VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

DATABASE SYSTEM (CO2014)

Assignment 1

HOSPITAL MANAGEMENT SYSTEM

Advisor: Do Thanh Thai
Students: Dinh Ba Khanh - 2252323
Nguyen Minh Khoi - 2252377
Tran Nguyen Anh Khoi - 2252383

HO CHI MINH CITY, OCTOBER 2024

Contents

1 Member list & Workload

No.	Fullname	Student ID	Task	Contribution
1	Dinh Ba Khanh	2252323	found entity types and their attributes, sketched ERD, did mapping, chosen technologies, made report	100%
2	Nguyen Minh Khoi	2252377	found entity types and their attributes, sketched ERD, did mapping, wrote MySQL statements, made report	100%
3	Tran Nguyen Anh Khoi	2252383	found entity types and their attributes, idenified constraints, sketched ERD, did mapping, made report	100%

2 Introduction

Hospitals play a vital role in our lives by offering the best medical care to individuals dealing with various illnesses, which may arise from factors such as changing climate conditions, heavy workloads, emotional stress, and more. Managing the daily operations and records of a hospital manually can be extremely challenging. Therefore, a database system is essential to efficiently store and manage all hospital activities and records.

3 Entities and Relationships

3.1 Strong Entities and Relationships

3.1.1 Strong Entities and their attributes

There are strong entities:

Medical Staff: represents for who work in hospital. There are medical staff attributes:

staffID: ID of staff.

staffSSN: social security number of staff.

staffName: name of the staff. It also includes *firstName, midName, lastName.*

staffDoB: date of birth of the staff.

gender: gender of the staff.

phoneNumber: phone number of the staff.

salary: salary of the staff.

Doctor: represents for the doctor who work in the hospital. There are doctor attributes:

license: license of the doctor.

specialization: specialization of the doctor.

Nurse: represents for the nurse who work in the hospital. There are nurse attributes:

yearExperience: the amount of year that the nurse has worked.

Patient: represents for the patient who go to the hospital for treatment. There are patient attributes:

patientID: the ID of the patient.

patientName: the name of the patient. It includes *firstName, midName, lastName.*

patientDoB: the date of birth of the patient.

address: patient's address. It includes *street, district, city*

gender: patient's gender.

patientSSN: social security number of patient.

Department: represents for the hospital departments such as Emergency, Cardiology, General Surgery, ... There are department attributes:

departmentID: ID of the department.

departmentName: name of the department.

Pharmacy: represents for the pharmacy of the hospital where patients purchase medicines based on doctor's presciption.

pharmacyID: ID of pharmacy.

pharmacyName: name of pharmacy.

Room: represents for the room in the hospital. There are room attributes:

roomID: ID of room.

roomType: type of room.

capacity: the capacity of the room.

numPatient: the number of patient in each room.

Bill: represents for the bill in the hospital. There are bill attributes:

billID: the ID of the bill

amount: the amount that customer has to pay.

createdDate: the date that this bill was printed.

paymentStatus: the status of payment.

paidDate: the date that this bill was paid.

Medicine: represents for patient's medicines. There are medicine attributes:

medicineID: the ID of each medicine.

medicineName: name of the medicine.

dosage: dose of medicine that patient has to take.

Treatment: the treatment method that the doctor proposes. There are treatment attributes:

treatmentID: the ID of the treatment.

treatmentDate: the date that it starts.

treatmentProcedure: the method of treatment.

Appointment: represents for the appointment of the patient and doctor.

appointmentID: the ID of the appointment.

appointmentDate: the date of the appointment.

appointmentTime: the time of the appointment.

Payment Approach: represents for the payment method.

Cash: represents for paying in cash.

cashierID: the ID of payment.

cashierName: name of the cashier that is responsible for the payment.

Bank: represents for paying through bank.

bankID: the ID of payment.

bankName: name of the bank that the customer uses.

Ewallet: represents for paying through Ewallet.

ewalletID: the ID of payment.

ewalletName: the name of ewallet that customer uses.

3.1.2 Relationships and their attributes

There are relationships:

Works for: This is a many-to-one (N:1) relationship where each medical staff entity can work for one department but a department can have multiple medical staff members.

