Highschool Database

CIS 3400 ETRA [17216]
Professor Qiang Gao

GROUP 5
Aryan Bousri (aryan.bousri@baruchmail.cuny.edu) [Group Leader]
Leonardo Cuapio (leonardo.cuapio@baruchmail.cuny.edu)
Pin Yuan Zhu Lee (pinyuan.zhulee@baruchmail.cuny.edu)
Fatima (fatima.tariq@baruchmail.cuny.edu)
Ranvir Saini (ranvir.saini@baruchmail.cuny.edu)

SUMMARY

For this project database, we as a group decided to use School/ educational institutions as our topic. Every school has a major system in place to manage the various amounts of data related to each specific student, which shows the relevance of the database when it comes to keeping records, maintaining organization, and allowing specific students to be differentiated from others. Through this database, not only can we keep track of students' performance and efforts, we can also keep track of courses, teachers, and non-academic activities. The database can also help with enrollment, and registration by identifying student's eligibility based on prior courses. Additionally, the database allows users to identify awards and grants for qualified students. This is done by utilizing student extracurriculars, grades, and academics.

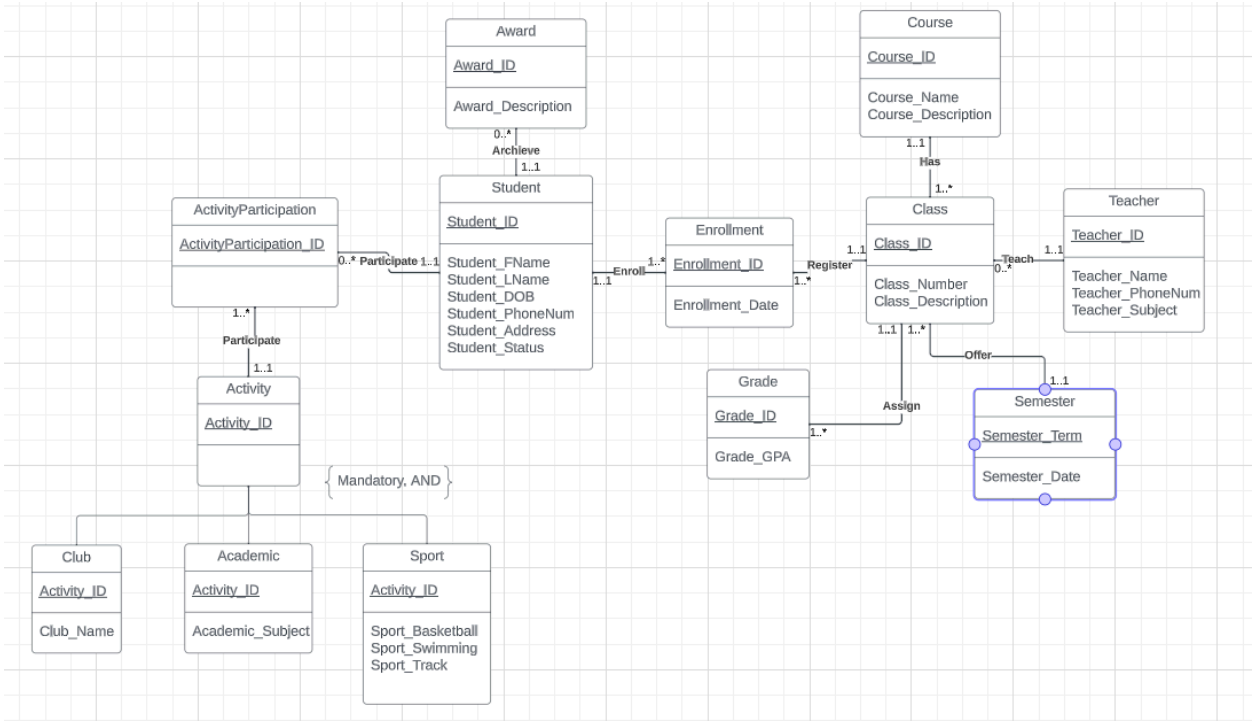
Our team began by building an ERD (entity relationship model) where the relationships between different entities, such as grades and classes were be made. We included a visual of the entity tables as well as the corresponding ERD sentences.

