

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

Лабораторная работа 5

Тема задания: «Процедуры, функции, триггеры в PostgreSQL»

Обучающийся: Анисимов Владислав Андреевич, группа К3240

Проверяющий: Говорова М.М., преподаватель

Санкт–Петербург,
2025

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ.....	2
Цель работы.....	3
Практическое задание.....	4
Выполнение.....	5
1 Создание процедур.....	5
2 Создание триггеров (7 штук).....	8
Выводы.....	16

Цель работы

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

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.

Вариант 2.2. 7 оригинальных триггеров - 7 баллов (max).

Выполнение

1 Создание процедур

- Выполнить списание автомобилей, выпущенных ранее заданного года.

```
CREATE OR REPLACE PROCEDURE write_off_old_cars(
    p_year INTEGER
)
LANGUAGE plpgsql AS
$$
BEGIN
    DELETE FROM accident
    WHERE instance_id IN (
        SELECT id FROM car_instance WHERE year <= p_year
    );

    DELETE FROM fine
    WHERE instance_id IN (
        SELECT id FROM car_instance WHERE year <= p_year
    );

    DELETE FROM contract
    WHERE instance_id IN (
        SELECT id FROM car_instance WHERE year <= p_year
    );

    DELETE FROM car_instance
    WHERE year <= p_year;

    COMMIT;
END;
$$;
```

17	17	1	M8460A178	JN8AZ28R59T103357	Авто невостробовано.	TOY2ARFEJPN789012	2025-03-20	3000000	72000	2021
18	22	4	a555aa78	idk	[null]	idk	2025-05-22	550000	147555	2011

Рис 1.1 - перед выполнением процедуры 1

16	16	7	E678EH78	XG0876543034567...	Авто невостробовано.	BMW58B30M3456...	2024-11-30	4300000	60000	2019
17	17	1	M8460A178	JN8AZ28R59T103357	Авто невостробовано.	TOY2ARFEJPN789012	2025-03-20	3000000	72000	2021

Рис 1.2 - результат выполнения процедуры 1

- Выдачи автомобиля и расчета стоимости с учетом скидки постоянным клиентам.

```
CREATE OR REPLACE PROCEDURE rent_car_with_discount(  
    p_car_instance_id INTEGER,  
    p_client_id INTEGER,  
    p_rent_emp_id INTEGER,  
    p_hours INTEGER,  
    p_price REAL,  
    OUT p_total_price REAL,  
    OUT p_contract_id INTEGER  
)  
  
LANGUAGE plpgsql AS  
$$  
DECLARE  
    v_client_discount REAL;  
    v_passport_id INTEGER;  
    v_date_end TIMESTAMP;  
BEGIN  
  
    SELECT COALESCE(discount, 0) INTO v_client_discount  
    FROM client  
    WHERE id = p_client_id;  
  
    SELECT id INTO v_passport_id  
    FROM passport_history  
    WHERE client_id = p_client_id AND date_end IS NULL;  
  
    p_total_price := p_hours * p_price / 24 * (1 -  
v_client_discount);  
  
    v_date_end := CURRENT_TIMESTAMP + (p_hours * INTERVAL '1  
hour');  
  
    INSERT INTO contract (  
        instance_id,  
        passport_id,  
        price,  
        date_start,  
        rent_emp_id  
    ) VALUES (  
        p_car_instance_id,
```

```

        v_passport_id,
        p_total_price,
        CURRENT_TIMESTAMP,
        p_rent_emp_id
    ) RETURNING id INTO p_contract_id;

    INSERT INTO contract_history (
        contract_id,
        emp_id,
        date_end
    ) VALUES (
        p_contract_id,
        p_rent_emp_id,
        v_date_end
    );

    COMMIT;
END;
$$;

```

Query Query History

1 CALL rent_car_with_discount(13, 8, 3, 14, 5000, 0, 0);

Data Output Messages Notifications

	p_total_price real	p_contract_id integer
1	2916.6667	1245

Рис 2 - результат выполнения процедуры 2

- Для вычисления количества автомобилей заданной марки.

```

CREATE OR REPLACE PROCEDURE count_cars_by_brand(
    p_brand_name TEXT,
    OUT cars_count INTEGER
)
LANGUAGE plpgsql AS
$$
BEGIN
    SELECT COUNT(*) INTO cars_count
    FROM car_instance

