Milestone 2: The Relational Model

COLLEGE DEPARTMENT DATABASE MANAGEMENT SYSTEM

Name: Denzill Loe

SFSU ID: 920606619

GitHub Username: Denzill7

Milestone / Version	Date
M1V1	03/15/2022
M2	04/12/2022

Table of Contents

Section 1: Project Description

This project is about creating a college department database management system. Its purpose is to correctly handle the data of the college departments, the faculty members and their courses of instruction, the students' names, school ID's, classes, and much more. Students will likely have classes that their name will keep a place in, and the database needs to correctly handle properly assigning these values as efficiently as possible to ensure a clean and efficient database.

This project helps to ensure better organization of all the responsibilities of all the faculty within the department and hence the department can focus on other concerns than managing the large volume of data involved with students adding or dropping classes. This keeps things on track and makes the college department much more efficient as a result.

The College Department Database Management System takes into consideration the student's satisfaction with the department. The database will keep track of classes that still need spots to fill, and it will keep track of any complaints filed by students in order to improve the available classes. Whenever there is a complaint or a notable lack of classes for a certain course, the database management system will help the faculty know to increase the amount of classes for that particular course.

Section 2: Use Cases

1. Use Case: Classes Not Being Filled

Actor: Department Chair (Carmen), Course, Class, Student

Description: Carmen is a chairperson of a computer science and engineering department at a university. She is upset because she is unable to properly enroll students into classes, resulting in some classes being only partially filled. She is now looking for a solution to better organize the remaining classes left by prominently displaying classes with low student numbers.

The College Department Database System is there to help Carmen by keeping track of the classes within the department and keeping track of the amount of students enrolled. The database system will be able to assess which classes are in need of filling.

2. Use Case: Lack of Classes for a Course

Actor: Instructor (Jose), Course, Class, Student (Denzill)

Description: Denzill, a student at a university, is trying to enroll in classes within his department in the university for the upcoming semester. Sometimes when trying to enroll in a certain class, the sections fill up so quickly that it's impossible for a student to enroll in a timely manner. Students then struggle to obtain add codes from professors, causing untold stress on many students within the department.

The College Department Database Management System will help students search for all available sections for a given course and show them the availability of each class. If there are no classes that the student is able to enroll in, the database should give the student resources on how to contact a professor and get an add code for a given section. The database will make note of what courses students had difficulties enrolling in to properly evaluate the amount of sections needed for next semester.

3. Use Case: Students Adding Classes Late

Actor: Student, Instructor, Class, Course, Add Code

Description: Related to the above situation, students may find themselves unable to enroll in sections of a course before they fill up completely. The standard protocol is usually to find a way to attend the first class and ask the instructor for an add code to enroll in the section after the school semester has started.

If a student requests an add code from an instructor and they permit it, they can ask the database to give them a unique code for the student to submit and properly enroll their student account in the class.

4. Use Case: Advertising Events and Open Classes to Students

Actor: Department Ambassador, Students, Fliers

Description: The department wants to promote open classes to students within the university's department. However, the department struggles to find the best way to reach out to students during the busy school semester.

The College Database Management System can help the faculty by automatically sending out promotional events to the emails of members of the department.

5. Use Case: Promoting the Department to the University and Other Places

Actor: Chair (Carmen), Office Manager, Faculty, Social Media Account, Brochure

Description: Carmen runs a department at a university. She wants to give the department more exposure within the university and to outside students looking to pursue a college education. She has an office manager of the department operate a few social media accounts and set up flyers advertising events within the department to invoke outside interest. Furthermore, the ambassador prints out a bunch of brochures to hand out to the faculty of the department in case people want to learn more about the department.

Section 3: Database Requirements (Business Rules)

1. Student

A student shall be able to create at most one account.

A student shall have at least one major.

A student shall have at least one transcript.

A student shall be able to enroll in many classes.

A student shall have at most one account.

2. Instructor

An instructor is a faculty member.

An instructor shall be able to generate many add codes.

3. Faculty Member

A faculty member is an instructor, administrator, researcher, office manager, chancellor.

