|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных**

**Отчет**

**по лабораторной работе № 2**

**Название:** Оптимизация запросов. Основы EXPLAIN в PostgreSQL.

Индексация.

**Дисциплина:** Технология параллельны систем баз данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-12М |  |  | Д.С. Каткова |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | А.Д. Пономарев |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

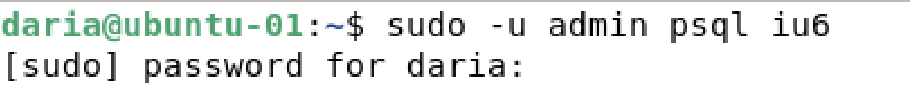
Москва, 2023

1. **Цель лабораторной работы**

Цель работы – формирование следующей компетенции: студент должен получить навыки работы с командой EXPLAIN. Он также должен познакомиться с эффективными методами индексации в PostgreSQL.

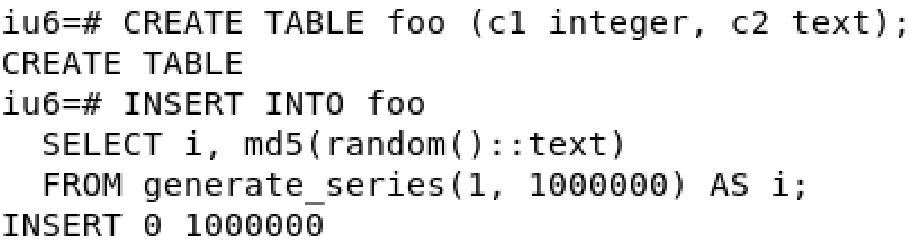
1. **Подключение к виртуальной машине и к базе данных**

Выполним подключение к базе данных.

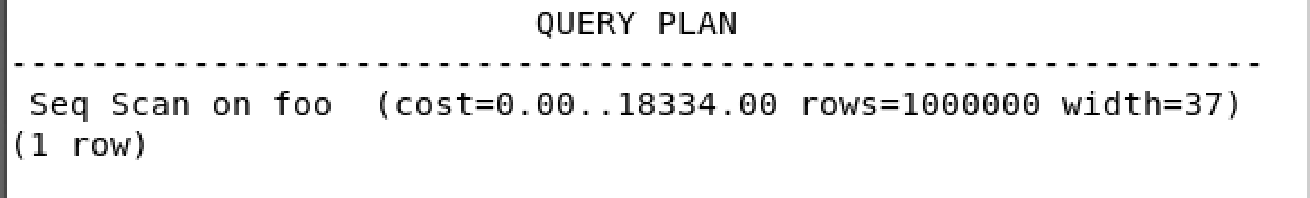


1. **Основы EXPLAIN**

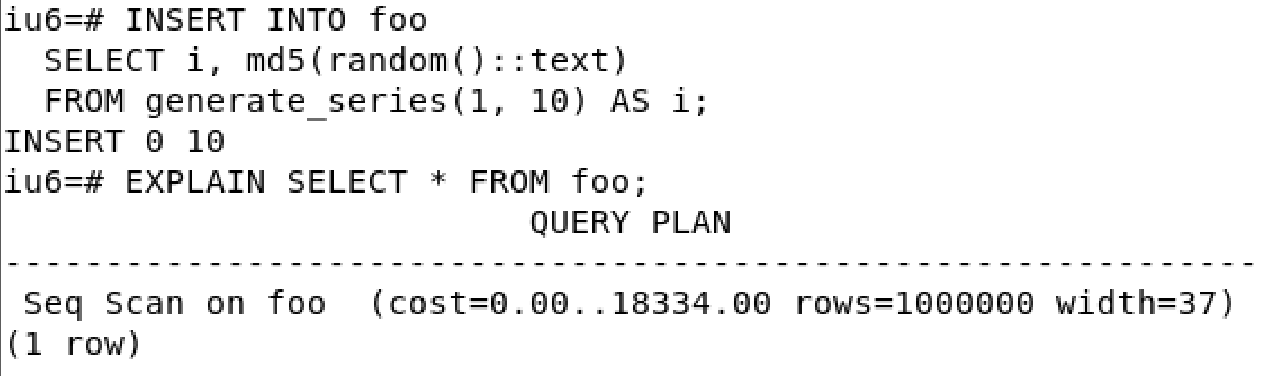
В соответствии с заданием, создадим тестовую таблицу и включим в нее 1 млн. записей.



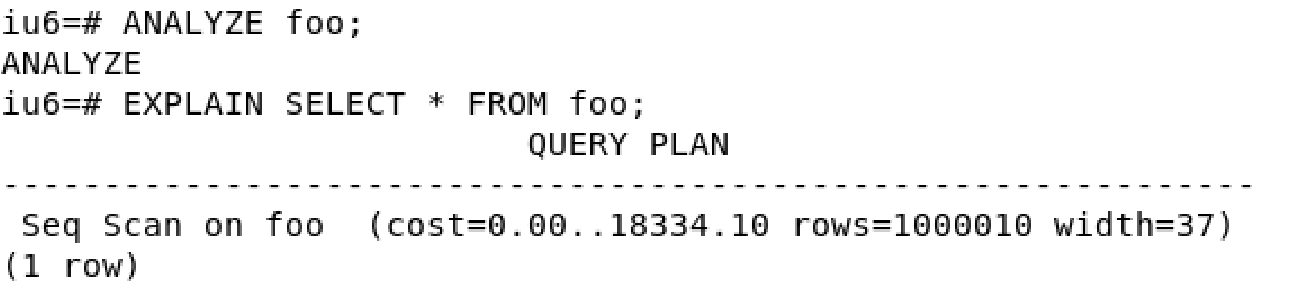
Прочитаем данные с помощью EXPLAIN.



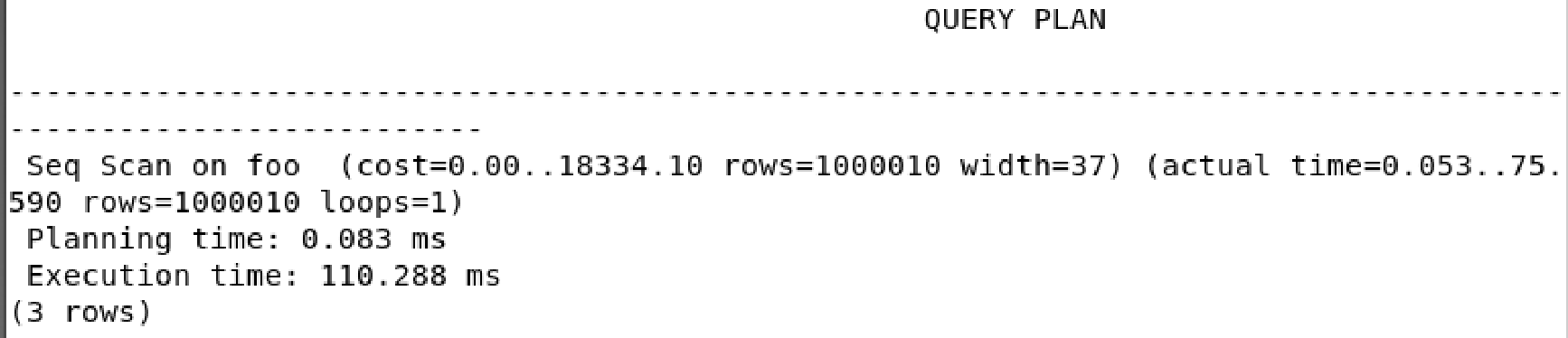
Добавим по аналогии 10 строк и повторим команду EXPLAIN.



