

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе «Процедуры, функции, триггеры в PostgreSQL»

по дисциплине «**Базы данных**»

Автор: Ким Даниил Дмитриевич

Факультет: ИКТ

Группа: K32391

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

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

Оборудование: компьютерный класс.

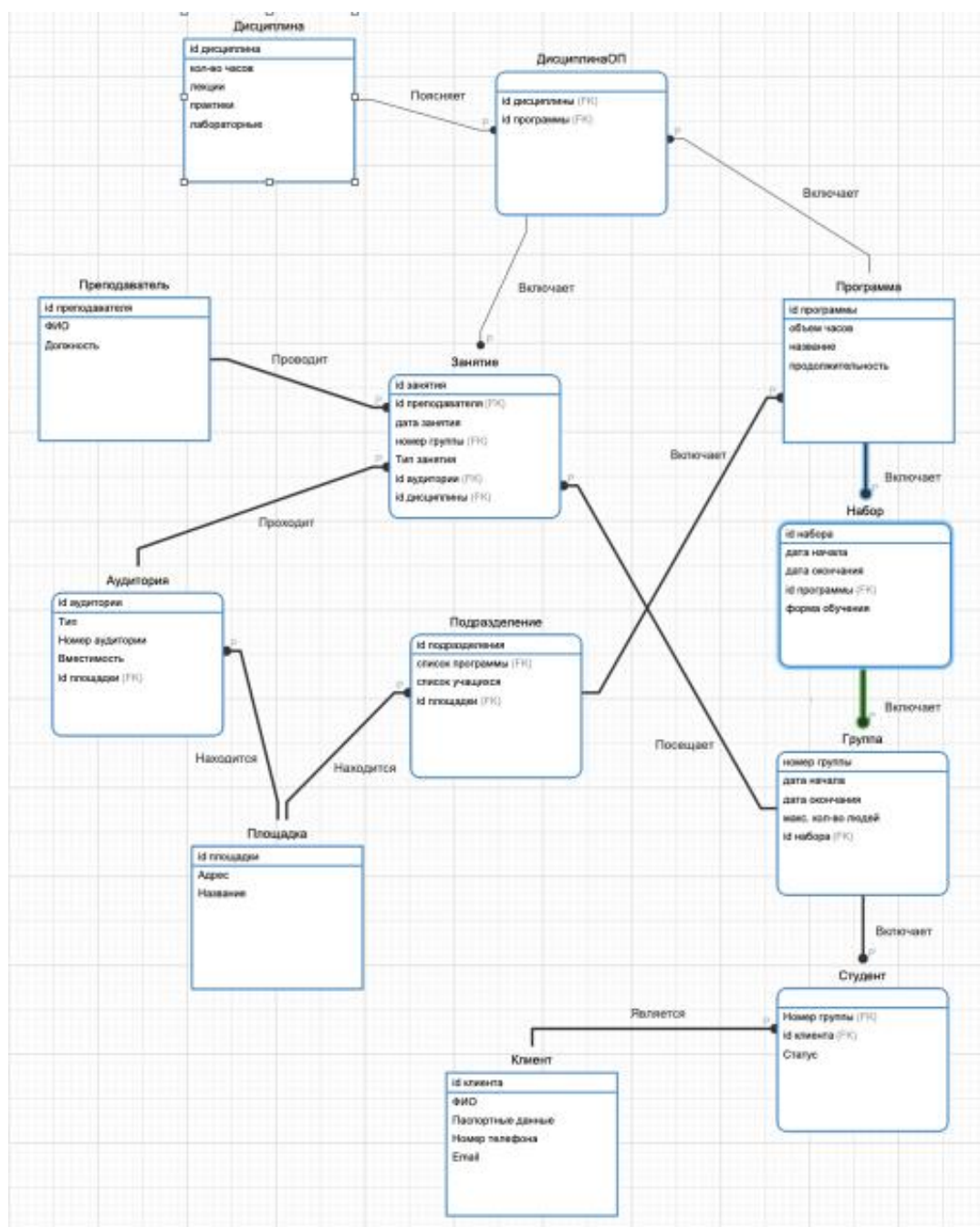
Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:

Вариант 1

- скрипты кода разработанных объектов (процедур/функций и триггера на логирование действий) и подтверждающие скриншоты работы и результатов в psql согласно индивидуальному заданию (часть 4 и 5).

