DOI 10.37882/2223-2966.2025.06.13

ОСОБЕННОСТИ ПРИМЕНЕНИЯ B-TREE ИНДЕКСОВ В ПОСТРЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ НА ПРИМЕРАХ POSTGRESQL И MONGODB

THE EFFECTIVENESS OF THE PRINCIPLES OF ADAPTIVE LAYOUT IN THE DEVELOPMENT OF USER INTERFACES

M. Vinogradova
B. Goryachkin
L. Litvinovich

Summary. *Problem statement*. Reducing the execution time of complex database queries becomes an urgent problem when working with tables and collections with many records [1–4, 9–17]. During the operation of information systems, the load on database management systems increases. Server performance in multiuser mode does not meet user expectations. This problem is found in all information systems that work with big data. When solving the problem, companies use tools such as vertical and horizontal scaling or query optimization. Vertical and horizontal scaling is significantly more expensive compared to optimizing database queries. Query optimization in multiuser mode involves fragmentation, analysis, and tracking the execution time of all queries [16]. Thus, correcting the situation of a long wait for a response for complex queries to tables and collections with many records requires a lot of material and time costs for companies. It is necessary to optimize the execution of complex database queries, helping companies create a database of their queries that will be completed in an acceptable time.
Results. The existing units of measurement are considered. It is determined how and why they are used in web design. The types of units of measurement are studied: absolute and relative, their differences and areas of applicability are indicated. The principles of adaptive layout are highlighted, as well as the tasks for solving which the principles of adaptive layout and the units of measurement considered were used.
Practical significance. The practical significance lies in reducing the execution time of such requests to a level below the time of visual perception by a human operator.

Keywords: information model, query optimizer, B-tree, composite indexes, covering indexes, selectivity, query ergonomics, query ergonomics.

Виноградова Мария Валерьевна
кандидат технических наук, доцент,
Московский государственный технический университет им. Н.Э. Баумана
vinogradova.m.iu5@yandex.ru
Горячкин Борис Сергеевич
кандидат технических наук, доцент,
Московский государственный технический университет им. Н.Э. Баумана
bsgor@mail.ru
Литвинович Людмила Валентиновна
Московский государственный технический университет им. Н.Э. Баумана
89167429943@mail.ru

Аннотация. *Постановка проблемы*. Снижение времени выполнения сложных запросов к базам данных становится актуальной проблемой при работе с таблицами и коллекциями с большим количеством записей [1-4, 9–17]. В процессе работы информационных систем нагрузка на системы управления базами данных возрастает. Производительность серверов при многопользовательском режиме не соответствует ожиданиям пользователей. Эта проблема встречается во всех информационных системах, которые работают с большими данными. При решении проблемы компании используют такие средства, как вертикальное и горизонтальное масштабирование или оптимизацию запросов. Вертикальное и горизонтальное масштабирование значительно дороже по сравнению с оптимизацией запросов к базам данных. Оптимизация запросов при многопользовательском режиме предусматривает фрагментацию, анализирование и отслеживание времени выполнения всех запросов [16]. Таким образом, исправление ситуации долгого ожидания ответа при сложных запросах к таблицам и коллекциям с большим количеством записей требует больших материальных и временных затрат для компаний. Необходимо оптимизировать выполнение сложных запросов к базам данных, помогая компаниям создавать базу своих запросов, которые будут выполняться за приемлемое время.
Результаты. Мы выбрали задачу оптимизации сложных запросов к базе данных, помогая компаниям создавать базу своих запросов, которые будут выполняться за определённое время. Время выполнения сложных запросов в нашем исследовании не превысило времени первых трёх фаз восприятия человеком зрительной информации. Таким образом, мы достигли успеха в своём исследовании.
Практическая значимость. Практическая значимость заключается в снижении времени выполнения таких запросов до уровня ниже времени зрительного восприятия человеком-оператором.

Ключевые слова: информационная модель, оптимизатор запросов, B-tree, составные индексы, покрывающие индексы, селективность, эргономичность запроса, эргономические показатели запроса.

Введение

Проблеме скорости выполнения SQL-запросов посвящены работы ряда авторов [1-4, 9–16]. На время выполнения запросов влияет множество факторов: индексы, объем данных, выбор плана выполнения запроса оптимизатором, текущая нагрузка на базу данных, задержки на уровне сети.

Необходимость оптимизации запросов заключается также в снижении нагрузки на СУБД. От структуры составленного запроса зависит, какое количество вычислительных ресурсов (оперативная память, процессорное время) будет использоваться. В данной работе разрабатываются информационные модели сложных поисковых запросов с B-tree индексами на примерах PostgreSQL и MongoDB. В соответствии с контекстом данных онлайн научной библиотеки предполагалось использование документо-ориентированной MongoDB. Выбор PostgreSQL объясняется лидерством этой СУБД перед другими при запросах к большим данным на чтение [3]. В обеих СУБД используется B-tree структура данных, в MongoDB используется B+tree. Данный факт также учитывался при выборе MongoDB в нашем эксперименте. Структура данных обеих СУБД позволяет использовать запросы с B-tree индексами. Запросы с B-tree индексами по умолчанию возвращают упорядоченные данные. Это значительно сокращает время выполнения запросов. Поэтому, мы выбрали исследование сложных запросов с применением B-tree индексов.

В статье применяются сокращения и обозначения.

