



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

**Институт информационных технологий (ИИТ)
Кафедра цифровой трансформации (ЦТ)**

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ
по дисциплине «Разработка баз данных»

Практическое занятие №6

Студенты группы *ИКБО-20-23 Комисарик М.А.*

(подпись)

Ассистент *Брайловский А.В.*

(подпись)

Отчет представлен «___»____2025 г.

Москва 2025 г.

ПОСТАНОВКА ЗАДАЧИ

Цель работы: Формирование углубленных практических навыков по управлению данными и реализации сложной бизнес-логики в СУБД PostgreSQL с использованием триггеров и курсоров.

Постановка задачи:

1. Проанализировать предметную область своей базы данных и выявить не менее трёх бизнес-правил, реализация которых в виде ограничений целостности возможна только с помощью триггеров.
2. Разработать два скрипта на PL/pgSQL, демонстрирующих оба способа обработки данных:
 - С использованием явного курсора (DECLARE / OPEN / FETCH / CLOSE).
 - С использованием неявного курсора (цикл FOR...IN).

ХОД РАБОТЫ

Начальные данные

	123 ↪ id_employee	123 ↪ id_job_position	A-Z first_name	A-Z second_name	A-Z third_name	A-Z app_account_password_hash	A-Z phone_number
1	1	1	Alice	Johnson	[NULL]	emp_hash_1	555-3333
2	2	2	Bob	Williams	Lee	emp_hash_2	555-4444
3	3	3	Carol	Martinez	[NULL]	emp_hash_3	555-8888
4	4	4	Dave	Anderson	Paul	emp_hash_4	555-9999
5	5	5	Eva	Garcia	Maria	emp_hash_5	555-0000
6	6	2	Frank	Taylor	[NULL]	emp_hash_6	555-1212
7	7	3	Grace	Thomas	Elizabeth	emp_hash_7	555-1313

Рисунок 1 – Содержание таблицы employee

	123 ↪ id_job_position	A-Z name	A-Z salary
1	1	Manager	50000
2	2	Chef	45000
3	3	Delivery Driver	30000
4	4	Cashier	28000
5	5	Assistant Manager	42000

Рисунок 2 – Содержание таблицы job_position

	123 ↪ id_product	A-Z name	A-Z description	123 price
1	1	Margherita Pizza	Classic pizza with tomato and cheese	\$10.99
2	2	Pepperoni Pizza	Pizza with pepperoni slices	\$12.99
3	3	Cola	Refreshing soft drink	\$2.99
4	4	Veggie Supreme	Pizza with assorted vegetables	\$14.99
5	5	Chocolate Cake	Rich chocolate dessert	\$6.99
6	6	Caesar Salad	Fresh salad with Caesar dressing	\$8.99
7	7	Garlic Bread	Toasted bread with garlic butter	\$4.99
8	8	Sprite	Sprite	\$3.49
9	9	Lipton tea	A 1 liter bottle of Lipton tea	\$4.49
10	10	Chocolate muffin	Tasty little chocolate dessert	\$4.99
11	11	Chocolate donut	A chocolate glazed donut	\$5.49

Рисунок 3 – Содержание таблицы product

	123 ↪ id_category	A-Z name	A-Z description
1	1	Pizza	Various types of pizzas
2	2	Beverage	Drinks and beverages
3	3	Appetizer	Starters and snacks
4	4	Dessert	Sweet treats
5	5	Salad	Fresh salads

Рисунок 4 – Содержание таблицы category

	l23 ↪ id_product_ingredient	l23 ↗ id_product	l23 ↗ id_ingredient	l23 ingredient_weight
1	1	1	1	500
2	2	1	2	200
3	3	1	3	150
4	4	2	1	500
5	5	2	2	200
6	6	2	3	150
7	7	2	4	100
8	8	4	1	500
9	9	4	2	200
10	10	4	3	150
11	11	4	5	80
12	12	4	6	70
13	13	4	7	60

Рисунок 5 – Содержание таблицы product_ingredient

	l23 ↪ id_product_category	l23 ↗ id_product	l23 ↗ id_category
1	1	1	1
2	2	2	1
3	3	3	2
4	4	4	1
5	5	5	4
6	6	6	5
7	7	7	3
8	8	8	2
9	9	9	2
10	10	10	4
11	11	11	4