4. Course

A course shall have a course number.

A course shall have at least one class section.

5. Section (Class)

A class shall have many students

A class section can have only one add code.

A class section can be permitted to add with an add code.

A class shall be linked to an account

6. Account

An account shall belong to one student.

An account shall have at least one payment method.

An account is linked to many classes.

7. Payment Method

A payment method can be linked to many accounts.

A payment method can have many credit cards linked to it.

8. Credit Card

A credit card is linked to a payment method.

9. Major

A major shall be had by many students.

A major shall have many courses.

A major has a department.

10. Department

A department shall have at least one faculty member.

A department shall have at least one department ambassador.

A department shall have at least one major.

A department shall have at least one department chair.

11. University

A university shall have at least one department.

A university shall have at least one address.

A university shall have a school name.

12. Transcript

A transcript can have many courses.

Many transcripts can belong to one student.

13. Administrator

An administrator is a faculty member.

An administrator has an account

14. Researcher

A researcher is a faculty member.

A researcher is an instructor.

15. Office Manager

An office manager is a faculty member.

An office manager has a department.

16. Add Code

An add code can only belong to one class.

An add code permits adding a class section

17. Chancellor

A chancellor is an instructor.

A chancellor is a faculty member.

A chancellor has an associated university.

Section 4: Detailed List of Main Entities, Attributes, and Keys

1. Student (Strong)
    - school_id: key, numeric
    - name: composite, alphanumeric
        1. first
        2. last
    - dob: multivalue, timestamp
    - age: derived, numeric
2. Instructor (Strong):
    - name: composite, alphanumeric
        1. first
        2. last
    - num_courses: numeric
    - years_teaching: numeric
3. Faculty Member (Strong):
    - faculty_id: key, numeric
    - salary, alphanumeric
    - name: composite, alphanumeric
        1. first
        2. last
4. Course (Strong):
    - course_id: key, numeric
    - title: key, alphanumeric
    - prerequisites: multivalue, alphanumeric
    - num_classes: numeric
    - units: numeric

  - price: numeric

5. Class Section (Strong):
    - section_id: key, alphanumeric
    - num_add_codes: numeric
    - num_students: numeric
    - semester: alphanumeric
6. Department (Strong)
    - name: key, alphanumeric
    - department_id: key, numeric
    - num_instructors: numeric
7. University (Strong):
    - school_name: key, alphanumeric
    - address: composite, alphanumeric
    - founding_year: numeric
    - num_students: numeric
8. Account (Strong):
    - account_id: key, numeric
    - name: composite, alphanumeric
    - email: alphanumeric
9. Add Code (Strong):

- code_id: key, numeric
- course: alphanumeric
- expiration_date: multivalue, timestamp

10. Researcher (Strong):
    - r_id: key, numeric
    - name: composite, alphanumeric

- field_of_research: alphanumeric
- grant: numeric

11. Paper (Weak):
    - paper_id: key, numeric
    - title: alphanumeric
    - conference: alphanumeric
12. Administrator (Strong):
    - name: composite, alphanumeric
    - admin_id: key, numeric
    - salary: numeric
13. Transcript (Strong):
    - transcript_id: key, numeric
    - name: composite, alphanumeric
    - num_courses: numeric
    - gpa: numeric
14. Major (Strong):
    - name: alphanumeric
    - major_id: key, numeric
    - num_students: numeric
15. Payment Method (Weak):
    - type: alphanumeric
    - payment_id: key, numeric
16. Credit Card (Weak):
    - name_on_card: composite, alphanumeric
        1. first name
        2. last name

- card_number: key, numeric
- card_id: key, numeric
- expiration_date: multivalue, timestamp

17. Check (Weak):

- name_on_check: composite, alphanumeric

  1. first name
  2. last name
- routing_num: numeric
- amount: numeric
- c_id: key, numeric

