Database Design Final Project Requirements

Introduction

The final project is a chance to work as part of a small team and put in practice many of the concepts we've learned throughout the semester. You are free to choose any domain your team decides as long as it meets the minimum requirements described in this document. Teams can be as small as a single person or as large as a 4 person team.

Data Model Requirements

Design a normalized database that meets the requirements listed in this document. Describe the database as a UML class diagram containing class names, properties, data types, relationships, and cardinality. Reify classes and relationships with the intent of implementing the database as a relational database. Your database should model at least one type of user, two domain objects, a relationship between a user and a domain object, and a relationship between domain objects. The following sections describe in more detail the requirements of the data model described in the UML class diagram.

User data model

Create a user data model that represents the users of your application. The datamodel should contain the following minimum set of properties:

- firstName: String
- lastName: String
- username: String
- password: String
- email: String
- dateOfBirth: Date

For graduate students, create at least two use data models.

Domain object data model

Create at least two domain object data models. A domain object is an object that is particular to the domain you choose. For instance, if you choose the domain of "movies", then "movies" and "theaters" are domain objects within the domain of "movies". If you choose the domain of "food", then "recipies", "ingredients", and "nutritional information" would be domain objects within the domain of "food". Another example: if you choose the domain "travel", then domain objects might include "plane tickets", "reservations", "hotels", and "airlines". Implement at least two domain objects. For graduate students, create at least three domain object data models.

Relationships

User to domain object relationship

Implement at least one relationship between your user data model and a domain object. The relationship should either be ***one to many*** or ***many to many***. For instance an "actor" user data model can be related to many "movies" domain object data model. For graduate students, implement at least two user to domain object relationships.

Domain object to domain object relationship

Implement at least one relationship between the two domain objects. The relationship should either be one to many or many to many. For instance, a record in "songs" could be related to many records in "play lists". For graduate students, implement at least two domain to domain object relationships.

User to user relationship (required for graduate students)

Implement a relationship between records in your user data model. The relationship should either be one to many or many to many. For instance, a "manager" can be related to many "employees" in the same "users" data model.

Inheritance relationship (required for graduate students)

Implement at least one inheritance relationship between two classes.

Portable Enumeration

Implement at least one portable enumeration. Do not use MySQL's enum keyword. For instance a "user" can have many "roles", e.g., "FACULTY", "STUDENT", and "STAFF". A "semester" can be any of several "quaters", e.g., "FALL", "SPRING", "SUMMER", "SUMMER1", or "SUMMER2". A "movie" can have a "genre" or "HORROR", "COMEDY", "DRAMA", or "SCIFI".

User Interface Requirements

Implement user interface screens that allow creating, reading, updating, and deleting each of the user models and domain object models. Implement relationships as navigations between related entities. For instance consider the following data model containing classes "students", "courses", and "sections", create the following screens:

- ***Student List*** - displays a list of all students or the students within a given section
- ***Student Editor*** - displays a particular user for editing or allows creating a new student, and navigate to sections for that student
- ***Course List*** - displays a list of all the courses

- ***Course Editor*** - displays a particular course for editing or allows creating a new course, and navigate to sections for that course
- ***Section List*** - displays a list of all sections or the sections a student is enrolled in or the sections for a given course
- ***Section Editor*** - displays a particular section for editing or allows creating a new section, and navigate to students for that section

If courses are related to many sections, then users should be able to navigate from a particular course record to the sections related to the course. If a student is enrolled in sections, then a student should be able to navigate to the sections they are enrolled in, and from a given section, they should be able to navigate to the students enrolled in the section.

Deliverables

As a deliverable, push all your code to a public git repository, and share the link to the repository on Canvas or Blackboard. Export your database to a local directory, zip it up into a single ZIP file called ***db_design_final_project_database.zip***, and include the file in your git repository. If the project needs more than one repository, submit all relevant repositories. If you've deployed your project to a remote server, include a link in a ***README*** file in your repository. In the ***README*** file, include the following:

1. Name of the project
2. Name of the team(s). At most two teams per project. One from each section, if applicable
3. At most 4 team members per project
4. Brief description of the project
5. Link to the latest data model as a single UML class diagram. The UML class diagram should be a single PDF document called ***db_design_final_project_UML.pdf*** and linked from the README
6. Description of user data model
7. Description of the two domain object data models
8. Description of the user to domain object relationship
9. Description of the domain object to domain object relationship
10. Description of the portable enumeration(s)
11. Description of the user interface requirements