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»

Выполнил:
студент II курса ФИКТ
группы К3240
Кобелев Л.К.

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_auditorium
(givendate date, givenclassnum integer, givengroup character
varying, givengroupyear character)
RETURNS TABLE(id int, auditorium_num character varying,
building character varying, address character varying)
AS $$
```

```
SELECT a.auditorium_id, a.auditorium_num, b.building_name,
b.building_address
FROM timetable."group" g
JOIN timetable.class_conducting cc
ON cc.group_id = g.group_id
JOIN timetable.auditorium a
ON a.auditorium_id = cc.auditorium_id
JOIN timetable.building b
ON b.building_id = a.building_id
AND g.group_year LIKE givengroupyear AND g.group_num LIKE
givengroup
AND a.auditorium_id NOT IN
(
        SELECT cc.auditorium_id
        FROM timetable.class_conducting cc
        WHERE cc.class_conducting_date = givendate
        AND cc.class_conducting_class_id = givenclassnum
)
$$
LANGUAGE SQL;
```

```
LabWork_S4_L1=# CREATE OR REPLACE FUNCTION free_audience
LabWork_S4_L1-# (givendate date, givenclassnum integer, givengroup character varying, givengroupyear character)
LabWork_S4_L1-# RETURNS TABLE(id int, auditorium_num character varying, building character varying, address character varying)
LabWork_S4_L1-# AS $$
LabWork_S4_L1$# SELECT a.auditorium_id, a.auditorium_num, b.building_name, b.building_address
LabWork_S4_L1$# FROM timetable."group" g
LabWork_S4_L1$# JOIN timetable.class_conducting cc
LabWork_S4_L1$# ON cc.group_id = g.group_id
LabWork_S4_L1$# JOIN timetable.auditorium a
LabWork_S4_L1$# ON a.auditorium_id = cc.auditorium_id
LabWork_S4_L1$# JOIN timetable.building b
LabWork_S4_L1$# ON b.building_id = a.building_id
LabWork_S4_L1$# AND g.group_year LIKE givengroupyear AND g.group_num LIKE givengroup
LabWork_S4_L1$# AND a.auditorium_id NOT IN
LabWork_S4_L1$# (
LabWork_S4_L1$# SELECT cc.auditorium_id
LabWork_S4_L1$# FROM timetable.class_conducting cc
LabWork_S4_L1$# WHERE cc.class_conducting_date = givendate
LabWork_S4_L1$# AND cc.class_conducting_class_id = givenclassnum
LabWork_S4_L1$# )
LabWork_S4_L1$# $$
LabWork_S4_L1-# LANGUAGE SQL;
CREATE FUNCTION
```

для проверки: `SELECT * FROM free_audience('2022-04-18', 3, 'M1130', '2022');`

Вывести расписание занятий для заданного преподавателя.

```
CREATE OR REPLACE FUNCTION educator_timetable(fio character
varying)
RETURNS TABLE(
        classdate date, weekday character varying, tbegin time
without time zone,
        tend time without time zone, groupnum character
varying,
        auditorium character varying, address character
varying
)
AS $$
        SELECT cc.class_conducting_date,
cc.class_conducting_weekday, c.class_time_begin,
c.class_time_end,
```

```
        g.group_num, a.auditorium_num, b.building_address
        FROM timetable.class_conducting cc
        JOIN timetable.educator e
        ON e.educator_id = cc.educator_id
        JOIN timetable.auditorium a
        ON a.auditorium_id = cc.auditorium_id
        JOIN timetable.building b
        ON b.building_id = a.building_id
        JOIN timetable."class" c
        ON c.class_id = cc.class_conducting_class_id
        JOIN timetable."group" g
        ON g.group_id = cc.group_id
        WHERE e.educator_fullname LIKE fio
$$
LANGUAGE SQL;
```

