информационные системы

Компьютерные инструменты в образовании, 2024
№ 4: 82–96
УДК: 004.65
http://cte.eltech.ru
doi:10.32603/2071-2340-2024-4-82-96

ОСОБЕННОСТИ ИНДЕКСИРОВАНИЯ В СУБД POSTGRESQL

Графеева Н. Г.[1], канд. физ.-мат. наук, nggrafeeva@corp.ifmo.ru
Назаров А. А.[1], студент, ✉ artem.a.nazarov@yandex.ru

[1]Национальный исследовательский университет ИТМО,
Кронверкский пр., 49, лит. А, 197101, Санкт-Петербург, Россия

Аннотация

В статье рассматриваются особенности процесса индексирования в СУБД PostgreSQL. Проведен обзор особенностей и сравнительный анализ индексов различных типов, продемонстрированы примеры эффективного применения индексов. На основе полученных результатов сделаны выводы об индексировании как средстве оптимизации SQL-запросов.

Ключевые слова: *индексирование, PostgreSQL, B-индексы.*

Цитирование: Графеева Н. Г., Назаров А. А. Особенности индексирования в СУБД PostgreSQL // Компьютерные инструменты в образовании. 2024. № 4. С. 82–96. doi:10.32603/2071-2340-2024-4-82-96

1. ВВЕДЕНИЕ

Вопросы оптимизации производительности запросов к базам данных становятся все более актуальными в контексте постоянно растущего количества самих данных. С точки зрения систем управления базами данных (далее — СУБД), подходов к оптимизации выполнения запросов существует несколько:

— подбор технических конфигураций СУБД (к таким относятся, например, возможность кэширования запросов, число параллельных обработчиков запроса, частота сбора статистики для планировщика запросов и т. п.);
— средства ускорения доступа к данным (в основном это индексирование и партиционирование, а иногда — их совместное использование);
— оптимизация самих конструкций, используемых в запросе (минимизация просмотров исходных таблиц за счет использования функций типа CASE, эффективного использования подзапросов, оконных функций и т .п.).

Что касается технических параметров, это может быть актуально для людей с достаточным набором знаний и навыков организации данных (например, администраторов баз данных в крупных компаниях) — достаточно редко рядовой пользователь СУБД сталкивается с необходимостью изменять значения подобных параметров (поскольку для СУБД «из коробки» уже указаны значения по умолчанию, подходящие для решения большинства стандартных задач).

Если вести речь об использовании специальных конструкций в запросе, то в этой области существует определенный потолок, который не позволяет дальше масштабировать подобные решения. К примеру, даже если для поиска первого совпадения использовать EXISTS, более эффективный, чем IN, при сотнях тысяч записей любое из решений будет работать достаточно медленно.

Тем интереснее рассмотреть средства ускорения доступа к данным, а именно технологии индексирования и партиционирования. Поскольку партиционирование не всегда применимо по ряду объективных причин (связанных, например, с трудностью выбора ключа), а также не обладает таким разнообразием возможных реализаций, индексирование кажется более интересным способом оптимизации запросов для рассмотрения и анализа. Однако важно отметить, что редко когда пользователь глубоко погружен в принцип работы индекса и его структуру, что может повлечь за собой не только отсутствие оптимизации, но порой и ухудшение итоговых метрик.

Стоит отметить, что проблема индексирования актуальна, и существует достаточно большое количество работ, посвященных обзору индексов различных типов (например, [1]), исследованию применения индексов для решения прикладных задач (например, [2]), тестированию производительности запросов с использованием индексов различных типов в зависимых от типов данных (например, [3]), а также подробному техническому описанию использования индексов в контексте конкретной СУБД (например, серия статей от разработчиков компании PostgresProfessional [4]).

В связи с этим кажется, что вопросы правильного подбора и использования индексов (с прозрачным техническим обоснованием и наглядными примерами) были и остаются очень важными в сфере оптимизации производительности запросов. Им и посвящена предложенная статья. Важно отметить, что в различных СУБД организация и использование индексов имеют свои особенности. В данной статье рассмотрено применение индексов на примере PostgreSQL.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ПО ИНДЕКСИРОВАНИЮ

Индексы являются объектами базы данных (наряду с функциями, таблицами, представлениями). При этом основное их назначение заключается в обеспечении прямого доступа к кортежу по ключу вместо сканирования таблицы целиком [5]. В свою очередь, индексирование — это сам процесс обеспечения быстрого доступа к данным.

При этом во многих источниках также отмечаются и основные недостатки индексов, связанные с:

— дополнительным объемом памяти, необходимым для хранения индексов;
— частой перестройкой индексов в связи с обновлением или вставкой данных в таблицы.

Классификация индексов по структуре для СУБД PostgreSQL (для 17-й версии, последней на момент написания этой статьи) включает в себя: B-деревья, хэш-индексы, GiST и SP-GiST, а также GIN и BRIN-индексы [6].

Все индексы в PostgreSQL хранятся вне области основных данных таблицы (вне «кучи») как массив страниц фиксированного размера. Соответственно, для получения данных в таком случае существуют два варианта реализации [7]:

1. Последовательно сканировать всю таблицу (так же, как это работает без применения индекса).

2. Обратиться к индексу, а потом — произвольно читать «кучу» несколько раз (так как близкое расположение элементов в индексе не гарантирует близкое расположение строк в основной области данных таблицы) [6].

Выбор, который был сделан оптимизатором в конкретном запросе, можно посмотреть с помощью команды EXPLAIN ANALYZE.

В данной статье авторам хотелось бы в первую очередь акцентировать внимание на различных типах индексов (и особенно пристально рассмотреть B-tree вследствие его широкой популярности). Хотя и в уровне применения индексов существуют несколько особенностей, заслуживающих внимания (например, порядок столбцов при использовании составного индекса, индексирование выражений или возможность частичного использования составного индекса в некоторых случаях).

3. ТИПЫ ИНДЕКСОВ В POSTGRESQL И ИХ ОСОБЕННОСТИ

3.1. B-индексы

Структура индекса B-дерева представлена на рис. 1.

