

Санкт-Петербургский национальный исследовательский университет ИТМО

Факультет Инфокоммуникационных технологий

**Лабораторная работа №3 по теме**

**«процедуры, функции, триггеры в PostgreSQL»**

**по дисциплине «Проектирование и реализация баз данных»**

Выполнил:

студент 2 курса К32421 группы

Козлов Всеволод Денисович

Преподаватель:

Говорова Марина Михайловна

Санкт-Петербург

# ЛАБОРАТОРНАЯ РАБОТА №3

## ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В PostgreSQL

**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. 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.

## Методы

### № . 1

**Описание:**

Исходное задание: метод для повышении стипендии отличникам на 10%.

Моя модификация: У меня нет отдельной стипендии для отличников, поэтому повышу социальную стипендию на 10%

**Код метода:**

```
create or replace function increase_social_scholarship()
returns void as
$$
declare
    prev_amount int;
    scholarship_name text;
begin
    scholarship_name := 'социальная имя';
    select money_amount into prev_amount
```

```

from scholarships
where name=scholarship_name;
update scholarships
set
    money_amount = prev_amount * 1.1
where
    name = scholarship_name;
end;
$$ language plpgsql;

```

## Выполнение метода:

До:

	id_scholarship	name	money_amount	type_of_scholarship
1	0	социальная имя	30000.00	социальная
2	1	академическая имя	2000.00	академическая
3	2	именная имя	100000.90	именная

После:

	id_scholarship	name	money_amount	type_of_scholarship
1	1	академическая имя	2000.00	академическая
2	2	именная имя	100000.90	именная
3	0	социальная имя	33000.00	социальная

## № . 2

### Описание:

Исходное задание: метод для перевода студента на следующий курс

Моя модификация: так как у меня нет курса студента, а есть только история групп студента, то вместо перевода на курс я реализую добавления новой группы в историю групп, что аналогично повышению курса студента, ведь каждая группа привязана к курсу

### Код метода:

```

create or replace function
    plus_course(id_record_book_inp int, id_new_group_inp int)
returns void as

```

```

$$
declare
    current_course int;
    new_group_course int;
begin
    select max(course) into current_course from groups
    where id_group in
            (select id_group from student_groups where
id_record_book=id_record_book_inp);

    select course into new_group_course from groups
    where id_group = id_new_group_inp;



    if new_group_course <> current_course+1 then
        raise exception 'new course of group is not larger than
current course of group by one';
    end if;

    insert into student_groups(id_record_book, id_group)
    values (id_record_book_inp, id_new_group_inp);
end;
$$ language plpgsql;

```

## Выполнение метода:

Выведем историю групп студента 20. Он сейчас находится на 1-м курсе в группе 4:

	 id_group_row ▾	 id_record_book ▾	 id_group ▾
1	40	20	4

Попытаемся добавить в группу 0 с курсом 1:

```
select plus_course(20, 0)
```

Получаем ошибку:

[2023-05-06 12:26:54] [P0001] ОШИБКА: new course of group is not larger than current course of group by one

[2023-05-06 12:26:54] Где: функция PL/pgSQL plus\_course(integer,integer), строка 14, оператор RAISE

Попытаемся добавить в группу 5 с курсом 2:

```
select plus_course(20, 5);
```

Посмотрим на результат:

	id_group_row	id_record_book	id_group
1	10042	20	5
2	40	20	4

## № 3

### Описание:

Метод для изменения оценки при успешной пересдаче экзамена

### Код метода:

```
create or replace function change_mark(input_id_study_course int,
input_id_group_row int, new_mark int) returns void as
$$
    DECLARE
        last_try_number int;
        var_id_teacher int;
    begin
        select max(try_number) into last_try_number from record_book
        where
            id_study_course = input_id_study_course and
            id_group_row = input_id_group_row;

        select id_teacher into var_id_teacher from record_book
        where
            id_study_course = input_id_study_course and
            id_group_row = input_id_group_row and
            try_number=last_try_number;

        if last_try_number = 3 then
            raise exception 'Невозможно пересдать экзамен, так как
были использованы все попытки';
        end if;

        insert into record_book(id_group_row, id_teacher,
id_study_course, mark, try_number)
            values (input_id_group_row, var_id_teacher,
input_id_study_course, new_mark, last_try_number+1);
    end;
$$ language plpgsql;
```

## Выполнение метода:

Посмотрим оценки для `id_group_row = 0 and id_study_course = 1`

	<code>id_row_record_book</code>	<code>id_group_row</code>	<code>id_teacher</code>	<code>id_study_course</code>	<code>mark</code>	<code>try_number</code>
1	1	0	1	1	4	2
2	401	0	1	1	2	1

Выполним метод:

```
select change_mark(2, 0, 5);
```

Результат:

	<code>id_row_record_book</code>	<code>id_group_row</code>	<code>id_teacher</code>	<code>id_study_course</code>	<code>mark</code>	<code>try_number</code>
1	1	0	1	1	4	2
2	819	0	1	1	5	3
3	401	0	1	1	2	1

Посмотрим оценки для `id_group_row = 0 and id_study_course = 2`.

