Индексы

Важность быстрого поиска

- 53% пользователей покидают страницу, если она грузится дольше 3 секунд
- По исследованиям Amazon, увеличение времени загрузки страницы на 100 ms приводит к снижению продаж на 1%. В пересчёте на текущий курс это +- 3,8 миллиарда долларов
- По исследованиям Google, увеличение загрузки страницы на 0.5 секунды приводит к потере 20% трафика
- Google и Яндекс отслеживает время загрузки страницы. Чем оно меньше, тем выше в поиске сайт будет отображен
- Будучи студентом отпадает желание поступать в ВУЗ, у которого лагает сайт

Домен

- Студенты - 1 миллион
- Преподаватели - 100
- Курсы - 50
- Записи на курс - 2 миллиона
- Оценки - 3 миллиона

План

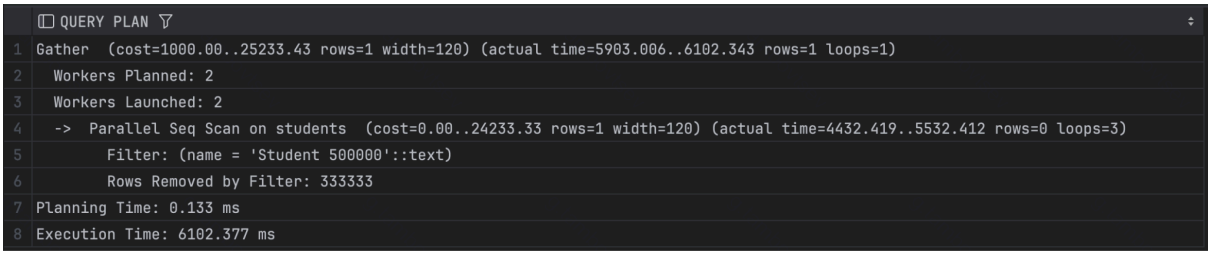
Запускаем PostgreSQL при помощи Docker. Обращаем внимание, что мы выставили параметр --cpus=”0,1”. В реалиях даже нашего объёма данных довольно сложно нагрузить БД так, чтобы запрос выполнялся хотя бы 1 секунду. Поэтому мы намеренно занижаем ресурсы контейнера, чтобы можно было отследить изменения в скорости выполнения запросов

```
docker run -d \
 --name hse-db-indexes-with-data \
 --cpus="0.1" \
 --memory="512m" \
 -e POSTGRES_USER=postgres \
 -e POSTGRES_PASSWORD=postgres \
 -e POSTGRES_DB=postgres \
 -p 5432:5432 \
 postgres:15
```

Проверяем, что база успешно запустилась и мы можем к ней подключиться

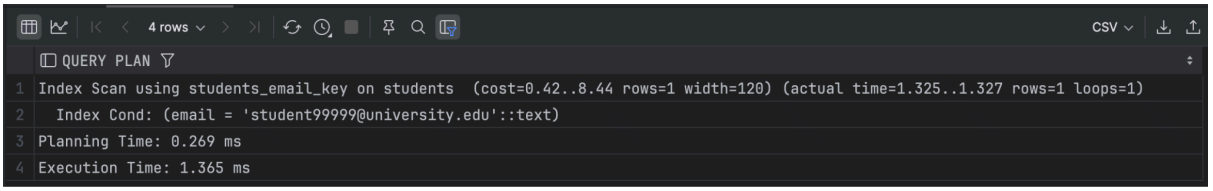
Создаем таблицы из файла `create_tables.sql`. Генерируем данные файликом `gen_data.sql` и обязательно фиксируем, сколько времени у нас это заняло (процесс небыстрый с учетом выделенных мощностей)

