DATABASE SPECIFICATIONS

For New Registration System

Registration System Database For Gotham University
Rahul Sagi, pzu0658@pzu.edu

School of Graduate Professional Studies
Information Science Department
INSC 521 - Introduction to Database Concepts

January, 2021

DOCUMENT CONTROL

Work carried out by:

Name	Email Address	Other
Rahul Sagi	**pzu0658@pzu.edu**	Phone Number: (682)-597-4568

Revision Sheet

Release No.	Date	Revision Description
1	**01/26/2021**	**Defined initial design providing detailed project overview, purpose by defining the core business requirements based on Gotham Universities' Software Requirement Specifications (SRS) document to produce the Core Requirements (CR) and identify the core entities**
2	**02/10/2021**	**Created a conceptual ERD model that helped define entities and the relationships between those entities. Furthermore, established total and partial participation constraints and displayed key assumptions such as relationships and the constraints and limitations of an ERD model**
3	**02/25/2021**	**Created a logical design based on the conceptual model in Milestone 2 using Crow-Foot Notation. Updated and improved upon the model in Milestone 2 as well as implement better Chen notation per professor feedback.**
4	**03/12/2021**	**Updated Milestone 2 and Mileston3 per professor feedback. Added and got rid of additional attributes to Student, Faculty, Professors, and Department during normalization. Identified primary and foreign keys. All tables in the database normalized to BCNF. Additional constraints added.**
5	**03/25/2021**	**Created the physical ERD based on the logical design. Built the tables through SQL in Oracle and input 20 records for each table. Updated naming conventions and attributes based upon professor feedback and updated Table of Contents.**
6	**04/09/2021**	**Provided various queries in relation to the SRS through a variety of commands. Updated a normalization error per feedback from the professor. Updated Table of Contents pages.**

DATABASE SPECIFICATIONS

TABLE OF CONTENTS

MILESTONE 1: DATA REQUIREMENTS

Purpose

The purpose of this specification is to provide specific information about the various entities that will make up the revamped database for Gotham University as well as detail the core requirements of the database. Furthermore, this specification should be used as a guideline and reference to understand the core requirements and the tenets of the individual elements that composite the new database. Having an understanding of these aspects will help the interested parties understand and eventually improve upon the contemporary version of the database for Gotham University.

System Overview

Gotham University wants to revamp their system and allow registration to be both be completed online and in person. This is so that registration will be more streamlined for users and more thorough for faculty and advisors. Furthermore, the new system will also innovate existing features and increase its capabilities. The new database will have multiple entities, such as departments, students, faculty, courses, cost, etc. Through these entities, the user can expect streamlined reports through information derived from the database based on their queries.

Entities:

Departments – Stores the location, name of the Department, DepartmentID, Faculty ID, and courses offered through CourseID

Courses – Stores the CourseID, name of the course, the cost of the course, Pre-Req, the major, Course Syllabus ID, course availability, and if they are on the waitlist or not

Cost – Stores the payment deadlines of various categories such as cost of rooming, and coursework. Also stores AdvisorID as each category falls under a specific advisor.

Faculty – Stores FacultyID, the students that fall under that advisor, the faculty advisor's first and last name, DepartmentID, the professors part of the faculty, and the department they are located in

Pre-Req – Stores whether or not that CourseID has a pre-req and what those pre-reqs are

Professors – Stores information about the ProfessorID, the faculty they're a part of, the courses they teach, their first and last name, and gender

Student – Stores StudentID, DepartmentID, first and last name, and phone number

Waitlist – Stores the WaitlistID, the StudentID, CourseID, and the student's first and last name

