Приложение 4

Создание ролей

Ввод [1]:
```
# Создам роль Алиса с возможностью подключения
# и создания других ролей

!psql -U postgres -c "DROP ROLE alice"
!psql -U postgres -c "CREATE ROLE alice LOGIN CREATEROLE;"
```

```
ERROR:  role "alice" does not exist
CREATE ROLE
```

Ввод [2]:
```
# Проверим может ли Алиса подключаться к серверу
# и создавать другие роли

# Создадим Алисой роль Боб

!psql -U alice -d postgres -c "DROP ROLE bob"
!psql -U alice -d postgres -c "CREATE ROLE bob LOGIN;"
```

```
ERROR:  role "bob" does not exist
CREATE ROLE
```

Ввод [3]:
```
# Действительно, получилось и подключиться
# и создать пользователя для Боба.
# Видим что в БД есть стандартная роль
# postgres, а также alice и bob

!psql -U alice -d postgres -c "\du"
```

```
                                   List of roles
 Role name |                         Attributes
| Member of
-----------+------------------------------------------------------------
-----+-----------
 alice     | Create role
| {}
 bob       |
| {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass
RLS | {}
```

Ввод [4]:
```
# Боб не может создавать другие роли

!psql -U bob -d postgres -c "CREATE ROLE charlie LOGIN;"
```

```
ERROR:  permission denied to create role
```

Ввод [5]:
```
# Существующие роли можно изменять.
# Например, Алиса может отобрать у Боба право входа:

!psql -U alice -d postgres -c "ALTER ROLE bob NOLOGIN;"
```

```
ALTER ROLE
```

Ввод [6]:
```
# Боб не может подключиться к серверу

!psql -U bob -d postgres -c "CREATE ROLE charlie LOGIN;"
```

```
psql: error: connection to server on socket "/var/run/postgresql/.
s.PGSQL.5432" failed: FATAL:  role "bob" is not permitted to log in
```

Владельцы

Когда Алиса создает какой-либо объект в базе данных, она становится его владельцем.

Ввод [7]:
```
!psql -U postgres -c "GRANT CREATE ON SCHEMA public TO alice;"
!psql -U alice -d postgres -c "CREATE TABLE alices_table(id inte
```

```
GRANT
CREATE TABLE
```

Ввод [8]:
```
# В колонке owner виден владелец объекта

!psql -U alice -d postgres -c "\dt"
```

```
           List of relations
 Schema |     Name     | Type  |  Owner
--------+--------------+-------+----------
 public | alices_table | table | alice
 public | test         | table | postgres
(2 rows)
```

Удаление ролей

Ввод [9]:
```
# Удалить роль можно, если нет объектов, которыми она владеет.

!psql -U postgres -c "DROP ROLE alice;"
```

```
ERROR:  role "alice" cannot be dropped because some objects depend
on it
DETAIL:  privileges for schema public
owner of table alices_table
```

Ввод [10]:
```
# Чтобы удалить роль Алисы, можно передать ее объекты другой рол

!psql -U postgres -c "REVOKE CREATE ON SCHEMA public FROM alice;
!psql -U postgres -c "REASSIGN OWNED BY alice TO bob;"
```

```
REVOKE
REASSIGN OWNED
```

Ввод [11]:
```
!psql -U postgres -c "DROP ROLE alice;"
```

```
DROP ROLE
```

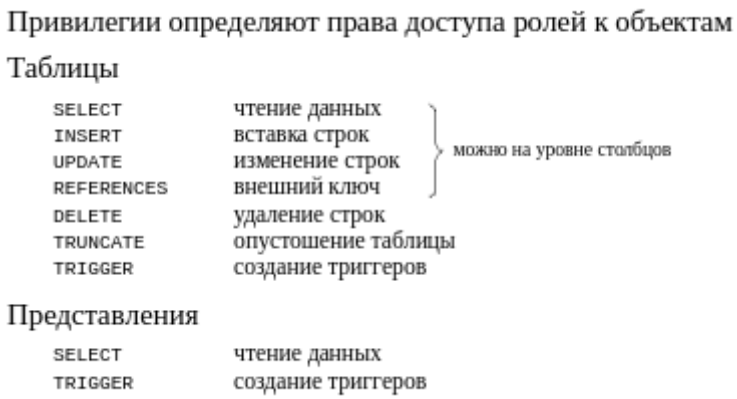
Ввод [12]:
```
# Другой вариант — удалить объекты, принадлежащие роли:

!psql -U postgres -c "DROP OWNED BY bob;"
!psql -U postgres -c "DROP ROLE bob;"
```

```
DROP OWNED
DROP ROLE
```

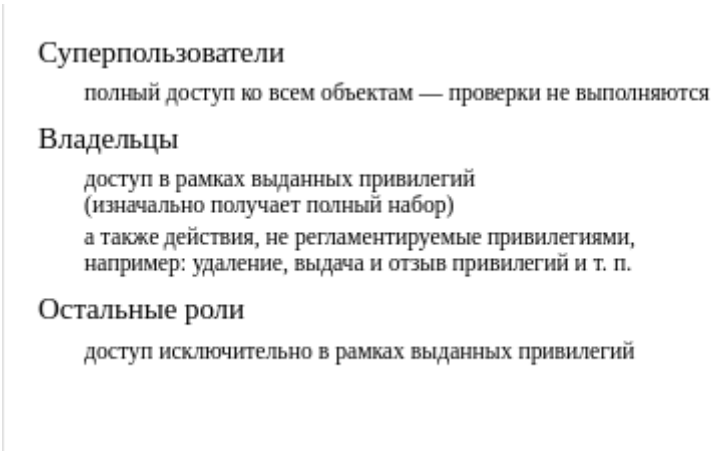
Выдача привилегий

Привилегии могут выдаваться на базы данных, схемы, табличные пространства, таблицы и представления.

