**Authors: Usman Abbas, Gaurav Kuwar, Zibin Luo**

CSC 336 - Database Systems (Spring 2022)
May 22, 2022

# Final Project - Course Schedule Database

**Motivation. Why did we choose this database?**

The main objective of this project is to create a relational database based on a specified topic of interest. The database that stood out the most to our group was the course schedule. The reason why the course schedule database became our group's number one pick is due to the fact that we are currently students. This is why we believed the course schedule database would relate strongly to us. As college students, we are responsible for looking through a series of courses in order to create our own schedule. When creating a schedule, it is essential to look at the course subject, term date, start and end time, room location, and professor for the class. This is something that we students have been doing for years and is why our group had a strong interest in choosing this database.

**Note about PrintScreen: We have a screenshot of all SQL queries running and instead of screenshots for the python script, we provided the HTML output of the Jupyter notebook with the output of all the functions, and how the relations were changed by the function.**

**Author's and roles?**

We collectively discussed the database system design through meetings.

Gaurav Kuwar - Worked on SQL code for creating users, relations and giving privileges to users, creating helper functions, functions for selecting from database, updating database, formatting Jupyter notebook, and drawing the E-R diagram.

Usman Abbas - Worked on functions for deleting records from the database, updating records in the database, and Main sections of the Final Report, and formatting the final report.

Zibin Luo - Worked on functions for inserting records into the database, creating CSV files, and functions for populating relations with data from the CSV files.

**The logic of the database**

We have a Room relation, which contains the room_id, room_num, and building_name. The room_id is the primary key, while the room_num and building_name would be like the NAC building_name and with room_num 511 since room numbers can have letters it is a string datatype.

The Professor relation has the professor_id, first_name, last_name, which we deemed any other information such as the department of the professor unnecessary, as it would not only make the database more complicated but it would also go out of the scope of the topic of the database.

The Course relation, contains the course_id, course_num, department, and title, the course_id is just so we can conveniently identify unique rows of the table, but course_num and department are primary keys. An example would be department=CSC, course_num=336 and title=Database Systems.

The Class relation, has the following attributes section_id, room_id, course_id, year, term, professor_id, start_time, end_time, and days_of_the_week. section_id, course_id, year, and term are the primary key, since the idea is that a course will have some sections in a semester (which would be a year and term), this would be able to be uniquely identified as a row. For simplicity, we decided to assume that every semester, a class will get access to one room. (which is unrealistic, but we discuss the challenge we faced in the conclusion). Hence, we add a UNIQUE(section_id, course_id, year, term, room_id) constraint to represent this. We also have that a term should only be the values "SPRING", "FALL", "SUMMER", and "WINTER". room_id is a foreign key from the Room relation, while the course_id is a foreign key from the Course relation. Lastly, the professor_id is a foreign key from the Professor relation.

**Explanation of the function**:

**getConfig**-reads and formats data from authFn returns it.
**colsToQuery**-Takes a dict colsNameVal, and puts it into a query format for WHERE, SET parts of the query
**addWhere**-Builds the where part of the query using values in dict colsNameVal and uses the separator "AND" by default.

**addSet**-Builds the set part of the query using values in dict colsNameVal and uses the separator ",".

**getColNameVal**-Gets a dict args_dict and filters values in the args_dict that are None or they are in the exclude list.

**populateRoom**-Populates Room relation with csvfile.

**populateCourse**-Populates Course relation with csvfile.

**populateProfessor**-Populates Professor relation with csvfile.

**populateClass**-Populates Class relation with csvfile.

**selectRelation**-Takes a relation in which the name of a relation in the database is a config and returns all rows in the relation. Not only that but it also has a where argument, which adds a WHERE part to the query using addWhere when WHERE is non-empty.

**selectProfessors**-Returns select query of Professor relation, when arguments are passed they are used to filter with WHERE in the select query.

**selectCourses**-Returns select query of Course relation, when arguments are passed they are used to filter with WHERE in the select query.

**selectRooms**-Returns select query of Room relation, when arguments are passed they are used to filter with WHERE in the select query.

