ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO

CƠ SỞ DỮ LIỆU

CHỦ ĐỀ: QUẢN LÝ BỆNH VIỆN

GIẢNG VIÊN: Trần Hồng Việt

NHÓM: Bing Chilling

Vũ Bảo Chinh: 23020341

Nguyễn Gia Khánh: 23020385

Hoàng Mạnh Hùng: 23020371

Lê Hồng Anh: 23020327

Hà Nội, tháng 12 năm 2024

Báo cáo bài tập môn học Cơ sở dữ liệu

I. Phát biểu bài toán

Hệ thống quản lý bệnh nhân trong bệnh viện là một phần mềm được thiết kế để hỗ trợ việc quản lý thông tin bệnh nhân, các cuộc hẹn khám chữa bệnh, hóa đơn, đơn thuốc và các tài nguyên khác như phòng bệnh, bác sĩ, nhân viên y tế. Mục tiêu của hệ thống là tạo ra một cơ sở dữ liệu tập trung, giúp nhân viên y tế và quản lý dễ dàng truy cập, cập nhật và theo dõi tình trạng bệnh nhân cũng như các hoạt động liên quan đến việc khám chữa bệnh.

II. Mô tả chi tiết hệ thống

1. Tổng quan:

- Hệ thống quản lý bệnh nhân được xây dựng để phục vụ các cơ sở y tế, bao gồm bệnh viện, phòng khám, trung tâm y tế, nhằm cung cấp một giải pháp quản lý toàn diện cho quá trình chăm sóc sức khỏe của bệnh nhân. Hệ thống này giúp các bác sĩ, nhân viên y tế và quản lý bệnh viện theo dõi và quản lý thông tin bệnh nhân, các cuộc hẹn khám chữa bệnh, hóa đơn, đơn thuốc, và các phòng bệnh.
- Mục tiêu của hệ thống là cải thiện hiệu quả công việc và nâng cao chất lượng dịch vụ chăm sóc bệnh nhân thông qua việc tự động hóa và tối ưu hóa các quy trình quản lý bệnh viện. Các chức năng chính bao gồm quản lý thông tin bệnh nhân, bác sĩ, phòng bệnh, lịch sử cuộc hẹn, hóa đơn, đơn thuốc, và thuốc. Hệ thống sẽ giúp giảm thiểu sai sót trong việc quản lý dữ liệu, đảm bảo thông tin luôn được cập nhật và chính xác.
- Hệ thống này cũng hỗ trợ việc tạo báo cáo chi tiết về tình trạng thanh toán, các loại dịch vụ đã sử dụng, các đơn thuốc đã được kê, và các phòng bệnh đang được sử dụng. Từ đó, việc lập kế hoạch, đánh giá chất lượng chăm sóc

sức khỏe, cũng như việc quản lý tài nguyên bệnh viện sẽ trở nên thuận tiện và chính xác hơn.

Các yếu tố quan trọng của hệ thống bao gồm:

- **Dễ sử dụng:** Giao diện đơn giản và dễ tiếp cận cho cả bác sĩ, nhân viên y tế và quản lý bệnh viện.
- **Bảo mật:** Đảm bảo an toàn thông tin bệnh nhân và các dữ liệu y tế nhạy cảm.
- **Tích hợp:** Các tính năng được tích hợp đầy đủ để hỗ trợ quá trình quản lý bệnh nhân từ đầu đến cuối.
- **Tối ưu hóa tài nguyên:** Quản lý hiệu quả các phòng bệnh, lịch hẹn, và bác sĩ, giúp sử dụng tài nguyên một cách hợp lý.

- Hệ thống sẽ tạo ra một nền tảng cho các bệnh viện, phòng khám và các cơ sở y tế khác để cung cấp dịch vụ chăm sóc sức khỏe chất lượng cao, hỗ trợ bệnh nhân và nhân viên y tế trong việc theo dõi quá trình điều trị và chăm sóc sức khỏe.

2.Các thực thể và chức năng

BỆNH NHÂN (PATIENTS):

- **Chức năng: Lưu trữ và quản lý thông tin của bệnh nhân. Mỗi bệnh nhân đến bệnh viên được lưu lại các thông tin cá nhân.**
- **Thông tin chính**
  - RoomID : Mã phòng của bệnh nhân
  - PatientID: Mã bệnh nhân(khóa chính)
  - FullName: Tên đầy đủ của bệnh nhân
  - DateOfBirth: Ngày sinh của bệnh nhân
  - Gender: Giới tính của bệnh nhân
  - Address: Địa chỉ của bệnh nhân
  - PhoneNumber: Số điện thoại của bệnh nhân

- Email: Địa chỉ email của bệnh nhân

BÁC SĨ (DOCTORS):

- **Chức năng:** Lưu trữ và quản lý thông tin của bác sĩ. Thông tin về chuyên môn và lịch làm việc của bác sĩ sẽ được cập nhật.
- **Thông tin chính:**
  - **DoctorID**: Mã bác sĩ (khóa chính).
  - **FullName**: Tên đầy đủ của bác sĩ.
  - **Specialty**: Chuyên khoa của bác sĩ.
  - **PhoneNumber**: Số điện thoại của bác sĩ.
  - **Email**: Địa chỉ email của bác sĩ.
  - **HireDate**: Ngày bắt đầu làm việc của bác sĩ.

NHÂN VIÊN (STAFF):

- **Chức năng:** Lưu trữ và quản lý thông tin của nhân viên trong bệnh viện (không bao gồm bác sĩ).
- **Thông tin chính:**
  - **StaffID**: Mã nhân viên (khóa chính).
  - **FullName**: Tên đầy đủ của nhân viên.
  - **Role**: Vai trò hoặc chức vụ của nhân viên.
  - **PhoneNumber**: Số điện thoại của nhân viên.
  - **Email**: Địa chỉ email của nhân viên.
  - **HireDate**: Ngày bắt đầu làm việc của nhân viên.

PHÒNG BỆNH (ROOMS):

- **Chức năng:** Quản lý thông tin về các phòng bệnh trong bệnh viện, bao gồm loại phòng và tình trạng phòng.
- **Thông tin chính:**

- **RoomID**: Mã phòng (khóa chính).
- **RoomNumber**: Số phòng bệnh.
- **RoomType**: Loại phòng bệnh (General, Private, ICU).
- **Capacity**: Sức chứa của phòng bệnh.
- **AvailabilityStatus**: Trạng thái phòng bệnh (Available, Occupied).

LỊCH HẸN KHÁM (APPOINTMENTS):

- **Chức năng:** Quản lý lịch khám bệnh giữa bệnh nhân và bác sĩ, bao gồm ngày giờ và lý do khám.
- **Thông tin chính:**
  - **AppointmentID**: Mã lịch hẹn khám (khóa chính).
  - **PatientID**: Mã bệnh nhân (khóa ngoại tham chiếu Patients).
  - **DoctorID**: Mã bác sĩ (khóa ngoại tham chiếu Doctors).
  - **AppointmentDate**: Ngày giờ hẹn khám bệnh.
  - **Reason**: Lý do hẹn khám bệnh.

THUỐC (MEDICINES):

- **Chức năng:** Lưu trữ và quản lý thông tin về các loại thuốc có sẵn trong bệnh viện.
- **Thông tin chính:**
  - **MedicineID**: Mã thuốc (khóa chính).
  - **MedicineName**: Tên của loại thuốc.
  - **Description**: Mô tả chi tiết về thuốc.
  - **Price**: Giá của thuốc.

ĐƠN THUỐC (PRESCRIPTIONS):

- **Chức năng:** Quản lý thông tin đơn thuốc được bác sĩ kê cho bệnh nhân sau mỗi buổi khám.

- **Thông tin chính:**
  - **PrescriptionID**: Mã đơn thuốc (khóa chính).
  - **AppointmentID**: Mã lịch hẹn khám (khóa ngoại tham chiếu Appointments).
  - **IssueDate**: Ngày cấp đơn thuốc.

CHI TIẾT ĐƠN THUỐC (PRESCRIPTION DETAILS):

- **Chức năng:** Quản lý các chi tiết thuốc trong từng đơn thuốc cụ thể.
- **Thông tin chính:**
  - **PrescriptionDetailID**: Mã chi tiết đơn thuốc (khóa chính).
  - **PrescriptionID**: Mã đơn thuốc (khóa ngoại tham chiếu Prescriptions).
  - **MedicineID**: Mã thuốc (khóa ngoại tham chiếu Medicines).
  - **Quantity**: Số lượng thuốc trong đơn.

HÓA ĐƠN (BILLS):

- **Chức năng:** Lưu trữ và quản lý thông tin hóa đơn thanh toán của bệnh nhân.
- **Thông tin chính:**
  - **BillID**: Mã hóa đơn (khóa chính).
  - **PatientID**: Mã bệnh nhân (khóa ngoại tham chiếu Patients).
  - **TotalAmount**: Tổng số tiền cần thanh toán.
  - **IssueDate**: Ngày xuất hóa đơn.
  - **PaidStatus**: Trạng thái thanh toán (Paid, Unpaid).

CHI TIẾT HÓA ĐƠN (BILL DETAILS):

- **Chức năng:** Quản lý chi tiết các khoản mục trong mỗi hóa đơn cụ thể.
- **Thông tin chính:**
  - **BillDetailID**: Mã chi tiết hóa đơn (khóa chính).
  - **BillID**: Mã hóa đơn (khóa ngoại tham chiếu Bills).
  - **Description**: Mô tả khoản mục trong hóa đơn.

- **Amount**: Số tiền tương ứng với khoản mục.

3. Quy trình làm việc của hệ thống

Bước 1: Quản lý thông tin cơ bản

- Quản trị viên nhập thông tin cơ bản của hệ thống, bao gồm:
  - **Thông tin bệnh nhân (PATIENTS):** Cung cấp các thông tin về bệnh nhân như tên, ngày sinh, giới tính, số điện thoại, địa chỉ và email. Mỗi bệnh nhân sẽ được gắn một ID duy nhất để theo dõi thông tin.
  - **Thông tin bác sĩ (DOCTORS):** Bao gồm tên, chuyên môn, số điện thoại, email và ngày bắt đầu làm việc của bác sĩ. Mỗi bác sĩ cũng sẽ có một ID riêng để theo dõi các cuộc hẹn và điều trị cho bệnh nhân.
  - **Thông tin nhân viên (STAFFS):** Các nhân viên y tế, bao gồm cả các nhân viên hỗ trợ khác như điều dưỡng, quản lý bệnh viện. Thông tin này giúp phân công công việc và theo dõi hoạt động của nhân viên.
  - **Thông tin phòng bệnh (ROOMS):** Mỗi phòng bệnh sẽ có các thông tin như số phòng, loại phòng (phòng VIP, phòng thường), tình trạng phòng (có sẵn hay đã đầy), và sức chứa của phòng.
  - **Lịch hẹn khám (APPOINTMENTS) và đơn thuốc (MEDICINES):** Mỗi bệnh nhân có thể đặt lịch hẹn khám với bác sĩ, và hệ thống sẽ theo dõi các thông tin về lý do khám, thời gian và bác sĩ điều trị. Các đơn thuốc kê cho bệnh nhân sẽ được ghi lại trong hệ thống.
  - **Thông tin hóa đơn (BILLS):** Khi bệnh nhân điều trị xong, thông tin về hóa đơn sẽ được ghi lại, bao gồm các dịch vụ sử dụng, thuốc men, chi phí khám chữa bệnh và trạng thái thanh toán.

Bước 2: Lập hóa đơn viện phí

- Sau khi bệnh nhân hoàn tất quá trình khám và điều trị, nhân viên bệnh viện sẽ tiến hành lập hóa đơn viện phí. Các bước thực hiện bao gồm:
  - **Tạo hóa đơn (CREATE BILL):** Dựa trên thông tin đã ghi nhận về các dịch vụ, thuốc men và cuộc hẹn khám, nhân viên sẽ tạo một hóa đơn cho bệnh

nhân. Hóa đơn sẽ bao gồm tổng số tiền phải trả và các khoản chi tiết như chi phí khám chữa bệnh, thuốc men, và các phí phát sinh khác.

- **Cập nhật thông tin chi tiết hóa đơn (BILL DETAILS):** Hóa đơn sẽ được chia thành các mục chi tiết như phí khám bệnh, chi phí thuốc men, chi phí dịch vụ (nếu có), và tổng số tiền. Mỗi mục sẽ được ghi nhận riêng biệt.
- **Trạng thái thanh toán:** Hệ thống ghi nhận trạng thái thanh toán của hóa đơn (đã thanh toán hoặc chưa thanh toán). Điều này giúp theo dõi tình trạng tài chính của bệnh nhân và bệnh viện.