— Информационная модель — совокупность информации, характеризующая существенные свойства и состояния объекта, процесса, явления, а также взаимосвязь с внешним миром.
— Оптимизатор запросов — выбор данных из базы, движок СУБД, который пытается наилучшим (наиболее быстрым, менее затратным) образом использовать ресурсы. В качестве результата оптимизатор выдает «План выполнения запроса».
— B-tree — это структура данных, которая предоставляет отсортированные данные и позволяет выполнять поиск, последовательный доступ, вложения и удаления в отсортированном порядке. B-tree обладает высокой способностью хранить системы, которые записывают большие блоки данных. B-tree упрощает двоичное дерево поиска, допуская узлы с более чем двумя дочерними элементами; одна метастраница хранится в фиксированной позиции в начале первого сегментного файла индекса, все остальные страницы являются либо конечными страницами, либо внутренними страницами.
— Составные индексы — индексы в MongoDB, которые состоят из нескольких полей; порядок полей в составном индексе имеет значение, например, если составной индекс состоит из {author: 1, name : -1 }, индекс сортируется сначала по полю author и потом в пределах поля author по поля name.
— Покрывающие индексы — индексы, в выражении которых присутствуют все поля, используемые в запросе. Оптимизатору не надо обращаться к таблице в памяти, он получает информацию из таблицы индекса;
— Селективность — количество строк в таблице, удовлетворяющих запросу, относительно к общему количеству строк в наборе; чем меньше строк для поиска, тем выше селективность.
— Эргономичность запроса — время выполнения запроса (производительность СУБД). При запросах к большим данным выбор оптимизатором плана сканирования индекса имеет огромное значение потому, что отображается во времени ожидания ответа для пользователя. Создание условий, в которых человек может работать комфортно и эффективно относится к области знаний эргономики, поэтому под эргономичностью запроса мы будем понимать время его выполнения.
— Эргономические показатели запроса — селективность полей в индексе B-tree, учет NULL значений в полях индекса B-tree, покрывающие индексы B-tree.
— Человек-оператор — человек, осуществляющий трудовую деятельность, основу которой составляет взаимодействие с объектом воздействия, машиной и средой на рабочем месте при использовании информационной модели и органов управления [1].
— Суперкласс — это компоненты наследования данных, в которых определенные атрибуты и характеристики сущности наследуются от родительского объекта к дочерним объектам или сущностям.
— СУБД — система управления базами данных.

Методика и организация исследования

Модель наших запросов с использованием B-tree индексов предполагает выполнение определённых условий оптимизации. В этом исследовании изучались условия оптимизации сложных запросов к базе данных. Исследование проводилось на запросах к PostgreSQL и MongoDB. Особенности архитектуры PostgreSQL и MongoDB определили разный подход в изучении оптимизации сложных запросов к базам данных с большим количеством записей. Репликации и шардирование позволяют MongoDB использовать дополнительную оперативную память в многопользовательском режиме работы. Подсистема хранения данных WiredTiger обеспечивает оптимальное использование оперативной памяти и сокращает количество обращений к файловой

системе. Мы исключили шардирование для чистоты исследования времени работы WiredTiger, времени выполнения одиночных сложных запросов и времени выполнения этих же запросов в серии запросов в MongoDB. Особенности структуры данных, B-tree в PostgreSQL и B+tree в MongoDB, позволили провести сравнительное исследование запросов с применением B-tree индексов в сложных запросах в обеих СУБД.

В PostgreSQL используется структура данных B-tree. В PostgreSQL файлы данных разбиваются на сегменты размером сегмента 1 ГБ. Данные хранятся на страницах фиксированного размера 8 КБ. Для каждой строки в PostgreSQL создаётся внутренний идентификатор. Первая цифра в идентификаторе — это номер страницы, а вторая — номер кортежа. Данные представляют строки, для доступа к ним используются ссылки. Ссылки содержат логическое смещение на странице. В отдельной области хранятся индексы и ссылки на соседние узлы в B-tree.

Вместо таблиц MongoDB использует коллекции, где хранятся документы; MongoDB хранит данные в виде JSON-документов с различным уровнем глубины, кодируя их в бинарном формате BSON. У документов в коллекции может различаться набор полей и их типы данных. Запросы к документам с вложенными полями считаются сложными запросами. В MongoDB используется структура данных B+tree. При создании документа в MongoDB создаётся идентификатор объекта. Его 12-байтовая формула ObjectId включает в себя 4-байтовую отметку времени создания, 3-байтовый машинный идентификатор, 2 обработанных байта и 3-байтовый счётчик, начинающийся со случайного значения. Индексы в MongoDB занимают много памяти и затратны при считывании.

На рисунке 1 представлены структуры даных B-tree и B+tree.

На рисунке 1 (справа) изображён процесс сегментирования для дальнейшего шардирования и горизонтального масштабирования MongoDB. В отличие от B-tree внутренние узлы в индексе B+tree не имеют ссылок на страницы документа, но содержат только ссылки на соседние узлы и на узлы уровнем ниже; только листовые узлы B+tree имеют информацию страниц документа. Такая особенность и позволяет сегментировать данные без потерь. Для оптимизации работы в оперативной памяти в MongoDB разработана подсистема хранения данных WiredTiger. WiredTiger хранит данные таблицы в памяти, используя структуру данных B+tree, ссылаясь на узлы B+tree как на страницы. Также поддерживается сжатие для всех коллекций и индексов. WiredTiger кэширует часть таблицы, к которой в данный момент обращается приложение для чтения или записи в основной памяти. Таким образом, сокращает число обращений к диску.

В качестве изучаемых критериев оптимизации сложных запросов к таблицам и коллекциям с большим количеством записей были выбраны:

— селективность,
— фильтрация NULL-значений в запросе,
— покрытие условия запроса,
— серийность сложных запросов

Изучение функций метода доступа к индексу B-tree в документации PostgresPro [5-8], а также результаты нашего исследования, позволили нам определить факторы, влияющие на стоимость доступа к таблице индексов в PostgreSQL в информационную модель, представленную на рисунке 2.