Manages: This is a one-to-one (1:1) relationship where a doctor can manage a Department. Each department can be managed by only one doctor, but not all doctor will have a managerial role.

startDate: date when the doctor starts managing the department

Takes care: This is many-to-many (N:M) relationship where nurses can take care of multiple patients, and a patient can be taken care of by multiple nurses.

Assists: This is many-to-many (N:M) relationship where nurses can assist in multiple treatments, and each treatment can be assisted by multiple nurses.

Performs: This is many-to-many (N:M) relationship where doctors can perform multiple treatments, and each treatment can be performed by multiple doctors.

Undergoes: This is many-to-many (N:M) relationship where patients can undergo multiple treatments, and each treatment can involve multiple patients.

Recorded: This is many-to-one (N:1) relationship where each treatment is recorded in one Medical Record, but a record can include multiple treatments.

Belongs to: This is many-to-one (N:1) relationship where Medical Record belongs to one patient, but a patient can have multiple medical records.

Schedules: This is many-to-many (M:N) relationship where doctors can schedule multiple appointments, and each appointment may involve multiple doctors.

Attends: This is many-to-many (M:N) relationship where patients can attend multiple appointments, and each appointment can involve multiple patients.

Prescribes: This is many-to-many-to-many relationship where a doctor prescribes certain medicines to a patient. Each doctor can prescribe multiple medicines to multiple patients, and each patient can receive prescriptions for multiple medicines from multiple doctors. This complex relationship links doctors, patients, and medicines, reflecting the prescription process within the hospital system.

prescribesDate: date of the prescription written

Received at: This is many-to-many (M:N) relationship where medicines can be received at multiple pharmacies, and each pharmacy can carry multiple medicines.

Admitted to: This is many-to-many (M:N) relationship where patients can be admitted to multiple rooms at different times, and a room can accommodate multiple patients.

admittedDate:: the date patient admitted the room.

dischargedDate: the date patient discharge the room.

Pays: This is one-to-many (1:N) relationship where a patient can pay multiple bills, but each bill is associated with one patient.

Processed by: This is many-to-one (N:1) relationship where each bill is processed by one payment approach, but a payment method can be used for multiple bills.

3.2 Weak Entities

Patient's Family: represents for family's member. This weak entity associates with strong entity **Patient** through relationship **Family of**, this is many-to-one relationship where each patient's family entity is connected to one patient, but a patient can have multiple family members listed. The **Patient's Family** entity depends on the **Patient** entity for its existence and identification, meaning it cannot exist without being linked to a specific patient.

familyID: the ID of the family, dependent on the associated patient.

name: name of family member.

relationship: the relationship with the patient.

phoneNumber: family member's phone number.

Medical Record: represents for patient's record. This weak entity associates with strong entity **Patient** through relationship **Belongs to**, this is many-to-one relationship where each medical record is associated with one patient, but a patient can have multiple medical records over time. The **Medical Record** is dependent on the **Patient** because it records information specific to the individual's treatments, and it cannot exist without being tied to a patient. There are record attributes:

recordID: the ID of the record, may be partially reliant on the "patientID" for unique identification.

recordDate: the date medical record is created.

diagnosis: the diagnosis of the patient

testResult: the result after test.

