create database Расписание\_экзаменов

-- Создание таблицы "Студенты"

CREATE TABLE Студенты (

id\_студента INT PRIMARY KEY IDENTITY(1,1),

ФИО VARCHAR(100) NOT NULL,

Группа VARCHAR(50) NOT NULL,

Дата\_рождения DATE,

Адрес VARCHAR(100),

Телефон VARCHAR(15),

Email VARCHAR(50)

);

-- Создание таблицы "Предметы"

CREATE TABLE Предметы (

id\_предмета INT PRIMARY KEY IDENTITY(1,1),

Название VARCHAR(100) NOT NULL,

Преподаватель VARCHAR(100) NOT NULL,

Кафедра VARCHAR(100),

Описание TEXT,

Количество\_часов INT

);

-- Создание таблицы "Экзамены"

CREATE TABLE Экзамены (

id\_экзамена INT PRIMARY KEY IDENTITY(1,1),

id\_предмета INT,

Дата DATE,

Время TIME,

Аудитория VARCHAR(20),

Тип\_экзамена VARCHAR(50),

FOREIGN KEY (id\_предмета) REFERENCES Предметы (id\_предмета)

);

-- Создание таблицы "Результаты"

CREATE TABLE Результаты (

id\_результата INT PRIMARY KEY IDENTITY(1,1),

id\_студента INT,

id\_экзамена INT,

Оценка INT,

FOREIGN KEY (id\_студента) REFERENCES Студенты (id\_студента),

FOREIGN KEY (id\_экзамена) REFERENCES Экзамены (id\_экзамена)

);

-- Заполнение таблицы "Студенты"

INSERT INTO Студенты (ФИО, Группа, Дата\_рождения, Адрес, Телефон, Email)

VALUES

('Иванов Иван Иванович', 'Группа 1', '1990-01-01', 'г. Москва, ул. Ленина, д. 1', '+7 123 456-7890', 'ivanov@example.com'),

('Петров Петр Петрович', 'Группа 2', '1992-03-15', 'г. Санкт-Петербург, ул. Пушкина, д. 10', '+7 987 654-3210', 'petrov@example.com'),

('Сидорова Анна Васильевна', 'Группа 1', '1991-06-25', 'г. Москва, ул. Гагарина, д. 5', '+7 999 888-7777', 'sidorova@example.com');

-- Заполнение таблицы "Предметы"

INSERT INTO Предметы (Название, Преподаватель, Кафедра, Описание, Количество\_часов)

VALUES

('Математика', 'Иванова Мария Сергеевна', 'Кафедра математики', 'Основы высшей математики', 60),

('Физика', 'Петров Алексей Иванович', 'Кафедра физики', 'Механика, электромагнетизм', 80),

('Информатика', 'Сидоров Павел Николаевич', 'Кафедра информатики', 'Программирование, базы данных', 70);

-- Заполнение таблицы "Экзамены"

INSERT INTO Экзамены (id\_предмета, Дата, Время, Аудитория, Тип\_экзамена)

VALUES

(1, '2023-01-15', '09:00', 'Аудитория 101', 'Письменный'),

(2, '2023-02-20', '14:00', 'Аудитория 202', 'Устный'),

(3, '2023-03-10', '10:30', 'Аудитория 305', 'Практический');

-- Заполнение таблицы "Результаты"

INSERT INTO Результаты (id\_студента, id\_экзамена, Оценка)

VALUES

(1, 1, 4),

(2, 2, 5),

(3, 3, 5);

--для отображения информации о экзаменах и соответствующих кафедрах:

CREATE VIEW Экзамены\_Кафедра AS

SELECT

Экзамены.id\_экзамена,

Экзамены.Дата,

Экзамены.Время,

Предметы.Название AS Предмет,

Предметы.Кафедра

FROM

Экзамены

INNER JOIN Предметы ON Экзамены.id\_предмета = Предметы.id\_предмета;

select \* from Экзамены\_Кафедра

--для отображения информации о студентах и их оценках:

CREATE VIEW Студенты\_Результаты AS

SELECT

Студенты.ФИО AS Студент,

Студенты.Группа,

Результаты.id\_экзамена,

Предметы.Название AS Предмет,

Результаты.Оценка

FROM

Студенты

INNER JOIN Результаты ON Студенты.id\_студента = Результаты.id\_студента

INNER JOIN Предметы ON Результаты.id\_экзамена = Предметы.id\_предмета;

select \* from Студенты\_Результаты

--для отображения количества студентов, зарегистрированных на каждом экзамене:

CREATE VIEW Количество\_Студентов\_Экзамена AS

SELECT

Экзамены.id\_экзамена,

Экзамены.Дата,

Экзамены.Время,

COUNT(Результаты.id\_студента) AS Количество\_Студентов

FROM

Экзамены

LEFT JOIN Результаты ON Экзамены.id\_экзамена = Результаты.id\_экзамена

GROUP BY

Экзамены.id\_экзамена,

Экзамены.Дата,

Экзамены.Время;

select \* from Количество\_Студентов\_Экзамена

--для добавления нового студента в таблицу "Студенты":

CREATE PROCEDURE ДобавитьСтудента

@id\_студента INT,

@ФИО VARCHAR(100),

@Группа VARCHAR(50),

@Дата\_рождения DATE,

@Адрес VARCHAR(100),

@Телефон VARCHAR(15),

@Email VARCHAR(50)

