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

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 сообщение.

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))

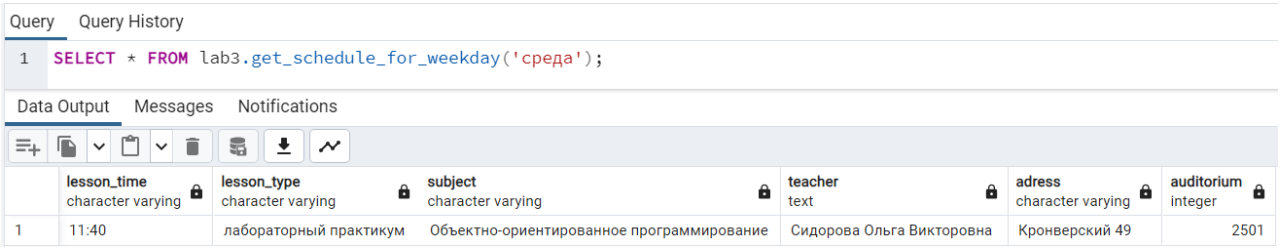
RETURNS TABLE (lesson_time VARCHAR(5), lesson_type VARCHAR(24), subject VARCHAR(48), teacher TEXT, adress VARCHAR(36), auditorium INTEGER)

LANGUAGE plpgsql
AS $$
BEGIN

RETURN QUERY (
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" l
JOIN lab3."Schedule" s
ON l.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;

$$



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."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;
```

$$

Query | Query History

```
CREATE OR REPLACE PROCEDURE lab3.join_course(id_param INTEGER, surname_param VARCHAR(24), name_param VARCHAR(24), middle_name_param VARCHAR(

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."Student in group" sg JOIN lab3."Group on programm" g ON sg.id_group = g.id_group WHERE sg.id_student = id
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
 (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;

$$
```

Data Output | Messages | Notifications

```
CREATE PROCEDURE

Query returned successfully in 174 msec.
```

✓ Query returned successfully in 174 msec. ✕

✓ Query returned successfully in 42 msec. ✕

Query | Query History

```
CALL lab3.join_course (15, 'Кадникова', 'Юлия', 'Владимировна', 200340837, 'katekatekd@mail.ru', 3, 'бюджетный');
SELECT * FROM lab3."Student";
```

Data Output | Messages | Notifications

	id_student [PK] integer	student_surname character varying (64)	student_name character varying (64)	student_middle_name character varying (64)	passport_number integer	contacts character varying (64)
1	1	Кадникова	Екатерина	Михайловна	111111111	tg/@rrrwwwaaarrr
2	2	Кузнецова	Кира	Александровна	111111112	tg/@kikirkw
3	3	Тарасов	Алексей	Михайлович	111131111	tg/@tarrrasa
4	4	Иванов	Иван	Иванович	111116112	ivanov@mail.com
5	5	Петров	Петр	Петрович	111111111	petrov@mail.com
6	6	Сидоров	Сидор	Сидорович	111111182	sidorov@mail.com
7	7	Кузнецов	Кузьма	Кузьмич	111931111	kuznetsov@mail.com
8	8	Алексеев	Алексей	Алексеевич	111116112	alekseev@mail.com
9	9	Смирнов	Семен	Семенович	112351111	smirnov@mail.com
10	10	Николаев	Николай	Николаевич	111436543	nikolaev@mail.com
11	11	Морозов	Михаил	Михайлович	119721111	morozov@mail.com
12	12	Васильев	Василий	Васильевич	111890112	vasiliev@mail.com
13	13	Козлов	Константин	Константинович	999111111	kozlov@mail.com
14	15	Кадникова	Юлия	Владимировна	200340837	katekatekd@mail.ru

Query Query History

```
SELECT * FROM lab3."Student in group"
ORDER BY id_student_in_group ASC
```

Data Output Messages Notifications

	id_student integer	id_group integer	education_document integer	id_student_in_group [PK] integer	type_recruit character varying
1	1	1	45678294	1	платный
2	2	1	58358792	2	платный
3	3	2	58392010	3	платный
4	4	2	16479403	4	бюджетный
5	5	3	9274718	5	платный
6	6	3	16363849	6	бюджетный
7	7	4	34464839	7	платный
8	8	4	12456789	8	платный
9	9	5	87654322	9	бюджетный
10	10	5	2234567	10	платный
11	11	1	52000	11	платный
12	12	1	833223993	12	платный
13	12	2	828291	13	платный
14	15	1	833223994	14	бюджетный

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

```sql
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;
$$
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 42 msec.
```

Query | Query History

```sql
SELECT * FROM lab3.get_free_auds_for_weekday('среда');
```

Data Output | Messages | Notifications

	adress character varying	aud_type character varying	number integer
1	Чайковского 11	лекционная	101
2	Ломоносова 9	лабораторная	2102
3	Ломоносова 9	практическая	2201
4	Ломоносова 9	коворкинг	4301
5	Биржевая 14	офис	302
6	Биржевая 14	лекционная	401
7	Ломоносова 9	практическая	4402
8	Кронверский 49	лекционная	2501
9	Чайковского 11	лабораторная	502

Query Query History

