

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение  
высшего образования  
«Северный (Арктический) федеральный университет имени М.В. Ломоносова»

Высшая школа информационных технологий и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №11-12**

По дисциплине: Защита информации в системах управления базами данных

На тему: Защита хоста

Выполнил обучающийся:

Грозов Илья Владимирович

Направление подготовки / специальность:

10.03.01 Информационная безопасность

Курс: 3

Группа: 151113

Руководитель: Зубарев Александр Андреевич, ст.

преподаватель

Отметка о зачете

Руководитель

А.А. Зубарев.

Архангельск 2024

## ЗАДАНИЕ

Получить практический навык защиты сети при эксплуатации СУБД

## ХОД РАБОТЫ

### 1 УСТАНОВКА POSTGRES

Установим postgres используя команду: `sudo apt-get install postgresql postgresql-contrib`. Установка postgres отображена на рисунке 1

```
root@ZiVSYBD-LR11:/home/huguenot-lr11# sudo apt-get install postgresql postgresql-contrib
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libpq5 postgresql-13 postgresql-client-13 postgresql-client-common postgresql-common sysstat
Suggested packages:
  postgresql-doc postgresql-doc-13 libjson-perl isag
The following NEW packages will be installed:
  libpq5 postgresql-13 postgresql-client-13 postgresql-client-common postgresql-common postgresql-contrib sysstat
0 upgraded, 8 newly installed, 0 to remove and 0 not upgraded.
Need to get 17.9 MB of archives.
After this operation, 59.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://deb.debian.org/debian bullseye/main amd64 postgresql-client-common all 225+deb11u1 [89.4 kB]
Get:2 http://security.debian.org/debian-security bullseye-security/main amd64 libpq5 amd64 13.14-0+deb11u1 [182 kB]
Get:3 http://deb.debian.org/debian bullseye/main amd64 postgresql-common all 225+deb11u1 [238 kB]
Get:4 http://security.debian.org/debian-security bullseye-security/main amd64 postgresql-client-13 amd64 13.14-0+deb11u1 [1,513 kB]
Get:5 http://deb.debian.org/debian bullseye/main amd64 postgresql all 13+225+deb11u1 [64.9 kB]
Get:6 http://deb.debian.org/debian bullseye/main amd64 postgresql-contrib all 13+225+deb11u1 [64.9 kB]
Get:7 http://deb.debian.org/debian bullseye/main amd64 sysstat amd64 12.5.2-2 [603 kB]
Get:8 http://security.debian.org/debian-security bullseye-security/main amd64 postgresql-13 amd64 13.14-0+deb11u1 [15.2 MB]
Fetched 17.9 MB in 2s (10.6 MB/s)
Preconfiguring packages ...
Selecting previously unselected package libpq5:amd64.
(Reading database ... 165440 files and directories currently installed.)
Preparing to unpack .../0-libpq5_13.14-0+deb11u1_amd64.deb ...
Unpacking libpq5:amd64 (13.14-0+deb11u1) ...
Selecting previously unselected package postgresql-client-common.
Preparing to unpack .../1-postgresql-client-common_225+deb11u1_all.deb ...
Unpacking postgresql-client-common (225+deb11u1) ...
Selecting previously unselected package postgresql-client-13.
Preparing to unpack .../2-postgresql-client-13_13.14-0+deb11u1_amd64.deb ...
Unpacking postgresql-client-13 (13.14-0+deb11u1) ...
Selecting previously unselected package postgresql-common.
Preparing to unpack .../3-postgresql-common_225+deb11u1_all.deb ...
Adding 'diversion of /usr/bin/pg_config to /usr/bin/pg_config.libpq-dev by postgresql-common'
Unpacking postgresql-common (225+deb11u1) ...
Selecting previously unselected package postgresql-13.
Preparing to unpack .../4-postgresql-13_13.14-0+deb11u1_amd64.deb ...
Unpacking postgresql-13 (13.14-0+deb11u1) ...
```

Рисунок 1 – Установка postgres

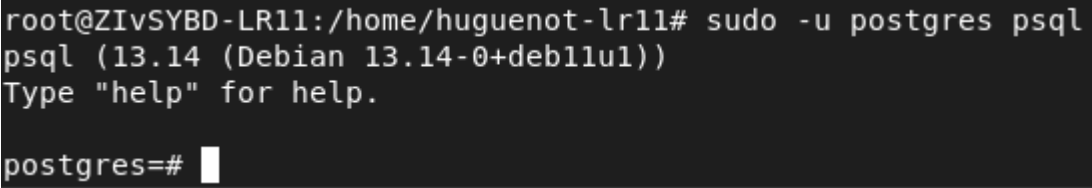
При помощи команды: `sudo systemctl status postgresql` проверим запущен ли системный процесс. Проверка запуска отображена на рисунке 2

```
root@ZiVSYBD-LR11:/home/huguenot-lr11# sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
   Active: active (exited) since Sat 2024-06-01 20:22:25 MSK; 48s ago
     Main PID: 4021 (code=exited, status=0/SUCCESS)
       Tasks: 0 (limit: 19138)
      Memory: 0B
         CPU: 0
       CGroup: /system.slice/postgresql.service

Jun 01 20:22:25 ZiVSYBD-LR11 systemd[1]: Starting PostgreSQL RDBMS...
Jun 01 20:22:25 ZiVSYBD-LR11 systemd[1]: Finished PostgreSQL RDBMS.
root@ZiVSYBD-LR11:/home/huguenot-lr11#
```

Рисунок 2 – Проверка запуска

Войдем в оболочку postgres при помощи команды: `sudo -u postgres psql`.  
Вход в оболочку postgres отображен на рисунке 3



```
root@ZIVSYBD-LR11:/home/huguenot-lr11# sudo -u postgres psql
psql (13.14 (Debian 13.14-0+deb11u1))
Type "help" for help.

postgres=#
```

Рисунок 3 – Вход в оболочку postgres

## 2 РАЗВЕРТЫВАНИЕ БАЗЫ ДАННЫХ

Загрузим архив с предложенной базой данных при помощи команды: `wget https://edu.postgrespro.ru/demo_medium.zip`. Загрузка архива с базой данных отображена на рисунке 4

```
root@ZiVSYBD-LR11:/home/huguenot-lr11# wget https://edu.postgrespro.ru/demo_medium.zip
--2024-06-01 20:33:24-- https://edu.postgrespro.ru/demo_medium.zip
Resolving edu.postgrespro.ru (edu.postgrespro.ru)... 213.171.56.196
Connecting to edu.postgrespro.ru (edu.postgrespro.ru)[213.171.56.196]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 64544914 (62M) [application/zip]
Saving to: 'demo_medium.zip'

demo_medium.zip                               100%[=====] 61.55M  35.5MB/s   in 1.7s
2024-06-01 20:33:26 (35.5 MB/s) - 'demo_medium.zip' saved [64544914/64544914]
root@ZiVSYBD-LR11:/home/huguenot-lr11#
```