AS

BEGIN

INSERT INTO Студенты (id\_студента, ФИО, Группа, Дата\_рождения, Адрес, Телефон, Email)

VALUES (@id\_студента, @ФИО, @Группа, @Дата\_рождения, @Адрес, @Телефон, @Email)

END

--для получения расписания экзаменов по заданной дате:

CREATE PROCEDURE ПолучитьРасписание

@Дата DATE

AS

BEGIN

SELECT

Экзамены.id\_экзамена,

Экзамены.Дата,

Экзамены.Время,

Экзамены.Аудитория,

Экзамены.Тип\_экзамена,

Предметы.Название AS Название\_предмета,

Предметы.Преподаватель

FROM

Экзамены

INNER JOIN Предметы ON Экзамены.id\_предмета = Предметы.id\_предмета

WHERE

Экзамены.Дата = @Дата

END

--для обновления оценки по заданному идентификатору результата:

CREATE PROCEDURE ОбновитьРезультат

@id\_результата INT,

@Оценка INT

AS

BEGIN

UPDATE Результаты

SET Оценка = @Оценка

WHERE id\_результата = @id\_результата

END

--Оценка должна быть в диапазоне от 1 до 5.

CREATE TRIGGER ПроверкаОценки

ON Результаты

INSTEAD OF INSERT

AS

BEGIN

DECLARE @Оценка INT

SELECT @Оценка = Оценка FROM inserted

IF @Оценка < 1 OR @Оценка > 5

BEGIN

RAISERROR ('Неверное значение оценки. Оценка должна быть в диапазоне от 1 до 5.', 16, 1)

ROLLBACK TRANSACTION

END

ELSE

BEGIN

-- Если значение оценки корректно, выполняется операция INSERT

INSERT INTO Результаты (id\_результата, id\_студента, id\_экзамена, Оценка)

SELECT id\_результата, id\_студента, id\_экзамена, Оценка FROM inserted

END

END;

--при удалении студента из системы, все связанные с ним результаты также будут удалены.

CREATE TRIGGER АвтоУдалениеРезультатов

ON Студенты

AFTER DELETE

AS

BEGIN

DELETE FROM Результаты

WHERE id\_студента IN (SELECT id\_студента FROM deleted)

END

--при удалении предмета из системы, все связанные с ним экзамены также будут удалены.

CREATE TRIGGER АвтоУдалениеПредмета

ON Предметы

AFTER DELETE

AS

BEGIN

DELETE FROM Экзамены

WHERE id\_предмета IN (SELECT id\_предмета FROM deleted)

END

CREATE FUNCTION ПолучитьЭкзаменыПоПредмету(@id\_предмета INT)

RETURNS TABLE

AS

RETURN (

SELECT Экзамены.\*, Предметы.Название AS НазваниеПредмета

FROM Экзамены

INNER JOIN Предметы ON Экзамены.id\_предмета = Предметы.id\_предмета

WHERE Экзамены.id\_предмета = @id\_предмета

);

CREATE FUNCTION ПолучитьРезультатыПоЭкзамену(@id\_экзамена INT)

RETURNS TABLE

AS

RETURN (

SELECT Результаты.\*, Студенты.ФИО AS ФИОСтудента, Экзамены.Дата, Экзамены.Время, Экзамены.Аудитория

FROM Результаты

INNER JOIN Студенты ON Результаты.id\_студента = Студенты.id\_студента

INNER JOIN Экзамены ON Результаты.id\_экзамена = Экзамены.id\_экзамена

WHERE Результаты.id\_экзамена = @id\_экзамена

);

CREATE FUNCTION ПолучитьСтудентовПоГруппе(@группа VARCHAR(50))

RETURNS TABLE

AS

RETURN (

SELECT Студенты.\*

FROM Студенты

WHERE Студенты.Группа = @группа

);

1. **Перечислите нормальные формы БД**

Вот перечень нормальных форм баз данных (НФБД):

1. Первая нормальная форма (1НФ): Все атрибуты в таблице должны быть атомарными, то есть не должны содержать повторяющихся или составных значений.

2. Вторая нормальная форма (2НФ): Таблица должна быть в 1НФ, и каждый неключевой атрибут должен полностью зависеть от всего составного первичного ключа.

3. Третья нормальная форма (3НФ): Таблица должна быть в 2НФ, и каждый неключевой атрибут должен зависеть только от первичного ключа, а не от других неключевых атрибутов.

4. Четвёртая нормальная форма (4НФ): Таблица должна быть в 3НФ, и неключевые атрибуты не должны иметь зависимых многозначных фактов.

5. Пятая нормальная форма (5НФ) или нормальная форма проекции-соединения (ПСНФ): Таблица должна быть в 4НФ, и любая зависимость должна быть выражена только через ключевые атрибуты.

6. Доменно-ключевая нормальная форма (ДКНФ): Все атрибуты, не входящие в ключевой атрибут, должны полностью зависеть от каждого ключевого атрибута, а не от их комбинации.

7. Шестая нормальная форма (6НФ): Цель этой нормальной формы состоит в устранении избыточности данных, путём разделения многозначных фактов и сохранения их отношений.