Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет прикладной информатики
Образовательная программа Мобильные и сетевые технологии
Направление подготовки 09.03.03 Мобильные и сетевые технологии

О Т Ч Е Т

по лабораторной работе №5 Процедуры, функции, триггеры в PostgreSQL

По дисциплине «Проектирование и реализация баз данных»

Автор: Турищев А. И.
Факультет: ИКТ
Группа: К3340
Преподаватель: Говорова М. М.

Санкт-Петербург,
2025

Содержание

1 Цель работы **2**

2 Практическое задание **2**

3 Выполнение **2**

3.1 Процедуры 2

3.2 Триггеры 6

4 Выводы **9**

1 Цель работы

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

2 Практическое задание

Практическое задание:

1. Создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры. (3 балла)

2. Создать триггеры для индивидуальной БД согласно варианту: Вариант 2.1. 3 триггера - 3 балла (min). Допустимо использовать триггеры логирования из практического занятия по функциям и триггерам.

3 Выполнение

3.1 Процедуры

Процедура 1 - о текущей сумме вклада и сумме начисленного за месяц процента для заданного клиента;

```
lab-3=# CREATE OR REPLACE PROCEDURE get_client_deposit_info(
    p_client_id INTEGER,
    ref REFCURSOR DEFAULT 'client_deposit_cursor'
)
LANGUAGE plpgsql AS $$
BEGIN
    OPEN ref FOR
    WITH deposit_calc AS (
        SELECT
            v.Номер_договора_вклада,
            v.Начальная_сумма_вклада,
            vv.Процент_по_вкладу,
            ROUND(
                v.Начальная_сумма_вклада::NUMERIC *
                POWER(
                    1 + vv.Процент_по_вкладу::NUMERIC / 100.0 / 12,
                    EXTRACT(YEAR FROM AGE(CURRENT_DATE, v.Дата_начала_вклада)) * 12 +
                    EXTRACT(MONTH FROM AGE(CURRENT_DATE, v.Дата_начала_вклада))
                ),
                2
            ) AS Текущая_сумма_вклада
        FROM lab_3.Вклады v
        JOIN lab_3."Виды вкладов" vv ON v."ID_вида_вклада" = vv."ID_вида_вклада"
        WHERE v."ID_клиента" = p_client_id
          AND v.Статус_вклада = 'открыт'
    )
    SELECT
        Номер_договора_вклада,
        Начальная_сумма_вклада,
        Процент_по_вкладу,
        Текущая_сумма_вклада,
        ROUND(
            Текущая_сумма_вклада * (Процент_по_вкладу::NUMERIC / 100.0 / 12),
            2
        ) AS Процент_за_месяц
    FROM deposit_calc;
END;
$$;
CREATE PROCEDURE
```

Рис. 1: Скрипт процедуры 1 в psql

```
lab-3=# BEGIN;
CALL get_client_deposit_info(1, 'my_cursor_123');
FETCH ALL FROM my_cursor_123;
COMMIT;
BEGIN
CALL
 Номер_договора_вклада | Начальная_сумма_вклада | Процент_по_вкладу | Текущая_сумма_вклада | Процент_за_месяц
-----------------------+------------------------+-------------------+----------------------+------------------
                 10003 |                 400000 |                 8 |            405351.11 |          2702.34
                 10001 |                 500000 |                 5 |            502083.33 |          2092.01
(2 rows)
```

Рис. 2: Результат выполнения процедуры 1

Процедура 2 - добавить данные о новом вкладе клиента

```
lab-3=# CREATE OR REPLACE PROCEDURE lab_3.add_new_deposit(
    p_contract_number INTEGER,
    p_employee_id INTEGER,
    p_currency_code INTEGER,
    p_client_id INTEGER,
    p_deposit_type_id INTEGER,
    p_initial_amount INTEGER,
    p_start_date DATE,
    p_length INTEGER
)
LANGUAGE plpgsql AS $$
DECLARE
    v_status VARCHAR(6) := 'открыт';
    v_percent INTEGER;
BEGIN
    SELECT Процент_по_вкладу INTO v_percent
    FROM lab_3."Виды вкладов"
    WHERE "ID_вида_вклада" = p_deposit_type_id;

    INSERT INTO lab_3.Вклады (
        Номер_договора_вклада,
        "ID_сотрудника",
        Код_валюты,
        "ID_клиента",
        "ID_вида_вклада",
        Начальная_сумма_вклада,
        Дата_начала_вклада,
        Срок_вклада,
        Статус_вклада
    ) VALUES (
        p_contract_number,
        p_employee_id,
        p_currency_code,
        p_client_id,
        p_deposit_type_id,
        p_initial_amount,
        p_start_date,
        p_length,
        v_status
    );
END;
$$;
CREATE PROCEDURE
```

