Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

ЛАБОРАТОРНАЯ РАБОТА №3

ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В PostgreSQL

по дисциплине:
«Проектирование и Реализация Баз Данных»

Выполнила:
студентка II курса ИКТ
группы К3241
Кормановская Д.

Санкт-Петербург
2022

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

Задачи:

1. 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.

Индивидуальное задание. Вариант 15.

Описание предметной области: БД образовательной организации содержит сведения об аудиториях и расписании проводимых в них занятий. Занятия проводятся на разных площадках. Время начала и окончания занятия по дням недели фиксировано. База данных используется для получения справок о наличии свободных аудиторий в указанное время, о месте и времени проведения определенных занятий.

БД должна содержать следующий минимальный набор сведений:

- Номер аудитории.
- Количество мест.
- Тип аудитории.
- Название площадки.
- Адрес площадки.
- Код дисциплины.
- Название дисциплины.
- Вид занятия.
- ФИО преподавателя.
- Должность преподавателя.
- Номер студенческой группы.
- Учебный год.
- Код направления.
- Название направления.
- Код подразделения.
- Название подразделения.
- Максимально возможное количество студентов для посещения занятия.
- Дата.
- День недели.
- Время начала занятия.
- Время окончания занятия.

Задание 4. Создать хранимые процедуры:

- Вывести список свободных аудиторий для проведения практических занятий заданной группы в заданное время.
- Вывести расписание занятий для заданного преподавателя.
- Вывести список аудиторий, в которых может разместиться заданная группа.

Задание 5. Создать необходимые триггеры.

Выполнение

Хранимые процедуры

Вывести список свободных аудиторий для проведения практических занятий заданной группы в заданное время.

```
CREATE OR REPLACE FUNCTION free_audience
(sday date, shour time, code character varying, syear integer)
RETURNS TABLE(id int, num character varying)
AS $$
     SELECT a.audience_id, a.audience_number
     FROM timetable."group" g
     JOIN timetable."class" c
```

```
        ON c.group_id = g.group_id
        JOIN timetable.audience a
        ON a.audience_id = c.audience_id
        WHERE g.group_numberofstudents <= a.audience_capacity
        AND g.group_yearstart = syear AND g.group_code LIKE code
        AND a.audience_id NOT IN
        (
             SELECT c.audience_id
             FROM timetable."class" c
             WHERE c.class_date = sday
             AND shour <= c.class_end AND shour >= c.class_start
        )
$$
LANGUAGE SQL;
```

```
DB_LAB1=# CREATE OR REPLACE FUNCTION free_audience
DB_LAB1-# (sday date, shour time, code character varying, syear integer)
DB_LAB1-# RETURNS TABLE(id int, num character varying)
DB_LAB1-# AS $$
DB_LAB1$# SELECT a.audience_id, a.audience_number
DB_LAB1$# FROM timetable."group" g
DB_LAB1$# JOIN timetable."class" c
DB_LAB1$# ON c.group_id = g.group_id
DB_LAB1$# JOIN timetable.audience a
DB_LAB1$# ON a.audience_id = c.audience_id
DB_LAB1$# WHERE g.group_numberofstudents <= a.audience_capacity
DB_LAB1$# AND g.group_yearstart = syear AND g.group_code LIKE code
DB_LAB1$# AND a.audience_id NOT IN
DB_LAB1$# (
DB_LAB1$# SELECT c.audience_id
DB_LAB1$# FROM timetable."class" c
DB_LAB1$# WHERE c.class_date = sday
DB_LAB1$# AND shour <= c.class_end AND shour >= c.class_start
DB_LAB1$# )
DB_LAB1$# $$
DB_LAB1-# LANGUAGE SQL;
CREATE FUNCTION
DB_LAB1=# SELECT *
DB_LAB1-# FROM free_audience('2022-04-11', '09:00:00', 'lklk43', 2019)
DB_LAB1-# ;
 id |  num
----+--------
  1 | 32-54
  2 | 93n234
  5 | 223
  5 | 223
  1 | 32-54
  2 | 93n234
(6 rows)
```

