МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Домашняя работа №__2__
по дисциплине «Постреляционные базы данных»

Тема: «Индексы в СУБД PostgreSQL»

ИСПОЛНИТЕЛЬ: __Журавлев Н.В.___
ФИО
группа ИУ5-24М _____
подпись
"26"_мая____2024 г.

ПРЕПОДАВАТЕЛЬ: _Виноградова М.В_
ФИО
_____
подпись
"__"_____202_ г.

Москва - 2024

Цель работы

Изучить понятие индексов и исследовать принципы их работы в СУБД PostgreSQL.

Задание

1. Создать и заполнить тестовую базу данных для СУБД PostgeSQL [1, 2]. Можно использовать БД из ЛР.
2. Создать копию БД и назначить для нее индексы (по варианту).
3. Провести ряд экспериментов (по варианту) по оценке времени выполнения запросов:
    1. С применением индексов [3, 4, 5, 6], фиксированным количеством записей (1000 шт.) и вариацией селективности (по таблице 1).
    2. Без применения индексов, фиксированным количеством записей (1000 шт.) и вариацией селективности (по таблице 2).
    3. С применением индексов, зафиксированной селективностью (10%) и вариацией количества записей (по таблице 1).
    4. Без применения индексов, зафиксированной селективностью (10%) и вариацией количества записей (по таблице 2).

Если в задании по варианту используется только одна таблица, то все вариации проводятся по одной таблице.

Под селективностью [7, 8, 9 ,10] понимается доля записей с определенным значением атрибута среди всех записей.

Для установки определенного уровня селективности рекомендуется использовать программные сценарии (процедуры), которые позволят быстро заполнить таблицу так, чтобы у определенного значения атрибута был заданный уровень селективности.

Рекомендуется для каждой пары значений {кол-во записей, уровень селективности} создавать новую таблицу или пересоздавать существующую. Процедура для генерации данных [11, 12, 13] с заданной селективностью может включать в себя:

- Скрипт создания таблицы с установкой необходимых индексов.

- Цикл, заполняющий таблицу записями со случайными значениями.

В цикле может быть задан счетчик общего числа значений, а также счетчик селективности для определенного значения атрибута.

Каждый эксперимент выполняется несколько раз, фиксируется каждое время выполнения и вычисляются среднее время и отклонение.

4. Для каждого эксперимента определить:
   • план выполнения запроса - explain,
   • ожидаемое время выполнения - explain, просуммировать все «правые» значения cost (Total Cost) в узлах запроса,
   • реальное время выполнения - explain analyze [4, 5].
5. Сравнить планы выполнения запросов и время их выполнения (с применением индексов и без них).
6. Варьирование выполнять:
   1. (Базовый):
      1. Селективность, %: {1, 3, 10, 20};
      2. Количество записей, шт.: {1.000 и 1.000.000}.
   2. (Расширенный и дополнительный):
      1. Селективность, %: {1, 2, 3, 5, 10, 15, 20, 30};
   3. (Дополнительный):
      1. Количество записей, шт.: {100, 1.000, 10.000, 100.000, 1.000.000}.

Если индекс не запускается, то необходимо обновить статистику и/или настроить оптимизатор.

Две таблицы: у первой индекс <Индекс 1>, у второй индекс <Индекс 2>. Выполнить запрос WHERE с подзапросом из другой таблицы: 1) с варьированием селективности поля в таблице подзапроса. 2) с варьированием селективности поля внешней таблицы, по которой происходит выборка WHERE. Также выполнить варьирование количества записей. Использованные индексы B-дерево и Хэш.

Ход работы

Запрос для экспериментов выглядит следующим образом:

SELECT * FROM first_table WHERE first_table.first_atr IN (SELECT second_table.first_atr FROM second_table WHERE second_table.first_atr = 0);

Из first_table выбираются все строки, у которых first_atr есть в тех first_atr в таблице second_table, которые равны 0.

План запроса с индексами представлен на рис. 1.

