



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 №6**  
по дисциплине «Разработка баз данных»

Студент группы *ИКБО-50-23. Павлов Н.С..*

\_\_\_\_\_  
(подпись)

Преподаватель *Мажей Я. В.*

\_\_\_\_\_  
(подпись)

Москва 2025 г.

## СОДЕРЖАНИЕ

1. ПОСТАНОВКА ЗАДАЧИ.....	3
2 ВЫПОЛНЕНИЕ РАБОТЫ.....	4
2.1 ИСХОДНЫЕ ТАБЛИЦЫ.....	4
2.2 ПОДГОТОВКА БД.....	7
2.3 ЗАДАНИЕ 1 .....	8
2.3.1 Бизнес-правило 1 .....	8
2.3.2 Бизнес-правило 2.....	10
2.3.3 Бизнес-правило 3.....	12
2.4 ЗАДАНИЕ 2 .....	14

## 1. ПОСТАНОВКА ЗАДАЧИ

**Цель:** формирование у студентов углубленных практических навыков по управлению данными и реализации сложной бизнес-логики в СУБД PostgreSQL с использованием триггеров и курсоров.

### **Задачи:**

#### **Задание №1: триггеры**

Проанализировать предметную область своей базы данных и выявить не менее трёх бизнес-правил, реализация которых в виде ограничений целостности возможна только с помощью триггеров.

Для каждого правила:

- описать алгоритм его работы, указав таблицу, событие и последовательность действий;
- написать код триггерной функции на PL/pgSQL и оператор CREATE TRIGGER;
- продемонстрировать работу триггера на примерах DML-операций, которые как успешно выполняются, так и корректно прерываются триггером (два запроса).

#### **Задание №2: курсоры**

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).

Каждый SQL-запрос сопровождать комментарием, объясняющим его назначение и логику работы с учетом специфики вашей базы данных.

## 2 ВЫПОЛНЕНИЕ РАБОТЫ

### 2.1 ИСХОДНЫЕ ТАБЛИЦЫ

	123 spare_part_id	A? title	A? specifications	123 price	123 supplier_id
1	4	Манетка Shimano Ultegra R8050	Электронная, 11-скоростная	8 500	5
2	5	Цепь Shimano HG701	11-скоростная, 116 звеньев	2 500	5
3	6	Звезды Shimano 105 R7000	Кассета 11-34T, 11 скоростей	3 800	5
4	7	Вилка RockShox Sid SL Ultimate	Карбон, ход 100mm, дисковые тормоза	25 000	8
5	8	Батарея Shimano Di2	BT-DN110, для электронного переключения	4 800	5
6	9	Подшипники Enduro	Комплект для втулок, промышленные	1 800	3
7	10	Трансмиссия SRAM Force eTap AXS	Беспроводная, 12-скоростная	45 000	6
8	32	Вилка Fox 36 Factory	Ход 160mm, воздушная пружина, амортизатор GRIP2, 29"/27.5", дисковые тормоза	125 000	9
9	33	Амортизатор Fox Float X2 Factory	Воздушный, регулировка высоко/низкоскоростного отскока и компрессии, 230x65mm	85 000	9
10	34	Вилка Fox 34 Performance Elite	Ход 130mm, воздушная пружина, амортизатор FIT4, 29"/27.5", дисковые тормоза	75 000	9
11	35	Амортизатор Fox DHX2 Factory	Пружинный, регулировка высоко/низкоскоростного отскока, 250x75mm	92 000	9
12	36	Вилка Fox 40 Factory	Двухкоронная, ход 190-203mm, воздушная пружина, 27.5", для даунхилла	140 000	9
13	37	Группа SRAM RED eTap AXS	Беспроводная электронная группа, 12-скоростная, гидравлические дисковые тормоза	280 000	6
14	38	Группа SRAM XX SL Eagle AXS	Беспроводная электронная группа для MTB, 12-скоростная, 10-52T	220 000	6
15	39	Тормоза SRAM Level Ultimate	Гидравлические дисковые тормоза, 4-поршневые, с технологией SwingLink	45 000	6
16	40	Система SRAM XX1 Eagle Carbon	Шатуны из карбона, 32T, каретка DUB, 12-скоростная	68 000	6
17	41	Кассета SRAM XG-1299 Eagle	12-скоростная, 10-52T, титановые шестерни, вес 268g	55 000	6
18	65	Тормозные диски Shimano RT800	Ротор 160mm, Centerlock, для горных велосипедов	3 200	5
19	68	Тормозные колодки Shimano	Для дисковых тормозов	1 200	5
20	1	Камера велосипедная 700x25-32	Резиновая, Presta, 48mm	935	1
21	2	Тормозные колодки Shimano R55C4	Для дисковых тормозов, керамические	1 260	5
22	3	Обод Mavic Ksyrium Elite	Новые характеристики	4 500	2

Рисунок 1 – Таблица spare\_part

	123 work_spare_part_id	123 work_id	123 spare_part_id
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10

Рисунок 2 – Таблица work\_spare\_part

	123 work_id	A? title	A? description_of_the_execution	123 lead_time	123 price
1	3	Правка обода	Выравнивание обода на станке, регулировка спиц	90	1 500
2	4	Регулировка переключения	Настройка переднего и заднего переключателей	60	1 000
3	5	Полная регулировка трансмиссии	Чистка, смазка, регулировка всей трансмиссии	120	2 000
4	6	Замена цепи и звезд	Замена цепи и кассеты, регулировка	75	1 200
5	7	Обслуживание вилки	Разборка, чистка, замена масла, сборка	180	3 000
6	8	Установка оборудования	Установка и настройка нового оборудования	60	1 000
7	9	Диагностика электроники	Проверка электронной системы, перепрошивка	90	1 500
8	10	Комплексное обслуживание	Полная диагностика и обслуживание всех систем	240	4 000
9	1	Замена камеры	Демонтаж покрышки, замена камеры, монтаж покрышки	30	585
10	2	Замена тормозных колодок	Снятие старых колодок, установка новых, регулировка	45	860

