Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5 «Процедуры, функции, триггеры в PostgreSQL» по дисциплине «Проектирование и реализация баз данных»

Автор: Кадникова Е.М.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

Вариант 7. БД «Курсы»	3
Ход работы	4
Вывол 1	12

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, pgadmin 4.

Практическое задание:

- 1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
- 2.1. 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).
 - 2.2. Создать авторский триггер по варианту индивидуального задания.

Вариант 7. БД «Курсы»

Описание предметной области: Сеть учебных подразделений НОУ ДПО 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.

Необходимо хранить информацию по аттестации обучающихся.

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 сообшение.

Задание 5. Создать необходимые триггеры.

Ход работы:

1. Процедура для получения расписания занятий для групп на определенный день недели:

CREATE OR REPLACE FUNCTION lab3.get_schedule_for_weekday(weekday_param VARCHAR(32))

RETURNS TABLE (lesson_time VARCHAR(5), lesson_type VARCHAR(24), subject VARCHAR(48), teacher TEXT, adress VARCHAR(36), auditorium INTEGER)

LANGUAGE plpgsql **AS \$\$ BEGIN** RETURN OUERY (SELECT l.lesson_time, s.lesson_type, sub.name_subject, concat(t.surname_teacher,'',t.name_teacher,'', t.middle name teacher) AS teacher, a.adress, a.number aud FROM lab3."Lesson" 1 JOIN lab3."Schedule" s $ON 1.id_{lesson} = s.id_{lesson}$ JOIN lab3."Teacher" t

ON s.id_teacher = t.id_teacher JOIN lab3."Study subject on programm" st

ON s.id_subject_on_programm = st.id_subject_on_programm

JOIN lab3. "Study subject" sub

ON st.id subject = sub.id subject

JOIN lab3."Auditorium" a

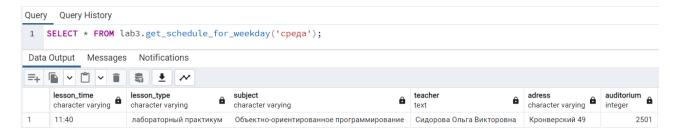
ON s.id aud = a.id aud

WHERE l.weekday = weekday_param);

END;

\$\$

```
1 CREATE OR REPLACE FUNCTION lab3.get_schedule_for_weekday(weekday_param VARCHAR(32))
    RETURNS TABLE (lesson_time VARCHAR(5), lesson_type VARCHAR(24), subject VARCHAR(48), teacher TEXT, adress VARCHAR(36), auditorium INTEGER)
5 LANGUAGE plpgsql
7 ▼ BEGIN
    SELECT l.lesson_time, s.lesson_type, sub.name_subject, concat(t.surname_teacher,' ',t.name_teacher,' ', t.middle_name_teacher) AS teacher,
    FROM lab3."Lesson" l
    JOIN lab3."Schedule"
   JOIN lab3."Teacher" t
ON s.id_teacher = t.id_teacher
    JOIN lab3."Study subject on programm" st
    ON s.id_subject_on_programm = st.id_subject_on_programm
JOIN lab3."Study subject" sub
ON st.id_subject = sub.id_subject
    JOIN lab3."Auditorium" a
    ON s.id_aud = a.id_aud
    WHERE l.weekday = weekday_param);
24 END:
Data Output Messages Notifications
CREATE FUNCTION
 Ouerv returned successfully in 42 msec.
```



2. Процедура для записи слушателя на курс:

CREATE OR REPLACE PROCEDURE lab3.join_course(id_param INTEGER, surname_param VARCHAR(24), name_param VARCHAR(24), middle_name_param VARCHAR(24), passport_param INTEGER, contacts VARCHAR(24), id_programm_param INTEGER, type_recruit_param VARCHAR(24))

LANGUAGE plpgsql

AS \$\$

BEGIN

IF NOT EXISTS (SELECT * FROM lab3."Student" s WHERE s.id_student = id_param) THEN INSERT INTO lab3."Student" VALUES (id_param, surname_param, name_param, middle_name_param, passport_param, contacts);
END IF:

IF NOT EXISTS (SELECT * FROM lab3."Group on programm" g WHERE (SELECT COUNT (*) FROM lab3."Student in group" sg WHERE g.id_group = sg.id_group) < g.max_people_count AND g.id_programm = id_programm_param) THEN

