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

Отчет

по лабораторной работе №3 «Процедуры, функции, триггеры в PostgreSQL»

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

Автор: Афонина Н.Р.

Факультет: ИКТ

Группа: К32421

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

Цель работы

Овладеть практическими 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.

Вариант 19. БД «Банк»

Создать хранимые процедуры:

- о текущей сумме вклада и сумме начисленного за месяц процента для заданного клиента (в моем случае процента, который будет выплачен в конце срока вклада, так как у меня все вклады такого типа);

```
CREATE OR REPLACE FUNCTION get_balance_and_interest_saving(p_id_client VARCHAR)
RETURNS TABLE(start_amount_saving_contract INT, interest_payment_saving INT)
AS $$
BEGIN
RETURN QUERY SELECT Sc.start_amount_saving_contract, Ssc.interest_payment_saving
FROM "Bank_schema".saving_contract AS Sc
JOIN "Bank_schema".schedule_saving_contract AS Ssc
ON Sc.id_saving_contract=Ssc.id_saving_contract
WHERE Sc.id_client = p_id_client;
END;
$$ LANGUAGE plpgsql;
```

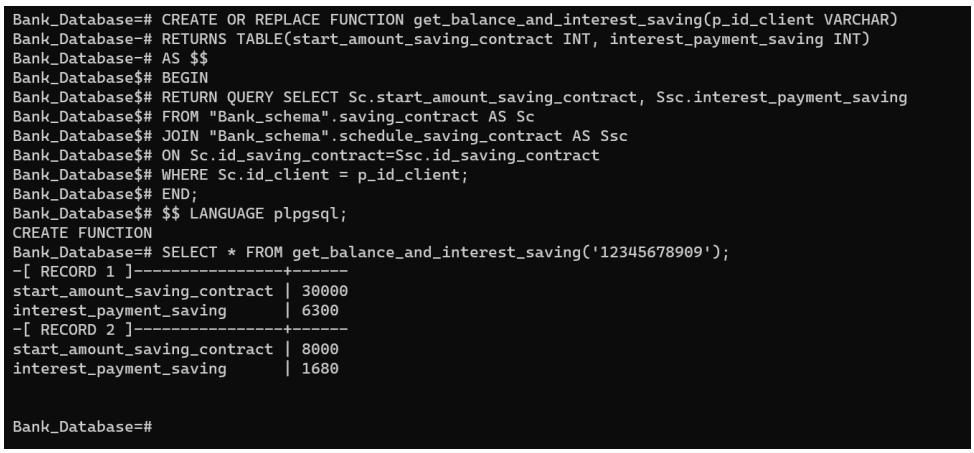

Рис. 1 – «Работа первой функции»

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

```
CREATE OR REPLACE FUNCTION get_client_with_max_loans()

RETURNS TABLE(name_client VARCHAR, number_of_loans INT) AS $$

BEGIN

RETURN QUERY

SELECT Cl.name_client, COUNT (Lc.id_client)::INT AS number_of_loans

FROM "Bank_schema".client Cl JOIN "Bank_schema".loan_contract Lc ON Cl.id_client=Lc.id_client

WHERE Lc.startdate_loan_contract <= CURRENT_DATE AND Lc.enddate_loan_contract >= CURRENT_DATE

GROUP BY Cl.name_client

HAVING COUNT (Lc.id_client) = (SELECT MAX(subquery.loan_count) FROM (SELECT COUNT(id_client) AS loan_count FROM "Bank_schema".loan_contract GROUP BY id_client) AS subquery);

END;

$$ LANGUAGE plpgsql;
```

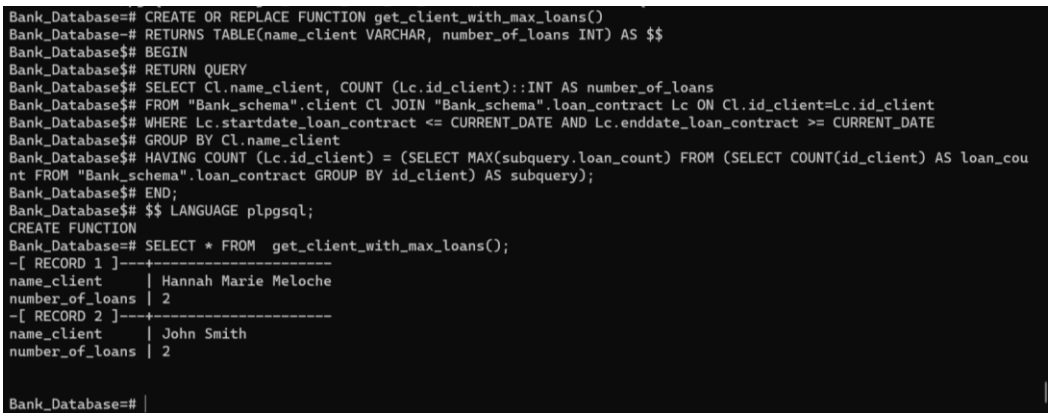

Рис. 2 – «Работа второй функции»

