FPT Education

FPT UNIVERSITY

DBI ASSIGNMENT REPORT

Student: Pham Xuan Chieu

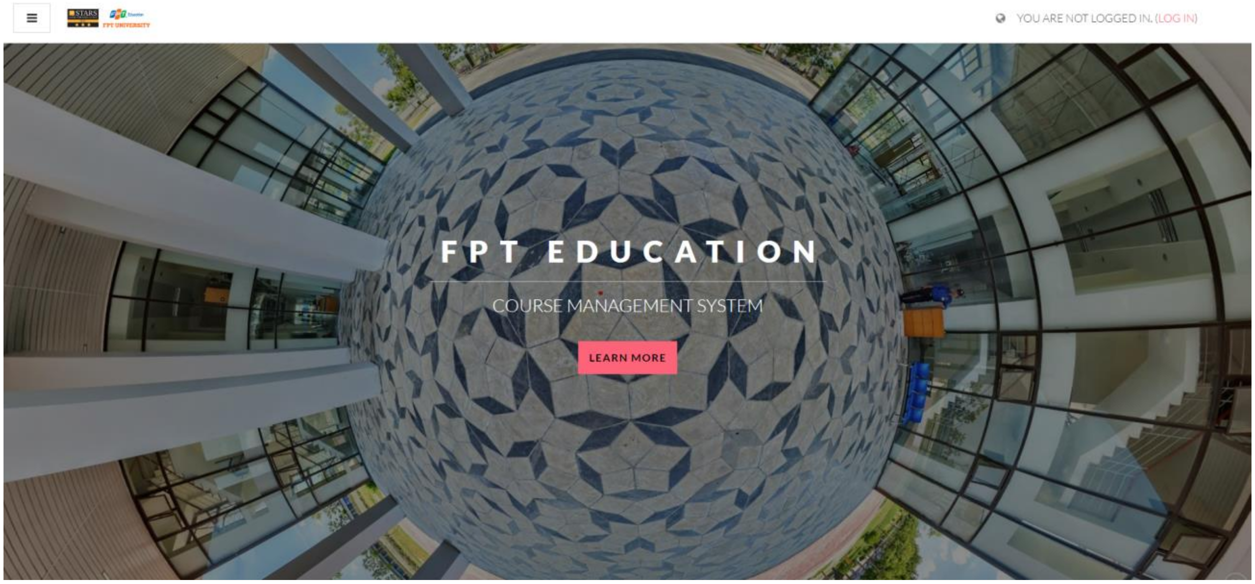
Student’s Roll Number: HE151312

Class: AI1706

Subject: Introduction to Databases (DBI202)

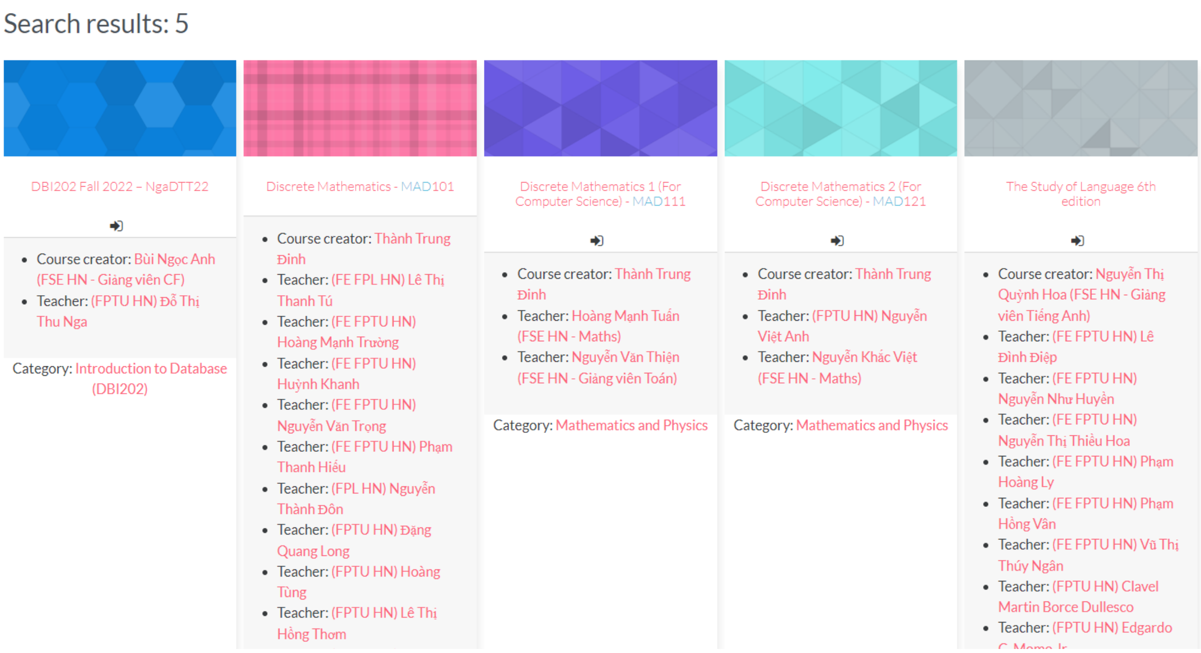
Instructor: Le Phuong Chi

A. Preview



CMS_ FPT is course management system of FPT university.

