Group 59
Group member names: Chris Sanchez, Mike Clancy

URL: http://flip2.engr.oregonstate.edu:5287/

Project Step 3 Draft Version: Student Records App

Fixes based on Feedback from Step 2

The following shows the feedback received from several reviewers including students and a TA. For completeness, I have copied all the feedback received during the entirety of the project.

The reviewer Carter Hawthorne mentioned that proper CASCADE operations were missing, so we went ahead and updated our SQL file by adding CASCADE ON DELETE SQL declarations. This reviewer also mentioned that the SQL DDL file looked messy, so we went ahead and cleaned up the file.

The reviewer Guyllian Dela Rosa, made some very good suggestions for improvement. We have implemented the following based on the reviewer's observation.

- The "state" attribute was added to the schema to match the outline. What had happened was that the schema snap shot was an outdated one.
- Under the Classes entity in the outline, the relationship details between the Classes and Professors was corrected
- Under the Classes entity in the outline, the relationship details between the Classes and Class Categories table was corrected
- We have changed the entity name for 'Class Categories' to 'Class_Categories' to include the underscore between the two words.
- The class_id attribute under the Professors entity in the outline was deleted to match the schema.
- Under the Registrations entity in the outline, the relationship details with the Students and Classes entities were corrected.

The reviewer Seongki Lee mentioned that there were a few crossed relationship lines in the schema but we do not have the same observation therefore we left the schema layout as it is.

Outside of any reviewer feedback we went ahead and corrected the relationship between Registrations and Classes to be M:1 instead of M:N to better match the schema

Fixes based on Prior Feedback

Clayton Loftus:

“Hi Mike and Chris,

“Nice job!

“Your overview clearly lays out the utility of your proposed student records application and database. The 1,000 student number helps me imagine about how many records you might be dealing with, however a rough number of classes and professors would also be helpful to think about how many related records there might be.

“You lay out four clear entities, each representing concrete, individual ideas and your outline does a good job describing the attributes and relationships of each.

“Your 1:M relationships look correctly laid out in the ERD diagram. I'm not sure the Registrations table relationship is diagrammed correctly, though. If I'm understanding it correctly, Registrations is essentially the intersection table between Students and Classes. So I'm not sure it's correct to show a 1:M relationship from Students to Registrations, an N:M relationship between Registrations and Classes, and also an N:M relationship between Students and Classes.

“Overall your naming looks consistent. One suggestion would be to simply use name as an attribute for Students, Classes, and Professors. So when you're referencing them you can simply use Students.name vs. Students.student_name.

“One other suggestion: you might consider adding a category table for Class_Types (or something) representing the classes, e.g. CS 340 is an instance of Class_Type, but maybe CS 340 Winter 2023 Section 400 is an instance of a Class that only happens once with a specific professor and set of students.

“Nice work so far. Good luck on your project!”

Eli Kurlbaum

“A student records system sounds like a situation where a website with a DB back en would be an excellent solution. It might have been a good idea to mention what you could do with the data in the database, rather than only mentioning what data it holds. This would give you a more clear direction to work towards.

“The overview lists that the university has 1,000 students. You could have also included other specific facts, such as how many professors and classes there are at the university, for a more clear picture of the problem you are solving.

“The following four entities are described: Students, Professors, Classes, Registrations. Each represents a single idea that will be stored as a list of various attributes for that entity. This part looks good to me.

“The outline details the purpose of each entity. For example, the description for students is, "records the details of all the students in the university." The attribute datatypes and constraints are also nicely listed, and there are relationships between each entity that are described.

“There are a couple of 1:M relationships (one between Students an Registrations, and another between Professors and Classes). Each one is logical and properly formulated. There are quite a few M:M relationships. The ERD does not show the intersection tables, but there should be some in the final product. The ERD diagram is logical, although a slight change I would make. I would move the entities so that there are no overlapping lines for the relationships since that is possible, although not something that is required; just a personal preference for readability.

“The entities are consistently capitalized and plural, and the attributes are lowercase and singular. I see no changes necessary with regards to your naming conventions.”

