Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інституті ім. Ігоря Сікорського”

Фізико-технічний інститут

**Розрахунково-графічна робота**

з предмету «Бази даних»

**Виконав:**

Студент 3 курсу,

ФТІ, групи ФБ-61

Вовчановський Павло

**Завдання на розрахунково-графічних робіт**

1. Промоделювати функціональні вимоги до системи, що проектується за допомогою DFD-діаграми, для чого:

* Ідентифікувати зовнішні об'єкти, що є постачальниками / приймачами інформації, і види інформації, які циркулюють між системою і зовнішніми об'єктами;
* Визначити основні функціональні вимоги до системи і представити їх у вигляді процесів;
* Визначити основні види інформації і представити їх у вигляді потоків між процесами і зовнішніми об'єктами;
* Задати сховища інформації.

Для побудови DFD-діаграми використовувати інструментальний засіб CASE Studio 2.

2. Побудувати концептуальну модель (ER-модель), для чого:

* За описом предметної області і поставленому завданню виявити необхідний набір сутностей, їх властивості і зв'язки між сутностями;
* Визначити необхідний набір атрибутів для кожної сутності, виділивши атрибути що ідентифікують сутність;
* Класифікувати сутності (стрижневі, асоціативні та інші);
* Визначити зв'язки між об'єктами;
* Задати обмеження цілісності;
* Описати отриману ER-модель.

Для побудови ER-моделі використовувати інструментальний засіб CASE Studio 2.

3. Отримати реляційну схему з ER-моделі, для чого:

* Побудувати набір необхідних таблиць бази даних;
* Виділити первинні та зовнішні ключі певних таблиць;
* Привести отримані таблиці до третьої нормальної форми.
* Дати математичний опис отриманої моделі з використанням апарату реляційної алгебри.

4. Використовуючи СУБД MS SQL Server створити спроектовану базу даних.

5. На мові SQL записати вирази для зазначених у варіанті завдання типів запитів. Перевірити працездатність написаних запитів в інтерактивному режимі.

6. Реалізувати закінчений додаток, що працює з створеною базою даних. Додаток має

* Заносити інформацію в таблиці створеної бази даних,;
* Виконувати необхідні дії по модифікації і видалення даних в таб-лицях створеної бази даних;
* Підтримувати цілісність бази даних;
* Виконувати запити з варіанта завдання.

**Варіант № 3**

**Предметна область** – Деканат.

**Задачі, що вирішує інформаційна система** – результати сесії.

**Завдання на обробку даних:**

1. Створити впорядковані списки:

- Студентів груп 'С' другого курсу (за прізвищами);

- Студентів, що мають максимальний середній бал у своїй групі (за середнім балом та прізвища);

- Дисциплін, за якими немає іспитів (за алфавітом).

2. обчислення:

- Обчислити стипендіальний фонд по групі, по курсу, факультету.

3. корекція:

- Зміна оцінки з дисципліни для обраного студента.

4. Звіти виду:

- "Студенти": факультет - курс - група - ПІБ - номер заліковки - середня оцінка.

- "Кількість студентів": факультет - курс - кількість студентів (за денною формою)

- "Кількість студентів": кількість студентів - форма навчання.

- "Успішність": група - середній бал по групі-різниця (бал - середній бал). У запиті впорядкувати за збільшення різниці.

**Виконання роботи**

**1. Побудова DFD-діаграми**

Відділ якості освіти та аналітики

Приймальна комісія

Методичний відділ

Викладач

Стипендіальнийвідділ

Додавання студента

Формування звітів

Внесення навчального плану

Внесення розмірів стипендій

Призначення стипендій

Внесення результатів

БД «Результати сесії»

