МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Практическое занятие № 2

Тема занятия: 1) Работа с триггерами

Выполнил студент 3 курса группы
ЭФБО-10-23 Ефремов А.И.

Проверил доцент Бочаров М.И.

Москва
2025 г.

Ефремов ЭФБО-10-23 Практическая 2

Практическая 2.1

1. Создать триггер, который при продаже будет выводить предупреждающее сообщение при разнице заявленной и продажной стоимости объекта недвижимости более чем на 20%.

```
CREATE OR REPLACE FUNCTION check_price_difference_yefremov()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    original_price INTEGER;
    price_diff_percent NUMERIC;
BEGIN
    -- Получаем исходную цену объекта недвижимости
    SELECT price INTO original_price
    FROM property
    WHERE id = NEW.property_id;

    -- Вычисляем разницу в процентах
    price_diff_percent := ABS((original_price - NEW.price) / original_price::NUMERIC) *
100;

    -- Если разница более 20%, выводим предупреждение
    IF price_diff_percent > 20 THEN
        RAISE WARNING 'Большая разница в цене! Объект %: заявленная цена %, продажная
цена %. Разница: %',
            NEW.property_id, original_price, NEW.price, ROUND(price_diff_percent, 2);
    END IF;

    RETURN NEW;
END;
$$;

-- Создание триггера
CREATE TRIGGER sale_price_check_yefremov
    BEFORE INSERT OR UPDATE ON sale
    FOR EACH ROW
    EXECUTE FUNCTION check_price_difference_yefremov();
```

```
INSERT INTO sale (property_id, sale_date, realtor_id, price)
VALUES (21, '2023-05-20', 2, 5500000);
```

```
mirea=# INSERT INTO sale (property_id, sale_date, realtor_id, price)
VALUES (21, '2023-05-20', 2, 5500000);
WARNING:  Большая разница в цене! Объект 21: заявленная цена 7200000, продажная цена 5500000. Разница: 23.61
INSERT 0 1
mirea=#
```

2. Создать триггер, который при добавлении новой продажи будет осуществлять проверку на повторную продажу объекта недвижимости. Если в таблице «Продажи» уже имеется запись о продаже данного объекта, выводить соответствующее сообщение и запрещать новую продажу.

```
CREATE OR REPLACE FUNCTION check_duplicate_sale_yefremov()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (SELECT 1 FROM sale WHERE property_id = NEW.property_id) THEN
        RAISE EXCEPTION 'Объект недвижимости % уже продан!', NEW.property_id;
    END IF;
    RETURN NEW;
END;
$$;

-- Создание триггера
CREATE TRIGGER prevent_duplicate_sale_yefremov
    BEFORE INSERT ON sale
    FOR EACH ROW
    EXECUTE FUNCTION check_duplicate_sale_yefremov();
```

```
INSERT INTO sale (property_id, sale_date, realtor_id, price)
VALUES (1, '2025-10-01', 1, 5000000);
```

```
mirea=# INSERT INTO sale (property_id, sale_date, realtor_id, price)
VALUES (1, '2025-10-01', 1, 5000000);
ERROR:  Объект недвижимости 1 уже продан!
CONTEXT:  PL/pgSQL function check_duplicate_sale_yefremov() line 4 at RAISE
mirea#
```

3. Создать триггер, который при добавлении новой записи в таблицу «Структура объекта недвижимости» проверяет несоответствие суммы площадей всех заявленных комнат (в большую сторону) с общей площадью объекта недвижимости. Выводить сообщение на сколько превышена площадь.

```
CREATE OR REPLACE FUNCTION check_room_areas_yefremov()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    total_room_area SMALLINT;
    property_area SMALLINT;
    discrepancy SMALLINT;
BEGIN
    -- Сумма площадей комнат для этого объекта (включая новую)
    SELECT SUM(area) INTO total_room_area
    FROM property_structure
    WHERE property_id = NEW.property_id;

    -- Общая площадь объекта
    SELECT area INTO property_area
    FROM property
```

```
    WHERE id = NEW.property_id;

    discrepancy := total_room_area - property_area;

    IF discrepancy > 0 THEN
        RAISE WARNING 'Сумма площадей комнат превышает общую площадь на % м² для объекта
%',
            discrepancy, NEW.property_id;
    END IF;

    RETURN NEW;
END;
$$;

-- Создание триггера
CREATE TRIGGER room_area_check_yefremov
    AFTER INSERT ON property_structure
    FOR EACH ROW
    EXECUTE FUNCTION check_room_areas_yefremov();
```

```
INSERT INTO property_structure (property_id, room_name, area) VALUES (1, 'Гараж', 20);
```

```
CREATE TRIGGER
mirea=# INSERT INTO property_structure (property_id, room_name, area) VALUES (1, 'Кухня', 10);
WARNING:  Сумма площадей комнат превышает общую площадь на 35 м² для объекта 1
INSERT 0 1
mirea=#
```

