Miniworld: Hospital Database

GROUP 1: Poorva Pisal (2020113001), Sneha Raghava Raju (2020101125), Yug Dedhia (2020115004)

Introduction

In this project, we are going to make a generic database of a hospital. The hospital's database should contain all the details required for the hospital's official use and the hospital's employees as the end-users, along with user restrictions.

Purpose

The purpose of this mini-world is to create a database for a hospital. This database allows the hospital to add, organize and update data effectively. This ensures the smooth running of all the hospital functions concerning the patients, equipment, inventory, rooms, and the hospital's finances.

Users

The users of this database are hospital staff, doctors, nurses, and administration that will enter patient information, keep records of all the employees, and store information about the different departments.

Applications

- To access/update information about patients, inventory, and rooms available.
- To store data related to the current employees, including the doctors, nurses, and other hospital staff.
- To store and retrieve information about the different departments and services provided by the hospital.

Database Requirements

Entities:

1. **Patient**
    - Attribute 1 - Patient ID (primary key) (ID uniquely corresponds to a person)

      Data Type: Integer
      Constraints: Unique positive integer that corresponds to a certain patient.

    - Attribute 2 - Name (Composite attribute)

      Attributes contained:
      1. First name
      2. Middle name
      3. Last name

      Data Type: String
      Constraints: Non-empty and should contain only letters.

    - Attribute 3 - Details (primary key)(Composite attribute)

      Attributes contained:
      1. Phone Number (Multivalued attribute)
         Data Type: Long
         Constraints: Unique 10 digit positive integer that corresponds to a certain patient.

      2. Address
         Data Type: String
         Constraints: Should contain only letters.

      3. DOB
         Data Type: Date
         Constraints: Should be a valid date

      4. Age - Derived attribute
         Data Type: Integer
         Constraints: Depends on validity of date of birth

2. **Services**
    - Attribute 1 - Service ID (Primary Key - A specific integer that identifies a Service).

      Data Type: Integer

Constraints: The Service ID should be a unique positive integer that corresponds to an existing service that is provided by the hospital.

- Attribute 2 - Service Name (Name of a the service)

  Data Type: String
  Constraints: Non-empty and should contain only letters.

- Attribute 3 - Cost (Cost of the Service)

  Data type: Float
  Constraints: A positive float.

- Attribute 4 - Description (Information about the service)

  Data Type: String (text)

3. Doctors

- Attribute 1 - Doctor's ID (Primary Key)
  Data type: Integer
  Constraints: The Doctor's ID should be a valid ID that identifies the doctor in the hospital. This should be a unique positive integer.

- Attribute 2 - Name
  Data type: String
  Constraints: The name should only contain alphabets.

- Attribute 3 - Specialization (i.e. type of doctor) (Multivalued attribute)
  Data type: String
  Constraints: Non-empty and should contain only letters.

- Attribute 4 - Department (linked to Department; Department ID)
  Data type: String
  Constraint: It should be a valid department and should exist in the Department table.

- Attribute 5 - Details (Composite attribute)

  Attributes contained:
  1. Phone Number - Multivalued attribute
     Data Type: Long
     Constraints: Unique 10 digit positive integer that corresponds to a certain patient.

2. Address
   Data Type: String
   Constraints: Should contain only letters.

3. Salary
   Data Type: Long
   Constraints: Should be a positive number

4. Nurses

- Attribute 1 - Nurse's ID (Primary Key)
  Data type: Integer
  Constraints: The Nurse's ID should be a valid ID that identifies the nurse in the hospital. This should be a unique positive integer.

- Attribute 2 - Name
  Data type: String
  Constraints: The name should only contain alphabets.

- Attribute 3 - Department (linked to Department; Department ID)
  Data type: String
  Constraint: It should be a valid department and should exist in the department table.

- Attribute 4 - Details (Composite attribute)

  Attributes contained:
  1. Phone Number - Multivalued attribute
     Data Type: Long
     Constraints: Unique 10 digit positive integer that corresponds to a certain patient.

  2. Address
     Data Type: String
     Constraints: Should contain only letters.

  3. Salary
     Data Type: Long
     Constraints: Should be a positive number

5. Staff

- Attribute 1 - Staff ID (Sub-class identifier)
  Data type: Integer

Constraint: The staff ID should be a valid ID that identifies the staff in the hospital. This should be a unique positive integer.