Изучение функций метода доступа к индексу B+tree в документации MongoDB [15,16,17], а также результаты нашего исследования, позволили нам определить фак-

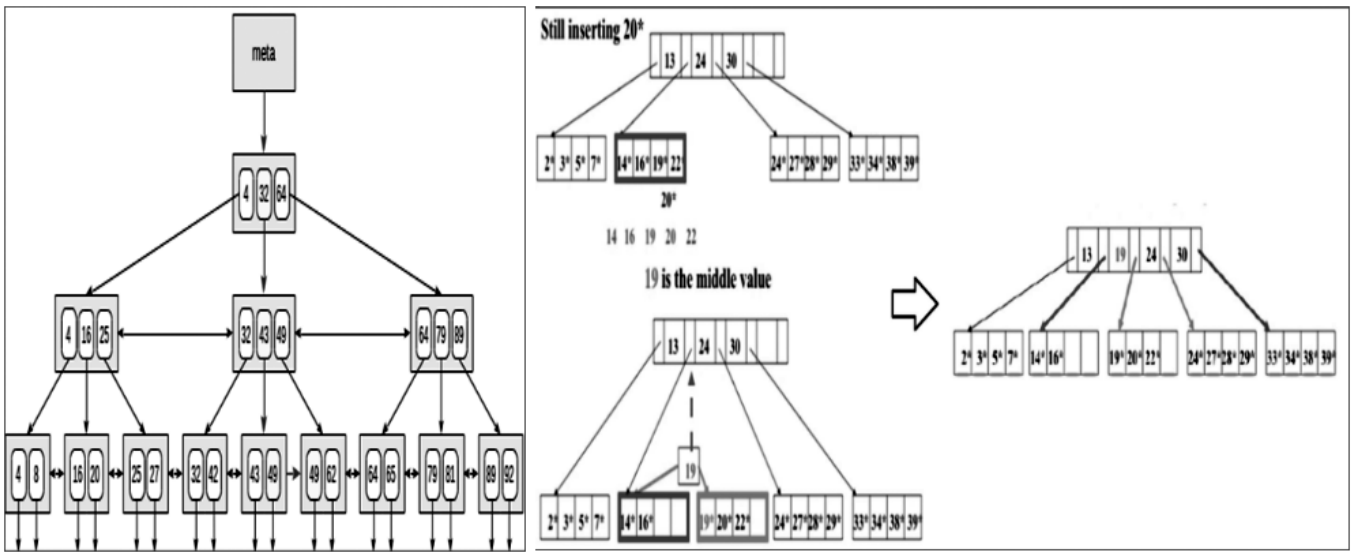

Рис. 1. Сруктуры данных B-tree(слева) и B+tree(справа)