Bước 3: Xử lý thanh toán hóa đơn

- Bệnh nhân có thể thanh toán hóa đơn qua các phương thức khác nhau như chuyển khoản ngân hàng, tiền mặt hoặc thẻ tín dụng. Sau khi thanh toán, nhân viên bệnh viện sẽ cập nhật trạng thái của hóa đơn trong hệ thống:
  - **Thanh toán hoàn tất:** Nếu bệnh nhân thanh toán đầy đủ, trạng thái hóa đơn sẽ được cập nhật là "Đã thanh toán".
  - **Chưa thanh toán:** Nếu bệnh nhân chưa thanh toán, trạng thái hóa đơn vẫn giữ là "Chưa thanh toán" cho đến khi được thanh toán.

Bước 4: Quản lý và theo dõi tình trạng bệnh nhân

- Sau khi bệnh nhân đã thanh toán hoặc hoàn thành quá trình điều trị, hệ thống tiếp tục theo dõi tình trạng của bệnh nhân qua các cuộc hẹn khám tiếp theo hoặc các điều trị dài hạn.
  - **Cập nhật lịch sử khám bệnh:** Hệ thống sẽ lưu lại lịch sử khám bệnh, bao gồm các bác sĩ đã khám, các thuốc đã kê, và các dịch vụ đã sử dụng.
  - **Quản lý tái khám:** Nếu cần, bệnh nhân sẽ được chỉ định lịch tái khám và hệ thống sẽ nhắc nhở bệnh nhân về lịch hẹn trong tương lai.

Bước 5: Quản lý phòng bệnh

- **Quản lý phòng bệnh:** Hệ thống sẽ theo dõi tình trạng phòng bệnh, khi có bệnh nhân được chỉ định vào phòng, hệ thống sẽ cập nhật lại tình trạng phòng (sẵn sàng hay đầy). Khi bệnh nhân xuất viện, hệ thống sẽ giải phóng phòng và cập nhật lại trạng thái phòng.

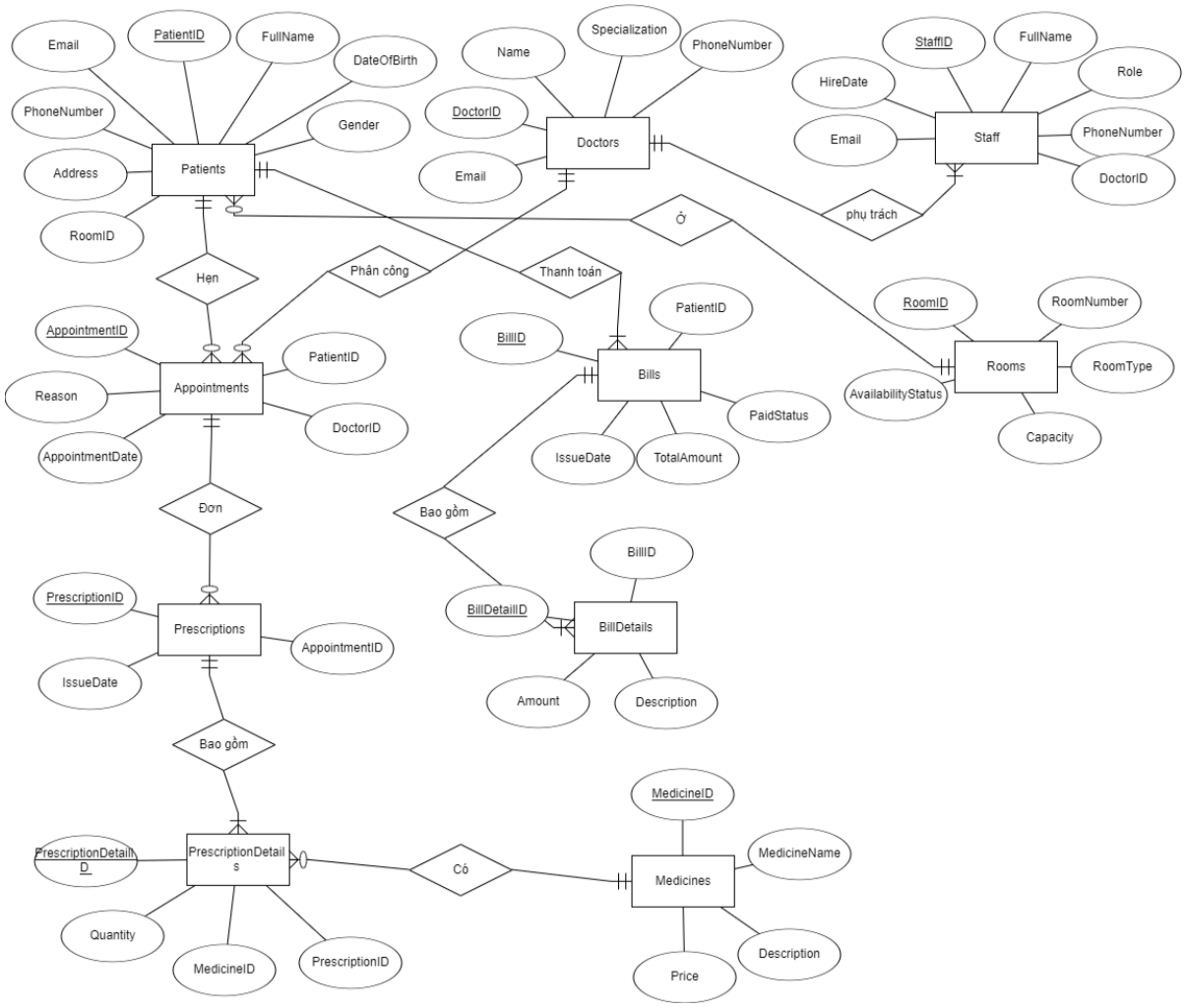
Bước 6: Báo cáo và phân tích dữ liệu

- Hệ thống cung cấp khả năng tạo báo cáo tổng hợp về các chỉ tiêu quan trọng, bao gồm:
    - **Báo cáo doanh thu:** Tổng hợp doanh thu từ hóa đơn của bệnh viện theo ngày, tuần, tháng hoặc năm.
    - **Báo cáo thanh toán:** Cung cấp thông tin về số lượng hóa đơn đã thanh toán và chưa thanh toán.
    - **Báo cáo tình trạng bệnh nhân:** Theo dõi tình trạng điều trị của bệnh nhân và các cuộc hẹn khám.
    - **Báo cáo về bác sĩ và nhân viên:** Cung cấp các thông tin về hiệu quả làm việc của bác sĩ và nhân viên y tế.

4. Ý nghĩa thực tế của hệ thống

- Hệ thống này hỗ trợ quản lý bệnh viện
    - Quản lý bệnh nhân hiệu quả: Dễ dàng theo dõi, quản lý bệnh nhân
    - Quản lí viện phí hiệu quả: Tính toán viện phí của bệnh nhân và lưu trữ
    - Báo cáo chi tiết và chính xác:

III. Sơ đồ ER(Entity Relationship)



Sơ đồ ER được thiết kế để mô tả các mối quan hệ giữa các thực thể trong hệ thống quản lý bệnh viện, bao gồm các đối tượng chính như bệnh nhân, bác sĩ, phòng bệnh, hóa đơn, thuốc và các dịch vụ đi kèm. Sơ đồ giúp người phát triển hiểu rõ cách thức các đối tượng liên kết với nhau và cách dữ liệu sẽ được lưu trữ trong cơ sở dữ liệu.

Các mối quan hệ giữa các thực thể:

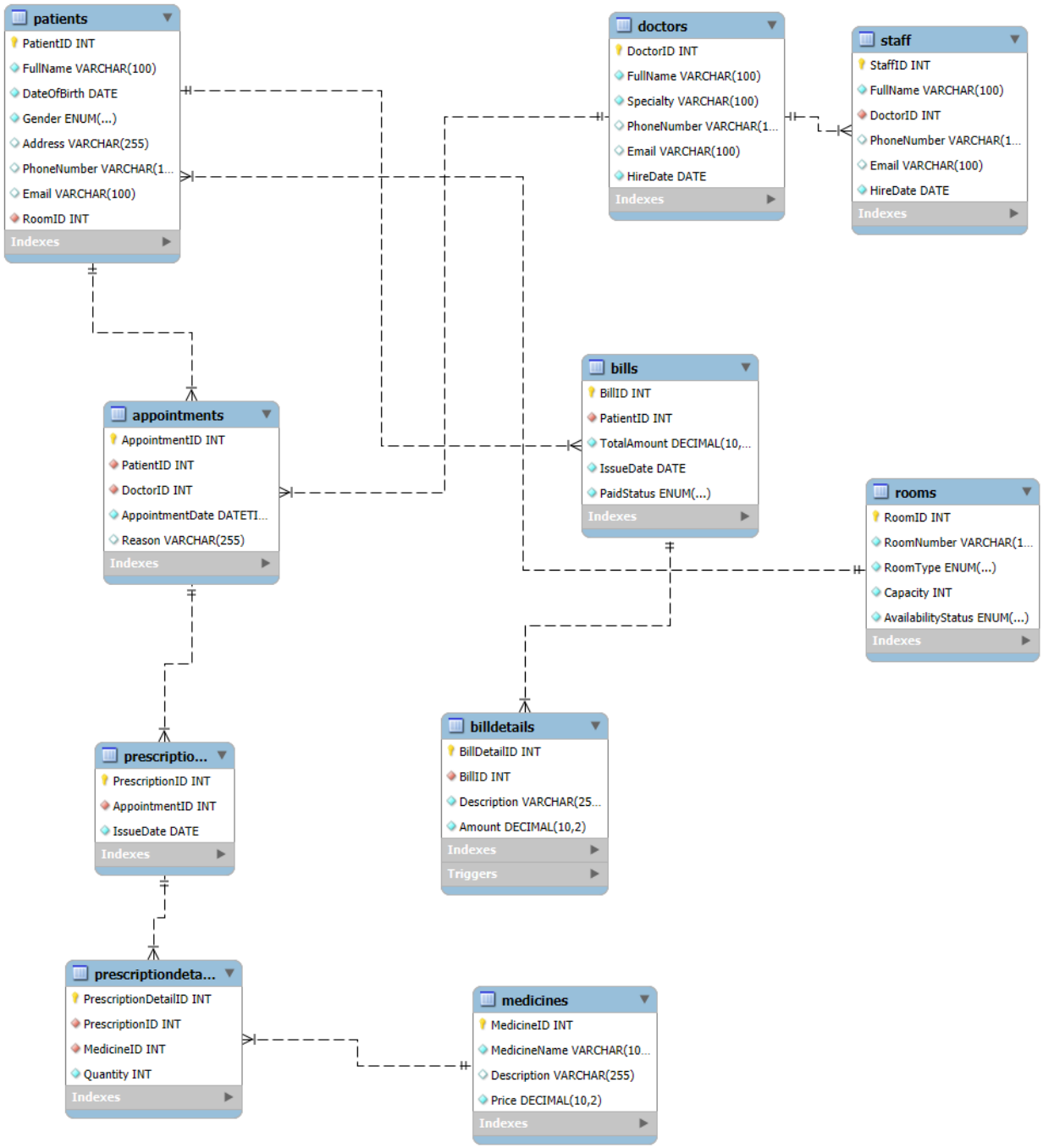
- **Patients và Appointments**: Mỗi bệnh nhân có thể có nhiều cuộc hẹn.
- **Appointments và Doctors**: Mỗi cuộc hẹn phải có một bác sĩ được chỉ định.
- **Bills và Patients**: Mỗi hóa đơn thuộc về một bệnh nhân.

- **Bills và BillDetails**: Mỗi hóa đơn có thể có nhiều chi tiết hóa đơn (mô tả các loại phí).
- **Prescriptions và Appointments**: Mỗi đơn thuốc được cấp cho một cuộc hẹn.
- **PrescriptionDetails và Medicines**: Mỗi chi tiết đơn thuốc bao gồm thuốc và số lượng sử dụng.

Ý nghĩa của sơ đồ ER:

Sơ đồ ER này giúp mô tả rõ ràng các đối tượng và mối quan hệ giữa chúng trong hệ thống quản lý bệnh viện. Nhờ vào sơ đồ này, việc phát triển hệ thống quản lý trở nên dễ dàng hơn vì nó cung cấp một cái nhìn tổng quan về cách dữ liệu được tổ chức và liên kết trong cơ sở dữ liệu.

IV. Chuyển sang lược đồ quan hệ và chuẩn hóa:



1. Giới thiệu chung

Lược đồ quan hệ (Relational Schema) là cách biểu diễn các bảng trong cơ sở dữ liệu và mối quan hệ giữa các bảng. Trong hệ thống cơ sở dữ liệu bệnh viện, chúng ta đã xác định được các bảng chủ yếu bao gồm thông tin bệnh nhân, bác sĩ, nhân

viên, cuộc hẹn, hóa đơn, chi tiết hóa đơn, đơn thuốc và thuốc. Mỗi bảng đều có các thuộc tính riêng và các mối quan hệ giữa các bảng đã được xác định rõ ràng.

Để đảm bảo dữ liệu được tổ chức hợp lý và tối ưu trong cơ sở dữ liệu, cần tiến hành chuẩn hóa các bảng theo các quy tắc chuẩn hóa cơ sở dữ liệu.

