Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИТМО»

Отчет

по лабораторной работе **«Процедуры, функции, триггеры в PostgreSQL»**
по дисциплине «Базы данных»

Автор:

Пырков Владислав

Факультет: Инфокоммуникационных технологий (ИКТ)

Группа: К32402

Преподаватель:
Говорова М. М.

Санкт-Петербург

2023 г.

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:

Вариант 1: 1. 2.Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4). 2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

Вариант 7. БД «Курсы»

Описание предметной области: Сеть учебных подразделений занимается организацией внебюджетного образования.

Имеется несколько образовательных программ краткосрочных курсов, предназначенных для определенных специальностей, связанных с программным обеспечением ИТ. Каждый программа имеет определенную длительность и свой перечень изучаемых дисциплин. Одна дисциплина может относиться к нескольким программам. На каждую программу может быть набрано несколько групп обучающихся. По каждой дисциплине могут проводиться лекционные, лабораторные/практические занятия и практика определенном объеме часов. По каждой дисциплине и практике проводится аттестация в формате экзамен/дифзачет/зачет.

Подразделение обеспечивает следующие ресурсы: учебные классы, лекционные аудитории и преподавателей. Необходимо составить расписание занятий.

БД должна содержать следующий минимальный набор сведений: Фамилия слушателя. Имя слушателя. Паспортные данные. Контакты. Код программы. Программа. Тип программы. Объем часов. Номер группы. максимальное количество человек в группе (для набора). Дата начала обучения. Дата окончания обучения. Название дисциплины. Количество часов. Дата занятий. Номер пары. Номер аудитории. Тип аудитории. Адрес площадки. Вид занятий (лекционные, практические или лабораторные). Фамилия преподавателя. Имя и отчество преподавателя. Должность преподавателя. Дисциплины, которые может вести преподаватель.

Задание 1.1 (ЛР 1 БД). Выполните инфологическое моделирование базы данных системы. (Ограничения задать самостоятельно.)

Задание 1.2. Создайте логическую модель БД, используя ИЛМ (задание 1.1). Используйте необходимые средства поддержки целостности данных в СУБД.

Задание 2. Создать запросы:

- Вывести все номера групп и программы, где количество слушателей меньше 10.
- Вывести список преподавателей с указанием количества программ, где они преподавали за истекший год.
- Вывести список преподавателей, которые не проводят занятия на третьей паре ни в один из дней недели.
- Вывести список свободных лекционных аудиторий на ближайший понедельник.
- Вычислить общее количество обучающихся по каждой программе за последний год.
- Вычислить среднюю загруженность компьютерных классов в неделю за последний месяц (в часах).
- Найти самые популярные программы за последние 3 года.

Задание 3. Создать представление:

- для потенциальных слушателей, содержащее перечень специальностей, изучаемых на них дисциплин и количество часов;
- общих доход по каждой программе за последний год.

Задание 4. Создать хранимые процедуры:

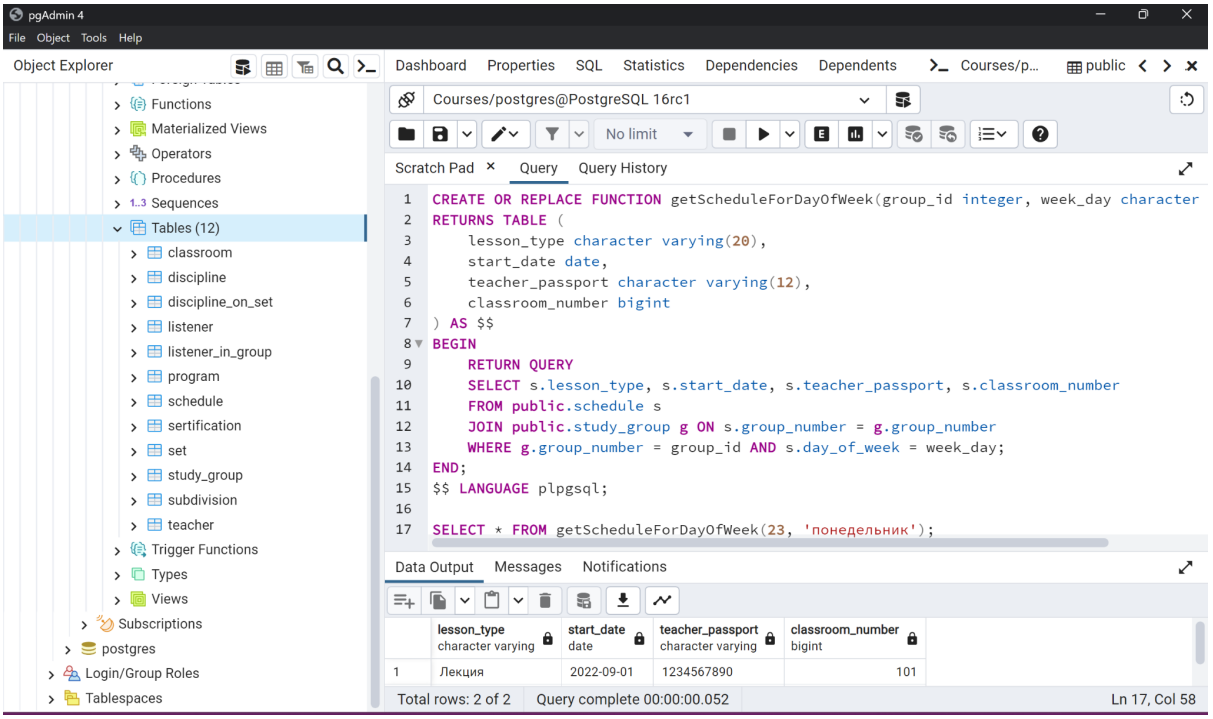
- Для получения расписания занятий для групп на определенный день недели.
- Записи на курс слушателя.
- Получения перечня свободных лекционных аудиторий на любой день недели. Если свободных аудиторий не имеется, то выдать соответствующее сообщение.

Задание 5. Создать необходимые триггеры.

Выполнение

Создать хранимые: процедуры

Для получения расписания занятий для групп на определенный день недели



CREATE OR REPLACE FUNCTION getScheduleForDayOfWeek(group_id integer, week_day character varying(20))

