Overview: In this assignment, you will build a database-driven information management system from ground up. We will give you an application domain to work on, your goal is to design the underlying database and define the application functionalities you will provide with the database, and implement this application using Oracle within a text-based JDBC program.

Assignment: In this assignment you are to implement a two-tier client-server architecture.

1. Database Back-End, which runs the Oracle DBMS on aloeh.cs.arizona.edu. Your job is to design the database relational schema, create tables and populate your tables with some initial data. We are requiring that you create an E–E–R diagram, analyze the FDs of each table and apply table normalization techniques to your schema to justify that your schema satisfies 3NF, and, if possible, BCNF.

2. JDBC Front-End, which is the client’s user interface. You need to design a text-based application that appropriately handles all the required functionalities. Your client application will run on lectura.

Application Domain: The problem description for the project is as follows:

Campus Health is the home of healthcare for the University of Arizona’s students. When they have any health needs such as general medicine, testing or immunizations, campus health is the place to go. Your task is to design a database and an associated manipulation/querying application for Campus Health.

While Campus Health provides multiple services, we will focus on 4 core ones: General Medicine, Counseling And Psychological Services (CAPS), Laboratory & Testing, and Immunization. To get any service you can either book an appointment or you can walk-in. If you’re a student on the student health insurance plan, you will be billed a certain rate. If you’re a student without any insurance, you will be charged at the student discounted rate. If you’re an employee then you get charged according to the mandatory employee insurance plan (even if the employee is also a student).

Campus health has different medical staff for different kinds of services. If an appointment is not canceled one hour before it is due to start, and the patient doesn’t show up, a no-show fee is billed to the patient’s bursar account (assume everyone has a bursar account). Patients also have an option to use the walk-in clinic. For those who opt for the walk-in clinic and have a severe illness, they will be seen immediately in the emergency ward. For those who walk-in and do not have a severe illness they will be asked to wait until the appropriate staff is available or will be asked to schedule an appointment for a later date.

Your E–R diagram must include at least one E–E–R concept.

(Continued...)
This description does not describe every detail. These are the essentials; we expect that your team will create logical and conceptual designs that incorporate all of these features, at minimum. You are free to add additional details that you feel are appropriate.

For each table you create, you need to populate a reasonable number of tuples in order to test your queries adequately. Some data basics are provided in the application domain description; the rest are left for you to determine, based on the scenario and your application’s needs. (What is ‘reasonable’ is difficult to define; a few dozen tuples per relation certainly would be; just a handful per relation may not provide sufficient variety.)

We realize that you are not an expert in this domain, but you have dealt with similar organizations in your life. Hopefully, you have enough experience that this problem description makes sense. If you have questions, please ask, and the TAs will help you clear things up.

Required functionalities: Within the framework provided above, your system is expected to perform examples of the following operations:

1. **Record insertion**: Your application should support inserting a new data record via a JDBC interface.
2. **Record deletion**: Your application should support deleting an existing data record via a JDBC interface.
3. **Record update**: Your application should support updating an existing data record via a JDBC interface.
4. **Queries**: Your application should support querying your database via a JDBC interface for the problem description given above. You are required to implement the four provided queries as well as at least one query of your own design. Details are provided below.

Specifically, the JDBC application’s interface should enable users to:

1. Add, update or delete a patient, employee, and appointment. When updating, the user is allowed to update everything except the IDs of the patients, employees, and appointments. When deleting, the entire row needs to be deleted. While inserting or updating an appointment, appointment time must not overlap with any existing appointment.
2. Update different service details of a staff member.
3. Schedule an immunization. If the immunization is for COVID-19, current availability rules apply. Specifically, after the 3rd dose (a.k.a. the first booster dose), if the person is 50 or over, there must be a date for the 4th dose scheduled (to encourage the best protection for the campus community). If the person is under the age of 50, their 4th dose date must be NULL (people under 50 are not allowed to schedule 4th doses at this time).

Here are the queries that your application is to be able to answer:

1. Print a list of patients who have their 2nd, 3rd or 4th doses of the COVID-19 vaccine scheduled by a certain date (given that the date is entered by the user).
2. Given a certain date, output which patients had a non–walk–in appointment. Sort in order by appointment time and group by type of service.
3. Print the schedule of staff given a certain date (input by the user). A schedule contains the list of staff members working that day (including those who were working that day as usual and those who were working to handle an appointment) and a staff member’s working hours (start and stop times).
4. Print the vaccine statistics of the two categories of patients (student, employees). The statistics include the count of patients that have completed all 4 doses of a vaccine series, the count of patients that have received three doses, but not the 4th, the count of patients that have received two doses but not the 3rd, and the count of patients who have only received the first dose.
5. A query of your choice, subject to these restrictions: The question must use more than two relations and must be constructed using at least one piece of information gathered from the user.

