Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

Лабораторная работа №3 «Процедуры, функции и триггеры в PostgreSQL» по дисциплине: «Проектирование и реализация баз данных»

Выполнила:
студентка II курса ИКТ
группы К3242
Скокова Алина Викторовна

Проверила:
Говорова Марина Михайловна

Санкт-Петербург
2022

Цель работы: овладеть практическими навыками создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Практическое задание (вариант 1).

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.

Выполнение.

Создание хранимых процедур/функций:

1. Вывести список всех звонков заданного абонента (рис. 1).

```
create function client_calls(character) returns setof payments.call_outgoing as
$$
    select call_id, subscription_id, time_call_start, time_call_end, zone_type, country_id, discount, status, date_payment
    from (payments.call_outgoing natural join payments.subscription) natural join payments.contract
    where passport = $1
$$ language sql;
```

```
telephone_exchange_db=# create function client_calls(character) returns setof payments.call_outgoing as
telephone_exchange_db-# $$
telephone_exchange_db$# select call_id, subscription_id, time_call_start, time_call_end, zone_type, country_id, discount, status, date_payment
telephone_exchange_db$# from (payments.call_outgoing natural join payments.subscription) natural join payments.contract
telephone_exchange_db$# where passport = $1
telephone_exchange_db$# $$ language sql;
CREATE FUNCTION
telephone_exchange_db=# select * from client_calls('1111 111111');
 call_id | subscription_id |   time_call_start   |    time_call_end    | zone_type | country_id | discount |  status  |    date_payment
---------+-----------------+---------------------+---------------------+-----------+------------+----------+----------+---------------------
      15 |               1 | 2022-09-25 20:00:00 | 2022-09-25 20:20:00 | страна    |            |      0.2 | оплачено | 2022-09-25 20:23:00
       3 |               1 | 2022-06-05 21:14:00 | 2022-06-05 21:17:00 | страна    |            |      0.2 | оплачено | 2022-06-05 21:18:00
       1 |               1 | 2022-06-03 08:00:00 | 2022-06-03 08:10:00 | страна    |            |      0.0 | оплачено | 2022-06-03 08:15:00
(3 строки)


telephone_exchange_db=# select * from client_calls('4900 836477');
 call_id | subscription_id |   time_call_start   |    time_call_end    | zone_type | country_id | discount |  status  |    date_payment
---------+-----------------+---------------------+---------------------+-----------+------------+----------+----------+---------------------
      14 |               6 | 2022-09-23 11:17:00 | 2022-09-23 11:20:00 | страна    |            |      0.0 | оплачено | 2022-09-23 11:21:00
(1 строка)
```

Рисунок 1 – Результаты работы функции 1

2. Вывести задолженность по оплате для заданного абонента (рис. 2).

```
create function client_debt(character) returns setof double precision as $$
select sum(p.price) as debt
from
(select s.price
from (select round(((extract(hour from (time_call_end - time_call_start)) * 60 +
extract(minute from (time_call_end - time_call_start)))
        * tariff_price) * (1-discount)) as price,
((extract(hour from (time_call_end - time_call_start)) * 60 + extract(minute from
(time_call_end - time_call_start)))
        * country_price) * (1-discount) as price2
from (((payments.call_outgoing natural left join payments.international_calls)
    natural join payments.subscription) natural left join payments.tariff)
    natural join payments.contract
where status = 'ожидание' and passport = $1) as s
where s.price2 is null
union all
select t.price2
from (select round(((extract(hour from (time_call_end - time_call_start)) * 60 +
extract(minute from (time_call_end - time_call_start)))
        * tariff_price) * (1-discount)) as price,
((extract(hour from (time_call_end - time_call_start)) * 60 + extract(minute from
(time_call_end - time_call_start)))
        * country_price) * (1-discount) as price2
from (((payments.call_outgoing natural left join payments.international_calls)
    natural join payments.subscription) natural left join payments.tariff)
    natural join payments.contract
where status = 'ожидание' and passport = $1) as t
where t.price2 is not null) as p;
$$ language sql;
```

