Student Record-Keeping Database

Tanisha Mehta, Mahek Shah, Melisa Hayalioglu, Melissa Pinto
CP363: Database Design
Dec 10, 2021

1. Introduction

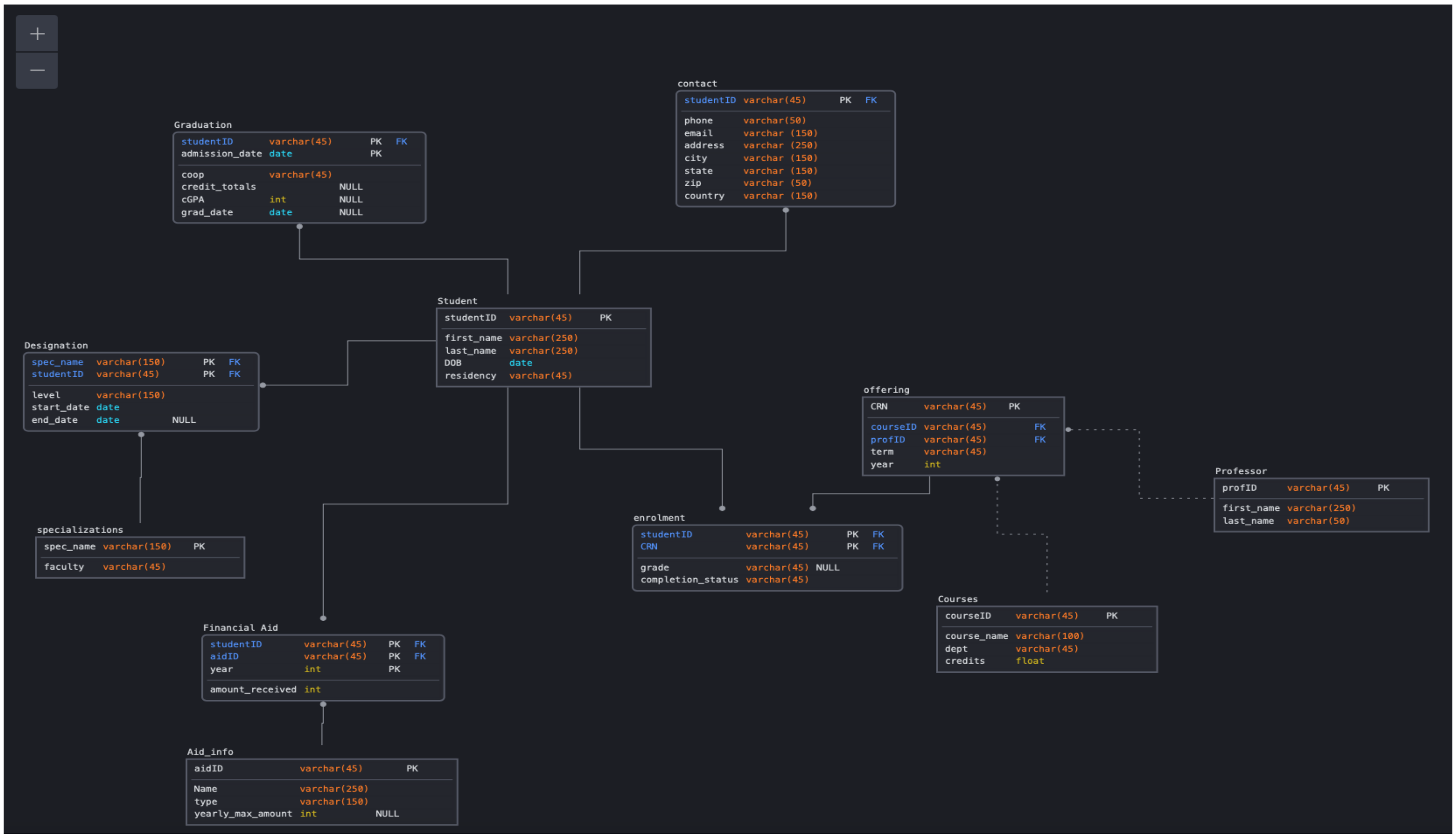
The purpose of creating this database is to create an organized system for a university to feasibly store student information and records. This database is designed to store information relevant to a university that is, it will manage relevant student information and it will be able to accommodate any necessary changes or updates with respect to: university programming (courses offered, financial aid types, specializations offered, etc.) and students. This includes general student information such as grades, courses taken by the student, the professors teaching each course, and the student's financial information. The information in the database should be able to provide information on a student's graduation records, their course re-enrollment and the degrees they pursue.

