

**Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**ОТЧЕТ**

**по Лабораторной работе № 5**

**«Запросы на выборку и модификацию данных. Представления. Работа с  
индексами»**

**по дисциплине «Проектирование и реализация баз данных»**

**Обучающиеся** Савченко Анастасия Сергеевна

**Факультет** прикладной информатики

**Группа** К3240

**Направление подготовки** 09.03.03 Прикладная информатика

**Образовательная программа** Мобильные и сетевые технологии 2023

**Преподаватель** Говорова Марина Михайловна

Санкт-Петербург  
2025

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 Выполнение.....	4
1.1 Разработанные объекты по индивидуальному заданию.....	4
– для получения расписания занятий для групп на определенный день недели.....	4
– записи на курс слушателя.....	6
– получения перечня свободных лекционных аудиторий на любой день недели.....	9
1.2 Триггеры.....	11
1. Автозаполнение дня недели по дате при создании или обновлении записи в расписании.....	11
2. Запрет на удаление группы, если в ней есть студенты.....	12
3. Проверка, что дата зачисления студента попадает в период существования группы.....	13
4. Предотвращение дублирования записи об участии студента в практике.....	15
5. Проверка, что преподаватель действительно привязан к дисциплине, при создании или обновлении записи в расписании....	17
6. Проверка, что преподаватель не назначен одновременно на две пары в одно и то же время и дату.....	18
7. Проверка вхождения дисциплины в программу группы при добавлении или изменении записи в расписании.....	20
ЗАКЛЮЧЕНИЕ.....	23



## ВВЕДЕНИЕ

**Цель работы** – овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

### **Практическое задание:**

Задание 4. Создать хранимые процедуры:

- Для получения расписания занятий для групп на определенный день недели.

- Записи на курс слушателя.

- Получения перечня свободных лекционных аудиторий на любой день недели. Если свободных аудиторий не имеется, то выдать соответствующее сообщение.

- создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры,

- 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).

Дополнительные баллы - 3:

Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу).

## 1 Выполнение

### 1.1 Разработанные объекты по индивидуальному заданию

Процедуры:

– *для получения расписания занятий для групп на определенный день недели*

```
CREATE OR REPLACE FUNCTION
courses_scheme.get_schedule_for_group_day_date(
    input_group_number VARCHAR,
    input_week_day courses_scheme.week_day_type,
    input_class_date DATE
)
RETURNS TABLE (
    group_number VARCHAR,
    class_id INTEGER,
    teacher_name VARCHAR,
    discipline_name VARCHAR,
    room_number VARCHAR,
    class_type VARCHAR,
    class_date DATE,
    week_day courses_scheme.week_day_type,
    class_number VARCHAR
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT
        g.group_number,
        c.class_id,
```

```

(t.last_name || ' ' || t.name_middlename)::VARCHAR AS teacher_name,
d.discipline_name,
r.room_number,
c.class_type::VARCHAR,
c.class_date,
c.week_day,
c.class_number::VARCHAR
FROM courses_scheme.classes c
JOIN courses_scheme.groups g ON c.group_id = g.group_id
JOIN courses_scheme.teachers t ON c.teacher_id = t.teacher_id
JOIN courses_scheme.disciplines d ON c.discipline_id = d.discipline_id
JOIN courses_scheme.rooms r ON c.room_id = r.room_id
WHERE g.group_number = input_group_number
      AND c.week_day = input_week_day
      AND c.class_date = input_class_date
ORDER BY c.class_number;

END;

$$;

```

Вызов процедуры:

```

SELECT * FROM
courses_scheme.get_schedule_for_group_day_date('WD_1',
'понедельник'::courses_scheme.week_day_type, '2025-05-05');

```

На рисунке 1 результат вызова.

```
lab5_restore=# SELECT * FROM courses_scheme.get_schedule_for_group_day_date('WD_1', 'понедельник'::courses_scheme.week_day_type, '2025-05-05');
```

group_number	class_id	teacher_name	discipline_name	room_number	class_type	class_date	week_day	class_number
WD_1	81	Зайцева Екатерина Петровна	Веб-разработка	111	лекционное	2025-05-05	понедельник	1
WD_1	82	Туманов Роман Викторович	UX/UI-дизайн	112	практическое	2025-05-05	понедельник	2
WD_1	83	Михайлов Кирилл Игоревич	Мобильная разработка	113	практическое	2025-05-05	понедельник	3

(3 строки)

Рисунок 1 – Результат выполнения.

