

Національний технічний університет України

“Київський політехнічний інститут”

БАЗИ ДАНИХ ТА ІНФОРМАЦІЙНІ СИСТЕМИ

**Розрахункова робота на тему:**

«Проектування предметної області - деканату»

Виконав:

Студент гр. ФІ-91

Варіант 4

Бурдун Вячелав

Перевірив:

Коломицев М.В.

Київ 2012

**Зміст**

[Вступ 3](#_Toc343702633)

[1. Проектування баз даних 4](#_Toc343702634)

[1.1. Системний аналіз предметної області 4](#_Toc343702635)

[1.2. Концептуальне проектування 6](#_Toc343702636)

[1.3. Логічне проектування 7](#_Toc343702637)

[2. Завдання 8](#_Toc343702638)

[3. Створення діаграми потоків даних (DFD-моделі) 9](#_Toc343702639)

[4. Створення концептуальної моделі (EF-моделі) 12](#_Toc343702640)

[5. Проектування бази даних у середовищі MS SQL 13](#_Toc343702641)

[6. Створення необхідного функціонування 19](#_Toc343702642)

[6.1. Створення впорядкованих списків 19](#_Toc343702643)

[6.2. Перевірка, що в кожного викладача не більше трех пар в один день 20](#_Toc343702644)

[6.3. Корегування даних 20](#_Toc343702645)

[6.4. Створення звіту 21](#_Toc343702646)

[7. Перевірка нормалізації розробленої моделі 25](#_Toc343702647)

[Висновок 26](#_Toc343702648)

# Вступ

Основною метою даної розрахункової роботи є освоєння навичок проектування баз даних, в частковому випадку бази даних постояльців у готелі.

Етап проектування даних зазвичай відбувається з участю потенційних користувачів. Іноді в процесі проектування даних створюються прототипи працюючих додатків, що дозволяють уточнити і доповнити вимоги до кінцевого продукту. Якщо передбачається, що впровадження нової інформаційної системи буде супроводжуватися виведенням з експлуатації її попередника, то приймаються рішення про те, яким чином використовувати успадковані дані, і в модель даних, що є підсумком цього етапу, вносяться необхідні зміни.

Автори багатьох робіт, присвячених загальним принципам розробки проектів інформаційних систем, стверджують, що вартість виправлення помилки, допущеної на попередньому етапі життєвого циклу, приблизно в десять разів перевищує витрати на її виправлення на поточному етапі. Зокрема, багато розробників стикаються з тим, що помилки проектування даних приводять іноді до написання коду великого обсягу, так чи інакше їх компенсуючого, і нерідко викликають проблеми на етапі супроводу готового продукту. Оскільки проектування даних слід безпосередньо за перед проектних обстеженням, дуже важливо, щоб ця частина роботи над проектом була виконана максимально якісно. Саме важливість цього етапу зумовила стрімке зростання популярності такої категорії програмного забезпечення, як засоби проектування даних, протягом останніх десяти років.

# Проектування баз даних

Проектування **бази даних** - це ітераційний, багатоетапний процес прийняття аргументованих рішень під час аналізу інформаційної моделі предметної області, вимог до даних з боку прикладних програмістів і користувачів, синтезу логічних і фізичних структур даних, аналізу та обґрунтування вибору програмних та апаратних засобів.

Проектування виконують за допомогою вивчення ПЗ та вимог, які ставляться до ІС. На цій «паперовій» стадії життя системи вибирають:

* структуру даних і стратегію їх зберігання у пам'яті ЕОМ;
* технологію обслуговування ІС та взаємодію з нею кінцевих користувачів;
* технічні та стандартні програмні засоби; а також розробку оригінальних програмних.

**Етапи проектування БД:**

* системний аналіз предметної області;
* концептуальне проектування;
* логічне проектування;
* фізичне проектування.

## Системний аналіз предметної області

**Системний аналіз** передбачає мовний опис реальних об'єктів предметної області, визначення зв'язків між об'єктами, дослідження характеристик об'єктів і зв'язків. Результати дослідження використовуються при концептуальному проектуванні БД.

Для визначення складу і структури предметної області застосовуються або функціональний, або предметний підходи.

***Функціональний підхід*** застосовує рух "від задач" і використовується у тих випадках, коли заздалегідь відомі функції майбутніх користувачів БД, а також відомі всі задачі, для інформаційних потреб яких створюються БД. В цьому випадку на основі виробничих документів, опитувань замовників можна чітко визначити мінімальний набір об'єктів предметної області та їх взаємозв'язок.

