****

**­**

**Министр науки и высшего образования Российской̆**

**Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

Рабочий протокол и отчет по лабораторной работе № 5

Партицирование в PostgreSQL

**Выполнил(а) студент группы** № **M34091**

Сидорцов Владимир Сергеевич

**Подпись:**

**Проверил:**

Санкт-Петербург

2023г.

1. **Цель работы.**

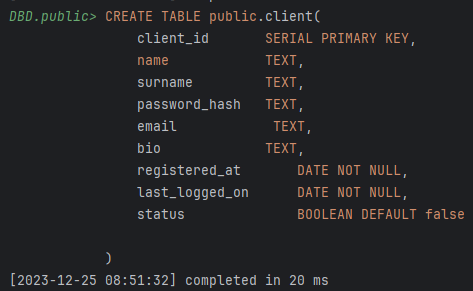
* Замаскировать поля с конфиденциальными данными.
* Провести анонимизацию данных.

1. **Задачи, решаемые при выполнении работы.**
2. **Объект исследования.**

* [Postgres partitioning](https://www.postgresql.org/docs/current/ddl-partitioning.html#DDL-PARTITIONING)

1. **Выполнение работы.**

Т.к. секционирование предназначено специально для очень больших количеств строк, я создам новую таблицу клиентов, которые обращались когда-нибудь в офис оператора:

****

Теперь заполним ее хаотичными данными с помощью SQL:



Теперь мы нуждаемся в секционировании. Мы можем полагаться на разные тактики разбиения:

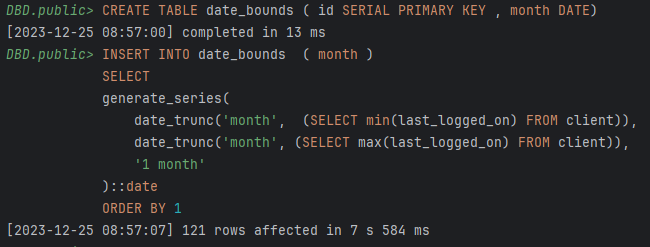
* по дате – например, выбирать партиции, основываясь на годе, в котором пользователь был создан;
* по диапазону идентификаторов – например, первый миллион пользователей, второй миллион пользователей, и так далее;

В моём случае я буду разбивать по каждому месяцу обращений.

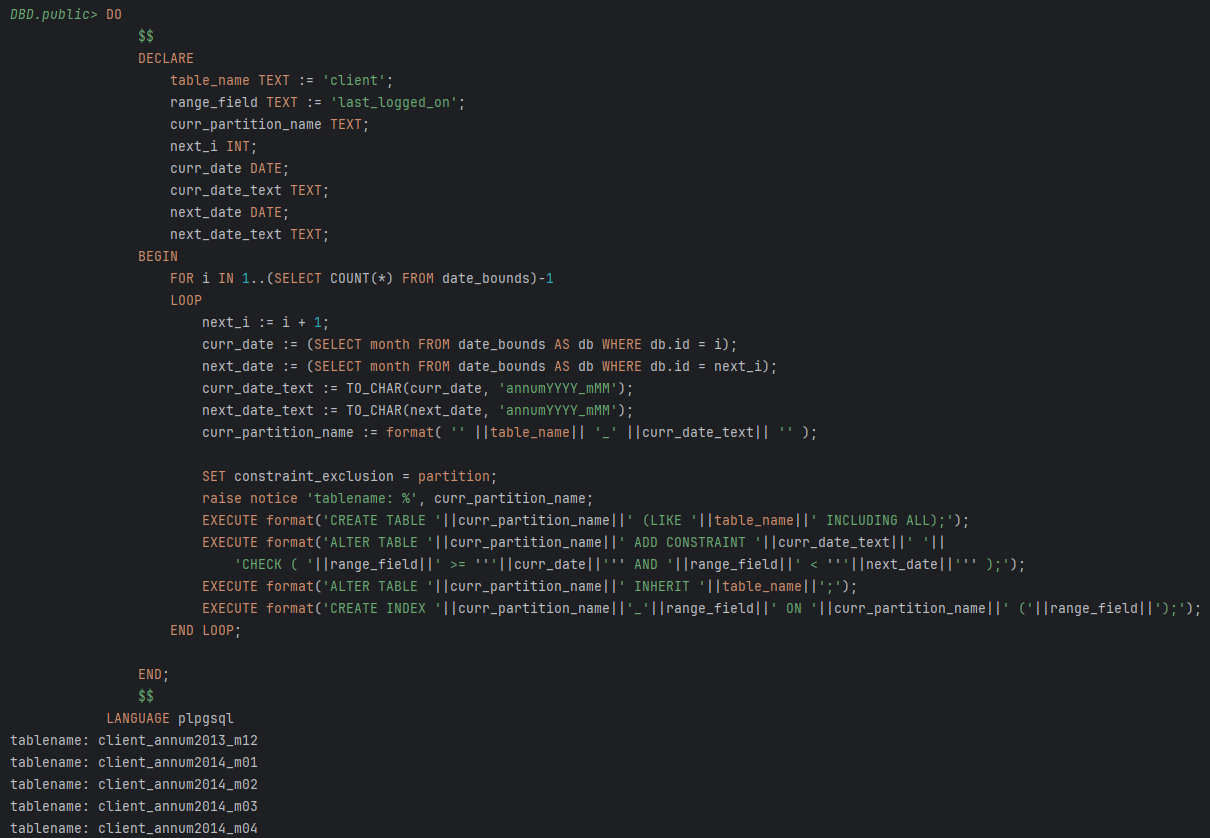
Начать стоит с того, что существует 2 подхода для создания партиции: создание таблицы оператором CREATE и «подвод» таблицы под родителя оператором ATTACH. В первом варианте мы можем удалить таблицу целиком вместо удаления блока строк, что значительно ускоряет поддержку данных. ATTACH же наоборот позволяет управлять данными более гибко, например, после DETACH у нас остаётся таблица, она уже отдельная, и мы можем ее анализировать.