4 Entity Relationship Diagram (ERD)

This is the ERD of the project or we can access here

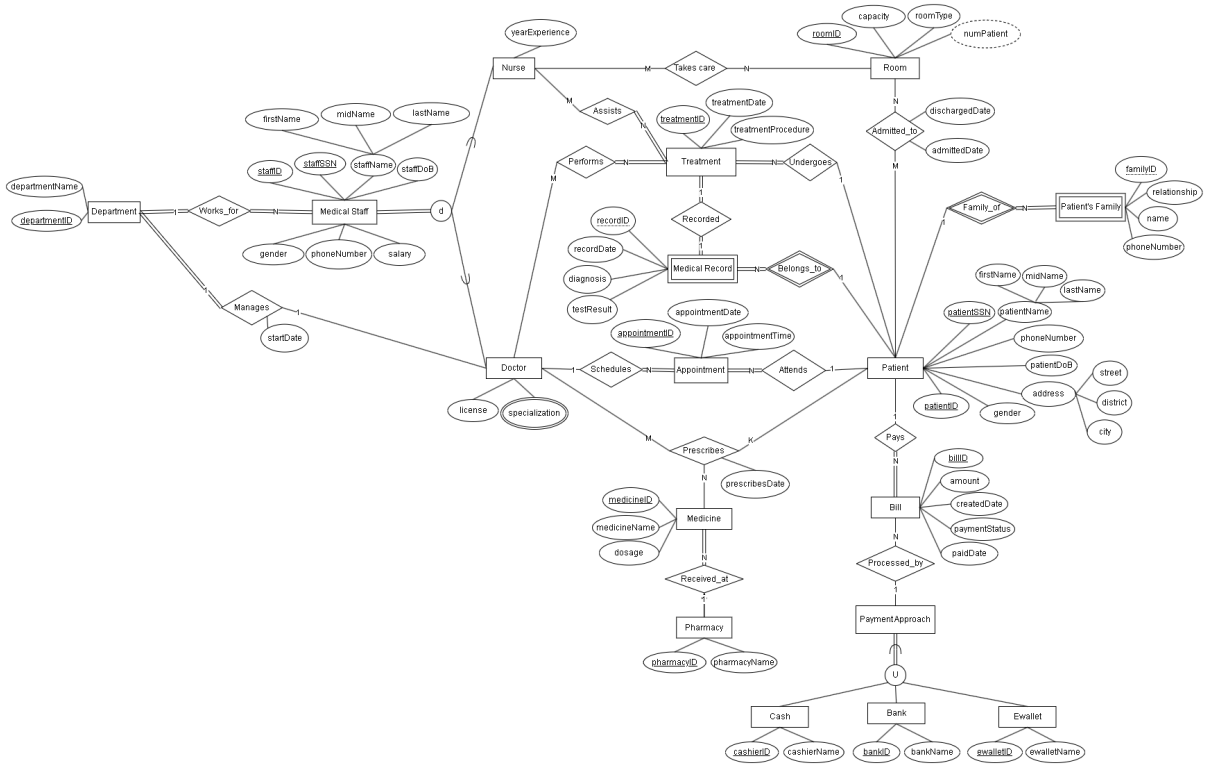


Figure 1: Entity Relationship Diagram

5 Relational database schema

5.1 Data Model Mapping

Department (departmentID, departmentName)

Primary key: departmentID

Medical_staff (staffID, staffSSN, firstName, midName, lastName, staffDoB, gender, phoneNumber, salary, departmentID)

Primary key: staffID

Secondary key (unique, not null): staffSSN

Foreign key: departmentID to Department.departmentID

Check total participation of Department.departmentID in Medical_Staff

Doctor (doctorID, license)

Primary key: doctorID

Foreign key: doctorID to Medical_Staff.staffID

Specialization (doctorID, aSpecialization)

Primary key: doctorID, aSpecialization

Foreign key: doctorID to Doctor.doctorID

Nurse (nurseID, yearExperience)

Primary key: nurseID

Foreign key: nurseID to Medical_Staff.staffID

Check the disjoint constraint between Doctor and Nurse

Check total participation of Medical_Staff.staffID in Doctor and Nurse

Manages (departmentID, doctorID, startDate)

Primary key: departmentID

Unique key: doctorID

Foreign key: departmentID to Department.departmentID, doctorID to Doctor.doctorID

Not null: doctorID, startDate

Room (roomID, capacity, roomType)

Primary key: roomID

Derived attribute: numPatient

Patient (patientID, patientSSN, firstName, midName, lastName, patientDoB, gender, phoneNumber, street, district, city)

Primary key: patientID

Secondary key (unique, not null): patientSSN

Admitted_to (patientID, roomID, admittedDate, dischargedDate)

Primary key: patientID, roomID

Foreign key: patientID to Patient.patientID, roomID to Room.roomID

Appointment (appointmentID, appointmentDate, appointmentTime, doctorID, patientID)

