Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

ПСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информатики и вычислительной техники

Кафедра вычислительной техники

ПРЕДМЕТ «Базы Данных»

# *КУРСОВОЙ ПРОЕКТ*

***«Вариант 6»***

РАЗРАБОТКА БАЗЫ ДАННЫХ В СУБД POSTGRESQL

Выполнил студент группы 0024-01

Невзоров Л.Е.

Проверил преподаватель

Вертешев А.С.

Псков

2022

**Задание**

Тема: учет успеваемости студентов в текущей сессии.

Объекты: факультеты, группы, студенты, экзамены, дисциплины, преподаватели.

Для моделирования задачи необходимо хранить следующую информацию:

\* Номер группы, в которой учится студент +

\* Факультет, к которому принадлежит группа

\* ФИО студента

\* Номер зачетной книжки студента

\* Наименование дисциплины, которую изучает группа студентов

\* ФИО лектора дисциплины

\* Оценка, полученная студентом за экзамен по каждой дисциплине, изучаемой

группой

\* Дата сдачи экзамена студентом

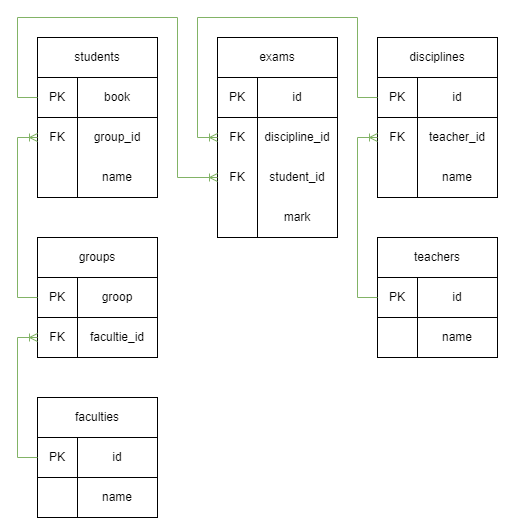
Дополнительные условия:

\* если студент пересдавал экзамен, то в таблице экзаменов может быть несколько

записей;

\* один лектор может читать несколько дисциплин.

**Схема**



**Создание таблиц**

CREATE TABLE "faculties" (

  id SERIAL PRIMARY KEY,

  name VARCHAR(50) UNIQUE NOT NULL

);

INSERT INTO faculties (name)

VALUES

  ('Institute of Engineering Sciences'),

  ('Institute of Mathematical Modeling and Igropractic'),

  ('Institute of Medicine and Experimental Biology');

CREATE TABLE "groups" (

  groop VARCHAR(7) PRIMARY KEY,

  facultie\_id INTEGER NOT NULL REFERENCES faculties(id) ON DELETE CASCADE

);

INSERT INTO groups (groop, facultie\_id)

VALUES

  ('0024-01', 1),

  ('0024-02', 1);

CREATE TABLE "students" (

  book VARCHAR(6) PRIMARY KEY,

  group\_id VARCHAR(7) NOT NULL REFERENCES groups(groop) ON DELETE CASCADE,

  name VARCHAR(50) NOT NULL

);

INSERT INTO students (book,group\_id,name)

VALUES

  ('667511','0024-01','Anastasia Warren'),

  ('717505','0024-01','Dante Lynn'),

  ('791997','0024-01','Yen Mcmillan'),

  ('418318','0024-01','Cameran Mathis'),

  ('351472','0024-01','Ivor Levy');

CREATE TABLE "exams" (

  id SERIAL PRIMARY KEY,

  discipline\_id INTEGER NOT NULL REFERENCES disciplines(id) ON DELETE CASCADE,

  student\_id VARCHAR(6) REFERENCES students(book) ON DELETE CASCADE,

  mark INTEGER DEFAULT NULL

);

CREATE TABLE "disciplines" (

  id SERIAL PRIMARY KEY,

  name VARCHAR(100) NOT NULL,

  teacher\_id INTEGER NOT NULL REFERENCES teachers(id) ON DELETE CASCADE

);

INSERT INTO disciplines (name,teacher\_id)

VALUES

  ('Programming in 1C',1),

  ('Database design',2),

  ('Web Programming',3),

  ('Circuit design',1),

  ('Modeling',4);

CREATE TABLE "teachers" (

  id SERIAL PRIMARY KEY,

  name VARCHAR(50) NOT NULL

);

INSERT INTO teachers (name)

VALUES

  ('Jada F. Barton'),

  ('Marsden G. Jacobson'),

  ('Zena U. Mejia'),

  ('Kyle S. Reid'),

  ('Stacy L. Hutchinson');