Zachary Johnson:

“Hi group 59, I thought this was interesting. You don't really list a problem or the need for a database/ the problem to be solved. It could be assumed that they are having difficulty keeping track of all of the students but it isn't clear on why the database is needed. You did have specific facts, and 4 entities that have a specific purpose. You did describe the items to be stored and there variable types, and also described the relationships. you hade M:M relationships and a proper 1:M relationships. You also had consistent with names, plurals, and capitalizations.

“I do see another issue with your project what about professors that are taking a class? What about post bachelor students that are studying and teaching? What about students that are PAs or TAs that are taking classes and helping grade and, or teach classes? I only throw these out there because I could see them coming up in a data base like this.

“Overall great work though!”

TA - James Cole

This is a great idea for a database and is an appropriate project for this course.

I do have some suggestions for improvement though:

The relationships lines in the ERD shouldn't be crossing over each other. Furthermore, the M:N relationship between Students and Classes is redundant as the Registrations table is already linking these two entities. If we remove the M:N relationship and the implied intersection table between Students and Classes, then that only leave you with 4 tables - which is not enough to meet the minimum project requirements. I propose adding a category table called 'ClassStandings' and connecting it to the Students table with a 1:M relationship. That additional table will fulfill the minimum requirements of the project.

I have drawn an ERD based on how I would suggest that you consider changing the database structure. It can be viewed here: https://bashify.io/images/ij90QL

I believe that the Overview could benefit from additional details. Perhaps consider explaining how each of the tables interact with one another. Explicitly state how the database is solving a problem. I.e. how does this database benefit the registration department?

Good job so far! I look forward to seeing how your project matures over this term.

Actions based on the feedback:

a. We received feedback that we needed to state what problem our database was meant to solve as well as provide more detail regarding the number of classes and professors the university has. Therefore, we have expanded the project overview section to include these facts.
b. The reviewer, Clayton Loftus, suggested we add a class category table and we went ahead and made that change to the project outline. He also suggested that we simplify some of the attribute names, which also has been updated in our outline. Clayton also suggested that the Registrations table is essentially the intersection table between Students and Classes. Therefore, for simplicity sake we removed the direct relationship between Students and Classes and left the Registrations table to act as an intersection table to enforce the M:N relationship between Students and Classes.
c. The reviewer, Eli Kurlbaum, suggested we add the intersection tables to the ERD and also suggested that we prevent the intersection of lines in the ERD. This also has now been implemented in the project.
d. The reviewer, Zachary Johnson, mentioned certain edge cases where a professor may also be a student or a student may be a TA. Therefore, we decided not to make any change in the database but rather communicate the narrow scope the database is meant to solve in the project overview section. The database is not meant to include information regarding all university staff and the current university policy does not allow current students to be professors.

e. The reviewer, James Cole, who is a TA, suggested that the ERD should not be crossing each other and that just make the Registrations table as the intersection table enforcing the M:N relationship between Students and Classes. The reviewer also suggested adding another entity to fulfill the minimum entity requirement, therefore, we added the Class Categories tables. The reviewer specifically suggested we add a ClassStandings table but instead we believed that a Class Categories table would better serve the goal of the database. Also, as other reviewers have suggested, James suggested that we add more detail to the Overview section, which we have, as specified in the project outline.

Project and Database Outline - Updated Version

Several changes were made to this section based on Step 2 feedback.

Overview

Student records application stores student records for a small-sized technology-education focused university with 1,000 students which is to be used by the registration department of the university. This application will hold registration information such as students, classes, and their associated professors. The University currently has 35 professors on staff and about 100 classes available for the students to take. Due to the university's small size, no students are currently allowed to be professors, but the university is open to this allowance as the university grows in size. The university is hoping that this database will help in keeping track of statistics such as the current teacher to student ratio, which classes are the most popular, and what professors are teaching which classes. This should help the university plan whether they should hire more professors and expand the number of classes available. This database is used exclusively for these statistics and by no means will encompass the entirety of all statistics that the university desires to track. The university is very eager to continue offering personalized education to their students; therefore, this database will facilitate the accomplishment of such a mission.

