Андрей Козлов, гр. 4538

1. Инкрементально приведите данное отношение в 5 нормальную форму.

Запишем функциональные зависимости из предыдущего задания.

Student_Id → Student_Name (1)
Group_Id → Group_Name (2)
Course_Id → Course_Name (3)
Lecturer_Id → Lecturer_Name (4)
Student_Id → Group_Id (5)
Student_Id Course_Id → Mark (6)
Group_Id Course_Id → Lecturer_Id (7)

Данное отношение находится в 1НФ, так как в нем нет повторяющихся групп, все атрибуты атомарны, у него есть ключ ({Student_Id, Course_Id}).

Приведем данное отношение в 2НФ, для этого проведем декомпозицию, получив следующие отношения:

Student_Id		Student_Name		
Group_Id		Group_Name		
Course_Id		Course_Name		
Lecturer_Id		Lecturer_Name		
Student_Id	Course_Id	Group_Id	Lecturer_Id	Mark (*)

Докажем, что все они находятся в 2НФ, то есть находятся в 1НФ и неключевые атрибуты функционально зависят от ключа в целом.

> Док-во: 1-4. Очевидно.
> 5. Множество атрибутов {Student_Id, Course_Id} является ключом, из ФЗ (5), (6), (7) выводима ФЗ Student_Id Course_Id → Group_Id Lecturer_Id Mark, значит отношение находится в 2НФ по определению.

Так как неключевые атрибуты непосредственно зависят от ключа, данное отношение находится также и в 3НФ.

Проводя декомпозицию отношения (*) (по ФЗ (7)), получаем следующие отношения:

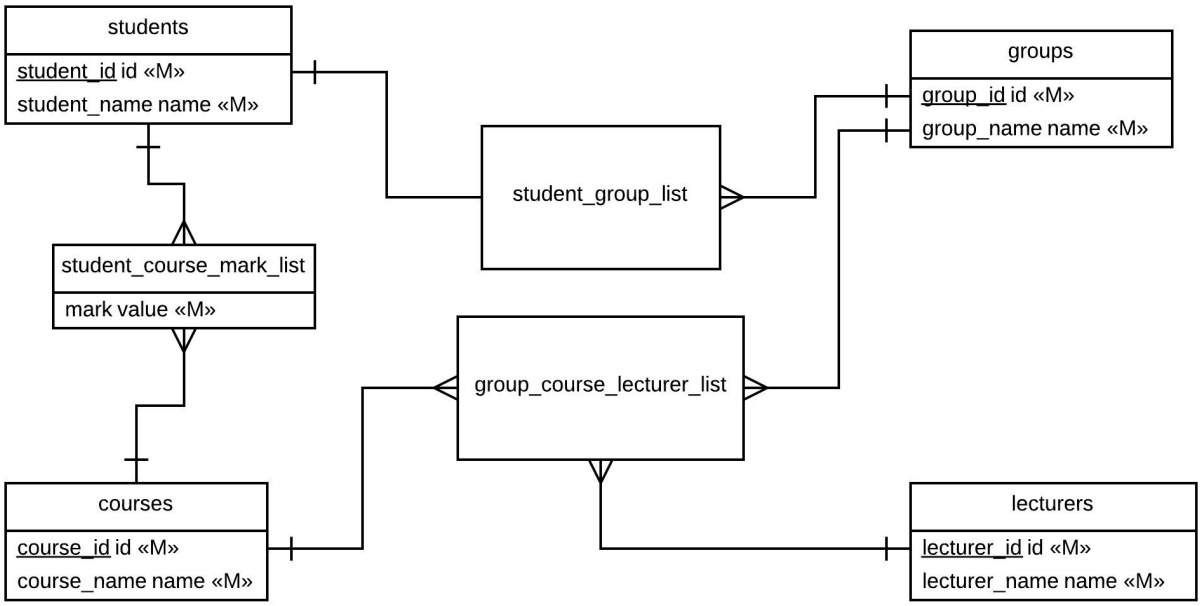
Group_Id		Course_Id	Lecturer_Id
Student_Id	Group_Id	Course_Id	Mark (**)