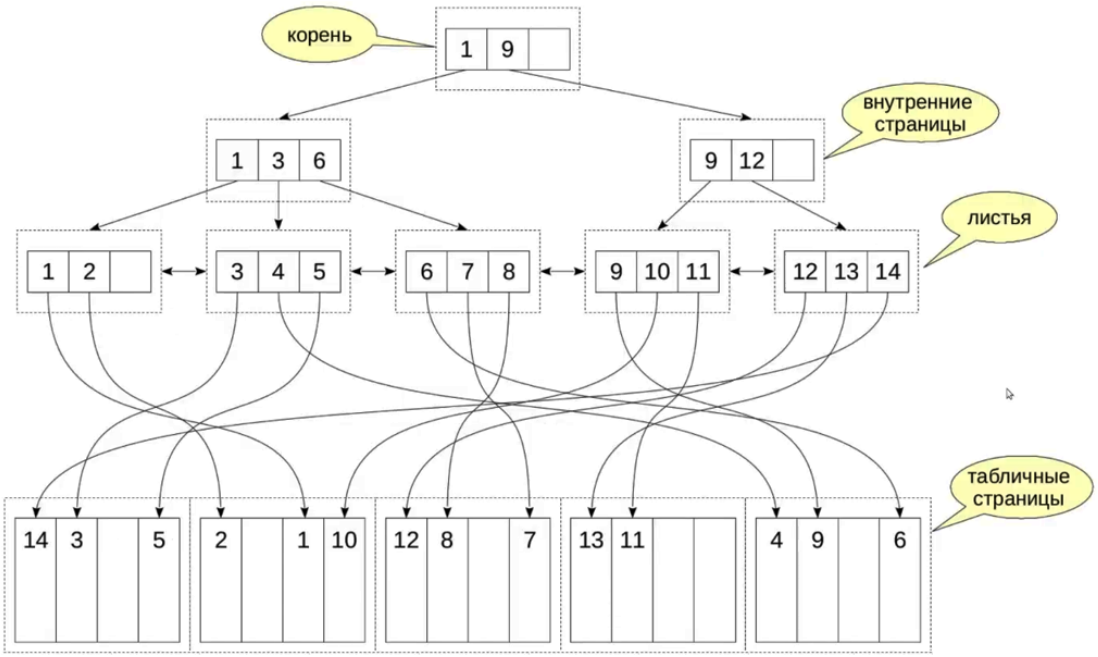

Рис. 1. Структура B-индекса [8]

При организации индекса в виде сбалансированного дерева определены несколько уровней: непосредственно корень дерева, внутренние страницы (1-ый уровень на схеме), листовые страницы (2-ой уровень в схеме) и сама таблица (последний уровень). В случае удаления, вставки или изменения значения может потребоваться перестраивать указатели на каждом из этих уровней, вследствие чего рекомендуется с осторожностью использовать B-индекс для часто изменяющихся таблиц.

B-индекс может быть применен при поиске и фильтрации значений, для которых используются операции сравнения, в том числе сравнения с шаблонами (*LIKE/ILIKE*), при

которых шаблон задается для проверки префикса строки [6]. Принцип его работы основан на алгоритме бинарного поиска со сложностью $O(\log n)$ в худшем случае, что приводит к наибольшей эффективности при большом разбросе значений в столбцах.

В СУБД PostgreSQL этот индекс строится по умолчанию при использовании соответствующей команды CREATE INDEX, а также автоматически создается при создании правил целостности PRIMARY KEY и UNIQUE.

3.2. Hash

Согласно документации, индекс этого типа хранит 32-битный код, полученный при использовании хэш-функции с каждым из значений соответствующего столбца. В связи с этим поддерживаются только операции равенства. Важно отметить, что временная сложность поиска конкретной строки составляет $O(1)$ в отличие от, например, $O(\log n)$ для B-индексов; но для проверок на диапазоны — все значительно сложнее: если для дерева можно сразу попасть в нужную внутреннюю страницу, в данном случае придется n раз брать хэш от каждого из значений диапазона. Также нельзя не отметить возможность коллизий, которые также приходится учитывать при проектировании и использовании индексов такого рода. Структура хэш-индекса представлена на рис. 2.

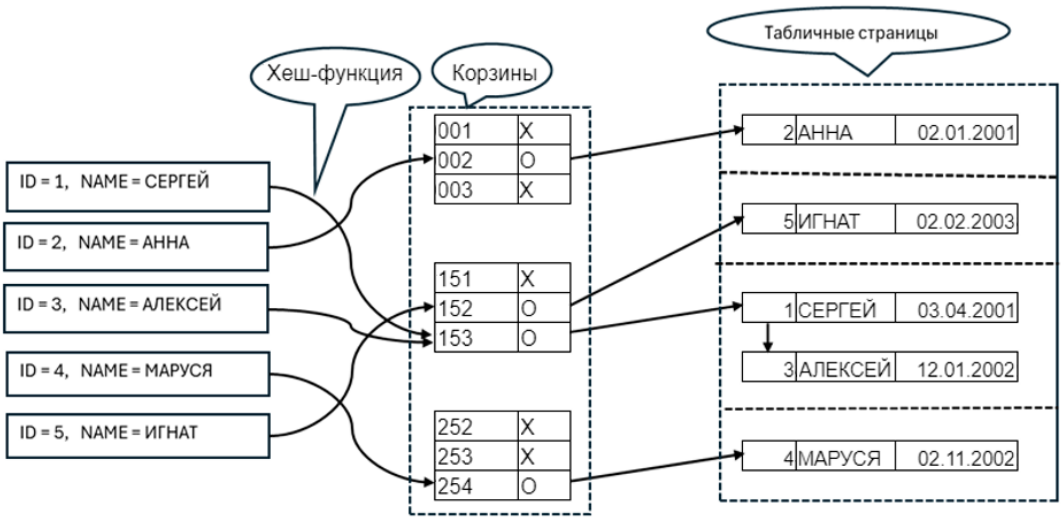

Рис. 2. Структура хэш-индекса

Используется хэш-индекс следующим образом:

— создается хэш-функция;
— создается индекс, состоящий из корзин (buckets);
— при выполнении запроса к значению применяется хэш, на основании которого выбирается одно из значений в корзине, в котором лежит указатель на страницу с подходящими строками из базы данных.

Для создания индекса используется команда CREATE INDEX ... USING HASH.

3.3. GiST и SP-GiST

GiST (Generalized Search Tree) — обобщенное поисковое дерево. Структура является схожей с B-индексом, но также подходит и для сравнения более сложных типов данных (на-

пример, геометрических точек, массивов, текстовых данных). Такой подход возможен за счет расширения стандартного дерева специальной функцией согласованности (реализуемой по-своему для каждого поддерживаемого семейства операторов). При использовании функция вызывается для каждой записи и определяет, нужно ли спускаться в соответствующее поддерево (или обращаться в «кучу» таблицы) [4].

