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

Отчет

по лабораторной работе «ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL»

по дисциплине «**Базы данных**»

Автор: Соколовская Арина Владимировна

Факультет: ИКТ

Группа: K32392

Преподаватель: Говорова М. М.

Дата: 03.05.2023

Санкт-Петербург 2023

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

2. **Оборудование:** компьютерный класс.

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

   Практическое задание:

   Вариант 1

   1. 2.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

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

```
CREATE OR REPLACE FUNCTION calculate_deposit_interest(agreement_id bigint)
    RETURNS TABLE(deposit_balance numeric, monthly_plus    numeric)
AS $$
DECLARE
    interest_rate       numeric;
    last_month_payments numeric;
    all_payments        numeric;
BEGIN
    -- Получение начальной суммы для заданного договора
    SELECT deposit_amount
    INTO deposit_balance
    FROM deposit_agreement
    WHERE deposit_agreement.contract_id = agreement_id;

    -- Расчет суммы начисленного процента в месяц
    SELECT deposit.interest_rate / 12
    INTO interest_rate
    FROM deposit, deposit_agreement
    WHERE deposit_agreement.contract_id = agreement_id
      and deposit.deposit_code = deposit_agreement.deposit_code;

        -- Получение суммы всех платежей по договору
    SELECT sum(payment_amount + payment_amount * interest_rate * date_part('month', age(pay_date)))
    INTO all_payments
    FROM deposit_payment_schedule, deposit_agreement
    WHERE deposit_agreement.contract_id = agreement_id;
```

```
    -- Получение суммы платежей за последний месяц по договору
    SELECT sum(payment_amount)
    INTO last_month_payments
    FROM deposit_payment_schedule, deposit_agreement
    WHERE deposit_agreement.contract_id = agreement_id AND pay_date between
        current_date - INTERVAL '2 month' and current_date - INTERVAL '1 month';

    deposit_balance := deposit_balance + all_payments;
    monthly_plus := last_month_payments + last_month_payments * interest_rate;

    RETURN NEXT;
END;
$$ LANGUAGE plpgsql;
```

```
 deposit_balance |      monthly_plus
-----------------+-------------------------
          2537.5 | 37.500000000000000000
(1 row)
```

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

```
CREATE OR REPLACE FUNCTION get_clients_with_most_credits()
    RETURNS TABLE (client_name VARCHAR(50))
AS $$
BEGIN
    RETURN QUERY
        SELECT c.full_name
        FROM client c
                 JOIN credit_agreement ca ON c.client_passport_number = ca.client_passport_number
        WHERE ca.status = 'Заключен'
        GROUP BY c.full_name
        HAVING COUNT(*) = (SELECT COUNT(*)
                           FROM credit_agreement
                           WHERE status = 'Заключен'
                           GROUP BY client_passport_number
                           ORDER BY COUNT(*) DESC
                           LIMIT 1);

END;
$$ LANGUAGE plpgsql;
```

```
        client_name
[------------------------------
 Евдокимова Кира Кирилловна
 Кузнецов Артём Георгиевич
(2 rows)
```

3. найти клиентов банка, не имеющих задолженности по кредитам

```sql
CREATE OR REPLACE FUNCTION get_clients_without_debts()
    RETURNS TABLE
            (
                client_name            VARCHAR(50),
                client_passport_number CHAR(10)
            )
AS
$$
BEGIN
    RETURN QUERY
        SELECT c.full_name, c.client_passport_number
        FROM client c
        WHERE EXISTS(
                SELECT 1
                FROM credit_agreement ca
                        JOIN credit_payment_schedule cps ON ca.contract_id = cps.contract_id
                WHERE c.client_passport_number = ca.client_passport_number
                  AND cps.payment_status = 'Не выплачено'
                  AND cps.planned_payment_date < current_date
            );

END;
$$ LANGUAGE plpgsql;
```

```
           client_name           | client_passport_number
---------------------------------+------------------------
 Кузнецов Артём Георгиевич       | 1993885725
 Евдокимова Кира Кирилловна      | 9517240027
 Андрианова Варвара Михайловна   | 6461063082
(3 rows)
```

- Создание триггеров

1. меняет работника в договоре кредита на работника с должностью "Консультант отделения банка по задолженностям", если появляется задолженность по кредиту

```sql
CREATE OR REPLACE FUNCTION update_employee_on_credit_default()
    RETURNS TRIGGER AS
$$
BEGIN
    IF EXISTS(
            SELECT 1
            FROM credit_payment_schedule cps
            WHERE cps.contract_id = NEW.contract_id
              AND cps.payment_status = 'Не выплачено'
              AND cps.planned_payment_date < current_date
        ) THEN
        UPDATE credit_agreement
        SET employee_passport_number = (SELECT employee_passport_number
                                        FROM employee
                                        WHERE position = 'Консультант отделения банка по задолженностям'
                                        LIMIT 1)
        WHERE contract_id = NEW.contract_id;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_employee_trigger
    AFTER INSERT OR UPDATE
    ON credit_payment_schedule
    FOR EACH ROW
EXECUTE FUNCTION update_employee_on_credit_default();
```

2. проверяет количество активных кредитов клиента при заключении нового кредита и выкидывает исключение, если у клиента уже есть 3 и более активных кредита

```sql
CREATE OR REPLACE FUNCTION check_max_credit_limit()
    RETURNS TRIGGER AS
$$
DECLARE
    client_credit_count INTEGER;
BEGIN
    -- Подсчитываем количество активных кредитов у клиента
    SELECT COUNT(*)
    INTO client_credit_count
    FROM credit_agreement
    WHERE client_passport_number = NEW.client_passport_number
      AND status = 'Заключен';

    -- Проверяем, превышает ли количество активных кредитов максимальное значение
    IF client_credit_count >= 3 THEN
        RAISE EXCEPTION 'Превышен максимальный лимит кредитов для клиента';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_max_credit_limit_trigger
    BEFORE INSERT
    ON credit_agreement
    FOR EACH ROW
EXECUTE FUNCTION check_max_credit_limit();
```

```
registration_of_deposits_and_credits=# INSERT INTO credit_agreement (contract_id, clien
ment_start_date, payment_end_date, maturity, credit_amount, interest_amount, total_amou
0, 11.200000000000017, 211.20000000000002);
ERROR:  Превышен максимальный лимит кредитов для клиента
```

Вывод:

В процессе выполнения данной лабораторной работы удалось овладеть навыками написания и использования процедур, функций и триггеров в PSQL.