```
LabWork_S4_L1=# CREATE OR REPLACE FUNCTION educator_timetable(fio character varying)
LabWork_S4_L1-# RETURNS TABLE(
LabWork_S4_L1(# classdate date, weekday character varying, tbegin time without time zone,
LabWork_S4_L1(# tend time without time zone, groupnum character varying,
LabWork_S4_L1(# auditorium character varying, address character varying
LabWork_S4_L1(# )
LabWork_S4_L1-# AS $$
LabWork_S4_L1$# SELECT cc.class_conducting_date, cc.class_conducting_weekday, c.class_time_begin, c.class_time_end,
LabWork_S4_L1$# g.group_num, a.auditorium_num, b.building_address
LabWork_S4_L1$# FROM timetable.class_conducting cc
LabWork_S4_L1$# JOIN timetable.educator e
LabWork_S4_L1$# ON e.educator_id = cc.educator_id
LabWork_S4_L1$# JOIN timetable.auditorium a
LabWork_S4_L1$# ON a.auditorium_id = cc.auditorium_id
LabWork_S4_L1$# JOIN timetable.building b
LabWork_S4_L1$# ON b.building_id = a.building_id
LabWork_S4_L1$# JOIN timetable."class" c
LabWork_S4_L1$# ON c.class_id = cc.class_conducting_class_id
LabWork_S4_L1$# JOIN timetable."group" g
LabWork_S4_L1$# ON g.group_id = cc.group_id
LabWork_S4_L1$# WHERE e.educator_fullname LIKE fio
LabWork_S4_L1$# $$
LabWork_S4_L1-# LANGUAGE SQL;
CREATE FUNCTION
```

для проверки: `SELECT * FROM educator_timetable('Эйлер Леонард');`

Вывести список аудиторий, в которых может разместиться заданная группа.

```
CREATE OR REPLACE FUNCTION suitable_audience(groupcode character varying, groupyear character varying)
RETURNS TABLE(anum character varying, acapacity integer, bname character varying)
AS $$
        SELECT a.auditorium_num, a.auditorium_capacity, b.building_name
        FROM timetable.auditorium a
        JOIN timetable.building b
        ON b.building_id = a.building_id
        WHERE a.auditorium_capacity >= (
                SELECT g.group_students_count
                FROM timetable."group" g
                WHERE g.group_num LIKE groupcode
                AND g.group_year = groupyear
        )
$$
LANGUAGE SQL;
```

```
LabWork_S4_L1=# CREATE OR REPLACE FUNCTION suitable_audience(groupcode character varying, groupyear character varying)
LabWork_S4_L1-# RETURNS TABLE(anum character varying, acapacity integer, bname character varying)
LabWork_S4_L1-# AS $$
LabWork_S4_L1$# SELECT a.auditorium_num, a.auditorium_capacity, b.building_name
LabWork_S4_L1$# FROM timetable.auditorium a
LabWork_S4_L1$# JOIN timetable.building b
LabWork_S4_L1$# ON b.building_id = a.building_id
LabWork_S4_L1$# WHERE a.auditorium_capacity >= (
LabWork_S4_L1$# SELECT g.group_students_count
LabWork_S4_L1$# FROM timetable."group" g
LabWork_S4_L1$# WHERE g.group_num LIKE groupcode
LabWork_S4_L1$# AND g.group_year = groupyear
LabWork_S4_L1$# )
LabWork_S4_L1$# $$
LabWork_S4_L1-# LANGUAGE SQL;
CREATE FUNCTION
```

для проверки: `SELECT * FROM suitable_audience('M1130', '2022');`

Триггеры

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

