



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 *ИНБО-12-23. Албахтин И.В.*

(подпись)

Ассистент *Брайловский А.В.*

(подпись)

Москва 2025 г.

ПРАКТИЧЕСКАЯ РАБОТА №6. ТРИГГЕРЫ И КУРСОРЫ В POSTGRESQL

Цель:

Целью данной практической работы является формирование у студентов углубленных практических навыков по управлению данными и реализации сложной бизнес-логики в СУБД PostgreSQL с использованием триггеров и курсоров.

Постановка задачи:

Для выполнения практической работы необходимо последовательно выполнить два задания, демонстрирующих применение триггеров и курсоров.

Ваша задача — адаптировать каждую из поставленных задач к логической структуре и предметной области вашей базы данных.

Задание №1: Триггеры

1. Аудит и логирование.

Запись факта изменения данных.

Пример: при любом изменении столбца salary в таблице employees создавать запись в salary_log, сохраняя OLD.salary, NEW.salary и CURRENT_USER.

2. Сложная валидация (проверка).

Запрет операции на основе данных из другой таблицы.

Пример: триггер BEFORE INSERT на таблицу order_items должен проверить, что NEW.quantity не превышает quantity_on_hand в таблице products. Если превышает – вызвать RAISE EXCEPTION.

3. Поддержание согласованности (денормализация).

Автоматическое обновление связанных данных.

Пример: триггер AFTER INSERT на таблицу sales автоматически обновляет столбец total_spent в таблице customers.

4. Защита данных.

Запрет определенных операций.

Пример: триггер BEFORE DELETE на таблицу departments может запретить удаление, если в таблице employees еще есть сотрудники, ссылающиеся на этот отдел.

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.

ВЫПОЛНЕНИЕ ПРАКТИЧЕСКОЙ РАБОТЫ

Таблица 1. Таблица worker (Сотрудник)

SELECT * FROM worker;

worker 1

SELECT * FROM worker | Введите SQL выражение чтобы отфильтровать результаты

	123 worker_id	A-Z name	A-Z position	A-Z phone
1	2	Ольга	Диагност	+79210009988
2	1	Иванов П.П.	Механик	+79112223344

Таблица 2. Таблица maintenance (ТО)

SELECT * FROM maintenance;

maintenance 1

SELECT * FROM maintenance | Введите SQL выражение чтобы отфильтровать результаты

	123 maintenance_id	123 car_id	123 worker_id	123 part_id	start_date	end_date	A-Z status
1	34	4	4	3	2025-02-10	[NULL]	waiting
2	35	5	5	2	2025-03-20	2025-09-10	completed
3	36	6	2	5	2025-03-20	[NULL]	in progress
4	37	7	3	4	2025-04-25	2025-09-11	completed
5	38	8	1	1	2025-04-25	[NULL]	planned
6	39	9	4	2	2025-05-15	2025-09-15	completed
7	40	10	5	3	2025-05-15	[NULL]	in progress
8	33	3	2	1	2025-11-05	2025-11-15	completed
9	31	1	1	2	2025-07-15	2025-07-20	completed
10	32	2	3	4	2025-09-10	2025-09-20	in progress

Таблица 3. Таблица maintenance_work

SELECT * FROM maintenance_work;

aintenance_work 1 X

* FROM maintenance_work | Введите SQL выражение чтобы отфильтровать результаты

123 maintenance_work	123 maintenance_id	123 work_type_id	123 part_id	123 quantity
1	31	1	[NULL]	0
2	32	2	[NULL]	0

