**СТУКТУРА БД**

**Таблица courses**

CourseID: int NOT NULL AUTO\_INCREMENT - Уникальный идентификатор курса.

SubjectID: int DEFAULT NULL - Идентификатор предмета, связанного с курсом.

TeacherID: int DEFAULT NULL - Идентификатор преподавателя, ведущего курс.

AcademicYear: varchar(9) NOT NULL - Академический год проведения курса.

Внешние ключи:

courses\_ibfk\_1: FOREIGN KEY (SubjectID) REFERENCES subjects (SubjectID).

courses\_ibfk\_2: FOREIGN KEY (TeacherID) REFERENCES teachers (TeacherID).

**Таблица grades**

GradeID: int NOT NULL AUTO\_INCREMENT - Уникальный идентификатор оценки.

StudentID: int DEFAULT NULL - Идентификатор студента.

SubjectID: int DEFAULT NULL - Идентификатор предмета.

Grade: int DEFAULT NULL - Оценка.

GradeDate: date NOT NULL - Дата выставления оценки.

Semester: enum ('Осенний', 'Весенний') - Семестр.

AcademicYear: varchar(9) NOT NULL - Академический год.

Внешние ключи:

grades\_ibfk\_1: FOREIGN KEY (StudentID) REFERENCES students (StudentID).

grades\_ibfk\_2: FOREIGN KEY (SubjectID) REFERENCES subjects (SubjectID).

Ограничения:

grades\_chk\_1: CHECK (Grade between 1 and 5).

**Таблица studentgroups**

GroupID: int NOT NULL AUTO\_INCREMENT - Уникальный идентификатор группы.

GroupName: varchar(255) NOT NULL - Название группы.

Таблица students

StudentID: int NOT NULL AUTO\_INCREMENT - Уникальный идентификатор студента.

FullName: varchar(255) NOT NULL - Полное имя студента.

DateOfBirth: date NOT NULL - Дата рождения студента.

ContactInfo: varchar(255) DEFAULT NULL - Контактная информация студента.

GroupID: int DEFAULT NULL - Идентификатор группы студента.

Внешний ключ:

FK\_Student\_Group: FOREIGN KEY (GroupID) REFERENCES studentgroups (GroupID).

**Таблица subjects**

SubjectID: int NOT NULL AUTO\_INCREMENT - Уникальный идентификатор предмета.

SubjectName: varchar(255) NOT NULL - Название предмета.

SubjectType: enum ('Гуманитарный', 'Математический') - Тип предмета.

**Таблица teachers**

TeacherID: int NOT NULL AUTO\_INCREMENT - Уникальный идентификатор преподавателя.

FullName: varchar(255) NOT NULL - Полное имя преподавателя.

DateOfBirth: date NOT NULL - Дата рождения преподавателя.

ContactInfo: varchar(255) DEFAULT NULL - Контактная информация преподавателя.

**ОТЧЕТ**

1) Была создана база данных университета с уже включенными ограничениями оценок значениями от 1 до 5 в таблицу Grades:

CREATE DATABASE University;

USE University;

CREATE TABLE Students (

StudentID INT AUTO\_INCREMENT PRIMARY KEY,

FullName VARCHAR(255) NOT NULL,

DateOfBirth DATE NOT NULL,

ContactInfo VARCHAR(255)

);

CREATE TABLE Teachers (

TeacherID INT AUTO\_INCREMENT PRIMARY KEY,

FullName VARCHAR(255) NOT NULL,

DateOfBirth DATE NOT NULL,

ContactInfo VARCHAR(255)

);

CREATE TABLE Subjects (

SubjectID INT AUTO\_INCREMENT PRIMARY KEY,

SubjectName VARCHAR(255) NOT NULL

);

