Project
<Database Design of Project Allocation System>

Student Name : Bingbing Li
Student ID : 18210705
Email : Bingbing.li@ucdconnect.ie

UCD Computer Science

Table of Contents

List of Figures

1. Introduction

The selection and allocation of projects to students in colleges and universities is one of the important process of college management.

When the number of students choosing projects increases, it is more efficient and fair to use a computational solution to manage the allocation process. At the same time, the online project selection system brings great convenience for students to choose projects, and the database play a very significant role in the designing of the application system.

Databases can efficiently store very large numbers of records with taking up a little space. Information is very quick and easy to find. A well-designed database system can provide developers with much convenience and easy system design and coding. Additionally, a well-designed database system can ensure the security of the data, and the data integrity is ensured by using various constrains for data.

This report will demonstrate the design process of the underlying information architecture to support the allocation application.

2. Database Plan: A Schematic View

In this section, this report will offer a high-level view of the database and motivate its design, and the report is divided into three parts: it will firstly illustrate the principal entities as well as their main attributes, then it will demonstrate the key relations that connect them and provide an E-R diagram to show the database plan in the end of the section.

2.1 Principal Entities and Main Attributes

The database system will retain a limited view of entities' attributes for information-processing purposes, it will store important data, however, actors such as entity’s age, nationality and gender will be dispensed.

There are 4 entities in the database system: student, staff, project, and preference matrix.

(1) Student part:

The main attributes of student entity are: student-id, student's full name, student's stream and student's GPA.

Each student must have one specific and unique student-id, so student-id can identify a student's information in our database system, it should be the primary key in student's entity table.

The student's full name, because it has higher intelligibility than student id, it is also necessary to add the student's full name to the system.

The student's stream is a central conceptual feature of the student's entity: either CS-only, or DS+CS. Since a student has to be assigned to a single project that is appropriate for their chosen stream, for example, CS-only students have to be assigned projects that are appropriate for students in the degree's main stream; CS+DS students should be assigned projects that consider their major in DS.

Our post-computation part tends to first consider allocating the projects to students with high grade points, so when other system perform allocation operations on the project, they need to call the student's GPA attribute.

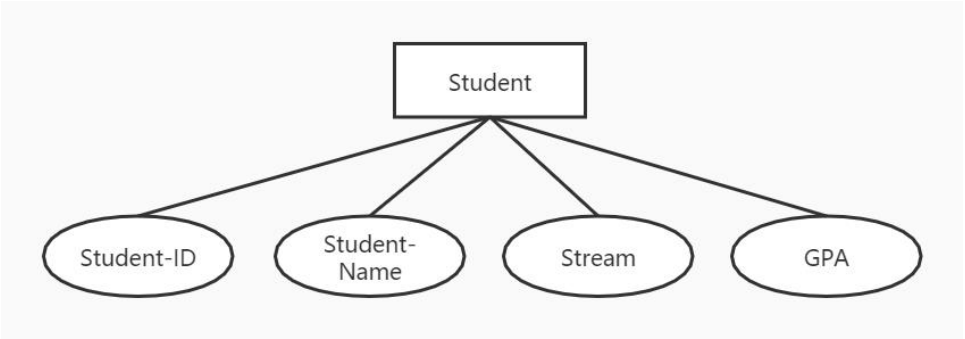


Figure 1. E-R diagram of Student Entity

(2) Staff part:

The main attributes of staff entity are: staff-id, staff's full name, and staff's stream.

Each staff must have one specific and unique staff-id, so staff-id can identify a staff's information in our database system, it should be the primary key in staff entity table.

The staff's full name is necessary, because it has higher intelligibility than staff id, and it It is convenient for students to know the teacher's information and contact their supervisor in the future.

The stream of staff member is needed, considering each staff member belongs to a specific research domain, and those who are teaching and researching in Cthulhu Studies (CS) will tend to propose only CS projects, while those who are teaching and researching Dagon Studies will tend to propose CS + DS projects, although either may also propose a project suitable for either stream students (CS or CS+DS).

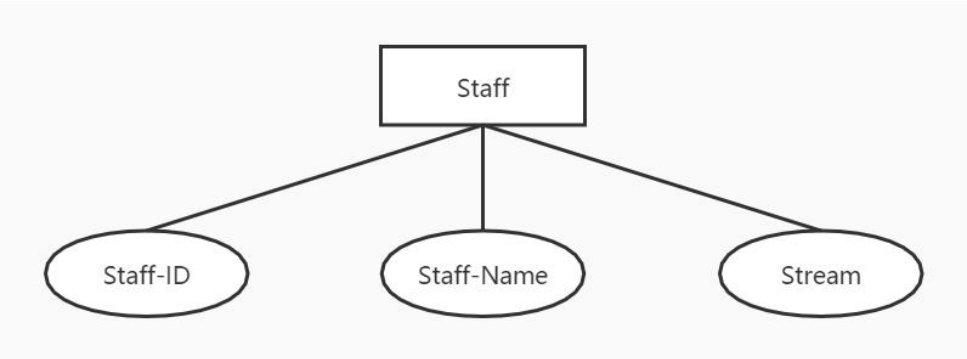


Figure 2. E-R diagram of Staff Entity

(3) Project part:

The main attributes of project entity are: project-id, project's title, supervisor' name, project's stream and the id of the self-specifying student (if any).

Each project must have one specific and unique project-id, so project-id can identify a project's information in our database system, it should be the primary key in project's entity table.

To help user's to read, the project's title should be displayed in other application, so store it in the database is also necessary, because the title is descriptive and unique, it can be one of the candidate key in the project's entity table.

In general, each project is proposed by a staff member,and supervisor's name of each project is important. for instance, to help user's such as student part to know the supervisor information.

Projects with the classification of CS or CS+DS are suitable for any and all students; those with either CS-only, or CS+DS, are suitable only for students in that particular domain. To facilitate student choosing their interested projects, the project's stream is necessary.

The allocation process allow students to raise their own self-proposed projects, so if there is, the id of student should be record in the project entity table.