System Name or Title

Registration System Database For Gotham University

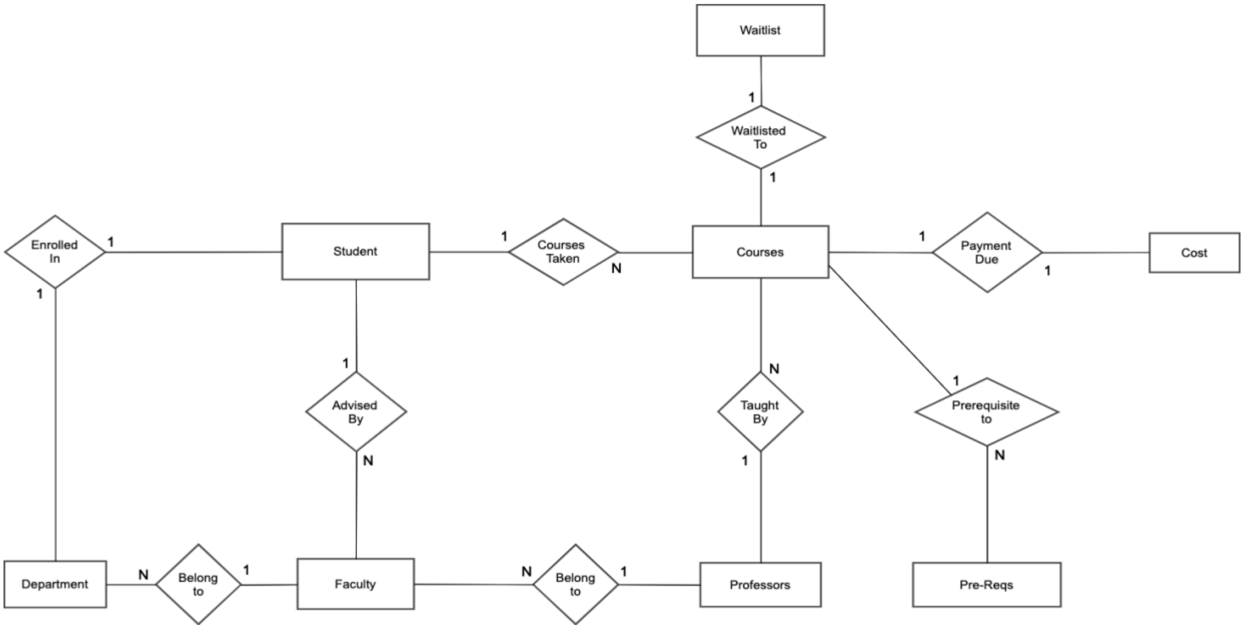
Core requirements

No.	Requirement	Referenced page in SRS	Referenced Section in SRS	Referenced Paragraph in Section
1	The system stores information about the departments the courses each department offers, the students that are a part of the department as well as the name of the department	4 14	1.1 7.3	1 1
2	The system stores information about the cost of each course as well as the multiple payment deadlines	13	9	1F
3	The system stores information about students, registrars, and faculty	4 7 13	1.1 2.3 9	1 Registrar 1 1F
4	The system stores information about class occupation and whether or not the class is available	12 16	7.3 9	3 6F
5	The System stores information about the pre-reqs for each course and whether or not the student satisfies the pre-reqs	10 19	5 9	5.1 12F
6	The system stores information about which classes are on the waitlist	4 11 13	1.1 5 8	4 5.1 8.1
7	The system stores information on each students GPA. This information is essential for students taking a 6th course	11	6	6.1
8	The system stores information about faculty and which students fall under that faculty	7 10 19	2 5 9	2.3 5.1 13F
9	The system stores information to allow faculty and administration to override certain requirements if need be	8 15 15	3.3 9 9	2 4F 5F
10	The system stores information about the course syllabus and other essential information for that particular course.	12 21	7.3 9	1 16F

11	The system stores information on which students receive greater priority. A person of that department will receive greater priority than a person who isn’t in that major.	17 18	9 9	8F 11F

MILESTONE 2: CONCEPTUAL DESIGN

Diagram



Assumptions and Constraints

- Unique attributes can’t be represented in the ERD
- Assumption that the university database is connected to the Internet
- Every table has a primary key
- Attributes like the age of students are implied given their enrollment date
- Assumption that the information in the database is accurate
- Assumption that the system runs smoothly and isn’t easily corrupted
- There’s a set limit for the number of students who are enrolled in a last or in the waitlist
- Each student can enroll in multiple courses, but there’s a limit to the number of courses a student can enroll in
- Unique attributes can’t be represented in the ERD
- Every course has a department, but a department doesn’t have every course
- Every course cost some amount of money, but not all courses cost the same
- Student entity will have fields such as enrollment year to help qualify students for certain courses
- Faculty group will have a n number of students based on an order sort like last name descending
- Every student has a faculty advisor, but a faculty group doesn’t have every student
- Every pre-req has a course, but not all courses have pre-reqs
- Every syllabus is a part of a course, and all courses have a syllabus
- There are enough users that are interested in the online registration system for it to be an effective database
- Mutated data can’t be displayed in an ERD effectively
- Some entities have a total participation constraint
- Not all constraints can be expressed in an ERD model
- Relatively abstract
- An ERD isn’t effective in displaying the minute details of a database

MILESTONE 3: LOGICAL DESIGN

Entity name: Departments

Attributes: dept_id, student_id, faculty_id, name

Keys: dept_id, student_id, faculty_id

Functional dependencies:

faculty_id → Faculty,

student_id → Student

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	dept_id		name, student_id, faculty_id

Attribute closures (if any): (dept_id)+ ={student_id, faculty_id, name};
(faculty_id)+ ={Faculty};
(student_id)+ ={Student};

Unique keys: dept_id

Entity name: Student

Attributes: Added student_id, faculty_id, dept_id, course_id, overallGPA, stud_fname, stud_lname, gender, enrollmentYear.

Keys: student_id, faculty_id, dept_id, course_id,

Functional dependencies:

faculty_id → Faculty,

dept_id → Department, and

course_id → Course.

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	student_id		overallGPA, stud_fname, stud_lname, gender, enrollmentYear, faculty_id, dept_id, course_id,

Attribute closures (if any): (student_id)+ ={faculty_id, dept_id, course_id, stud_name, gender, overallGPA, enrollmentYear};

(dept_id)+ ={Department};
(course_id)+ ={Courses};
(faculty_id)+ ={Faculty};

Unique keys: student_id

Entity name: Courses

Attributes: Added course_id, student_id, waitlist_id, cost_id, prereq_id, prof_id, capacity, credits, syllabus, coursename.

Keys: course_id, student_id, waitlist_id, cost_id, prereq_id, prof_id

Functional dependencies:

student_id→ Student,

waitlist_id → Waitlist,

cost_id → Cost,

prereq_id → Pre Req, and

prof_id → Professor

Attributes not in FD	**Attributes on the left**	**Attributes on both sides**	**Attributes on the right side**
	course_id		capacity, credits, syllabus, coursename, student_id, waitlist_id, cost_id, prereq_id, prof_id

Attribute closures (if any): (course_id)+ ={capacity, credits, syllabus, coursename, student_id, waitlist_id, cost_id, prereq_id, prof_id};
(prof_id)+ ={Professors};
(cost_id)+ ={Cost};
(prereq_id)+ ={Pre Reqs};
(student_id)+ ={Student};
(waitlist_id)+ ={Waitlist};

Unique keys: course_id

Entity name: Faculty

Attributes: Added faculty_id, student_id, dept_id, prof_id, email, phone number

Keys: student_id, faculty_id, dept_id, prof_id,

Functional dependencies:

student_id → Student,

dept_id → Department, and

prof_id → Professors.