Рисунок 3 – Таблица work

The screenshot shows the 'supplier' table in the dbstud application. The table has 5 columns: supplier\_id, organization\_title, address, and phone. The data is as follows:

supplier_id	organization_title	address	phone
1	Велокомплект	г. Москва, ул. Ленина, 25	+74951234567
2	СпортЗапчасти	г. Санкт-Петербург, Невский пр., 100	+78121234567
3	БайкТек	г. Екатеринбург, ул. Мира, 15	+73431234567
4	ВелоМир	г. Новосибирск, Красный пр., 50	+73831234567
5	Шимано Рус	г. Москва, Ленинградский пр., 80	+74959876543
6	СРАМ Дистрибьюшн	г. Казань, ул. Баумана, 30	+78431234567
7	Кампагноло	г. Москва, ул. Тверская, 45	+74957778899
8	РокШок Сервис	г. Краснодар, ул. Красная, 120	+78611234567
9	Фокс Раша	г. Сочи, ул. Курортная, 25	+86221234567
10	ДТ Швейцария	г. Москва, Кутузовский пр., 32	+74956667788

Рисунок 4 – Таблица supplier

The screenshot shows the 'client' table in the dbstud application. The table has 7 columns: client\_id, surname, name, fathers\_name, phone, and email. The data is as follows:

client_id	surname	name	fathers_name	phone	email
1	Иванов	Алексей	Петрович	+79161234567	alex.ivanov@mail.ru
2	Петрова	Мария	Сергеевна	+79161234568	maria.petrova@gmail.com
3	Сидоров	Дмитрий	Игоревич	+79161234569	dmitry.sidorov@yandex.ru
4	Козлова	Анна	Владимировна	+79161234570	anna.kozlova@mail.ru
5	Никитин	Сергей	Александрович	+79161234571	sergey.nikitin@gmail.com
6	Федорова	Елена	Дмитриевна	+79161234572	elena.fedorova@yandex.ru
7	Морозов	Иван	[NULL]	+79161234573	ivan.morozov@mail.ru
8	Павлова	Ольга	Алексеевна	+79161234574	olga.pavlova@gmail.com
9	Лебедев	Андрей	Викторович	+79161234575	andrey.lebedev@yandex.ru
10	Семенова	Татьяна	Николаевна	+79161234576	tatyana.semenova@mail.ru

Рисунок 5 – Таблица client

The screenshot shows the 'application' table in the dbstud application. The table has 5 columns: application\_id, client\_id, application\_date, and problem\_description. The data is as follows:

application_id	client_id	application_date	problem_description
1	1	2024-01-15	Прокол задней камеры, требуется замена
2	2	2024-01-16	Износ тормозных колодок, скрип при торможении
3	3	2024-01-17	Погнутый обод переднего колеса
4	4	2024-01-18	Проблемы с переключением передач
5	5	2024-01-19	Требуется полная регулировка трансмиссии
6	6	2024-01-20	Замена цепи и звезд
7	7	2024-01-21	Обслуживание вилки и амортизаторов
8	8	2024-01-22	Установка нового оборудования
9	9	2024-01-23	Диагностика электронной системы переключения
10	10	2024-01-24	Комплексное обслуживание после сезона
11	1	2024-02-01	Тестовый заказ

Рисунок 6 – Таблица application

user\_order

	user_order_id	application_id	status_id	start_date	end_date
1	1	1	5	2024-01-15	2024-01-16
2	2	2	4	2024-01-16	2024-01-18
3	3	3	3	2024-02-17	2024-02-20
4	4	4	2	2024-07-18	2024-07-19
5	5	5	1	2024-07-19	2024-07-22
6	6	6	6	2024-11-20	2024-11-21
7	7	7	4	2025-01-21	2025-01-23
8	8	8	5	2025-01-22	2025-01-23
9	9	9	2	2025-01-23	2025-01-25
10	10	10	1	2025-01-24	2025-01-26
11	11	11	1	2024-02-01	2024-02-05

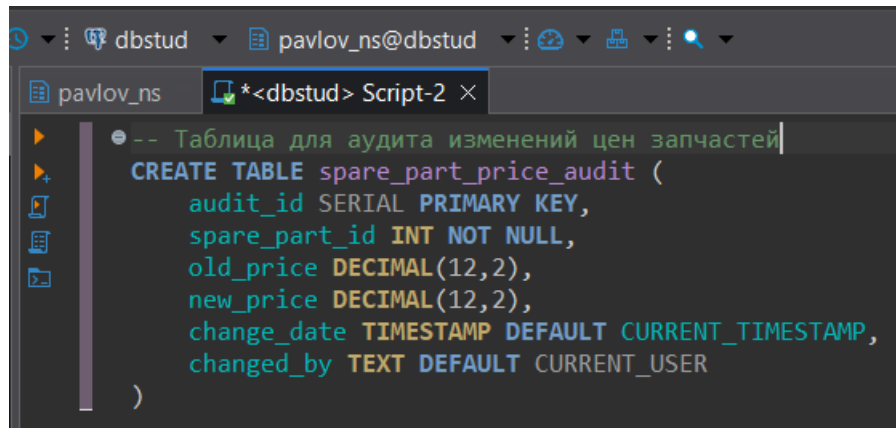
Рисунок 7 – Таблица user\_order

estimate

	estimate_id	user_order_id	price_of_work	price_of_spare_parts	total_price
1	1	1	500	850	1 350
2	2	2	800	1 200	2 000
3	3	3	1 500	4 500	6 000
4	4	4	1 000	8 500	9 500
5	5	5	2 000	2 500	4 500
6	6	6	1 200	3 800	5 000
7	7	7	3 000	25 000	28 000
8	8	8	1 000	4 800	5 800
9	9	9	1 500	1 800	3 300
10	10	10	4 000	45 000	49 000
11	11	11	2 000	1 500	3 500

