

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ «МИСИС»

Институт компьютерных наук

Контрольное домашнее задание

по дисциплине

«Базы данных»

на тему

«Кафедра вуза»

Выполнил:
студент 2-го курса,
гр. БПМ-22-3 Самсонов Н.О.

Проверил:
Новицкий Дмитрий

Москва, 2023

Оглавление

1. Постановка задачи.....	3
2. Описание структуры БД	3
2.1. Вербальная модель	3
2.2. Реляционная модель	5
2.3. Анализ функциональных зависимостей.....	7
3. Заполнение БД информацией.....	7
4. Описание представлений.....	9
5. Описание функций.....	10
6. Описание хранимых процедур.....	12
7. Описание триггеров	14
8. Пример работы с БД с использованием созданных объектов.....	16
9. Роли и пользователи.....	18
10. Создание копий	19
11. Список литературы	19

1. Постановка задачи

Вариант 3. Предметная область: Кафедра ВУЗа.

Необходимо разработать БД, которая должна обеспечивать информационную поддержку учебного процесса и организационной деятельности на кафедре ВУЗа. БД должна содержать учебный план, расписание занятий, списки групп, выпускаемых кафедрой, и списки аспирантов (с руководителями и темами исследований).

БД должна обеспечивать составление:

- расписания занятий на семестр (по группам);
- учебного плана (по семестрам) для каждого курса;
- расписания занятий для преподавателей;
- списка телефонов сотрудников;
- нагрузки по часам для преподавателей;
- списка научных кадров по научным направлениям;
- списков студентов-дипломников (по группам и по преподавателям).

2. Описание структуры БД

2.1. Вербальная модель

Сущность "Должность":

- Описывает различные должности в учебном учреждении, такие как преподаватель, администратор и т.д.
- Связана с сущностью "Сотрудник": каждый сотрудник занимает определенную должность. Тип связи: один ко многим.

Сущность "Сотрудник":

- Представляет информацию о сотрудниках учебного учреждения. Содержит данные о часах работы, контактной информации.
- Связана с сущностью "Преподаватель": каждый преподаватель является сотрудником. Тип связи: один к одному (реализация наследования).

Сущность "Преподаватель":

- Дополняет информацию о сотруднике для преподавателей, включая стаж и научную степень.
- Связана с сущностью "Расписание занятий": Каждый преподаватель может вести

несколько занятий по расписанию. У каждого занятия по расписанию может быть только один преподаватель. Тип связи: один ко многим.

Сущность "Расписание занятий":

- Описывает расписание занятий, включая привязку к преподавателю, паре, дисциплине и группе.
- Связана с сущностями "Пара", "Дисциплина" и "Группа": каждое занятие по расписанию связано с определенной парой, дисциплиной и группой. Тип связи: один ко многим.

Сущность "Пара":

- Содержит информацию о занятиях, включая день недели, время начала и окончания занятий, а также верхнюю или нижнюю неделю.

Сущность "Дисциплина":

- Содержит информацию о предметах, включая номер семестра, часы практики, лабораторных и лекций, а также количество зачетных единиц.
- Связана с сущностью "Направление подготовки": каждое направление подготовки включает в себя определённый набор дисциплин. Дисциплина может принадлежать к нескольким направлениям подготовки. Тип связи: многие ко многим.

Сущность "Направление подготовки":

- Содержит информацию о направлениях подготовки, таких как "Информационные технологии" и "Экономика".
- Связана с сущностью "Уровень подготовки": каждое направление подготовки имеет свой уровень подготовки. Тип связи: один ко многим.

Сущность "Уровень подготовки":

- Описывает различные уровни подготовки (бакалавр, магистр и т.д.).

Сущность "Группа":

- Представляет информацию о группах студентов, включая номер курса и название группы.
- Связана с сущностью "Студент": каждый студент принадлежит определенной группе. У каждой группы может быть много студентов, но каждый студент может быть только в одной группе. Тип связи: один ко многим.
- Связана с сущностью "Преподаватель": преподаватель может являться куратором нескольких групп. У группы есть один преподаватель. Тип отношения: один ко многим.
- Связана с сущностью "Направление подготовки": группа зачисляется на направление

подготовки. На одно направление подготовки может быть зачислено несколько групп.
Тип отношения: один ко многим.

Сущность "Студент":

- Содержит информацию о студентах, включая личную информацию, дату рождения и контактную информацию.
- Связана с сущностью "Преподаватель": преподаватель может являться научным руководителем нескольких студентов. Студент (аспирант) может иметь одного научного руководителя. Тип связи: один ко многим.

2.2. Реляционная модель

Таблица "Должность":

- Атрибуты: ID_должности (PK), Оклад, Ставка, Название_должности.

Таблица "Сотрудник":

- Атрибуты: ID_сотрудника (PK), ID_должности (FK), Электронная_почта, Номер_телефона, Имя, Фамилия, Отчество.
- Внешний ключ: ID_должности (ссылается на ID_должности в таблице "Должность").

Таблица "Преподаватель":

- Атрибуты: ID_преподавателя (PK), ID_сотрудника (FK), Стаж, Научная_степень.
- Внешний ключ: ID_сотрудника (ссылается на ID_сотрудника в таблице "Сотрудник").

