

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:  
Инфокоммуникацио  
нных технологий  
(ИКТ)

Группа: K32402

**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. Каждая 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:

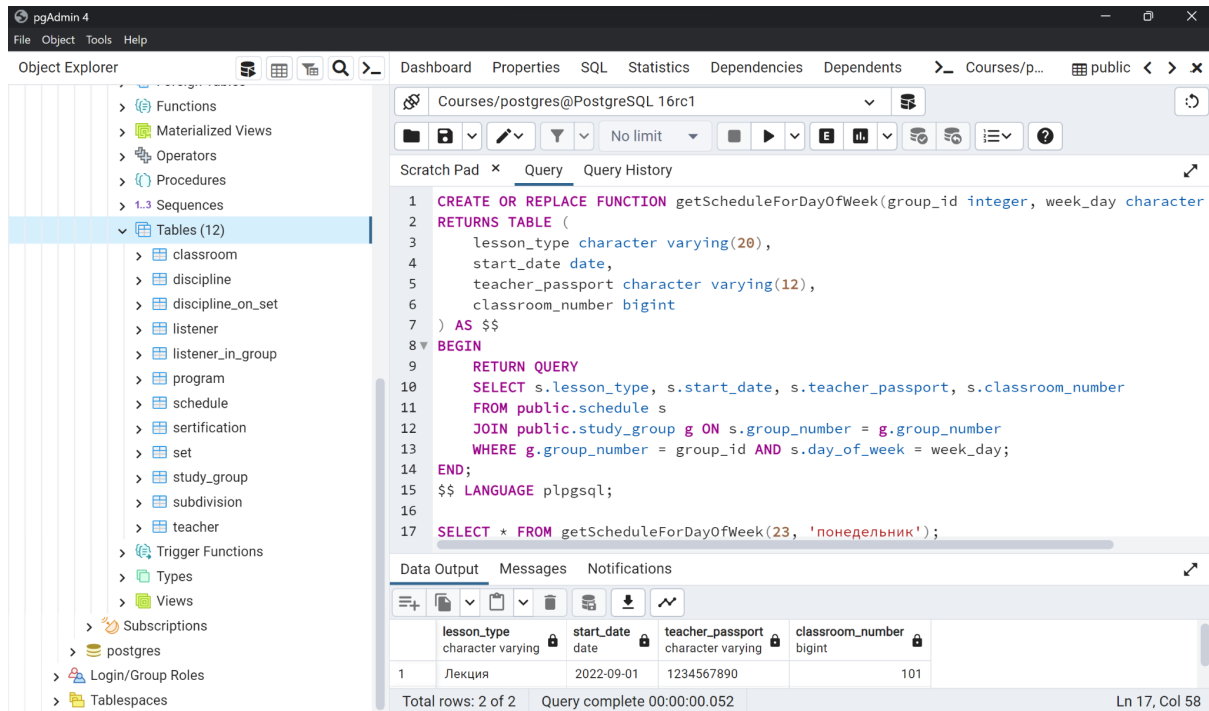
- Для получения расписания занятий для групп на определенный день недели.
- Записи на курс слушателя.
- Получения перечня свободных лекционных аудиторий на любой день недели. Если свободных аудиторий не имеется, то выдать соответствующее сообщение.

**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

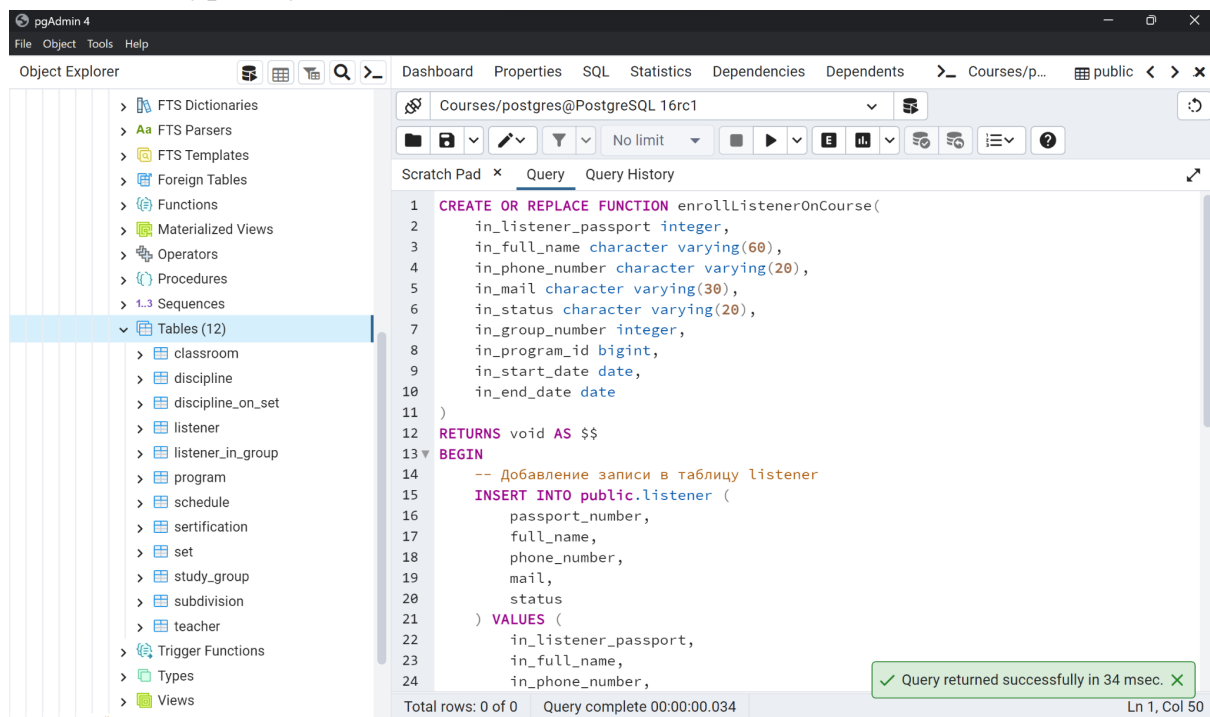
WHERE g.group_number = group_id AND s.day_of_week = week_day;