Рисунок 8 – Таблица estimate

## 2.2 ПОДГОТОВКА БД



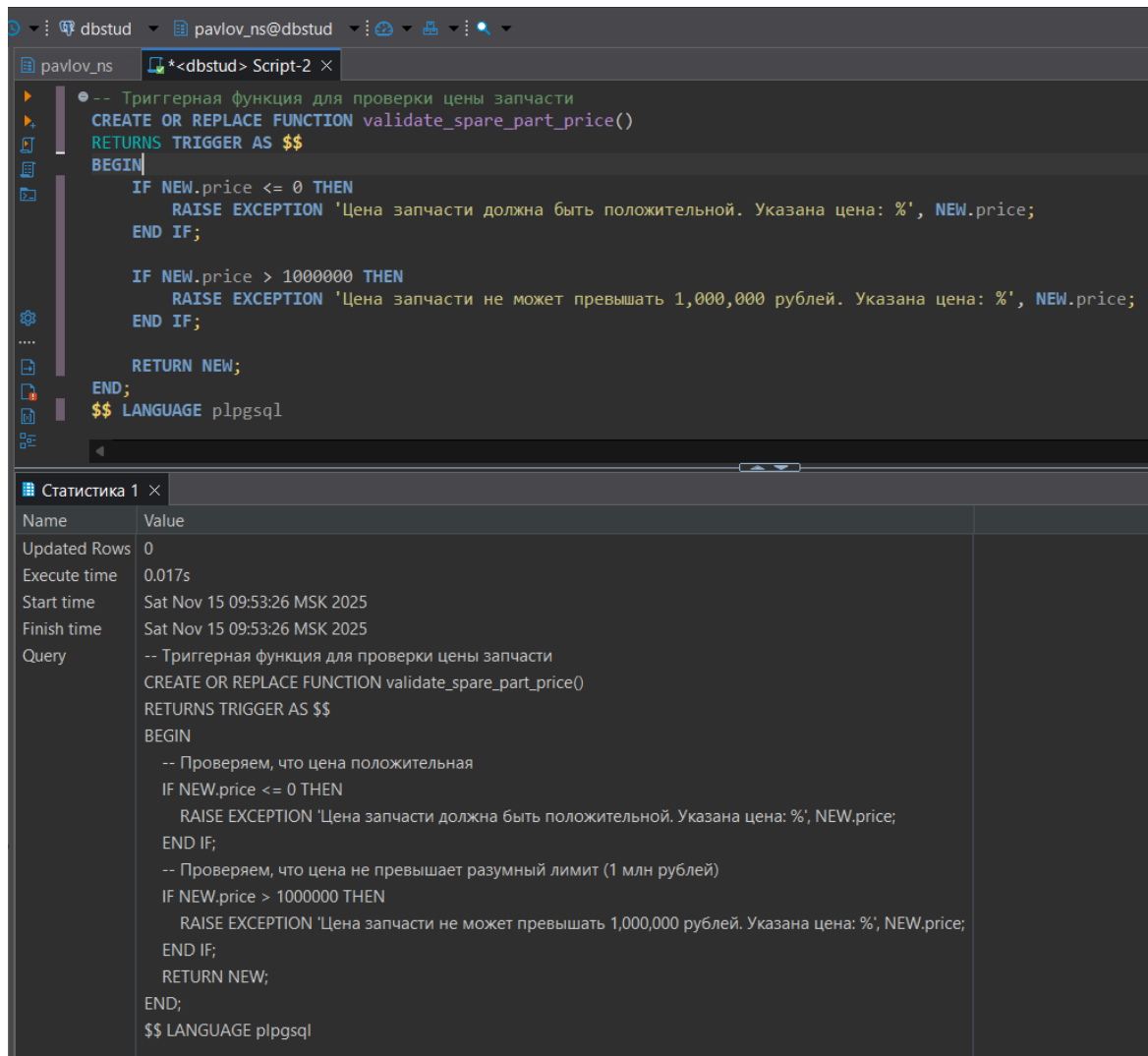
The screenshot shows a database client interface with a dark theme. The top bar displays the connection 'pavlov\_ns@dbstud'. Below it, a tab labeled 'pavlov\_ns' is active, and a script editor window titled '\*<dbstud> Script-2' is open. The script editor contains a SQL statement to create a table named 'spare\_part\_price\_audit'. The table has five columns: 'audit\_id' (SERIAL PRIMARY KEY), 'spare\_part\_id' (INT NOT NULL), 'old\_price' (DECIMAL(12,2)), 'new\_price' (DECIMAL(12,2)), 'change\_date' (TIMESTAMP DEFAULT CURRENT\_TIMESTAMP), and 'changed\_by' (TEXT DEFAULT CURRENT\_USER). A comment at the top of the script reads '-- Таблица для аудита изменений цен запчастей'.

```
-- Таблица для аудита изменений цен запчастей
CREATE TABLE spare_part_price_audit (
    audit_id SERIAL PRIMARY KEY,
    spare_part_id INT NOT NULL,
    old_price DECIMAL(12,2),
    new_price DECIMAL(12,2),
    change_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    changed_by TEXT DEFAULT CURRENT_USER
)
```