Рисунок 4 – Загрузка архива с базой данных

Распакуем архив с базой данных при помощи команды: `unzip demo_medium.zip`. Распаковка архива отображена на рисунке 5

```
root@ZiVSYBD-LR11:/home/huguenot-lr11# unzip demo_medium.zip
Archive:  demo_medium.zip
  inflating: demo_medium-20170815.sql
root@ZiVSYBD-LR11:/home/huguenot-lr11#
```

Рисунок 5 – Распаковка архива

Переместим распакованную базу данных в более удобное место при помощи команды `mv`. Перемещение базы данных отображено на рисунке 6

```
huguenot-lr11@ZiVSYBD-LR11:~$ su
Password:
root@ZiVSYBD-LR11:/home/huguenot-lr11# mv demo_medium-20170815.sql /home/
root@ZiVSYBD-LR11:/home/huguenot-lr11#
```

Рисунок 6 – Перемещение базы данных

При помощи команды `sudo -u postgres psql` выполним вход в оболочку и запустим скрипт для базы данных при помощи команды: `\i /home/demo_medium-20170815.sql`. Запуск скрипта для базы данных отображен на рисунке 7

```

root@ZiVSYBD-LR11:/home/huguenot-lr11# sudo -u postgres psql
psql (13.14 (Debian 13.14-0+deb11u1))
Type "help" for help.

postgres=# CREATE USER huguenot WITH PASSWORD 'admin';
CREATE ROLE
postgres=# \i /home/demo-medium-20170815.sql
SET
SET
SET
SET
SET
SET
SET
SET
SET
psql:/home/demo-medium-20170815.sql:17: ERROR:  database "demo" does not exist
CREATE DATABASE
You are now connected to database "demo" as user "postgres".
SET
SET
SET
SET
SET
SET
SET
SET
SET
SET
CREATE SCHEMA
COMMENT
CREATE EXTENSION
COMMENT
SET
CREATE FUNCTION
CREATE FUNCTION
COMMENT
SET
SET
CREATE TABLE
COMMENT
COMMENT
COMMENT
COMMENT

```

Рисунок 7 – Запуск скрипта для базы данных

База данных было автоматически восстановлена из sql скрипта. Восстановленная база данных отображена на рисунке 8

```

ALTER TABLE
ALTER TABLE
ALTER DATABASE
ALTER DATABASE
demo=#


```

Рисунок 8 – Восстановленная база данных из sql скрипта

### 3 РАЗГРАНИЧЕНИЕ ПРАВ ПОЛЬЗОВАТЕЛЕЙ

#### 3.1 Действия перед разграничением

Перед созданием новых пользователей и разграничением их прав, выполним просмотр списка всех существующих таблиц. Пользователи с правами будут созданы на их основе. Просмотр списка существующих таблиц выполняется при помощи команды \dt в оболочке postgres базы данных demo. Существующие таблицы отображены на рисунке 9



```
demo=# \dt
```

List of relations			
Schema	Name	Type	Owner
bookings	aircrafts_data	table	postgres
bookings	airports_data	table	postgres
bookings	boarding_passes	table	postgres
bookings	bookings	table	postgres
bookings	flights	table	postgres
bookings	seats	table	postgres
bookings	ticket_flights	table	postgres
bookings	tickets	table	postgres

(8 rows)

```
demo=#
```

Рисунок 9 – Существующие таблицы

В развернутой базе данных demo существует 8 таблиц:

- aircrafts\_data;
- airports\_data;
- boarding\_passes;
- bookings;
- flights;
- seats;
- ticket\_flights;
- tickets

Для существующих таблиц возможно минимально создать 4 пользователя:

- администратор – администратор базы данных, имеет доступ ко всем таблицам;
- диспетчер – имеет доступ к таблицам, необходимым для организации воздушного пространства;
- контроллер – имеет доступ к таблицам, необходимым для производства посадки на борт;
- продавец – имеет доступ к таблицам, необходимым для получения информации о самолетах, маршрутах, бронировании и продажи билетов

На основе приведенных данных составим таблицу для разграничения прав пользователей

Таблица 1 – Разграничение прав пользователей

Таблицы БД		Пользователи БД			
		administrator	dispatcher	controller	salesman
1	aircrafts_data				
2	airports_data				
3	boarding_passes				
4	bookings				
5	flights				
6	seats				
7	ticket_flights				
8	tickets				

### 3.2 Создание пользователей и разграничение прав

Создадим пользователя administrator используя составленную таблицу 1. Выдадим пользователю administrator необходимые права для доступа к таблицам базы данных при помощи команд, отраженных в листинге 1

Листинг 1 – Команды для создания пользователя administrative и выдача прав

```
CREATE USER administrator WITH PASSWORD 'administrator';
```



```

GRANT INSERT, UPDATE, DELETE ON aircrafts_data TO administrator;
GRANT INSERT, UPDATE, DELETE ON airports_data TO administrator;
GRANT INSERT, UPDATE, DELETE ON boarding_passes TO administrator;
GRANT INSERT, UPDATE, DELETE ON bookings TO administrator;
GRANT INSERT, UPDATE, DELETE ON flights TO administrator;
GRANT INSERT, UPDATE, DELETE ON seats TO administrator;
GRANT INSERT, UPDATE, DELETE ON ticket_flights TO administrator;
GRANT INSERT, UPDATE, DELETE ON tickets TO administrator;

```

Создание пользователя administrator и выдача прав к таблицам отображена на рисунке 10

```

demo=# CREATE USER administrator WITH PASSWORD 'administrator';
CREATE ROLE
demo=# GRANT INSERT, UPDATE, DELETE ON aircrafts_data TO administrator;
GRANT
demo=# GRANT INSERT, UPDATE, DELETE ON airports_data TO administrator;
GRANT
demo=# GRANT INSERT, UPDATE, DELETE ON boards_passes TO administrator;
ERROR:  relation "boards_passes" does not exist
demo=# GRANT INSERT, UPDATE, DELETE ON boarding_passes TO administrator;
GRANT
demo=# GRANT INSERT, UPDATE, DELETE ON bookings TO administrator;
GRANT
demo=# GRANT INSERT, UPDATE, DELETE ON flights TO administrator;
GRANT
demo=# GRANT INSERT, UPDATE, DELETE ON seats TO administrator;
GRANT
demo=# GRANT INSERT, UPDATE, DELETE ON ticket_flights TO administrator;
GRANT
demo=# GRANT INSERT, UPDATE, DELETE ON tickets TO administrator;
GRANT
demo=# █

```

Рисунок 10 – Создание пользователя administrator и выдача прав к таблицам

Создадим пользователя dispatcher используя составленную таблицу 1. Выдадим пользователю dispatcher необходимые права для доступа к таблицам базы данных при помощи команд, отраженных в листинге 2

