Университет ИТМО

Факультет инфокоммуникационных технологий

Дисциплина:
«ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ БАЗ ДАННЫХ»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«Процедуры, функции, триггеры в PostgreSQL»

Вариант 19

Специальность:
09.03.03 Мобильные и сетевые технологии

Выполнила:
Студентка группы K3240

Вахрушева К.А

Проверила:
Говорова М.М.

Дата: «__» ____ 2022 г.
Оценка _____

Санкт-Петербург
2022г.

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

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.

База данных: «Банк»

Выполнение:

Задание 4. Создать хранимые процедуры:

- о текущей сумме вклада и сумме начисленного за месяц процента для заданного клиента;

```
call deposit_sum_and_last_percent(1);
select dc.return_sum as "Текущая сумма", dc.last_percent as "Последняя выплата" from "default"."Deposit_contract" dc;


create or replace procedure deposit_sum_and_last_percent(arg_deposit_id int)
language plpgsql as $$
begin
    update "default"."Deposit_contract"
    set last_percent = ((return_sum - deposit_sum) / (date_part('month', return_date::date) -
    date_part('month', deposit_date::date)) / interest * 0.01)/nullif(date_part('month', return_date::date) -
    date_part('month', deposit_date::date), 0)
where depositor_id = arg_deposit_id;
commit;
end;
$$;
```

posit_contract 1

ect dc.return_sum as "Текущая сумма", dc.last_perc — Введите SQL выражение чтобы отфильтровать результаты

Текущая сумма	Последняя выплата
170 000	1 000
250 000	2 500
350 000	3 000
350 000	2 000
700 000	10 000
700 000	7 000

- найти клиента банка, имеющего максимальное количество кредитов на текущий день;

```
create or replace function max_credits()
returns table (id integer, name varchar[], total bigint)
language plpgsql as $$
begin
    return query (
    select dep.depositor_id, dep."name", count(*) as "Кредиты" from "default"."Depositor" dep
    left join "default"."Credit_contract" cc on cc.depositor_id = dep.depositor_id
    group by dep.depositor_id  having count(*) = (select count(*) from "default"."Credit_contract" cco group by depositor_id order by count(*) desc limit 1)
    );
end;
$$;

select max_credits();
```

Результат 1

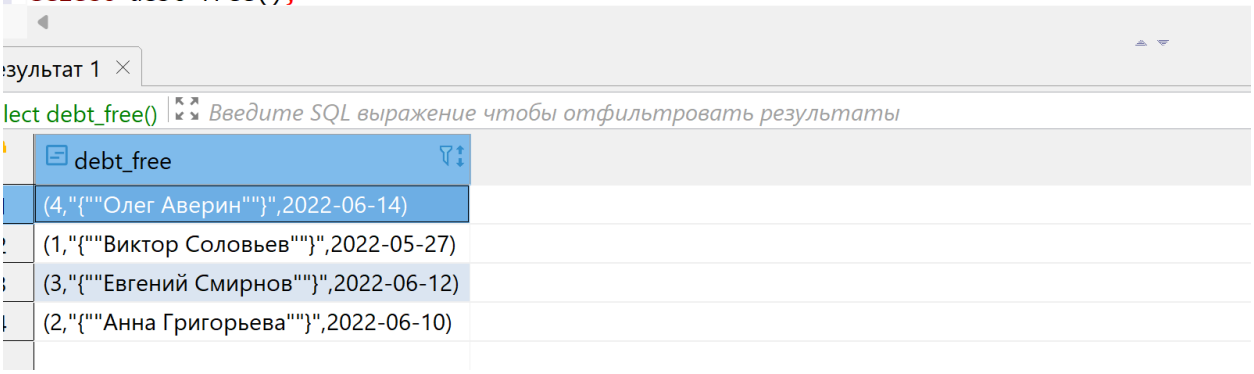
select max_credits() — Введите SQL выражение чтобы отфильтровать результаты

	max_credits
1	(2,"{""Анна Григорьева""}",3)

- найти клиентов банка, не имеющих задолженности по кредитам.

```sql
create or replace function debt_free()
returns table (id integer, firstname varchar[], pay_date date) language plpgsql as $$
begin
    return query (
    select dep.depositor_id , dep."name", p.pay_date from "default"."Depositor" dep
    join "default"."Credit_contract" cc on cc.depositor_id = dep.depositor_id
    join "default"."Payment" p on p.credit_cont_id = cc.credit_cont_id
    group by dep.depositor_id, p.pay_date
    );
end;
$$;

select debt_free();
```



Задание 5. Создать необходимые триггеры.

Триггер для автоматического добавления даты при создании новой строки в таблице договоров по кредитам.
Ограничения на null отключены для наглядности примера.

```sql
CREATE FUNCTION edit_credit_contract_date()
RETURNS trigger AS $$
begin
    update "default"."Credit_contract"
   set credit_date = current_date where credit_cont_id = new.credit_cont_id;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER edit_credit_cont
BEFORE insert or UPDATE ON "default"."Credit_contract"
FOR EACH ROW execute function edit_credit_contract_date();

insert into "default"."Credit_contract" (credit_cont_id) values (10);
select * from "default"."Credit_contract" cc ;
```

redit_contract 1

```
elect * from "default"."Credit_contract" cc
```

Введите SQL выражение чтобы отфильтровать результаты

	credit_cont_id	credit_date	repay_date	credit_sum	repay_sum	comment	worker_id	depositor_id	credit_id
1	1	2022-04-19	2022-10-20	100 000	120 000	{}	1	2	1
2	2	2022-04-15	2022-10-20	70 000	80 000	{}	1	2	2
3	3	2022-05-15	2022-10-20	55 000	60 000	{}	1	3	3
4	4	2022-05-10	2023-01-10	70 000	100 000	{}	2	1	2
5	5	2022-03-12	2022-04-12	100 000	10 000	{}	1	2	4
6	6	2022-05-16	2023-05-16	1 000 000	1 000 250	NULL	3	4	5
7	7	2022-05-10	2023-07-10	1 000 000	1 000 270	NULL	2	3	6
8	10	2022-06-15	[NULL]	[NULL]	[NULL]	NULL	[NULL]	[NULL]	[NULL]

Триггер, который отрабатывает перед удалением строки из таблицы Deposit. При удалении вклада в контракте сумма автоматически увеличивается на 5%.

```sql
CREATE OR REPLACE FUNCTION increase_deposit() RETURNS trigger AS
$$
BEGIN
   UPDATE "default"."Deposit_contract" SET return_sum = return_sum + return_sum * 0.05
      WHERE old.deposit_id = deposit_id;
   RETURN OLD;
END;$$ LANGUAGE plpgsql;

CREATE TRIGGER before_delete_deposit
   AFTER DELETE ON "default"."Deposit" FOR EACH ROW
   EXECUTE PROCEDURE increase_deposit();

delete from "default"."Deposit" where deposit_id = 3;
select * from "default"."Deposit_contract" dc where deposit_id = 3;
```

Deposit_contract 1

```
elect * from "default"."Deposit_contract" dc where de
```

Введите SQL выражение чтобы отфильтровать результаты

	deposit_cont_id	deposit_date	return_date	deposit_sum	return_sum	comment	worker_id	depositor_id	deposit_id	last_percent
1	4	2022-05-01	2025-01-05	300 000	367 500	NULL	1	3	3	2 000

Вывод: мной были получены навыки работы с хранимыми процедурами и триггерами в PostgreSQL.