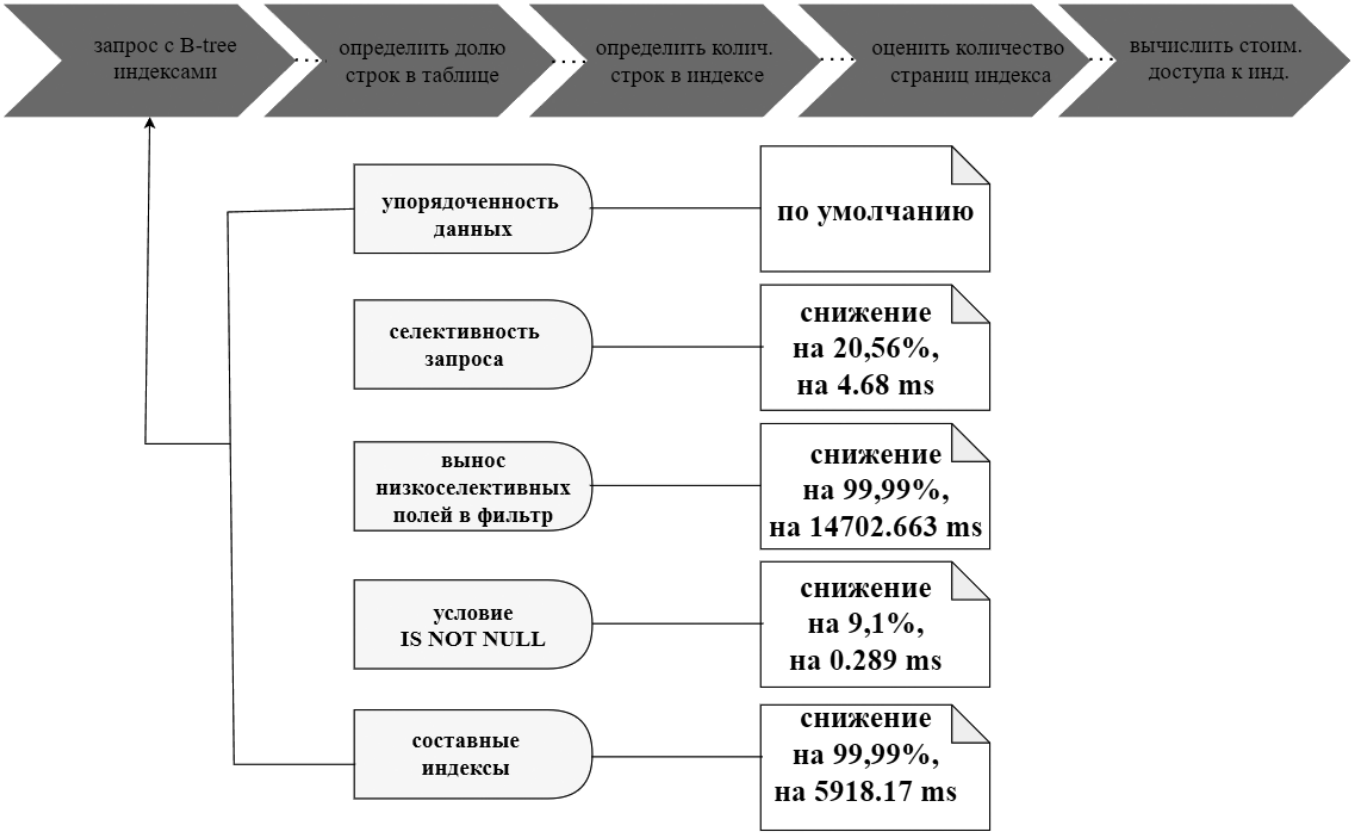

Рис. 2. Информационная модель стоимость доступа к таблице индексов в PostgreSQL

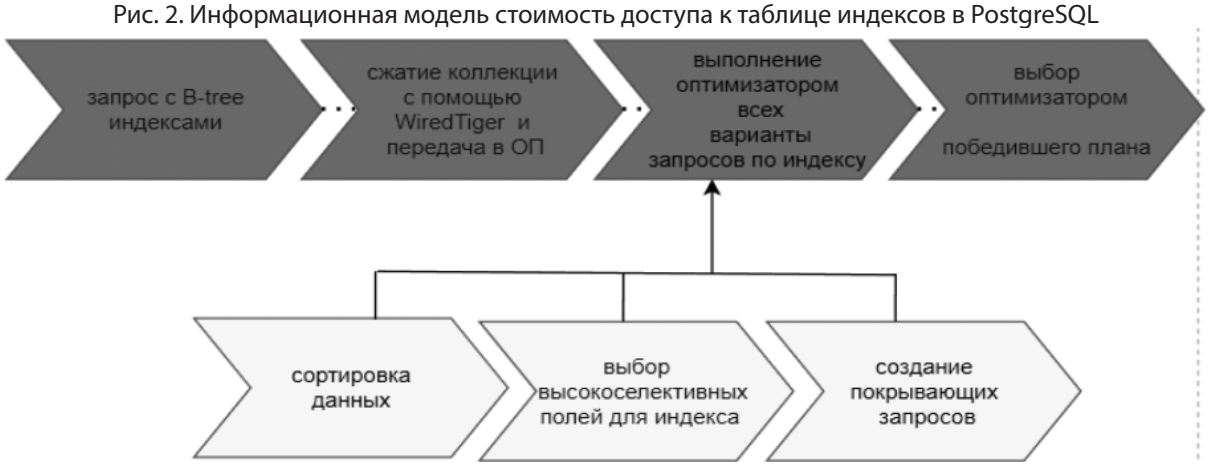

Рис. 3. Информационная модель доступа к индексам и к коллекции

торы, влияющие на стоимость доступа к таблице индексов в MongoDB. На рисунке 3 представлена информационная модель для MongoDB. Для запросов в MongoDB мы не рассматривали горизонтальное масштабирование для чистоты эксперимента и исключения влияния факторов распараллеливания запросов на время выполнения запроса. Поэтому в информационной модели для MongoDB будет рассматриваться только запрос на одном компьютере.

Из этапов в информационной модели расчёта стоимости доступа к индексу предлагается формула расчёта времени выполнения сложного запроса с использованием индекса B-tree в PostgreSQL:

$$T_{req\_index} = t_{plan} + (t_{index\_read} + t_{check\_ref}) * n + [t_{inp\_oup} + t_{system}] \quad (1),$$

где T_{req_index} — время чтения коллекции,

t_{plan} — время работы планировщика,

t_{index_read} — время чтения таблиц индекса,

t_{check_ref} — время чтения таблицы

t_{inp_oup} — время на операции ввода/вывода,

t_{system} — время, зависящее от пропускной способности системы,

n — число соединённых таблиц.

Из этапов в информационной модели расчёта стоимости доступа к индексу предлагается формула расчёта

времени выполнения сложного запроса с использованием индекса B+tree в MongoDB:

$$T_{req\_index} = t_{WT} + t_{plan} + (t_{index\_read} + t_{check\_ref}) * n + \\ +[t_{inp\_oup} + t_{system}] \quad (1),$$

где T_{req_index} — время чтения коллекции,
t_{WT} — время работы Wired Tiger,
t_{plan} — время работы планировщика,
t_{index_read} — время чтения индекса,
t_{check_ref} — время чтения документа в коллекции
t_{inp_oup} — время на операции ввода/вывода,
t_{system} — время, зависящее от пропускной способности системы

Методы и алгоритмы

В нашем исследовании для получения информации о длительности этапов запроса, об объёме данных, обрабатываемых операторами, о назначении планировщиком параллельных процессов чтения использовались методы анализа запросов. Для получения информации о свободном объёме оперативной памяти использовался метод просмотра диспетчера задач. Для получения информации об объёме памяти, который занимает таблица, применяли метод взвешивания таблиц.

В результате эксперимента был проведён анализ планов выполнения запросов в обеих СУБД, построены логические схемы выбора оптимизатором плана выполнения запроса с B-tree индексами в PostgreSQL и MongoDB. На рисунке 4 представлена логическая схема выбора оптимизатором плана выполнения запроса с B-tree индексами в PostgreSQL.