Database Outline

a. Students: records the details of all the students in the university
   i. student_id: int, auto_increment, unique, not NULL, PK
   ii. name: varchar, not NULL
   iii. street_address: varchar, not NULL
   iv. city: varchar, not NULL
   v. state: varchar, not NULL

- vi. zip_code: varchar, not NULL
- vii. email: varchar, not NULL
- viii. phone_number: varchar, not NULL
- ix. Relationship:
    1. A M:N relationship between Students and Classes is implemented with the Registration table that serves as an intersection table with student_id as FK inside the Registrations table.

b. Classes: records the details of all the classes that the university offers
   - i. class_id: int, auto_increment, unique, not NULL, PK
   - ii. class_category_id: int, not NULL, FK
   - iii. name: varchar, not NULL
   - iv. current_enrollment: int, not NULL
   - v. max_enrollment: int, not NULL
   - vi. professor_id: int, not NULL, FK
   - vii. Relationship:
       1. A M:N relationship between classes and students is implemented with the Registration table that serves as an intersection table with class_id as FK inside the Registrations table.
       2. A M:1 relationship between Classes and Professors implemented with professor_id as FK inside Classes
       3. A M:1 relationship between Classes and Class_Categories implemented with class_category_id as a FK in the Classes table.

c. Registrations: shows details about students' registrations to classes
   - i. registration_id: int, auto_increment, unique, not NULL, PK
   - ii. student_id: int, not NULL, FK
   - iii. class_id: int, not NULL, FK
   - iv. date_time_of_registration: date, not NULL
   - v. semester: varchar, not NULL
   - vi. Relationship:
       1. A M:1 relationship between Registrations and Students implemented with student_id as a FK inside Registrations
       2. A M:1 relationship between Registrations and Classes implemented with class_id as a FK inside Registrations

