|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Министерство науки и высшего образования РФ | | | | | |
| Федеральное государственное автономное | | | | | |
| образовательное учреждение высшего образования | | | | | |
| **«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»** | | | | | |
|  | | | | | |
| Институт космических и информационных технологий | | | | | |
| институт | | | | | |
| Программная инженерия | | | | | |
| кафедра | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ** | | | | | |
| Повышение производительности | | | | | |
| тема | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| Преподаватель | |  |  |  | А. Д. Вожжов |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ23-17/1б, 032320521 |  |  |  | А. С. Лысаковский |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |
|  | | | | | |
|  | | | | | |
| Красноярск 2025 | | | | | |

# ВВЕДЕНИЕ

## Цель работы

Изучить теоретический материал по теме «Повышение производительности». Выполнить задания.

## Задачи

В рамках данной практической работы необходимо выполнить следующие задачи:

1. изучить теоретический материал по предложенной теме;
2. выполнить задание;
3. предоставить отчёт преподавателю.

## Задание

Задание данной практической работы состоит из следующих частей:

1. Выполнить задания из главы 10 из книги на е-курсах.

# ХОД РАБОТЫ

## Задание 1

Задание. Как вы думаете, почему при сканировании по индексу оценка стоимости ресурсов, требующихся для выдачи первых результатов, не равна нулю, хотя используется индекс, совпадающий с порядком сортировки?

На рисунке 1 показан результат выполнения задания.

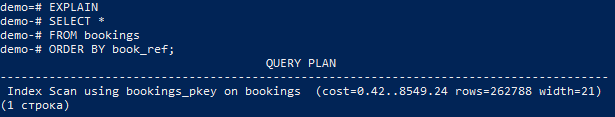


Рисунок 1 – Запрос

Для выдачи первых результатов необходимо сначала найти индекс в специальной таблице. Поэтому начальные затраты и не равны нулю.

## Задание 2

Задание. Как вы думаете, если в запросе присутствует предложение ORDER BY, и создан индекс по тем столбцам, которые фигурируют в предложении ORDER BY, то всегда ли будет использоваться этот индекс или нет? Почему? Проверьте ваши предположения с помощью команды EXPLAIN.

Ответ. Не всегда.

На рисунке 2 показан результат выполнения задания.

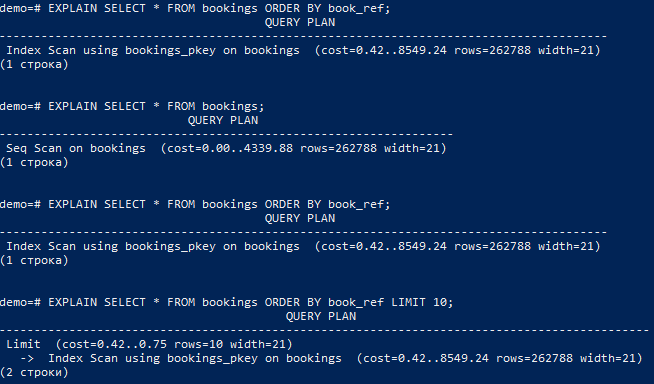


Рисунок 2 – Эксперименты

Как видно из примеров, не всегда используется поиск по индексу. Потому-что планировщик подготавливает несколько планов выполнения запросов и выбирает лучший. Не всегда выгодно обращаться к индексной таблице.

## Задание 3

Задание. Самостоятельно выполните команду EXPLAIN для запроса, содержащего общее табличное выражение (CTE). Посмотрите, на каком уровне находится узел плана, отвечающий за это выражение, как он оформляется. Учтите, что общие табличные выражения всегда материализуются, т.е. вычисляются однократно и результат их вычисления сохраняется в памяти, а затем все последующие обращения в рамках запроса направляются уже к этому материализованному результату.

На рисунке 3 показан прогресс работы.

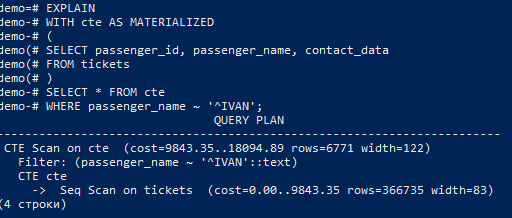


Рисунок 3 – CTE запрос

CTE находится в самом низу.

## Задание 4

Прокомментируйте следующий план, попробуйте объяснить значения всех его узлов и параметров

На рисунке 4 показан результат выполнения задания.

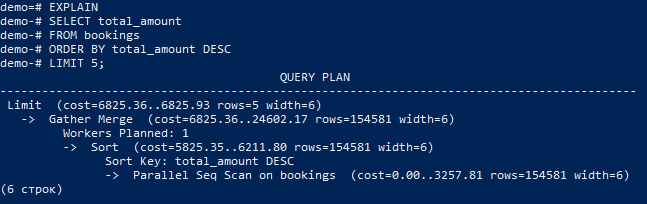


Рисунок 4 – Анализ

Производится запрос на возврат из таблицы bookings столбца total\_amount. Затем данные сортируются по убыванию столбца total\_amount. Стоит отметить, что данные задачи выполняются параллельно. Следующим шагом происходит объединение параллельной работы запросов выше. Наконец, ограничение вывода первыми пятью результатами с учётом сортировки.

## Задание 5

Задание. В подавляющем большинстве городов только один аэропорт, но есть и такие города, в которых более одного аэропорта. Давайте их выявим.

Как вы думаете, чем можно объяснить, что вторая оценка стоимости в параметре cost для узла Seq Scan, равная 3,04, не совпадает с первой оценкой стоимости в параметре cost для узла HashAggregate?

На рисунке 5 показан прогресс работы.

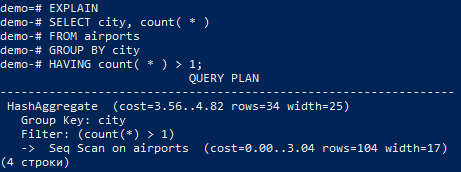


Рисунок 5 – Анализ запроса

Расходы не сходятся, потому что создаются дополнительные структуры данных, например хэш-таблица, и фильтрация. Эти действия создают дополнительные расходы.

## Задание 6

Выполните команду EXPLAIN для запроса, в котором использована какая-

нибудь из оконных функций. Найдите в плане выполнения запроса узел с именем WindowAgg. Попробуйте объяснить, почему он занимает именно этот уровень в плане.

На рисунках 6 показан результат выполнения задания.

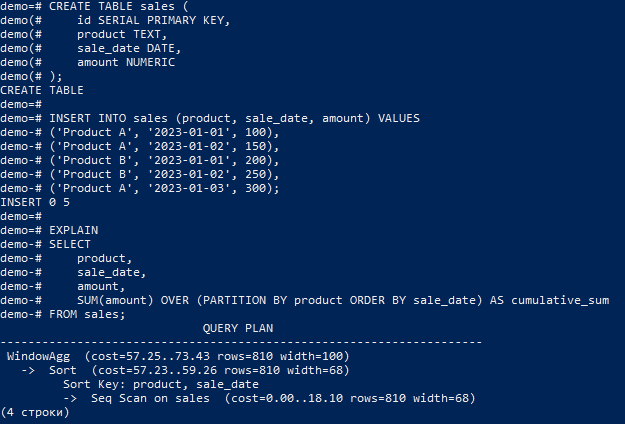


Рисунок 6 – Анализ

Такая позиция WindowAgg обусловлена тем, что оконные функции выполняются после сортировок и фильтров, но до ограничений выводов (LIMIT).

## Задание 7

Проанализируйте план выполнения операций вставки и удаления строк. Причем сделайте это таким образом, чтобы данные в таблицах фактически изменены не были.

На рисунке 7 показан прогресс работы.

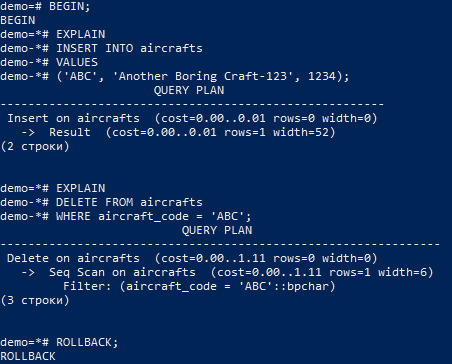


Рисунок 7 – Анализ операций

## Задание 8

Задание. Замена коррелированного подзапроса соединением таблиц является одним из способов повышения производительности. Предположим, что мы задались вопросом: сколько маршрутов обслуживают самолеты каждого типа? При этом нужно учитывать, что может иметь место такая ситуация, когда самолеты какого-либо типа не обслуживают ни одного маршрута. Поэтому необходимо использовать не только представление «Маршруты» (routes), но и таблицу «Самолеты» (aircrafts).

Это первый вариант запроса, в нем используется коррелированный подзапрос.

EXPLAIN ANALYZE

SELECT a.aircraft\_code AS a\_code,

a.model,

( SELECT count( r.aircraft\_code )

FROM routes r

WHERE r.aircraft\_code = a.aircraft\_code

) AS num\_routes

FROM aircrafts a

GROUP BY 1, 2

ORDER BY 3 DESC;

А в этом варианте коррелированный подзапрос раскрыт и заменен внешним соединением:

EXPLAIN ANALYZE

SELECT a.aircraft\_code AS a\_code,

a.model,

count( r.aircraft\_code ) AS num\_routes

FROM aircrafts a

LEFT OUTER JOIN routes r

ON r.aircraft\_code = a.aircraft\_code

GROUP BY 1, 2

ORDER BY 3 DESC;

Причина использования внешнего соединения в том, что может найтись модель самолета, не обслуживающая ни одного маршрута, и если не использовать внешнее соединение, она вообще не попадет в результирующую выборку.

Исследуйте планы выполнения обоих запросов. Попытайтесь найти объяснение различиям в эффективности их выполнения. Чтобы получить усредненную картину, выполните каждый запрос несколько раз. Поскольку таблицы, участвующие в запросах, небольшие, то различие по абсолютным затратам времени выполнения будет незначительным. Но если бы число строк в таблицах было большим, то экономия ресурсов сервера могла оказаться заметной.