The entities we included are: course, class, grade, student, semester, enrollment, teacher, activity, and ActivtyParticipation.

Our ERD model is then converted into an RDM (relational model) which closes in towards our final goal of implementing a proper database. While the ERD abstractly shows the relationships between entities and attributes, the RDM defines them in a way a database could interpret them. We then entered a normalization phase, where our main objective was to eliminate data redundancy and to produce a clearer data model. The next step is SQL implementation, which allows our team to bring our initially abstract database idea into a completed form, where now we can manipulate our database in Microsoft Access with various scenarios to ensure that it serves its purpose.

The scenarios show exactly how this database can be used. Our examples demonstrate how students can be identified for different scholarships/grants, programs, registration requirements, and class information. From figuring out which students in the robotics team won a money prize to identifying seniors who qualify for the prime scholarship, our database can be used to cover many areas.

ERD Model



ERD Sentences

Entities: course, class
One course has one or many classes. 1..*
One class belongs to one and only one course. 1..1

Entities: class, semester
One semester offers one or many classes. 1..*
One class is under one and only one semester. 1..1

Entities: student, enrollment, class
The student enrolls in one or many times. 1..*
The enrolled student is one and only one student. 1..1
Class has one or many enrolled students. 1..*
An enrolled student belongs to one and only one class. 1..1

Entities: student, award
A class assigns one or many grades. 1..*
A grade belongs to one and only one class. 1..1

Entities: grade, class
A student has one or many grades. 1..*
A grade belongs to one and only one student. 1..1

Entities: teacher, class
A teacher teaches zero or many classes. 0..*
A class is taught by one and only one teacher. 1..1

Entities: ActivityParticipation, student
A student has participation if the student participates.
A student participation belongs to one and only one student.

Entities: ActivityParticipation, Activity
One activity may have one or many participants. 1..*
One activitypartcipation belongs to one and only one student. 1..1

Entities. Activity
Activity can be classified as 3 main categories: club, academic competition, and sport team.

Converting ERD to RDM

Course (Course_ID, Course_Name, Course_Description)

Class (Class_ID, Class_Number, Class_Description, Course_ID(fk), Teacher_ID(fk), Semester_Term(fk))

Teacher (Teacher_ID, Teacher_Name, Teacher_PhoneNum, Teacher_Subject)

Semester (Semester_Term, Semester_Date)

Grade (Grade_ID, Grade_GPA, Class_ID(fk))

Enrollment (Enrollment_ID, Enrollment_Date, Class_ID(fk), Student_ID(fk))

Student (Student_ID, Student_FName, Student_LName, Student_DOB, Student_PhoneNum, Student_Address, Student_Status)

Award (Award_ID, Award_Description, Student_ID(fk))

ActivityParticipation (ActivityParticipation_ID, Student_ID(fk), Activity_ID(fk))

Activity (Activity_ID, Club_Name, Academic_Subject, Sport_Basketball, Sport_Swimming, Sport_Track, Type_Activity)

Normalization

Course (Course_ID, Course_Name, Course_Description)

Step 1: Key? Yes, (Course_ID)

Step 2: Partial key dependency? No

Step 3: Transitive dependency? No

Final relationship: Course (Course_ID, Course_Name, Course_Description)

Class (Class_ID, Class_Number, Class_Description, Course_ID(fk), Teacher_ID(fk), Semester_Term(fk))

Step 1: Key? Yes, (Class_ID)

Step 2: Partial key dependency? No

Step 3: Transitive dependency? No

Final relationship: Class (Class_ID, Class_Number, Class_Description, Course_ID(fk), Teacher_ID(fk), Semester_Term(fk))

Teacher (Teacher_ID, Teacher_Name, Teacher_PhoneNum, Teacher_Subject)

Step 1: Key? Yes, (Teacher_ID)

Step 2: Partial key dependency? No

Step 3: Transitive dependency? No

Final relationship: Teacher (Teacher_ID, Teacher_Name, Teacher_PhoneNum, Teacher_Subject)

Semester (Semester_Term, Semester_Date)

Step 1: Key? Yes, (Semester_Term)

