Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

«ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL»

Специальность:
3. Мобильные и сетевые технологии

Проверил:
Говорова М.М._____
Дата: «__»_____20__г.
Оценка _____

Выполнил:
студент группы K3241
Балцат К.

Санкт-
Петербург2022

ЦЕЛЬ РАБОТЫ

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Оборудование: компьютерный класс.
Программное обеспечение: СУБД PostgreSQL 1X, pgAdmin 4.

Практическое задание:

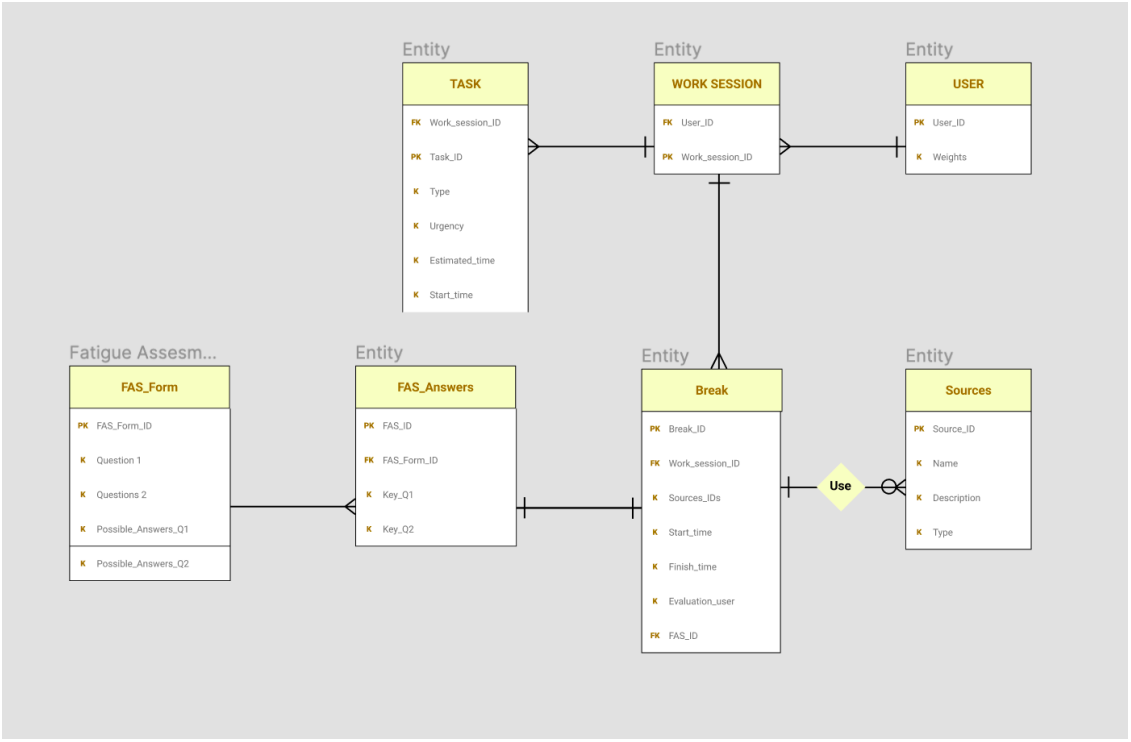
1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
2. Создать триггер для логирования событий вставки, удаления, редактирования данных
в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

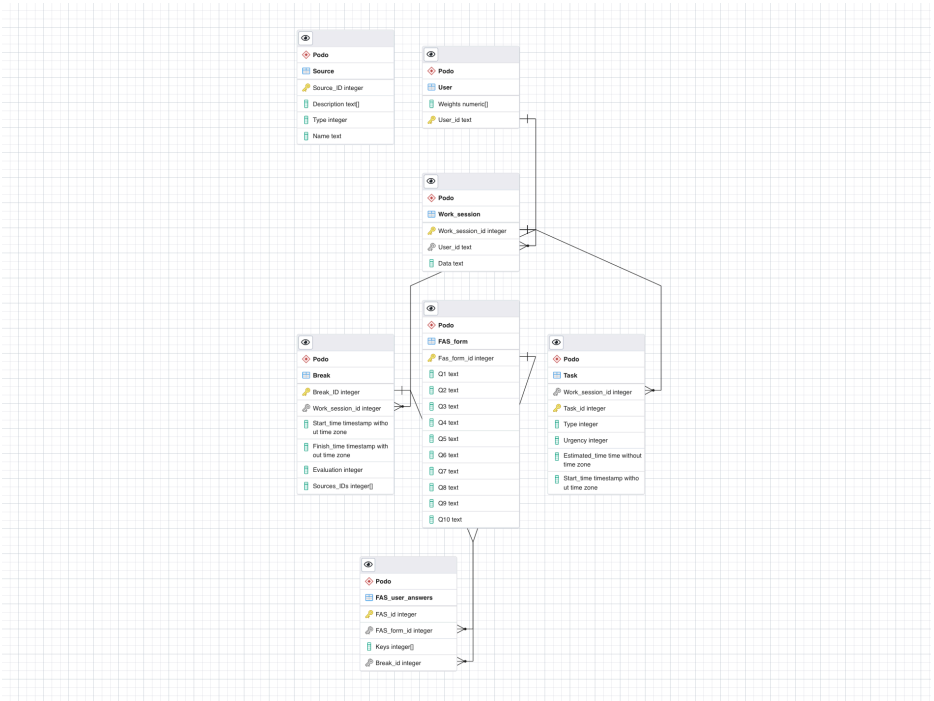
1.

