CPSC 437/537: Introduction to Database Systems
Course Final Project
Fall 2022

Contents

1 Project Description

1.1 Objectives

The main goal of the course project is to introduce the use of database systems in a practical application. As part of developing this project, you will exercise: schema design, data extraction, entity resolution, SQL queries, and other tools to help realize your application. The project is intentionally left open-ended to offer flexibility in choosing features to implement, and to think about how to make these features efficient. The project will account for 20% of the total grade, but is meant to be fun and creative – it is not meant to feel like hard work.

There have been many amazing projects in the past, but to inspire teams this year we will have all participating teams and the course staff vote for the **best demo project** at the end of the year, and we will post the winners and their project on the course website.

1.2 Requirements

While leaving most of the project open-ended, we require the following of all projects:

- Real data: Must use at least two (2) sources of real-world data. (See Resources, Section 4.)
- Data cleaning: Frequently, online data is incorrect and incomplete. As you import the data, you should run simple scripts to check for errors.
- Entity resolution: Since you are importing data from multiple sources, you will need to “connect” the data. For instance, check for misspellings of names and places, capitalization, etc.
- Database management: You should store and query data from well-designed tables; have a normalized relational schema.
- Complex SQL query: Include at least one (1) complex SQL query. (i.e., with multiple joins, subqueries, aggregation, etc. These can be in the application queries, or part of the schema design.)

2 Project Deliverables

Project submission method will be through Canvas; we will give the details later.

2.1 Milestones

- (unofficial) **Milestone 1: October 6** - Form teams of 3 and discuss project idea with course staff.

  You can use any development technology, programming language, and database system that your project requires.

  Each group member might also consider applying for AWS Educate, which grants $100 in usage credits. You need to use your **.yale.edu** email address to register. The approving process may take about one week, so **apply early**!

  With a total group amount of about $300, you should have enough to complete the project. However, if you exceed this amount through carelessness you will be responsible for overages. By this, we mean that you should turn off instances whenever they are not being used and NEVER publicly share your id and password or put them somewhere that they can be compromised. There have been incidents where hackers used the AWS Keys and deployed several EC2 instances and a bill of more than $1000 was generated.

- **Milestone 2: October 20** - Project outline and schema design.

  At this point, you should have a plan for completing the project and schema design. Submit a document with team members and brief project summary.

- (unofficial) **Milestone 3: October 27** - Populate tables.

- (unofficial) **Milestone 4: November 10** - Check progress on application.

- (unofficial) **Milestone 4: November 17** - Demo basic functionality.

  At this stage, you should have some basic functionality working and mostly prepare for the demo, not last-minute debugging.

2.2 Final Deliverable - December 1

There are two parts to the deliverable, the demo and the submission of materials. We will have a **Demo Day** on the last day that the class meets, on **December 8** for you to present your project. For the final submission, include:

- **Project Report.** Your report should include:
  – Introduction and project goals.
  – Basic architecture (high-level, please do not just dump all the classes you used).
  – Key features of the project.
  – Description of data.
  – Technical challenges.
- **Code.** Please include the entire source code (and submit one zip file with your report), or a link to a Github repository that the course staff can access at the end of your project report.

The official due date for the writeup and code is **December 1**, but we will accept submissions up until December 9 without any penalty.

3 Grading

Grading will roughly consider the following:

- Data cleaning
- Schema + indexing + optimization
- Idea + architecture
- Result (success from inception to realization, lucidity in code organization)
- Demo presentation

4 Resources: Datasets and Project Ideas

4.1 Some Datasets (hyperlinked)

- Summer Olympics dataset (can retrieve in CSV format)
- Wikidata
- DBpedia
- World Bank Open Data
- World Factbook
- Greatest Sports Nation

You may also be able to find data from Kaggle competitions or datathons. However, there are many interesting datasets out there so please feel free to explore and look for data that would suit your application.

4.2 Past Projects

The following are actual projects done by former students of the class. We hope that these will give you some ideas for your own project.

- MLB (Major League Baseball) Predictions.

  For this project, we plan on taking data from the Lahman's Baseball Database (http://www.seanlahman.com/baseball-archive/statistics), which has team statistics and standings, along with individual player statistics from 1871 to 2018. We will show why the database is in an appropriate normal format for the project. We will build an interface that uses Python and SQL to make player or team predictions. We might implement some machine learning techniques to help with the predictions and will display data on some public web portal.