Таблица "Пара":

- Атрибуты: ID_пары (PK), День_недели, Время_окончания, Время_начала, Верхняя_нижняя_неделя.

Таблица "Уровень_подготовки":

- Атрибуты: ID_уровня_подготовки (PK), Название_уровня_подготовки, Форма_обучения.

Таблица "Направление_подготовки":

- Атрибуты: ID_направления_подготовки (PK), ID_уровня_подготовки (FK), Название_направления_подготовки, Область_образования.
- Внешний ключ: ID_уровня_подготовки (ссылается на ID_уровня_подготовки в таблице "Уровень_подготовки").

Таблица "Дисциплина":

- Атрибуты: ID_дисциплины (PK), ID_направления_подготовки (FK), Номер_семестра, Название_дисциплины, Часы_практики, Часы_лабораторных, Часы_лекций, Количество_зачётных_единиц.
- Внешний ключ: ID_направления_подготовки (ссылается на ID_направления_подготовки в таблице "Направление_подготовки").

Таблица "Группа":

- Атрибуты: ID_группы (PK), ID_куратора (FK), Номер_курса, Название_группы, Количество_студентов, ID_направления_подготовки.
- Внешний ключ: ID_куратора (ссылается на ID_преподавателя в таблице "Преподаватель").
- Внешний ключ: ID_направления_подготовки (ссылается на ID_направления_подготовки в таблице "Направление_подготовки").

Таблица "Расписание_занятий":

- Атрибуты: (ID_преподавателя (FK), ID_пары (FK))(PK), ID_дисциплины (FK), ID_группы (FK), Номер_аудитории.
- Внешние ключи: ID_преподавателя (ссылается на ID_преподавателя в таблице "Преподаватель"), ID_пары (ссылается на ID_пары в таблице "Пара"), ID_дисциплины (ссылается на ID_дисциплины в таблице "Дисциплина"), ID_группы (ссылается на ID_группы в таблице "Группа").

Таблица "Студент":

- Атрибуты: ID_студента (PK), ID_научного_руководителя (FK), ID_группы (FK),

Дата_рождения, Электронная_почта, Имя, Фамилия, Отчество.

- Внешние ключи: ID_научного_руководителя (ссылается на ID_преподавателя в таблице "Преподаватель"), ID_группы (ссылается на ID_группы в таблице "Группа").