ХОД РАБОТЫ

1) Наименование БД:
Podo Data Base

2) Схема логической модели:

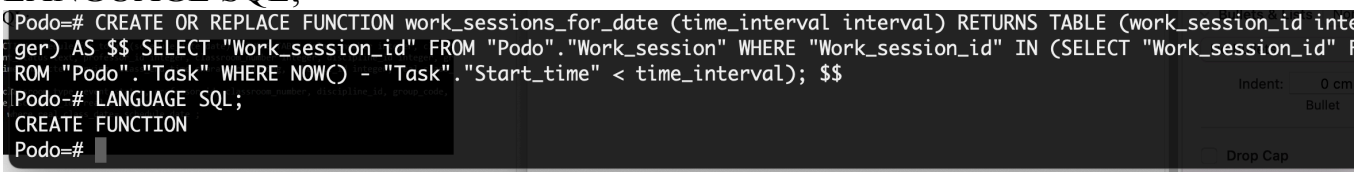




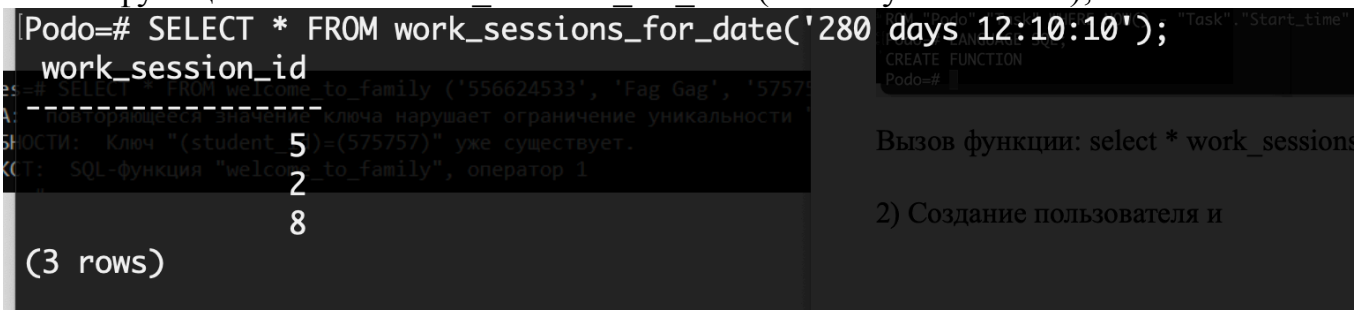
3) Создание хранимых процедур

1) Для получения рабочих сессий, открытых в последнее время

CREATE OR REPLACE FUNCTION work_sessions_for_date (time_interval interval) RETURNS TABLE (work_session_id integer) AS $$ SELECT "Work_session_id" FROM "Podo"."Work_session" WHERE "Work_session_id" IN (SELECT "Work_session_id" FROM "Podo"."Task" WHERE (NOW() - "Task"."Start_time" < time_interval)); $$
LANGUAGE SQL;



Вызов функции: select * work_sessions_for_date(‘280 days 12:10:10’);

