STUDENT RECORD KEEPING SYSTEM DATABASE

Luis Alfaro

920648466

lalfaro4

Milestone/Version	Date
M2V1	7/22/2021
M1V2	7/22/2021

Table of Contents

Table of Contents 2

Project Summary 3

Use Cases 4

Database Requirements 6

- Student 6
- Course 6
- Department 6
- Room 6
- City 7
- Mailing List 7
- Grader 7
- Advisor 7
- Payment 7
- School Program 7
- Exam 7
- Animal 7
- Professor 7
- Lecturer 8
- Book 8
- Resource Links 8
- Classroom 8
- Time Slot 8
- Section 8
- Grader 8
- Faculty Member 8
- Dorm 8
- Bank Account 8
- Credit Card 9
- Author 9

Main Entities, Attributes, and Keys 10

- After Testing Table 12

Entity Relationship Diagram 13

Testing Table 14

Constraints Table 18

Update/Delete Testing Table 22

Project Summary

The purpose of this project is to build a Student Record Keeping System Database for a university. This project aims to better manage student's records in order to allow the university to operate more efficiently. Students will be able to enroll/wailist a course. The database will take care of the books needed for courses in order to inform the students what books they should purchase. Students will be able to make payments for their fees using different payment methods. The database will save dorms in which a student can rent a room. With a mailing list, the university will be able to send the university news to Students that sign up for the mailing list. This database will better categorize time slots and sections for multiple courses to use and save memory. This can save a lot of time from people typing the time span of a class rather than picking a time slot.

Use Cases

1. **Use Case:** Adding a Course
   Actor: James (Student)
   Description: James already researched what classes he wants to take before his enrollment date this semester. It is now Jame's enrollment date so he decides to go to the university's website and log in. After logging in, James tries to add a course but it is full. James decides to look at the other section numbers for that course and adds the section 2 course to his cart and checks out with his debit card.

2. **Use Case:** Books Needed
   Actor: Lisa (Student)
   Description: Lisa already has her upcoming semester classes and is logged in to the University's student portal. Lisa likes to be prepared and notices that it says she has 4 books to purchase on one of her courses. Then Lisa decides to use the Author and Publisher information to search online stores for the books to purchase.

3. **Use Case:** Fees due
   Actor: Logan (Student)
   Description: Logan is already logged in to the website. Logan has fees that he has to pay before adding courses for next semester. Logan is trying to add 2 classes for next semester. Logan tries to add a course and he receives a message that he still has fees due. Logan then goes to pay his remaining dues and is asked whether he is going to pay by credit card or bank account. After paying his dues. Logan proceeds to add the course again. This time the computer science course he wanted to take is now full, so he decides to waitlist the course. He then checkouts his other course with a different payment method and is successful.

4. **Use Case:** Adding Housing
   Actor: Pam (Student)
   Description: Pam is looking for housing, in order to attend school. She notices that her courses are not on the main campus but in another campus in another city. Pam goes to the student portal and rents a dorm room that is close to her classes. Pam is able to choose from multiple dorms. After Pam picks a dorm, she then picks a room with her favorite number 13.

5. **Use Case:** New Faculty Member Hire
   Actor: Cam (Employer) Bob(New Lecturer)
   Description: Cam wants to hire Bob to be a faculty member for the university's Math department. The system asks Cam whether he wants Bob to be hired as an Advisor, Professor, or Lecturer. Bob is hired as a Lecturer who will teach no classes this semester. Next semester he is

scheduled to teach multiple math courses and a computer science course. Cam decides to add a new school program for students to join and discuss Bob's new ideas.

6. **Use Case:** Joining School Program
   Actor: Ridley (New Incoming Homeless Student)
   Description: Ridley is a new student who still is undecided in his major and doesn't want to complete a minor. He registers for the University mailing list to stay up to date with the University news. He is thinking about getting a major or two, one in the department of Computer Science. Ridley wants to take online courses and currently does not reside in a single city since he is homeless and always moving. Ridley decides to join a financial housing program provided by the school to help him afford housing.