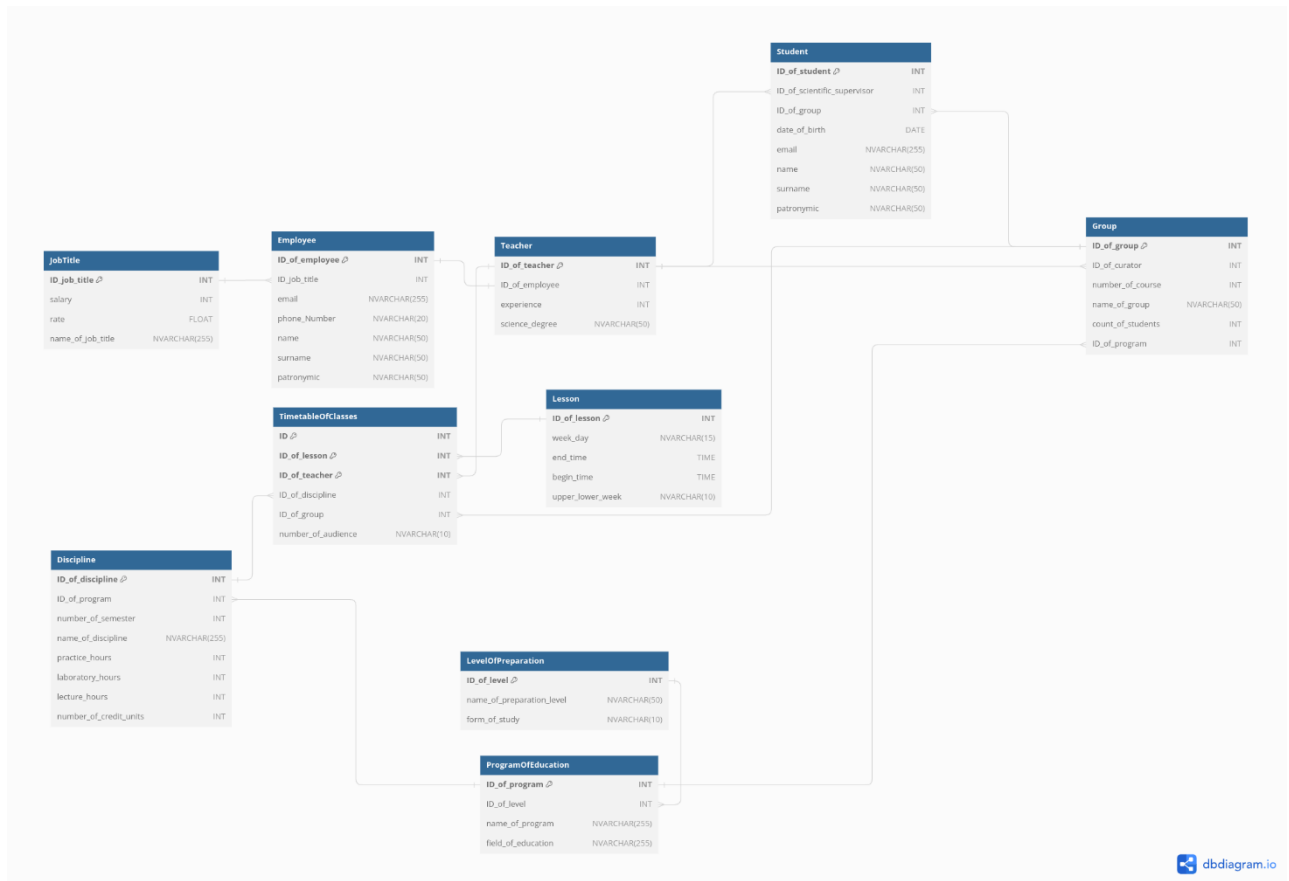


Рис.1. Диаграмма базы данных

2.3. Анализ функциональных зависимостей

Все отношения находятся в 1НФ, так как все их атрибуты являются простыми, все используемые домены содержат только скалярные значения.

Все отношения находятся в 2НФ, так как находятся в 1НФ и все неключевые атрибуты функционально зависят от полного первичного ключа.

Все отношения находятся в 3НФ, так как находятся во 2НФ и не имеют транзитивных зависимостей неключевых атрибутов от первичного ключа.

3. Заполнение БД информацией

Данные для заполнения таблиц генерируются с помощью ChatGPT и заполняются в ручном режиме.

Заполнение таблицы "Сотрудник":

```

INSERT INTO Сотрудник (ID_должности, Имя, Фамилия, Отчество,
Электронная_почта, Номер_телефона)
VALUES
  (1, 'Анна', 'Иванова', 'Петровна', 'anna@example.com',
'+7 (999) 123-45-67'),
  (2, 'Дмитрий', 'Петров', 'Александрович',
'dmitry@example.com', '+7 (999) 987-65-43'),
  (3, 'Елена', 'Сидорова', 'Ивановна', 'elena@example.com',
'+7 (999) 111-22-33'),
  (4, 'Алексей', 'Козлов', 'Сергеевич', 'alexey@example.com',
'+7 (999) 555-44-33'),

  (5, 'Мария', 'Смирнова', 'Алексеевна',
'maria@example.com', '+7 (999) 876-54-32');

```

Заполнение таблицы "Сотрудник":

```

INSERT INTO Направление_подготовки (ID_уровня_подготовки,
Название_направления_подготовки, Область_образования)
VALUES
  (1, 'Информационные технологии', 'Технические науки'),
  (1, 'Экономика и управление', 'Экономические науки'),
  (2, 'Психология', 'Гуманитарные науки'),
  (2, 'Медицина', 'Медицинские науки'),

  (3, 'Право', 'Юридические науки');

```

Заполнение таблицы "Группа":

```

INSERT INTO Группа (Название_группы, ID_куратора, Номер_курса,
ID_направления_подготовки)
VALUES
  ('БПМ-22-1', 1, 2, 1),
  ('БПМ-22-2', 1, 2, 1),
  ('БПМ-23-1', 1, 1, 1),
  ('БПМ-23-2', 1, 1, 1),
  ('БивТ-22-1', 2, 2, 1),
  ('БивТ-22-2', 2, 2, 1),
  ('БивТ-20-1', 2, 4, 1),
  ('БивТ-21-1', 4, 3, 1),

  ('БЛГ-22-1', 4, 2, 2);

```

Количество записей:

"Должность" – 9

"Сотрудник" – 5

"Преподаватель" – 4

"Пара" – 50

"Уровень_подготовки" – 6

"Направление_подготовки" – 5

"Дисциплина" – 5

"Группа" – 9

"Расписание_занятий" – 50

"Студент" – 24

4. Описание представлений

v_DiplomaStudents – список студентов-дипломников с научными руководителями

v_GroupCurator - информация о группах и их кураторах

v_GetSalary – информация о зарплате сотрудников (рассчитывается исходя из оклада и ставки)

v_TeachersList - список всех преподавателей с их должностями

Представление v_DiplomaStudents:

```
CREATE VIEW v_DiplomaStudents AS
SELECT
    Студент.ID_студента,
    Студент.Имя AS Студент_Имя,
    Студент.Фамилия AS Студент_Фамилия,
    Студент.Отчество AS Студент_Отчество,
    Студент.ID_группы,
    Сотрудник.Имя AS Руководитель_Имя,
    Сотрудник.Фамилия AS Руководитель_Фамилия,
    Сотрудник.Отчество AS Руководитель_Отчество,
    Преподаватель.Научная_степень,
    Сотрудник.Электронная_почта AS Руководитель_Электронная_почта
FROM
    Студент
LEFT JOIN Преподаватель ON Студент.ID_научного_руководителя =
    Преподаватель.ID_преподавателя
LEFT JOIN Сотрудник ON Преподаватель.ID_сотрудника =
    Сотрудник.ID_сотрудника;
```