Таблица 4. Таблица part

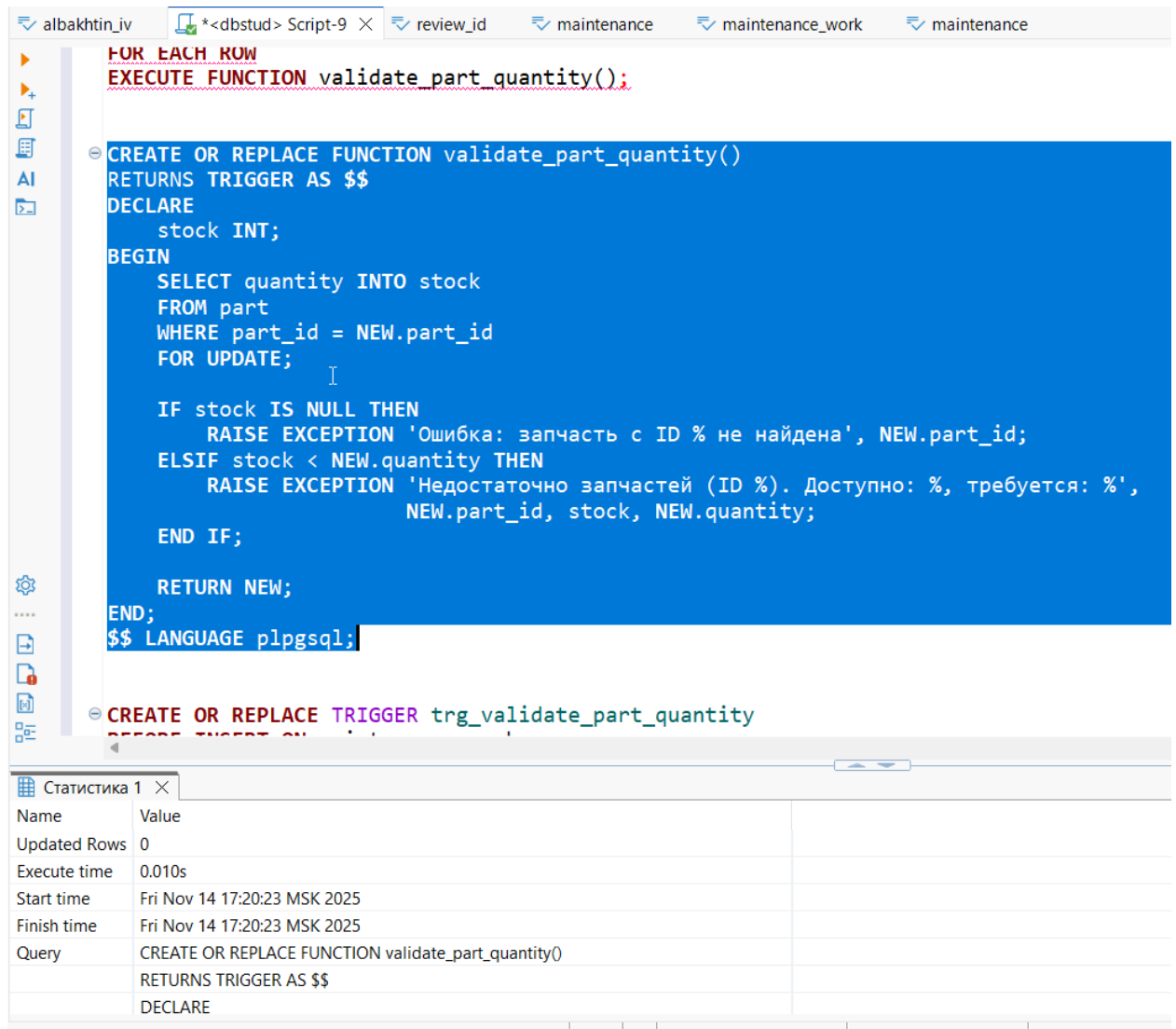
SELECT * FROM part;

X

* FROM part | Введите SQL выражение чтобы отфильтровать результаты

123 part_id	AZ name	123 price	123 supplier_id	123 quantity
2	Тормозные колодки	1 200	2	5
1	Фильтр масляный	750	1	7

Задание 1. Триггеры



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

CREATE OR REPLACE FUNCTION validate_part_quantity()
RETURNS TRIGGER AS $$
DECLARE
    stock INT;
BEGIN
    SELECT quantity INTO stock
    FROM part
    WHERE part_id = NEW.part_id
    FOR UPDATE;