***Предметний підхід*** застосовується у тому випадку, коли інформаційні потреби майбутніх користувачів чітко не визначені. В цьому випадку не можна чітко визначити мінімальний набір об'єктів предметної області. В опис предметної області включаються об'єкти та зв'язки, які є найбільш характерними та найбільш суттєвими для неї. БД називається предметною і може використовуватися для розв’язання задач, які заздалегідь не визначені.

У практичній діяльності використовується комплексний підхід, який з одного боку дозволяє розв’язувати конкретні інформаційні та функціональні задачі, а з іншого боку – враховує можливість додавання нових застосувань. У загальному випадку існує два підходи до проектування БД: низхідне проектування і висхідне проектування.

***Низхідне проектування*** починається з визначення наборів даних, потім визначаються елементи даних для кожного з таких наборів. Цей процес включає в себе ідентифікацію різних типів сутностей і визначення атрибутів кожної сутності.

Низхідне проектування включає операції декомпозиції, що передбачає заміну вихідної множини відношень, що входять в схему БД, іншою множиною відношень, які є проекціями вихідних відношень. Цей підхід рекомендується застосовувати у тих випадках, коли кількість, різноманітність та складність сутностей, зв'язків і транзакцій значна за розмірами. Найбільш поширеними моделями для цього проектування є моделі "сутність − зв'язок" (ER-моделі, Entity-Relationship model).

***Висхідне проектування*** починається з виявлення елементів даних, які потім групуються в набори даних. Спочатку визначаються атрибути, які потім об'єднуються в сутності. Висхідне проектування включає операції синтезу, що передбачає виконання компоновки із заданої множини функціональних залежностей між об'єктами предметної області вихідних відношень схеми БД.

Цей підхід рекомендується застосовувати у тому випадку, якщо розробляється невелика БД з незначною кількістю об'єктів, атрибутів і транзакцій.

## Концептуальне проектування

З **концептуального проектування** починається створення концептуальної схеми БД, в основі якої лежить концептуальна модель даних.

***Концептуальна модель*** представляє загальний погляд на дані. Розрізняють два головних підходи до моделювання даних при концептуальному проектуванні:

− семантичні моделі;

− об'єктні моделі.

*Семантичні моделі* головну увагу приділяють структурі даних. Найбільш поширеною семантичною моделлю є модель *"сутність – зв'язок"* (Entity Relationship model, ER-модель). ER-модель складається із сутностей, зв'язків, атрибутів, доменів атрибутів, ключів. Моделювання даних відображає логічну структуру даних, так само, як блок-схеми алгоритмів відображають логічну структуру програми.

*Об'єктні моделі* головну увагу приділяють поведінці об'єктів даних і засобам маніпуляції даними. Головне поняття таких моделей − об'єкт, тобто сутність, яка має стан і поведінку. Стан об'єкта визначається сукупністю його атрибутів, а поведінка об'єкта визначається сукупністю операцій специфікованих для нього.

Зближення цих моделей реалізується в розширеному ER-моделюванні (Extended Entity Relationship model, EER-модель).

## Логічне проектування

Логічне проектування полягає в створенні логічної моделі на основі вибраної моделі даних. На цьому етапі необхідно вже знати яка СУБД буде застосовуватися в системі (ієрархічна, мережна, реляційна, об'єктно-орієнтована). Для перевірки вірності логічної моделі застосовується нормалізація. Крім того логічна модель перевіряється на умову забезпечення всіх транзакцій користувачів. Фізичне проектування полягає в описі засобів фізичної реалізації логічного проекту БД. Фізичні моделі визначають засоби розміщення даних в середовищі зберігання і засоби доступу до цих даних, які підтримуються на фізичному рівні.

У процесі проектування визначається структура реляційної БД (склад таблиць, їх структура і логічні зв'язки). Структура таблиці визначається складом стовпців, типом даних і розмірами стовпців, ключами таблиці.

# Завдання

**Предметная область** – Деканат.

**Решаемые задачи** – учебная нагрузка преподавателей.

**Задание:**

1. Создать упорядоченные по кафедрам списки :

* кафедра – преподаватель – дисциплина;
* преподавателей, у которых должность не соответствует учёной степени (для кандидата – доцент, для доктора – профессор).

1. Вычисления:

Убедиться, что у каждого преподавателя не более 3-х занятий в день.

1. Коррекция:

изменение расписания для преподавателя (аудитория, день недели, время).