```

```
JOIN car ON car_instance.car_id = car.id
JOIN car_info ON car.info_id = car_info.id
JOIN brand ON car_info.brand_id = brand.id
WHERE brand.name = p_brand_name;

END;

$$;
```

```
Query      Query History
1  CALL count_cars_by_brand('BMW', 0);

Data Output  Messages  Notifications

cars_count integer
1            3
```

Рис 3 - результат выполнения процедуры 3

2 Создание триггеров (7 штук)

- Выдача скидок клиентам, которые арендовали много автомобилей за год

```
CREATE OR REPLACE FUNCTION apply_discount()
RETURNS TRIGGER AS $$
DECLARE
    contract_count INTEGER;
    c_client_id INTEGER;
BEGIN
    SELECT COUNT(*) INTO contract_count
    FROM contract
    JOIN passport_history ON contract.passport_id =
passport_history.id
    JOIN client ON client.id = passport_history.client_id
    WHERE NEW.passport_id = passport_history.id
    AND extract(year from contract.date_start) = EXTRACT(YEAR
FROM CURRENT_DATE);

    SELECT passport_history.client_id INTO c_client_id
    FROM passport_history
    WHERE NEW.passport_id = passport_history.id;
```



```

    IF contract_count >= 10 THEN
        UPDATE client SET discount = 0.15
        WHERE c_client_id = client.id;
    ELSIF contract_count >= 5 THEN
        UPDATE client SET discount = 0.1
        WHERE c_client_id = client.id;
    ELSIF contract_count >= 3 THEN
        UPDATE client SET discount = 0.05
        WHERE c_client_id = client.id;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

```

CREATE OR REPLACE TRIGGER give_discount
BEFORE INSERT ON contract
FOR EACH ROW
EXECUTE FUNCTION apply_discount();

```

45	45	Lothaire Paladini	+62 (996) 490-7913	16315 Maple Wood Terrace	[null]
46	46	Kata Roberti	+86 (265) 741-9532	18502 Eagan Alley	0.05
47	47	Lurette Dommerque	+55 (640) 102-0875	1 Logan Alley	[null]

Рис 4 - результат работы триггера 1

- Проверка авто на доступность перед арендой

```

CREATE OR REPLACE FUNCTION check_car_availability()
RETURNS TRIGGER AS $$
DECLARE
    is_available BOOLEAN;
BEGIN
    SELECT NOT EXISTS (
        SELECT 1 FROM contract
        WHERE instance_id = NEW.instance_id
        AND return_emp_id IS NULL
    ) INTO is_available;

    IF NOT is_available THEN
        RAISE EXCEPTION 'Автомобиль с ID % уже арендован',
NEW.instance_id;

```

```

        END IF;

        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_availability
BEFORE INSERT ON contract
FOR EACH ROW
EXECUTE FUNCTION check_car_availability();

```

ERROR: Автомобиль с ID 4 уже арендован

CONTEXT: функция PL/pgSQL check_car_availability(), строка 13, оператор RAISE

SQL-оператор: "INSERT INTO contract (

```

    instance_id,
    passport_id,
    price,
    date_start,
    rent_emp_id
) VALUES (
    p_car_instance_id,
    v_passport_id,
    p_total_price,
    CURRENT_TIMESTAMP,
    p_rent_emp_id
) RETURNING id"
```

функция PL/pgSQL rent_car_with_discount(integer,integer,integer,integer,real), строка 20, оператор SQL-оператор

ОШИБКА: Автомобиль с ID 4 уже арендован

SQL state: P0001

Рис 5 - результат работы триггера 2

- Проверка новой даты при продлении аренды

```

CREATE OR REPLACE FUNCTION validate_contract_dates()
RETURNS TRIGGER AS $$
DECLARE
    last_date TIMESTAMP;
BEGIN
    SELECT MAX(date_end) INTO last_date
    FROM contract_history
    WHERE contract_history.contract_id = NEW.contract_id;

    IF NEW.date_end < last_date THEN
        RAISE EXCEPTION 'Дата окончания не может быть раньше даты
начала';
    END IF;