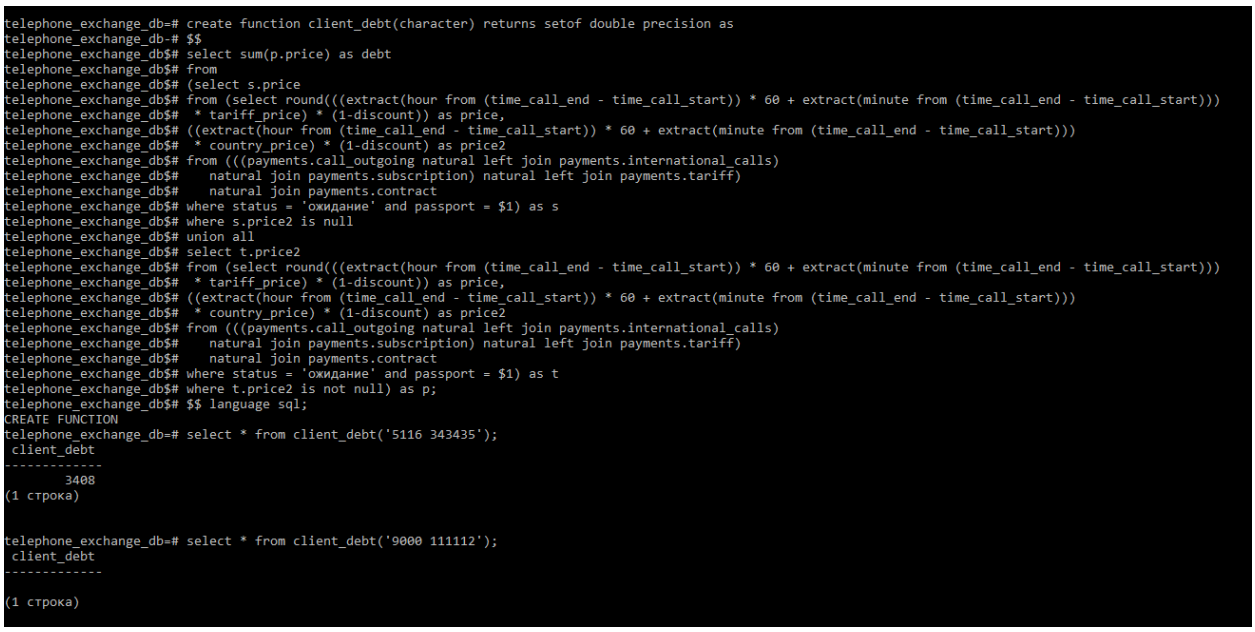

Рисунок 2 – Результаты работы функции 2

3. Рассчитать общую стоимость звонков по каждой зоне за истекшую неделю (рис. 3).

```
create function prev_week_zone_sum() returns table(zone_type character varying, sum_price double precision) as
$$
select p.zone_type, sum(p.price) as sum_price
from
(select s.price, s.zone_type
from (select zone_type, round(((extract(hour from (time_call_end - time_call_start)) * 60 + extract(minute from (time_call_end - time_call_start)))
        * tariff_price) * (1-discount)) as price,
round(((extract(hour from (time_call_end - time_call_start)) * 60 + extract(minute from (time_call_end - time_call_start)))
        * country_price) * (1-discount)) as price2
from (((payments.call_outgoing natural left join payments.international_calls)
    natural join payments.subscription) natural left join payments.tariff)
    natural join payments.contract
where date_part('week', time_call_start) = date_part('week', current_timestamp) - 1) as s
```

```
where s.price2 is null
union all
select t.price2, t.zone_type
from (select zone_type, round(((extract(hour from (time_call_end - time_call_start)) * 60 + extract(minute from (time_call_end - time_call_start)))
        * tariff_price) * (1-discount)) as price,
round(((extract(hour from (time_call_end - time_call_start)) * 60 + extract(minute from (time_call_end - time_call_start)))
        * country_price) * (1-discount)) as price2
from (((payments.call_outgoing natural left join payments.international_calls)
    natural join payments.subscription) natural left join payments.tariff)
    natural join payments.contract
where date_part('week', time_call_start) = date_part('week', current_timestamp) - 1) as t
where t.price2 is not null) as p
group by zone_type
order by sum_price desc
$$ language sql;
```

```
telephone_exchange_db=# create function prev_week_zone_sum() returns table(zone_type character varying, sum_price double precision) as
telephone_exchange_db-# $$
telephone_exchange_db$# select p.zone_type, sum(p.price) as sum_price
telephone_exchange_db$# from
telephone_exchange_db$# (select s.price, s.zone_type
telephone_exchange_db$# from (select zone_type, round(((extract(hour from (time_call_end - time_call_start)) * 60 + extract(minute from (time_call_end - time_call_start
)))
telephone_exchange_db$#  * tariff_price) * (1-discount)) as price,
telephone_exchange_db$# round(((extract(hour from (time_call_end - time_call_start)) * 60 + extract(minute from (time_call_end - time_call_start)))
telephone_exchange_db$#  * country_price) * (1-discount)) as price2
telephone_exchange_db$# from (((payments.call_outgoing natural left join payments.international_calls)