Прогоняем запросы из файлы `slow_queries.sql` и фиксируем значения полученные из `EXPLAIN ANALYZE`. На примере первого запроса мы видим, что выполняется **Seq Scan**, что означает проход по всей таблице



```
QUERY PLAN
Gather  (cost=1000.00..25233.43 rows=1 width=120) (actual time=5903.006..6102.343 rows=1 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  ->  Parallel Seq Scan on students  (cost=0.00..24233.33 rows=1 width=120) (actual time=4432.419..5532.412 rows=0 loops=3)
        Filter: (name = 'Student 500000'::text)
        Rows Removed by Filter: 333333
Planning Time: 0.133 ms
Execution Time: 6102.377 ms
```

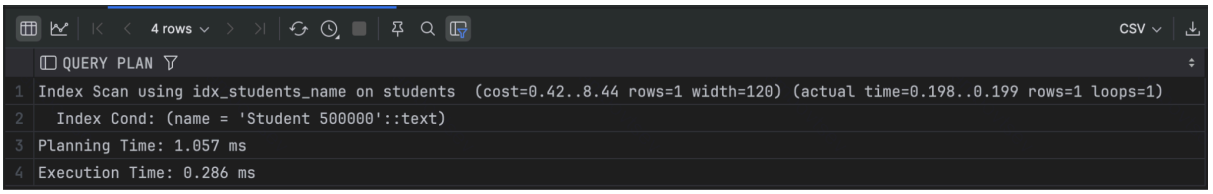
В то время как поиск по `email` гораздо быстрее и поиск выполняется по **Index Scan**. Странно, вроде никаких индексов ещё не добавили. Но вернемся к этому позже



```
4 rows                                                                                    CSV
QUERY PLAN
Index Scan using students_email_key on students  (cost=0.42..8.44 rows=1 width=120) (actual time=1.325..1.327 rows=1 loops=1)
  Index Cond: (email = 'student99999@university.edu'::text)
Planning Time: 0.269 ms
Execution Time: 1.365 ms
```

Таким образом проходимся по всем запросам, фиксируя где-нибудь время выполнения и сам план запроса

Начинаем накидывать индексы из файла `indexes.sql`, что тоже занимает время, фиксируем. Начинаем гонять запросы и сравниваем время их выполнения - радуемся! На примере поиска по имени студента. По плану запроса у нас теперь идет поиск **Index Scan**



```
4 rows                                                                                    CSV
QUERY PLAN
Index Scan using idx_students_name on students  (cost=0.42..8.44 rows=1 width=120) (actual time=0.198..0.199 rows=1 loops=1)
  Index Cond: (name = 'Student 500000'::text)
Planning Time: 1.057 ms
Execution Time: 0.286 ms
```

Возвращаясь к полю `email`, вспоминаем, что на нём стоит `UNIQUE`, что по сути тоже является индексом

Вспоминаем, что на построение индексов требуется время - грустим. Обсуждаем, на какие таблицы и поля стоит добавлять индексы, а на какие нет

- Приходим к выводу, что студенты, преподаватели и курсы меняются редко
- Оценки добавляются часто
- Что делать с оценками?
    - Отключать формирование индексов на момент сессии, когда студенты активно сдают экзамены и идет постоянная запись оценок
    - partial indexes (рассказываем и пробуем)
    - Можно попробовать партиции, но про это поговорим потом

Приходим к выводу, что индексы - это не волшебная таблетка. Могут как ускорить, так и замедлить работу. Подходят в нужном месте в нужное время

ДЗ

На данный момент у вас уже есть таблицы, которые вы создали. Ваша задача, создать файл, в котором вы опишите:

- Продумать, на какие таблицы будет чаще делаться запись, а на какие чаще чтение
- Какие запросы будут использоваться при работе вашего приложения. Привести примеры

Далее необходимо создать `indexes.sql` и в нём:

- Создать индексу
- Аргументировать каждый созданный индекс

Пример:

```
-- Хэш индекс по internal_id, так как поиск осуществляется только
-- при помощи '='
CREATE INDEX idx_student_internal_id ON students USING
hash(internal_id);
```

ВАЖНО

Перед вами не стоит задача добавить как можно больше индексов. Ваша основная задача аргументировать ваш выбор. Можете даже описать почему вы **НЕ** добавили тот или иной индекс