Рисунок 9 – Добавление таблицы для аудита изменения цен

## 2.3 ЗАДАНИЕ 1

### 2.3.1 Бизнес-правило 1



The screenshot shows the dbstud interface with a script editor and a statistics window. The script editor contains the following SQL code:

```
-- Триггерная функция для проверки цены запчастей
CREATE OR REPLACE FUNCTION validate_spare_part_price()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.price <= 0 THEN
        RAISE EXCEPTION 'Цена запчасти должна быть положительной. Указана цена: %', NEW.price;
    END IF;

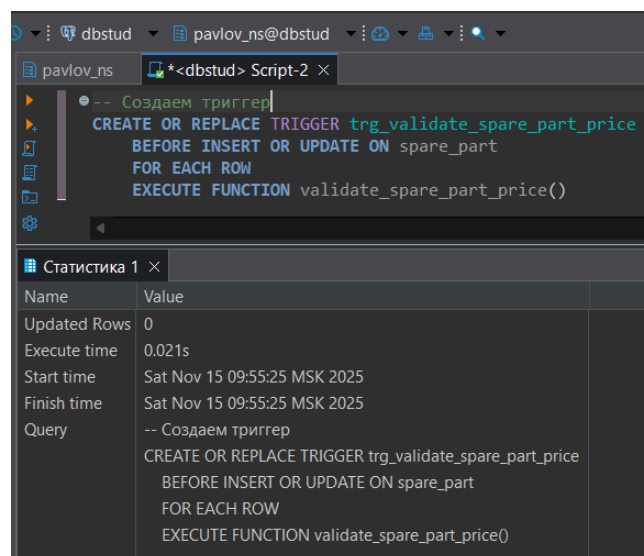
    IF NEW.price > 1000000 THEN
        RAISE EXCEPTION 'Цена запчасти не может превышать 1,000,000 рублей. Указана цена: %', NEW.price;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql
```

The statistics window shows the following information:

Name	Value
Updated Rows	0
Execute time	0.017s
Start time	Sat Nov 15 09:53:26 MSK 2025
Finish time	Sat Nov 15 09:53:26 MSK 2025
Query	-- Триггерная функция для проверки цены запчастей CREATE OR REPLACE FUNCTION validate_spare_part_price() RETURNS TRIGGER AS \$\$ BEGIN -- Проверяем, что цена положительная IF NEW.price <= 0 THEN RAISE EXCEPTION 'Цена запчасти должна быть положительной. Указана цена: %', NEW.price; END IF; -- Проверяем, что цена не превышает разумный лимит (1 млн рублей) IF NEW.price > 1000000 THEN RAISE EXCEPTION 'Цена запчасти не может превышать 1,000,000 рублей. Указана цена: %', NEW.price; END IF; RETURN NEW; END; \$\$ LANGUAGE plpgsql

Рисунок 10 – Создание триггерной функции



The screenshot shows the dbstud interface with a script editor and a statistics window. The script editor contains the following SQL code:

```
-- Создаем триггер
CREATE OR REPLACE TRIGGER trg_validate_spare_part_price
BEFORE INSERT OR UPDATE ON spare_part
FOR EACH ROW
EXECUTE FUNCTION validate_spare_part_price()
```

The statistics window shows the following information:

Name	Value
Updated Rows	0
Execute time	0.021s
Start time	Sat Nov 15 09:55:25 MSK 2025
Finish time	Sat Nov 15 09:55:25 MSK 2025
Query	-- Создаем триггер CREATE OR REPLACE TRIGGER trg_validate_spare_part_price BEFORE INSERT OR UPDATE ON spare_part FOR EACH ROW EXECUTE FUNCTION validate_spare_part_price()