    RETURN NEW;

```

```

END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER validate_contract_h_dates
BEFORE INSERT OR UPDATE ON contract_history
FOR EACH ROW
EXECUTE FUNCTION validate_contract_dates();

```

Query Query History

```

1  INSERT INTO contract_history(contract_id, emp_id, date_end)
2  VALUES(1245, 3, NOW() - interval '2 month')

```

Data Output Messages Notifications

ERROR: Дата окончания не может быть раньше даты начала
CONTEXT: функция PL/pgSQL validate_contract_dates(), строка 10, оператор RAISE

ОШИБКА: Дата окончания не может быть раньше даты начала
SQL state: P0001

Рис 6 - результат работы триггера 3

- Проверка скидки на валидность

```

CREATE OR REPLACE FUNCTION validate_client_discount()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.discount > 0.3 THEN
        RAISE EXCEPTION 'Скидка не может превышать 30%%';
    END IF;

    IF NEW.discount < 0 THEN
        RAISE EXCEPTION 'Скидка не может быть отрицательной';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER validate_discount
BEFORE INSERT OR UPDATE ON client
FOR EACH ROW
EXECUTE FUNCTION validate_client_discount();

```

ERROR: Скидка не может быть отрицательной
CONTEXT: функция PL/pgSQL validate_client_discount(), строка 8, оператор RAISE

ОШИБКА: Скидка не может быть отрицательной
SQL state: P0001

Рис 7 - результат работы триггера 4

- Проверка валидности должности у работника

```
CREATE OR REPLACE FUNCTION validate_emp_dates()
RETURNS TRIGGER AS $$
DECLARE
    last_date TIMESTAMP;
BEGIN
    SELECT MAX(date_end) INTO last_date
    FROM positions_history
    WHERE positions_history.emp_id = NEW.emp_id;

    IF NEW.date_start < last_date THEN
        RAISE EXCEPTION 'Дата начала работы не может быть раньше
даты конца';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER validate_emp_h_dates
BEFORE INSERT OR UPDATE ON positions_history
FOR EACH ROW
EXECUTE FUNCTION validate_emp_dates();
```

ERROR: Дата начала работы не может быть раньше даты конца
CONTEXT: функция PL/pgSQL validate_emp_dates(), строка 10, оператор RAISE

ОШИБКА: Дата начала работы не может быть раньше даты конца
SQL state: P0001

Рис 8 - результат работы триггера 5

- Автоматическая установка даты окончания при обновлении паспорта

```
CREATE OR REPLACE FUNCTION update_passport_status()
RETURNS TRIGGER AS $$
```

```

BEGIN
    UPDATE passport_history
    SET date_end = CURRENT_DATE
    WHERE client_id = NEW.client_id
    AND date_end IS NULL
    AND id != NEW.id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER update_passport_status
AFTER INSERT ON passport_history
FOR EACH ROW
EXECUTE FUNCTION update_passport_status();

```

86	86	86	2024-05-14	[null]	8299411898
87	87	87	2021-02-13	2025-05-22	6089283715
88	88	88	2023-04-20	[null]	7539175067
89	89	89	2021-01-14	[null]	3980069585

Рис 9 - результат работы триггера 6

- Установка специальных отметок автомобилям

```

CREATE OR REPLACE PROCEDURE check_car_condition(
    i_car_id INTEGER,
    OUT result TEXT
)
LANGUAGE plpgsql AS
$$
DECLARE
    v_last_maintenance TIMESTAMP;
    v_is_needed BOOLEAN;
    v_mileage INTEGER;
BEGIN
    SELECT last_maintenance INTO v_last_maintenance
    FROM car_instance
    WHERE i_car_id = car_instance.id;

    result := '';

    IF v_last_maintenance < NOW() - INTERVAL '6 month' THEN
        result := result || 'Необходимо провести ТО. ';
    END IF;
END;

```

```

END IF;

SELECT EXISTS (
    SELECT 1
    FROM car_instance
    JOIN contract ON contract.instance_id = i_car_id
    WHERE i_car_id = car_instance.id
    AND contract.date_start >= NOW() - INTERVAL '3 month'
) INTO v_is_needed;

IF NOT v_is_needed THEN
    result := result || 'Авто не востребовано. ';
END IF;