*– записи на курс слушателя*

```
CREATE OR REPLACE PROCEDURE
courses_scheme.enroll_student_to_program(
    IN p_student_id INTEGER,
    IN p_program_id INTEGER
)
LANGUAGE plpgsql
AS $$
DECLARE
    v_group_id INTEGER;
    v_start_date DATE;
    v_is_enrolled BOOLEAN; -- Пременная "записан ли
BEGIN
    -- Проверяем, не записан ли уже студент на эту программу
    SELECT EXISTS (
        SELECT 1
        FROM courses_scheme.current_students cs
        JOIN courses_scheme.groups g ON cs.group_id = g.group_id
        WHERE cs.student_id = p_student_id
        AND g.program_id = p_program_id
    ) INTO v_is_enrolled;

    IF v_is_enrolled THEN
        RAISE NOTICE 'Студент % уже записан на программу %',
p_student_id, p_program_id;
        RETURN;
    END IF;

    -- Ищем подходящую группу со свободными местами
```

```

SELECT
    g.group_id,
    g.start_date
INTO
    v_group_id,
    v_start_date
FROM courses_scheme.groups g
    LEFT JOIN courses_scheme.current_students cs ON g.group_id =
cs.group_id
WHERE g.program_id = p_program_id
    AND g.max_students > (
        SELECT COUNT(*)
        FROM courses_scheme.current_students
        WHERE group_id = g.group_id
    )
    AND CURRENT_DATE BETWEEN g.start_date AND g.end_date --
Проверка что текущая дата попадает в период группы
ORDER BY g.start_date
LIMIT 1;

-- Если группа не найдена
IF v_group_id IS NULL THEN
    RAISE EXCEPTION 'Нет доступных групп для программы с id %',
p_program_id;
END IF;

-- Запись студента в группу
INSERT INTO courses_scheme.current_students (
    group_id,

```



```

        student_id,
        status_edu,
        date_start_edu
    ) VALUES (
        v_group_id,
        p_student_id,
        'студент',
        v_start_date
    );

    RAISE NOTICE 'Студент % успешно записан в группу %',
p_student_id, v_group_id;

END;
$$;

```

Вызов процедуры:

```
CALL courses_scheme.enroll_student_to_program(331, 2);
```

На рисунке 2 результат вызова процедуры 2.

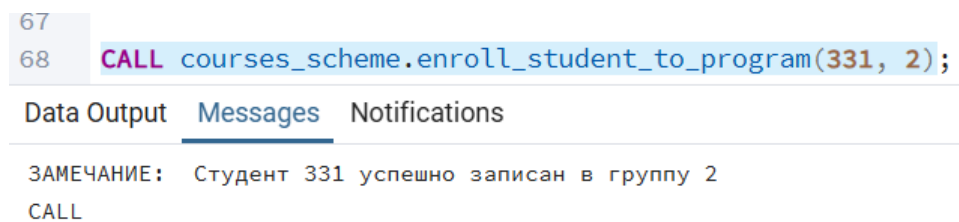


Рисунок 2 – Результат выполнения процедуры 2.

*– получения перечня свободных лекционных аудиторий на любой день недели*

*Если свободных аудиторий не имеется, то выдать соответствующее сообщение*

```
CREATE OR REPLACE FUNCTION
courses_scheme.get_free_lecture_rooms(input_week_day VARCHAR, input_date
DATE)
RETURNS TABLE (
    room_number VARCHAR,
    room_id INTEGER
) AS $$
BEGIN
    -- Выбор аудиторий
    RETURN QUERY
    SELECT r.room_number, r.room_id
    FROM courses_scheme.rooms r
    -- Проверка занятости
    LEFT JOIN courses_scheme.classes c
    ON r.room_id = c.room_id
    AND c.week_day::VARCHAR = input_week_day
    AND c.class_date = input_date
    -- Фильтрация - только свободные лекционные
    WHERE r.room_type = 'лекционная'
    AND c.class_id IS NULL;
    IF NOT FOUND THEN
        RAISE NOTICE 'Нет свободных лекционных аудиторий на
указанный день и дату.';
    END IF;
END;
```

```
$$ LANGUAGE plpgsql;
```

Вызов процедуры:

```
SELECT * FROM courses_scheme.get_free_lecture_rooms('понедельник',  
'2025-04-06');
```

На рисунке 3 результат вызова.

```
lab5_restore=# SELECT * FROM courses_scheme.get_free_lecture_rooms('понедельник', '2025-04-06');  
 room_number | room_id  
-----+-----  
 111         |      1  
 118         |      8  
 215         |     15  
 311         |     16  
 315         |     20  
 122         |     22  
 128         |     28  
 220         |     30  
 224         |     34  
 321         |     38  
(10 строк)
```

Рисунок 3 – Результат выполнения.