1. Отчеты вида:

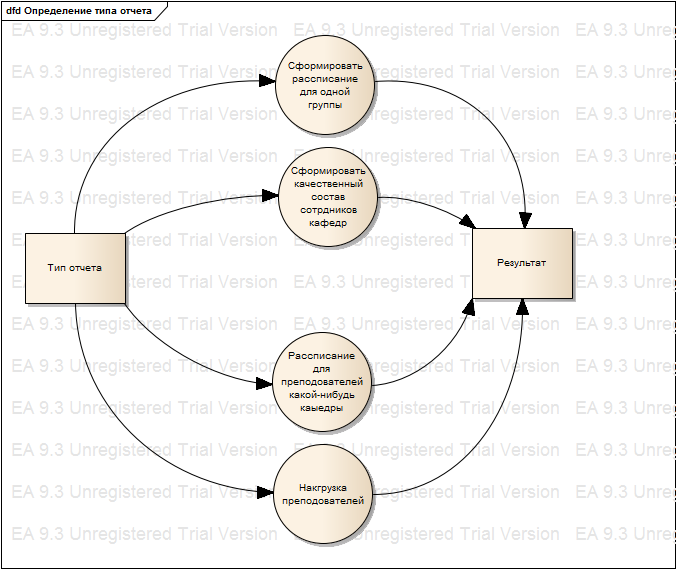
* Расписание занятий для одной произвольной группы.
* Качественный состав сотрудников кафедр": кафедра – количество профессоров – количество доцентов – количество старших преподавателей;
* Расписание для преподавателей кафедры «Х».
* "Нагрузка преподавателей": кафедра – преподаватель – количество дисциплины, которые он ведёт и количество часов по этим дисциплинам

# Створення діаграми потоків даних (DFD-моделі)

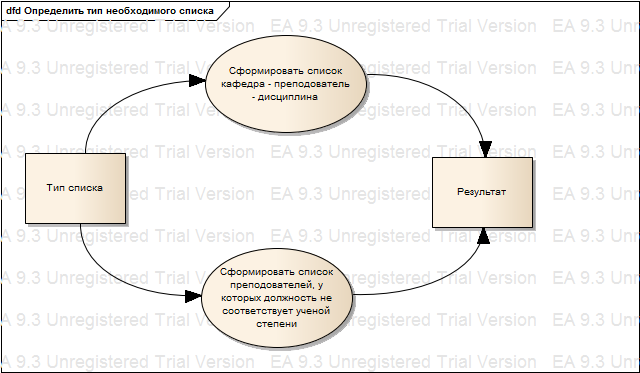
Після аналізу завдання, розробляємо можливу діаграму потоків даних. Для створення DFD-діаграми будемо користуватися програмою Enterprise Architect 9.

Для більш компактного і структурованого проектування визначемо декілька підсистем.

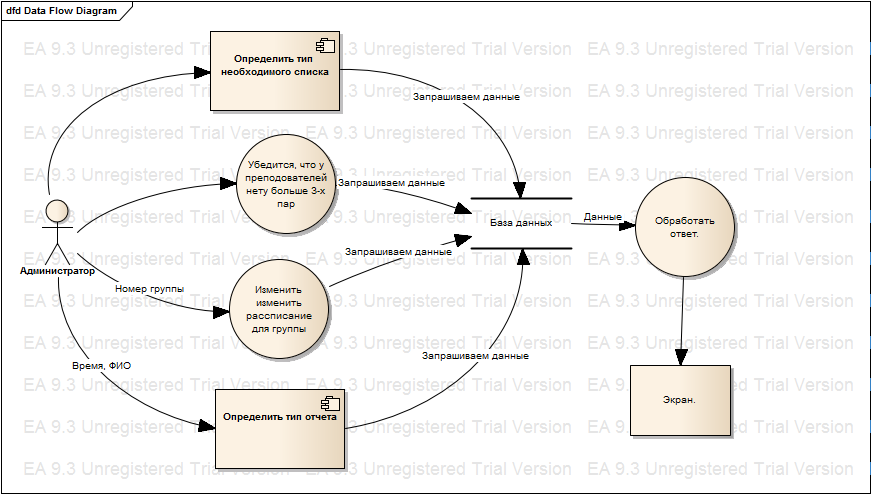
Результат роботи можно побачити на мал. 1,2,3.



Мал. 1. Підсистема «Визначення типу звіту».