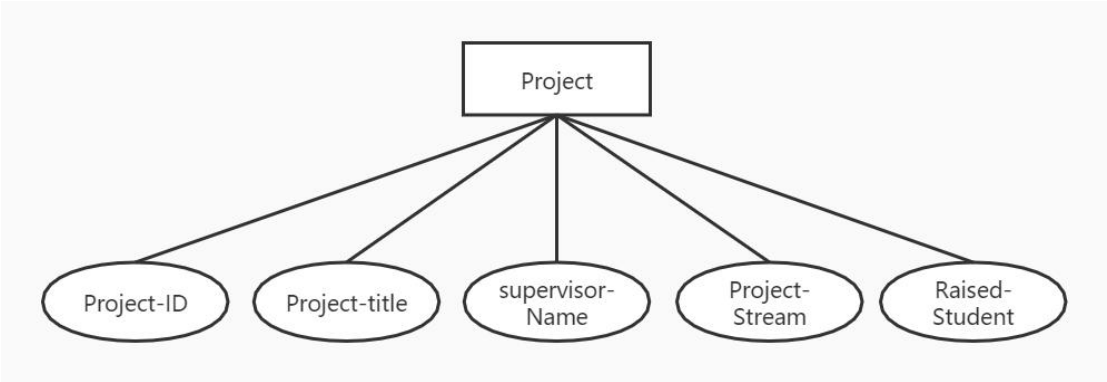


Figure 3. E-R diagram of Project Entity

(4) preference matrix:

The main attributes of preference matrix are: student-id, and 20 project-ids.

Each student will fill in the 20 project ids they most intend to choose, and the student-id will specify each row of preference matrix, so it should be the primary key of preference matrix table.

For each student, a corresponding row in the matrix contains their twenty preferred projects, from their most preferred (the first element in a row) to their least preferred (the last element in a row), so the preference matrix will have 20 columns of project-ids.

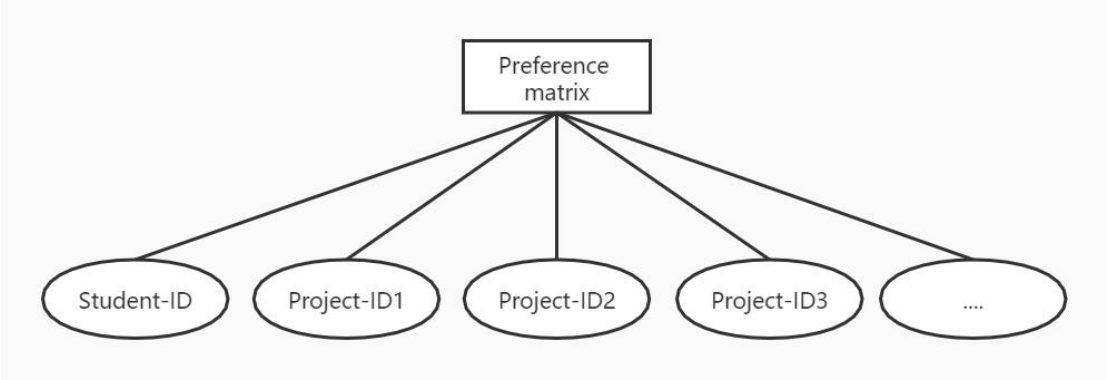


Figure 4. E-R diagram of Preference Matrix Entity

2.2 Key Relations connecting Entities

There are 3 key relations in the database system: project proposal, project allocation, and mentorship.

(1) Project proposal:

The project proposal relation is between projects and staff members.

One project belongs to one staff member who proposed it, while one staff member can propose many projects, so the relation between Staff to Projects is 1 to N.

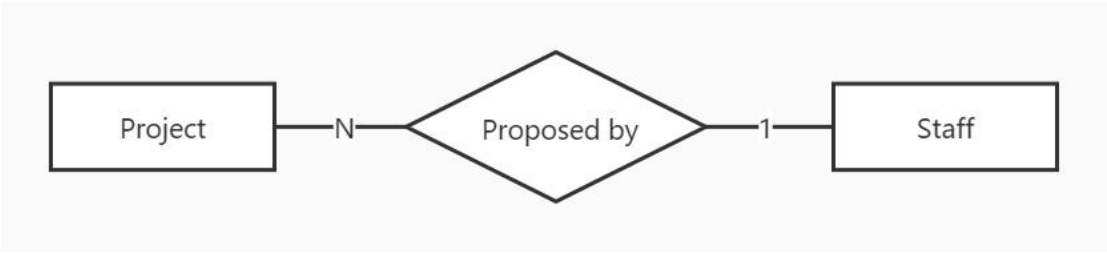


Figure 5. E-R diagram of Proposed by relation

(2) Project allocation:

The project allocation relation is between projects and students.

After the other high-level system computes the allocation process, one student will be assigned one distinct project, and one project will have one student doing it.

So the allocation relation between Student and Project is 1 to 1.

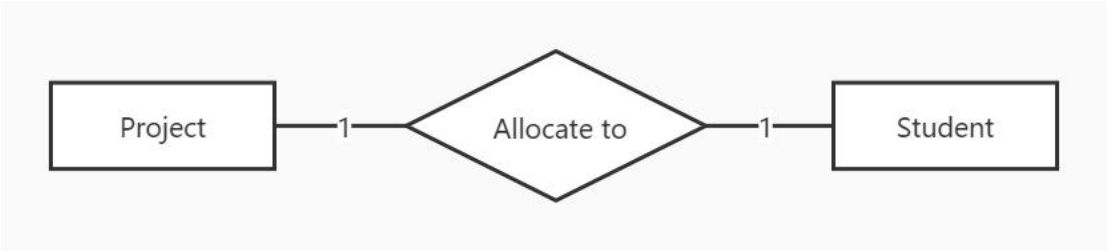


Figure 6. E-R diagram of Allocate to relation

(3) Mentorship:

The mentorship relation is between Staffs and students.
After a student is assigned a projects, he or she will have one staff member who proposed the project title as their mentor, however, one staff can mentor many students. So the mentorship relation between Staff and Student is 1 to N.

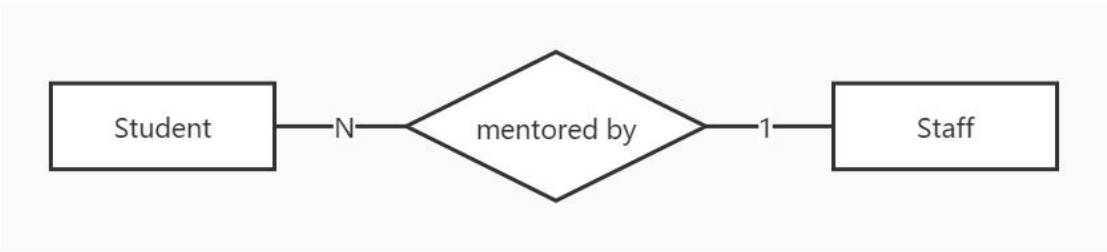


Figure 7. E-R diagram of Mentored by relation

2.3 E-R Diagram

After analyzing the main entities and attributes associated with them, as well as the main relations connecting the entities, a integrated E-R diagram can be generated.