Рис. 3: Скрипт процедуры 2 в psql

```
lab-3=# call add_new_deposit(10006, 1, 2, 1, 1, 50000, '08-20-2025', 12)
;
CALL
```

Рис. 4: Вызов процедуры 2

	Номер_договора_вклада [PK] integer	ID_сотрудника integer	Код_валюты integer	ID_клиента integer	ID_вида_вклада integer	Суммарная_выплата real	Начальная_сумма_вклада real	Дата_начала_вклада date	Срок_вкл integer
1	10001	2	1	1	1	5116	500000	2025-08-01	
2	10002	1	1	2	2	86444	500000	2025-08-01	
3	10003	1	4	1	2	69155	400000	2025-07-01	
4	10004	1	4	4	2	8644	50000	2025-07-01	
5	10006	1	2	1	1	50000	50000	2025-08-20	

Рис. 5: Результат выполнения процедуры 2

Процедура 3 - найти клиентов банка, не имеющих задолженности по кредитам.

```
lab-3=# CREATE OR REPLACE PROCEDURE lab_3.get_clients_without_debt(
    ref REFCURSOR DEFAULT 'clients_no_debt_cursor'
)
LANGUAGE plpgsql AS $$
BEGIN
    OPEN ref FOR
    SELECT DISTINCT
        k."ID_клиента",
        k."ФИО_клиента"
    FROM lab_3.Клиенты k
    WHERE NOT EXISTS (
        -- Есть ли у клиента хотя бы одна НЕВЫПЛАЧЕННАЯ выплата по кредиту?
        SELECT 1
        FROM lab_3.Кредиты kr
        JOIN lab_3."Выплаты по кредиту" v
            ON kr."Номер_договора_кредита" = v."Номер_договора_кредита"
        WHERE kr."ID_клиента" = k."ID_клиента"
          AND v."Статус_выплаты_по_кредиту" = 'не выплачена'
    );
END;
$$;
CREATE PROCEDURE
```

Рис. 6: Скрипт процедуры 1 в psql

```
lab-3=# begin;
BEGIN
lab-3=*# call lab_3.get_clients_without_debt()
CALL
lab-3=*# FETCH ALL FROM clients_no_debt_cursor
;
 ID_клиента |      ФИО_клиента
------------+------------------------
          1 | Иванов Иван Иванович
          2 | Петрова Анна Сергеевна
          4 | Воробьева Ангелина
          5 | ТЕСТ
(4 rows)

lab-3=*# end;
COMMIT
```

Рис. 7: Результат выполнения процедуры 1

3.2 Триггеры

Триггер 1 и 2 - проверка правильности введенных данных в поля начальная сумма вклада, срок вклада, процентная ставка и срок кредита.

```
lab-3=# CREATE TRIGGER trg_check_vklad_params
BEFORE INSERT OR UPDATE ON lab_3.Вклады
FOR EACH ROW
EXECUTE FUNCTION lab_3.check_vklad_params();
ERROR:  trigger "trg_check_vklad_params" for relation "Вклады" already exists
lab-3=# CREATE OR REPLACE FUNCTION lab_3.check_vklad_params()
RETURNS TRIGGER AS $$
DECLARE
    min_sum INTEGER;
    max_sum INTEGER;
    min_term INTEGER;
    max_term INTEGER;
BEGIN
    SELECT
        Минимальная_сумма_вклада,
        Максимальная_сумма_вклада,
        Минимальный_срок_вклада,
        Максимальный_срок_вклада
    INTO min_sum, max_sum, min_term, max_term
    FROM lab_3."Виды вкладов"
    WHERE "ID_вида_вклада" = NEW."ID_вида_вклада";

    IF NEW.Начальная_сумма_вклада < min_sum OR NEW.Начальная_сумма_вклада > max_sum THEN
        RAISE EXCEPTION 'Сумма вклада (%) должна быть между % и % для вида вклада %',
            NEW.Начальная_сумма_вклада, min_sum, max_sum, NEW."ID_вида_вклада";
    END IF;

    IF NEW.Срок_вклада < min_term OR NEW.Срок_вклада > max_term THEN
        RAISE EXCEPTION 'Срок вклада (%) должен быть между % и % месяцев для вида вклада %',
            NEW.Срок_вклада, min_term, max_term, NEW."ID_вида_вклада";
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
lab-3=# CREATE TRIGGER trg_check_vklad_params
BEFORE INSERT OR UPDATE ON lab_3.Вклады
FOR EACH ROW
EXECUTE FUNCTION lab_3.check_vklad_params();
CREATE TRIGGER
```