Ход работы:



Создать хранимые процедуры:

- Для получения расписания занятий для групп на определенный день недели.

Query Query History

```
1 CREATE OR REPLACE FUNCTION lesson.GetScheduleForGroupByDayOfWeek2(  
2     p_group_number VARCHAR(9),  
3     p_day_of_week INTEGER  
4 )  
5 RETURNS TABLE (lesson_date DATE, lesson_type VARCHAR, audience_number VARCHAR)  
6 AS $$  
7 BEGIN  
8     RAISE NOTICE 'Schedule for group % on %:', p_group_number, to_char(p_day_of_week, 'FMDay');  
9  
10    RETURN QUERY  
11    SELECT l.lesson_date, l.lesson_type, CAST(a.audience_number AS VARCHAR)  
12    FROM lesson._group g  
13    JOIN lesson.lesson l ON g.group_number = l.group_number  
14    JOIN lesson.audience a ON l.id_audience = a.id_audience  
15    WHERE g.group_number = p_group_number  
16           AND EXTRACT(ISODOW FROM l.lesson_date) = p_day_of_week;  
17 END;  
18 $$  
19 LANGUAGE plpgsql;  
20
```

Data Output Messages Notifications

CREATE FUNCTION









Query returned successfully in 71 msec.

Вывод:

Query Query History

```
1 SELECT * FROM lesson.GetScheduleForGroupByDayOfWeek2('K32391', 5);  
2
```

Data Output Messages Notifications



	lesson_date date	lesson_type character varying	audience_number character varying
1	2023-03-24	Лекция	302

- Записи на курс слушателя.

Query	Query History
1	CREATE OR REPLACE PROCEDURE lesson.EnrollStudentOnProgram(2 p_student_id INTEGER , 3 p_client_id INTEGER , 4 p_program_id INTEGER 5) 6 LANGUAGE plpgsql 7 AS \$\$ 8 ▼ BEGIN 9 10 INSERT INTO lesson.student (id_student, id_client, group_number, student_status) 11 VALUES (p_student_id, p_client_id, 'K32391', 'Зачислен'); 12 13 INSERT INTO lesson.division (id_division, id_place, id_program, students_list) 14 VALUES (1, 1, p_program_id, ARRAY [p_student_id]); 15 16 RAISE NOTICE 'Студент успешно записан на программу.'; 17 END ; 18 \$\$; 19
<div>Data Output <u>Messages</u> Notifications</div> <div>CREATE PROCEDURE</div> <div>Query returned successfully in 34 msec.</div>	

Вывод:

Query	Query History
1	CALL lesson.EnrollStudentOnProgram(1, 1, 1);
<div>Data Output <u>Messages</u> Notifications</div> <div>NOTICE: Студент успешно записан на программу. CALL</div> <div>Query returned successfully in 35 msec.</div>	

- Получения перечня свободных лекционных аудиторий на любой день недели. Если свободных аудиторий не имеется, то выдать соответствующее сообщение.

Query









Query History

```
1 CREATE OR REPLACE FUNCTION lesson.get_free_lecture_auditoriums(day_of_week integer)
2 RETURNS TABLE (audience_number integer, capacity integer)
3 AS $$
4 BEGIN
5     RETURN QUERY
6     SELECT lesson.audience.audience_number, lesson.audience.capacity
7     FROM lesson.audience
8     WHERE lesson.audience._type = 'Лекция'
9     AND lesson.audience.id_audience NOT IN (
10         SELECT lesson.lesson.id_audience
11         FROM lesson.lesson
12         WHERE extract(dow from lesson.lesson_date) = day_of_week
13     );
14
15 IF NOT FOUND THEN
16     RAISE NOTICE 'Свободные лекционные аудитории на указанный день недели отсутствуют.';
17 END IF;
18
19 RETURN;
20 END;
21 $$ LANGUAGE plpgsql;
22
23 SELECT * FROM lesson.get_free_lecture_auditoriums(2);
24
```