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	faculty_id		email, faculty_id, dept_id, prof_id,

Attribute closures (if any): (faculty_id)+ ={student_id, dept_id, prof_id, email};
(dept_id)+ ={Department};
(student_id)+ ={Student};
(prof_id)+ ={Professors};

Unique keys: faculty_id

Entity Name: Professors

Attributes: Added prof_id, faculty_id, course_id, gender, prof_name,

Keys: prof_id, faculty_id, course_id

Functional dependencies:

faculty_id → Faculty, and

course_id → Courses

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	prof_id		gender, prof_fname, prof_lname, faculty_id, course_id

Attribute closures (if any): (prof_id)+ ={faculty_id, course_id, gender, prof_name};
(faculty_id)+ ={Faculty};
(course_id)+ ={Courses};

Unique keys: prof_id

Entity Name: Pre-Reqs

Attributes: Added prereq_id, course_id

Keys: prereq_id, course_id

Functional dependencies:

course_id → Courses

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	prereq_id		Pre Reqs
	course_id		Courses

Attribute closures (if any): (prereq_id)+ ={course_id, Courses};
(course_id)+ ={Courses};

Unique keys: prereq_id

Entity Name: Waitlist

Attributes: Added waitlist_id, course_id, waitlistPriority

Keys: waitlist_id, course_id

Functional dependencies:

course_id → Courses

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	waitlist_id		waitlistPriority, course_id

Attribute closures (if any): (waitlist_id)+ ={course_id, waitlistPriority};
(course_id)+ ={Courses};

Unique keys: waitlist_id

Entity Name: Cost

Attributes: Added cost_id, course_id, costCourse

Keys: cost_id, course_id

Functional dependencies:

course_id → Courses

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	waitlist_id		costCourse, course_id

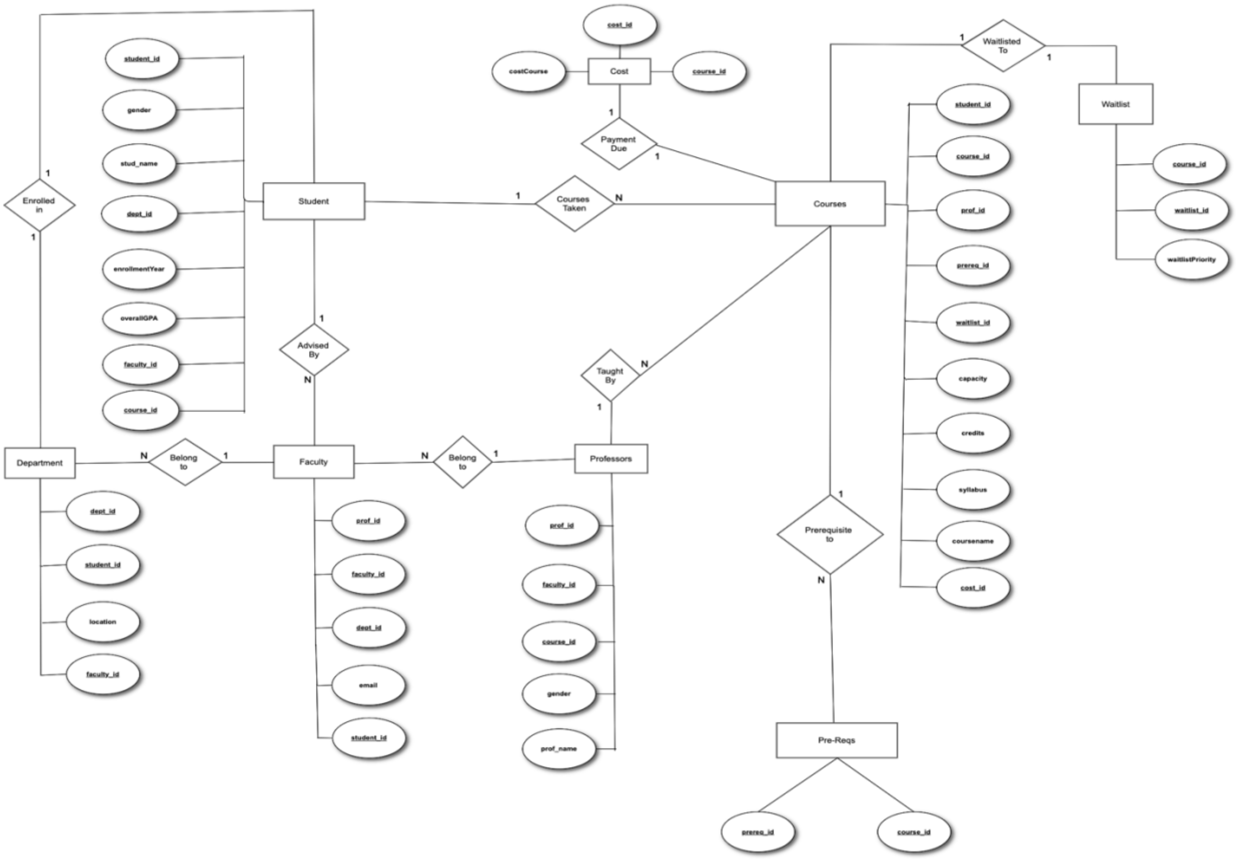
Attribute closures (if any): (cost_id)+ ={course_id, costCourse};

(course_id)+ ={Courses};

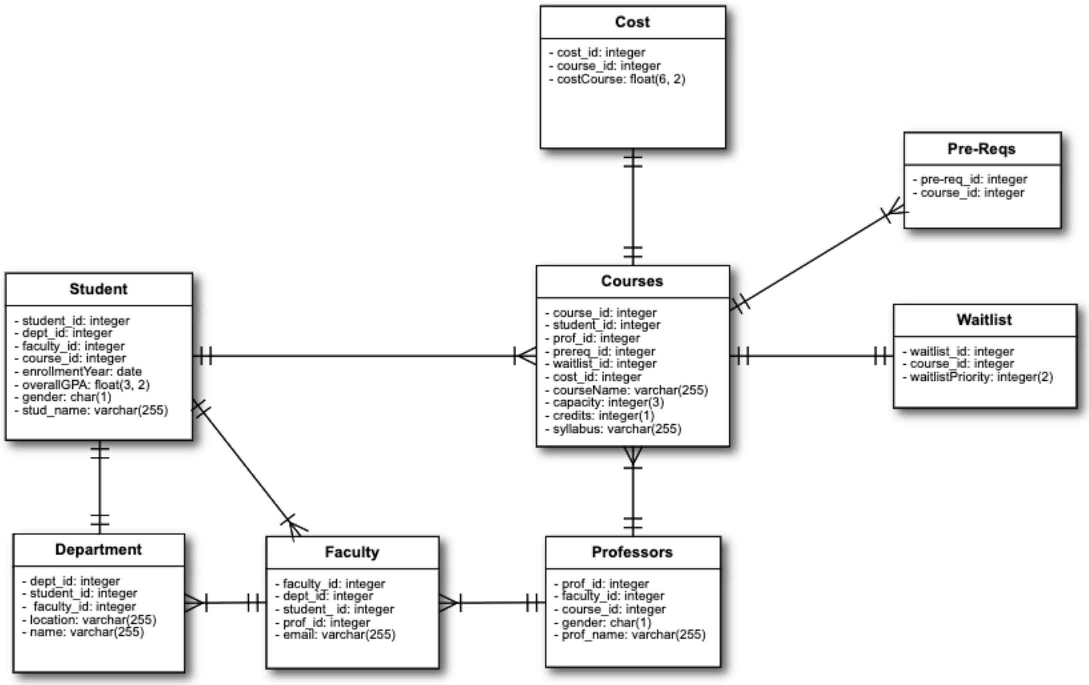
Unique keys: cost_id

Entity Relationship Diagram

Chen Notation:



Crow-Foot Notation



Assumptions and Constraints

- Unique attributes can’t be represented in the ERD
- Assumption that the university database is connected to the Internet
- Every table has a primary key
- Attributes like the age of students are implied given their enrollment date
- Assumption that the information in the database is accurate
- Assumption that the system runs smoothly and isn’t easily corrupted
- There’s a set limit for the number of students who are enrolled in a last or in the waitlist
- Each student can enroll in multiple courses, but there’s a limit to the number of courses a student can enroll in
- Unique attributes can’t be represented in the ERD
- Every course has a department, but a department doesn’t have every course
- Every course cost some amount of money, but not all courses cost the same
- Student entity will have fields such as enrollment year to help qualify students for certain courses
- Faculty group will have a n number of students based on an order sort like last name descending
- Every student has a faculty advisor, but a faculty group doesn’t have every student
- Every pre-req has a course, but not all courses have pre-reqs
- Every syllabus is a part of a course, and all courses have a syllabus
- There are enough users that are interested in the online registration system for it to be an effective database
- Mutated data can’t be displayed in an ERD effectively
- Some entities have a total participation constraint
- Not all constraints can be expressed in an ERD model
- Relatively abstract
- An ERD isn’t effective in displaying the minute details of a database

MILESTONE 4: NORMALIZATION AND

MILESTONE 5: PHYSICAL DESIGN

Assumptions and Constraints

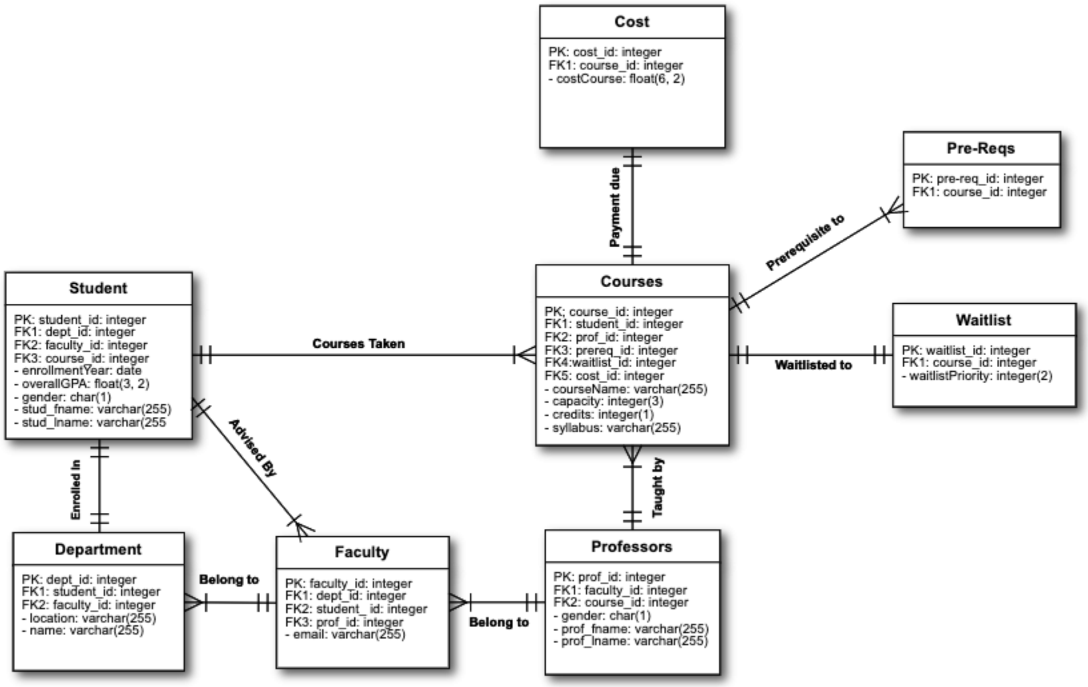
- Unique attributes can't be represented in the ERD
- Assumption that the university database is connected to the Internet
- A potential constraint is that the database isn't too normalizaed
- Every table has a primary key
- Attributes like the age of students are implied given their enrollment date
- Assumption that the information in the database is accurate
- Assumption that the system runs smoothly and isn't easily corrupted
- There's a set limit for the number of students who are enrolled in a last or in the waitlist
- Each student can enroll in multiple courses, but there's a limit to the number of courses a student can enroll in
- A potential constraint is that the database will not suffer from a degradation in performance after normalization
- Unique attributes can't be represented in the ERD
- Every course has a department, but a department doesn't have every course
- Every course cost some amount of money, but not all courses cost the same
- Student entity will have fields such as enrollment year to help qualify students for certain courses
- Faculty group will have a n number of students based on an order sort like last name descending
- Every student has a faculty advisor, but a faculty group doesn't have every student
- Every pre-req has a course, but not all courses have pre-reqs
- Every syllabus is a part of a course, and all courses have a syllabus
- There are enough users that are interested in the online registration system for it to be an effective database
- Mutated data can't be displayed in an ERD effectively
- Some entities have a total participation constraint
- Not all constraints can be expressed in an ERD model
- Relatively abstract
- An ERD isn't effective in displaying the minute details of a database

Naming Conventions

The naming standards I used are in tune and followed by the entity names for easier understanding. Every primary key is unique. My naming standards are consistent regarding letter case, are not verbose, and use underscores when required.