RETURNS TABLE (

lesson_type character varying(20),

start_date date,

teacher_passport character varying(12),

classroom_number bigint

) AS $$

BEGIN

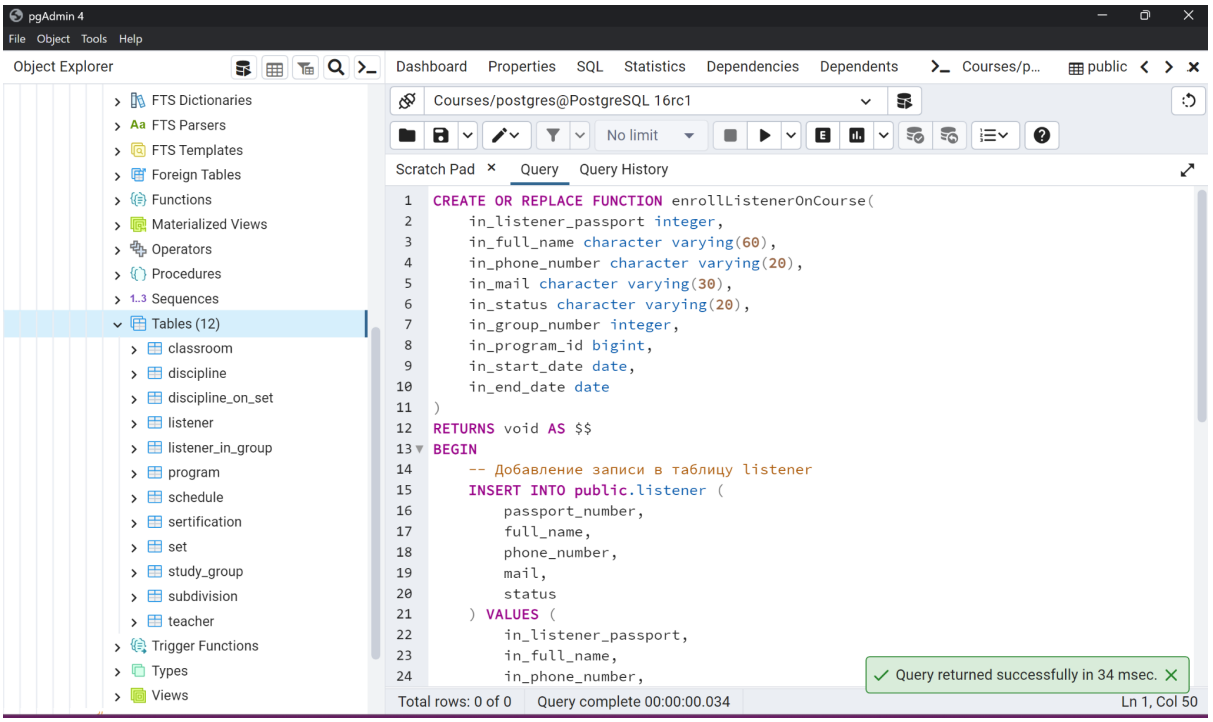
RETURN QUERY

SELECT s.lesson_type, s.start_date, s.teacher_passport, s.classroom_number

```
    FROM public.schedule s
    JOIN public.study_group g ON s.group_number = g.group_number
    WHERE g.group_number = group_id AND s.day_of_week = week_day;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM getScheduleForDayOfWeek(23, 'понедельник');
```

Записи на курс слушателя



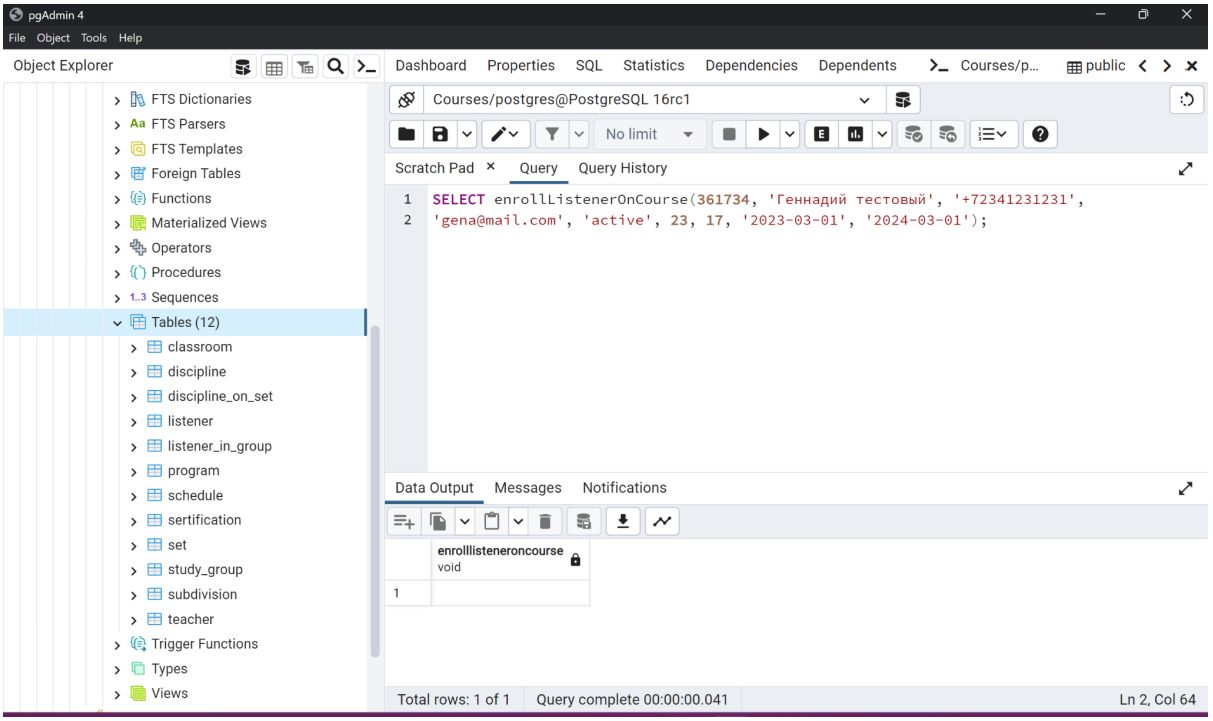

```
CREATE OR REPLACE FUNCTION enrollListenerOnCourse(
    in_listener_passport integer,
    in_full_name character varying(60),
    in_phone_number character varying(20),
    in_mail character varying(30),
    in_status character varying(20),
    in_group_number integer,
    in_program_id bigint,
    in_start_date date,
    in_end_date date
)
RETURNS void AS $$
BEGIN
```