```
CREATE OR REPLACE FUNCTION add_to_log() RETURNS TRIGGER AS $$
DECLARE
    mstr varchar(30);
    groupnum character varying;
        groupyear character varying;
    retstr varchar(254);
BEGIN
    IF TG_OP = 'INSERT' THEN
        groupnum = NEW.group_num;
                groupyear = NEW.group_year;
        mstr := 'Add new group ';
        retstr := mstr||groupnum||' '||groupyear;
             INSERT  INTO  timetable.logs(logs_text,  logs_time)
values (retstr, NOW());
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        groupnum = NEW.group_num;
                groupyear = NEW.group_year;
        mstr := 'Update group ';
        retstr := mstr||groupnum||' '||groupyear;
             INSERT  INTO  timetable.logs(logs_text,  logs_time)
values (retstr, NOW());
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
                groupnum = OLD.group_num;
                groupyear = OLD.group_year;
        mstr := 'Remove group ';
        retstr := mstr||groupnum||' '||groupyear;
             INSERT  INTO  timetable.logs(logs_text,  logs_time)
values (retstr, NOW());
        RETURN OLD;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Создание триггера:

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

```
LabWork_S4_L1=# CREATE OR REPLACE FUNCTION add_to_log() RETURNS TRIGGER AS $$
LabWork_S4_L1$# DECLARE
LabWork_S4_L1$#     mstr varchar(30);
LabWork_S4_L1$#     groupnum character varying;
LabWork_S4_L1$# groupyear character varying;
LabWork_S4_L1$#     retstr varchar(254);
LabWork_S4_L1$# BEGIN
LabWork_S4_L1$#     IF TG_OP = 'INSERT' THEN
LabWork_S4_L1$#         groupnum = NEW.group_num;
LabWork_S4_L1$# groupyear = NEW.group_year;
LabWork_S4_L1$#         mstr := 'Add new group ';
LabWork_S4_L1$#         retstr := mstr||groupnum||' '||groupyear;
LabWork_S4_L1$#         INSERT INTO timetable.logs(logs_text, logs_time) values (retstr, NOW());
LabWork_S4_L1$#         RETURN NEW;
LabWork_S4_L1$#     ELSIF TG_OP = 'UPDATE' THEN
LabWork_S4_L1$#         groupnum = NEW.group_num;
LabWork_S4_L1$# groupyear = NEW.group_year;
LabWork_S4_L1$#         mstr := 'Update group ';
LabWork_S4_L1$#         retstr := mstr||groupnum||' '||groupyear;
LabWork_S4_L1$#         INSERT INTO timetable.logs(logs_text, logs_time) values (retstr, NOW());
LabWork_S4_L1$#         RETURN NEW;
LabWork_S4_L1$#     ELSIF TG_OP = 'DELETE' THEN
LabWork_S4_L1$# groupnum = OLD.group_num;
LabWork_S4_L1$# groupyear = OLD.group_year;
LabWork_S4_L1$#         mstr := 'Remove group ';
LabWork_S4_L1$#         retstr := mstr||groupnum||' '||groupyear;
LabWork_S4_L1$#         INSERT INTO timetable.logs(logs_text, logs_time) values (retstr, NOW());
LabWork_S4_L1$#         RETURN OLD;
LabWork_S4_L1$#     END IF;
LabWork_S4_L1$# END;
LabWork_S4_L1$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
LabWork_S4_L1=# CREATE TRIGGER t_group AFTER INSERT OR UPDATE OR DELETE ON timetable."group" FOR EACH ROW EXECUTE PRO
URE add_to_log();
CREATE TRIGGER
```

Далее после различных манипуляций с данными получаем следующие логи:

```
LabWork_S4_L1=# SELECT * FROM timetable.logs
LabWork_S4_L1-# ;
 logs_id |        logs_text         |    logs_time
---------+--------------------------+-----------------
       1 | Add new group N1130 2021 | 22:26:51.479799
       2 | Add new group P3110 2029 | 22:26:51.479799
       3 | Update group P1120 2029  | 22:27:14.984429
       4 | Update group P1120 2020  | 22:27:38.259092
       5 | Remove group N1130 2021  | 22:28:41.094719
(5 rows)
```

Выводы

Созданы хранимые процедуры и триггеры в SQL Shell.