In this application, I implemented Spring Data JPA repositories to manage data access efficiently within the Hospital Information System. The main goal was to leverage the repository pattern, which offers a streamlined way to interact with the database, making the application more modular and easier to maintain.

To start, I defined a series of Spring Data JPA repositories for the key entities such as Employee , Doctor , Nurse , Patient , Department , and Ward . Each of these repositories extends JpaRepository , which provides built-in methods for handling basic CRUD operations. For example, the save() method was used to add new records to the database or update existing ones. I used **findById()** and **findAll()** to retrieve specific records or all records of a particular type. To delete records, I utilized the **deleteById()** method. This setup ensured that basic data manipulation tasks were straightforward and required minimal boilerplate code.

Beyond basic CRUD operations, I also created custom query methods using Spring Data JPA's method naming conventions. These methods enabled more specific data retrieval operations that matched the needs of the application. For instance, I implemented a method like findBySpecialty(String specialty) in the DoctorRepository to find doctors based on their specialty. Similarly, in the PatientRepository , I included a method like findByWard_WardNumber(int wardNumber) to retrieve patients assigned to a specific ward. These custom queries allowed the application to fetch targeted data sets efficiently without writing explicit SQL queries, making the code more readable and easier to maintain.

Overall, integrating Spring Data repositories into my application significantly enhanced data management by providing a structured approach to database interaction, ensuring that the application remains scalable and easy to modify as new features or requirements emerge.

Proposed ERD diag