---

2. Các bảng trong lược đồ quan hệ

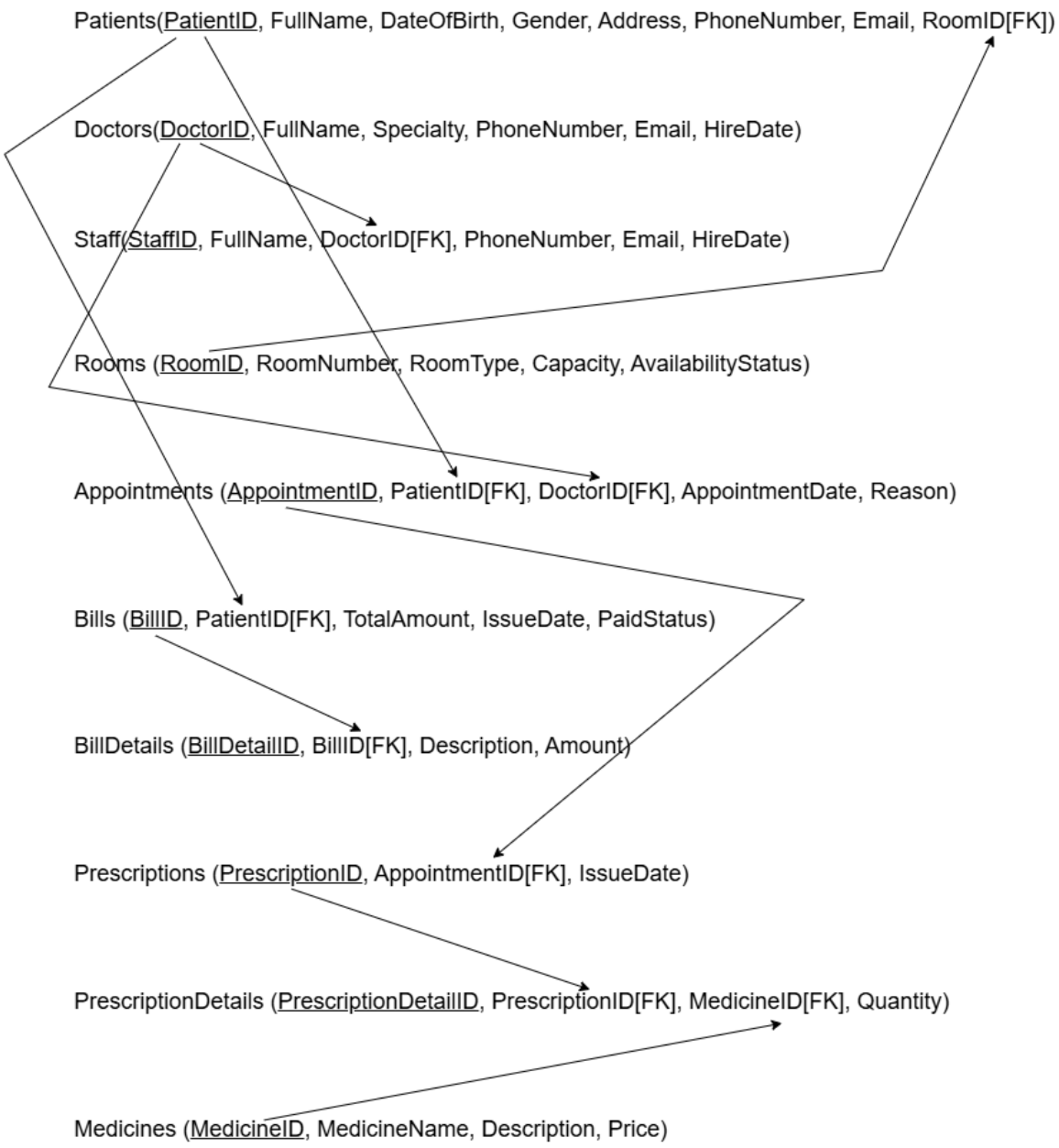
Dựa trên lược đồ quan hệ, chúng ta có các bảng chính sau:

1. **Patients (Bệnh nhân)**
   - Các thuộc tính: `PatientID`, `FullName`, `DateOfBirth`, `Gender`, `PhoneNumber`, `Address`, `Email`, `RoomID`.
   - Mối quan hệ: Một bệnh nhân có thể có nhiều cuộc hẹn (Appointments) và hóa đơn (Bills). Mỗi bệnh nhân chỉ có một phòng (Rooms).
2. **Doctors (Bác sĩ)**
   - Các thuộc tính: `DoctorID`, `FullName`, `Specialty`, `PhoneNumber`, `Email`, `HireDate`.
   - Mối quan hệ: Một bác sĩ có thể có nhiều cuộc hẹn với bệnh nhân (Appointments).
3. **Staff (Nhân viên)**
   - Các thuộc tính: `StaffID`, `FullName`, `PhoneNumber`, `Email`, `HireDate`.
   - Mối quan hệ: Một nhân viên có thể hỗ trợ một bác sĩ (Doctors).
4. **Appointments (Cuộc hẹn)**
   - Các thuộc tính: `AppointmentID`, `PatientID`, `DoctorID`, `AppointmentDate`, `Reason`.
   - Mối quan hệ: Mỗi cuộc hẹn liên kết với một bệnh nhân (Patients) và một bác sĩ (Doctors).
5. **Bills (Hóa đơn)**
   - Các thuộc tính: `BillID`, `PatientID`, `TotalAmount`, `IssueDate`, `PaidStatus`.
   - Mối quan hệ: Mỗi hóa đơn liên kết với một bệnh nhân (Patients) và có nhiều chi tiết hóa đơn (BillDetails).

6. **BillDetails (Chi tiết hóa đơn)**
    - Các thuộc tính: `BillDetailID`, `BillID`, `Description`, `Amount`.
    - Mối quan hệ: Mỗi chi tiết hóa đơn thuộc một hóa đơn (Bills).
7. **Prescriptions (Đơn thuốc)**
    - Các thuộc tính: `PrescriptionID`, `AppointmentID`, `IssueDate`.
    - Mối quan hệ: Mỗi đơn thuốc liên kết với một cuộc hẹn (Appointments).
8. **PrescriptionDetails (Chi tiết đơn thuốc)**
    - Các thuộc tính: `PrescriptionDetailID`, `PrescriptionID`, `MedicineID`, `Quantity`.
    - Mối quan hệ: Mỗi chi tiết đơn thuốc liên kết với một đơn thuốc (Prescriptions) và một loại thuốc (Medicines).
9. **Medicines (Thuốc)**
    - Các thuộc tính: `MedicineID`, `MedicineName`, `Description`, `Price`.
    - Mối quan hệ: Mỗi loại thuốc có thể xuất hiện trong nhiều đơn thuốc (PrescriptionDetails).
10. **Rooms (Phòng)**
    - Các thuộc tính: `RoomID`, `RoomNumber`, `RoomType`, `Capacity`, `AvailabilityStatus`.
    - Mối quan hệ: Một phòng có thể chứa nhiều bệnh nhân (Patients).

---

3. Mô hình dữ liệu quan hệ , chuẩn hóa lược đồ quan hệ sang dạng chuẩn 3NF



Danh sách quan hệ và phụ thuộc hàm tương ứng:

PATIENTS(PatientID [PK], FullName, DateOfBirth, Gender, Address, PhoneNumber, Email, RoomID [FK])
Phụ thuộc hàm: {(PatientID) → (FullName, DateOfBirth, Gender, Address, PhoneNumber, Email, RoomID)}

DOCTORS(DoctorID [PK], FullName, Specialty, PhoneNumber, Email, HireDate)
Phụ thuộc hàm: {(DoctorID) → (FullName, Specialty, PhoneNumber, Email, HireDate)}

STAFF(StaffID [PK], FullName, PhoneNumber, Email, HireDate, DoctorID [FK])
Phụ thuộc hàm: {(StaffID) → (FullName, PhoneNumber, Email, HireDate)}

ROOMS(RoomID [PK], RoomNumber, RoomType, Capacity, AvailabilityStatus)
Phụ thuộc hàm: {(RoomID) → (RoomNumber, RoomType, Capacity, AvailabilityStatus)}

APPOINTMENTS(AppointmentID [PK], PatientID [FK], DoctorID [FK], AppointmentDate, Reason)
Phụ thuộc hàm: {(AppointmentID) → (PatientID, DoctorID, AppointmentDate, Reason)}

BILLS(BillID [PK], PatientID [FK], TotalAmount, IssueDate, PaidStatus)
Phụ thuộc hàm: {(BillID) → (PatientID, TotalAmount, IssueDate, PaidStatus)}

BILL_DETAILS(BillDetailID [PK], BillID [FK], Description, Amount)
Phụ thuộc hàm: {(BillDetailID) → (BillID, Description, Amount)}

PRESCRIPTIONS(PrescriptionID [PK], AppointmentID [FK], IssueDate)
Phụ thuộc hàm: {(PrescriptionID) → (AppointmentID, IssueDate)}

PRESCRIPTION_DETAILS(PrescriptionDetailID [PK], PrescriptionID [FK], MedicineID [FK], Quantity)
Phụ thuộc hàm: {(PrescriptionDetailID) → (PrescriptionID, MedicineID, Quantity)}

MEDICINES(MedicineID [PK], MedicineName, Description, Price)
Phụ thuộc hàm: {(MedicineID) → (MedicineName, Description, Price)}

Chuẩn hóa lược đồ quan hệ sang chuẩn 3NF:

1. Patients:

Phụ thuộc hàm: (PatientID) → (FullName, DateOfBirth, Gender, Address, PhoneNumber, Email, RoomID).
Đã ở dạng chuẩn 3NF.

2. Doctors:

Phụ thuộc hàm: (DoctorID) → (FullName, Specialty, PhoneNumber, Email, HireDate).
Đã ở dạng chuẩn 3NF.

3. Staff:

Phụ thuộc hàm: (StaffID) → (FullName, PhoneNumber, Email, HireDate).
Đã ở dạng chuẩn 3NF.

4. Rooms:

Phụ thuộc hàm: (RoomID) → (RoomNumber, RoomType, Capacity, AvailabilityStatus).
Đã ở dạng chuẩn 3NF.

5. Appointments:

Phụ thuộc hàm: (AppointmentID) → (PatientID, DoctorID, AppointmentDate, Reason).
Đã ở dạng chuẩn 3NF.

6. Bills:

Phụ thuộc hàm: (BillID) → (PatientID, TotalAmount, IssueDate, PaidStatus).
Đã ở dạng chuẩn 3NF.

7. BillDetails:

Phụ thuộc hàm: (BillDetailID) → (BillID, Description, Amount).
Đã ở dạng chuẩn 3NF.

8. Prescriptions:

Phụ thuộc hàm: (PrescriptionID) → (AppointmentID, IssueDate).
Đã ở dạng chuẩn 3NF.

9. PrescriptionDetails:

Phụ thuộc hàm: (PrescriptionDetailID) → (PrescriptionID, MedicineID, Quantity).
Đã ở dạng chuẩn 3NF.

10. Medicines:

Phụ thuộc hàm: (MedicineID) → (MedicineName, Description, Price).
Đã ở dạng chuẩn 3NF.

4. Ứng dụng chuẩn hóa vào cơ sở dữ liệu

Dựa trên các quy tắc chuẩn hóa, lược đồ quan hệ của cơ sở dữ liệu bệnh viện hiện tại đã được thiết kế với các bảng được chuẩn hóa. Các bảng đã loại bỏ sự dư thừa thông tin và tối ưu hóa việc quản lý dữ liệu:

- **Bảng** `Appointments` đã loại bỏ sự phụ thuộc bán khóa khi lưu trữ thông tin bệnh nhân và bác sĩ riêng biệt thay vì lưu trực tiếp vào bảng này.
- **Bảng** `Bills` và `BillDetails` đã được phân tách để lưu trữ thông tin tổng thể hóa đơn và chi tiết hóa đơn, giúp giảm thiểu sự dư thừa và đảm bảo tính chính xác khi cập nhật dữ liệu.
- **Bảng** `Prescriptions` và `PrescriptionDetails` giúp tách rời thông tin về đơn thuốc và chi tiết thuốc, làm cho cơ sở dữ liệu dễ dàng mở rộng và duy trì.

---

5. Kết luận

Việc chuyển đổi từ lược đồ thực thể (ERD) sang lược đồ quan hệ và thực hiện chuẩn hóa cơ sở dữ liệu giúp tối ưu hóa việc lưu trữ và truy vấn dữ liệu. Cơ sở dữ liệu đã được thiết kế hợp lý với các mối quan hệ rõ ràng và các bảng được chuẩn hóa, từ đó nâng cao hiệu quả trong việc quản lý và duy trì hệ thống cơ sở dữ liệu bệnh viện.