Вывести расписание занятий для заданного преподавателя.

```
CREATE OR REPLACE FUNCTION teacher_timetable(staffnumber
character varying)
```

```
RETURNS TABLE(gday date, gstart time, gend time, discname
character varying, classtypename character varying, code
character varying, audience character varying, office
character varying)
AS $$
     SELECT c.class_date, c.class_start, c.class_end,
     d.discipline_name, ct.classtype_name, g.group_code,
     a.audience_number, o.office_name
     FROM timetable."class" c
     JOIN timetable.teacher t
     ON t.teacher_id = c.teacher_id
     JOIN timetable.audience a
     ON a.audience_id = c.audience_id
     JOIN timetable."group" g
     ON g.group_id = c.group_id
     JOIN timetable.discipline d
     on d.discipline_id = c.discipline_id
     JOIN timetable.classtype ct
     ON ct.classtype_id = c.classtype_id
     JOIN timetable.office o
     on o.office_id = a.office_id
     WHERE t.teacher_staffnumber LIKE staffnumber
$$
LANGUAGE SQL;
```

```
DB_LAB1=# CREATE OR REPLACE FUNCTION teacher_timetable(staffnumber character varying)
DB_LAB1-# RETURNS TABLE(gday date, gstart time, gend time, discname character varying,
DB_LAB1(#  classtypename character varying, code character varying,
DB_LAB1(#  audience character varying, office character varying)
DB_LAB1-# AS $$
DB_LAB1$# SELECT c.class_date, c.class_start, c.class_end,
DB_LAB1$# d.discipline_name, ct.classtype_name, g.group_code,
DB_LAB1$# a.audience_number, o.office_name
DB_LAB1$# FROM timetable."class" c
DB_LAB1$# JOIN timetable.teacher t
DB_LAB1$# ON t.teacher_id = c.teacher_id
DB_LAB1$# JOIN timetable.audience a
DB_LAB1$# ON a.audience_id = c.audience_id
DB_LAB1$# JOIN timetable."group" g
DB_LAB1$# ON g.group_id = c.group_id
DB_LAB1$# JOIN timetable.discipline d
DB_LAB1$# on d.discipline_id = c.discipline_id
DB_LAB1$# JOIN timetable.classtype ct
DB_LAB1$# ON ct.classtype_id = c.classtype_id
DB_LAB1$# JOIN timetable.office o
DB_LAB1$# on o.office_id = a.office_id
DB_LAB1$# WHERE t.teacher_staffnumber LIKE staffnumber
DB_LAB1$# $$
DB_LAB1-# LANGUAGE SQL;
CREATE FUNCTION
```

Вывести список аудиторий, в которых может разместиться заданная группа.

```
CREATE OR REPLACE FUNCTION audience_for_group(code character
varying, gyear integer)
RETURNS TABLE(an character varying, ac integer, ofn character
varying)
AS $$
```

```
     SELECT       a.audience_number,       a.audience_capacity,
o.office_name
     FROM timetable.audience a
     JOIN timetable.office o
     ON o.office_id = a.office_id
     WHERE a.audience_capacity >= (
          SELECT g.group_numberofstudents
          FROM timetable."group" g
          WHERE g.group_code LIKE code
          AND g.group_yearstart = gyear
     )
$$
LANGUAGE SQL;
```

```
DB_LAB1=# CREATE OR REPLACE FUNCTION audience_for_group(code character varying, gyear integer)
DB_LAB1-# RETURNS TABLE(an character varying, ac integer, ofn character varying)
DB_LAB1-# AS $$
DB_LAB1$# SELECT a.audience_number, a.audience_capacity, o.office_name
DB_LAB1$# FROM timetable.audience a
DB_LAB1$# JOIN timetable.office o
DB_LAB1$# ON o.office_id = a.office_id
DB_LAB1$# WHERE a.audience_capacity >= (
DB_LAB1$# SELECT g.group_numberofstudents
DB_LAB1$# FROM timetable."group" g
DB_LAB1$# WHERE g.group_code LIKE code
DB_LAB1$# AND g.group_yearstart = gyear
DB_LAB1$# )
DB_LAB1$# $$
DB_LAB1-# LANGUAGE SQL;
CREATE FUNCTION
```