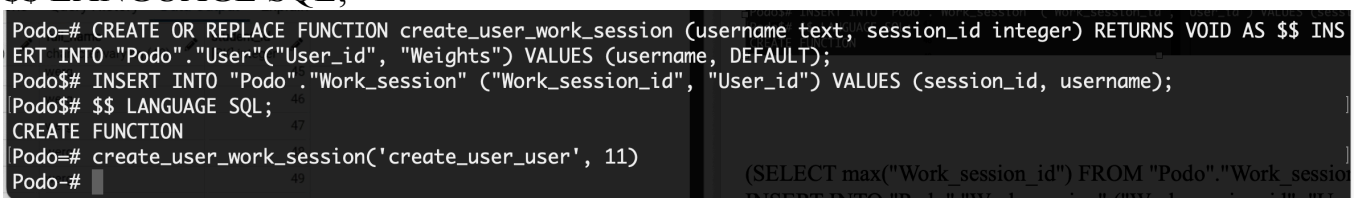


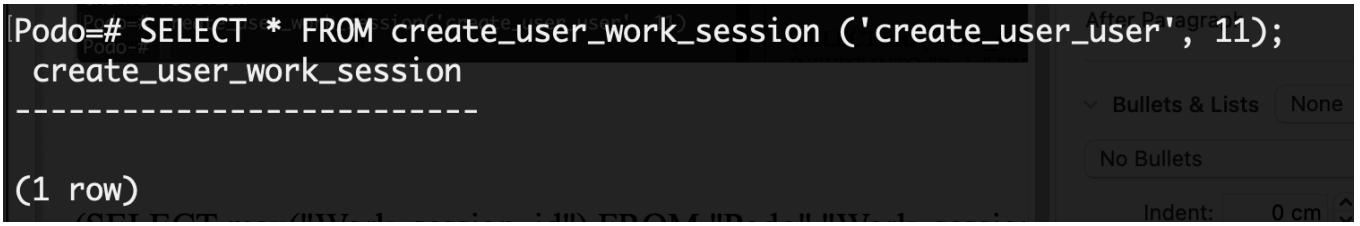
Вызов функции: select * work_sessions

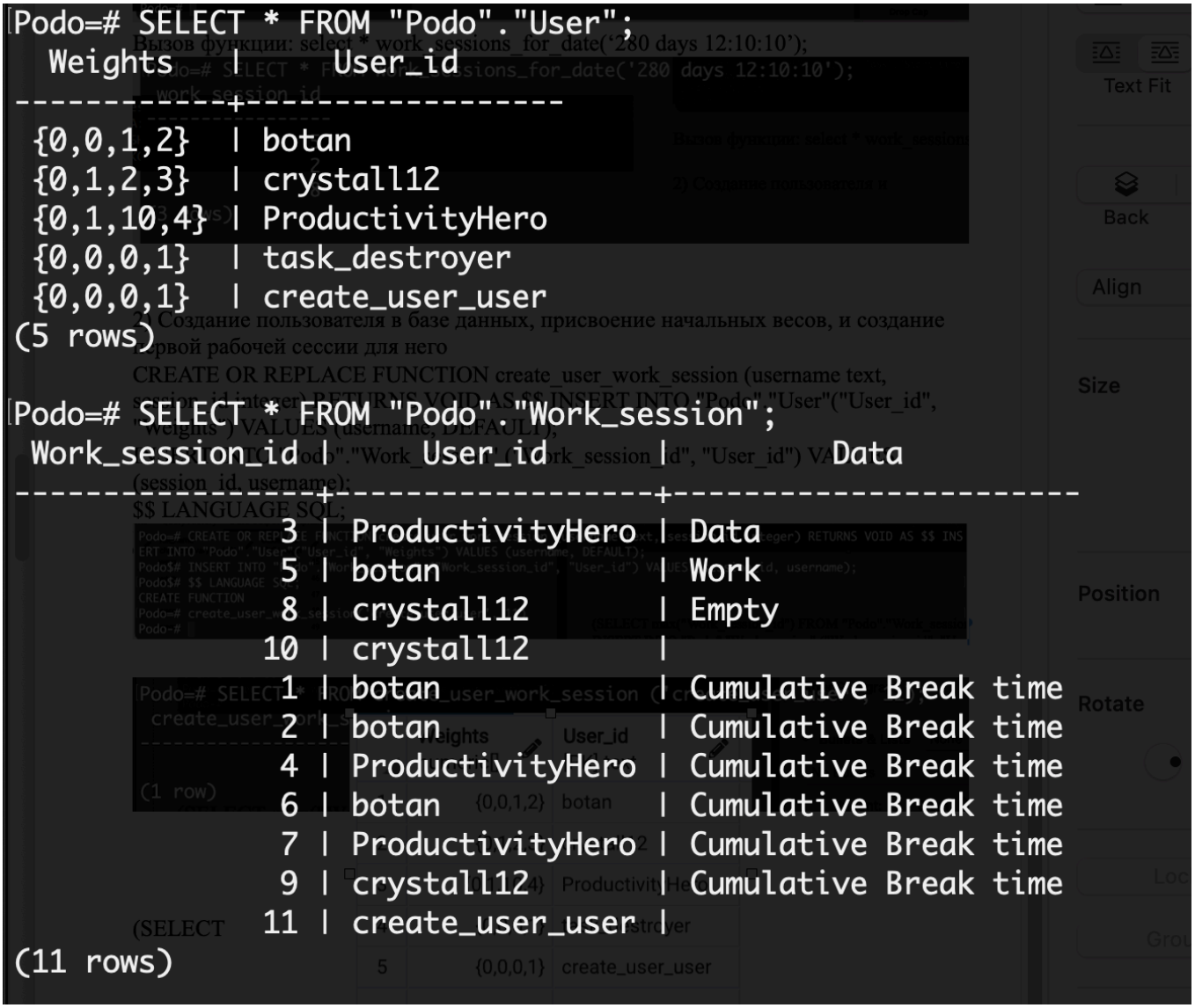
2) Создание пользователя и

2) **Создание пользователя в базе данных, присвоение начальных весов, и создание первой рабочей сессии для него**

CREATE OR REPLACE FUNCTION create_user_work_session (username text, session_id integer) RETURNS VOID AS $$ INSERT INTO "Podo"."User"("User_id", "Weights") VALUES (username, DEFAULT);
INSERT INTO "Podo"."Work_session" ("Work_session_id", "User_id") VALUES (session_id, username);
$$ LANGUAGE SQL;







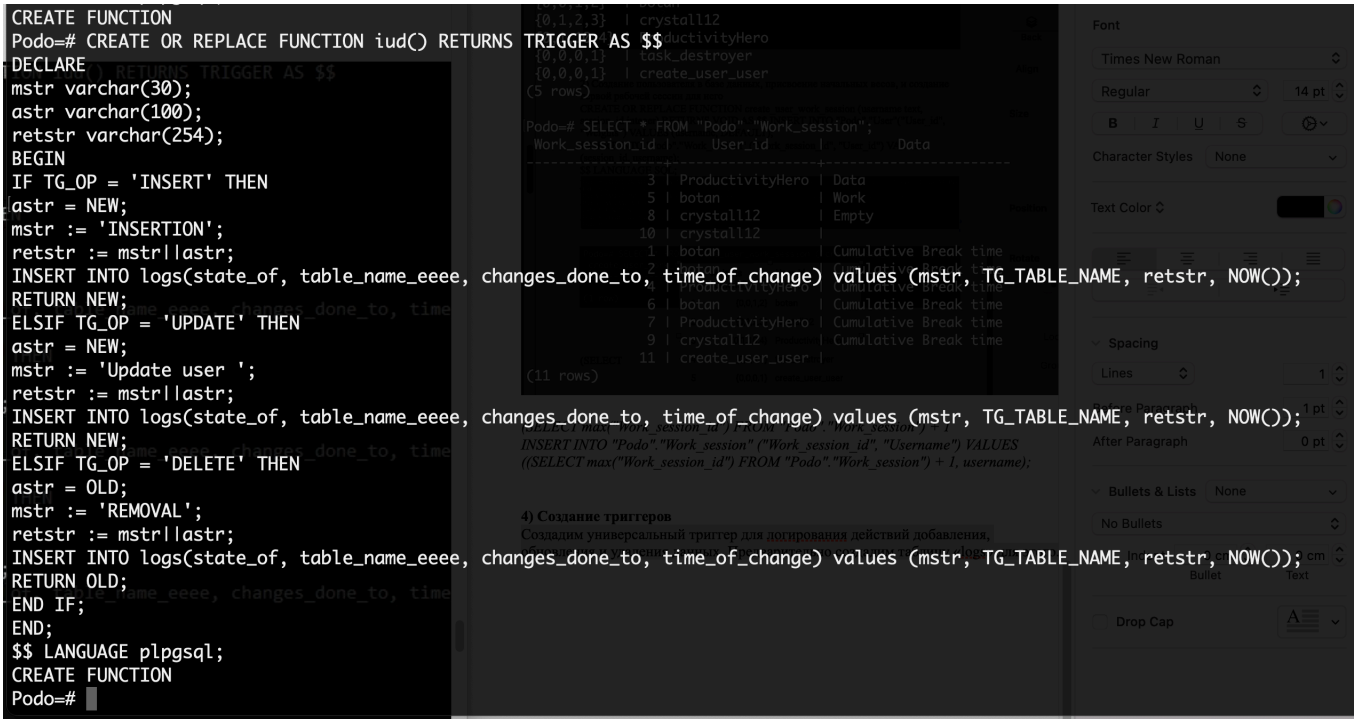
```
Podo=# SELECT * FROM "Podo"."User";
  Weights   |     User_id
------------+------------------
 {0,0,1,2}  | botan
 {0,1,2,3}  | crystall12
 {0,1,10,4} | ProductivityHero
 {0,0,0,1}  | task_destroyer
 {0,0,0,1}  | create_user_user
(5 rows)

Podo=# SELECT * FROM "Podo"."Work_session";
 Work_session_id |     User_id      |         Data
-----------------+------------------+----------------------
               3 | ProductivityHero | Data
               5 | botan            | Work
               8 | crystall12       | Empty
              10 | crystall12       |
               1 | botan            | Cumulative Break time
               2 | botan            | Cumulative Break time
               4 | ProductivityHero | Cumulative Break time
               6 | botan            | Cumulative Break time
               7 | ProductivityHero | Cumulative Break time
               9 | crystall12       | Cumulative Break time
              11 | create_user_user |
(11 rows)
```