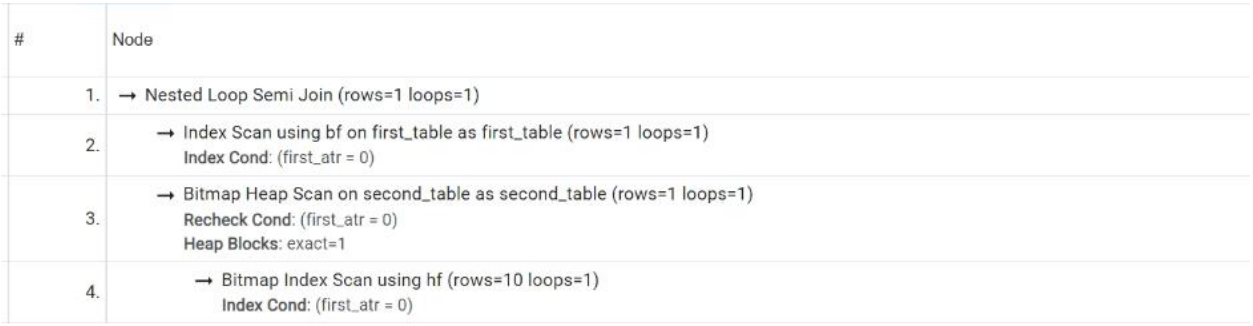

Рисунок 1 - План запроса с индексами

План запроса без индексов представлен на рис. 2.

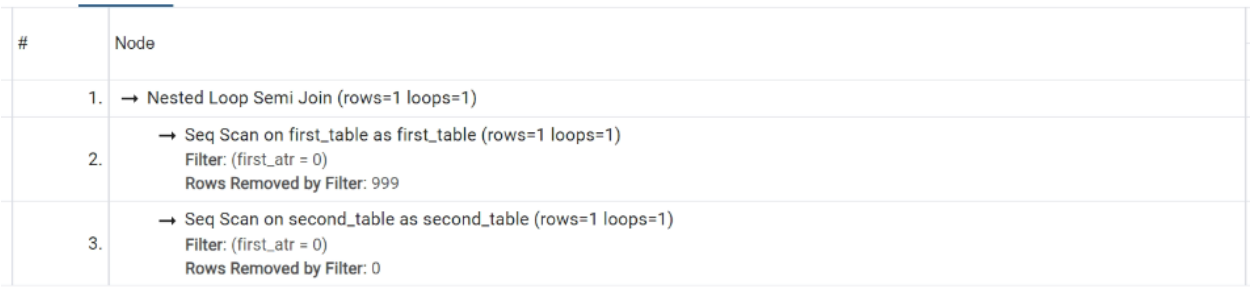

Рисунок 2 - План запроса без индексов

Сравнение времени и стоимости выполнения запросов в виде графика (с индексами и без индексов) – зависимость времени и стоимости выполнения запросов от % селективности и количества записей, представлены на рис. 3-18.

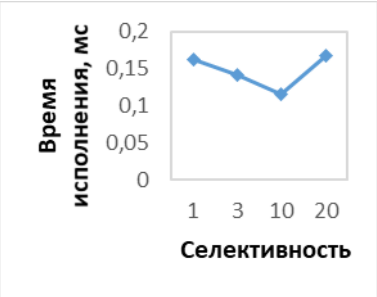

Рисунок 3 – Зависимость селективности от времени для поля в подзапросе без индексов

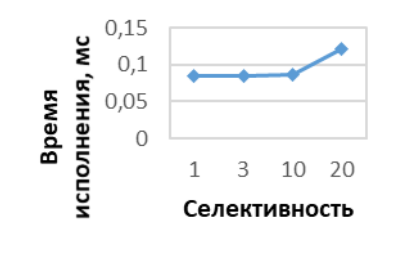

Рисунок 4 – Зависимость селективности от времени для поля в подзапросе с индексами

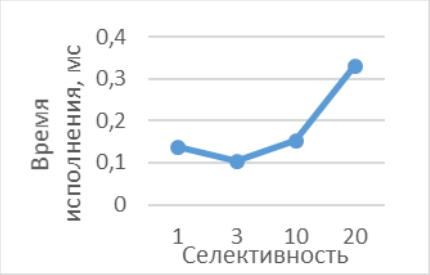

Рисунок 5 – Зависимость селективности от времени для поля внешней таблицы без индексов

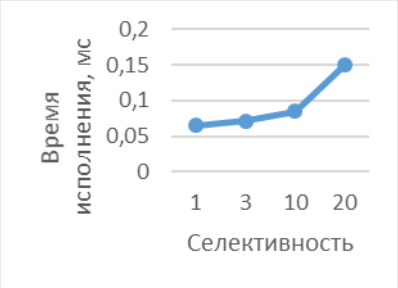

Рисунок 6 – Зависимость селективности от времени для поля внешней таблицы с индексами