2. Detailed Database Design

2.1 Database design

2.1.1 Conceptual Diagram

Figure 1: UML diagram for all table relationships.[1]



[1] Created using (1).

2.1.2 Description

Figure 1 displays the conceptual model of the student record management database. The database is designed to store information that the university would need in order to manage the academic records and progress of students, along with supplementary information regarding professors and course registration. There are 3 tables– the aid_info, specializations and courses tables– that store general bulk information regarding financial aid types, all degrees offered at the university and all courses offered by the university respectively. Then, each of the three tables are related to 3 sub-tables which store specific information regarding each student– as is the case with designation and financial_aid tables – or regarding each course's offering. In this manner, these three tables are subsets of the aforementioned three bulk-information tables. Designing the database like this enables it to be more flexible (for instance, the university can add a new specialization at any time and this will not affect any relationships between tables) and it helps prevent any insertion anomalies that might otherwise occur. Also, the breadth of information stored in the database enables a user to quickly access any pertinent student information such as their: degrees (majors and minors), courses taken, marks obtained, contact information, repeated courses, etc.

2.1.3 Purpose of Each Table

1. Student Table
   The student table contains basic student information. Its attributes are the student's ID, first name, last name, date of birth and residency (whether the student is international or domestic). The primary key in this table is the **studentID** and its set to data type varchar which will take a string length up to 45 characters of letters, number, and special characters which will support our student ID of studxxxxx partial letters of a prefix "stud", and then 5 numbers. All students must have a unique **studentID** - and as it is set to a primary key, it will have a unique constraint placed upon it. **studentID** is also referenced in other tables as well, and it will often act as the foreign key of other tables in order to link student data together. Additionally, other attributes are **first_name**, **last_name**, and **residency**, which are all datatype varchar which will take a string length up to 45 characters. Along with the **date** attribute using a built in date data type to store the birthdate of each student. The student table is one of the key tables that connects with other tables to further navigate and store information through the database.

2. Graduation Table
   The graduation table stores necessary information that is needed for the student's graduation - including date of admission, date of graduation, credits, co-op status, and the student's GPA. This information will all be necessary and useful for the administration to track the student's progress and eligibility to graduate. In the graduation table, the primary key is a composite key, composed of the **studentID** and

the **admission_date**. The **studentID** is referenced from the Student table, and in the graduation table, **studentID** acts as both a primary key and foreign key, enabling relationship between the two tables. Additionally, a date data type is used for the attribute **admission_date** as part of the composite primary key. If a student is part of the school's co-op program, then the anticipated graduation date will be pushed back thus changing the value of the grad_date column. Additionally, the student's cumulative GPA and total credits will be stored via the attributes cGPA and credit_totals respectively.

3. <u>Designation Table</u>
   The designation table stores information about a student's degree programs, the names of any majors or minors being taken by a student, as well as the dates enrolled (**start_date**) and graduated– **end_date** (this date is Null if the student hasn't yet finished their degree). The designation table contains a composite primary key, composed of the specialization name (**spec_name**) and student ID (**studentID**), enabling a relationship between the student table and the specializations table. The relationship between the specializations table references **spec_name** column which gives information about the type of faculty the student is part of. Based on that information, the **level** column is set if the degree is a minor or major. The designations table has been designed for students to take multiple majors or minors in combination. As the primary key consists of the **studentID** and the **spec_name**, the unique restriction is placed upon a student attempting to enroll in the same major or minor multiple times - this would not be permitted.

4. <u>Specializations table</u>
   The specializations table stores information regarding all of the different programs offered by the university, and the associated faculty. This table was necessary to create in order to allow students to have multiple majors and minors without redundancy of information in the designation table and also to enable the university to add new specializations without affecting any other tables. The specialization table contains a primary key for **spec_name**, and based on that, the name of faculty is given with the varchar data type. The specialization table stores general information on all available specializations at the university, and then the designation table references it and in this way, each student's group of majors & minors is a subset of all specializations stored in the specialization table.

5. <u>Financial Aid Table</u>
   The financial aid table keeps track of the students' financial aid information, which is necessary for the university when calculating tuition and the amount owed by students. The financial aid table contains a primary composite key consisting of **studentID** and **aidID**. It also contains two foreign keys, **studentID** which references the student table and **aidID** which references the aid_info table. Since the **aidID** is a foreign key, it establishes a relationshipwith the aid_info table, obtaining more specific information about the aid being awarded to the student. Overall, the financial aid table links a

student (via the **studentID**) and the aid given to that student over the course of their degree.