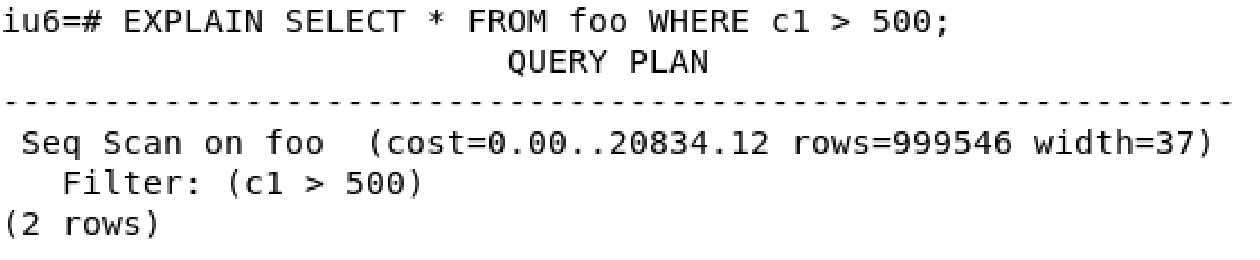
Можно заметить, что вывод не изменился. Это произошло потому, что хоть записи и были добавлены в таблицу, ее статистика обновлена не была. Чтобы обновить статистику по таблице, используем команду ANALYZE и повторим команду EXPLAIN. Теперь статистика обновлена.



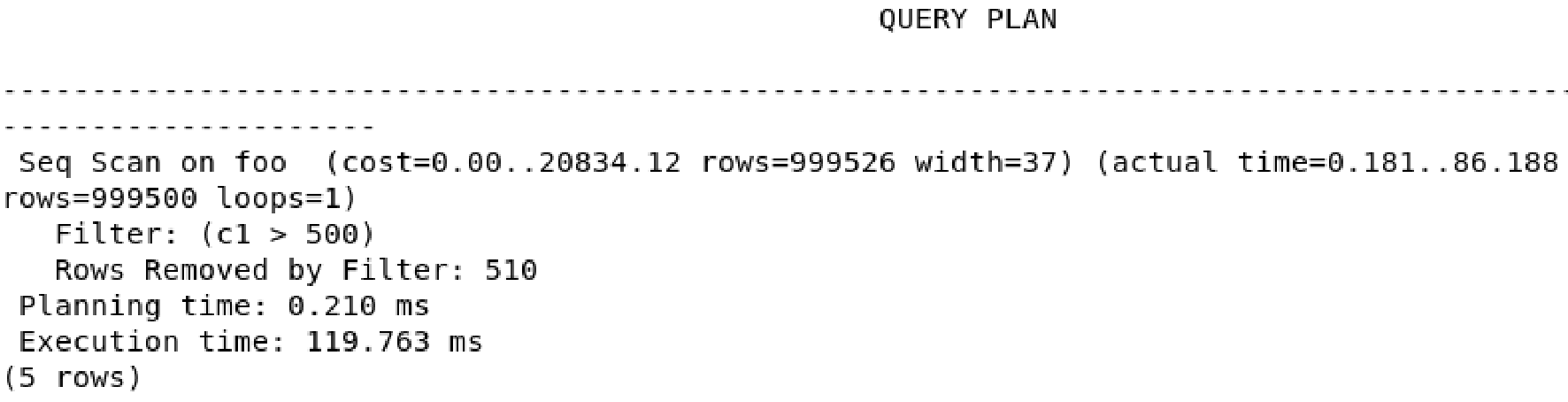
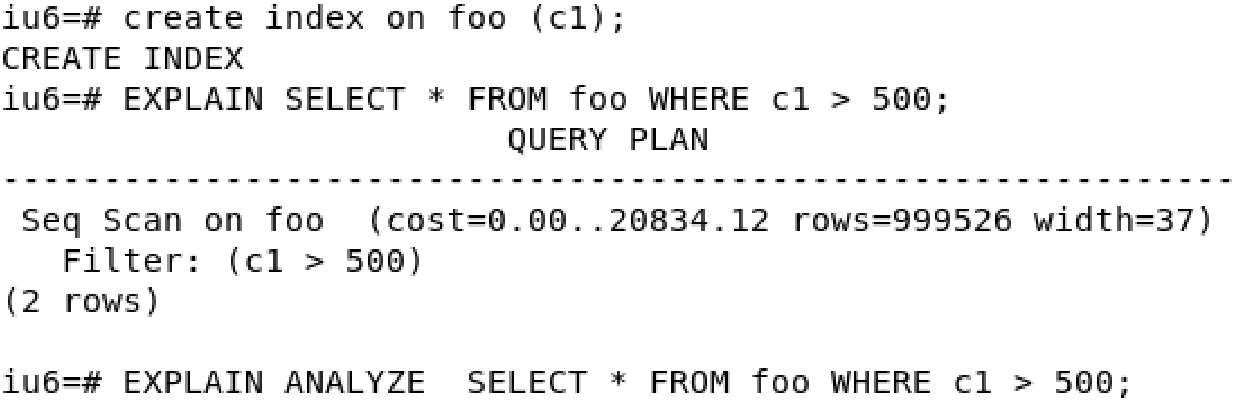
Далее используем параметры EXPLAIN и ANALYZE вместе.



Добавим в запрос новое условие.

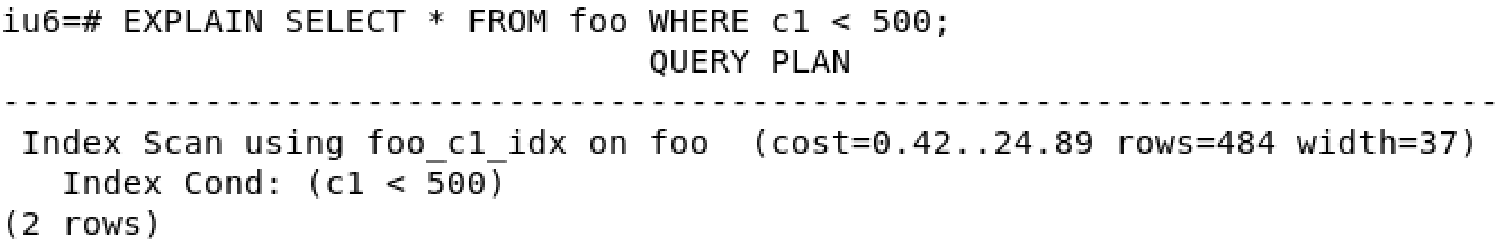


Как можно увидеть, индекс в данном запросе не используется, о чем говорит надпись Seq Scan. Теперь создадим индекс и повторим команды EXPLAIN и EXPLAIN ANALYZE.