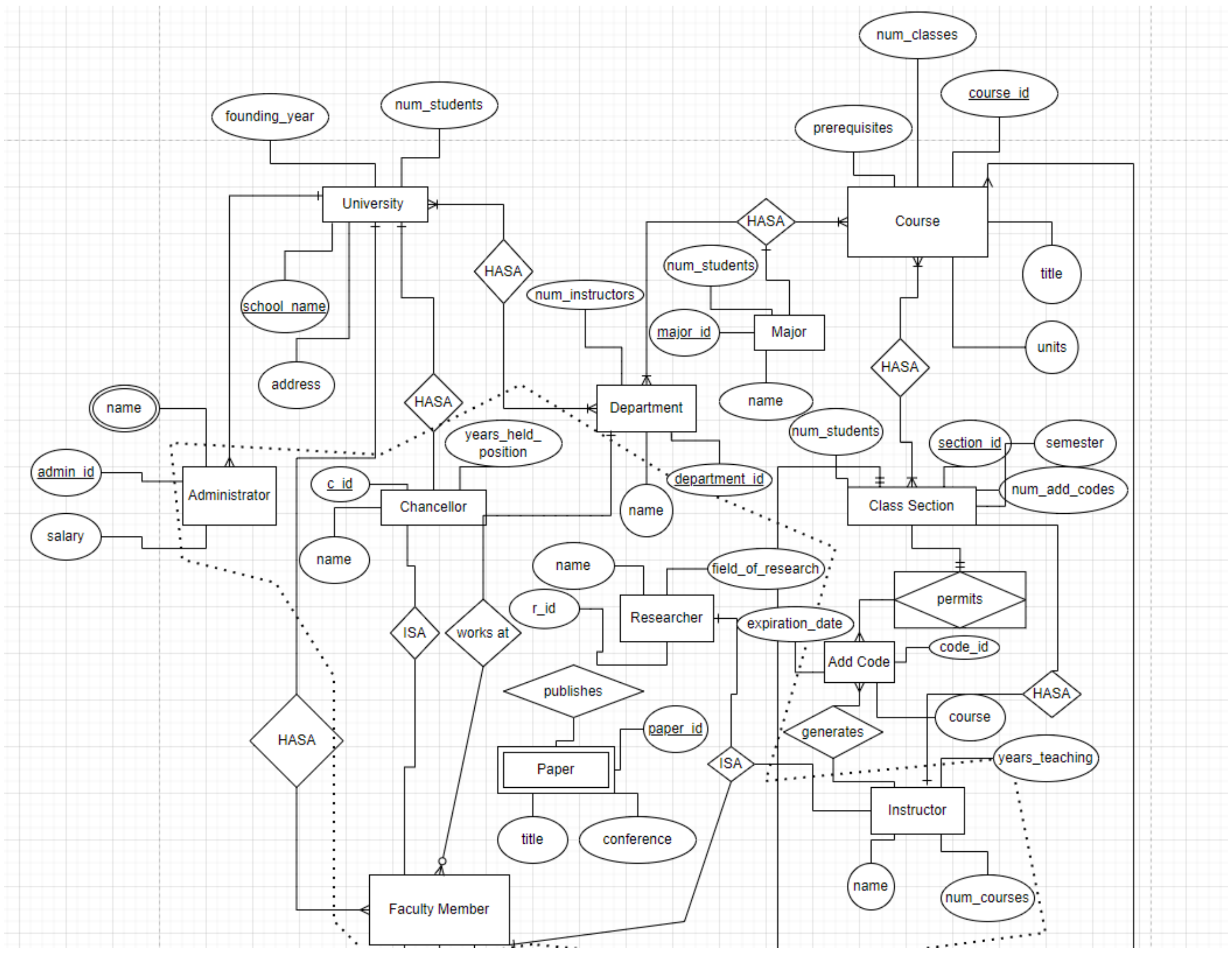
18. Chancellor (Strong):