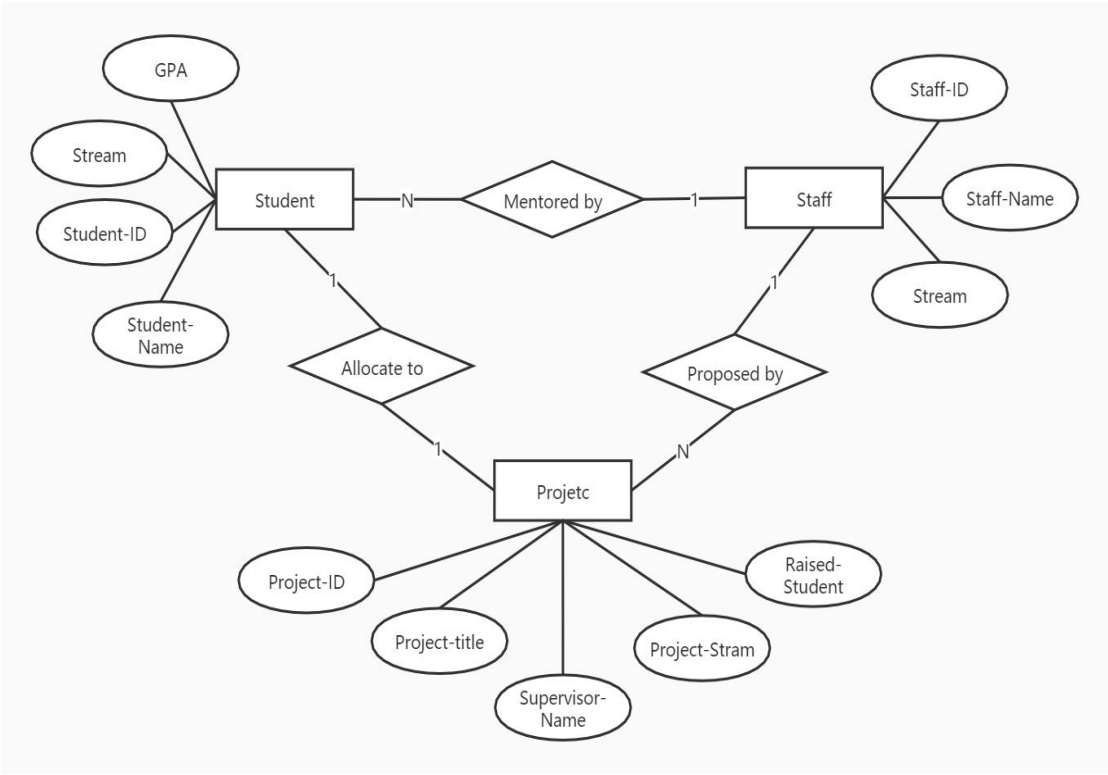


Figure 8. integrated E-R diagram

3. Database Structure: A Normalized View

In this section, this report will firstly give related key terms an explanation, then it will describe the main tables in the database system and the role played by each and prove that whether the main tables meet the condition of 1-NF, 2-NF,3-NF, or BCNF. Finally, after establishing the relationships between the tables and creating foreign key constrains, the report will display the Enhanced entity-relationship Diagram of the database system.

3.1 Explanation of Key Items

Before describing the main tables, we need to give the 1-NF, 2-NF, 3-NF, BCNF and some constrains in the database an explanation:

1-NF:

Ensure there are no redundant, repeating groups of data by imposing a unique

PRIMARY KEY on every table.[1]

2-NF:

a relation is in 2NF if it is in 1NF and every non-prime attribute of the relation is dependent on the whole of every candidate key, namely, it does not have any non-prime attribute that is functionally dependent on any subset of any candidate key of the relation.

3-NF:

A relation that is in First and Second Normal Form and in which no non-primary-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3NF).

If A->B and B->C are two FDs then A->C is called transitive dependency.

BCNF:

BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Foreign key:

The object of the foreign key is to identify a specific row of the referenced table, in some row of the primary table, it is usually necessary that the foreign key be equal to the candidate key.[2] This helps in referential integrity between the two tables in the database.

3.2 Description of Main Tables

According to the E-R diagram in Section 2, the conceptual model in the system can be converted into two kinds of tables : Map entities onto tables and map relations onto tables to connect entity tables.

3.2.1 Entity Tables:

(1) Student table:

Column Name	Information	Data type	Constrain
Student_ID	student's id	VARCHAR(7)	Primary key, NOT null
Student_Name	student's name	VARCHAR(20)	Not null
GPA	student's grade	VARCHAR(15)	Not null
Stream	CS/CS+DS	DECIMAL(3,2)	Not null

Figure 9. description of Student table

Student_ID	Student_Name	GPA	Stream
3906145	Will VonRueden	2.63	CS+DS
3912076	Cesar Borer	3.12	CS+DS
3973076	Tyson Lueilwitz	3.96	CS
3981423	Eda Armstrong	1.53	CS+DS
4093754	Darien Keebler	3.77	CS+DS
4146664	Omari Harvey	1.93	CS+DS
4150466	Hosea Boyer IV	2.00	CS
4165377	Verlie Kuhlman	2.62	CS
4175826	Janelle Wintheiser	2.83	CS
4239679	Esta Lubowitz	3.80	CS+DS

Figure 10. illustration Student table

Student table contains the necessary information relating to the process of project allocation of each student including: student-id, student's full name, student's GPA, and the stream of each student. It is one of the referenced table.

Constrains in the Student table:

Candidate key:{Student_ID}

Primary key:{Student_ID}

Functional dependencies in the Student table:

Student_ID -> {Student_Name, GPA, Stream}

The Student table is in 1-NF because each attribute has atomic values and one Student_ID can identify one record.

The Student table is in 2-NF because non-key attribute Student_Name, GPA and Stream are dependent on Whole key.