For Milestone 5, I followed the rules to convert to convert the logical schema into a relational schema. These rules are that each attribute of an entity becomes an attribute in a related table, each entity will be transformed into a table in the database, and lastly, every relationship in the database schema has a corresponding foreign key constraint.

Entity Relationship Diagram (Physical Design)



Tables

Name of the table	***Department***			
Description	The department is where faculty members work and students of a certain major are a part of.			
Attribute	**Description**	**Type**	**Examples of values**	**Notes**
dept_id	ID of the department	Integer	Between 1 and 9999999	
name	name of the department	VarChar(255)	Department of Mathematics	Can’t be null
Functional Dependencies and Keys				
Functional dependencies	faculty_id → Faculty student_id → Student			
Candidate keys	dept_id			
Normalization				
1NF	**Yes**	Every cell has a unique value		
2NF	**Yes**	Each table is in 1NF and no non-prime attribute is dependent on the proper subset of any candidate key of table		
3NF	**Yes**	Every non-key attribute depends only on a key		
BCNF	**Yes**	Relation is in 3NF		
Physical Design				
Primary Key	**dept_id**			

Foreign Keys	**student_id, faculty_id**
SQL Code	CREATE TABLE Department (dept_id int PRIMARY KEY, name varchar(255) NOT NULL, FOREIGN KEY(faculty_id) REFERENCES Faculty(faculty_id), FOREIGN KEY(student_id) REFERENCES Student(student_id));
Count of records in the table	**20**

…

Name of the table	***Student***			
Description	A person who's enrolled in the University. Can either be an undergraduate or a graduate			
Attribute	**Description**	**Type**	**Examples of values**	**Notes**
student_id	ID of a student	Integer	Between 1 and 9999999	
overallGPA	overall GPA of a student	Float(3,2)	3.45	Can't be null
stud_fname	name of the student	VarChar(255)	Bob	Can't be null
stud_lname	name of the student	VarChar(255)	Jones	Can't be null
gender	gender of the student	Char(1)	M, F	
enrollmentYear	year of enrollment	year	2017	Has to be larger than 2015, but less than 2021
Functional Dependencies and Keys				
Functional dependencies	student_id → Student, faculty_id → Faculty, dept_id → Department, course_id → Course			
Candidate keys	student_id			
Normalization				
1NF	**Yes**	Every cell has a unique value		
2NF	**Yes**	Each table is in 1NF and no non-prime attribute is dependent on the proper subset of any candidate key of table		
3NF	**Yes**	Every non-key attribute depends only on a key		
BCNF	**Yes**	Relation is in 3NF		
Physical Design				
Primary Key	**student_id**			

Foreign Keys	**faculty_id, dept_id, course_id**
SQL Code	CREATE TABLE Student (student_id int PRIMARY KEY, overallGPA float(3,2) NOT NULL, stud_fname varchar(255) NOT NULL, stud_lname varchar(255) NOT NULL, gender char(1), enrollmentYear date, FOREIGN KEY(faculty_id) REFERENCES Faculty(faculty_id), FOREIGN KEY(dept_id) REFERENCES Department(dept_id), FOREIGN KEY(course_id) REFERENCES Courses(course_id));
Count of records in the table	**20**

Name of the table	***Courses***			
Description	The classes offered in a university that a student enrolls in and an instructor teaches in			
Attribute	**Description**	**Type**	**Examples of values**	**Notes**
course_id	id of the course	Integer	Between 1 and 9999999	
capacity	number of students that can still enroll in the class	Integer(3)	30	Can’t be null
credits	number of credits in the course	Integer(1)	3	
syllabus	the syllabus of the course	VarChar(255)	https://psu.instructure.com/courses/2114308/assignments/syllabus	
coursename	the name of the course	VarChar(255)	Intro to Statistics	Can’t be null
Functional Dependencies and Keys				
Functional dependencies	course_id → Courses student_id→ Student, waitlist_id → Waitlist, cost_id → Cost, prereq_id → Pre Req, and prof_id → Professor			
Candidate keys	course_id			

Normalization		
1NF	**Yes**	Every cell has a unique value
2NF	**Yes**	Each table is in 1NF and no non-prime attribute is dependent on the proper subset of any candidate key of table
3NF	**Yes**	Every non-key attribute depends only on a key
BCNF	**Yes**	Relation is in 3NF
Physical Design		
Primary Key	**course_id**	
Foreign Keys	**student_id, prof_id, prereq_id, cost_id, waitlist_id**	
SQL Code	CREATE TABLE Courses (course_id int PRIMARY KEY, syllabus varchar(255), coursename varchar(255) NOT NULL, credits int(1), capacity int(3) NOT NULL, FOREIGN KEY(student_id) REFERENCES Student(student_id), FOREIGN KEY(prof_id) REFERENCES Professors(prof_id), FOREIGN KEY(prereq_id) REFERENCES PreReqs(prereq_id), FOREIGN KEY(cost_id) REFERENCES Cost(cost_id), FOREIGN KEY(waitlist_id) REFERENCES Waitlist(waitlist_id));	
Count of records in the table	**20**	

Name of the table	***Faculty***			
Description	The staff members of a University that are available for students as resources			
Attribute	**Description**	**Type**	**Examples of values**	**Notes**
faculty_id	id of faculty	Integer	Between 1 and 9999999	
email	email of faculty member	VarChar(255)	rahulsagitx@gmail.com	
Functional Dependencies and Keys				
Functional dependencies	faculty_id → Faculty student_id → Student, dept_id → Department prof_id → Professors.			
Candidate keys	faculty_id			
Normalization				
1NF	**Yes**	Every cell has a unique value		
2NF	**Yes**	Each table is in 1NF and no non-prime attribute is dependent on the proper subset of any candidate key of table		

