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.

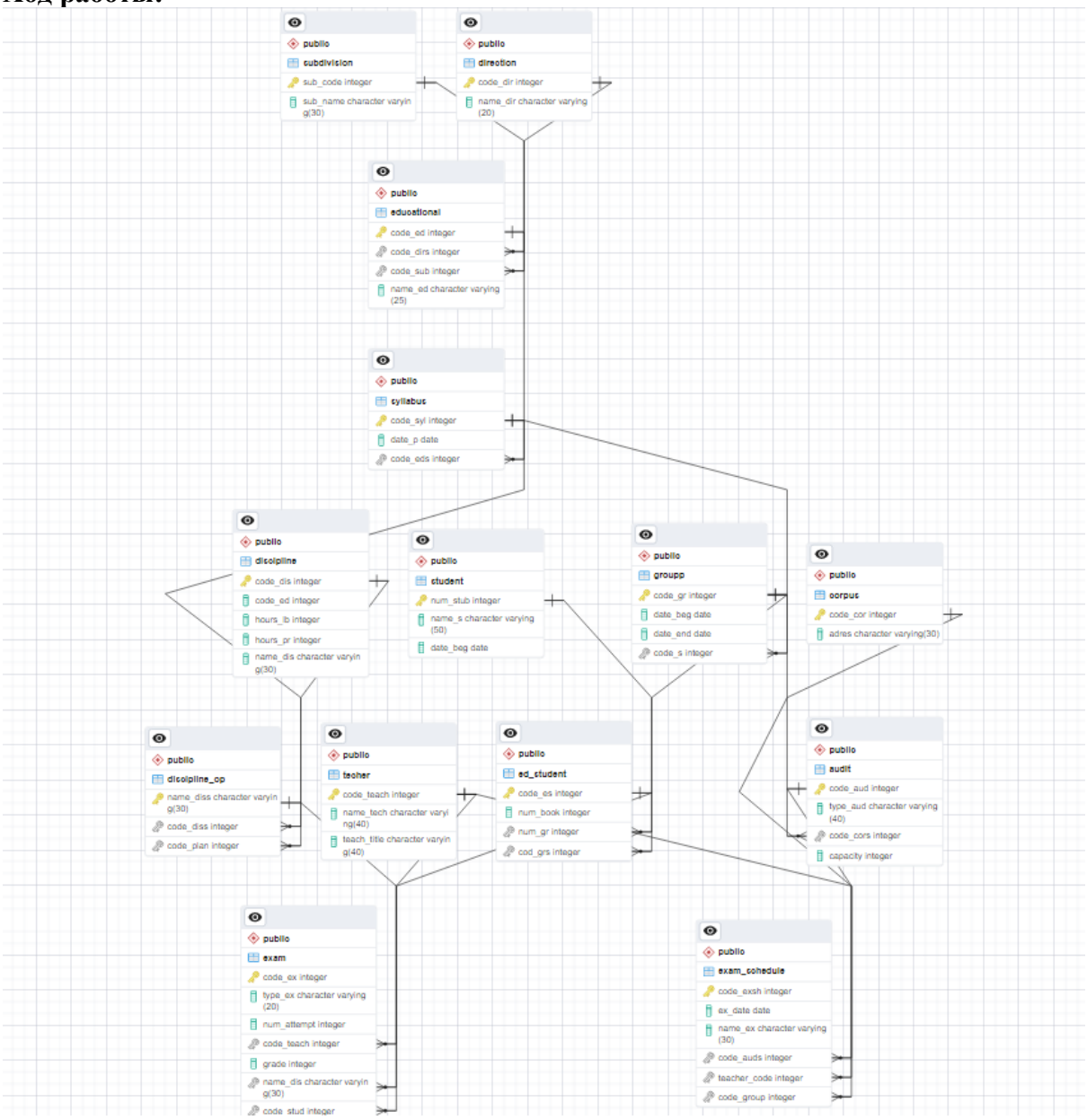
Оборудование: компьютерный класс.

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 IncreaseScholarship()
RETURNS TABLE (student_id INT, scholarship 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 exam
            WHERE grade = 4
        );
BEGIN
    -- Обновление стипендии для отличников
    UPDATE ed_student
    SET scholarship = ed_student.scholarship * 1.1
    WHERE code_es IN (
        SELECT code_stud
        FROM exam
        WHERE grade = 4
    );

    -- Открытие курсора для получения результатов
    OPEN cur_result;

    -- Возврат результатов из курсора
    RETURN QUERY FETCH ALL FROM cur_result;

    -- Закрытие курсора
    CLOSE cur_result;

END;
$$ LANGUAGE plpgsql;
```

Вывод:

```
1  SELECT * FROM IncreaseScholarship();
```

Data Output | Сообщения | Notifications

	student_id integer	scholarship numeric
1	2	1331.0

- Для перевода студентов на следующий курс.

```
CREATE OR REPLACE FUNCTION MoveStudentsToNextCourse()
RETURNS TABLE (student_id INT, cours int) AS $$
DECLARE
    cur_result CURSOR FOR
        SELECT code_es, ed_student.cours
        FROM ed_student
        WHERE code_es IN (
            SELECT code_stud
            FROM exam
            WHERE grade > 2
        );
BEGIN
    -- Обновление стипендии для отличников
    UPDATE ed_student
    SET cours = ed_student.cours + 1
    WHERE code_es IN (
        SELECT code_stud
        FROM exam
        WHERE grade > 2
    );

    -- Открытие курсора для получения результатов
    OPEN cur_result;

    -- Возврат результатов из курсора
    RETURN QUERY FETCH ALL FROM cur_result;

    -- Закрытие курсора
    CLOSE cur_result;

END;
$$ LANGUAGE plpgsql;
```

Вывод:

```
1  SELECT * FROM MoveStudentsToNextCourse()
```

Data Output | Сообщения | Notifications

	student_id integer	cours integer
1	2	3

- 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

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.