The Student table is in 3-NF because no non-primary-key attribute is transitively dependent on the Student_ID.

The Student table is in BCNF because the only one determinant Student_ID is the super key of Student table.

(2) Staff table:

Column Name	Information	Data type	Constrain
Staff_ID	staff's id	VARCHAR(7)	Primary key, NOT null
Staff_Name	staff's name	VARCHAR(20)	Not null
Research_Stream	CS/CS+DS	VARCHAR(15)	Not null

Figure 11. description of Staff table

Staff_ID	Staff_Name	Research_Stream
CS80000	Rashawn Berge	CS
CS80001	Thad Quigley	CS
CS80002	Christy Kris	CS
CS80003	Vita Cole	CS
CS80004	Jany Oberbrunner	CS
CS80005	Armando Kihn	CS
CS80006	Bernadette Romag	CS
CS80007	Jonathan Altenwerth	CS
CS80008	Tremayne Ledner	CS
DS91110	Alberto Russel	CS+DS
DS91111	Sheridan Considine	CS+DS
DS91112	Eldon Rowe	CS+DS
DS91113	Amaya Wuckert	CS+DS
DS91114	Jackeline Streich	CS+DS
DS91115	Mozell Frami	CS+DS
DS91116	Gretchen Bernhard	CS+DS
DS91117	Nadia Howe	CS+DS
NULL	NULL	NULL

Figure 12. illustration of Staff table

Staff table contains the necessary information of each staff menmbers including: staff-id, staff's full name, and the teaching and researching stream of each staff. It is one of the referenced table.

Constrains in the Staff table:

Candidate key:{Staff_ID}

Primary key:{Staff_ID}

Functional dependencies in the Staff table:

Staff_ID -> {Staff_Name, Research_Stream}

The Staff table is in 1-NF because each attribute has atomic values and one Staff_ID can identify one record.

The Staff table is in 2-NF because non-key attribute Staff_Name and Research_Stream are dependent on Whole key.

The Staff table is in 3-NF because no non-primary-key attribute is transitively dependent on the Staff_ID.

The Staff table is in BCNF because the only one determinant Staff_ID is the super key of Staff table.

(3) Project table:

Column Name	Information	Data type	Constrain
Project_ID	project's id	VARCHAR(5)	Primary key, NOT null
Project_Name	project's title	VARCHAR(60)	Not null
Supervisor_Name	supervisor's name	VARCHAR(20)	Not null
Stream	CS/CS+DS/ CS or CS+DS	VARCHAR(15)	Not null
Raised_Student	student's id	VARCHAR(7)	

Figure 13. description of Project table

Project_ID	Project_Name	Supervisor_Name	Stream	Raised_Student
DS000	Dagonmarathon apps	Alberto Russel	CS+DS	NULL
DS008	Dagon Online Exam System	Alberto Russel	CS+DS	NULL
DS003	Dagon p2p file sharing	Amaya Wuckert	CS+DS	NULL
DS011	Dagon Mobile phone supermarket	Amaya Wuckert	CS+DS	5139874
CS005	Cthulhu Electronic address book	Armando Kihn	CS	NULL
CSDS0	Human Resource Management System	Armando Kihn	CS or CS+DS	NULL
CS006	Cthulhu Personal finance system	Bernadette Romag	CS	NULL
CS002	Cthulhu lease Management System of Audio-vis...	Christy Kris	CS	4165377
CS011	Cthulhu Community Property Management System	Christy Kris	CS	NULL

Figure 14. illustration of Project table

Project table contains information of each Project including: Project_ID, Project_Name, Supervisor_Name, Stream, Raised_Student. It is one of the referenced table.

Constrains in the Project table:

Candidate key:{Project_ID, Project_Name}

Primary key:{Project_ID}

Functional dependencies in the Project table:

Project_ID -> {Project_Name, Supervisor_Name, Stream, Raised_Student}

Project_Name -> {Project_Name, Supervisor_Name, Stream, Raised_Student}

The Project table is in 1-NF because each attribute has atomic values and the primary key Project_ID can identify one record.

The Project table is in 2-NF because non-key attributes Supervisor_Name, Stream, Raised_Student are functionally dependent on the whole key(Project_ID).

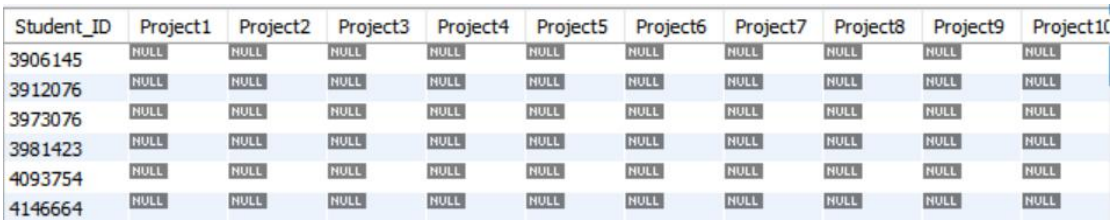
The Project table is not in 3-NF because non-key attributes Supervisor_Name, Stream, Raised_Student are not only dependent on Project_ID, but also dependent on one of the Candidate key Project_Name which means they are transitively dependent on the primary key (Project_ID).

Because the Project table is not in 3-NF, so it is not in the BCNF.

(4) Matrix table:

Column Name	Information	Data type	Constrain
Student_ID	student's id	VARCHAR(7)	Primary key, NOT null
Project1	first preferred project's id	VARCHAR(5)	
Project2	second preferred project's id	VARCHAR(5)	
...	...	...	...
Project20	last preferred project's id	VARCHAR(5)	

Figure 15. description of Matrix table



Student_ID	Project1	Project2	Project3	Project4	Project5	Project6	Project7	Project8	Project9	Project10
3906145	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3912076	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3973076	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3981423	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4093754	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4146664	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 16. illustration of Matrix table

Matrix table contains the 20 selected projects of all students. The columns of a table including: Student_ID, and project1, Project2, ...Project20.

Constrains in the Matrix table:

Candidate key:{Student_ID}

Primary key:{Student_ID}

Functional dependencies in the Matrix table:

Student_ID -> {Project1, Project2,...Project20}

The Matrix table is in 1-NF because each attribute has atomic values and one Student_ID can identify one record.

The Matrix table is in 2-NF because non-key attribute Project1..Project20 are dependent on Whole key.