7. **Use Case:** Creating A Course
   Actor: John **(**Faculty Member)
   Description: John wants to create a new course that is not currently being taught at the University. Due to the recent pandemic, John wants to make the class asynchronous so there is not designated Time Slot and Classroom for the course. John is trying to make some money, so he decided to write a book and make it required for his course. Usually, for one of John's veterinary classes, he gives students animals that they will be responsible for. Due to the pandemic, this class will not have animals assigned to students.

8. **Use Case:** Advisement
   Actor: Summer **(**Second Semester Student)
   Description: Summer is a 2nd-semester freshman that has to enroll in the Math Assessment Exam in order to find out what math course she will be placed in. After taking the exam, Summer's Advisor advises Summer to take Math 226. Summer looks at the Math 226 resource links in order to find out more information about the course. In order to pay her fees, she decides to become a Grader for a previous class she has taken.

Database Requirements

Student

1. A Student shall be enrolled in many Courses.
2. A Student shall waitlist many Courses.
3. A Student shall minor in 0 or 1 Department.
4. A Student shall major in 0 or 1 Department.
5. A Student shall rent many Rooms.
6. A Student shall reside in 0 or 1 City.
7. A Student shall signup to many Mailing Lists.
8. A Student shall be a Grader or not.
9. A Student shall be advised by 0 or 1 Advisor at a time.
10. A Student shall make many payments.
11. A Student shall join many School Programs.
12. A Student shall enroll in many Exams.
13. A Student shall research many Animals.

Course

1. A Course shall enroll many Students.
2. A Course shall be waitlisted by many Students.
3. A Course shall be taught either by 0 or 1 Professor or 1 Lecturer
4. A Course shall require 0 or 1 Book.
5. A Course shall link many Resource Links.
6. A Course shall be located in many Cities.
7. A Course shall have many Classrooms.
8. A Course shall meet during 0 or 1 Time Slot.
9. A Course shall have 0 or 1 Section.
10. A Course shall be graded by many Graders.

Department

1. A Department shall minor many Students.
2. A Department shall major many Students.
3. A Department shall appoint 1 or more Faculty Members
4. A Department shall have many School Programs

