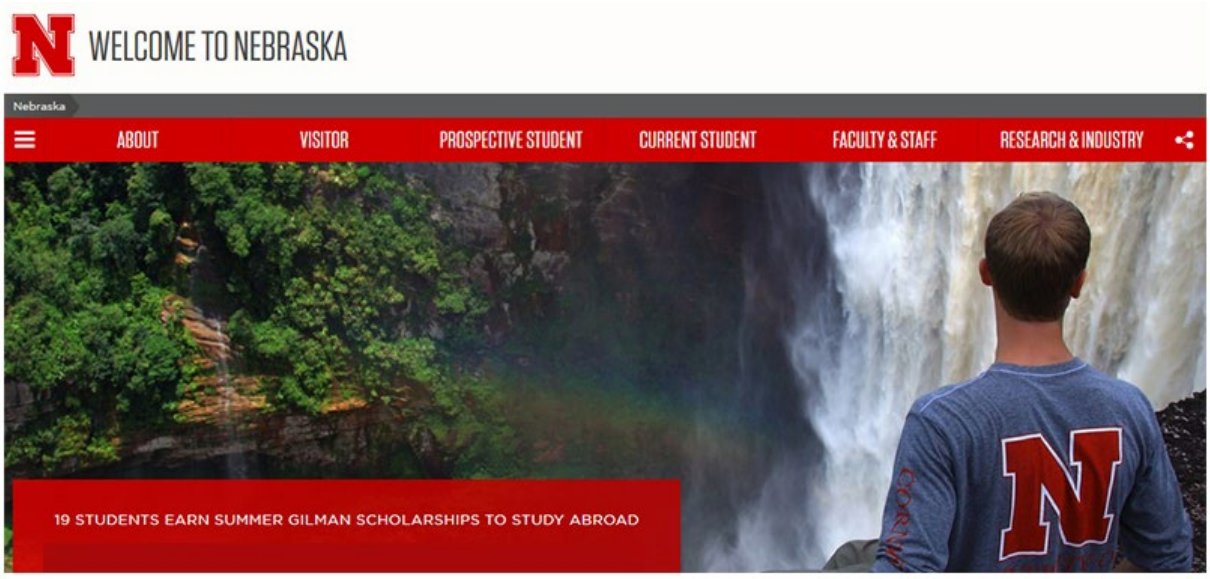
CSCE 608 – Project 1
University Database project

Name: Nimoshika Jayaraman
UIN : 631000852

a. Project Description:

This application is designed for a university DB. This consists of basic information like the courses provided, instructor details, students information, budget of the department, location of the instructor, first name and last name of a person (both instructor and students), hire date of an instructor, enrollment date of a student, name of the course, start date of the department, grade of the student ,credits for a particular course. This application both displays the details of DB and helps in manipulation too. For this purpose, we use IDs to improve the uniqueness. We use course ID, department ID, instructor ID, enrollment ID, student ID.

The details are categorized into 6 divisions as Department, course instructor, enrollment, course, person, and office assignment. Our DB is designed in such a way that a particular instructor can teach more than one course. A student can enroll in more than one course. A course must be taught by only one instructor. Application was designed having simple webpage like the below in mind.



b. <u>The Entity-Relationship diagram of Database:</u>

As discussed in the previous section, we have 6 entities. Then, we can list the attributes for each entity.

For **Department** entity, we will have attributes like: Department ID, Instructor ID, Name, Budget and Start Date. Department ID is unique for the department. It is the primary key. Instructor ID is unique for the instructor. Name specifies the name of the department. Start Date denote the start date of the department. Budget denote the budget for the specific department.

For **Course** entity, we will have attributes like: Course ID, Title and Credits. Course ID is unique for a given course. It is the primary key. Credits is the number of credits a subject hold. Title is the title of the course.

For **Person** entity, we will have attributes like: ID, First Name, Last Name, Hire Date and Enrollment Date. ID is unique for a person. It is the primary key. First name denotes the first name of a person. Last name denotes the last name of a person. Hire date denotes the hire date of an instructor. Enrollment date denotes the date in which the student is enrolled.

For **Course Instructor** entity, we have attributes like: Course ID and Instructor ID. Course ID is unique for a given course and Instructor ID is unique for a given instructor. Instructor ID is the primary key.