V. Thiết lập cơ sở dữ liệu bệnh viện

- Tạo cơ sở quản lý bệnh viện

```
DROP DATABASE IF EXISTS Hospital_Management_System ;
CREATE DATABASE IF NOT EXISTS Hospital_Management_System ;
USE Hospital_Management_System;
```

- Tạo bảng lưu thông tin bệnh nhân:

```sql
create table Patients (
    PatientID INT AUTO_INCREMENT PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    DateOfBirth DATE NOT NULL,
    Gender ENUM('Male', 'Female', 'Other') NOT NULL,
    Address VARCHAR(255),
    PhoneNumber VARCHAR(15),
    Email VARCHAR(100),
    RoomID INT NOT NULL,
    FOREIGN KEY(RoomID) REFERENCES Rooms(RoomID)
);
```

- Tạo bảng lưu thông tin bác sĩ:

```sql
CREATE TABLE Doctors (
    DoctorID INT AUTO_INCREMENT PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    Specialty VARCHAR(100) NOT NULL,
    PhoneNumber VARCHAR(15),
    Email VARCHAR(100),
    HireDate DATE NOT NULL
);
```

- Tạo bảng lưu thông tin nhân viên:

```sql
create table Staff (
    StaffID INT AUTO_INCREMENT PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    DoctorID INT NOT NULL,
    PhoneNumber VARCHAR(15),
    Email VARCHAR(100),
    HireDate DATE NOT NULL,
    FOREIGN KEY(DoctorID) REFERENCES Doctors(DoctorID)
);
```

- Tạo bảng lưu thông tin phòng bệnh:

```sql
CREATE TABLE Rooms (
    RoomID INT AUTO_INCREMENT PRIMARY KEY,
    RoomNumber VARCHAR(10) NOT NULL,
    RoomType ENUM('General', 'Private', 'ICU') NOT NULL,
    Capacity INT NOT NULL,
    AvailabilityStatus ENUM('Available', 'Occupied') NOT NULL
);
```

- Tạo bảng lưu thông tin lịch khám bệnh:

```sql
CREATE TABLE Appointments (
    AppointmentID INT AUTO_INCREMENT PRIMARY KEY,
    PatientID INT NOT NULL,
    DoctorID INT NOT NULL,
    AppointmentDate DATETIME NOT NULL,
    Reason VARCHAR(255),
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
);
```

- Tạo bảng lưu thông tin thuốc:

```sql
CREATE TABLE Medicines (
    MedicineID INT AUTO_INCREMENT PRIMARY KEY,
    MedicineName VARCHAR(100) NOT NULL,
    Description VARCHAR(255),
    Price DECIMAL(10, 2) NOT NULL
);
```

- Tạo bảng lưu thông tin đơn thuốc:

```sql
CREATE TABLE Prescriptions (
    PrescriptionID INT AUTO_INCREMENT PRIMARY KEY,
    AppointmentID INT NOT NULL,
    IssueDate DATE NOT NULL,
    FOREIGN KEY (AppointmentID) REFERENCES Appointments(AppointmentID)
);
```

- Tạo bảng lưu thông tin chi tiết đơn thuốc :

```sql
CREATE TABLE PrescriptionDetails (
    PrescriptionDetailID INT AUTO_INCREMENT PRIMARY KEY,
    PrescriptionID INT NOT NULL,
    MedicineID INT NOT NULL,
    Quantity INT NOT NULL,
    FOREIGN KEY (PrescriptionID) REFERENCES Prescriptions(PrescriptionID),
    FOREIGN KEY (MedicineID) REFERENCES Medicines(MedicineID)
);
```

- Tạo bảng lưu thông tin hóa đơn:

```sql
CREATE TABLE Bills (
    BillID INT AUTO_INCREMENT PRIMARY KEY,
    PatientID INT NOT NULL,
    TotalAmount DECIMAL(10, 2) NOT NULL,
    IssueDate DATE NOT NULL,
    PaidStatus ENUM('Paid', 'Unpaid') NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID)
);
```

- Tạo bảng lưu thông tin chi tiết hóa đơn :

```
CREATE TABLE BillDetails (
    BillDetailID INT AUTO_INCREMENT PRIMARY KEY,
    BillID INT NOT NULL,
    Description VARCHAR(255) NOT NULL,
    Amount DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (BillID) REFERENCES Bills(BillID)
);
```

VI. Tạo ràng buộc

- **Ràng buộc 1: Giới tính của bệnh nhân phải là Nam hoặc Nữ**

```
ALTER TABLE Patients
ADD CONSTRAINT chk_Gender CHECK (Gender IN ('Male', 'Female'));
```

- **Ràng buộc 2: Số phòng trong bệnh viện không được phép âm**

```
ALTER TABLE Rooms
ADD CONSTRAINT chk_Capacity CHECK (Capacity > 0);
```

- **Ràng buộc 3: Viện phí thanh toán không được phép âm**

```
ALTER TABLE Bills
ADD CONSTRAINT chk_TotalAmount CHECK (TotalAmount >= 0);
```

- **Ràng buộc 4: Giá thuốc không được phép âm**

```
ALTER TABLE Medicines
ADD CONSTRAINT chk_Price CHECK (Price > 0);
```

- **Ràng buộc 5: Giá đơn thuốc không được phép âm**

```
ALTER TABLE PrescriptionDetails
ADD CONSTRAINT chk_Quantity CHECK (Quantity > 0);
```

VII. Nhập dữ liệu đầu vào

- Nhập dữ liệu mẫu cho bảng Patients:

```
insert into Patients (FullName, DateOfBirth, Gender, Address, PhoneNumber, Email, RoomID)
values ('Nguyễn Thị Ly', '1995-05-15', 'Female', 'Hà Nội, Việt Nam', '0987654321', 'lynguyen@gmail.com', 1),
       ('Trần Khánh Toàn', '1985-03-20', 'Male', 'Hồ Chí Minh , Việt Nam', '0912345678', 'toantran@gmail.com', 1),
       ('Hoàng Thị Loan', '2000-08-10', 'Female', 'Đà Nẵng, Việt Nam', '0932112233', 'loan123@gmail.com', 2),
       ('Trần Minh Duy', '1992-07-25', 'Male', 'Thái Bình, Việt Nam', '0988776655', 'duytran1992@gmail.com', 4),
       ('Nguyễn Minh Tuấn', '1988-02-14', 'Male', 'Hưng Yên, Vietnam', '0902345678', 'tuannguyen1988@gmail.com', 5),
       ('Trần Thị Mai', '1995-12-30', 'Female', 'Nghệ An, Việt Nam', '0967554433', 'mai567@gmail.com', 6),
       ('Nguyễn Thị Lan Anh', '2002-09-15', 'Female', 'Hải Dương, Việt Nam', '0977223344', 'lananh0303@gmail.com', 7),
       ('Nguyễn Minh Quang', '1986-01-10', 'Male', 'Thanh Hóa, Việt Nam', '0911887766', 'quangnguyen1986@gmail.com', 8),
       ('Nguyễn Thị Mai', '1999-04-05', 'Female', 'Huế, Việt Nam', '0944556677', 'mainguyen234@gmail.com', 10),
       ('Nguyễn Thái Sơn ', '1993-11-19', 'Male', 'Nam Định, Việt Nam', '0901223344', 'sonnguyen123@gmail.com', 10);
```

- Nhập dữ liệu mẫu cho bảng Doctors :

```
insert into Doctors (FullName, Specialty, PhoneNumber, Email, HireDate)
values ('Mr.Nguyễn Hữu Trình', 'Cardiologist', '0968736655', 'trinhnguyen03@hospital.com', '2020-01-10'),
       ('Mr. Trần Minh Hoàng', 'Dermatologist', '0912345678', 'hoangtran98@hospital.com', '2019-07-15'),
       ('Mrs. Vũ Thị Mai', 'Neurologist', '0932112233', 'maivu24@hospital.com', '2021-08-20'),
       ('Mr. Phạm Tiến Minh', 'Orthopedic', '0987654321', 'minhpham68@hospital.com', '2022-03-10'),
       ('Mr. Nguyễn Văn Tuấn', 'Pediatrician', '0902345678', 'tuannguyen78@hospital.com', '2018-12-01'),
       ('Mrs. Nguyễn Phương Linh', 'Surgeon', '0967554433', 'linhnguyen23@hospital.com', '2021-06-25'),
       ('Mrs. Nguyễn Thu Phương', 'General Practitioner', '0977223344', 'phuongnguyen12@hospital.com', '2020-09-10'),
       ('Mr. Nguyễn Khánh Duy', 'Dentist', '0911887766', 'duynguyen88@hospital.com', '2017-05-17'),
       ('Mr. Nguyễn Khánh Huyền', 'Ophthalmologist', '0944556677', 'khanhuyen234@hospital.com', '2022-02-05'),
       ('Mr. Nguyễn Thanh Sơn', 'Endocrinologist', '0901223344', 'sonnguyen@hospital.com', '2019-04-22');
```

- Nhập dữ liệu mẫu cho bảng Staff :

```
INSERT INTO Staff (FullName, DoctorID, PhoneNumber, Email, HireDate)
values ('Nguyễn Thị Mai', 1, '0988776655', 'mainurse@hospital.com', '2018-01-10'),
       ('Trần Minh Hoàng', 2, '0912345678', 'receptionist@hospital.com', '2017-02-20'),
       ('Lê Thị Lan', 3, '0932112233', 'labtech@hospital.com', '2020-08-05'),
       ('Phạm Quốc Duy', 4, '0987654321', 'pharmacist@hospital.com', '2019-03-15'),
       ('Nguễn Tuấn Minh', 5, '0902345678', 'admin@hospital.com', '2021-12-12'),
       ('Nguyễn Phương Lan', 6, '0967554433', 'physio@hospital.com', '2020-06-10'),
       ('Phạm Lan Anh', 7, '0977223344', 'billing@hospital.com', '2021-11-20'),
       ('Bùi Minh Quang', 8, '0911887766', 'security@hospital.com', '2018-07-17'),
       ('Vũ Thị Mai', 9, '0944556677', 'medrecords@hospital.com', '2019-02-25'),
       ('Bùi Thái Sơn', 10, '0901223344', 'cleaner@hospital.com', '2022-03-01');
```

- Nhập dữ liệu mẫu cho bảng Rooms :

```sql
insert into Rooms (RoomNumber, RoomType, Capacity, AvailabilityStatus)
values ('101', 'General', 2, 'Available'),
       ('102', 'Private', 1, 'Occupied'),
       ('103', 'ICU', 1, 'Available'),
       ('104', 'General', 2, 'Available'),
       ('205', 'Private', 1, 'Occupied'),
       ('106', 'General', 2, 'Available'),
       ('307', 'ICU', 1, 'Occupied'),
       ('108', 'General', 2, 'Available'),
       ('109', 'Private', 1, 'Available'),
       ('310', 'General', 2, 'Occupied');
```

- Nhập dữ liệu mẫu cho bảng Appointments :

```sql
insert into Appointments (PatientID, DoctorID, AppointmentDate, Reason)
values (1, 1, '2024-08-15 08:00:00', 'Heart checkup'),
       (2, 2, '2024-02-08 09:30:00', 'Skin rash'),
       (3, 3, '2024-04-12 10:00:00', 'Migraine'),
       (4, 4, '2024-06-13 11:15:00', 'Joint pain'),
       (5, 5, '2024-03-11 14:30:00', 'Routine checkup'),
       (6, 6, '2024-02-20 15:00:00', 'Physiotherapy'),
       (7, 7, '2024-05-18 16:30:00', 'General checkup'),
       (8, 8, '2024-07-25 17:00:00', 'Dental issue'),
       (9, 9, '2024-01-05 09:30:00', 'Vision problems'),
       (10, 10, '2024-12-20 11:00:00', 'Thyroid test');
```

- Nhập dữ liệu mẫu cho bảng Medicines :

```sql
insert into Medicines (MedicineName, Description, Price)
values ('Paracetamol', 'Pain reliever', 5.00),
       ('Amoxicillin', 'Antibiotic', 10.00),
       ('Ibuprofen', 'Anti-inflammatory', 8.00),
       ('Aspirin', 'Pain reliever', 6.50),
       ('Metformin', 'Diabetes medication', 12.00),
       ('Ciprofloxacin', 'Antibiotic', 15.00),
       ('Cetirizine', 'Antihistamine', 7.00),
       ('Lisinopril', 'Blood pressure medication', 20.00),
       ('Simvastatin', 'Cholesterol medication', 25.00),
       ('Omeprazole', 'Antacid', 10.50);
```

- Nhập dữ liệu mẫu cho bảng Prescriptions :

```
insert into Prescriptions (AppointmentID, IssueDate)
values (1, '2024-12-15'),
       (2, '2024-12-16'),
       (3, '2024-12-17'),
       (4, '2024-12-18'),
       (5, '2024-12-19'),
       (6, '2024-12-20'),
       (7, '2024-12-21'),
       (8, '2024-12-22'),
       (9, '2024-12-23'),
       (10, '2024-12-24');
```