Рисунок 11 – Создание триггера



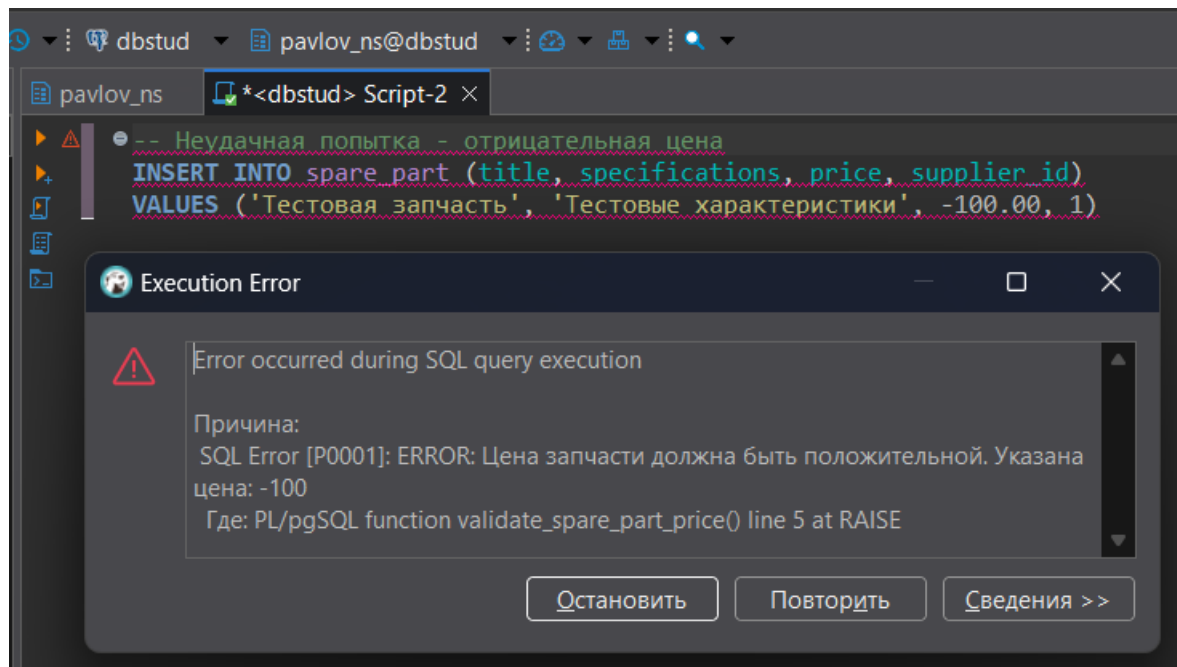


Рисунок 12 – Попытка добавления значения с отрицательной ценой

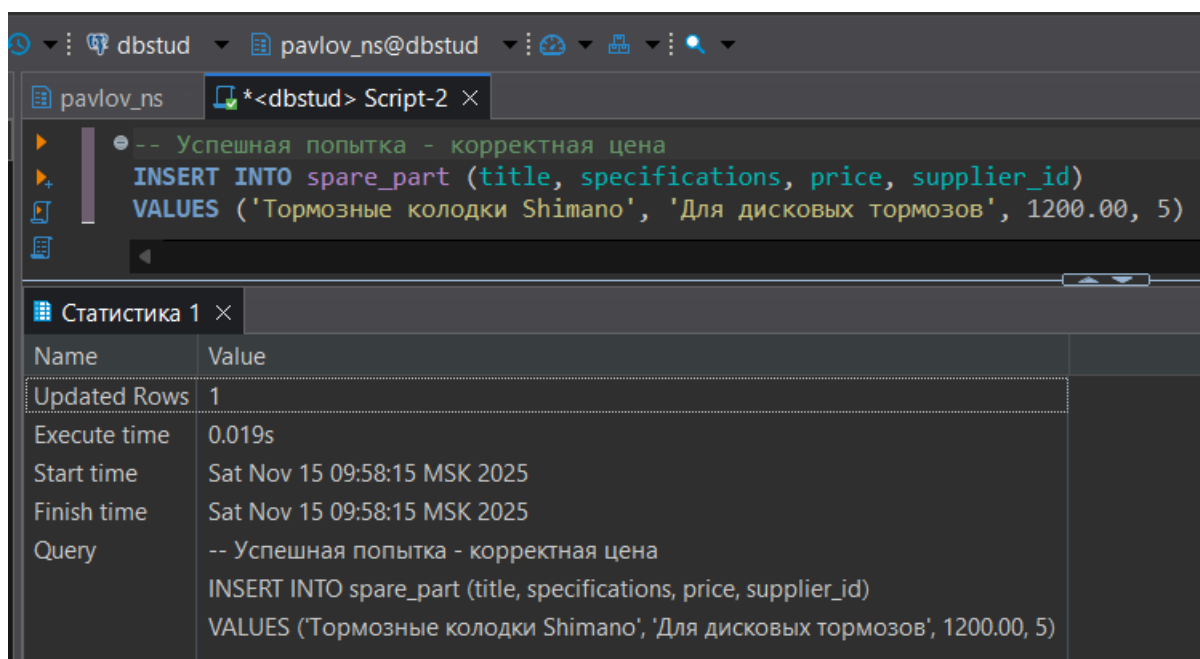


Рисунок 13 – Попытка с корректными значениями

### 2.3.2 Бизнес-правило 2

The screenshot displays a database IDE interface. The top pane shows a SQL script for creating a trigger function. The script is as follows:

```
-- Триггерная функция для пересчета стоимости работ
CREATE OR REPLACE FUNCTION update_work_prices_after_part_change()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'UPDATE' AND NEW.price != OLD.price THEN
        UPDATE work w
        SET price = w.price + (NEW.price - OLD.price) * (
            SELECT COUNT(*)
            FROM work_spare_part wsp
            WHERE wsp.work_id IN (
                SELECT work_id FROM work_spare_part WHERE spare_part_id = NEW.spare_part_id
            )
        )
        WHERE w.work_id IN (
            SELECT work_id FROM work_spare_part WHERE spare_part_id = NEW.spare_part_id
        );
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql
```

The bottom pane, titled "Статистика 1", shows the execution statistics for the query. The statistics are as follows:

Name	Value
Updated Rows	0
Execute time	0.030s
Start time	Sat Nov 15 10:08:18 MSK 2025
Finish time	Sat Nov 15 10:08:18 MSK 2025
Query	-- Триггерная функция для пересчета стоимости работ CREATE OR REPLACE FUNCTION update_work_prices_after_part_change() RETURNS TRIGGER AS \$\$ BEGIN IF TG_OP = 'UPDATE' AND NEW.price != OLD.price THEN UPDATE work w SET price = w.price + (NEW.price - OLD.price) * ( SELECT COUNT(*) FROM work_spare_part wsp WHERE wsp.work_id IN ( SELECT work_id FROM work_spare_part WHERE spare_part_id = NEW.spare_part_id ) ) WHERE w.work_id IN ( SELECT work_id FROM work_spare_part WHERE spare_part_id = NEW.spare_part_id ); END IF; RETURN NEW; END; \$\$ LANGUAGE plpgsql