```
1  SELECT * FROM lab3."Aud_buzyness"
2  ORDER BY id_buzy ASC
```

Data Output Messages Notifications

	id_buzy [PK] integer	id_aud integer	date_lesson date	date_time character varying (12)	buzyness character varying (24)	weekday character varying
1	1	1	2023-09-11	11:40	занята	понедельник
2	2	1	2023-09-11	13:10	занята	понедельник
3	3	1	2023-09-11	10:00	свободна	понедельник
4	4	9	2022-02-09	11:40	занята	четверг
5	5	4	2023-10-10	15:20	занята	вторник
6	6	4	2023-01-10	17:00	занята	вторник
7	7	4	2023-10-10	11:40	свободна	вторник
8	8	4	2023-01-11	15:20	занята	среда
9	9	4	2022-10-19	15:20	занята	среда
10	10	4	2023-05-10	15:20	свободна	среда
11	11	4	2023-07-10	8:20	занята	пятница

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

if
new.is_out_punch = (select tps.is_out_punch from time_punch tps
        where tps.employee_id = new.employee_id order by tps.id desc limit 1 )
or
new.punch_time>now()
or
new.punch_time <= (select tps.punch_time from time_punch tps
where tps.employee_id = new.employee_id order by tps.id desc limit 1 )

then return null;
end if; return new;
end;

$psql$ language plpgsql;



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();
```

```
emp_time=# create or replace function fn_check_time_punch() returns trigger as $psql$ begin
emp_time$#
emp_time$# if
emp_time$# new.is_out_punch = (select tps.is_out_punch from time_punch tps
emp_time$# where tps.employee_id = new.employee_id order by tps.id desc limit 1 )
emp_time$# or
emp_time$# new.punch_time>now()
emp_time$# or
emp_time$# new.punch_time <= (select tps.punch_time from time_punch tps
emp_time$# where tps.employee_id = new.employee_id order by tps.id desc limit 1 )
emp_time$#
emp_time$# then return null;
emp_time$# end if; return new;
emp_time$# end;
emp_time$#
emp_time$# $psql$ language plpgsql;
CREATE FUNCTION
emp_time=# drop trigger if exists check_time_punch on time_punch; create trigger check_time_punch
DROP TRIGGER
emp_time-# before insert on time_punch for each row
emp_time-#
emp_time-# execute procedure fn_check_time_punch();
CREATE TRIGGER
emp_time=# select * from time_punch;
 id | employee_id | is_out_punch |     punch_time
----+-------------+--------------+---------------------
  1 |           1 | f            | 2021-01-01 10:00:00
  2 |           1 | t            | 2021-01-01 11:30:00
  3 |           1 | f            | 2023-10-15 10:00:00
  4 |           1 | t            | 2023-10-15 17:30:00
  5 |           1 | f            | 2021-01-02 14:00:00
  7 |           1 | t            | 2021-01-02 16:00:00
  8 |           1 | f            | 2023-10-15 17:00:00
  9 |           1 | t            | 2023-10-15 10:30:00
 10 |           1 | f            | 2023-10-15 17:00:00
 11 |           1 | t            | 2023-10-15 17:00:00
 12 |           3 | f            | 2023-03-04 10:00:00
 13 |           3 | t            | 2023-03-04 15:00:00
(12 строк)


emp_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');
INSERT 0 0
```

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();
```

Query | Query History

```
1  create or replace function lab3.drop_student_from_course()
2  returns trigger
3  language plpgsql
4  as $$
5  begin
6
7  delete from lab3."Student" s where s.id_student not in (select sg.id_student from lab3."Student in group" sg);
8  return new;
9
10 end;
11 $$
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 126 msec.
```

Query | Query History

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

Data Output | Messages | Notifications

```
CREATE TRIGGER

Query returned successfully in 96 msec.
```

Query | Query History

```
1  delete from lab3."Student in group" where id_student = 4;
```

Data Output | Messages | Notifications

```
DELETE 1

Query returned successfully in 112 msec.
```

Query | Query History

```
1  SELECT * FROM lab3."Student"
2  ORDER BY id_student ASC
```

Data Output | Messages | Notifications

	id_student [PK] integer	student_surname character varying (64)	student_name character varying (64)	student_middle_name character varying (64)	passport_number integer	contacts character varying (64)
1	1	Кадникова	Екатерина	Михайловна	111111111	tg/@rrrwwwaaarrr
2	2	Кузнецова	Кира	Александровна	111111112	tg/@kikirkw
3	3	Тарасов	Алексей	Михайлович	111131111	tg/@tarrrasa
4	5	Петров	Петр	Петрович	111111111	petrov@mail.com
5	6	Сидоров	Сидор	Сидорович	111111182	sidorov@mail.com
6	7	Кузнецов	Кузьма	Кузьмич	111931111	kuznetsov@mail.com
7	8	Алексеев	Алексей	Алексеевич	111116112	alekseev@mail.com
8	9	Смирнов	Семен	Семенович	112351111	smirnov@mail.com
9	10	Николаев	Николай	Николаевич	111436543	nikolaev@mail.com
10	11	Морозов	Михаил	Михайлович	119721111	morozov@mail.com
11	12	Васильев	Василий	Васильевич	111890112	vasiliev@mail.com
12	15	Кадникова	Юлия	Владимировна	200340837	katekatekd@mail.ru

Вывод

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