- Nhập dữ liệu mẫu cho bảng PrescriptionDetails :

```
insert into PrescriptionDetails (PrescriptionID, MedicineID, Quantity)
values (1, 1, 10),
       (1, 3, 5),
       (2, 2, 7),
       (2, 4, 8),
       (3, 5, 3),
       (3, 6, 2),
       (4, 7, 10),
       (4, 8, 4),
       (5, 9, 5),
       (5, 10, 6);
```

- Nhập dữ liệu mẫu cho bảng Bills :

```
insert into Bills (PatientID, TotalAmount, IssueDate, PaidStatus)
values (1, 100.00, '2024-12-15', 'Unpaid'),
       (2, 80.00, '2024-12-16', 'Paid'),
       (3, 70.00, '2024-12-17', 'Unpaid'),
       (4, 90.00, '2024-12-18', 'Unpaid'),
       (5, 50.00, '2024-12-19', 'Paid'),
       (6, 75.00, '2024-12-20', 'Unpaid'),
       (7, 60.00, '2024-12-21', 'Paid'),
       (8, 110.00, '2024-12-22', 'Unpaid'),
       (9, 85.00, '2024-12-23', 'Paid'),
       (10, 95.00, '2024-12-24', 'Unpaid');
```

- Nhập dữ liệu mẫu cho bảng BillDetails :

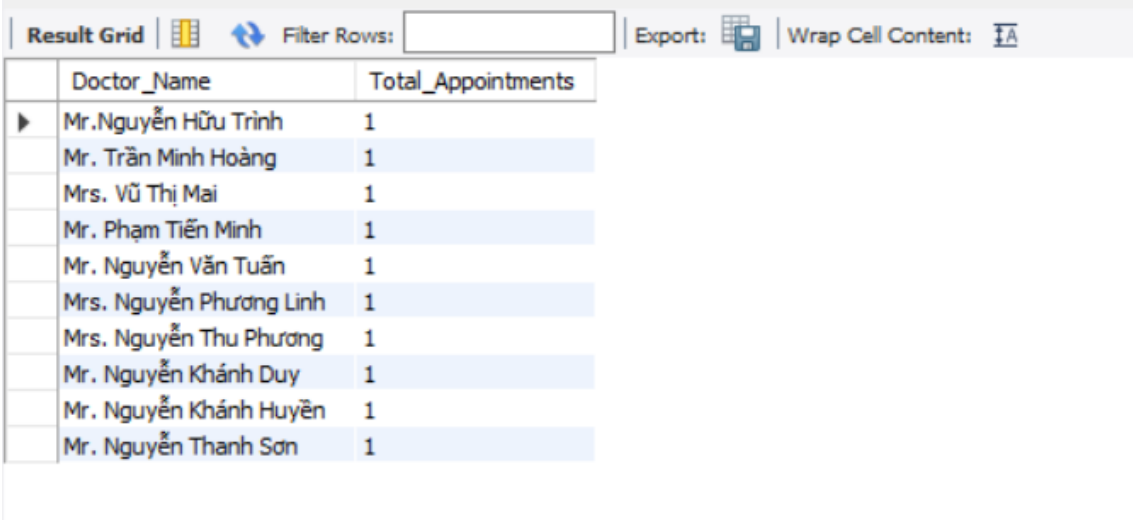
```
insert into BillDetails (BillID, Description, Amount)
values (1, 'Consultation Fee', 30.00),
       (1, 'Medicine Fee', 20.00),
       (2, 'Consultation Fee', 25.00),
       (2, 'Medicine Fee', 15.00),
       (3, 'Consultation Fee', 40.00),
       (3, 'Medicine Fee', 30.00),
       (4, 'Consultation Fee', 35.00),
       (4, 'Medicine Fee', 25.00),
       (5, 'Consultation Fee', 20.00),
       (5, 'Medicine Fee', 30.00);
```

VIII.Queries

- Truy vấn dùng **INNER JOIN:**
  - Truy vấn nhằm đếm tổng số lượng lịch hẹn của từng bác sĩ trong hệ thống, đồng thời sắp xếp kết quả theo thứ tự giảm dần dựa trên số lượng lịch hẹn.
    - **Bảng chính:** `Doctors` chứa thông tin bác sĩ, bao gồm tên và mã định danh (`DoctorID`).
    - **Bảng phụ:** `Appointments` chứa thông tin lịch hẹn, trong đó `DoctorID` là khóa ngoại liên kết với bảng `Doctors`.
    - **Phương pháp:**
      - **INNER JOIN:** Liên kết hai bảng dựa trên cột `DoctorID` chung.
      - **COUNT():** Hàm tổng hợp để đếm số lượng lịch hẹn tương ứng cho mỗi bác sĩ.
      - **GROUP BY:** Nhóm kết quả theo bác sĩ (`Doctor_Name`) để tính tổng lịch hẹn.
      - **ORDER BY:** Sắp xếp kết quả theo thứ tự giảm dần (`DESC`) dựa trên số lượng lịch hẹn.
    - Câu lệnh :

```sql
SELECT d.FullName AS Doctor_Name, COUNT(a.AppointmentID) AS Total_Appointments
FROM Doctors d
        INNER JOIN Appointments a ON d.DoctorID = a.DoctorID
GROUP BY d.FullName
ORDER BY Total_Appointments DESC;
```

    - Kết quả :



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Doctor_Name	Total_Appointments
▶	Mr.Nguyễn Hữu Trình	1
	Mr. Trần Minh Hoàng	1
	Mrs. Vũ Thị Mai	1
	Mr. Phạm Tiến Minh	1
	Mr. Nguyễn Văn Tuấn	1
	Mrs. Nguyễn Phương Linh	1
	Mrs. Nguyễn Thu Phương	1
	Mr. Nguyễn Khánh Duy	1
	Mr. Nguyễn Khánh Huyền	1
	Mr. Nguyễn Thanh Sơn	1

- **Ý nghĩa:** Truy vấn này giúp:

1. Đánh giá khối lượng công việc của từng bác sĩ.
2. Xác định bác sĩ nào có lịch hẹn nhiều nhất, để điều phối lại tài nguyên nếu cần.
3. Cung cấp số liệu thống kê hỗ trợ quản lý hiệu quả hệ thống bệnh viện.

- Truy vấn SQL trên thực hiện việc kết hợp dữ liệu từ ba bảng để lấy thông tin về các cuộc hẹn (appointments), bệnh nhân (patients), và bác sĩ (doctors).
  - Câu lệnh:

```
select a.AppointmentID, p.FullName AS Patient_Name, d.FullName AS Doctor_Name, a.AppointmentDate
from Appointments a
        inner join Patients p on a.PatientID = p.PatientID
        inner join Doctors d ON a.DoctorID = d.DoctorID;
```

  - Kết quả:

AppointmentID	Patient_Name	Doctor_Name	AppointmentDate
1	Nguyễn Thị Ly	Mr.Nguyễn Hữu Trình	2024-08-15 08:00:00
2	Trần Khánh Toàn	Mr. Trần Minh Hoàng	2024-02-08 09:30:00
3	Hoàng Thị Loan	Mrs. Vũ Thị Mai	2024-04-12 10:00:00
4	Trần Minh Duy	Mr. Phạm Tiến Minh	2024-06-13 11:15:00
5	Nguyễn Minh Tuấn	Mr. Nguyễn Văn Tuấn	2024-03-11 14:30:00
6	Trần Thị Mai	Mrs. Nguyễn Phương Linh	2024-02-20 15:00:00
7	Nguyễn Thị Lan Anh	Mrs. Nguyễn Thu Phương	2024-05-18 16:30:00
8	Nguyễn Minh Quang	Mr. Nguyễn Khánh Duy	2024-07-25 17:00:00
9	Nguyễn Thị Mai	Mr. Nguyễn Khánh Huyền	2024-01-05 09:30:00
10	Nguyễn Thái Sơn	Mr. Nguyễn Thanh Sơn	2024-12-20 11:00:00

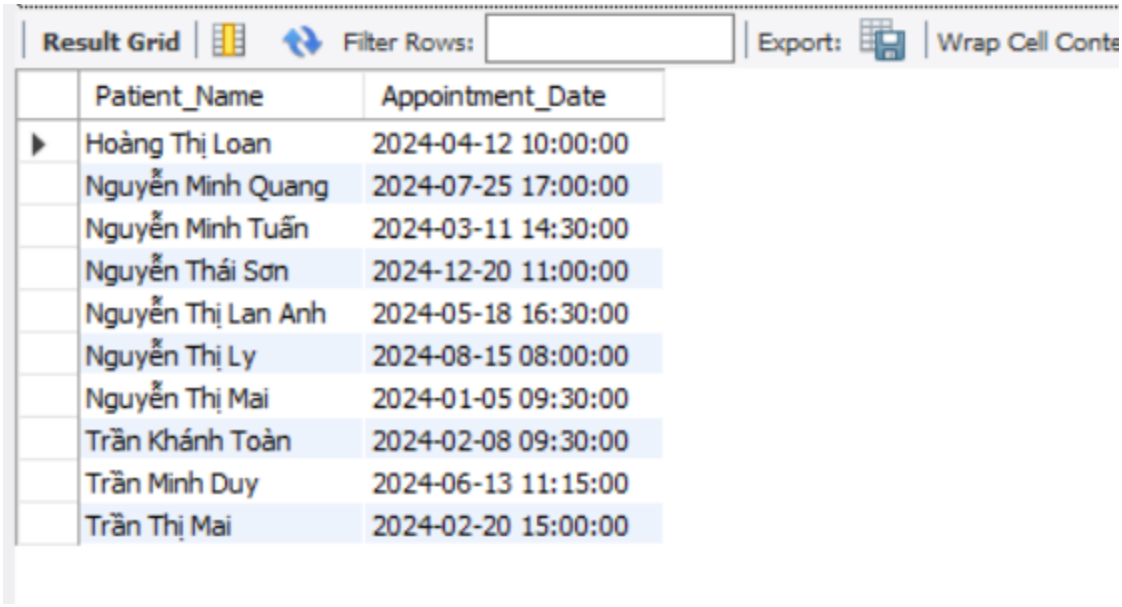
- Truy vấn dùng **OUTER JOIN:**
  - Truy vấn này nhằm liệt kê tất cả bệnh nhân trong hệ thống cùng với ngày hẹn tương ứng. Đối với những bệnh nhân chưa có lịch hẹn, kết quả sẽ hiển thị giá trị mặc định là `"No Appointment"`.
    - **Bảng chính:** `Patients` chứa thông tin về bệnh nhân, bao gồm tên và mã định danh (`PatientID`).
    - **Bảng phụ:** `Appointments` chứa thông tin về lịch hẹn, trong đó `PatientID` là khóa ngoại liên kết với bảng `Patients`.
    - **Phương pháp:**
      - **LEFT OUTER JOIN:** Liên kết bảng `Patients` với bảng `Appointments`, đảm bảo rằng tất cả bản ghi từ bảng `Patients` sẽ được hiển thị, bất kể có khớp với bảng `Appointments` hay không.
      - **COALESCE():** Hàm xử lý giá trị NULL. Nếu bệnh nhân không có lịch hẹn (`AppointmentDate` là NULL), thì kết quả sẽ hiển thị `"No Appointment"`.
      - **ORDER BY:** Sắp xếp kết quả theo thứ tự bảng chữ cái của tên bệnh nhân (`Patient_Name`).
    - **Ý nghĩa:**
      - Truy vấn này cung cấp cái nhìn toàn cảnh về tình trạng lịch hẹn của bệnh nhân.

- Hỗ trợ nhân viên quản lý dễ dàng xác định những bệnh nhân chưa có lịch hẹn để theo dõi và điều phối lịch trình phù hợp.

- Câu lệnh:

```
SELECT p.FullName AS Patient_Name, COALESCE(a.AppointmentDate, 'No Appointment') AS Appointment_Date
FROM Patients p
        LEFT OUTER JOIN Appointments a ON p.PatientID = a.PatientID
ORDER BY Patient_Name;
```

- Kết quả:



Result Grid | Filter Rows: | Export: | Wrap Cell Conte

Patient_Name	Appointment_Date
Hoàng Thị Loan	2024-04-12 10:00:00
Nguyễn Minh Quang	2024-07-25 17:00:00
Nguyễn Minh Tuấn	2024-03-11 14:30:00
Nguyễn Thái Sơn	2024-12-20 11:00:00
Nguyễn Thị Lan Anh	2024-05-18 16:30:00
Nguyễn Thị Ly	2024-08-15 08:00:00
Nguyễn Thị Mai	2024-01-05 09:30:00
Trần Khánh Toàn	2024-02-08 09:30:00
Trần Minh Duy	2024-06-13 11:15:00
Trần Thị Mai	2024-02-20 15:00:00

- Truy vấn SQL trên sẽ **lấy tất cả các bệnh nhân** từ bảng **Patients** và ghép thêm thông tin ngày hẹn từ bảng **Appointments** nếu có.

