Group 4:
Hunter Ryan
Keaton Harvey
Elijah Posiulai
Daymien Zavala

Databases Final Report

(university) Database:

The university database manages information related to university degree programs, courses, semesters, instructors, sections of courses offered each term, and evaluations of how well these sections meet various degree goals. The schema is designed to ensure data integrity, reflect the underlying relationships, and support features like adding courses, linking them to degrees, setting goals, and recording evaluation outcomes.

Degree Table:

The Degree table defines distinct academic programs offered by the university. Each record corresponds to a unique degree, such as a Bachelor of Science (BS) in Computer Science or a Master of Science (MS) in Mathematics.

- degreeID: A short identifier for the degree (e.g., "0001" for "Computer Science, Bachelor of Science").
- name: A descriptive name for the degree program (e.g., "Computer Science").
- level: Indicates the academic level of the degree. It must be one of the approved values ('BA', 'BS', 'MS', 'Ph.D.', 'Cert').
- The primary key is degreeID, ensuring each degree is uniquely identified.
- There is a uniqueness constraint on (name, level) to prevent creating multiple degrees that have the same name and level.

- A CHECK constraint ensures that level is restricted to a predefined set of values.

Course Table:

The Course table stores information about the courses offered by the university. Each record corresponds to a single course, identified by a course number and associated with a descriptive name.

- courseNumber: A unique code identifying the course (e.g., "CSCI101").
- name: A human-readable title of the course (e.g., "Introduction to Computer Science").

Semester Table:

The Semester table enumerates the academic terms during which courses can be offered. Each row combines a year and a term (e.g., Spring 2024) to represent a specific teaching period.

- year: A four-digit integer representing the calendar year (e.g., 2024).
- term: A value from the predefined set {Spring, Summer, Fall}, indicating the portion of the year.
- The combination (year, term) serves as the primary key, ensuring each semester is uniquely identified.

Instructor Table:

The Instructor table stores information about instructors (faculty members) who teach courses.

- instructorID: A unique identifier for the instructor (e.g., "00012345").
- name: The instructor’s full name.

Course_Degree Table:

The Course_Degree table establishes a many-to-many relationship between Course and Degree. It indicates which courses are part of which degrees, and whether each course is considered a core requirement.

- degreeID: References a degree defined in the Degree table.
- courseNumber: References a course defined in the Course table.
- isCore: A boolean value indicating if the course is a core requirement (TRUE) or an elective (FALSE) for that degree.
- Primary key is (degreeID, courseNumber), ensuring uniqueness of the association.
- Foreign keys enforce referential integrity with the Degree and Course tables.

Goal Table:

The Goal table represents learning outcomes or objectives that each degree aims for its students to achieve. Goals are often tied to curricular requirements, and courses can be mapped to these goals.

- goalCode: A short code (4 characters) identifying the goal.
- degreeID: Indicates which degree this goal is associated with.
- description: A textual description explaining what the goal entails.
- Primary key is (goalCode, degreeID).
- A foreign key ensures that the goal is linked to an existing degree.

Section Table:

The Section table models an instance of a course being offered in a given semester. For example, "CSCI101 Section 001 in Fall 2024." Each row includes the specific instructor teaching that section and how many students are enrolled.

- courseNumber: References a course from the Course table.
- sectionID: A short identifier (3 characters) for a specific section of a course.
- year and term: Identify the semester from the Semester table during which this section is offered.
- instructorID: References the Instructor teaching this section.
- enrollmentCount: The number of students enrolled, must be non-negative.
- The primary key (courseNumber, sectionID, year, term) uniquely identifies a particular section offering.
- Foreign keys ensure that each section references valid courses, instructors, and semesters.

Evaluation Table:

The Evaluation table records assessment results of how well a particular course section meets the goals of associated degrees. This table ties together a section of a course, a degree's goals, and the actual performance indicators (grades and evaluation type).

- courseNumber, sectionID, year, term: Identify the specific section being evaluated.
- degreeID, goalCode: Identify the specific degree goal being evaluated.
- evaluationType: Describes the form of assessment (Homework, Project, etc.).
- gradeCountA, gradeCountB, gradeCountC, gradeCountF: Record how many students achieved each grade category.

