CST363 Introduction to Database Systems

Project 1 – Degree Progress Checker

By Ivan Alejandre
Randy Son

Overview

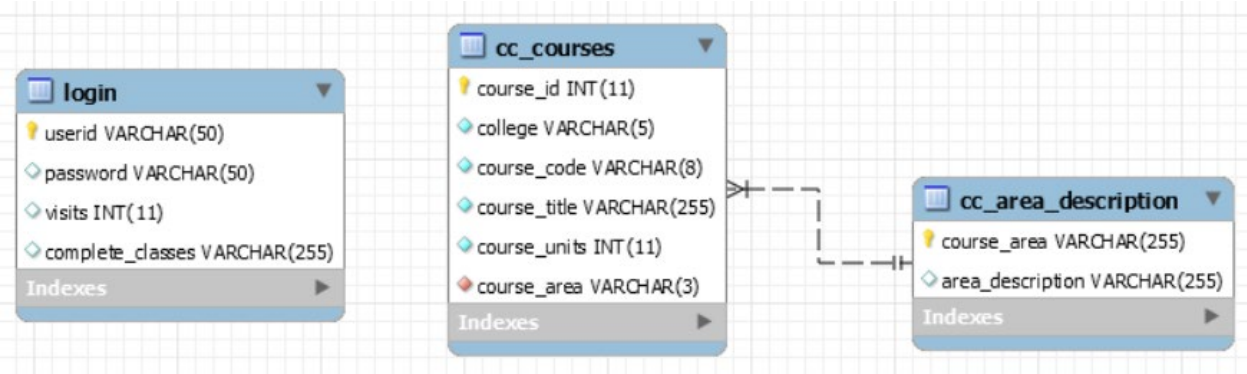
In the current state, we students in the CSUMB Computer Science program utilize a Word document called an Individual Learning Plan or ILP for short to obtain an overview of our degree progress. This file details the classes we have taken and which of them satisfy which areas for general education requirements for Cal State and each one of them is a lengthy and manual process to complete. Multiply that effort with the number of students in the program. Wouldn't it be easier to have an app that automates that for you?

That is where this app comes in. It seeks to reduce the amount of work needed by an academic adviser and provides a proof of concept that a degree roadmap is accomplishable.

To facilitate this, we took class list data from Assist, and parsed it into a tabular format that is ideal for a MySQL database.

In connection with the data, we utilized a similar login system from an earlier assignment that mimics a student signing into the college web portal. The web application, after verifying credentials, then executes a query for each course area, e.g. (A, B, C, D, E) and displays the available classes for each area. Then after having selected the classes they have completed, we store this data in the database for future use. The python script executes conditionals based on what the student selected. Then the student is able to see how many units they have left to satisfy each general education area.

ER Diagram



Currently, the tables in our schema are normalized in that we sought to reduce duplicate data across the schema. In addition, we record completed classes under the complete_classes column in the login table not as rows but as comma delimited data. We recognized the effects of this and will amend it in the next portion of the project.

In next design iteration for part 2, we plan on creating another table called course_history that will house the classes students have taken. Going this approach will bridge the gap between login and cc_courses, by entertaining a primary key and foreign key relationship between the tables. And by doing this, we will be fulfilling the requirements of third normal form of having all columns dependent on a primary key, making OLAP queries in the future much easier.

In addition, we chose not to pursue a denormalized approach to our schema to make queries simple and to avoid anomalies that are created when inserting, updating and deleting records.

Operational tables • **at least 3 tables with primary and foreign keys** • **sql script to create table and load with initial data**	**20 points**	
Web application • **2 or 3 (or more) html file and CGI *.py scripts for OLTP**	**20**	
PDF document showing • **Brief explanation of your application, its design and web application** • **ER diagram of operational tables** • **tables are normalized or explained why you chose not to** • **pages are numbers** • **title page with names of team members**	**10**	
Git account containing • **pdf file** • **sql script files** • **html and py files** • ***.mdb model file from Workbench**		
OLAP tables • **star schema design** • ***.mdb model file from Workbench** • **updated PDF with ER diagram of the OLAP tables** • ***.sql scripts to extract and load the OLAP tables**	**40**	
OLAP queries • ***.sql script with 5 select queries that might be typical of the kind of question that an OLAP user might ask. Use comments in the script to explain what the query is asking.**	**40**	