Primary key: appointmentID

Foreign key: doctorID to Doctor.doctorID, patientID to Patient.patientID

Not null: doctorID, patientID

Pharmacy (pharmacyID, pharmacyName)

Primary key: pharmacyID

Medicine (medicineID, medicineName, dosage, pharmacyID)

Primary key: medicineID

Foreign key: pharmacyID to Pharmacy.pharmacyID

Not null: pharmacyID

Payment_Approach (PAID)

Primary key: PAID (surrogate key)

Cash (cashierID, cashierName, PAID)

Primary key: cashierID

Foreign key: PAID to Payment_Approach.PAID

Bank (bankID, bankName, PAID)

Primary key: bankID

Foreign key: PAID to Payment_Approach.PAID

Ewallet (ewalletID, ewalletName, PAID)

Primary key: ewalletID

Foreign key: PAID to Payment_Approach.PAID

Check total participation of Payment_Approach.PAID in Cash, Bank and Ewallet

Bill (billID, patientID, amount, createdDate, paymentStatus, paidDate, PAID)

Primary key: billID

Foreign key: patientID to Patient.patientID, PAID to Payment_Approach.PAID

Not null: patientID, PAID

Treatment (treatmentID, treatmentDate, treatmentProcedure, patientID)

Primary key: treatmentID

Foreign key: patientID to Patient.patientID

Not null: patientID

Medical_Record (recordID, patientID, recordDate, diagnosis, testResult, treatmentID)

Primary key: recordID, patientID

Foreign key: patientID to Patient.patientID, treatmentID to Treatment.treatmentID

Not null: patientID, treatmentID

Check total participation of Treatment.treatmentID in Medical_Record

Performs (doctorID, treatmentID)

Primary key: doctorID, treatmentID

Foreign key: doctorID to Doctor.doctorID, treatmentID to Treatment.treatmentID

Check total participation of Treatment.treatmentID in Performs

Assists (nurseID, treatmentID)

Primary key: nurseID, treatmentID

Foreign key: nurseID to Nurse.nurseID, treatmentID to Treatment.treatmentID

Check total participation of Treatment.treatmentID in Assists

Takes_care (nurseID, roomID)

Primary key: nurseID, roomID

Foreign key: nurseID to Nurse.nurseID, roomID to Room.roomID

Prescribes (medicineID, patientID,doctorID, prescribesDate)

Primary key: medicineID, patientID, doctorID

Foreign key: doctorID to Doctor.doctorID, patientID to Patient.patientID, medicineID to Medicine.medicineID

Patient's_Family (familyID, patientID, relationship, name, phoneNumber)

Primary key: familyID, patientID

Foreign key: patientID to Patient.patientID

5.2 Data Model Mapping Diagram

This is relational mapping or we can access here

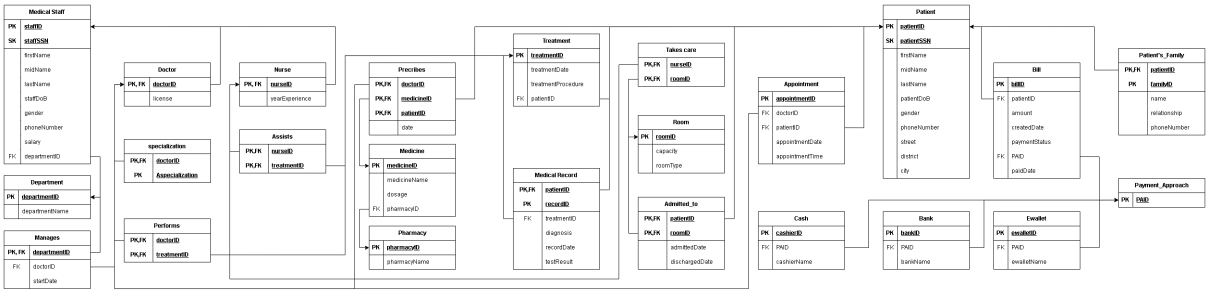


Figure 2: Relational Mapping

5.3 Constraints not shown

- A doctor cannot be scheduled to more than one appointment concurrently
- A patient cannot attend more than one appointment at the same time
- A doctor can only perform one treatment at a period of time
- A nurse can only assists in one treatment at a period of time
- The salary and bill amount can not be negative
- The created date must be earlier than or equal to the paid date.
- A nurse should be assigned to work on at most two adjacent rooms only
- A patient can only be admitted to one room at a time
- The admission date must be earlier than or equal to the discharge date
- Any room with admitted patients must have at least one assigned nurse for patient care
- The number of patients admitted to a room must not exceed the room's capacity