Рисунок 14 – Создание триггерной функции

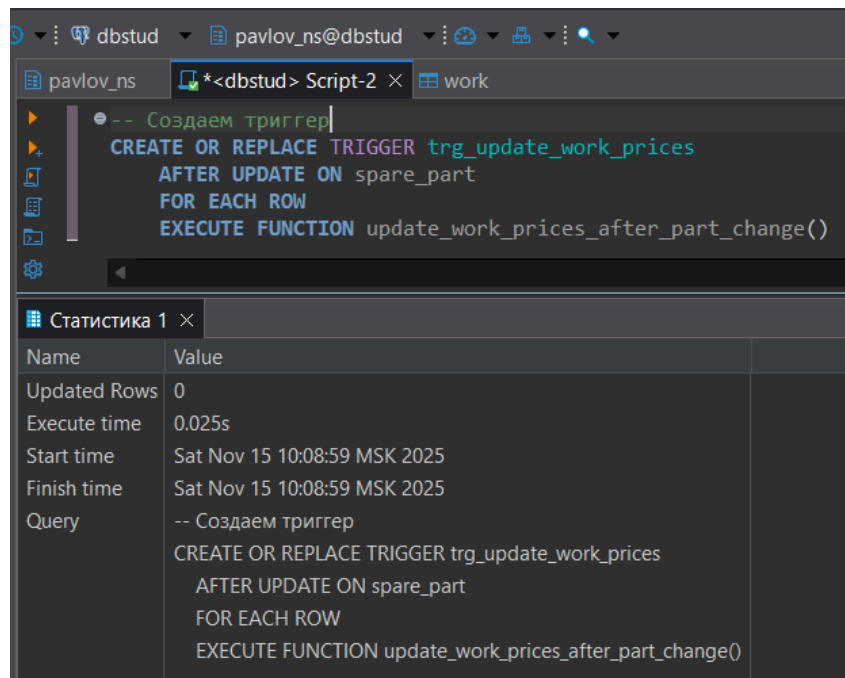


Рисунок 15 – Создание триггера

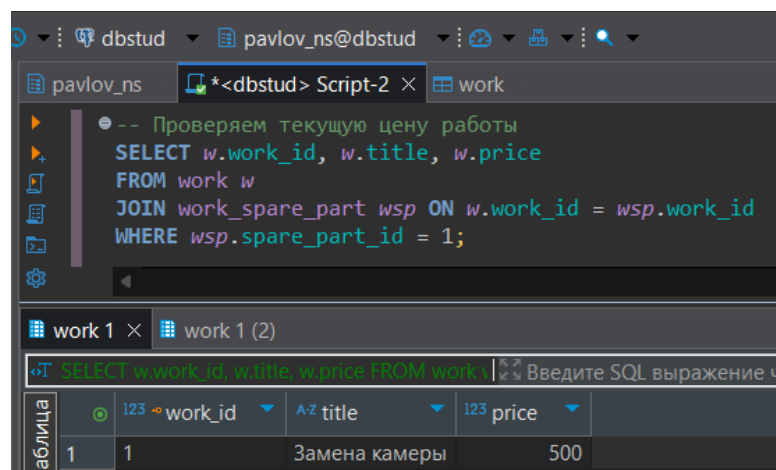


Рисунок 16 – Вывод исходного состояния

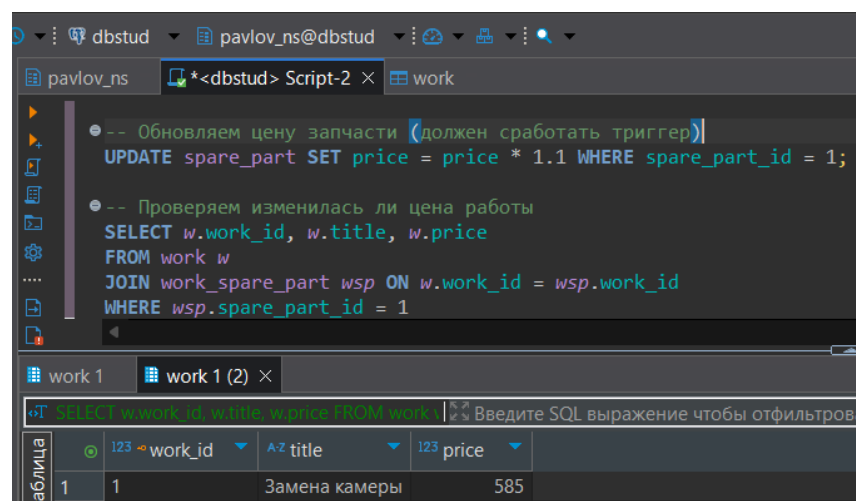


Рисунок 17 – Проверка работы триггера

### 2.3.3 Бизнес-правило 3

The screenshot displays a database IDE interface. The top pane shows a SQL script for creating a trigger function. The script is as follows:

```
-- Триггерная функция для аудита изменений цен
CREATE OR REPLACE FUNCTION audit_spare_part_price_changes()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'UPDATE' AND NEW.price IS DISTINCT FROM OLD.price THEN
        INSERT INTO spare_part_price_audit (spare_part_id, old_price, new_price)
        VALUES (NEW.spare_part_id, OLD.price, NEW.price);
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql
```

The bottom pane, titled "Статистика 1", shows the execution statistics for the query:

Name	Value
Updated Rows	0
Execute time	0.029s
Start time	Sat Nov 15 10:17:31 MSK 2025
Finish time	Sat Nov 15 10:17:53 MSK 2025
Query	-- Триггерная функция для аудита изменений цен CREATE OR REPLACE FUNCTION audit_spare_part_price_changes() RETURNS TRIGGER AS \$\$ BEGIN IF TG_OP = 'UPDATE' AND NEW.price IS DISTINCT FROM OLD.price THEN INSERT INTO spare_part_price_audit (spare_part_id, old_price, new_price) VALUES (NEW.spare_part_id, OLD.price, NEW.price); END IF; RETURN NEW; END; \$\$ LANGUAGE plpgsql