- Attribute 2 - Name
  Data type: String
  Constraints: The name should only contain alphabets.

- Attribute 3 - Details (Composite attribute)

  Attributes contained:
  1. Phone Number (Multivalued attribute)
     Data Type: Long
     Constraints: Unique 10 digit positive integer that corresponds to a certain patient.

  2. Address
     Data Type: String
     Constraints: Non-empty and containing alphanumeric characters.

  3. Salary
     Data Type: Long
     Constraints: Should be a positive number

- Subclasses (Departments):
  1. Administrative
     - Department
  2. Maintenance
     - Area (Floor/Rooms they are assigned to)
  3. Security
     - Area (Location they are assigned to)
  4. Transport
     - Vehicle Type
     - Licence & Registration

6. Inventory (Weak Entity)

- Attribute 1 - Department ID (Foriegn Key - From ‘Departments’ table.)
  Data Type: Integer
  Constraints: Unique positive integer that corresponds to a certain department.

- Attribute 2 - Item name
  Data Type: String
  Constraints: Unique positive integer that corresponds to a certain department.

- Attribute 3 - Current count
  Data Type: Integer
  Constraints: Positive integer

- Attribute 4 - Needed count
  Data Type: Integer
  Constraints: Positive integer

- Attribute 5 - Count to be ordered (derived attribute)
  Data Type: Integer
  Constraints: Positive integer

- Attribute 6 - Price
  Data Type: Float
  Constraints: A positive float.

7. Departments

- Attribute 1 - Department ID (Primary key)
  Data Type: Integer
  Constraint: The department ID should be a valid ID that identifies the department in the hospital. This should be a unique positive integer.

- Attribute 2 - Name
  Data type: String
  Constraint: Should be a valid department name.

- Attribute 3 - Location
  Data Type: String (Floor number, Room number etc.)
  Constraints: Should contain only letters.

8. Appointments

- Attribute 1 - Appointment ID (Primary Key)
  Data Type: Integer
  Constraints: Unique positive integer that corresponds to a certain appointment.

- Attribute 2 - Patients ID (Foreign Key - from the Patient Table)
  Data Type: Integer
  Constraints: Unique positive integer that corresponds to a certain patient.

- Attribute 3 - Doctors ID (Foriegn Key - from Doctor table)
  Data Type: Integer
  Constraints: Unique positive integer that corresponds to a certain doctor.

- Attribute 4 - Nurses ID (Foreign key - from Nurses table)
  Data Type: Integer
  Constraints: Unique positive integer that corresponds to a certain nurse.

- Attribute 5 - Services (Foreign key - from Services table)
  Data Type: Integer
  Constraints: Unique positive integer that corresponds to a certain service.

- Attribute 6 - Date
  Data Type: Date
  Constraints: Constraints: Should be a valid date

- Attribute 7 - Time (i.e start time)
  Data Type: Timestamp
  Constraints: A valid time during working hours.

- Attribute 8 - Room type
  Data Type: Integer
  Constraints: Null for no room / any positive int for a type.

9. Illness (Weak Entity)

- Attribute 1 - Patient ID (Primary Key)
  Data type: Integer
  Constraints: The Patient ID should be a valid ID that identifies the patient in the hospital. This should be a unique positive integer.

- Attribute 2 - Name of illness
  Data type: String
  Constraints: Should contain only letters.

- Attribute 3 - Medicine
  Data type: String
  Constraints: Should contain only alphanumeric characters.

Weak Entity:

A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities. It is dependent on a strong entity to ensure its existence. Unlike a strong entity, a weak entity does not have a key attribute.

1. **Inventory** is a weak entity as its existence is dependent on the existence of the **Departments** entity. If a department exists then only will the inventory for it exist.
2. **Illness** is a weak entity as its existence is dependent on the existence of the **Patient** entity. If a patient exists then only will the details for their illness exist.