telephone_exchange_db$#    natural join payments.subscription) natural left join payments.tariff)
telephone_exchange_db$#    natural join payments.contract
telephone_exchange_db$# where date_part('week', time_call_start) = date_part('week', current_timestamp) - 1) as s
telephone_exchange_db$# where s.price2 is null
telephone_exchange_db$# union all
telephone_exchange_db$# select t.price2, t.zone_type
telephone_exchange_db$# from (select zone_type, round(((extract(hour from (time_call_end - time_call_start)) * 60 + extract(minute from (time_call_end - time_call_start
)))
telephone_exchange_db$#  * tariff_price) * (1-discount)) as price,
telephone_exchange_db$# round(((extract(hour from (time_call_end - time_call_start)) * 60 + extract(minute from (time_call_end - time_call_start)))
telephone_exchange_db$#  * country_price) * (1-discount)) as price2
telephone_exchange_db$# from (((payments.call_outgoing natural left join payments.international_calls)
telephone_exchange_db$#    natural join payments.subscription) natural left join payments.tariff)
telephone_exchange_db$#    natural join payments.contract
telephone_exchange_db$# where date_part('week', time_call_start) = date_part('week', current_timestamp) - 1) as t
telephone_exchange_db$# where t.price2 is not null) as p
telephone_exchange_db$# group by zone_type
telephone_exchange_db$# order by sum_price desc
telephone_exchange_db$# $$ language sql;
CREATE FUNCTION
telephone_exchange_db=# select * from prev_week_zone_sum();
     zone_type      | sum_price
--------------------+-----------
 СНГ                |      2928
 дальнее зарубежье  |      1700
 страна             |      1080
 город              |       570
(4 строки)
```

Рисунок 3 – Результаты работы функции 2

Создание триггера для логирования событий:

Создадим таблицу с будущими логами, функцию триггера и сам триггер (рис. 4).

```
CREATE OR REPLACE FUNCTION payments.logging() RETURNS TRIGGER AS $$
DECLARE
   text_entry varchar(30);
   value_entry varchar(100);
   full_entry varchar(254);
BEGIN
   IF TG_OP = 'INSERT' THEN
      value_entry = NEW.call_id;
      text_entry := 'Added new call ';
      full_entry := text_entry || value_entry;
      INSERT INTO payments.logs(action_logged) values (full_entry);
      RETURN NEW;
   ELSIF TG_OP = 'UPDATE' THEN
      value_entry = NEW.call_id;
      text_entry := 'Updated call ';
      full_entry := text_entry || value_entry;
      INSERT INTO payments.logs(action_logged) values (full_entry);
      RETURN NEW;
   ELSIF TG_OP = 'DELETE' THEN
      value_entry = OLD.call_id;
      text_entry := 'Removed call ';
      full_entry := text_entry || value_entry;
      INSERT INTO payments.logs(action_logged) values (full_entry);
      RETURN OLD;
   END IF;
END;
```

$$ LANGUAGE plpgsql;