Рисунок 6 – Содержание таблицы product_category

	l23 ↪ id_ingredient	A-Z name
1	1	Flour
2	2	Cheese
3	3	Tomato Sauce
4	4	Pepperoni
5	5	Mushrooms
6	6	Onions
7	7	Olives

Рисунок 7 – Содержание таблицы ingredient

	l23 ↪ id_pizzeria	A-Z phone_number	l23 ↗ id_address
1	1	555-0001	1
2	2	555-0002	2
3	3	555-0003	3
4	4	555-0004	4
5	5	555-0005	5

Рисунок 8 – Содержание таблицы pizzeria

	123 → id_pizzeria_storage	123 ↗ id_pizzeria	123 ↗ id_storage
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5

Рисунок 9 – Содержание таблицы pizzeria_storage

	123 → id_storage_ingredient	123 ↗ id_storage	123 ↗ id_ingredient	123 ingredient_weight
1	4	2	4	2 000
2	5	3	5	1 500
3	6	4	6	1 800
4	7	5	7	1 200
5	2	1	2	5 000
6	1	1	1	10 000
7	3	1	3	3 000

Рисунок 10 – Содержание таблицы storage_ingredient

	123 → id_client	AZ first_name	AZ second_name	AZ third_name	AZ phone_number	123 ↗ id_delivery_address	AZ account_password_id_hash
1	1	John	Doe	[NULL]	555-1111	1	hashed_password_123
2	2	Jane	Smith	Marie	555-2222	2	hashed_password_456
3	3	Mike	Johnson	Robert	555-3333	3	hashed_password_789
4	4	Sarah	Wilson	[NULL]	555-4444	4	hashed_password_101
5	5	David	Brown	James	555-5555	5	hashed_password_112
6	6	Emily	Davis	Anne	555-6666	6	hashed_password_131
7	7	Chris	Miller	Thomas	555-7777	7	hashed_password_415

Рисунок 11 – Содержание таблицы client

	123 → id_address	AZ city	AZ street	123 house_number	123 floor	123 unit_number
1	1	New York	Main St	123	[NULL]	[NULL]
2	2	Los Angeles	Oak Ave	456	3	5
3	3	Chicago	Pine Rd	789	2	[NULL]
4	4	Boston	Maple St	321	1	2
5	5	Seattle	Cedar Ave	654	4	8
6	6	Miami	Palm Blvd	987	[NULL]	[NULL]
7	7	Denver	Elm St	147	2	3

Рисунок 12 – Содержание таблицы address

	123 → id_order	123 ↗ id_pizzeria	123 ↗ id_client	123 ↗ id_delivery_address	123 order_amount	date_ofFormation	123 ↗ delivery_method_id	123 ↗
1	4	1	4	4	\$18.98	2025-09-08	1	1
2	8	1	1	1	\$38.97	2025-11-21	2	1
3	11	1	4	4	\$45.97	2025-06-16	1	2
4	12	1	5	5	\$32.97	2025-06-10	2	1
5	1	1	1	1	\$25.97	2025-11-05	2	3
6	2	1	2	2	\$15.98	2025-02-06	1	2
7	3	1	3	3	\$32.97	2025-09-07	2	3
8	5	1	5	5	\$45.96	2025-02-09	2	2
9	6	1	6	6	\$12.99	2025-06-10	1	3
10	7	1	7	7	\$28.97	2025-09-11	1	4
11	9	1	2	2	\$22.98	2025-02-07	1	2
12	10	1	3	3	\$19.98	2025-11-21	2	1
13	15	1	1	1	\$32.97	2025-11-03	1	3

Рисунок 13 – Содержание таблицы _order.

	123 id_order_product	123 id_order	123 id_product	123 product_count
1	1	1	1	2
2	2	1	3	1
3	3	2	2	1
4	4	3	4	2
5	5	3	3	2
6	6	4	6	1
7	7	4	7	2
8	8	5	5	3
9	9	5	3	3
10	10	6	2	1
11	11	7	1	1
12	12	7	4	1
13	13	7	3	2
14	14	8	2	2
15	15	8	3	2
16	16	9	4	1
17	17	9	7	3
18	18	10	6	2
19	19	10	8	1
20	20	11	1	3
21	21	11	3	1
22	22	12	5	2
23	23	12	9	2
24	24	12	11	1

Рисунок 14 – Содержание таблицы order_product.