  - Câu lệnh:

```
select p.FullName AS Patient_Name, a.AppointmentDate
from Patients p
        left outer join Appointments a ON p.PatientID = a.PatientID;
```

  - Kết quả:

	Patient_Name	AppointmentDate
▶	Nguyễn Thị Ly	2024-08-15 08:00:00
	Trần Khánh Toàn	2024-02-08 09:30:00
	Hoàng Thị Loan	2024-04-12 10:00:00
	Trần Minh Duy	2024-06-13 11:15:00
	Nguyễn Minh Tuấn	2024-03-11 14:30:00
	Trần Thị Mai	2024-02-20 15:00:00
	Nguyễn Thị Lan Anh	2024-05-18 16:30:00
	Nguyễn Minh Quang	2024-07-25 17:00:00
	Nguyễn Thị Mai	2024-01-05 09:30:00
	Nguyễn Thái Sơn	2024-12-20 11:00:00

- Truy vấn dùng **SUBQUERY** trong **WHERE:**
  - Truy vấn dưới sử dụng **SUBQUERY** trong mệnh đề **WHERE** để lọc và lấy thông tin các bác sĩ có cuộc hẹn trong một khoảng thời gian từ 20/2/2024 đến 18/5/2024
    - Câu lệnh:

```
select d.FullName
from Doctors d
where d.DoctorID IN (select a.DoctorID
                     from Appointments a
                     where a.AppointmentDate between '2024-02-20' AND '2024-05-18');
```

    - Kết quả:

	FullName
▶	Mrs. Vũ Thị Mai
	Mr. Nguyễn Văn Tuấn
	Mrs. Nguyễn Phương Linh

  - Truy vấn nhằm tìm danh sách các bác sĩ đã tham gia vào các lịch hẹn có liên quan đến kiểm tra sức khỏe (*checkup*). Kết quả sẽ trả về tên và email của các bác sĩ phù hợp.

1. **Bảng chính:**

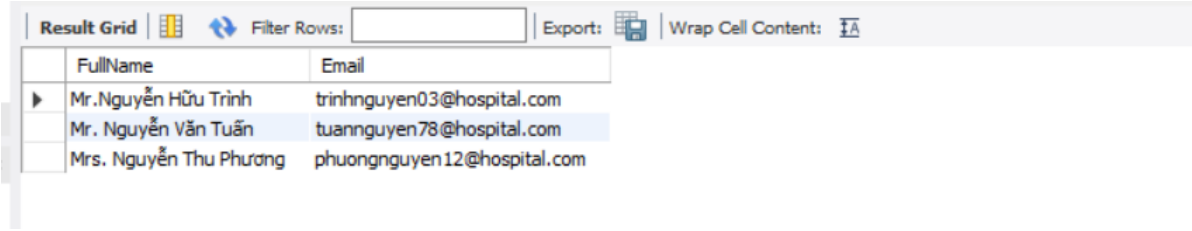
- Bảng `Doctors` chứa thông tin bác sĩ, bao gồm các cột `DoctorID`, `FullName`, và `Email`.

2. **Truy vấn con (subquery):**
   - Truy vấn con thực thi trên bảng `Appointments` để lọc ra các `DoctorID` có lịch hẹn mà lý do (`Reason`) chứa từ khóa `"checkup"`.
   - Điều kiện sử dụng toán tử `LIKE '%checkup%'` để tìm kiếm văn bản chứa cụm từ này ở bất kỳ vị trí nào trong chuỗi.
3. **Truy vấn chính:**
   - Truy vấn chính trên bảng `Doctors` sử dụng kết quả từ subquery để lọc các bác sĩ có `DoctorID` nằm trong danh sách trả về từ subquery.
   - Chỉ trả về các cột `FullName` và `Email`.

- Câu lệnh:

```
SELECT FullName, Email
FROM Doctors
WHERE DoctorID IN (
    SELECT DoctorID
    FROM Appointments
    WHERE Reason LIKE '%checkup%'
);
```

- Kết quả:



- **Ý nghĩa:**

        - Truy vấn này cung cấp thông tin hữu ích để quản lý và đánh giá hoạt động của các bác sĩ liên quan đến kiểm tra sức khỏe.
        - Hỗ trợ xác định các bác sĩ có tần suất tham gia kiểm tra sức khỏe cao để điều phối hợp lý tài nguyên y tế.
- Truy vấn dùng **SUBQUERY** trong **FROM**:
    - Truy vấn SQL trên thực hiện **đếm số lượng bệnh nhân** cho từng bác sĩ trong khoảng thời gian từ 25/7/2024 đến 20/12/2024.
        - Câu lệnh:

```
select d.FullName AS DoctorName, count(a.PatientID) AS Patient_Amount
from Doctors d
        join (select PatientID, DoctorID
              from Appointments
              where AppointmentDate between '2024-07-25' and '2024-12-20') a on d.DoctorID = a.DoctorID
group by d.FullName;
```

        - Kết quả:

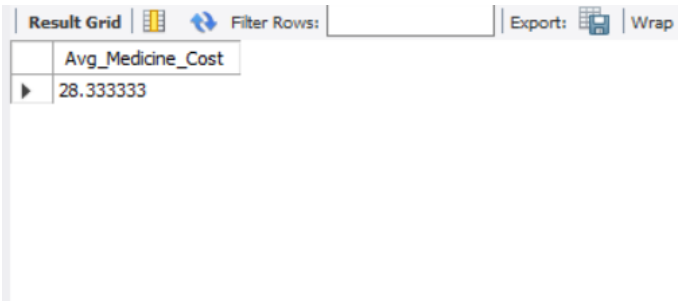
	DoctorName	Patient_Amount
▶	Mr.Nguyễn Hữu Trinh	1
	Mr. Nguyễn Khánh Duy	1

    - Truy vấn nhằm tính **chi phí trung bình của thuốc** trên các hóa đơn, bằng cách sử dụng một **subquery** để tổng hợp chi phí thuốc trong từng hóa đơn trước khi tính trung bình.
    - **Subquery (truy vấn con):**
        - Truy vấn con trong phần `FROM` thực hiện nhiệm vụ tổng hợp chi phí thuốc (`Medicine Fee`) trên từng hóa đơn (`BillID`).
        - **Các bảng tham gia:**
            - `Bills` (`b`): Đại diện cho thông tin hóa đơn.
            - `BillDetails` (`ms`): Chứa chi tiết của từng hóa đơn, bao gồm mô tả và chi phí.

        - **Điều kiện JOIN:** Liên kết hai bảng dựa trên `BillID`, chỉ lấy các bản ghi trong `BillDetails` có mô tả là `'Medicine Fee'`.
        - **Hàm tổng hợp:** `SUM(ms.Amount)` tính tổng chi phí thuốc cho từng hóa đơn.
        - **GROUP BY:** Nhóm kết quả theo `BillID` để tính tổng chi phí thuốc cho từng hóa đơn riêng lẻ.
    - **Truy vấn chính:**
        - Sử dụng kết quả từ subquery (được đặt tên là `MedicineSummary`) để tính trung bình chi phí thuốc trên tất cả các hóa đơn.
        - Hàm `AVG()` được áp dụng trên cột `MedicineCost` từ kết quả subquery.
    - Câu lệnh:

```
SELECT AVG(MedicineCost) AS Avg_Medicine_Cost
FROM (
        SELECT b.BillID, SUM(ms.Amount) AS MedicineCost
        FROM Bills b
                JOIN BillDetails ms ON b.BillID = ms.BillID AND ms.Description = 'Medicine Fee'
        GROUP BY b.BillID
    ) AS MedicineSummary;
```

    - Kết quả:



- Sử dụng **GROUP BY** và hàm tổng hợp**::**
    - Truy vấn SQL này thực hiện việc tính tổng số tiền của các hóa đơn cho mỗi bệnh nhân.
        - Câu lệnh:

```sql
select p.FullName AS PatientName, SUM(b.TotalAmount) AS Total_Amount
from Bills b
        join Patients p on b.PatientID = p.PatientID
group by p.FullName;
```

- Kết quả

PatientName	Total_Amount
Nguyễn Thị Ly	100.00
Trần Khánh Toàn	80.00
Hoàng Thị Loan	70.00
Trần Minh Duy	90.00
Nguyễn Minh Tuấn	50.00
Trần Thị Mai	75.00
Nguyễn Thị Lan Anh	60.00
Nguyễn Minh Quang	110.00
Nguyễn Thị Mai	85.00
Nguyễn Thái Sơn	95.00

- Truy vấn nhằm nhóm các phòng theo loại phòng (`RoomType`) để:

1. Đếm số lượng phòng của từng loại (`Total_Rooms`).
2. Tính tổng sức chứa của các phòng thuộc từng loại (`Total_Capacity`).
3. Sắp xếp kết quả theo tổng sức chứa giảm dần (`DESC`).

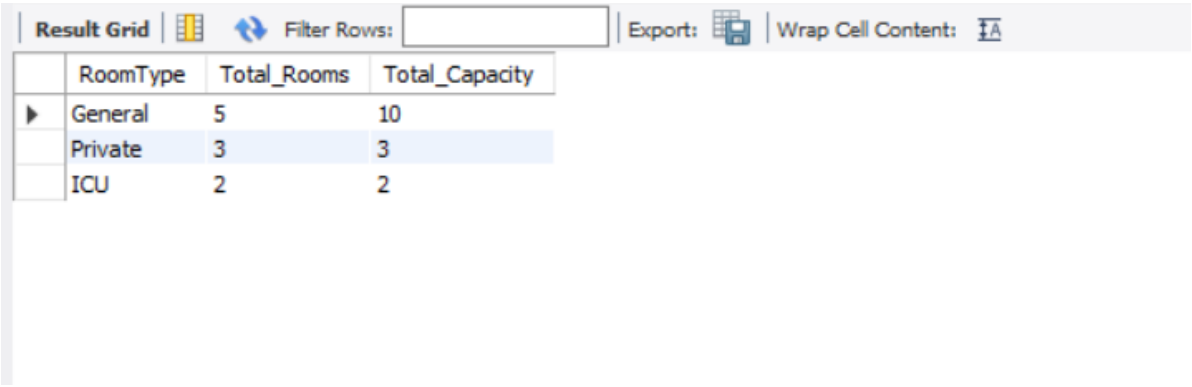
- **Các bảng tham gia:**
  - **Bảng `Rooms` (`r`)**: Lưu thông tin về các phòng trong bệnh viện, bao gồm loại phòng (`RoomType`), mã phòng (`RoomID`), và sức chứa (`Capacity`).
- **Cách thức thực hiện:**
  - **`GROUP BY RoomType`**: Nhóm các phòng lại theo loại (`RoomType`) để thực hiện các phép tính tổng hợp trên từng nhóm.
  - **Hàm tổng hợp:**
    - `COUNT(r.RoomID)`: Đếm số lượng phòng trong mỗi loại và đặt tên là `Total_Rooms`.

- `SUM(r.Capacity)`: Tính tổng sức chứa của các phòng trong mỗi loại và đặt tên là `Total_Capacity`.
- `ORDER BY Total_Capacity DESC`: Sắp xếp kết quả theo tổng sức chứa giảm dần, giúp loại phòng có sức chứa lớn nhất hiển thị trước.

- Câu lệnh:

```
SELECT r.RoomType, COUNT(r.RoomID) AS Total_Rooms, SUM(r.Capacity) AS Total_Capacity
FROM Rooms r
GROUP BY r.RoomType
ORDER BY Total_Capacity DESC;
```

- Kết quả:



- **Ý nghĩa:**
  - Truy vấn này giúp quản lý bệnh viện nắm bắt tổng quan về phân bổ phòng và sức chứa, từ đó tối ưu hóa việc sử dụng các loại phòng phù hợp với nhu cầu bệnh nhân.

- Truy vấn này nhằm mục đích cung cấp thông tin tổng hợp về bệnh nhân, bác sĩ, các phòng và chi phí của các dịch vụ y tế đã được thanh toán. Các thông tin chi tiết bao gồm:
  - Tên bệnh nhân, ngày sinh, giới tính, số điện thoại và email.
  - Tên bác sĩ khám cho bệnh nhân.
  - Số phòng bệnh nhân được phân công.
  - Mã hóa đơn, tổng số tiền hóa đơn, và trạng thái thanh toán.
  - Tổng chi phí dịch vụ và thuốc đã sử dụng.