    IF stock IS NULL THEN
        RAISE EXCEPTION 'Ошибка: запчасть с ID % не найдена', NEW.part_id;
    ELSIF stock < NEW.quantity THEN
        RAISE EXCEPTION 'Недостаточно запчастей (ID %). Доступно: %, требуется: %',
            NEW.part_id, stock, NEW.quantity;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg_validate_part_quantity
BEFORE INSERT OR UPDATE ON part
FOR EACH ROW
EXECUTE FUNCTION validate_part_quantity();
```

Name	Value
Updated Rows	0
Execute time	0.010s
Start time	Fri Nov 14 17:20:23 MSK 2025
Finish time	Fri Nov 14 17:20:23 MSK 2025
Query	CREATE OR REPLACE FUNCTION validate_part_quantity() RETURNS TRIGGER AS \$\$ DECLARE

Рисунок 1 – Запрет на списание запчастей, если их не хватает на складе

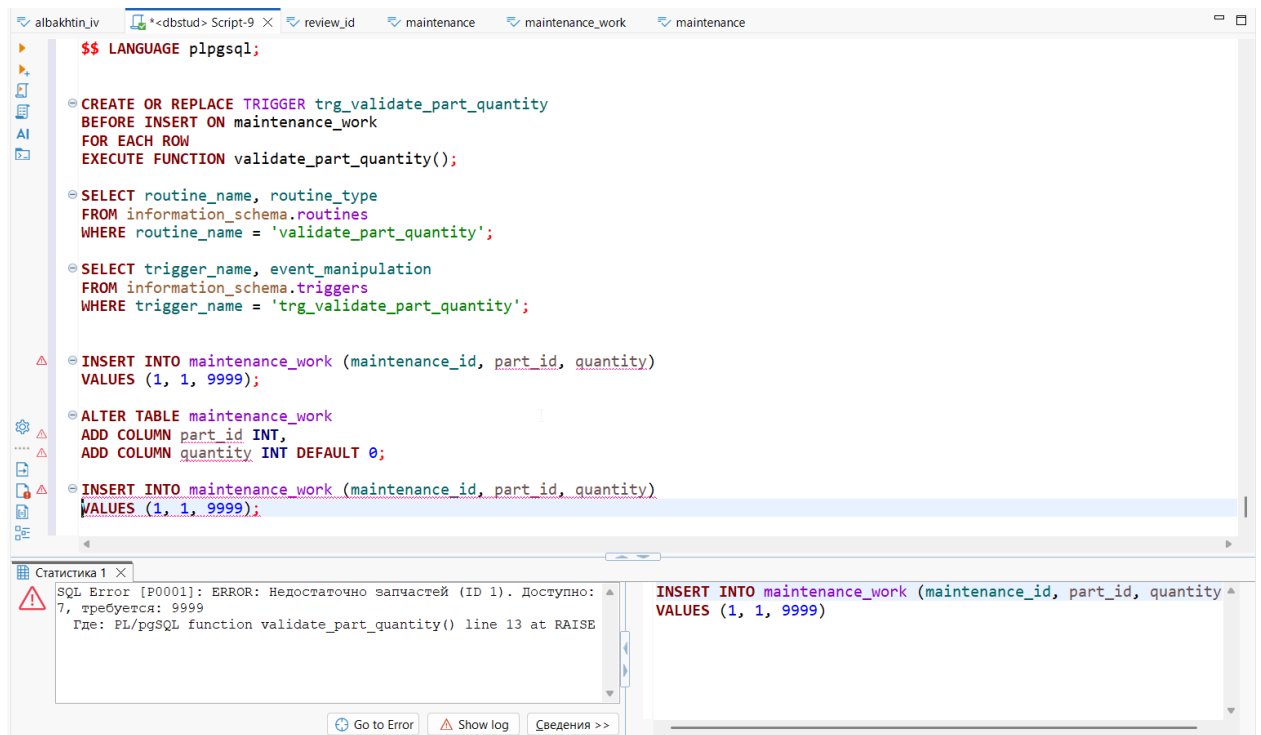


Рисунок 2 – Проверка

albakhtin_iv

*<dbstud> Script-9

review_id

maintenance

maintenance_work

▶

⚠

📄

AI

📁

```

ADD COLUMN quantity INT DEFAULT 0;

INSERT INTO maintenance_work (maintenance_id, part_id, quantity)
VALUES (1, 1, 9999);