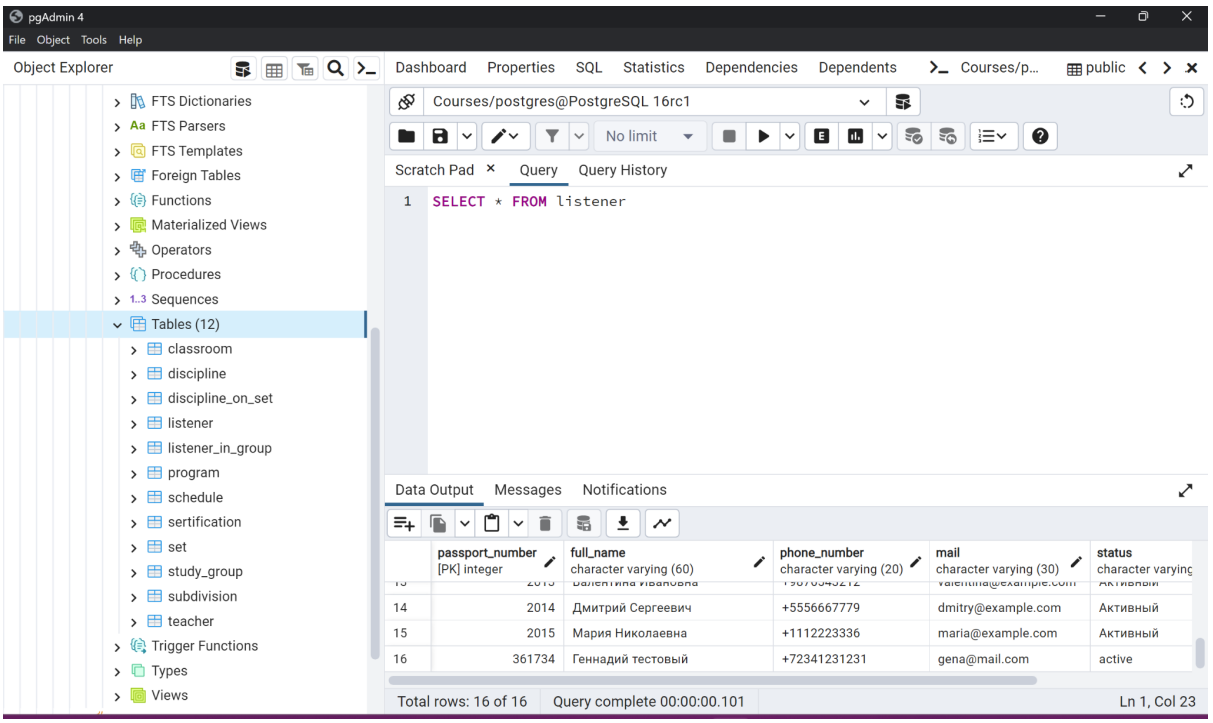
```
INSERT INTO public.listener (
    passport_number,
    full_name,
    phone_number,
    mail,
    status
) VALUES (
    in_listener_passport,
    in_full_name,
    in_phone_number,
    in_mail,
    in_status
);

INSERT INTO public.listener_in_group (
    passport_number,
    group_number,
    program_id,
    status,
    start_date,
    end_date
) VALUES (
    in_listener_passport,
```