Полезно подчеркнуть, что механизм GiST индекса является масштабируемым и позволяет реализовать идею индексирования данных в СУБД PostgreSQL в зависимости от их типизации.

SP-GiST, в отличие от GiST, является несбалансированным деревом, что позволяет в дополнении к вышеперечисленным реализовать широкий спектр несбалансированных дисковых структур данных, например, квадродеревья, деревья kd и R-деревья.

В большинстве случаев индексы SP-GiST также являются полезными при сложных комплексных структурах данных и реализации сравнения по нестандартным операторам.

Для создания индексов используется традиционная команда с USING GIST/SPGIST.

3.4. GIN

Структура GIN-индекса отражена на рис. 3 и наиболее понятна в контексте задачи текстового поиска.

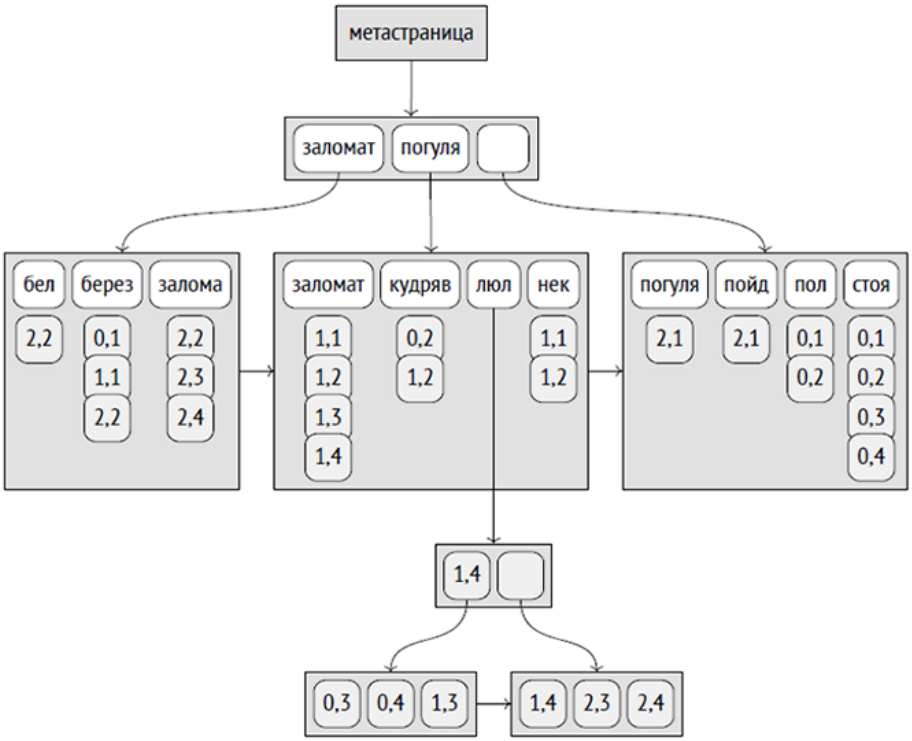

Рис. 3. Структура GIN-индекса [9]

Название индекса расшифровывается как инвертированный индекс (Generalized Inverted Index). Этот метод доступа работает с типами данных, значения которых являются составными и состоят из элементов (так, например, в контексте полнотекстового поиска документы состоят из лексем). При этом индексируются не сами значения,

а только их элементы, причем от каждого элемента по индексу можно перейти ко всем значениям, в которых этот элемент встречается. Хорошая аналогия для этого метода — предметный указатель в конце книги. В указателе собраны все важные термины, и для каждого приведен список страниц, на которых этот термин упоминается [8]. Из особенностей можно также отметить, что сами значения, содержащие элементы, могут изменяться, но сами элементы из индекса не удаляются никогда [4].

Для создания индекса используется команда CREATE INDEX ... USING GIN.

3.5. BRIN

В отличие от остальных индексов, BRIN-индексы (Block Range Index) ориентированы не для быстрого поиска нужных строк, а для того, чтобы избежать просмотра заведомо ненужных. Этот метод доступа создавался в расчете на таблицы размером в десятки терабайт. Индекс делит таблицу на зоны (range) и хранит границы значений внутри каждой из зон. Сами индексы не хранят по записи на каждую строку (в отличие от других типов индексов), а хранят обобщенную информацию на уровне зон, за счет чего более экономно расходуют память.

Нулевая страница BRIN-индекса — метастраница с информацией о структуре индекса. Далее находятся страницы со сводной информацией. Каждая индексная строка на такой странице содержит сводку по какой-то одной зоне. Между метастраницей и сводными данными располагается карта зон (range map), которая иногда называется обратной картой (reverse range map или revmap).

Отличие от B-индекса можно проследить на рис. 4: вместо хранения всех значений на нижнем уровне хранятся только границы зон.

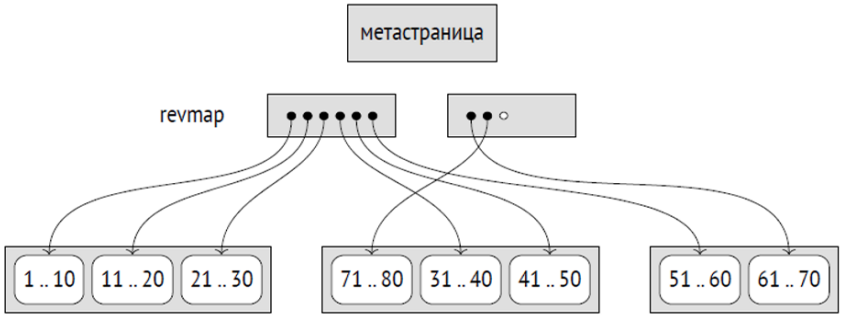

Рис. 4. Особенности структуры BRIN-индексов [9]

Для создания индекса используется команда CREATE INDEX ... USING BRIN.

Таким образом, по результатам анализа различных типов индексов в СУБД PostgreSQL была составлена сравнительная таблица 1.

4. ПРИМЕРЫ ЭФФЕКТИВНОГО ПРИМЕНЕНИЯ ИНДЕКСОВ В СУБД PostgreSQL

При тестировании внимание авторов в первую очередь было обращено на B-индексы вследствие их широкого распространения и адаптации под большое число разнохарактерных задач. Однако также было проведено сравнение применения B-индексов и Hash-индексов для некоторого типа запросов. Ниже представлено описание нескольких экспериментов и выводы, основанные на результатах проведенных экспериментов.