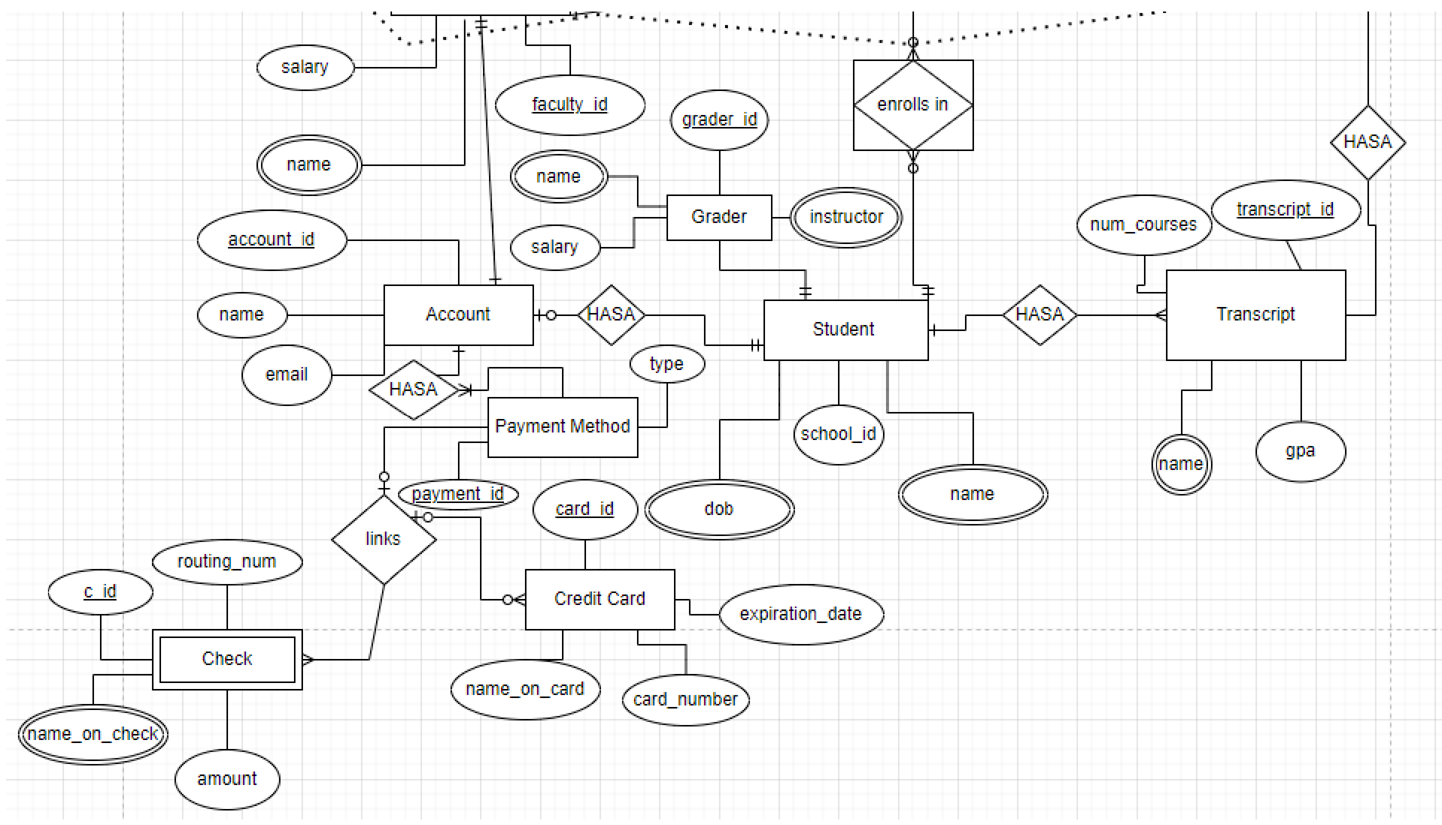
- c_id: key, numeric
- name: composite, alphanumeric
  1. first name
  2. last name
- years_held_position: numeric

19. Grader (Strong):

- grader_id: key, numeric
- salary: numeric
- name: composite, alphanumeric
  1. first name
  2. last name
- instructor: composite, alphanumeric
  1. first name
  2. last name

Section 5: Entity Relationship Diagram (ERD)

