Dec 10th 2018

CS 316 Final Project

Group 15

Eddie Yang, Frank Yin, Frank Qu, Jack Claar

Duke Course Review Database

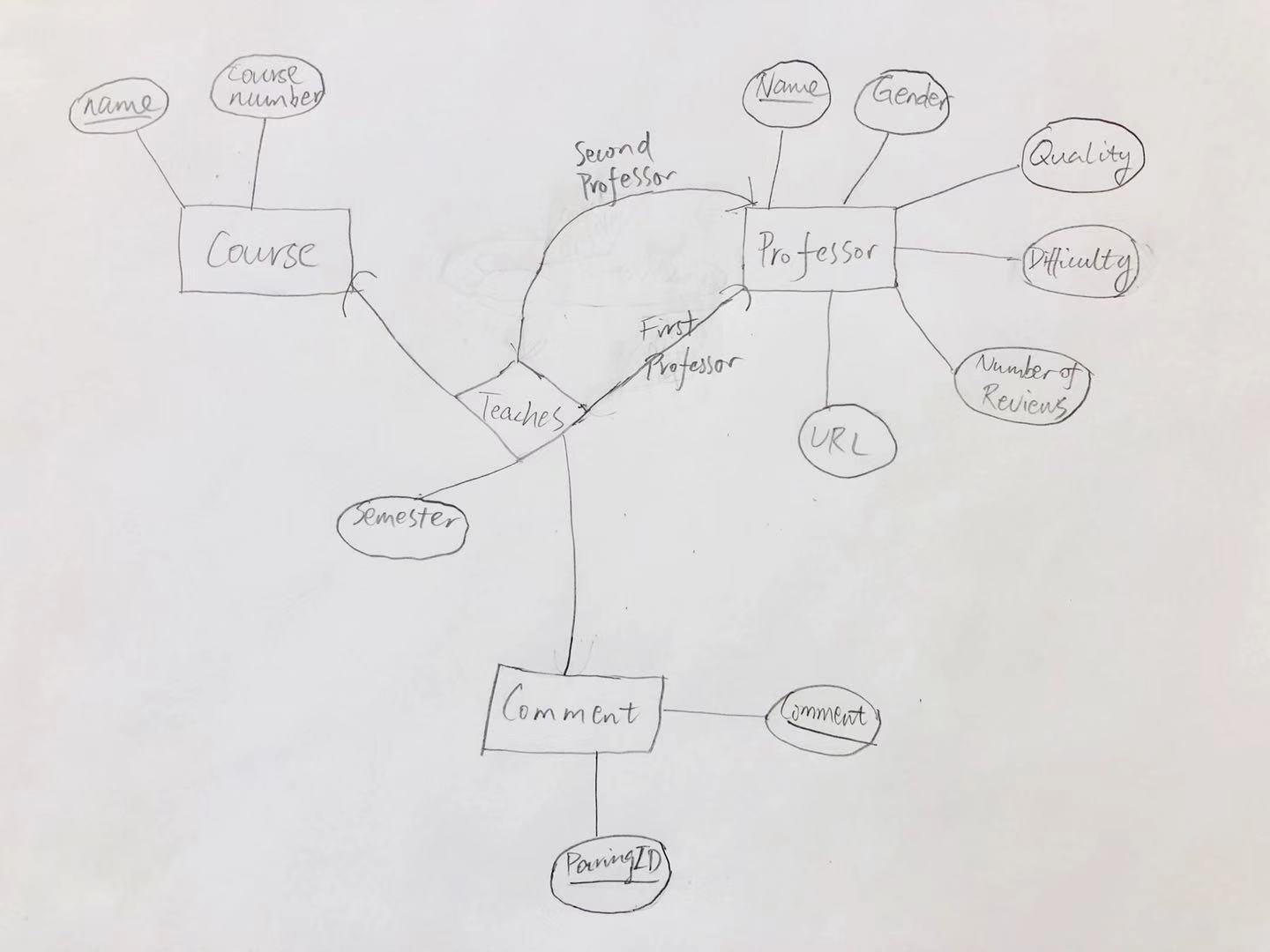
Application Description:

Having trouble to find reviews of courses you want to take? Want to know comments about courses taught by a specific professor? Our database includes thousands of exclusive reviews about Duke courses, collected from previous students who took the course.

The motivation behind our application is to design a platform that helps users to find specific information and past reviews about courses they are interested in. In addition, the application allows users to find professors they are interested in, their “ratemyprofessor” ratings, home pages and courses they taught in the past. Moreover, users can also contribute toward sharing their own comments about the courses listed in the application for others to view.