CMS_FPT offers courses in both professional subjects and soft skills. A student can enroll in multiple courses taught by faculty members on campus. In courses, teachers also provide materials to help students learn. And after a learning process, students will take the exam and get updated results



Search results: 5

DBI202 Fall 2022 – NgaDTT22

- Course creator: Bùi Ngọc Anh (FSE HN - Giảng viên CF)
- Teacher: (FPTU HN) Đỗ Thị Thu Nga

Category: Introduction to Database (DBI202)

Discrete Mathematics - MAD101

- Course creator: Thành Trung Đinh
- Teacher: (FE FPL HN) Lê Thị Thanh Tú
- Teacher: (FE FPTU HN) Hoàng Mạnh Trường
- Teacher: (FE FPTU HN) Huỳnh Khanh
- Teacher: (FE FPTU HN) Nguyễn Văn Trọng
- Teacher: (FE FPTU HN) Phạm Thanh Hiếu
- Teacher: (FPL HN) Nguyễn Thành Đôn
- Teacher: (FPTU HN) Đặng Quang Long
- Teacher: (FPTU HN) Hoàng Tùng
- Teacher: (FPTU HN) Lê Thị Hồng Thơm

Discrete Mathematics 1 (For Computer Science) - MAD111

- Course creator: Thành Trung Đinh
- Teacher: Hoàng Mạnh Tuấn (FSE HN - Maths)
- Teacher: Nguyễn Văn Thiện (FSE HN - Giảng viên Toán)

Category: Mathematics and Physics

Discrete Mathematics 2 (For Computer Science) - MAD121

- Course creator: Thành Trung Đinh
- Teacher: (FPTU HN) Nguyễn Việt Anh
- Teacher: Nguyễn Khắc Việt (FSE HN - Maths)

Category: Mathematics and Physics

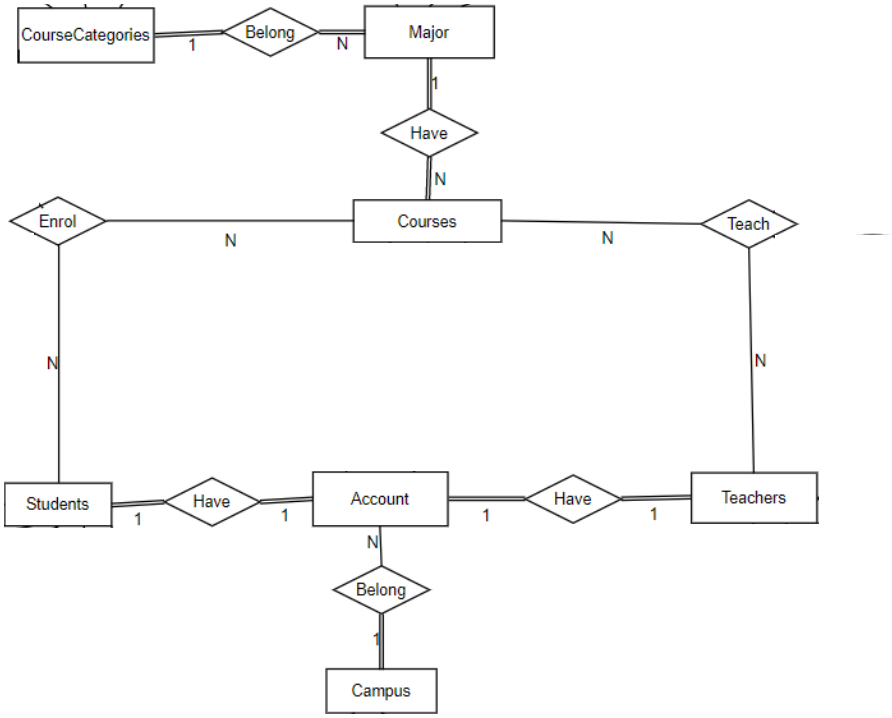
The Study of Language 6th edition

- Course creator: Nguyễn Thị Quỳnh Hoa (FSE HN - Giảng viên Tiếng Anh)
- Teacher: (FE FPTU HN) Lê Đình Điệp
- Teacher: (FE FPTU HN) Nguyễn Như Huyền
- Teacher: (FE FPTU HN) Nguyễn Thị Thiều Hoa
- Teacher: (FE FPTU HN) Phạm Hoàng Ly
- Teacher: (FE FPTU HN) Phạm Hồng Vân
- Teacher: (FE FPTU HN) Vũ Thị Thúy Ngân
- Teacher: (FPTU HN) Clavel Martin Borce Dullesco
- Teacher: (FPTU HN) Edgardo C. Momo Jr