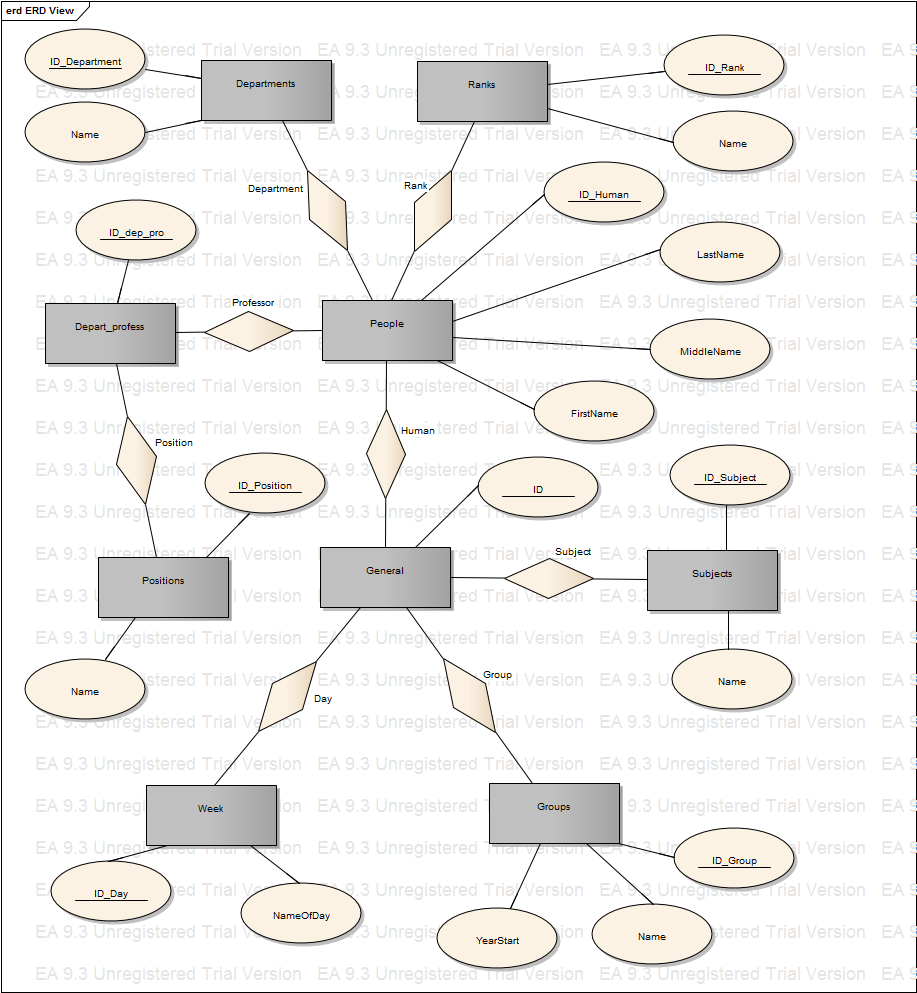
Мал. 2. Підсистема «Визначення типу необхідного списку».



Мал. 3. Головна діаграма потоків даних (DFD).

