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

Отчёт

по лабораторной работе №5 «Процедуры, функции и триггеры в PostgreSQL»

По дисциплине **«Проектирование и реализация баз данных»**

Автор: Сергеев В. Ю.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М. М.

Санкт-Петербург, 2023

Оглавление

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
2. Модифицировать триггер на проверку корректности входа и выхода сотрудника с максимальным учётом «узких» мест некорректных данных по входу и выходу
3. Создать авторский триггер по варианту индивидуального задание

Вариант 19. БД «Банк»

Описание предметной области: Система обеспечивает работу с вкладами и кредитами клиентов банка.

Клиенты банка имеют вклады и кредиты различных видов. Для вкладов и кредитов может использоваться различная валюта.

Сотрудники банка заключают договоры с клиентами. Фиксируется сотрудник, заключивший договор.

Ежемесячно начисляется процент по вкладу, и полученная сумма добавляется к сумме вклада заказчика. Вкладчик имеет право снимать проценты по вкладу или всю сумму вклада с процентами по истечении срока вклада. При снятии денег до истечения срока вклада процент за текущий месяц не начисляется.

Кредит выдается на определенный срок. Формируется график выплат, который получает клиент при заключении договора. Хранится информация по своевременности ежемесячных выплат.

БД должна содержать следующий минимальный набор сведений: ФИО сотрудника. Возраст сотрудника. Адрес сотрудника. № телефона сотрудник. Паспортные данные сотрудника. Должность сотрудника. Оклад сотрудника (зависит от категории). Наименование вклада. Описание вклада. Минимальный срок вклада. Минимальная сумма вклада. Процент по вкладу. Срок вклада. Процентная ставка. Код валюты. Наименование валюты. ФИО вкладчика. Адрес вкладчика. Телефон вкладчика. E-mail вкладчика. Паспортные данные. Номер договора. Дата вклада. Дата возврата. Сумма вклада. Сумма возврата. Данные по кредиту.

Задание 4. Создать хранимые процедуры:

- о текущей сумме вклада и сумме начисленного за месяц процента для заданного клиента;
- найти клиента банка, имеющего максимальное количество кредитов на текущий день;
- найти клиентов банка, не имеющих задолженности по кредитам.

Задание 5. Создать необходимые триггеры.

Выполнение

Процедуры

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

```
create or replace procedure
"bankDB".get_deposit_payment(agreement_number integer)
language sql as
$$
update "bankDB"."DepositAgreement" da
set "SummaryPayment" = "StartSum" +
(select coalesce(sum("Payment"), 0)
 from "bankDB"."DepositPaySchedule" dps
 where not ("FactPaymentDate" is null)
 and da."AgreementNumber" = dps."AgreementNumber")
where da."AgreementNumber" = agreement_number;
$$
```

```
Bank=#
Bank=# create or replace procedure
Bank-# "bankDB".get_deposit_payment(agreement_number integer)
Bank-# language sql as
Bank-# $$
Bank$# update "bankDB"."DepositAgreement" da
Bank$# set "SummaryPayment" = "StartSum" +
Bank$# (select coalesce(sum("Payment"), 0)
Bank$#  from "bankDB"."DepositPaySchedule" dps
Bank$#  where not ("FactPaymentDate" is null)
Bank$#  and da."AgreementNumber" = dps."AgreementNumber")
Bank$# where da."AgreementNumber" = agreement_number;
Bank$# $$
Bank-# ;
CREATE PROCEDURE
Bank=#
```

```
Bank=# select "AgreementNumber", "SummaryPayment" from "bankDB"."DepositAgreement";
 AgreementNumber | SummaryPayment
-----------------+----------------
              21 |        2353.83
              22 |        2123.04
              24 |             0
              25 |             0
              26 |             0
              27 |             0
              39 |             0
              23 |             0
(8 строк)


Bank=# update "bankDB"."DepositPaySchedule" dps
Bank-# set "FactPaymentDate" = current_date
Bank-# where "AgreementNumber" = 23;
UPDATE 24
Bank=# call "bankDB".get_deposit_payment(23);
CALL
Bank=# select "AgreementNumber", "SummaryPayment" from "bankDB"."DepositAgreement";
 AgreementNumber | SummaryPayment
-----------------+----------------
              21 |        2353.83
              22 |        2123.04
              24 |             0
              25 |             0
              26 |             0
              27 |             0
              39 |             0
              23 |     250137.72
(8 строк)


Bank=#
```