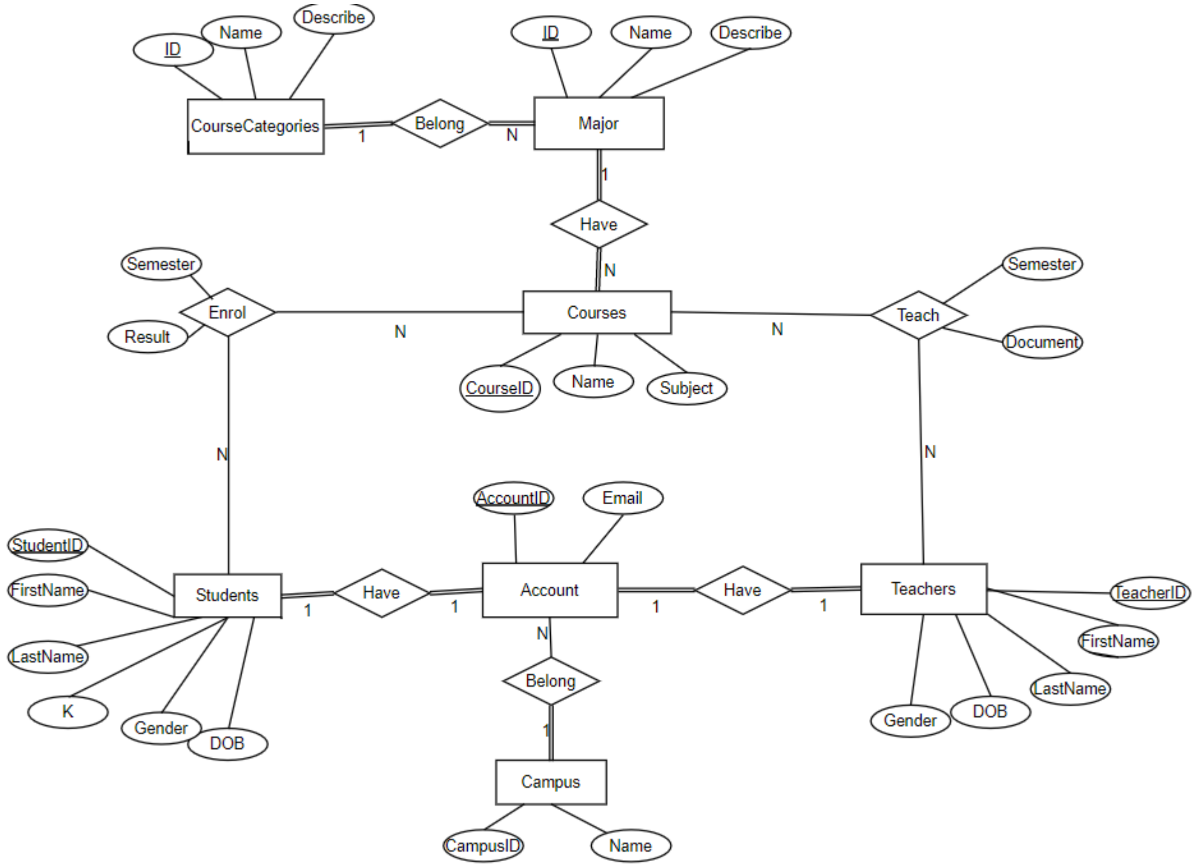
B. Entity Relationship Diagram (Using Chen 's Notation)

I. Entity Relationship Diagram (ERD) for Database

1. Simplified Entity Relationship Diagram

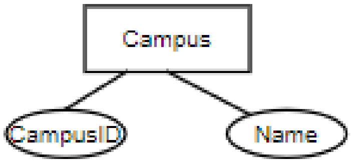


2. Full Entity Relationship Diagram for Database



II. Explanations for entities

1.Campus

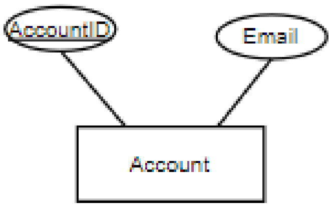


- Definition: the buildings of a college or university, or of a large organization, and the land that surrounds them. FPT university education system consists of 4 campuses: Hanoi, Da Nang, Ho Chi Minh and Quy Nhon.

- Attributes of Campus: CampusID, Name

- CampusID: number in ascending order by geographical location from north to south Vietnam =>This is primary key for the Campus entity

2. Account



- Definition: each student or teacher owns a unique account to join or manage their course on the CMS system

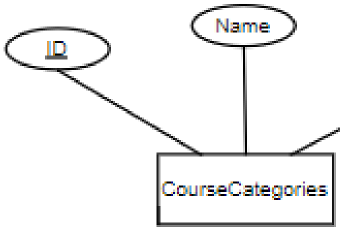
- Attributes of Account: AccountID and Email

- AccountID : formed from the name of the student or teacher along with their code, each of which has its own unique code

=> This is primary key for the Account entity

-Email: Every student or teacher has an email with a structure similar to AccountID so I decided not to make it the key to reduce complexity.

3. Course Categories



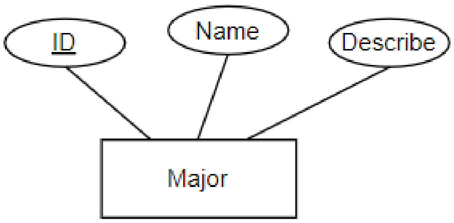
-Definition : In this case, it is defined as a discipline or block of knowledge. Example : Fundamental includes: English, Chinese, Soft Skills, etc. Software Engineering includes: Computing Fundamentals, Graphic Design, Information Assurance, etc.

- Attributes of Account : ID, Name, Describe

- ID: it is the initials of Course Categories, so it is usually unique.

=> This is primary key for the Course Categories entity

4. Major



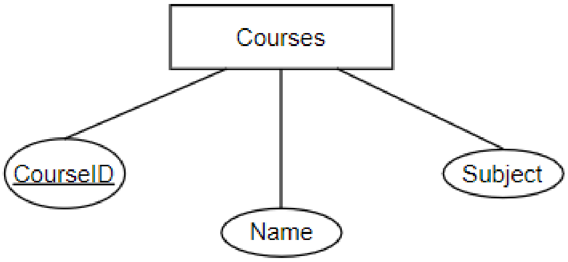
- Definition : In this case, major include the courses which students have to learn and teacher may teach.

- Attributes of Major: ID, Name, Describe, CourseCategories

- ID: it is the initials of Major, so it is usually unique.