Триггеры

Логгирование `INSERT, DELETE, UPDATE` для таблицы преподавателей.
Функция для триггера:

```
CREATE OR REPLACE FUNCTION add_to_log() RETURNS TRIGGER AS $$
DECLARE
    mstr varchar(30);
    namestr character varying;
     surnamestr character varying;
    retstr varchar(254);
BEGIN
    IF TG_OP = 'INSERT' THEN
        namestr = NEW.teacher_name;
          surnamestr = NEW.teacher_surname;
        mstr := 'Add new teacher ';
        retstr := mstr||surnamestr||namestr;
             INSERT  INTO  timetable.logs(logs_text,  logs_time)
values (retstr, NOW());
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        namestr = NEW.teacher_name;
          surnamestr = NEW.teacher_surname;
        mstr := 'Update user ';
        retstr := mstr||surnamestr||namestr;
             INSERT  INTO  timetable.logs(logs_text,  logs_time)
values (retstr, NOW());
        RETURN NEW;
```

```
    ELSIF TG_OP = 'DELETE' THEN
        namestr = OLD.teacher_name;
          surnamestr = OLD.teacher_surname;
        mstr := 'Remove user ';
        retstr := mstr||surnamestr||namestr;
             INSERT INTO timetable.logs(logs_text, logs_time)
values (retstr, NOW());
        RETURN OLD;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Создание триггера:

```
CREATE TRIGGER t_teacher AFTER INSERT OR UPDATE OR DELETE ON
timetable.teacher FOR EACH ROW EXECUTE PROCEDURE add_to_log();
```

```
DB_LAB1=# CREATE OR REPLACE FUNCTION add_to_log() RETURNS TRIGGER AS $$
DB_LAB1$# DECLARE
DB_LAB1$#     mstr varchar(30);
DB_LAB1$#     namestr character varying;
DB_LAB1$# surnamestr character varying;
DB_LAB1$#     retstr varchar(254);
DB_LAB1$# BEGIN
DB_LAB1$#     IF TG_OP = 'INSERT' THEN
DB_LAB1$#         namestr = NEW.teacher_name;
DB_LAB1$# surnamestr = NEW.teacher_surname;
DB_LAB1$#         mstr := 'Add new teacher ';
DB_LAB1$#         retstr := mstr||surnamestr||namestr;
DB_LAB1$#         INSERT INTO logs(logs_text, logs_time) values (retstr, NOW());
DB_LAB1$#         RETURN NEW;
DB_LAB1$#     ELSIF TG_OP = 'UPDATE' THEN
DB_LAB1$#         namestr = NEW.teacher_name;
DB_LAB1$# surnamestr = NEW.teacher_surname;
DB_LAB1$#         mstr := 'Update user ';
DB_LAB1$#         retstr := mstr||surnamestr||namestr;
DB_LAB1$#         INSERT INTO logs(logs_text, logs_time) values (retstr, NOW());
DB_LAB1$#         RETURN NEW;
DB_LAB1$#     ELSIF TG_OP = 'DELETE' THEN
DB_LAB1$#         namestr = OLD.teacher_name;
DB_LAB1$# surnamestr = OLD.teacher_surname;
DB_LAB1$#         mstr := 'Remove user ';
DB_LAB1$#         retstr := mstr||surnamestr||namestr;
DB_LAB1$#         INSERT INTO logs(logs_text, logs_time) values (retstr, NOW());
DB_LAB1$#         RETURN OLD;
DB_LAB1$#     END IF;
DB_LAB1$# END;
DB_LAB1$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
DB_LAB1=# CREATE TRIGGER t_teacher AFTER INSERT OR UPDATE OR DELETE ON timetable.teacher FOR EACH ROW EXECUTE PROCEDURE
add_to_log();
CREATE TRIGGER
```

Далее после различных манипуляций с данными получаем следующие логи:

	logs_id [PK] bigint	logs_text character varying (500)	logs_time timestamp without time zone
1	1	Add new teacher КофейкинЧай	2022-05-05 20:08:10.545625
2	2	Update user КуропаткинаМария	2022-05-05 20:08:44.111168
3	3	Update user КофейкинЧай	2022-05-05 20:08:44.111168
4	4	Add new teacher ыаываывываы	2022-05-05 20:09:20.336082
5	5	Remove user ыаываывываы	2022-05-05 20:11:19.474547

Выводы

Созданы хранимые процедуры и триггеры в SQL Shell.