1 Триггеры

1.1 Первый триггер

- Таблица: order_product
- Событие: INSERT
- Время срабатывания: BEFORE
- Уровень: FOR EACH ROW
- Логика действий:
 1. Для каждого ингредиента, необходимого для приготовления добавляемого продукта (NEW.id_product), проверить, достаточно ли его веса на складе,
 2. Если хотя бы одного ингредиента недостаточно, прервать операцию с сообщением об ошибке (RAISE EXCEPTION),
 3. Если всех ингредиентов достаточно, разрешить вставку.

Код триггерной функции и триггера:

```

CREATE OR REPLACE FUNCTION check_ingredient_availability_correct()
RETURNS TRIGGER AS $$ 
DECLARE
    v_pizzeria_id INT;
    ingredient_record RECORD;
    required_weight REAL;
    available_weight REAL;
BEGIN
    SELECT id_pizzeria INTO v_pizzeria_id FROM _order WHERE id_order = NEW.id_order;

    FOR ingredient_record IN
        SELECT id_ingredient, ingredient_weight FROM product_ingredient WHERE id_product = NEW.id_product
    LOOP
        required_weight := ingredient_record.ingredient_weight * NEW.product_count;

        SELECT SUM(s_i.ingredient_weight)
        INTO available_weight
        FROM storage_ingredient as s_i
        JOIN pizzeria_storage as p_s ON s_i.id_storage = p_s.id_storage
        WHERE p_s.id_pizzeria = v_pizzeria_id AND s_i.id_ingredient = ingredient_record.id_ingredient;

        available_weight := COALESCE(available_weight, 0);

        IF available_weight < required_weight THEN
            RAISE EXCEPTION 'Недостаточно ингредиента (ID: %) на складах пиццерии (ID: %). Требуется: %, в наличии: %',
                ingredient_record.id_ingredient,
                v_pizzeria_id,
                required_weight,
                available_weight;
        END IF;
    END LOOP;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE or replace TRIGGER before_order_product_insert
BEFORE INSERT ON order_product
FOR EACH ROW
EXECUTE FUNCTION check_ingredient_availability_correct();

```

Рисунок 15

Name	Value
Start time	Mon Nov 10 05:32:25 MSK 2025
Finish time	Mon Nov 10 05:32:25 MSK 2025
Query	CREATE OR REPLACE FUNCTION check_ingredient_availability_correct() RETURNS TRIGGER AS \$\$ DECLARE v_pizzeria_id INT; ingredient_record RECORD; required_weight REAL; available_weight REAL; BEGIN SELECT id_pizzeria INTO v_pizzeria_id FROM _order WHERE id_order = NEW.id_order; FOR ingredient_record IN SELECT id_ingredient, ingredient_weight FROM product_ingredient WHERE id_product = NEW.id_product LOOP required_weight := ingredient_record.ingredient_weight * NEW.product_count; SELECT SUM(s_i.ingredient_weight) INTO available_weight FROM storage_ingredient as s_i JOIN pizzeria_storage as p_s ON s_i.id_storage = p_s.id_storage WHERE p_s.id_pizzeria = v_pizzeria_id AND s_i.id_ingredient = ingredient_record.id_ingredient; available_weight := COALESCE(available_weight, 0); IF available_weight < required_weight THEN RAISE EXCEPTION 'Недостаточно ингредиента (ID: %) на складах пиццерии (ID: %). Требуется: %, в наличии: %', ingredient_record.id_ingredient, v_pizzeria_id, required_weight, available_weight; END IF; END LOOP; RETURN NEW; END; \$\$ LANGUAGE plpgsql; CREATE or replace TRIGGER before_order_product_insert BEFORE INSERT ON order_product FOR EACH ROW EXECUTE FUNCTION check_ingredient_availability_correct();

