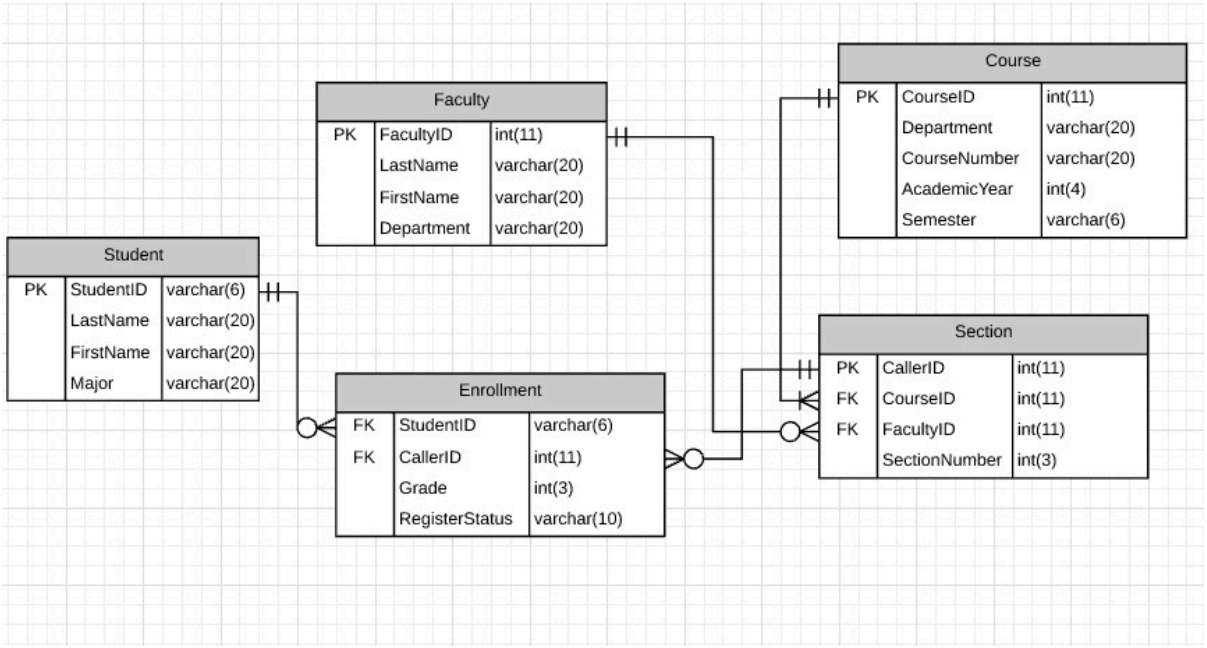

Student Entity

- StudentID: It is easy to understand that we have to set this as a primary key of this entity. Students can have the same name and major but every student will have a unique student id.
- LastName: Student's last name can't be null.
- FirstName: Student's first name can't be null.
- Major: Student's major will set to be null if not specified because some undergraduate students may not have chosen a major yet.

Generally, the student entity is an independent entity that provides information about students.

Faculty Entity

- FacultyID: This is similar to the StudentID field of Student Entity. It is the primary key for faculties. It is called FacultyID instead of StudentID because faculties are not student anymore, though FacultyID and StudentID may have the same format as they are both given by the university.
- LastName: Faculty's last name can't be null.
- FirstName: Faculty's first name can't be null.
- Department: The department is also default null as student entity's major because some temporary faculties may not have direct relationship to any department.

The Faculty entity is also an independent entity that provides information about faculties.

Course Entity

- CourseID: Primary Key for identifying the courses.
- Department: This specifies which department does this course belong to. This field can't be null. Ex: COMS, ELEN…
- CourseNumber: This specifies the specific course number under certain department. This field can't be null. Ex: 4111, 4701….
- Year: This specifies which year is/was this course offered. This can't be null and should be a 4-digit integers. Ex: 2017,2016…
- Semester: This specifies which semester is this course offered. This can't be null. Ex: summer,fall,winter….

The course entity is an independent entity. It provides information about a course. I was previously thinking making Department + CourseNumber (EX COMS + 4111) as primary key, but I then realized that I also need to consider same courses in previous semesters. Therefore, I set CourseID as primary key. Another advantage is that it is easier to link section entity to the course if we have only one primary key for course entity.

Section Entity

- CallerID: Primary Key for identifying the sections. This is inspired by our university registration system: each section has a unique callerID.
- CourseID: Foreign Key that relates to the course entity. Each section must belong to a course. A course entity will have at least one related section. This can't be null.
- FacultyID: Foreign Key that relates to the faculty entity. Each section must have a faculty. A faculty entity may have 0,1 or many sections. This can be null at the beginning because school could open a section first then assign a professor teaching that section later.
- SectionNumber: This is the each section's number. This can't be null. Ex: 001, 002, H01…

The Section entity is considered to be an associative entity of course entity because a section must be associated with a course.

Enrollment Entity

- StudentID: Foreign Key that relates to the student entity. Each enrollment entity must have a studentID. Each student entity can have 0,1 or more enrollments.
- CallerID: Foreign Key that relates to the section entity. Each enrollment entity must have a callerID. Each section entity can have 0,1 or more enrollments.
- Grade: This is the grade of the student who is in this section. It can be null and then set to a specific number after final exam. Ex: null, 80, 95, 99….
- RegisterStatus: The status can be either "enrollment" or "waitlist"

The enrollment entity is a weak entity connecting student entity and section entity. It provides information about student's registration status and grade for his/her section.