- improvementNote: An optional note for improvement suggestions or remarks on goal attainment.
- Primary key (courseNumber, sectionID, year, term, degreeID, goalCode) ensures each evaluation record is unique to a specific section-goal combination.
- Foreign keys link evaluations to existing Sections and Goals.
- CHECK constraints ensure grade counts are non-negative.

Updated Schema:

```
CREATE DATABASE IF NOT EXISTS university;
USE university;

CREATE TABLE IF NOT EXISTS Degree (
    degreeID VARCHAR(10) PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    level VARCHAR(10) NOT NULL,
    CONSTRAINT unique_name_level UNIQUE (name, level),
    CONSTRAINT valid_level CHECK (level IN ('BA', 'BS', 'MS', 'Ph.D.', 'Cert'))
);

CREATE TABLE IF NOT EXISTS Course (
    courseNumber VARCHAR(50) PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);
```

```
CREATE TABLE IF NOT EXISTS Semester (
    year INT NOT NULL,
    term ENUM('Spring', 'Summer', 'Fall') NOT NULL,
    PRIMARY KEY (year, term)
);

CREATE TABLE IF NOT EXISTS Instructor (
    instructorID VARCHAR(8) PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS Course_Degree (
    degreeID VARCHAR(10),
    courseNumber VARCHAR(50),
    isCore BOOLEAN NOT NULL DEFAULT FALSE,
    PRIMARY KEY (degreeID, courseNumber),
    FOREIGN KEY (degreeID)
        REFERENCES Degree(degreeID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (courseNumber)
        REFERENCES Course(courseNumber)
        ON DELETE CASCADE
```

```
        ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS Goal (
    goalCode VARCHAR(4),
    degreeID VARCHAR(10),
    description TEXT NOT NULL,
    PRIMARY KEY (goalCode, degreeID),
    UNIQUE KEY goal_degree_idx (degreeID, goalCode),
    FOREIGN KEY (degreeID)
        REFERENCES Degree(degreeID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS Section (
    courseNumber VARCHAR(50),
    sectionID VARCHAR(3),
    year INT,
    term ENUM('Spring', 'Summer', 'Fall'),
    instructorID VARCHAR(8),
    enrollmentCount INT NOT NULL DEFAULT 0,
    PRIMARY KEY (courseNumber, sectionID, year, term),
```

```
    FOREIGN KEY (courseNumber)
        REFERENCES Course(courseNumber)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (instructorID)
        REFERENCES Instructor(instructorID)
        ON DELETE SET NULL
        ON UPDATE CASCADE,
    FOREIGN KEY (year, term)
        REFERENCES Semester(year, term)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CHECK (enrollmentCount >= 0)
);

CREATE TABLE IF NOT EXISTS Evaluation (
    courseNumber VARCHAR(50),
    sectionID VARCHAR(3),
    year INT,
    term ENUM('Spring', 'Summer', 'Fall'),
    degreeID VARCHAR(10),
    goalCode VARCHAR(4),
```

```
    evaluationType ENUM('Homework', 'Project', 'Quiz', 'Oral Presentation', 'Report', 'Mid-term',
'Final Exam', 'Other'),
    gradeCountA INT NOT NULL DEFAULT 0,
    gradeCountB INT NOT NULL DEFAULT 0,
    gradeCountC INT NOT NULL DEFAULT 0,
    gradeCountF INT NOT NULL DEFAULT 0,
    improvementNote TEXT,
    PRIMARY KEY (courseNumber, sectionID, year, term, degreeID, goalCode),
    FOREIGN KEY (courseNumber, sectionID, year, term)
        REFERENCES Section(courseNumber, sectionID, year, term)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (degreeID, goalCode)
        REFERENCES Goal(degreeID, goalCode)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CHECK (gradeCountA >= 0 AND gradeCountB >= 0 AND gradeCountC >= 0 AND
gradeCountF >= 0)
);
```

Brief Installation/User Manual:

This manual provides detailed instructions for installing and running the University Course Management System. The intended audience is a junior-level Computer Science student who is familiar with basic programming but has not yet studied databases. This guide will walk you through the setup process, running the program, and using its graphical interface, all without requiring prior database knowledge.

Step 1:

Python 3 installed

- Check if Python 3 is installed by opening a terminal or command prompt and typing:
  python3 --version
- If this command returns a version number (e.g., Python 3.10.0), you're set.
- If not, visit https://www.python.org/downloads/ to download and install Python 3 for your operating system.