Рисунок 7 – Зависимость селективности от стоимости для поля в подзапросе без индексов

Рисунок 8 – Зависимость селективности от стоимости для поля в подзапросе с индексами

Рисунок 9 – Зависимость селективности от стоимости для поля внешней таблицы без индексов

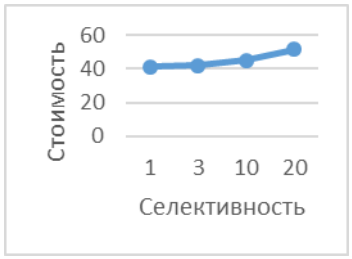

Рисунок 10 – Зависимость селективности от стоимости для поля внешней таблицы с индексами

Рисунок 11 – Зависимость количества записей от времени для поля в подзапросе без индексов

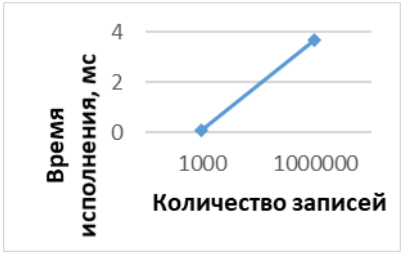

Рисунок 12 – Зависимость количества записей от времени для поля в подзапросе с индексами

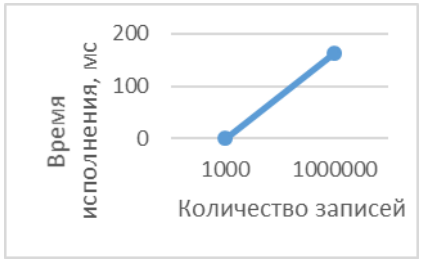

Рисунок 13 – Зависимость количества записей от времени для поля внешней таблицы без индексов

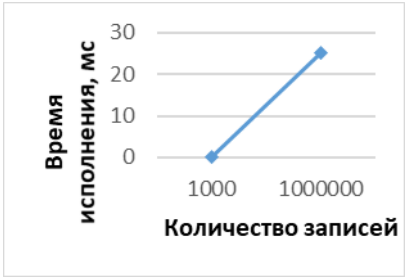

Рисунок 14 – Зависимость количества записей от времени для поля внешней таблицы с индексами

Рисунок 15 – Зависимость количества записей от стоимости для поля в подзапросе без индексов

Рисунок 16 – Зависимость количества записей от стоимости для поля в подзапросе с индексами

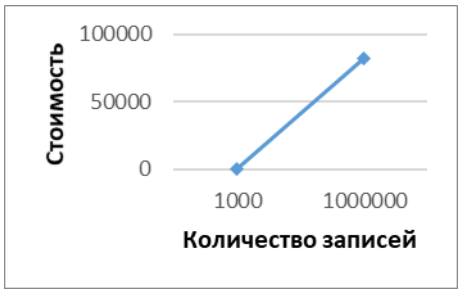

Рисунок 17 – Зависимость количества записей от стоимости для поля внешней таблицы без индексов

Рисунок 18 – Зависимость количества записей от стоимости для поля внешней таблицы с индексами

Дерево запроса с индексами в терминах расширенной реляционной алгебры:

$$\pi_{*}\left(\sigma_{bf.firstatr=\pi_{secondtable.firstatr}\left(\sigma_{\pi_{TID}\left(\sigma_{value=0}(hf)\right)=secondtable.id}(secondtable)\right)}(bf)\right)$$

Дерево запроса без индексов в терминах расширенной реляционной алгебры:

$$\pi_{*}\left(\sigma_{firsttable.firstatr=\pi_{secondtable.firstatr}\left(\sigma_{secondtable.firstatr=0}(secondtable)\right)}(firsttable)\right)$$