(Continued...)

**Working in Groups:** In industry, such a project is usually the work of multiple developers, because it involves several different components. Good communication is a vital key to the success of the project. This assignment provides an opportunity for just this sort of teamwork. Therefore, we are accepting team sizes of between two and four members (inclusive). Working alone is not permitted.

Early on, you will need to agree on a reasonable workload distribution plan for your team, with well-defined responsibilities, deliverables, and expected completion dates. Such a plan will minimize conflicts and debugging effort in the actual implementation.

**Late days:** Late days can be used on this assignment, but only on the third due date. How many a team has to use is determined as follows: Team members total their remaining late days, and divide by the number of members in the team (integer division), producing the number of late days the team has available, **to a max of two days late.** (Why a max of two? The TAs need to get grading done soon after the due date, you need time to study for your final exams, and the department has a rule about assignments needing to be due before the start of finals.) For example, a team whose three members have 1, 1, and 3 late days remaining have \(\left\lfloor \frac{1+1+3}{3} \right\rfloor = 1\) late day to use, if needed.

**Hand In:** Here are the ‘deliverables’ for each of the assignment’s three due dates:

1. **Team Composition:** By the first due date (see the top of the front page of this handout), one member of your team must create a PRIVATE post on Piazza using the “program4” folder with the names and NetIDs of the members of your team. Failure to do so by the start of class on this date will cost your team the corresponding points listed in the Grading Criteria section (below).

2. **E–E–R Diagram:** As stated in the Assignment section, your team will need to create an E–E–R diagram that describes your database design. Before the second due date, your team will need to prepare a draft of your E–E–R diagram and a member of your team will need to submit it through turnin to the cs460p4 folder. The purpose of this requirement is to allow the TAs to review your schema and make suggestions for improvement. The sooner you create your design and discuss it with the TAs, the more time you will have to refine your final E–E–R diagram. If TAs need further explanation of your E–E–R Diagram, they’ll send out an email to make an appointment to have an additional meeting.

3. **Final Product:** On or before the third due date, a member of your team must submit a .tar file of your well-documented application program file(s) via turnin to the folder cs460p4. The tar file should contain all of the following:
   (a) The source code for your application.
   (b) A PDF file called “design.pdf” containing the following sections in this order:
      i. **Conceptual database design:** Your final E–E–R diagram along with your design rationale and any necessary high-level text description of the data model (e.g., constraints, or anything you were not able to show in the E–E–R diagram but that is necessary to help people understand your database design).
      ii. **Logical database design:** The conversion of your E–E–R schema into a relational database schema. Provide the schemas of the tables resulting from this step.
      iii. **Normalization analysis:** For each of your entity sets (tables), provide all of the FDs of the table and justify why your the table adheres to 3NF / BCNF.
      iv. **Query description:** Describe your self-designed query. Specifically, what question is it answering, and what is the utility of including such a query in the system?
   (c) A ReadMe.txt describing:
      i. Compilation and execution instructions, to enable the TAs to execute your application and exercise the required functionalities.
      ii. The workload distribution among team members (that is, which people were responsible for which parts of the project).

   **In addition,** each team must schedule a time slot (~20 minutes) to meet with a TA, demonstrate your system, and answer questions about it. Closer to the first due date, we’ll let you know how to sign up.

(Continued...)
**Grading Criteria:** Total: 100 points

1. Team Composition (1st due date): 5


3. Final Submission (3rd due date): 75
   
   (a) Coding / Implementation: 55
   
   - Documentation 15
   - Style and organization 10
   - Record insertion: 5
   - Record deletion: 5
   - Record update: 10
   - Record query: 10

   (b) Database design: 20
   
   - Final E–E–R diagram: 10
   - Normalization analysis: 10

**Grading Notes:**

1. Unless we receive verifiable complaints about inadequate contributions, each member of a team will receive the same score on this assignment.

2. We won’t put much weight at all on the appearance of the text application; concern yourselves with the application's functionality instead. The main points of the assignment are the DB design and its implementation.