E/R Diagrams:

After many alterations and discussions between team members, the final application has the following E/R diagram:



The list of database tables with keys declared:

Course(name, course\_number)

Professor(name, gender, quality, difficulty, number\_of\_reviews,url)

Teaches(prof\_name, sec\_prof\_name, semester, course\_name, pairingID)

Comment(pairingID, comment)

The “Course” table has two attributes: name and course\_number. The name of the course is uniquely identified because different courses do not have the same course name, ex. “Introduction to Databases”. In the case of cross-listing between departments, the application aggregate all of the course numbers to be the attribute, ex. “AAAS 261/CULANTH 261”.

The “Professor” table has a list of attributes for professors, including name, gender, quality, difficulty, number\_of\_reviews, and url. For quality and difficulty, the data are collected from ratemyprofessor on Dec 9th, 2018. The number of reviews for a professor is counted based on how many course comments in the database are taught by the professor. Url of a professor is a website link for users to go to professor homepage to learn more about the professor.

The “Teaches” table is a record of relationships between courses and professors that taught them. The table includes primary and secondary professors that taught a specific class during a specific semester, it is assigned with a unique pairingID that functions as the key for each section.

The “Comment” table has two attributes: pairingID and comment. For each class, there maybe multiple comments, which means they share the same pairingID. Each pair of comment and pairingID creates a primary key for the tuple.

Key assumptions for our application are:

1. The application assumes there are at most two professors co-teaching the same section together. For instance, professor Astrachan and professor Forbes may teach the same CS210 section. That is the reason why the application included a secondary professor for our teaches table. For classes that are taught by three or more professors, only two are professors are included because they are rare.
2. The database assumes the names of courses are unique keys for entity “Course”.
3. The application assumes the names of professors are unique keys for entity “Professors”, since our “Professor” table is unlikely to be large enough for duplicates.
4. The application assumes that two comments cannot be the same. If they are the same, they are probably really short and not very helpful to our users.
5. 00s for professors’ quality/difficulty ratings means they are not rated yet.

Application Implementation:

The application uses JAVA/PLAY as the basic framework to create the database implementation. The CourseDB.java in app/model contains models and ways to interact with the course database. It has different classes such as “ProfessorInfo” and “CourseInfo” that hold various information about professors and courses. The model also has functionalities that allow users to retrieve professors and courses they searched and comments related with specific courses.

The Application.java in app/controllers controls the interaction between CourseDB.java and html. It calls the method in CourseDB.java to get the results back from the database and provide them for the corresponding html files to be displayed on the website application.

The scala/html files in app/views hold the application’s templates, which present Java objects they receive as inputs as HTML views.

Web Interface:

The main page of application allows users to search by professors and search by courses. The search course method utilizes concatenate method which allows users to search by both course number and course name. In addition, it allows users to look at all professors and courses in our database. Moreover, the users can sort professors by name, quality, and difficulty.

Once users entered their course name/professor name, the application returns a list of courses/professors that contains what the users entered. After users click into their desired findings, the course pages contain courses information, comments, and professors who taught the class. The professor pages contains professors’ information such as gender, ratings, url, and classes that he or she taught.

We also included a google survey link that allows users to contribute to sharing their experiences. The database can be updated on the backend by inserting new courses and related comments in the future. The reason behind is that the database is not very robust against spelling mistakes so having it checked before inserting is the better way to avoid potential duplicates and misinterpretations.

Evaluation for the system:

We believe that our database structure is intuitive and can support the application well. Compare with more professional websites such as ratemyprofessor.com, our database has original reviews that are not available anywhere else, and have more Duke courses and professors’ information such as Url links. The database is Duke-specific and does not have other universities courses or related professors’ ratings. In order to build this database, it required a lot of time and effort to clean up our dataset. For instance, professors’ name may be reversed or written incorrectly, courses name maybe abbreviation of its actual name. For professor names, some of them are also study abroad professors or visiting professors that consumed more time to correct them.

Any open issue or future directions:

This is the first time for all of our group members using JAVA/PLAY and Scala html files, there was a lot of learning and we encountered quite a few difficulties. Overall, there are several things we can further improve on in the future:

1. We can further improve our application’s searching ability by allowing ability to do fuzzy string search.
2. We would like to allow users to add more comments to our database directly through our UI. This will require us to identify whether a course, a professor, or a teaching pairing exists in our database accurately. This goes back to our first future direction.
3. We would like to allow users to give quality and difficulty ratings for professors and courses.
4. We would like to improve our UI to make it prettier.