INFSCI 2710 Database Management, Fall 2024

Consider a Hospital Management System. This system will manage patients, doctors, nurses, treatments, rooms and billing information:

1. Patient
    - Each patient has a unique patient ID, name, date of birth, gender, contact details, and admission date.
    - Each patient can be assigned to a room and may receive one or more treatments.
    - Each patient will have a billing record based on treatments and room charges
2. Doctors
    - Each doctor has a unique doctor ID, name, specialization, years of experience, and contact details.
    - A doctor can attend to multiple patients, and each patient can be attended by multiple doctors (many-to-many relationship).
3. Nurses
    - Each nurse has a unique nurse ID, name, and shift details.
    - Nurses are assigned to rooms and assist patients.
4. Rooms
    - Each room has a unique room ID, type (e.g., ICU, general), capacity, and availability status.
    - Patients are assigned to rooms, and each room may house multiple patients (up to its capacity).
5. Treatments
    - Each treatment has a unique treatment ID, treatment name, and cost.
    - Patients receive treatments, which are administered by doctors.
6. Billing
    - Billing is associated with each patient and includes the treatment cost and room charges.
    - Each billing record has a unique bill ID, patient ID, total amount, and billing date.

Q1 [5 pt] What are the main entities and their attributes in this system? Identify primary keys for each entity.

SOLUTION:

Main Entities and Their Attributes

1. **Patient**
    - Attributes:
        - Patient ID (Primary Key)
        - Name
        - Date of Birth
        - Gender
        - Contact Details
        - Admission Date
2. **Doctor**
    - Attributes:
        - Doctor ID (Primary Key)
        - Name
        - Specialization
        - Years of Experience
        - Contact Details
3. **Nurse**
    - Attributes:
        - Nurse ID (Primary Key)
        - Name
        - Shift Details
4. **Room**
    - Attributes:
        - Room ID (Primary Key)
        - Type (e.g., ICU, General)
        - Capacity
        - Availability Status
5. **Treatment**
    - Attributes:
        - Treatment ID (Primary Key)
        - Treatment Name
        - Cost
6. **Billing**
    - Attributes:
        - Bill ID (Primary Key)
        - Patient ID (Foreign Key)
        - Total Amount
        - Billing Date

Q2 [5 pt] How would you represent the many-to-many relationship between patients and doctors in the database?

SOLUTION:

To represent the many-to-many relationship between **Patients** and **Doctors**, a **relation table** is required. This table will capture the association between the two entities.

Table: Attend

- **Attributes**:
  - Patient ID (Foreign Key referencing the **Patient** table)
  - Doctor ID (Foreign Key referencing the **Doctor** table)
  - Interaction Date (optional, to record when a patient was attended by a doctor)

Primary Key:

- A composite primary key consisting of **Patient ID** and **Doctor ID** ensures the uniqueness of each relationship.

Q3 [5 pt] How would you enforce room capacity limits in your schema design?

SOLUTION:

Patient

- Attributes:
  - Patient ID (Primary Key)
  - Name
  - Date of Birth
  - Gender
  - Contact Details
  - Admission Date

Room

- Attributes:
  - Room ID (Primary Key)
  - Type (e.g., ICU, General)
  - Capacity
  - Availability Status

Relation between Patient and Room can be made through creating a table called **Assign_Room** table which contains the primary keys of **Patient** table and **Room** table.