Рисунок 16

Name	Value
Updated Rows	0
Execute time	0.0s
Start time	Mon Nov 10 05:37:04 MSK 2025
Finish time	Mon Nov 10 05:37:04 MSK 2025
Query	CREATE or replace TRIGGER before_order_product_insert BEFORE INSERT ON order_product FOR EACH ROW EXECUTE FUNCTION check_ingredient_availability_correct()

Рисунок 17

Неудачный запрос:

The screenshot shows a PostgreSQL error window. At the top, there is a toolbar with icons for back, forward, and search, followed by a status bar showing 'Execution Error'. Below the toolbar is a message area with a warning icon and the text 'Error occurred during SQL query execution'. Underneath, it says 'Причина:' (Cause:) followed by the error message: 'SQL Error [P0001]: ERROR: Недостаточно ингредиента (ID: 1) на складах пиццерии (ID: 1). Требуется: 50000, в наличии: 8500' and 'Где: PL/pgSQL function check_ingredient_availability_correct() line 24 at RAISE'. At the bottom of the window are buttons for 'Остановить' (Stop) and 'Повторить' (Repeat). A scrollable log area below the message window displays the same error message twice.

```
• INSERT INTO order_product (id_order, id_product, product_count)
VALUES (1, 1, 100);
```

Рисунок 18

Удачный запрос представлен в следующем пункте.

1.2 Второй триггер

- Таблица: order_product
- Событие: INSERT, UPDATE, DELETE
- Время срабатывания: AFTER
- Уровень: FOR EACH ROW
- Логика действий:
 1. Определить id_order, для которого произошло изменение,

2. После изменения в `order_product` пересчитать полную стоимость всех товаров для этого заказа, суммируя произведения количества (`product_count`) на цену (`price`) из таблицы `product`,
3. Обновить поле `order_amount` в таблице `_order` полученным значением.

Код триггерной функции и триггера:

```

●CREATE OR REPLACE FUNCTION update_order_amount()
RETURNS TRIGGER AS $$$
DECLARE
    target_order_id INT;
    new_total_amount MONEY;
BEGIN
    IF (TG_OP = 'DELETE') THEN
        target_order_id := OLD.id_order;
    ELSE
        target_order_id := NEW.id_order;
    END IF;

    SELECT SUM(p.price * o_p.product_count)
    INTO new_total_amount
    FROM order_product o_p
    JOIN product p ON o_p.id_product = p.id_product
    WHERE o_p.id_order = target_order_id;

    UPDATE _order
    SET order_amount = COALESCE(new_total_amount, '$0.00')
    WHERE id_order = target_order_id;

    IF (TG_OP = 'DELETE') THEN
        RETURN OLD;
    ELSE
        RETURN NEW;
    END IF;
END;
$$ LANGUAGE plpgsql;

●CREATE TRIGGER after_order_product_change
AFTER INSERT OR UPDATE OR DELETE ON order_product
FOR EACH ROW
EXECUTE FUNCTION update_order_amount();