=> This is the primary key of the Major

5. Courses



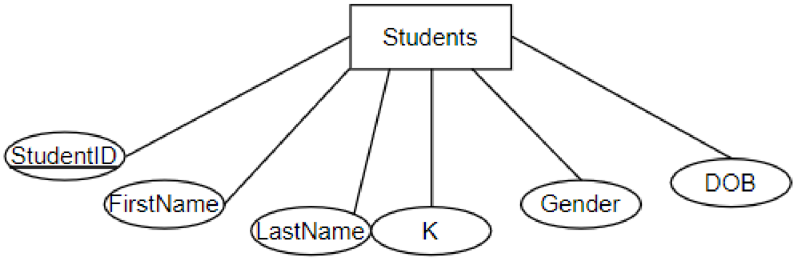
- Definition: Course is a set of classes or a plan of study on a particular subject, usually leading to an exam or qualification

- Attributes of Courses: CourseID, Name,Subject, Major

- CourseID: it stands for public subject with its code . Example: CSD203 stands for Data Structures and Algorithm with Python mean while CSD201 stand for Data Structures and Algorithms. It is always true that it will be different subject or subject code so the courses will be different so students can distinguish them

=> This is the primary key of the Courses

6. Students



- Definition: it includes all information of a students.

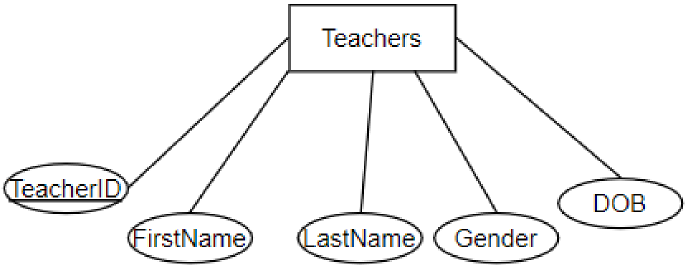
- Attributes of Students: StudentID,Email,FirstName,LastName,K,DOB,Gender

- StudentID: it stands for the student's registration number on the campus. This is an ascending sequence of numbers, and the studentids of the campuses have nothing to do with each other

=> This is the primary key of the Students

- K: It is a symbol of the school year in which students enter the school

7. Teachers



- Definition: it includes all information of a teachers.

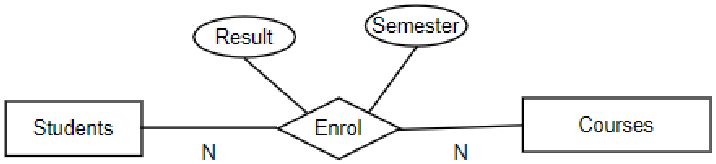
- Attributes of Teachers: TeacherID, Email, FirstName, LastName, DOB, Gender

- TeacherID: it stands for LastName, FirstName and code of the teacher. It is unique

=> This is the primary key of the Teachers

III. Explanation for Entity Relationships

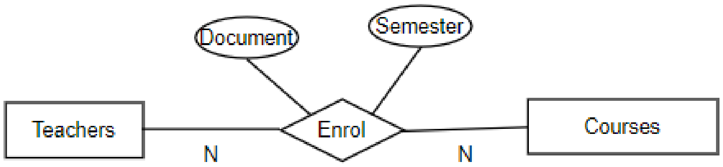
1.Students_Courses



- One student can enrol for multiple courses in a particular semester. It is sure that one course is enrolled by many students. The result Course of a student will be update at the end of semester.

- One student may not register for any courses. At the same time, one course may be not enrolled by any students.

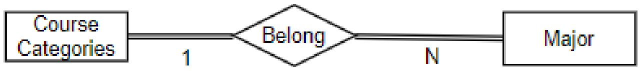
2. Teachers_Courses



- One teacher can teach courses in a particular semester. They also have private documents about the course which they were teaching. One course can teach by many teachers.

- One teacher may not teach any course. At the same time, one course may not be taught by any teacher.

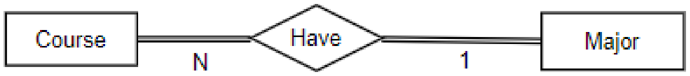
3. CourseCategories_Major



- Majors can be belong one CourseCategory

- One major have to belong a specific CourseCategory. A CourseCategory must have at least one major.

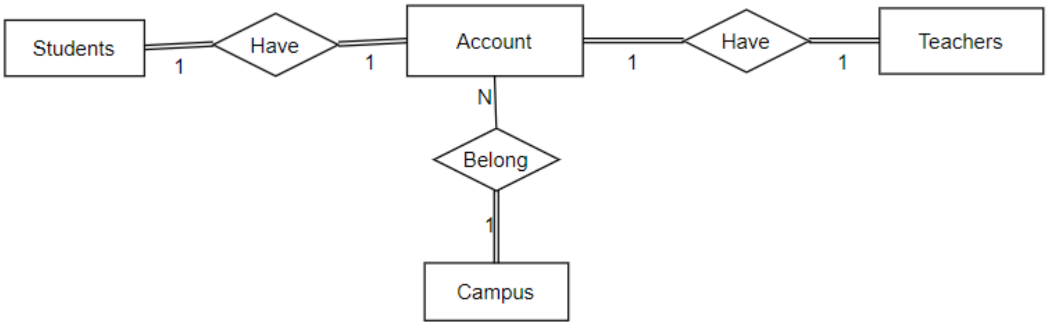
4. Major_Course



- One major can have courses but one course just belong to one major.

- One major have at least one Course. A Course must belong to one major.

5. Campus_Account_Students_Teachers



- When one student or teacher joined a specific campus, they will be give a unique account

- One account have to belong one student or one teacher

- One account can not owner by many people.

D. Relational Mapping

I. Step 1: Mapping Regular Entity

- For each regular (not weak) entity E, create a relation R including all necessary attributes of E, one column for each attribute.

