Student Information System Database Project - Project Step 8

Sheng Bian

URL: http://access.engr.oregonstate.edu:8259

Fixes based on Feedback from Step 1

Feedback by the Grader:

Recommend storing phone numbers as integer.

What I chose to fix:

I chose to change the type of phone number to integer.

What I chose not to fix and why I decided so:

I chose to fix everything so there is nothing to put in this section.

Other changes that I decided to make to the Step 1:

I decided to add participation of entities in relationships section. The purpose of this update is to make it clear of participation so that I can create the ER diagram correctly.

Fixes based on Feedback from Step 2 and 3

Feedback from the Grader:

Step 2:

ERD: Primary Keys not underlined (5)

I went back and looked at step 1, phone number was initially an integer. I am sorry I meant to write "recommend NOT storing phone numbers as integer," the reason is because integers can be manipulated with arithmetic, which you won't want, or storing this way might remove preceding 0's or do other undesirable things. Please use String or VARCHAR

Step 3:

SQL file syntactically correct. Data types appropriate. Foreign keys correctly defined. Relationship tables Are present.

Feedback from Peer Reviewer:

Step 2:

The ERD looks very well laid out and everything is present that is required except the primary keys are not underlined. The cardinality of relationships as well as participation matches the outline.

The Schema is good, except I believe in the Student_Course table all three columns should be underlined since studentNumber and courseCode come from primary keys.

As far as general comments, I think it would make your database easier to follow if instead of using names like number and courseCode as primary keys, you named then studentID, personID, paymentID, etc. That way all of your primary keys are both unique and follow the same pattern.

Very good explaination of the entities and their attributes. The ER diagram and schema match the outline.

My only suggestion would be for the outline and database itself. Normally a database like this would require contact information. Thus, I'd consider not allowing the phone number and email address to be blank. Maybe birthdate should not be blank either.

Step 3:

The SQL is syntactically correct, data types are appropriate, foreign keys are correctly defined, and relationship tables are present compared to the ERD and schema. There aren't any other suggestions that could be made as the database seems correct. This was verified by importing it into phpmyadmin.

What I chose to fix and how I fixed it:

I chose to change the type of phone number to VARCHAR.

I chose to underline Primary keys in ERD.

What I chose not to fix and why I decided so:

In schema, I chose not to underline all three columns in Student_Course table since it's not required by notation from SymbolKey.pdf.

I chose not to rename the primary keys to studentID, personID, paymentID, etc. The first reason is the it's not required by project specification. The second reason is courseCode like "CS165" is more descriptive than coudeID.

I chose not to change the phone number, email and birthdate to NOT NULL because this information usually can be blank in student information system.

Other changes that I decided to make on my own:

No other changes that I want to make.

Fixes based on Feedback from Step 4

Feedback by the Grader:

Looks good, no feedback this time, please proceed.

What I chose to fix:

None.

Fixes based on Feedback from Step 5

Feedback by the Peer Review:

Displaying entities worked. on all pages. However, I was unable to find a search function. The styling of the website could be improved, however that wasn't the focus of this step so I think it is acceptable.

What I chose to fix:

Add search function and improve style.

Fixes based on Feedback from Step 6

Feedback by the Peer Review:

I was able to add entries to entities using the provided forms for every entity except for the M-to-M relationship (Student-Courses). I was able to delete entries from every entity using the delete buttons for every entity, including the M-to-M relationship.

What I chose to fix:

Fix adding function in Student-Course page.

Fixes based on Feedback from Step 6

Feedback by the Grader:

None.

What I chose to fix:

None.

Updated Version: Project Outline + Database Outline

Project Outline

I will be making a database representing student information in a university. Student Information System is widely used by educational institutions to manage student data. Student Information System provides capabilities for checking student financial information, registering students in courses, documenting grades and etc. People can usually retrieve abundant useful information from Student Information Database by different queries. Additionally, Student Information Database has many different relationships between entities. The wide usage and complexity makes Student Information Database a great project candidate.