3NF	**Yes**	Every non-key attribute depends only on a key
BCNF	**Yes**	Relation is in 3NF
Physical Design		
Primary Key	**faculty_id**	
Foreign Keys	**student_id, dept_id, prof_id**	
SQL Code	CREATE TABLE Faculty (faculty_id int PRIMARY KEY, email varchar(255), FOREIGN KEY(student_id) REFERENCES Student(student_id), FOREIGN KEY(dept_id) REFERENCES Department(dept_id), FOREIGN KEY(prof_id) REFERENCES Professors(prof_id));	
Count of records in the table	**20**	

Name of the table	***Professors***			
Description	Member of the faculty that teaches students the courses the students are enrolled in			
Attribute	**Description**	**Type**	**Examples of values**	**Notes**
prof_id	The id of the professor	Integer	Between 1 and 9999999	
prof_fname	name of professor	VarChar(255)	Sam	Can’t be null
prof_lname	name of professor	VarChar(255)	Smith	Can’t be null
gender	gender of the professor	Char(1)	M/F	
Functional Dependencies and Keys				
Functional dependencies	prof_id → Professors faculty_id → Faculty course_id → Courses			
Candidate keys	prof_id			
Normalization				
1NF	**Yes**	Every cell has a unique value		
2NF	**Yes**	Each table is in 1NF and no non-prime attribute is dependent on the proper subset of any candidate key of table		
3NF	**Yes**	Every non-key attribute depends only on a key		
BCNF	**Yes**	Relation is in 3NF		
Physical Design				
Primary Key	**prof_id**			
Foreign Keys	**course_id, faculty_id**			

	SQL Code	CREATE TABLE Professors (prof_id int PRIMARY KEY, prof_fname varchar(255) NOT NULL, prof_lname varchar(255) NOT NULL, gender char(1), FOREIGN KEY(course_id) REFERENCES Courses(course_id), FOREIGN KEY(faculty_id) REFERENCES Faculty(faculty_id));
	Count of records in the table	**20**

Name of the table	***PreReqs***			
Description	The pre-requisites required for some courses			
Attribute	**Description**	**Type**	**Examples of values**	**Notes**
prereq_id	The id of the pre-requisite	Integer	Between 1 and 9999999	
Functional Dependencies and Keys				
Functional dependencies	prereq_id → Pre-Reqs, course_id → Courses			
Candidate keys	prereq_id			
Normalization				
1NF	**Yes**	Every cell has a unique value		
2NF	**Yes**	Each table is in 1NF and no non-prime attribute is dependent on the proper subset of any candidate key of table		
3NF	**Yes**	Every non-key attribute depends only on a key		
BCNF	**Yes**	Relation is in 3NF		
Physical Design				
Primary Key	**prereq_id**			
Foreign Keys	**course_id**			

SQL Code	CREATE TABLE PreReqs (
	prereq_id int PRIMARY KEY,
	FOREIGN KEY(course_id) REFERENCES Courses(course_id)
	);			

Count of records in the table	**20**			

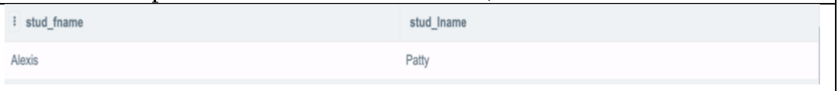
Name of the table	***Waitlist***
Description	The number of people who could not enroll in a class due to the capacity of the class being a its maximum

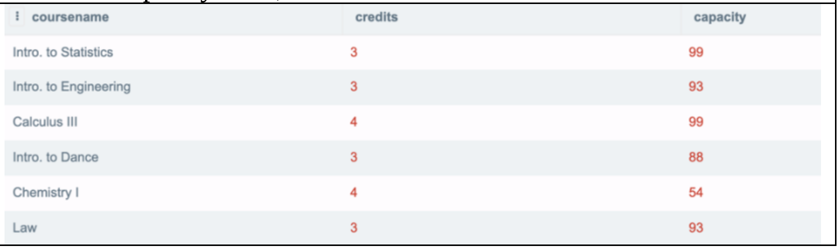
Attribute	Description	Type	Examples of values	Notes
waitlist_id	The id of the waitlist	Integer	Between 1 and 9999999	
waitlistPriority	the priority of the waitlist in the course	Integer(2)	03	
Functional Dependencies and Keys				
Functional dependencies	waitlist_id → Waitlist course_id → Courses			
Candidate keys	waitlist_id			
Normalization				
1NF	**Yes**	Every cell has a unique value		
2NF	**Yes**	Each table is in 1NF and no non-prime attribute is dependent on the proper subset of any candidate key of table		
3NF	**Yes**	Every non-key attribute depends only on a key		
BCNF	**Yes**	Relation is in 3NF		
Physical Design				
Primary Key	**waitlist_id**			
Foreign Keys	**course_id**			
SQL Code	CREATE TABLE Waitlist (waitlist_id int PRIMARY KEY, waitlistPriority int(2), FOREIGN KEY(course_id) REFERENCES Courses(course_id));			
Count of records in the table	**20**			

Name of the table	***Cost***			
Description	Table depicting the costs of various courses			
Attribute	**Description**	**Type**	**Examples of values**	**Notes**
cost_id	The id of the cost of the course	Integer	Between 1 and 9999999	
costCourse	The cost of the course	Float(6, 2)	1245.25	
Functional Dependencies and Keys				
Functional dependencies	cost_id → Cost course_id → Courses			
Candidate keys	cost_id			
Normalization				
1NF	**Yes**	Every cell has a unique value		
2NF	**Yes**	Each table is in 1NF and no non-prime attribute is dependent on the proper subset of any candidate key of table		

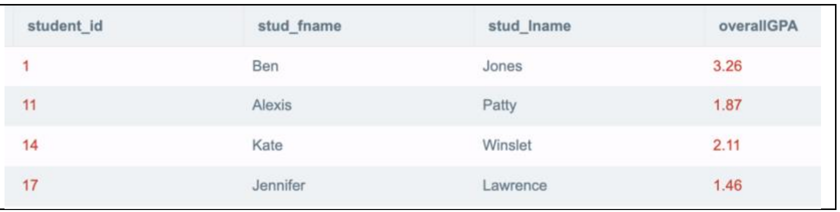
3NF	**Yes**	Every non-key attribute depends only on a key
BCNF	**Yes**	Relation is in 3NF
Physical Design		
Primary Key	**cost_id**	
Foreign Keys	**course_id**	
SQL Code	CREATE TABLE Cost (cost_id int PRIMARY KEY, costCourse float(6, 2), FOREIGN KEY(course_id) REFERENCES Courses(course_id));	
Count of records in the table	**20**	