Рисунок 18 – Создание триггерной функции

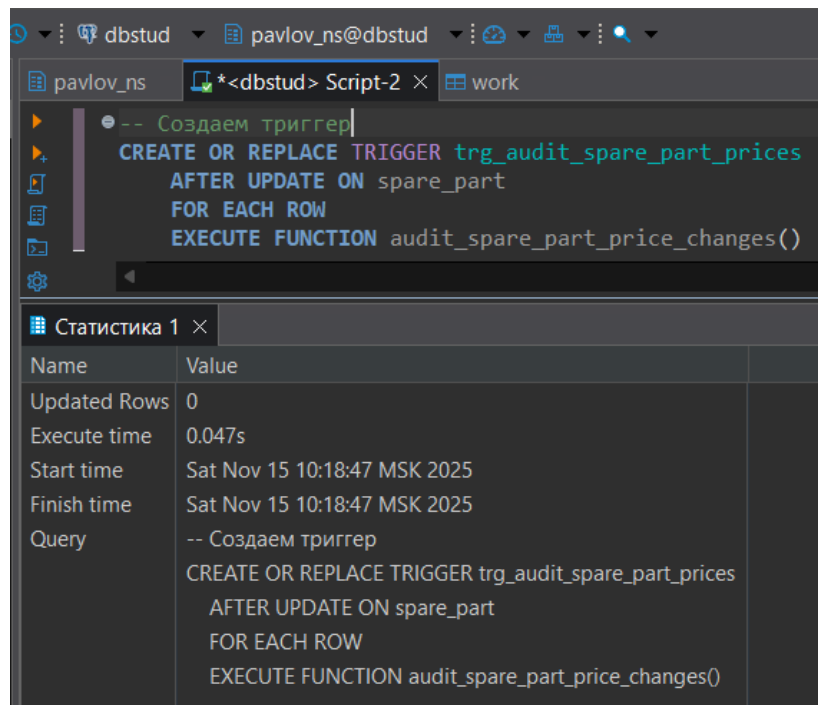


Рисунок 19 – Создание триггера

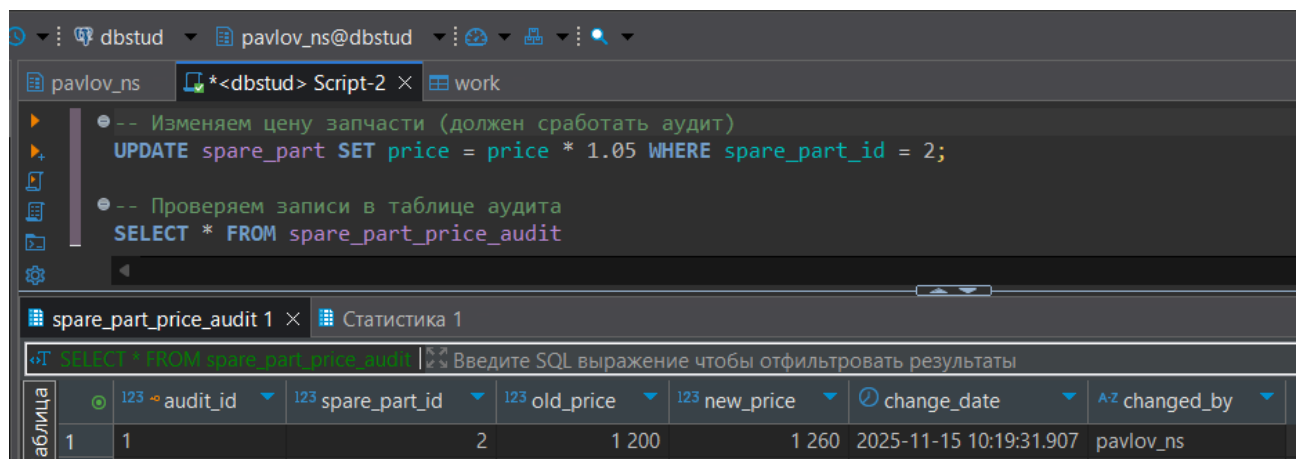


Рисунок 20 – Пример с активацией триггера

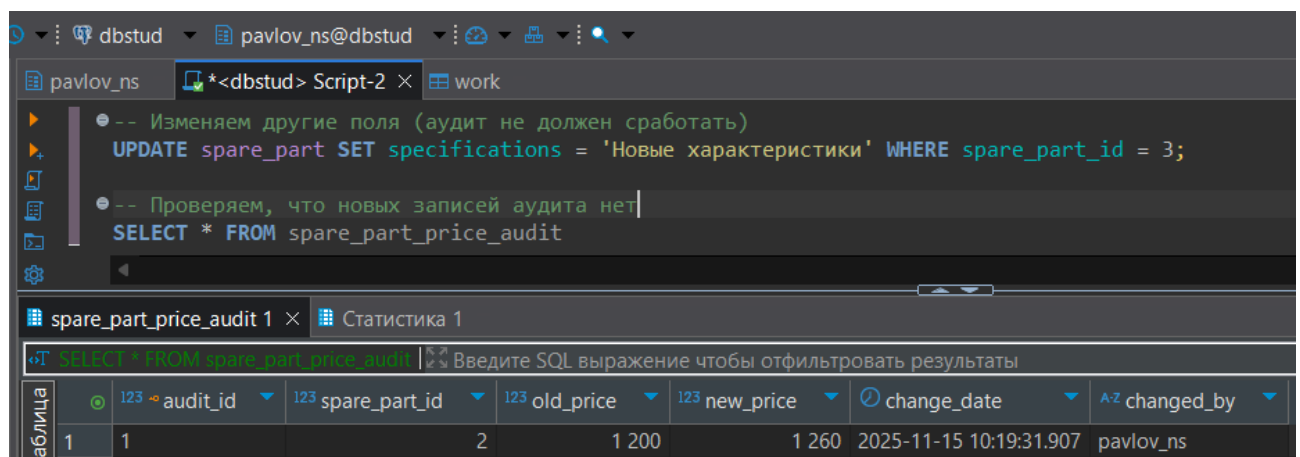


Рисунок 21 – Пример без активации триггера

## 2.4 ЗАДАНИЕ 2



```
-- Функция для анализа дорогих запчастей с использованием явного курсора
CREATE OR REPLACE FUNCTION analyze_expensive_parts(price_threshold DECIMAL(12,2))
RETURNS TABLE(
    part_id INT,
    part_name VARCHAR(100),
    part_price DECIMAL(12,2),
    supplier_name VARCHAR(100),
    price_category TEXT
) AS $$
DECLARE
    part_cursor CURSOR FOR
        SELECT sp.spare_part_id, sp.title, sp.price, s.organization_title
        FROM spare_part sp
        JOIN supplier s ON sp.supplier_id = s.supplier_id
        WHERE sp.price > price_threshold
        ORDER BY sp.price DESC;

    part_record RECORD;
BEGIN
    OPEN part_cursor;

    LOOP
        FETCH part_cursor INTO part_record;
        EXIT WHEN NOT FOUND;

        part_id := part_record.spare_part_id;
        part_name := part_record.title;
        part_price := part_record.price;
        supplier_name := part_record.organization_title;

        IF part_record.price > 50000 THEN
            price_category := 'Премиум';
        ELSIF part_record.price > 10000 THEN
            price_category := 'Высокий класс';
        ELSE
            price_category := 'Стандарт';
        END IF;

        RETURN NEXT;
    END LOOP;

    CLOSE part_cursor;

    RETURN;
END;
$$ LANGUAGE plpgsql
```