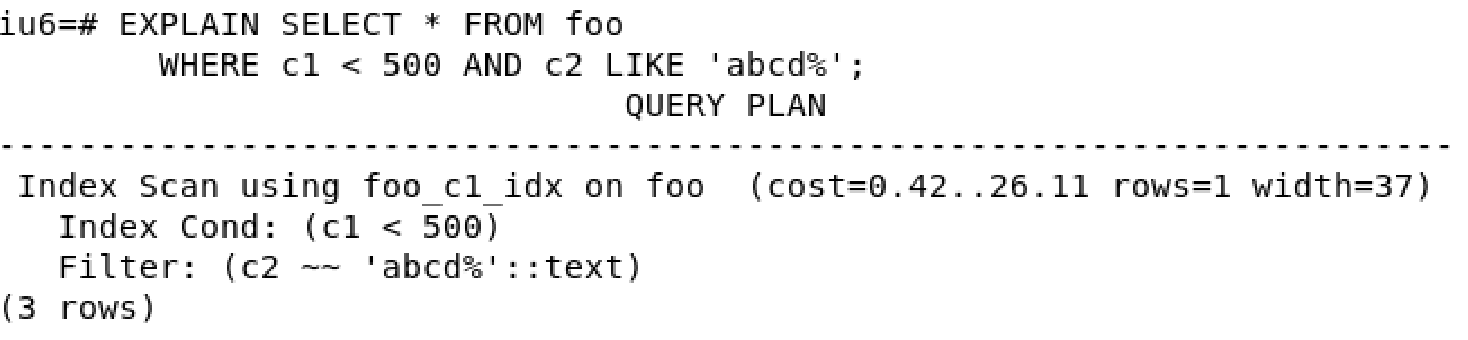
Хоть мы и создали индекс, он не использовался. Это произошло потому, что в данном случае для запроса выгоднее использовать последовательное сканирование. Индекс же будет использоваться только в том случае, если процент записей, удовлетворяющих условию, не превышает 5-10% от всех строк в таблице.

Немного изменим условие и снова произведем запрос.

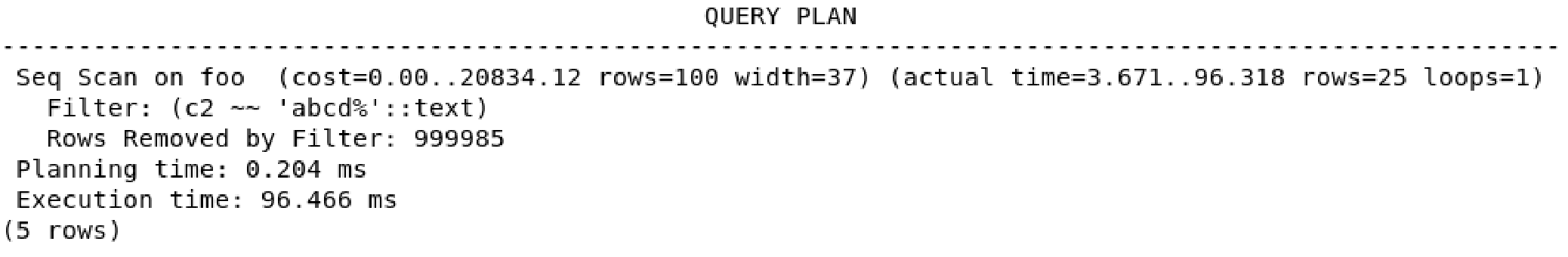


Уже в этом случае, как можно увидеть, индекс используется. Это произошло по причине небольшого количества строк, удовлетворяющих условию (543 строк из 10000010, что составляет 0,005% от всех строк).

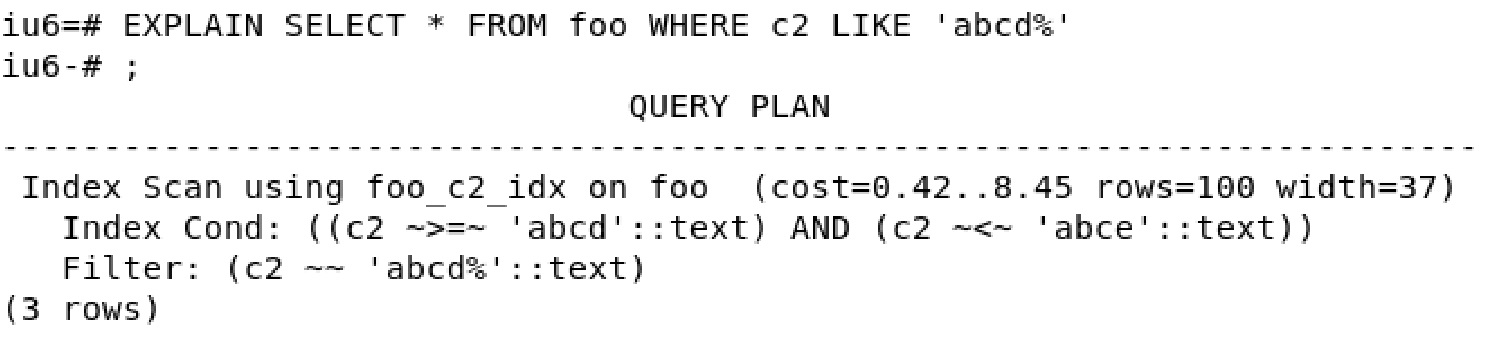
Дополнительно усложним условие. Используем еще одно текстовое поле и выполним команду EXPLAIN.



Как можно увидеть, в данном запросе все еще используется индекс. Уберем условие с проверкой численного поля и оставим только проверку текстового поля.

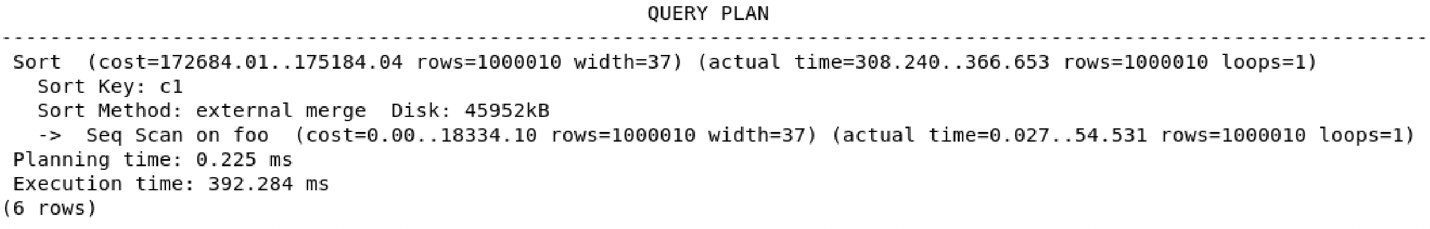
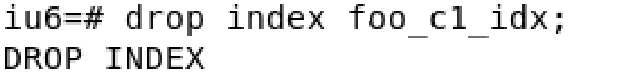


Как можно увидеть, здесь был произведен последовательный поиск. Это произошло по той причине, что никакого индекса на текстовой поле в нашей БД создано не было. Теперь создадим его и вновь проверим работу запроса.