Навчальний відділ

Отримання відомостей

Результати сесії

Результати сесії

Навчальний план

Навчальний план

Персональні дані студента

Персональні дані студента

Розмір стипендій

Розмір стипендій

Результати сесії

Протокол призначення

п

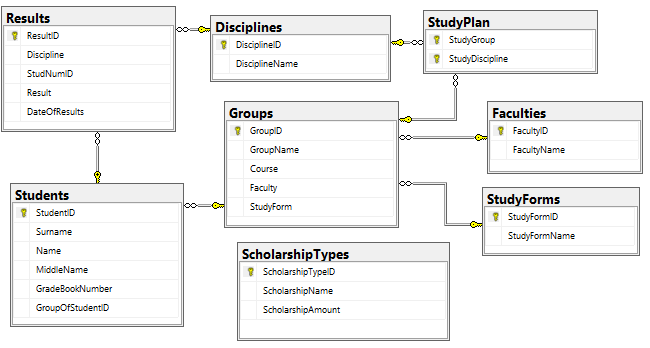
Результати сесії

Результати сесії

Відомість

Звіти

**2. Побудова ER-діаграми**

****

**3. Побудова реляційної моделі**

Перевіримо побудовану базу даних на відповідність до нормальних форм.

1НФ: Усі атрибути в нашій базі атомарні і не повторюються, відсутні групи які повторюються. Це означає що наша база даних знаходиться в 1 нормальній формі.

2НФ: В базі відсутні неповні функціональні залежності не ключових атрибутів первинного ключа, тобто будь-яке не ключове поле однозначно ідентифікується повним набором ключів. В нашій базі кожне поле унікальне і визначається набором ключів. База знаходиться в 2 нормальній формі.

3НФ: Перевіримо наявність транзитивних відношень в базі. З огляду на ER діаграму бачимо що у нас немає ніяких транзитивних залежностей між атрибутами та таблицями. (ні одне з ключових полів не ідентифікується за допомогою іншого ключового поля) База знаходиться в 3 нормальній формі.

Після проведеного аналізу було побудовано його математичну модель, що складається з 8 сутностей.

* Faculties(FacultyID INT IDENTITY PRIMARY KEY,FacultyName VARCHAR(70) NOT NULL UNIQUE);
* StudyForms(StudyFormID INT IDENTITY PRIMARY KEY, StudyFormName VARCHAR(20) NOT NULL UNIQUE);
* Groups(GroupID INT IDENTITY PRIMARY KEY, GroupName VARCHAR(4) NOT NULL UNIQUE, Course INT NOT NULL, Faculty INT NOT NULL, StudyForm INT NOT NULL);
* Students(StudentID INT IDENTITY PRIMARY KEY, Surname VARCHAR(15) NOT NULL, Name VARCHAR(15) NOT NULL, MiddleName VARCHAR(15) NOT NULL, GradeBookNumber VARCHAR(6) NOT NULL UNIQUE, GroupOfStudentID INT NOT NULL,
* Disciplines(DisciplineID INT IDENTITY PRIMARY KEY, DisciplineName VARCHAR(30) NOT NULL);
* Results(ResultID INT IDENTITY PRIMARY KEY, Discipline INT NOT NULL, StudNumID INT NOT NULL, Result INT NOT NULL, DateOfResults DATE NOT NULL);
* ScholarshipTypes(ScholarshipTypeID INT PRIMARY KEY, ScholarshipName VARCHAR(20) NOT NULL UNIQUE, ScholarshipAmount INT NOT NULL);
* StudyPlan(StudyGroup INT NOT NULL, StudyDiscipline INT NOT NULL, PRIMARY KEY (StudyGroup, StudyDiscipline));

# **4. Створення бази даних та обмежень за допомогою SQL-команд:**

CREATE TABLE Faculties(

FacultyID INT IDENTITY PRIMARY KEY,

FacultyName VARCHAR(70) NOT NULL UNIQUE,

);

CREATE TABLE StudyForms(

StudyFormID INT IDENTITY PRIMARY KEY,

StudyFormName VARCHAR(20) NOT NULL UNIQUE,

);