Представление v_GetSalary:

```
CREATE VIEW v_GetSalary
AS
```

```

SELECT
    Сотрудник.ID_сотрудника,
    Сотрудник.Имя,
    Сотрудник.Фамилия,
    Сотрудник.Отчество,
    Должность.Название_должности,
    Должность.Оклад,
    Должность.Ставка,
    (Должность.Оклад * Должность.Ставка) AS Зарплата
FROM
    Сотрудник
INNER JOIN
    Должность ON Сотрудник.ID_должности = Должность.ID_должности;

```

5. Описание функций

fn_CreateScheduleForTeacher (@Name, @SecondName, @Surname) - формирует расписание для преподавателя

fn_GetStudentsByGroup (@GroupName) - получение списка студентов по номеру группы

fn_GetEmployeePhoneList (
@FirstName = NULL,
@LastName = NULL,
@MiddleName = NULL) - возвращает контактную информацию сотрудника, если он указан, в противном случае – возвращается список контактной информации всех сотрудников

fn_GetListOfDisciplines(@DirectionName) – формирует список дисциплин для определенного направления

fn_CalculateWorkloadForTeacher(@TeacherID) – рассчитывает нагрузку по часам для преподавателя

Функция **fn_CreateScheduleForGroup** (табличная):

```

CREATE FUNCTION fn_CreateScheduleForTeacher (@Name NVARCHAR(50),
@SecondName NVARCHAR(50), @Surname NVARCHAR(50))
RETURNS TABLE
AS
RETURN
(
    SELECT
        Неделя,

```

```

        FORMAT(Время_начала, N'hh\:mm') AS Время_начала,
        FORMAT(Время_окончания, N'hh\:mm') AS Время_окончания,
        Понедельник,
        Вторник,
        Среда,
        Четверг,
        Пятница
FROM
(
    SELECT
        Пара.Верхняя_нижняя_неделя AS Неделя,
        Пара.Время_начала,
        Пара.Время_окончания,
        Пара.День_недели,
        CONCAT(Группа.Название_группы, ' ',
Дисциплина.Название_дисциплины, ' ',
Расписание_занятий.Номер_аудитории) AS Расписание

        FROM Расписание_занятий
        JOIN Преподаватель ON Расписание_занятий.ID_преподавателя
= Преподаватель.ID_преподавателя
        JOIN Сотрудник ON Преподаватель.ID_сотрудника =
Сотрудник.ID_сотрудника
        JOIN Пара ON Расписание_занятий.ID_пары = Пара.ID_пары
        JOIN Дисциплина ON Расписание_занятий.ID_дисциплины =
Дисциплина.ID_дисциплины
        JOIN Группа ON Расписание_занятий.ID_группы =
Группа.ID_группы
        WHERE Сотрудник.Имя = @Name AND Сотрудник.Фамилия =
@Surname AND Сотрудник.Отчество = @SecondName
    ) AS SourceTable
    PIVOT
    (
        MAX(Расписание) FOR День_недели IN ([Понедельник],
[Вторник], [Среда], [Четверг], [Пятница])
    ) AS PivotTable
);

```

Функция fn_CalculateWorkloadForTeacher (скалярная):

```

CREATE FUNCTION fn_CalculateWorkloadForTeacher(@TeacherID INT)
RETURNS INT
AS
BEGIN
    DECLARE @TotalWorkload INT = 0;

    SELECT @TotalWorkload = COUNT(*) * 2
    FROM Расписание_занятий
    WHERE ID_преподавателя = @TeacherID;

    RETURN @TotalWorkload * 2;
END;

```

6. Описание хранимых процедур

pr_CreateScheduleForGroup (@GroupName) - формирует расписание для конкретной группы

pr_RemoveStudent (@StudentID INT) – удаление(отчисление) студента

pr_AddNewStudent (
 @Дата_рождения,
 @Имя,
 @Фамилия,
 @Отчество = NULL,
 @ID_группы = NULL,
 @ID_научного_руководителя = NULL) - добавление нового студента. Если группа не указана, то студент добавляется в группу с наименьшим количеством человек. Также для нового студента автоматически формируется электронная почта.

pr_AddNewEmployee (
 @ID_должности,
 @Имя,
 @Фамилия,
 @Отчество = NULL,
 @Электронная_почта = NULL,
 @Номер_телефона = NULL,
 @Стаж = NULL,
 @Научная_степень = NULL) - добавление нового сотрудника с проверкой, что его должность соответствует должностям, которое есть на кафедре. Если сотрудник является преподавателем, то информация об этом сотруднике добавляется в таблицу "Преподаватель".

pr_RemoveEmployee (@ID_сотрудника) - удаление сотрудника. Если сотрудник является преподавателем, то сначала удаляется запись о нём как о преподавателе, а затем – как о сотруднике.

pr_TransferStudentToAnotherGroup (
 @ID_студента,
 @ID_группы_новой) - перевод студента в другую группу с проверкой, что студент с заданным ID и группа существуют.