Индексы-B-деревья в PostgreSQL представляют собой многоуровневые иерархические структуры, в которых каждый уровень дерева может использоваться как двусвязный список страниц. Единственная метастраница индекса хранится в фиксированной позиции в начале первого файла сегмента индекса. Все остальные страницы делятся на внутренние и на листовые. Листовые страницы находятся на самом нижнем уровне дерева. Все более высокие уровни состоят из внутренних страниц. Листовая страница содержит кортежи, указывающие на строки в таблице, а внутренняя страница — кортежи, указывающие на следующий уровень в дереве. Обычно листовые страницы составляют около 99% всех страниц индекса. Новые листовые страницы добавляются в B-дерево когда существующая листовая страница не может вместить новый поступающий кортеж. При этом выполняется операция разделения страницы, освобождающая место на переполнившейся странице, перенося подмножество изначально содержащихся на ней элементов на новую страницу. При разделении страницы в её родительскую страницу также должна быть добавлена ссылка вниз, что может потребовать произвести

разделение и этой родительской страницы. Разделение страниц «каскадно поднимается вверх» рекурсивным образом. Когда же и корневая страница не может вместить новую ссылку вниз, производится операция разделения корневой страницы. При этом в структуру дерева добавляется новый уровень, на котором оказывается новая корневая страница, стоящая над той, что была корневой ранее. Схема устройства B-дерева, представлена на рис 19.

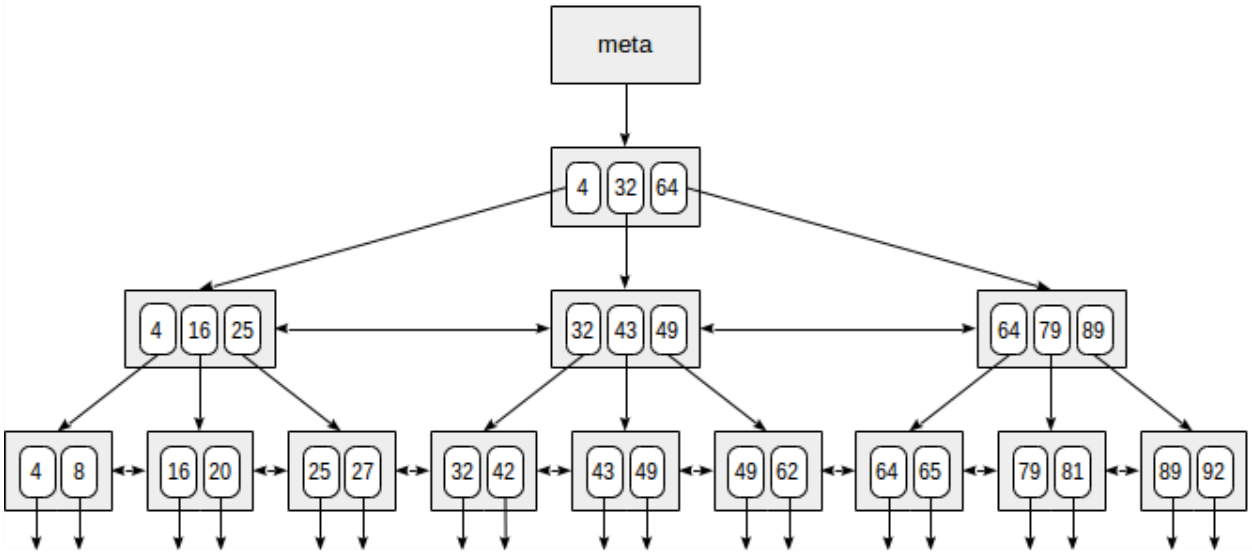

Рисунок 19 – Схема B-дерева

Идея хеширования состоит в том, чтобы значению любого типа данных сопоставить некоторое небольшое число (от 0 до N−1, всего N значений). Такое сопоставление называют хеш-функцией. Полученное число можно использовать как индекс обычного массива, куда и складывать ссылки на строки таблицы (TID). Элементы такого массива называют корзинами хеш-таблицы — в одной корзине могут лежать несколько TID-ов, если одно и то же проиндексированное значение встречается в разных строках. Хеш-функция тем лучше, чем равномернее она распределяет исходные значения по корзинам. Но даже хорошая функция будет иногда давать одинаковый результат для разных входных значений — это называется коллизией. Так что в одной корзине могут оказаться TID-ы, соответствующие разным ключам, и поэтому полученные из индекса TID-ы необходимо перепроверять. В индексе-B-дереве поиск должен проходить по дереву до тех пор, пока не будет найдена листовая страница. В таблицах с миллионами строк такой «спуск» может

увеличить время доступа к данным. Листовым страницам в хеш-индексе соответствуют страницы ячеек. В отличие от индекса-B-дерева хеш-индекс позволяет напрямую обращаться к страницам ячеек, тем самым потенциально сокращая время доступа к индексу в больших таблицах. Хеш-индексы поддерживают только оператор =, поэтому для предложений WHERE, в которых фигурируют проверки интервалов, хеш-индексы будут бесполезны. Схема устройства хэша, представлена на рис 20.