6 Defining User Groups and Permissions

User Group	Role	Permissions
Admin	Manage entire system	Has full access to everything in the database. This includes the ability to create, read, update, and delete, as well as manage user accounts and their roles.
Doctor	Handle patient treatment	Create, read, update, and delete treatments, prescriptions, appointments and medical records, . They can also view information about patients, other doctors, nurses, rooms, admitted patients, and medicines.
Nurse	Support patient care	View and update patient information, manage room details, and access treatment records. They also have the ability to view information on admitted patients and prescriptions.
Receptionist	Assist patients with completing necessary forms and documentation	Responsible for creating, viewing, updating, and deleting patient records and family information. They can also manage appointments, track admitted patients, and handle billing information.
Patient	Access personal health information	Can view their own medical records, appointments, prescriptions, family information, and billing details. This is done through dedicated views that display only their personal information.

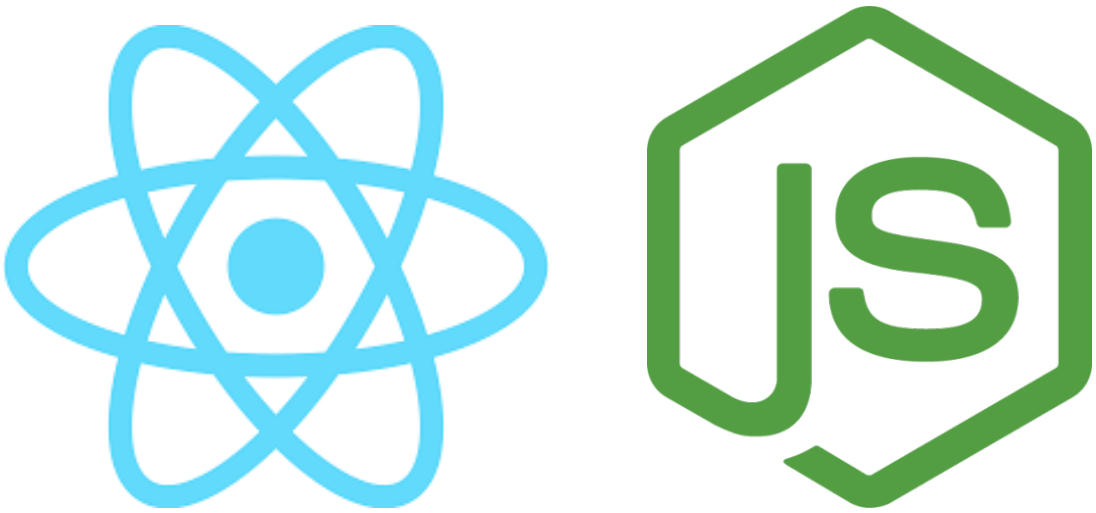
Table 1: User Groups and Access Control

7 Technologies for this project

In this assignment, first of all about the main phase, we want to practice our knowledge in this course, especially the knowledge about Database System, so we use **MySQL** as our Database Management System (DBMS).

Figure 3: MySQL

In another face, to make the user interface for this project, our group read many materials about many technologies for front-end and back-end and we decide to use **React** as our framework for front-end and **Node.js** as our framework for back-end. These 2 frameworks are very widespread in Vietnam as well as the world nowadays so we can approach the ecosystem of materials.