- Choose the key of entity E as the primary key of R.

- If the key of E includes more than one attributes, the set of corresponding columns in R will form its primary key

Following these steps about for my entity, I have the result:

- CourseCategories(CourseCategoriesID, Name, Describe) has CourseCategoriesID as Primary key

- Major(MajorID, Name, Describe) has MajorID as Primary key

- Courses(CourseID, Name, Subject) has CourseID as Primary key

- Campus(CampusID,Name,Descibe) has CampusID as Primary key

- Students(StudentID,FirstName,LastName,K,DOB,Gender) has StudentID as Primary key

- Teachers(TeacherID,FirstName,LastName,DOB,Gender) has TeacherID as Primary key

- Account(AccountID,Email) has AccountID,Email as Primary key

II. Step 2: Weak entities handling:

There is no weak-entiy in my ERD so no need to do this steps

III. Step 3: Sub – entities handling:

There is no sub-entity in my ERD so no need to do this steps

IV. Step 4: 1 – N (1 to many) relationships handling:

- On the many – relationship relation of the relationship, add its foreign keys refers to the keys of the 1 – relationship relation of the relationship.

- The 1 – N (1 to many) relationships:

CourseCategories (1) - be - Major(N)

Add foreign key CourseCategories from CourseCategories to Major

Major Major(ID, Name, Describe, CourseCategories)

Major (1) - be - Courses(N)

Add foreign key Major from Major to Courses

Courses(CourseID, Name, Subject,Major)

Campus (1) - be - Account(N)

Add foreign key CampusID from Campus to Account

Account(AccountID,Email,CampusID)

V. Step 5: 1 – 1 relationship handling:

- Choose a key from one entity and add it to other entity as foreign key.

- 1 – 1 relationship:

Account – owned by– Students

Add Email reference to Email from Account to Students

Students(StudentID,FirstName,LastName,K,DOB,Gender,Email)

Account – owned by– Teachers

Add Email reference to Email from Account to Teachers

Teachers(TeacherID,FirstName,LastName,DOB,Gender,Email)

VI. Step 6: Many to many (M - N) relationships handling:

- Steps:

Create a new table to represent the relationship

New table contains two foreign keys – one from each of the participants in the relationship

The primary key of the new table is the combination of the two foreign keys:

- M – N relationships:

Students – Enrol - Courses

Create a new table named Students_Courses with attributes from relationship attributes.

Students_Course (Semester, Result)

Add StudentID reference to StudentID from Students, CourseID reference to CourseID from Courses to Students_Courses

Students_Courses(StudentID,CourseID,Semester,Result) has Primary key is the combination of (StudentID,CourseID)

Teachers – Teach - Courses

Create a new table named Teachers_Courses with attributes from relationship attributes.

Teachers_Courses (Semester, Doccument)

Add TeacherID reference to TeacherID from Teachers, CourseID reference to CourseID from Courses to Teachers_Courses

Teachers_Courses(TeacherID,CourseID,Semester,Result) has Primary key is the combination of (TeacherID,CourseID)

VII. Muli – part and multi – value handling:

- Multi – part attribute: none.

- Multi – value attribute: none.

E. Logical Design

I. Relational Schema

List of relations in the database logical design:

- CourseCategories(CourseCategoriesID, Name, Describe) has CourseCategoriesID as Primary key

- Major(MajorID, Name, Describe, CourseCategories) has MajorID as Primary key

- Courses(CourseID, Name, Subject, Major) has CourseID as Primary key

- Campus(CampusID,Name,Descibe) has CampusID as Primary key

- Students(StudentID,FirstName,LastName,K,DOB,Gender,Email) has StudentID as Primary key

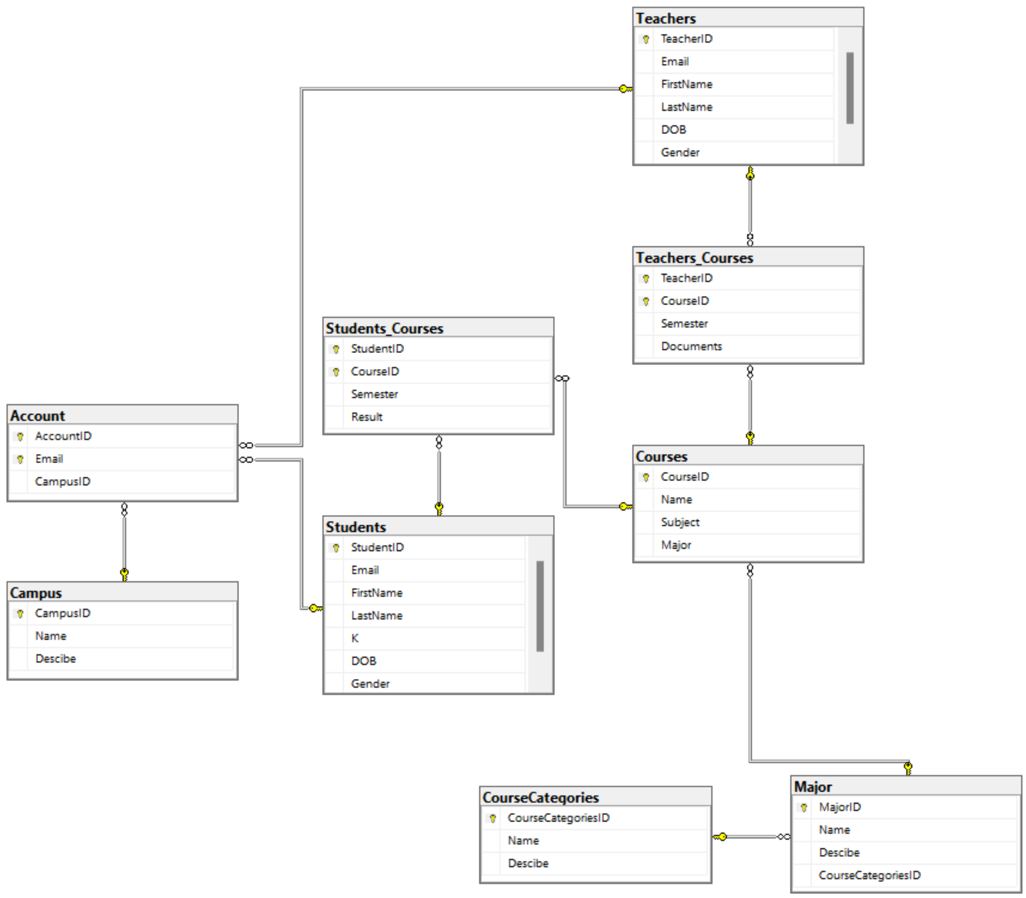
- Teachers(TeacherID,FirstName,LastName,DOB,Gender,Email) has TeacherID as Primary key