Листинг 2 – Команды для создания пользователя dispatcher и выдача прав

```

CREATE USER dispatcher WITH PASSWORD 'dispathcer';
GRANT SELECT, INSERT, UPDATE, DELETE ON aircrafts_data TO dispatcher;
GRANT SELECT, INSERT, UPDATE, DELETE ON airports_data TO dispatcher;
GRANT SELECT, INSERT, UPDATE, DELETE ON boarding_passes TO dispatcher;

```

Создание пользователя dispatcher и выдача прав к таблицам отображена на рисунке 11

```
demo=# CREATE USER dispatcher WITH PASSWORD 'dispathcer';
CREATE ROLE
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON aircrafts_data TO dispatcher;
GRANT
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON airports_data TO dispatcher;
GRANT
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON boards_passes TO dispatcher;
ERROR:  relation "boards_passes" does not exist
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON boarding_passes TO dispatcher;
GRANT
demo=#
```

Рисунок 11 – Создание пользователя dispatcher и выдача прав к таблицам

Создадим пользователя controller используя составленную таблицу 1. Выдадим пользователю controller необходимые права для доступа к таблицам базы данных при помощи команд, отраженных в листинге 3

Листинг 3 – Команды для создания пользователя controller и выдача прав

```
CREATE USER controller WITH PASSWORD 'controller';
GRANT SELECT, INSERT, UPDATE, DELETE ON boards_passes TO controller;
GRANT SELECT, INSERT, UPDATE, DELETE ON flights TO controller;
GRANT SELECT, INSERT, UPDATE, DELETE ON seats TO controller;
GRANT SELECT, INSERT, UPDATE, DELETE ON ticket_flights TO controller;
GRANT SELECT ON aircrafts_data TO controller;
GRANT SELECT ON airports_data TO controller;
```

Создание пользователя controller и выдача прав к таблицам отображена на рисунке 12

```
demo=# CREATE USER controller WITH PASSWORD 'controller';
CREATE ROLE
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON boarding_passes TO controller;
GRANT
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON flights TO controller;
GRANT
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON seats TO controller;
GRANT
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON ticket_flights TO controller;
GRANT
demo=# GRANT SELECT ON aircrafts_data TO controller;
GRANT
demo=# GRANT SELECT ON airports_data TO controller;
```

Рисунок 12 – Создание пользователя controller и выдача прав к таблицам

Создадим пользователя `salesman` используя составленную таблицу 1. Выдадим пользователю `salesman` необходимые права для доступа к таблицам базы данных при помощи команд, отраженных в листинге 4

#### Листинг 4 – Команды для создания пользователя `salesman` и выдача прав

```
CREATE USER salesman WITH PASSWORD 'salesman';
GRANT SELECT, INSERT, UPDATE, DELETE ON boards_passes TO salesman;
GRANT SELECT, INSERT, UPDATE, DELETE ON bookings TO salesman;
GRANT SELECT, INSERT, UPDATE, DELETE ON flights TO salesman;
GRANT SELECT, INSERT, UPDATE, DELETE ON seats TO salesman;
GRANT SELECT, INSERT, UPDATE, DELETE ON ticket_flights TO salesman;
GRANT SELECT, INSERT, UPDATE, DELETE ON tickets TO salesman;
```

Создание пользователя `salesman` и выдача прав к таблицам отображена на рисунке 13

```
demo=# CREATE USER salesman WITH PASSWORD 'salesman';
CREATE ROLE
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON boarding_passes TO salesman;
GRANT
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON bookings TO salesman;
GRANT
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON flights TO salesman;
GRANT
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON seats TO salesman;
GRANT
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON ticket_flights TO salesman;
GRANT
demo=# GRANT SELECT, INSERT, UPDATE, DELETE ON tickets TO salesman;
GRANT
demo=# █
```

#### Рисунок 13 – Создание пользователя `salesman` и выдача прав к таблицам

Нам требуется также выдать права на доступ к схемам созданным пользователям при помощи команд отраженных в листинге 5

#### Листинг 5 – Команды для выдачи доступа пользователям к схемам

```
GRANT USAGE ON SCHEMA bookings TO salesman;
GRANT USAGE ON SCHEMA bookings TO controller;
GRANT USAGE ON SCHEMA bookings TO dispatcher;
GRANT USAGE ON SCHEMA bookings TO administrator;
```

Выдача доступа к схемам отображена на рисунке 14

```
demo=# GRANT USAGE ON SCHEMA bookings TO salesman;
GRANT
demo=# GRANT USAGE ON SCHEMA bookings TO controller;
GRANT
demo=# GRANT USAGE ON SCHEMA bookings TO dispatcher;
GRANT
demo=# GRANT USAGE ON SCHEMA bookings TO administrator;
GRANT
demo=#
```

Рисунок 14 – Выдача доступа к схемам

### 3.3 Проверка разграничения прав пользователей

Для возможности авторизации под другими пользователями нам необходимо сконфигурировать файл `pg_hba.conf` открыв его при помощи команды: `sudo nano /etc/postgresql/13/main/pg_hba.conf`. Открытие файла конфигурации отображено на рисунке 15

```
root@ZIVSYBD-LR11:~# sudo nano /etc/postgresql/13/main/pg_hba.conf
```

Рисунок 15 – Открытие файла конфигурации

Изменим все существующие значения `peer` на `md5` кроме пользователя `postgres`, на приведенном рисунке это значение было изменено ошибочно. Настройки авторизации для пользователей отображены на рисунке 16

```
# Database administrative login by Unix domain socket
local    all             postgres              md5

# TYPE  DATABASE  USER  ADDRESS  METHOD

# "local" is for Unix domain socket connections only
local    all             all                  md5
# IPv4 local connections:
host     all             all              127.0.0.1/32      md5
# IPv6 local connections:
host     all             all              ::1/128           md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local    replication     all                  md5
host     replication     all              127.0.0.1/32      md5
host     replication     all              ::1/128           md5
```

Рисунок 16 – Настройки авторизации для пользователей

Выполним перезагрузку сервиса postgres при помощи команды `sudo service postgresql restart`. Выполнение перезагрузки сервиса postgres отображено на рисунке 17

```
root@ZIVSYBD-LR11:~# sudo service postgresql restart
```