Рисунок 22 – Создание функции с использованием явного курсора

dbstud pavlov\_ns@dbstud

pavlov\_ns \*dbstud> Script-2

```
-- Демонстрация работы функции с явным курсором
SELECT * FROM analyze_expensive_parts(5000)
```

Результат 1

SELECT \* FROM analyze\_expensive\_parts(5000) | Введите SQL выражение чтобы отфильтровать результаты

	part_id	part_name	part_price	supplier_name	price_category
1	37	Группа SRAM RED eTap AXS	280 000	CPAM Дистрибьюшн	Премиум
2	38	Группа SRAM XX SL Eagle AX	220 000	CPAM Дистрибьюшн	Премиум
3	36	Вилка Fox 40 Factory	140 000	Фокс Раша	Премиум
4	32	Вилка Fox 36 Factory	125 000	Фокс Раша	Премиум
5	35	Амортизатор Fox DHX2 Facto	92 000	Фокс Раша	Премиум
6	33	Амортизатор Fox Float X2 Fa	85 000	Фокс Раша	Премиум
7	34	Вилка Fox 34 Performance Eli	75 000	Фокс Раша	Премиум
8	40	Система SRAM XX1 Eagle Car	68 000	CPAM Дистрибьюшн	Премиум
9	41	Кассета SRAM XG-1299 Eagle	55 000	CPAM Дистрибьюшн	Премиум
10	10	Трансмиссия SRAM Force eT	45 000	CPAM Дистрибьюшн	Высокий класс
11	39	Тормоза SRAM Level Ultimat	45 000	CPAM Дистрибьюшн	Высокий класс
12	7	Вилка RockShox Sid SL Ultima	25 000	РокШок Сервис	Высокий класс
13	4	Манетка Shimano Ultegra R8	8 500	Шимано Рус	Стандарт

Рисунок 23 – Пример с использованием явного курсора

The image shows a screenshot of a database IDE window. The title bar indicates the user is 'pavlov\_ns@dbstud'. The main editor area displays a SQL script for creating a function named 'get\_client\_order\_statistics'. The script is written in Russian and uses PL/SQL syntax. It includes a comment at the top: '-- Функция для анализа клиентов и их заказов с использованием неявного курсора'. The function returns a table with columns: client\_id (INT), client\_name (TEXT), total\_orders (INT), total\_spent (DECIMAL(12,2)), and status\_summary (TEXT). The function body starts with a DECLARE section for local variables: client\_record (RECORD), order\_count (INT), total\_amount (DECIMAL(12,2)), and status\_text (TEXT). It then enters a BEGIN block with a FOR loop over client\_record. Inside the loop, it performs a SELECT query joining client, application, user\_order, and estimate tables to calculate order counts and total amounts. It then uses an IF-ELSIF-ELSE block to assign a status text based on the total amount: 'VIP клиент' for amounts over 10000, 'Постоянный клиент' for amounts over 5000, 'Новый клиент' for amounts greater than 0, and 'Без заказов' otherwise. Finally, it assigns the calculated values to the function's return table and uses RETURN NEXT to process the next row. The script ends with END; and RETURN NEXT;.

```
-- Функция для анализа клиентов и их заказов с использованием неявного курсора
CREATE OR REPLACE FUNCTION get_client_order_statistics()
RETURNS TABLE(
    client_id INT,
    client_name TEXT,
    total_orders INT,
    total_spent DECIMAL(12,2),
    status_summary TEXT
) AS $$
DECLARE
    client_record RECORD;
    order_count INT;
    total_amount DECIMAL(12,2);
    status_text TEXT;
BEGIN
    FOR client_record IN
        SELECT c.client_id, c.surname || ' ' || c.name as full_name
        FROM client c
        ORDER BY c.client_id
    LOOP
        SELECT
            COUNT(uo.user_order_id),
            COALESCE(SUM(e.total_price), 0)
        INTO order_count, total_amount
        FROM client c2
        LEFT JOIN application a ON c2.client_id = a.client_id
        LEFT JOIN user_order uo ON a.application_id = uo.application_id
        LEFT JOIN estimate e ON uo.user_order_id = e.user_order_id
        WHERE c2.client_id = client_record.client_id;

        IF total_amount > 10000 THEN
            status_text := 'VIP клиент';
        ELSIF total_amount > 5000 THEN
            status_text := 'Постоянный клиент';
        ELSIF total_amount > 0 THEN
            status_text := 'Новый клиент';
        ELSE
            status_text := 'Без заказов';
        END IF;

        client_id := client_record.client_id;
        client_name := client_record.full_name;
        total_orders := order_count;
        total_spent := total_amount;
        status_summary := status_text;

        RETURN NEXT;
    END LOOP;
END;
```

Рисунок 24 – Создание функции с использованием неявного курсора



dbstud pavlov\_ns@dbstud

pavlov\_ns \*<dbstud> Script-2 ×

```
-- Демонстрация работы функции с неявным курсором
SELECT * FROM get_client_order_statistics()
```

Результат 1 ×

Введите SQL выражение чтобы отфильтровать результаты

	123 client_id	A-Z client_name	123 total_orders	123 total_spent	A-Z status_summary
1	1	Иванов Алексей	2	4 850	Новый клиент
2	2	Петрова Мария	1	2 000	Новый клиент
3	3	Сидоров Дмитрий	1	6 000	Постоянный клиент
4	4	Козлова Анна	1	9 500	Постоянный клиент
5	5	Никитин Сергей	1	4 500	Новый клиент
6	6	Федорова Елена	1	5 000	Новый клиент
7	7	Морозов Иван	1	28 000	VIP клиент
8	8	Павлова Ольга	1	5 800	Постоянный клиент
9	9	Лебедев Андрей	1	3 300	Новый клиент
10	10	Семенова Татьяна	1	49 000	VIP клиент

Рисунок 25 – Пример с использованием неявного курсора