Далее проводя декомпозицию отношения (**) (по ФЗ (5)), получаем следующие отношения:

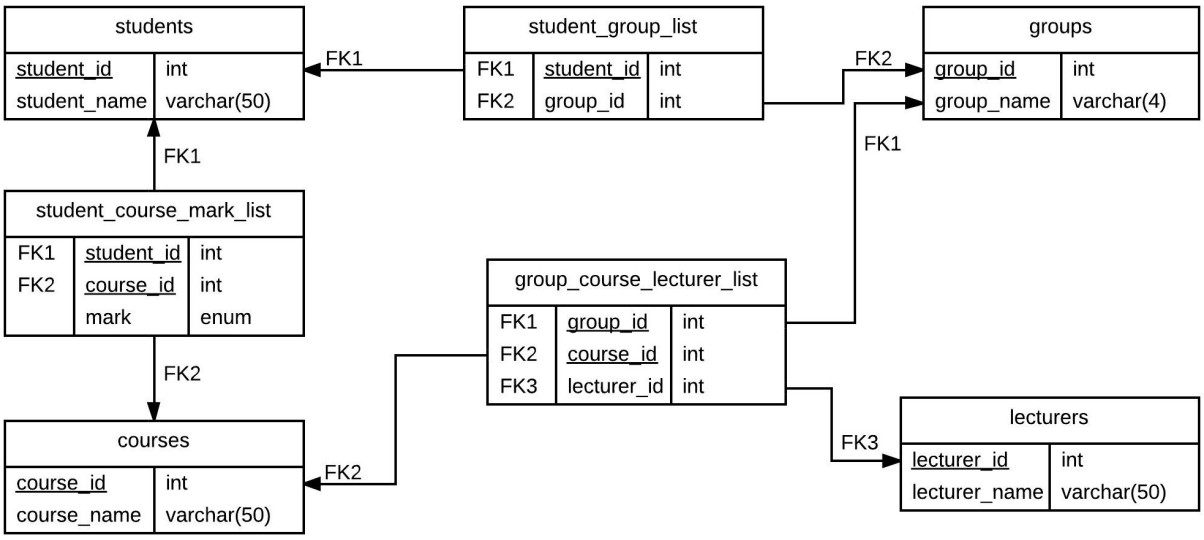
Student_Id		Group_Id
Student_Id	Course_Id	Mark

Полученные отношения находятся в 4НФ и 5НФ.