На рисунке 5 представлена логическая схема выбора оптимизатором плана выполнения запроса с B+tree индексами в MongoDB.

Исходные данные

Эксперимент проводился на сгенерированных данных онлайн научной библиотеки. Созданы таблицы в PostgreSQL: «resource» с 70 млн записей, «article» с 30 млн записей и «userlib» с 30 млн записей. В MongoDB созданы коллекция «resource» с 23 млн документов. Выполнялись сложные запросы к двум таблицам в PostgreSQL и сложные запросы ко вложенному полю в документе коллекции MongoDB. Исследовался план выполнения запросов, возвращаемый планировщиком. Изучались особенности применения B-tree индексов в этих запросах для каждой СУБД.

Ресурсы научной онлайн библиотеки состоят из нескольких типов: статьи, книги, презентации, видео, методические указания. В таблицах PostreSQL у этих типов ресурсов есть одинаковые поля: название источника, автор, тип источника, путь к файлу, год издания. Таблица «resource» содержит поля «id» — идентификационный номер, name — название источника, author — автор, type — тип источника, «path» — путь к файлу, «year» — год издания. Ресурс является суперклассом для статьи, видео и т.д. Таблица «article» включает поля: «id», «DOI», «ISBN» и «resource_id» для связи с суперклассом «resource». Аналогично связаны другие классы видео, презентация, научный том и методические указания с суперклассом ресурс через поле «resource_id». Таблица «userLibrs» включает в себя поля «id», «name» — имя пользователя, «rights» — права_пользователя и «resource_id» для связи с суперклассом «resource».

В коллекции MongoDB собраны документы разного типа с разными полями; статья с полями «_id», «name» — название источника, «author» — автор, «type» — тип источника, «path» — путь к файлу, «year» — год издания, «DOI», «ISBN» и другие типы ресурсов со своими полями. Количество полей у разных типов документов было разное.

В PostgreSQL выполнялись запросы с B-tree индексами к двум таблицам «resource» и «article», «resource» и «userLib».

В MongoDB выполнялись запросы с B+tree индексами к вложенному низкоселективному полю «language» документов типа «scientificVolume», вложенным высокоселективному «DOI» и низкоселективному полю «ISBN» документов типа «article»; также проводились запросы с покрывающими индексами, где явно указывались «_id».

Методы

Целью оптимизации поисковых запросов к отношениям/коллекциям с большим количеством записей было снижение времени выполнения запросов с B-tree индексами до времени первых трех фаз зрительного анализатора человека-оператора.

Методами взвешивания таблиц были выбраны:
— pg_size_pretty(pg_relation_size('userLibr_2')),
— обращение к статистике метатаблиц:
— select * from bt_metap('index_btree_us_2'),
— bt_page_stats('index_btree_us_2', 3)

Методами анализа запросов были выбраны:
— EXPLAIN ANALYZE для запросов в PostgreSQL,
— explain(«executionStats») для запросов в MongoDB.

Результаты исследования и их обсуждение

В таблице 1 представлены данные по количеству записей в таблицах PostgreSQL и коллекции «resource»