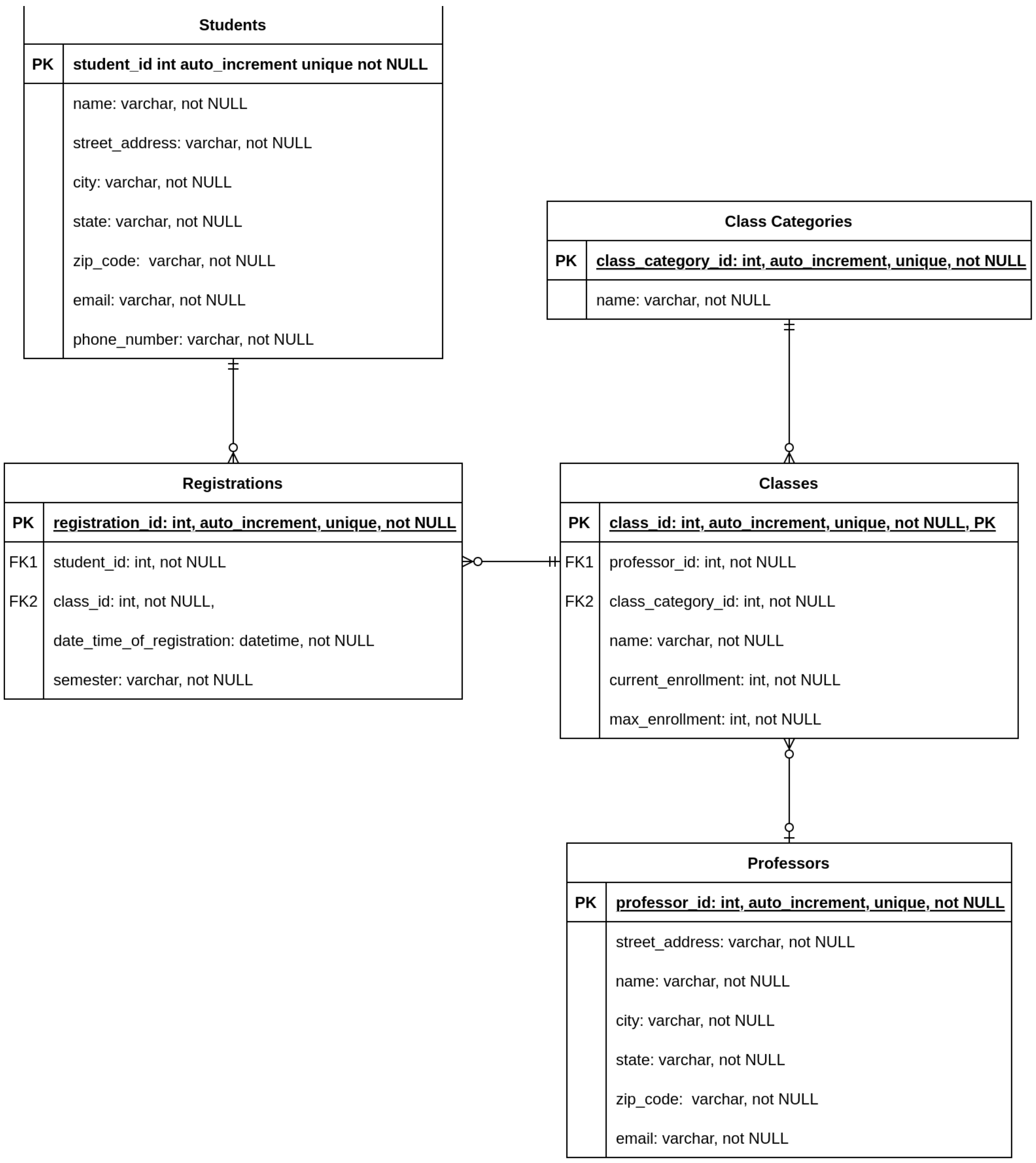
d. Class_Categories: this is a category table that shows what category a particular class belongs to
   - i. class_category_id: int, auto_increment, unique, not NULL, PK
   - ii. name: varchar, not NULL
   - iii. Relationship:

            1. A 1:M relationship between Class_Categories and Classes implemented with class_category_id as a FK inside Classes

    e. Professors: records the details of all the Professors in the university
        i. professor_id: int, auto_increment, unique, not NULL, PK
        ii. name: varchar, not NULL
        iii. street_address: varchar, not NULL
        iv. city: varchar, not NULL
        v. state: varchar, not NULL
        vi. zip_code: varchar, not NULL
        vii. email: varchar, not NULL
        viii. Relationship:
            1. A 1:M relationship between Professors and Classes implemented with professor_id as a FK inside Classes

Entity Relationship Diagram

No additional feedback was given to us after the submission of the Step 2 final draft. For completeness. No normalization changes have been made to this ERD. The ERD is copied below.



Students

PK	**student_id int auto_increment unique not NULL**
	name: varchar, not NULL
	street_address: varchar, not NULL
	city: varchar, not NULL
	state: varchar, not NULL
	zip_code: varchar, not NULL
	email: varchar, not NULL
	phone_number: varchar, not NULL

Class Categories

PK	**class_category_id: int, auto_increment, unique, not NULL**
	name: varchar, not NULL

Registrations

PK	**registration_id: int, auto_increment, unique, not NULL**
FK1	student_id: int, not NULL
FK2	class_id: int, not NULL,
	date_time_of_registration: datetime, not NULL
	semester: varchar, not NULL