CREATE TRIGGER t_calls
AFTER INSERT OR UPDATE OR DELETE ON payments.call_outgoing FOR EACH ROW EXECUTE PROCEDURE payments.logging();

```
telephone_exchange_db=# create table payments.logs (
telephone_exchange_db(# id_log serial primary key,
telephone_exchange_db(# action_logged varchar(254) not null,
telephone_exchange_db(# time_logged timestamp without time zone DEFAULT current_timestamp
telephone_exchange_db(# );
CREATE TABLE
telephone_exchange_db=# CREATE OR REPLACE FUNCTION payments.logging() RETURNS TRIGGER AS $$
telephone_exchange_db$# DECLARE
telephone_exchange_db$#     text_entry varchar(30);
telephone_exchange_db$#     value_entry varchar(100);
telephone_exchange_db$#     full_entry varchar(254);
telephone_exchange_db$# BEGIN
telephone_exchange_db$#     IF    TG_OP = 'INSERT' THEN
telephone_exchange_db$#         value_entry = NEW.call_id;
telephone_exchange_db$#         text_entry := 'Added new call ';
telephone_exchange_db$#         full_entry := text_entry || value_entry;
telephone_exchange_db$#         INSERT INTO payments.logs(action_logged) values (full_entry);
telephone_exchange_db$#         RETURN NEW;
telephone_exchange_db$#     ELSIF TG_OP = 'UPDATE' THEN
telephone_exchange_db$#         value_entry = NEW.call_id;
telephone_exchange_db$#         text_entry := 'Updated call ';
telephone_exchange_db$#         full_entry := text_entry || value_entry;
telephone_exchange_db$#         INSERT INTO payments.logs(action_logged) values (full_entry);
telephone_exchange_db$#         RETURN NEW;
telephone_exchange_db$#     ELSIF TG_OP = 'DELETE' THEN
telephone_exchange_db$#         value_entry = OLD.call_id;
telephone_exchange_db$#         text_entry := 'Removed call ';
telephone_exchange_db$#         full_entry := text_entry || value_entry;
telephone_exchange_db$#         INSERT INTO payments.logs(action_logged) values (full_entry);
telephone_exchange_db$#         RETURN OLD;
telephone_exchange_db$#     END IF;
telephone_exchange_db$# END;
telephone_exchange_db$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
telephone_exchange_db=# CREATE TRIGGER t_calls
telephone_exchange_db-# AFTER INSERT OR UPDATE OR DELETE ON payments.call_outgoing FOR EACH ROW EXECUTE PROCEDURE payments.logging();
CREATE TRIGGER
```

Рисунок 4 – Создание триггера в консоли

Проверим работоспособность триггера (рис. 5-7).

```
telephone_exchange_db=# insert into payments.call_outgoing (subscription_id, time_call_start, time_call_end, zone_type, discount, status)
telephone_exchange_db-# values (4, current_timestamp - interval '10 minutes', current_timestamp, 'город', 0, 'ожидание');
INSERT 0 1
telephone_exchange_db=# select * from payments.logs;
 id_log |   action_logged    |        time_logged
--------+--------------------+---------------------------
      1 | Added new call 21 | 2022-06-23 08:17:08.750251
(1 строка)


telephone_exchange_db=# select * from payments.call_outgoing;
 call_id | subscription_id |      time_call_start       |       time_call_end        |     zone_type     | country_id | discount |  status   |    date_payment
---------+-----------------+----------------------------+----------------------------+-------------------+------------+----------+-----------+---------------------
       7 |              13 | 2022-06-14 14:00:00        | 2022-06-14 14:10:00        | дальнее зарубежье |          5 |      0.0 | оплачено  | 2022-06-14 14:15:00
       8 |              14 | 2022-06-14 14:00:00        | 2022-06-14 14:19:00        | город             |            |      0.0 | оплачено  | 2022-06-14 14:15:00
      11 |              10 | 2022-08-29 02:00:00        | 2022-08-29 02:05:00        | город             |            |      0.0 | ожидание  |
      12 |              13 | 2022-08-29 12:00:00        | 2022-08-29 12:25:00        | дальнее зарубежье |          5 |      0.0 | оплачено  | 2022-08-29 12:25:00
       1 |               1 | 2022-06-03 08:00:00        | 2022-06-03 08:10:00        | страна            |            |      0.0 | оплачено  | 2022-06-03 08:15:00
       2 |               4 | 2022-06-03 08:00:00        | 2022-06-03 08:30:00        | страна            |            |      0.0 | ожидание  |
       3 |               1 | 2022-06-05 21:14:00        | 2022-06-05 21:17:00        | страна            |            |      0.2 | оплачено  | 2022-06-05 21:18:00
       9 |               4 | 2022-06-14 17:54:00        | 2022-06-14 18:30:00        | страна            |            |      0.0 | оплачено  | 2022-06-14 19:00:00
      14 |               6 | 2022-09-23 11:17:00        | 2022-09-23 11:20:00        | страна            |            |      0.0 | оплачено  | 2022-09-23 11:21:00
      15 |               1 | 2022-09-25 20:00:00        | 2022-09-25 20:20:00        | страна            |            |      0.2 | оплачено  | 2022-09-25 20:23:00
       5 |               7 | 2022-06-13 20:00:00        | 2022-06-13 21:17:00        | СНГ               |          1 |      0.2 | ожидание  |
       6 |               7 | 2022-06-13 22:00:00        | 2022-06-13 22:45:00        | СНГ               |          1 |      0.2 | ожидание  |
      16 |               7 | 2022-09-25 20:00:00        | 2022-09-25 20:20:00        | СНГ               |          1 |      0.2 | ожидание  |
      19 |               4 | 2022-06-23 07:38:19.776346 | 2022-06-23 08:01:19.776346 | город             |            |        0 | ожидание  |
      20 |               4 | 2022-06-23 07:47:56.992725 | 2022-06-23 08:02:56.992725 | город             |            |        0 | ожидание  |
      21 |               4 | 2022-06-23 08:07:08.750251 | 2022-06-23 08:17:08.750251 | город             |            |        0 | ожидание  |
(16 строк)
```