RAISE NOTICE 'Нет доступных групп';

ELSE

IF NOT EXISTS (SELECT * FROM lab3."Student in group" sg JOIN lab3."Group on programm" g ON sg.id_group = g.id_group WHERE sg.id_student = id_param AND g.id_programm = id_programm_param) THEN

INSERT INTO lab3."Student in group"

VALUES

(id_param,

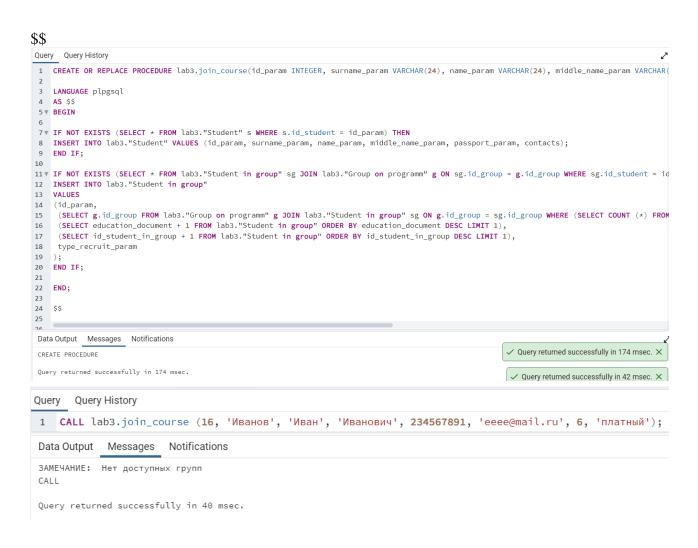
(SELECT g.id_group FROM lab3."Group on programm" g JOIN lab3."Student in group" sg ON g.id_group = sg.id_group WHERE (SELECT COUNT (*) FROM lab3."Student in group" sg WHERE g.id_group = sg.id_group) < g.max_people_count ORDER BY g.id_group LIMIT 1),

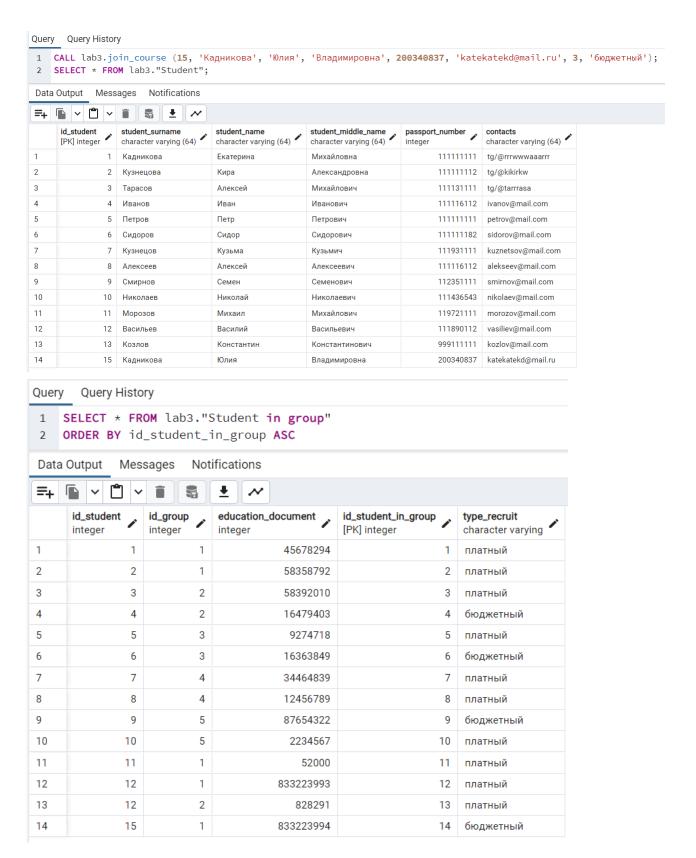
(SELECT education_document + 1 FROM lab3."Student in group" ORDER BY education_document DESC LIMIT 1),

(SELECT id_student_in_group + 1 FROM lab3."Student in group" ORDER BY id_student_in_group DESC LIMIT 1),

```
type_recruit_param
);
END IF;
END IF;
```

END;





3. Процедура для получения перечня свободных лекционных аудиторий на любой день недели:

CREATE OR REPLACE FUNCTION lab3.get_free_auds_for_weekday(weekday_param VARCHAR(16))

RETURNS TABLE(adress VARCHAR(48), aud_type VARCHAR(24), number INTEGER)

LANGUAGE plpgsql

```
AS $$
BEGIN

RETURN QUERY (SELECT a.adress, a.type_aud, a.number_aud
FROM lab3."Auditorium" a

WHERE id_aud NOT IN (SELECT DISTINCT a.id_aud FROM lab3."Auditorium" a

JOIN lab3."Aud_buzyness" b

ON a.id_aud = b.id_aud

WHERE b.buzyness = 'занята' AND b.weekday = weekday_param));

END;

$$

Query Query History

1 CREATE OR REPLACE FUNCTION lab3.get_free_auds_for_weekday(weekday_auditorium)

1 CREATE OR REPLACE FUNCTION lab3.get_free_auds_for_weekday(weekday_auditorium)

2 DETURNS TABLE (c.b. auditorium) lab3.get_free_auds_for_weekday(weekday_auditorium)
```

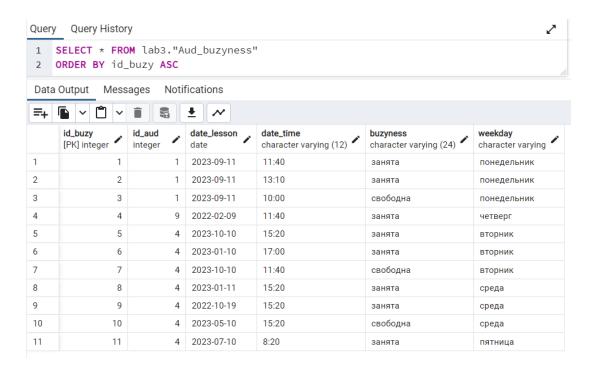
```
CREATE OR REPLACE FUNCTION lab3.get_free_auds_for_weekday(weekday_param VARCHAR(16))
 3
     RETURNS TABLE(adress VARCHAR(48), aud_type VARCHAR(24), number INTEGER)
 4
 5
    LANGUAGE plpgsql
    AS $$
 6
 7 ▼ BEGIN
 8
 9
    RETURN QUERY (SELECT a.adress, a.type_aud, a.number_aud
    FROM lab3."Auditorium" a
10
    WHERE id_aud NOT IN (SELECT DISTINCT a.id_aud FROM lab3."Auditorium" a
11
       JOIN lab3."Aud_buzyness" b
12
13
       ON a.id_aud = b.id_aud
       WHERE b.buzyness = 'занята' AND b.weekday = weekday_param));
14
15
16
    END;
17
    $$
Data Output
             Messages
                         Notifications
CREATE FUNCTION
Query returned successfully in 42 msec.
Query Query History
 1 SELECT * FROM lab3.get_free_auds_for_weekday('среда');
Data Output
            Messages
                       Notifications
=+
                       S
                           #
                      aud_type
                                      number
                                            a
     character varying
                      character varying
                                      integer
      Чайковского 11
                      лекционная
                                           101
2
      Ломоносова 9
                      лабораторная
                                          2102
                                          2201
3
      Ломоносова 9
                      практическая
      Ломоносова 9
                                          4301
4
                      коворкинг
5
      Биржевая 14
                      офис
                                           302
6
      Биржевая 14
                                           401
                      лекционная
7
      Ломоносова 9
                                          4402
                      практическая
8
      Кронверский 49
                      лекционная
                                          2501
```

502

9

Чайковского 11

лабораторная



4. Модифицировать триггер на проверку корректности входа ивыхода сотрудника (имеющиеся проблемы: может быть отрицательное время работы, человек зашел/вышел в будущем)

create or replace function fn_check_time_punch() returns trigger as \$psql\$ begin

drop trigger if exists check_time_punch on time_punch; create trigger check_time_punch before insert on time_punch for each row