Предложите аналогичную пару запросов к базе данных «Авиаперевозки». Проведите необходимые эксперименты с вашими запросами.

На рисунках с 8 по 15 показан прогресс работы.

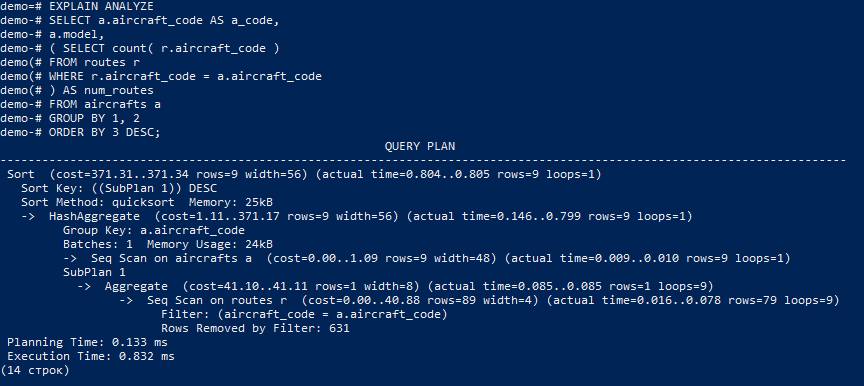


Рисунок 8 – Коррелированный подзапрос 1

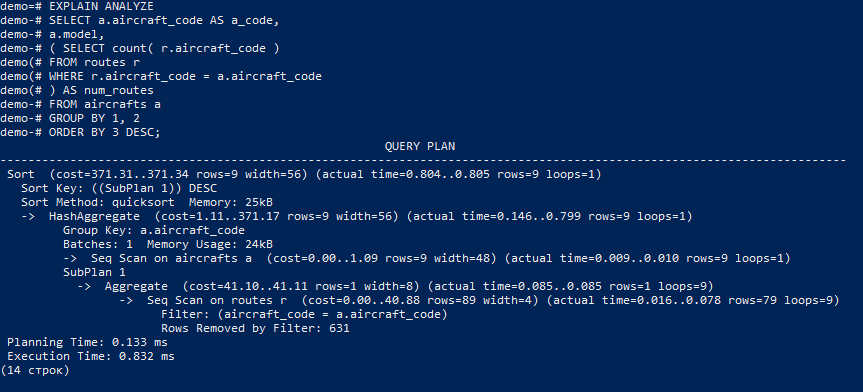


Рисунок 9 – Коррелированный подзапрос 2

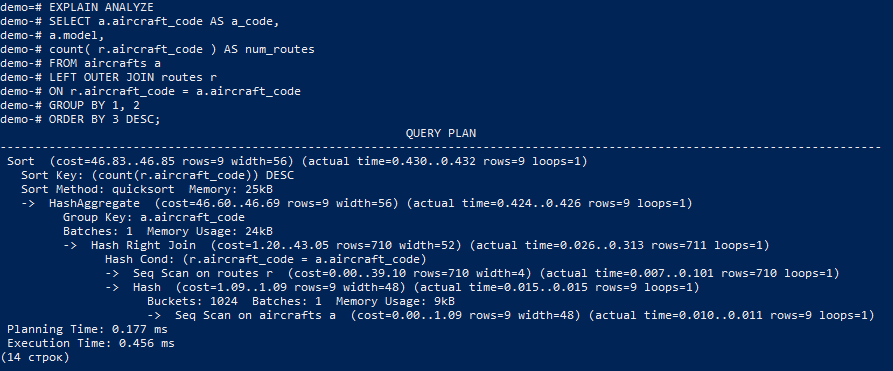


Рисунок 10 – Внешнее соединение 1

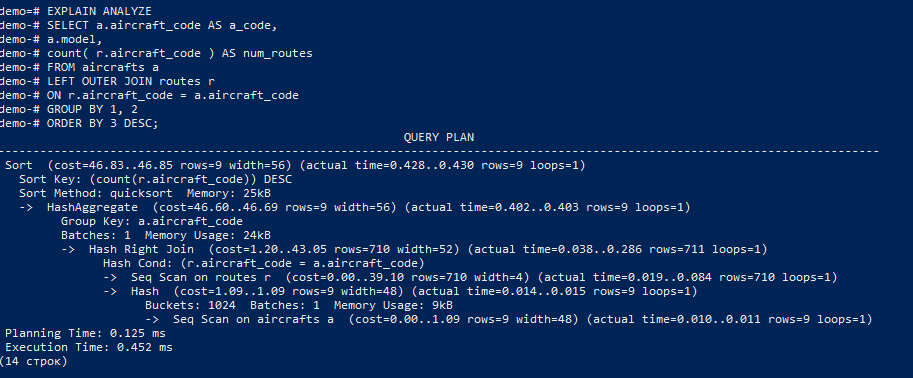


Рисунок 11 – Внешнее соединение 2

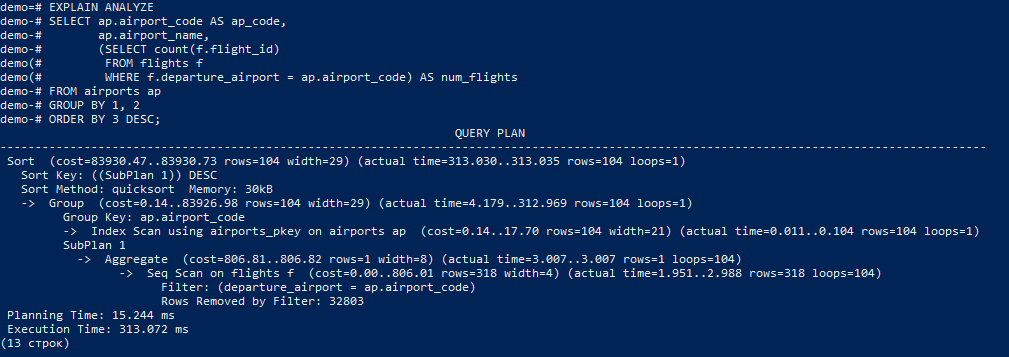


Рисунок 12 – Свой коррелированный подзапрос 1

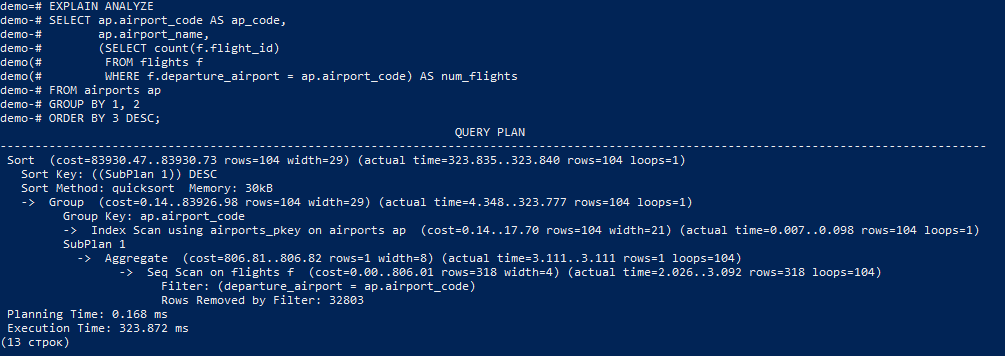


Рисунок 13 – Свой коррелированный подзапрос 2

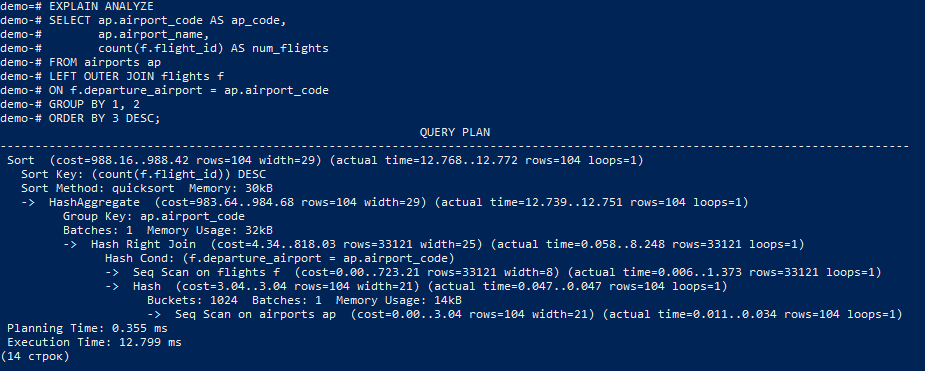


Рисунок 14 – Свой запрос с внешним соединением 1

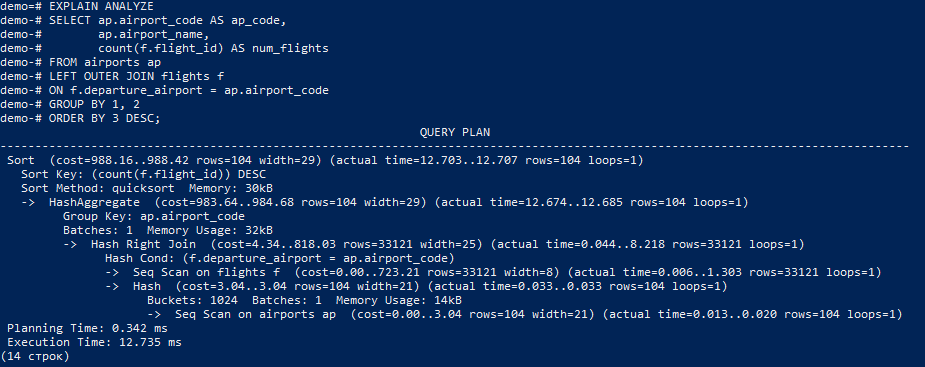


Рисунок 15 – Свой запрос с внешним соединением 2

Низкая скорость коррелированных подзапросов обусловлена в необходимости анализировать таблицу много раз. Внешние соединения позволяют выполнить анализ таблиц условно 1 раз и далее уже выполнять группировку. По итогу внешние соединения использовать выгоднее на больших выборках, нежели коррелированные подзапросы.