Рисунок 5 – Проверка работы триггера на вставку записи

```
telephone_exchange_db=# delete from payments.call_outgoing where call_id in (19, 20);
DELETE 2
telephone_exchange_db=# select * from payments.logs;
 id_log |   action_logged    |        time_logged
--------+--------------------+---------------------------
      1 | Added new call 21 | 2022-06-23 08:17:08.750251
      2 | Removed call 19   | 2022-06-23 08:19:58.999224
      3 | Removed call 20   | 2022-06-23 08:19:58.999224
(3 строки)


telephone_exchange_db=# select * from payments.call_outgoing;
 call_id | subscription_id |      time_call_start       |       time_call_end        |     zone_type     | country_id | discount |  status  |    date_payment
---------+-----------------+----------------------------+----------------------------+-------------------+------------+----------+----------+--------------------
       7 |              13 | 2022-06-14 14:00:00        | 2022-06-14 14:10:00        | дальнее зарубежье |          5 |      0.0 | оплачено | 2022-06-14 14:15:00
       8 |              14 | 2022-06-14 14:00:00        | 2022-06-14 14:19:00        | город             |            |      0.0 | оплачено | 2022-06-14 14:15:00
      11 |              10 | 2022-08-29 02:00:00        | 2022-08-29 02:05:00        | город             |            |      0.0 | ожидание |
      12 |              13 | 2022-08-29 12:00:00        | 2022-08-29 12:25:00        | дальнее зарубежье |          5 |      0.0 | оплачено | 2022-08-29 12:25:00
       1 |               1 | 2022-06-03 08:00:00        | 2022-06-03 08:10:00        | страна            |            |      0.0 | оплачено | 2022-06-03 08:15:00
       2 |               4 | 2022-06-03 08:00:00        | 2022-06-03 08:30:00        | страна            |            |      0.0 | ожидание |
       3 |               1 | 2022-06-05 21:14:00        | 2022-06-05 21:17:00        | страна            |            |      0.2 | оплачено | 2022-06-05 21:18:00
       9 |               4 | 2022-06-14 17:54:00        | 2022-06-14 18:30:00        | страна            |            |      0.0 | оплачено | 2022-06-14 19:00:00
      14 |               6 | 2022-09-23 11:17:00        | 2022-09-23 11:20:00        | страна            |            |      0.0 | оплачено | 2022-09-23 11:21:00
      15 |               1 | 2022-09-25 20:00:00        | 2022-09-25 20:20:00        | страна            |            |      0.2 | оплачено | 2022-09-25 20:23:00
       5 |               7 | 2022-06-13 20:00:00        | 2022-06-13 21:17:00        | СНГ               |          1 |      0.2 | ожидание |
       6 |               7 | 2022-06-13 22:00:00        | 2022-06-13 22:45:00        | СНГ               |          1 |      0.2 | ожидание |
      16 |               7 | 2022-09-25 20:00:00        | 2022-09-25 20:20:00        | СНГ               |          1 |      0.2 | ожидание |
      21 |               4 | 2022-06-23 08:07:08.750251 | 2022-06-23 08:17:08.750251 | город             |            |        0 | ожидание |
(14 строк)
```