## 1.2 Триггеры

Создать триггеры для индивидуальной БД согласно варианту:

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

### ***1. Автозаполнение дня недели по дате при создании или обновлении записи в расписании***

```
CREATE OR REPLACE FUNCTION courses_scheme.fn_set_class_week_day()
  RETURNS trigger AS
$$
BEGIN
  NEW.week_day := CASE EXTRACT(ISODOW FROM NEW.class_date)
    WHEN 1 THEN 'понедельник'::courses_scheme.week_day_type
    WHEN 2 THEN 'вторник'::courses_scheme.week_day_type
    WHEN 3 THEN 'среда'::courses_scheme.week_day_type
    WHEN 4 THEN 'четверг'::courses_scheme.week_day_type
    WHEN 5 THEN 'пятница'::courses_scheme.week_day_type
    WHEN 6 THEN 'суббота'::courses_scheme.week_day_type
    WHEN 7 THEN 'воскресенье'::courses_scheme.week_day_type
  END;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_set_class_week_day
  BEFORE INSERT OR UPDATE ON courses_scheme.classes
  FOR EACH ROW
  EXECUTE FUNCTION courses_scheme.fn_set_class_week_day();
```

Запрос для проверки работы триггера:

```
INSERT INTO courses_scheme.classes
```

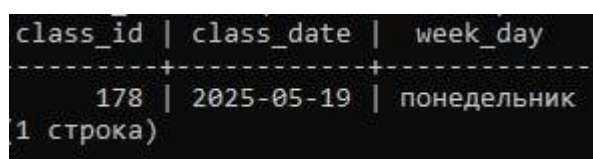
(teacher\_id, discipline\_id, group\_id, room\_id, class\_type, class\_date, class\_number, class\_status)

VALUES

(5, 9, 2, 2, 'практическое', '2025-05-19', '2', 'Yes')

RETURNING class\_id, class\_date, week\_day;

На рисунке 4 результат проверки триггера 1.



class_id	class_date	week_day
178	2025-05-19	понедельник

1 строка)

Рисунок 4 – Результат проверки триггера 1.

## ***2. Запрет на удаление группы, если в ней есть студенты***

CREATE OR REPLACE FUNCTION

courses\_scheme.prevent\_group\_deletion\_if\_students\_exist()

RETURNS TRIGGER

LANGUAGE plpgsql

AS \$\$

BEGIN

IF EXISTS (

SELECT 1

FROM courses\_scheme.current\_students

WHERE group\_id = OLD.group\_id

AND status\_edu = 'студент'

) THEN

RAISE EXCEPTION

'Удаление невозможно: в группе % есть учащиеся со статусом "студент"',

OLD.group\_id;

END IF;

RETURN OLD;

END;

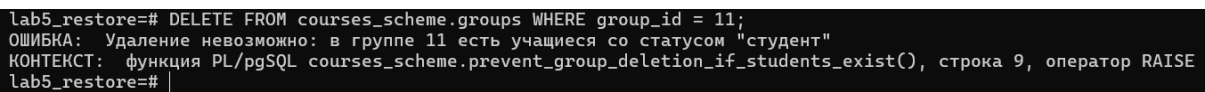
\$\$;

```
CREATE TRIGGER trg_prevent_group_deletion_when_students_exist
BEFORE DELETE ON courses_scheme.groups
FOR EACH ROW
EXECUTE FUNCTION courses_scheme.prevent_group_deletion_if_students_exist();
```

Запрос для проверки работы триггера:

```
DELETE FROM courses_scheme.groups
WHERE group_id = 11;
```

На рисунке 5 результат проверки триггера 2.



```
lab5_restore=# DELETE FROM courses_scheme.groups WHERE group_id = 11;
ОШИБКА: Удаление невозможно: в группе 11 есть учащиеся со статусом "студент"
КОНТЕКСТ: функция PL/pgSQL courses_scheme.prevent_group_deletion_if_students_exist(), строка 9, оператор RAISE
lab5_restore=#
```

Рисунок 5 – Результат проверки триггера 2.

