Assignment 3.
Developing data products using data analytics and embedded SQL

This is the final assignment for this course, where you combine skills learned in Assignments 1 and 2 to produce a viable data product: CEA Course Recommender (CR). Users interact with the database through your application. This project is expected to improve your SQL programming skills and the ability to ask and answer questions about data through a comprehensive User Interface.

You can develop this application in a programming language of your choice[1] using either a local SQLite database, or a PostgreSQL database on the CDF server. You can even use any other DBMS, including MySQL and Oracle, however you will need to install the system by yourself and make sure that you are able to run your program during the final project demo.

Program specifications.

1. Introduction

The goal of this program is to use data about courses, topics and skills - collected in the CEA database - to produce course recommendations, similar to movie and music recommendations in such applications as Netflix or Pandora.

The idea of an automatic recommender is quite simple: find a group of users most similar to an active user and ask what did they rank highly. Use these highly ranked items as recommendations for an active user. If you want to explore the topic of recommenders in more details, you can read a book chapter excerpt provided on the assignment page.

When a new active user logs in, your application should collect data about this new user in order to find his/her nearest neighbors – students most similar to him/her. In addition, you should verify what courses this active user has already taken - in order to exclude them from your recommendations.

Once you collect all the required information, you present the user with the list of recommended courses, ranked according to the user-specified criteria.

2. Database

The database instance represents a snapshot of the CEA database - so familiar to you: all the included entities, attributes and relationships have the same meaning as in Assignment 1. SQL scripts for recreating an SQLite instance of this database are provided in cea_db.zip.

The database structure is summarized in the E/R diagram in Figure 1, and in the database schema below.

[1] Those of you who want your JDBC score for Assignment 1 updated, develop a comprehensive version using JDBC, and you will get the full mark for the JDBC part of Assignment 1.

Note that artificial integer IDs have been added to the week entities such as Course and Course Edition, to make joins more efficient. Also the ternary relationships between student, course edition and skill (topic) contain an additional attribute course id, to ensure that any combination of (skill, course) is valid: foreign key constraints.

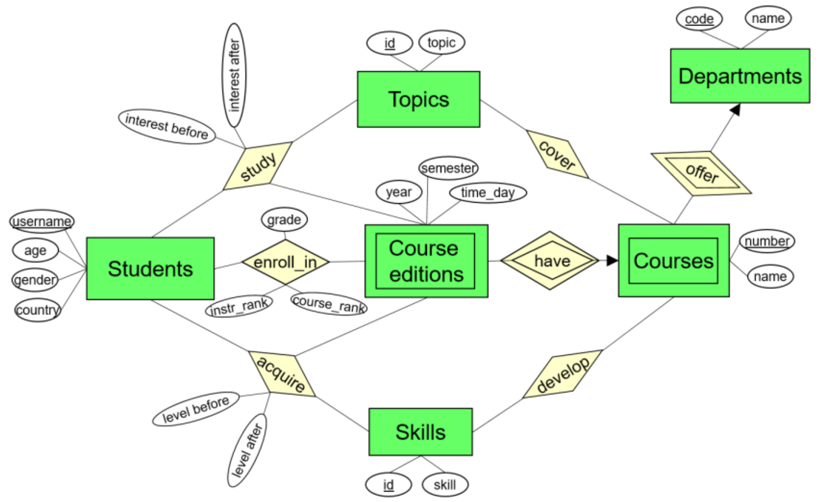

Figure 1. E/R diagram of the CEA database

Relational schema:

Departments (dept_code, dept_name)
Courses (course_id, dept_code, course_number, course_name)
Course_editions (edition_id, course_id, year, semester, total_students, time_day)
Students (username, permission, age, gender, native_country)
Skills (skill_id, skill)
Topics (topic_id, topic)
Course_topics (topic_id, course_id)
Course_skills (skill_id, course_id)
Enrollments (username, edition_id, letter_grade, course_ranking, instr_ranking)
Skill_rankings (username, edition_id, course_id, skill_id, rank_before, rank_after)
Topic_interests (username, edition_id, course_id, topic_id, interest_before, interest_after)
Letter_grades (letter_grade, min_grade, max_grade, gpv)

3. Functionality

3.1. Initial user information

An active user logs in and if his[2] user name is not found in the database, the initial demographic information is collected and recorded into the Students table. Otherwise the existing information is used.