Step 2: Partial key dependency? No

Step 3: Transitive dependency? No

Final relationship: Semester (Semester_Term, Semester_Date)

Grade (Grade_ID, Grade_GPA, Class_ID(fk))

Step 1: Key? Yes, (Grade_ID)

Step 2: Partial key dependency? No

Step 3: Transitive dependency? No

Final relationship: Grade (Grade_ID, Grade_GPA, Class_ID(fk))

Enrollment (Enrollment_ID, Enrollment_Date, Class_ID(fk), Student_ID(fk))

Step 1: Key? Yes, (Enrollment_ID)

Step 2: Partial key dependency? No

Step 3: Transitive dependency? No

Final relationship: Enrollment (Enrollment_ID, Enrollment_Date, Class_ID(fk), Student_ID(fk))

Student (Student_ID, Student_FName, Student_LName, Student_DOB, Student_PhoneNum, Student_Address, Student_Status)

Step 1: Key? Yes, (Student_ID)

Step 2: Partial key dependency? No

Step 3: Transitive dependency? No

Final relationship: Student (Student_ID, Student_FName, Student_LName, Student_DOB, Student_PhoneNum, Student_Address, Student_Status)

Award (Award_ID, Award_Description, Student_ID(fk))

Step 1: Key? Yes, (Award_ID)

Step 2: Partial key dependency? No

Step 3: Transitive dependency? No

Final relationship: Award (Award_ID, Award_Description, Student_ID(fk))

ActivityParticipation (ActivityParticipation_ID, Student_ID(fk), Activity_ID(fk))

Step 1: Key? Yes, (ActivityParticipation_ID)

Step 2: Partial key dependency? No

Step 3: Transitive dependency? No

Final relationship: ActivityParticipation (ActivityParticipation_ID, Student_ID(fk), Activity_ID(fk))

Activity (Activity_ID, Club_Name, Academic_Subject, Sport_Basketball, Sport_Swimming, Sport_Track, Type_Activity)

Step 1: Key? Yes, (Activity_ID)

Step 2: Partial key dependency? No

Step 3: Transitive dependency? No

Final relationship: Activity (Activity_ID, Club_Name, Academic_Subject, Sport_Basketball, Sport_Swimming, Sport_Track, Type_Activity)

Final Relationships

Course (Course_ID, Course_Name, Course_Description)

Class (Class_ID, Class_Number, Class_Description, Course_ID(fk), Teacher_ID(fk), Semester_Term(fk))

Teacher (Teacher_ID, Teacher_Name, Teacher_PhoneNum, Teacher_Subject)

Semester (Semester_Term, Semester_Date)

Grade (Grade_ID, Grade_GPA, Class_ID(fk))

Enrollment (Enrollment_ID, Enrollment_Date, Class_ID(fk), Student_ID(fk))

Student (Student_ID, Student_FName, Student_LName, Student_DOB, Student_PhoneNum, Student_Address, Student_Status)

Award (Award_ID, Award_Description, Student_ID(fk))

ActivityParticipation (ActivityParticipation_ID, Student_ID(fk), Activity_ID(fk))

Activity (Activity_ID, Club_Name, Academic_Subject, Sport_Basketball, Sport_Swimming, Sport_Track, Type_Activity)