Процедура pr_AddNewStudent:

```
CREATE PROCEDURE pr_AddNewStudent (
```

```

        @Дата_рождения DATE,
        @Имя NVARCHAR(50),
        @Фамилия NVARCHAR(50),
        @Отчество NVARCHAR(50) = NULL,
        @ID_группы INT = NULL,
        @ID_научного_руководителя INT = NULL
    )
AS
BEGIN
    DECLARE @SelectedGroup INT;

    IF @ID_группы IS NULL
    BEGIN
        WITH SelectedGroup AS (
            SELECT TOP 1 ID_группы
            FROM Группа
            ORDER BY Количество_студентов ASC
        )
        SELECT @SelectedGroup = ID_группы FROM SelectedGroup;
    END

    INSERT INTO Студент (Имя, Фамилия, Отчество, ID_группы,
        Дата_рождения, ID_научного_руководителя)
    VALUES (@Имя, @Фамилия, @Отчество, @SelectedGroup,
        @Дата_рождения, @ID_научного_руководителя);

    DECLARE @ID INT = SCOPE_IDENTITY();
    DECLARE @Электронная_почта NVARCHAR(255) = CONCAT(@ID,
        '@example.com');

    UPDATE Студент
    SET Электронная_почта = @Электронная_почта
    WHERE ID_студента = @ID;

    PRINT CONCAT('Студент успешно добавлен в ', @SelectedGroup, '
    группу. ');
END;

```

Процедура pr_AddNewEmployee:

```

CREATE PROCEDURE pr_AddNewEmployee (
    @ID_должности INT,
    @Имя NVARCHAR(50),
    @Фамилия NVARCHAR(50),
    @Отчество NVARCHAR(50) = NULL,
    @Электронная_почта NVARCHAR(255) = NULL,
    @Номер_телефона NVARCHAR(20) = NULL,
    @Стаж INT = NULL,
    @Научная_степень NVARCHAR(50) = NULL
)
AS
BEGIN
    BEGIN TRY

```

```

        IF NOT EXISTS (SELECT 1 FROM Должность WHERE ID_должности =
@ID_должности)
        BEGIN
            THROW 50000, 'Должности с таким ID не существует', 1;
        END

        BEGIN TRANSACTION
            INSERT INTO Сотрудник (ID_должности, Имя, Фамилия, Отчество,
Электронная_почта, Номер_телефона)
            VALUES (@ID_должности, @Имя, @Фамилия, @Отчество,
@Электронная_почта, @Номер_телефона);
            DECLARE @ID_сотрудника INT;
            SET @ID_сотрудника = SCOPE_IDENTITY();

            DECLARE @Название_должности NVARCHAR(255) = (
                SELECT Название_должности
                FROM Должность
                JOIN Сотрудник ON Сотрудник.ID_должности =
Должность.ID_должности
                WHERE ID_сотрудника = @ID_сотрудника);

            IF @Название_должности IN ('Профессор', 'Доцент', 'Старший
преподаватель', 'Преподаватель')
            BEGIN
                INSERT INTO Преподаватель (ID_сотрудника, Стаж,
Научная_степень)
                VALUES (@ID_сотрудника, @Стаж, @Научная_степень);
            END
        COMMIT TRANSACTION
    END TRY

    BEGIN CATCH
        IF @@TRANCOUNT > 0
        BEGIN
            ROLLBACK TRANSACTION;
            THROW;
        END
        ELSE
        BEGIN
            PRINT ERROR_MESSAGE();
        END;
    END CATCH
END;

```

7. Описание триггеров

tr_ReplaceTeacherInSchedule – триггер, срабатывающий вместо удаления преподавателя. Перед его удалением триггер сначала заменяет ID_преподавателя, которого удаляем, в расписании на ID другого преподавателя. Также, если преподавателя является научным руководителем студента, то ID_научного_руководителя для студента становится NULL.

tr_UpdateStudentCount – триггер, срабатывающий после любых изменений в таблице "Студент". Триггер обновляет количество студентов в группах, в которые были добавлены/из которых были удалены студенты.

tr_UpdateTeacherSalary – триггер, который увеличивает оклад преподавателя на 5% за каждый год стажа. Срабатывает после вставки или обновления записи в таблице "Преподаватель".

Триггер tr_UpdateStudentCount:

```
CREATE TRIGGER tr_UpdateStudentCount
ON Студент
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    DECLARE @GroupID INT;
    DECLARE @StudentCount INT;

    DECLARE Group_Cursor CURSOR FOR
    SELECT DISTINCT ID_группы
    FROM inserted
    UNION
    SELECT DISTINCT ID_группы
    FROM deleted;

    OPEN Group_Cursor;

    FETCH NEXT FROM Group_Cursor INTO @GroupID;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        SELECT @StudentCount = COUNT(ID_студента)
        FROM Студент
        WHERE ID_группы = @GroupID;

        UPDATE Группа
        SET Количество_студентов = @StudentCount
        WHERE ID_группы = @GroupID;

        FETCH NEXT FROM Group_Cursor INTO @GroupID;
    END;

    CLOSE Group_Cursor;
    DEALLOCATE Group_Cursor;
END;
```