## Задание 9

Одним из способов повышения производительности является изменение схемы данных, связанное с денормализацией, а именно: создание материализованных представлений. В главе 5 было описано такое материализованное представление - "Маршруты" (routes). Команда для его создания была приведена в главе 6.

Проведите эксперимент: сначала выполните выборку из готового представления, а затем ту выборку, которая это представление формирует.

EXPLAIN ANALYZE

SELECT \* FROM routes;

EXPLAIN ANALYZE

WITH f3 AS (SELECT f2.flight\_no, ...

Поскольку второй запрос очень громоздкий, то можно поступить таким образом: сначала сохраните его в текстовом файле, а затем выполните с помощью команды \i утилиты psql.

Вы увидите, что затраты времени отличаются практически на два порядка. Конечно, нужно помнить, что материализованные представления необходимо периодически обновлять, чтобы их содержимое было актуальным.

На рисунках 16-17 показан прогресс работы.

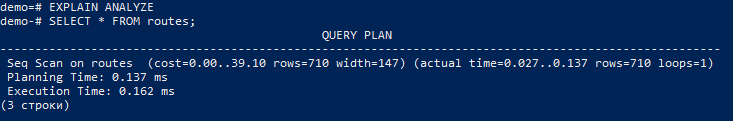


Рисунок 16 – Запрос к представлению

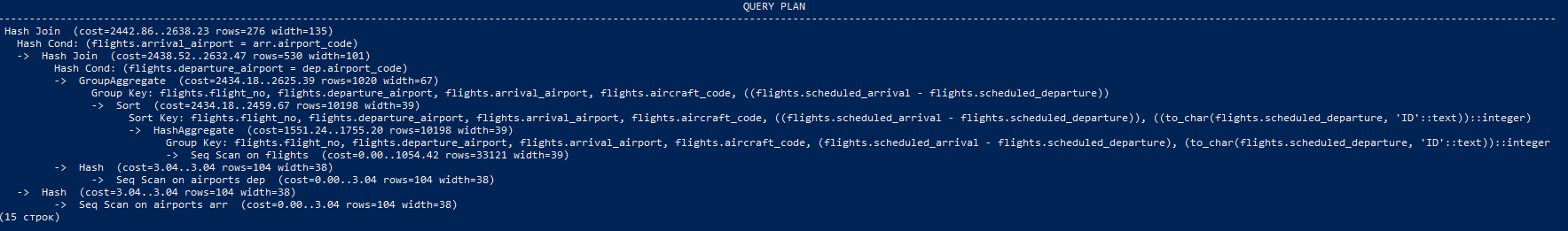


Рисунок 17 – Запрос к исходному запросу представления

## Задание 10

Одним из способов повышения производительности является изменение схемы данных, связанное с денормализацией, а именно: использование вычисляемых столбцов. Для примера рассмотрим таблицу "Бронирования" (bookings). В ней столбец "Полная сумма бронирования" (total\_amount) является вычисляемым. Мы не будем сейчас говорить о том, каким образом его значения синхронизируются с данными в таблице "Перелеты" (ticket\_flights), а лишь рассмотрим два запроса, возвращающие полные суммы бронирований.

Предположим, что указанного столбца в таблице bookings не было бы. Тогда запрос, возвращающий полные суммы бронирований, выглядел бы так:

EXPLAIN ANALYZE

SELECT b.book\_ref, sum(tf.amount)

FROM bookings b, tickets t, ticket\_flights tf

WHERE b.book\_ref = t.book\_ref

AND t.ticket\_no = tf.ticket\_no

GROUP BY 1

ORDER BY 1;

Но благодаря наличию вычисляемого столбца total\_amount те же сведения можно получить с гораздо меньшими затратами ресурсов:

EXPLAIN ANALYZE

SELECT book\_ref, total\_amount

FROM bookings

ORDER BY 1;

В качестве другого примера можно предложить добавить вычисляемый столбец "Количество билетов" (tickets\_count) в таблицу bookings, который будет хранить количество билетов в каждом бронировании. Это позволит избежать сложных соединений таблиц при подсчете количества билетов:

-- Без вычисляемого столбца:

EXPLAIN ANALYZE

SELECT b.book\_ref, count(t.ticket\_no)

FROM bookings b JOIN tickets t ON b.book\_ref = t.book\_ref

GROUP BY 1;

-- С вычисляемым столбцом:

EXPLAIN ANALYZE

SELECT book\_ref, tickets\_count

FROM bookings;

Эксперименты покажут значительное сокращение времени выполнения запроса и уменьшение нагрузки на сервер, особенно при частых обращениях к этой информации. Однако следует помнить о необходимости поддержания актуальности данных в вычисляемом столбце при изменении связанных таблиц.

На рисунках 18-19 показан результат работы.

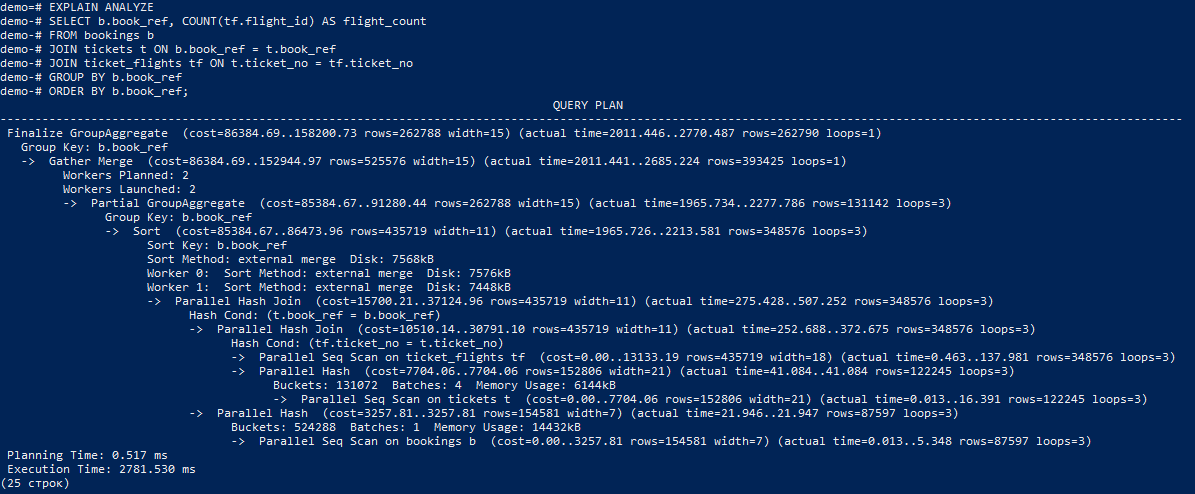


Рисунок 18 – Запрос без вычисляемого столбца

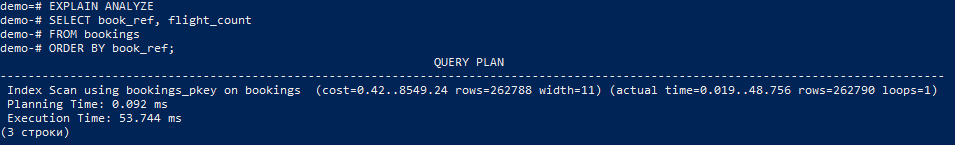


Рисунок 19 – Запрос с вычисляемым столбцом

## Задание 11

Одним из способов повышения производительности является использование временных таблиц. Если нужно сделать много выборок из представления "Рейсы" (flights\_v), стоит создать временную таблицу:

CREATE TEMP TABLE flights\_tt AS SELECT \* FROM flights\_v;

Сравним планы выполнения запросов:

EXPLAIN ANALYZE SELECT \* FROM flights\_v;

EXPLAIN ANALYZE SELECT \* FROM flights\_tt;

План для представления многоуровневый, потому что представление каждый раз пересчитывает данные из исходных таблиц. Для временной таблицы план проще, так как данные уже сохранены.

Для более сложных запросов:

EXPLAIN ANALYZE SELECT departure\_city, count(\*) FROM flights\_v WHERE status = 'Scheduled' GROUP BY 1;

EXPLAIN ANALYZE SELECT departure\_city, count(\*) FROM flights\_tt WHERE status = 'Scheduled' GROUP BY 1;

На рисунках показан прогресс работы.

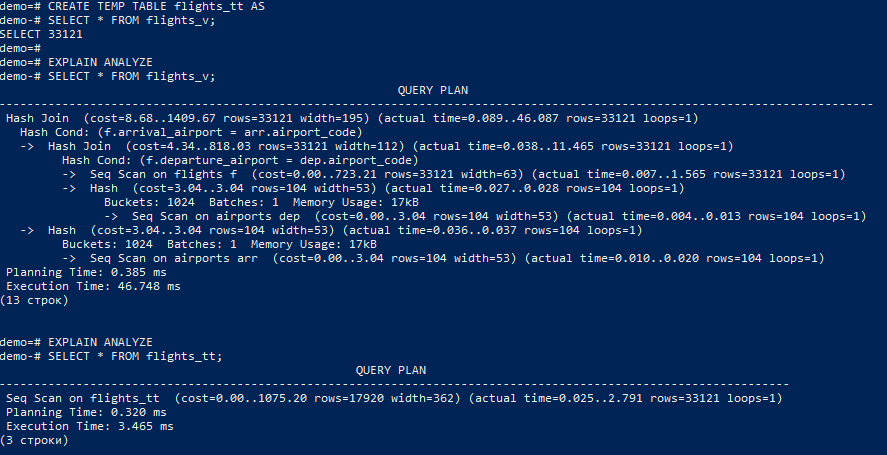


Рисунок 20 – Работа

Большой план к представлению обусловлен тем, что представление – это просто ссылка на сохранённый запрос. Потому, когда мы инспектируем представление, мы инспектируем запрос этого представления.