The Matrix table is in 3-NF because no non-primary-key attribute is transitively dependent on the Student_ID.

The Matrix table is in BCNF because the only one determinant Student_ID is the super key of Matrix table.

3.2.2 Relation tables:

(1) Proposal table:

Column Name	Information	Data type	Constrain
Project_ID	project's id	VARCHAR(5)	Primary key, Foreign key, NOT null
Staff_ID	supervisor's id	VARCHAR(7)	Foreign key, NOT null

Figure 17. description of Proposal table

Project_ID	Staff_ID
CS000	CS80000
CS009	CS80000
CSDS1	CS80000
CS001	CS80001
CS010	CS80001
CS002	CS80002
CS011	CS80002
CS003	CS80003
CS012	CS80003

Figure 18. illustration of Proposal table

After gathering and reviewing of proposals from staff members, we can build a proposal table which the relation is appeared before the establishment of Project table, and we can use it to connect Staff table and Project table. The information of Proposal table includes: Project_ID and Staff_ID.

Constrains in the Proposal table:

Candidate key:{Project_ID}

Primary key:{Project_ID}

Foreign key:{Project_ID referenced table: Project table,

Staff_ID referenced table: Staff table}

Functional dependencies in the Proposal table:

Project_ID -> {Staff_ID}

The Proposal table is in 1-NF because each attribute has atomic values and one Project_ID can identify one record.

The Proposal table is in 2-NF because non-key attribute Stuff_ID are dependent on Whole key.

The Proposal table is in 3-NF because no non-primary-key attribute is transitively dependent on the Project_ID.

The Proposal table is in BCNF because the only one determinant Project_ID is the super key of Proposal table.

(2) Allocation table:

Column Name	Information	Data type	Constrain
Student_ID	student's id	VARCHAR(7)	Primary key, Foreign key NOT null
Student_Name	student's name	VARCHAR(20)	NOT null
project_ID	Project's id	VARCHAR(5)	Foreign key, NOT NULL

Figure 19. description of Allocation table

Student_ID	Student_Name	Project_ID
3906145	Will VonRueden	DS000
3912076	Cesar Borer	DS001
3973076	Tyson Lueilwitz	CS000
3981423	Eda Armstrong	DS002
4093754	Darien Keebler	DS003
4146664	Omari Harvey	DS004
4150466	Hosea Boyer IV	CS001
4165377	Verlie Kuhlman	CS002
4175826	Janelle Wintheiser	CS003

Figure 20. illustration of Allocation table

After the high-level application computes the satisfaction algorithm, and allocate a unique project to each student, we can create a relation between Student table and Project table, when mapping relation onto table, the information includes:

Student_ID, Student_Name, and Project_ID.

Constrains in the Allocation table:

Candidate key:{Student_ID, Project_ID}

Primary key:{Student_ID}

Foreign key:{Student_ID referenced table: Student table,

Project_ID referenced table: Project table}

Functional dependencies in the Allocation table:

Student_ID -> {Student_Name, Project_ID}

Project_ID -> {Student_ID, Student_Name}

The Allocation table is in 1-NF because each attribute has atomic values and one Student_ID can identify one record.

The Allocation table is in 2-NF because non-key attribute Student_Name are dependent on the whole key(Student_ID).

The Allocation table is not in 3-NF because non-key attributes(Student_Name) are not only dependent on Student_ID, but also dependent on Project_ID and because Project_ID dependents on Student_ID, so the non-key attribute Student_Name transitively dependent on the primary key (Student_ID).

Because the Allocation table is not in 3-NF, so it is not in the BCNF.

(3) Mentorship table:

Column Name	Information	Data type	Constrain
Student_ID	student's id	VARCHAR(7)	Primary key, Foreign key NOT null
Staff_ID	mentor's id	VARCHAR(7)	Foreign key, NOT null

Figure 21. description of Mentorship table

Student_ID	Staff_ID
3973076	CS80000
4857721	CS80000
5193123	CS80000
4150466	CS80001
4877188	CS80001
4165377	CS80002
5025900	CS80002
4175826	CS80003
5061857	CS80003

Figure 22. illustration of Mentorship table

After the allocation process, each student would have an mentor relationship with their supervisors, we can build a relation table named Mentorship connecting the relation between Student table and Staff table, including the information: Student_ID and Staff_ID.

Constrains in the Mentorship table:

Candidate key:{Student_ID}

Primary key:{Student_ID}

Foreign key:{Student_ID referenced table: Student table,

Staff_ID referenced table: Staff table}

Functional dependencies in the Mentorship table:

Student_ID -> {Staff_ID}

The Mentorship table is in 1-NF because each attribute has atomic values and one Student_ID can identify one record.

The Mentorship table is in 2-NF because non-key attribute Stuff_ID are dependent on Whole key(Student_ID).

The Mentorship table is in 3-NF because no non-primary-key attribute is transitively dependent on the Student_ID.

The Mentorship table is in BCNF because the only one determinant Student_ID is the super key of Mentorship table.

3.3 Enhanced Entity-Relationship Diagram

After the normalization of tables and adding constrains to tables, an enhanced EER-diagram can be generated as follows:

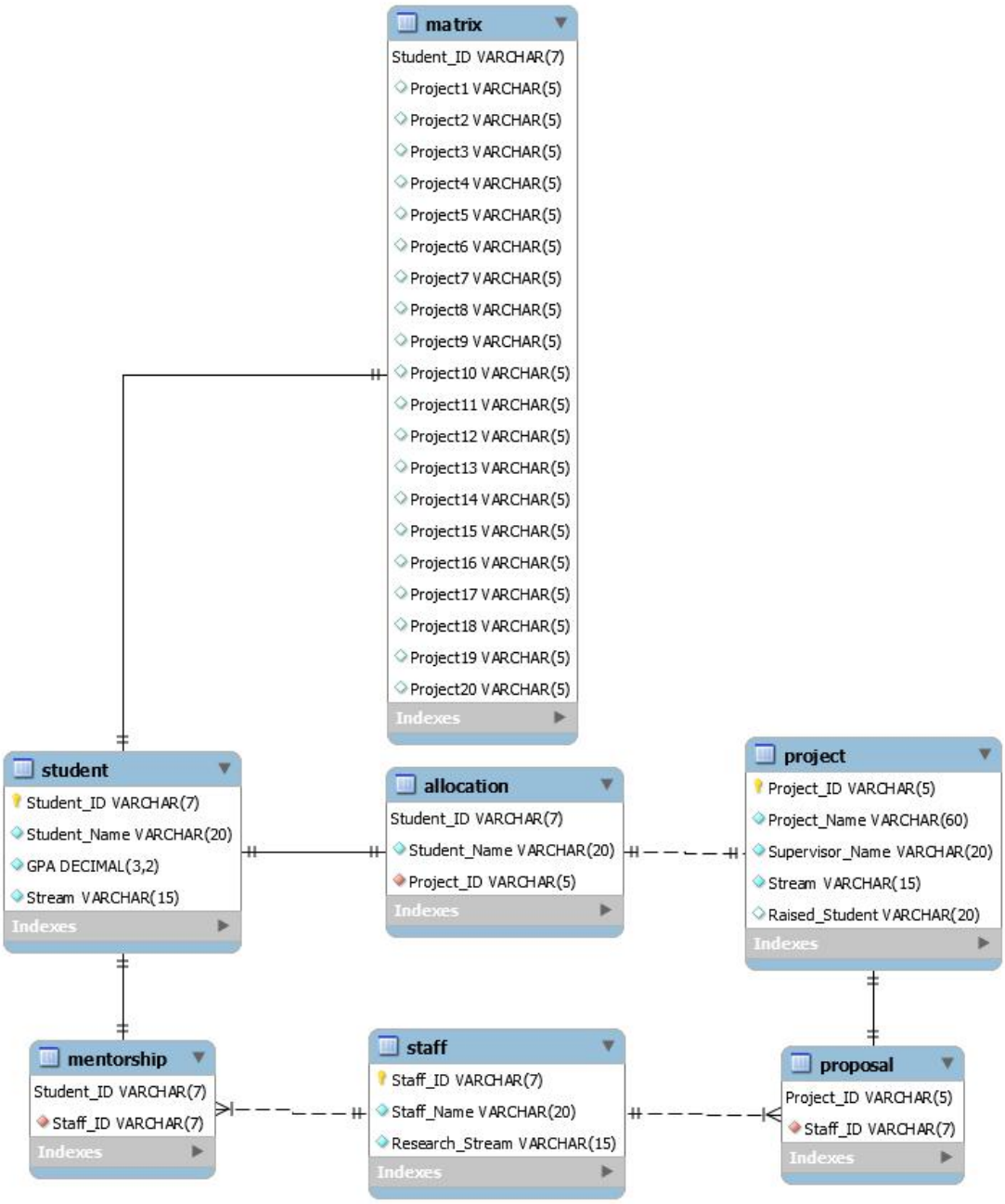


Figure 23. EER Diagram

4. Database Views

A view is a virtual table that combines data from one or more tables or other views. The main aim of a view in SQL is therefore to usefully combine data from several sources without the need to construct yet another database table to store the data.

Because a query determines the view, we can use a view to limit what data is being returned from the source tables and views used to produce the view. A view helps the view author to return data that an end user wants while shielding potentially sensitive data from disclosure in the source tables.[3]

Therefore, It is wise and necessary to represent a certain relation as an SQL view.

In my database system, the views are divided into student, supervisor and comprehensive perspective. For student's perspective, they can browse projects information belonging to different types of streams: CS, CS+DS and CS or CS+DS. For supervisor's perspective, they can browse student's after ranking information belonging to two streams: CS, CS+DS. For comprehensive perspective, students and staff members can see the final allocation information after allocation process.

4.1 Student's Perspective

It is the responsibility of other systems to gather the preferences of students for projects, and it is to be hoped that those other systems ensure that students only see the projects that are fit for their streams. So we create student's part of view.

(1) Project_cs

Project_ID	Project_Name	Supervisor_Name	Stream	Raised_Student
CS000	Cthulhu skiing apps,	Rashawn Berge	CS	NULL
CS001	Cthulhu Public Transportation Inquiry System	Thad Quigley	CS	NULL
CS002	Cthulhu lease Management System of Audio-vis...	Christy Kris	CS	4165377
CS003	Cthulhu office automation system	Vita Cole	CS	NULL
CS004	Cthulhu Local monitoring and remote port scann...	Jany Oberbrunner	CS	NULL
CS005	Cthulhu Electronic address book	Armando Kihn	CS	NULL
CS006	Cthulhu Personal finance system	Bernadette Romag	CS	NULL
CS007	Cthulhu Laboratory Resource Management Syst...	Jonathan Altenwerth	CS	NULL
CS008	Cthulhu Backgammon game design	Tremayne Ledner	CS	NULL
CS009	Cthulhu Bank Account Management System	Rashawn Berge	CS	NULL

Figure 24. Project_cs view

This view will show the project-id, project's title, supervisor's name and raised students' id of projects belonging to the CS stream.

(2) Project_ds

Project_ID	Project_Name	Supervisor_Name	Stream	Raised_Student
DS000	Dagonmarathon apps	Alberto Russel	CS+DS	NULL
DS001	Dagon Eldritch fitness monitors	Sheridan Considine	CS+DS	NULL
DS002	Dagon Medicine Management System	Eldon Rowe	CS+DS	NULL
DS003	Dagon p2p file sharing	Amaya Wuckert	CS+DS	NULL
DS004	Dagon Online learning apps	Jackeline Streich	CS+DS	NULL
DS005	Dagon Travel website	Mozell Frami	CS+DS	NULL
DS006	Dagon Operation Management System	Gretchen Bernhard	CS+DS	NULL
DS007	Dagon Drug sales data management System	Nadia Howe	CS+DS	NULL
DS008	Dagon Online Exam System	Alberto Russel	CS+DS	NULL
DS009	Dagon Online laboratory appointment	Sheridan Considine	CS+DS	NULL
DS010	Dagon Online book purchase	Eldon Rowe	CS+DS	NULL
DS011	Dagon Mobile phone supermarket	Amaya Wuckert	CS+DS	5139874

Figure 25. Project_ds view

This view will show the project-id, project's title, supervisor's name and raised students' id of projects belonging to the CS+DS stream.

(3) Project_csds

Project_ID	Project_Name	Supervisor_Name	Stream	Raised_Student
CSDS0	Human Resource Management System	Armando Kihn	CS or CS+DS	NULL
CSDS1	Student Status Management System	Rashawn Berge	CS or CS+DS	NULL
CSDS2	Student Information Management System	Mozell Frami	CS or CS+DS	NULL
CSDS3	Student Online Course Selection System	Jany Oberbrunner	CS or CS+DS	NULL

Figure 26. Project_csds view

This view will show the project-id, project's title, supervisor's name and raised students' id of project belonging to the CS or CS+DS stream.