Рисунок 17 – Выполнение перезагрузки сервиса postgres

### 3.3.1 Проверка пользователя salesman

Для проверки настроенных прав пользователя salesman выполним авторизацию под ним и отобразим список таблиц при помощи команды `\dt`. Авторизация под пользователем salesman и отображение списка таблиц отображено на рисунке 18

```
root@ZIVSYBD-LR11:/home/huguenot-lr11# psql -U salesman -d demo
Password for user salesman:
psql (13.14 (Debian 13.14-0+deb11u1))
Type "help" for help.

demo=> \dt
          List of relations
  Schema |      Name      | Type  | Owner
  ---+---+---+---
 bookings | aircrafts_data | table | postgres
 bookings | airports_data  | table | postgres
 bookings | boarding_passes | table | postgres
 bookings | bookings       | table | postgres
 bookings | flights        | table | postgres
 bookings | seats          | table | postgres
 bookings | ticket_flights | table | postgres
 bookings | tickets         | table | postgres
(8 rows)

demo=> █
```

Рисунок 18 – Авторизация под пользователем salesman и отображение списка таблиц

Из таблицы 1 мы знаем, что у нас есть доступ к таблице tickets. Выполним запрос в данной таблице для извлечения первых 10 значений при помощи команды: `SELECT * FROM tickets LIMIT 10`; Выполнение запроса к таблице tickets пользователя salesman отображено на рисунке 19

ticket_no	book_ref	passenger_id	passenger_name	contact_data
0005432000860	8BBCD2	4750 122452	VLADIMIR FROLOV	{"phone": "+70125366530"}
0005432000861	DA7166	3889 683019	NINA BELOVA	{"email": "belovanina11041976@postgrespro.ru", "phone": "+70048667971"}
0005432000862	DA7166	3554 024596	KIRA SIDOROVA	{"email": "sidorova.kira_101971@postgrespro.ru", "phone": "+70398785493"}
0005432000863	78E2D2	2836 125969	GENNADIY NIKITIN	{"email": "nikitin_g_111965@postgrespro.ru", "phone": "+70556069198"}
0005432000864	B8F5BC	1665 656774	FEDOR SHEVCHENKO	{"email": "shevchenkofedor-1963@postgrespro.ru", "phone": "+70644329898"}
0005432000865	B8F5BC	6427 408050	DAMIR TIMOFEEV	{"phone": "+70125956007"}
0005432000866	BD402C	6243 166891	EVGENIY MAKAROV	{"email": "makarov-e011968@postgrespro.ru", "phone": "+70390705622"}
0005432000867	9328AF	5496 753314	ALFIYA FROLOVA	{"email": "alfiya-frolova1963@postgrespro.ru", "phone": "+70224059780"}
0005432000868	9328AF	7886 931683	NADEZHDA KUZMINA	{"email": "kuzmina-nadezhda.121975@postgrespro.ru", "phone": "+70308198082"}
0005432000869	9328AF	5164 327476	TATYANA ANDREEVA	{"phone": "+70875110463"}

(10 rows)

Рисунок 19 – Выполнение запроса к таблице tickets пользователя salesman

Доступ есть, данные были извлечены. Из таблицы 1 мы знаем, что для пользователя salesman отсутствует доступ к таблице airports\_data. Выполним запрос в данной таблице для извлечения первых 10 значений при помощи команды: `SELECT * FROM airports_data LIMIT 10`; чтобы убедиться в том, что у данного пользователя нет прав на просмотр. Выполнение запроса к таблице airports\_data пользователя salesman отображено на рисунке 20

```
demo=> SELECT * FROM airports_data LIMIT 10;
ERROR:  permission denied for table airports_data
demo=>
```

Рисунок 20 – Выполнение запроса к таблице airports\_data пользователя salesman

### 3.3.2 Проверка пользователя controller

Для проверки настроенных прав пользователя controller выполним авторизацию под ним и отобразим список таблиц при помощи команды `\dt`. Авторизация под пользователем controller и отображение списка таблиц отображено на рисунке 21

```

root@ZIVSYBD-LR11:/home/huguenot-lr11# psql -U controller -d demo
Password for user controller:
psql (13.14 (Debian 13.14-0+deb11u1))
Type "help" for help.

demo=> \dt

```

List of relations			
Schema	Name	Type	Owner
bookings	aircrafts_data	table	postgres
bookings	airports_data	table	postgres
bookings	boarding_passes	table	postgres
bookings	bookings	table	postgres
bookings	flights	table	postgres
bookings	seats	table	postgres
bookings	ticket_flights	table	postgres
bookings	tickets	table	postgres

```

(8 rows)

demo=>

```

Рисунок 21 – Авторизация под пользователем controller и отображение списка таблиц

Из таблицы 1 мы знаем, что у нас есть доступ к таблице seats. Выполним запрос в данной таблице для извлечения первых 10 значений при помощи команды: `SELECT * FROM seats LIMIT 10;` Выполнение запроса к таблице seats пользователя controller отображено на рисунке 22

```

demo=> SELECT * FROM seats LIMIT 10;

```

aircraft_code	seat_no	fare_conditions
319	2A	Business
319	2C	Business
319	2D	Business
319	2F	Business
319	3A	Business
319	3C	Business
319	3D	Business
319	3F	Business
319	4A	Business
319	4C	Business

Рисунок 22 – Выполнение запроса к таблице seats пользователя controller

Доступ есть, данные были извлечены. Из таблицы 1 мы знаем, что для пользователя controller отсутствует доступ к таблице bookings. Выполним запрос в данной таблице для извлечения первых 10 значений при помощи команды: `SELECT * FROM bookings LIMIT 10;` чтобы убедиться в том, что у данного пользователя нет прав на просмотр. Выполнение запроса к таблице bookings пользователя controller отображено на рисунке 23

```
demo=> SELECT * FROM bookings LIMIT 10;
ERROR:  permission denied for table bookings
demo=> █
```

Рисунок 23 – Выполнение запроса к таблице bookings пользователя controller

### 3.3.3 Проверка пользователя dispatcher

Для проверки настроенных прав пользователя dispatcher выполним авторизацию под ним и отобразим список таблиц при помощи команды `\dt`. Авторизация под пользователем dispatcher и отображение списка таблиц отображено на рисунке 24

```
root@ZiVSYBD-LR11:/home/huguenot-lr11# psql -U dispatcher -d demo
Password for user dispatcher:
psql (13.14 (Debian 13.14-0+deb11u1))
Type "help" for help.

demo=> \dt
          List of relations
  Schema |      Name      | Type  | Owner
-----+-----+-----+-----
 bookings | aircrafts_data | table | postgres
 bookings | airports_data  | table | postgres
 bookings | boarding_passes | table | postgres
 bookings | bookings       | table | postgres
 bookings | flights        | table | postgres
 bookings | seats          | table | postgres
 bookings | ticket_flights | table | postgres
 bookings | tickets        | table | postgres
(8 rows)

demo=> █
```



Рисунок 24 – Авторизация под пользователем dispatcher и отображение списка таблиц