Relationships:

1. **Relationship 1**
    - Degree - Binary
    - Participating Entities - Patient and Appointments
    - Cardinality ratio - 1:N (where N is the number appointments)
2. **Relationship 2**
    - Degree - Binary
    - Entities - Patient and Illness
    - Cardinality ratio - 1:N
3. **Relationship 3**
    - Degree - Binary
    - Entities - Doctors and Departments
    - Cardinality ratio - 1:1
4. **Relationship 4**
    - Degree - Binary
    - Entities - Nurses and Departments
    - Cardinality ratio - 1:1
5. **Relationship 5**

    Subclass Relation: In a HOSPITAL database SUPERVISION relationship between STAFF (role -administrative) and STAFF(role- maintenance)

    - Degree - Binary
    - Subclasses - Administrative and Maintenance
    - Cardinality ratio - 1:N (one admin manages N maintenance staff members)
6. **Relationship 6**
    - Degree - Binary
    - Entities - Department and Inventory
    - Cardinality ratio - 1:N (one department has N inventory)
7. **Relationship 7**
    - Degree - Five-Degree (all five can access each other)
    - Entities - Appointment, Patient, Doctor, Nurse, Services
    - Cardinality:
        1. Nurse -> Appointment (1:N)
        2. Patient -> Appointment (1:N)
        3. Doctor -> Appointment (1:N)
        4. Service -> Appointment (1:N)