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

```
create or replace function
"bankDB".get_biggest_loaner()
returns table(number bigint)
language plpgsql
as
$$
begin
    return query
    (select cast(la."PassportNumber" as bigint)
    from "bankDB"."LoanAgreement" la
    where "Status" = 'Open'
    group by "PassportNumber"
```

```
    having count(la.*) =
    (select max(tbl.counter) from
    (select count(la.*) as counter
    from "bankDB"."LoanAgreement" la
    group by la."PassportNumber") tbl));
end;
$$;
```

```
Bank=#
Bank=# create or replace function
Bank-# "bankDB".get_biggest_loaner()
Bank-# returns table(number bigint)
Bank-# language plpgsql
Bank-# as
Bank-# $$
Bank$# begin
Bank$# return query
Bank$# (select cast(la."PassportNumber" as bigint)
Bank$# from "bankDB"."LoanAgreement" la
Bank$# where "Status" = 'Open'
Bank$# group by "PassportNumber"
Bank$# having count(la.*) =
Bank$# (select max(tbl.counter) from
Bank$# (select count(la.*) as counter
Bank$# from "bankDB"."LoanAgreement" la
Bank$# group by la."PassportNumber") tbl));
Bank$# end;
Bank$# $$;
CREATE FUNCTION
```

```
Bank=# select * from "bankDB".get_biggest_loaner();
  number
-----------
 324629513
 129798791
 121748022
 123609496
  51690385
 571140644
(6 строк)


Bank=#
```

Найти клиентов банка, не имеющих задолженности по кредитам.

```
create or replace function
"bankDB".get_loyal_loaners()
returns table(passport bigint)
language plpgsql
as
$$
begin
    return query
    (select cast(la."PassportNumber" as bigint)
    from "bankDB"."LoanAgreement" la
    where la."PassportNumber" not in
     (select distinct la."PassportNumber"
     from "bankDB"."LoanAgreement" la
     join "bankDB"."LoanPaySchedule" lps
     on la."AgreementNumber"=lps."AgreementNumber"
     where
     (lps."FactPaymentDay" is null
      and lps."PlanPaymentDate" < current_date
      and la."PassportNumber" is not null)));
end;
$$;
```

```
Bank=#
Bank=# create or replace function
Bank-# "bankDB".get_loyal_loaners()
Bank-# returns table(passport bigint)
Bank-# language plpgsql
Bank-# as
Bank-# $$
Bank$# begin
Bank$# return query
Bank$# (select cast(la."PassportNumber" as bigint)
Bank$# from "bankDB"."LoanAgreement" la
Bank$# where la."PassportNumber" not in
Bank$#  (select distinct la."PassportNumber"
Bank$#  from "bankDB"."LoanAgreement" la
Bank$#  join "bankDB"."LoanPaySchedule" lps
Bank$#  on la."AgreementNumber"=lps."AgreementNumber"
Bank$#  where
Bank$#  (lps."FactPaymentDay" is null
Bank$#   and lps."PlanPaymentDate" < current_date
Bank$#   and la."PassportNumber" is not null)));
Bank$# end;
Bank$# $$;
CREATE FUNCTION
```

```
Bank=# select * from "bankDB".get_loyal_loaners();
 passport
----------
(0 строк)


Bank=# insert into "bankDB"."LoanAgreement"(
Bank(# "LoanDate", "PaymentDay", "PlanCloseDate", "StartSum", "Debt", "TabelNumber", "PassportNumber", "IdCurrency", "TypeId")
Bank-# values ('2023-02-20', 9, '2025-02-20', '154000', '154000', 100015, 856093670, 2, 6);
INSERT 0 1
Bank=# select "bankDB".get_loyal_loaners() as number;
  number
-----------
 856093670
(1 строка)


Bank=#
```

Модификация триггера

В практическом задании была продемонстрирована работа следующей триггерной функции:

```
create or replace function fn_check_time_punch() returns trigger as $psql$
    begin
    if new.is_out_punch = (
           select tps.is_out_punch
           from time_punch tps
           where tps.employee_id = new.employee_id
           order by tps.id desc limit 1
         )
        then return null;
         end if;
         return new;
    end;
$psql$ language plpgsql;
```

Однако данная функция имеет несколько уязвимостей:

- Время входа или выхода может быть некорректным: время последнего панча меньше, чем предыдущего; или время последнего выхода меньше, чем время последнего входа, что порождает отрицательную разницу;
- Вбитое время может быть больше настоящего;