(a) React (b) Node.js

Figure 4: 2 frameworks for front-end and back-end

8 First MySQL code

We can also access our MySQL code here

```sql
DROP SCHEMA IF EXISTS HMS;
CREATE SCHEMA HMS;
USE HMS;

CREATE TABLE HMS.Department (
    departmentID CHAR(2),
    departmentName VARCHAR(50),
    PRIMARY KEY (departmentID)
);

CREATE TABLE HMS.Medical_Staff (
    staffID CHAR(7),
    staffSSN CHAR(10) NOT NULL UNIQUE,
    firstName VARCHAR(20) NOT NULL,
    midName  VARCHAR(50),
    lastName VARCHAR(20),
    staffDoB DATE,
    gender  ENUM ('F', 'M'),
    phoneNumber CHAR(10),
    salary  INT UNSIGNED,
    departmentID CHAR(2),

    PRIMARY KEY (staffID),
    FOREIGN KEY (departmentID) REFERENCES Department (departmentID)
);

CREATE TABLE HMS.Doctor (
    doctorID CHAR(7),
    license CHAR(12),

    PRIMARY KEY (doctorID),
    FOREIGN KEY (doctorID) REFERENCES Medical_Staff (staffID)
);

CREATE TABLE HMS.Manages (
    departmentID CHAR(2),
    doctorID CHAR(7) NOT NULL UNIQUE,
    startDate DATE NOT NULL,

    PRIMARY KEY (departmentID),
    FOREIGN KEY (departmentID) REFERENCES Department (departmentID),
    FOREIGN KEY (doctorID) REFERENCES Doctor (doctorID)
);

CREATE TABLE HMS.Specialization (
    doctorID CHAR(7),
    aSpecialization VARCHAR(20),

    PRIMARY KEY (doctorID, aSpecialization),
    FOREIGN KEY (doctorID) REFERENCES Doctor (doctorID)
```

```
);

CREATE TABLE HMS.Nurse (
    nurseID CHAR(7),
    yearExperience CHAR(12),

    PRIMARY KEY (nurseID),
    FOREIGN KEY (nurseID) REFERENCES Medical_Staff (staffID)
);

CREATE TABLE HMS.Room (
    roomID CHAR(3),
    capacity INT,
    roomType VARCHAR(20),

    PRIMARY KEY (roomID)
);

CREATE TABLE HMS.Patient (
    patientID CHAR(7),
    patientSSN CHAR(10) NOT NULL UNIQUE,
    firstName VARCHAR(20) NOT NULL,
    midName VARCHAR(50),
    lastName VARCHAR(20),
    patientDoB DATE,
    gender ENUM ('F', 'M'),
    phoneNumber CHAR(10),
    street VARCHAR(50),
    district VARCHAR(50),
    city VARCHAR(50),

    PRIMARY KEY (patientID)
);

CREATE TABLE HMS.Patients_Family (
    familyID INT,
    patientID CHAR(7) NOT NULL,
    relationship VARCHAR(50),
    phoneNumber CHAR(10),

    PRIMARY KEY (patientID, familyID),
    FOREIGN KEY (patientID) REFERENCES Patient (patientID)
);

CREATE TABLE HMS.Admitted_to (
    patientID CHAR(7),
    roomID CHAR(3),
    admittedDate DATE,
    dischargedDate DATE,

    PRIMARY KEY (patientID, roomID),
    FOREIGN KEY (patientID) REFERENCES Patient (patientID),
    FOREIGN KEY (roomID) REFERENCES Room (roomID)
```

```
);

CREATE TABLE HMS.Appointment (
    appointmentID INT,
    patientID CHAR(7),
    doctorID CHAR(7),
    appointmentDate DATE,
    appointmentTime TIME,

    PRIMARY KEY (appointmentID),
    FOREIGN KEY (patientID) REFERENCES Patient (patientID),
    FOREIGN KEY (doctorID) REFERENCES Doctor (doctorID)
);

CREATE TABLE HMS.Pharmacy (
    pharmacyID CHAR(2),
    pharmacyName VARCHAR(50),