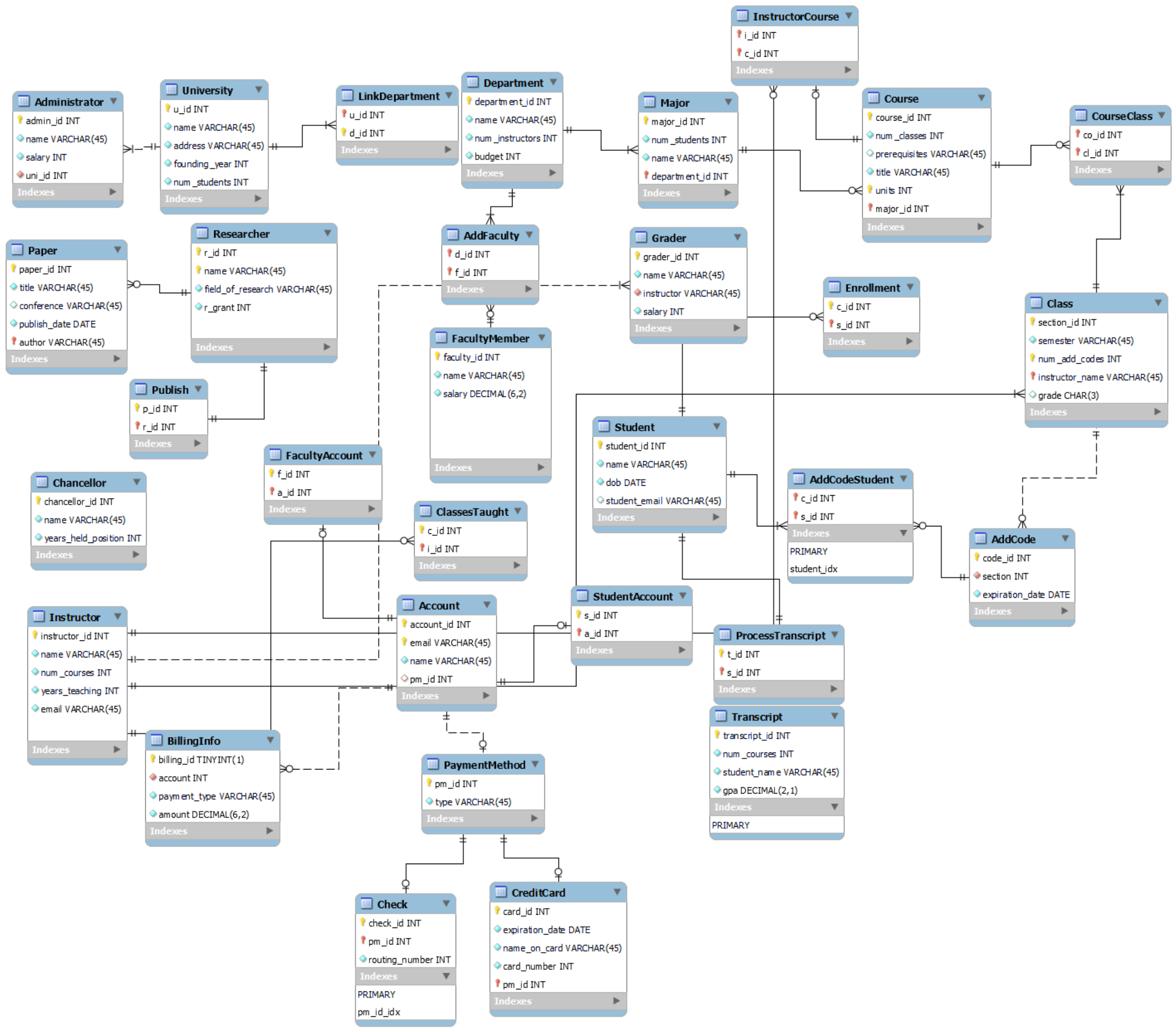



salary
faculty_id
name
grader_id
enrolls in
HASA
name
account_id
salary
Grader
instructor
num_courses
transcript_id
name
Account
HASA
Student
HASA
Transcript
email
HASA
type
Payment Method
school_id
name
gpa
payment_id
card_id
dob
name
links
routing_num
c_id
Credit Card
expiration_date
Check
name_on_card
card_number
name_on_check
amount