CREATE TABLE Grades (

GradeID INT AUTO\_INCREMENT PRIMARY KEY,

StudentID INT,

SubjectID INT,

Grade INT CHECK (Grade BETWEEN 1 AND 5),

GradeDate DATE NOT NULL,

FOREIGN KEY (StudentID) REFERENCES Students(StudentID),

FOREIGN KEY (SubjectID) REFERENCES Subjects(SubjectID)

);

CREATE TABLE Courses (

CourseID INT AUTO\_INCREMENT PRIMARY KEY,

SubjectID INT,

TeacherID INT,

FOREIGN KEY (SubjectID) REFERENCES Subjects(SubjectID),

FOREIGN KEY (TeacherID) REFERENCES Teachers(TeacherID)

);

Добавлена градация предметов по типу:

ALTER TABLE Subjects ADD SubjectType ENUM('Гуманитарный', 'Математический') NOT NULL;

В таблицу оценок добавлены столбцы о семестре и учебном годе

ALTER TABLE Grades

ADD Semester ENUM('Осенний', 'Весенний') NOT NULL,

ADD AcademicYear VARCHAR(9) NOT NULL;

Создана таблица групп студентов:

CREATE TABLE StudentGroups (

GroupID INT AUTO\_INCREMENT PRIMARY KEY,

GroupName VARCHAR(255) NOT NULL

);

добавлен столбец для группы в таблицу Students:

ALTER TABLE Students ADD GroupID INT;

Затем добавьте внешний ключ для связи с таблицей Studentgroups:

ALTER TABLE Students ADD CONSTRAINT FK\_Student\_Group FOREIGN KEY (GroupID) REFERENCES studentgroups(GroupID);

Добавлены группы:

INSERT INTO studentgroups (GroupName) VALUES ('Группа А'), ('Группа Б');

Студенты распределены по группам:

UPDATE Students SET GroupID = 1 WHERE StudentID IN (1, 3, 5, 7, 9, 11, 13);

UPDATE Students SET GroupID = 2 WHERE StudentID IN (2, 4, 6, 8, 10, 12);

2) Предусмотрены механизмы для предотвращения дублирования записей или ввода некорректных данных.

ALTER TABLE Students ADD UNIQUE (FullName, DateOfBirth);

ALTER TABLE Teachers ADD UNIQUE (FullName, DateOfBirth);

ALTER TABLE Subjects ADD UNIQUE (SubjectName);

3) Заданы следующие возможность:

вывод списка студентов по определённому предмету:

SELECT DISTINCT s.FullName

FROM Students s

JOIN Grades g ON s.StudentID = g.StudentID

WHERE g.SubjectID = [SubjectID];

вывод списка предметов, которые преподаёт конкретный преподаватель:

SELECT sb.SubjectName

FROM Subjects sb

JOIN Courses c ON sb.SubjectID = c.SubjectID

WHERE c.TeacherID = [TeacherID];

вывод среднего балла студента по всем предметам:

SELECT AVG(g.Grade)

FROM Grades g

WHERE g.StudentID = [StudentID];

вывод рейтинга преподавателей по средней оценке студентов:

SELECT t.FullName, AVG(g.Grade) as AverageGrade

FROM Teachers t

JOIN Courses c ON t.TeacherID = c.TeacherID

JOIN Grades g ON c.SubjectID = g.SubjectID

GROUP BY t.TeacherID

ORDER BY AverageGrade DESC;

вывод списка преподавателей, которые преподавали более 3 предметов за последний год:

SELECT

t.TeacherID,

t.FullName,

COUNT(DISTINCT c.SubjectID) AS NumberOfSubjects

FROM Teachers t

JOIN Courses c ON t.TeacherID = c.TeacherID

WHERE c.AcademicYear = [AcademicYear]

GROUP BY t.TeacherID

HAVING NumberOfSubjects > 3;

вывод списка студентов, которые имеют средний балл выше 4 по математическим предметам, но ниже 3 по гуманитарным:

SELECT

s.StudentID,

s.FullName,

AVG(CASE WHEN sb.SubjectType = 'Математический' THEN g.Grade END) AS AvgMathGrade,