CREATE OR REPLACE FUNCTION update_part_stock()
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'INSERT') THEN
        UPDATE part
        SET quantity = quantity - NEW.quantity
        WHERE part_id = NEW.part_id;
    ELSIF (TG_OP = 'DELETE') THEN
        UPDATE part
        SET quantity = quantity + OLD.quantity
        WHERE part_id = OLD.part_id;
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg_update_part_stock
AFTER INSERT OR DELETE ON maintenance_work
FOR EACH ROW
EXECUTE FUNCTION update_part_stock();

```

⚙️

📄

🔒

📄

🔍

Статистика 1

Name	Value	
Updated Rows	0	
Execute time	0.012s	
Start time	Fri Nov 14 17:24:10 MSK 2025	
Finish time	Fri Nov 14 17:24:10 MSK 2025	
Query	CREATE OR REPLACE FUNCTION update_part_stock() RETURNS TRIGGER AS \$\$ BEGIN	

Рисунок 3 – Автоматически уменьшать или возвращать остаток на складе после вставки или удаления записи

albakhtin_iv

*<dbstud> Script-9 X

review_id

maintenance

maintenance_work

n

▶

+

📄

📊

AI

🔍

⚙️

📄

📄

📄

📄

```

WHERE part_id = OLD.part_id;
END IF;

RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg_update_part_stock
AFTER INSERT OR DELETE ON maintenance_work
FOR EACH ROW
EXECUTE FUNCTION update_part_stock();

INSERT INTO maintenance_work (maintenance_id, part_id, quantity)
VALUES (1, 1, 2);

DELETE FROM maintenance_work
WHERE maintenance_id = 1 AND part_id = 1 AND quantity = 2;

```

Статистика 1 X

Name	Value	
Updated Rows	2	
Execute time	0.014s	
Start time	Fri Nov 14 17:24:43 MSK 2025	
Finish time	Fri Nov 14 17:24:43 MSK 2025	
Query	INSERT INTO maintenance_work (maintenance_id, part_id, quantity)	
	VALUES (1, 1, 2);	
	DELETE FROM maintenance_work	
	WHERE maintenance_id = 1 AND part_id = 1 AND quantity = 2	

Рисунок 4 – Проверка

albakhtin_iv *<dbstud> Script-9 × review_id maintenance maintenance_work mainte

```

change_time TIMESTAMP DEFAULT NOW()
);

CREATE OR REPLACE FUNCTION log_worker_change()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.name IS DISTINCT FROM OLD.name
    OR NEW.position IS DISTINCT FROM OLD.position THEN
        INSERT INTO worker_audit (worker_id, old_name, old_position)
        VALUES (OLD.worker_id, OLD.name, OLD.position);
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg_log_worker_change
AFTER UPDATE ON worker
FOR EACH ROW
EXECUTE FUNCTION log_worker_change();

```

Статистика 1 ×

Name	Value
Updated Rows	0
Execute time	0.019s
Start time	Fri Nov 14 17:27:44 MSK 2025
Finish time	Fri Nov 14 17:27:44 MSK 2025
Query	CREATE OR REPLACE FUNCTION log_worker_change() RETURNS TRIGGER AS \$\$ BEGIN IF NEW.name IS DISTINCT FROM OLD.name OR NEW.position IS DISTINCT FROM OLD.position THEN INSERT INTO worker_audit (worker_id, old_name, old_position) VALUES (OLD.worker_id, OLD.name, OLD.position); END IF; RETURN NEW;

Рисунок 6 – При изменении имени или должности работника сохранять старое значение

albakhtin_iv * <dbstud> Script-9 X review_i

```

RETURNS TRIGGER AS $$
BEGIN
    IF NEW.name IS DISTINCT FR
    OR NEW.position IS DIST
    INSERT INTO worker_aud
    VALUES (OLD.worker_id,
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg_
AFTER UPDATE ON worker
FOR EACH ROW
EXECUTE FUNCTION log_worker_ch

UPDATE worker
SET name = 'Иванов П.П.'
WHERE worker_id = 1;

```

Статистика 1 X

Name	Value
Updated Rows	1
Execute time	0.017s
Start time	Fri Nov 14 17:28:11 MSK 2025
Finish time	Fri Nov 14 17:28:11 MSK 2025
Query	UPDATE worker
	SET name = 'Иванов П.П.'
	WHERE worker_id = 1

Рисунок 7 - Проверка

Задание 2. Курсоры

The screenshot shows a database IDE with a script editor and a console window. The script editor contains the following PL/SQL code:

```
RETURN;  
END;  
$$ LANGUAGE plpgsql;  
  
