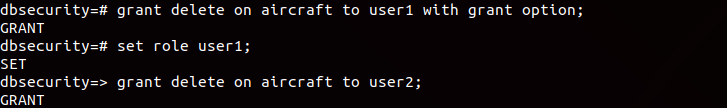
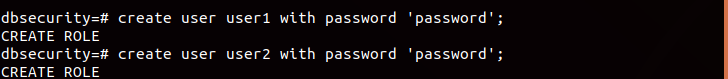
**Контроль доступа и системы аудита**

Существует множество подходов к реализации системы контроля доступа внутри базы данных. В данной практической работе описаны некоторые из них.

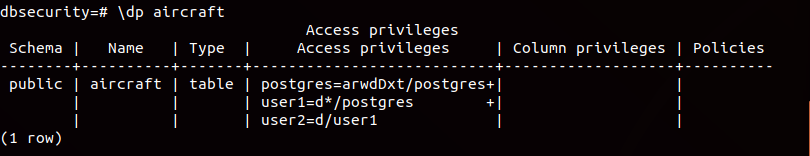
Во-первых, PostgreSQL имеет возможность устанавливать права на запросы ***select/update/delete/ insert*** с помощью команд ***grant/revoke***. Также PostgreSQL позволяет делегировать управление правами пользователям, а те могут делиться собственными правами друг с другом, используя опцию «***with grant option***».

Например, мы можем создать пользователя *user1* и дать ему право на удаление строк в таблице и возможность передать это право другому пользователю. После этого *user1* от своего имени может выдать разрешение на удаление строк другому пользователю *user2*.

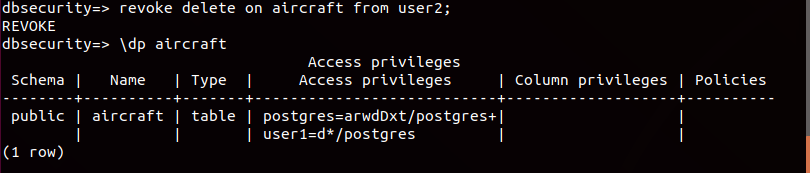




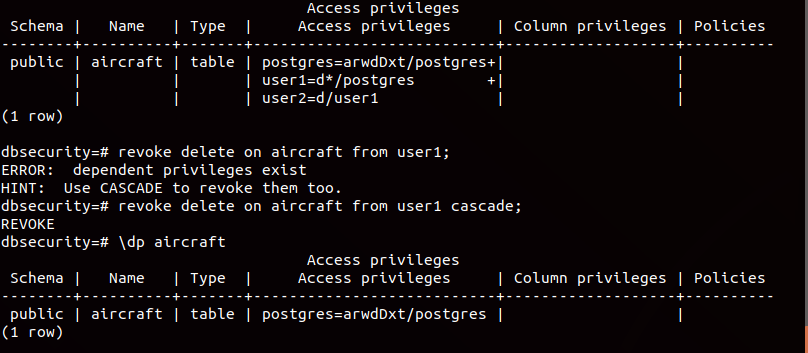
Чтобы просмотреть настройки прав, установленные для таблиц на текущий момент, надо ввести встроенную короткую команду ***\dp*.**



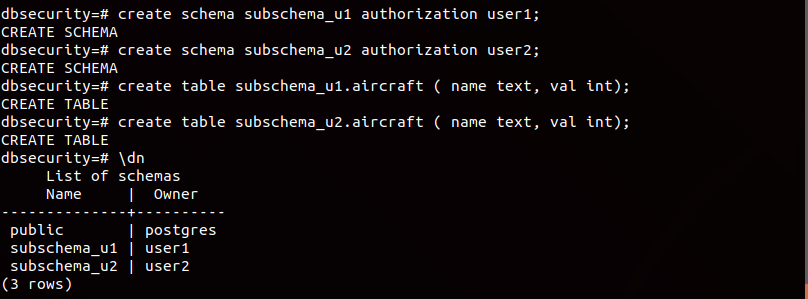
Чтобы отозвать у пользователя выданные права, нужно использовать команду ***revoke***.



Фактически, мы можем удалить выданные права со всеми случаями делегирования одной командой, применив опцию «***cascade***».

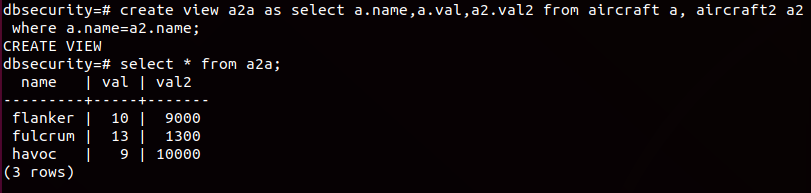


Еще одним способом управления доступом пользователей к объектам внутри базы данных являются **подсхемы**. Подсхемы позволяют выделить отдельные наборы таблиц для каждого пользователя. Последовательность команд для создания подсхем представлена ​​на рисунке ниже.