Сравнение сложных планов представлено на рисунке 21.

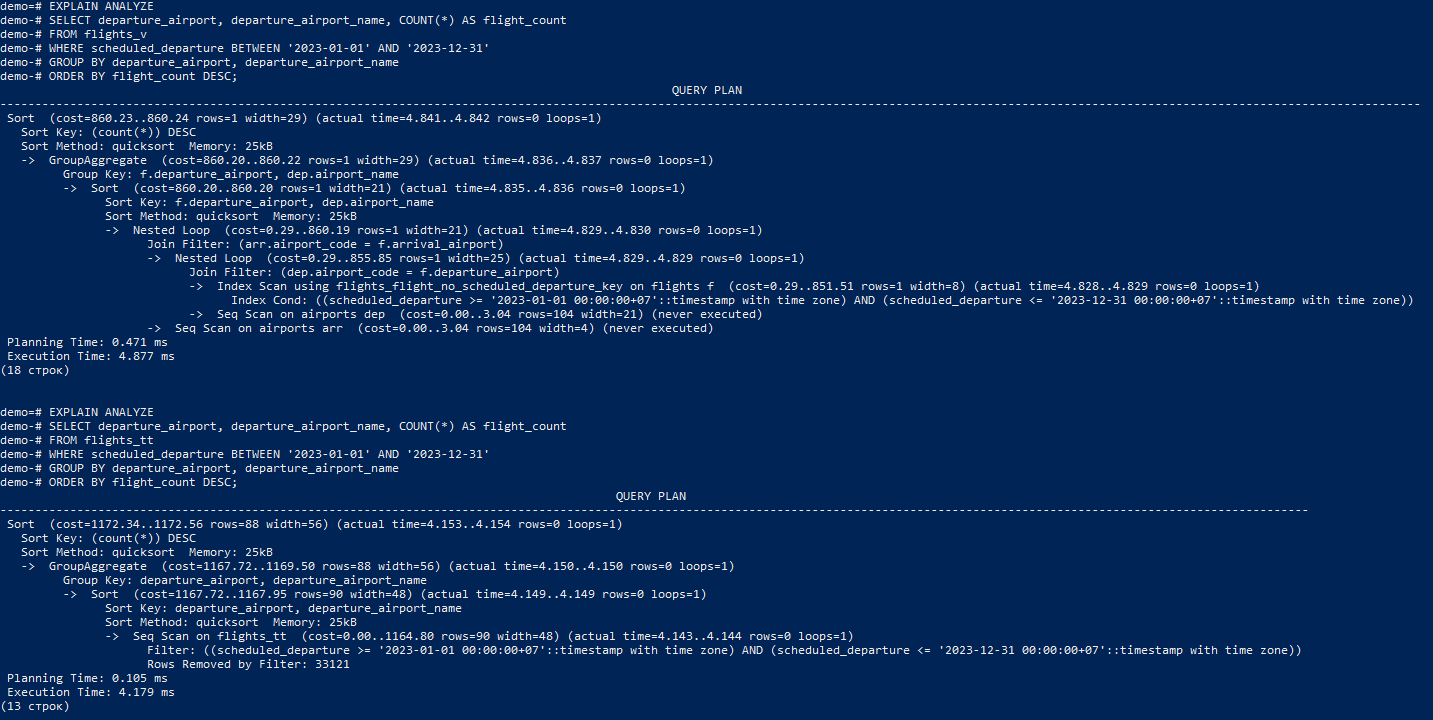


Рисунок 21 – Сравнение запросов

Временная таблица выигрывает по времени.

Решение для базы данных «Авиаперевозки» представлено на рисунках 22-24.

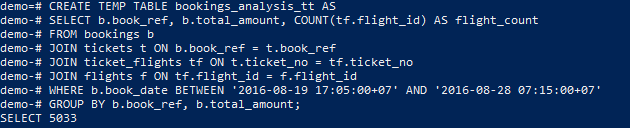


Рисунок 22 – Создание временной таблицы

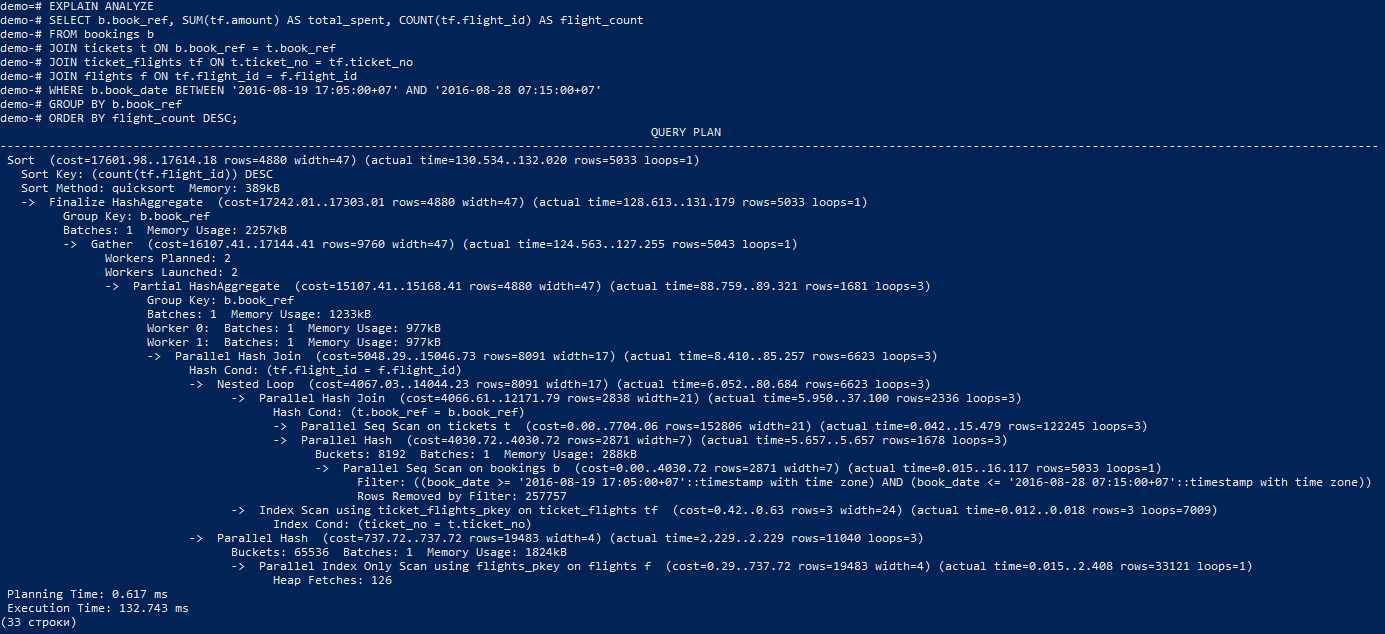


Рисунок 23 – Запрос без временной таблицы

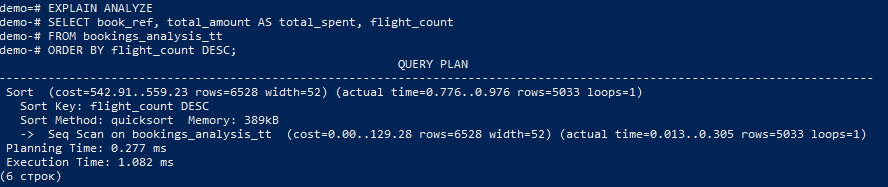


Рисунок 24 – Запрос с временной таблицей

## Задание 12

Одним из способов повышения производительности является изменение схемы данных, связанное с денормализацией, а именно: создание индексов. Выполните следующий простой запрос к таблице «Билеты»:

EXPLAIN ANALYZE SELECT count( \* ) FROM tickets WHERE passenger\_name = 'IVAN IVANOV';

Создайте индекс по столбцу passenger\_name:

CREATE INDEX passenger\_name\_key ON tickets ( passenger\_name );

Теперь повторите запрос и сравните полученные планы и фактические результаты. Предложите какой-нибудь запрос к базе данных «Авиаперевозки», для выполнения которого было бы целесообразно создать индекс. Создайте индекс и повторите запрос. Изучите полученный план, посмотрите, используется ли индекс планировщиком.

На рисунках 25-26 представлен прогресс работы.