Эти проблемы решаются модификацией имеющейся функции:

```
create or replace function fn_check_time_punch() returns trigger as $psql$
    begin
    if new.is_out_punch = (
           select tps.is_out_punch
           from time_punch tps
           where tps.employee_id = new.employee_id
           order by tps.id desc limit 1)
          or new.punch_time <=
          (select tps.punch_time
           from time_punch tps
           where tps.employee_id = new.employee_id
           order by tps.id desc limit 1)
          or new.punch_time > now()
        then return null;
         end if;
         return new;
    end;
$psql$ language plpgsql;
```

```
emp_time=# create or replace function fn_check_time_punch() returns trigger as $psql$
emp_time$#     begin
emp_time$# if new.is_out_punch = (
emp_time$#           select tps.is_out_punch
emp_time$#           from time_punch tps
emp_time$#           where tps.employee_id = new.employee_id
emp_time$#           order by tps.id desc limit 1)
emp_time$#   or new.punch_time <=
emp_time$#   (select tps.punch_time
emp_time$#           from time_punch tps
emp_time$#           where tps.employee_id = new.employee_id
emp_time$#           order by tps.id desc limit 1)
emp_time$#   or new.punch_time > now()
emp_time$# then return null;
emp_time$#         end if;
emp_time$#         return new;
emp_time$#     end;
emp_time$# $psql$ language plpgsql;
CREATE FUNCTION
emp_time=#
emp_time=# create or replace trigger check_time_punch before insert on time_punch
emp_time-# for each row execute procedure fn_check_time_punch();
CREATE TRIGGER
```

```
emp_time=# select * from time_punch
emp_time-# ;
 id | employee_id | is_out_punch |         punch_time
----+-------------+--------------+----------------------------
  1 |           1 | f            | 2021-01-01 10:10:00
  2 |           1 | t            | 2021-01-01 11:40:00
  4 |           1 | f            | 2021-01-01 11:50:00
  6 |           1 | t            | 2021-01-01 11:54:00
  7 |           1 | f            | 2020-01-01 12:00:00
  8 |           1 | t            | 2019-01-01 12:00:00
 10 |           1 | f            | 2023-11-24 00:00:00
 11 |           1 | t            | 2023-11-24 00:00:00
 13 |           1 | f            | 2023-11-24 16:37:39.979481
 14 |           2 | t            | 2023-11-24 16:41:08.931541
 16 |           2 | f            | 2023-11-24 16:53:03.755275
 17 |           2 | t            | 2023-11-24 16:53:57.80578
 21 |           1 | t            | 2023-11-01 12:00:00
(13 строк)
emp_time=# insert into time_punch(employee_id, is_out_punch, punch_time)
emp_time-# values(1, false, '2025-01-01 12:00:00'),
emp_time-# (1, true, '2023-12-01 12:00:00'),
emp_time-# (1, false, '2023-12-01 12:00:00'),
emp_time-# (1, true, '2023-11-01 12:00:00'),
emp_time-# (1, true, '2023-12-01 12:00:00'),
emp_time-# (1, true, '2023-12-02 12:00:00');
INSERT 0 2
emp_time=# select * from time_punch;
 id | employee_id | is_out_punch |         punch_time
----+-------------+--------------+----------------------------
  1 |           1 | f            | 2021-01-01 10:10:00
  2 |           1 | t            | 2021-01-01 11:40:00
  4 |           1 | f            | 2021-01-01 11:50:00
  6 |           1 | t            | 2021-01-01 11:54:00
  7 |           1 | f            | 2020-01-01 12:00:00
  8 |           1 | t            | 2019-01-01 12:00:00
 10 |           1 | f            | 2023-11-24 00:00:00
 11 |           1 | t            | 2023-11-24 00:00:00
 13 |           1 | f            | 2023-11-24 16:37:39.979481
 14 |           2 | t            | 2023-11-24 16:41:08.931541
 16 |           2 | f            | 2023-11-24 16:53:03.755275
 17 |           2 | t            | 2023-11-24 16:53:57.80578
 21 |           1 | t            | 2023-11-01 12:00:00
 26 |           1 | f            | 2023-12-01 12:00:00
 29 |           1 | t            | 2023-12-02 12:00:00
(15 строк)
```

Свой триггер

У нас есть уже написана выше процедура, которая считает текущую выплату по вкладу для конкретного клиента. Напишем триггер, который пересчитывает сумму и каждом обновлении таблицы с выплатами.

Триггерная функция:

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

Триггер:

```
create or replace trigger change_total_deposit_payment
after update on "bankDB"."DepositPaySchedule"
for each row execute function "bankDB".count_total_deposit_payment();
```

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

```
Bank=# select "AgreementNumber", "SummaryPayment" from "bankDB"."DepositAgreement";
 AgreementNumber | SummaryPayment
-----------------+----------------
              22 |        2123.04
              24 |             0
              25 |             0
              26 |             0
              27 |             0
              39 |             0
              23 |      250137.72
              21 |        107229
(8 строк)

Bank=# update "bankDB"."DepositPaySchedule"
Bank-# set "FactPaymentDate" = current_date
Bank-#
Bank-# where "AgreementNumber" = 25;
UPDATE 24
Bank=# select "AgreementNumber", "SummaryPayment" from "bankDB"."DepositAgreement";
 AgreementNumber | SummaryPayment
-----------------+----------------
              22 |        2123.04
              24 |             0
              26 |             0
              27 |             0
              39 |             0
              23 |      250137.72
              21 |        107229
              25 |       178177.1
(8 строк)
```

Вывод

В ходе лабораторной работы были написаны функции и триггеры для собственной базы данных в PostgreSQL. Была изучена чужая триггерная функция и исправлены некоторые уязвимости.