- найти клиентов банка, не имеющих задолженности по кредитам.

```
CREATE OR REPLACE FUNCTION get_clients_with_no_debt()
RETURNS TABLE(name_client VARCHAR)
AS $$
BEGIN
RETURN QUERY
SELECT DISTINCT Cl.name_client
FROM (("Bank_schema".client Cl JOIN "Bank_schema".loan_contract Lc ON Cl.id_client=Lc.id_client) JOIN "Bank_schema".schedule_loan_contract Slc ON Lc.id_loan_contract=Slc.id_loan_contract)
WHERE state_payment_loan = true;
END;
$$ LANGUAGE plpgsql;
```

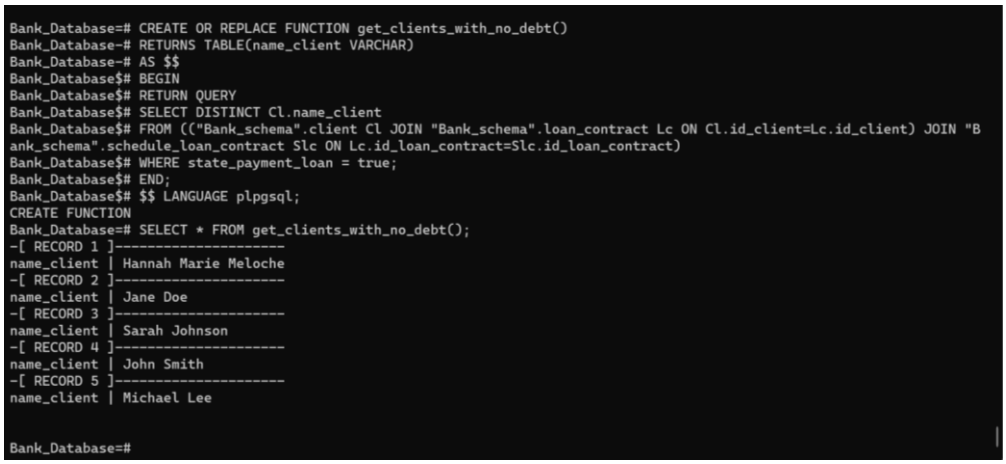

Рис. 3 – «Работа третьей функции»

Создать необходимые триггеры.

- Триггер на логирование вставки данных в таблицу Client

```
CREATE OR REPLACE FUNCTION log_client_insert()
RETURNS TRIGGER AS $$
BEGIN
INSERT INTO "Bank_schema".client_log(action_type, client_id, name_client, email_client, log_date)
VALUES ('INSERT', NEW.id_client, NEW.name_client, NEW.email_client, CURRENT_TIMESTAMP);
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER client_insert_trigger
AFTER INSERT ON "Bank_schema".client
FOR EACH ROW
EXECUTE FUNCTION log_client_insert();
```