Database Outline

The entities in my database are:

Student – Student is the most important entity and has a relationship with any other entities in this database. It has the following attributes:

- number: The number of the student, which is a 10-digit integer. It's the primary key. It cannot be blank and there is no default.
- programCode: A programCode is the program that the student enrolls in. It is a string of maximum 8 characters. It cannot be blank and there is no default.
- isInternational: The Boolean value to show if the student is international student. It cannot be blank and the default value is false.
- collegeId: The collegeId of the student and it's different from number. The collegeId is 8-digit number. It cannot be blank and there is no default.
- balance: The balance in student financial account, which is a float number. It cannot be blank. It can be positive, negative and 0.

Person – The personal information of student is stored in my database using person entity.

- number: The number of the person, which is the same as the number of the student that person is associated to. Number is the primary key and it is a 10-digit integer. It cannot be blank and there is no default.
- lastName: The lastName of the person which is a string of maximum 100 characters. It cannot be blank and there is no default.
- firstName: The firstName of the person which is a string of maximum 100 characters. It cannot be blank and there is no default.
- phoneNumber: A phoneNumber is a string of maximum 10 characters. It can be blank and there is no default.
- email: The email of the person which is a string of maximum 100 characters. It can be blank and there is no default.
- birthdate: The birthdate of the person, which is stored as date. It can be blank and there is no default.

Course – The different courses that the students can register for using course entity.

- courseCode: A courseCode is the primary key, which is a string of maximum 10 characters. It cannot be blank and there is no default.
- name: Name of the course which is a string of maximum 100 characters. It cannot be blank and there is no default.
- hours: The hours of the course which is an integer of maximum number 80. It cannot be blank and there is no default.
- credits: The credits of the course which is an integer and it can only be 3, or 4, or 5 or 6. It cannot be blank and the default credits is 4.

Program – The different program information is stored in my database using program entity.

- programCode: A programCode is the primary key, which is a string of maximum 8 characters. It cannot be blank and there is no default.
- name: Name of the program which is a string of maximum 100 characters. It cannot be blank and there is no default.
- college: Name of the college the program belongs to, which is a string of maximum 100 characters. It cannot be blank and there is no default.

Payment – Each time students pay for the college, the transaction information will be recorded in database using payment entity.

- id: This number is automatically assigned to each payment when they are recorded in database. An auto-incrementing number which is the primary key. It cannot be blank.
- studentNumber: A studentNumber is the number of the student that the payment associated to. A studentNumber is a 10-digit integer. It cannot be blank and there is no default.

- invoiceNumber: An invoiceNumber is a 12-digit number. It is generated by the system and it cannot be blank.
- transactionDate: A transactionDate is the date that the student makes the payment. It is generated by the system and it cannot be blank.
- amount: Amount is a float number. The maximum amount is 99999. It cannot be blank and there is no default.

The relationships in my database are:

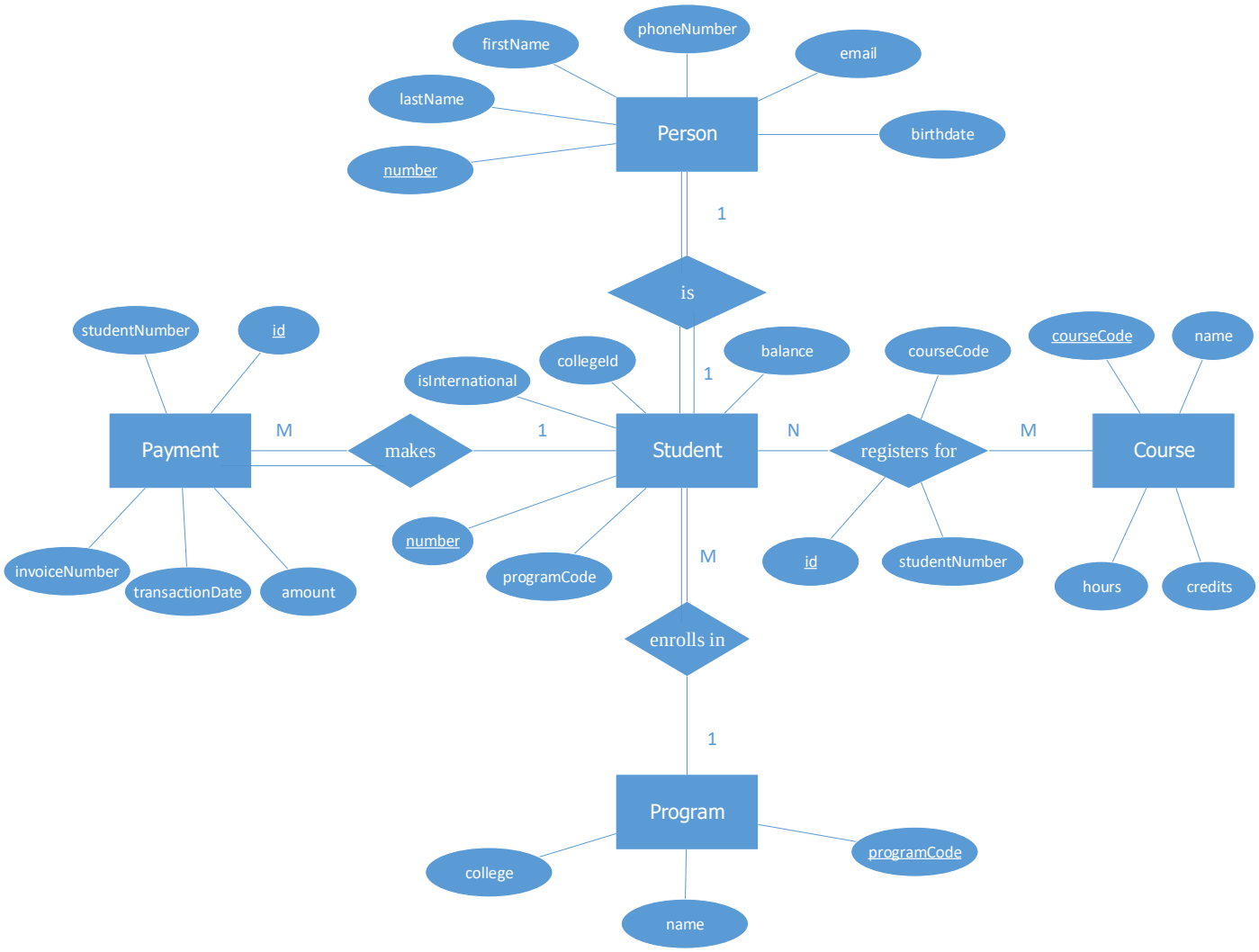
Student is person - A student can only be associated to one person. So, the student and person entities are in a one-to-one relationship. A student can't exist without being associated with a person but a person will always be associated with a student.

Student enrolls in program - A student can only enroll in one Program, but a program can have many students. So, this is a one-to-many relationship. A program can exist without being enrolled by any students but a student must enroll in a program.

