Project

CS 410/510 - Databases

Due **December 12, 2018 at 11:59 p.m.** This project will be graded out of 100 points, and will be worth 15% of your grade.

You may work on this project with a partner. If you do so, only submit one copy.

Revision Log

R3: December 6, 2018

- Fixed user ID / user name confusion in 'grade' command

R2: November 14, 2018

- Fixed section selection commands to be consistent with multiple sections.
- Add a comment about violating normal forms to implement constraints.
- Use 'points' instead of 'weight' in item commands to be consistent with data model description.
- Added comment about testing and Eclipse.

Summary

In this project, you will design and implement a Java application for managing grades in a class. This needs to be a command shell application using Cliché, like the examples we are building in class.

Problem Setup

Simplifying assumptions: this application will be used by a single instructor (so we do not need to handle multiple users), but it may be used for multiple classes.

- We need to be able to track grades for multiple classes. Each class has a course number (e.g. CS410), a year, a term (Fall/Spring/Summer), a section number, and a description, along with other fields you think might be relevant.
- Each class has multiple *categories*, each of which has a name and a weight.

- The class has multiple items, each of which also has a name, description, and point value, and is in one category. No two items in the same class can have the same name.
- We can have multiple students, each of which has a username (i.e. the first part of their e-mail address), a student ID, and a name.
- We can assign a student a grade for an item.

Part 1: Data Model

Draw an E-R model for the data needed for this problem.

Part 2: PostgreSQL Schema

Write SQL CREATE TABLE statements to implement your model. Include foreign key relationships and suitable indexes. Your database must be in the 4th Normal Form; if you need normal form violations for other desirable things such as constraints, that is acceptable; document in your schema file with comments. The comment should identify the normal form violated and explain why the violation is necessary.

Part 3: Java Program

Write a Java command shell program (using Cliché) to implement the application. You need to implement the following commands, along with others you deem necessary or helpful:

Class Management

- Create a class: `new-class CS410 Fall 2018 1 "Databases"`
- Activate a class:
    - `select-class CS410` selects the only section of CS410 in the most recent term, if there is only one such section; if there are multiple sections it fails.
    - `select-class CS410 Fall 2018` selects the only section of CS410 in Fall 2018; if there are multiple such sections, it fails.
    - `select-class CS410 Fall 2018 1` selects a specific section
- `show-class` shows the currently-active class

All other commands are to be interpreted in the context of the *currently active class*. This is like your current directory in a Unix command line.

Category and Item Management

- `show-categories` – list the categories with their weigghts
- `add-category "Name" weight` – add a new category
- `show-items` – list the items with their point values, grouped by category
- `add-item name "Category" "Description" points` – add a new item

Student Management

- `add-student username studentid "Last, First"`
- `show-students` – show all students
- `show-students EKS` – show all students with 'EKS' in their name or username (case-insensitive)
- `grade itemname username 20` – assign a grade of 20 for student with user name 'username' for item 'itemname'. If the student already has a grade for that item, replace it.

Grade Reporting

- `student-grades username` – show student's current grade: all items, visually grouped by category, with the student's grade (if they have one). Show subtotals for each category, along with the overall grade in the class.
- `gradebook` – show the current class's gradebook: students (username, student ID, and name), along with their total grades in the class.

Grade Calculation

For reporting grades, calculate grades out of 100. Rescale category weights so they sum to 100; within each category, compute the fraction of possible points a student has achieved (divide their total grade points in that category by the total possible points based on item point counts).

For both `student-grades` and `gradebook`, report grades two ways: a *total* grade, based on total possible points (including items for which the student does not have a grade at all), and an *attempted* grade, that is based on the point values of the items for which they have a grade.

You should do as much of the grade computations as possible in SQL – minimize your Java computations.

Extra Credit

Add a command '`import-grades itemname gradefile.csv`' that loads multiple students' grades for an item from a CSV file where student usernames are in the first column and student grades are in the second.

Submission

Submit a zip file containing the following files:

- `model.pdf`, a diagram of your data model in E-R syntax.
- `schema.sql`, an SQL script containing your DDL (database schema).
- `dump.sql`, a PostgreSQL dump file containing a dump of your database with example data.
- Your Java source code under `src/`.
- `pom.xml`, the Maven POM file used to import and build your project. A version of this file is attached to the project submission. It assumes that your main shell class is `edu.boisestate.cs410.gradebook.GradeBookShell`.

I need to be able to run '`mvn package`' in your unzipped submission to build your project and prepare it to run. If you have your project properly configured to use Maven, and zip up the project directory, that should work.

Test with the Maven-based build. Eclipse's compiler is unhelpfully forgiving.