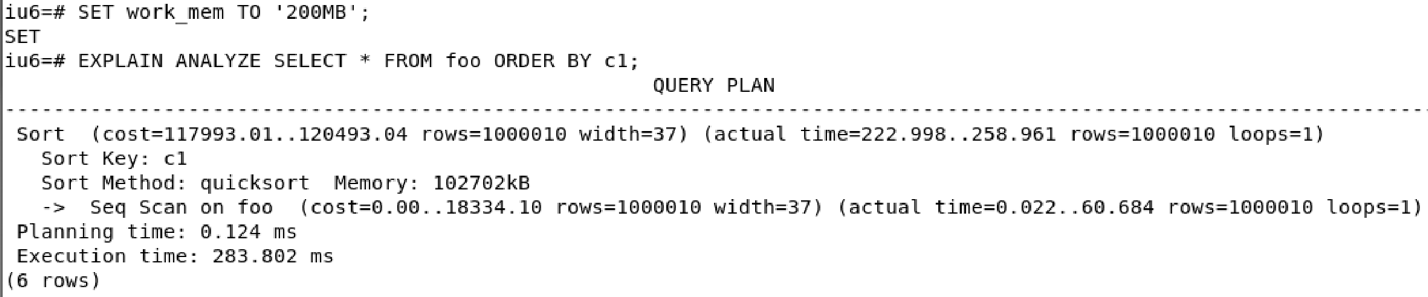


Как можно увидеть, после создания индекса он начал использоваться.

Удалим индекс, созданный для атрибута с1 таблицы foo. Выполним новую команду.

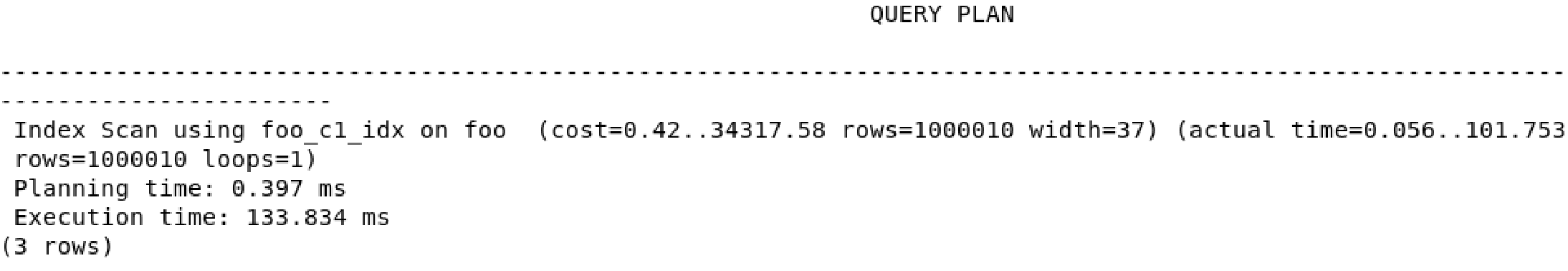


Сейчас методом сортировки используется external merge. Увеличим объем используемой памяти и повторим данный запрос.



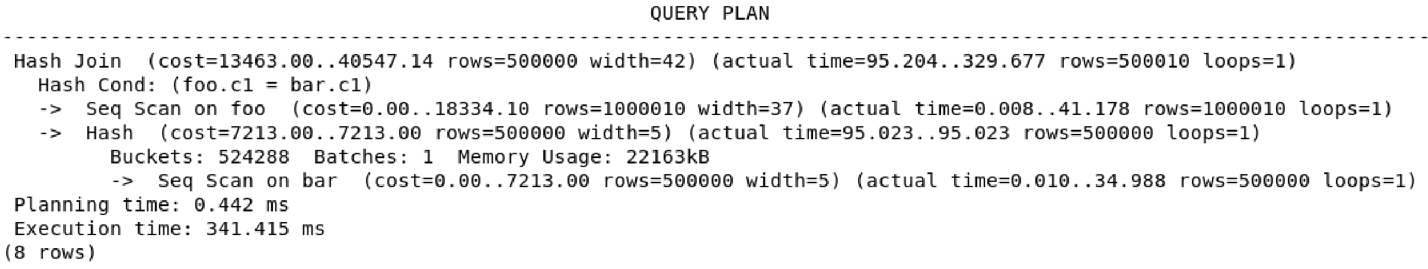
Как можно увидеть, после увеличения количества используемого объема памяти метод сортировки поменялся с external merge на quicksort. Это произошло по той причине, что хоть quicksort и является самой быстрой, она расходует достаточно много памяти. Из-за этого, раньше использовать ее было бы нерационально.

Вновь создадим индекс для с1 таблицы foo и повторим предыдущий запрос.



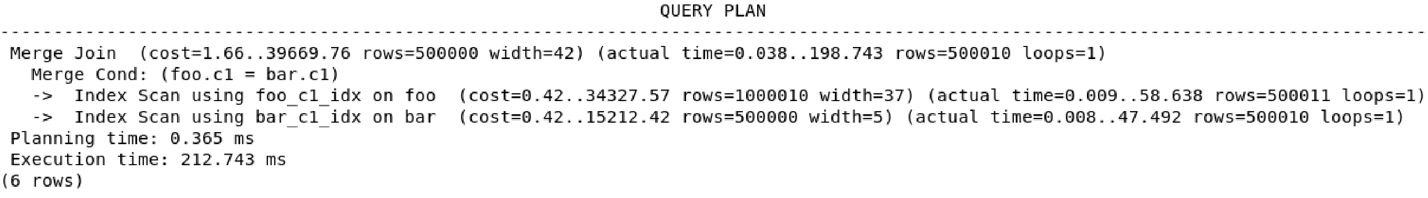
Как можно увидеть, теперь и для сортировки записей в таблице используется индекс.

Создадим новую таблицу, соберем с нее статистику и выполним новую команду.