Section 6: Testing Table

Rule	Entity A	Relation	Entity B	Cardinality	Pass/Fail	Error Description
1	Student	create	Account	0-to-1	Fail	Student could create multiple accounts
2	Student	has	Major	M-to-1	Pass	
3	Student	has	Transcript	1-to-1	Pass	
4	Student	enrolls	Class	1-to-M	Pass	
5	Student	has	Account	1-to-1	Pass	
6	Instructor	generate	Add Code	1-to-M	Pass	
7	Course	has	Section	1-to-M	Pass	
8	Class Section	has	Student	1-to-M	Pass	
9	Account	has	Payment Method	1-to-1	Pass	
10	Payment Method	linked	Account	M-to-1	Pass	
11	Credit Card	linked	Payment Method	1-to-1	Pass	
12	Major	has	Student	1-to-M	Pass	
13	Major	has	Course	1-to-M	Pass	
14	Department	has	Faculty Member	1-to-M	Pass	
15	Department	has	Major	1-to-M	Pass	
16	University	has	Department	1-to-M	Pass	

Section 7: Database Model / EER



Note: I realized after the fact that you cannot have duplicate FK names, so some of these names were slightly changed after the making of this section.

Table	FK	ON UPDATE	ON DELETE	Description
LinkDepartment	university	CASCADE	CASCADE	If university is deleted, the departments linked with it should also be deleted.
LinkDepartment	department	CASCADE	CASCADE	If a department is deleted, it should no longer be linked to that university.

Major	department_id	CASCADE	CASCADE	If a department is updated or deleted, the majors belonging to that department should be updated or deleted.
Course	major_name	CASCADE	CASCADE	If the major_name is updated or deleted, the major_name in the course table should be updated or deleted as well.
CourseClass	course	CASCADE	CASCADE	If the course_id is updated, the course_id that links the class should be updated.
CourseClass	class	CASCADE	CASCADE	If the class_id is updated or deleted, the class belonging to that course should no longer exist in the db or be updated accordingly.
Class	instructor_name	CASCADE	CASCADE	If the name is updated, the instructor's name in the class table should be updated too. If it's deleted, delete it too.
ClassesTaught	c_id	CASCADE	CASCADE	Same reason as InstructorCourse
ClassesTaught	s_id	CASCADE	CASCADE	Same reason as InstructorCourse
Enrollment	c_id	CASCADE	NO ACTION	If the class's id gets deleted, we should still retain a record of the student enrolling in the class for transcript records. If the class_id the student is enrolling in gets changed, it should also be changed in this table to reflect that.
Enrollment	s_id	CASCADE	NO ACTION	If the student's id gets deleted, we should still retain a record of the student enrolling in the class for transcript records. If the student_id the student is enrolling with gets changed, it should also be changed in this table to reflect that.
AddCode	class_section	CASCADE	CASCADE	If the class_id is changed, the c_id in the add code should be changed as well. If the class is deleted, the add codes should be

				deleted too.
AddCodeStudent	code	CASCADE	NO ACTION	If the code’s id is updated, the code’s id in the AddCodeStudent table should be updated too. It should retain a record of adding the student if the add code gets deleted.
AddCodeStudent	student	CASCADE	NO ACTION	If the student’s id is updated, the student’s id in the AddCodeStudent table should be updated too. It should retain a record of adding the student if the student_id gets deleted.
InstructorCourse	instructor	CASCADE	CASCADE	If the instructor_id is updated it should be changed in here, If it’s deleted, it should be deleted from here because the instructor no longer teaches the same course
InstructorCourse	course	CASCADE	CASCADE	If the course_id is updated, it should be changed here. If it’s deleted, then the instructor can’t teach the course so it should be deleted.
AddFaculty	department	CASCADE	NO ACTION	If the department is deleted, the faculty member shouldn’t be deleted because they could be working for another department.
AddFaculty	faculty	CASCADE	CASCADE	If the faculty member is deleted from the table, the thing in this table should be deleted because they no longer work for the department.
FacultyAccount	faculty	CASCADE	CASCADE	If the faculty member’s id is deleted, their account should be deleted as well. Same for updating.
FacultyAccount	account	CASCADE	CASCADE	If their account’s id gets deleted, the faculty account shouldn’t exist anymore.
StudentAccount	student	CASCADE	CASCADE	If the student id gets changed or deleted, it should be reflected in here.