4.Создать триггер, который будет выводить предупреждающее сообщение, что дата продажи раньше, чем дата размещения объявления.

```
CREATE OR REPLACE FUNCTION check_sale_date_yefremov()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    listing_date_val DATE;
BEGIN
    SELECT listing_date INTO listing_date_val
    FROM property
    WHERE id = NEW.property_id;

    IF NEW.sale_date < listing_date_val THEN
        RAISE WARNING 'Дата продажи (%) раньше даты размещения (%) для объекта %',
            NEW.sale_date, listing_date_val, NEW.property_id;
    END IF;

    RETURN NEW;
END;
$$;

-- Создание триггера
CREATE TRIGGER sale_date_check_yefremov
    BEFORE INSERT OR UPDATE ON sale
```

```
    FOR EACH ROW
    EXECUTE FUNCTION check_sale_date_yefremov();
```

```
INSERT INTO sale (property_id, sale_date, realtor_id, price) VALUES (30, '2020-01-01',
1, 5000000);
```

```
mirea=# INSERT INTO sale (property_id, sale_date, realtor_id, price) VALUES (30, '2020-01-01', 1, 5000000);
WARNING:  Дата продажи (2020-01-01) раньше даты размещения (2023-03-15) для объекта 30
WARNING:  Большая разница в цене! Объект 30: заявленная цена 3100000, продажная цена 5000000. Разница: 61.29
INSERT 0 1
mirea=#
```

5. Создать таблицы «Журнал»: дата и время операции, операция (INSERT, UPDATE, DELETE), пользователь. Создать триггер, который будет формировать новую запись в этой таблице при добавлении/изменении/удалении строки в таблицу «Продажи».

```
CREATE TABLE journal (
    operation_time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    operation_type VARCHAR(10) NOT NULL,
    user_name VARCHAR(255) NOT NULL DEFAULT CURRENT_USER,
    sale_id INTEGER
);

CREATE OR REPLACE FUNCTION log_sale_operations_yefremov()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO journal (operation_type, sale_id) VALUES ('INSERT', NEW.id);
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO journal (operation_type, sale_id) VALUES ('UPDATE', NEW.id);
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO journal (operation_type, sale_id) VALUES ('DELETE', OLD.id);
        RETURN OLD;
    END IF;
    RETURN NEW;
END;
$$;

-- Создание триггера
CREATE TRIGGER sale_log_yefremov
    AFTER INSERT OR UPDATE OR DELETE ON sale
    FOR EACH ROW
    EXECUTE FUNCTION log_sale_operations_yefremov();
```

```
INSERT INTO sale (property_id, sale_date, realtor_id, price) VALUES (23, '2025-10-01',
1, 5000000);
UPDATE sale SET price = 6000000 WHERE id = (SELECT MAX(id) FROM sale);
DELETE FROM sale WHERE id = (SELECT MAX(id) FROM sale);
```

```
SELECT * FROM journal ORDER BY operation_time;
```

```
mirea=# SELECT * FROM journal ORDER BY operation_time;
       operation_time       | operation_type | user_name | sale_id
----------------------------+----------------+-----------+---------
 2025-09-30 09:59:35.869517 | INSERT         | postgres  |      28
 2025-09-30 10:04:20.643995 | INSERT         | postgres  |      29
 2025-09-30 10:04:20.658726 | UPDATE         | postgres  |      29
 2025-09-30 10:04:20.661757 | DELETE         | postgres  |      29
(4 rows)
```

Практическая 2.2

1. Добавить таблицу «Бонусы», в которой будет две колонки: код риэлтора и размер накопленных бонусов. Размер бонуса рассчитывается по формуле – `стоимость продажи * 5%`. Создать триггер, который будет автоматически увеличивать размер накопленного бонуса при добавлении новой продажи. Необходимо учесть, что продажа риэлтора может быть впервые, следовательно необходимо добавить новую запись в таблицу. При удалении данных о продажи, размер накопленного бонуса должен уменьшиться.

```
CREATE TABLE bonuses (
    realtor_id INTEGER PRIMARY KEY REFERENCES realtor(id),
    bonus_amount DOUBLE PRECISION DEFAULT 0
);

CREATE OR REPLACE FUNCTION update_bonus_yefremov()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    -- При INSERT: Добавляем бонус
    IF TG_OP = 'INSERT' THEN
        INSERT INTO bonuses (realtor_id, bonus_amount)
        VALUES (NEW.realtor_id, NEW.price * 0.05)
        ON CONFLICT (realtor_id) DO UPDATE
        SET bonus_amount = bonuses.bonus_amount + EXCLUDED.bonus_amount;

    -- При DELETE: Уменьшаем бонус
    ELSIF TG_OP = 'DELETE' THEN
        UPDATE bonuses
        SET bonus_amount = bonus_amount - (OLD.price * 0.05)
        WHERE realtor_id = OLD.realtor_id;
    END IF;
    RETURN NEW;
END;
$$;

-- Создание триггера
CREATE TRIGGER bonus_update_yefremov
    AFTER INSERT OR DELETE ON sale
    FOR EACH ROW
    EXECUTE FUNCTION update_bonus_yefremov();
```

```
INSERT INTO sale (property_id, sale_date, realtor_id, price) VALUES (28, '2025-10-01',
1, 1000000);
```

```sql
SELECT * FROM bonuses WHERE realtor_id = 1;
```

```
mirea=# INSERT INTO sale (property_id, sale_date, realtor_id, price) VALUES (28, '2025-10-01', 1, 1000000);
WARNING:  Большая разница в цене! Объект 28: заявленная цена 5400000, продажная цена 1000000. Разница: 81.48
INSERT 0 1
mirea=# SELECT * FROM bonuses WHERE realtor_id = 1;
 realtor_id | bonus_amount
------------+--------------
          1 |       100000
(1 row)
```