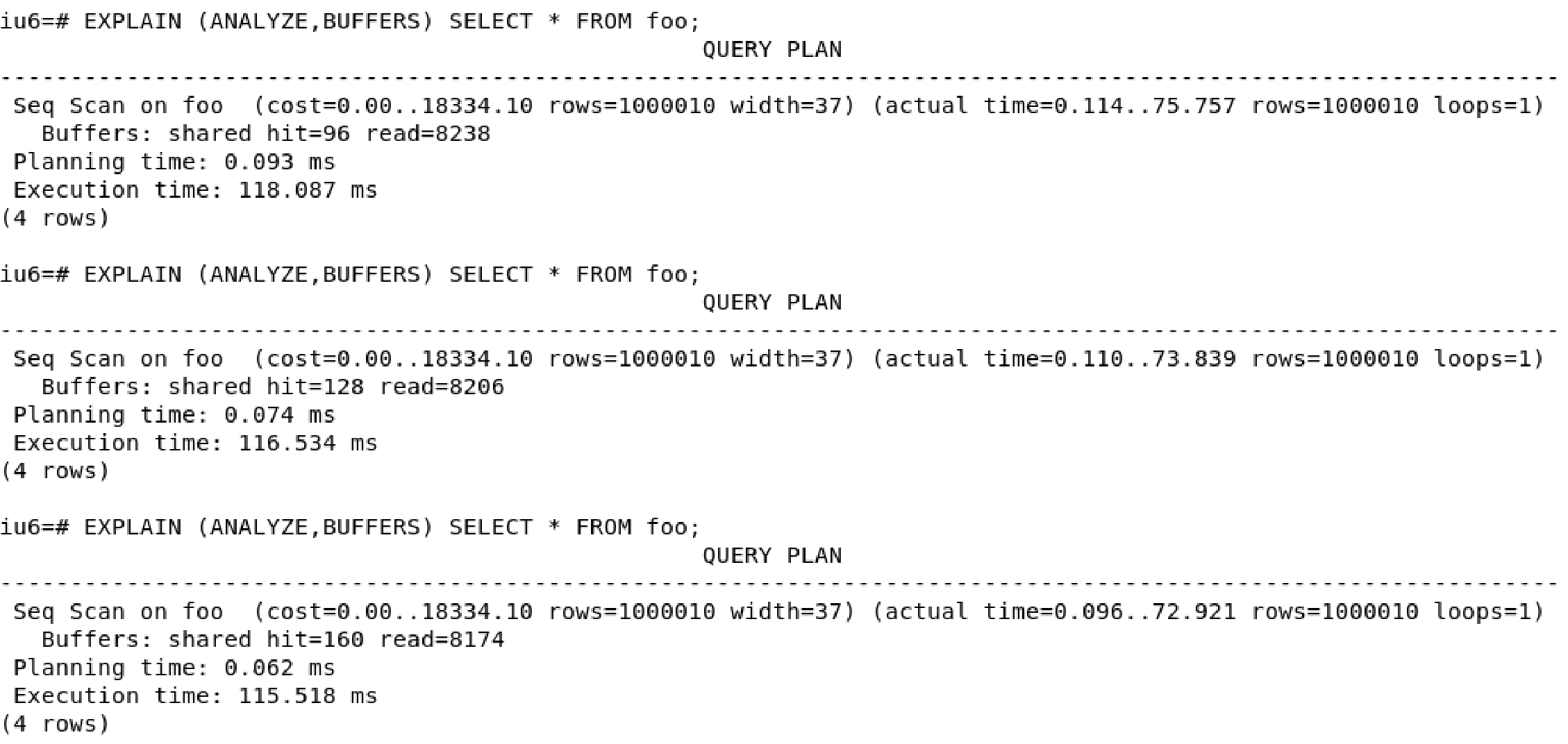
Как можно увидеть, здесь планировщик выбирает соединение по хешу, при котором строки одной таблицы записываются в хеш-таблицу в памяти, после чего сканируется другая таблица и для каждой ее строки проверяется соответствие по хеш-таблице.

Далее создадим индекс для атрибута с1 таблицы bar и проверим изменения.



После создания индекса время выполнения запроса уменьшилось.

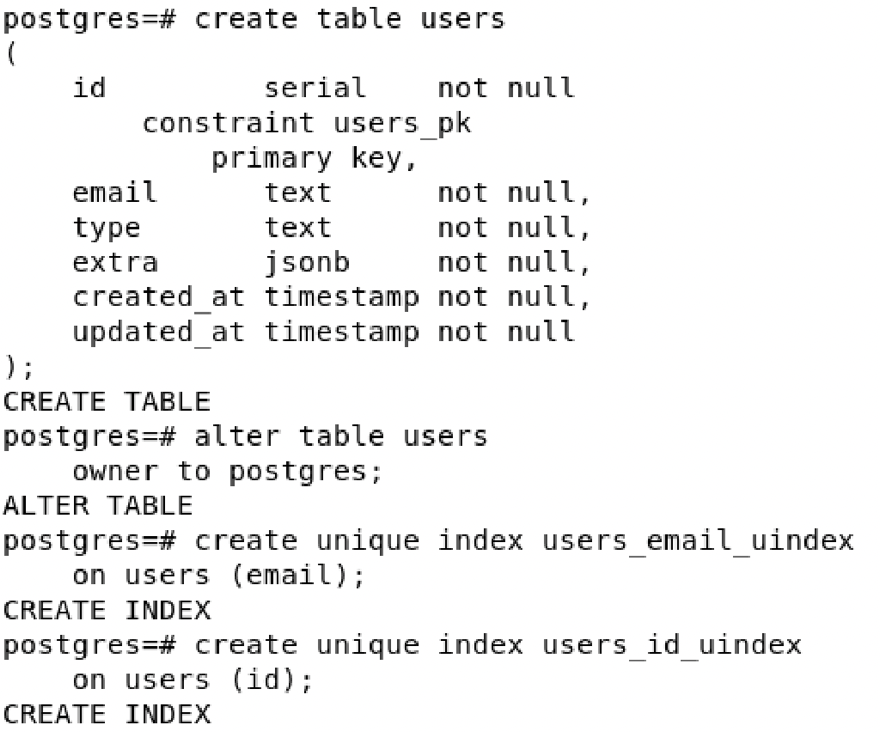
Далее удалим индекс, созданный для с1 таблицы foo, выйдем из консоли, остановим PostgreSQL, принудительно зафиксируем изменения в файловой системе, очистим кеш и заново запустим PostgreSQL. Войдем в консоль и сделаем запрос из задания три раза.