3.2. Collecting user courses

The first step is to identify which courses this student has taken. This applies to both a new and a returning user. The courses are not added to the existing database yet, but are recorded into an appropriate data structure for the future use.

[2] *His, he, him* is used in a gender-neutral sense throughout the text.

3.3. Collecting user interests
Then the user needs to express his interest in the available topics. The topics are presented grouped by departments, to make navigation and selection easier. The user does not have to rank all the available topics.

3.4. Collecting user skills
Next the user accesses his level of different skills, which are also presented grouped by the departments. Again, the user has the right to omit some skills.

3.5. Computing potential recommendations
The program then finds 15 users which are most similar to an active user by their demographic characteristics, level of skills before the course, and topic interests before the course. It extracts all courses which these top 15 users have ranked, except the ones which have been already taken by the active user. Do not forget to exclude an active user himself from the set of his nearest neighbors.

3.6. Presenting recommendations based on the user-specified criteria

At this point the user is presented with the choice of how he wants his recommendations to be ranked.

- "Recommend courses with the best predicted grade"
  The first choice is by the expected grade. For each course that program has identified as possible recommendation, it computes an average grade, based on the numeric max_grade corresponding to the letter grade, and converts it to the letter grade again. Then all the recommendations are sorted by this average grade in descending order and 5 top are presented to the active user.

- "Recommend courses which promote my interests"
  The second choice is based on the development of student interests in different topics. The program will rank the potential candidate courses by an average increase in topic interest after the course, taking into account only those topics that the user expressed some interest in.

- "Recommend courses which improve my skills"
  The third choice is based on an average skill improvement after each course. Again, the courses with the best skill improvement are presented as course recommendations. For both option 2 and option 3, the relevant improvements are summed up for each course before averaging them among all 15 nearest neighbors.

- "Recommend courses which make me happy" (with the best predicted evaluation score)
  Finally, the user may select to see the recommended courses based on the best average course satisfaction.

  You do not have to apply additional weights to all the values computed above, however if you want you may weigh each value by the inverse of the distance from an active user (see book examples for details)

4. Data collection
After the user explored all his recommendations and before he exits, he is prompted to add his own data to the CEA database. For each course he has already taken (recorded in step 3.2), the regular questions such as grade, course rank and instructor rank are asked. The user may want to add new

topics and skills, and record skill improvements and topic interest dynamics. You may reuse your code from Assignment 1 for this part. This step of the data collection should be optional for the user.

Minimum requirements for the user interface:

- Comprehensive interaction with the user by maintaining a dialog (text UI is OK)
- Preventing exposure to the internals of the database implementation
- Easy to use, intuitive interface
- Friendly and comprehensive error messages following an invalid data entry
- You program should not crash under any circumstances – after any user action

Marking scheme

Each piece of functionality is graded separately:

- Initial user information: 5 points
- Collecting user courses: 10 points
- Collecting user interests and skills: 15 points
- Computing potential recommendations: 25 points
- Recommending courses with the best predicted grade: 10 points
- Recommending courses which promote my interests: 10 points
- Recommending courses which improve my skills: 10 points
- Recommending courses with the best predicted evaluation score: 10 points
- Data collection: 30 points
- Overall UI: 10 points
- Quality of SQL queries and embedded SQL: 15 points.

150 points in total for 15% of the course grade.

Bonus points

This application relies on collective intelligence. The best solution for this type of applications is a web application, where you can collect and analyze data from the large amount of users. Therefore, the students who will develop the Course Recommender as a web application will be awarded 5 bonus points towards the total course grade. A tutorial for embedding SQLite queries into PHP can be found here. However, you are free to develop in any other language/framework such as NodeJS[3].

[3] Warning about using NodeJS with PostgreSQL: last time I tried the NodeJS did not have a bug-free PostgreSQL driver, probably because this combination was not popular enough. SQLite or MySQL is the best choice of the RDBMS for NodeJS.