Триггер tr_ReplaceTeacherInSchedule:

```
CREATE TRIGGER tr_ReplaceTeacherInSchedule
ON Преподаватель
INSTEAD OF DELETE
AS
```

```

BEGIN
    SET NOCOUNT ON;

    CREATE TABLE #DeletedTeachers (
        ID_преподавателя INT
    );

    INSERT INTO #DeletedTeachers (ID_преподавателя)
    SELECT ID_преподавателя
    FROM DELETED;

    UPDATE Расписание_занятий
    SET ID_преподавателя = (
        SELECT TOP 1 ID_преподавателя
        FROM Преподаватель
        WHERE ID_преподавателя <> (SELECT ID_преподавателя FROM
#DeletedTeachers)
    )
    WHERE ID_преподавателя IN (SELECT ID_преподавателя FROM
#DeletedTeachers);

    DROP TABLE #DeletedTeachers;

    UPDATE Студент
    SET ID_научного_руководителя = NULL
    WHERE ID_научного_руководителя IN (SELECT ID_преподавателя
FROM deleted);

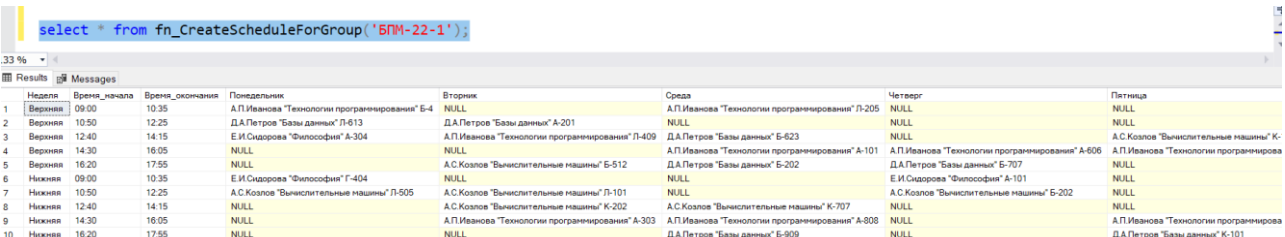
    DELETE FROM Преподаватель
    WHERE ID_преподавателя IN (SELECT ID_преподавателя FROM
DELETED);
END;

```

8. Пример работы с БД с использованием созданных объектов

1. Пусть куратор группы БПМ-22-1 перед началом семестра хочет получить расписание группы и список всех студентов этой группы.

Тогда куратор может воспользоваться функциями `fn_CreateScheduleForGroup` и `fn_GetStudentsByGroup`



The screenshot shows a SQL query window with the command `select * from fn_CreateScheduleForGroup('БПМ-22-1');` and its results. The results are displayed in a table with columns for Week, Day, Start Time, End Time, and the specific course and teacher for each day of the week.

Неделя	Время_начала	Время_окончания	Понедельник	Вторник	Среда	Четверг	Пятница
1	Верхняя	09:00	10:35	А.П.Иванова "Технологии программирования" Б-4	NULL	А.П.Иванова "Технологии программирования" Л-205	NULL
2	Верхняя	10:50	12:25	Д.А.Петров "Базы данных" Л-613	Д.А.Петров "Базы данных" А-201	NULL	NULL
3	Верхняя	12:40	14:15	Е.И.Сидорова "Философия" А-304	А.П.Иванова "Технологии программирования" Л-409	Д.А.Петров "Базы данных" Б-623	NULL
4	Верхняя	14:30	16:05	NULL	NULL	А.П.Иванова "Технологии программирования" А-101	А.П.Иванова "Технологии программирования" А-606
5	Верхняя	16:20	17:55	NULL	А.С.Козлов "Вычислительные машины" Б-512	Д.А.Петров "Базы данных" Б-202	Д.А.Петров "Базы данных" Б-707
6	Нижняя	09:00	10:35	Е.И.Сидорова "Философия" Г-404	NULL	NULL	Е.И.Сидорова "Философия" А-101
7	Нижняя	10:50	12:25	А.С.Козлов "Вычислительные машины" Л-505	А.С.Козлов "Вычислительные машины" Л-101	NULL	А.С.Козлов "Вычислительные машины" Б-202
8	Нижняя	12:40	14:15	NULL	А.С.Козлов "Вычислительные машины" К-202	А.С.Козлов "Вычислительные машины" К-707	NULL
9	Нижняя	14:30	16:05	NULL	А.П.Иванова "Технологии программирования" А-303	А.П.Иванова "Технологии программирования" А-808	А.П.Иванова "Технологии программирования" А-808
10	Нижняя	16:20	17:55	NULL	NULL	Д.А.Петров "Базы данных" Б-909	Д.А.Петров "Базы данных" К-101