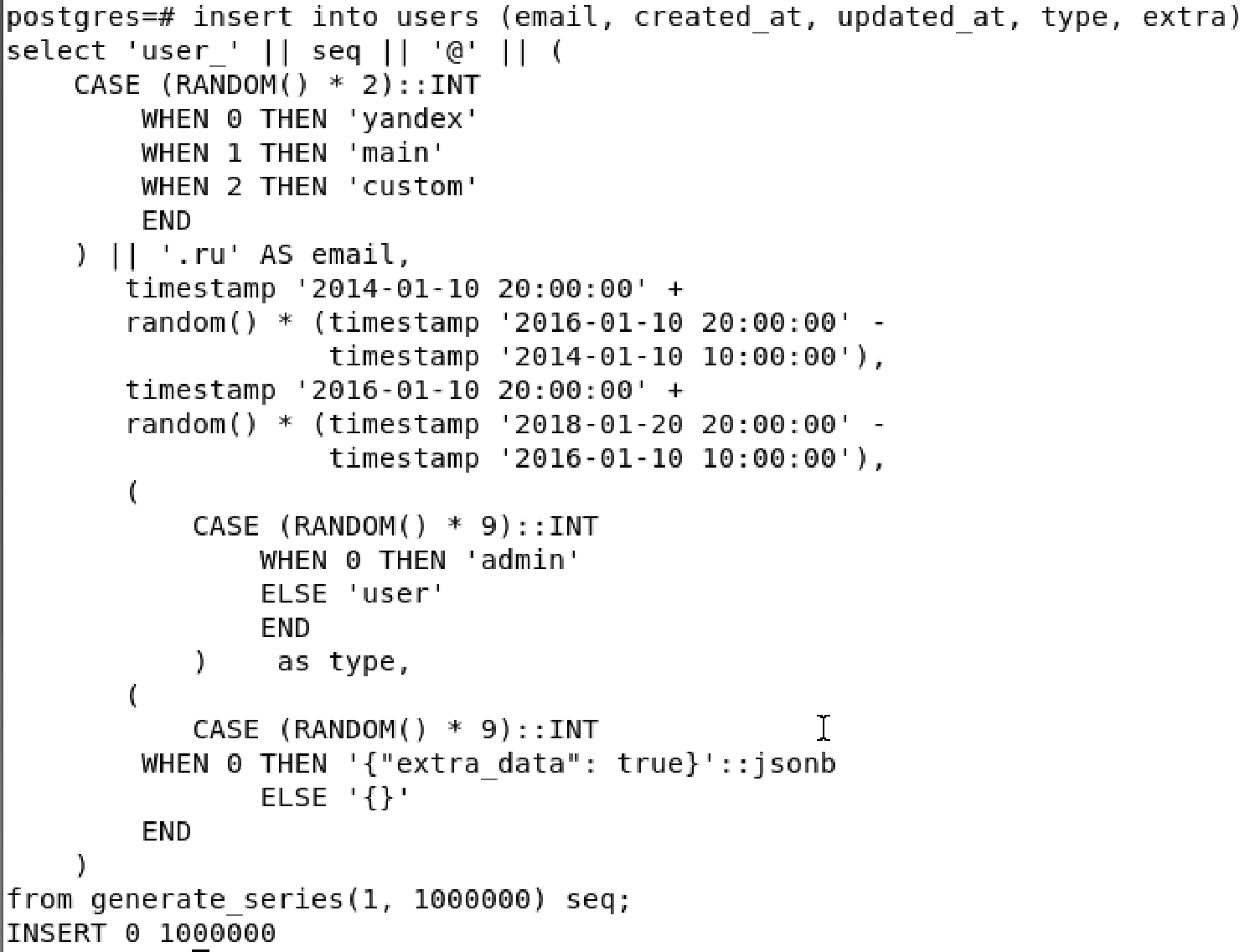
Как можно увидеть, с каждым выполнением запроса количество блоков, считанных из кэша PostgreSQL, становится все меньше.

1. **Эффективные методы индексации в PostgreSQL**

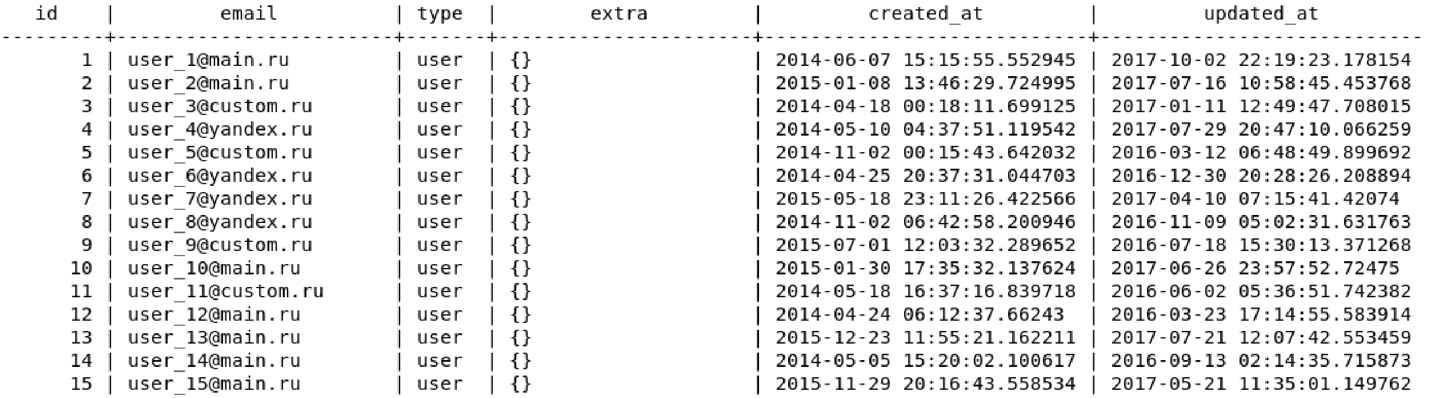
Выйдем из консоли PostgreSQL и подключимся к базе postgres как пользователь postgres, после чего создадим базу данных.



Теперь заполним базу данных.



Итоговая таблица выглядит следующим образом.



**Задание 1. Создать индекс для поиска по контексту значения поля email.**

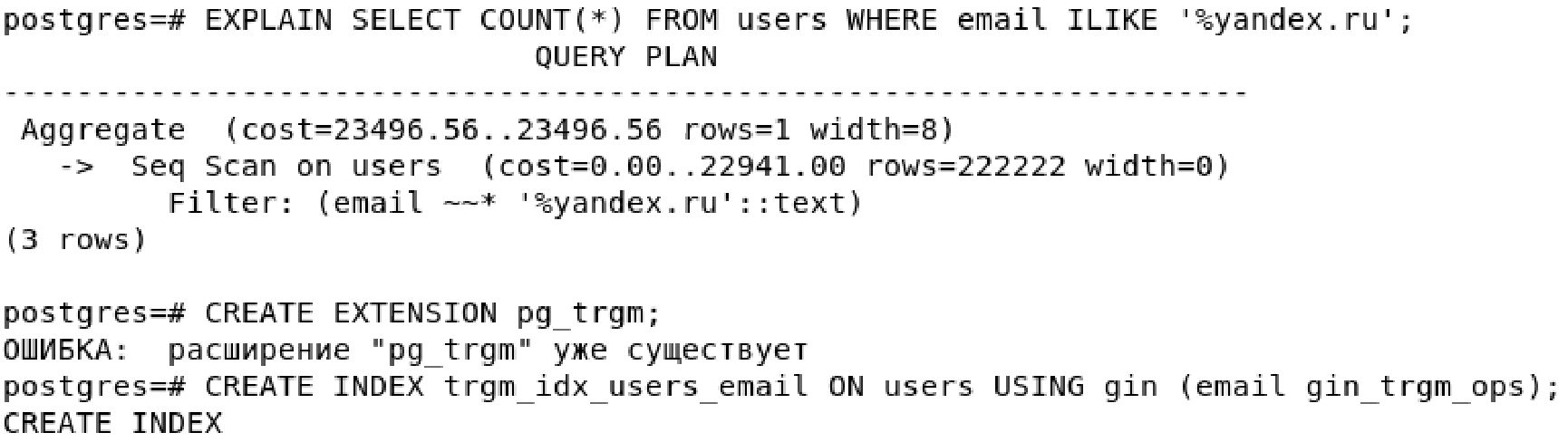
При выполнении будем использовать модуль pg\_trgm и оператор индекса gin. Выполнение самого SELECT выглядит следующим образом.



