

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: Никитин.П

Факультет: ИКТ

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.

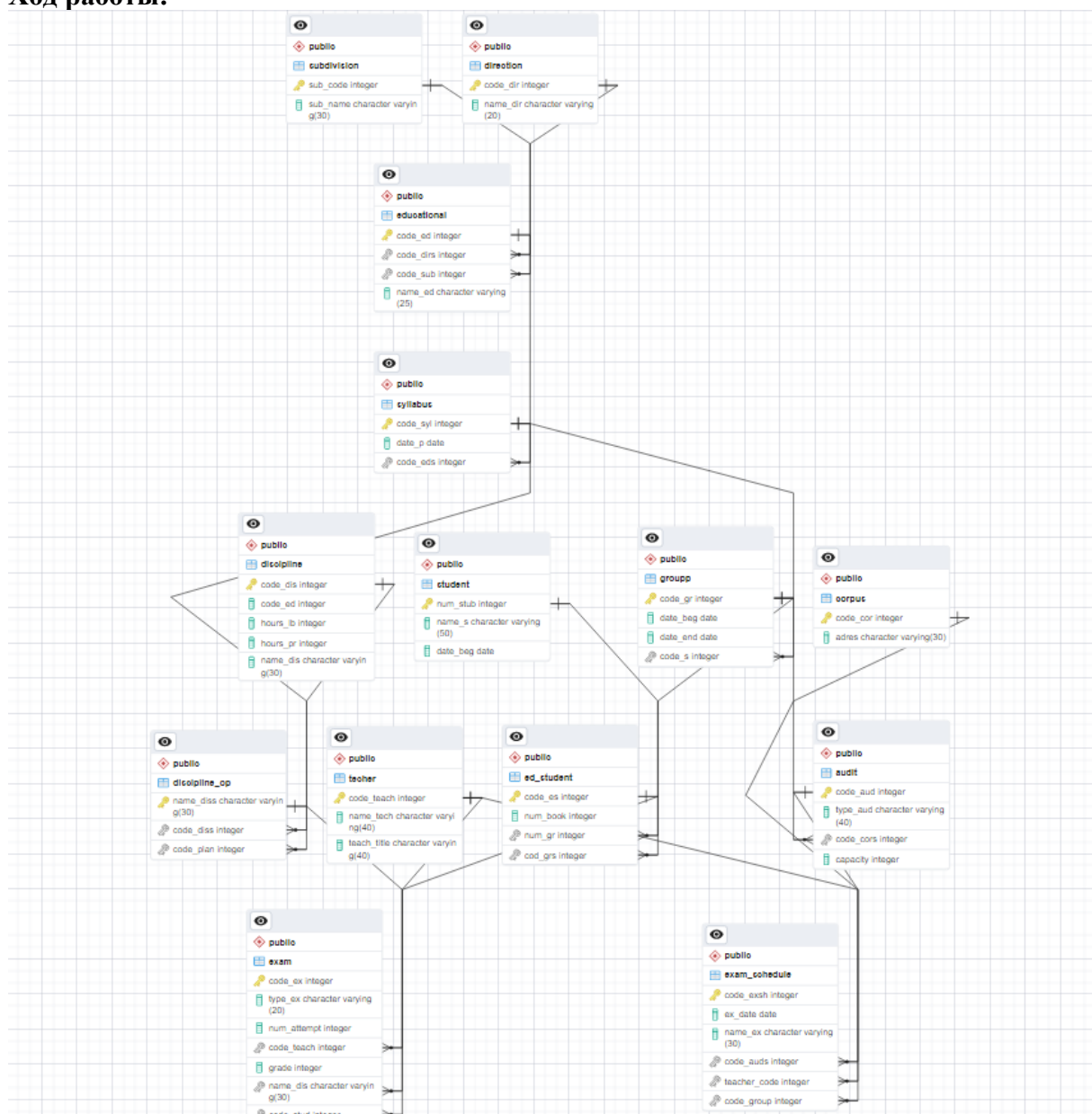
Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

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 повышения стипендии отличникам на 10%.

```
create or replace function IncreaseScholarship2()
returns table (student_id INT, scholarrrship DECIMAL) as $$
declare
    cur_result cursor for
        select code_es, ed_student.scholarship * 1.1 from ed_student where code_es in (select code_stud from
            (select code_stud, avg (grade) as grd from exam group by code_stud) as d
            where grd = 5);
begin
    update ed_student
    set scholarship = ed_student.scholarship * 1.1
    where code_es in (
        select code_es from ed_student where code_es in (select code_stud from
            (select code_stud, avg (grade) as grd from exam group by code_stud) as d
            where grd = 5
            ));
    open cur_result;
    return query fetch all from cur_result;
    close cur_result;
end;
$$ language plpgsql;
```