```

Рисунок 19

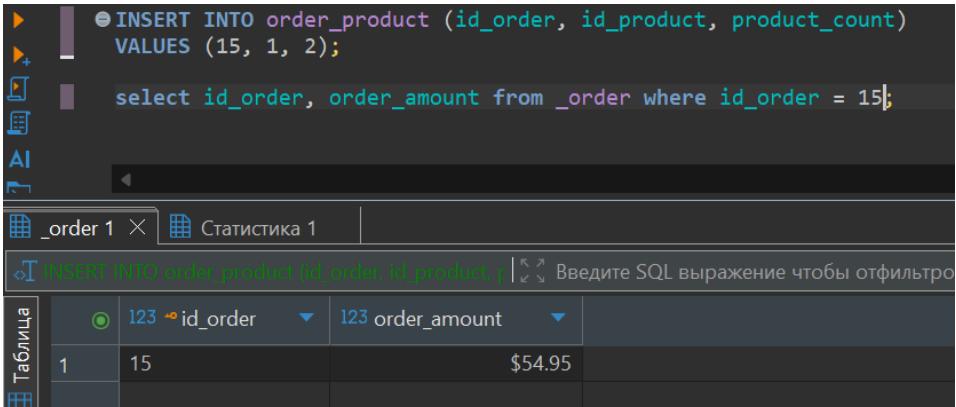
Статистика 1	
Name	Value
Updated Rows	0
Execute time	0.0s
Start time	Mon Nov 10 05:49:28 MSK 2025
Finish time	Mon Nov 10 05:49:28 MSK 2025
Query	<pre> CREATE OR REPLACE FUNCTION update_order_amount() RETURNS TRIGGER AS \$\$\$ DECLARE target_order_id INT; new_total_amount MONEY; BEGIN IF (TG_OP = 'DELETE') THEN target_order_id := OLD.id_order; ELSE target_order_id := NEW.id_order; END IF; SELECT SUM(p.price * o_p.product_count) INTO new_total_amount FROM order_product o_p JOIN product p ON o_p.id_product = p.id_product WHERE o_p.id_order = target_order_id; UPDATE _order SET order_amount = COALESCE(new_total_amount, '\$0.00') WHERE id_order = target_order_id; IF (TG_OP = 'DELETE') THEN RETURN OLD; ELSE RETURN NEW; END IF; END; \$\$ LANGUAGE plpgsql; CREATE TRIGGER after_order_product_change AFTER INSERT OR UPDATE OR DELETE ON order_product FOR EACH ROW EXECUTE FUNCTION update_order_amount(); </pre>

Рисунок 20

Name	Value
Updated Rows	0
Execute time	0.0s
Start time	Mon Nov 10 05:50:23 MSK 2025
Finish time	Mon Nov 10 05:50:23 MSK 2025
Query	CREATE TRIGGER after_order_product_change AFTER INSERT OR UPDATE OR DELETE ON order_product FOR EACH ROW EXECUTE FUNCTION update_order_amount()

Рисунок 21

Тестирование вставки:



```

INSERT INTO order_product (id_order, id_product, product_count)
VALUES (15, 1, 2);

select id_order, order_amount from _order where id_order = 15;

```

Статистика 1

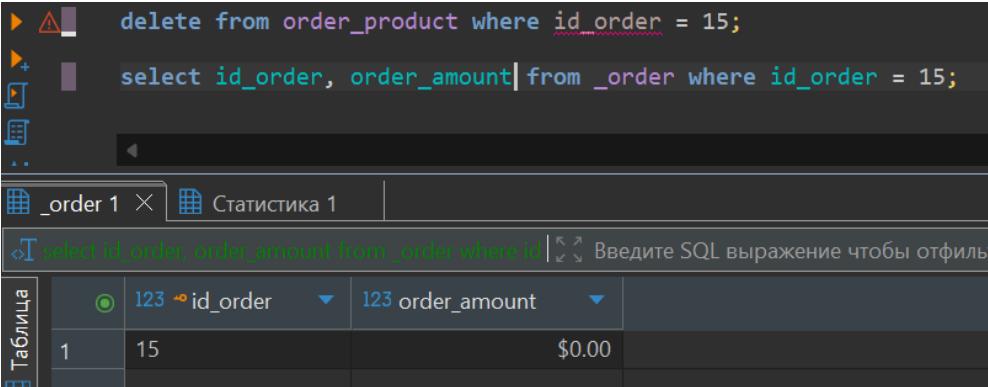
_order 1 × Статистика 1

Ведите SQL выражение чтобы отфильтровать

Таблица	123 id_order	123 order_amount
	15	\$54.95

Рисунок 22

Тестирование удаления:



```

delete from order_product where id_order = 15;

select id_order, order_amount from _order where id_order = 15;

```

Статистика 1

Ведите SQL выражение чтобы отфильтровать

Таблица	123 id_order	123 order_amount
	15	\$0.00

Рисунок 23

1.3 Третий триггер

- Таблица: job_position
- Событие: UPDATE
- Время срабатывания: AFTER

- Уровень: FOR EACH ROW
- Логика действий:
 1. Проверить, изменилась ли зарплата (salary).
 2. Если да, выполнить INSERT в таблицу job_position_audit, используя старое значение из OLD, новое из NEW, а также NOW().

