Emergency Room Management System (ERMS)

CS4400: Introduction to Database Systems
Course Project: Fall 2025 Semester

Project Purpose

In this project, you will analyze, specify, design, implement, and document an online system based on the provided scenario description. You are required to use the classical methodology for relational database development. The system will be implemented using a relational DBMS that supports standard SQL queries. You will use your localhost MySQL Server (Version 8.0 or above) to implement your database and the application. You cannot use any other software like Access or SQLite. Please ask the Instructors and TAs if you have questions.

Project Phases

	Phase I	Phase II	Phase III
We provide you with...	• Scenario Description • Sample Data	• Solution EERD • Initial Data Set (in a Non-Normalized Format)	• Physical Database Schema with Data Sets Inserted • View & Stored Procedure Outlines
You turn in to us... (+ CATME ratings due after each phase)	• Enhanced Entity Relationship Diagram (EERD)	• Relational Schema • Physical Database Schema with Data Sets Inserted • Unhandled constraints	• Implemented Views & Stored Procedures

Phase II Directions

In Phase II, your tasks are to:

- Translate the **provided** EERD (NOT your Phase I submission) into a **relational schema**
- Translate the relational schema into **create table statements** with the appropriate data types, primary and unique keys, update/delete behavior, check constraints, and foreign keys
- Write **insert statements** to import the **sample data & denormalized data** into the tables you've created
- Document all **unhandled constraints** based on your design & implementation choices

Relational Schema [35%]

Convert the Enhanced Entity-Relationship Diagram (EERD) that we've provided into a relational schema. Identify primary keys and foreign keys using the text-based notation introduced in class and covered in the slides.

**** **DO NOT USE YOUR EERD FROM PHASE I.** We require you to **use the provided EERD** for this assignment as the focus is on making sure that you understand and can apply the conversion process correctly. This also ensures you won't be double penalized for mistakes in Phase I.*

Update and Delete Behavior [5%]

Determine the update and delete behavior for each of the foreign keys specified in the relational schema that you have developed. Each decision should be justified based on how it would impact the functionality of the database and reflect the information provided in the scenario description. Here is an example of how this would look with an example relation from the Global Company Database schema:

- works_on (essn [fk5], pno [fk6], hours)
    - fk5: essn → employee (ssn)
        - **On update restrict** – the ssn of an employee should not change. If there is an attempt to change it, then this will help prevent it from occurring
        - **On delete cascade** – if an employee is fired or leaves the company, we no longer want to store their information in our system, including how long they spent working on projects
    - fk6: pno → project (pnumber)
        - **On update cascade** – the work an employee puts into a project should be tracked even if the project number changes
        - **On delete restrict** – we need to keep record of the hours an employee has worked, even on projects that are no longer being contributed to

Note: There can be multiple valid solutions for update and delete behavior. What's important is that you select one that you can properly justify.

Create Table Statements [25%]

Building off the SQL template provided in Canvas files, develop the MySQL CREATE TABLE statements, including domain constraints, integrity constraints, primary keys and uniqueness constraints, foreign key constraints, and check constraints. You will also need to specify ON UPDATE and ON DELETE clauses that you defined as well.

Insert Statements [25%]

Before deciding to use a database to track patient, staff, appointment, and order information, the emergency room used a very disorganized method to record information. Half of the data was written down in a text format [the **Sample Data in the Scenario Description**] and the other half was recorded in a spreadsheet [in **Canvas files**]. The ER needs to combine the data from both data sources to insert into the final tables for this database.

Building off your CREATE TABLE statements, you must insert all information from the provided **initial data spreadsheet file** *and* **sample data from the scenario description**. You should write the MySQL INSERT statements in the same MySQL file as the CREATE TABLE statements.

The data is provided in a "non-normalized" manner, meaning that it is not structured in a way that can naturally inserted into your final table structures. You will have to sift through and manipulate the data to get it into a format that is usable. This is a very real problem that occurs in industry due to how data gets logged in software. Obviously, this is a miniscule amount of data compared to what a real emergency room would have.

Unhandled Constraints [10%]

List all the constraints expressed in the written requirements and EERD that are **not** supported by your CREATE TABLE statements. If a constraint is already being handled by one of the following components, then you should NOT list it as an unhandled constraint:

- check constraint declaration
- data type declaration
- primary key declaration
- foreign key declaration, including ON UPDATE/DELETE behaviors
- uniqueness declaration
- not null declaration
- default value declaration

The relational model is powerful, but it is not powerful enough to express all constraints. The intent of this portion of the assignment is for you to determine the other constraints that will need to be accounted for as you continue to develop the system. You do NOT need to provide solutions for managing the unhandled constraints that you've identified – you only must acknowledge them in a clear and concise manner (**maximum of 2 sentences per constraint**). You will implement solutions for these constraints in the last phase of the project.

How should you list your unhandled constraints? Here are some possible examples from the Global Company database, using its relational schema as reference:

- Ensure that each department has at least one employee that works for that department.
  - Why? There is no field in department tracking how many employees work for that department. The Employee entity only has information about the department each employee works for, and not the other way around (because it is 1:N).
- Ensure that employee works on at least one project.
  - Why? The works_on table is separated from the Employee table because it is an N:M relationship. This means we do not know if an employee is assigned a project only from the Employee table.
- Ensure that each project has at least one employee working on that project.
  - Why? The works_on table is separated from the Project table because it is an N:M relationship. This means we do not know if a project has any employees assigned from the Project table only.

Here are some other possible constraints that aren't required by the company database example from class, but they are reasonably realistic and will give you an idea of other possibilities:

- Ensure that an employee doesn't supervise (directly) more than seven other employees.
  - Why? An Employee has information about their one manager in their table entry, but a manager does not have a list of the employees they supervise in their row entry.
- Ensure that no supervisee makes more (i.e., has a higher salary) than their supervisor.
  - Why? The Employee table only has the supervisor_ssn and not supervisor_salary too.
- Ensure that the manager for a department also works for that department.
  - Why? The Department table only has the manager_ssn and not the manager_deptID too.

Submission Checklist

Each team needs **one of its members** to upload the deliverables to Gradescope. When submitting, ensure that **all members are added to the submission**. Failure to do so will result in a **2-point penalty** to their overall project score. The other team members should log in to their individual Gradescope accounts and check to ensure that all files have been uploaded correctly and that they are linked to their team's submission. Include your team number in every file name.

Your submission must include the following **three distinct files (do not zip them together)**:

- A file named **cs4400_phase2_schema_team#.pdf** containing your relational schema and the update and delete behaviors, with justifications, for each foreign key
- A file named **cs4400_phase2_constraints_team#.pdf** containing your unhandled constraints
- A file named **cs4400_phase2_database_team#.sql** containing your create table and insert statements
    - **You must submit the original MySQL statements that you've hand typed, NOT the SQL dump/export file. The file format MUST be ".sql".**
    - **Your SQL file must run in MySQL Workbench without error for you to receive credit for these statements.**

Version History

Version	**Date**	**Notes**
0	September 15, 2025	Initial Release