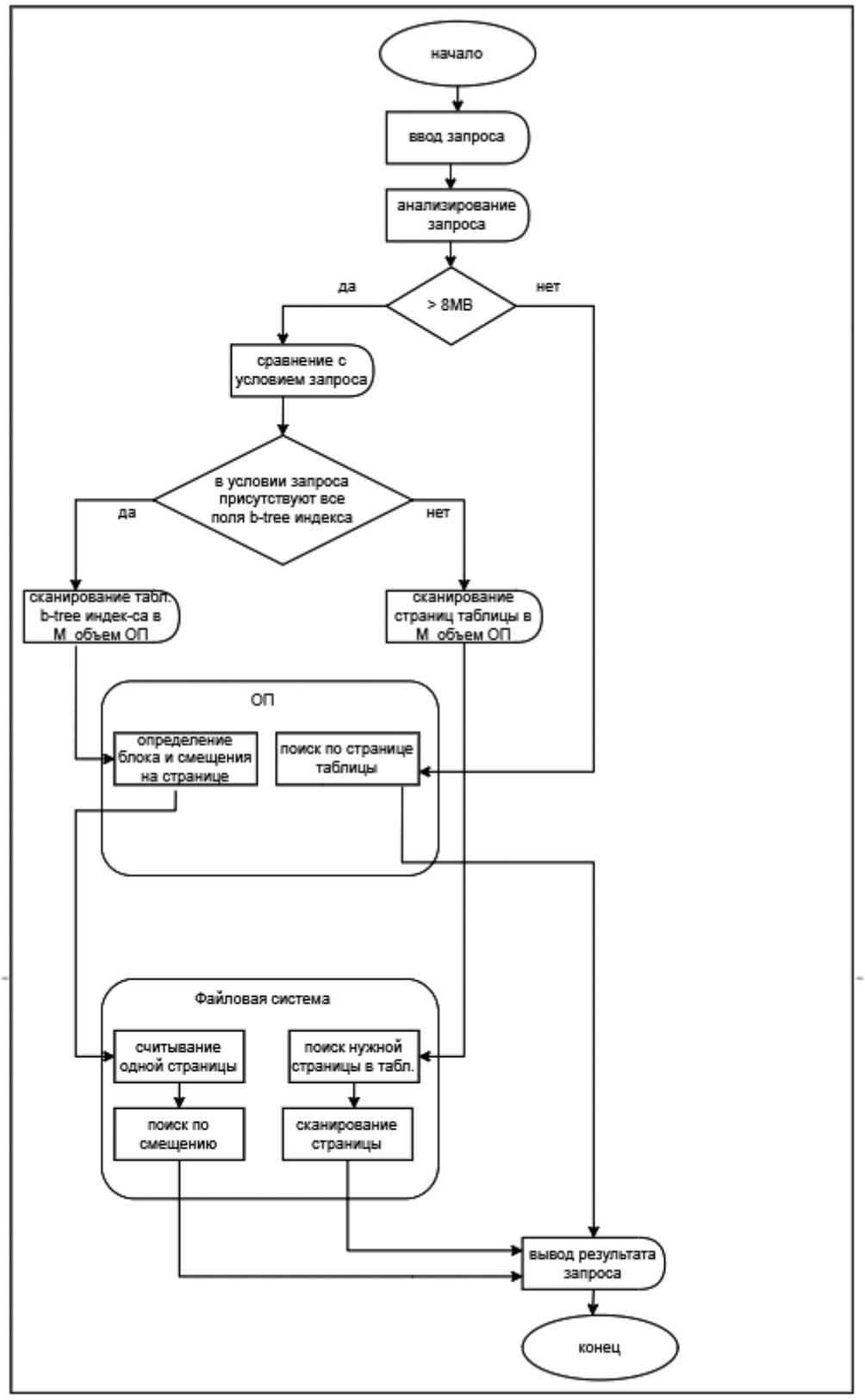

Рис. 4. Логическая схема выбора оптимизатором плана выполнения запроса с B-tree индексами в PostgreSQL

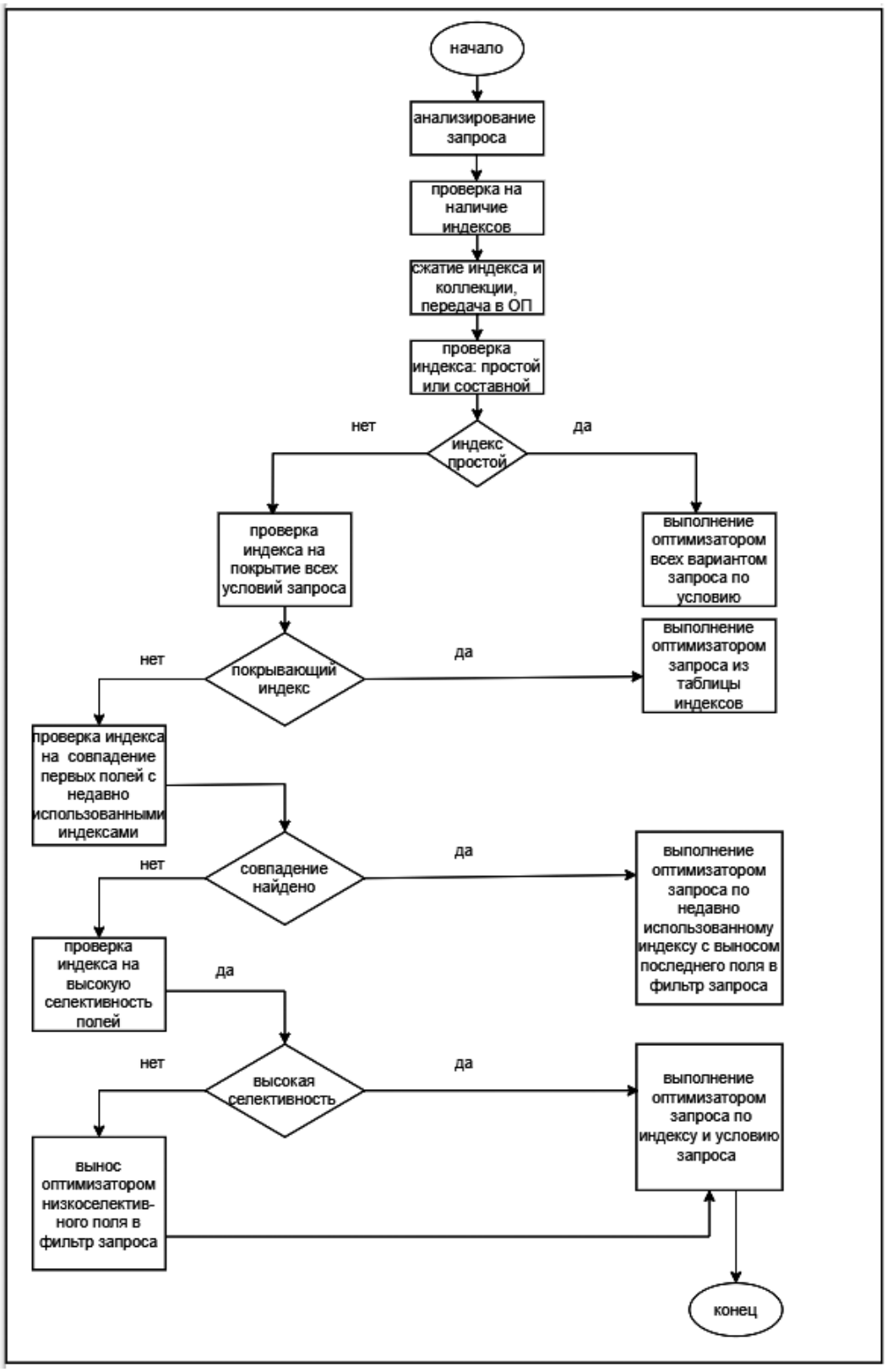

Рис. 5. Логическая схема выбора оптимизатором плана выполнения запроса с B-tree индексами в MongoDB

MongoDB и время полного считывания в сложных запросах (запросы №№ 1, 2).