- Account(AccountID,Email,CampusID) has AccountID,Email as Primary key

- Students_Courses(StudentID,CourseID,Semester,Result) has Primary key is the combination of (StudentID,CourseID)

- Teachers_Courses(TeacherID,CourseID,Semester,Result) has Primary key is the combination of (TeacherID,CourseID)

II. Database Diagram



III. Table Analysis

1. CourseCategories

Attributes	Data type	Allow null
CourseCategoriesID (Primary key)	Varchar(50)	No
Name	Varchar(50)	No
Describe	Varchar(500)	No

Constraints Descriptions:

All columns in this table must be filled so each column has NOT NULL constraint.

Each Course Category has a CourseCategoriesID using for identifier so CourseCategoriesID is primary key of CourseCategories

2.Major

Attributes	Data type	Allow null
MajorID(Primary key)	Varchar(50)	No
Name	Varchar(50)	No
Describe	Varchar(500)	No
CourseCategoriesID	Varchar(50)	No

Constraints Descriptions:

All columns in this table must be filled so each column has NOT NULL constraint.

Each Major has a MajorID using for identifier so MajorID is primary key of Major

3.Courses

Attributes	Data type	Allow null
CourseID(Primary key)	Varchar(50)	No
Name	Varchar(50)	No
Subject	Varchar(50)	No
MajorID	Varchar(50)	No

Constraints Descriptions:

All columns in this table must be filled so each column has NOT NULL constraint.

Each Course has a CourseID using for identifier so CourseID is primary key of Courses

4.Campus

Attributes	Data type	Allow null
CampusID(Primary key)	Int	No
Name	Varchar(50)	No
Describe	Varchar(500)	No

All columns in this table must be filled so each column has NOT NULL constraint.

Each Campus has a CampusID using for identifier so CampusID is primary key of Campus

5.Students

Attributes	Data type	Allow null

StudentID(Primary key)	Varchar(10)	No
FirstName	Varchar(50)	No
LastName	Varchar(10)	No
K	Int	No
DOB	Date	No
Gender	Varchar(10)	No
Email (Unique key)	Varchar(50)	No

All columns in this table must be filled so each column has NOT NULL constraint.

Each Student has a StudentID using for identifier so StudentID is primary key of Students

Each Student has a Email unique so Email is a unique key of Students

6.Teachers

Attributes	Data type	Allow null
TeacherID(Primary key)	Varchar(10)	No
FirstName	Varchar(50)	No
LastName	Varchar(10)	No
DOB	Date	No
Gender	Varchar(10)	No
Email (Unique key)	Varchar(50)	No

All columns in this table must be filled so each column has NOT NULL constraint.

Each Teacher has a TeacherID using for identifier so TeacherID is primary key of Teachers.

Each Teacher has a Email unique so Email is a unique key of Teachers

7. Account

Attributes	Data type	Allow null
AccountID(Primary key)	Varchar(10)	No
Email(Primary key)	Varchar(50)	No
CampusID	Int	No

All columns in this table must be filled so each column has NOT NULL constraint.

Each Account has a AccountID or Email using for identifier so AccountID,Email is primary key of Account.

8.Students_Courses

Attributes	Data type	Allow null
StudentID(Primary key)	Varchar(10)	No
CourseID(Primary_key)	Varchar(10)	No
Semester	Varchar(10)	No
Result	Float	No

All columns in this table must be filled so each column has NOT NULL constraint.

The combination of (StudentID, CourseID) make each record unique so it is primary key of this table

Primary key(StudentID,CourseID)

The data in columns (StudentID, CourseID) must be consistent with the data from the original table. So there are foreign keys between those table

StudentID varchar(10) foreign key references Students(StudentID),

CourseID varchar(10) foreign key references Courses(CourseID)

9.Teacher_Course

Attributes	Data type	Allow null
TeacherID(Primary key)	Varchar(10)	No
CourseID(Primary_key)	Varchar(10)	No
Semester	Varchar(10)	No
Document	Varchar(50)	No

All columns in this table must be filled so each column has NOT NULL constraint.