Data Output

Messages

Notifications



	audience_number integer	capacity integer
1	201	50

Триггер для логирования событий вставки, удаления и обновления данных в таблице

Вывод

```
Query Query History
1  -- Создание таблицы для лога событий
2  CREATE TABLE IF NOT EXISTS lesson.event_log (
3      id SERIAL PRIMARY KEY,
4      event_type VARCHAR(20) NOT NULL,
5      event_timestamp TIMESTAMPTZ DEFAULT NOW(),
6      table_name VARCHAR(50) NOT NULL,
7      old_data JSONB,
8      new_data JSONB
9  );
10
11 -- Триггер для логирования событий вставки, удаления и обновления данных
12 CREATE OR REPLACE FUNCTION lesson.log_event()
13 RETURNS TRIGGER AS $$
14 BEGIN
15     IF (TG_OP = 'INSERT') THEN
16         INSERT INTO lesson.event_log (event_type, table_name, new_data)
17         VALUES ('INSERT', TG_TABLE_NAME, row_to_json(NEW));
18     ELSIF (TG_OP = 'DELETE') THEN
19         INSERT INTO lesson.event_log (event_type, table_name, old_data)
20         VALUES ('DELETE', TG_TABLE_NAME, row_to_json(OLD));
21     ELSIF (TG_OP = 'UPDATE') THEN
22         INSERT INTO lesson.event_log (event_type, table_name, old_data, new_data)
23         VALUES ('UPDATE', TG_TABLE_NAME, row_to_json(OLD), row_to_json(NEW));
24     END IF;
25
26     RETURN NEW;
27 END;
28 $$ LANGUAGE plpgsql;
29
30 -- Привязка триггера к таблице
31 CREATE TRIGGER log_event_trigger
32 AFTER INSERT OR UPDATE OR DELETE ON lesson.student
33 FOR EACH ROW EXECUTE FUNCTION lesson.log_event();
34
```

Data Output Messages Notifications

CREATE TRIGGER

Проверим работу:

QueryQuery History

1INSERT INTO lesson.student (id_student, id_client, group_number, student_status)

2VALUES (22, 1, 'K32391', 'Зачислен');

3

4

5UPDATE lesson.student

6SET student_status = 'Отчислен'

7WHERE id_student = 1;

8

9

10DELETE FROM lesson.student

11WHERE id_student = 1;

12

13

14

15SELECT * FROM lesson.event_log;

16

Data OutputMessagesNotifications

	id [PK] integer	event_type character varying (20)	event_timestamp timestamp with time zone	table_name character varying (50)	old_data jsonb
1	1	INSERT	2023-05-25 02:56:49.415324+03	student	[null]
2	2	UPDATE	2023-05-25 02:56:49.415324+03	student	{"id_client": 1, "id_student": 1, "group_number": "K32391", "student_status": "Зачислен"}
3	3	DELETE	2023-05-25 02:56:49.415324+03	student	{"id_client": 1, "id_student": 1, "group_number": "K32391", "student_status": "Отчислен"}

old_data jsonb	new_data jsonb
[null]	{"id_client": 1, "id_student": 22, "group_number": "K32391", "student_status": "Зачислен"}
{"id_client": 1, "id_student": 1, "group_number": "K32391", "student_status": "Зачислен"}	{"id_client": 1, "id_student": 1, "group_number": "K32391", "student_status": "Отчислен"}
{"id_client": 1, "id_student": 1, "group_number": "K32391", "student_status": "Отчислен"}	[null]

В ходе лабораторной работы я научился создавать и использовать процедуры, функции и триггеры в базе данных PostgreSQL. Также, я понял, что функции и процедуры в SQL недостаточно гибкие, как в ЯП.