# Створення концептуальної моделі (EF-моделі)

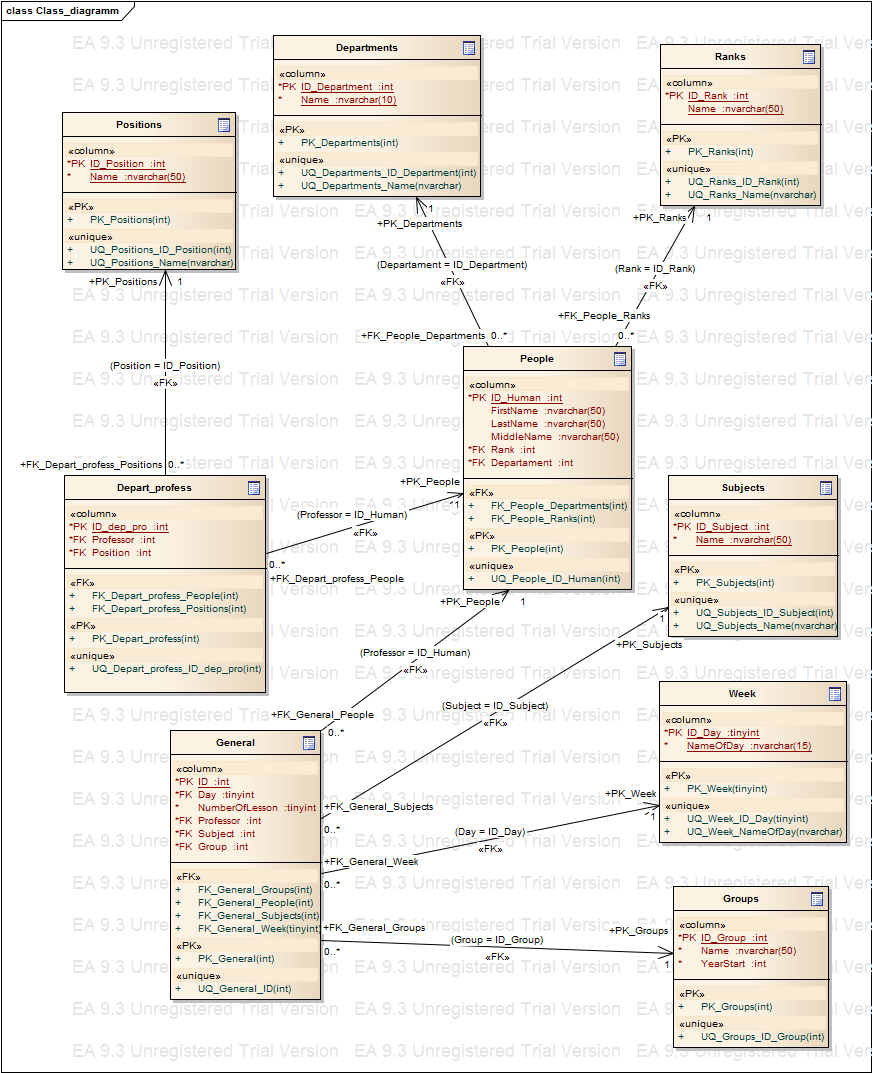
Проаналізувавши діаграму потоків даних, а також мету проектування і засоби для створення, будуємо концептуальну модель (мал. 4).



Мал. 4. Концептуальна модель даних

# Проектування бази даних у середовищі MS SQL

Проаналізувавши усі діаграми, а також предметну область, побудуємо діаграму класів, яка однозначно задасть сутності, атрибути і ключі для MS SQL бази даних (мал. 6).



Мал. 6. Діаграма класів

Виходячи з цього, SQL-код для створення потрібних сутностей буде:

CREATE TABLE Depart\_profess (

ID\_dep\_pro int NOT NULL,

Departament int NOT NULL,

Professor int NOT NULL,

Position int NOT NULL);

CREATE TABLE Departments (

ID\_Department int NOT NULL,

Name nvarchar(10) NOT NULL);

CREATE TABLE General (

ID int NOT NULL,

Day tinyint NOT NULL,

NumberOfLesson tinyint NOT NULL,

Professor int NOT NULL,

Subject int NOT NULL,

Group int NOT NULL);

CREATE TABLE Groups (

ID\_Group int NOT NULL,

Name nvarchar(50) NOT NULL,

YearStart int NOT NULL);

CREATE TABLE People (

ID\_Human int NOT NULL,

FirstName nvarchar(50),

LastName nvarchar(50),

MiddleName nvarchar(50),

Rank int NOT NULL,

Departament int NOT NULL);

CREATE TABLE Positions (

ID\_Position int NOT NULL,

Name nvarchar(50) NOT NULL);

CREATE TABLE Ranks (

ID\_Rank int NOT NULL,

Name nvarchar(50));

CREATE TABLE Subjects (

ID\_Subject int NOT NULL,

Name nvarchar(50) NOT NULL);

CREATE TABLE Week (

ID\_Day tinyint NOT NULL,

NameOfDay nvarchar(15) NOT NULL);

ALTER TABLE Depart\_profess

ADD CONSTRAINT UQ\_Depart\_profess\_ID\_dep\_pro UNIQUE (ID\_dep\_pro);

ALTER TABLE Departments

ADD CONSTRAINT UQ\_Departments\_ID\_Department UNIQUE (ID\_Department);

ALTER TABLE Departments

ADD CONSTRAINT UQ\_Departments\_Name UNIQUE (Name);

ALTER TABLE General

ADD CONSTRAINT UQ\_General\_ID UNIQUE (ID);

ALTER TABLE Groups

ADD CONSTRAINT UQ\_Groups\_ID\_Group UNIQUE (ID\_Group);

ALTER TABLE People

ADD CONSTRAINT UQ\_People\_ID\_Human UNIQUE (ID\_Human);