execute procedure fn_check_time_punch();

```
replace function fn_check_time_punch() returns trigger as $psql$ begin
 mp_time$#
mp_time$# if
 mp_time$# new.is_out_punch = (select tps.is_out_punch from time_punch tps
mp_time$# where tps.employee_id = new.employee_id order by tps.id desc limit 1 )
  np time$# new.punch time>now()
  np_imes# of
up_time$# new.punch_time <= (select tps.punch_time from time_punch tps
np_time$# where tps.employee_id = new.employee_id order by tps.id desc limit 1 )
  np_time$#
np_time$# then return null;
 mp_time$# end if; return new;
mp_time$# end;
 mp_time$# $psql$ language plpgsql;
REATE FUNCTION

mp_time=# drop trigger if exists check_time_punch on time_punch; create trigger check_time_punch

ROP TRIGGER
 mp_time-# before insert on time_punch for each row
imp_time-# execute procedure fn_check_time_punch();
REATE TRIGGER
cheart mades
emp_time=# select * from time_punch;
id | employee_id | is_out_punch |
                                                                                                                                                 punch_time
                                                                                                                                 2021-01-01 10:00:00
                                                                                                                               2021-01-01 10:00:00 2023-10-15 10:00:00 2023-10-15 10:00:00 2023-10-15 17:30:00 2023-10-10 20 14:00:00 2021-01-02 16:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 17:00:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 10:00 2023-10-15 
                                                                                                                                 2023-10-15 17:00:00
2023-10-15 17:00:00
                                                                                                                                2023-03-04 10:00:00
2023-03-04 15:00:00
   mp_time=# insert into time_punch(employee_id, is_out_punch, punch_time) values (1, false, '2023-01-22 13:13:13'), (1, true, '2022-01-22 13:13:13');
```

5. Авторский триггер для обновления списка студентов при отчислении с курса:

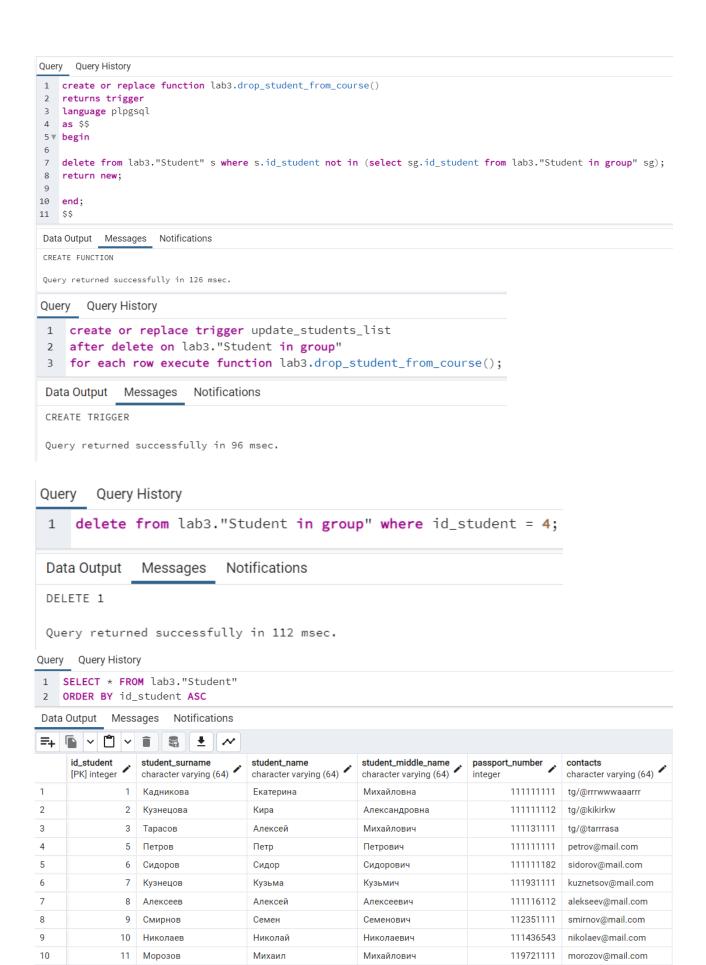
```
create or replace function lab3.drop_student_from_course()
returns trigger
language plpgsql
as $$
begin
```

delete from lab3."Student" s where s.id_student not in (select sg.id_student from lab3."Student in group" sg);

return new;

end; \$\$

create or replace trigger update_students_list after delete on lab3."Student in group" for each row execute function lab3.drop_student_from_course();



Васильевич

Владимировна

111890112

200340837

vasiliev@mail.com

katekatekd@mail.ru

Василий

Юлия

11

12

12

15

Васильев

Кадникова

Вывод

В ходе выполнения данной лабораторной работы научились создавать и работать с процедурами, функциями и триггерами в PostgreSQL.