(SELECT max("Work_session_id") FROM "Podo"."Work_session") + 1
INSERT INTO "Podo"."Work_session" ("Work_session_id", "Username") VALUES
((SELECT max("Work_session_id") FROM "Podo"."Work_session") + 1, username);

CREATE OR REPLACE FUNCTION create_user_work_session (username text)
RETURNS VOID AS $$ INSERT INTO "Podo"."User"("User_id", "Weights") VALUES
(username, DEFAULT);
INSERT INTO "Podo"."Work_session" ("Work_session_id", "User_id") VALUES
((SELECT max("Work_session_id") FROM "Podo"."Work_session") + 1, username);
$$ LANGUAGE SQL;

4) Создание триггеров

Создадим универсальный триггер для логирования действий добавления, обновления и удаления данных. Предварительно создадим таблицу «logs» для этого.




```
CREATE OR REPLACE FUNCTION iud() RETURNS TRIGGER AS $$
DECLARE
mstr varchar(30);
astr varchar(100);
retstr varchar(254);
BEGIN
IF TG_OP = 'INSERT' THEN
astr = NEW;
mstr := 'INSERTION';
retstr := mstr||astr;
INSERT INTO "Podo".logs(state_of, table_name_eeee, changes_done_to, time_of_change)
values (mstr, TG_TABLE_NAME, retstr, NOW());
RETURN NEW;
ELSIF TG_OP = 'UPDATE' THEN
astr = NEW;
mstr := 'Update user ';
retstr := mstr||astr;
INSERT INTO "Podo".logs(state_of, table_name_eeee, changes_done_to, time_of_change)
values (mstr, TG_TABLE_NAME, retstr, NOW());
RETURN NEW;
ELSIF TG_OP = 'DELETE' THEN
astr = OLD;
mstr := 'REMOVAL';
retstr := mstr||astr;
INSERT INTO "Podo".logs(state_of, table_name_eeee, changes_done_to, time_of_change)
values (mstr, TG_TABLE_NAME, retstr, NOW());
RETURN OLD;
END IF;
```

END;
$$ LANGUAGE plpgsql;

Создадим триггер и сделаем запросы:

```
Podo=# CREATE TRIGGER tt AFTER INSERT OR UPDATE OR DELETE ON "Podo"."User" FOR EACH ROW
  EXECUTE PROCEDURE iud();
CREATE TRIGGER
```

```
Podo=# UPDATE "Podo"."User" SET "Weights" = '{0,0,1,1}' where "User_id" = 'botan';
UPDATE 1
Podo=# INSERT INTO "Podo"."User" ("User_id") VALUES ('triggered_user');
INSERT 0 1
Podo=# DELETE FROM "Podo"."User" WHERE "User_id" = 'task_destroyer';
DELETE 1
Podo=#
```

Результаты:

```
  state_of    | table_name_eeee |            changes_done_to             |       time_of_change
--------------+-----------------+----------------------------------------+----------------------------
 Update user  | User            | Update user ("{0,0,1,1}",botan)        | 2022-06-01 12:50:15.152452
 Update user  | User            | Update user ("{0,0,1,1}",botan)        | 2022-06-01 13:09:50.021479
 INSERTION    | User            | INSERTION("{0,0,0,1}",triggered_user)  | 2022-06-01 13:10:35.666518
 REMOVAL      | User            | REMOVAL("{0,0,0,1}",task_destroyer)    | 2022-06-01 13:11:15.176821
(4 rows)
```

Вывод:

В ходе выполнения лабораторной работы я узнал принцип создания хранимых процедур и триггеров.