Вывод:

```
1 select * from IncreaseScholarship2();|
```

Data Output Messages Notifications



	student_id integer	scholarrrship numeric
1	1	2420.0
2	3	2420.0

- Для перевода студентов на следующий курс.

```
create or replace function moveStudentsToNextCourse4()
returns table (student_id INT, cours int) as $$
declare
    cur_result cursor for
        select code_es, ed_student.cours + 1 from ed_student where not (code_es in (select code_stud from exam where grade <= 2))
begin
    update ed_student
    set cours = ed_student.cours + 1
    where code_es in (
        select code_es from ed_student where not (code_es in (select code_stud from exam where grade <= 2)));
    open cur_result;
    return query fetch all from cur_result;
    close cur_result;
end;
$$ language plpgsql;
```

Вывод:

```
1 select * from moveStudentsToNextCourse4()
```

Data Output Messages Notifications

	student_id integer	cours integer
1	2	6
2	1	6
3	3	6

- Для изменения оценки при успешной пересдаче экзамена.

```
CREATE OR REPLACE PROCEDURE UpdateExamGrade(
    ex_id INT,
    studentID INT,
    newGrade INT
)
LANGUAGE plpgsql
AS $$
DECLARE
    currentGrade INT;
BEGIN
    -- Получаем текущую оценку студента
    SELECT grade INTO currentGrade
    FROM exam
    WHERE code_stud = studentID and code_ex=ex_id;

    -- Проверяем, если новая оценка выше текущей
    IF newGrade > currentGrade THEN
        -- Обновляем оценку студента
        UPDATE exam
        SET grade = newGrade
        WHERE code_stud = studentID and code_ex=ex_id;

        RAISE NOTICE 'Оценка студента успешно изменена.';
    ELSE
        RAISE NOTICE 'Новая оценка должна быть выше текущей оценки.';
    END IF;
END;
$$;

1 CALL UpdateExamGrade(2,2, 4);
```

Data Output Сообщения Notifications

ЗАМЕЧАНИЕ: Оценка студента успешно изменена.
CALL

Запрос завершён успешно, время выполнения: 46 msec.

	code_ex [PK] integer	type_ex character varying (20)	num_attempt integer	code_teach integer	grade integer	name_dis character varying (30)	code_stud integer
1	1	Мат	1	1	5	Математика	2
2	2	Мат	1	1	4	Математика	2

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

Вывод

```
CREATE OR REPLACE FUNCTION LogTriggerFunction()
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'INSERT') THEN
        INSERT INTO LogTable (TableName, ActionType)
        VALUES (TG_TABLE_NAME, 'Insert');
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO LogTable (TableName, ActionType)
        VALUES (TG_TABLE_NAME, 'Update');
    ELSIF (TG_OP = 'DELETE') THEN
        INSERT INTO LogTable (TableName, ActionType)
        VALUES (TG_TABLE_NAME, 'Delete');
    END IF;












    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Привязка триггера к таблице
CREATE TRIGGER LogTrigger
AFTER INSERT OR UPDATE OR DELETE ON student
FOR EACH ROW
EXECUTE FUNCTION LogTriggerFunction();
```

Проверим работу:

```
1 insert into student
2 values (22 , 'Ron', '10-01-2023');
```

Data Output Сообщения Notifications

       				
	logid [PK] integer 	tablename character varying (100) 	actiontype character varying (50) 	actiondate timestamp without time zone 
1	1	student	Insert	2023-05-27 15:54:24.916548

```

create or replace function add_to_log()
returns trigger as $$
declare
mstr varchar(30);
astr varchar(100);
retstr varchar(254);
begin
if TG_OP = 'INSERT' then
astr = NEW;
mstr := 'Add data ';
retstr := mstr||astr;
INSERT INTO logs(text, added, table_name) values
(retstr, NOW(), TG_TABLE_NAME);
return new;
ELSIF TG_OP = 'UPDATE' THEN
astr = NEW;
mstr := 'Update data ';
retstr := mstr||astr;
INSERT INTO logs(text, added, table_name) values
(retstr, NOW(), TG_TABLE_NAME);
RETURN NEW;
ELSIF TG_OP = 'DELETE' THEN
astr = OLD;
mstr := 'Remove data ';
retstr := mstr || astr;
INSERT INTO logs(text, added, table_name) values
(retstr, NOW(), TG_TABLE_NAME);
return old;
end if;
end;
$$ language plpgsql;

```

Query Query History

```

1 create trigger t_client after insert or update or delete on
2 ed_student for each row execute procedure
3 add_to_log ();
4
5

```

Query Query History

```

1 SELECT * FROM public.logs
2

```

Data Output Messages Notifications



	text character varying	table_name character varying	added character varying
1	Update data (2,2,1,121,1210,6)	ed_student	2023-06-05 02:3...
2	Update data (1,2,1,121,2200,6)	ed_student	2023-06-05 02:3...
3	Update data (3,2,1,121,2200,6)	ed_student	2023-06-05 02:3...

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 гиб