Database Implementation and Inserting Data

```
CREATE TABLE Course (
Course_ID NUMBER NOT NULL,
Course_Name VARCHAR(250),
Course_Description VARCHAR(250),
CONSTRAINT pk_Course PRIMARY KEY (Course_ID)
)

INSERT INTO Course VALUES (1, “Calculus 1”, “Limit & derivative”)

CREATE TABLE Teacher (
Teacher_ID NUMBER NOT NULL,
Teacher_Name VARCHAR(250),
Teacher_PhoneNum NUMBER,
Teacher_Subject VARCHAR(250),
CONSTRAINT pk_Teacher PRIMARY KEY (Teacher_ID)
)

INSERT INTO Teacher VALUES (1, "Jake",       5687459631, "   Math")

CREATE TABLE Semester (
Semester_Term VARCHAR(250),
Semester_Date DATE,
CONSTRAINT pk_Semester PRIMARY KEY (Semester_Term)
)

INSERT INTO Semester VALUES ("Fall 2023", "7/20/2023")

CREATE TABLE Class (
Class_ID NUMBER NOT NULL,
Class_Number VARCHAR(250),
Class_Description VARCHAR(250),
Course_ID NUMBER,
Teacher_ID NUMBER,
Semester_Term VARCHAR(250),
CONSTRAINT pk_Class PRIMARY KEY (Class_ID),
CONSTRAINT fk1_Class FOREIGN KEY (Course_ID) REFERENCES  Course(Course_ID),
CONSTRAINT fk2_Class FOREIGN KEY (Teacher_ID) REFERENCES Teacher(Teacher_ID),
CONSTRAINT fk3_Class FOREIGN KEY (Semester_Term) REFERENCES Semester(Semester_Term)
)

INSERT INTO Class VALUES (1, "MTH 1001", "Math", 1, 1, "Fall 2023")
```

```
CREATE TABLE Grade (
Grade_ID NUMBER NOT NULL,
Grade_GPA NUMBER,
Class_ID NUMBER,
CONSTRAINT pk_Grade PRIMARY KEY (Grade_ID),
CONSTRAINT fk_Grade FOREIGN KEY (Class_ID) REFERENCES  Class(Class_ID)
)

INSERT INTO Grade VALUES (101, 62, 1)

CREATE TABLE Student (
Student_ID NUMBER NOT NULL,
Student_FName VARCHAR(250),
Student_LName VARCHAR(250),
Student_DOB DATE,
Student_PhoneNum NUMBER,
Student_Address VARCHAR(250),
Student_Status VARCHAR(250),
CONSTRAINT pk_Student PRIMARY KEY (Student_ID)
)

INSERT INTO Student VALUES (101, "Jayce", "Thomas", "9/11/2006", 5461268463, "155 W. Prospect
Ave.", "Freshman")

CREATE TABLE Enrollment (
Enrollment_ID NUMBER NOT NULL,
Enrollment_Date DATE,
Class_ID NUMBER,
Student_ID NUMBER,
CONSTRAINT pk_Enrollment PRIMARY KEY (Enrollment_ID),
CONSTRAINT fk1_Enrollment FOREIGN KEY (Class_ID) REFERENCES  Class(Class_ID),
CONSTRAINT fk2_Enrollment FOREIGN KEY (Student_ID) REFERENCES  Student(Student_ID)
)

INSERT INTO Enrollment VALUES(1, "6/25/2023",       4, 101)

CREATE TABLE Award (
Award_ID NUMBER NOT NULL,
Award_Description VARCHAR(250),
Student_ID NUMBER,
CONSTRAINT pk_Award PRIMARY KEY (Award_ID),
CONSTRAINT fk_Award FOREIGN KEY (Student_ID) REFERENCES  Student (Student_ID)
```

```
)

INSERT INTO Award VALUES (1001, "Math Team 3rd Place", 103)

CREATE TABLE Activity (
Activity_ID NUMBER NOT NULL,
Club_Name VARCHAR(250),
Academic_Subject VARCHAR(250),
Sport_Basketball VARCHAR(250),
Sport_Swimming VARCHAR(250),
Sport_Track VARCHAR(250),
Type_Activity VARCHAR(250),
CONSTRAINT pk_Activity PRIMARY KEY (Activity_ID)
)

INSERT INTO Activity(Activity_ID, Sport_Basketball) VALUES (1, "Yes")

CREATE TABLE ActivityParticipation (
ActivityParticipation_ID NUMBER NOT NULL,
Student_ID NUMBER,
Activity_ID NUMBER,
CONSTRAINT pk_ActivityParticipation PRIMARY KEY (ActivityParticipation_ID),
CONSTRAINT fk1_ActivityParticipation  FOREIGN KEY (Student_ID) REFERENCES  Student
(Student_ID),
CONSTRAINT fk2_ActivityParticipation  FOREIGN KEY (Activity_ID) REFERENCES  Activity
(Activity_ID)
)

INSERT INTO ActivityParticipation VALUES (1, 101, 5)
```