ALTER TABLE Positions

ADD CONSTRAINT UQ\_Positions\_ID\_Position UNIQUE (ID\_Position);

ALTER TABLE Positions

ADD CONSTRAINT UQ\_Positions\_Name UNIQUE (Name);

ALTER TABLE Ranks

ADD CONSTRAINT UQ\_Ranks\_ID\_Rank UNIQUE (ID\_Rank);

ALTER TABLE Ranks

ADD CONSTRAINT UQ\_Ranks\_Name UNIQUE (Name);

ALTER TABLE Subjects

ADD CONSTRAINT UQ\_Subjects\_ID\_Subject UNIQUE (ID\_Subject);

ALTER TABLE Subjects

ADD CONSTRAINT UQ\_Subjects\_Name UNIQUE (Name);

ALTER TABLE Week

ADD CONSTRAINT UQ\_Week\_ID\_Day UNIQUE (ID\_Day);

ALTER TABLE Week

ADD CONSTRAINT UQ\_Week\_NameOfDay UNIQUE (NameOfDay);

ALTER TABLE Depart\_profess ADD CONSTRAINT PK\_Depart\_profess

PRIMARY KEY CLUSTERED (ID\_dep\_pro);

ALTER TABLE Departments ADD CONSTRAINT PK\_Departments

PRIMARY KEY CLUSTERED (ID\_Department);

ALTER TABLE General ADD CONSTRAINT PK\_General

PRIMARY KEY CLUSTERED (ID);

ALTER TABLE Groups ADD CONSTRAINT PK\_Groups

PRIMARY KEY CLUSTERED (ID\_Group);

ALTER TABLE People ADD CONSTRAINT PK\_People

PRIMARY KEY CLUSTERED (ID\_Human);

ALTER TABLE Positions ADD CONSTRAINT PK\_Positions

PRIMARY KEY CLUSTERED (ID\_Position);

ALTER TABLE Ranks ADD CONSTRAINT PK\_Ranks

PRIMARY KEY CLUSTERED (ID\_Rank);

ALTER TABLE Subjects ADD CONSTRAINT PK\_Subjects

PRIMARY KEY CLUSTERED (ID\_Subject);

ALTER TABLE Week ADD CONSTRAINT PK\_Week

PRIMARY KEY CLUSTERED (ID\_Day);

ALTER TABLE Depart\_profess ADD CONSTRAINT FK\_Depart\_profess\_Departments

FOREIGN KEY (Departament) REFERENCES Departments (ID\_Department)

ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Depart\_profess ADD CONSTRAINT FK\_Depart\_profess\_People

FOREIGN KEY (Professor) REFERENCES People (ID\_Human)

ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Depart\_profess ADD CONSTRAINT FK\_Depart\_profess\_Positions

FOREIGN KEY (Position) REFERENCES Positions (ID\_Position)

ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE General ADD CONSTRAINT FK\_General\_Groups

FOREIGN KEY (Group) REFERENCES Groups (ID\_Group)

ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE General ADD CONSTRAINT FK\_General\_People

FOREIGN KEY (Professor) REFERENCES People (ID\_Human)

ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE General ADD CONSTRAINT FK\_General\_Subjects

FOREIGN KEY (Subject) REFERENCES Subjects (ID\_Subject)

ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE General ADD CONSTRAINT FK\_General\_Week

FOREIGN KEY (Day) REFERENCES Week (ID\_Day)

ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE People ADD CONSTRAINT FK\_People\_Departments

FOREIGN KEY (Departament) REFERENCES Departments (ID\_Department)