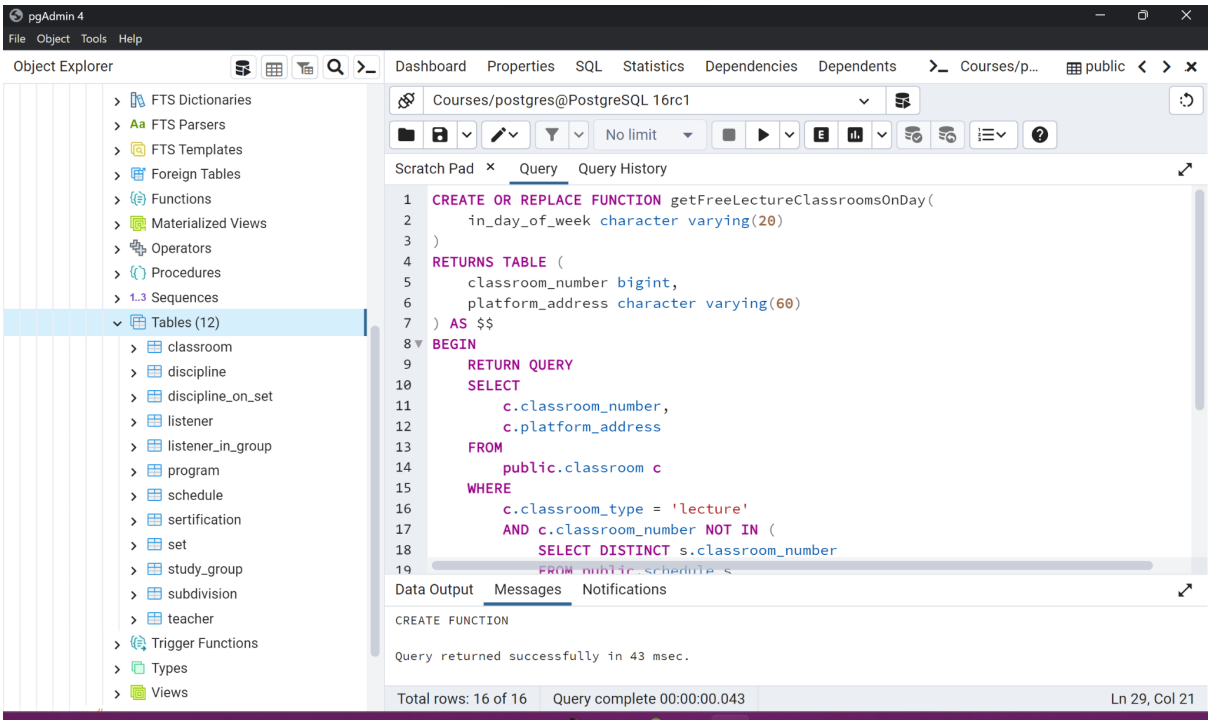
```
        in_group_number,
        in_program_id,
        in_status,
        in_start_date,
        in_end_date
    );

    EXCEPTION
    WHEN others THEN
        RAISE EXCEPTION 'Ошибка при записи слушателя на курс';
END;
$$ LANGUAGE plpgsql;
```


pgAdmin 4
File Object Tools Help
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents Courses/p... public
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (12)
classroom
discipline
discipline_on_set
listener
listener_in_group
program
schedule
sertification
set
study_group
subdivision
teacher
Trigger Functions
Types
Views
Courses/postgres@PostgreSQL 16rc1
No limit
Scratch Pad
Query
Query History
1 SELECT enrollListenerOnCourse(361734, 'Геннадий тестовый', '+72341231231',
2 'gena@mail.com', 'active', 23, 17, '2023-03-01', '2024-03-01');
Data Output
Messages
Notifications
enrolllisteneroncourse
void
1
Total rows: 1 of 1
Query complete 00:00:00.041
Ln 2, Col 64


pgAdmin 4
File Object Tools Help
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents Courses/p... public
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (12)
classroom
discipline
discipline_on_set
listener
listener_in_group
program
schedule
sertification
set
study_group
subdivision
teacher
Trigger Functions
Types
Views
Courses/postgres@PostgreSQL 16rc1
No limit
Scratch Pad
Query
Query History
1 SELECT * FROM listener
Data Output
Messages
Notifications
passport_number [PK] integer
full_name character varying (60)
phone_number character varying (20)
mail character varying (30)
status character varying
14 2014 Дмитрий Сергеевич +5556667779 dmitry@example.com Активный
15 2015 Мария Николаевна +1112223336 maria@example.com Активный
16 361734 Геннадий тестовый +72341231231 gena@mail.com active
Total rows: 16 of 16
Query complete 00:00:00.101
Ln 1, Col 23