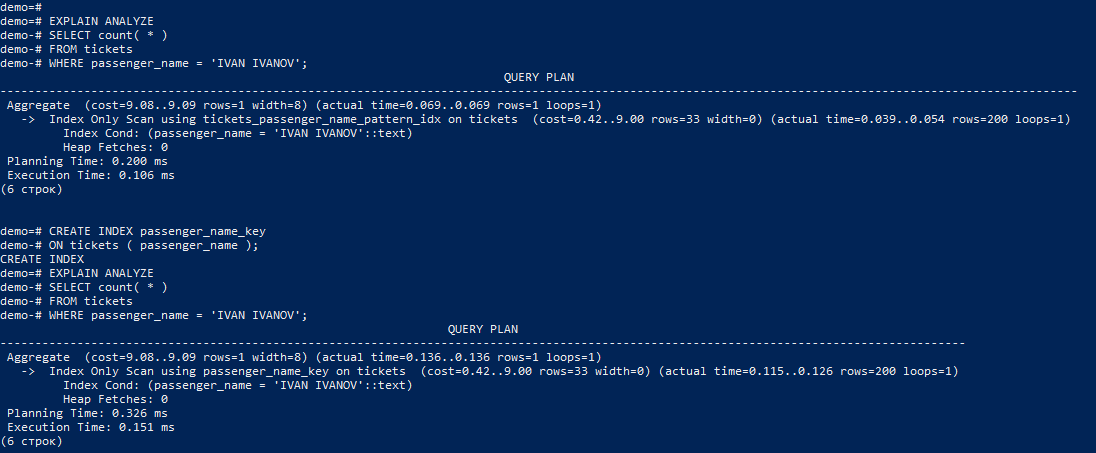


Рисунок 25 – Прогресс работы

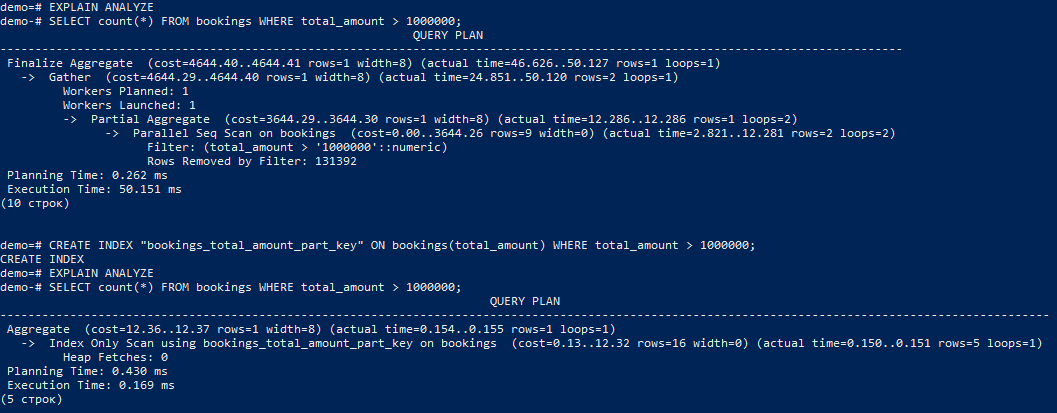


Рисунок 26 – Своё решение для индексов

## Задание 13

В самом конце главы мы выполняли оптимизацию запроса путем создания индекса и модификации текста запроса. Был сформирован такой запрос:

EXPLAIN ANALYZE

SELECT num\_tickets, count() AS num\_bookings

FROM

(SELECT b.book\_ref, count()

FROM bookings b, tickets t

WHERE date\_trunc('mon', b.book\_date) = '2016-09-01'

AND t.book\_ref = b.book\_ref

GROUP BY b.book\_ref

) AS count\_tickets(book\_ref, num\_tickets)

GROUP by num\_tickets

ORDER BY num\_tickets DESC;

Мы экспериментировали с параметрами планировщика enable\_hashjoin и enable\_nestloop при наличии индекса по таблице tickets:

SET enable\_hashjoin = off;

SET enable\_nestloop = off;

Однако полученные планы детально рассмотрены не были.

Задание. Проанализируйте эти планы. Посмотрите, в каких случаях используются и в каких не используются индексы по таблицам bookings и tickets. Вспомните о таком понятии, как селективность, т.е. доля строк, выбираемых из таблицы.

На рисунках 27-29 показан прогресс работы.

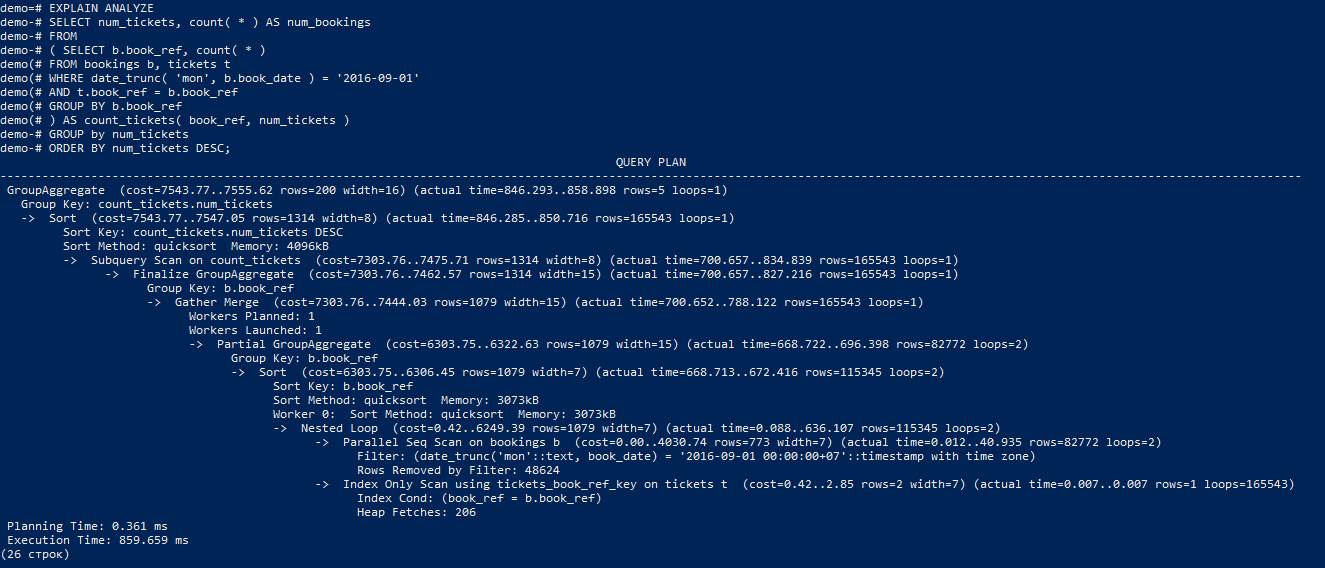


Рисунок 27 – Запрос 1

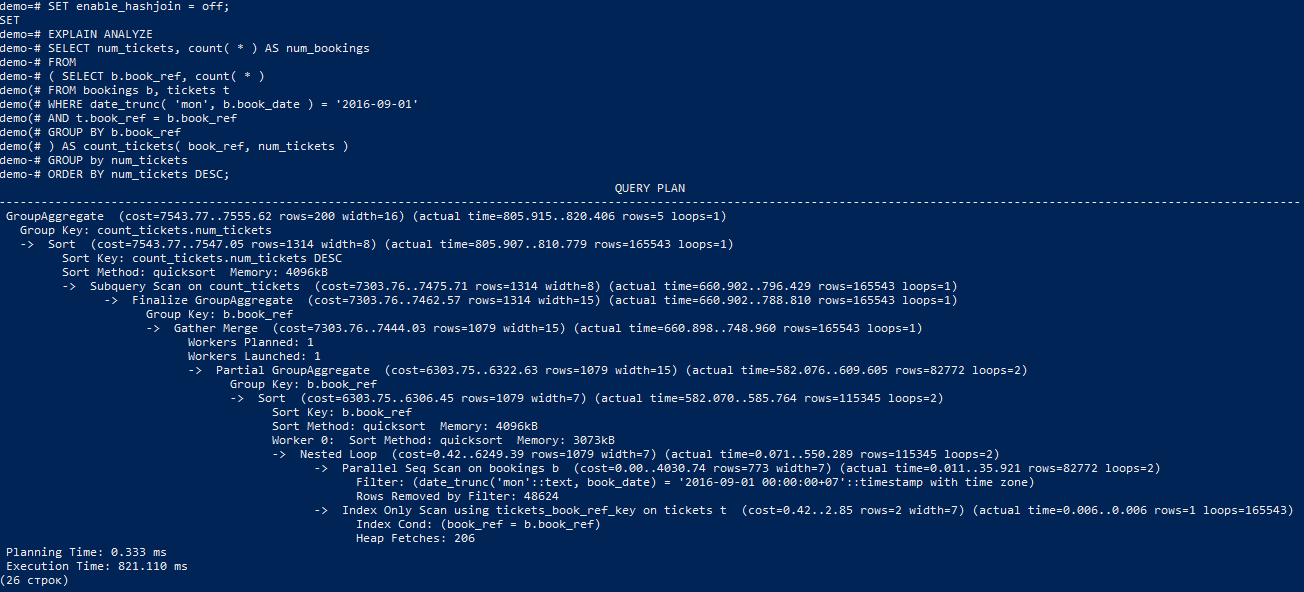


Рисунок 28 – Запрос 2

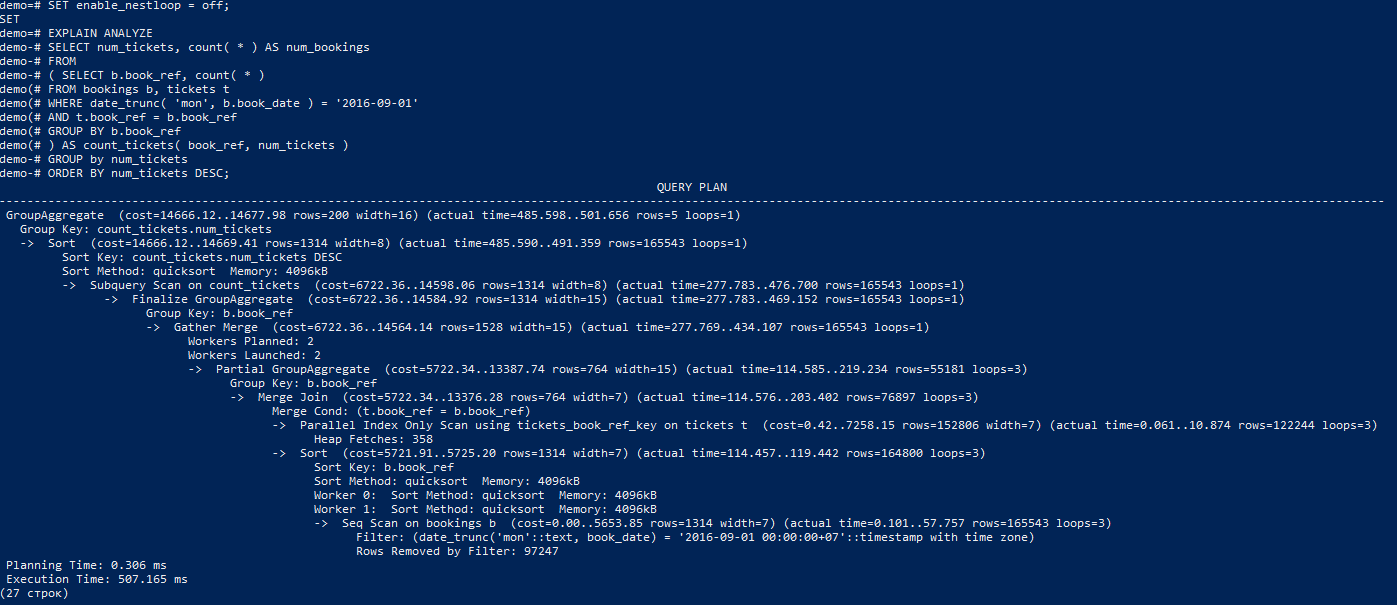


Рисунок 29 – Запрос 3

Индексы в таблицах используются, когда селективность высокая. То есть объём выборки по отношению к таблице мал. Индексы не используются, когда селективность низкая. То есть объём выборки по отношению к таблице велик.