Room

1. A Room shall be rented by many Students.

2. A Room shall be contained by one Dorm.

City

1. A City shall have many Students who Reside in it.
2. A City shall have many Courses located in it.

Mailing List

1. A Mailing List shall be signup-ed by many Students.

Grader

1. A Grader ISA Student.
2. A Grader shall grade 0 or 1 Course at a time.

Advisor

1. An Advisor shall advise many Students.
2. An Advisor ISA Faculty Member

Payment

1. A Payment can be made by only 1 Student.
2. A Payment can be either a Bank Account or a Credit Card.

School Program

1. A School Program shall have many Students.
2. A School Program shall have 0 or 1 Department.

Exam

1. An Exam can be enrolled by many Students.

Animal

1. An Animal can be researched by many students.

Professor

1. A Professor shall teach many Courses.
2. A Professor ISA one and only one Faculty Member

Lecturer

1. A Lecturer shall teach many Courses
2. A Lecturer ISA one and only one Faculty Member

Book

1. A Book shall be required by many courses.
2. A Book shall be written by at least 1 Author.

Resource Links

1. A Resource Link shall be linked to many Courses

Classroom

1. A Classroom shall have many Courses.

Time Slot

1. A Time Slot shall be met by many Courses.

Section

1. A Section shall have many courses.

Grader

1. A Grader ISA student.
2. A Grader can grade many Courses.

Faculty Member

1. A Faculty Member is either a Professor, Lecturer, or Advisor.
2. A Faculty Member shall b appointed to 1 Department.

Dorm

1. A Dorm shall have many Rooms

Bank Account

1. A Bank Account ISA Payment

Credit Card

1. A Credit Card ISA Payment

Author

1. A Author shall write a Book.

Main Entities, Attributes, and Keys

1. Student (Strong)
    a. student_id: key, numeric
    b. name: alphanumeric
    c. dob: multivalue, timestamp
    d. assigned_animal: binary
    e. email: alphanumeric
    f. advisor: weak key, numeric
    g. gpa: numeric
    h. advisor: weak key, numeric
    i. monitor: weak key, numeric
2. Course (Strong)
    a. course_id: key, numeric
    b. section: key, numeric
    c. time Slot: weak key, numeric
    d. classroom: weak key, numeric
    e. resource_links: weak key, numeric
    f. city: key, numeric
3. Department (Strong)
    a. department_id: key, numeric
    b. name: alphanumeric
    c. email: alphanumeric
    d. phone_number: numeric
4. Room (Weak)
    a. room_id: key, numeric
5. City (Strong)
    a. city_id: key, numeric
    b. name: alphanumeric
    c. is_there_courses: binary
6. Mailing List (Strong)
    a. mail_id: key, numeric
    b. address: alphanumeric
    c. content: alphanumeric
7. Grader (Weak)
    a. grader_id: key, numeric
    b. student: weak key, numeric
    c. course: weak key, numeric
8. Advisor (Weak)
    a. faculty: key, numeric
9. Payment (Strong)
    a. payment_id: key, numeric
    b. amount: numeric

    c. student: weak key, numeric

10. School Program (Strong)
    a. program_id: key, numeric
    b. name: alphanumeric
    c. department: weak key, numeric
11. Exam (Strong)
    a. exam_id: key, numeric
    b. name: alphanumeric
    c. next_date: datetime
12. Animal (Strong)
    a. animal_id: key, numeric
    b. name: alphanumeric
    c. dob: multivalue, timestamp
    d. animal_type: alphanumeric
13. Professor (Weak)
    a. faculty: key, numeric
    b. first_taught: numeric
    c. years_taught: numeric
14. Lecturer (Weak)
    a. faculty: key, numeric
    b. first_taught: numeric
    c. years_taught: numeric
15. Book (Weak)
    a. isbn: key, numeric
    b. title: alphanumeric
    c. publisher: alphanumeric
    d. publish_date: Date
    e. category: alphanumeric
16. Resource Links (Strong)
    a. link_id: key, numeric
    b. url: alphanumeric
    c. title: alphanumeric
    d. category: alphanumeric
17. Classroom (Strong)
    a. classroom_id: key, numeric
    b. max_occupancy: numeric
    c. is_lab: binary
18. Time Slot (Strong)
    a. time_id: key, numeric
    b. start_time: time
    c. end_time: time
19. Section (Strong)
    a. Section_id: key, numeric
    b. total_sections: numeric