Рис.2. Пример работы функции `fn_CreateScheduleForGroup`

`select * from fn_GetStudentsByGroup('БПМ-22-1');`

161 %

Results Messages

	ID_студента	Имя	Фамилия	Отчество	Дата_рождения	Электронная_почта	Название_группы
1	2200000	Мария	Игнатова	Алексеевна	1999-09-22	maria@example.com	БПМ-22-1
2	2200004	Иван	Петров	Александрович	1999-02-15	ivan@example.com	БПМ-22-1
3	2200005	Наталья	Сидорова	Васильевна	2000-11-03	natalia@example.com	БПМ-22-1
4	2200006	Денис	Кузнецов	Игоревич	1997-08-27	denis@example.com	БПМ-22-1
5	2200007	Александра	Федорова	Андреевна	2001-07-09	alexandra@example.com	БПМ-22-1
6	2200008	Максим	Иванов	Дмитриевич	1998-03-18	maxim@example.com	БПМ-22-1
7	2200009	Анна	Козлова	Игоревна	1999-05-21	anna@example.com	БПМ-22-1
8	2200010	Сергей	Титов	Сергеевич	2000-09-30	sergey@example.com	БПМ-22-1
9	2200011	Ольга	Макарова	Олеговна	1998-12-14	olga@example.com	БПМ-22-1
10	2200012	Дмитрий	Капустин	Павлович	2001-01-25	dmitriy@example.com	БПМ-22-1
11	2200013	Елена	Соловьева	Игоревна	1999-06-05	elena@example.com	БПМ-22-1
12	2200014	Артем	Воронцов	Андреевич	2000-08-17	artem@example.com	БПМ-22-1
13	2200015	Кристина	Исаева	Петровна	1997-04-12	kristina@example.com	БПМ-22-1
14	2200016	Роман	Кудрявцев	Алексеевич	2001-10-28	roman@example.com	БПМ-22-1
15	2200017	Юлия	Смирнова	Сергеевна	1998-07-02	yulia@example.com	БПМ-22-1
16	2200018	Владислав	Белкин	Васильевич	1999-11-14	vladislav@example.com	БПМ-22-1
17	2200019	Анастасия	Полякова	Игоревна	2000-02-19	anastasia@example.com	БПМ-22-1
18	2200020	Игорь	Родин	Дмитриевич	1997-09-23	igor@example.com	БПМ-22-1
19	2200021	Марина	Сергеева	Александровна	2001-12-31	marina@example.com	БПМ-22-1
20	2200022	Артемий	Куликов	Петрович	1998-06-17	artemiy@example.com	БПМ-22-1
21	2200023	Елена	Андреева	Владимировна	1999-03-22	elena@example.com	БПМ-22-1

Рис.3. Пример работы функции fn_GetStudentsByGroup

2. Пусть в начале семестра нужно зачислить нового студента из другого вуза. Также другой студент написал заявление о переводе в другую группу. Для добавления нового студента и перевода другого можно воспользоваться функциями pr_AddNewStudent и pr_TransferStudentToAnotherGroup.

При добавлении нового студента можно не указывать ID группы, процедура добавит студента в группу с наименьшим числом студентов. Это нужно для того, чтобы группы заполнялись равномерно.

`EXEC pr_AddNewStudent '2004.10.01', 'Иван', 'Иванов', 'Иванович';`

146 %

Messages

(1 row affected)

(1 row affected)

Студент успешно добавлен в 7 группу.

Рис.4. Пример работы процедуры pr_AddNewStudent

Допустим, что мы допустили ошибку при попытке перевода студента в другую группу – указали несуществующий ID студента. Тогда выведет сообщение о нашей ошибке.

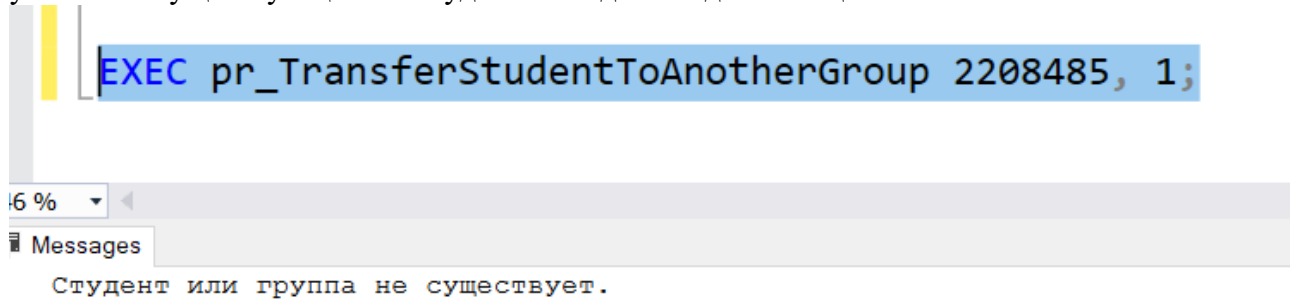


Рис.5. Пример работы процедуры `pr_TransferStudentToAnotherGroup` с ошибкой

Далее, выполним запрос с корректным ID студента. Увидим сообщение об успешном выполнении перевода.