Assign_Room

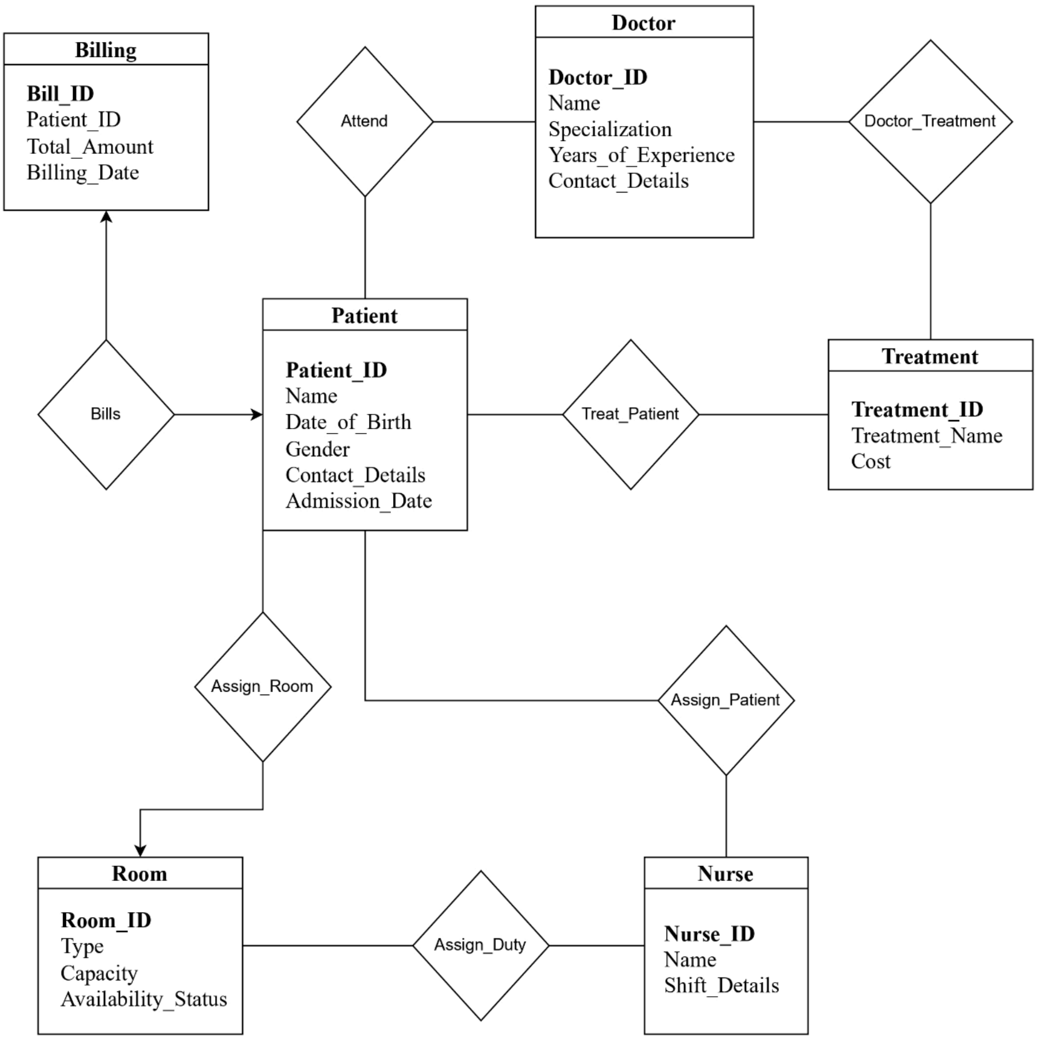
- Attributes:
  - Patient ID (Foreign Key referencing the **Patient** table)
  - Room ID (Foreign Key referencing the **Room** table)

Constraint:

Using Triggers:

```
Create Trigger enforce_room_capacity

Before Insert On Assign_Room

For Each Row

Declare

   current_capacity INT

   room_capacity INT

Begin

   Select Count(*)

   Into current_capacity

   From Assign_Room RA

   Where RA.Room_ID = :NEW.Room_ID;

   Select Capacity

   Into room_capacity

   From Room

   Where Room_ID = :NEW.Room_ID;

   If current_capactiy >= room_capacity Then

    Raise_Application_Error(-20001, 'Room Capacity Exceeded');

   End If;

End;
```