Step 2:

MySQL Server Installed and Running

- This application uses MySQL to store its data.
- Download and install MySQL from: https://dev.mysql.com/downloads/.
- Make sure to note the root password you set during installation.
- After installation, start the MySQL server. On most systems, the server may start automatically, or you may have to start it manually.
- You can test if MySQL is running by opening a terminal and typing:
  mysql -u root -p

- Enter your MySQL root password when prompted. If you see a MySQL command prompt, the server is running.

Step 3:

Python Packages

- This program uses mysql-connector-python to connect to MySQL and tkinter for its graphical interface.
- mysql-connector-python can be installed by running:

  pip install mysql-connector-python
- If you have Python 3 installed, you can install the Tkinter package with:

  sudo apt-get update

  sudo apt-get install python3-tk
- After installation, you can verify Tkinter by running a Python shell and trying:

  import tkinter

Step 4:

Downloading the Program Files

- Obtain the program's source code files (Database.py) and the database schema file (UniversitySchema.sql). These will be downloaded from a GitHub repository called DatabasesProject
- Place the files in a single folder.
- Make sure Database.py and UniversitySchema.sql are in the same directory so that the program can properly locate and execute the schema when first run.

Step 5:

Configuring the Database

- The code assumes a MySQL user and password are set as follows:

  User: cs5330

  Password: pw5330

- Update user and password to match your MySQL user and password. If you are using the MySQL root user for simplicity, replace user="cs5330" with user="root" and password="pw5330" with your root password. If you created a dedicated user, use their credentials instead.

Step 6:

Running the program

- Open a Terminal/Command Prompt and Navigate to the directory containing Database.py
- Run the application and start the program by typing:

  python3 Database.py

- On the first run, the program will attempt to use the university database.
- If the database does not exist, it will create it automatically and run the UniversitySchema.sql file to build all the required tables.
- After the database setup is complete, a graphical user interface (GUI) will appear.

Step 7:

Using the Graphical Interface

- The GUI presents several tabs, each corresponding to different actions:
  - **Add Degree:** Enter a unique Degree ID (e.g., "0001"), the name of the degree (e.g., "Computer Science"), and the Level (e.g., "BS"). Requirements include

degree names being alphabetic characters and the valid levels being BA, BS, MS, Ph.D, and Cert.

- **Add Course:** Enter a Course Number (e.g., "CSCI101") and a Course Name (e.g., "Intro to CS"). Requirements include course names being alphabetic characters and valid course number format (2-4 uppercase letters + 4 digits).
- **Add Instructor:** Enter an Instructor ID (e.g., “12345678”), the Instructor Name (e.g., “Bob Smith”). Requirements include the instructor ID being exactly 8 numeric characters.
- **Add Goal:** Enter a Goal Code (e.g., “g100”), the unique Degree ID (e.g., "0001"), and description (e.g., “Do well on your goal”). Requirements include the goal code being any single character followed by 3 positive numbers.
- **Add Semester:** Enter a year (e.g., “2024”) and term (e.g., “Fall”). Requirements include the year being a valid 4 digit number and the terms being either Summer, Fall, or Spring.
- **Add Section:** Enter a Course Number (e.g., "CSCI101"), the SectionID (e.g., "001"), enter a year (e.g., “2024”), term (e.g., “Fall”), enter an Instructor ID (e.g., “12345678”), and Enrollment Count (e.g., “100”). Requirements include the SectionID being exactly 3 characters and the enrollment count being a non-negative integer.
- **Associate Course to Degree:** Enter a Course Number (e.g., "CSCI101"), the unique Degree ID (e.g., "0001"), and isCore (e.g., "1").
- **Associate Course with Goal:** Enter a Course Number (e.g., "CSCI101"), the unique Degree ID (e.g., "0001"), and enter a Goal Code (e.g., “g100”).

- **Add Course to Semester:** Enter a year (e.g., “2024”), term (e.g., “Fall”), enter a Course Number (e.g., "CSCI101"), the SectionID (e.g., "001"), enter an Instructor ID (e.g., “12345678”), and Enrollment Count (e.g., “100”).
- **Queries and Evaluations:** This area lets you view evaluation statuses, enter/update evaluation data (like grades and improvement notes), and run various queries (e.g., sections above a certain pass percentage). Select the appropriate sub-tab, fill in the required fields, and click the corresponding button to run queries or save evaluations.