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

- 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

Dashboard Properties SQL Statistics Dependencies Dependents Courses/p... public

Courses/postgres@PostgreSQL 16rc1

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

- 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

Dashboard Properties SQL Statistics Dependencies Dependents Courses/p... public

Courses/postgres@PostgreSQL 16rc1

Scratch Pad Query Query History

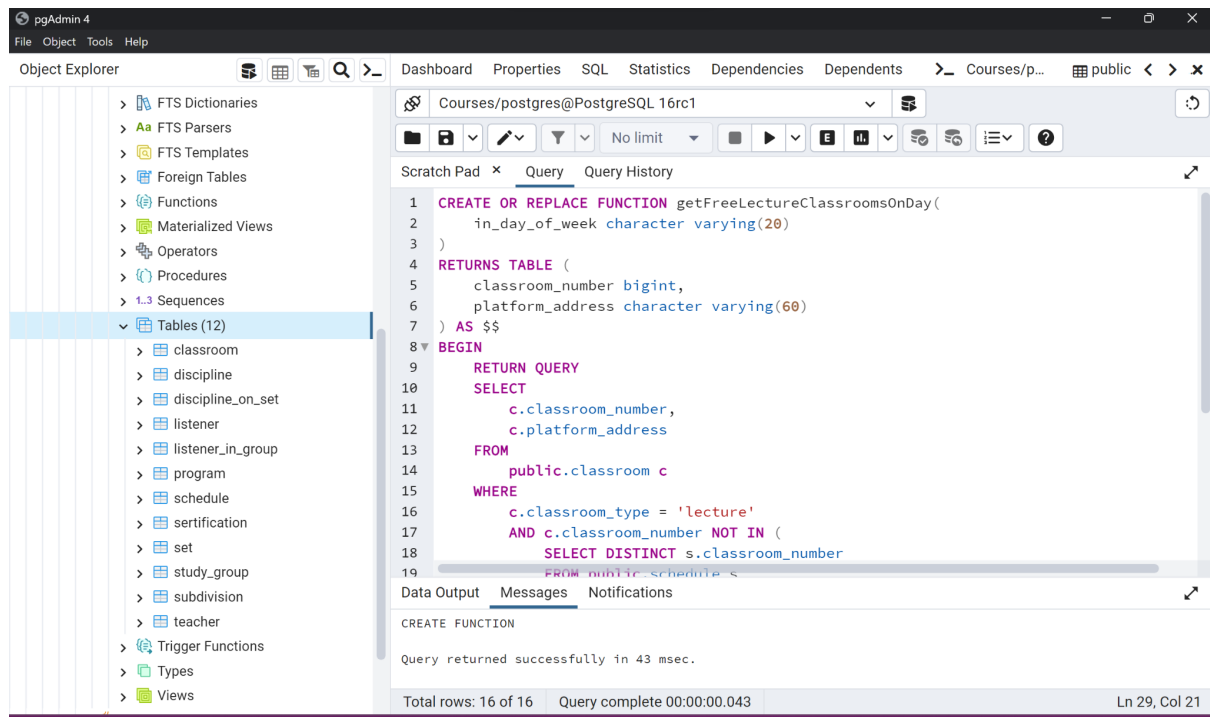
```
1 SELECT * FROM listener
```

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
2013	Валентина Ивановна	+7070049212	valentina@example.com	Активный
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;

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Object Explorer' pane is open, showing a tree of database objects. The 'Tables (12)' folder is expanded, listing tables such as 'classroom', 'discipline', 'discipline\_on\_set', 'listener', 'listener\_in\_group', 'program', 'schedule', 'sertification', 'set', 'study\_group', 'subdivision', 'teacher', 'Trigger Functions', 'Types', and 'Views'. The 'classroom' table is selected.