Рисунок 6 – Проверка работы триггера на удаление нескольких записей

```
telephone_exchange_db=# update payments.call_outgoing set status = 'оплачено', date_payment = current_timestamp where call_id = 21;
UPDATE 1
telephone_exchange_db=# select * from payments.logs;
 id_log |   action_logged    |        time_logged
--------+--------------------+---------------------------
      1 | Added new call 21 | 2022-06-23 08:17:08.750251
      2 | Removed call 19   | 2022-06-23 08:19:58.999224
      3 | Removed call 20   | 2022-06-23 08:19:58.999224
      4 | Updated call 21   | 2022-06-23 08:22:32.222872
(4 строки)


telephone_exchange_db=# select * from payments.call_outgoing;
 call_id | subscription_id |      time_call_start       |       time_call_end        |     zone_type     | country_id | discount |  status  |        date_payment
-
---------+-----------------+----------------------------+----------------------------+-------------------+------------+----------+----------+----------------------------
       7 |              13 | 2022-06-14 14:00:00        | 2022-06-14 14:10:00        | дальнее зарубежье |          5 |      0.0 | оплачено | 2022-06-14 14:15:00
       8 |              14 | 2022-06-14 14:00:00        | 2022-06-14 14:19:00        | город             |            |      0.0 | оплачено | 2022-06-14 14:15:00
      11 |              10 | 2022-08-29 02:00:00        | 2022-08-29 02:05:00        | город             |            |      0.0 | ожидание |
      12 |              13 | 2022-08-29 12:00:00        | 2022-08-29 12:25:00        | дальнее зарубежье |          5 |      0.0 | оплачено | 2022-08-29 12:25:00
       1 |               1 | 2022-06-03 08:00:00        | 2022-06-03 08:10:00        | страна            |            |      0.0 | оплачено | 2022-06-03 08:15:00
       2 |               4 | 2022-06-03 08:00:00        | 2022-06-03 08:30:00        | страна            |            |      0.0 | ожидание |
       3 |               1 | 2022-06-05 21:14:00        | 2022-06-05 21:17:00        | страна            |            |      0.2 | оплачено | 2022-06-05 21:18:00
       9 |               4 | 2022-06-14 17:54:00        | 2022-06-14 18:30:00        | страна            |            |      0.0 | оплачено | 2022-06-14 19:00:00
      14 |               6 | 2022-09-23 11:17:00        | 2022-09-23 11:20:00        | страна            |            |      0.0 | оплачено | 2022-09-23 11:21:00
      15 |               1 | 2022-09-25 20:00:00        | 2022-09-25 20:20:00        | страна            |            |      0.2 | оплачено | 2022-09-25 20:23:00
       5 |               7 | 2022-06-13 20:00:00        | 2022-06-13 21:17:00        | СНГ               |          1 |      0.2 | ожидание |
       6 |               7 | 2022-06-13 22:00:00        | 2022-06-13 22:45:00        | СНГ               |          1 |      0.2 | ожидание |
      16 |               7 | 2022-09-25 20:00:00        | 2022-09-25 20:20:00        | СНГ               |          1 |      0.2 | ожидание |
      21 |               4 | 2022-06-23 08:07:08.750251 | 2022-06-23 08:17:08.750251 | город             |            |        0 | оплачено | 2022-06-23 08:22:32.222872
(14 строк)
```

Рисунок 7 – Проверка работы триггера на редактирование данных

Выводы.

Я овладела практическими навыками создания и использования процедур, функций и триггеров в базе данных PostgreSQL.