Таблица 1.

Данные по количеству записей в таблицах PostgreSQL и коллекции «resource» MongoDB и время полного считывания в сложных запросах

СУБД	PostgreSQL	MongoDB
объём таблиц/коллекций (количество записей)	«resource»: 70 000 000; «article»: 30 000 000; «userlab»: 30 000 000	«resource»: 23 000 000;
время полного сканирования таблиц/коллекции (мс)	92683.329	28326

В таблице 2 представлены результаты выполнения сложных запросов в PostgreSQL и MongoDB.

В запросе №1 происходит полное считывание из двух таблиц resource и article, время выполнения 92683.329 мс.

В запросе №2 происходит полное считывание из коллекции resource до вложенного уровня включительно, время выполнения 28326 мс.

В запросе №3 присваиваем пользователю с именем “user2000” право “admin”, повышая селективность поля максимально и выполняем запрос по индексу B-tree к двум таблицам resource и userlibr. В плане выполнения запроса получаем чтение только индексных таблиц.

В запросе №5 к двум таблицам resource и article по индексам B-tree оптимизатор при выборе плана по индексам будет читать все строки таблицы индексов с учетом значений null.

Для запроса №6 создаем индекс с учетом null значений и, тем самым, снижаем время выполнения запроса.

В запросе №9 к таблицам resource и article планировщик выбирает план чтения по индексам в таблице resource и полного сканирования таблицы article по причине отсутствия индекса для этой таблицы.

В запросе №10 пример покрывающих индексов, когда происходит чтение из таблиц индексов для обеих таблиц resource и article, не происходит обращение к самим таблицам.

В запросе №12 по индексу B+tree к вложенному полю коллекции resource происходит чтение таблицы индексов и чтение одного документа:

В запросе №13 по индексу B+tree к вложенному полю коллекции resource происходит чтение только таблицы индексов и не происходит обращение к таблице, т.е. пример покрывающих индексов. Это получилось достигнуть с помощью явного указания “_id” = 0:

В запросе 14 по индексу B+tree с низкоселектиным полем ISBN к вложенному полю планировщик использует вложенное поле документа в индексе, как фильтр запроса и проводит поиск по имеющемуся в памяти ключу, первые поля которого совпадают с созданным нами ключом для этого запроса:

В запросе 15 по индексу B+tree с высокоселектиным полем DOI к вложенному полю оптимизатор также использует вложенное поле документа в индексе, как фильтр запроса и проводит поиск по имеющемуся в памяти ключу, первые поля которого совпадают с созданным нами ключом для этого запроса:

Таким образом, оптимизатор выносит вложенное поле документа в фильтр плана выполнения запроса, использует частично подходящий имеющийся в кэше индекс по другим полям индекса.

Таблица 2.

Результаты выполнения сложных запросов в PostgreSQL и MongoDB

	PostgreSQL			MongoDB		
	№ запроса	Время (мс)	Снижение времени %	№ запроса	Время (мс)	Снижение времени %
селективность	4	9.74	99,99	14	83	99,71
	3	5.06	99,99	15	77	99,73
фильтрация NULL-значений в запросе	6	2.88	99,99	—	—	—
	5	3.17	99,99	—	—	—
покрытие условия	9	5924.29	93,61	12	118	99,58
	10	6.12	99,99	13	72	99,75
серия сложных запросов	[9, 10, 3, 4]	[5924.29, 6.2, 5.7, 9.48]	->99,99	[12, 13, 15, 14]	[118, 9, 2, 0]	->100

Для выявления времени сжатия и передачи данных в память подсистемой хранения WiredTiger проведена серия запросов. Сравнивалось время выполнения первого запроса из серии запросов и время выполнения этого же запроса после очищения буфера, повторного создания индекса на эту коллекцию и выполнения серии запросов. Выяснилось, что время выполнения одного и того же запроса сильно сократилось с 118 мс до 22 мс в результате предварительного запроса на создание индекса на эту коллекцию. В условиях приближения к чистому эксперименту можно считать разницу во времени выполнения этих запросов временем работы WiredTiger.

Аналогичная серия запросов была проведена и в PostgreSQL. Сравнивалось время выполнения запросов в режиме после очищения буфера и в режиме серии запросов. Как видно из таблицы 2, время одиночных запросов очень близко к времени выполнения этих запросов в серии запросов для PostgreSQL. В то время, как для MongoDB время выполнения запросов в режиме серии запросов значительно сокращается по сравнению с временем выполнения этих запросов в одиночном режиме после очищения буфера. Это время в серии запросов стремиться к нулю даже при низкоселективном поле в фильтре запроса (см. Таблицу 2). Такое наблюдение является подтверждением, что в результате сжатия вся коллекция находится в оперативной памяти. Этот пример позволяет убедиться, что для MongoDB рациональным будет выполнение серии запросов к большой коллекции по сравнению с одиночными запросами. Для PostgreSQL проведение одиночных запросов или серии запросов не влияет на время их проведения.

Время зрительного анализатора человека-оператора можно рассматривать, как «время зрительной фиксации (для простых фигур — 0,2 сек и для ознакомления с ситуацией — 0,65 сек) $t_{фикс}$, время сохранения зрительного ощущения (инерция зрения) $t_{ин}$ и время зрительного восприятия $t_{воспр}$. Это неполный набор ЗА ЧО» [2]. Нас интересуют первые перечисленные три фазы.

В нашем эксперименте время сложного запроса на чтение без индексов к отношению resource с 70 млн записей составило 92,683 сек. Это значительно превышает время зрительного анализатора человека-оператора.