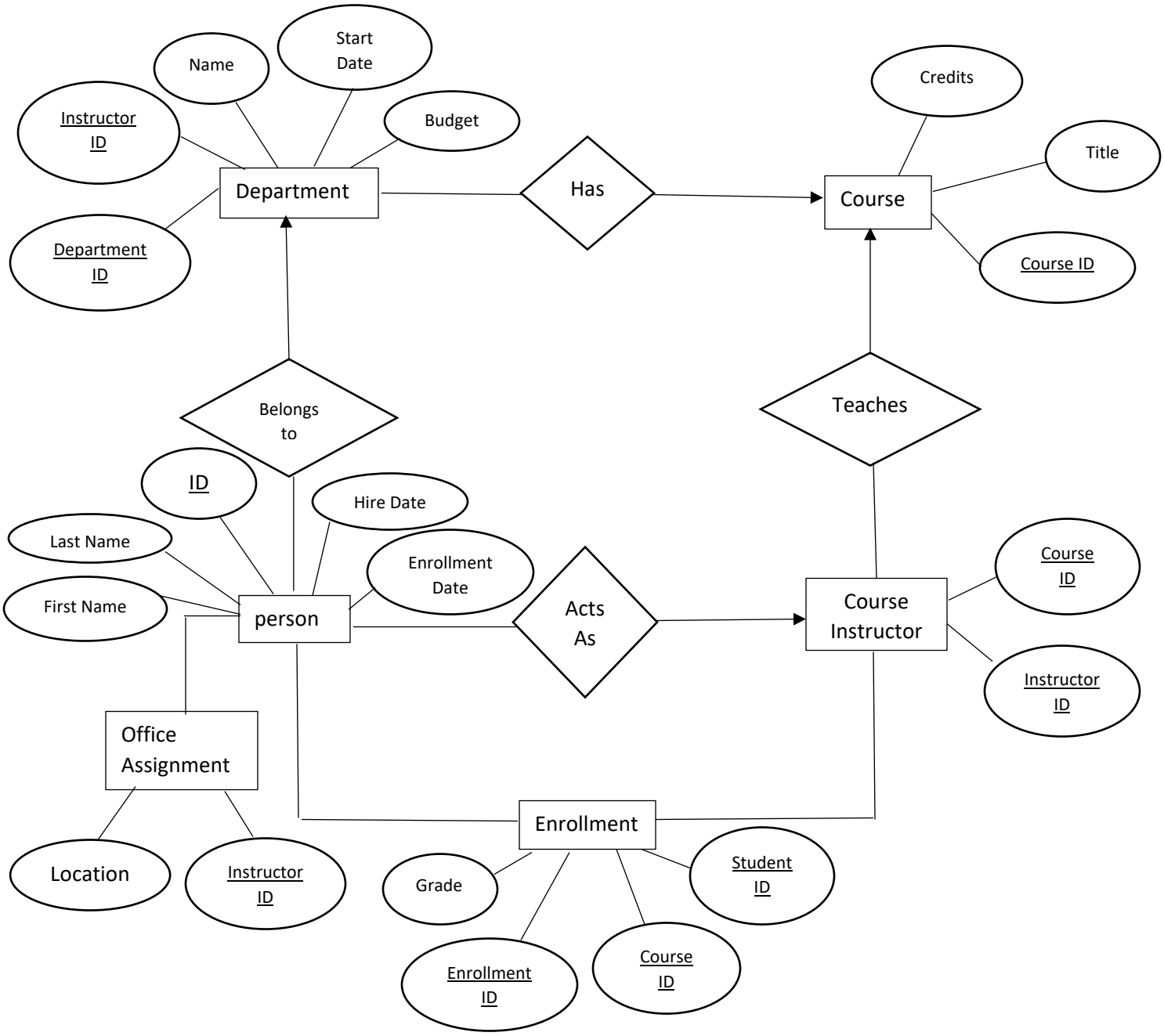
For **Enrollment** entity, we have attributes like: Enrollment ID, Course ID, Student ID and Grade. Enrollment ID is unique for an enrollment. Enrollment ID is the primary key. Course ID is unique for a given course. Student ID unique for a particular student. Grade denotes the grade scored by a particular student in a particular course. Same grade can be scored by different students.

For **office assignment** entity, we have attributes like: Instructor ID and Location. Instructor ID denotes the ID of an instructor. It is unique for an instructor. Location specifies the location in which the instructor is located.

For this DB system we will have few relations as below:

1. A course instructor teaches a course. One course has only one course instructor while one instructor can teach more than one course. So, it is a many to one relationship.
2. A person belongs to a department. One person belongs to a particular department while a department has any person in it. So, it is a many to one relationship.
3. A department has a course. One course belongs to a department while one department has many courses in it. So, it is a many to one relationship.
4. A person as course instructor. One course instructor is a person while one person can be more than one course instructor. So, it is a many to one relationship.

Based on our above discussion, we can draw the E/R diagram as below:



C. Database schema and Normalization:

Transfer the E/R diagram to relations as below:

1. Department (Department ID , Instructor ID , Name , Budget , Start Date)
2. Belongs To (Department ID , ID)
3. Person (ID , First Name , Last Name , Hire Date , Enrollment Date)
4. Office Assignment (Instructor ID , Location)
5. Acts As (ID , Instructor ID)
6. Enrollment (Enrollment ID , Course ID , Student ID , Grade)
7. Course Instructor (Course ID , Instructor ID)
8. Teaches (Instructor ID , Course ID)
9. Course (Course ID , Title , Credits)
10. Has (Dept ID , Course ID)

The Nontrivial FDs in each relation:

1. Department's FD : { Department ID -> { Name , Budget , Start Date }}
2. Belongs to 's FD : ϕ
3. Person's FD : { ID -> First Name , Last Name , Hire Date , Enrollment Date }
4. Office Assignment's FD : { Instructor ID -> Location}
5. Acts As's FD : ϕ
6. Enrollment's FD : {{Student ID , Course ID } -> Grade }
7. Course Instructor's FD : ϕ
8. Teacher's FD : ϕ
9. Course's FD : {Course ID -> { Title , Credits }}
10. Has's FD : ϕ

Checking whether the relations are in BCNF:

1. For Department, Department ID is the primary key. The left side are all super keys. So the relation is in BCNF.
2. For Belongs to, as the FD is an empty set it is in BCNF.