### ***3. Проверка, что дата зачисления студента попадает в период существования группы***

```
CREATE OR REPLACE FUNCTION
courses_scheme.validate_student_enrollment_date()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    grp_start date;
    grp_end date;
BEGIN
    -- Получаем даты начала и окончания группы
```

```

SELECT start_date, end_date
INTO grp_start, grp_end
FROM courses_scheme.groups
WHERE group_id = NEW.group_id;

-- Проверяем что дата зачисления не раньше начала группы
IF NEW.date_start_edu < grp_start THEN
    RAISE EXCEPTION
        'Дата зачисления % раньше начала группы % (%)',
        NEW.date_start_edu, NEW.group_id, grp_start;
END IF;

-- не позже окончания группы
IF grp_end IS NOT NULL AND NEW.date_start_edu > grp_end THEN
    RAISE EXCEPTION
        'Дата зачисления % позже окончания группы % (%)',
        NEW.date_start_edu, NEW.group_id, grp_end;
END IF;

RETURN NEW;
END;
$$;

CREATE TRIGGER validate_student_enrollment_dates
BEFORE INSERT OR UPDATE OF group_id, date_start_edu ON
courses_scheme.current_students
FOR EACH ROW
EXECUTE FUNCTION courses_scheme.validate_student_enrollment_date();

```

Запрос на создание записи для проверки работы триггера :

```
INSERT INTO courses_scheme.current_students
  (group_id, student_id, status_edu, date_start_edu)
VALUES
  (2, 331, 'студент', '2024-01-15');
```

На рисунке 6 результат проверки триггера 3 при создании записи.

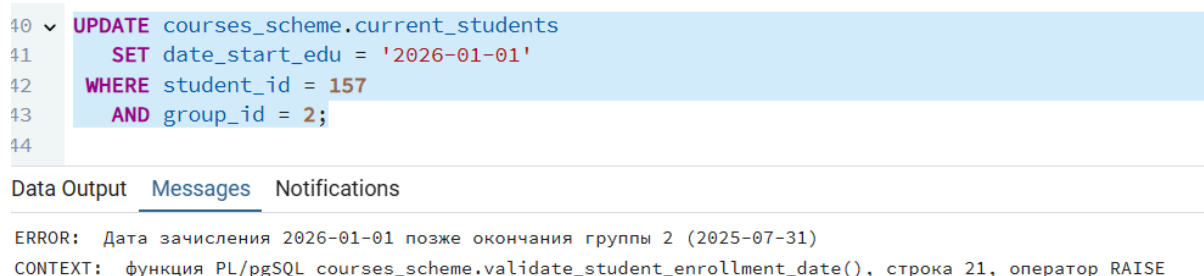
ОШИБКА: Дата зачисления 2024-01-15 раньше начала группы 2 (2025-02-01)  
КОНТЕКСТ: функция PL/pgSQL courses\_scheme.validate\_student\_enrollment\_date(), строка 14, оператор RAISE

Рисунок 6 – Результат проверки триггера 3 при создании записи.

Запрос для проверки работы триггера (обновление записи):

```
UPDATE courses_scheme.current_students
  SET date_start_edu = '2026-01-01'
WHERE student_id = 157
AND group_id = 2;
```

На рисунке 7 результат проверки триггера 3 при обновлении записи.



The screenshot shows a SQL query editor with the following SQL code:

```
40 UPDATE courses_scheme.current_students
41   SET date_start_edu = '2026-01-01'
42   WHERE student_id = 157
43   AND group_id = 2;
44
```

Below the code, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is selected, showing an error message:

ERROR: Дата зачисления 2026-01-01 позже окончания группы 2 (2025-07-31)  
CONTEXT: функция PL/pgSQL courses\_scheme.validate\_student\_enrollment\_date(), строка 21, оператор RAISE

Рисунок 7 – Результат проверки триггера 3 при обновлении записи.