CREATE TABLE Groups(

GroupID INT IDENTITY PRIMARY KEY,

GroupName VARCHAR(4) NOT NULL UNIQUE,

Course INT NOT NULL,

Faculty INT NOT NULL,

StudyForm INT NOT NULL,

CONSTRAINT CourseCheck CHECK(Course between 1 and 6),

CONSTRAINT fk\_faculty FOREIGN KEY (Faculty)

REFERENCES Faculties(FacultyID) ON DELETE CASCADE,

CONSTRAINT fk\_studyform FOREIGN KEY (StudyForm)

REFERENCES StudyForms(StudyFormID) ON DELETE CASCADE,

);

CREATE TABLE Students(

StudentID INT IDENTITY PRIMARY KEY,

Surname VARCHAR(15) NOT NULL,

Name VARCHAR(15) NOT NULL,

MiddleName VARCHAR(15) NOT NULL,

GradeBookNumber VARCHAR(6) NOT NULL UNIQUE,

GroupOfStudentID INT NOT NULL,

CONSTRAINT fk\_group\_studID FOREIGN KEY (GroupOfStudentID)

REFERENCES Groups(GroupID) ON DELETE CASCADE,

);

CREATE TABLE Disciplines(

DisciplineID INT IDENTITY PRIMARY KEY,

DisciplineName VARCHAR(30) NOT NULL,

);

CREATE TABLE Results(

ResultID INT IDENTITY PRIMARY KEY,

Discipline INT NOT NULL,

StudNumID INT NOT NULL,

Result INT NOT NULL,

DateOfResults DATE NOT NULL,

CONSTRAINT ResultCheck check(Result between 0 and 100),

CONSTRAINT DateOfResultCheck CHECK (DateOfResults BETWEEN '2000.01.01' AND GETDATE()),

CONSTRAINT uc\_ResultData UNIQUE(StudNumID, DateOfResults, Discipline),

CONSTRAINT fk\_stud\_numID FOREIGN KEY (StudNumID)

REFERENCES Students(StudentID) ON DELETE CASCADE,

CONSTRAINT fk\_discipline\_ID FOREIGN KEY (Discipline)

REFERENCES Disciplines(DisciplineID) ON DELETE CASCADE,

);

CREATE TABLE ScholarshipTypes(

ScholarshipTypeID INT PRIMARY KEY,

ScholarshipName VARCHAR(20) NOT NULL UNIQUE,

ScholarshipAmount INT NOT NULL,

);

CREATE TABLE StudyPlan(

StudyGroup INT NOT NULL,

StudyDiscipline INT NOT NULL,

PRIMARY KEY (StudyGroup, StudyDiscipline),

CONSTRAINT fk\_group\_studyplan FOREIGN KEY (StudyGroup)

REFERENCES Groups(GroupID) ON DELETE CASCADE,

CONSTRAINT fk\_discipline\_studyplan FOREIGN KEY (StudyDiscipline)

REFERENCES Disciplines(DisciplineID) ON DELETE CASCADE,

););

Проаналізувавши сутності системи, я визначив, які обмеження на дані потрібно вводити та створив необхідні тригери на вставку та оновлення недопустимих кортежів, для забезпечення цілісності даних в таблиці Results:

CREATE TRIGGER StudyCHK ON Results

FOR INSERT, UPDATE

AS

IF NOT EXISTS(SELECT 'True' FROM StudyPlan WHERE CONCAT(StudyDiscipline, StudyGroup)=(SELECT CONCAT(Discipline, GroupOfStudentID) FROM INSERTED INNER JOIN Students ON inserted.StudNumID=Students.StudentID))

BEGIN

ROLLBACK TRAN

RAISERROR('Дана дисципліна не передбачена у навчальному плані студента!', 16, 1)

END

# **Завдання розрахункової:**

1. Створити впорядковані списки:

- Студентів груп 'С' другого курсу (за прізвищами):