Таблица 1. Особенности различных типов индексов (составлено авторами)

Название индекса	**B**	**Hash**	**GiST**
Структура	Сбалансированное дерево	Хэш-функция и корзины с указателями	Сбалансированное дерево
Поддерживаемые операции	<, <=, =, >=, >	=	Набор операторов различается в зависимости от типа данных столбца
Применимость	При поиске маленького % данных с большим размахом возможных значений	При точной проверке на равенство	Для сравнения сложных комплексных структур (геометрических, массивов)
Команда создания	CREATE INDEX	CREATE INDEX USING HASH	CREATE INDEX USING GIST

Название индекса	**SP-GiST**	**GIN**	**BRIN**
Структура	Несбалансированное дерево	B-дерево списков или B-деревьев	Блочная структура
Поддерживаемые операции	Набор операторов различается в зависимости от типа данных столбца	Набор операторов различается в зависимости от типа данных столбца	Набор операторов различается в зависимости от типа данных столбца
Применимость	Для сравнения несбалансированных структур (например, построения R-деревьев)	Для индексирования составных частей элемента, например текстов	Для отсортированных значений
Команда создания	CREATE INDEX USING SPGIST	CREATE INDEX USING GIN	CREATE INDEX USING BRIN

5. УСЛОВИЯ ЭКСПЕРИМЕНТОВ

Для тестирования производительности с использованием B и Hash-индексов была создана таблица Person с синтетическими данными. Скрипт создания представлен на рис. 5.

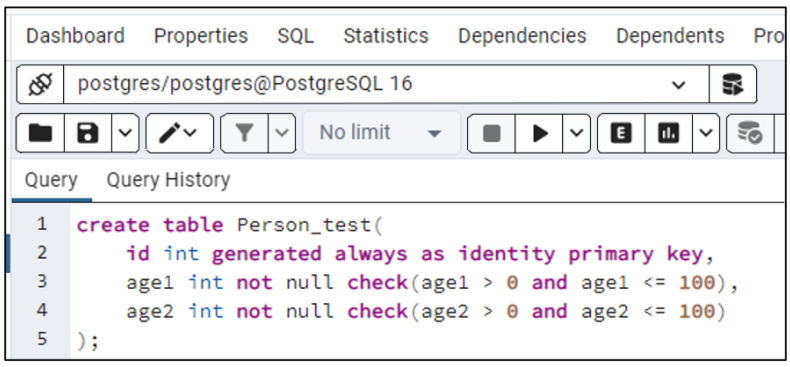

Рис. 5. Скрипт создания тестовой таблицы

Эксперимент 1

Утверждение: B-индекс эффективен при отборе маленького процента строк от общего объема таблицы.

Описание эксперимента: для одного из столбцов таблицы Person (age2) был построен индекс, при этом столбец age2 определен точно так же, как и столбец age1. Далее таблица была равномерно заполнена значениями в диапазоне от 1 до 100: на каждое значение столбцов age1 и age2 приходилось порядка 1 % записей. Процедура заполнения представлена на рис. 6.

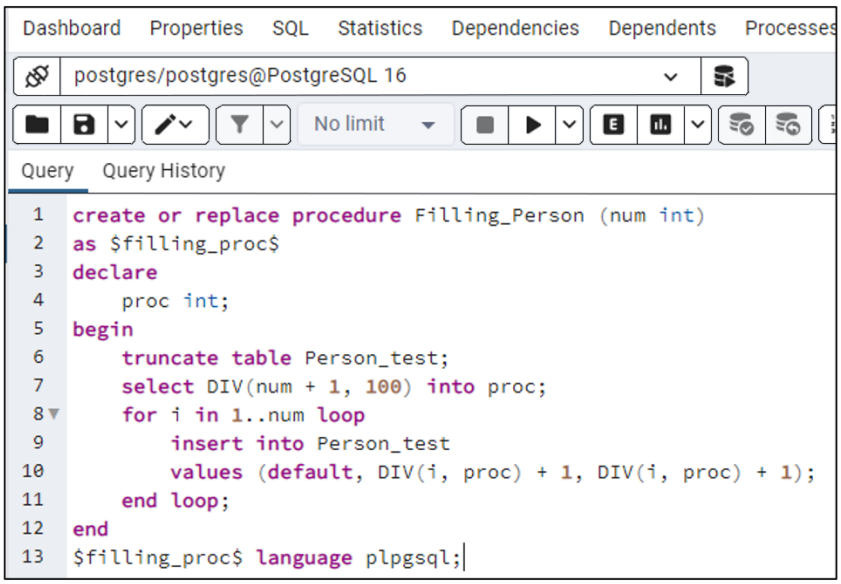

Рис. 6. Процедура для генерации данных

Впоследствии была произведена выборка значений с условием age1 < 2 (~ 1 % записей от общего числа) и age2 < 2 при условии разного объема таблицы Person.

Результаты представлены на рис. 7.

Выводы: можно отметить, что при росте числа строк в таблице индексное сканирование, предложенное оптимизатором для age2, оказывается значительно эффективнее последовательного сканирования таблицы.

Эксперимент 2

Утверждение: использование B-индекса может быть неэффективным при отборе большого процента строк от общего объема таблицы.

Описание эксперимента: по сравнению с предыдущим тестом было изменено условие выборки (age2 > 2, ~ 98 % от общего числа строк). Измерение было проведено при явном требовании на индексное сканирование (поскольку оптимизатор предлагает последовательное сканирование без использования индекса).

Результаты представлены на рис. 8.

Выводы: при выборке большого числа строк явное использование индекса начинает проигрывать в производительности, поскольку тратится дополнительное время на обращение к индексу вместо последовательного сканирования таблицы.