#### ***4. Предотвращение дублирования записи об участии студента в практике***

```
CREATE OR REPLACE FUNCTION
courses_scheme.fn_prevent_duplicate_practice()
  RETURNS trigger AS
$$
BEGIN
  IF EXISTS (
```



```

SELECT 1
  FROM courses_scheme.student_practice
 WHERE curr_stud_id = NEW.curr_stud_id
       AND practice_id = NEW.practice_id
) THEN
  RAISE EXCEPTION
    'Студент % уже назначен на практику %',
    NEW.curr_stud_id, NEW.practice_id;
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_prevent_duplicate_practice
  BEFORE INSERT ON courses_scheme.student_practice
  FOR EACH ROW EXECUTE FUNCTION
courses_scheme.fn_prevent_duplicate_practice();

```

Запрос для проверки работы триггера:

```

INSERT INTO courses_scheme.student_practice (practice_id, curr_stud_id)
VALUES (1, 3);

```

На рисунке 7 результат проверки триггера.

```

ОШИБКА: Студент 3 уже назначен на практику 1
КОНТЕКСТ: функция PL/pgSQL courses_scheme.fn_prevent_duplicate_practice(), строка 9, оператор RAISE
lab5_restore=#

```

Рисунок 7 – Результат проверки.

**5. Проверка, что преподаватель действительно привязан к дисциплине, при создании или обновлении записи в расписании**

CREATE OR REPLACE FUNCTION

courses\_scheme.validate\_teacher\_discipline()

RETURNS TRIGGER

LANGUAGE plpgsql

AS \$\$

BEGIN

-- Проверяем, ведет ли преподаватель указанную дисциплину

IF NOT EXISTS (

SELECT 1

FROM courses\_scheme.teacher\_discipline

WHERE teacher\_id = NEW.teacher\_id

AND discipline\_id = NEW.discipline\_id

) THEN

-- для сообщения

DECLARE

t\_name TEXT := (SELECT last\_name FROM courses\_scheme.teachers  
WHERE teacher\_id = NEW.teacher\_id);

d\_name TEXT := (SELECT discipline\_name FROM  
courses\_scheme.disciplines WHERE discipline\_id = NEW.discipline\_id);

BEGIN

RAISE EXCEPTION

'Преподаватель "%" (ID:%) не ведет дисциплину "%" (ID:%)',

t\_name, NEW.teacher\_id, d\_name, NEW.discipline\_id;

END;

END IF;

RETURN NEW;

END;

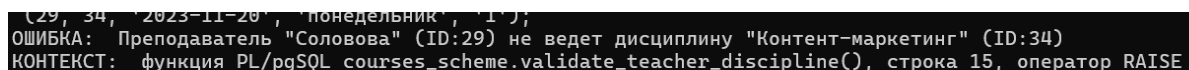
\$\$;

```
-- Триггер для проверки перед вставкой/обновлением
CREATE TRIGGER check_teacher_qualification
BEFORE INSERT OR UPDATE OF teacher_id, discipline_id ON
courses_scheme.classes
FOR EACH ROW
EXECUTE FUNCTION courses_scheme.validate_teacher_discipline();
```

Запрос для проверки работы триггера:

```
INSERT INTO courses_scheme.classes (teacher_id, discipline_id, class_date,
week_day, class_number)
VALUES (29, 34, '2023-11-20', 'понедельник', '1');
```

На рисунке 8 результат проверки триггера.



```
(29, 34, '2023-11-20', 'понедельник', '1');
ОШИБКА: Преподаватель "Соловова" (ID:29) не ведет дисциплину "Контент-маркетинг" (ID:34)
КОНТЕКСТ: функция PL/pgSQL courses_scheme.validate_teacher_discipline(), строка 15, оператор RAISE
```

Рисунок 8 – Результат проверки триггера.

***6. Проверка, что преподаватель не назначен одновременно на две пары в одно и то же время и дату***

```
CREATE OR REPLACE FUNCTION
courses_scheme.fn_check_teacher_schedule_conflict()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
```

```

    conflict_count int;

BEGIN

    -- Проверяем количество пересечений в расписании
    SELECT COUNT(*)
    INTO conflict_count
    FROM courses_scheme.classes
    WHERE teacher_id = NEW.teacher_id
        AND class_date = NEW.class_date
        AND class_number = NEW.class_number
        AND class_id <> COALESCE(NEW.class_id, -1); -- Исключаем текущую
запись при обновлении

    -- Если найдены пересечения, запрещаем операцию
    IF conflict_count > 0 THEN

        RAISE EXCEPTION 'Преподаватель % уже назначен на пару № % в этот
день %',

            NEW.teacher_id, NEW.class_number, NEW.class_date;

    END IF;

    RETURN NEW; -- Разрешаем операцию, если конфликтов нет
END;

$$;

CREATE TRIGGER trg_check_teacher_schedule_conflict
BEFORE INSERT OR UPDATE ON courses_scheme.classes