The combination of (TeacherID, CourseID) make each record unique so it is primary key of this table

Primary key(TeacherID, CourseID)

The data in columns (TeacherID, CourseID) must be consistent with the data from the original table. So there are foreign keys between those table

TeacherID varchar(10) foreign key references Teachers(TeacherID),

CourseID varchar(10) foreign key references Courses(CourseID)

IV. Database statements used to create table

```
---CREATE DATABASE---
```

```
CREATE DATABASE CMS_FPT

USE CMS_FPT
```

1.CourseCategories

```
create table CourseCategories

(

	CourseCategoriesID varchar(50) primary key,

	[Name] varchar(50) not null,

	Descibe varchar(500) not null

)
```

2.Major

```
create table Major

(

	MajorID varchar(50) primary key,

	[Name] varchar(50) not null,

	Descibe varchar(500) not null,

	CourseCategoriesID varchar(50) foreign key references CourseCategories(CourseCategoriesID)

)
```

3.Courses

```
create table Courses

(

	CourseID varchar(50) primary key,

	[Name] varchar(50)not null,

	[Subject] varchar(50)not null,

	MajorID varchar(50) foreign key references Major(MajorID)

)
```

4.Campus

```
create table Campus
(
        CampusID int primary key,
        [Name] varchar(50)not null,
        Descibe varchar(500)not null
)
```

5.Students

```
create table Students
(
        StudentID varchar(10) ,
        Email varchar(50) unique ,
        FirstName varchar(50)not null,
        LastName varchar(10)not null,
        K int not null,
        DOB date,
        Gender varchar(10) check(Gender ='Male' or Gender ='Female')
        primary key (StudentID)
)
```

6.Teachers

```
create table Teachers
(
        TeacherID varchar(10),
        Email varchar(50) unique,
        FirstName varchar(50) not null,
        LastName varchar(10)not null,
        DOB date,
        Gender varchar check(Gender ='Male' or Gender ='Female')
        primary key (TeacherID)
)
```

7.Account

```
Create table Account
(
        AccountID varchar(10) ,
        Email varchar(50),
        primary key (AccountID,Email),
        CampusID int foreign key references Campus(CampusID),
        foreign key(Email) references Students(Email),
        foreign key(Email) references Teachers(Email)
)
```

8.Students_Courses

```
create table Students_Courses
(
        StudentID varchar(10) foreign key references Students(StudentID),
        CourseID varchar(10) foreign key references Courses(CourseID),
        Semester varchar(10)not null,
        Result float not null,
        primary key(StudentID,CourseID)
)
```

9.Teachers_Courses

```
create table Teachers_Courses
(
       TeacherID varchar(10) foreign key references Teachers(TeacherID),

       CourseID varchar(10) foreign key references Courses(CourseID),

       Semester varchar(10) not null,

       Documents varchar(50)not null,

       primary key(TeacherID,CourseID)

)
```

F. Queries, Store Procedures and Trigger

I. Sample Queries

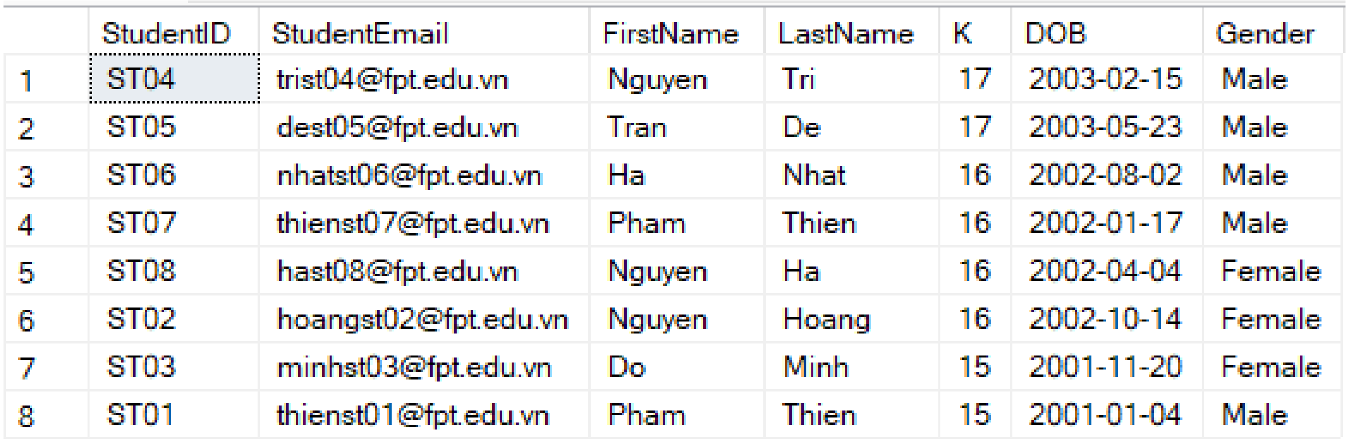
1. Query using ORDER BY

- Question: Display all student information sorted by K in descending direction

- Query:

```
select *
from Students
order by K desc
```

-Result:



	StudentID	StudentEmail	FirstName	LastName	K	DOB	Gender
1	ST04	trist04@fpt.edu.vn	Nguyen	Tri	17	2003-02-15	Male
2	ST05	dest05@fpt.edu.vn	Tran	De	17	2003-05-23	Male
3	ST06	nhatst06@fpt.edu.vn	Ha	Nhat	16	2002-08-02	Male
4	ST07	thienst07@fpt.edu.vn	Pham	Thien	16	2002-01-17	Male
5	ST08	hast08@fpt.edu.vn	Nguyen	Ha	16	2002-04-04	Female
6	ST02	hoangst02@fpt.edu.vn	Nguyen	Hoang	16	2002-10-14	Female
7	ST03	minhst03@fpt.edu.vn	Do	Minh	15	2001-11-20	Female
8	ST01	thienst01@fpt.edu.vn	Pham	Thien	15	2001-01-04	Male