StudentAccount	account	CASCADE	CASCADE	If the account id gets changed or deleted, it should be reflected in here.
Account	s_name	CASCADE	NO ACTION	If the name of the student gets changed, the account name should get changed too, but if it gets deleted, their account should still stay in the table.
Account	i_name	CASCADE	NO ACTION	Same logic as above.
Account	pm_id	CASCADE	SET NULL	If the payment method is deleted, the account should simply remain without a payment method attached to it.
Student	email	CASCADE	NO ACTION	If the email in the account is deleted, the student should still keep the email.
Grader	instructor_name	CASCADE	NO ACTION	If the instructor's name is updated it should be updated here too. If it's deleted, nothing should happen.
Grader	student_id	CASCADE	CASCADE	If student id is deleted or updated, it should be reflected here.
Grader	student_name	CASCADE	CASCADE	If the student's name is deleted or updated, it should be reflected in this table.
Instructor	name	CASCADE	CASCADE	If the faculty member's name is deleted or updated, it should be reflected here.
Instructor	email	CASCADE	NO ACTION	Same logic as Student.email
Check	pm_id	CASCADE	CASCADE	If the payment method id gets deleted, the spot in the check table should also be deleted.
CreditCard	pm_id	CASCADE	CASCADE	Same logic as above.
Transcript	student_name	CASCADE	NO ACTION	If the student gets deleted from the Student table, the transcript should still remain intact as a record of their courses at the school.

ProcessTranscript	t_id	CASCADE	NO ACTION	If the transcript id is changed it should change in this table.
ProcessTranscript	s_id	CASCADE	NO ACTION	Same as above.
Chancellor	name	CASCADE	CASCADE	If they get deleted from the FacultyMember table, they should be deleted here too.
Researcher	name	CASCADE	CASCADE	Same reason as above.
Paper	author_name	CASCADE	NO ACTION	If the researcher gets deleted, their papers shouldn't get deleted.

Section 9: Testing Table

Entity	**SQLQuery**	**Pass / Fail**	**Error Description**	**Possible Solution**
University	DELETE	Pass	None	None
University	UPDATE	Pass	None	None
Department	DELETE	Fail	-- Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (\`collegedepartmentdb\`.\`courseclass\`, CONSTRAINT \`course\` FOREIGN KEY (\`co_id\`) REFERENCES \`course\` (\`course_id\`))	
Department	UPDATE	Pass	None	None
LinkDepartment	DELETE	Pass	None	None
LinkDepartment	UPDATE	Pass	None	None
Administrator	DELETE	Pass	None	None
Administrator	UPDATE	Pass	None	None
Major	DELETE	Fail	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (\`collegedepartmentdb\`.\`courseclass\`, CONSTRAINT \`course\` FOREIGN KEY (\`co_id\`) REFERENCES \`course\` (\`course_id\`))	
Major	UPDATE	Pass	None	None

Course	DELETE	Fail	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`collegedepartmentdb`.`courseclass`, CONSTRAINT `course` FOREIGN KEY (`co_id`) REFERENCES `course` (`course_id`))	
Course	UPDATE	Pass	None	None
Student	DELETE	Pass	None	None
Student	UPDATE	Fail	Error Code: 1062. Duplicate entry 'test' for key 'student.name_UNIQUE'	make this key not unique
ProcessTranscript	DELETE	Pass	None	None
ProcessTranscript	UPDATE	Fail	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`collegedepartmentdb`.`processtranscript`, CONSTRAINT `s_id_pt` FOREIGN KEY (`s_id`) REFERENCES `student` (`student_id`) ON UPDATE CASCADE)	
Transcript	DELETE	Pass	None	None
Transcript	UPDATE	Pass	None	None

