изменять эти оценки непредсказуемым способом. Следовательно, практическая реализация предлагаемого метода должна включать вспомогательный алгоритм для проверки адекватности рассчитанной параметрической модели, перед её применением в алгоритмах автоматизации автономной базы данных.

Литература

1. Gromey D. D., Lebedenko E. V. Automation of data distribution for the DBMS with competitive requests // Modern informatization problems in simulation and social technologies Proceedings of the XX-th International Open Science Conference. Yelm, WA, USA, 2015. P. 167–173.
2. Chaudhuri S., Narasayya V., Ramamurthy R. Exact cardinality query optimization for optimizer testing // Proceedings of the VLDB Endowment. 2009. Vol. 2, Issue 1. P. 994–1005. doi: <https://doi.org/10.14778/1687627.1687739>
3. Борчук Л. Е. Стоимостные оптимизаторы для СУБД: вчера и сегодня // Открытые системы. СУБД. 2016. № 1. С. 36–39.
4. ERMIA / Kim K., Wang T., Johnson R., Pandis I. // Proceedings of the 2016 International Conference on Management of Data – SIGMOD '16. 2016. doi: <https://doi.org/10.1145/2882903.2882905>
5. Neumann T., Mühlbauer T., Kemper A. Fast Serializable Multi-Version Concurrency Control for Main-Memory Database Systems // Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data – SIGMOD '15. 2015. doi: <https://doi.org/10.1145/2723372.2749436>
6. Rheinländer A., Leser U., Graefe G. Optimization of Complex Dataflows with User-Defined Functions // ACM Computing Surveys. 2017. Vol. 50, Issue 3. P. 1–39. doi: <https://doi.org/10.1145/3078752>
7. Cost-Model Oblivious Database Tuning with Reinforcement Learning / Basu D., Lin Q., Chen W., Vo H. T., Yuan Z., Senellart P., Bressan S. // Lecture Notes in Computer Science. 2015. P. 253–268. doi: https://doi.org/10.1007/978-3-319-22849-5_18
8. Automated Demand-driven Resource Scaling in Relational Database-as-a-Service / Das S., Li F., Narasayya V. R., König A. C. // Proceedings of the 2016 International Conference on Management of Data – SIGMOD '16. 2016. doi: <https://doi.org/10.1145/2882903.2903733>
9. Mozafari B., Niu N. A Handbook for Building an Approximate Query Engine // IEEE Data Eng. Bull. 2015. Vol. 38, Issue 3. P. 3–29.
10. Yoon D. Y., Niu N., Mozafari B. DBSherlock // Proceedings of the 2016 International Conference on Management of Data – SIGMOD '16. 2016. doi: <https://doi.org/10.1145/2882903.2915218>

11. Radhika G., Chhabra P., Kumari R. Consistency models in distributed shared memory systems // International Journal of Computer Science and Mobile Computing. 2014. Vol. 3, Issue 9. P. 196–201.
12. Arulraj J., Pavlo A., Dulloor S. R. Let's Talk About Storage & Recovery Methods for Non-Volatile Memory Database Systems // Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data – SIGMOD '15. 2015. doi: <https://doi.org/10.1145/2723372.2749441>
13. Query-based Workload Forecasting for Self-Driving Database Management Systems / Ma L., Van Aken D., Hefny A., Mezerhane G., Pavlo A., Gordon G. J. // Proceedings of the 2018 International Conference on Management of Data – SIGMOD '18. 2018. doi: <https://doi.org/10.1145/3183713.3196908>
14. Arulraj J., Pavlo A., Menon P. Bridging the Archipelago between Row-Stores and Column-Stores for Hybrid Workloads // Proceedings of the 2016 International Conference on Management of Data – SIGMOD '16. 2016. doi: <https://doi.org/10.1145/2882903.2915231>
15. Nithya B. An Analysis on Applications of Machine Learning Tools, Techniques and Practices in Health Care System // International Journal of Advanced Research in Computer Science and Software Engineering. 2016. Vol. 6, Issue 6.
16. Hogland J., Anderson N. Function Modeling Improves the Efficiency of Spatial Modeling Using Big Data from Remote Sensing // Big Data and Cognitive Computing. 2017. Vol. 1, Issue 1. P. 3. doi: <https://doi.org/10.3390/bdcc1010003>
17. Multi-source data analysis and evaluation of machine learning techniques for SQL injection detection / Ross K., Moh M., Moh T.-S., Yao J. // Proceedings of the ACMSE 2018 Conference on – ACMSE '18. 2018. doi: <https://doi.org/10.1145/3190645.3190670>
18. Janitza S., Celik E., Boulesteix A.-L. A computationally fast variable importance test for random forests for high-dimensional data // Advances in Data Analysis and Classification. 2018. Vol. 12, Issue 4. P. 885–915. doi: <https://doi.org/10.1007/s11634-016-0276-4>
19. Data Exploration with SQL using Machine Learning Techniques / Cum-in J., Petit J., Scuturici V., Surdu S. // Open Proceedings. 2017. P. 96–107. doi: <http://doi.org/10.5441/002/edbt.2017.10>