Больше всего привилегий определено для таблиц. Некоторые из них можно определить не только для всей таблицы, но и для отдельных столбцов



При этом существуют три категории ролей: суперпользователи, владельцы объектов и прочие пользователи.



Привилегии выдаются командой `GRANT` и отзываются командой `REVOKE`

Ввод [13]:
```
# В моем примере Алиса будет владельцем нескольких объектов в св

!psql -U postgres -c "CREATE ROLE alice LOGIN;"
!psql -U postgres -c "CREATE SCHEMA alice;"
!psql -U postgres -c "GRANT CREATE, USAGE ON SCHEMA alice TO ali
!psql -U alice -d postgres -c "CREATE TABLE t1(n integer);"
!psql -U alice -d postgres -c "CREATE TABLE t2(n integer, m inte
```

```
CREATE ROLE
CREATE SCHEMA
GRANT
CREATE TABLE
CREATE TABLE
```

Ввод [14]:
```
# Создадим роль Боб
# Боб пробует обратиться к таблице t1.

!psql -U postgres -c "CREATE ROLE bob LOGIN;"
!psql -U bob -d postgres -c "SELECT * FROM alice.t1;"
```

```
CREATE ROLE
ERROR:  permission denied for schema alice
LINE 1: SELECT * FROM alice.t1;
                      ^
```

У Боба нет доступа к схеме, так как он не суперпользователь, не владелец схемы, и не имеет нужных привилегий.

Ввод [15]:
```
!psql -U bob -d postgres -c "\dn+ alice"
```

```
                    List of schemas
 Name  |  Owner   |  Access privileges   | Description
-------+----------+----------------------+-------------
 alice | postgres | postgres=UC/postgres+|
       |          | alice=UC/postgres    |
(1 row)
```

Ввод [16]:
```
# Чтобы Боб мог прочесть данные таблицы
# Алисе надо выдать Бобу доступ к своей схеме.

!psql -U alice -d postgres -c "GRANT CREATE, USAGE ON SCHEMA ali
```

```
WARNING:  no privileges were granted for "alice"
GRANT
```

Привилегия не выдалась потому что Алиса не является владельцем схемы. Сменим владельца схемы на Алису

Ввод [17]:
```
!psql -U postgres -c "ALTER SCHEMA alice OWNER TO alice;"
```

```
ALTER SCHEMA
```

Ввод [18]:

```
# Повторю запрос - теперь права выдались корректно
# Попробуем Бобом прочитать данные таблицы

!psql -U alice -d postgres -c "GRANT CREATE, USAGE ON SCHEMA ali
!psql -U bob -d postgres -c "SELECT * FROM alice.t1;"
```

```
GRANT
ERROR:  permission denied for table t1
```

Ввод [19]:

```
# Теперь вернулась ошибка нехватки прав
# уже не на схему а на таблицу
# Выдам права чтения на таблицу

!psql -U alice -d postgres -c "GRANT SELECT ON TABLE alice.t1 TO
!psql -U bob -d postgres -c "SELECT * FROM alice.t1;"
```

```
GRANT
 n
---
(0 rows)
```

Ввод [20]:

```
# Права выдались и Боб смог прочесть таблицу.
# Кстати она пустая!

# Попробуем Бобом положить туда какие-то данные

!psql -U bob -d postgres -c "INSERT INTO alice.t1 VALUES (911);"
```

```
ERROR:  permission denied for table t1
```

Ввод [21]:

```
# Опять вернулась ошибка потому что у Боба
# есть только права на чтение (SELECT)
# Чтобы об мог вставлять новые строки в таблицу
# ему нужны права на INSERT

# Попросим Алису выдать Бобу нужные права

!psql -U alice -d postgres -c "GRANT INSERT ON TABLE alice.t1 TO

# Повторим попытку вставки

!psql -U bob -d postgres -c "INSERT INTO alice.t1 VALUES (911);"
```

```
GRANT
INSERT 0 1
```

Ввод [22]:

```
# Кажется удалось. Прочтем таблицу заново

!psql -U bob -d postgres -c "SELECT * FROM alice.t1;"
```

```
  n
-----
 911
(1 row)
```

Таким же образом Алиса или суперюзер postgres (или любой другой суперюзер) может выдать права на:

- SELECT
- INSERT
- UPDATE
- DELETE
- TRUNCATE
- TRIGGER
- REFERENCES

или для выдачи всех прав разом можно просто указать ALL

Ввод [23]:
```
1 !psql -U alice -d postgres -c "GRANT ALL ON TABLE alice.t1 TO bo
```

```
GRANT
```

Ввод [24]:
```
1 # Теперь Боб может удалить строку из
2 # таблицы или полностью очистить её
3
4 !psql -U bob -d postgres -c "TRUNCATE alice.t1;"
5 !psql -U bob -d postgres -c "SELECT * FROM alice.t1;"
```

```
TRUNCATE TABLE
 n
---
(0 rows)
```

Ввод [25]:
```
1 # Удалим в конце все созданные объекты
2
3 !psql -U postgres -c "DROP OWNED BY alice;"
4 !psql -U postgres -c "DROP OWNED BY bob;"
5 !psql -U postgres -c "DROP ROLE alice;"
6 !psql -U postgres -c "DROP ROLE bob;"
```

```
DROP OWNED
DROP OWNED
DROP ROLE
DROP ROLE
```