Q4 [10 pt] Draw an ER diagram for the system above



Q5 [20 pt] Translate the ER diagram from Q4 into SQL DDL statements create table queries

```sql
CREATE TABLE Patient (

  Patient_ID INT PRIMARY KEY,

  Name VARCHAR(100) NOT NULL,

  Date_of_Birth DATE NOT NULL,

  Gender VARCHAR(20) NOT NULL,

  Contact_Details VARCHAR(255),

  Admission_Date DATE NOT NULL

);


CREATE TABLE Doctor (

  Doctor_ID INT PRIMARY KEY,

  Name VARCHAR(100) NOT NULL,

  Specialization VARCHAR(100),

  Years_of_Experience INT,

  Contact_Details VARCHAR(255)

);


CREATE TABLE Treatment (

  Treatment_ID INT PRIMARY KEY,

  Treatment_Name VARCHAR(100) NOT NULL,

  Cost DECIMAL(10, 2) NOT NULL

);
```

```sql
CREATE TABLE Room (

  Room_ID INT PRIMARY KEY,

  Type VARCHAR(20) NOT NULL,

  Capacity INT NOT NULL,

  Availability_Status BOOLEAN NOT NULL

);


CREATE TABLE Nurse (

  Nurse_ID INT PRIMARY KEY,

  Name VARCHAR(100) NOT NULL,

  Shift_Details VARCHAR(100)

);


CREATE TABLE Billing (

  Bill_ID INT PRIMARY KEY,

  Patient_ID INT,

  Total_Amount DECIMAL(10, 2) NOT NULL,

  Billing_Date DATE NOT NULL,

  FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)

);
```

```sql
CREATE TABLE Attend (

  Patient_ID INT NOT NULL,

  Doctor_ID INT NOT NULL,

  PRIMARY KEY (Patient_ID, Doctor_ID),

  FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),

  FOREIGN KEY (Doctor_ID) REFERENCES Doctor(Doctor_ID)

);


CREATE TABLE Assign_Duty (

  Nurse_ID INT NOT NULL,

  Room_ID INT NOT NULL,

  PRIMARY KEY (Nurse_ID, Room_ID),

  FOREIGN KEY (Nurse_ID) REFERENCES Nurse(Nurse_ID),

  FOREIGN KEY (Room_ID) REFERENCES Room(Room_ID)

);


CREATE TABLE Assign_Room (

  Patient_ID INT NOT NULL,

  Room_ID INT NOT NULL,

  PRIMARY KEY (Patient_ID, Room_ID),

  FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),

  FOREIGN KEY (Room_ID) REFERENCES Room(Room_ID)

);
```

```sql
CREATE TABLE Assign_Patient (

  Patient_ID INT NOT NULL,

  Nurse_ID INT NOT NULL,

  PRIMARY KEY (Patient_ID, Nurse_ID),

  FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),

  FOREIGN KEY (Nurse_ID) REFERENCES Nurse(Nurse_ID)

);


CREATE TABLE Treat_Patient (

  Patient_ID INT NOT NULL,

  Treatment_ID INT NOT NULL,

  PRIMARY KEY (Patient_ID, Treatment_ID),

  FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),

  FOREIGN KEY (Treatment_ID) REFERENCES
Treatment(Treatment_ID)

);

CREATE TABLE Doctor_Treatment (

  Doctor_ID INT NOT NULL,

  Treatment_ID INT NOT NULL,

  PRIMARY KEY (Doctor_ID, Treatment_ID),

  FOREIGN KEY (Doctor_ID) REFERENCES Doctor(Doctor_ID),

  FOREIGN KEY (Treatment_ID) REFERENCES
Treatment(Treatment_ID)

);
```

Q6 [15 pts] Consider Table (a) which shows part of the records in relation R. Complete the table (b) for given functional dependencies (FD). Please just answer yes, no or unknown

Solutions:

FD	**Satisfied on given records (yes/ no/unknown)**	**Hold on R (yes/no/unknown)**	**Trivial (yes/no)**
A→ B	No	No	No
B→ A	No	No	No
AC→ D	Yes	Unknown	No
AC→B	No	No	No
AD→ B	No	No	No
ABC→ AC	Yes	Yes	Yes
BC→ D	Yes	Unknown	No
C→ AC	No	No	No
AB→ A	Yes	Yes	Yes
BD→ A	No	No	No

Q7 [10 pts] Consider a relation R1(A,B,C,D,E,F,G) and a set of functional dependencies FD = {AB → E, C → D, DE → F, DA → G} which hold on R1. Using Armstrong's axioms verify if the following functional dependencies hold on R1

FD	Yes/No	Proof if yes
AC → G	Yes	A → A (By Reflexivity) C → D (Given) AC → AD (By Union) AD → G (Given) AC → G (By Transitivity)
AC→ F	No	-
ABD→F	Yes	AB → E (Given) ABD → ED (By Augmenting D) DE → F (Given) ABD → F (By Transitivity)
BCD→ F	No	-
ABC→EG	Yes	AB → E (Given) ABC → AB (By Reflexivity) ABC → E (By Transitivity)

		ABC → C (By Reflexivity) C → D (Given) ABC → D (By Transitivity) ABC → DE (By Union) DE → F (Given) ABC → F (By Transitivity) DA → G (Given) ABC → D (Derived from above) ABC → DA (By Augmenting A) ABC → G (By Transitivity) ABC → E (Derived from above) ABC → EG (By Union)

Q8 [10 pts] Consider a relation R(A,B,C,D,E,F) and a set of functional dependencies, which hold on R : { CD->BE, B->CA, D->F} Are decompositions in the table lossless and why?

Decomposition	Lossless? (Yes) / (No)	Why
R1(ABF) and R2(CDE)	No	R1 ∩ R2 - None Since the intersection of R1 and R2 is none, the decomposition is not lossless.
R1(ABCEF) and R2(CDE)	Yes	R1 ∩ R2 - CE **To Prove:** CE → ABCEF CE → CDE **Proof: CE → ABCEF** B → CA (Given) D → F (Given) BD → ACF (By Union) CBD → ACF (By Augmenting C) BE → ACF (By Given) E → E (By Reflexivity) BE → ACEF B → C (Gives) CE → ACEF BCE → ABCEF (By Augmenting B) **CE → ABCEF** Hence Proved **CE acts as a candidate key for R1** **Proof: CE → CDE**

		This is trivial for R2 and **CE acts as a candidate key for R2 as well**. Therefore this is a **lossless decomposition.**
R1(ABDE) and R2(BDF)	Yes	R1 ∩ R2 - BD **To Prove:** BD → ABDE BD → BDF **Proof: BD → BDF** D → D (By Reflexivity) B → CA (Given) BD → ACD (By Union) CD → BE (Given) BD → ABE (Substituted) D → D (By Reflexivity) BD → ABDE (By Union) Hence Proved **BD acts as the candidate key for R1** **Proof: BD → BDF** D → F (Given) BD → BF (By Augmenting B) D → D (By Reflexivity) BD → BDF (By Union) Hence Proved **BD acts as the candidate key for R2** Therefore this is a **lossless decomposition.**
R1(ACDF) and R2(BCDE)	Yes	R1 ∩ R2 - CD **To Prove:** CD → ACDF CD → BCDE **Proof: CD → ACDF** CD → BE (Given) CD → CD (By Reflexivity) CD → BCDE (By Union) CD → B (Gives) B → CA (Given) CD → CA (By Transitivity) CD → ABCDE (By Union) D → F (Given) CD → ABCDEF (Substituted) **CD → ACDF**