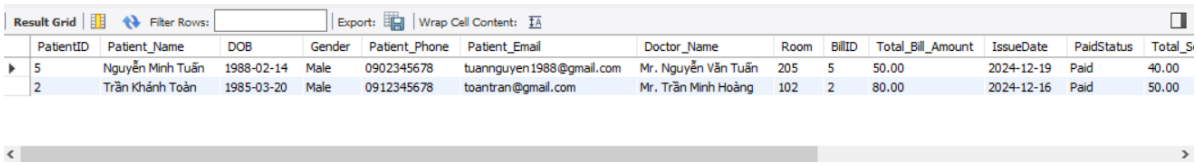
- Tổng số đơn thuốc và số lần hẹn khám của bệnh nhân.
- Tổng chi phí dịch vụ cộng với chi phí thuốc (GrandTotal).
- **Các bảng tham gia:**
  - **Patients (p):** Bảng chứa thông tin về bệnh nhân, bao gồm mã bệnh nhân (PatientID), tên (FullName), ngày sinh (DateOfBirth), giới tính (Gender), số điện thoại (PhoneNumber), và email.
  - **Bills (b):** Bảng chứa thông tin về hóa đơn, bao gồm mã hóa đơn (BillID), tổng số tiền (TotalAmount), ngày lập hóa đơn (IssueDate), và trạng thái thanh toán (PaidStatus).
  - **BillDetails (bs, ms):** Bảng chứa thông tin chi tiết về các chi phí dịch vụ (Consultation Fee) và thuốc (Medicine Fee), được tính tổng hợp.
  - **Appointments (a):** Bảng chứa thông tin các cuộc hẹn, bao gồm mã cuộc hẹn (AppointmentID) và thông tin về bác sĩ (DoctorID) và phòng (RoomID).
  - **Doctors (d):** Bảng chứa thông tin về bác sĩ khám chữa cho bệnh nhân, bao gồm tên bác sĩ (FullName).
  - **Rooms (r):** Bảng chứa thông tin về phòng khám của bệnh nhân (RoomNumber).
  - **PrescriptionDetails (pd):** Bảng chứa thông tin về đơn thuốc, được sử dụng để tính tổng số đơn thuốc.
- **Cách thức thực hiện:**
  - **JOIN các bảng:** Các bảng liên kết với nhau để thu thập thông tin cần thiết về bệnh nhân, bác sĩ, phòng, hóa đơn và các dịch vụ thuốc.
  - **SUM(bs.Amount) và SUM(ms.Amount):** Tính tổng chi phí dịch vụ (Consultation Fee) và thuốc (Medicine Fee) của từng bệnh nhân.
  - **COUNT(pd.PrescriptionID) và COUNT(a.AppointmentID):** Đếm số lượng đơn thuốc và số lần hẹn khám của bệnh nhân.
  - **GROUP BY:** Các bệnh nhân và hóa đơn được nhóm lại để tính toán các thông tin tổng hợp.

- **ORDER BY GrandTotal DESC:** Sắp xếp kết quả theo tổng chi phí dịch vụ và thuốc (GrandTotal) giảm dần.

- Câu lệnh:

```
SELECT p.PatientID,
       p.FullName AS Patient_Name,
       p.DateOfBirth AS DOB,
       p.Gender,
       p.PhoneNumber AS Patient_Phone,
       p.Email AS Patient_Email,
       d.FullName AS Doctor_Name,
       r.RoomNumber AS Room,
       b.BillID,
       b.TotalAmount AS Total_Bill_Amount,
       b.IssueDate,
       b.PaidStatus,
       SUM(bs.Amount) AS Total_ServiceCost,
       SUM(ms.Amount) AS Total_MedicineCost,
       COUNT(pd.PrescriptionID) AS Prescription_Count,
       COUNT(a.AppointmentID) AS Appointment_Count,
       (SUM(bs.Amount) + SUM(ms.Amount)) AS GrandTotal
FROM Patients p
JOIN Bills b ON p.PatientID = b.PatientID
JOIN BillDetails bs ON b.BillID = bs.BillID AND bs.Description = 'Consultation Fee'
JOIN BillDetails ms ON b.BillID = ms.BillID AND ms.Description = 'Medicine Fee'
JOIN Appointments a ON p.PatientID = a.PatientID
JOIN Doctors d ON a.DoctorID = d.DoctorID
JOIN Rooms r ON a.AppointmentID = r.RoomID
LEFT JOIN PrescriptionDetails pd ON a.AppointmentID = pd.PrescriptionID
WHERE b.PaidStatus = 'Paid'
GROUP BY p.PatientID, b.BillID, a.AppointmentID
ORDER BY GrandTotal DESC;
```

- Kết quả:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PatientID	Patient_Name	DOB	Gender	Patient_Phone	Patient_Email	Doctor_Name	Room	BillID	Total_Bill_Amount	IssueDate	PaidStatus	Total_S
5	Nguyễn Minh Tuấn	1988-02-14	Male	0902345678	tuannguyen1988@gmail.com	Mr. Nguyễn Văn Tuấn	205	5	50.00	2024-12-19	Paid	40.00
2	Trần Khánh Toàn	1985-03-20	Male	0912345678	toantran@gmail.com	Mr. Trần Minh Hoàng	102	2	80.00	2024-12-16	Paid	50.00

- **Ý nghĩa:**

Truy vấn này cung cấp cái nhìn tổng quan về các bệnh nhân và các dịch vụ y tế mà họ đã sử dụng. Các thông tin tổng hợp về chi phí và số lần hẹn khám sẽ giúp bệnh viện quản lý tài chính và tối ưu hóa quá trình chăm sóc bệnh nhân. Các dữ liệu này có thể được sử dụng để phân tích chi phí, tối ưu hóa nguồn lực, và cải thiện dịch vụ chăm sóc sức khỏe cho bệnh nhân.

IX. Transaction using Rollback

Transaction1:

- **Chèn hóa đơn cho bệnh nhân:**

  Một hóa đơn mới được thêm vào bảng `Bills` với thông tin bệnh nhân có mã `1`, tổng số tiền là `100.00`, ngày lập hóa đơn `2024-12-14` và trạng thái thanh toán là `Unpaid`.
  Sau đó,
  `@LastBillID` được sử dụng để lưu mã hóa đơn vừa chèn.

- **Chèn chi tiết hóa đơn:**

  Các chi tiết hóa đơn bao gồm "Consultation Fee" với giá trị `50.00` và "Medicine Fee" với giá trị `50.00` được chèn vào bảng `BillDetails` với mã hóa đơn `@LastBillID`.

- **Chèn thêm phí quá mức (Overcharge Fee):**

  Một phí "Overcharge Fee" với giá trị `200.00` được thêm vào bảng `BillDetails` cùng mã hóa đơn `@LastBillID`.

- **Rollback giao dịch:**

  Cuối cùng, giao dịch này bị rollback, nghĩa là tất cả các thao tác trên bảng `Bills` và `BillDetails` sẽ không được lưu vào cơ sở dữ liệu, không có thay đổi thực sự.

- Câu lệnh:

```
START TRANSACTION;
INSERT INTO Bills (PatientID, TotalAmount, IssueDate, PaidStatus)
VALUES (1, 100.00, '2024-12-14', 'Unpaid');
SET @LastBillID = LAST_INSERT_ID();

INSERT INTO BillDetails (BillID, Description, Amount)
VALUES (@LastBillID, 'Consultation Fee', 50.00),
       (@LastBillID, 'Medicine Fee', 50.00);

INSERT INTO BillDetails (BillID, Description, Amount)
VALUES (@LastBillID, 'Overcharge Fee', 200.00);

COMMIT;
ROLLBACK;
```

Transaction 2

- **Chèn bệnh nhân mới:**

  Một bệnh nhân mới có thông tin như sau: Tên là "Trần Văn Nam", ngày sinh `1990-01-15`, giới tính `Male`, địa chỉ `Hà Nội, Việt Nam`, số điện thoại `0988112233`, và email `namtran@gmail.com` được thêm vào bảng `Patients`. `@NewPatientID` được sử dụng để lưu mã bệnh nhân vừa chèn.

- **Chèn cuộc hẹn cho bệnh nhân:**

  Một cuộc hẹn mới cho bệnh nhân vừa chèn được thêm vào bảng `Appointments` với bác sĩ có mã `1`, ngày hẹn là `2024-12-20 10:00:00`, lý do hẹn là "Routine checkup".

- **Chèn hóa đơn cho bệnh nhân mới:**

  Một hóa đơn mới được thêm vào bảng `Bills` với bệnh nhân vừa tạo, tổng số tiền là `150.00`, ngày lập hóa đơn `2024-12-20`, và trạng thái thanh toán là `Unpaid`.

- **Rollback giao dịch:**

  Giao dịch này cũng bị rollback, do đó tất cả các thao tác trên bảng `Patients`, `Appointments`, và `Bills` không được lưu vào cơ sở dữ liệu.

- Câu lệnh:

```sql
START TRANSACTION;

INSERT INTO Patients (FullName, DateOfBirth, Gender, Address, PhoneNumber, Email, RoomID)
VALUES ('Trần Văn Nam', '1990-01-15', 'Male', 'Hà Nội, Việt Nam', '0988112233', 'namtran@gmail.com', 3);

SET @NewPatientID = LAST_INSERT_ID();

INSERT INTO Appointments (PatientID, DoctorID, AppointmentDate, Reason)
VALUES (@NewPatientID, 1, '2024-12-20 10:00:00', 'Routine checkup');

INSERT INTO Bills (PatientID, TotalAmount, IssueDate, PaidStatus)
VALUES (@NewPatientID, 150.00, '2024-12-20', 'Unpaid');
ROLLBACK;
```

Ý nghĩa:

1. **Giao dịch với trạng thái Rollback:**

   Việc sử dụng `ROLLBACK` trong cả hai giao dịch đảm bảo rằng các thay đổi tạm thời đối với cơ sở dữ liệu không được lưu lại. Điều này rất hữu ích trong trường hợp cần kiểm tra tính chính xác của dữ liệu trước khi quyết định lưu trữ hoặc khi gặp lỗi trong quá trình thao tác.

2. **Kiểm soát và nhất quán dữ liệu:**

   Giao dịch giúp duy trì tính nhất quán của cơ sở dữ liệu bằng cách đảm bảo rằng các thao tác chèn vào bảng `Bills` và `BillDetails`, cũng như bảng `Patients` và `Appointments`, chỉ được thực hiện hoàn chỉnh nếu không có lỗi nào xảy ra. Nếu có lỗi hoặc nếu muốn huỷ thao tác, việc rollback sẽ đưa cơ sở dữ liệu về trạng thái ban đầu.

Kết luận:

Các thao tác SQL đã thực hiện quản lý việc thêm dữ liệu một cách có kiểm soát thông qua giao dịch. Tuy nhiên, do việc rollback, không có thay đổi thực sự nào được lưu vào cơ sở dữ liệu. Đây là một phương pháp quan trọng trong việc phát triển và kiểm thử các thao tác với cơ sở dữ liệu, đảm bảo tính toàn vẹn và nhất quán.

X. Trigger

TRIGGER CẬP NHẬT TÔNG SỐ TIỀN SAU KHI XÓA CHI TIẾT HÓA ĐƠN

Mục tiêu thực hiện trigger:
Trigger này được tạo ra để tự động cập nhật tổng số tiền (`TotalAmount`) trong bảng `Bills` mỗi khi có một bản ghi bị xóa trong bảng `BillDetails`. Mục đích của việc này là đảm bảo rằng khi một chi tiết hóa đơn bị xóa, tổng số tiền của hóa đơn (ở bảng `Bills`) sẽ được điều chỉnh tương ứng.

Chi tiết kỹ thuật:

1. **Thao tác thực hiện:**
    - **Loại Trigger:** `AFTER DELETE`
    - Trigger này được thiết lập để thực hiện ngay sau khi một bản ghi bị xóa trong bảng `BillDetails`.
    - **Hoạt động:**Khi một bản ghi bị xóa trong bảng `BillDetails`, trigger sẽ được kích hoạt và thực hiện các bước sau:
        - Trigger sẽ tính lại tổng số tiền (`TotalAmount`) cho hóa đơn trong bảng `Bills` bằng cách sử dụng hàm `SUM(Amount)` trên tất cả các bản ghi còn lại của cùng một hóa đơn (dựa trên `BillID`).
        - Sau khi tính toán, tổng số tiền mới sẽ được cập nhật lại vào trường `TotalAmount` trong bảng `Bills` cho đúng với số tiền của hóa đơn sau khi một chi tiết bị xóa.

Câu lệnh:

```
CREATE TRIGGER UpdateTotalAmountAfterDelete
AFTER DELETE ON BillDetails
FOR EACH ROW
BEGIN
    UPDATE Bills
    SET TotalAmount = (
        SELECT SUM(Amount)
        FROM BillDetails
        WHERE BillID = OLD.BillID
    )
    WHERE BillID = OLD.BillID;
END $$
```

- **AFTER DELETE ON BillDetails:**

Trigger này sẽ được kích hoạt ngay sau khi một bản ghi bị xóa khỏi bảng `BillDetails`.

- **Cập nhật tổng số tiền:**

  Câu lệnh `UPDATE Bills` sẽ cập nhật lại trường `TotalAmount` trong bảng `Bills` với giá trị mới, tính từ các bản ghi còn lại trong bảng `BillDetails` cho cùng một `BillID` của bản ghi vừa bị xóa.

- **OLD.BillID:**

  `OLD.BillID` là giá trị `BillID` của bản ghi bị xóa trong bảng `BillDetails`. Trigger sử dụng giá trị này để cập nhật tổng số tiền của hóa đơn liên quan trong bảng `Bills`.