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

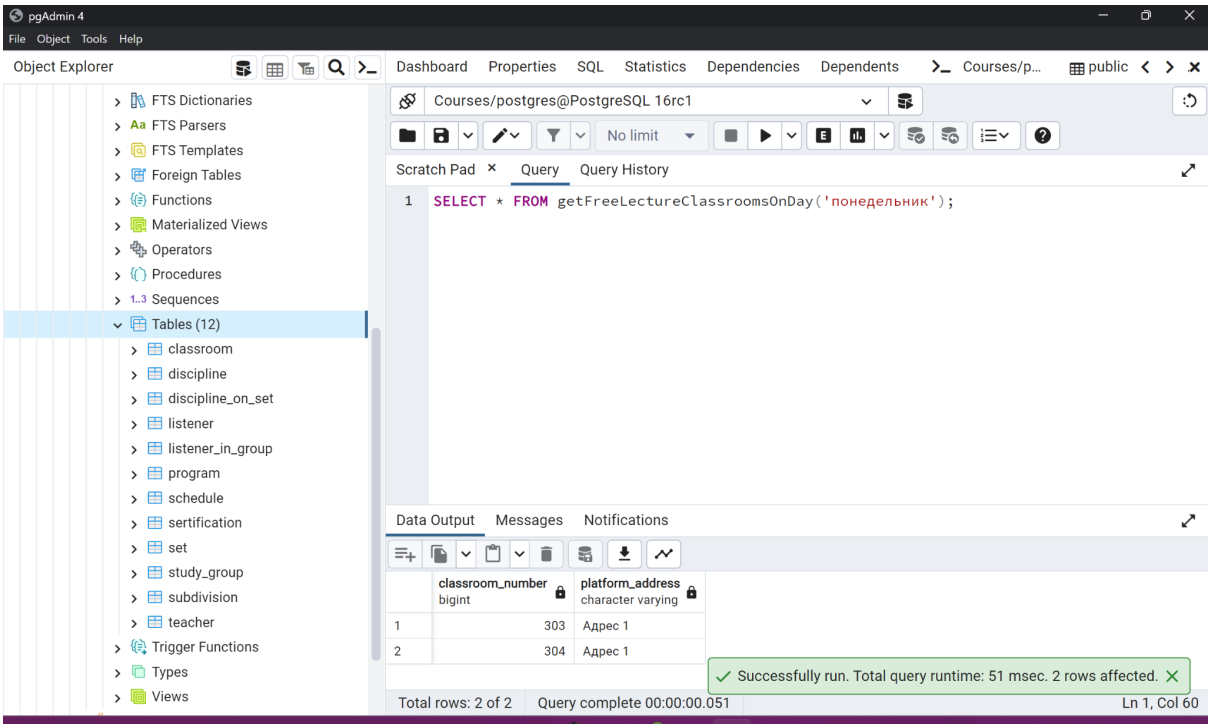



```
CREATE OR REPLACE FUNCTION getFreeLectureClassroomsOnDay(
    in_day_of_week character varying(20)
)
RETURNS TABLE (
    classroom_number bigint,
    platform_address character varying(60)
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        c.classroom_number,
        c.platform_address
    FROM
```

```
        public.classroom c
    WHERE
        c.classroom_type = 'lecture'
        AND c.classroom_number NOT IN (
            SELECT DISTINCT s.classroom_number
            FROM public.schedule s
            WHERE s.day_of_week = in_day_of_week
        )
    ORDER BY
        c.classroom_number;


    IF NOT FOUND THEN
        RAISE EXCEPTION 'Свободных лекционных аудиторий на % не найдено.',
in_day_of_week;
    END IF;
END;
```

$$ LANGUAGE plpgsql;



SELECT * FROM getFreeLectureClassroomsOnDay('понедельник');

