CSC 411 Course Programming Project

Option 3: Electronics Vendor

Due date: 11: 59 PM, April 25, 2022

Checkpoint	Task	Deadline
1	Project groups formation due date	February, 14
2	Web Application Development Learning (Chapter 9, Reference books Options 1~2)	February, 16
3	Project Website Design & Initial Implementation	March, 2
4	E-R Diagram	March, 19
5	Relation Schema	March, 29
6	DB Implementation and Queries	April, 12
7	Report	April, 25

Goal: The goal of this project is to provide a realistic experience in the conceptual design, logical design, implementation, operation, and maintenance of a relational database and associated applications. First, I shall describe the application, then the categories of requirements, and then some suggestions on how deeply you need to go in each category. A real project of this sort would require a substantial development team working for several months (or more). You will do this alone over several weeks. I have chosen to go with individual rather than group projects because the goal of this project is for you to gain a personal appreciation of the depth and breadth of issues that go into the design of a database application, rather than to have you specialize in just one aspect (and rely on others for the rest).

The project can go well beyond the minimal requirements I outline at the end. I encourage such extensions. They could turn into a senior design project or other independent work.

Application description: The application is an **electronics vendor** that operates both a Web site and a chain of many physical stores. Examples include Best Buy and Circuit City. To find out more about this application, think about any experiences you may have had making purchases both online and in-store, and browse their Web sites.

In our hypothetical company, it has been decided to redesign a major part of the database that underlies company operations. Unfortunately, the manager assigned to solicit database design proposals is not very computer literate and is unable to provide a very detailed specification at the technical level. Fortunately, you are able to do that.

Here are a few points to consider:

- There are many different products, grouped into a variety of (possibly overlapping) categories. Groupings can by type of product (cameras, phones, etc.), by manufacturer (Sony, Apple, etc.), or by other means (for example, a Gateway PC might be packaged with a Sony monitor and an HP printer and marketed as a package).

- Some customers have a contract with the company and bill their purchases to an account number. They are billed monthly. Other customers are infrequent customers and pay with a

credit or debit card. Card information may be stored for online customers, but not for in-store customers.

- Online sales must be sent to a shipper. The company needs to store the tracking number for the shipping company so it can respond to customer inquiries.

- Inventory must be accurate both in stores and in warehouses used to replenish stores and to ship to online customers. When inventory is low, a reorder should be sent to the manufacturer and listed in the database. When goods arrive, inventory should be updated and reorders marked as having been filled.

- Sales data are important for corporate planning. Marketers may want to look at sales data by time period, product, product grouping, season, region (for stores), etc.

Client Requests:

1. **E-R Model**

   - Construct an E-R diagram representing the conceptual design of the database.
   - Be sure to identify primary keys, relationship cardinalities, etc.

2. **Relational Model**

   - After creating an initial relational design from your E-R design, refine it based on the principles of relational design (Chapter 7).

   - Create the relations in a DBMS system, such as MySQL.

   - Create indices and constraints (optional).

   - If as you refine your design, you discover flaws in the E-R design, go back and change it (even if the earlier design passed the checkpoint.) Your final E-R design must be consistent with your relational design.

3. **Populate Relations**

   - Include enough data to make answers to your queries interesting and nontrivial for test purposes.

   - You may find it helpful to write a program to generate test data.

4. **Queries**: You should run a number of test queries to see that you have loaded your database in the way you intended. The queries listed below are those that your client (the manager from the package delivery company) wants turned in. They may provide further hints about database design, so think about them at the outset of the project.

- Assume the package shipped by USPS with tracking number 123456 is reported to have been destroyed in an accident. Find the contact information for the customer. Also, find the contents of that shipment and create a new shipment of replacement items.

- Find the customer who has bought the most (by price) in the past year.

- Find the top 2 products by dollar-amount sold in the past year.

- Find the top 2 products by unit sales in the past year.

- Find those products that are out-of-stock at every store in California.

- Find those packages that were not delivered within the promised time.

- Generate the bill for each customer for the past month.

5. **Interfaces**: There are several types of users who access the database. Each may need a special application

    - The database administrator (you) may use SQL either via the command line or SQL Developer.

    - Customer service needs a lookup application to check inventory both locally and at nearby stores.

    - Online customers need an elegant Web interface. **However, for this project, a command-line interface will still be acceptable (but will minus 10 points for this item) if your Web and/or GUI skills are not up to the challenge. (After all, this is a database course, not the Web Apps course, nor the User Interface course.)**

    - Call center staff need an application that allows quick access to customer data and the ability quickly to enter phone orders.

