

# Задание 1. Создание таблицы и настройка RLS

Step 1: Создать тестовую таблицу, назовите её **skillbox\_test**. Таблица должна содержать поля: «Идентификатор», «Логин», «Размер заработной платы», «Контактный номер телефона».

Step 2: Создать тестовые учётные записи пользователей БД.

Для сотрудников отдела сбыта:

Семёнов Вадим (login SEMIONOV)

Данилов Иван (login DANILOV)

Для бухгалтерии:

Романова Светлана (login ROMANOVA)

Step 3: Заполнить таблицу следующими данными:

1, SEMIONOV, 50000, 791101234567

2, DANILOV, 90000, 795301234567

3, ROMANOVA, 70000, 790401234567

```
postgres=# create database test_db
postgres=# ;
CREATE DATABASE
postgres=# create table test_skillbox;
ERROR:  syntax error at or near ";"
LINE 1: create table test_skillbox;
          ^

postgres=# create table test_skillbox (identifier integer, username text, salary integer, phone number bigint);
ERROR:  syntax error at or near "bigint"
LINE 1: ...nteger, username text, salary integer, phone number bigint);
                                     ^

postgres=# create table test_skillbox (identifier integer, username text, salary integer, phone_number bigint);
CREATE TABLE
postgres=# insert into test_skillbox values ('1', 'Semionov', '50000', '791101234567');
INSERT 0 1
postgres=# insert into test_skillbox values ('2', 'Danilov', '90000', '795301234567');
INSERT 0 1
postgres=# insert into test_skillbox values ('3', 'Romanova', '70000', '790401234567');
INSERT 0 1
postgres=# select * from test_skillbox;
 identifier | username | salary | phone_number
-----+-----+-----+-----
         1 | Semionov | 50000 | 791101234567
         2 | Danilov | 90000 | 795301234567
         3 | Romanova | 70000 | 790401234567
(3 rows)

postgres=# grant all on table test_skillbox to semionov, danilov, romanova;
ERROR:  role "semionov" does not exist
postgres=# create user semionol nosuperuser login;
ERROR:  syntax error at or near ","
LINE 1: create user semionol nosuperuser, login;
                                     ^

postgres=# create user semionol nosuperuser login;
CREATE ROLE
postgres=# create user danilov nosuperuser login;
CREATE ROLE
postgres=# create user romanova nosuperuser login;
CREATE ROLE
postgres=# grant all on table test_skillbox to semionov, danilov, romanova;
ERROR:  role "semionov" does not exist
postgres=# alter role semionol
BYPASSRLS          NOCREATEDB          REPLICATION
CONNECTION LIMIT   NOCREATEROLE         RESET
CREATEDB           NOINHERIT              SET
CREATEROLE         NOLOGIN                SUPERUSER
ENCRYPTED PASSWORD  NOREPLICATION          VALID UNTIL
INHERIT            NOSUPERUSER            WITH
```

Тут 3 в 1 — создание пользователей, таблицы и её заполнение.

Step 4: Настроить правила RLS.

Для сотрудников отдела сбыта: видят только свои личные оклады (одну запись со своим логином).

Для сотрудника бухгалтерии: видит все данные.

```

test_db=# CREATE FUNCTION read_me(text) RETURNS boolean as
test_db=# $$
test_db$# BEGIN
test_db$#     RAISE NOTICE 'called as sessio_user=%, current_user=% for "%" ',
test_db$#         session_user, current_user, $1;
test_db$#     RETURN true;
test_db$# END;
test_db$# $$ LANGUAGE 'plpgsql';
ERROR:  unrecognized exception condition "notice"
CONTEXT:  compilation of PL/pgSQL function "read_me" near line 3
test_db=# CREATE FUNCTION read_me(text) RETURNS boolean as
test_db=# $$
test_db$# BEGIN
test_db$#     RAISE NOTICE 'called as sessio_user=%, current_user=% for "%" ',
test_db$#         session_user, current_user, $1;
test_db$#     RETURN true;
test_db$# END;
test_db$# $$ LANGUAGE 'plpgsql';
test_db=# CREATE FUNCTION
test_db=# GRANT ALL ON FUNCTION read_me TO romanova, semionov, danilov;
test_db=# GRANT
test_db=#