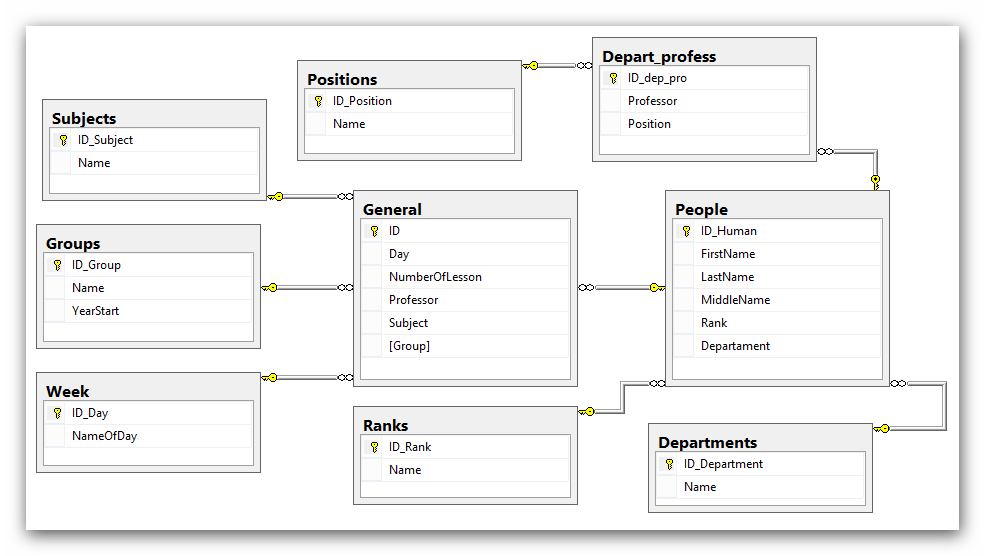
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE People ADD CONSTRAINT FK\_People\_Ranks

FOREIGN KEY (Rank) REFERENCES Ranks (ID\_Rank)

ON DELETE CASCADE ON UPDATE CASCADE;

Якщо побудувати діаграму створеної бази даних, то можемо побачити таблиці та зв’язки між ними (мал. 7).



Мал. 6. Діаграма усіх таблиць бази даних Hotel

# Створення необхідного функціонування

За допомогою представлень, збережених процедур і функцій, створюємо необхідний функціонал.

## Створення впорядкованих списків

Створюємо списки, згідно з варіантом завдання.

А) Список кафедра-викладач-предмет:

CREATE PROCEDURE [dbo].[Dep\_prof\_subj]

AS

BEGIN

SET NOCOUNT ON;

SELECT P.[LastName],P.[FirstName],P.[MiddleName],S.[Name] AS [Subject],D.[Name] AS Department

FROM General G

INNER JOIN (People P INNER JOIN Departments D ON P.[Departament]=D.[ID\_Department]) ON P.[ID\_Human]=G.[Professor]

INNER JOIN Subjects S ON S.[ID\_Subject]=G.[Subject]

END

GO

Б) Список викладачів, у яких посада відповідає науковій степені:

CREATE PROCEDURE [dbo].[List1]

AS

BEGIN

SET NOCOUNT ON;

SELECT P.[LastName],P.[FirstName],P.[MiddleName]

FROM Depart\_profess DP

INNER JOIN People P ON P.[ID\_Human]=DP.[Professor]

INNER JOIN Positions Pos ON Pos.[ID\_Position]=DP.[Position]

WHERE DP.[Position]!=P.[Rank]

END

GO

## Перевірка, що в кожного викладача не більше трех пар в один день

Напишемо процедуру, яка буде виводити «ТАК» чи «НІ».

CREATE PROCEDURE [dbo].[NumberOfLessons]

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (SELECT Number

FROM General G

INNER JOIN People P ON P.[ID\_Human]=G.[Professor]

INNER JOIN

(SELECT Professor, [Day], count(\*) as Number

FROM General G

GROUP BY Professor, [Day])

GP ON GP.[Professor]=P.[ID\_Human]

WHERE Number>3)

BEGIN

PRINT('YES')

END

ELSE

BEGIN

PRINT('NO')

END

END

GO

## Корегування даних

Зміна групи для викладача при заданому дню тижня й номера пари:

CREATE PROCEDURE [dbo].[Correct]

@LastName nvarchar(50),

@FirstName nvarchar(50),

@Day nvarchar(50),

@NumberOfLesson int,

@Group nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @Prof\_ID INT

SET @Prof\_ID=(SELECT ID\_Human FROM [People] WHERE LastName=@LastName AND FirstName=@FirstName)

DECLARE @GROUP\_ID INT

SET @GROUP\_ID=(SELECT ID\_Group FROM [Groups] WHERE Name=@Group)

DECLARE @DAY\_ID INT

SET @DAY\_ID=(SELECT ID\_Day FROM [Week] WHERE NameOfDay=@Day)

UPDATE General

SET [Day]=@Day, [NumberOfLesson]=@NumberOfLesson, [Professor]=@Prof\_ID

WHERE [Group]=@Group

END

GO

## Створення звіту

А) Розклад заннять для конктреної групи

CREATE PROCEDURE [dbo].[Schedule]

@Group nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @Group\_ID INT

SET @Group\_ID=(SELECT ID\_Group FROM [Groups] WHERE Name=@Group)