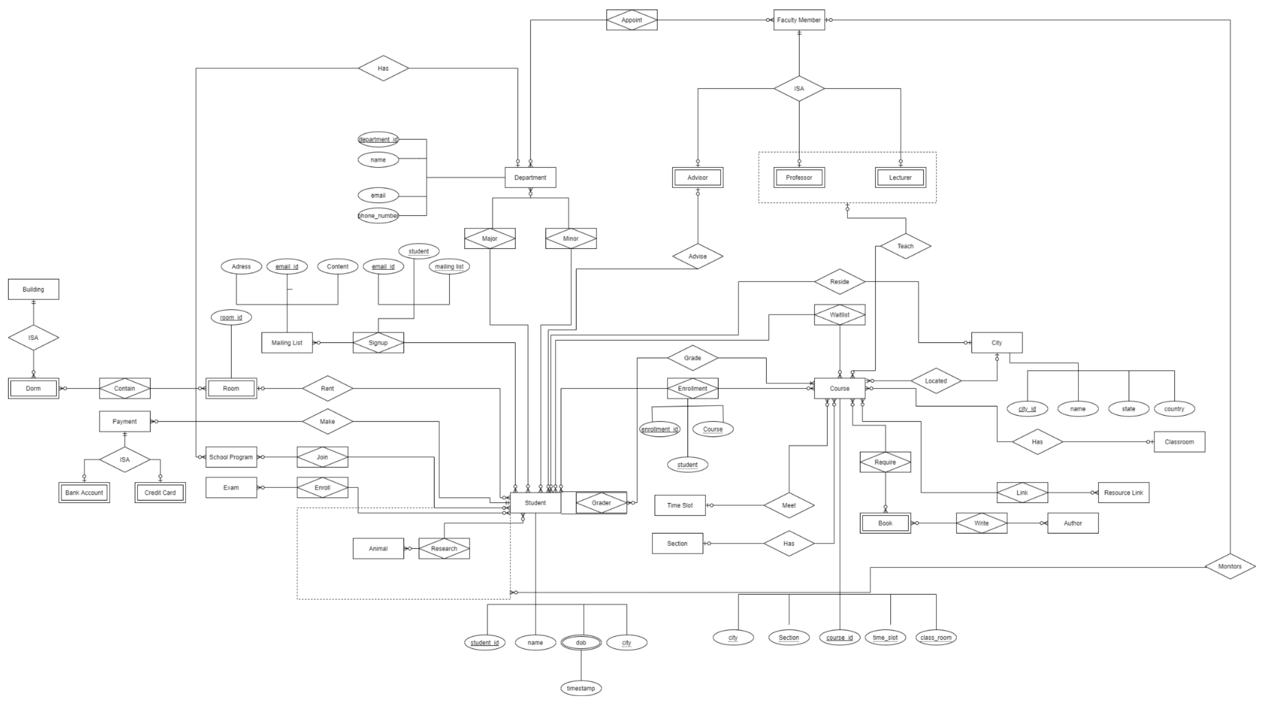
    c. next_available_section: numeric

20. Faculty Member (Strong)
    a. faculty_id: key, numeric
    b. name: alphanumeric
    c. first_joined: numeric
21. Dorm (Strong)
    a. dorm_id: key, numeric
    b. building: alphanumeric
    c. address: alphanumeric
    d. capacity: numeric
    e. residents: numeric
22. Bank Account (Weak)
    a. bank_id: key, numeric
23. Credit Card (Weak)
    a. credit_id: key, numeric
24. Author (Strong)
    a. author_id: key, numeric
    b. name: alphanumeric
    c. age: alphanumeric

After Testing Table

25. Building (Strong)
    a. building_id: key, numeric
    b. address: alphanumeric
    c. capacity: numeric
26. Dorm (weak)
    a. dorm_id: key, numeric
    b. building: weak key, numeric
    c. residents: numeric

Entity Relationship Diagram



Testing Table

Rule	Entity A	Relation	Entity B	Cardinality	Pass/Fail	Error Description
1	**Student**	Enroll	**Course**	M-to-N	pass	none
2	**Student**	Waitlist	**Course**	M-to-one	pass	none
3	**Student**	Minor	**Department**	M-to-one	fail	Students can have multiple Minors
4	**Student**	Major	**Department**	M-to-one	fail	Students can have more than 1 Major
5	**Student**	Rent	**Room**	M-to-N	fail	A Student can only rent 1 room at a time
6	**Student**	Reside	**City**	one-to-M	pass	none
7	**Student**	Signup	**Mailing List**	M-to-N	pass	none
8	**Student**	ISA/Recursive	**Grader**	none	pass	none
9	**Student**	Advised	**Advisor**	M-to-one	pass	none
10	**Student**	Make	**Payment**	one-to-M	pass	none
11	**Student**	Join	**School Program**	M-to-N	pass	none
12	**Student**	Enroll	**Exam**	M-to-N	pass	none
13	**Student**	Research	**Animal**	M-to-N	pass	none
14	**Course**	Enroll	**Student**	M-to-N	pass	none
15	**Course**	Waitlisted	**Student**	M-to-N	pass	none
16	**Course**	Taught	**Professor**	M-to-one	pass	none
17	**Course**	Taught	**Lecturer**	M-to-one	pass	none
18	**Course**	Require	**Book**	one-to-M	fail	A writing course can