6. Aid_info table
   The aid_into table contains general information about each financial aid. This table is designed to supplement and provide info for student financial aid– such as looking up the name of an aid using its aidID and also the type of aid (is it a scholarship, loan, grant, etc.). Storage of general aid information in this way (in a separate table) is useful in helping prevent data anomalies and redundancy. Additionally, it allows the university to update the aid_info table (ex: if a new scholarship is being offered) without affecting any other tables. The primary key for the aid_info table is the **aidID**, which serves to link it with the financial_aid table. The aid_info table also contains the name of the aid, whether it is funded by the government or a scholarship, if it's a loan scholarship or award, and the total yearly max amount of each given aid if applicable (some financial aid's such as OSAP have a yearly maximum).

7. Contact Table
   The contact table is used to store the contact information for each student. This information is important to provide information to the university for reaching the student when needed. The primary key for the contact table is the **studentID** that is referenced from the Student table, as well as storing information related to the phone, email, address, city, state, country, and zip code. When NATURAL JOIN is performed on the contact and student table, the resulting table will contain all the student's personal information.

8. Offering table
   The offering table allows for the university to store and handle information regarding course registration. The **CRN** serves as an identification for each course, and is thus set to be the primary key of the offering table. It consists of the **courseID**, followed by a letter designating the term it is offered, and then two numbers to indicate the year it is offered and any additional characters to specify section (as some courses have multiple sections each term). The design of the **CRN** in this way enables a single course to have multiple offerings at the same time and also to distinguish between each offering (ex: in fall 2021 vs. winter 2022, etc.). By using the **CRN**, the database is able to differentiate between the same course taught in different semesters or different years, and then link that information with each student as needed. For example, a join can be performed between the offering and enrollment table in order to consolidate information on courses each student has taken, and the grade obtained in the course. The **CRN** is set to varchar data type with a string length of 25. Additionally, **courseID**, **profID**, and **term** are also set to varchar which will take a string length of 45 characters and are the foreign keys in this table which establishes relationships between the professor, courses, and enrollment tables since courseID references courses(**courseID**) and profID references Professor(**profID**).

9. <u>Courses</u>
   The courses table will store pertinent information regarding all courses available at the university. The courses table focuses specifically on information regarding a specific course, such as credits, and faculty, while the offering table stores relevant, but different, information– namely, the term each course is offered and which professors are teaching it. As the database was moved into 3NF, it was necessary to separate these two tables to prevent data anomalies, specifically that there would be significant amounts of repeated information regarding departments and credits. In the courses table, the **courseID** is a primary key, which connects it to the offering table. Both course name and department are set as varchar datatype. The credits are float since the value of the course can be 1.0 or 0.5.

10. <u>Professor</u>
    The professor table stores basic information regarding all professors at the university. It has only a few attributes. The professor table includes a primary key of the professor ID which is referenced in the offering table to visualize which professor has taught which course and when. The **profID** is the primary key, which enforces a unique constraint to ensure that no professor is entered multiple times into the table. Each professor ID is associated with the respective professor's first name and last name– both attributes of which are set to the varchar data type.

11. <u>Enrolment table</u>
    The enrolment table stores all information pertaining to a student's course history and performance in each course. Additionally, it contains an attribute: **completion_status** that indicates whether the student has withdrawn from a specific offering, passed, failed or deferred its exam, which provides a useful extra layer to filter through courses that have a high failure rate for example, or to see if a student has re-enrolled in a course after failing it on the first try. All this information can then be used by the university for student transcripts. The primary key for the table is a composite key consisting of **studentID** and **CRN**. Both of these fields are also foreign keys that reference the Student and offering tables respecively. Searching the enrolment table using a studentID will display all of the CRN's that student has registered in during their time at the university. Additionally, information about the student's grade and completion status will also displayed. As briefy mentioned before, completion status is necessary because it allows the university to track whether a student has failed a specific course on the first try and whether they have subsequently re-enrolled at a later offering. Of course, completion status also encodes more information other than a simple pass or failure; it can be set to any of the following: withdrawn (if a student has withdrawn from a course offering), failed, satisfactory (if a student took the course and passed), deferred (if the exam was deferred for one reason or other), in progress (if the course is still in progress).

Conclusion

In conclusion, the student record-keeping database includes important student information that should be easily and efficiently accessible. It was designed to allow easy access to all relevant details and to accommodate any changes to the university curriculum in the future without storing redundant data. This design document outlines the tables, their attributes and relationships to other tables and clarifies any design considerations needed when actually creating the database system.

References

1. *SQLDBM (n.d.)- Online Database Modeler*. SQLDBM. Retrieved December 1, 2021, from https://app.sqldbm.com/.
2. Bois, C. (n.d.). *CSV generator*. Toolbox for developers. Retrieved December 10, 2021, from https://extendsclass.com/csv-generator.html.