Из таблицы 1 мы знаем, что у нас есть доступ к таблице airports\_data. Выполним запрос в данной таблице для извлечения первых 10 значений при помощи команды: `SELECT * FROM airports_data LIMIT 10;` Выполнение запроса к таблице airports\_data пользователя dispatcher отображено на рисунке 25

```
demo=> SELECT * FROM airports_data LIMIT 10;
```

airport_code	airport_name	city	coordinates	timezone
YKS	{ "en": "Yakutsk Airport", "ru": "Якутск" }	{ "en": "Yakutsk", "ru": "Якутск" }	(129.77099609375, 62.093299865722656)	Asia/Yakutsk
MJZ	{ "en": "Mirny Airport", "ru": "Мирный" }	{ "en": "Mirnyj", "ru": "Мирный" }	(114.03908146484375, 62.534698486328125)	Asia/Yakutsk
KHV	{ "en": "Khabarovsk Novy Airport", "ru": "Хабаровск-Новый" }	{ "en": "Khabarovsk", "ru": "Хабаровск" }	(135.18800354004, 48.52799987793)	Asia/Vladivostok
PXS	{ "en": "Yelizovo Airport", "ru": "Елизово" }	{ "en": "Petropavlovsk", "ru": "Петропавловск-Камчатский" }	(158.45399475097656, 53.16780005544922)	Asia/Kamchatka
UUS	{ "en": "Yuzhno-Sakhalinsk Airport", "ru": "Хомутово" }	{ "en": "Yuzhno-Sakhalinsk", "ru": "Южно-Сахалинск" }	(142.71800231933594, 46.88869857788086)	Asia/Sakhalin
VVO	{ "en": "Vladivostok International Airport", "ru": "Владивосток" }	{ "en": "Vladivostok", "ru": "Владивосток" }	(132.1479949951172, 43.39899826049885)	Asia/Vladivostok
LED	{ "en": "Pulkovo Airport", "ru": "Пулково" }	{ "en": "St. Petersburg", "ru": "Санкт-Петербург" }	(30.262500762930453, 59.80630059814453)	Europe/Moscow
KGD	{ "en": "Khrabrovo Airport", "ru": "Храброво" }	{ "en": "Kaliningrad", "ru": "Калининград" }	(20.392599860774414, 54.8899938964044)	Europe/Kaliningrad
KEJ	{ "en": "Kemerovo Airport", "ru": "Кемерово" }	{ "en": "Kemerovo", "ru": "Кемерово" }	(86.1072006225586, 55.27009963989258)	Asia/Novokuznetsk
CEK	{ "en": "Chelyabinsk Balandino Airport", "ru": "Челябинск" }	{ "en": "Chelyabinsk", "ru": "Челябинск" }	(61.5033, 55.305801)	Asia/Yekaterinburg

```
(10 rows)
```

Рисунок 25 – Выполнение запроса к таблице airports\_data пользователя dispatcher

Доступ есть, данные были извлечены. Из таблицы 1 мы знаем, что для пользователя dispatcher отсутствует доступ к таблице tickets\_flights. Выполним запрос в данной таблице для извлечения первых 10 значений при помощи команды: `SELECT * FROM tickets_flights LIMIT 10;` чтобы убедиться в том, что у данного пользователя нет прав на просмотр. Выполнение запроса к таблице tickets\_flights пользователя dispatcher отображено на рисунке 26

```
demo=> SELECT * FROM ticket_flights LIMIT 10;
ERROR: permission denied for table ticket_flights
demo=> █
```

Рисунок 26 – Выполнение запроса к таблице tickets\_flights пользователя dispatcher

### 3.3.3 Проверка пользователя administrator

Для проверки настроенных прав пользователя administrator выполним авторизацию под ним и отобразим список таблиц при помощи команды \dt. Авторизация под пользователем administrator и отображение списка таблиц отображено на рисунке 27

```
Password for user administrator:
psql (13.14 (Debian 13.14-0+deb11u1))
Type "help" for help.

demo=> \dt
          List of relations
Schema |      Name      | Type  | Owner
-----+-----+-----+-----
bookings | aircrafts_data | table | postgres
bookings | airports_data  | table | postgres
bookings | boarding_passes | table | postgres
bookings | bookings       | table | postgres
bookings | flights        | table | postgres
bookings | seats          | table | postgres
bookings | ticket_flights | table | postgres
bookings | tickets        | table | postgres
(8 rows)

demo=>
```

Рисунок 27 – Авторизация под пользователем administrator и отображение списка таблиц

Из таблицы 1 мы знаем, что у нас есть доступ ко всем таблицам. Разграничение прав доступа не выполнялось, но функция SELECT была упущена и не добавлена пользователю administrator

## 4 НАСТРОЙКА РЕЗЕРВНОГО КОПИРОВАНИЯ

Для настройки резервного копирования создадим папку backup\_demo при помощи команды: `mkdir backup_demo`. Создание папки для резервного копирования отображено на рисунке 28

```
root@ZivSYBD-LR11:/home# mkdir backup_demo
root@ZivSYBD-LR11:/home#
```

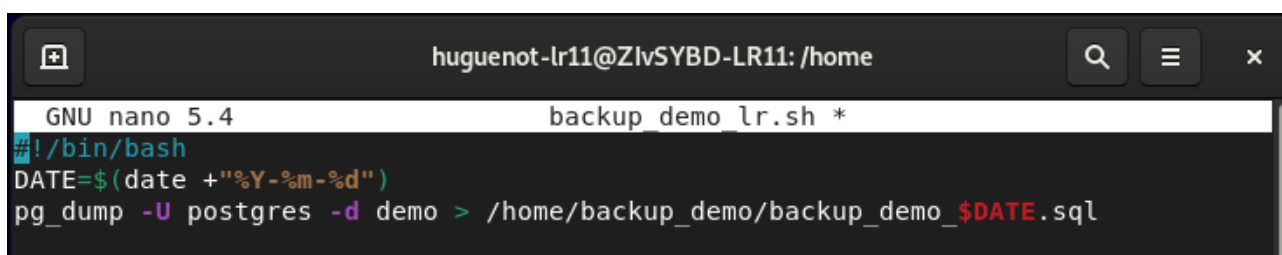
Рисунок 28 – Создание папки для резервного копирования

При помощи команды: `touch backup_demo_lr.sh` создадим файл будущего скрипта для выполнения резервного копирования и откроем его при помощи команды `sudo nano backup_demo_lr.sh`. Создание файла резервного копирования и открытие скрипта для редактирования отображено на рисунке 29

```
root@ZivSYBD-LR11:/home# touch backup_demo_lr.sh
root@ZivSYBD-LR11:/home# sudo nano backup_demo_lr.sh
root@ZivSYBD-LR11:/home#
```

Рисунок 29 – Создание файла резервного копирования и открытие скрипта для редактирования

Напишем скрипт для резервного копирования. Написанный скрипт для резервного копирования отображен на рисунке 30



```
huguenot-lr11@ZivSYBD-LR11: /home
GNU nano 5.4 backup_demo_lr.sh *
#!/bin/bash
DATE=$(date +%Y-%m-%d)
pg_dump -U postgres -d demo > /home/backup_demo/backup_demo_$(date +%Y-%m-%d).sql
```