    PRIMARY KEY (pharmacyID)
);

CREATE TABLE HMS.Medicine (
    medicineID CHAR(11),
    medicineName VARCHAR(50),
    dosage VARCHAR(1000),
    pharmacyID CHAR(2) NOT NULL,

    PRIMARY KEY (medicineID),
    FOREIGN KEY (pharmacyID) REFERENCES Pharmacy (pharmacyID)
);

CREATE TABLE HMS.Payment_Approach (
    PAID CHAR(3),
    PRIMARY KEY (PAID)
);

CREATE TABLE HMS.Cash (
    cashierID CHAR(3),
    cashierName VARCHAR(50),
    PAID CHAR(3) NOT NULL,

    PRIMARY KEY (cashierID),
    FOREIGN KEY (PAID) REFERENCES Payment_Approach (PAID)
);

CREATE TABLE HMS.Bank (
    bankID CHAR(3),
    bankName VARCHAR(50),
    PAID CHAR(3) NOT NULL,

    PRIMARY KEY (bankID),
    FOREIGN KEY (PAID) REFERENCES Payment_Approach (PAID)
);
```

```
157
158 CREATE TABLE HMS.Ewallet (
159     ewalletID CHAR(3),
160     ewalletName VARCHAR(50),
161     PAID CHAR(3) NOT NULL,
162
163     PRIMARY KEY (ewalletID),
164     FOREIGN KEY (PAID) REFERENCES Payment_Approach (PAID)
165 );
166
167 CREATE TABLE HMS.Bill (
168     billID INT,
169     patientID CHAR(7) NOT NULL,
170     amount INT UNSIGNED,
171     createdDate DATE,
172     paymentStatus BOOLEAN,
173     PAID CHAR(3),
174     paidDate DATE,
175
176     PRIMARY KEY (billID),
177     FOREIGN KEY (patientID) REFERENCES Patient (patientID),
178     FOREIGN KEY (PAID) REFERENCES Payment_Approach (PAID)
179 );
180
181 CREATE TABLE HMS.Treatment (
182     treatmentID INT,
183     patientID CHAR(7) NOT NULL,
184     treatmentDate DATE,
185     treatmentProcedure VARCHAR(1000),
186
187     PRIMARY KEY (treatmentID),
188     FOREIGN KEY (patientID) REFERENCES Patient (patientID)
189 );
190
191 CREATE TABLE HMS.Medical_Record (
192     recordID INT,
193     patientID CHAR(7) NOT NULL,
194     recordDate DATE,
195     treatmentID INT NOT NULL,
196     diagnosis VARCHAR(1000),
197     testResult VARCHAR(1000),
198
199     PRIMARY KEY (patientID, recordID),
200     FOREIGN KEY (patientID) REFERENCES Patient (patientID),
201     FOREIGN KEY (treatmentID) REFERENCES Treatment (treatmentID)
202 );
203
204 CREATE TABLE HMS.Performs (
205     doctorID CHAR(7) NOT NULL,
206     treatmentID INT NOT NULL,
207
208     PRIMARY KEY (doctorID, treatmentID),
209     FOREIGN KEY (doctorID) REFERENCES Doctor (doctorID),
```

```
    FOREIGN KEY (treatmentID) REFERENCES Treatment (treatmentID)
);

CREATE TABLE HMS.Assists (
    nurseID CHAR(7) NOT NULL,
    treatmentID INT NOT NULL,

    PRIMARY KEY (nurseID, treatmentID),
    FOREIGN KEY (nurseID) REFERENCES Nurse (nurseID),
    FOREIGN KEY (treatmentID) REFERENCES Treatment (treatmentID)
);

CREATE TABLE HMS.Takes_care (
    nurseID CHAR(7) NOT NULL,
    roomID CHAR(3) NOT NULL,

    PRIMARY KEY (nurseID, roomID),
    FOREIGN KEY (nurseID) REFERENCES Nurse (nurseID),
    FOREIGN KEY (roomID ) REFERENCES Room (roomID )
);

CREATE TABLE HMS.Prescribes (
    medicineID CHAR(11) NOT NULL,
    patientID CHAR(7) NOT NULL,