```

Не знаю, зачем добавил данную функцию.

```

test_db=# CREATE POLICY sem_info on test_skillbox
test_db=#   for select
test_db=#   to semionov
test_db=#   using (read_me(Username) and status = 'Employee 1');
test_db=# CREATE POLICY
test_db=# CREATE POLICY dan_info on test_skillbox
test_db=#   for select
test_db=#   to danilov
test_db=#   using (read_me(Username) and status = 'Employee 2');
test_db=# CREATE POLICY
test_db=#

```

Тут я создал 2 политики, где 2 данных пользователя могут видеть только их соответствующие ряды.

```

test_db=# CREATE POLICY admin_all on test_skillbox to romanova using (true) with check (true);
test_db=# CREATE POLICY
test_db=#

```

Тут политика, где данный пользователь может видеть всю таблицу.

```

test_db=# alter table test_skillbox enable row level security;
test_db=# ALTER TABLE
test_db=#

```

И самое главное, включить безопасность на уровне ряда.

```

ERROR: must be owner of table test_skillbox
test_db=> \c test_db postgres
You are now connected to database "test_db" as user "postgres".
test_db=# alter table test_skillbox enable row level security;
ALTER TABLE
test_db=# \c test_db romanova
You are now connected to database "test_db" as user "romanova".
test_db=> select * from test_skillbox;
   status   | username | salary | phone_number
-----+-----+-----+-----
Employee 1 | semionov | 50000  | 791101234567
Employee 2 | danilov  | 90000  | 795301234567
Accountant | romanova | 70000  | 790401234567
(3 rows)

test_db=> \c test_db semionov
You are now connected to database "test_db" as user "semionov".
test_db=> select * from test_skillbox;
NOTICE: called as sessio_user=semionov, current_user=semionov for "semionov"
   status   | username | salary | phone_number
-----+-----+-----+-----
Employee 1 | semionov | 50000  | 791101234567
(1 row)

test_db=> \c test_db danilov
You are now connected to database "test_db" as user "danilov".
test_db=> select * from test_skillbox;
NOTICE: called as sessio_user=danilov, current_user=danilov for "danilov"
   status   | username | salary | phone_number
-----+-----+-----+-----
Employee 2 | danilov  | 90000  | 795301234567
(1 row)

test_db=> █

```

Итоговый результат.

P.S. Хотя в задании было указано так - «Создать SQL-скрипт, выполнив описанные ниже действия». Но я решил сделать по своему.

## Задание 2. Настройка маскирования

Step 1: Подключить расширение PostgreSQL Anonymizer.

```
(kali@kali)-[~]
$ sudo docker run -d -e POSTGRES_PASSWORD=x -p 6543:5432 registry.gitlab.com/dalibo/postgresql_anonymizer
Unable to find image 'registry.gitlab.com/dalibo/postgresql_anonymizer:latest' locally
latest: Pulling from dalibo/postgresql_anonymizer
578acb154839: Download complete
25417907e653: Download complete
2fb3d089144a: Download complete
d9e5621a4ac2: Download complete
41435b4dcf75: Downloading [=====]
22a19612d6d7: Downloading [=====]
8a44e3712846: Waiting
9a7b0e8e2d10: Waiting
28c03e657381: Waiting
68b985b60e92: Waiting
044a9e28cc77: Waiting
54af805c8770: Waiting
1b3fa3e19558: Waiting
d967a8cb5bbf: Waiting
30ca43136912: Waiting
216798779aa8: Waiting
223a451d7829: Waiting
2336fd99a639: Waiting
64aa298249a6: Waiting
852e7b6b7c17: Waiting
70c46ee7597d: Waiting
7a0aa6e42d73: Waiting
```

```
kali@kali:~$ sudo apt update
kali@kali:~$ ] 4.424MB/8.068MB
kali@kali:~$ ] 811.1kB/1.195MB
kali@kali:~$
kali@kali:~$ sudo systemctl enable docker
kali@kali:~$
kali@kali:~$ docker
kali@kali:~$

You can now get started with using docker, with 'sudo -i fy

kali@kali:~$ sudo usermod -aG docker $USER
kali@kali:~$
```