3. For person, ID is a primary key. The left side are all super keys. So the relation is in BCNF.
4. For office assignment, instructor ID is a primary key. The left side are all super keys. So, the relation is in BCNF.
5. For Acts as, The FD is an empty set .So it's in BCNF.
6. For Enrollment, enrollment ID is the primary key. While student ID and Course ID and student ID are foreign keys. As the left side are all super keys, the relation is in BCNF.
7. For course Instructor, FD is an empty set. So its in BCNF.
8. For Teaches, FD is an empty set. So its in BCNF.
9. For Has, FD is an empty set. So its in BCNF.

Therefore, we can conclude all these relations are in BCNF.

Since we have some many-one relationships, we can try to merge them to the "many" sides of entity.

After merging, we can get a set of new relations as:

1. Course (course ID, Department ID, Title, Credits)
2. Course Instructor (Course ID, Instructor ID)
3. Department (Department ID, Instructor ID, Name, Budget, Start Date)
4. Enrollment (Enrollment ID, Course ID, Student ID, Grade)
5. Office Assignment (Instructor ID, Location)
6. Person (ID, Last Name, First Name, Hire Date, Enrollment Date)

The Nontrivial FDs in these new relations are:

1. Course's FD: { Course ID -> {Title} , { Department ID , Course ID } -> Credits
2. Course Instructor's FD: ϕ
3. Department's FD : { Department ID -> { Name , Budget , Start date}
4. Enrollment FD : {Enrollment ID , Course ID , Student ID } -> Grade
5. Office Assignment : { Instructor ID -> Location }
6. Person { ID -> First Name , Last Name , Hire Date , Enrollment Date }

After the changes the relations are still in BCNF. It's hard to improve the schemas by 3NF and 4NF because the FDs of current schemas are mostly in the pattern that the primary key determine the other attributes.

D. Table creation:

Based on the discussion on the above section, We can design the SQL schemas like the following :

Table1: Course

Attribute Name	Data Type	Comment
Course ID	INT	Primary Key, Not Null
Title	Nvarchar(50)	Includes Null
Credits	INT	Not Null
Department ID	INT	Foreign Key Between Department and Course, Not Null

Table2: Course Instructor

Attribute Name	Data Type	Comment
Course ID	INT	Primary Key, Foreign Key to course, Not Null
Instructor ID	INT	Primary Key, Foreign Key to person, Not null

Table3:Department

Attribute Name	Data Type	Comment
Department ID	INT	Primary Key, Not null
Name	Nvarchar(50)	Null
Budget	Money	Not Null

Start Date	Datetime	Not Null
Instructor ID	INT	Null

Table4: Enrollment

Attribute Name	Data Type	Comment
Enrollment ID	INT	Not Null
Course ID	INT	Not Null
Student ID	INT	Not Null
Grade	INT	Null

Table5: Office Assignment

Attribute Name	Data Type	Comment
Instructor ID	INT	Primary key, Foreign Key to person, Not Null
Location	Nvarchar(50)	Null

Table6: Person

Attribute Name	Data Type	Comment
ID	INT	Primary Key, Not Null
Last Name	Nvarchar(50)	Not Null
First Name	Nvarchar(50)	Not Null
Hire Date	Datetime	Null
Enrollment Date	Datetime	Null

d**.Data collection:**

I chose to get my data randomly whereas, the key attributes here are IDs which are generated uniquely. So, they are said to be unique. Some of the interesting joins are:

Course Instructor uses course ID and Course Title together on courses column (join)
Department uses First name and last name together on administrator column (join)
Add/update a course (Insert/Update/delete)
Add/update a Department (Insert/update/delete)
Create new instructor (Insert)
Create new department (Insert)

e. User Interface and functions:

I used .netframework 4.5 with c# for developing the user interface and functions. MVC model framework is used with HTML CSS and c#. Various tabs like student, instructor, course, and department are used. Home displays the welcome screen. About displays the student body statistics.

Students: search by name, create new, edit, details and delete

Courses: select department, create new, edit, details and delete.

Instructors: create new, edit, select, details and delete.

Departments: create new, edit, details and delete.

Above are the basic functionalities provided by each tab in the application. The source file for SQL and Microsoft visual studio provides information on how these functionalities are implemented.

Discussion:

I enjoyed doing the project. I wanted to do a university application because, of all the theory that I learned in materials uses this as example. Making it in real time was a good experience. I have used the MCV framework earlier but integrating it with SQL is what I learnt from this project. Now I have the confidence to work on real time DB projects. Thank you for providing this opportunity and experience.