    - The stocking clerks at the warehouses need an application to help them record incoming shipments and update inventory.

    - The marketing department needs sales reports and may want to do special data mining and analysis. Because we do not cover these topics until late in the course, these features may be left to version 2, planned for the year after you graduate USM.

    These interfaces may, in some cases, use existing tools, in which case all you need to provide are instructions (e.g. run Query SQL in Oracle). In other cases, the target user of the interface is not someone we would expect to be fluent in SQL. Those interfaces can be built as

    - Web applications using Java applets or a scripting language.

- A standalone Java application using Swing to create a GUI

- Other GUI development tools you may know (but be sure they are platform independent, see note below)

- Since this course is not a Java course, nor a GUI course, I will accept a simple command-line interface. In fact, even if you are expert in GUI development, you may want to start with a simple command-line and then upgrade later.

6. **Concurrency**: The company stock will go down rapidly if the database cannot support more than one customer at a time making a purchase or if it cannot deal with more than one item being reordered. Be sure the MySQL/Other DBMS transaction mechanism is providing the needed guarantees. By running the various queries and applications in separate sessions (you can run multiple JBDC connections at once), you can simulate the real-life operation of your enterprise. Test concurrency carefully: don't fire up several processes that submit customer market baskets until you are sure things work (and don't scale this up to hundreds of customers or the system is likely to crash or bog down - it is only a P4 with 4GB of memory!)

What to turn in and how much I actually expect you to do:

The checkpoints are not graded. Usually, I find the fourth checkpoint requires some discussion, while checkpoint 5 can often be handled quickly. Checkpoint 6 is mainly in place to ensure that the projects are on schedule. Please talk to me about questions at any point; don't wait for a checkpoint.

The final version of the project is to be turned in as a single zip file on **Canvas**. I will accept paper or scanned PDF for the ER diagram since we are not covering drawing tools for these diagrams, but PowerPoint does work fairly well for this purpose.

1. E-R diagram, plus any explanatory notes. At minimum you must include all the entity and relationship sets implied by this handout. You may go beyond the minimum. Remember that the manager who defined the specifications is not computer literate so the specifications should not be viewed as necessarily being precise and complete.

2. Relational schema. It is likely for many of you that your ER design will be sufficiently extensive that we agree that only a part of the resulting relational design will actually be implemented. This is something on which we'll agree before Checkpoint 5. This is the point where we cut implementation effort and data-entry time to something realistic for the course time frame.

3. You may choose not turning in a listing of all your data which can be seen online. I will do if I find it necessary. In this case, I, as DBA, will have access to your database online and will use that if your submission leaves me with some unresolved questions. Please create a DBA account for that purpose. Otherwise, you may have to upload all of your data to Canvas as well.

4. A set of sample queries and the code you wrote for each of the listed queries, and the screenshot result from running the queries.

5. The code to implement the various interfaces. (Your code should be platform independent. I will accept quite basic interfaces (command line with a modest command set), but encourage more elegant interfaces. Depending on the degree of sophistication you plan for each interface, we can agree to fewer than the 7 interfaces requested by the client.

6. Please avoid platform-specific solutions. It is a bit hard for me to debug custom installations in the time-frame I have to grade the projects. Please check with me before making any design decisions that bind your application to a specific hardware or software platform. Platform-specific solutions will become a nightmare in the data-integration phase.

   If you use Java, please submit your java code as .java files that I can compile and run. Please do not submit them embedded in a NetBeans project or any other IDE. For other programming language, you also need to provide related instructions to guide me to compile and run your programs.

7. A README file in the top-level folder that explains what is where, etc. Include usage instructions for the interfaces.

8. Everything should be in a single zip file so that when I unzip it, I can read the README file, follow the directions, and run your project.

9. For group submission, one submission with a group report is enough. The group report needs to include a brief statement of individual contribution, i.e., which group member was responsible for which parts of the project and submitted material.

Grading:

I shall use the following approximate template for grading:

1. ER design: 20 points

2. Relational design (and constraints and indices, if applicable): 20 points

3. Data creation: sufficient quantity, reasonable realism, sufficiently “interesting”: 20 points

4. User interfaces, including proper features, proper updating of the database, etc.: 30 points

5. Concurrent operation of interfaces: 10 points

6. I reserve the right to give extra points for exception solutions to parts of the project.