- Determining Factors that Affect TV Revenue in US sporting events.

  Our preliminary project idea is to gather data on sporting events and build a model that is able to determine how different factors affect the TV revenue generated. Factors we will look at time of game, location, teams involved, etc.. The end goal is to create an application that can provide an estimate on how much revenue will be generated for an upcoming game based on these same factors.

- Automated Analysis of SQL queries using First-order Theorem Provers.

  Here is the overall idea:

  – Automated Analysis of SQL queries using First-order Theorem Provers.
  – Test case generation for SQL queries using symbolic methods.
  – Automatically populate test database based on SQL schema including database integrity constraints.

  For the front-end, I consider using Flex/Bison (https://www.gnu.org/software/bison/) to generate lexer and parser for a fragment of SQL and a Schema Definition language and implementing the abstract syntax tree and compiler in C++. For the back-end, I aim to employ Z3 theory solver (https://github.com/Z3Prover/z3) and/or KodKod model finder (https://github.com/emina/kodkod) for automated reasoning.

- Movie recommendation database system.

  Users will be able to search movies based on certain attributes and to create a profile and either comment or like a certain movie. These changes will be updated in the movie database as well and will be reflected in further searches.

- Timetable planner for Yale courses.

  Functionality: a calendar for users to search, add, organize courses at Yale, and also add todo list such as assignment dues in the timetable.

- League of Legends Winning Predictor.

  Designed to be a web application. Allow users to pick specific champions as a team, the application will calculate the winning rate between two teams. Use data from Riot game. Allow users to check specific champion's information and the winning rate prediction for the next season. Predict the best champion pick for a team.

- Recommendation of movies and television shows.

  As a slight modification on one of the recommendations in the project description, we will be constructing a cross-platform system for the recommendation of movies and television shows. We will link data from multiple platforms (e.g., Netflix, IMDB, etc.) in a relational database in order to construct more comprehensive show/movie preference profiles for more robust recommendations. Additionally, we will visualize interesting trends in the dataset.

- A price recommendation tool for New York Airbnb landlords.

  Given the large growth of Airbnb, more landlords are joining in this community. However, they may face a problem that they do not know how to price their apartments. Thus, we find a large dataset with 48.9K rows of data of Airbnb in New York. And we plan to build a rational database with this dataset and use machine learning techniques to predict the price of new apartments. Thus, the landlords can get a sense of the market trend and use the recommended price to maximize their profits. The technical challenges are building a good rational database schema and choosing features to predict the price. Moreover, since this is a large dataset, tuning the hyperparameters used in machine learning is also challenging and important.

- Analyzes climate-change related weather patterns given NOAA datasets.

  Develop a project that analyzes climate-change related weather patterns given NOAA datasets. We would utilize the publicly available NOAA datasets in order to track where intense weather such as, draughts, floods, hurricanes, or heat waves occur. We believe that this would be not only an interesting project, but also one that would help people understand the impacts of climate change.

- School rankings and more.

  Fetch universities' data (like faculties and publications) online, store it in database, and perform queries and ranking based on the data. Basic operation is to make queries with locations, subjects and other conditions specified by users, and then display the ranking list. Advanced functions will include searching institutions by name (accurate or vague searches) and adding privilege controls on database to distinguish system managers, users and other possible roles. Our stretch goal is to enable connection-based queries, such as finding similar schools, or searching for research partners (or potential partners) of certain faculties. If possible, we will also compare different database designs to see how the efficiency of these advanced features are influenced.

- Player Stats Comparison.

  An NBA or LOL player stats website that provides players' statistics and various tools and metrics to compare two or more players.

- Predicting a Movie's Economic Success.

  The general idea is to leverage a simple 3-layer feed-forward neural network to predict just how good of an investment a given movie should be, given past trends. Performance will be quantified in terms of gross revenue divided by production budget. Relevant details from a large, randomly sampled selection of movies from will be scraped from IMDb (Internet Movie Database, imdb.com) via the Python package IMDbPY.

- Spotify's search.

  We plan to use online data sets from Kaggle containing the top 100 most popular songs from Spotify and their musical features. We plan to supplement these datasets by querying more data from Spotify API to retrieve their most popular songs based on genre so that we can create at least two datasets from which to query. Our web application will be a more robust version of Spotify's search, allowing for additional parameters to be specified such as mood or occasion (which we will define as our own categories) or more specific features such as musical key or mode.