Создание триггеров

Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

```
Courses=# -- Создание функции для логирования событий
Courses=# CREATE OR REPLACE FUNCTION log_classroom_changes()
Courses-# RETURNS TRIGGER AS $$
Courses$# BEGIN
Courses$#     IF TG_OP = 'INSERT' THEN
Courses$#         RAISE NOTICE 'INSERT: classroom_number = %, platform_address = %, classroom_type = %',
Courses$#                      NEW.classroom_number, NEW.platform_address, NEW.classroom_type;
Courses$#     ELSIF TG_OP = 'UPDATE' THEN
Courses$#         RAISE NOTICE 'UPDATE: classroom_number = %, platform_address = %, classroom_type = %',
Courses$#                      NEW.classroom_number, NEW.platform_address, NEW.classroom_type;
Courses$#     ELSIF TG_OP = 'DELETE' THEN
Courses$#         RAISE NOTICE 'DELETE: classroom_number = %, platform_address = %, classroom_type = %',
Courses$#                      OLD.classroom_number, OLD.platform_address, OLD.classroom_type;
Courses$#     END IF;
Courses$#     RETURN NEW;
Courses$# END;
Courses$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
Courses=#
Courses=# -- Создание триггера для логирования событий в таблице "classroom"
Courses=# CREATE TRIGGER log_classroom_changes_trigger
Courses-# AFTER INSERT OR UPDATE OR DELETE ON classroom
Courses-# FOR EACH ROW
Courses-# EXECUTE FUNCTION log_classroom_changes();
CREATE TRIGGER
Courses=#
```

-- Создание функции для логирования событий

CREATE OR REPLACE FUNCTION log_classroom_changes()

RETURNS TRIGGER AS $$

BEGIN

IF TG_OP = 'INSERT' THEN

RAISE NOTICE 'INSERT: classroom_number = %, platform_address = %, classroom_type = %',

NEW.classroom_number, NEW.platform_address, NEW.classroom_type;

ELSIF TG_OP = 'UPDATE' THEN

RAISE NOTICE 'UPDATE: classroom_number = %, platform_address = %, classroom_type = %',

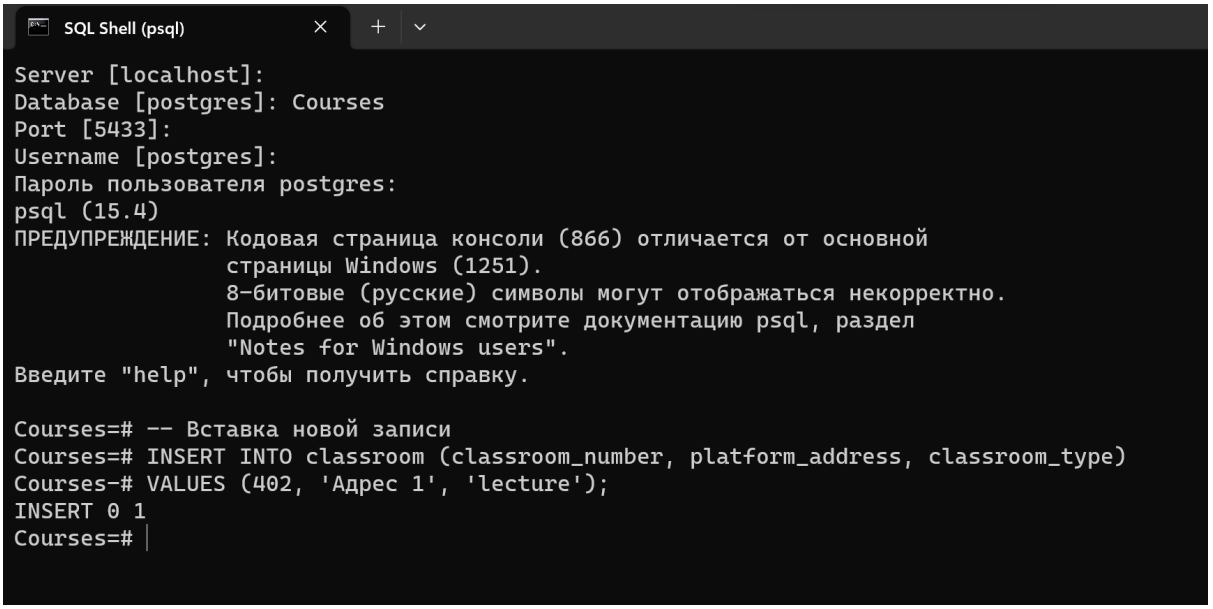
NEW.classroom_number, NEW.platform_address, NEW.classroom_type;