INSERT INTO exams (discipline\_id,student\_id,mark)

VALUES

  (1,'667511',4),

  (2,'667511',4),

  (3,'667511',5),

  (4,'667511',4);

**Запросы**

1)Три запроса на выборку с использованием join, использовать не мене 3 таблиц

SELECT st.name,

       gr.groop,

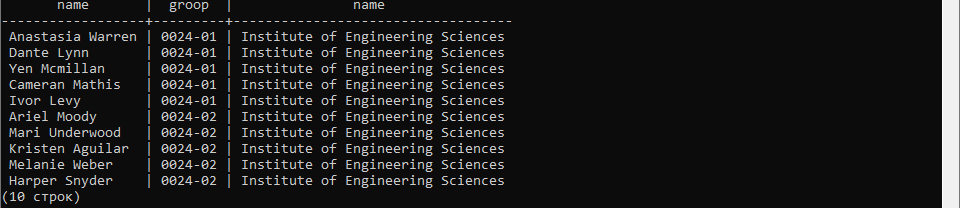
       fc.name

FROM students AS st

JOIN groups AS gr ON st.group\_id=gr.groop

JOIN faculties AS fc ON gr.facultie\_id=fc.id

LIMIT 10;



SELECT gr.groop,

       st.book,

       ex.mark

FROM exams AS ex

JOIN students AS st ON st.book=ex.student\_id

JOIN groups AS gr ON gr.groop=st.group\_id

LIMIT 10;



SELECT te.name,

       di.name,

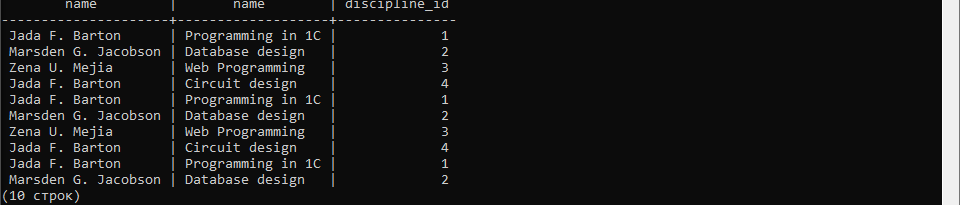
       ex.discipline\_id

FROM exams AS ex

JOIN disciplines AS di ON di.id=ex.discipline\_id

JOIN teachers AS te ON te.id=di.teacher\_id

LIMIT 10;



[2) Запросы на объединение, пересечение, разность, произведение, проекцию, деление](#_Toc91305837)

SELECT name,

       book

FROM students

WHERE name LIKE '%S%'

UNION

SELECT name,

       book

FROM students

WHERE name LIKE '%M%';



SELECT student\_id,

       mark

FROM exams

WHERE mark <= 5

  AND mark >= 3

INTERSECT

SELECT student\_id,

       mark

FROM exams

WHERE mark = 4;



SELECT student\_id,

       mark

FROM exams

EXCEPT

SELECT student\_id,

       mark

FROM exams

WHERE mark <= 3;



[3) Запросы на соединение: left join, right join, full outer join](#_Toc91305838)

SELECT gr.groop,

       st.name

FROM groups AS gr

LEFT OUTER JOIN students AS st ON gr.groop=st.group\_id

LIMIT 5;



SELECT gr.groop,

       st.name

FROM groups AS gr

RIGHT OUTER JOIN students AS st ON gr.groop=st.group\_id

LIMIT 5;



SELECT gr.groop,

       st.name

FROM groups AS gr

FULL OUTER JOIN students AS st ON gr.groop=st.group\_id

LIMIT 5;



4) 2 Подзапроса: Простой, Сложный

SELECT st.name,

       gr.groop

FROM groups AS gr

LEFT OUTER JOIN students AS st ON gr.groop=st.group\_id

WHERE st.name LIKE '%B%' OR st.name LIKE '%S%'

LIMIT 5;



SELECT st.name,

       ex.mark

FROM exams AS ex

LEFT OUTER JOIN students AS st ON st.book=ex.student\_id

WHERE ex.mark > (SELECT AVG(mark)

                 FROM exams

                 WHERE mark > (SELECT AVG(mark)

                              FROM exams))

LIMIT 10;



5) Запросы с различными уровнями вложенности подзапросов в частях SELECT, FROM, WHERE, HAVING

SELECT st.name,

       ex.mark

FROM exams AS ex

LEFT OUTER JOIN students AS st ON st.book=ex.student\_id

WHERE ex.mark > (SELECT AVG(mark)

                 FROM exams

                 WHERE mark > (SELECT AVG(mark)

