-- Создание таблицы "Студенты"

CREATE TABLE Студенты (

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

Имя VARCHAR(100) NOT NULL,

Фамилия VARCHAR(100) NOT NULL,

Группа VARCHAR(50) NOT NULL

);

-- Создание таблицы "Преподаватели"

CREATE TABLE Преподаватели (

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

Имя VARCHAR(100) NOT NULL,

Фамилия VARCHAR(100) NOT NULL,

Должность VARCHAR(50) NOT NULL

);

-- Создание таблицы "Предметы"

CREATE TABLE Предметы (

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

Название VARCHAR(100) NOT NULL,

Описание TEXT,

id\_преподавателя INT,

FOREIGN KEY (id\_преподавателя) REFERENCES Преподаватели(id\_преподавателя)

);

-- Создание таблицы "Результаты"

CREATE TABLE Результаты (

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

id\_студента INT,

id\_предмета INT,

Оценка INT,

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

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

);

-- Заполнение таблицы "Студенты"

INSERT INTO Студенты (Имя, Фамилия, Группа)

VALUES

('Иван', 'Иванов', 'Группа 1'),

('Петр', 'Петров', 'Группа 2'),

('Анна', 'Сидорова', 'Группа 1'),

('Мария', 'Кузнецова', 'Группа 3'),

('Алексей', 'Смирнов', 'Группа 2');

-- Заполнение таблицы "Преподаватели"

INSERT INTO Преподаватели (Имя, Фамилия, Должность)

VALUES

('Александр', 'Иванов', 'Профессор'),

('Елена', 'Петрова', 'Доцент'),

('Иван', 'Сидоров', 'Преподаватель'),

('Ольга', 'Кузнецова', 'Старший преподаватель'),

('Сергей', 'Смирнов', 'Ассистент');

-- Заполнение таблицы "Предметы"

INSERT INTO Предметы (Название, Описание, id\_преподавателя)

VALUES

('Математика', 'Основы математического анализа', 1),

('Физика', 'Механика и термодинамика', 2),

('История', 'Всемирная история нового времени', 3),

('Программирование', 'Основы программирования на языке Python', 4),

('Литература', 'Классическая литература XIX века', 5);

-- Заполнение таблицы "Результаты"

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

VALUES

(1, 1, 5),

(1, 2, 4),

(2, 1, 3),

(2, 3, 5),

(3, 2, 4);

CREATE VIEW Студенты\_Оценки AS

SELECT

Студенты.id\_студента,

Студенты.Имя AS Имя\_студента,

Студенты.Фамилия AS Фамилия\_студента,

Предметы.id\_предмета,

Предметы.Название AS Название\_предмета,

Результаты.Оценка

FROM

Студенты

JOIN

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

JOIN

Предметы ON Результаты.id\_предмета = Предметы.id\_предмета;

select \* from Студенты\_Оценки

CREATE VIEW Преподаватели\_Предметы AS

SELECT

Преподаватели.\*,

Предметы.Название AS Название\_предмета,

Предметы.Описание

FROM

Преподаватели

JOIN Предметы ON Преподаватели.id\_преподавателя = Предметы.id\_преподавателя;

select \* from Преподаватели\_Предметы

CREATE VIEW Студенты\_Преподаватели AS

SELECT

Студенты.id\_студента,

Студенты.Имя AS Имя\_студента,

Студенты.Фамилия AS Фамилия\_студента,

Преподаватели.id\_преподавателя,

Преподаватели.Имя AS Имя\_преподавателя,

Преподаватели.Фамилия AS Фамилия\_преподавателя

FROM

Студенты

JOIN

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

JOIN

Предметы ON Результаты.id\_предмета = Предметы.id\_предмета

JOIN

Преподаватели ON Предметы.id\_преподавателя = Преподаватели.id\_преподавателя;

select \* from Студенты\_Преподаватели

CREATE PROCEDURE ПолучитьСтудентовПоГруппе

@Группа VARCHAR(50)

AS

BEGIN

SELECT Имя, Фамилия

FROM Студенты

WHERE Группа = @Группа;

END;

CREATE PROCEDURE ДобавитьСтудента

@Имя VARCHAR(100),

@Фамилия VARCHAR(100),

@Группа VARCHAR(50)

AS

BEGIN

INSERT INTO Студенты (Имя, Фамилия, Группа)

VALUES (@Имя, @Фамилия, @Группа);

END;

CREATE PROCEDURE ОбновитьОценкуСтудента

@id\_студента INT,

@id\_предмета INT,

@Оценка INT

AS

BEGIN

UPDATE Результаты

SET Оценка = @Оценка

WHERE id\_студента = @id\_студента

AND id\_предмета = @id\_предмета;

END;

CREATE TRIGGER ПроверкаОценки

ON Результаты

AFTER INSERT

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE Оценка < 2 OR Оценка > 5)

BEGIN

RAISERROR('Недопустимая оценка. Оценка должна быть от 2 до 5.', 16, 1);

ROLLBACK;

END;

END;

CREATE TRIGGER УдалениеРезультатовСтудента

ON Студенты

AFTER DELETE

AS

BEGIN

DELETE FROM Результаты

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

END;

CREATE TRIGGER УдалениеПреподавателя

ON Преподаватели

AFTER DELETE

AS

BEGIN

PRINT 'Преподаватель был удален!';

END;

CREATE FUNCTION ПолучитьСписокСтудентовПоГруппе(@группа VARCHAR(50))

RETURNS TABLE

AS

RETURN

(

SELECT Имя, Фамилия

FROM Студенты

WHERE Группа = @группа

);

CREATE FUNCTION ПолучитьСписокПредметовПреподавателя(@id\_преподавателя INT)

RETURNS TABLE

AS

RETURN

(

SELECT Название, Описание

FROM Предметы

WHERE id\_преподавателя = @id\_преподавателя

);

CREATE FUNCTION ПолучитьСреднийБаллСтудента(@id\_студента INT)

RETURNS FLOAT

AS

BEGIN

DECLARE @среднийБалл FLOAT;

SELECT @среднийБалл = AVG(Оценка)

FROM Результаты

WHERE id\_студента = @id\_студента;

RETURN @среднийБалл;

END;

1. **Алгоритм проведения процедуры резервного копирования.**

Алгоритм проведения процедуры резервного копирования может включать следующие шаги:

1. Определение объема данных

2. Выбор типа резервного копирования

3. Планирование расписания

4. Выбор места хранения

5. Использование соответствующих инструментов

6. Инициирование процесса резервного копирования

7. Проверка целостности

8. Хранение и управление копиями.

9. Тестирование восстановления

10. Мониторинг и обновление процедуры

11. Документация и доверенность