CREATE VIEW CStudentListView

AS

SELECT CONCAT(Surname,' ', Name, ' ', MiddleName) AS 'ПІБ' From Students WHERE GroupOfStudentID IN (SELECT GroupID FROM Groups WHERE Course=2 AND GroupName LIKE 'С%');

GO

SELECT \* FROM CStudentListView ORDER BY 'ПІБ'

- Студентів, що мають максимальний середній бал у своїй групі (за середнім балом та прізвища):

CREATE VIEW MaxInGroupView

AS

WITH AvgResults AS (SELECT StudNumID, AVG(Result) AS AvgRes FROM Results GROUP BY StudNumID)

SELECT CONCAT(Surname,' ', Name, ' ', MiddleName) AS 'ПІБ', GroupName, AvgRes FROM Students

INNER JOIN Groups ON Students.GroupOfStudentID=Groups.GroupID

INNER JOIN AvgResults ON Students.StudentID=AvgResults.StudNumID

WHERE EXISTS(SELECT \* FROM (SELECT GroupOfStudentID , MAX(AvgRes) AS MaxRes FROM Students

INNER JOIN AvgResults ON Students.StudentID=AvgResults.StudNumID

GROUP BY GroupOfStudentID) GroupMax

WhERE Students.GroupOfStudentID=GroupMax.GroupOfStudentID AND AvgResults.AvgRes=GroupMax.MaxRes);

GO

SELECT \* FROM MaxInGroupView ORDER BY AvgRes, 'ПІБ';

- Дисциплін, за якими немає іспитів (за алфавітом):

CREATE VIEW DisciplineNoExamsVIew

AS

SELECT DisciplineName FROM Disciplines WHERE DisciplineID NOT IN (SELECT StudyDiscipline FROM StudyPlan);

GO

SELECT \* From DisciplineNoExamsView ORDER BY DisciplineName;2. обчислення:

- Обчислити стипендіальний фонд по групі, по курсу, факультету:

CREATE VIEW ScholarsView

AS

WITH ScholarsDef AS (SELECT StudNumID FROM Results

GROUP BY StudNumID

HAVING SUM(CASE WHEN Result>=60 THEN 1 ELSE 0 END)=(SELECT COUNT(\*) FROM StudyPlan

INNER JOIN Students ON StudyPlan.StudyGroup=Students.GroupOfStudentID

WHERE StudentID=StudNumID) AND AVG(Result)>=75),

ScholarsAdv AS (SELECT StudNumID FROM Results

GROUP BY StudNumID

HAVING SUM(CASE WHEN Result>=95 THEN 1 ELSE 0 END)=(SELECT COUNT(\*) FROM StudyPlan

INNER JOIN Students ON StudyPlan.StudyGroup=Students.GroupOfStudentID

WHERE StudentID=StudNumID))

SELECT StudNumID, (CASE WHEN StudNumID IN (SELECT \* FROM ScholarsAdv) THEN (SELECT ScholarshipAmount FROM ScholarshipTypes WHERE ScholarshipTypeID=2)

ELSE (SELECT ScholarshipAmount FROM ScholarshipTypes WHERE ScholarshipTypeID=1) END)

AS Scholarship FROM ScholarsDef

GO

CREATE VIEW MoneyByGroupView

AS

SELECT GroupName, SUM(Scholarship) AS 'Стипендіальний фонд' FROM ScholarsView

INNER JOIN Students ON ScholarsView.StudNumID=Students.StudentID

INNER JOIN Groups ON Students.GroupOfStudentID=Groups.GroupID

GROUP BY GroupName

GO

CREATE VIEW MoneyByCourseView

AS

SELECT Course, SUM(Scholarship) AS 'Стипендіальний фонд' FROM ScholarsView

INNER JOIN Students ON ScholarsView.StudNumID=Students.StudentID

INNER JOIN Groups ON Students.GroupOfStudentID=Groups.GroupID

GROUP BY Course

GO

CREATE VIEW MoneyByFacultyView

AS

SELECT FacultyName, SUM(Scholarship) AS 'Стипендіальний фонд' FROM ScholarsView

INNER JOIN Students ON ScholarsView.StudNumID=Students.StudentID

INNER JOIN Groups ON Students.GroupOfStudentID=Groups.GroupID

INNER JOIN Faculties ON Groups.GroupID=Faculties.FacultyID

GROUP BY FacultyName

3. корекція:

- Зміна оцінки з дисципліни для обраного студента.