4.2 Staff's Perspective

It is necessary for staff members to know student's information, and the student record's in descending order is useful for staff's to check. So we create the staff's perspective of view to browse student's information.

(1) Student_CSRanking

Student_ID	Student_Name	GPA	Stream
3973076	Tyson Lueilwitz	3.96	CS
5186529	Bettye Wunsch S	3.95	CS
4244610	Carlos Ortiz	3.81	CS
4877188	Aleen McKenzie	3.73	CS
5061857	Easter Schiller	3.63	CS
5154148	Jermey Haag	3.26	CS
4751231	Declan Roob	3.03	CS
5025900	Esperanza Yost	2.93	CS
4175826	Janelle Wintheiser	2.83	CS
4857721	Titus Effertz	2.69	CS
4165377	Verlie Kuhlman	2.62	CS

Figure 27. Student_CSRanking view

This view will show the students belonging to CS stream, the staffs can check students' basic information and their GPA. The records are sorted by GPA in descending order so the supervisors can clearly see the ranking order of students.

(2) Student_DSRanking

Student_ID	Student_Name	GPA	Stream
5193123	Mike Spendlove	4.00	CS+DS
4239679	Esta Lubowitz	3.80	CS+DS
4093754	Darien Keebler	3.77	CS+DS
4924912	Ervin Price	3.59	CS+DS
3912076	Cesar Borer	3.12	CS+DS
5187390	Sandra Schamberger	2.87	CS+DS
4974001	Josefa Veum I	2.71	CS+DS
5047013	Will Dare	2.69	CS+DS

Figure 28. Student_DSRanking view

This view will show the students belonging to CS+DS stream, the staffs can check students' basic information and their GPA. The records are sorted by GPA in descending order so the supervisors can clearly see the ranking order of students.

4.3 Comprehensive View

When the process of assigning functions and the project topic selection process is completed, we will generate a view that is more readable for both students and staffs to see.

Student_ID	Student_Name	Project_Name	Supervisor_Name
3906145	Will VonRueden	Dagonmarathon apps	Alberto Russel
3912076	Cesar Borer	Dagon Eldritch fitness monitors	Sheridan Considine
3973076	Tyson Lueilwitz	Cthulhu skiing apps,	Rashawn Berge
3981423	Eda Armstrong	Dagon Medicine Management System	Eldon Rowe
4093754	Darien Keebler	Dagon p2p file sharing	Amaya Wuckert
4146664	Omari Harvey	Dagon Online learning apps	Jackeline Streich
4150466	Hosea Boyer IV	Cthulhu Public Transportation Inquiry System	Thad Quigley
4165377	Verlie Kuhlman	Cthulhu lease Management System of Audio-vis...	Christy Kris
4175826	Janelle Wintheiser	Cthulhu office automation system	Vita Cole

Figure 29. Student-Project-Supervisor view

As above shows, this sps view will show the student-id, student's full name, the title of project they have chosen, and the name of the supervisor of both student and his or her project.

5. Procedural Elements

A database trigger is a special stored procedure which is executed when specific actions take place within a database. When changes are made to a table's data, triggers are defined to run. Triggers can be specified to run on actions such as INSERT, UPDATE, and DELETE instead of or after Data Manipulation Language.

In this database system, it have already used relationships and constrains like foreign key to enforce referential integrity, but foreign key does not apply to all the situations, so I intend to define some triggers to maintain the format and consistency of the data. The tables I implement triggers in are Student table and Staff table.

5.1 Student Table:

(1) student_BEFORE_INSERT trigger:

```
DELIMITER $$
USE `project_18210705`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `student_BEFORE_INSERT`
BEFORE INSERT ON `student` FOR EACH ROW BEGIN
    IF NEW.GPA < 0 THEN
        SET NEW.GPA = 0;
    ELSEIF NEW.GPA > 4.2 THEN
        SET NEW.GPA = 4.2;
    END IF;
    IF New.Stream='DS' THEN
        SET NEW.Stream='CS+DS';
    END IF;
END$$
DELIMITER ;
```

Figure 30. Student_BEFORE_INSERT trigger

The student_BEFORE_INSERT trigger help the database administrator management the Student table's data format. For instance, if we insert a record in which Student.GPA=4.3, the before_insert trigger will run and automatically change it to GPA=4.2 and insert the record in the new line. If we insert a record in which the Student.Stream= 'DS', the trigger will run and automatically change it to 'CS+DS' and insert the record in the new line of the table.

(2)student_BEFORE_UPDATE trigger:

```
DELIMITER $$
USE `project_18210705`$$
CREATE DEFINER = CURRENT_USER TRIGGER `project_18210705`.`student_BEFORE_UPDATE`
BEFORE UPDATE ON `student` FOR EACH ROW
BEGIN
    IF NEW.GPA < 0 THEN
        SET NEW.GPA = 0;
    ELSEIF NEW.GPA > 4.2 THEN
        SET NEW.GPA = 4.2;
    END IF;
    IF New.Stream='DS' THEN
        SET NEW.Stream='CS+DS';
    END IF;
END$$
DELIMITER ;
```

Figure 31. Student_BEFORE_UPDATE trigger

The main function of student_BEFORE_UPDATE trigger is similar to student_BEFORE_UPDATE trigger, but it will run when the UPDATE actions are executed on the Student's table.

(3) student_AFTER_UPDATE trigger:

```
DELIMITER $$
USE `project_18210705`$$
CREATE DEFINER = CURRENT_USER TRIGGER `project_18210705`.`student_AFTER_UPDATE`
AFTER UPDATE ON `student` FOR EACH ROW
BEGIN
    IF OLD.Student_Name <> NEW.Student_Name THEN
        UPDATE Allocation
        SET Allocation.Student_Name=NEW.Student_Name
        WHERE Allocation.Student_Name=OLD.Student_Name;
    END IF;
END$$
DELIMITER ;
```

Figure 32. Student_AFTER_UPDATE trigger

student_AFTER_UPDATE trigger will help to enforce referential integrity of the database system. If we alter the Student_Name in Student table, the corresponding Student_Name in Allocation table should also be changed, and additionally, the student_Name is not the foreign key of Allocation table, so we should define the AFTER_UPDATE trigger to enforce referential integrity of students' information.