                              FROM exams))

LIMIT 10;



SELECT tbl.name,

       tbl.mark

FROM (SELECT \*

      FROM students AS st

      JOIN exams AS ex ON st.book=ex.student\_id) AS tbl

WHERE tbl.mark > (SELECT AVG(mark)

                  FROM exams

                  WHERE mark > (SELECT AVG(mark)

                                FROM exams))

LIMIT 10;



SELECT st.name,

       AVG(ex.mark)

FROM students AS st

JOIN exams AS ex ON st.book=ex.student\_id

GROUP BY st.name

HAVING AVG(ex.mark) > (SELECT AVG(mark)

                       FROM exams

                       WHERE mark > (SELECT AVG(mark)

                                     FROM exams))

LIMIT 10;



[6) Создание ролей пользователей user, guest](#_Toc91305840)

DROP ROLE IF EXISTS test;

-- создание пользователя

CREATE ROLE test WITH LOGIN PASSWORD 'passworduser';

-- выдаем привелегии

GRANT ALL ON DATABASE grade TO test;

-- просмотр зарегистрированных пользователей

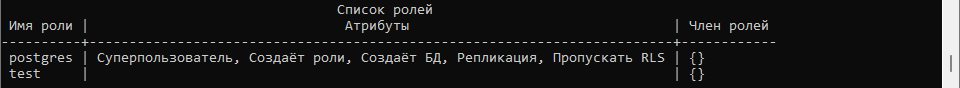
\du

-- забираем привелегии

REVOKE ALL ON DATABASE grade FROM test;

-- удаляем пользователя

DROP ROLE test;



DROP ROLE IF EXISTS guest;

-- создание пользователя

CREATE ROLE guest WITH LOGIN PASSWORD 'password';

-- выдаем привелегии

GRANT ALL ON students TO guest;

-- просмотр зарегистрированных пользователей

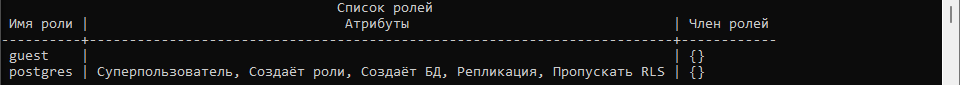
\du

-- забираем привелегии

REVOKE ALL ON students FROM guest;

-- удаляем пользователя

DROP ROLE guest;



Проверка, может ли GUEST (гость) зайти в недоступные ему таблицы в базе данных:





[7) Создание двух транзакций с тремя точками восстановления на удаление, создание транзакции на вставку](#_Toc91305841)

BEGIN;

-- транзакция 1

UPDATE exams SET mark = mark - 1

    WHERE student\_id = '667511';

SAVEPOINT my\_savepoint;

UPDATE exams SET mark = mark + 1

    WHERE student\_id = '717505';

SAVEPOINT my\_savepoint;

-- транзакция 2

UPDATE exams SET mark = mark - 1

    WHERE student\_id = '667511';

SAVEPOINT my\_savepoint;

UPDATE exams SET mark = mark + 1

    WHERE student\_id = '791997';

ROLLBACK TO my\_savepoint; -- откат на операцию 2.1 и отмена операции 2.2

-- корректировка транзакции 2

UPDATE exams SET mark = mark + 1

    WHERE student\_id = '717505';

COMMIT;

ДО:



ПОСЛЕ:



[8) Создание двух представлений: изменяемое, неизменяемое](#_Toc91305842)

-- создадим изменяемое представление

CREATE OR REPLACE VIEW students\_view AS

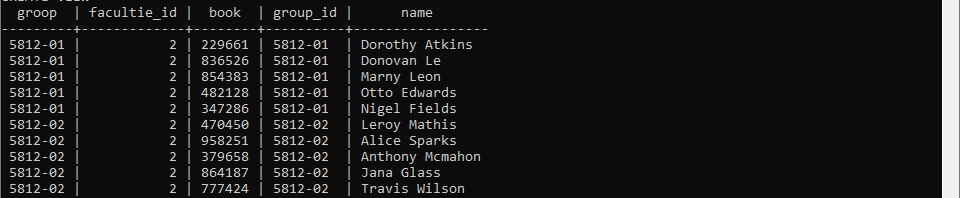
    SELECT \*

    FROM groups AS gr

    LEFT OUTER JOIN students AS st ON gr.groop=st.group\_id

    WHERE gr.groop = '5812-01'

       OR gr.groop = '5812-02';