CREATE PROC AddResult

@Discipline INT,

@GradeBookNumber VARCHAR(6),

@Result INT,

@DateOfResults DATE = NULL

AS

SET @DateOfResults=ISNULL(@DateOfResults, GETDATE());

IF EXISTS (SELECT \* FROM Results WHERE Discipline=@Discipline AND StudNumID=(SELECT StudentID From Students WHERE GradeBookNumber=@GradeBookNumber))

UPDATE Results SET Result=@Result, DateOfResults=@DateOfResults WHERE Discipline=@Discipline AND StudNumID=(SELECT StudentID From Students WHERE GradeBookNumber=@GradeBookNumber);

ELSE

INSERT INTO Results(Discipline, StudNumID, Result, DateOfResults) VALUES (@Discipline, (SELECT StudentID From Students WHERE GradeBookNumber=@GradeBookNumber), @Result, @DateOfResults);

4. Звіти виду:

- "Студенти": факультет - курс - група - ПІБ - номер заліковки - середня оцінка.

CREATE VIEW StudentsView

AS

WITH AvgResults AS (SELECT StudNumID, AVG(Result) AS AvgRes FROM Results GROUP BY StudNumID)

SELECT FacultyName, Course, GroupName, CONCAT(Surname,' ', Name, ' ', MiddleName) AS 'ПІБ', GradeBookNumber, AvgRes FROM Students

INNER JOIN Groups ON Students.GroupOfStudentID=Groups.GroupID

INNER JOIN Faculties ON Groups.Faculty=Faculties.FacultyID

INNER JOIN AvgResults ON Students.StudentID=AvgResults.StudNumID

GO

SELECT \* FROM StudentsView

- "Кількість студентів": факультет - курс - кількість студентів (за денною формою)

CREATE VIEW StudentsNumber

AS

SELECT FacultyName, Course, COUNT(\*) as NumberOfStundents FROM Students

INNER JOIN Groups ON Students.GroupOfStudentID=Groups.GroupID

INNER JOIN Faculties ON Groups.Faculty=Faculties.FacultyID

WHERE Groups.StudyForm=(SELECT StudyFormID FROM StudyForms WHERE StudyFormName='денна')

GROUP BY FacultyName, Course;

GO

SELECT \* FROM StudentsNumber

- "Кількість студентів": кількість студентів - форма навчання.

CREATE VIEW StudentsNumberbyStudyFormView

AS

SELECT StudyFormName, COUNT(\*) as NumberOfStundents FROM StudyForms

INNER JOIN Groups ON StudyForms.StudyFormID=Groups.StudyForm

INNER JOIN Students ON Groups.GroupID=Students.GroupOfStudentID

GROUP BY StudyFormName;

GO

SELECT \* FROM StudentsNumberbyStudyFormView

- "Успішність": група - середній бал по групі-різниця (бал - середній бал). У запиті впорядкувати за збільшення різниці.

CREATE PROC Progress

@Result INT=NULL

AS

SET @Result = ISNULL(@Result, (SELECT AVG(Result) From Results))

SELECT GroupName, AVG(Result), @Result-AVG(Result) AS 'Різниця' FROM Results

INNER JOIN Students ON Results.StudNumID=Students.StudentID

INNER JOIN Groups ON Students.GroupOfStudentID=Groups.GroupID

GROUP BY GroupName

ORDER BY 'Різниця'

GO

EXEC Progress