SELECT mileage INTO v_mileage
FROM car_instance
WHERE i_car_id = car_instance.id;

IF v_mileage > 150000 THEN
    result := result || 'У авто большой пробег. ';
END IF;
END;
$$;

CREATE OR REPLACE FUNCTION check_car_condition_con()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE car_instance SET special_marks = NULL
    WHERE car_instance.id = NEW.instance_id;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION check_car_condition_car()
RETURNS TRIGGER AS $$
DECLARE
    result TEXT;
BEGIN
    IF NEW.special_marks IS NULL THEN
        CALL check_car_condition(NEW.id, result);
        UPDATE car_instance SET special_marks = result
        WHERE car_instance.id = NEW.id;
    END IF;
END;

```

```

        END IF;
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_car_c
AFTER UPDATE ON contract
FOR EACH ROW
EXECUTE FUNCTION check_car_condition_con();

CREATE TRIGGER check_car
AFTER UPDATE ON car_instance
FOR EACH ROW
EXECUTE FUNCTION check_car_condition_car();

```

	id [PK] integer	car_id integer	reg_code text	vin_code text	special_marks text	engine_code text	last_maintenance date	price integer	mileage integer	year integer
1	1	16	A123AB78	XTA210990Y1234567	Необходимо провести ТО. Авто невостребован...	VWEA2114TSI12345	2024-10-15	850000	120000	2015
2	2	7	B234BC78	Y6TF51234A9876543	Необходимо провести ТО.	BMW58B30M1234...	2024-11-20	4200000	60000	2019
3	3	22	C345CT78	Z7VG98765B4567890		CHVLT1V86212345	2024-12-05	3800000	50000	2017
4	4	3	E456EK78	XWB654321C78901...	Необходимо провести ТО.	TYTA25AFKS123456	2024-10-25	2100000	75000	2018
5	5	10	K567KM78	YTC321098D2345678	Необходимо провести ТО.	MBM27420T123456	2024-11-10	3200000	90000	2017
6	6	16	M678MH78	ZUD210987E3456789	У авто большой пробег.	VWEA2114TSI67890	2024-12-15	780000	187524	2015
7	7	25	H789HO78	XVF987654F4567890	Необходимо провести ТО.	HYTTTHETA2023456	2024-10-30	1850000	55000	2019
8	8	7	O890OP78	YWG876543G56789...	Авто невостребовано.	BMW58B30M7890...	2024-11-25	4500000	60000	2019
9	9	13	P901PY78	ZXH765432H67890...		AUDEA88820T1234	2024-12-20	2800000	110000	2016
10	10	22	C012CX78	XAI654321I7890123	Необходимо провести ТО. Авто невостребован...	CHVLT1V86267890	2024-11-05	3700000	50000	2017
11	11	4	T123TA78	YBJ54321OJ8901234	Авто невостребовано.	HONL15B15T12345	2024-12-10	2200000	45000	2020
12	12	10	Y234YB78	ZCK432109K9012345	Необходимо провести ТО.	MBM27420T789012	2024-10-20	3100000	90000	2017
13	13	16	X345XE78	XDL321098L0123456	Необходимо провести ТО. Авто невостребован...	VWEA2114TSI34567	2024-11-15	800000	120000	2015
14	14	19	A456AK78	YEM210987M12345...		FORDCOYOTE50123	2024-12-25	5200000	30000	2021
15	15	25	B567BM78	ZFN987654N2345678	Необходимо провести ТО. Авто невостребован...	HYTTTHETA2078901	2024-10-10	1950000	55000	2019
16	16	7	E678EH78	XGO876543O34567...	Авто невостребовано.	BMW58B30M3456...	2024-11-30	4300000	60000	2019
17	17	1	M846OA178	JN8AZ28R59T103357	Авто невостребовано.	TOY2ARFEJPN789012	2025-03-20	3000000	72000	2021

Рис 10 - результат работы триггера 7

Выводы

В ходе выполнения лабораторной работы я научился создавать процедуры, функции и триггеры в PostgreSQL. Были созданы 3 процедуры и 7 оригинальных триггеров. В результате, оказалось, что очень многие процессы при работе с базой данных можно упростить и автоматизировать. Я понял, что триггеры - это очень важная часть при работе с БД, которая нужна в первую очередь для удобства.