Партиции, триггеры, хранимки

Партиции

Снова возвращаемся к нашей модели со студентами. До этого мы хоть и накидывали различные индексы на наши таблицы, чтобы ускорить поиск, но при этом так или иначе все данные по сути лежали “в одной куче”, хотя всё и так уже логически поделено на части

- Оценки поделены на семестры
- Студенты поделены на курсы

Вспоминаем какие бывают партиции

Практика

Всё также поднимаем Postgresql в Docker и создаём таблицы из файлика create_tables.sql и генерим данные из файлика insert_data.sql. Рассуждаем каким образом мы можем разделить

Из файлика partitions.sql создаём таблицу и партиции для оценок. Вот незадача - на существующие таблицы нельзя накинуть партиции, если она не была для этого приспособлена. Обсуждаем как нам перетещить данные из одной таблицы в другую

В файлике migration.sql смотрим один из вариантов переноса данных. Также делаем SELECT в партиционированную таблицу(из того же файла), используя EXPLAIN ANALYZE и видим, что PostgreSQL сама определила нужную партицию

```
QUERY PLAN
Seq Scan on grades_2024_03 grades  (cost=0.00..31.25 rows=8 width=20) (actual time=0.011..0.013 rows=1 loops=1)
  Filter: (created_at = '2024-03-05 00:00:00+00'::timestamp with time zone)
  Rows Removed by Filter: 29
Planning Time: 0.083 ms
Execution Time: 0.038 ms
```

Также обязательно упоминаем, что мы можем напрямую ходить в нужную нам партицию, без указания даты в WHERE

Возвращаемся к migration.sql к следующему типу партиций LIST. Создаем таблицу и накидываем партиции. Обращаем внимание на `PRIMARY KEY (id, faculty)`

И далее хэш партиционирование, где мы делим все записи на две части

```
(MODULUS 2, REMAINDER 0);
(MODULUS 2, REMAINDER 1);
```

Можем немного упомянуть Cassandra

Триггеры

Рассказываем историю о том, что студенты часто приходят переписывать работы, чтобы исправить оценки. Заведующие кафедры это не запрещают, но им хотелось бы отслеживать все изменения в оценках. В файлике trigger.sql создаем триггер и пробуем изменить любую оценку в таблице gardes, после чего проверяем таблицу grade_audit. Подробно рассматриваем новую таблицу, созданную функцию и сам триггер

Хранимые процедуры

Бывает такое, что запланированные идеи так и не воплощаются в жизнь. Поэтому напишем хранимую процедуру, которая будет удалять курс, если на нём не выставили ни одной оценки. Заходим в procedure.sql, создаем хранимку и вызываем её при помощи `CALL delete_course_if_no_grades(1);`