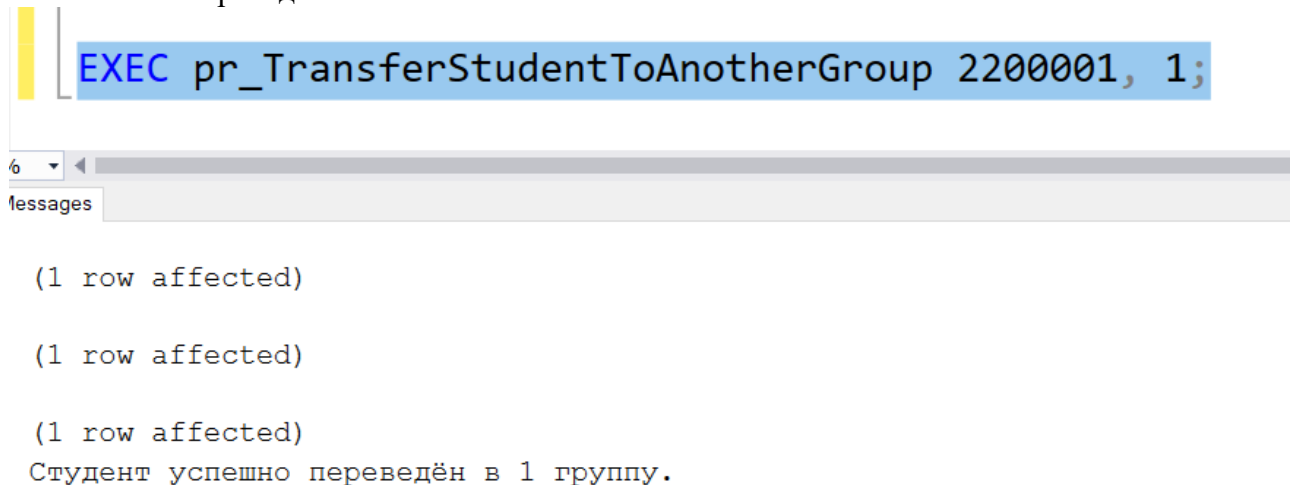


Рис.6. Пример работы процедуры `pr_TransferStudentToAnotherGroup`

9. Роли и пользователи

В базе данных создаются роли и пользователи, а также назначаются соответствующие права для каждой из ролей. Вот какие действия доступны каждой из ролей:

1. **Роль `db_admin`:**
 - Полномочия `CONTROL` (полный контроль) над базой данных.
 - Эта роль предполагается для администратора базы данных и предоставляет полные права на управление базой данных.
2. **Роль `db_user`:**
 - Права `SELECT`, `INSERT`, `UPDATE`, `DELETE`.
 - Эта роль предназначена для пользователей, в том числе менеджеров, которые могут выполнять основные операции чтения и записи данных.
3. **Роль `role_teacher`:**
 - Права `SELECT`, `INSERT`, `UPDATE`, `DELETE` на таблицу "Сотрудник".
 - Эта роль предполагается для преподавателей, которые имеют доступ к данным о сотрудниках и могут выполнять операции добавления, обновления и удаления.
4. **Роль `role_student`:**
 - Право `SELECT` на таблицу "Студент".

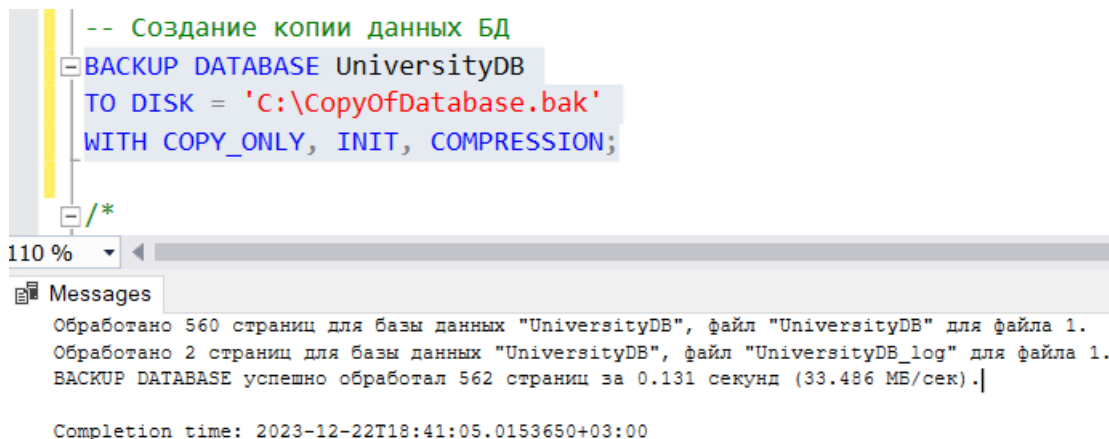
- Эта роль предназначена для студентов, которые имеют только право на чтение данных о студентах.
5. Роль **role_group_curator**:
- Права SELECT, UPDATE на таблицу "Группа".
 - Эта роль предполагается для кураторов групп, которые могут просматривать и обновлять информацию о группах.

10. Создание копий

```
-- Создание копии структуры БД
IF NOT EXISTS (SELECT name FROM sys.databases WHERE name =
'CopyOfDatabase')
BEGIN
    CREATE DATABASE CopyOfDatabase;
END;

-- Создание копии данных БД
BACKUP DATABASE UniversityDB
TO DISK = 'C:\CopyOfDatabase.bak'
WITH COPY_ONLY, INIT, COMPRESSION;
```

Результат выполнения команды создания копии данных БД:



```
-- Создание копии данных БД
BACKUP DATABASE UniversityDB
TO DISK = 'C:\CopyOfDatabase.bak'
WITH COPY_ONLY, INIT, COMPRESSION;
```

110 %

Messages

Обработано 560 страниц для базы данных "UniversityDB", файл "UniversityDB" для файла 1.
 Обработано 2 страниц для базы данных "UniversityDB", файл "UniversityDB_log" для файла 1.
 BACKUP DATABASE успешно обработал 562 страниц за 0.131 секунд (33.486 МБ/сек).|

Completion time: 2023-12-22T18:41:05.0153650+03:00

11. Список литературы

- Документация по Transact SQL Microsoft - <https://learn.microsoft.com/ru-ru/sql/t-sql/language-reference?view=sql-server-ver16>
- ChatGPT для генерации данных - <https://chat.openai.com>