

**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**

Document Control i

Work carried out by: i

Revision Sheet i

Milestone 1: Data Requirements 1

Purpose 1

Outcomes 1

System Overview 1

System Name or Title 1

Core requirements 2-3

Milestone 2: Conceptual Design 4

Purpose 4

Outcomes 4

Entity Relationship Diagram 4

Assumptions and Constraints 4

Milestone 3: Logical Design 5-9

Purpose 9

Outcomes 9

Entity Relationship Diagram 9-10

Assumptions and Constraints 10

Milestone 4: Normalization and 11

Milestone 5: Physical Design 11

Purpose 11

Outcomes 11

Assumptions and Constraints 11

Naming Conventions 11

Entity Relationship Diagram (Physical Design) 12

Tables 12-19

**Examples of values** 12-19

**Notes** 12-19

Milestone 6: SQL queries and 20-22

Purpose 20

Outcomes 20

# 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 | 1.1  5 | 4  5.1 |
|  |  | 13 | 8 | 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

Diagram

Description automatically generated

## 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

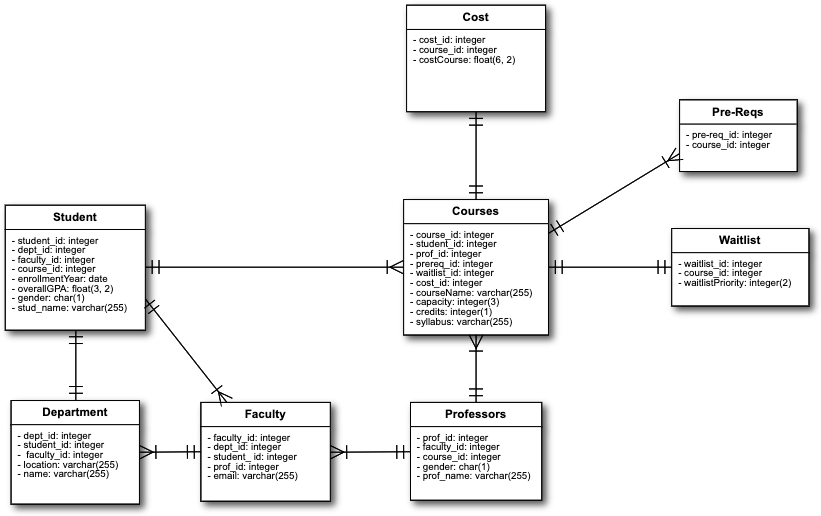
## Entity Relationship Diagram

**Chen Notation:**

Diagram

Description automatically generated

**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)

Diagram

Description automatically generated

## 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  stud\_lname | name of the student  name of the student | | VarChar(255)  VarChar(255) | Bob  Jones | Can’t be null  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  prof\_lname | name of professor  name of professor | | VarChar(255)  VarChar(255) | Sam  Smith | Can’t be null  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

|  |  |
| --- | --- |
| **Query 1** |  |
| **English version** | Return the student/students that are in the Department of Education |
| **Source for the query need in the SRS document** | SRS Document Page 14, Section 7.3, Paragraph 1 |
| **SQL sentence** | SELECT stud\_fname, stud\_lname  FROM Department  INNER JOIN Student  ON Department.student\_id = student.student\_id  WHERE Department.name = 'Education'; |
| **Example of returned rows (cropped screen caption)** |  |

|  |  |
| --- | --- |
| **Query 2** |  |
| **English version** | Return the coursename, credits, and classes with over 50 course capacity remaining |
| **Source for the query need in the SRS document** | SRS Document Page 12, Section 7.3, Paragraph 1 |
| **SQL sentence** | SELECT coursename, credits, capacity  FROM Courses  WHERE capacity > 50; |
| **Example of returned rows (cropped screen caption)** | A picture containing background pattern  Description automatically generated |

|  |  |
| --- | --- |
| **Query 3** |  |
| **English version** | Return the information of students who are taking multiple classes and their corresponding information |
| **Source for the query need in the SRS document** | SRS Document Page 4, Section 1.1, Paragraph 1 |
| **SQL sentence** | 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(\*) > 1; |
| **Example of returned rows (cropped screen caption)** | A picture containing application  Description automatically generated |

|  |  |
| --- | --- |
| **Query 4** |  |
| **English version** | Return the students who have a overall GPA above 3.0 and their department |
| **Source for the query need in the SRS document** | SRS Document Page 11, Section 6, Paragraph 6.1 |
| **SQL sentence** | 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); |
| **Example of returned rows (cropped screen caption)** | Table  Description automatically generated with medium confidence |

|  |  |
| --- | --- |
| **Query 5** |  |
| **English version** | Return the classes that don’t have any seats remaining and are waitlisted |
| **Source for the query need in the SRS document** | SRS Document Page 11, Section 5, Paragraph 5.1 |
| **SQL sentence** | 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; |
| **Example of returned rows (cropped screen caption)** | Table  Description automatically generated with medium confidence |

|  |  |
| --- | --- |
| **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)** | Table  Description automatically generated |

|  |  |
| --- | --- |
| **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)** | Graphical user interface, application, Teams  Description automatically generated |

…