Final Project's Information

The following information describes what you will need to turn in for the final project. There is an FAQ section at the end that you might find useful too.

Overview

You will be making a database driven website using HTML, Node.js (or PHP or any other language of your choice) and MariaDB/MySQL. HTML is a markup language used for laying out your website. PHP/Node or any other platform you choose will be the server side portion which will use a MySQL/MariaDB database to store, read and manipulate the information that is used to populate fields on the website.

You may work in pairs.

Your database should have at least 4 entities and at least 4 relationships, one of which must be a many to many relationship. Your database should be pre-populated with sample data.

It should be possible to add entries to every table individually and every table should be used in at least one select query. For the select queries, it is fine to just display the content of the tables, but your website needs to also have the ability to search using text or filter using a dynamically populated list of properties to filter on.

You also need to include one delete and one update function in your website. In addition, it should be possible to add and remove things from at least one many-to-many relationship and it should be possible to add things to all relationships.

The Various Pieces

Outline 5%

This should give us an overview of the topic. If we need some context about an obscure topic to understand how your site functions, it should go here. Basically introduce us to your topic.

Database Outline, in Words 5%

You should describe, in detail, the entities, properties and relationships of your project. The ER diagram and Schema will be graded based on if they match your description of how the entities and relationships work. If you don't go into detail, the grader will use their best judgment and it is final. For example, if we think a product should only be on one aisle in a grocery store but you think it should be able to be in multiple aisles and you don't clearly state it in this outline, you will likely lose points in the next three sections. Be as thorough as possible here.

ER Diagram 10%

This should be an ER diagram that matches your database outline. Anything that does not match the outline or uses incorrect notation will cause a deduction. Please stick to the notations from **SymbolKey.pdf (https://oregonstate.instructure.com/courses/1662151/files/69434610/preview)**

Schema 10%

This schema should follow the database outline exactly. It will again be graded on the extent to which it matches the outline with an emphasis on if relationships and keys are set up correctly. Again, please use only the notation from **SymbolKey.pdf (https://oregonstate.instructure.com/courses/1662151/files/69434610/preview)**

Data Definition Queries 15%

These are the queries that are used to generate your database. It is likely that these will only ever be run once, and that will be manually by you. These are not the queries that are used to do things like add, remove or view data in your site. These are going to be the queries to create tables and foreign keys.

All data types should be appropriate, foreign keys should exist and be correct and it should match the database outline.

Data Manipulation Queries 25%

These are going to be the queries that your website uses to let your users interact with data. They will be the things that the data in your forms are being submitted to. Anything that is a variable that you expect the user to fill in should be enclosed in square braces. For example

SELECT salary FROM employee WHERE salary > [salaryInput];

We only want to see SQL here. Do not include any of the JavaScript or PHP used to process the data.

Website Functionality 20%

It should be possible to add data to every table and view data from every table. It is generally not appropriate to have only a single query that joins all tables and displays them.

When dealing with relationships you cannot expect the user to manually type any data. You should allow the user to select things to relate to each other via drop-down menus or some other UI element where the user picks from existing items to add them to a relationship. When picking items you should display the name that makes the most sense to the user. This is very likely not the primary key. For example, in context of the BSG database, when adding a row to the relation between bsg_people and bsg_cert, we would expect the bsg_cert.title values to be shown in a dropdown/set of radio button instead of allowing entry of bsg_cert.id

It should be possible to both add and remove things from relationships.

Style 10%

Your website should be reasonably easy to navigate. We are not expecting CSS. But tabular data should be displayed in tables and form elements should be reasonably grouped. Your write-up should be well formatted with queries that are easy to read and commented if they are complex.

What to turn-in?

You need to turn in two files in Week 10. A PDF that contains all diagrams and written info, and a .zip that contains all of the source code for your project.

The queries (including the CREATE TABLE statements) **should be** one section in the PDF, otherwise you will lose some points (don't forget to use square brackets to act as place holders for variables that will be user provided). In the comment area of the submission form **include a link to your hosted, functioning website** (for most of you this will be web.engr.oregonstate.edu/~yourusername or the link to your Node.js server running on flip). The website should be accessible to all until the end of the last month of this term.

Frequently Asked Questions:

Q. I am confused. How many and on what all things should the website provide the various add, update and delete functionalities?

A. *Consider the **bsg_database (https://oregonstate.instructure.com/courses/1662151/pages/bsg-database?module_item_id=17798704)** (though it does not have the same number of entities and relationships as required). A website for it **must** have the functionality to add, remove and update instances of the many-to-many relationship between bsg_people and bsg_certs. Then, it is **required** that website provide ways to perform the same operations on the relation between bsg_planet and bsg_people (a one-to-many relationship). Further, it **should** also provide a way to insert new records* to *all the entities i.e. bsg_people, bsg_certs and bsg_planets. Finally, a user should be able to UPDATE and DELETE rows in **at least one** of these entities.*

Q. What does "Add and remove things from a relationship" mean?

A. *In a many-to-one relationship like bsg_people to bsg_planets, it means that you should be able to set the homeworld value to NULL on a person in bsg_people. That removes the relationship.*

It is a little more interesting in a many-to-many relationship because one would need to delete a row from a table. That would be the case with bsg_people and bsg_certifications. One should be able to add and remove certifications for a person without deleting either bsg_people rows or bsg_certification rows.

Q. Should the search/filter facility be for all the relationships and entities?

A. *At least for one entity.*

Final Notes:

1. Be sure to have a look at all discussions tagged "project" on Piazza.
2. For the frontend, you may use libraries/extensions like **handlebars.js (http://handlebarsjs.com/)** for templating the user interface.
3. Each student in the pair needs to have the application deployed.
4. This space will be updated with any major clarifications/changes.