To sum up, the student, faculty and course entities are considered independent. The section entity is associated with course entity but also has a foreign key to faculty entity. The enrollment entity connects student entity and section entity as a weak entity. Another thing I would like to mention is that I found a hole in my design after finishing all my work. For enrollment entity, I do not set any primary key, which would allow student to register more than one section that belong to the same course. This could be avoid by adding a CourseID foreign key that points to course entity's CourseID and make CourseID + StudentID together to be the primary key for enrollment entity.

Then, it's time to implement these tables on sql workbench. The specific implementation is in "ddl.sql" file.

After creating those five tables, I inserted some test data into the table as shown below.

FacultyID	LastName	FirstName	Department
1	Jett	June	cs
2	Voegele	Virgina	NULL
3	Tow	Kaci	cs
4	Yelton	Lanny	cs
5	Hug	Kylee	ee
NULL	NULL	NULL	NULL

Figure1. Faculty Table

I made up five faculties. The Department column can be null but the other three columns cannot be null.

StudentID	LastName	FirstName	Major
bs8726	Santa	Binford	cs
cb1311	Bellis	Clyde	math
cm2829	Mcclaren	Chris	NULL
gt9377	Trenton	Gunnells	math
rm3719	Merry	Rutz	cs
NULL	NULL	NULL	NULL

Figure2. Student Table

Similarly, I made up five students. The major column can be null but the other three columns cannot be null. For test purpose, I only created five students. There of course should be more than just five students.

CourseID	Department	CourseNumber	AcademicYear	Semester
1	COMS	4111	2017	FALL
2	COMS	4111	2016	FALL
3	CSEE	4119	2017	FALL
4	COMS	4701	2017	FALL
5	COMS	4711	2017	FALL
6	COMS	6998	2017	FALL
7	COMS	4231	2017	FALL
8	CSEE	6998	2017	FALL
NULL	NULL	NULL	NULL	NULL

Figure3. Course Table

I created eight courses for test purpose. There are two COMS4111 courses, one is for 2017 Fall and one is for 2016 Fall. This would be demonstrated through user stories section.

CallerID	CourseID	FacultyID	SectionNumb...
10000	1	1	1
10001	1	2	2
10002	2	3	1
10003	3	4	1
10004	3	1	2
10005	3	5	3
10006	4	4	1
10007	5	5	1
10008	6	1	1
10009	7	2	1
10010	8	3	1
NULL	NULL	NULL	NULL

Figure4. Section Table

I create several sections and make each course has at least one related section. I also randomly assigned faculties to the sections for test purpose.

Stude...	CallerID	Grade	RegisterStatus
bs8726	10002	95	enrolled
bs8726	10003	NULL	waitlist
bs8726	10006	NULL	enrolled
bs8726	10007	NULL	waitlist
cb1311	10002	98	enrolled
cb1311	10010	NULL	enrolled
cb1311	10009	NULL	waitlist
cm2829	10000	NULL	waitlist
cm2829	10007	NULL	enrolled
cm2829	10006	NULL	waitlist
gt9377	10000	NULL	enrolled
gt9377	10003	NULL	enrolled
gt9377	10006	NULL	enrolled
gt9377	10009	NULL	enrolled
rm3719	10001	NULL	enrolled
rm3719	10003	NULL	enrolled
rm3719	10008	NULL	enrolled
rm3719	10009	NULL	enrolled
NULL	NULL	NULL	NULL

Figure5. Enrollment Table

The enrollment table is the largest table. For test purpose, I assign 3-4 sections (including sections in previous year) to each student and I also randomly assign the RegisterStatus.
Now, let's test with some interesting user stories. The specific sql statements can be found in "story.sql". I would only describe what the story is and paste the result.

Story #1: Delete a faculty entity. We expect this to fail because every faculty relates to at least one section. Therefore, it will violate the foreign key constraint. The output result is shown below.

16:33:57 Delete From Faculty where Faculty.LastName="Jett" Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`4111_hw2`.`section`, CONSTRAINT `FK_Section_FacultyID` FOREIGN KEY (`FacultyID`) REFERENCES `Faculty` (`FacultyID`)) 0.079 sec

Story #2: Registration office is thinking about who may be a good professor for teaching COMS4111 next semester. The registration office would like to see which faculties have taught COMS4111. The answer should also include what time does the faculty teach that course.

LastName	FirstName	CallerID	Department	CourseNumber	Semester	AcademicYear
Jett	June	10000	COMS	4111	FALL	2017
Voegele	Virgina	10001	COMS	4111	FALL	2017
Tow	Kaci	10002	COMS	4111	FALL	2016

Figure6. Output for second story

Story #3: The current COMS4111 professor is interested in selecting a TA. Therefore, the professor would like to know the students who took COMS4111 before (Year < 2017) and obtained a good grade (Grade > 95). The professor would also like to know when that student took the course and who was his/her professor at that time. (This involved information from all five tables)

s_lastname	s_firstname	Department	CourseNumber	AcademicYear	Semester	CallerID	Grade	f_lastname	f_firstnan
Bellis	Clyde	COMS	4111	2016	FALL	10002	98	Tow	Kaci

Figure7. Output for third story

As the result shows, Student Clyde Bellis took COMS4111 in 2016 FALL with faculty Kaci Tow and obtained 98 points.

Story #4: A student named Rutz Merry would like to know all her professors' names for her enrolled class.

LastName	FirstName
Voegele	Virgina
Yelton	Lanny
Jett	June

Figure8. Output for fourth story

Again, please refer to "ddl.sql" for sql statements creating the five tables and "story.sql" for sql statements supporting the user stories.