Bonus

1. Considering the Patient entity, the following are the types of keys:
    - Super Keys:

      {Patient ID}

      {Details}

      {Patient ID, Details}

      {Patient ID, Name}

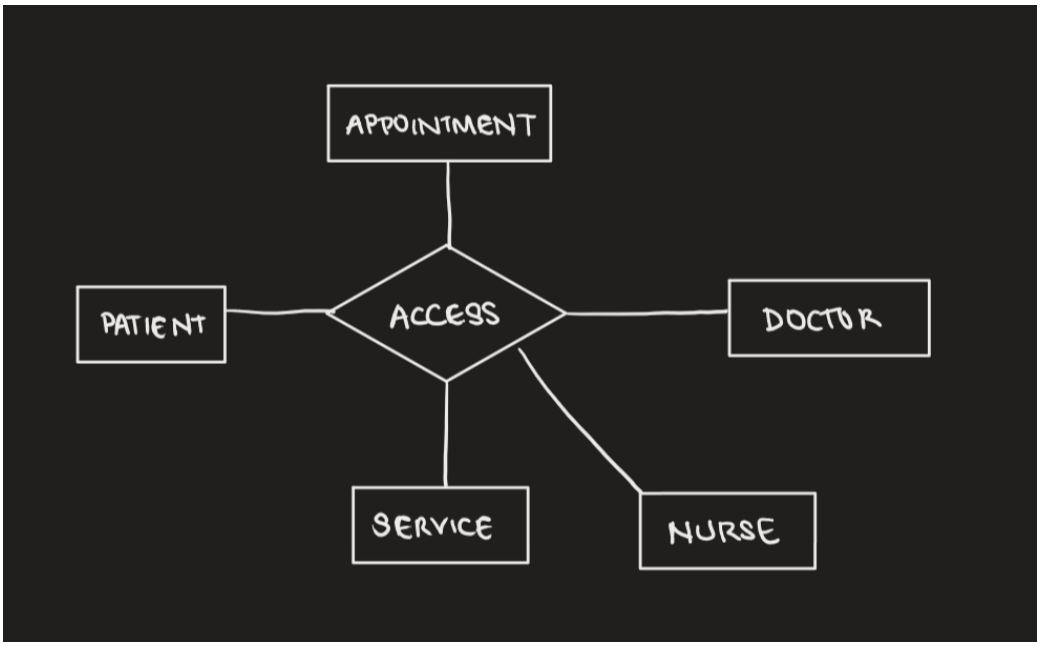
      {Patient ID, Details, Name}

      {Details, Name}

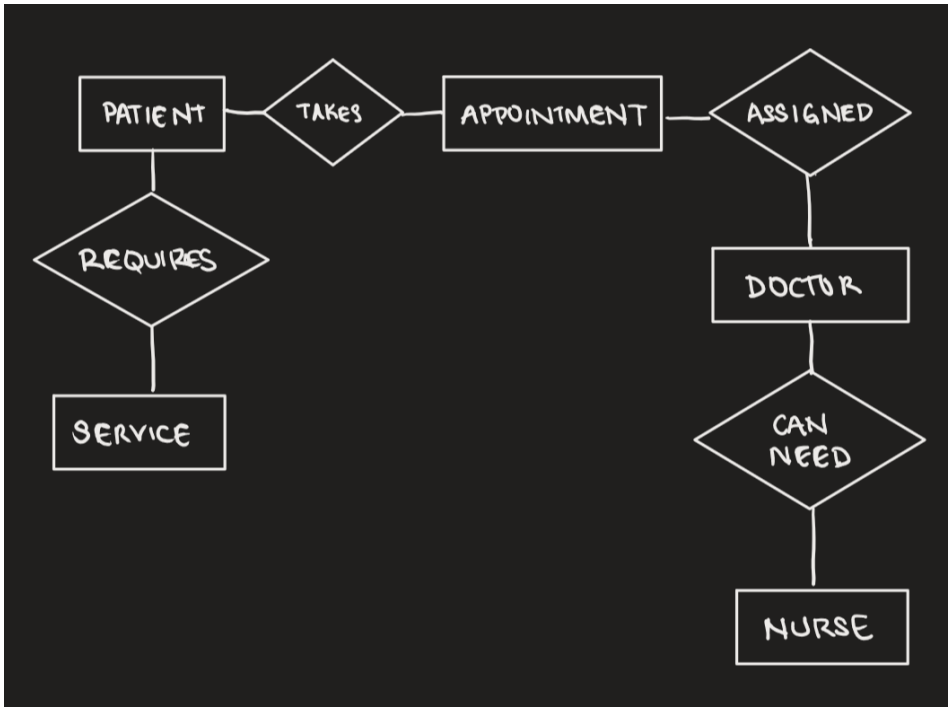
    - Candidate Keys: Patient ID, Details (Both of these can uniquely identify a row. Name cannot, because two people can have the same name.) and the rest of the super keys are redundant.
    - Alternate Keys: If we choose Patient ID to be the Primary Key then Details will become the Alternate Key and vice versa.

2. Considering the 5-degree relationship of Appointments the following is the breakup into 5 binary relationships:

   5 degree relation:



5 binary relations:



3. In a HOSPITAL database SUPERVISION relationship between STAFF (role -administrative) and STAFF (role- maintenance) in the Staff entity is an example for a relation type with the same participating entity type in distinct roles.

Functional Requirements

Modifications:

1. **Insert:**
    - Addition of details of a new patient or a new illness.
    - Addition of details of a new employee falling under doctors/nurses/staff.
    - Addition of a new appointment.
    - Addition of new services or new items in the inventory.

    Integrity Constraints: The primary key of Patient, Doctor/Nurses/Staff, Appointment and Services should be unique. When adding a new appointment, the foreign key (eg: Doctor ID, Nurse ID) should either be NULL or should be a value that exists in the primary key of the respective table.

2. **Delete:**
    - Deleting the details of a staff member/doctor/nurse who has resigned or has been dismissed.
    - Deleting a cancelled appointment.
    - Deleting discontinued services.
3. **Update:**
    - Updating details of a patient or their illnesses.
    - Updating details of doctors/nurses/staff members.
    - Updating the details of inventory (eg.: count of items)
    - Updating details of a service.
    - Updating details of a department (eg.: change of location)

Retrievals:

1. **Selection:**
    - Retrieving data of all the doctors in a specific department
    - Retrieving data of all the nurses in a specific department.

    Eg.: SELECT * FROM Doctors WHERE DEPARTMENT=$dep

2. **Projection:**
    - All patients whose age > n (where n is a threshold age)
    - All doctors of a certain specialization
    - All nurses of a certain department

Eg.: SELECT * FROM PATIENTS WHERE AGE> $n

3. **Aggregate:**
    - Finding the average age of the patients in the hospital..
    - Finding the total salary (sum of all employees' salary) of a department

    Eg: SELECT AVG(Age) AS AverageAge FROM Patients;

4. **Search:**
    - Searching for details of a patient of a particular birth year.
    - Searching for details of a patient using some keywords.
    - Searching for details of a particular department using some keywords.

    Eg.: SELECT * FROM PATIENTS WHERE YEAR= $Year

5. **Analysis:**
    - The number of Appointments each Doctor had.
    - The number of Patients that took a particular Service.
    - The total number of nurses/doctors working in a certain department.