Рисунок 30 - Написанный скрипт для резервного копирования

Проверим написанный скрипт для резервного копирования запустив его при помощи команды: `./backup_demo_lr.sh`. Запуск скрипта резервного копирования отображен на рисунке 31

```
root@ZiVSYBD-LR11:/home# ./backup_demo_lr.sh
```

Рисунок 31 – Запуск скрипта резервного копирования

Во избежание ошибки отображенной на рисунке 32 необходимо изменить метод аутентификации для пользователя. Ошибка аутентификации пользователя отображена на рисунке 32

```
root@ZiVSYBD-LR11:/home# ./backup_demo_lr.sh
pg_dump: error: connection to database "demo" failed: FATAL: Peer authentication failed for user "postgres"
```

Рисунок 32 – Ошибка аутентификации пользователя

После изменения метода аутентификации пользователя в указанной папке будет располагаться файл базы данных после резервного копирования. Файл базы данных после резервного копирования отображен на рисунке 33

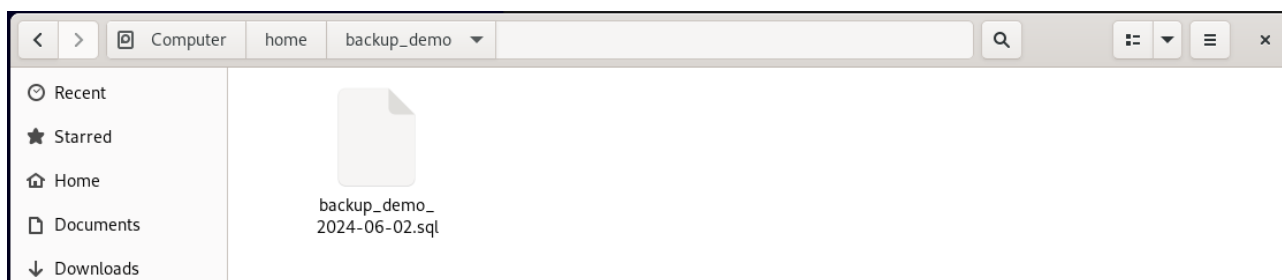


Рисунок 33 - Файл базы данных после резервного копирования

Добавим скрипт резервного копирования базы данных для автоматического выполнения в crontab. Откроем crontab при помощи команды: crontab -e. Открытие crontab отображено на рисунке 34

```
root@ZiVSYBD-LR11:/home# crontab -e
```

Рисунок 34 – Открытие crontab

Отредактируем crontab внеся строку следующего содержания: 0 0 \* \* \* /home/backup\_demo\_lr.sh. Данные изменения позволят автоматически выполнять резервное копирование базы данных каждый день в 00:00. Редактирование crontab отображено на рисунке 35

```
GNU nano 5.4
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow  command
0 0 * * * /home/backup_demo_lr.sh
```

Рисунок 35 – Редактирование crontab

## 5 ХЭШИРОВАНИЕ ХРАНЯЩИХСЯ ДАННЫХ ПО SHA-1

Выполним хэширование хранящихся данных по sha-1. Изменим выбранную таблицу airports\_data добавив в нее столбец sha1\_hashed при помощи команды: ALTER TABLE airports\_data ADD COLUMN sha1\_hashed TEXT; Изменение таблицы airports\_data отображено на рисунке 36

```
demo=# ALTER TABLE airports_data ADD COLUMN sha1_hashed TEXT;  
ALTER TABLE
```

Рисунок 36 – Изменение таблицы airports\_data

Для выполнения хэширование хранящихся данных по sha-1 нам необходимо добавить расширение pgcrypto при помощи команды: CREATE EXTENSION pgcrypto; Добавление расширения pgcrypto отображено на рисунке 37

```
demo=# CREATE EXTENSION pgcrypto;  
CREATE EXTENSION
```

Рисунок 37 – Добавление расширения pgcrypto

Обновим таблицу airports\_data при помощи команды: UPDATE airports\_data SET sha1\_hashed = encode(digest(airport\_name::text, 'sha1'), 'hex'); Обновление таблицы airports\_data отображено на рисунке 38

```
demo=# UPDATE airports_data SET sha1_hashed = encode(digest(airport_name::text, 'sha1'), 'hex');  
UPDATE 104
```

Рисунок 38 – Обновление таблицы airports\_data

При помощи команды: SELECT \* FROM airports\_data LIMIT 10; проверим, что хэширования данных было выполнено. Выполненное хэширования отображено на рисунке 39

huguenot-lr11@ZHSYBD-LR11: ~						
airport_code	airport_name	city	coordinates	timezone	sha1_hashed	
YKS	{ "en": "Yakutsk Airport", "ru": "Якутск" }	{ "en": "Yakutsk", "ru": "Якутск" }	{ 129.7709669375, 62.89329865722656 }	Asia/Yakutsk	{ c55d39f5e2108933a6e45662ae26c5f18c7c79a a7beaa42fbf8c0238d409417b6b81317fa32151	
KHV	{ "en": "Khabarovsk-Novy Airport", "ru": "Хабаровск-Новый" }	{ "en": "Khabarovsk", "ru": "Хабаровск" }	{ 135.18800354004, 48.52799987793 }	Asia/Vladivostok	{ 0197387e4773bc508399c057d5a64e978fae85c 30fc00120bc20f340e40c30f564d9993b70b2ee	
KNC	{ "en": "Yelizovo Airport", "ru": "Елизово" }	{ "en": "Petropavlovsk", "ru": "Петропавловск-Камчатский" }	{ 138.4539947507656, 53.10790008540922 }	Asia/Kamchatka	{ 30fc00120bc20f340e40c30f564d9993b70b2ee 32c715c99c0d0732c6b114025426f5c48226e	
UIS	{ "en": "Yuzhno-Sakhalinsk Airport", "ru": "Жуковское" }	{ "en": "Yuzhno-Sakhalinsk", "ru": "Южно-Сахалинск" }	{ 142.7180021933504, 46.88800937788066 }	Asia/Sakhalin	{ 87b49bde60c214adeef104e1ef4cead0d3ac0b2 fa3422e40db1f1349875ec3e3a3e42ad66675a	
VVO	{ "en": "Vladivostok International Airport", "ru": "Владивосток" }	{ "en": "Vladivostok", "ru": "Владивосток" }	{ 132.147949951172, 43.39899826049805 }	Asia/Vladivostok	{ 87b49bde60c214adeef104e1ef4cead0d3ac0b2 fa3422e40db1f1349875ec3e3a3e42ad66675a	
LED	{ "en": "Pulkovo Airport", "ru": "Пулково" }	{ "en": "St. Petersburg", "ru": "Санкт-Петербург" }	{ 30.2625076239453, 59.88030059814453 }	Europe/Moscow	{ d0e6d6e4c324c4e633c0d173f4f4f26c23a3 964edf1a29d6af07e7cbebc0d3c985d754519f1a	
KGD	{ "en": "Khabarov Airport", "ru": "Хабарово" }	{ "en": "Khabarovsk", "ru": "Хабаровск" }	{ 129.7709669375, 62.89329865722656 }	Asia/Yakutsk	{ c55d39f5e2108933a6e45662ae26c5f18c7c79a a7beaa42fbf8c0238d409417b6b81317fa32151	
KEJ	{ "en": "Kemerovo Airport", "ru": "Кемерово" }	{ "en": "Kemerovo", "ru": "Кемерово" }	{ 86.1072006225586, 55.27009963989258 }	Asia/Novokuznetsk	{ 964edf1a29d6af07e7cbebc0d3c985d754519f1a f60733a65bae592f106d53257f0edc7b8d6e880	
CEK	{ "en": "Chelyabinsk Balandino Airport", "ru": "Челябинск" }	{ "en": "Chelyabinsk", "ru": "Челябинск" }	{ 61.5033, 55.305801 }	Asia/Yekaterinburg	{ f60733a65bae592f106d53257f0edc7b8d6e880 f60733a65bae592f106d53257f0edc7b8d6e880	