```

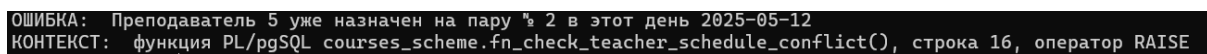
FOR EACH ROW

EXECUTE FUNCTION courses\_scheme.fn\_check\_teacher\_schedule\_conflict();

Проверка:

INSERT INTO courses\_scheme.classes (teacher\_id, discipline\_id, group\_id, room\_id, class\_type, class\_date, class\_number, class\_status) VALUES (5, 9, 4, 5, 'практическое', '2025-05-12', '2', 'Yes');

На рисунке 9 результат проверки триггера.



ОШИБКА: Преподаватель 5 уже назначен на пару № 2 в этот день 2025-05-12  
КОНТЕКСТ: функция PL/pgSQL courses\_scheme.fn\_check\_teacher\_schedule\_conflict(), строка 16, оператор RAISE

Рисунок 9 – Результат проверки триггера.

***7. Проверка вхождения дисциплины в программу группы при добавлении или изменении записи в расписании***

CREATE OR REPLACE FUNCTION

courses\_scheme.fn\_check\_discipline\_in\_group\_program()

RETURNS TRIGGER

LANGUAGE plpgsql

AS \$\$

DECLARE

cnt integer;

BEGIN

-- Проверяем, есть ли дисциплина в программе группы

SELECT COUNT(\*)

INTO cnt

FROM courses\_scheme.program\_discipline pd

```

JOIN courses_scheme.groups g ON g.program_id = pd.program_id
WHERE g.group_id = NEW.group_id
    AND pd.discipline_id = NEW.discipline_id;
IF cnt = 0 THEN
    RAISE EXCEPTION 'Дисциплина с id % не входит в программу группы с id %',
        NEW.discipline_id, NEW.group_id;
END IF;
RETURN NEW;
END;
$$;

```

```

CREATE TRIGGER trg_check_discipline_in_group_program
BEFORE INSERT OR UPDATE ON courses_scheme.classes
FOR EACH ROW
EXECUTE FUNCTION courses_scheme.fn_check_discipline_in_group_program();

```

Проверка:

```

INSERT INTO courses_scheme.classes (teacher_id, discipline_id, group_id, room_id,
class_type, class_date, class_number, class_status) VALUES (5, 10, 2, 3,
'лекционное', '2025-05-21', '1', 'Yes');

```

На рисунке 10 результат проверки.

```

ОШИБКА: Дисциплина с id 10 не входит в программу группы с id 2
КОНТЕКСТ: функция PL/pgSQL courses_scheme.fn_check_discipline_in_group_program(), строка 13, оператор RAISE

```

Рисунок 10 – Результат проверки триггера.

## **ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы были успешно освоены практические аспекты работы с хранимыми процедурами, функциями и триггерами в PostgreSQL. В рамках задания реализованы процедуры с входными и выходными параметрами, разработана серверная логика с использованием триггеров, а также выполнены дополнительные проверки для обеспечения корректности данных и предотвращения ошибок.

Были разработаны следующие ключевые компоненты:

три хранимые процедуры:

- Получение расписания для группы на определенный день недели;
- Запись студента на курс с проверкой доступности мест;
- Поиск свободных лекционных аудиторий с обработкой случая их отсутствия.

семь оригинальных триггеров, реализующих:

- Автоматическое заполнение дня недели по дате;
- Запрет удаления группы с активными студентами;
- Проверка даты зачисления студента в рамках периода существования группы;
- Предотвращение дублирования записей о практике;
- Проверка соответствия преподавателя и дисциплины;
- Контроль занятости преподавателя в одно и то же время;
- Проверка вхождения дисциплины в программу группы.

Лабораторная работа способствовала приобретению ценных навыков в области серверной разработки на PostgreSQL. В их числе: создание и использование хранимых процедур, разработка триггерных функций для автоматизации и валидации данных, реализация проверок и автоматизация процессов в СУБД. Прделанная работа позволила закрепить ключевые механизмы PostgreSQL, направленные на поддержание целостности данных.