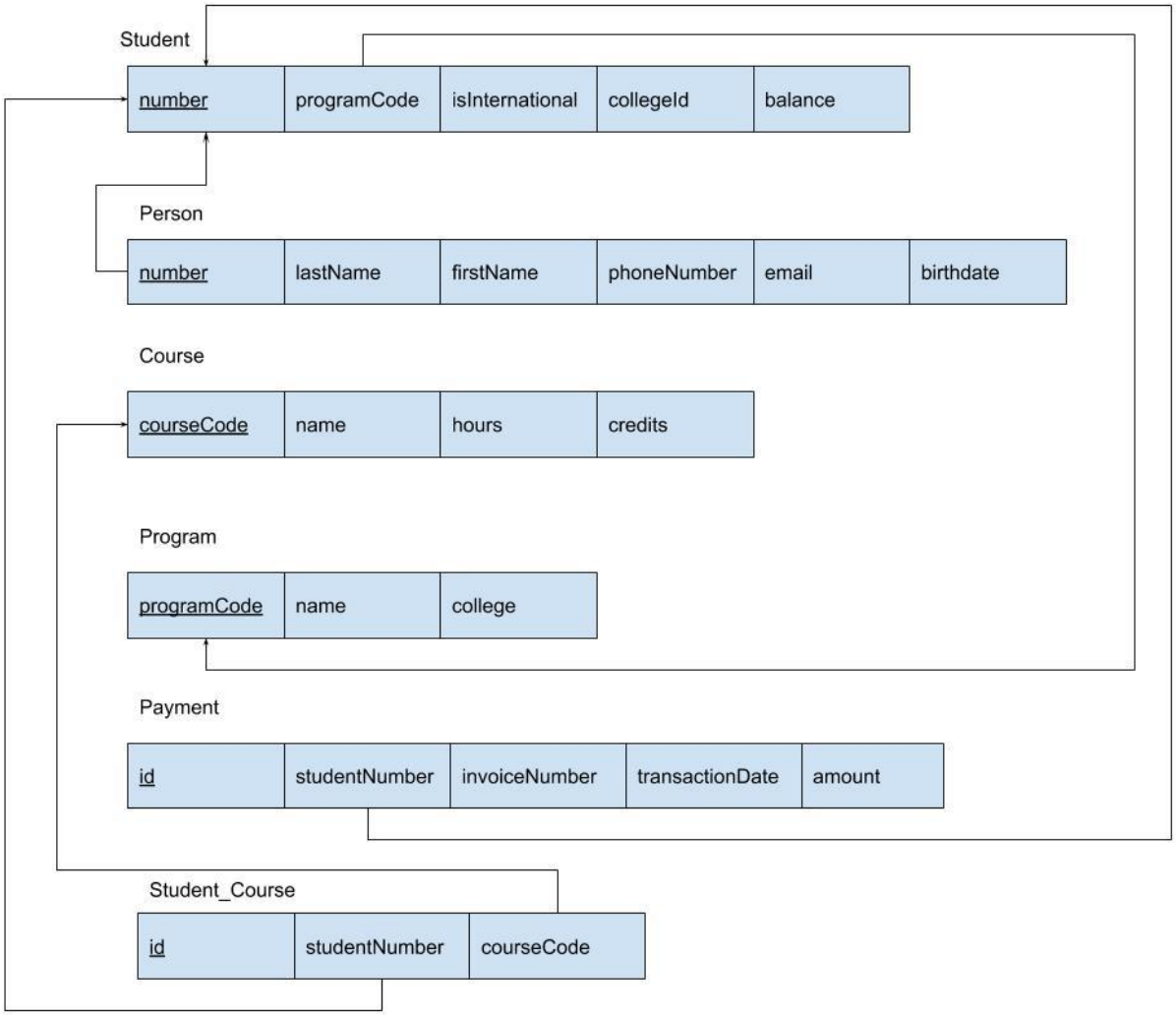
Student register for course - A student can register for many courses, and a course can have many students registered. So, this is a many-to-many relationship. A student can exist without registering a course and a course can exist without having any student being registered.

Student makes payment - A student can make many payments, but one payment is only associated with a student. So, this is a one-to-many relationship. A student can exist without making any payment but a payment must be associated with a student.

Entity-Relationship Diagram



Schema


Student
number
programCode
isInternational
collegeId
balance
Person
number
lastName
firstName
phoneNumber
email
birthdate
Course
courseCode
name
hours
credits
Program
programCode
name
college
Payment
id
studentNumber
invoiceNumber
transactionDate
amount
Student_Course
id
studentNumber
courseCode