CSCE 156 – Assignment 4

Fall 2013

Introduction

In the previous phase of this project, you built an application framework that modeled the given problem and processed data files to produce a report. In this phase, you will design a data model that will support your application. You will do this by designing and implementing an SQL database.

Phase III – Database Design

You will design a (MySQL) database to store data that models the problem in the previous phase. You will need to design your tables to support data related to the entities that you identified and implemented in the previous phase as well as the relationship(s) between these entities.

The exact details (including naming conventions, column/field types, and relational design) are design decisions whose details are your responsibility. You must follow good database design principles. You should make your database flexible enough that records can be easily added/modified/removed without data integrity problems.

You should implement your database using a DDL (document data language) file and thoroughly test it using your MySQL account on cse. Included in your DDL you should include several queries to thoroughly test your database design as described below.

Design Process

While designing your database keep in mind the following:

- What tables are necessary to support the application you designed in the previous phase? In each table, what columns are necessary and what values should be allowed and/or excluded?
- How can data integrity be preserved? What primary, non-primary, and foreign keys should be defined?
- What are the relations between tables? What are the many-to-one or many-to-many relations? What opportunities are there for further data integrity and normalization?

Artifacts

Your database design will be realized by writing SQL script files which will contain all the necessary queries to create your database, insert some test data, and execute some queries to modify and remove data.

Database Schema File

You will hand in an SQL script (name it `portfolioDB.sql`) that contains all of the necessary SQL queries to create your database. Be sure that you do the following:

- Use conditional DROP TABLE IF EXISTS statements to clear out old versions of tables with the same name.
- Do *NOT* include a "USE database" statement: we will be running your DDL on our own database instance and will not have access to *your* database.
- After creating your tables, include all necessary insert statements (along with any nested select queries) to populate your database with some test data (this should be the same data as in the test cases used in the previous phase if you wish).

Database Query File

Provide, in a separate file (name it `portfolioQueries.sql`), queries to perform the following operations. This should be considered as a *bare minimum* of the testing that should be done. The script should operate from the cse command line.

1. A query to retrieve the major fields for every person
2. A query to retrieve the email(s) of a given person
3. A query to add an email to a specific person
4. A query to change the email address of a given email record
5. A query (or series of queries) to remove a given person record
6. A query to create a person record
7. A query to get all the assets in a particular portfolio
8. A query to get all the assets of a particular person
9. A query to create a new asset record
10. A query to create a new portfolio record
11. A query to associate a particular asset with a particular portfolio
12. A query to find the total number of portfolios owned by each person
13. A query to find the total number of portfolios managed by each person
14. A query to find the *total* value of all stocks in each portfolio (hint: you can take an aggregate of a mathematical expression)
15. A query to detect an invalid distribution of private investment assets (that is, a query to add up the stake percentage of each such asset and return a list of investments whose percentage exceeds 100%)

Clearly number these queries (using comments) in your SQL script and add any additional testing queries at the end.

Design Write-up

You will update and modify your Design Document draft that you started in the previous phase to include details on you database and data model design. An updated printed hardcopy will be handed in 1 week prior to the assignment due date.

In the new section, make sure to given enough details so that a technically competent individual would be able to reasonably reproduce an implementation of your design. Be sure to address *at least* the following.

- What tables and fields are in your database?
- What are its primary and non-primary keys?
- How are your tables related? What foreign key constraints did you use?
- How does your database design support the application and problem statement requirements?
- You are highly encouraged to include an Entity-Relation diagram for your database

Tips & Common Errors

- Make sure that your tables are *normalized* to a reasonable degree.
- When using MySQL, be sure to use `InnoDB` to ensure that foreign and primary key constraints are respected
- When using MySQL, use a case-sensitive character set to ensure correct data
- External identifiers should remain external. Alpha-numeric identifiers from sources "outside" the database (not generated or managed by the database) do not necessarily make good primary or foreign keys
- Keys should be integers, not floats nor VARCHAR; though VARCHAR columns may be made unique if appropriate
- Take care that you choose the correct data types for each of your columns
- Leverage the power of SQL queries to aggregate data—don't simply "flatten" the data and process it as in the previous phase
- Be careful with keys and relations between tables
- Be careful when defining nullable fields—what can/should be allowed to be null and what shouldn't?
- Your data model should be general—it should reasonably handle and model situations that may not necessarily be accounted for in your (or our) test cases