Classes

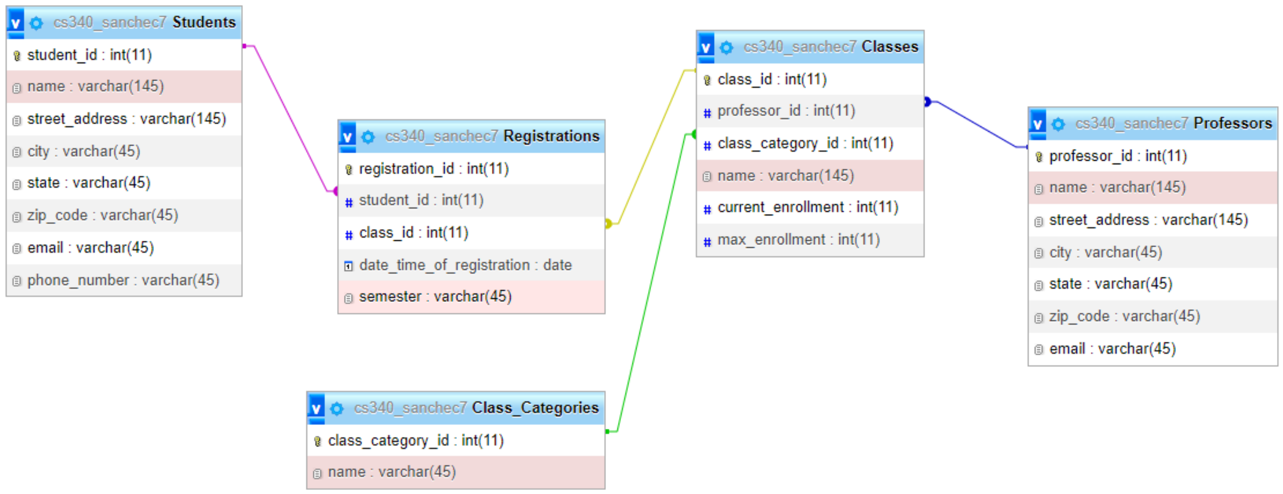
PK	**class_id: int, auto_increment, unique, not NULL, PK**
FK1	professor_id: int, not NULL
FK2	class_category_id: int, not NULL
	name: varchar, not NULL
	current_enrollment: int, not NULL
	max_enrollment: int, not NULL

Professors

PK	**professor_id: int, auto_increment, unique, not NULL**
	street_address: varchar, not NULL
	name: varchar, not NULL
	city: varchar, not NULL
	state: varchar, not NULL
	zip_code: varchar, not NULL
	email: varchar, not NULL

Schema

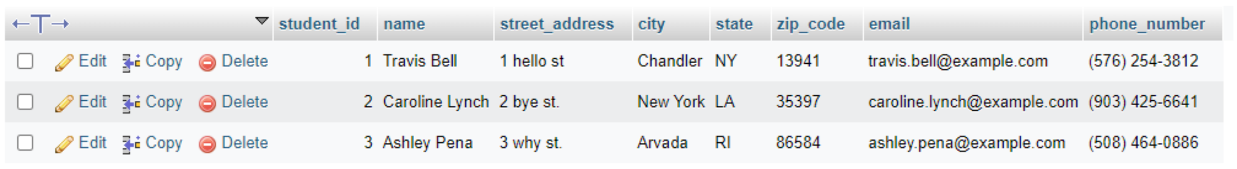
A change was made to the Schema based on Step 2 feedback.



Example Data

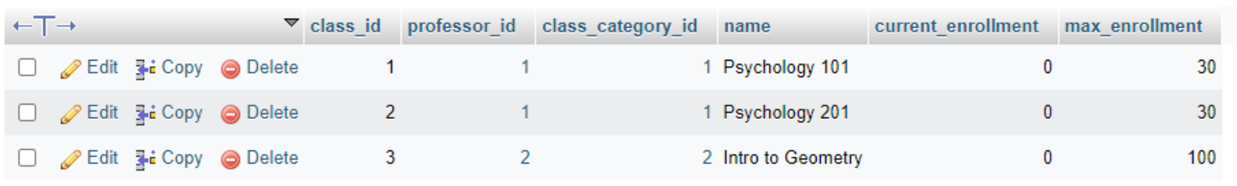
No changes were made to the Example Data based Step 2 feedback.