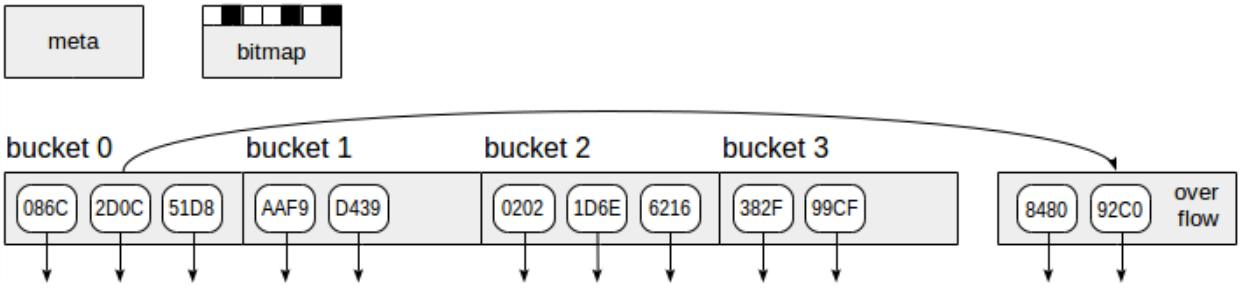

Рисунок 20 – Схема Хэша

Было сделано 3 скрипта. Создание таблицы, что представлен в листинге 1. Для проведения экспериментов – листинг 2. Заполнение таблиц – листинг 3.

Листинг 1 – Создание таблиц

```
CREATE OR REPLACE PROCEDURE reset_tables()
LANGUAGE plpgsql
AS $$
BEGIN
	DROP TABLE first_table;
CREATE TABLE first_table
(
	id SERIAL PRIMARY KEY,
	first_atr int,
	second_atr int
);
DROP TABLE second_table;
CREATE TABLE second_table
(
	id SERIAL PRIMARY KEY,
	first_atr int,
```

```
        second_atr int
);
END;
$$;
call reset_tables();
```

Листинг 2 – Проведение экспериментов

```
CALL reset_tables();
-- 1) С варьированием селективности поля в таблице подзапроса. (second_table.first_atr)
-- 1
CALL fill_tab_jank(1000, 'first_table');
CALL fill_tab(1, 1000, 'second_table');
-- 2
CALL fill_tab_jank(1000, 'first_table');
CALL fill_tab(3, 1000, 'second_table');
-- 3
CALL fill_tab_jank(1000, 'first_table');
CALL fill_tab(10, 1000, 'second_table');
-- 4
CALL fill_tab_jank(1000, 'first_table');
CALL fill_tab(20, 1000, 'second_table');
-- 5
CALL fill_tab_jank(1000000, 'first_table');
CALL fill_tab(10, 1000000, 'second_table');
-- 2) С варьированием селективности поля внешней таблицы, по которой происходит выборка WHERE. (first_table.first_atr)
-- 1
CALL fill_tab_jank(1000, 'second_table');
CALL fill_tab(1, 1000, 'first_table');
-- 2
CALL fill_tab_jank(1000, 'second_table');
CALL fill_tab(3, 1000, 'first_table');
-- 3
CALL fill_tab_jank(1000, 'second_table');
```

```
CALL fill_tab(10, 1000, 'first_table');
-- 4
CALL fill_tab_jank(1000, 'second_table');
CALL fill_tab(20, 1000, 'first_table');
-- 5
CALL fill_tab_jank(1000000, 'second_table');
CALL fill_tab(10, 1000000, 'first_table');
-- С индексами
create index bf on first_table using btree (first_atr);
create index hf on second_table using hash (first_atr);
-- Без индексов
drop index bf;
drop index hf;
-- Экспиримент
EXPLAIN ANALYZE SELECT * FROM first_table WHERE first_table.first_atr IN (
	SELECT second_table.first_atr FROM second_table WHERE second_table.first_atr =
0
);
-- План
EXPLAIN SELECT * FROM first_table WHERE first_table.first_atr IN (
	SELECT second_table.first_atr FROM second_table WHERE second_table.first_atr =
0
);
```