В нашем эксперименте время сложного запроса на чтение с B-tree индексами к отношению resource с 70 млн записей в PostgreSQL составило:

— с применением селективности: 5,06 мс,
— с учетом значений IS NOT NULL в индексе: 2.88 мс,
— с покрывающими индексами: 6.12 мс,
— к коллекции с 23 000 000 записей в MongoDB:
— с применением селективности: 77 мс
— с покрывающими индексами: 72 мс

Все значения меньше времени зрительного анализатора человека-оператора (первые три фазы). Так мы доказали эргономичность наших запросов.

Заключение

— применение индексов B-tree в сложных запросах к отношениям/ коллекциям с большим количеством записей в PostgreSQL и MongoDB сокращает время запроса на 99%;
— для PostgreSQL наиболее информативными для снижения времени сложных запросов к таблицам с большим количеством записей оказались факторы применения в запросах покрывающих индексов, высокой селективности полей индексов, явное указание значений NULL в таблице индексов;
— для запросов с B-Tree индексами к таблицам PostgreSQL важен порядок полей в индексе; уникальные и высокоселективные поля рационально прописывать на первой позиции;
— для MongoDB наиболее информативным выявлен фактор кэширования в оперативную память таблицы индексов и таблицы данных, к которым на текущий момент происходит обращение. Поэтому, важно выполнять запросы серий запросов к одной коллекции;
— при выполнении серии сложных запросов к одной коллекции время работы WiredTiger прибавляется ко времени первого запроса в серии запросов к одной коллекции; время следующих запросов в серии сокращается и стремится к 0;
— при выполнении серии сложных запросов с B-tree индексами первым запросом актуально выполнять самый короткий запрос с учетом покрывающих индексов и селективности; это сократит время всей серии запросов;
— при выполнении вышеперечисленных условий сложных запросов с B-tree индексами, время выполнения не превышает времени первых трех фаз восприятия светового сигнала зрительным анализатором человека-оператора. Поэтому мы можем утверждать, что такие запросы являются эргономичными.

ЛИТЕРАТУРА

1. Горячкин Б.С. Проектная эргономика автоматизированных систем обработки отображения информации и управления: Монография. М.: «Спутник+», 2023. Т. 1. С. 165–178. ISBN: 978-5-9973-6761-9.
2. Горячкин Б.С. Проектная эргономика автоматизированных систем обработки отображения информации и управления: Монография. М.: «Спутник+», 2023. Т. 2. С.136–171. ISBN: 978-5-9973-6782-4.
3. Елисеева Е.А., Горячкин Б.С., Виноградова М.В., Черненький М.В. Оценка времени выполнения поисковых запросов в nosql и объектно-реляционной базах данных // Динамика сложных систем — XXI век. 2022. Т.16. №2. С. 44–51. DOI: 10.18127/j19997493-202202-05.
4. Виноградов В.И., Виноградова М.В. Постреляционные модели данных и языки запросов: Учебное пособие. М.: МГТУ им. Н.Э. Баумана. 2017. 96 с. ISBN 978-5-7038-4283-6
5. Документация PostgreSQL. 64.1. https://www.postgresql.org/docs/current/indextypes.html
6. Документация PostgresPro документация 14.1.1, 14.1.2. https://postgrespro.ru/docs/postgrespro/9.5/using-explain
7. Документация PostgreSQL 65.6. Разметка страницы. https://postgrespro.ru/docs/postgrespro/9.5/planner-stats-details
8. Документация PostgresPRO. 15.1. Как работают параллельно выполняемые запросы. https://postgrespro.ru/docs/postgresql/10/how-parallel-query-works
9. Официальный сайт компании Хабр. https://habr.com/ru/companies/postgrespro/articles/330544/ (дата обращения 15.10.2024).
10. Егор Рогов. PostgreSQL 16 изнутри. М.: ДМК Пресс, 2024. 664 с. ISBN 978-5-93700-305-8
11. Позняк А.А., Горячкин Б.С. Оценка времени выполнения SQL-запросов в озерах данных // Наукосфера. 2024. № 5-1. С. 60–70 DOI: 10.5281/zenodo.11208215
12. Официальный сайт компании Хабр. Paul Randa. Почему оптимизатор запросов не анализирует содержимое буферного пула. https://habr.com/ru/companies/otus/articles/649573/ (дата обращения 17.10.2024).
13. Официальный сайт компании Хабр. Григорчук С. Оптимизация запросов базы данных на примере B2B сервиса для строителей. https://habr.com/ru/articles/461071/ (дата обращения 24.10.2024).
14. Официальный сайт компании Хабр. Robert Sheldon. 14 questions about indexes in SQL Server that you were embarrassed to ask when parsing the end of the file: перевод с англ. https://habr.com/ru/articles/247373/ (дата обращения 17.10.2024).
15. MongoDB Documentation. https://www.mongodb.com/docs/
16. Адамо. Тонете. Учебное пособие по шардингу MongoDB с лучшими практиками и рекомендациями по его включению. https://www-percona-com.translate.goog/blog/when-should-i-enable-mongodb-sharding/?_x_tr_sl=en&_x_tr_tl=ru&_x_tr_hl=ru&_x_tr_pto=rq#:~:text=Auto%20sharding%20in%20MongoDB%20automatically,datasets%20and%20high%20user%20demands (September 1, 2023)
17. Adamo Tonete A. Tutorial on MongoDB Sharding with Best Practices & When to Enable ItSeptember https://www.mongodb.com/docs/manual/tutorial/analyze-query-plan/ (1 сентября, 2023)

Журнал «Современная наука: актуальные проблемы теории и практики»