Scenario

Seniors are qualified for a $5,000 Prime Scholarship. Write a query to display the students (first and last name) who qualify for this scholarship.

```
SELECT Student_ID, Student_FNAME, Student_Lname
FROM Student
WHERE Student_Status = "Senior"
```

Student_ID	Student_FNAME	Student_Lname
106	Isobelle	Adams
108	Lana	Young
109	Anna	Martin
111	Lala	Hall

The Swimming Organization is giving out $1,000 for students (first and last name)who participate in swimming. Write a query to display the students who can get the $1,000.

```
SELECT A.Student_ID, S.Student_FNAME, S.Student_Lname
FROM Student S INNER JOIN ActivityParticipation A
ON A.Student_ID = S.Student_ID
WHERE A.Activity_ID = 2
```

Student_ID	Student_FNAME	Student_Lname
105	Antonia	White
110	Joy	Lee

Students who have participated in a math competition and gotten an award are invited to participate in the MathHouse Internship Program. Write a query to display the information of students who can apply for this internship and the award they achieve.

```
SELECT S.Student_ID, S.Student_FName, S.Student_LName, S.Student_DOB, S.Student_PhoneNum,
S.Student_Address, S.Student_Status, A.Award_Description
FROM Student S INNER JOIN Award A
ON A.Student_ID = S.Student_ID
WHERE A.Award_Description LIKE "*Math*"
```

Student_ID	Student_FName	Student_LName	Student_DOB	Student_PhoneNum	Student_Address	Student_Status	Award_Description
103	Lose	Scott	2/29/2008	1648964523	7133 Fieldstone Road	Freshman	Math Team 3rd Place
107	Charlotte	Jackson	1/29/2009	5897468964	90 Bayport Ave.	Freshman	Global Math Solving Award

The semester is ending. Only students who are taking calculus 1 can register for calculus 2. Write a query to display the first and last name of the students who can register for calculus 2.

```
SELECT S.Student_ID, S.Student_FName, S.Student_LName
FROM Student S INNER JOIN Enrollment E
ON E.Student_ID = S.Student_ID
WHERE E.Class_ID IN (1,2)
```

Student_ID	Student_FName	Student_LName
103	Lose	Scott
104	Rachel	Taylor
105	Antonia	White
106	Isobelle	Adams
108	Lana	Young

NASA is hosting a robot competition. Participants can win up to $10,000 and NASA will cover the tuition of all students of the winning team. Write a query to display the member(s) of the Robotic Team.

```
SELECT A.Academic_Subject, S.Student_FName, S.Student_LName
FROM Activity A, ActivityParticipation P, Student S
WHERE A.Activity_ID=P.Activity_ID AND P.Student_ID=S.Student_ID AND  A.Academic_Subject =
"Robotic Team"
```

Academic_Subject	Student_FName	Student_LName
Robotic Team	Katy	Wilson

During the enrollment period, students want to know the teacher of the class they want to enroll in. Write a query to display the class number and teacher's name.

```
SELECT  DISTINCT Class_Number, Teacher_Name
FROM class c, teacher t
WHERE c.Teacher_ID = t.Teacher_ID
```

Class_Numbe	Teacher_Name
ENG 1001	Blue
MTH 1001	Jake
MTH 1002	Luck
PHY 1001	Skyle

Conclusion

The most challenging part of the project we as a group experienced was the creation of the ERD. It required us to consider which entities to include while thinking about how they would link. Our group had many discussions and brainstorming sessions on this matter. This is what made the process difficult since everyone had different and clashing ideas. Once we agreed on which entities we would use for our school database, we needed to figure out the attributes and the relationships between the entities. After multiple revisions and discussions, we created our ERD model. The project's easiest part was converting the ERD to RDM. This is because as long as the rules of the different relationships are followed, it is an easy and smooth process. A set of relations was the result of the process which was derived by converting each entity using the relationships.

If we were to do this project again, it is essential to think ahead of the possible scenarios and create links and relationships for them. For example, the student entity and grade entity were not linked, so there was no way to make a scholarship/grant for students with high grades/GPA.

Our school database system works as intended, and as proposed. It identifies specific students that are eligible for different scholarships/grants. Additionally it allows users to track students' performance and efforts, while also keeping track of courses, teachers, and non-academic activities.