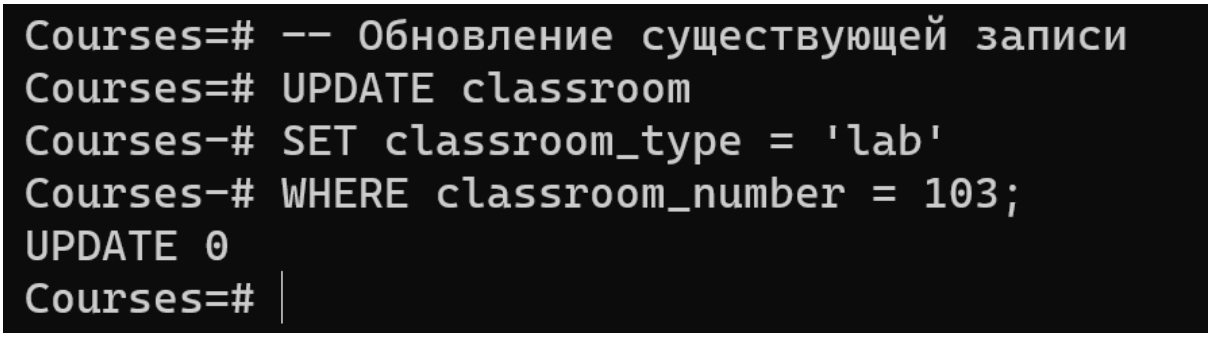
ELSIF TG_OP = 'DELETE' THEN

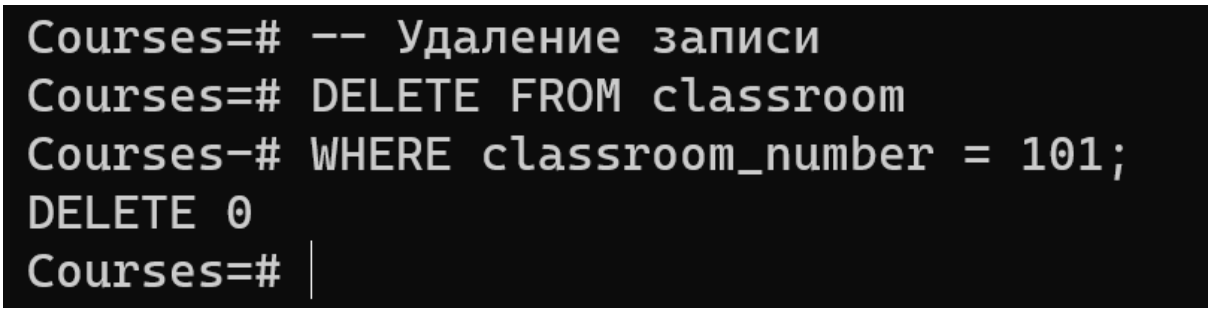
RAISE NOTICE 'DELETE: classroom_number = %, platform_address = %, classroom_type = %',

```
            OLD.classroom_number, OLD.platform_address, OLD.classroom_type;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;


-- Создание триггера для логирования событий в таблице "classroom"
CREATE TRIGGER log_classroom_changes_trigger
AFTER INSERT OR UPDATE OR DELETE ON classroom
FOR EACH ROW
EXECUTE FUNCTION log_classroom_changes();
```


SQL Shell (psql)
Server [localhost]:
Database [postgres]: Courses
Port [5433]:
Username [postgres]:
Пароль пользователя postgres:
psql (15.4)
ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной
страницы Windows (1251).
8-битовые (русские) символы могут отображаться некорректно.
Подробнее об этом смотрите документацию psql, раздел
"Notes for Windows users".
Введите "help", чтобы получить справку.
Courses=# -- Вставка новой записи
Courses=# INSERT INTO classroom (classroom_number, platform_address, classroom_type)
Courses-# VALUES (402, 'Адрес 1', 'lecture');
INSERT 0 1
Courses=#


Courses=# -- Обновление существующей записи
Courses=# UPDATE classroom
Courses-# SET classroom_type = 'lab'
Courses-# WHERE classroom_number = 103;
UPDATE 0
Courses=#


Courses=# -- Удаление записи
Courses=# DELETE FROM classroom
Courses-# WHERE classroom_number = 101;
DELETE 0
Courses=#

Выводы

В ходе выполнения работы были приобретены навыки по созданию процедур и триггеров, полезных при наличии большого количества данных, которые необходимо постоянно отслеживать.