Проанализировав данные запросы, можно сказать, что планировщик может допускать ошибки и выбирать не эффективные планы.

## Задание 14

В столбцах таблиц могут содержаться значения NULL. При сортировке строк по значениям таких столбцов СУБД по умолчанию ведет себя так, как будто значение NULL превосходит по величине любые другие значения. В результате получается, что если задан возрастающий порядок сортировки, то значения NULL будут идти последними, если же порядок сортировки убывающий, тогда они будут первыми. Принимая решение о создании индексов, нужно учитывать требуемый порядок сортировки и желаемое расположение строк со значениями NULL в выборке.

Давайте создадим таблицу, содержащую такое число строк, что использование индекса планировщиком становится очень вероятным:

CREATE TABLE nulls AS

SELECT num::integer, 'TEXT' || num::text AS txt

FROM generate\_series(1, 200000) AS gen\_ser(num);

Проиндексируем таблицу по числовому столбцу:

CREATE INDEX nulls\_ind

ON nulls(num);

Добавим в таблицу одну строку, содержащую значение NULL в индексируемом столбце:

INSERT INTO nulls

VALUES (NULL, 'TEXT');

Проверим использование индекса:

EXPLAIN

SELECT \*

FROM nulls

ORDER BY num;

Убедимся, что строка со значением NULL окажется в выводе самой последней:

SELECT \*

FROM nulls

ORDER BY num

OFFSET 199995;

Модифицируем запрос, указав, что значения NULL должны располагаться в начале выборки:

EXPLAIN

SELECT \*

FROM nulls

ORDER BY num NULLS FIRST;

Задание 1. Проверьте, будет ли использоваться индекс:

EXPLAIN

SELECT \*

FROM nulls

ORDER BY num DESC NULLS FIRST;

Задание 2. Модифицируйте команду создания индекса так, чтобы он использовался при выполнении выборки:

SELECT \*

FROM nulls

ORDER BY num NULLS FIRST;

Задание 3. Выполните аналогичные эксперименты, задавая убывающий порядок сортировки с помощью DESC и изменяя расположение значений NULL с помощью NULLS FIRST и NULLS LAST. Проверьте фактическое время выполнения команд с помощью EXPLAIN ANALYZE.

На рисунках 30-32 показан прогресс работы.

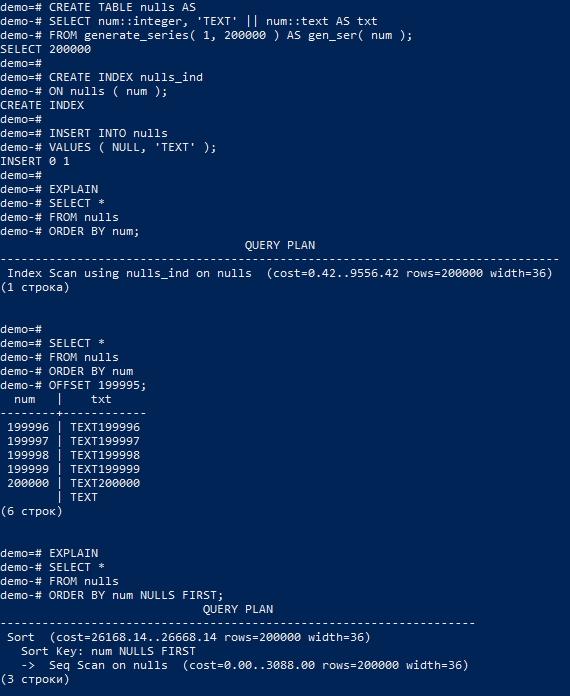


Рисунок 30 – Прогресс работы

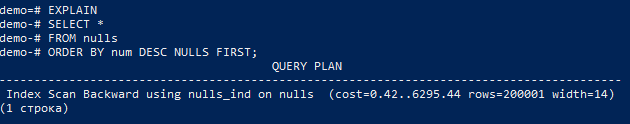


Рисунок 31 – Проверка гипотезы

Слово Backward в плане означает, что поиск по таблице индексов будет производиться снизу вверх. По умолчанию «PostgreSQL» создаёт таблицы индексов в возрастающем порядке.

На рисунке 32 показано под задание 2.

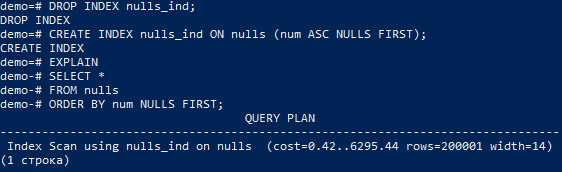


Рисунок 32 – Решение для использования индекса

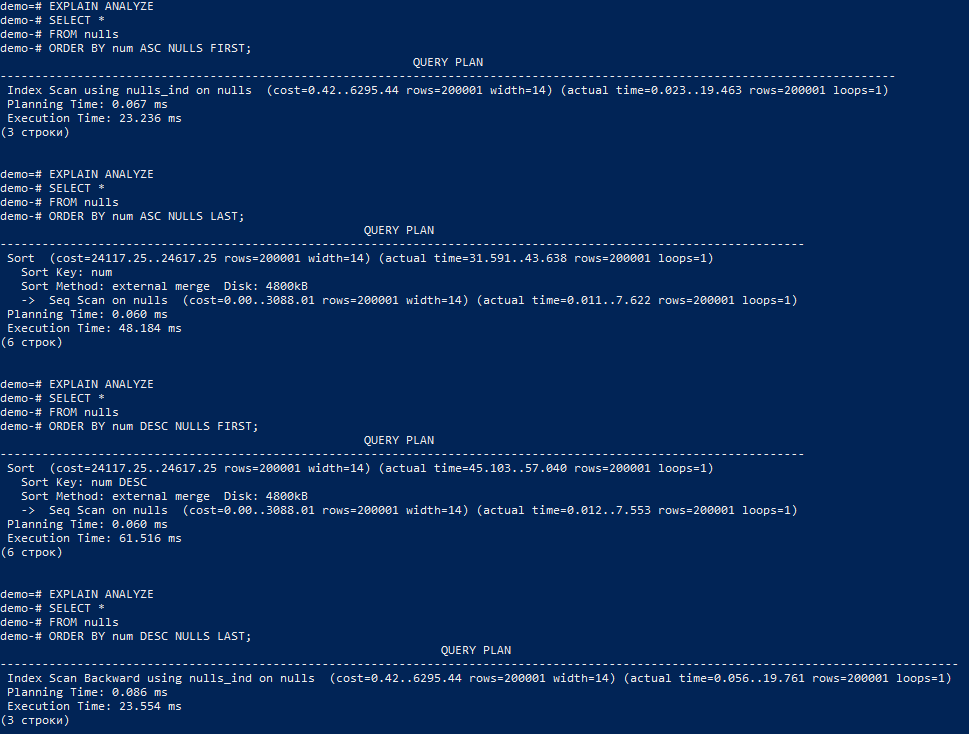


Рисунок 33 – Эксперименты с запросами

## Задание 15

Обратитесь к запросам в главе 6. Выполните команду EXPLAIN для всех этих запросов и ознакомьтесь с планами, которые создаст планировщик. В планах могут встречаться наименования методов, которые не были рассмотрены в тексте главы, однако они должны быть вам интуитивно понятны.

На рисунках 34-42 показан прогресс работы.