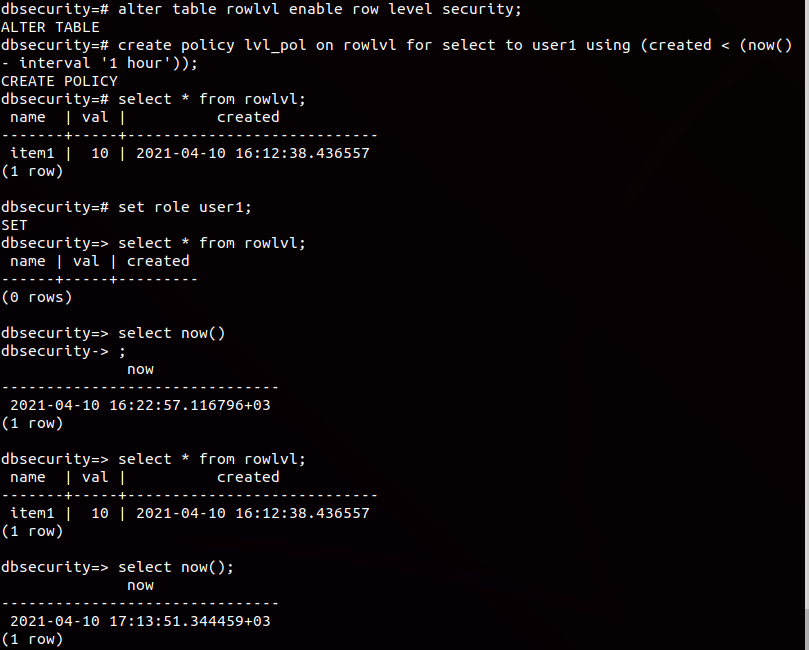


Пользователь *user1* из примера может видеть только таблицу из своей подсхемы.

Если необходимо управление доступом на уровне столбцов таблицы, то в PostgresSQL можно использовать **представления.** Пример команды для создания представления с целью управления доступом можно найти на рисунке ниже.

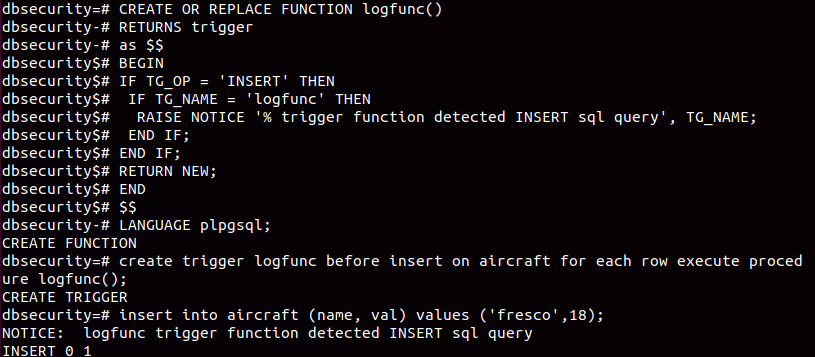


Следующим уровнем детализации при разграничении доступа к объектам внутри базы данных является управление доступом на уровне строк. Функция управления доступом активируется путем ввода команды «***enable row level security***», а также с помощью применения политики для вывода только соответствующих ей строк. Пример такой настройки представлен на рисунке ниже.



Наконец, давайте посмотрим, как мы можем реализовать систему аудита на основе встроенных функций базы данных PostgreSQL. Основная идея – запускать обработчик для каждого действия в базе данных. Хорошим способом для реализации системы аудита является использование функции триггера, которая позволяет обработать каждый оператор SQL во время его выполнения.

Нам нужно создать функцию триггера и сопоставить ее с некоторой таблицей событий, например, для оператора ***insert***. В приведенном ниже примере создается функция, которая проверяет, является ли тип запроса вставкой, и вызывает триггер, который вставляет уведомление в таблицу событий о том, какое точное значение и в какую таблицу было вставлено.



**Задачи для практической работы:**

1. Подготовьте таблицы для выполнения перечисленных ниже задач. Достаточно 2-3 таблиц для п. 1-5 ниже.

2. Выдайте права 3 пользователям. Пользователь User1 должен иметь полный доступ к таблице. User2 должен иметь право на вставку, select-запросы и обновление значений в таблицах. User3 должен иметь право на удаление строк из таблиц, а также возможность делегировать свои права любому пользователю.

3. Предоставьте право на удаление от пользователя User3 пользователю User4 и проверьте все выданные права.

4. Отмените все предоставленные выше права.

5. Создайте подсхему авторизации для User1 и User2 с различным набором таблиц (служебное слово AUTHORIZATION у команды CREATE SCHEMA).

6. Создайте представление как объединенный набор столбцов из разных таблиц. С помощью команды grant ограничьте доступ к представлению.

7. Настройте безопасность на уровне строк (RLS), политика должна быть создана на основе текущего пользователя, и протестируйте ее.

8. Создайте триггер для регистрации вставки, обновления и удаления содержимого в определенных таблицах.

**Вопросы к защите ЛР:**

1. Модели управления доступом. Способы реализации контроля доступа к БД.
2. Синтаксис по разграничению доступа к объектам с помощью команды grant. К каким объектам БД может быть ограничен доступ? Виды привилегий
3. Ролевая модель доступа. Атрибуты роли.
4. Методы аутентификации и способы хранения аутентификационных данных.
5. Примеры механизмов обеспечения целостности БД.
6. Основные виды угроз в СУБД
7. Классификация методов защиты на основе функций СУБД
8. Функции первичного ключа в реляционной модели базы данных
9. Правила и условия контроля целостности в реляционной модели базы данных