Enrollment	DELETE	Pass	None	None
Enrollment	UPDATE	Fail	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`collegedepartmentdb`.`enrollment`, CONSTRAINT `s_id` FOREIGN KEY (`s_id`) REFERENCES `student` (`student_id`) ON UPDATE CASCADE)	
Account	DELETE	Fail	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`collegedepartmentdb`.`enrollment`, CONSTRAINT `s_id` FOREIGN KEY (`s_id`) REFERENCES `student` (`student_id`) ON UPDATE CASCADE)	
Account	UPDATE	Fail	Error Code: 1366. Incorrect integer value: 'test' for column 'pm_id' at row 5 (`account_id`))	
StudentAccount	DELETE	Pass	None	None
StudentAccount	UPDATE	Pass	None	None
FacultyAccount	DELETE	Pass	None	None

FacultyAccount	UPDATE	Pass	None	None
Instructor	DELETE	Fail	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`collegedepartmentdb`.`classestaught`, CONSTRAINT `i_id_ct` FOREIGN KEY (`i_id`) REFERENCES `instructor` (`instructor_id`))	
Instructor	UPDATE	Pass	None	None
ClassesTaught	DELETE	Pass	None	None
ClassesTaught	UPDATE	Pass	None	None
InstructorCourse	DELETE	Pass	None	None
InstructorCourse	UPDATE	Fail	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`collegedepartmentdb`.`instructorcourse`, CONSTRAINT `instructor_ic` FOREIGN KEY (`i_id`) REFERENCES `instructor` (`instructor_id`) ON DELETE CASCADE ON UPDATE CASCADE)	None
Class	DELETE	Pass	None	None
Class	UPDATE	Pass	None	None
CourseClass	DELETE	Pass	None	None

CourseClass	UPDATE	Pass	None	None
AddCode	DELETE	Fail	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (\`collegedepartmentdb\`.\`addcodestudent\`, CONSTRAINT \`code\` FOREIGN KEY (\`c_id\`) REFERENCES \`addcode\` (\`code_id\`) ON UPDATE CASCADE)	
AddCode	UPDATE	Pass	None	None
AddCodeStudent	DELETE	Pass	None	None
AddCodeStudent	UPDATE	Fail	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (\`collegedepartmentdb\`.\`addcodestudent\`, CONSTRAINT \`code\` FOREIGN KEY (\`c_id\`) REFERENCES \`addcode\` (\`code_id\`) ON UPDATE CASCADE)	
Researcher	DELETE	Fail	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (\`collegedepartmentdb\`.\`publish\`, CONSTRAINT \`researcher\`	

			FOREIGN KEY (\`r_id\`) REFERENCES \`researcher\` (\`r_id\`))	
Researcher	UPDATE	Pass	None	None
Publish	DELETE	Pass	None	None
Publish	UPDATE	Fail	-- Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (\`collegedepartmentdb\`.\`publish\`, CONSTRAINT \`researcher\` FOREIGN KEY (\`r_id\`) REFERENCES \`researcher\` (\`r_id\`))	
Paper	DELETE	Pass	None	None
Paper	UPDATE	Pass	None	None
FacultyMember	DELETE	Pass	None	None
FacultyMember	UPDATE	Pass	None	None
Chancellor	DELETE	Pass	None	None
Chancellor	UPDATE	Pass	None	None
Grader	DELETE	Pass	None	None
Grader	UPDATE	Pass	None	None
PaymentMethod	DELETE	Pass	None	None
PaymentMethod	UPDATE	Pass	None	None
CreditCard	DELETE	Pass	None	None
CreditCard	UPDATE	Pass	None	None
Check	DELETE	Pass	None	None
Check	UPDATE	Pass	None	None

BillingInfo	DELETE	Pass	None	None
BillingInfo	UPDATE	Pass	None	None