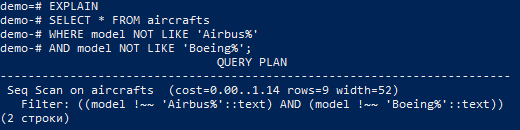


Рисунок 34 – EXPLAIN Запросов, часть 1

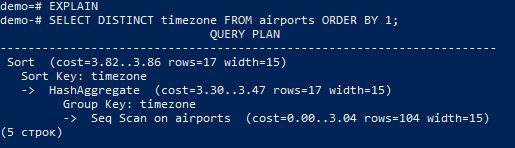


Рисунок 35 – EXPLAIN Запросов, часть 2

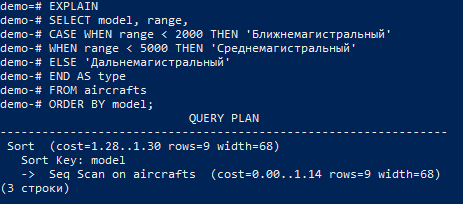


Рисунок 36 – EXPLAIN Запросов, часть 3

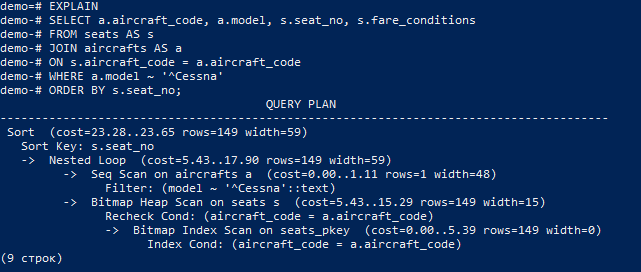


Рисунок 37 – EXPLAIN Запросов, часть 4

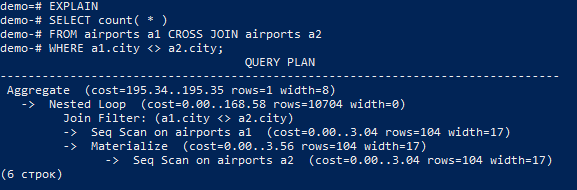


Рисунок 38 – EXPLAIN Запросов, часть 5

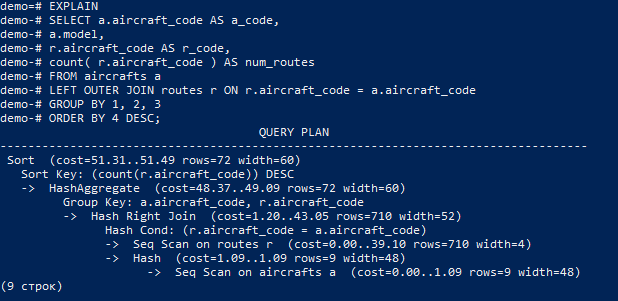


Рисунок 39 – EXPLAIN Запросов, часть 6

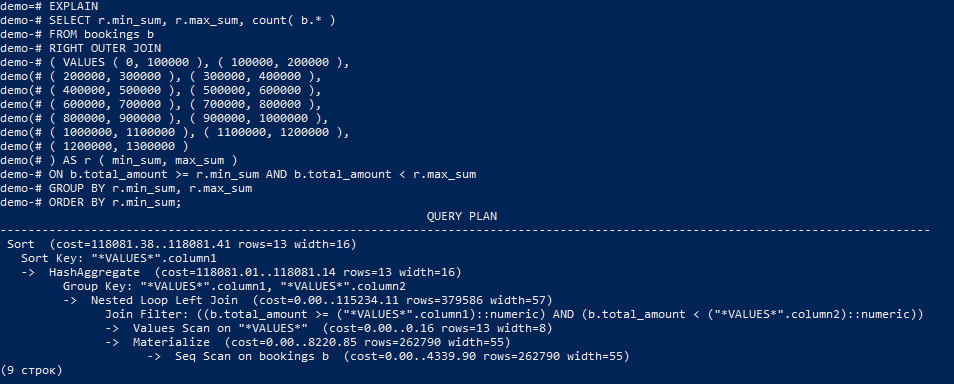


Рисунок 40 – EXPLAIN Запросов, часть 7

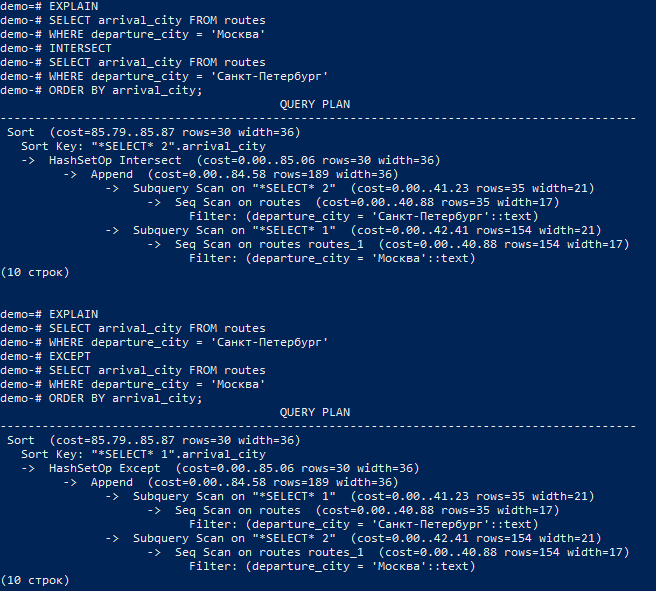


Рисунок 41 – EXPLAIN Запросов, часть 8

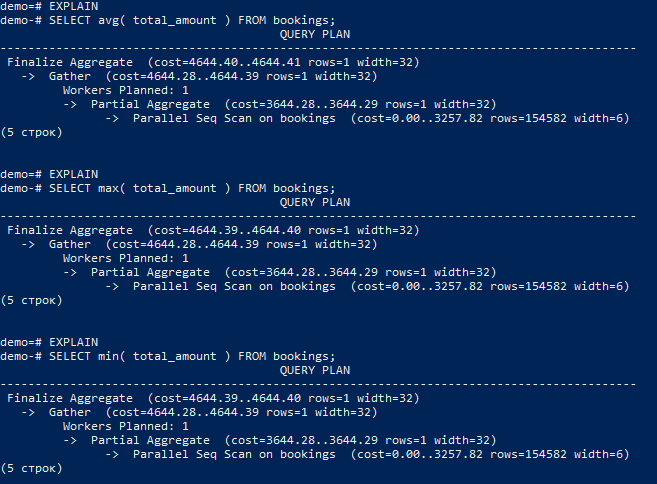


Рисунок 42 – EXPLAIN Запросов, часть 9

## Задание 16

В разделе документации 19.7 «Планирование запросов» приведены параметры, с помощью которых можно влиять на решения, принимаемые планировщиком. В тексте главы мы уже говорили о параметрах, управляющих выбором способа соединения наборов строк, и показали простой пример. Также было сказано и о том, что при установке значений параметров enable\_hashjoin, enable\_mergejoin и enable\_nestloop в off не накладывается полного запрета на использование соответствующих методов. Вместо этого конкретному методу назначается очень высокая стоимость.

Давайте проведем следующий эксперимент: запретим использование всех методов соединения наборов строк и выполним запрос, в котором соединяются две таблицы:

SET enable\_hashjoin = off;

SET enable\_mergejoin = off;

SET enable\_nestloop = off;

Запрос выводит информацию о числе мест в самолетах всех моделей:

EXPLAIN

SELECT a.model, count(\*)

FROM aircrafts a, seats s

WHERE a.aircraft\_code = s.aircraft\_code

GROUP BY a.aircraft\_code;

Обратите внимание на оценки стоимости выполнения запроса. Резкое повышение оценок происходит именно в узле, отвечающем за соединение наборов строк. Эти оценки не означают, что время выполнения запроса будет стремиться к бесконечности. С помощью команды EXPLAIN ANALYZЕ выполните запрос и убедитесь в этом сами.

Задание. Самостоятельно ознакомьтесь с содержанием раздела документации 19.7 «Планирование запросов», а также раздела 14.3 «Управление планировщиком с помощью явных предложений JOIN» и проведите эксперименты с запросами, приведенными в главе 6 пособия, получая различные варианты планов и сравнивая их.

Ваша задача — понять, как изменения значений этих параметров влияют на план выполнения запроса. Однако для того чтобы понимать, когда и почему нужно изменять значения конкретных параметров, правильно оценивать степень и направленность их влияния, понимать взаимосвязь параметров, требуется опыт и изучение документации.

На рисунке 43 представлен прогресс работы.

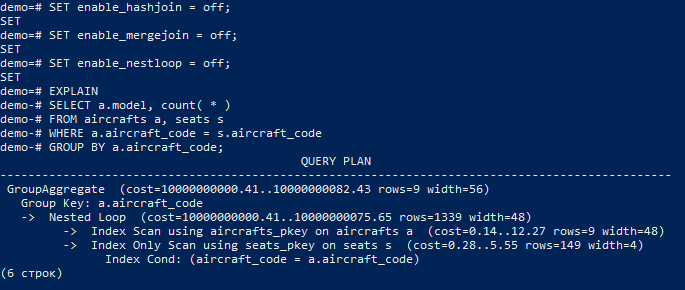


Рисунок 43 – Анализ запроса

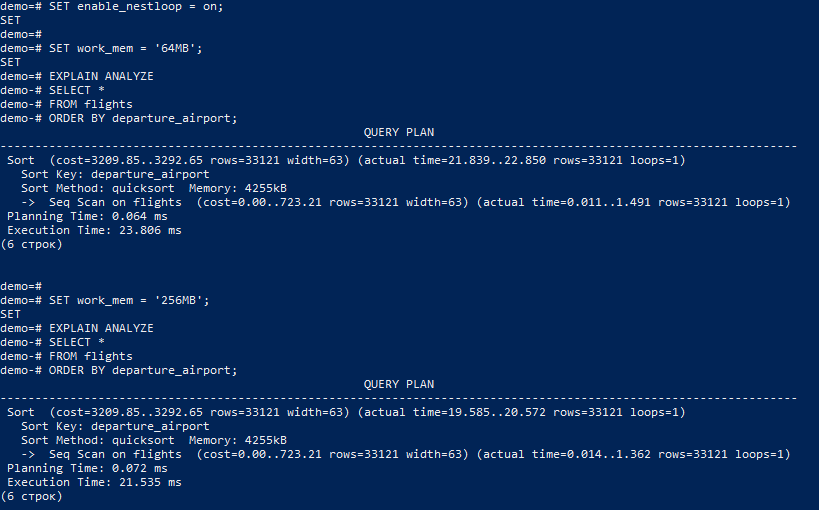


Рисунок 44 – Анализ «nestloop»

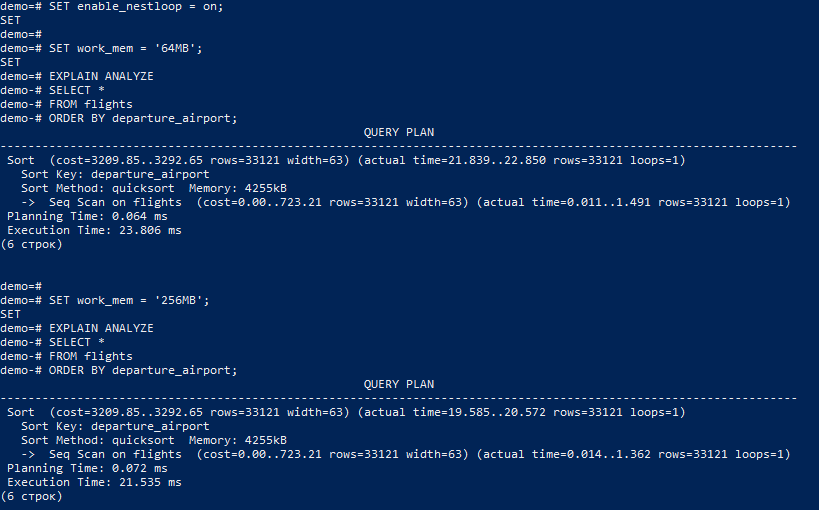


Рисунок 45 – Анализ «work\_mem»

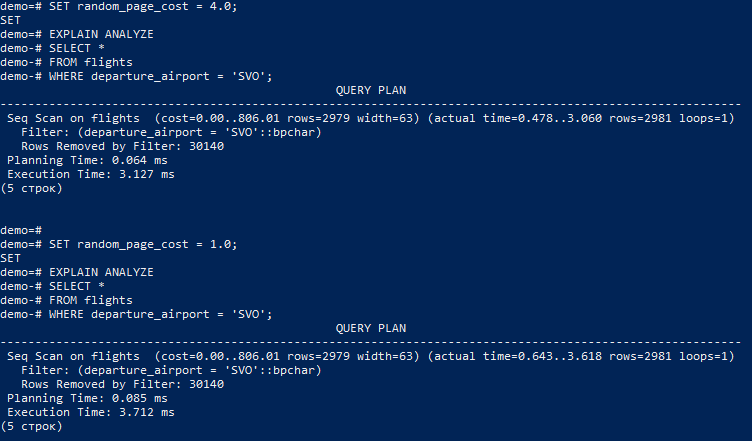


Рисунок 46 – Анализ «random\_page\_count»

## Задание 17

Самостоятельно ознакомьтесь с разделом документации 14.2 «Статистика, используемая планировщиком».