The main pane displays a SQL query in the 'Query' tab:

```
1 SELECT * FROM getFreeLectureClassroomsOnDay('понедельник');
```

The 'Data Output' tab shows the results of the query in a table format:

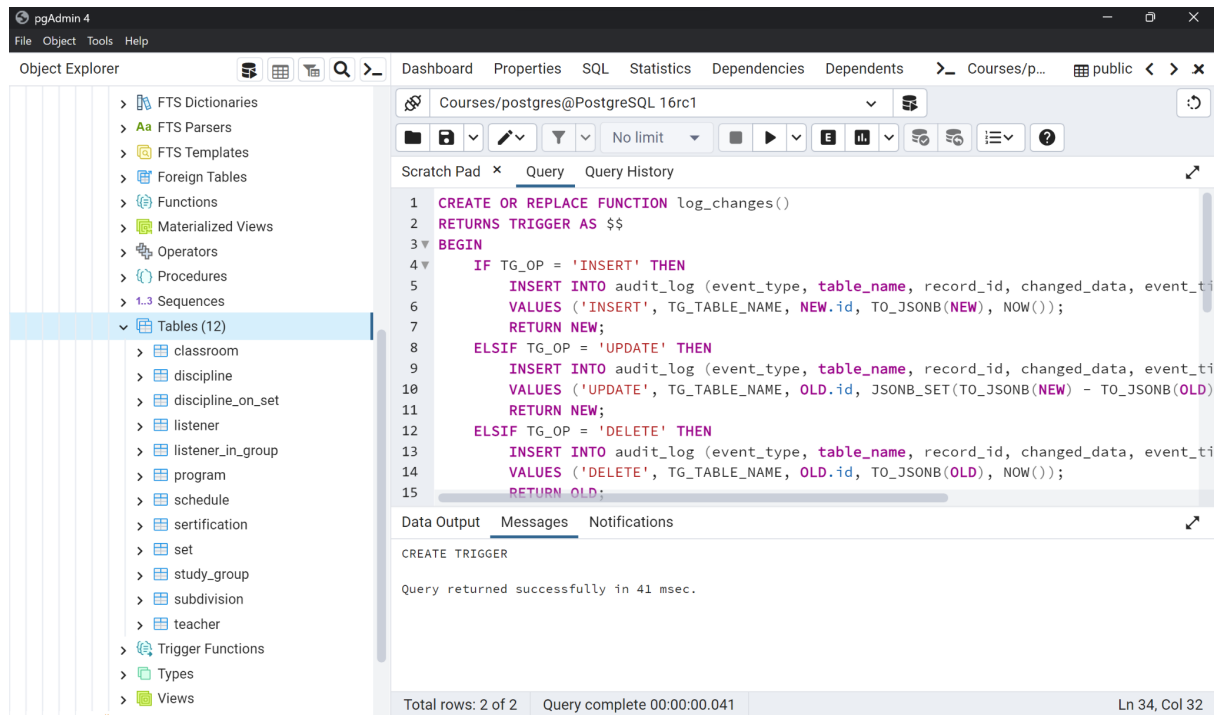
	classroom_number bigint	platform_address character varying
1	303	Адрес 1
2	304	Адрес 1

Below the table, a green status bar indicates: 'Successfully run. Total query runtime: 51 msec. 2 rows affected.' The bottom status bar shows 'Total rows: 2 of 2' and 'Query complete 00:00:00.051'.

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.



```
CREATE OR REPLACE FUNCTION log_changes()
```

```
RETURNS TRIGGER AS $$
```

```
BEGIN
```

```
    IF TG_OP = 'INSERT' THEN
```

```
        INSERT INTO audit_log (event_type, table_name, record_id, changed_data, event_timestamp)
```

```
        VALUES ('INSERT', TG_TABLE_NAME, NEW.id, TO_JSONB(NEW), NOW());
```

```
        RETURN NEW;
```

```
    ELSIF TG_OP = 'UPDATE' THEN
```

```
        INSERT INTO audit_log (event_type, table_name, record_id, changed_data, event_timestamp)
```

```
        VALUES ('UPDATE', TG_TABLE_NAME, OLD.id, JSONB_SET(TO_JSONB(NEW) - TO_JSONB(OLD), '{id}', TO_JSONB(OLD->'id')), NOW());
```

```
        RETURN NEW;

    ELSIF TG_OP = 'DELETE' THEN

        INSERT INTO audit_log (event_type, table_name, record_id, changed_data,
event_timestamp)

            VALUES ('DELETE', TG_TABLE_NAME, OLD.id, TO_JSONB(OLD), NOW());

        RETURN OLD;

    END IF;

    RETURN NULL;

END;

$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER log_insert

AFTER INSERT ON schedule

FOR EACH ROW

EXECUTE FUNCTION log_changes();
```

```
CREATE TRIGGER log_update

AFTER UPDATE ON schedule

FOR EACH ROW

EXECUTE FUNCTION log_changes();
```

```
CREATE TRIGGER log_delete

AFTER DELETE ON schedule

FOR EACH ROW

EXECUTE FUNCTION log_changes();
```



## **Выводы**

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