Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по Лабораторной Работе № 5

по дисциплине **«Базы Данных»**

Автор: Акулов Даниил Даниилович

Факультет: ИКТ

Группа: К3239

Преподаватель: Говорова Марина Михайловна

Санкт-Петербург, 2023

Содержание работы

Цель работы:

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

Практическое задание:

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
2. Создать авторский триггер по варианту индивидуального задания.

Выполнение работы:

Создание хранимых процедур:

1. О текущей сумме вклада и сумме начисленного за месяц процента для заданного клиента.

```
create or replace procedure
get_deposit_payment(agreement_id integer)
language sql as
$$
update deposit_agreement da
set amountofpayments = (select coalesce(sum(dps.paymentamount), 0)
from deposit_payment_schedule dps
where dps.actualpaymentdate is not null
and da.id = dps.depositagreementid)
where da.id = agreement_id;
$$;
```

```
postgres=# create or replace procedure
postgres-# get_deposit_payment(agreement_id integer)
postgres-# language sql as
postgres-# $$
postgres$# update deposit_agreement da
postgres$# set amountofpayments = (select coalesce(sum(dps.paymentamount), 0)
postgres$# from deposit_payment_schedule dps
postgres$# where dps.actualpaymentdate is not null
postgres$# and da.id = dps.depositagreementid)
postgres$# where da.id = agreement_id;
postgres$# $$
postgres-#
postgres-# ;
CREATE PROCEDURE
```

```
postgres=# select id, amountofpayments from deposit_agreement;
 id | amountofpayments
----+------------------
  2 |                0
  1 |                0
(2 строки)


postgres=# call get_deposit_payment(2);
CALL
postgres=# select id, amountofpayments from deposit_agreement;
 id | amountofpayments
----+------------------
  1 |                0
  2 |            15000
(2 строки)
```

2. Найти клиента банка, имеющего максимальное количество кредитов на текущий день.

```
create or replace function
get_biggest_count_credit()
returns table(number bigint)
language plpgsql as
$$
begin
 return query
 select cast(c.passportid as bigint) from client c join (select ca.clientid
 from credit_agreement ca
 where dateofclosing is null
 group by clientid
 having count(ca.*) =
 (select max(cnt) from
 (select count(ca.*) as cnt
 from credit_agreement ca
 group by ca.clientid))) as ca on ca.clientid = c.id;
end;
$$;
```

```
postgres=# create or replace function
postgres-# get_biggest_count_credit()
postgres-# returns table(number bigint)
postgres-# language plpgsql as
postgres-# $$
postgres$# begin
postgres$#  return query
postgres$#  select cast(c.passportid as bigint) from client c join (select ca.clientid
postgres$#  from credit_agreement ca
postgres$#  where dateofclosing is null
postgres$#  group by clientid
postgres$#  having count(ca.*) =
postgres$#  (select max(cnt) from
postgres$#  (select count(ca.*) as cnt
postgres$#  from credit_agreement ca
postgres$#  group by ca.clientid))) as ca on ca.clientid = c.id;
postgres$# end;
postgres$# $$;
CREATE FUNCTION
```

```
postgres=# select * from get_biggest_count_credit();
   number
------------
 1914323849
 2809346819
(2 строки)
```

3. Найти клиентов банка, не имеющих задолженности по кредитам.

```
create or replace function
get_good_credit_clients()
returns table(passportid bigint)
language plpgsql
as
$$
begin
 return query
 select cast(c.passportid as bigint) from client c join (select distinct ca.clientid
 from credit_agreement ca
 where ca.clientid not in
 (select distinct ca.clientid
 from credit_agreement ca
 join credit_payment_schedule dps
 on ca.id=dps.creditagreementid
 where dps.actualpaymentdate is null
 and dps.plannedpaymentdate < current_date)) as ca on ca.clientid = c.id;
end;
$$;
```

```
postgres=# create or replace function
postgres-# get_good_credit_clients()
postgres-# returns table(passportid bigint)
postgres-# language plpgsql
postgres-# as
postgres-# $$
postgres$# begin
postgres$#  return query
postgres$#  select cast(c.passportid as bigint) from client c join (select distinct ca.clientid
postgres$#  from credit_agreement ca
postgres$#  where ca.clientid not in
postgres$#  (select distinct ca.clientid
postgres$#  from credit_agreement ca
postgres$#  join credit_payment_schedule dps
postgres$#  on ca.id=dps.creditagreementid
postgres$#  where dps.actualpaymentdate is null
postgres$#  and dps.plannedpaymentdate < current_date)) as ca on ca.clientid = c.id;
postgres$# end;
postgres$# $$;
CREATE FUNCTION
```

```
postgres=# select * from get_good_credit_clients();
 passportid
------------
 2809346819
```

Создание авторского триггера:

Триггер обновляет сумму выплат по вкладу при каждом обновлении таблицы с расписанием платежей:

```
create or replace function count_total_deposit_payment()
returns trigger as $$
begin
call get_deposit_payment(new.id);
return new;
end;
$$ language plpgsql;

create or replace trigger change_total_deposit_payment
after update on deposit_payment_schedule
for each row execute function count_total_deposit_payment();
```

```
postgres=# create or replace function count_total_deposit_payment()
postgres-# returns trigger as $$
postgres$# begin
postgres$# call get_deposit_payment(new.id);
postgres$# return new;
postgres$# end;
postgres$# $$ language plpgsql;
CREATE FUNCTION
postgres=# create or replace trigger change_total_deposit_payment
postgres-# after update on deposit_payment_schedule
postgres-# for each row execute function count_total_deposit_payment();
CREATE TRIGGER
```

```
postgres=# select id, amountofpayments from deposit_agreement;
 id | amountofpayments
----+------------------
  2 |            15000
  1 |                0
(2 строки)


postgres=# update deposit_payment_schedule set actualpaymentdate = current_date;
UPDATE 6
postgres=# select id, amountofpayments from deposit_agreement;
 id | amountofpayments
----+------------------
  1 |            10500
  2 |            15000
(2 строки)
```

Вывод

В ходе лабораторной работы были освоены практические навыки по созданию, функций в PostgreSQL с использованием инструмента управления pgAdmin 4 и PSQL tool. Были созданы функции на выборку данных, а также были созданы необходимые триггеры.