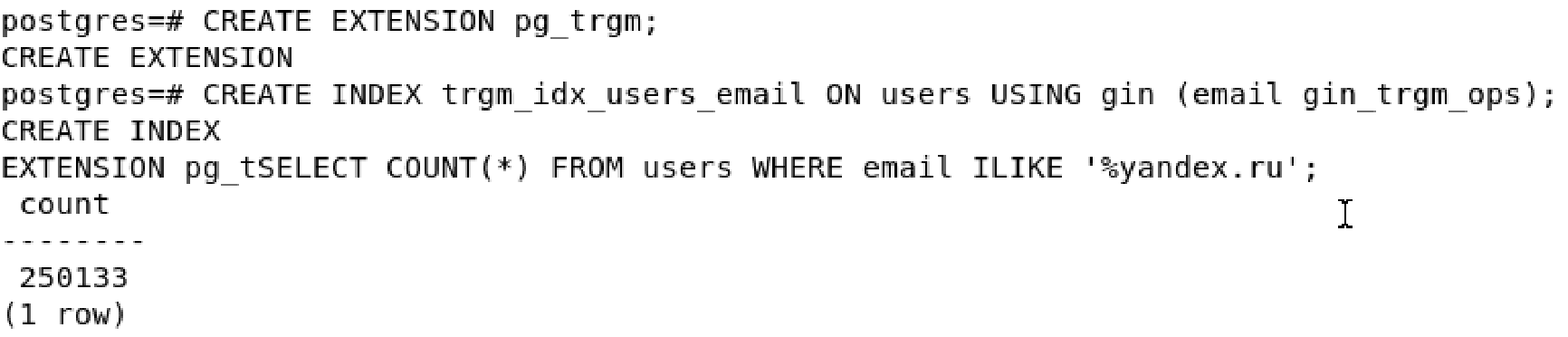
Выполнение SELECT

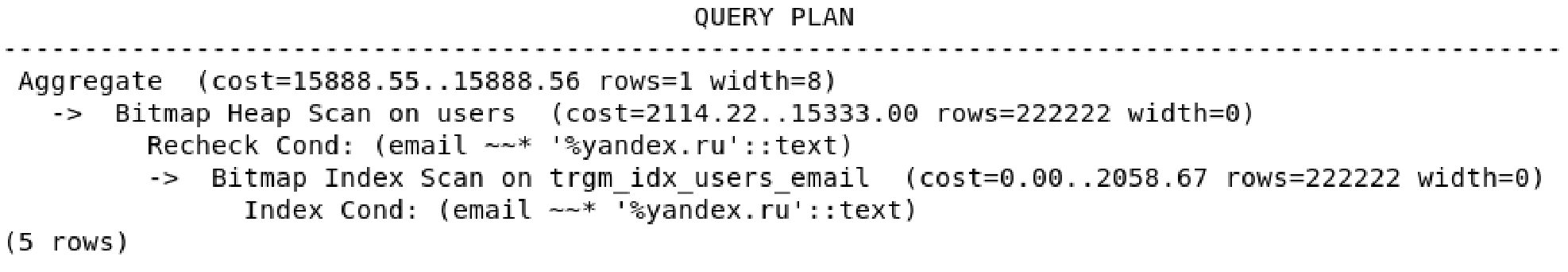
Выполним следующую команду до создания индекса.

EXPLAIN SELECT COUNT(\*) FROM users WHERE email ILIKE '%yandex.ru';



Создадим индекс и выполним предыдущую команду еще раз.

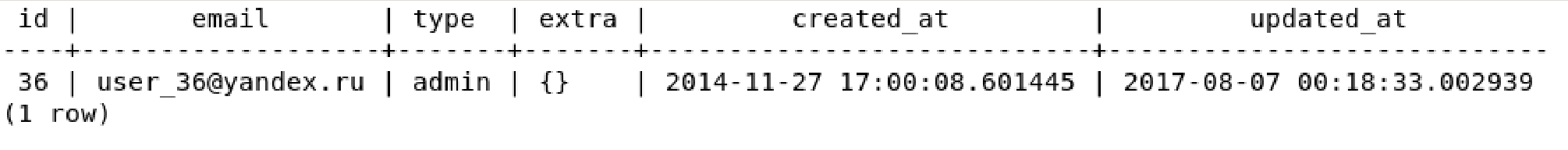




В отличие от первого раза, теперь планировщиком используется индекс.

**Задание 2. Создать индекс по update\_at пользователей, но только для админов (type = ‘admin’)**

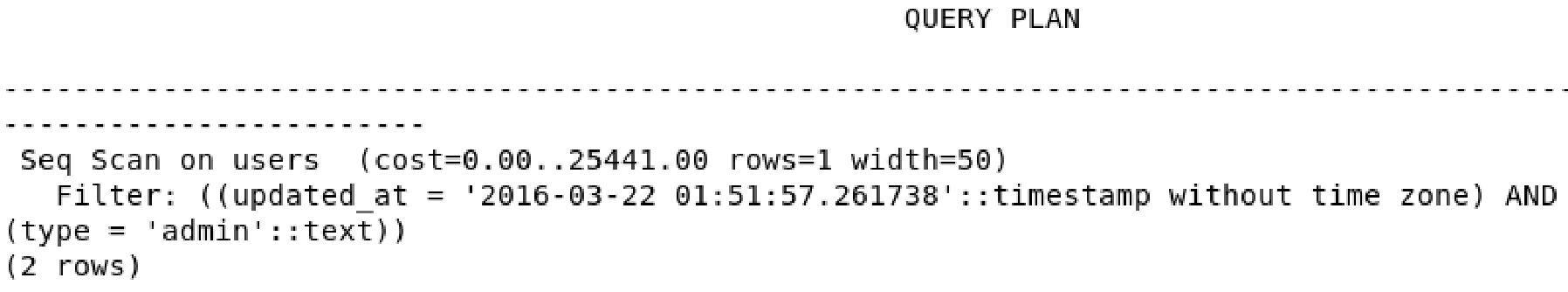
При выполнении будем использовать частичные индексы. Выполнение самого SELECT выглядит следующим образом.



Была найдена одна строка соответствующая условию type = ‘admin’ и update\_at = '2017-08-07 00:18:33.002939'

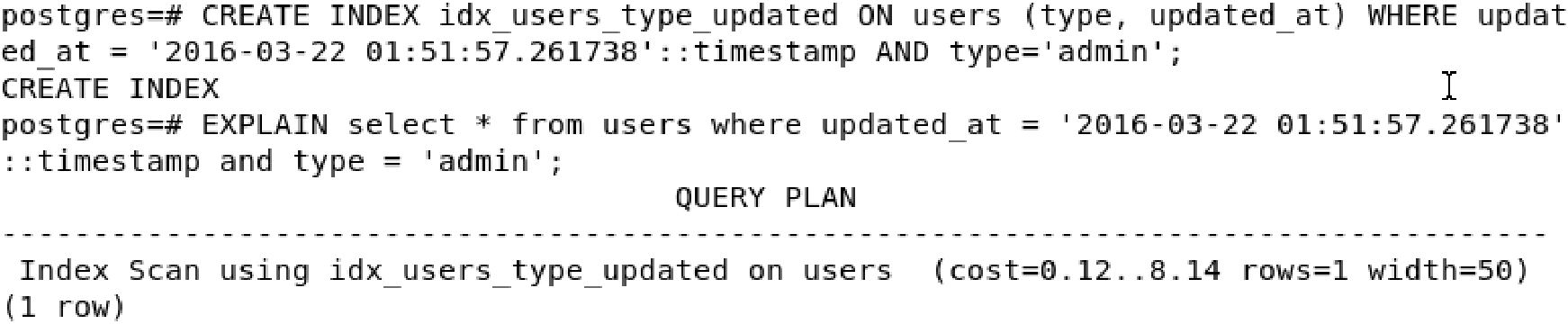
Выполним следующую команду до создания индекса.