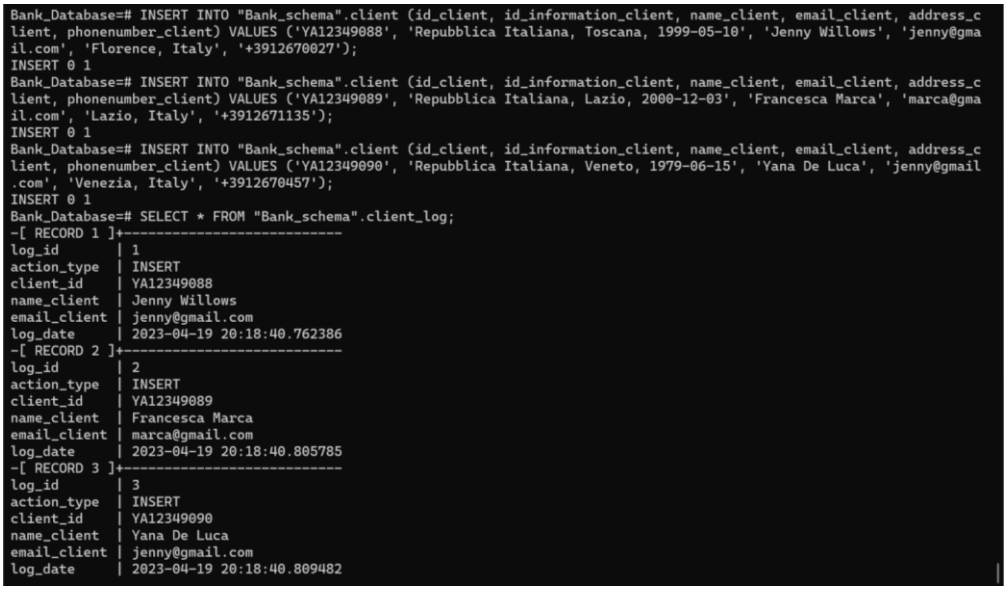

Рис. 4 – «Работа триггера на insert»

- Триггер на логирование обновления данных в таблице Client

```
CREATE OR REPLACE FUNCTION log_client_update()
RETURNS TRIGGER AS $$
BEGIN
```

```
INSERT INTO "Bank_schema".client_log(action_type, client_id, name_client, email_client,
log_date)
VALUES ('UPDATE', OLD.id_client, NEW.name_client, NEW.email_client,
CURRENT_TIMESTAMP);
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER client_update_trigger
AFTER UPDATE ON "Bank_schema".client
FOR EACH ROW
EXECUTE FUNCTION log_client_update();
```

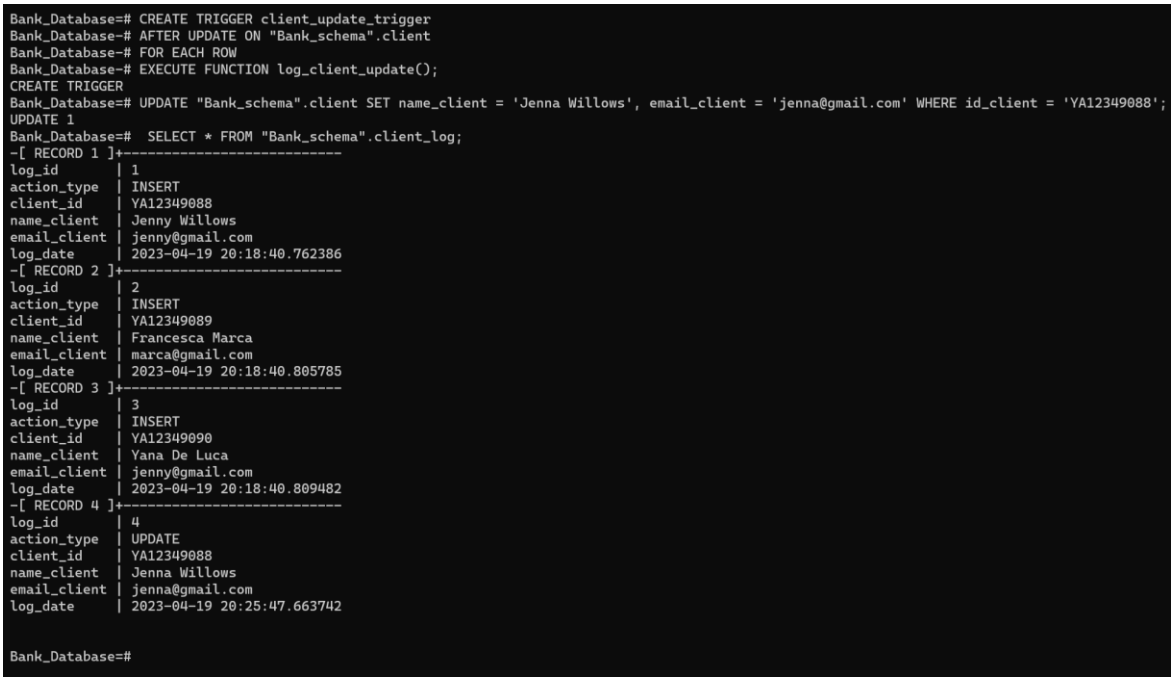

Рис. 5 – «Работа триггера на update»

