Final Project (26% of total grade)

Purpose: In this final project, you will design, build, and document a MySQL database system. Your project will cover various database-related concepts and practices, including views, triggers, stored procedures, functions, normalization, data types, keys, and constraints. This project is worth 26% of your final grade.

Overview

Proposal - Part 1 (6%)

Due Week 12

A. Written, not coded
B. Database design, including database tools

Presentation / Documents - Part 2 (5% / 15%)

Due Week 14

A. SQL code that creates and demonstrates your database
B. 5-minute presentation in class
C. Supporting materials

Part 1: Proposal (6%)

Purpose: Your assignment is to propose a database architecture, (which you will later build, and present to the class). You will be creating a real world scenario that will use a database to solve real world problems with data. You will be the individual managing this database.

Name your file: HTTP5126-FinalProjectProposal-*LastNameFirstName*.pdf, replace *LastNameFirstName* with your name as displayed in Blackboard.

Submission: Make a copy of the proposal template provided. Follow the instructions in both this document and the proposal template. When completed, save your document as a PDF file, name your file as above. Upload and submit to Blackboard (Assessments > Final Project > Proposal).

Proposal Overview (24 marks)

Real World Scenario (2 marks)

What real world scenario are you putting yourself in as the person in charge of managing this database? “Real world” meaning human centered perspective, a situation real people could be placed in.

- Examples:
    - You are managing the data for a movie theater.
    - You are in charge of managing data at a pet store.
    - You are creating and managing a personal database for you and your friends.

Problems & Features (6 marks)

Describe 2 ways the people using your database may want to manipulate it. These will provide problems that will need to be solved with features. Choose a specific user group to focus on – for example, customers, or managers, or employees. What 2 real world human problems are these groups looking to have solved in your scenario? Describe the problem and the tables that are relevant to each problem. (2 marks; 1 per problem)

How can you use your database and SQL to solve the 2 real world problems you identified? Describe the features that would solve the problems. 1 feature should be the solution to 1 real-world problem. If there are 2 solutions needed for 1 problem, you may actually have 2 problems combined into 1. Conversely, the 2 features should not do the same thing in different ways. (4 marks, 2 for each proposed feature)

- These are examples of problems that need features to solve them:
    - When a movie joins or leaves the theater offerings how can the database easily update its tables?

    - When a sale is made how can the appropriate tables reflect the sale occurring?
    - When a friend wants to enter new data, how can you simplify the process?

Architecture Description (16 marks)

Outline the proposed architecture of your database. These should relate to your scenario, problems, and features.

Include the justification for the database architecture you created. In other words, state why each table, view, trigger, and function/procedure are necessary for the premise/features you have identified.

- Give your database a name (1 mark)
- Database schema with normalized tables (9 marks)
    - Column Names (1 mark)
    - Data types (1 mark)
    - Keys (1 mark)
    - Constraints (1 mark)
    - Relationships between tables (1 mark)
    - Normalization (2 marks)
    - Justification of table architecture (2 marks)
- Database tools (6 marks)
    - At least 1 view (1 mark)
    - At least 1 trigger (1 mark)
    - At least 1 function OR 1 procedure (1 mark)
    - Justification of tool architecture (3 marks; 1 per tool)

Notes

- The steps above do not need to be completed in order, they should inform each other and make sense when combined.
- Feel free to reuse the ideas from your lab 6/7, use a similar idea to your Pet Project from your JavaScript Frontend course, or come up with something completely new.
- If you decide to reuse your Lab 6/7 tables, you must ensure the scenario and problems make sense with the database you created previously. You will most likely need to alter those tables by adding to them or by adding new tables to appropriately solve the problems you define.
- Essentially, do not just squeeze a scenario, problems, and features into an old database, you are free to alter that schema so all the pieces described above make sense together.

Example Scenario

This is an example scenario of managing the data at an animal sanctuary. This is a list of problems that need features to solve them. There are many possible database features that could assist in the sanctuary's operation.

- If an animal joins or leaves the sanctuary, how can you make it easy to update all necessary tables?
- Assume that donors have the option to earmark their donations for certain types of animals. How can you see the total donated funds for each type of animal?
- Sleep schedules of certain animals could determine when exhibits are open. How can we generate a schedule of when exhibits are open?
- Dietary requirements of certain animals could determine a feeding schedule. How can we generate a feeding schedule?
- The animal sanctuary is open to the public within certain hours, excluding certain holidays. How can we generate a calendar that shows when the sanctuary is open?
- Employees must be scheduled for different shifts. How can we schedule different types of employees for different shifts, while making sure employees don't work more than 88 hours per two-week period?

Conventions Overview

Deductions [-0.25 marks per]

- General
    - Full words not abbreviations and acronyms
    - Avoid redundancy, do not prefix names with the name of their parent
    - Names should be meaningful and self-explanatory
    - db/table/column name should reflect their real world purpose
    - Names should be lowercase since SQL keywords are UPPERCASE
    - snake_case: underscore_in_place_of_spaces
- Databases
    - Singular name that describes information held in db
- Tables
    - Names should be nouns, 1 or 2 words
    - Table names may be singular OR plural *BUT BE CONSISTENT
- Columns
    - Names should be 1 or 2 words and singular

Convention Suggestions

- Views
    - Names should try not to exceed 4 words
    - Names should describe what the views purpose is
    - When possible include tables used in view in the view name
- Triggers
    - Prefix triggers with trg_
    - Trigger name typically in action then table name format
        - trg_<action>_<table_name>
        - trg_insert_movie
- Functions
    - Prefix functions with fn_
    - Function name typically in verb then noun format
        - fn_verb_noun
        - fn_get_title
- Procedures
    - Prefix procedures with usp_
    - Procedure name typically in verb then noun format
        - usp_verb_noun
        - usp_get_name