**selectClasses**-Returns select query of Class relation, when arguments are passed they are used to filter with WHERE in the select query.

**roomsInBuilding**-Get all rooms in a building

**selectDistinct**-Get list of distinct col_name in relation

**allBuildings**-Get list of all buildings in Room relation

**allDepartments**-Get list of all departments in Course relation

**allSectionsOfCourseBySemester**-Returns a the sections for a specific course by semester using course_num, department, year, term

**classInfo**-Returns all information a Student would need for attending a class, section_id, department, course_num, title, term, year, days_of_the_week, start_time, end_time, room_num, building_name, professor name, using join on multiple relations

**addRoom**-Insert new record to Room relation

**addProfessor**-Insert new record to Professor relation

**addCourse**-Insert new record to Course relation

**addClass**-Insert new record to Class relation

**deleteRoom**-Delete a row from class relation

**deleteProfessor**-Delete a row from class relation

**deleteCourse**-Delete a row from class relation

**deleteClass**-Delete a row from class relation

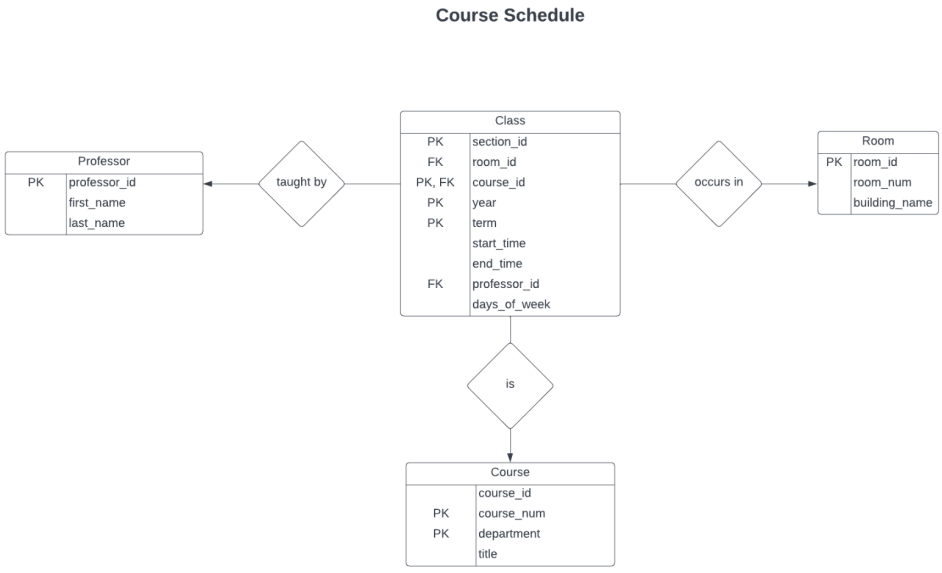
**updateRelation**-Update a row from class relation

**updateRoom**-Update a row from Room relation
**updateProfessor**-Update a row from Professor relation

**updateCourse**-Update a row from Course relation

**updateClass**-Update a row from Class relation

## E-R Diagram




## Conclusion

In conclusion, our group was successful in creating a relational database system for the course schedule. The course schedule database had many key implementations such as the four important relations as the Course, Class, Room, and Professor. Each of the listed relations included the necessary attributes that would be important implementation to the course schedule database. Not only that but it was also important that each of the data that were being imported into the database were unique. This is why we made sure to make heavy use of the primary key, foreign keys, and also constraints where ever we saw it necessary. However, when trying to build a constraint for the start_time and end_time, for every day of the week the class would occur, in a room, we realized that this constraint was extremely difficult to implement, and since we did not really have much time to complete the project because of finals, we decided to go with the simpler approach described in the “logic of the database” section.

The next portion of the project was to create python functions/modules that can be used to conduct various operations on the database. These function operations consisted of inserting data into the database, deleting data, populating the database and etc. We created an er-diagram so that we can analyze the relationship between each relation, as well as provide an outline of the functionality of our system.

This project has allowed us to test our learned skills from this course in order to implement this project. Overall, this project has given us the opportunity to improve our skills in database design and to get a better understanding of how this may be implemented in the real world.