						require many books
19	**Course**	Links	**Resource Links**	M-to-N	pass	none
20	**Course**	Located	**City**	M-to-N	fail	Course can only exist at 1 place at a time.
21	**Course**	Have	**Classroom**	M-to-N	fail	Course can only exist at 1 place at a time.
22	**Course**	Meet	**Time Slot**	M-to-one	pass	none
23	**Course**	Has	**Section**	M-to-one	pass	none
24	**Course**	Graded	**Grader**	M-to-N	pass	none
25	**Department**	Minor	**Student**	M-to-one	fail	A student can have more than 1 Minor at a time.
26	**Department**	Major	**Student**	M-to-one	fail	A student can have more than 1 Major at a time.
27	**Department**	Appoint	**Faculty Member**	M-to-one	fail	A faculty member can belong to more than 1 Department
28	**Department**	Has	**School Program**	M-to-one	pass	none
29	**Room**	Rented	**Student**	M-to-N	fail	A Student should only rent 1 room at a time.
30	**Room**	Contained	**Dorm**	M-to-one	fail	A room number can belong to multiple dorms.

31	**City**	Resides	**Students**	M-to-one	pass	none
32	**City**	Locates	**Course**	M-to-N	pass	Course should only exist at 1 place at a time
33	**Mailing List**	Signuped	**Students**	M-to-N	pass	none
34	**Grader**	ISA/Recursi ve	**Student**	none	pass	none
35	**Grader**	Grade	**Course**	M-to-N	pass	none
36	**Advisor**	Advice	**Student**	M-to-one	pass	none
37	**Advisor**	ISA	**Faculty Member**	one-to-one	pass	none
38	**Payment**	Made	**Student**	M-to-one	pass	none
39	**Payment**	ISA	**Bank Account**	one-to-one	pass	none
40	**Payment**	ISA	**Credit Card**	one-to-one	pass	none
41	**School Program**	Joined	**Student**	M-to-N	pass	none
42	**School Program**	Has	**Department**	M-to-one	pass	none
43	**Exam**	Enrolled	**Student**	M-to-N	pass	none
44	**Animal**	Researched	**Student**	M-to-N	pass	none
45	**Professor**	Teach	**Course**	M-to-one	pass	none
46	**Professor**	ISA	**Faculty Member**	one-to-one	pass	none
47	**Lecturer**	Teach	**Course**	M-to-one	pass	none
48	**Lecturer**	ISA	**Faculty Member**	one-to-one	pass	none
49	**Book**	Required	**Courses**	M-to-one	fail	A course can have multiple books required.
50	**Book**	Written	**Author**	M-to-N	pass	none
51	**Resource Link**	Linked	**Course**	M-to-N	pass	none

52	**Classroom**	Has	**Course**	M-to-N	fail	A Course can only exist at 0 or 1 classroom at a time.
53	**Timeslot**	Met	**Course**	M-to-one	pass	none
54	**Section**	Has	**Course**	M-to-one	pass	none
55	**Grader**	ISA	**Student**	none	pass	none
56	**Grader**	Grade	**Course**	M-to-N	pass	none
57	**Faculty Member**	ISA	**Professor**	one-to-one	pass	none
58	**Faculty Member**	ISA	**Lecturer**	one-to-one	pass	none
59	**Faculty Member**	ISA	**Advisor**	one-to-one	pass	none
60	**Faculty Member**	Appointed	**Department**	M-to-one	fail	A faculty member can be part of multiple departments.
61	**Dorm**	Contain	**Room**	M-to-one	fail	There can be multiple dorms with the same room number.
62	**Bank Account**	ISA	**Payment**	one-to-one	pass	none
63	**Credit Card**	ISA	**Payment**	one-to-one	pass	none
64	**Author**	Write	**Book**	M-to-N	pass	none