Я для себя выбрал метод создания таблиц, чтобы каждый месяц иметь возможность удалить данные старого месяца и сохранить объем данных в норме.  
  
Далее мы должны будем определиться с границами наших партиций, чтобы однозначно определять их ответственность. Наложение границ = ошибка. При чем ограничения должны быть примитивными типами данных, без никаких вычислений, чисто декларативными, чтобы postgres мог корректно применить свои правила.

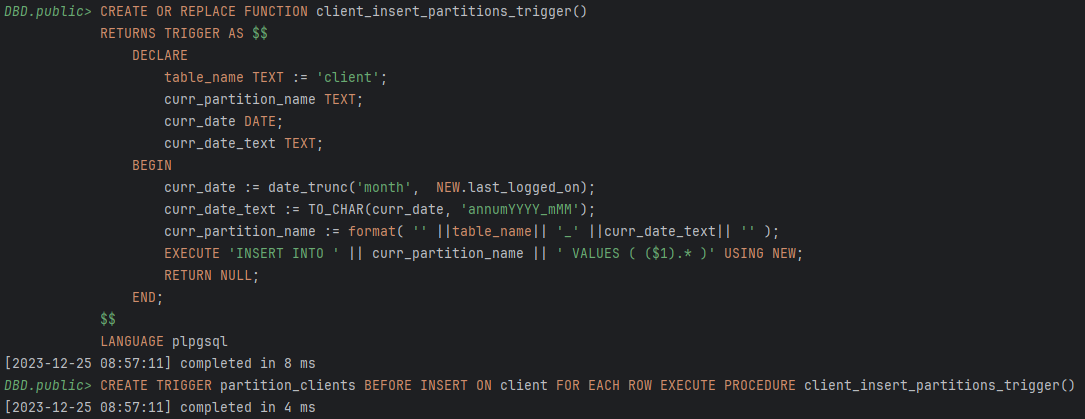
Для однозначного определения границ партиций я создал новую таблицу и заполнил ее каждым новым месяцем из сгенерированного диапазона:



Вслед за этим я пишу функцию, которая в цикле для каждой границы вычисляет имя таблицы, ее границы и создает ее. ALTER TABLE ADD CONSTRAINT создается самостоятельно чтобы сохранить силы и не держать ACCESS EXCLUSIVE lock на родительской таблице (блокирование чтения и записи).



Отлично, теперь у нас есть границы и созданные партиции.. Но как мы поймем в какую именно дочернюю таблицу класть новую строчку? Триггер!

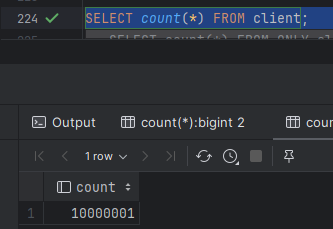


Здесь мы создали специальную функцию, которая будет всякий раз перед вставкой строки выполняться: определять границы и имя партиции по такому же точно принциу, что и раньше.

Отлично, теперь у нас есть границы и распределение новых строк! Протестируем! Вставим 1 новую строчку:



Сколько у нас строчек всего?



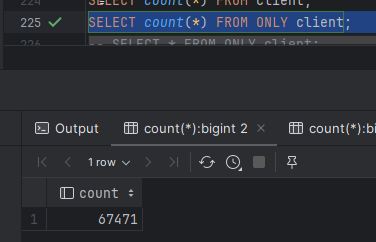
Класс!