-- изменим представление

CREATE OR REPLACE VIEW students\_view AS

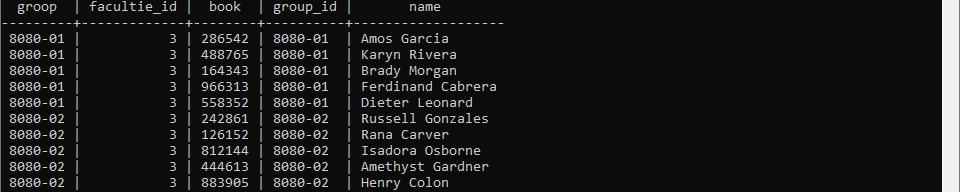
    SELECT \*

    FROM groups AS gr

    LEFT OUTER JOIN students AS st ON gr.groop=st.group\_id

    WHERE gr.groop = '8080-01'

       OR gr.groop = '8080-02';



-- создадим неизменяемое представление

CREATE VIEW students\_view AS

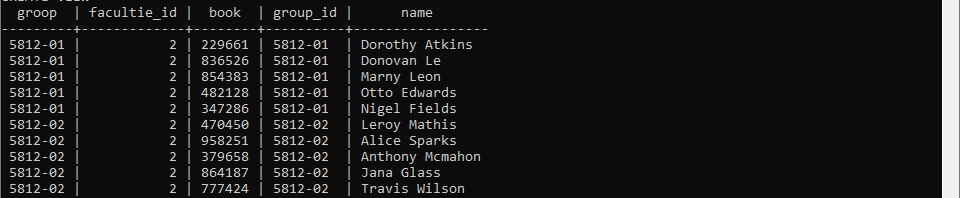
    SELECT \*

    FROM groups AS gr

    LEFT OUTER JOIN students AS st ON gr.groop=st.group\_id

    WHERE gr.groop = '5812-01'

       OR gr.groop = '5812-02';



-- попробуем его изменить

CREATE VIEW students\_view AS

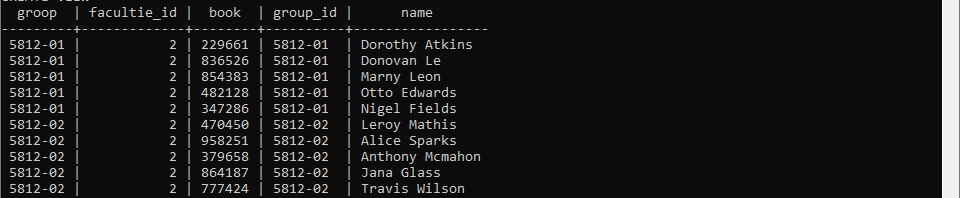
    SELECT \*

    FROM groups AS gr

    LEFT OUTER JOIN students AS st ON gr.groop=st.group\_id

    WHERE gr.groop = '8080-01'

       OR gr.groop = '8080-02';



[9) Создание функции](#_Toc91305843)

Посчитаем общую среднюю оценку для всех студентов:

CREATE FUNCTION mean(x float, y float) RETURNS INTEGER AS $$

     SELECT x / y;

$$ LANGUAGE SQL;

SELECT mean((SELECT SUM(mark) FROM exams),

            (SELECT COUNT(mark) FROM exams)) AS mean\_mark;



[10) Создание триггера](#_Toc91305844)

-- создание функции

CREATE FUNCTION exams\_stamp() RETURNS trigger AS $exams\_stamp$

    BEGIN

        IF NEW.discipline\_id IS NULL THEN

            RAISE EXCEPTION 'discipline\_id cannot be null';

        END IF;

        IF NEW.student\_id IS NULL THEN

            RAISE EXCEPTION '% student\_id cannot be null', NEW.student\_id;

        END IF;

        IF NEW.mark < 2 THEN

            RAISE EXCEPTION '% mark cannot be < 2', NEW.mark;

        END IF;

        RETURN NEW;

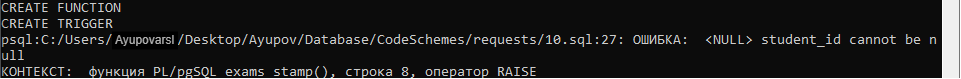
    END;

$exams\_stamp$ LANGUAGE plpgsql;

-- создание триггера

CREATE TRIGGER exams\_stamp BEFORE INSERT OR UPDATE ON exams

    FOR EACH ROW EXECUTE FUNCTION exams\_stamp();



[11). Создание процедуры](#_Toc91305845)

-- удалние через процедуру

CREATE PROCEDURE clean\_emp() AS '

    DELETE FROM exams

        WHERE mark < 3;

' LANGUAGE SQL;

CALL clean\_emp();



[12) Шифрование столбцов](#_Toc91305846)