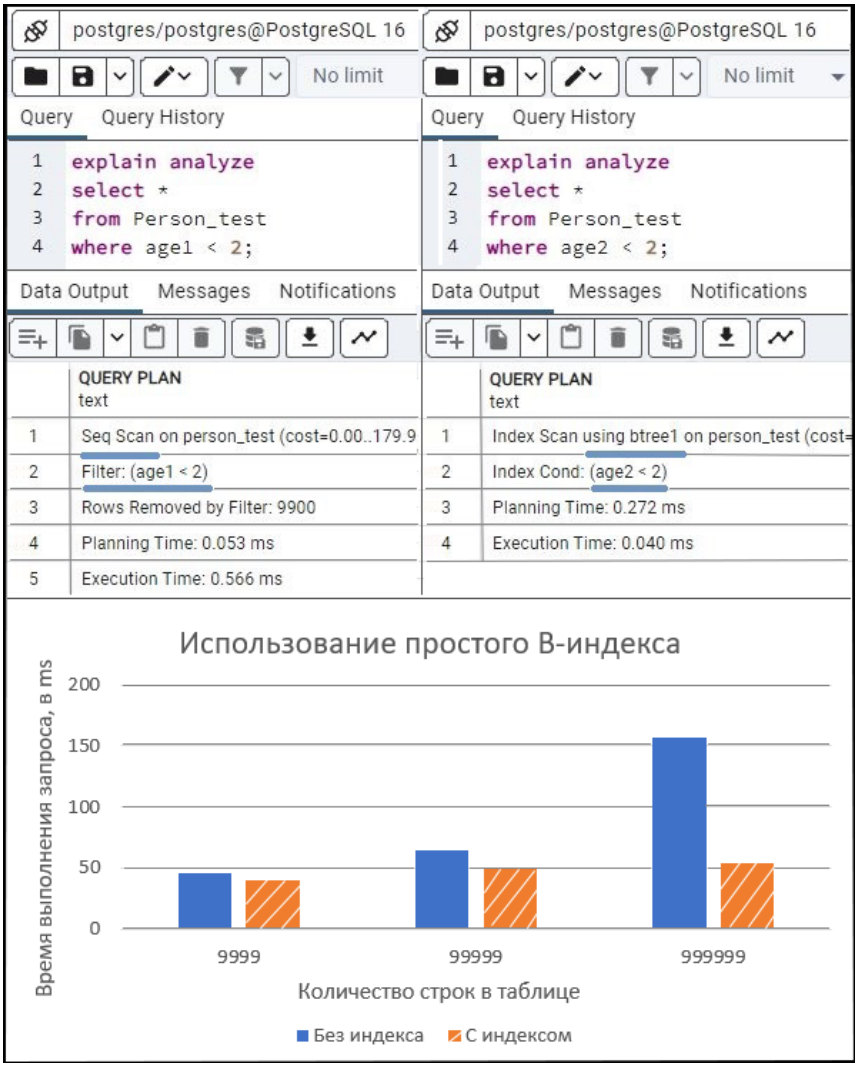

Рис. 7. Результаты тестирования эффективности простого B-индекса

Эксперимент 3

Утверждение: при грамотном использовании составного индекса он может быть частично использован оптимизатором как простой (для фильтрации выборок, в которых используются только первые столбцы из определения индекса).

Описание эксперимента: Для проведения эксперимента был создан составной индекс для таблицы Person по полям age1 и age2, а затем проведена выборка записей с условием (age1 < 2 and age2 < 2) и условием: (age1 < 2).

Результаты представлены на рис. 9.

Выводы: планировщик использует составной индекс как простой при выборке записей по условию, в котором используются только первые столбцы индекса.

Эксперимент 4

Утверждение: индекс будет применяться только при точном использовании столбцов индекса при фильтрации выбираемых записей (то есть нельзя допускать построение выражений из индексных полей).

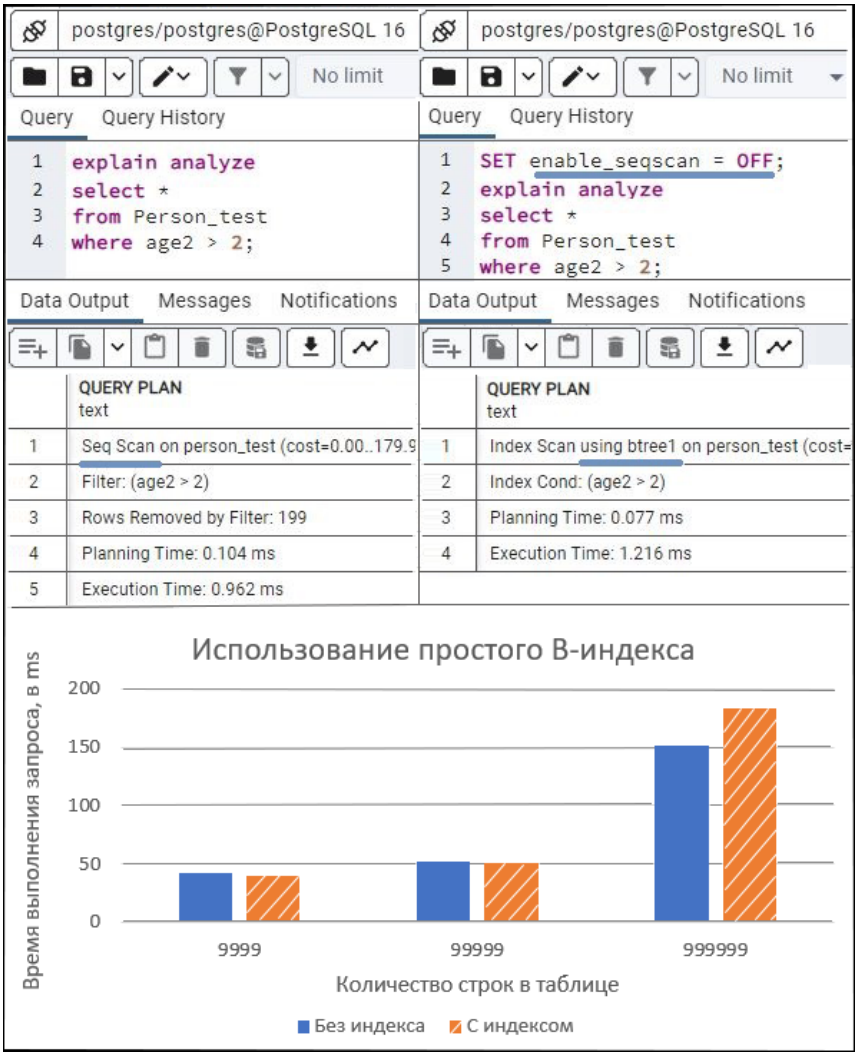

Рис. 8. Результаты теста на неэффективное использование индекса

Описание эксперимента: были использованы данные первого эксперимента с выборкой по условию age2 < 2 с небольшим изменением. Теперь условие было изменено на age1+1 < 3 (количество строк для выборки остается таким же).

Результаты представлены на рис. 10.

Выводы: при использовании индексируемого столбца с выражением планировщик перестает рекомендовать применение индекса, что при больших данных может существенно ухудшить производительность запроса (хотя следует отметить, что специальные приемы для построения индекса по выражениям также существуют [6], но они не будут рассмотрены в настоящей статье).