Теперь наша задача удалить строки из родительской и переместить строки в соответствующую дочернюю таблицу:

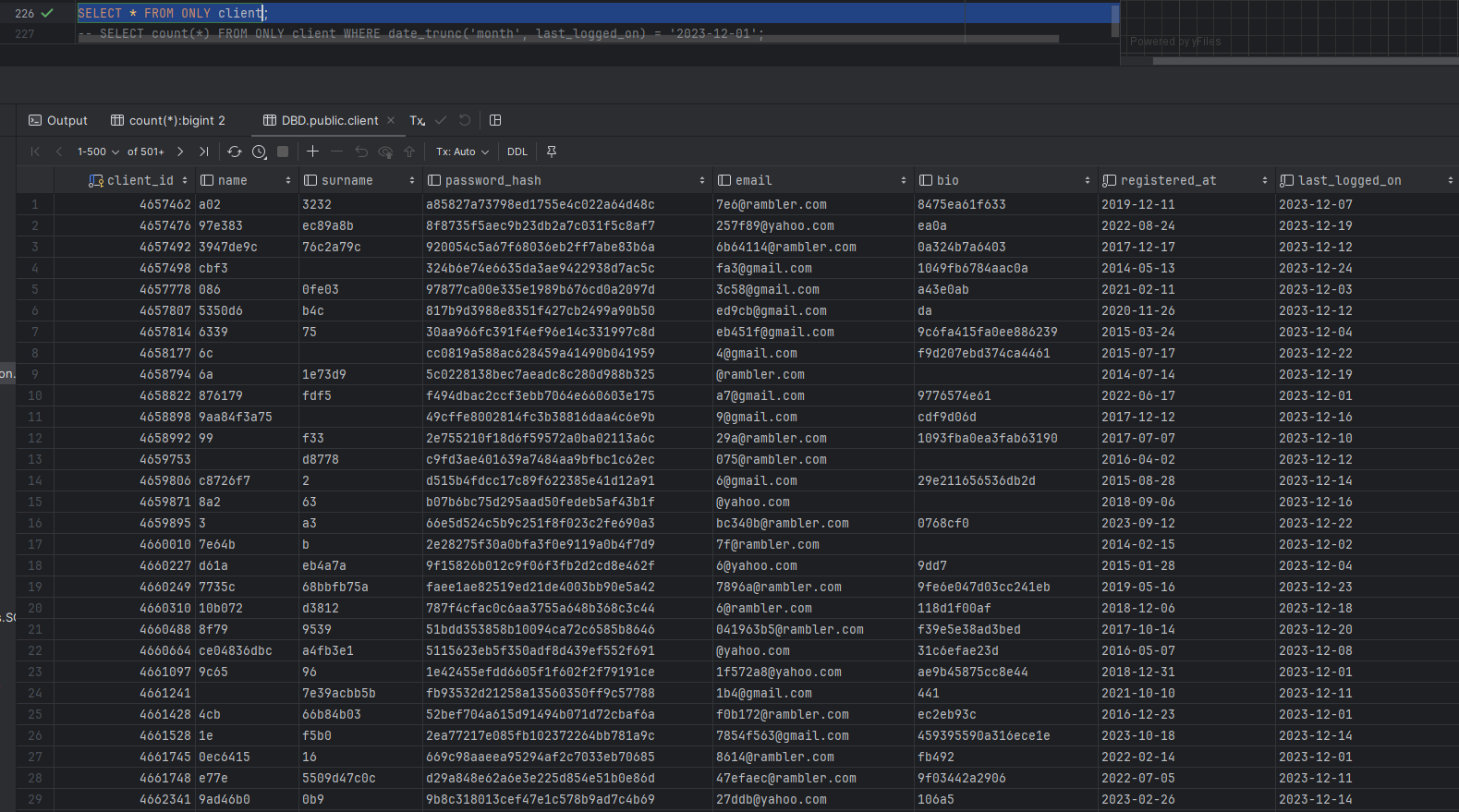


Для каждой партиции мы удаляем из главной таблицы те строчки, которые удовлетворяют именно ее границам, а после вставляем в партицию.

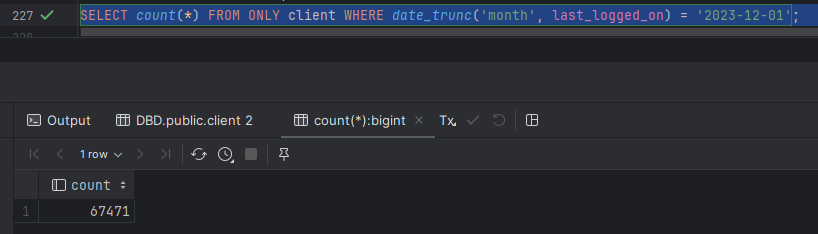
Наконец, смотрим на финальное количество строк в родителе:

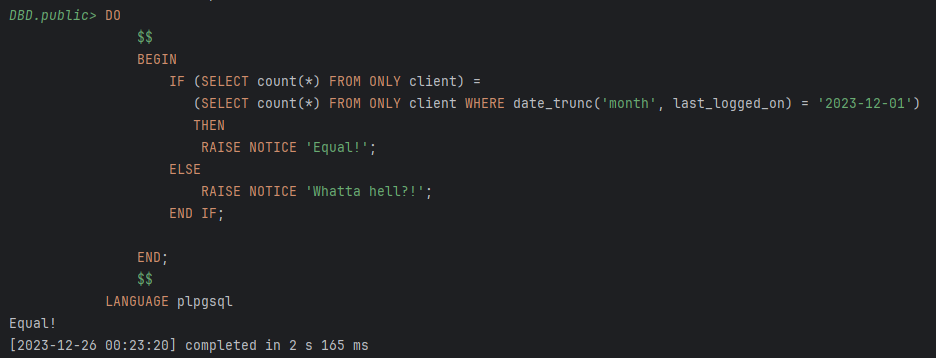


О нет! Что же не так?



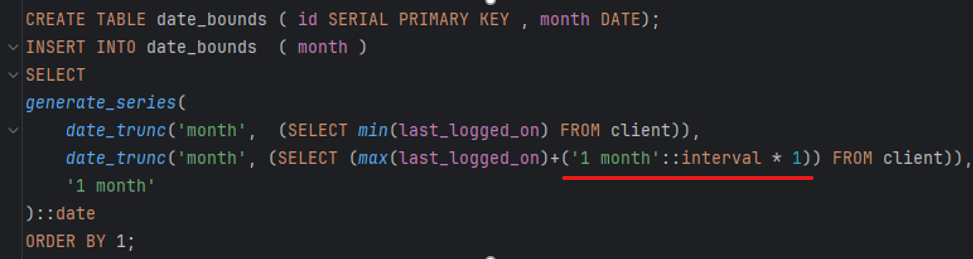
После недлительного анализа видим, что last\_logged\_on по месяцу у них один и тот же:



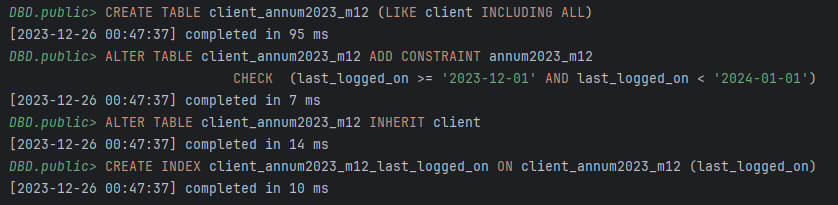




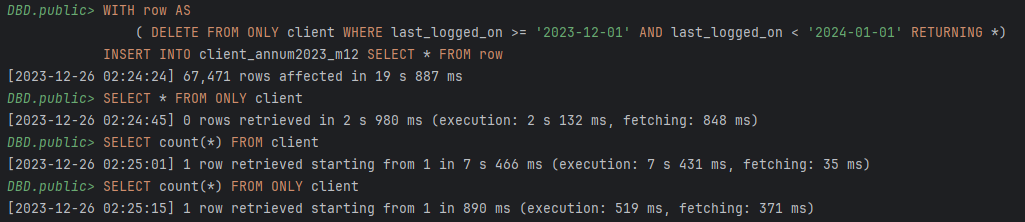
Где же ошибка? В грацах цикла создания границ партиций:

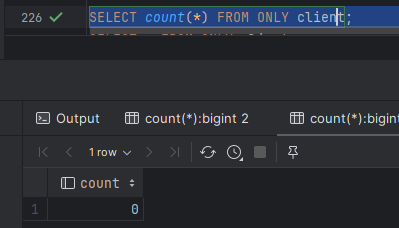


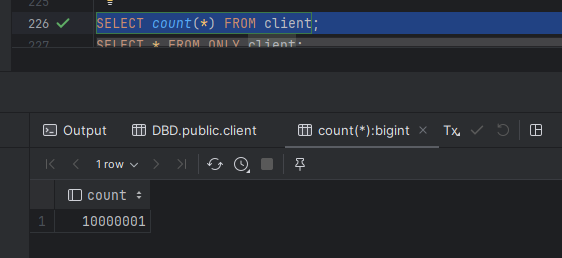
Оставшуюся партицию создаем вручную



И переносим данные тоже вручную:







1. **Выводы и анализ результатов работы.**

Научился создавать секционирование таблиц.

Научился создавать триггеры.

Научился писать функции, циклы, условия в plpgsql.

1. **Дополнительные задания.**
2. **Выполнение дополнительных заданий.**
3. **Замечания преподавателя (исправления, вызванные замечаниями преподавателя, также помещают в этот пункт)**