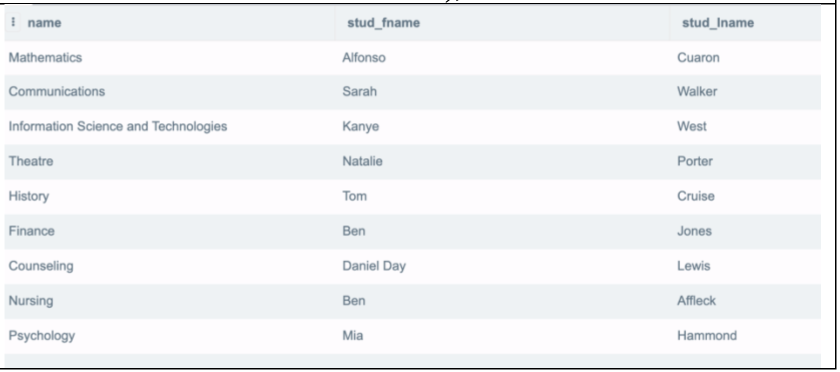
MILESTONE 6: SQL QUERIES

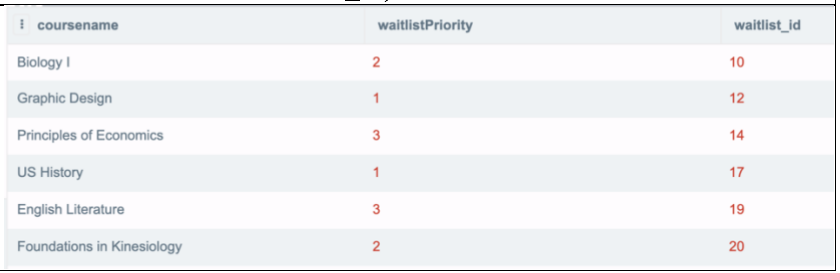
<table>
<tr><th>Query 1</th><th></th></tr>
<tr><td><b>English version</b></td><td>Return the student/students that are in the Department of Education</td></tr>
<tr><td><b>Source for the query need in the SRS document</b></td><td>SRS Document Page 14, Section 7.3, Paragraph 1</td></tr>
<tr><td><b>SQL sentence</b></td><td>SELECT stud_fname, stud_lname
FROM Department
INNER JOIN Student
ON Department.student_id = student.student_id
WHERE Department.name = 'Education';</td></tr>
<tr><td><b>Example of returned rows (cropped screen caption)</b></td><td>
<table>
<tr><th>stud_fname</th><th>stud_lname</th></tr>
<tr><td>Alexis</td><td>Patty</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Query 2</th><th></th></tr>
<tr><td><b>English version</b></td><td>Return the coursename, credits, and classes with over 50 course capacity remaining</td></tr>
<tr><td><b>Source for the query need in the SRS document</b></td><td>SRS Document Page 12, Section 7.3, Paragraph 1</td></tr>
<tr><td><b>SQL sentence</b></td><td>SELECT coursename, credits, capacity
FROM Courses
WHERE capacity &gt; 50;</td></tr>
<tr><td><b>Example of returned rows (cropped screen caption)</b></td><td>
<table>
<tr><th>coursename</th><th>credits</th><th>capacity</th></tr>
<tr><td>Intro. to Statistics</td><td>3</td><td>99</td></tr>
<tr><td>Intro. to Engineering</td><td>3</td><td>93</td></tr>
<tr><td>Calculus III</td><td>4</td><td>99</td></tr>
<tr><td>Intro. to Dance</td><td>3</td><td>88</td></tr>
<tr><td>Chemistry I</td><td>4</td><td>54</td></tr>
<tr><td>Law</td><td>3</td><td>93</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Query 3</th><th></th></tr>
<tr><td><b>English version</b></td><td>Return the information of students who are taking multiple classes and their corresponding information</td></tr>
<tr><td><b>Source for the query need in the SRS document</b></td><td>SRS Document Page 4, Section 1.1, Paragraph 1</td></tr>
<tr><td><b>SQL sentence</b></td><td>SELECT Student.student_id, Student.stud_fname, Student.stud_lname, Student.overallGPA
FROM Student
INNER JOIN Courses ON Courses.student_id = Student.student_id
GROUP BY Student.student_id
HAVING COUNT(*) &gt; 1;</td></tr>
</table>