Эксперимент 5

Утверждение: оптимизатор может ошибиться в аргументации использования индекса, если будет использовать устаревшую статистику.

Описание эксперимента: Для реализации эксперимента было использовано условие age2 < 2 (когда размер выборки составляет 1 % от всей таблицы и, как следствие, индекс эффективен). Такое условие выборки было рассмотрено ранее в эксперименте 1. Затем

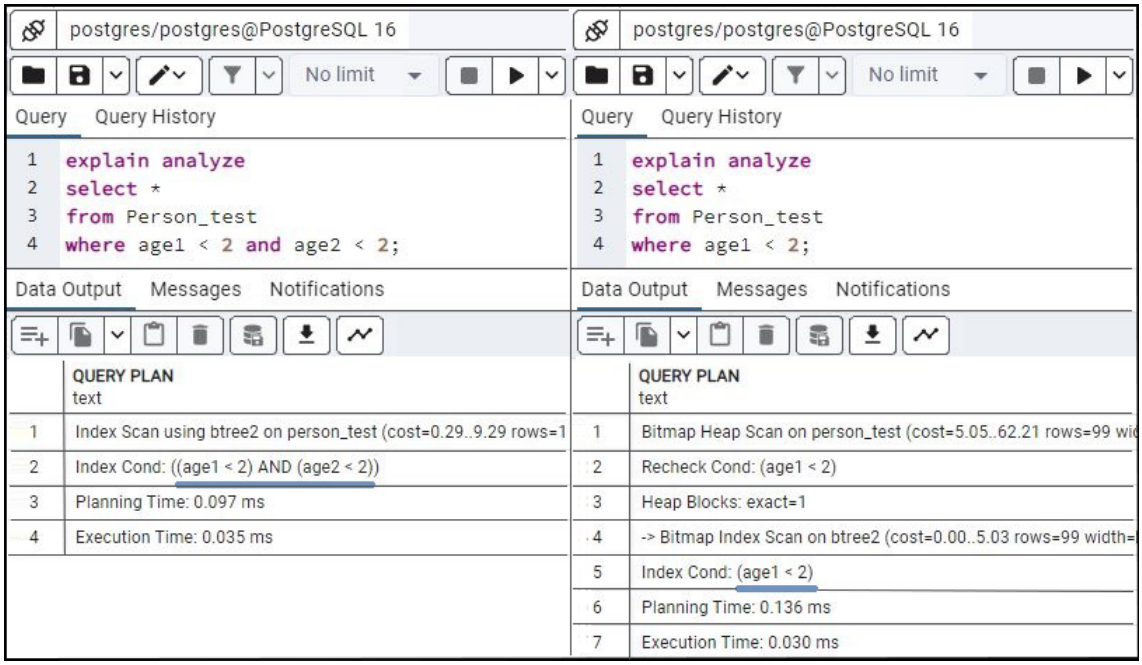

Рис. 9. Планы выполнения запросов при использовании составного индекса

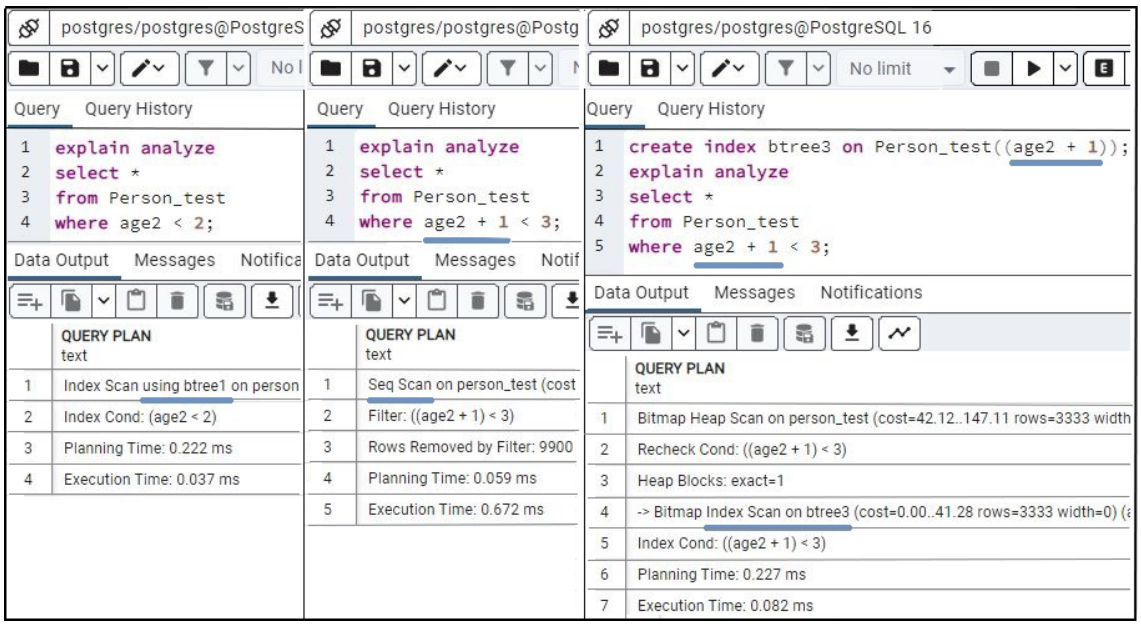

Рис. 10. Планы выполнения запросов при использовании индекса с выражением

было произведено удаление всех строк, не удовлетворяющих этому условию, и снова реализована выборка с условием age2 < 2 (когда размер выборки составляет всю таблицу целиком — и индекс неэффективен).

Результаты представлены на рис. 11.