SELECT * FROM get_worker_stats();  
  
DO $$  
DECLARE  
    cur_parts CURSOR FOR SELECT part_id, name, quantity FROM  
    row_data RECORD;  
BEGIN  
    OPEN cur_parts;  
    LOOP  
        FETCH cur_parts INTO row_data;  
        EXIT WHEN NOT FOUND;  
        RAISE NOTICE 'ID: %, Наименование: %, Остаток: %',  
            row_data.part_id, row_data.name, row_data.quantity;  
    END LOOP;  
    CLOSE cur_parts;  
END $$;
```

The console window shows the output of the script:

```
relation "worker_audit" already exists, skipping  
ID: 2, Наименование: Тормозные колодки, Остаток: 5  
ID: 1, Наименование: Фильтр масляный, Остаток: 7
```

Below the console window is a statistics table:

Name	Value
Updated Rows	0
Execute time	0.012s
Start time	Fri Nov 14 17:41:57 MSK 2025
Finish time	Fri Nov 14 17:41:57 MSK 2025
Query	DO \$\$ DECLARE cur_parts CURSOR FOR SELECT part_id, name, quantity FROM part; row_data RECORD; BEGIN OPEN cur_parts; LOOP FETCH cur_parts INTO row_data; EXIT WHEN NOT FOUND;

Рисунок 8 – Вывести список запчастей и их остатков построчно в консоль (Явный (DECLARE / OPEN / FETCH / CLOSE))

Rollback Auto dbstud albakhtin_iv@dbstud

albakhtin_iv *<dbstud> Script-9 review_id maintenance maintenance_work maintenance

```

WHERE worker_id = 1;

CREATE OR REPLACE FUNCTION get_worker_stats()
RETURNS TABLE(worker_id INT, worker_name TEXT, total_services INT) AS $$
BEGIN
  FOR worker_id, worker_name, total_services IN
  SELECT
    w.worker_id,
    w.name,
    COUNT(m.maintenance_id) AS total_services
  FROM worker w
  LEFT JOIN maintenance m ON w.worker_id = m.worker_id
  GROUP BY w.worker_id, w.name
  LOOP
    RETURN NEXT;
  END LOOP;
  RETURN;
END;
$$ LANGUAGE plpgsql;

```

Статистика 1

Name	Value
Updated Rows	0
Execute time	0.013s
Start time	Fri Nov 14 17:34:07 MSK 2025
Finish time	Fri Nov 14 17:34:07 MSK 2025
Query	CREATE OR REPLACE FUNCTION get_worker_stats() RETURNS TABLE(worker_id INT, worker_name TEXT, total_services INT) AS \$\$ BEGIN FOR worker_id, worker_name, total_services IN SELECT w.worker_id, w.name, COUNT(m.maintenance_id) AS total_services FROM worker w

MSK RU Запись Имя таблицы

Рисунок 9 – Вернуть таблицу работников с количеством их ТО (Неявный (FOR ... IN))

albakhtin_iv

*<dbstud> Script-9

review_id

maintenance

▶

+

📄

📊

AI

📄

⚙️

📄

📄

📄

🔧

```

CREATE OR REPLACE FUNCTION get_worker_stats()
RETURNS TABLE(worker_id INT, worker_name TEXT, to
BEGIN
    FOR worker_id, worker_name, total_services IN
    SELECT
        w.worker_id,
        w.name,
        COUNT(m.maintenance_id) AS total_serv
    FROM worker w
    LEFT JOIN maintenance m ON w.worker_id =
    GROUP BY w.worker_id, w.name
    LOOP
        RETURN NEXT;
    END LOOP;
    RETURN;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM get_worker_stats();

```

Результат 1

🔍

SELECT * FROM get_worker_stats()

Введите SQL выражение чтобы отфильтровать

	123 worker_id	A-Z worker_name	123 total_services
1	1	Иванов П.П.	2
2	2	Ольга	2

Рисунок 10 - Проверка

ВЫВОД

В ходе выполнения практической работы были реализованы триггеры для валидации, аудита и автоматического обновления данных, а также два примера курсоров.

Практика позволила закрепить навыки программирования на PL/pgSQL и научиться применять триггеры и курсоры для автоматизации бизнес-логики в базе данных.