Установка данной утилиты (вот почему я это делаю на Kali Linux, а не на Ubuntu).

```
ubuntu@u: ~
9ad9b1166fde: Pulling fs layer
286c4682b24d: Waiting
1d3679a4a1a1: Waiting
5f2e6cdc8503: Waiting
0f7dc70f54e8: Waiting
a090c7442692: Waiting
81bfe40fd0f6: Waiting
8ac8c22bbb38: Waiting
96e51d1d3c6e: Waiting
667bd4154fa2: Waiting
87267fb600a9: Waiting
6b15cab3bc5c: Waiting
d6993b043863: Waiting
6f6881948ca4: Waiting
96b1926a06b7: Waiting
a9ad00784d8b: Waiting
0847367da048: Waiting
96ffab8968b1: Waiting
d7d7a34852ed: Waiting
error pulling image configuration: download failed after attempts=1: error parsing HTTP 403 response body: invalid character '<' looking for beginning of value:
"<?xml version='1.0' encoding='UTF-8'?><Error><Code>AccessDenied</Code><Message>Access denied.</Message></Error>"
ubuntu@u:~$
```

Ошибка доступа (или статус 403). Сделав то же самое на Kali Linux, ошибки не было.

Step 2: Настроить правила маскирования на таблицу **skillbox\_test** (из задания 1) так, чтобы сотрудникам отдела сбыта в поле «Телефон» показывались бы только четыре символа слева, остальные символы выводились бы символом «\*», сотрудники бухгалтерии должны видеть данные без маскирования (без назначения роли superuser).

```

kali@kali: ~
File Actions Edit View Help

t
(1 row)

postgres=# create role semionov nosuperuser login;
CREATE ROLE
postgres=# create role danilov nosuperuser login;
CREATE ROLE
postgres=# create role romanova nosuperuser login;
CREATE ROLE
postgres=# security label for anon on role semionov is 'MASKED';
SECURITY LABEL
postgres=# security label for anon on role danilov is 'MASKED';
SECURITY LABEL
postgres=# create table (Identifier serial, username text, salary text, phone text);
ERROR: syntax error at or near "("
LINE 1: create table
                  ^
postgres=# create table test_skillbox (Identifier serial, username text, salary text, phone text);
CREATE TABLE
postgres=# insert into test_skillbox values ('1', 'semionov', '50000', '+79101234567');
INSERT 0 1
postgres=# insert into test_skillbox values ('2', 'danilov', '90000', '+795301234567');
INSERT 0 1
postgres=# insert into test_skillbox values ('3', 'romanova', '70000', '+790401234567');
INSERT 0 1
postgres=# security label for anon on column test_skillbox.phone is 'MASKED WITH FUNCTION anon.partial(phone)';
ERROR: function anon.partial(text, integer, unknown) does not exist
LINE 1: ...killbox AS SELECT identifier,username,salary,CAST(anon parti...
                                ^
HINT: No function matches the given name and argument types. You might need to add explicit type casts.
QUERY: CREATE OR REPLACE VIEW mask_test_skillbox AS SELECT identifier,username,salary,CAST(anon.partial(phone),4,$$*****$) AS text) AS phone FROM public.test_skillbox
CONTEXT: PL/pgSQL function anon.mask_create_view(oid) line 3 at EXECUTE SQL statement "SELECT anon.mask_create_view(oid)"
FROM pg_catalog.pg_class
WHERE relnamespace=quote_ident(pg_catalog.current_setting('anon.sourceschema'))::REGNAMESPACE
AND reldkind IN ('r','p','f') -- relations or partitions or foreign tables"
PL/pgSQL function anon.mask_update() line 15 at PERFORM SQL statement "SELECT anon.mask_update()"
PL/pgSQL function anon.trg_mask_update() line 5 at PERFORM SQL statement "SELECT anon.mask_update()"
postgres=# create extension
adminpack btree_gist dict_xsyn hstore isn moddatetime pg_prewarm pg_trgm seg tsm_system_time
cardinality old_snapshot pgrowlocks pg_visibility pg_unaccent
cube faker anonymization insert_username jsonb_locales plpgsql plythonzu plythonzu tablefunc row_publishing "uuid-osp" "v4-t
bloom file_fdw intagg ltree pg_buffercache pgstattuple postgres_fdw tcn xml2
btree_gin dict_int fuzzystmatch intarray ltree_plython3u pg_freepacemap pg_surgery refint tsm_system_rows

postgres=# create extension anon
CASCAD E VERSION WITH SCHEMA
postgres=# create extension anon cascade;
ERROR: extension "anon" already exists
postgres=# select anon.init();