SELECT P.[LastName],P.[FirstName],P.[MiddleName],W.[NameOfDay],NumberOfLesson,S.[Name]

FROM General G

INNER JOIN People P ON P.[ID\_Human]=G.[Professor]

INNER JOIN Subjects S ON S.[ID\_Subject]=G.[Subject]

INNER JOIN [Week] W ON W.[ID\_Day]=G.[Day]

WHERE G.[Group]=@Group\_ID

END

GO

Б) Якісний склад працівників на кафедрі

CREATE PROCEDURE [dbo].[Quality]

@Department nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @Dep\_ID INT

SET @Dep\_ID=(SELECT ID\_Department FROM [Departments] WHERE Name=@Department)

SELECT R.[Name], COUNT(\*)

FROM People P

INNER JOIN Ranks R ON P.[Rank]=R.[ID\_Rank]

WHERE P.[Departament]=@Dep\_ID

GROUP BY P.[Rank], R.[Name]

END

GO

В) Розклад занять для викладачів

CREATE PROCEDURE [dbo].[Schedule\_for\_prof]

@Department nvarchar(10)

AS

BEGIN

SET NOCOUNT ON;

SELECT W.[NameOfDay] as [Day],NumberOfLesson,Gr.[Name] as [Group],P.[LastName],P.[FirstName],P.[MiddleName],S.[Name] as [Subject]

FROM General G

INNER JOIN Subjects S ON G.[Subject]=S.[ID\_Subject]

INNER JOIN (People P INNER JOIN Departments D ON P.[Departament]=D.[ID\_Department]) ON P.[ID\_Human]=G.[Professor]

INNER JOIN [Week] W ON W.ID\_Day = G.[Day]

INNER JOIN [Groups] Gr ON G.[Group]=Gr.[ID\_Group]

WHERE D.[Name]=@Department

END

GO

Г) Навантаженість викладачів:

ALTER PROCEDURE [dbo].[LastList]

@Department nvarchar(10),

@LastName nvarchar(50),

@FirstName nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @ID\_Human INT

SET @ID\_Human=(SELECT ID\_Human FROM People WHERE FirstName=@FirstName AND LastName=@LastName AND Departament=(SELECT ID\_Department FROM Departments WHERE Name=@Department))

SELECT S.[Name] as [Subject], count(\*) as NumberOfLessons

FROM General G

INNER JOIN Subjects S ON S.[ID\_Subject]=G.[Subject]

WHERE G.[Professor]=@ID\_Human

GROUP BY S.[Name]

DECLARE @COUNT INT

SET @COUNT=(SELECT COUNT(\*) FROM

(SELECT S.[Name] as [Subject], count(\*) as NumberOfLessons

FROM General G

INNER JOIN Subjects S ON S.[ID\_Subject]=G.[Subject]

WHERE G.[Professor]=@ID\_Human

GROUP BY S.[Name]) P)

PRINT(@COUNT)

END

GO

# 

# Перевірка нормалізації розробленої моделі

Для того, щоб це перевірити, переввіремо чи належить дана модель до третьої нормальної форми. Для цього модель повинна належати до першої і другої форми, а також не повинна мати транзитивних відношень, тобто залежностей неключевого атрибуту від іншого неключового.

Для початку перевіремо належність до першої нормальної форми. Для цього усі атрибути повинні буди атомарні і повинні бути відсутні групи, які повторюються. Це виконується, виходячи з того, що в таблицях немає повторювальних даних чи груп.

Далі перевіремо належність до другої нормальної групи. Це буде виконуватись, якщо модель належить до першої нормальної групи (це вже перевірено), а також будуть відсутні неповні функуіональні залежності неключевих атрибутів первинного ключа., тобто будь-яке неключеве поле має однозначно ідентифікуватись повним набором ключів. Це виконується, тому що будь-яке поле унікальне і визначається набором ключів.

Залишилось перевірити відсутність транзітивних відношень, тобто ні однез ключевих полів не має ідентифікуватися за допомогою іншого ключевого поля. У побудованій моделі немає полів які ідентифікуються в залженості від іншого ключевого поля. З цього слідує що розроблена модель належить до тертої нормальної форми.

З цього слідує, що процес нормалізаціїї чи приведення до третьої нормальної форми не потрібен, бо він вже досягнут.

# Висновок

У ході виконання розрахунково-графічної роботи, я дослідив методи проектування баз даних, навчився проектувати власну базу даних у програмі Enterprise Erchitect, а також створювати різного роду функціонування для потреб користувача.