		Hence Proved **CD acts as the candidate key for R1** **Proof: CD → BCDE** CD → BE (Given) CD → CD (By Reflexivity) CD → BCDE (By Union) **CD → BCDE** Hence Proved **CD acts as the candidate key for R2** Therefore this is a **lossless decomposition.**
R1(ABEF) and R2(BCDF)	Yes	R1 ∩ R2 - BF **To Prove:** BF → ABEF BF → BCDF **Proof: BF → ABEF** B → CA (Given) BD → ACD (By Augmenting D) BD → ABCD (By Reflexivity) D → F (Given) BF → ABCF (Substituted) CD → E (Gives) CF → BE (Gives) BF → ABE (Substituted) BF → ABCF BF → ABCEF (By Union) **BF → ABEF** Hence Proved **BF acts as the candidate key for R1** **Proof: BF → BCDF** B → CA (Given) BD → ACD (By Augmenting D) BD → ABCD (By Reflexivity) D → F (Given) BD → ABCF (Substituted) BD → ABCDF BD → BCDF **BF → BCDF** Hence Proved **BF acts as the candidate key for R2** Therefore this is a **lossless decomposition.**

Q9 [10 pts] Consider the following relations with the associated functional dependencies. Decide, whether those relations are in (a) BCNF, (b)3NF, (c) neither in BCNF nor 3NF normal form.

Relation, FD	Answer (a, b, or c)	Reason
R1(A,B,C,D) {AB→ C, C→ D}	**(c)**	**Keys:** AB → AB (By Reflexivity) AB → C (Given) C → D (Given) AB → D (By Transitivity) AB → ABCD (By Union) **AB is the candidate key** **F1:** (AB)Left part is candidate key so it is good for BCNF **F2:** C is not the candidate key so in order to check for 3NF. Verify right part (D) which is not the subset of the keys generated. **Therefore R1 is neither BCNF nor 3NF.**
R2(A,B,C,D), {AC→ BD}	**(a)**	**Keys:** AC → AC (By Reflexivity) AC → BD (Given) AC → ABCD (By Union) **AC is the candidate key** **F1:** (AC) Left part is candidate key so it is good for BCNF **Therefore R2 is in BCNF.**
R3(A,B,C,D) {AB→ CD, D→A}	**(b)**	**Keys:** AB → AB (By Reflexivity) AB → CD (Given) AB → ABCD (By Union) **AB is the candidate key** D → A (Given)

		D → AD (By Union) BD → ABD (By Augmenting B) BD → ABCD (By derived) **BD is the candidate key** **F1:** (AB) Left part is candidate key so it is good for BCNF **F2:** (D) is not the candidate key so in order to check for 3NF. Verify right part (A) which is the subset of the keys generated. So, it is good for 3NF. **Therefore R3 is in 3NF.**
R4(A,B,C,D,E), {AC→D, D→B}	**(c)**	**Keys:** AC → AC (By Reflexivity) AC → D (Given) AC → ACD (By Union) AC → B (By Transitivity) AC → ABCD (By Union) ACE → ABCDE (By Augmenting E) **ACE is the candidate key** **F1:** (AC) is not the candidate key so in order to check for 3NF. Verify right part (D) which is not the subset of the keys generated. **F2:** D is not the candidate key so in order to check for 3NF. Verify right part (B) which is not the subset of the keys generated. **Therefore R4 is neither BCNF nor 3NF.**
R5(A,B,C,D,E) {CD→ ABE, B→ C}	**(b)**	**Keys:** CD → CD (By Reflexivity) CD → ABE (Given) CD → ABCDE (By Union) **CD is the candidate key** B → C (Given)

		$B \rightarrow B$ $B \rightarrow BC$ (By Union) $BD \rightarrow BCD$ (By Augmenting D) $BD \rightarrow ABCDE$ (By derived) **BD is the candidate key** **F1:** (CD) Left part is candidate key so it is good for BCNF **F2:** (B) is not the candidate key so in order to check for 3NF. Verify right part (C) which is the subset of the keys generated. So, it is good for 3NF. **Therefore R5 is in 3NF.**

Q10 [10pts] Draw a valid B+ tree below for the search keys (1, 2, 3, 4, …, 12). Assume the keys are inserted in their natural order. The order of the tree is 3