```

Мне пришлось создавать ту ДБ заново, поскольку фактически это другой инстанс (ту ДБ делал в самом PostgreSQL, а утилита была установлена и использована через Docker).

```
kali@kali: ~
File Actions Edit View Help
postgres=# insert into test_skillbox values ('2', 'danilov', '90000', '+795301234567');
INSERT 0 1
postgres=# insert into test_skillbox values ('3', 'romanova', '70000', '+790401234567');
INSERT 0 1
postgres=# select * from test;
 identifier | username | salary | phone
-----+-----+-----+-----
(0 rows)

postgres=# insert into test values ('3', 'romanova', '70000', '+790401234567');
INSERT 0 1
postgres=# insert into test values ('2', 'danilov', '90000', '+795301234567');
INSERT 0 1
postgres=# insert into test values ('1', 'semionov', '50000', '+791101234567');
INSERT 0 1
postgres=# select * from test;
 identifier | username | salary | phone
-----+-----+-----+-----
(3 rows)

postgres=# security label for anon on column test.phone is 'MASKED WITH FUNCTION anon.partial(phone,4,$$*****$$,2)';
SECURITY LABEL
postgres=# security label for anon on column test.phone is 'MASKED WITH FUNCTION anon.partial(phone,4,$$*****$$)';
ERROR:  function anon.partial(text, integer, unknown) does not exist
LINE 1: ... sk.test AS SELECT identifier,username,salary,CAST(anon parti ...
HINT:  No function matches the given name and argument types. You might need to add explicit type casts.
QUERY:  CREATE OR REPLACE VIEW mask.test AS SELECT identifier,username,salary,CAST(anon.partial(phone,4,$$*****$$) AS text) AS phone FROM public.test
CONTEXT:  PL/pgSQL function anon.mask_create_view(oid) line 3 at EXECUTE
SQL statement "SELECT anon.mask_create_view(oid)"
FROM pg_catalog.pg_class
WHERE relnamespace=quote_ident(pg_catalog.current_setting('anon.sourceschema'))::REGNAMESPACE
AND relkind IN ('r','p','f') -- relations or partitions or foreign tables
PL/pgSQL function anon.mask_update() line 15 at PERFORM
SQL statement "SELECT anon.mask_update()"
PL/pgSQL function anon.trg_mask_update() line 5 at PERFORM
postgres=# select * from test;
 identifier | username | salary | phone
-----+-----+-----+-----
(3 rows)

postgres=# \q
(kali@kali)~$ psql --host=localhost -U danilov -p 6543
Completing 'PostgreSQL user'
Completing PostgreSQL user
```

Фактически повторение первых 3 шагов из задания 1 — самая первая команда была «select anon.start\_dynamic\_masking();». Там была не одна попытка написать данную функцию про номер телефона, но вылезала ошибка — получилось пока так (phone,4,\$\$\*\*\*\*\*\$\$,2), хотя в задании указано только 4 символа слева.

```
postgres=# select * from test;
 identifier | username | salary | phone
-----+-----+-----+-----
(3 rows)

postgres=# \c postgres semionov
Password for user semionov:
psql (16.1 (Debian 16.1-1), server 14.10 (Debian 14.10-1.pgdg120+1))
You are now connected to database "postgres" as user "semionov".
postgres=# select * from test;
 identifier | username | salary | phone
-----+-----+-----+-----
(3 rows)

postgres=# \c postgres romanova;
Password for user romanova:
psql (16.1 (Debian 16.1-1), server 14.10 (Debian 14.10-1.pgdg120+1))
You are now connected to database "postgres" as user "romanova".
postgres=# select * from test;
 identifier | username | salary | phone
-----+-----+-----+-----
(3 rows)

postgres=#
```

Пользователь Романова видит все данные, а Semionov и Danilov видят только часть своих номеров телефона (если что, RLS там не включён).

P.S. Хотя в задании было указано так - «Создать SQL-скрипт, выполнив описанные ниже действия». Но я решил сделать по своему.