2. **Постройте соответствующую модель сущность-связь.**



3. **Постройте соответствующую физическую модель.**



4. **Реализуйте SQL-скрипты, создающие схему базы данных.**

```
DROP DATABASE IF EXISTS deanery;
CREATE DATABASE deanery;
USE deanery;

CREATE TABLE students (
      student_id INT NOT NULL AUTO_INCREMENT,
      student_name VARCHAR(50) NOT NULL,
      PRIMARY KEY(student_id)
);
```

```
CREATE TABLE groups (
      group_id INT NOT NULL AUTO_INCREMENT,
      group_name VARCHAR(4) NOT NULL,
      PRIMARY KEY(group_id)
);

CREATE TABLE courses (
      course_id INT NOT NULL AUTO_INCREMENT,
      course_name VARCHAR(50) NOT NULL,
      PRIMARY KEY(course_id)
);

CREATE TABLE lecturers (
      lecturer_id INT NOT NULL AUTO_INCREMENT,
      lecturer_name VARCHAR(50) NOT NULL,
      PRIMARY KEY(lecturer_id)
);

CREATE TABLE group_course_lecturer_list (
      group_id INT NOT NULL,
      course_id INT NOT NULL,
      lecturer_id INT NOT NULL,
      PRIMARY KEY(group_id, course_id),
      FOREIGN KEY(group_id) REFERENCES groups(group_id),
      FOREIGN KEY(course_id) REFERENCES courses(course_id),
      FOREIGN KEY(lecturer_id) REFERENCES lecturers(lecturer_id)
);

CREATE TABLE student_group_list (
      student_id INT NOT NULL,
      group_id INT NOT NULL,
      PRIMARY KEY(student_id),
      FOREIGN KEY(student_id) REFERENCES students(student_id),
      FOREIGN KEY(group_id) REFERENCES groups(group_id)
);

CREATE TABLE student_course_mark_list (
      student_id INT NOT NULL,
      course_id INT NOT NULL,
      mark ENUM('E', 'D', 'C', 'B', 'A'),
      PRIMARY KEY(student_id, course_id),
      FOREIGN KEY(student_id) REFERENCES students(student_id),
      FOREIGN KEY(course_id) REFERENCES courses(course_id)
);
```

```
DELIMITER //

DROP TRIGGER IF EXISTS check_student_course_mark_insert//
CREATE TRIGGER check_student_course_mark_insert
	BEFORE INSERT ON student_course_mark_list FOR EACH ROW
	BEGIN
		DECLARE message VARCHAR(255);
		IF EXISTS (SELECT group_id FROM student_group_list WHERE student_id
= NEW.student_id AND group_id NOT IN (SELECT group_id FROM
group_course_lecturer_list WHERE course_id = NEW.course_id)) THEN
			SET message = CONCAT('Student \"', CAST(NEW.student_id AS
CHAR), '\" cannot have a mark for \"', CAST(NEW.course_id AS CHAR), '\"
course.');
			SIGNAL SQLSTATE '45000' SET message_text = message;
		END IF;
	END//

DROP TRIGGER IF EXISTS check_student_course_mark_update//
CREATE TRIGGER check_student_course_mark_update
	BEFORE UPDATE ON student_course_mark_list FOR EACH ROW
	BEGIN
		DECLARE message VARCHAR(255);
		IF EXISTS (SELECT group_id FROM student_group_list WHERE student_id
= NEW.student_id AND group_id NOT IN (SELECT group_id FROM
group_course_lecturer_list WHERE course_id = NEW.course_id)) THEN
			SET message = CONCAT('Student \"', CAST(NEW.student_id AS
CHAR), '\" cannot have a mark for \"', CAST(NEW.course_id AS CHAR), '\"
course.');
			SIGNAL SQLSTATE '45000' SET message_text = message;
		END IF;
	END//

DELIMITER ;
```

5. **Заполните базу тестовыми данными.**

```
INSERT INTO students (student_name) VALUES
	('Васин'),
	('Козлов'),
	('Шагал'),
	('Васильев'),
	('Дешевой')
;

INSERT INTO groups (group_name) VALUES
	('4538'),
	('4539')
;
```

```
INSERT INTO courses (course_name) VALUES
      ('Математический анализ'),
      ('Дискретная математика'),
      ('Алгоритмы и структуры данных')
;

INSERT INTO lecturers (lecturer_name) VALUES
      ('Додонов'),
      ('Станкевич')
;

INSERT INTO group_course_lecturer_list (group_id, course_id, lecturer_id) VALUES
      (1, 1, 1),
      (2, 1, 1),
      (1, 2, 2),
      (2, 2, 2),
      (1, 3, 2),
      (2, 3, 2)
;

INSERT INTO student_group_list (student_id, group_id) VALUES
      (1, 1),
      (2, 1),
      (3, 1),
      (4, 2),
      (5, 2)
;

INSERT INTO student_course_mark_list (student_id, course_id, mark) VALUES
      (1, 1, 'D'),
      (1, 2, 'E'),
      (2, 1, 'D'),
      (2, 2, 'C'),
      (3, 1, 'D'),
      (3, 2, 'E'),
      (4, 1, 'A'),
      (4, 2, 'A'),
      (5, 1, 'E'),
      (5, 2, 'E')
;
```