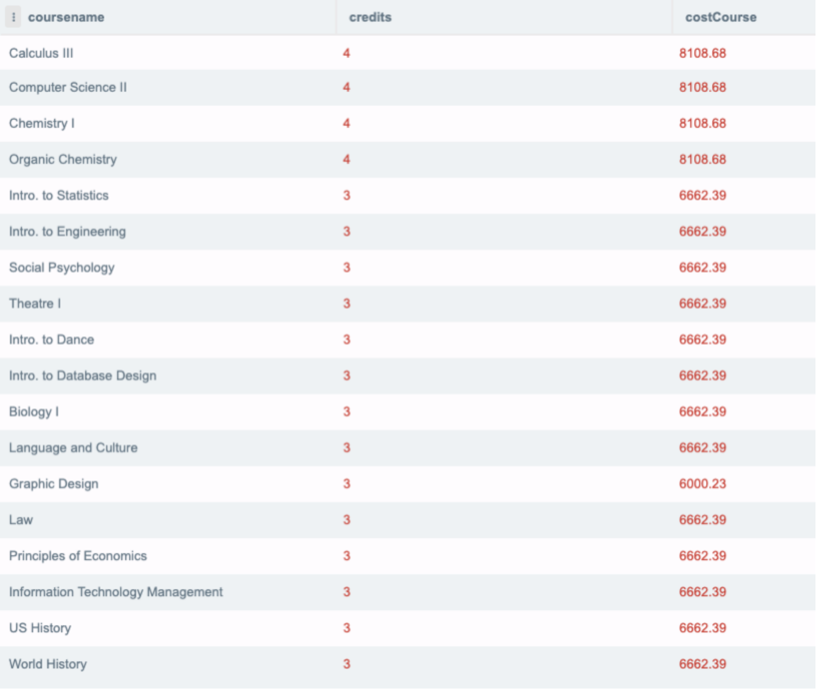
<table>
<tr><td><b>Example of returned rows (cropped screen caption)</b></td><td>
<table>
<tr><th>student_id</th><th>stud_fname</th><th>stud_lname</th><th>overallGPA</th></tr>
<tr><td>1</td><td>Ben</td><td>Jones</td><td>3.26</td></tr>
<tr><td>11</td><td>Alexis</td><td>Patty</td><td>1.87</td></tr>
<tr><td>14</td><td>Kate</td><td>Winslet</td><td>2.11</td></tr>
<tr><td>17</td><td>Jennifer</td><td>Lawrence</td><td>1.46</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Query 4</th><th></th></tr>
<tr><td><b>English version</b></td><td>Return the students who have a overall GPA above 3.0 and their department</td></tr>
<tr><td><b>Source for the query need in the SRS document</b></td><td>SRS Document Page 11, Section 6, Paragraph 6.1</td></tr>
<tr><td><b>SQL sentence</b></td><td><pre>SELECT Department.name, stud_fname, stud_lname
FROM Student
INNER JOIN Department ON Department.student_id = Student.student_id
WHERE Student.student_id IN (SELECT student_id
                FROM Student
                Where overallGPA > 3.0);</pre></td></tr>
<tr><td><b>Example of returned rows (cropped screen caption)</b></td><td>
<table>
<tr><th>name</th><th>stud_fname</th><th>stud_lname</th></tr>
<tr><td>Mathematics</td><td>Alfonso</td><td>Cuaron</td></tr>
<tr><td>Communications</td><td>Sarah</td><td>Walker</td></tr>
<tr><td>Information Science and Technologies</td><td>Kanye</td><td>West</td></tr>
<tr><td>Theatre</td><td>Natalie</td><td>Porter</td></tr>
<tr><td>History</td><td>Tom</td><td>Cruise</td></tr>
<tr><td>Finance</td><td>Ben</td><td>Jones</td></tr>
<tr><td>Counseling</td><td>Daniel Day</td><td>Lewis</td></tr>
<tr><td>Nursing</td><td>Ben</td><td>Affleck</td></tr>
<tr><td>Psychology</td><td>Mia</td><td>Hammond</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Query 5</th><th></th></tr>
<tr><td><b>English version</b></td><td>Return the classes that don’t have any seats remaining and are waitlisted</td></tr>
<tr><td><b>Source for the query need in the SRS document</b></td><td>SRS Document Page 11, Section 5, Paragraph 5.1</td></tr>
<tr><td><b>SQL sentence</b></td><td><pre>SELECT Courses.coursename, waitlistPriority, Waitlist.waitlist_id
FROM Waitlist
INNER JOIN Courses ON Courses.course_id = Waitlist.course_id
WHERE waitlistpriority > 0
GROUP BY Waitlist.waitlist_id;</pre></td></tr>
<tr><td><b>Example of returned rows (cropped screen caption)</b></td><td>
<table>
<tr><th>coursename</th><th>waitlistPriority</th><th>waitlist_id</th></tr>
<tr><td>Biology I</td><td>2</td><td>10</td></tr>
<tr><td>Graphic Design</td><td>1</td><td>12</td></tr>
<tr><td>Principles of Economics</td><td>3</td><td>14</td></tr>
<tr><td>US History</td><td>1</td><td>17</td></tr>
<tr><td>English Literature</td><td>3</td><td>19</td></tr>
<tr><td>Foundations in Kinesiology</td><td>2</td><td>20</td></tr>
</table></td></tr>
</table>

Query 6	
English version	Return the courses, credits, and cost of the course
Source for the query need in the SRS document	SRS Document Page 13, Section 9, Paragraph 1F

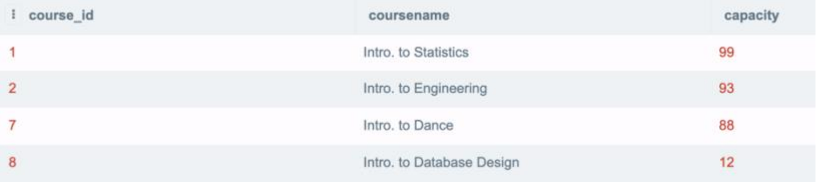
SQL sentence	SELECT Courses.coursename, Courses.credits, costcourse
	FROM Cost
	INNER JOIN Courses
	ON Courses.course_id = Cost.course_id
	ORDER BY credits DESC;

Example of returned rows (cropped screen caption)	

coursename	credits	costCourse
Calculus III	4	8108.68
Computer Science II	4	8108.68
Chemistry I	4	8108.68
Organic Chemistry	4	8108.68
Intro. to Statistics	3	6662.39
Intro. to Engineering	3	6662.39
Social Psychology	3	6662.39
Theatre I	3	6662.39
Intro. to Dance	3	6662.39
Intro. to Database Design	3	6662.39
Biology I	3	6662.39
Language and Culture	3	6662.39
Graphic Design	3	6000.23
Law	3	6662.39
Principles of Economics	3	6662.39
Information Technology Management	3	6662.39
US History	3	6662.39
World History	3	6662.39

Query 7	
English version	Return courses that are introductory courses for incoming students and pre-reqs for current students
Source for the query need in the SRS document	SRS Document Page 10, Section 5, Paragraph 5.1

SQL sentence	SELECT course_id, coursename, capacity
	FROM Courses
	WHERE coursename LIKE '%Intro%'
	ORDER BY course_id;

Example of returned rows (cropped screen caption)	

course_id	coursename	capacity
1	Intro. to Statistics	99
2	Intro. to Engineering	93
7	Intro. to Dance	88
8	Intro. to Database Design	12

…