Листинг 3 – Заполнение таблиц

```
CREATE OR REPLACE PROCEDURE fill_tab(selecting float, count_row float,
name_create_table text)
LANGUAGE plpgsql
AS $$
DECLARE
count_rep integer := 0;
count_sel integer := 0;
BEGIN
	count_sel := selecting / 100 * count_row;
```

```
	LOOP
		IF count_rep = count_row THEN
			EXIT;
		END IF;

		IF count_sel > count_rep THEN
			EXECUTE format('INSERT INTO ' || name_create_table || ' (first_atr, second_atr) VALUES(0, 0)');
		ELSE
			EXECUTE format('INSERT INTO ' || name_create_table || ' (first_atr, second_atr) VALUES($1, $1)') USING count_rep;
		END IF;
		count_rep := count_rep + 1;
	END LOOP;
END;
$$;
CREATE OR REPLACE PROCEDURE fill_tab_jank(count_row float, name_create_table text)
LANGUAGE plpgsql
AS $$
DECLARE
count_rep integer := 0;
BEGIN
	LOOP
		IF count_rep = count_row THEN
			EXIT;
		END IF;
		EXECUTE format('INSERT INTO ' || name_create_table || ' (first_atr, second_atr) VALUES($1, $1)') USING count_rep;
		count_rep := count_rep + 1;
	END LOOP;
END;
$$;
```

Вывод

При применении индексов время исполнения запроса меньше примерно в 2 раза на маленьком количестве данных, как и стоимость.

При увеличении селективности незначительно возрастает время исполнения запроса при использовании индексов и без, при этом использовании исполняется быстрее.

При увеличении данных, чем больше количество записей, тем быстрее сильнее заметка разница в исполнении запросов с индексами, чем без них.

Список используемой литературы

1. Виноградов В.И., Виноградова М.В. Постреляционные модели данных и языки запросов: Учебное пособие. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2017. – 100с. - ISBN 978-5-7038-4283-6.
2. Рогов Е. В. PostgreSQL 16 изнутри. — М.: ДМК Пресс, 2024. — 664 с. ISBN 978-5-93700-305-8 – URL: https://edu.postgrespro.ru/postgresql_internals-16.pdf
3. Глава 11. Индексы. – Текст. Изображение: электронные // Компания Postgres Professional: [сайт]. – URL: https://postgrespro.ru/docs/postgresql/16/indexes
4. Chapter 11. Indexes. – Текст. Изображение: электронные // PostgreSQL: [сайт]. – URL: https://www.postgresql.org/docs/16/indexes.html
5. Индексы в PostgreSQL — 1. – Текст. Изображение: электронные // Habr: [сайт]. – URL: https://habr.com/ru/companies/postgrespro/articles/326096/
6. Индексы в PostgreSQL — 2. – Текст. Изображение: электронные // Habr: [сайт]. – URL: https://habr.com/ru/companies/postgrespro/articles/326106/
7. Язык SQL. Лекция 6. Индексы. – Текст. Изображение: электронные // Компания Postgres Professional: [сайт]. – URL: https://edu.postgrespro.ru/sqlprimer/sqlprimer-2019-msu-06.pdf
8. Статистика. – Текст. Изображение: электронные // Компания Postgres Professional: [сайт]. – URL:

https://edu.postgrespro.ru/dba29.5/dba2_17_statistics.pdf

9. Оптимизация запросов. Выполнение запросов. – Текст. Изображение: электронные // Компания Postgres Professional: [сайт]. – URL: https://edu.postgrespro.ru/qpt/qpt_02_query.pdf
10. Оптимизация запросов. Статистика. – Текст. Изображение: электронные // Компания Postgres Professional: [сайт]. – URL: https://edu.postgrespro.ru/qpt/qpt_09_statistics.pdf
11. Атака не клонов, или Генерация и анализ тестовых данных для нагрузки. Часть 2. – Текст. Изображение: электронные // Habr: [сайт]. – URL: https://habr.com/ru/companies/oleg-bunin/articles/589543/
12. Генерация данных — творчество или рутина? – Текст. Изображение: электронные // Habr: [сайт]. – URL: https://habr.com/ru/articles/723202/
13. Легко сгенерируйте Mock данные с помощью PostgreSQL. – Текст. Изображение: электронные // DevGang: [сайт]. – URL: https://devgang.ru/article/legko-sgeneriruite-mock-dannye-s-pomosczu-postgresqlgy714we9hp/
14. Глава 14. Оптимизация производительности. – Текст. Изображение: электронные // Компания Postgres Professional: [сайт]. – URL: https://postgrespro.ru/docs/postgresql/16/performance-tips
15. Chapter 14. Performance Tips. – Текст. Изображение: электронные // PostgreSQL: [сайт]. – URL: https://www.postgresql.org/docs/current/performance-tips.html
16. Оптимизация запросов. Индексный доступ. – Текст. Изображение: электронные // Компания Postgres Professional: [сайт]. – URL: https://edu.postgrespro.ru/qpt-13/qpt_04_indexscan.html
17. Индексы в PostgreSQL — 4. – Текст. Изображение: электронные // Habr: [сайт]. – URL: https://habr.com/ru/companies/postgrespro/articles/330544/
18. Глава 67. Индексы B-деревья. – Текст. Изображение: электронные // Компания Postgres Professional: [сайт]. – URL:

https://postgrespro.ru/docs/postgresql/16/btree

19. Chapter 67. B-Tree Indexes. – Текст. Изображение: электронные // PostgreSQL: [сайт]. – URL: https://www.postgresql.org/docs/16/btree.html
20. Индексы в PostgreSQL — 3. – Текст. Изображение: электронные // Habr: [сайт]. – URL: https://habr.com/ru/companies/postgrespro/articles/328280/
21. Глава 72. Хеш-индексы. – Текст. Изображение: электронные // Компания Postgres Professional: [сайт]. – URL: https://postgrespro.ru/docs/postgresql/16/hash-index
22. Chapter 72. Hash Indexes. – Текст. Изображение: электронные // PostgreSQL: [сайт]. – URL: https://www.postgresql.org/docs/16/hashindex.html

Приложение 1

		Селективность 1%; Записей 1000				Селективность 3%; Записей 1000				Селективность 10%; Записей 1000				Селективность 20%; Записей 1000				Селективность 10%; Записей 1000000			
		1 эксп	2 эксп	3 эксп	Среднее	1 эксп	2 эксп	3 эксп	Среднее	1 эксп	2 эксп	3 эксп	Среднее	1 эксп	2 эксп	3 эксп	Среднее	1 эксп	2 эксп	3 эксп	Среднее
С варьированием селективности поля в таблице подзапроса																					
С индексами	Время	0,087	0,075	0,094	0,085333	0,086	0,072	0,096	0,084667	0,072	0,102	0,085	0,086333	0,141	0,126	0,098	0,121667	3,97	3,398	3,693	3,687
	Стоимость	41,08	41,08	41,08	41,08	42,04	42,04	42,04	42,04	45,41	45,41	45,41	45,41	62,2	62,2	62,2	62,2	20760,91	20760,91	20760,91	20760,91
Без индексов	Время	0,226	0,148	0,112	0,162	0,124	0,143	0,155	0,140667	0,087	0,105	0,155	0,115667	0,179	0,177	0,147	0,167667	101,001	112,178	200,57	137,9163
	Стоимость	74,02	74,02	74,02	74,02	74,02	74,02	74,02	74,02	74,02	74,02	74,02	74,02	74,02	74,02	74,02	74,02	69655,21	69655,21	69655,21	69655,21
С варьированием селективности поля внешней таблицы, по которой происходит выборка WHERE																					
С индексами	Время	0,052	0,08	0,064	0,065333	0,084	0,054	0,077	0,071667	0,079	0,095	0,08	0,084667	0,192	0,092	0,165	0,149667	21,818	31,779	21,525	25,04067
	Стоимость	41,08	41,08	41,08	41,08	42,04	42,04	42,04	42,04	45,13	45,13	45,13	45,13	51,63	51,63	51,63	51,63	7595,91	7595,91	7595,91	7595,91
Без индексов	Время	0,145	0,128	0,137	0,136667	0,1	0,109	0,103	0,104	0,108	0,228	0,123	0,153	0,304	0,329	0,358	0,330333	140,261	193,291	152,564	162,0387
	Стоимость	92,63	92,63	92,63	92,63	92,88	92,88	92,88	92,88	93,75	93,75	93,75	93,75	95,52	95,52	95,52	95,52	82508,8	82508,8	82508,8	82508,8