Результат:

	<code>id_row_record_book</code>	<code>id_group_row</code>	<code>id_teacher</code>	<code>id_study_course</code>	<code>mark</code>	<code>try_number</code>
1	2	0	2	2	3	3
2	402	0	2	2	2	1
3	403	0	2	2	2	2

Видим, что у студента использованы все попытки => ему нельзя пересдать предмета

Выполним метод:

```
select change_mark(2, 0, 5);
```

Получили ошибку:

[2023-05-06 15:49:24] [P0001] ERROR: Невозможно пересдать экзамен, так как были использованы все попытки

[2023-05-06 15:49:24] Где: PL/pgSQL function change\_mark(integer,integer,integer) line 18 at RAISE

## №4. Бонус

Описание:

У меня почему-то сбились последовательности для primary key. Я решил написать метод, который устанавливает значение последовательностей на максимальное значение PK + 1.

Код метода:

```
CREATE OR REPLACE FUNCTION reset_serial_sequences() RETURNS void AS
$$
DECLARE
    name_of_table text;
    name_of_column text;
    sequence_name text;
    max_value bigint;
```

```

BEGIN
    FOR name_of_table, name_of_column IN
        SELECT table_name, column_name FROM information_schema.columns
        WHERE column_default LIKE 'nextval%'
    LOOP
        sequence_name := pg_get_serial_sequence(name_of_table,
name_of_column);

        if name_of_column is not null then
            EXECUTE format('SELECT max(%I) FROM %I', name_of_column,
name_of_table) INTO max_value;

            EXECUTE format('SELECT setval(%L, %s)', sequence_name,
max_value + 1);
        end if;

    END LOOP;
END;
$$ LANGUAGE plpgsql;

```

## Триггеры

### №1

#### Описание:

Триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL

#### Код:

```

-- Создание таблицы для логов
create table subdivision_logs(
    id_log serial primary key,
    operation_type cud_operation not null,
    operation_time timestamp without time zone,
    affected_id_subdivision int null,
    affected_subdivision_name text null
);
-- Создание метода для триггера
CREATE OR REPLACE FUNCTION subdivision_add_to_log() RETURNS TRIGGER
AS $$

```

```

DECLARE
    var_operation_type cud_operation;
BEGIN
    IF TG_OP = 'INSERT' THEN
        var_operation_type := 'insert';
        insert into
            subdivision_logs(operation_type, operation_time,
affected_id_subdivision, affected_subdivision_name)
        VALUES
            (var_operation_type, now(), new.id_subdivision, new.name);
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        var_operation_type := 'update';
        insert into
            subdivision_logs(operation_type, operation_time,
affected_id_subdivision, affected_subdivision_name)
        VALUES
            (var_operation_type, now(), old.id_subdivision, old.name);
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
        var_operation_type := 'delete';
        insert into
            subdivision_logs(operation_type, operation_time,
affected_id_subdivision, affected_subdivision_name)
        VALUES
            (var_operation_type, now(), old.id_subdivision, old.name);
        RETURN OLD;
    END IF;
END
$$ LANGUAGE plpgsql;
-- Создание триггера
create trigger log_trigger
    after insert or update or delete on subdivisions
    for each row execute procedure subdivision_add_to_log();

```

## Выполнение:

### Выполнение insert:

	id_log	operation_type	operation_time	affected_id_subdivision	affected_subdivision_name
1	1	insert	2023-05-06 13:29:18.406493	11	new subdivision

### Выполнение update:

	id_log	operation_type	operation_time	affected_id_subdivision	affected_subdivision_name
1	1	insert	2023-05-06 13:29:18.406493		11 new subdivision
2	2	update	2023-05-06 13:30:22.449229	9	fancy name
3	3	update	2023-05-06 13:31:02.105800	9	fancy name
4	4	update	2023-05-06 13:31:35.469823	9	name 9



## Выполнение delete:

	id_log	operation_type	operation_time	affected_id_subdivision	affected_subdivision_name
1	1	insert	2023-05-06 13:29:18.406493	11	new subdivision
2	2	update	2023-05-06 13:30:22.449229	9	fancy name
3	3	update	2023-05-06 13:31:02.105800	9	fancy name
4	4	update	2023-05-06 13:31:35.469823	9	name 9
5	5	delete	2023-05-06 13:32:23.247818	11	new subdivision

## №2

### Описание:

Триггер, который предотвращает вставку оценки, если дисциплина отсутствует в учебном плане группы.

### Код:

```
-- Метод для триггера
create or replace function check_study_course() returns trigger as
$$
declare
    var_id_study_courses int[];
begin
    select array_agg(id_study_course) into var_id_study_courses from
study_courses
    where id_study_plan =
        (select id_study_plan from groups where id_group =
            (select id_group from student_groups where
id_group_row=new.id_group_row));

    if new.id_study_course <> any(var_id_study_courses) then
        raise EXCEPTION 'Нет такого учебного курса в учебном плане
группы';
    end if;

    return new;
end
$$ language plpgsql;
-- Создание триггера
create trigger study_course_trigger
before insert on record_book
for each row execute procedure check_study_course();
```

### Выполнение:

Попытаемся поставить оценку по дисциплине, которой нет у студента в учебном плане:

```
insert into record_book(id_group_row, id_teacher, id_study_course, mark, try_number)
VALUES (0, 10, 10, 5, 1);
```

Получаем ошибку:

[2023-05-06 17:18:10] [P0001] ERROR: Нет такого учебного курса в учебном плане группы

[2023-05-06 17:18:10] Где: PL/pgSQL function check\_study\_course() line 11 at RAISE

## Выводы

В процессе выполнения данной лабораторной работы удалось овладеть навыками написания и использования процедур, функций и триггеров в PSQL.