Students



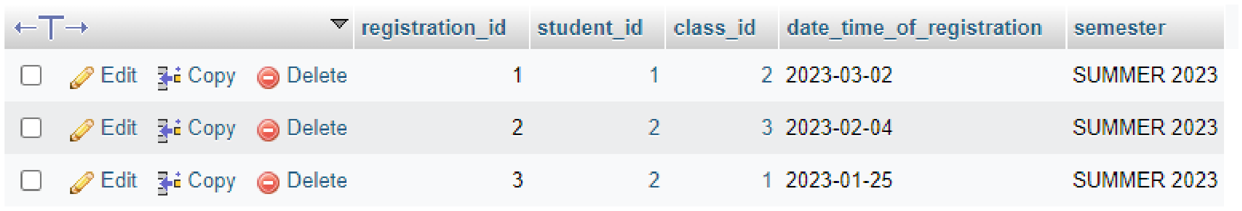
student_id	name	street_address	city	state	zip_code	email	phone_number
1	Travis Bell	1 hello st	Chandler	NY	13941	travis.bell@example.com	(576) 254-3812
2	Caroline Lynch	2 bye st.	New York	LA	35397	caroline.lynch@example.com	(903) 425-6641
3	Ashley Pena	3 why st.	Arvada	RI	86584	ashley.pena@example.com	(508) 464-0886

Classes



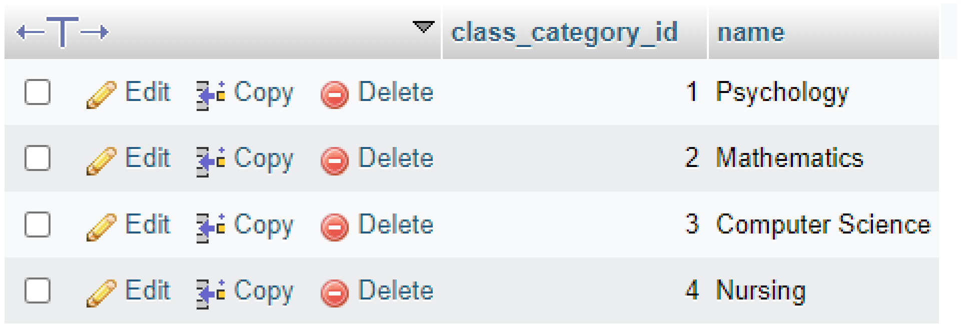
class_id	professor_id	class_category_id	name	current_enrollment	max_enrollment
1	1	1	Psychology 101	0	30
2	1	1	Psychology 201	0	30
3	2	2	Intro to Geometry	0	100

Registrations



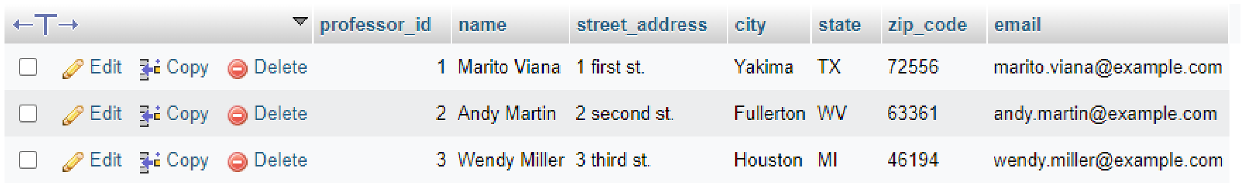
registration_id	student_id	class_id	date_time_of_registration	semester
1	1	2	2023-03-02	SUMMER 2023
2	2	3	2023-02-04	SUMMER 2023
3	2	1	2023-01-25	SUMMER 2023

Class_Categories



class_category_id	name
1	Psychology
2	Mathematics
3	Computer Science
4	Nursing

Professors



professor_id	name	street_address	city	state	zip_code	email
1	Marito Viana	1 first st.	Yakima	TX	72556	marito.viana@example.com
2	Andy Martin	2 second st.	Fullerton	WV	63361	andy.martin@example.com
3	Wendy Miller	3 third st.	Houston	MI	46194	wendy.miller@example.com