- Триггер на логирование удаления данных в таблице Client

```
CREATE OR REPLACE FUNCTION log_client_delete()
RETURNS TRIGGER AS $$
BEGIN
   INSERT INTO "Bank_schema".client_log(action_type, client_id, name_client, email_client,
log_date)
   VALUES ('DELETE', OLD.id_client, OLD.name_client, OLD.email_client,
CURRENT_TIMESTAMP);
```

```
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER client_delete_trigger
AFTER DELETE ON "Bank_schema".client
FOR EACH ROW
EXECUTE FUNCTION log_client_delete();
```

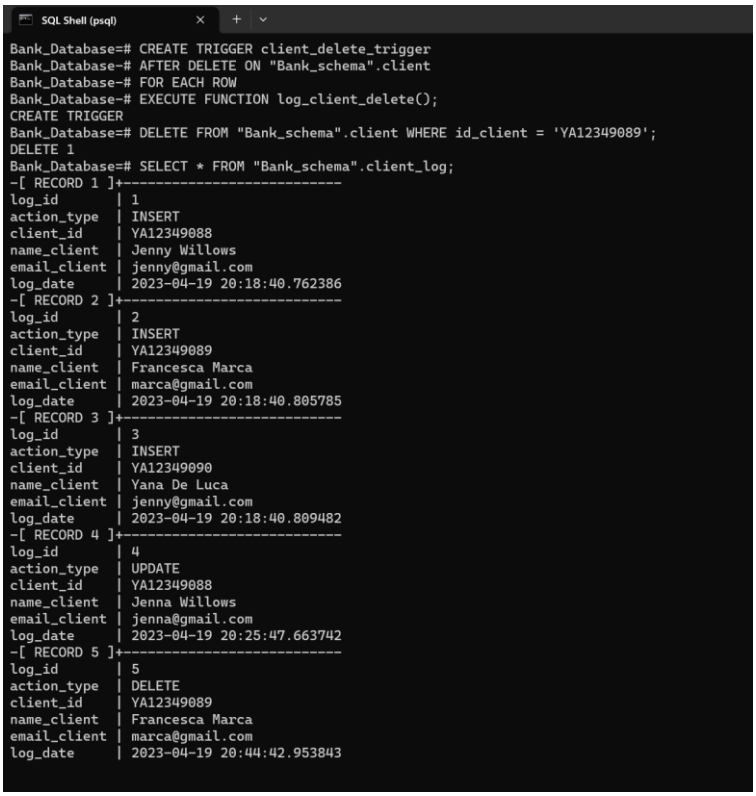

Рис. 6 – «Работа триггера на delete»

- Триггер, который автоматически рассчитывает окончательную сумму для кредитного договора на основе суммы кредиты, процентной ставки и срока кредита.

```
CREATE OR REPLACE FUNCTION calculate_loan_payment()
RETURNS TRIGGER AS $$
DECLARE
    loan_amount INTEGER;
    interest_rate NUMERIC;
    loan_term VARCHAR;
```

```
        loan_term_int INTEGER;
    monthly_rate NUMERIC;
    monthly_payment NUMERIC;
BEGIN

    loan_amount := NEW.amount_loan_contract;
    SELECT loan.interest_loan INTO interest_rate FROM "Bank_schema".loan loan JOIN
"Bank_schema".loan_contract contract ON loan.id_loan = contract.id_loan WHERE
contract.id_loan_contract = NEW.id_loan_contract;
    SELECT loan.terms_loan INTO loan_term FROM "Bank_schema".loan loan JOIN
"Bank_schema".loan_contract contract ON loan.id_loan = contract.id_loan WHERE
contract.id_loan_contract = NEW.id_loan_contract;

        loan_term_int := substring(regexp_replace(loan_term, '[^\d]', '', 'g'), '\d+')::INTEGER;
        monthly_rate := interest_rate / 12;
    monthly_payment := (loan_amount * monthly_rate) / (1 - POWER(1 + monthly_rate, -
loan_term_int * 12));



    UPDATE "Bank_schema".loan_contract SET total_amount_loan_contract = loan_amount +
monthly_payment * loan_term_int WHERE id_loan_contract = NEW.id_loan_contract;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;



CREATE TRIGGER loan_payment_trigger
AFTER INSERT ON "Bank_schema".loan_contract
FOR EACH ROW
EXECUTE FUNCTION calculate_loan_payment();
```

Рис. 7 – «Работа личного триггера»

Вывод

В данной лабораторной работе было выполнено создание и использование функций, так как их использование я посчитала более целесообразным для заданий моего варианта. Также я поработала с триггерами на логирование добавления, изменения и удаления данных и создала собственный триггер, который бы самостоятельно рассчитывал окончательную сумму кредита при добавлении нового контракта.