Выводы: можно заметить, что сразу после удаления планировщик продолжает предлагать использовать индекс, хотя это уже не является эффективным, поскольку область

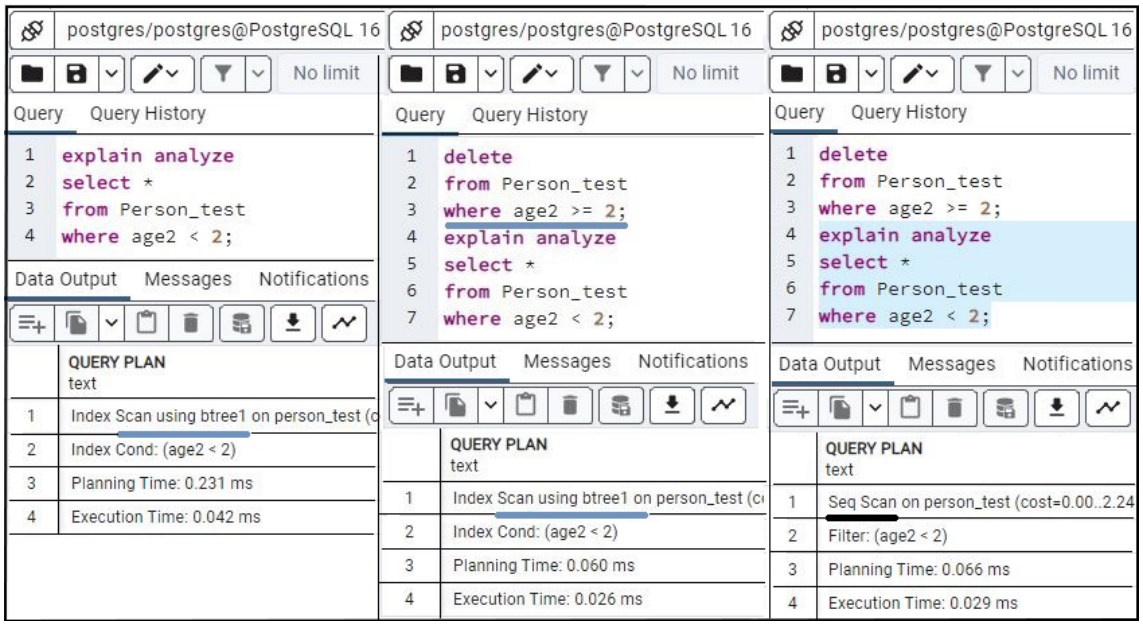

Рис. 11. Планы выполнения запросов при использовании индекса до и после сборки мусора

выборки полностью совпадает с объемом всей таблицы. Это связано с тем, что после удаления строк еще не был проведен сбор статистики (по умолчанию был установлен интервал для сбора статистики в 1 минуту). Если же попробовать произвести тот же самый запрос через минуту — уже предлагается сканирование таблицы. Для решения подобных проблем можно либо изменять значения конфигурационных параметров самой СУБД, либо использовать явное указание оптимизатору на использование индексов (или на запрет их использования в зависимости от задачи).

Эксперимент 6

Утверждение: при условиях проверки на равенство Hash-индекс может быть более эффективен, чем B-индекс.

Описание эксперимента: для проверки была использована все та же таблица Person, при этом на одном из полей оставался построенный ранее B-индекс (*age2*), а для другого был построен Hash-индекс для поля *age1*. Далее было проведена выборка по условию на равенство с константой.

Результаты: представлены на рис. 12.

Выводы: можно отметить, что на большом количестве строк hash-индекс не уступает, а даже чуть быстрее, чем B-индекс. При этом также стоит отметить, что hash-индекс может быть еще более эффективен при снижении количества строк с указанным значением индексируемого столбца (и стремлении к уникальности столбца в целом).

6. ЗАКЛЮЧЕНИЕ

Таким образом, по результатам проведенных экспериментов можно сделать ряд выводов:

1. В СУБД PostgreSQL существует большое количество индексов разных типов, имеющих разную структуру и оптимизированных для решения задач различного характера.

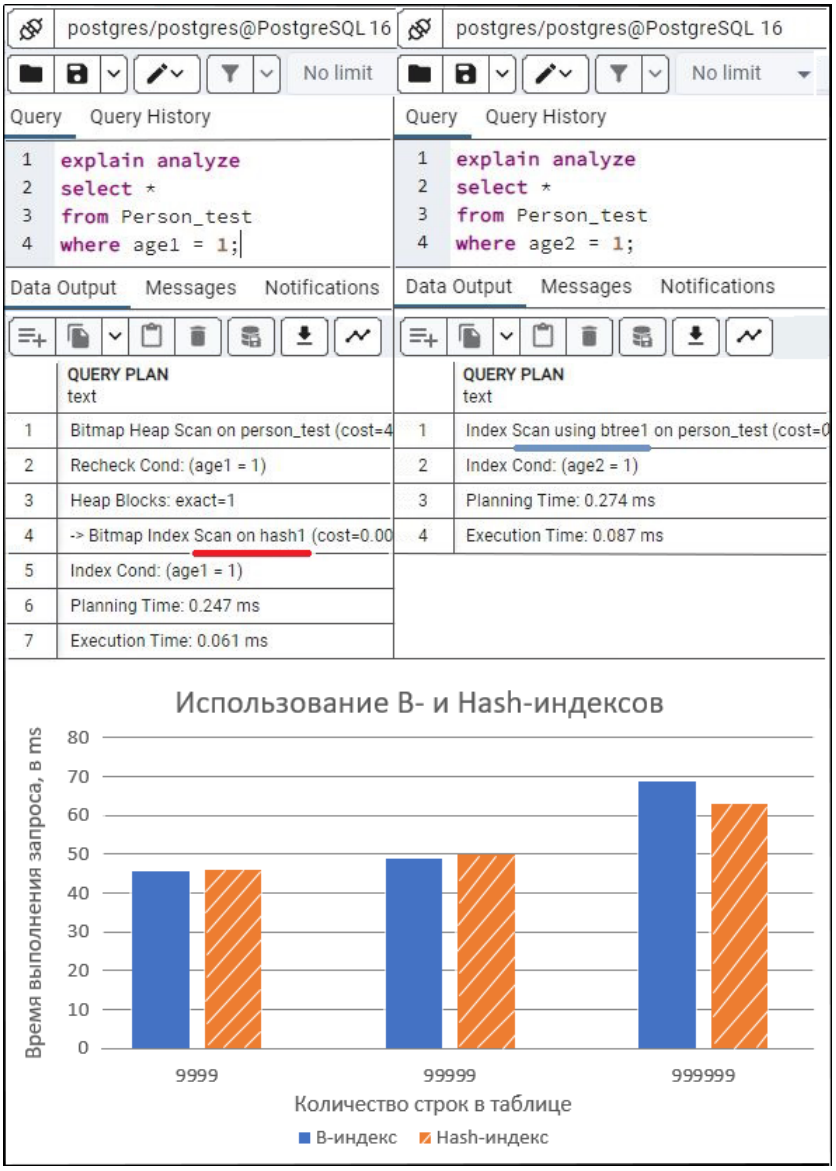

Рис. 12. Результаты тестирования производительности при использовании Hash- и B-индексов

2. При правильном использовании индексов возможно получить выигрыш в несколько раз в производительности запросов; но при неэффективном использовании — потерять примерно столько же.
3. Индексы не всегда являются спасением: требуется тщательно оценивать каждую конкретную ситуацию и индивидуально принимать решение о внедрении тех или иных индексов в таблицы.
4. В большинстве случаев оптимизатор выбирает наиболее эффективный план запроса по умолчанию, однако бывают ситуации, когда это требуется делать пользователю вручную.
5. Для того чтобы правильно оценить применимость индексов, пользователю необходимо разбираться как в структуре индексов, так и в принципах работы СУБД, связанных с использованием индексов в различного рода запросах.