Constraints Table

Table	FK	ON DELETE	ON UPDATE	DESCRIPTION
student	city	SET NULL	CASCADE	If a city gets renamed, it should be updated. If a City is deleted/ doesn't exist anymore we set null.
student	room	SET NULL	CASCADE	room is updated, we update it for student, but if a room is deleted a student can no longer live there.
student	advisor	SET NULL	CASCADE	Set null in order to not delete a student if an advisor is deleted, but cascade on update of the advisor.
student	faculty_member	SET NULL	CASCADE	Don't want to delete a student if we delete the faculty member that monitors the student.
minor	student	CASCADE	CASCADE	If a student is deleted, we want to delete their records in minor.
minor	department	CASCADE	CASCADE	If a department is deleted, we want to delete its records in minor.
major	student	CASCADE	CASCADE	If a student is deleted, we want to delete their records in major.
major	department	CASCADE	CASCADE	If a department is deleted, we want to delete its records in major.
appoint	faculty_member	CASCADE	CASCADE	If a faculty_member is deleted, their appointed records should be deleted.
appoint	department	CASCADE	CASCADE	If a department is deleted, its appointed records should be deleted.
school_program	department	SET NULL	CASCADE	When we delete a department, we don't want to delete all the school programs that belong to that department.
join	student	CASCADE	CASCADE	When a student is deleted, their records of school_programs joined should be deleted.
join	school_pr	CASCADE	CASCADE	When a school_program is deleted, we

	ogram			want to delete the record of students in that program.
sign_up	student	CASCADE	CASCADE	When a student is deleted, we don’t want them to be part of a mailing_list.
sign_up	mailing_li st	CASCADE	CASCADE	When a mailing_list is deleted the record should be deleted for students that are signed up to that mailing_list
contain	dorm	CASCADE	CASCADE	When a dorm is deleted, the contain table should delete the records which were part of the dorm.
contain	room	SET NULL	CASCADE	When a room is deleted, I set it to null in case that 1 dorm does have that room.
bank_accou nt	payment	CASCADE	CASCADE	When a payment is deleted/updated, the bank_accont should be deleted/updated too.
credit_card	payment	CASCADE	CASCADE	When a payment is deleted/updated, the credit_card should be deleted/updated too.
grader	student	CASCADE	CASCADE	When a student is deleted, the grader should be deleted too because a grader is a student
grades	grader	CASCADE	CASCADE	When the grader is deleted, the record of what that grader grades should be deleted.
grades	course	CASCADE	CASCADE	On deletion of a course, the records of students grading that course should be deleted too.
grades	student	CASCADE	CASCADE	On deletion of student, we should delete all records of a grader grading that student.
research	student	CASCADE	CASCADE	When a student is deleted, their record of researching an animal should be deleted.
research	animal	CASCADE	CASCADE	When an animal is deleted, the record of that animal being researched should be deleted.
enrollment	student	CASCADE	CASCADE	When a student is deleted, their enrollment record should be deleted.
enrollment	course	CASCADE	CASCADE	When a course is deleted, its enrollment