CREATE EXTENSION IF NOT EXISTS pgcrypto;

CREATE TABLE exam\_table (

   id SERIAL PRIMARY KEY,

   discipline\_id VARCHAR(150) NOT NULL,

   student\_id VARCHAR(150) NOT NULL,

   mark INTEGER NOT NULL

);

INSERT INTO exam\_table (discipline\_id,student\_id,mark)

VALUES

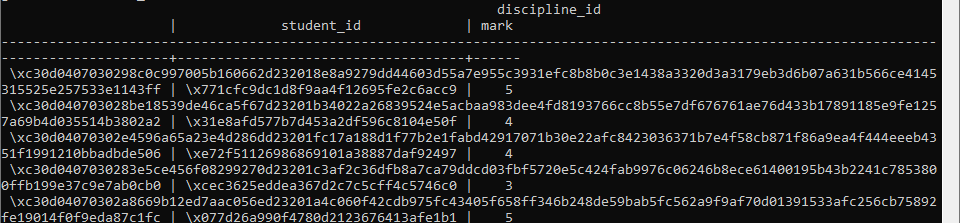
  (pgp\_sym\_encrypt('6','arSlan!PassWorD'), encrypt('667511','arSlan!PassWorD', 'aes'), 5),

  (pgp\_sym\_encrypt('1','arSlan!PassWorD'), encrypt('717505','arSlan!PassWorD', 'aes'), 4),

  (pgp\_sym\_encrypt('7','arSlan!PassWorD'), encrypt('791997','arSlan!PassWorD', 'aes'), 4),

  (pgp\_sym\_encrypt('3','arSlan!PassWorD'), encrypt('418318','arSlan!PassWorD', 'aes'), 3),

  (pgp\_sym\_encrypt('5','arSlan!PassWorD'), encrypt('351472','arSlan!PassWorD', 'aes'), 5);



13) Расшифровка столбцов

SELECT  pgp\_sym\_decrypt(discipline\_id::bytea, 'arSlan!PassWorD'),

        convert\_from(decrypt(student\_id::bytea, 'arSlan!PassWorD', 'aes'), 'SQL\_ASCII'),

        mark

FROM exam\_table



14) Создание объектных типов данных

CREATE TYPE estimation AS ENUM ('excellent', 'good', 'unsatisfactory');

CREATE TABLE estimation\_table (

    id SERIAL,

    name TEXT,

    est estimation

);

INSERT INTO estimation\_table (id, name, est)

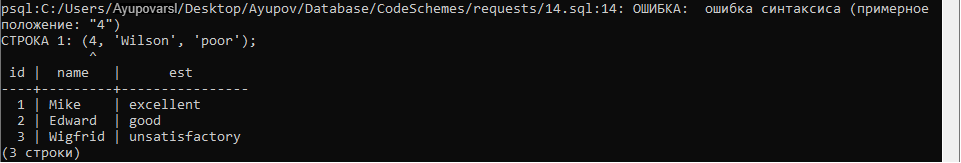
VALUES

    (1, 'Mike', 'excellent'),

    (2, 'Edward', 'good'),

    (3, 'Wigfrid', 'unsatisfactory');

    (4, 'Wilson', 'poor');



16) Запрос на перевод в формат JSON

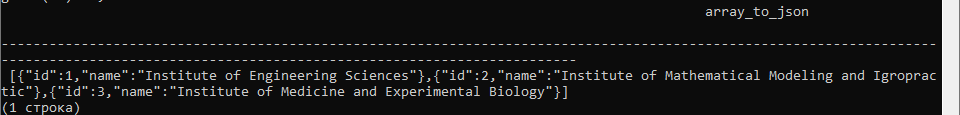
SELECT array\_to\_json(array\_agg(row\_to\_json (fc))) FROM (

    SELECT fc.id,

           fc.name

    FROM faculties AS fc

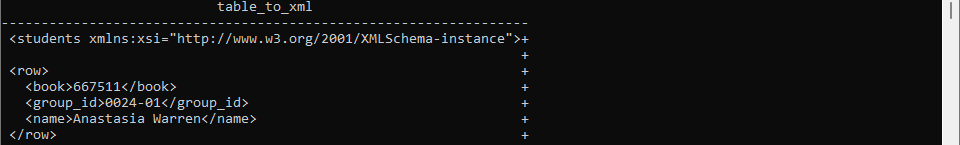
) fc;



17) Запрос на перевод в формат XML

-- таблица в XML

SELECT table\_to\_xml('students', true, false, '');



-- query запрос в XML

SELECT query\_to\_xml('SELECT \* FROM students', true, false, '');