Рис. 8: Скрипт триггера 1

	Номер_договора_вклада [PK] integer	ID_сотрудника integer	Код_валюты integer	ID_клиента integer	ID_вида_вклада integer	Начальная_сумма_вклада real	Дата_начала_вклада date	Срок_вклада integer	Статус_вклада character varying
1	10001	2	1	1	1	500000	2025-08-01	12	открыт
2	10002	1	1	2	2	500000	2025-08-01	24	открыт
3	10003	1	4	1	2	400000	2025-07-01	24	открыт
4	10004	1	4	4	2	50000	2025-07-01	24	открыт
5	10006	1	2	1	1	50000	2025-08-20	100	открыт

Срок вклада (100) должен быть между 1 и 12 месяцев для вида вклада 1
CONTEXT: PL/pgSQL function lab_3.check_vklad_params() line 23 at RAISE

Рис. 9: Результат работы триггера 1

```
lab-3=# CREATE OR REPLACE FUNCTION lab_3.check_kredit_params()
RETURNS TRIGGER AS $$
DECLARE
    min_rate INTEGER;
    max_rate INTEGER;
    min_term INTEGER;
    max_term INTEGER;
BEGIN
    SELECT
        Минимальная_процентная_ставка,
        Максимальная_процентная_ставка,
        Минимальный_срок_кредита,
        Максимальный_срок_кредита
    INTO min_rate, max_rate, min_term, max_term
    FROM lab_3."Виды кредитов"
    WHERE "ID_вида_кредита" = NEW."ID_вида_кредита";

    IF NEW.Процентная_ставка < min_rate OR NEW.Процентная_ставка > max_rate THEN
        RAISE EXCEPTION 'Процентная ставка (%) должна быть между % и % для вида кредита %',
            NEW.Процентная_ставка, min_rate, max_rate, NEW."ID_вида_кредита";
    END IF;

    IF NEW.Срок_кредита < min_term OR NEW.Срок_кредита > max_term THEN
        RAISE EXCEPTION 'Срок кредита (%) должен быть между % и % месяцев для вида кредита %',
            NEW.Срок_кредита, min_term, max_term, NEW."ID_вида_кредита";
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
lab-3=# CREATE TRIGGER trg_check_kredit_params
BEFORE INSERT OR UPDATE ON lab_3.Кредиты
FOR EACH ROW
EXECUTE FUNCTION lab_3.check_kredit_params();
CREATE TRIGGER
```

Рис. 10: Скрипт триггера 2

	вора_кредита	ID_сотрудника integer	Код_валюты integer	ID_клиента integer	ID_вида_кредита integer	Процентная_ставка real	Сумма_кредита real	Дата_начала_кредита date	Срок_кредита integer	Статус_кре character va
1	20001	2	1	1	1	12	200000	2025-08-01	24	открыт
2	20002	2	1	2	2	10	3e+06	2025-08-01	60	открыт
3	20003	2	3	3	2	8	4e+06	2025-07-01	60	открыт
4	20004	2	3	4	1	10	100000	2025-07-01	100	открыт

Срок кредита (100) должен быть между 6 и 24 месяцев для вида кредита 1
CONTEXT: PL/pgSQL function lab_3.check_kredit_params() line 23 at RAISE

Рис. 11: Результат работы триггера 2

Триггеры 3 и 4 - Заполнение таблиц выплаты по вкладу и выплаты по кредиту на основе новой записи в таблице вклады или кредиты

```
lab-3=# CREATE OR REPLACE FUNCTION lab_3.generate_vklad_payments()
RETURNS TRIGGER AS $$
DECLARE
    i INTEGER;
    payment_date DATE;
    monthly_payment NUMERIC;
    total_on_vkald NUMERIC := NEW.Начальная_сумма_вклада;
    total_months INTEGER := NEW.Срок_вклада;
    percent NUMERIC;
BEGIN
    SELECT Процент_по_вкладу
    INTO percent
    FROM lab_3."Виды вкладов"
    WHERE "ID_вида_вклада" = NEW."ID_вида_вклада";

    FOR i IN 1..total_months LOOP
        monthly_payment := (total_on_vkald * percent) / 100 / 12;
        payment_date := NEW.Дата_начала_вклада + (i || ' months')::INTERVAL;

        INSERT INTO lab_3."Выплаты по вкладу" (
            Номер_договора_вклада,
            Сумма_выплаты_по_вкладу,
            Статус_выплаты_по_вкладу,
            Дата_выплаты_по_вкладу,
            Фактическая_дата_выплаты_по_вкладу
        )
        VALUES (
            NEW.Номер_договора_вклада,
            monthly_payment,
            'не выплачена',
            payment_date,
            NULL
        );
        total_on_vkald := monthly_payment + total_on_vkald;
    END LOOP;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
NOTICE:  identifier "Фактическая_дата_выплаты_по_вкладу" will be truncated to "Фактическая_дата_выплаты_по_вклад"
NOTICE:  identifier "Фактическая_дата_выплаты_по_вкладу" will be truncated to "Фактическая_дата_выплаты_по_вклад"
LINE 20:         INSERT INTO lab_3."Выплаты по вкладу" (
                 ^
CREATE FUNCTION
lab-3=# CREATE TRIGGER trg_generate_vklad_payments
AFTER INSERT ON lab_3.Вклады
FOR EACH ROW
EXECUTE FUNCTION lab_3.generate_vklad_payments();
CREATE TRIGGER
```