				record should be deleted.
waitlist	student	CASCADE	CASCADE	When a student is deleted, their waitlist record should be deleted.
waitlist	course	CASCADE	CASCADE	When a course is deleted, its waitlist record should be deleted.
enroll_exam	student	CASCADE	CASCADE	When a student is deleted, their exam enrollment record should be deleted.
enroll_exam	exam	CASCADE	CASCADE	When a exam is deleted, its exam enrollment record should be deleted.
advisor	faculty_member	CASCADE	CASCADE	When the faculty member fk is deleted, the child, advisor, should be deleted too.
lecturer	faculty_member	CASCADE	CASCADE	When the faculty member fk is deleted, the child, lecturer, should be deleted too.
professor	faculty_member	CASCADE	CASCADE	When the faculty member fk is deleted, the child, professor, should be deleted too.
lecturer_course	lecturer	CASCADE	CASCADE	When the lecturer is deleted, all the records of the courses they are teaching should be deleted.
lecturer_course	course	CASCADE	CASCADE	When the course is deleted, all the records of the course that is being taught should be deleted.
professor_course	professor	CASCADE	CASCADE	When the professor is deleted, all the records of the courses they are teaching should be deleted.
professor_course	course	CASCADE	CASCADE	When the course is deleted, all the records of the course that is being taught should be deleted.
course	city	CASCADE	CASCADE	If a city is deleted, the courses available at that city should be deleted too.
course	classroom	SET NULL	CASCADE	When a classroom is deleted, we want to set to null since the classroom can change and classroom is not required if online.
course	time_slot	SET NULL	CASCADE	When a time_slot is deleted, we want to set to null since the time_slot can change and time_slot is not required if online.

course	section	SET NULL	CASCADE	When a section is deleted, we want to set to null since the section can change, and section is not required for a course.
links	resource_link	CASCADE	CASCADE	When a resource_link, we need to delete the record of courses linking to the resource_link.
links	course	CASCADE	CASCADE	When a course is deleted, we need to delete the record of that course linking to resource_link
require	course	CASCADE	CASCADE	When a course is deleted, we want to delete the required record of books needed for that course.
require	book	CASCADE	CASCADE	When a book is deleted, we want to delete the required record of courses requiring that book.
write	author	NO ACTION	CASCADE	When an author is deleted, we still want to keep the record of books they wrote.
write	book	NO ACTION	CASCADE	When a book is deleted, we still want to keep the record of the book and its authors.

Update/Delete Testing Table

Entity	SQLQuery	Pass/ Fail	Error Description	Possible Solution
Course	DELETE	Pass	None	None
Course	FAIL	Pass	Capacity of a course does not increase when adding a student	Make default value 0 rather than null
Time_Slot	DELETE	Fail	Deletes courses associated with that Time_Slot	On delete we can set to null
Time_Slot	UPDATE	Pass	None	None
Classroom	DELETE	Pass	None	None
Classroom	UPDATE	Pass	None	None
Section	DELETE	Pass	None	None
Section	UPDATE	Pass	None	None
Resource_Link	DELETE	Passl	None	None
Resource_Link	UPDATE	None	None	None
Payment	DELETE	Fail	None	None
Payment	UPDATE	Pass	None	None
Building	DELETE	Pass	None	None
Building	UPDATE	Pass	None	None
Room	DELETE	Pass	None	None
Room	UPDATE	Pass	None	None
Mailing_List	DELETE	Pass	None	None
Mailing_List	UPDATE	Pass	None	None
School_Program	DELETE	Pass	None	None
School_Program	UPDATE	Pass	None	None

Department	DELETE	Pass	None	None
Department	UPDATE	Pass	None	None
Faculty_Member	DELETE	Pass	None	None
Faculty_Member	UPDATE	Fail	advisor_id is not updated when the faculty_id is updated.	Add a trigger on update to update the advisor_id as well.
Advisor	DELETE	Pass	None	None
Advisor	UPDATE	Fail	faculty_id not updated when advisor_id is updated	Add a trigger on update to update the faculty_id as well
Author	DELETE	Pass	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (\`universitydb\`.\`writes\`, CONSTRAINT \`FK_WRITE_AUTHOR\` FOREIGN KEY (\`author\`) REFERENCES \`author\` (\`author_id\`) ON UPDATE CASCADE)	Allow Author to be nullable in the Write table
Author	UPDATE	Fail	None	None
Book	DELETE	Fail	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (\`universitydb\`.\`writes\`, CONSTRAINT \`FK_WRITE_BOOK\` FOREIGN KEY (\`book\`) REFERENCES \`book\` (\`book_id\`) ON UPDATE CASCADE)	Should not delete Write record in order to save the record of Books written by Authors.
Book	UPDATE	Pass	None	None