Рисунок 39 – Выполненное хэширования

## 6 СОСТАВЛЕНИЕ ЗАПРОСОВ ДОСТУПА

Составим простые команды для выполнения запросов доступа.

При помощи команды: `SELECT * FROM aircrafts_data LIMIT 10;` получим первые 10 строк таблицы `aircrafts_data`. Получение первых 10 строк таблицы `aircrafts_data` отображено на рисунке 40

```
demo=# SELECT * FROM aircrafts_data LIMIT 10;
```

aircraft_code	model	range
773	{"en": "Boeing 777-300", "ru": "Боинг 777-300"}	11100
763	{"en": "Boeing 767-300", "ru": "Боинг 767-300"}	7900
SU9	{"en": "Sukhoi Superjet-100", "ru": "Сухой Суперджет-100"}	3000
320	{"en": "Airbus A320-200", "ru": "Аэробус A320-200"}	5700
321	{"en": "Airbus A321-200", "ru": "Аэробус A321-200"}	5600
319	{"en": "Airbus A319-100", "ru": "Аэробус A319-100"}	6700
733	{"en": "Boeing 737-300", "ru": "Боинг 737-300"}	4200
CN1	{"en": "Cessna 208 Caravan", "ru": "Сессна 208 Караван"}	1200
CR2	{"en": "Bombardier CRJ-200", "ru": "Бомбардье CRJ-200"}	2700

(9 rows)

Рисунок 40 – Получение первых 10 строк таблицы `aircrafts_data`

При помощи команды: `SELECT * FROM airports_data LIMIT 10;` получим первые 10 строк таблицы `airports_data`. Получение первых 10 строк таблицы `airports_data` отображено на рисунке 41

airport_code	airport_name	city	coordinates	timezone	shai hashed
VKS	{"en": "Yakutsk Airport", "ru": "Якутск"}	{"en": "Yakutsk", "ru": "Якутск"}	(129.77899609375, 62.093298865722658)	Asia/Yakutsk	c5d39f5e21809534e4a5662ae26c5f18c7c79a
KJZ	{"en": "Kiry Airport", "ru": "Нерюнели"}	{"en": "Kiry", "ru": "Нерюнели"}	(114.03800146484375, 62.5346984803328125)	Asia/Yakutsk	a7f8ea452f7f8c023864984176580132f3a32151
KHV	{"en": "Khabarovsk-Navy Airport", "ru": "Хабаровск-Новый"}	{"en": "Khabarovsk", "ru": "Хабаровск"}	(135.18800354804, 48.5279987793)	Asia/Vladivostok	0197387e4f738c508399c057d5a64e9787bae85c
KRC	{"en": "Yelizovo Airport", "ru": "Елизово"}	{"en": "Petropavlovsk", "ru": "Петропавловск-Камчатский"}	(158.45399475897656, 53.16790008544922)	Asia/Kamchatka	3bf0012dbc28f34d69dc30f564d99936b7db2ee
ULS	{"en": "Yuzhno-Sakhalinsk Airport", "ru": "Хомутово"}	{"en": "Yuzhno-Sakhalinsk", "ru": "Южно-Сахалинск"}	(142.7180023193594, 46.88869837788886)	Asia/Sakhalin	52c715d998c067520b114025426ff5cc88256e
VVO	{"en": "Vladivostok International Airport", "ru": "Владивосток"}	{"en": "Vladivostok", "ru": "Владивосток"}	(132.147994951172, 43.39899826849885)	Asia/Vladivostok	87b49ebd66c214adeef104e1ef4cead0d3aceb2
LED	{"en": "Pulkovo Airport", "ru": "Пулково"}	{"en": "St. Petersburg", "ru": "Санкт-Петербург"}	(30.262500762939453, 59.80030059814453)	Europe/Moscow	fa342e48db31f349875ec3d6a3e42ad6666675a
KGD	{"en": "Khabarovsk Airport", "ru": "Хабаровск"}	{"en": "Khabarovsk", "ru": "Хабаровск"}	(120.522599868774414, 54.8899933864844)	Europe/Kaliningrad	db8669c0c324c06339c8d173f4f4f758220c3
KEJ	{"en": "Kemerovo Airport", "ru": "Кемерово"}	{"en": "Kemerovo", "ru": "Кемерово"}	(86.1072006225586, 55.27089963989258)	Asia/Novosibirsk	964edf1a29d6af07e7cbe003c85d7545191a
CEK	{"en": "Chelyabinsk Balandino Airport", "ru": "Челябинск"}	{"en": "Chelyabinsk", "ru": "Челябинск"}	(61.5033, 55.385881)	Asia/Yekaterinburg	f60733a65abes592f186d53257f0edc7b8d6e880

(10 rows)

Рисунок 41 – Получение первых 10 строк таблицы `airports_data`

При помощи команды: `SELECT * FROM boarding_passes LIMIT 10;` получим первые 10 строк таблицы `boarding_passes`. Получение первых 10 строк таблицы `boarding_passes` отображено на рисунке 42

ticket_no	flight_id	boarding_no	seat_no	hash256
0005435208229	60731	1	1H	
0005435208224	60731	2	2A	
0005435208191	60731	3	2D	
0005435208233	60731	4	2H	
0005435208178	60731	5	3A	
0005435208193	60731	6	3D	
0005435208217	60731	7	4G	
0005435208174	60731	8	5A	
0005432210903	60731	9	11K	
0005432210909	60731	10	12C	

(10 rows)