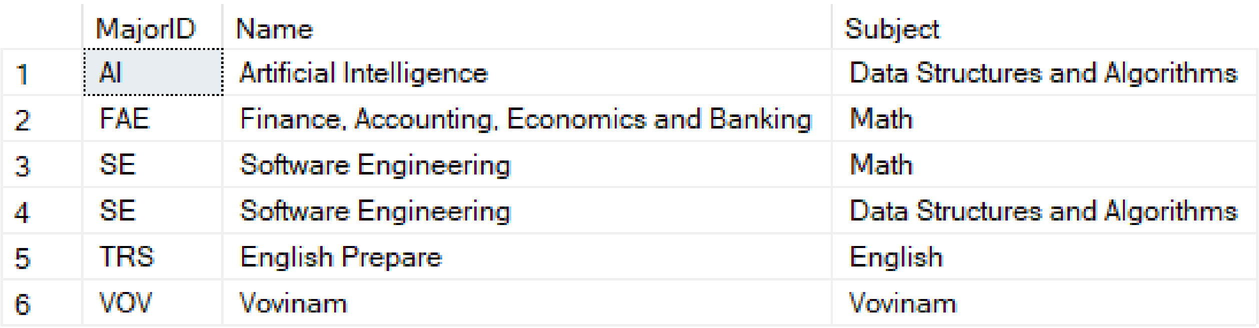
2.Query using INNER JOINS

- Question: Display the name of the subject taught by each major

- Query:

```
select m.MajorID,m.[Name],Subject
from Major m inner join Courses c
on m.MajorID = c.MajorID
order by MajorID asc
```

- Result:



	MajorID	Name	Subject
1	AI	Artificial Intelligence	Data Structures and Algorithms
2	FAE	Finance, Accounting, Economics and Banking	Math
3	SE	Software Engineering	Math
4	SE	Software Engineering	Data Structures and Algorithms
5	TRS	English Prepare	English
6	VOV	Vovinam	Vovinam

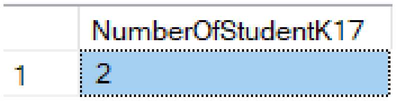
3. A query that uses aggregate functions

- Question: show total number of students as K17

- Query:

```
select count(*)[NumberOfStudentK17]
from Students
where K= 17
group by K
```

- Result:



	NumberOfStudentK17
1	2

4. A query that uses the GROUP BY and HAVING clauses

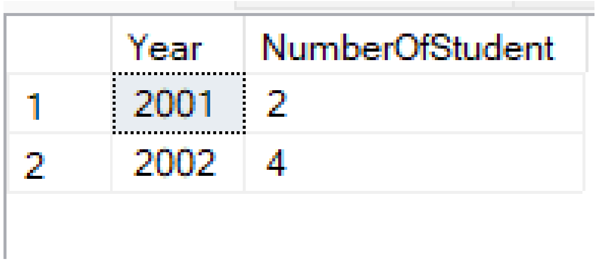
-Question:

Displays the number of students born in 2002 and before

- Query:

```
select  year(DOB)[year],count(*) [NumberOfStudent]
from Students
group by year(DOB)
having  year(DOB)<=2002
```

-Result:



	Year	NumberOfStudent
1	2001	2
2	2002	4

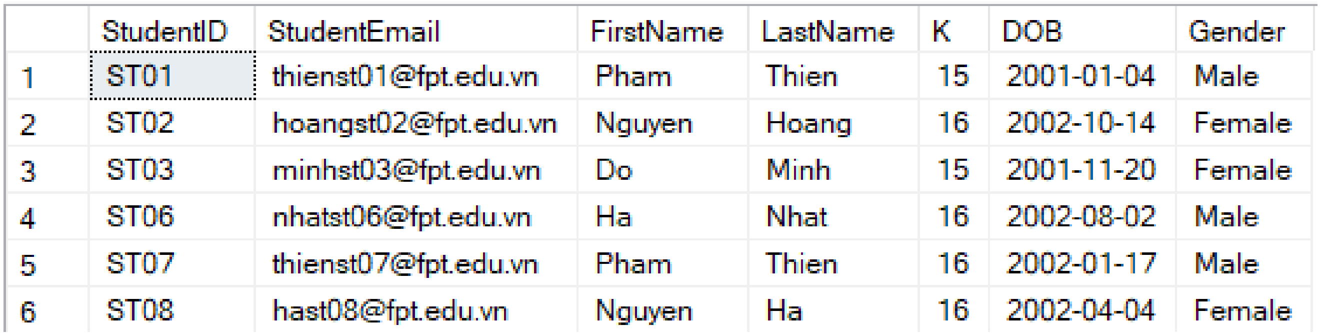
5. Queries that use partial matching in the WHERE clause

- Question: display all information of students with ‘h’ in Lastname

-Query:

```
select *
from Students
where LastName like '%h%'
```

- Result:



	StudentID	StudentEmail	FirstName	LastName	K	DOB	Gender
1	ST01	thienst01@fpt.edu.vn	Pham	Thien	15	2001-01-04	Male
2	ST02	hoangst02@fpt.edu.vn	Nguyen	Hoang	16	2002-10-14	Female
3	ST03	minhst03@fpt.edu.vn	Do	Minh	15	2001-11-20	Female
4	ST06	nhatst06@fpt.edu.vn	Ha	Nhat	16	2002-08-02	Male
5	ST07	thienst07@fpt.edu.vn	Pham	Thien	16	2002-01-17	Male
6	ST08	hast08@fpt.edu.vn	Nguyen	Ha	16	2002-04-04	Female

6.