AVG(CASE WHEN sb.SubjectType = 'Гуманитарный' THEN g.Grade END) AS AvgHumanGrade

FROM Students s

JOIN Grades g ON s.StudentID = g.StudentID

JOIN Subjects sb ON g.SubjectID = sb.SubjectID

GROUP BY s.StudentID, s.FullName

HAVING AvgMathGrade > 4 AND AvgHumanGrade < 3;

определение предмета, по которым больше всего двоек в текущем семестре:

SELECT

sb.SubjectName,

COUNT(\*) AS CountOfTwos

FROM Grades g

JOIN Subjects sb ON g.SubjectID = sb.SubjectID

WHERE g.Grade = 2 AND g.Semester = ' [Semester]' AND g.AcademicYear = '[AcademicYear]'

GROUP BY g.SubjectID

ORDER BY CountOfTwos DESC

LIMIT 1;

вывод студентов, которые получили высший балл по всем своим экзаменам (средняя оценка по предмету) и преподавателей, которые вели эти предметы:

SELECT

g.SubjectID,

su.SubjectName,

s.StudentID,

s.FullName AS StudentName,

t.TeacherID,

t.FullName AS TeacherName,

AVG(g.Grade) AS AverageGrade

FROM Grades g

JOIN Students s ON g.StudentID = s.StudentID

JOIN Courses c ON g.SubjectID = c.SubjectID

JOIN Teachers t ON c.TeacherID = t.TeacherID

JOIN Subjects su ON g.SubjectID = su.SubjectID

GROUP BY g.SubjectID, s.StudentID, t.TeacherID

HAVING AVG(g.Grade) = (

SELECT MAX(AvgGrade)

FROM (

SELECT

AVG(g2.Grade) AS AvgGrade

FROM Grades g2

WHERE g2.SubjectID = g.SubjectID

GROUP BY g2.StudentID

) AS SubQuery

)

просмотр изменения среднего балла студента по годам обучения:

SELECT

s.StudentID,

s.FullName,

g.AcademicYear,

AVG(g.Grade) AS AverageGrade

FROM Students s

JOIN Grades g ON s.StudentID = g.StudentID

GROUP BY s.StudentID, g.AcademicYear

ORDER BY s.StudentID, g.AcademicYear;

определение группы, в которой средний балл выше, чем в других, по аналогичным предметам:

SELECT

s.GroupID,

g.SubjectID,

AVG(g.Grade) AS AverageGrade

FROM Grades g

JOIN Students s ON g.StudentID = s.StudentID

GROUP BY s.GroupID, g.SubjectID

HAVING AVG(g.Grade) >= ALL (

SELECT AVG(g2.Grade)

FROM Grades g2

JOIN Students s2 ON g2.StudentID = s2.StudentID

WHERE g2.SubjectID = g.SubjectID AND s2.GroupID != s.GroupID

GROUP BY s2.GroupID

)

6. Вставьте записи о новом студенте с его личной информацией: ФИО, дата рождения, контактные данные и др.

INSERT INTO Students (FullName, DateOfBirth, ContactInfo) VALUES

('Иван Иванов', '2000-01-01', 'ivanov@example.com');

7. Имеется возможность:

обновления контактной информации преподавателя, например, электронной почты или номера телефона, на основе его идентификационного номера или ФИО:

UPDATE Teachers

SET ContactInfo = 'new\_email@example.com'

WHERE TeacherID = [TeacherID]

удаления записи о предмете, который больше не преподают в учебном заведении:

START TRANSACTION;

INSERT INTO Subjects (SubjectName) VALUES ('Удаленный предмет');

SET @PlaceholderSubjectID = LAST\_INSERT\_ID();

UPDATE Courses SET SubjectID = @PlaceholderSubjectID WHERE SubjectID = [SubjectID];

DELETE FROM Subjects WHERE SubjectID = [SubjectID];

COMMIT;