5.2 Staff Table:

(1) staff_BEFORE_INSERT trigger:

```
DELIMITER $$
USE `project_18210705`$$
CREATE DEFINER = CURRENT_USER TRIGGER `project_18210705`.`staff_BEFORE_INSERT`
BEFORE INSERT ON `staff` FOR EACH ROW
BEGIN
    IF NEW.Research_Stream='DS' THEN
        SET NEW.Research_Stream='CS+DS';
    END IF;
END$$
DELIMITER ;
```

Figure 33. Staff_BEFORE_INSERT trigger

The staff_BEFORE_INSERT trigger help the database administrator management the Staff table's data format. For instance, if we insert a record in which the Staff.Research_Stream= 'DS', the trigger will run and automatically change it to 'CS+DS' and insert the record in the new line of the table.

(2) staff_BEFORE_UPDATE trigger:

```
DELIMITER $$
USE `project_18210705`$$
CREATE DEFINER = CURRENT_USER TRIGGER `project_18210705`.`staff_BEFORE_UPDATE`
BEFORE UPDATE ON `staff` FOR EACH ROW
BEGIN
    IF NEW.Research_Stream='DS' THEN
        SET NEW.Research_Stream='CS+DS';
    END IF;
END$$
DELIMITER ;
```

Figure 34. Staff_BEFORE_UPDATE trigger

The main function of staff_BEFORE_UPDATE trigger is similar to staff_BEFORE_UPDATE trigger, but it will run when the UPDATE actions are executed on the Staff's table.

(3) staff_AFTER_UPDATE trigger:

```
DELIMITER $$
USE `project_18210705`$$
CREATE DEFINER = CURRENT_USER TRIGGER `project_18210705`.`staff_AFTER_UPDATE`
AFTER UPDATE ON `staff` FOR EACH ROW
BEGIN
    IF OLD.Staff_Name<>NEW.Staff_Name THEN
        UPDATE Project
        SET Project.Supervisor_Name=NEW.Staff_Name
        WHERE Project.Supervisor_Name=OLD.Staff_Name;
    END IF;
END$$
DELIMITER ;
```

Figure 35. Staff_AFTER_UPDATE trigger

staff_AFTER_UPDATE trigger will help to enforce referential integrity of the database system. If we alter the Staff_Name in Staff table, the attribute Supervisor_Name with corresponding name in Project table should also be changed, and additionally, the Supervisor_Name is not the foreign key of Project table, so we need to define the AFTER_UPDATE trigger to enforce referential integrity of staffs' information.

6. Example Queries: Your Database In Action

In this part, this report will provide some sample queries over database and the result of the queries will be illustrated.

(1) Students information mentored by a certain supervisor

```
SELECT a.Student_ID,a.Student_Name,a.GPA,a.Stream
FROM Student a
INNER JOIN Mentorship b ON a.Student_ID=b.Student_ID
INNER JOIN Staff c ON b.Staff_ID=c.Staff_ID
WHERE Staff_Name='Rashawn Berge';
```

Figure 36. Query 1

The query above is used to show the information of students who was mentored the the supervisor 'Rashawn Berge', and the results is as follows:

Student_ID	Student_Name	GPA	Stream
3973076	Tyson Lueilwitz	3.96	CS
4857721	Titus Effertz	2.69	CS
5193123	Mike Spendlove	4.00	CS+DS

Figure 37. Query 1 result

(2) Preference Matrix of a certain stream by GPA ranking

```
SELECT *
FROM Matrix
INNER JOIN Student a ON Matrix.Student_ID=a.Student_ID
WHERE a.Stream='CS'
ORDER BY a.GPA DESC;
```

Figure 38. Query 2

The query above is used to show the ranked preference Matrix of students who was 'CS' stream, and the result of the query will display the preference matrix in GPA's descending order as follows:

Student_ID	Project1	Project2	Project3	Project4	Project5	Project6	Project7
3973076	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5186529	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4244610	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4877188	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5061857	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5154148	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4751231	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 39. Query 2 result

(3) Self-proposed Student information

```
SELECT a.Student_ID,a.Student_Name,a.GPA,a.Stream
FROM Student a
INNER JOIN Project b ON a.Student_ID=b.Raised_Student
WHERE b.Raised_Student is not null;
```

Figure 40. Query 3

The query above is used to show the information of students whose project's title were proposed by themselves, the result is as follows:

Student_ID	Student_Name	GPA	Stream
4165377	Verlie Kuhlman	2.62	CS
5139874	Janis Roberts	1.51	CS+DS

Figure 41. Query 3 result

7. Conclusions

In the course of designing the project allocation database system, I realized the significant role of database playing in the complete design process. From shallow to deep, the gradual and detailed hierarchical understanding is particularly important in the database system design.

A fine-grained schematic view is the fundamental part of database design, it forms the structure of the whole database system, and the proper normalized view is also important in our database design, considering one of the main principles is eliminating the duplicate information because that the space is wasted and the probability of errors and discrepancies is increased. Further more correctness and completeness of information is necessary in database design, that is one of the reason why need to set up constrains including primary key, not null, and foreign key. I also learned that we need to use relationships to enforce referential integrity as our first choice and avoid using trigger to do that part, for the reason that trigger can be hard to troubleshoot and can cause other triggers to fire.

Through designing the database system, I mainly focus on the database needs that the allocation system must entail, and in the future, I intend to study further with this application, and improve deficiencies of my design during this process.

Acknowledgements

I Acknowledge that the work was written by my own, and I would like to express my gratitude to Dr.Tony for his generous instruction and many times of Q&A session regarding to the project's idea helped me a lot.

In addition, I would like to thank our teaching assistances, their answer to my questions were vital in inspiring me to think about my ideas when creating my database system.

References

[1] Elmasri, Ramez; Navathe, Shamkant B. (July 2003). Fundamentals of Database Systems, Fourth Edition. Pearson. p. 315. ISBN 0321204484.

[2] Elmasri, Ramez (2011). Fundamentals of Database Systems. Addison-Wesley. pp. 73–74. ISBN 978-0-13-608620-8.

[3] Jack Pine(2020). Available at: https://www.quora.com/Why-are-Views-used-in-SQL [Accessed 8 Jan. 2020]