Список литературы

1. *Морозов С. В., Нестеров С. А.* Сравнительный анализ типов индексов в СУБД SQLServer и PostgreSQL // Системный анализ в проектировании и управлении: сборник научных трудов XXVIII международной научно-практической конференции. Т. 2. СПб., 2024. С. 485–491.
2. *Головлев А. А.* Исследование и разработка подходов к индексированию больших текстов в СУБД Oracle // Теория и практика современной науки. 2018. № 12. С. 108–111.
3. *Вендин А. С.* Использование индексов в реляционных СУБД // Успехи современной науки. 2017. Т. 6, № 3. С. 38–41.
4. Индексы в PostgreSQL (серия статей) [электронный ресурс]. URL: https://habr.com/ru/companies/postgrespro/articles/326096/ (дата последнего обращения: 10.11.2024).
5. *Кузнецов С. Д.* Основы современных баз данных [Электронный ресурс]. URL: https://citforum.ru/database/osbd/contents.shtml (дата последнего обращения: 20.12.2024).
6. Документация СУБД PostgreSQL (раздел Индексы) [Электронный ресурс]. URL: https://postgrespro.ru/docs/postgresql/16/indexes (дата последнего обращения 20.12.2024).
7. *Дейт К. Дж.* Введение в системы баз данных, 8-е издание: Пер. с англ. М.: Издательский дом «Вильямс». 2005. 1328 с.: ил.
8. Оптимизация запросов. Индексный доступ [Электронный ресурс]. URL: https://edu.postgrespro.ru/qpt-13/qpt_04_indexscan.html (дата последнего обращения 20.12.2024).
9. *Рогов Е. В.* PostgreSQL 16 изнутри. М.: ДМК Пресс, 2024. 664 с.
10. *Новиков Б. А., Горшкова Е. А., Графеева Н. Г.* Основы технологий баз данных: учеб. пособие (под ред. Е. В. Рогова). 2-е изд., М.: ДМК Пресс, 2020. 582 с.

Поступила в редакцию 27.11.2024, окончательный вариант — 20.12.2024.

Графеева Наталья Генриховна, канд. физ.-мат. наук, доцент, доцент, университет ИТМО, nggrafeeva@corp.ifmo.ru

Назаров Артем Александрович, студент магистратуры, университет ИТМО, ✉ artem.a.nazarov@yandex.ru

Computer tools in education, 2024

№ 4: 82–96

http://cte.eltech.ru

doi:10.32603/2071-2340-2024-4-82-96

Characteristic of Indexing in the PostgreSQL Database Management System

Grafeeva N. G.[1], Cand. Sc., Associate Professor, nggrafeeva@corp.ifmo.ru
Nazarov A. A.[1], Student, ✉ artem.a.nazarov@yandex.ru

[1]ITMO University, 49 Kronverksky, bldg. A, 197101, Saint Petersburg, Russia

Abstract

The article discusses the features of the indexing process in the PostgreSQL database. The authors conducted a review and comparative analysis of indexes of various types, demonstrated examples of effective use of indexes. Based on the obtained results, conclusions about indexing as one of the means of optimizing SQL queries were drawn.

Keywords: *indexing, PostgreSQL, B-indexes.*

Citation: N. G. Grafeeva and A. A. Nazarov, “Characteristic of Indexing in the PostgreSQL Database Management System,” *Computer tools in education*, no. 4, pp. 82–96, 2024 (in Russian); doi:10.32603/2071-2340-2024-4-82-96

References

1. S. V. Morozov and S. A. Nesterov, “Comparative Analysis of Index Types in SQL Server And PostgreSQL DBMS,” in *Proc. of 28th International Scientific, Educational and Practical Conference SYSTEM ANALYSIS IN ENGINEERING AND CONTROL. St. Petersburg 16 Jul 2024*, vol. 2, pp. 485–491, 2024 (in Russian); doi:10.18720/SPBPU/2/id24-202
2. A. A. Golovlev, “Research and Development of Approaches to Indexing of Big Texts in Oracle DBMS,” *Teoriya i praktika sovremennoi nauki*, vol. 12, pp. 108–111, 2018 (in Russian).
3. A. S. Vendin, [Using Indexes in RDBMS] “Ispol’zovanie indeksov v relyatsionnykh SUBD,” *Uspekhi sovremennoi nauki*, vol.6, no. 3, pp. 38–41, 2017 (in Russian).
4. Postgres Professional, “Indexes in PostgreSQL,” in *habr.com*, 2017 (in Russian). [Online]. Available: https://habr.com/ru/companies/postgrespro/articles/326096/
5. S. D. Kuznetsov, “Basics of modern databases,” in *citforum.ru*, 2024 (in Russian). [Online]. Available: https://citforum.ru/database/osbd/contents.shtml
6. The PostgreSQL Global Development Group, “PostgreSQL 16.6 Documentation. Chapter 11. Indexes,” in *postgrespro.ru*, 2024 (in Russian). [Online]. Available: https://postgrespro.ru/docs/postgresql/16/indexes
7. C. J. Date, *Introduction to Database Systems*, 8th edition, Moscow: Williams Pbl. h., 2005 (in Russian).
8. Postgres Professional, “Query Optimization. Index Access,” in *postgrespro.ru*, 2024. [Online]. Available: https://edu.postgrespro.ru/qpt-13/qpt_04_indexscan.html
9. E. V. Rogov, *PostgreSQL 16 inside*, Moscow: DMK Press, 2024 (in Russian).
10. B. A. Novikov, E. A. Gorshkova, and N. G. Grafeeva, *Fundamentals of Database Technologies: a textbook* E. V. Rogov ed., 2nd edition, Moscow: DMK Press, 2020 (in Russian).

Received 27-11-2024, the final version — 20-12-2024.

Natalia Grafeeva, Cand. of Sciences (Phys.-Math.), Associate Professor, ITMO University, nggrafeeva@corp.ifmo.ru

Artem Nazarov, Master’s Degree student, ITMO University, ✉ artem.a.nazarov@yandex.ru