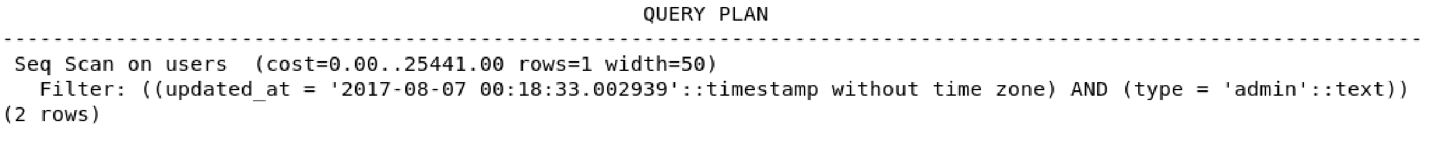
EXPLAIN SELECT \* FROM users WHERE updated\_at = '2017-08-07 00:18:33.002939'::timestamp and type = 'admin';



Создадим индекс и выполним предыдущую команду еще раз.

CREATE INDEX idx\_users\_type\_updated ON users (type, updated\_at) WHERE updated\_at = '2016-03-22 01:51:57.261738'::timestamp AND type='admin';

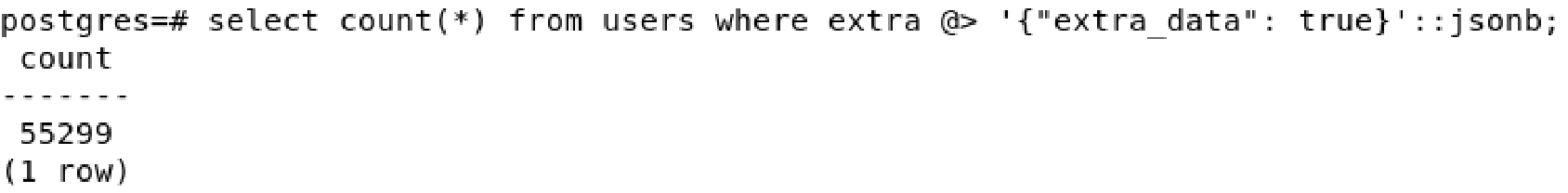




В отличие от первого раза, теперь планировщиком используется индекс.

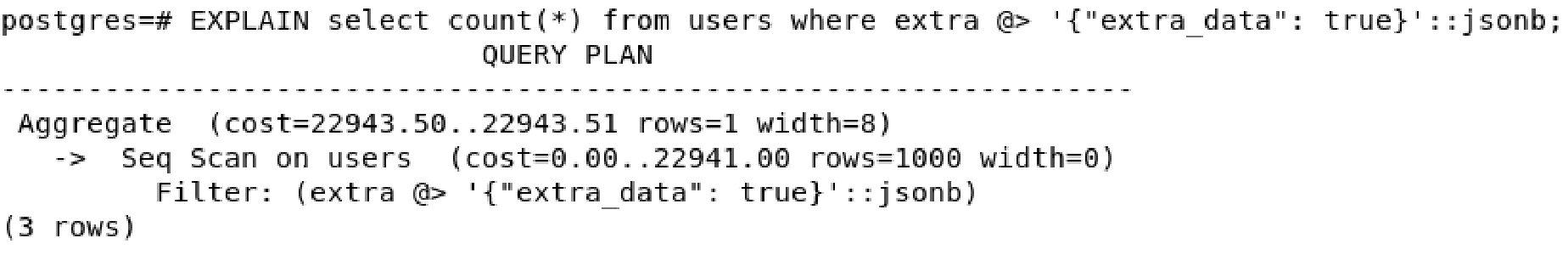
**Задание 3. Создать индекс для поля extra.**

При выполнении будем использовать эффективное индексирование данных jsonb. Выполнение самого SELECT выглядит следующим образом.



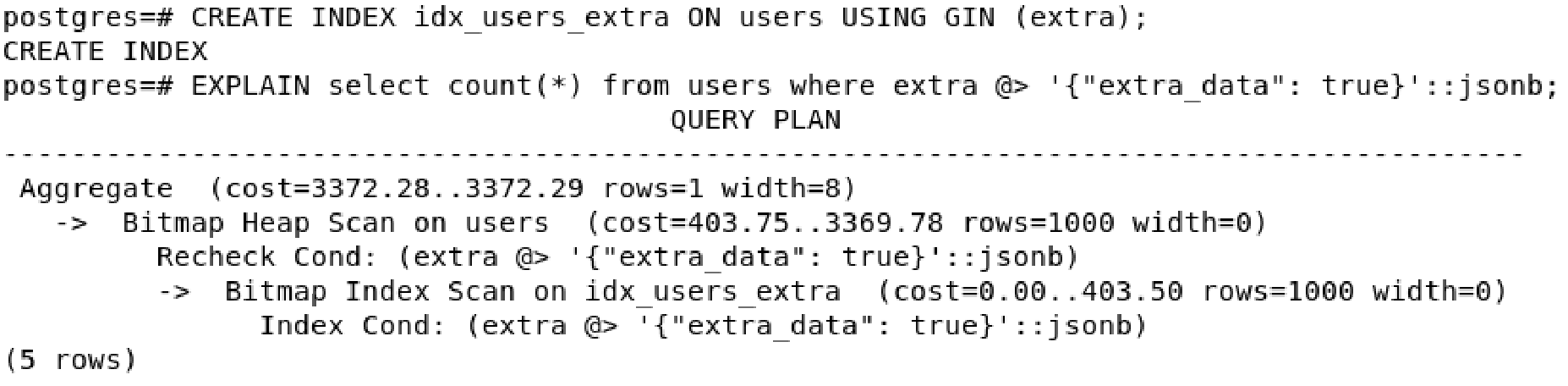
Выполним следующую команду до создания индекса.

EXPLAIN SELECT COUNT(\*) FROM users WHERE extra @> '{"extra\_data": true}'::jsonb;



Создадим индекс и выполним предыдущую команду еще раз.

CREATE INDEX idx\_users\_extra ON users USING GIN (extra);



В отличие от первого раза, теперь планировщиком используется индекс.

В соответствии с заданием каждый запрос был покрыт индексом, вследствие чего при выполнении каждого запроса seq scan не наблюдался.

1. **Вывод**

В ходе лабораторной работы были получены навыки работы с командой EXPLAIN. Было произведено знакомство с эффективными методами индексации в PostgreSQL. Все задания были успешно выполнены, а их результаты соответствуют требованиям.