Рис. 12: Скрипт триггера 3

7	10005	41.666668	не выплачена	2025-09-20	[null]
8	10005	41.84028	не выплачена	2025-10-20	[null]
9	10005	42.014614	не выплачена	2025-11-20	[null]
10	10005	42.189674	не выплачена	2025-12-20	[null]
11	10005	42.365463	не выплачена	2026-01-20	[null]
12	10005	42.541985	не выплачена	2026-02-20	[null]
13	10005	42.719246	не выплачена	2026-03-20	[null]
14	10005	42.89724	не выплачена	2026-04-20	[null]

Рис. 13: Результат работы триггера 3

```
lab-3=# CREATE OR REPLACE FUNCTION lab_3.generate_kredit_payments()
RETURNS TRIGGER AS $$
DECLARE
    i INTEGER;
    payment_date DATE;
    monthly_principal NUMERIC;
    monthly_interest NUMERIC;
    total_months INTEGER := NEW.Срок_кредита;
    remaining_debt NUMERIC := NEW.Сумма_кредита;
BEGIN
    monthly_principal := NEW.Сумма_кредита / total_months;

    FOR i IN 1..total_months LOOP
        payment_date := NEW.Дата_начала_кредита + (i || ' months')::INTERVAL;

        monthly_interest := (remaining_debt * NEW.Процентная_ставка) / 100 / 12;

        INSERT INTO lab_3."Выплаты по кредиту" (
            Номер_договора_кредита,
            Сумма_выплаты_по_кредиту,
            Статус_выплаты_по_кредиту,
            Дата_выплаты_по_кредиту,
            Фактическая_дата_выплаты_по_кредиту,
            Сумма_выплаты_по_процентам
        )
        VALUES (
            NEW.Номер_договора_кредита,
            monthly_principal,
            'не выплачена',
            payment_date,
            NULL,
            monthly_interest
        );
        remaining_debt := remaining_debt - monthly_principal;
    END LOOP;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
NOTICE:  identifier "Фактическая_дата_выплаты_по_кредиту" will be truncated to "Фактическая_дата_выплаты_по_креди"
NOTICE:  identifier "Фактическая_дата_выплаты_по_кредиту" will be truncated to "Фактическая_дата_выплаты_по_креди"
LINE 18:         INSERT INTO lab_3."Выплаты по кредиту" (
                 ^
CREATE FUNCTION
lab-3=# CREATE TRIGGER trg_generate_kredit_payments
AFTER INSERT ON lab_3.Кредиты
FOR EACH ROW
EXECUTE FUNCTION lab_3.generate_kredit_payments();
CREATE TRIGGER
```

Рис. 14: Скрипт триггера 4

	ора_кредита	Сумма_выплаты_по_кредиту real	Статус_выплаты_по_кредиту character varying (12)	Дата_выплаты_по_кредиту date	Фактическая_дата_выплаты_по_креди date	Сумма_выплаты_по_процентам real
7	20005	833.3333	не выплачена	2025-09-01	[null]	83.333336
8	20005	833.3333	не выплачена	2025-10-01	[null]	76.388885
9	20005	833.3333	не выплачена	2025-11-01	[null]	69.44444
10	20005	833.3333	не выплачена	2025-12-01	[null]	62.5
11	20005	833.3333	не выплачена	2026-01-01	[null]	55.555557
12	20005	833.3333	не выплачена	2026-02-01	[null]	48.61111
13	20005	833.3333	не выплачена	2026-03-01	[null]	41.666668
14	20005	833.3333	не выплачена	2026-04-01	[null]	34.72222
15	20005	833.3333	не выплачена	2026-05-01	[null]	27.777779

Рис. 15: Результат работы триггера 4

4 Выводы

В ходе работы я ознакомился с функциями, процедурами и триггерами в PostgreSQL, научился их создавать в консоли psql. Были созданы 3 процедуры и 4 триггера по индивидуальному заданию, так же была проверена их работа.