Раздел «Статистика, используемая планировщиком» был изучен.

## Задание 18

Команда EXPLAIN имеет опцию BUFFERS. Ознакомьтесь с ней самостоятельно по разделу документации 14.1 «Использование EXPLAIN».

На рисунках 47-48 показан прогресс работы.

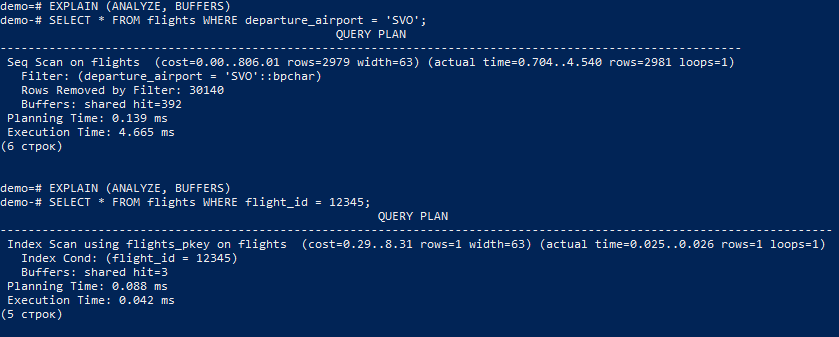


Рисунок 47 – Анализ BUFFERS, часть 1

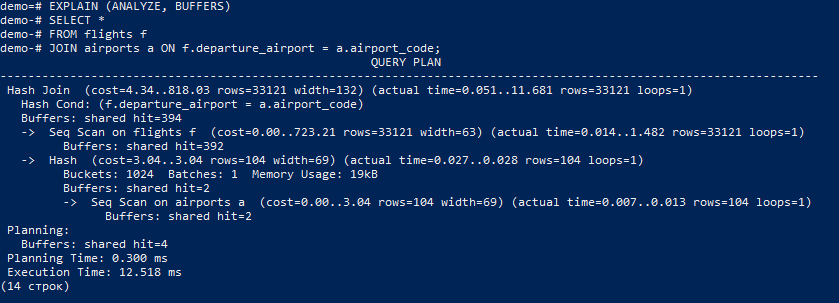


Рисунок 48 – Анализ BUFFERS, часть 2

## Задание 19

При массовом вводе данных в базу данных производительность СУБД может снижаться по ряду причин. Например, при наличии индексов они обновляются при вводе каждой новой строки в таблицу, что требует дополнительных затрат ресурсов.

Для повышения производительности СУБД в подобных ситуациях в документации предлагается ряд мер. В частности, рекомендуется удаление индексов перед началом массового ввода данных и их пересоздание после завершения такого ввода.

Ознакомьтесь с этими мерами самостоятельно по разделу документации 14.4 «Наполнение базы данных».

На рисунках 49-52 показан прогресс работы.

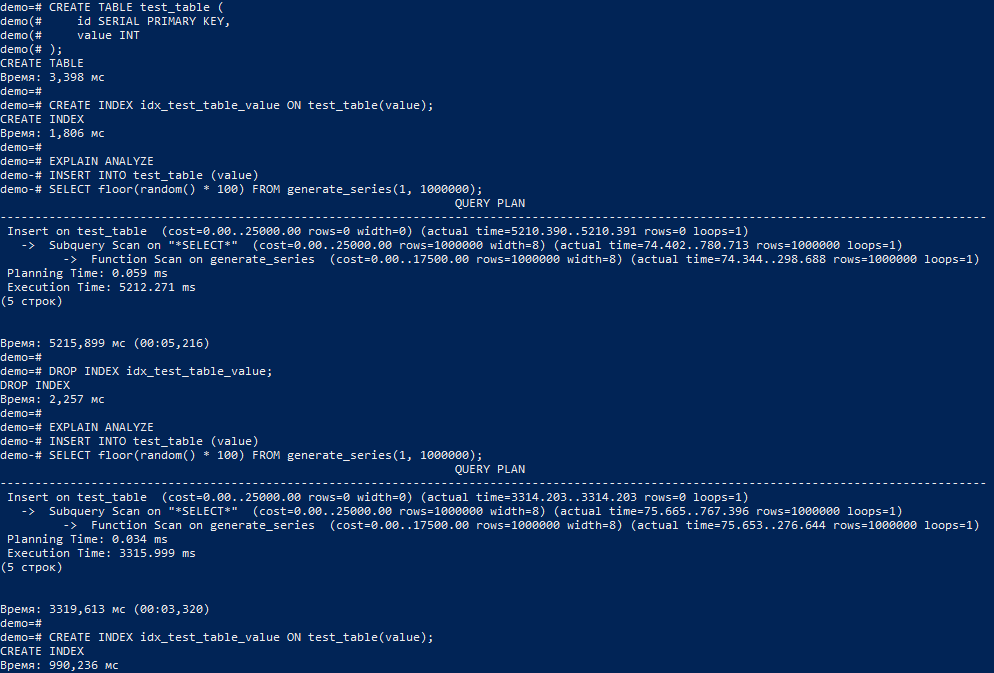


Рисунок 49 – Эксперименты, часть 1

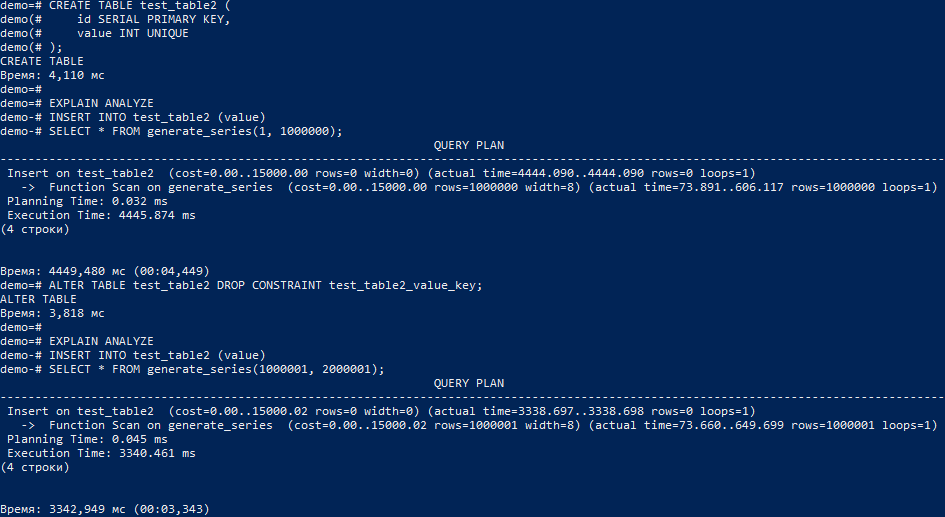


Рисунок 50 – Эксперименты, часть 2

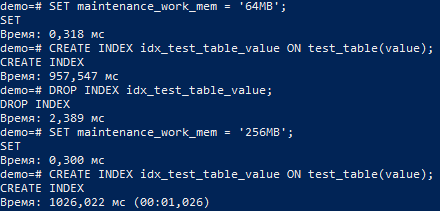


Рисунок 51 – Эксперименты, часть 3

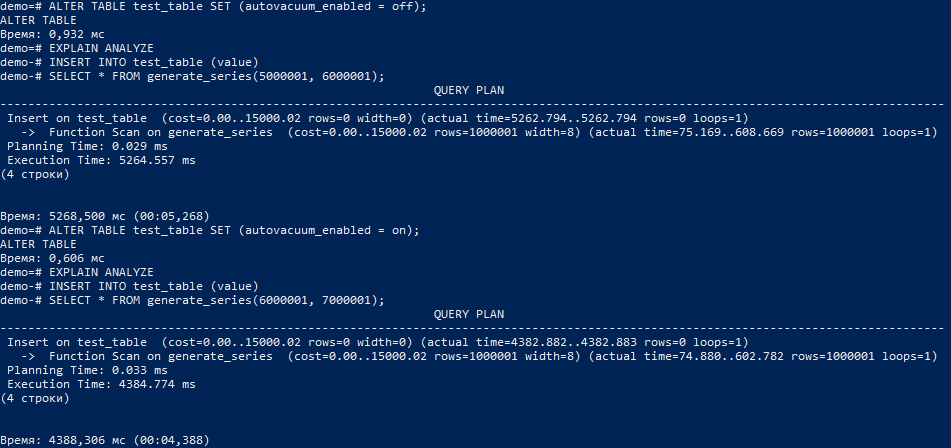


Рисунок 52 – Эксперименты, часть 4

# ЗАКЛЮЧЕНИЕ

По результатам работы был изучен теоретический материал по теме «Повышение производительности». Все поставленные цели и задачи были выполнены.