Код триггерной функции и триггера:

```

CREATE OR REPLACE FUNCTION log_job_position_changes()
RETURNS TRIGGER AS $$ 
BEGIN
    IF OLD.name IS DISTINCT FROM NEW.name OR OLD.salary IS DISTINCT FROM NEW.salary THEN
        INSERT INTO job_position_audit (
            job_position_id,
            old_salary,
            new_salary,
            change_timestamp
        )
        VALUES (
            OLD.id_job_position,
            OLD.salary,
            NEW.salary,
            NOW()
        );
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_job_position_update
AFTER UPDATE ON job_position
FOR EACH ROW
EXECUTE FUNCTION log_job_position_changes();

```

Рисунок 24

Статистика 1	
Name	Value
Updated Rows	0
Execute time	0.0s
Start time	Mon Nov 10 06:31:44 MSK 2025
Finish time	Mon Nov 10 06:31:44 MSK 2025
Query	<pre> CREATE OR REPLACE FUNCTION log_job_position_changes() RETURNS TRIGGER AS \$\$ BEGIN IF OLD.name IS DISTINCT FROM NEW.name OR OLD.salary IS DISTINCT FROM NEW.salary THEN INSERT INTO job_position_audit (job_position_id, old_salary, new_salary, change_timestamp) VALUES (OLD.id_job_position, OLD.salary, NEW.salary, NOW()); END IF; RETURN NEW; END; \$\$ LANGUAGE plpgsql; CREATE TRIGGER after_job_position_update AFTER UPDATE ON job_position FOR EACH ROW EXECUTE FUNCTION log_job_position_changes(); </pre>

Рисунок 25

Name	Value
Updated Rows	0
Execute time	0.0s
Start time	Mon Nov 10 06:24:08 MSK 2025
Finish time	Mon Nov 10 06:24:08 MSK 2025
Query	<pre>CREATE TRIGGER after_job_position_update AFTER UPDATE ON job_position FOR EACH ROW EXECUTE FUNCTION log_job_position_changes()</pre>

Рисунок 26

Запрос без изменений:

```

update job_position
set salary = 30000
where id_job_position = 3;

select count(*) from job_position_audit;

```

Результат 1 × Статистика 1

count
0

Рисунок 27

Запрос с изменениями:

```

update job_position
set salary = 31000
where id_job_position = 3;

select * from job_position_audit;

```

job_position_audit 1 × Статистика 1

audit_id	job_position_id	old_salary	new_salary	change_timestamp
1	3	30000	31000	2025-11-10 06:32:43.653 +0300

Рисунок 28

2 Курсоры

2.1 Явный курсор

The screenshot shows a PostgreSQL terminal window with a dark theme. On the left is a vertical toolbar with icons for file operations, settings, and help. The main area contains a SQL script and its output. The script is as follows:

```
DO $$  
DECLARE  
    employee_cursor CURSOR FOR  
        SELECT e.first_name, e.second_name, j_p.name AS job_name  
        FROM employee e  
        JOIN job_position as j_p ON e.id_job_position = j_p.id_job_position;  
    employee_record RECORD;  
BEGIN  
    OPEN employee_cursor;  
  
    LOOP  
        FETCH employee_cursor INTO employee_record;  
        EXIT WHEN NOT FOUND;  
  
        RAISE NOTICE 'Сотрудник: %, Должность: %',  
            employee_record.first_name || ' ' || employee_record.second_name,  
            employee_record.job_name;  
    END LOOP;  
  
    CLOSE employee_cursor;  
END;  
$$ LANGUAGE plpgsql;
```

Below the script, there are two tabs: "Статистика 1" and "Вывод". The "Вывод" tab is selected, showing the following output:

```
Сотрудник: Alice Johnson, Должность: Manager  
Сотрудник: Bob Williams, Должность: Chef  
Сотрудник: Carol Martinez, Должность: Delivery Driver  
Сотрудник: Dave Anderson, Должность: Cashier  
Сотрудник: Eva Garcia, Должность: Assistant Manager  
Сотрудник: Frank Taylor, Должность: Chef  
Сотрудник: Grace Thomas, Должность: Delivery Driver
```

Рисунок 29

2.2 Неявный курсор

Выполнено в пункте 1.1.