2. Создать триггер, который будет информировать о превышении размера накопленного бонуса риэлтором. Максимальный размер бонуса установить самостоятельно.

```sql
CREATE OR REPLACE FUNCTION check_bonus_limit_yefremov()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    max_bonus CONSTANT DOUBLE PRECISION := 1000000;
BEGIN
    IF NEW.bonus_amount > max_bonus THEN
        RAISE WARNING 'Превышен максимальный бонус для риэлтора %: текущий %',
            NEW.realtor_id, NEW.bonus_amount;
    END IF;
    RETURN NEW;
END;
$$;

-- Создание триггера
CREATE TRIGGER bonus_limit_check_yefremov
    AFTER UPDATE ON bonuses
    FOR EACH ROW
    EXECUTE FUNCTION check_bonus_limit_yefremov();
```

```sql
UPDATE bonuses SET bonus_amount = 1100000 WHERE realtor_id = 1;
```

```
mirea=# UPDATE bonuses SET bonus_amount = 1100000 WHERE realtor_id = 1;
WARNING:  Превышен максимальный бонус для риэлтора 1: текущий 1100000
UPDATE 1
mirea=#
```

3. В таблицу «Риэлторы» добавить колонку «Паспортные данные». Создать триггер, который будет проверять корректность паспортных данных по маске: `XXXX YYYYYY`, где `X` – серия паспорта, `Y` – номер паспорта. Между серией и номером паспорта пробел. При несоответствии вводимой информации маске, выводить сообщение.

```sql
ALTER TABLE realtor ADD COLUMN passport_data VARCHAR(11);

CREATE OR REPLACE FUNCTION check_passport_yefremov()
RETURNS TRIGGER
LANGUAGE plpgsql
```

```
AS $$
BEGIN
    IF NEW.passport_data !~ '^\d{4} \d{6}$' THEN
        RAISE EXCEPTION 'Некорректные паспортные данные: % (ожидается формат XXXX
УУУУУУ)', NEW.passport_data;
    END IF;
    RETURN NEW;
END;
$$;

-- Создание триггера
CREATE TRIGGER passport_check_yefremov
    BEFORE INSERT OR UPDATE ON realtor
    FOR EACH ROW
    EXECUTE FUNCTION check_passport_yefremov();
```

```
UPDATE realtor SET passport_data = '1234567890' WHERE id = 1;
```

```
mirea=# UPDATE realtor SET passport_data = '1234567890' WHERE id = 1;
ERROR:  Некорректные паспортные данные: 1234567890 (ожидается формат XXXX УУУУУУ)
CONTEXT:  PL/pgSQL function check_passport_yefremov() line 4 at RAISE
mirea#
```

4. Создать триггер, который при добавлении нового риэлтора формат телефона: `79996667788` преобразует в: `+7 (999) 666 77 88`.

```
CREATE OR REPLACE FUNCTION format_phone_yefremov()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    IF NEW.phone_number ~ '^\d{11}$' THEN
        NEW.phone_number := '+7 (' || SUBSTRING(NEW.phone_number, 2, 3) || ') ' ||
                            SUBSTRING(NEW.phone_number, 5, 3) || ' ' ||
                            SUBSTRING(NEW.phone_number, 8, 2) || ' ' ||
                            SUBSTRING(NEW.phone_number, 10, 2);
    END IF;
    RETURN NEW;
END;
$$;

-- Создание триггера
CREATE TRIGGER phone_format_yefremov
    BEFORE INSERT OR UPDATE ON realtor
    FOR EACH ROW
    EXECUTE FUNCTION format_phone_yefremov();
```

```
INSERT INTO realtor (last_name, first_name, phone_number)
VALUES ('Test', 'Test', '79996667788');
```

```
SELECT phone_number FROM realtor WHERE last_name = 'Test'; -- +7 (999) 666 77 88
```

```
mirea=# INSERT INTO realtor (last_name, first_name, phone_number)
VALUES ('Test', 'Test', '79996667788');
INSERT 0 1
mirea=# SELECT phone_number FROM realtor WHERE last_name = 'Test'; -- +7 (999) 666 77 88
    phone_number
--------------------
 +7 (999) 666 77 88
(1 row)
```