    doctorID CHAR(7) NOT NULL,
    prescribesDate DATE,

    PRIMARY KEY (patientID, medicineID, doctorID),
    FOREIGN KEY (patientID) REFERENCES Patient (patientID),
    FOREIGN KEY (doctorID) REFERENCES Doctor (doctorID),
    FOREIGN KEY (medicineID) REFERENCES Medicine (medicineID)
);

DROP ROLE IF EXISTS admin_role, doctor_role, nurse_role,
    receptionist_role, patient_role;

CREATE ROLE 'admin_role';
GRANT ALL PRIVILEGES ON HMS.* TO 'admin_role';

CREATE ROLE 'doctor_role';
GRANT SELECT, INSERT, UPDATE, DELETE ON HMS.Appointment TO 'doctor_role';
GRANT SELECT, INSERT, UPDATE, DELETE ON HMS.Medical_Record TO '
    doctor_role';
GRANT SELECT, INSERT, UPDATE, DELETE ON HMS.Treatment TO 'doctor_role';
GRANT SELECT, INSERT, UPDATE, DELETE ON HMS.Prescribes TO 'doctor_role';
GRANT SELECT ON HMS.Patient TO 'doctor_role';
GRANT SELECT ON HMS.Doctor TO 'doctor_role';
GRANT SELECT ON HMS.Nurse TO 'doctor_role';
GRANT SELECT ON HMS.Room TO 'doctor_role';
GRANT SELECT ON HMS.Admitted_to TO 'doctor_role';
GRANT SELECT ON HMS.Medicine TO 'doctor_role';

CREATE ROLE 'nurse_role';
```

```
GRANT SELECT, UPDATE ON HMS.Patient TO 'nurse_role';
GRANT SELECT ON HMS.Room TO 'nurse_role';
GRANT SELECT ON HMS.Admitted_to TO 'nurse_role';
GRANT SELECT ON HMS.Treatment TO 'nurse_role';
GRANT SELECT ON HMS.Prescribes TO 'nurse_role';

CREATE ROLE 'receptionist_role';
GRANT SELECT, INSERT, UPDATE, DELETE ON HMS.Patient TO 'receptionist_role
    ';
GRANT SELECT, INSERT, UPDATE, DELETE ON HMS.Patients_Family TO '
    receptionist_role';
GRANT SELECT, INSERT, UPDATE, DELETE ON HMS.Appointment TO '
    receptionist_role';
GRANT SELECT, INSERT, UPDATE, DELETE ON HMS.Admitted_to TO '
    receptionist_role';
GRANT SELECT, INSERT, UPDATE, DELETE ON HMS.Bill TO 'receptionist_role';

CREATE ROLE 'patient_role';
CREATE VIEW HMS.Patient_View AS
    SELECT * FROM HMS.Patient WHERE patientID = SUBSTRING_INDEX(USER(), '
    @', 1);
CREATE VIEW HMS.Patients_Family_View AS
    SELECT * FROM HMS.Patients_Family WHERE patientID = SUBSTRING_INDEX(
    USER(), '@', 1);
CREATE VIEW HMS.Appointment_View AS
    SELECT * FROM HMS.Appointment WHERE patientID = SUBSTRING_INDEX(USER
    (), '@', 1);
CREATE VIEW HMS.Medical_Record_View AS
    SELECT * FROM HMS.Medical_Record WHERE patientID = SUBSTRING_INDEX(
    USER(), '@', 1);
CREATE VIEW HMS.Prescribes_View AS
    SELECT * FROM HMS.Prescribes WHERE patientID = SUBSTRING_INDEX(USER()
    , '@', 1);
CREATE VIEW HMS.Bill_View AS
    SELECT * FROM HMS.Bill WHERE patientID = SUBSTRING_INDEX(USER(), '@',
     1);
GRANT SELECT ON HMS.Patient_View TO 'patient_role';
GRANT SELECT ON HMS.Patients_Family_View TO 'patient_role';
GRANT SELECT ON HMS.Appointment_View TO 'patient_role';
GRANT SELECT ON HMS.Medical_Record_View TO 'patient_role';
GRANT SELECT ON HMS.Prescribes_View TO 'patient_role';
GRANT SELECT ON HMS.Bill_View TO 'patient_role';

FLUSH PRIVILEGES;
```