Program #4: Database Design and Implementation

Due Date: May 4th, 2020, at 12:30 p.m. MST

Overview: Ordinarily, this assigned is a small–group project and includes a web interface. Due to our circumstances, I’ve cut it back to an individual assignment and dropped the web interface component. You’ll still get to do the rest: Design a DB for a specific application domain, create a relational schema based on your design, normalize your schema, implement it in Oracle, and write a Java program that allows data manipulation and querying.

Assignment: You are to do all of the following:

1. Design a Database Using E–R Modeling. Here is the description of the problem:

Dr. Denton’s Delightful Dentistry (4D) cares about your dental health! “Add 4D Teeth to Your 3D Vision!” is their advertising motto. Dr. Denton has a growing dental practice with multiple dentists, dental hygienists, and office staffers all in one large office. Naturally, the practice needs to keep track of basic information about its employees, its patients, the visits those patients make to the office, the dental procedures that are scheduled to be performed, the procedures actually performed and who performed them and when, and charges for the procedures and laboratory services.

Patients can make multiple visits. The office needs to keep track of those visits, what was done for the patient during each of them, and what those services cost the patient. Dr. Denton contracts with three laboratories for work that can’t be performed in the office, such as making bite guards, analyzing tissue samples, etc.

Every procedure and service has a base charge. Patients with insurance are billed 25% of the base charges, regardless of their insurance company.

This description does not describe every detail, just the essential framework. We expect that you will create an E–R Model diagram that incorporates all of these basics and adds others (e.g., patient attributes, relationships between entity sets, etc.) to make a complete design. We do not have a secret list of mystery expectations! Rather, we expect that you’ve been to a dentist regularly, and so have some idea how a dental office works and what dentists and hygienists do for patients. If you want information on dental care beyond your personal knowledge, consult Dr. Google. Note that the required queries (see Step 5(d)) specify some attributes that you’ll need to include in your design.

You may use any of the E–R Modeling diagramming schemes that we covered in this class. You are welcome to use an E–R diagramming tool, but a clear, neatly–hand–drawn diagram is fine, so long as you can incorporate it into the document (see the Hand In section, below).

2. Create a Relational Schema for Your Design. Convert your E–R Model to a collection of relations and attributes, with identified keys, relationships, etc.

3. Normalize your Schema to 3NF/BCNF. As described in the lecture videos and slides, the goal is BCNF, with 3NF being acceptable if BCNF is not achievable. If you were thoughtful during the previous steps, this step could be easy. If you instead designed a DB with odd collections of unrelated information, normalization could take a while.

(Continued...)
4. **Implement Your DB in Oracle.** Create and populate tables in Oracle via `lectura`, as you’ve done before. Creating your own example data is part of this step (and can be fun!). Again, feel free to consult outside sources for information you don’t know, such as the names of common dental procedures. You don’t have to have dozens of tuples in each table; rather, create enough that you can ask suitable queries and get interesting results from them.

**IMPORTANT:** Use the `GRANT` command to make all of your tables accessible by PUBLIC! You did this on Program #3; if necessary, review the information on the `GRANT` command provided there. (The `GRANT` command will also be covered in Topic 14.)

5. **Program a Java/JDBC Front–End.** This is the client’s user interface. Your program must:
   a. Run on `lectura` and communicate with Oracle
   b. Access your tables with the “<your netid>.” prefix so that the TAs can successfully test your code
   c. Allow users to insert and delete patients
   d. Allow users to ask the following questions:
      i. Who are the current employees and their titles?
      ii. Which patients have appointments on a given date?
      iii. For each patient with an upcoming appointment, which services are scheduled to be provided?
      iv. For a given patient, what was the bill for their last visit? The bill must include the patient’s name and contact info, whether or not they have insurance, a list of all services performed and their costs, and the total charge for the visit.
      v. Your Choice! Create your own query on your database. That’s it; no restrictions. (But we hope that you’ll implement something interesting that a user might want to know.)

You are welcome to add additional functionality, but you don’t need to do so!

**Hand In:** On or before the due date, submit via `turnin` on `lectura` to the folder `cs460p4` all of these items:

1. A single PDF document containing:
   a. Your final E–R diagram.
   b. Your final normalized relational schema, including your justifications that each of your relations are in at least 3NF. (Yes, that means you will need to include functional dependency information.)


**For More Information:**

- Know next to nothing about what dentists do? Here’s a starting point:
  

- Want to use realistic base costs for dental procedures?
  

**Other Requirements and Hints:**

- By department policy, no assignments can be due after the last day of classes. So, no matter how many late days you have left, you MUST turn this in by midnight on May 6. Note that submissions received after 12:30 p.m. on the 6th will be three days late.

- **Start Early!** Yes, you have two weeks, but there are several steps in this assignment, it’s the end of the semester when lots of big projects are due, and we’re working remotely. Recommendation: Pretend that the assignment is due three days earlier, and set your own intermediate sub–due–dates for each of the steps.