Рисунок 42 – Получение первых 10 строк таблицы boarding\_passes

При помощи команды: `SELECT * FROM bookings LIMIT 10;` получим первые 10 строк таблицы bookings. Получение первых 10 строк таблицы bookings отображено на рисунке 43

book_ref	book_date	total_amount
00000F	2017-07-05 03:12:00+03	265700.00
000012	2017-07-14 09:02:00+03	37900.00
00002D	2017-05-20 18:45:00+03	114700.00
000068	2017-08-15 14:27:00+03	18100.00
0000C9	2017-06-30 15:52:00+03	54600.00
000104	2017-05-15 07:51:00+03	18600.00
000112	2017-06-14 23:50:00+03	22000.00
00015C	2017-05-30 02:26:00+03	56000.00
00015D	2017-06-05 20:26:00+03	281500.00
000181	2017-08-10 13:28:00+03	131800.00

(10 rows)

Рисунок 43 – Получение первых 10 строк таблицы bookings

При помощи команды: `SELECT * FROM flights LIMIT 10;` получим первые 10 строк таблицы flights. Получение первых 10 строк таблицы flights отображено на рисунке 43

flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport	arrival_airport	status	aircraft_code	actual_departure	actual_arrival
182	PG0402	2017-09-01 12:25:00+03	2017-09-01 13:20:00+03	DME	LED	Scheduled	321		
1996	PG0335	2017-08-26 09:30:00+03	2017-08-26 11:35:00+03	DME	JOK	Scheduled	CN1		
5979	PG0384	2017-08-26 12:10:00+03	2017-08-26 12:40:00+03	DME	BZK	Scheduled	SU9		
8136	PG0138	2017-08-28 10:15:00+03	2017-08-28 11:20:00+03	VKO	RTW	Scheduled	CR2		
10455	PG0277	2017-09-12 11:45:00+03	2017-09-12 15:10:00+03	SVO	OVF	Scheduled	773		
11531	PG0456	2017-08-16 16:15:00+03	2017-08-16 18:30:00+03	SVO	PES	On Time	CN1		
13774	PG0460	2017-08-25 14:30:00+03	2017-08-25 15:25:00+03	SVO	ULV	Scheduled	SU9		
16088	PG0407	2017-08-20 12:10:00+03	2017-08-20 13:05:00+03	LED	DME	Scheduled	321		
16181	PG0230	2017-08-28 10:20:00+03	2017-08-28 11:10:00+03	LED	VKO	Scheduled	321		
19239	PG0077	2017-09-04 17:45:00+03	2017-09-04 19:10:00+03	LED	CEE	Scheduled	CN1		

(10 rows)



Рисунок 43 – Получение первых 10 строк таблицы flights

При помощи команды: `SELECT * FROM seats LIMIT 10;` получим первые 10 строк таблицы seats. Получение первых 10 строк таблицы seats отображено на рисунке 44

aircraft_code	seat_no	fare_conditions
319	2A	Business
319	2C	Business
319	2D	Business
319	2F	Business
319	3A	Business
319	3C	Business
319	3D	Business
319	3F	Business
319	4A	Business
319	4C	Business
(10 rows)		

Рисунок 44 – Получение первых 10 строк таблицы seats

При помощи команды: `SELECT * FROM ticket_flights LIMIT 10;` получим первые 10 строк таблицы ticket\_flights. Получение первых 10 строк таблицы ticket\_flights отображено на рисунке 45

ticket_no	flight_id	fare_conditions	amount
0005432081075	11002	Business	99800.00
0005433845814	11047	Business	99800.00
0005432003470	27484	Business	99800.00
0005433568595	23503	Business	105900.00
0005432003656	27415	Business	99800.00
0005433415623	35652	Business	199800.00
0005432661827	43274	Business	185300.00
0005432873244	2394	Business	115000.00
0005432661894	43278	Business	185300.00
0005433569575	23477	Business	105900.00
(10 rows)			

Рисунок 45 – Получение первых 10 строк таблицы ticket\_flights

При помощи команды: `SELECT * FROM tickets LIMIT 10;` получим первые 10 строк таблицы `tickets`. Получение первых 10 строк таблицы `tickets` отображено на рисунке 46

ticket_no	book_ref	passenger_id	passenger_name	contact_data
0005432000860	8BBCD2	4750 122452	VLADIMIR FROLOV	{"phone": "+70125366530"}
0005432000861	DA7166	3889 683019	NINA BELOVA	{"email": "belovanina11041976@postgrespro.ru", "phone": "+70048667971"}
0005432000862	DA7166	3554 024596	KIRA SIDOROVA	{"email": "sidorova.kira_101971@postgrespro.ru", "phone": "+70398785493"}
0005432000863	78E2D2	2836 125969	GENNADIY NIKITIN	{"email": "nikitin_g_111965@postgrespro.ru", "phone": "+70556069198"}
0005432000864	B8F5BC	1665 656774	FEDOR SHEVCHENKO	{"email": "shevchenkofedor-1963@postgrespro.ru", "phone": "+70644329898"}
0005432000865	B8F5BC	6427 408050	DAMIR TIMOFEEV	{"phone": "+70125956007"}
0005432000866	BD402C	6243 166891	EVGENIY MAKAROV	{"email": "makarov-e011968@postgrespro.ru", "phone": "+70390705622"}
0005432000867	9328AF	5496 753314	ALFIYA FROLOVA	{"email": "alfiya-frolova1963@postgrespro.ru", "phone": "+70224059780"}
0005432000868	9328AF	7886 931683	NADEZHDA KUZMINA	{"email": "kuzmina-nadezhda.121975@postgrespro.ru", "phone": "+70308198082"}
0005432000869	9328AF	5164 327476	TATYANA ANDREEVA	{"phone": "+70875110463"}

(10 rows)

Рисунок 46 – Получение первых 10 строк таблицы `tickets`

## КОНТРОЛЬНЫЕ ВОПРОСЫ

### 1. Что такое хэш функция?

Хеш — это уникальная строка символов, полученная после применения специальной хеш-функции к определенным входным данным

### 2. Какие алгоритмы поддерживает postgres pro?

Шифрование данных:

- AES;
- RSA;

Хэширование данных:

- MD5;
- SHA-1;
- SHA-256

### 3. Какая модель разграничения прав доступа реализована в postgres pro в чем ее преимущества?

PGSQL Pro реализует модель разграничения прав доступа на основе ролей (Role-Based Access Control, RBAC). Преимущества данной модели: управление доступом на основе ролей использует принцип наименьших привилегий, поэтому пользователи располагают только теми правами, которые им нужны при исполнении своей работы, автоматические настройки и создание групп облегчают назначение прав, охватывают всю систему, снижают риски неправильно назначенной роли

## ВЫВОД

В ходе выполнения лабораторной работы по теме: «Защита хоста» получили практический навык защиты сети при эксплуатации СУБД