  Ý nghĩa:

  1. **Tự động hóa việc cập nhật dữ liệu:**

     Trigger này giúp tự động điều chỉnh tổng số tiền trong bảng `Bills` mỗi khi có thay đổi về các chi tiết hóa đơn trong bảng `BillDetails`. Điều này giúp duy trì tính nhất quán của dữ liệu mà không cần can thiệp thủ công.

  2. **Duy trì tính toàn vẹn của dữ liệu:**

     Khi chi tiết hóa đơn bị xóa, việc cập nhật lại tổng số tiền giúp đảm bảo rằng số tiền trong bảng `Bills` luôn chính xác, phản ánh đúng các chi tiết hóa đơn còn lại.

  3. **Giảm thiểu sai sót:**

     Bằng cách sử dụng trigger, hệ thống tự động thực hiện các tính toán và cập nhật mà không cần sự can thiệp từ người dùng, giúp giảm thiểu khả năng sai sót hoặc quên cập nhật.

TRIGGER CẬP NHẬT TỔNG SỐ TIỀN SAU KHI CẬP NHẬT CHI TIẾT HÓA ĐƠN

Mục tiêu thực hiện trigger:
Trigger này được tạo ra để tự động cập nhật tổng số tiền (`TotalAmount`) trong bảng `Bills` mỗi khi có một bản ghi trong bảng `BillDetails` bị cập nhật. Mục đích của trigger là đảm bảo rằng tổng số tiền của hóa đơn trong bảng `Bills` sẽ luôn chính xác sau khi có bất kỳ thay đổi nào đối với các chi tiết hóa đơn.

Chi tiết kỹ thuật:

1. **Thao tác thực hiện:**
    - **Loại Trigger:** `AFTER UPDATE`
    - Trigger này sẽ được kích hoạt ngay sau khi một bản ghi trong bảng `BillDetails` bị cập nhật.
    - **Hoạt động:**Khi một bản ghi trong bảng `BillDetails` bị thay đổi (ví dụ: thay đổi số tiền của dịch vụ hoặc thuốc), trigger sẽ thực hiện các bước sau:
        - Trigger tính lại tổng số tiền (`TotalAmount`) của hóa đơn trong bảng `Bills` bằng cách sử dụng hàm `SUM(Amount)` để cộng tổng số tiền của tất cả các chi tiết hóa đơn còn lại với `BillID` tương ứng.
        - Sau khi tính toán tổng số tiền mới, trigger sẽ cập nhật lại trường `TotalAmount` trong bảng `Bills` cho hóa đơn có `BillID` tương ứng.

Câu lệnh:

```
CREATE TRIGGER UpdateTotalAmountAfterUpdate
AFTER UPDATE ON BillDetails
FOR EACH ROW
BEGIN
    UPDATE Bills
    SET TotalAmount = (
        SELECT SUM(Amount)
        FROM BillDetails
        WHERE BillID = NEW.BillID
    )
    WHERE BillID = NEW.BillID;
END $$

DELIMITER ;
```

- **AFTER UPDATE ON BillDetails:**

  Trigger này sẽ được kích hoạt ngay sau khi có sự thay đổi (cập nhật) trong bảng `BillDetails`.

- **Cập nhật tổng số tiền:**

  Câu lệnh `UPDATE Bills` sẽ cập nhật lại trường `TotalAmount` trong bảng `Bills` bằng cách tính tổng số tiền của tất cả các bản ghi còn lại trong bảng `BillDetails` cho cùng một `BillID` của bản ghi đã được cập nhật.

- **NEW.BillID:**

`NEW.BillID` là giá trị `BillID` của bản ghi trong bảng `BillDetails` sau khi cập nhật. Trigger sử dụng giá trị này để tìm và cập nhật tổng số tiền trong bảng `Bills`.

Ý nghĩa:

1. **Tự động hóa việc cập nhật dữ liệu:**

   Trigger này giúp tự động điều chỉnh tổng số tiền trong bảng `Bills` ngay khi có sự thay đổi trong bảng `BillDetails`, đảm bảo rằng dữ liệu luôn được cập nhật mà không cần sự can thiệp thủ công.

2. **Duy trì tính toàn vẹn của dữ liệu:**

   Khi một chi tiết hóa đơn được cập nhật (ví dụ, thay đổi số tiền), tổng số tiền trong bảng `Bills` sẽ được tính lại để phản ánh chính xác sự thay đổi này. Điều này giúp duy trì tính chính xác và nhất quán giữa các bảng liên quan.

3. **Giảm thiểu sai sót:**

   Việc sử dụng trigger giúp tránh được lỗi mà người quản lý có thể quên cập nhật tổng số tiền trong bảng `Bills` sau khi thay đổi chi tiết hóa đơn. Trigger sẽ tự động thực hiện tính toán và cập nhật dữ liệu một cách chính xác.

TRIGGER CẬP NHẬT TỔNG SỐ TIỀN SAU KHI THÊM CHI TIẾT HÓA ĐƠN

Mục tiêu thực hiện trigger:
Trigger này được tạo ra để tự động cập nhật tổng số tiền (`TotalAmount`) trong bảng `Bills` mỗi khi một chi tiết hóa đơn mới được thêm vào bảng `BillDetails`. Mục đích của trigger là đảm bảo rằng tổng số tiền của hóa đơn trong bảng `Bills` luôn được cập nhật chính xác sau khi có thêm bất kỳ chi tiết hóa đơn nào.

Chi tiết kỹ thuật:

1. **Thao tác thực hiện:**

   - **Loại Trigger:** `AFTER INSERT`
   - Trigger này sẽ được kích hoạt ngay sau khi một bản ghi mới được chèn vào bảng `BillDetails`.

- **Hoạt động:**Sau khi thêm một chi tiết hóa đơn mới vào bảng `BillDetails`, trigger thực hiện các bước sau:
  - Trigger tính lại tổng số tiền (`TotalAmount`) của hóa đơn trong bảng `Bills` bằng cách sử dụng hàm `SUM(Amount)` để cộng tổng số tiền của tất cả các chi tiết hóa đơn (bao gồm cả chi tiết mới được thêm vào) với `BillID` tương ứng.
  - Sau khi tính toán tổng số tiền mới, trigger sẽ cập nhật lại trường `TotalAmount` trong bảng `Bills` cho hóa đơn có `BillID` tương ứng.

Câu lệnh:

```
DELIMITER $$
CREATE TRIGGER UpdateBillTotalAfterInsert
AFTER INSERT ON BillDetails
FOR EACH ROW
BEGIN
    UPDATE Bills
    SET TotalAmount = (
        SELECT SUM(Amount)
        FROM BillDetails
        WHERE BillID = NEW.BillID
    )
    WHERE BillID = NEW.BillID;
END $$
DELIMITER ;
```

- **AFTER INSERT ON BillDetails:**

  Trigger này sẽ được kích hoạt ngay sau khi một bản ghi mới được thêm vào bảng `BillDetails`.

- **Cập nhật tổng số tiền:**

  Câu lệnh `UPDATE Bills` sẽ cập nhật lại trường `TotalAmount` trong bảng `Bills` bằng cách tính tổng số tiền của tất cả các bản ghi trong bảng `BillDetails` có cùng `BillID`. Hàm `SUM(Amount)` sẽ tính tổng số tiền của tất cả chi tiết hóa đơn của một hóa đơn cụ thể.

- **NEW.BillID:**

  `NEW.BillID` là giá trị `BillID` của bản ghi vừa được chèn vào bảng `BillDetails`. Trigger sử dụng giá trị này để tìm và cập nhật tổng số tiền trong bảng `Bills`.

Trigger `UpdateBillTotalAfterInsert` mang lại hiệu quả cao trong việc duy trì sự chính xác và nhất quán của dữ liệu giữa các bảng. Việc tự động cập nhật tổng số tiền

của hóa đơn giúp tối ưu hóa quy trình quản lý hóa đơn, tiết kiệm thời gian và giảm thiểu các lỗi do thao tác thủ công.

XI. Procedure

1.Procedure AddDoctor

Mục tiêu:
Thủ tục
`AddDoctor` được thiết kế để thêm thông tin bác sĩ mới vào cơ sở dữ liệu. Bằng cách sử dụng thủ tục này, người dùng có thể dễ dàng nhập các thông tin cần thiết của bác sĩ vào bảng `Doctors`.

Chi tiết kỹ thuật:

- **Tham số đầu vào**:
  - `p_FullName`: Tên bác sĩ (kiểu dữ liệu `VARCHAR(100)`).
  - `p_Specialty`: Chuyên khoa của bác sĩ (kiểu dữ liệu `VARCHAR(100)`).
  - `p_PhoneNumber`: Số điện thoại liên lạc của bác sĩ (kiểu dữ liệu `VARCHAR(15)`).
  - `p_Email`: Địa chỉ email của bác sĩ (kiểu dữ liệu `VARCHAR(100)`).
  - `p_HireDate`: Ngày bắt đầu làm việc của bác sĩ (kiểu dữ liệu `DATE`).
- **Quá trình thực hiện**:
  - Thủ tục thực hiện việc chèn một bản ghi mới vào bảng `Doctors` với các thông tin được cung cấp từ tham số đầu vào.
- **Câu lệnh:**

```
CREATE PROCEDURE AddDoctor (
    IN p_FullName VARCHAR(100),
    IN p_Specialty VARCHAR(100),
    IN p_PhoneNumber VARCHAR(15),
    IN p_Email VARCHAR(100),
    IN p_HireDate DATE
)
BEGIN
    INSERT INTO Doctors (FullName, Specialty, PhoneNumber, Email, HireDate)
    VALUES (p_FullName, p_Specialty, p_PhoneNumber, p_Email, p_HireDate);
END $$
```

- **Ý nghĩa**:
  Thủ tục này giúp tự động hóa quá trình thêm bác sĩ vào cơ sở dữ liệu, đảm bảo rằng các bác sĩ mới được ghi nhận đầy đủ thông tin và tránh việc nhập liệu thủ công nhiều lần.

---

2. Procedure AddPatient

Mục tiêu:
Thủ tục
`AddPatient` được sử dụng để thêm thông tin bệnh nhân mới vào cơ sở dữ liệu, giúp quy trình nhập liệu trở nên tự động và dễ dàng hơn.

Chi tiết kỹ thuật:

- **Tham số đầu vào**:
  - `p_FullName`: Tên bệnh nhân (kiểu dữ liệu `VARCHAR(100)`).
  - `p_DateOfBirth`: Ngày sinh của bệnh nhân (kiểu dữ liệu `DATE`).
  - `p_Gender`: Giới tính của bệnh nhân, có thể là `Male`, `Female` hoặc `Other` (kiểu dữ liệu `ENUM`).
  - `p_Address`: Địa chỉ của bệnh nhân (kiểu dữ liệu `VARCHAR(255)`).
  - `p_PhoneNumber`: Số điện thoại của bệnh nhân (kiểu dữ liệu `VARCHAR(15)`).
  - `p_Email`: Địa chỉ email của bệnh nhân (kiểu dữ liệu `VARCHAR(100)`).

- **Quá trình thực hiện**:
  - Thủ tục này chèn một bản ghi mới vào bảng `Patients` với các thông tin được cung cấp từ tham số đầu vào.
- **Câu lệnh:**

```
CREATE PROCEDURE AddPatient (
    IN p_FullName VARCHAR(100),
    IN p_DateOfBirth DATE,
    IN p_Gender ENUM('Male', 'Female', 'Other'),
    IN p_Address VARCHAR(255),
    IN p_PhoneNumber VARCHAR(15),
    IN p_Email VARCHAR(100),
    IN p_Room INT
)
BEGIN
    INSERT INTO Patients (FullName, DateOfBirth, Gender, Address, PhoneNumber, Email, RoomID)
    VALUES (p_FullName, p_DateOfBirth, p_Gender, p_Address, p_PhoneNumber, p_Email, p_Room);
END $$
```

- **Ý nghĩa**:
  Việc sử dụng thủ tục này giúp đơn giản hóa quá trình nhập thông tin bệnh nhân mới vào cơ sở dữ liệu, giảm thiểu sai sót do nhập liệu thủ công và tăng tính nhất quán trong cơ sở dữ liệu.

---

3. Procedure GetUnpaidBills

Mục tiêu:
Thủ tục
`GetUnpaidBills` được thiết kế để tính tổng số tiền chưa thanh toán từ bảng hóa đơn và trả về kết quả. Đây là một công cụ hữu ích để theo dõi tổng số tiền còn nợ trong hệ thống.

Chi tiết kỹ thuật:

- **Tham số đầu ra**:
  - `p_TotalUnpaid`: Tổng số tiền chưa thanh toán (kiểu dữ liệu `DECIMAL(10, 2)`).
- **Quá trình thực hiện**:

- Thủ tục này thực hiện một phép tính tổng số tiền từ cột `TotalAmount` của bảng `Bills` với điều kiện là hóa đơn chưa được thanh toán (`PaidStatus = 'Unpaid'`).
- Kết quả tổng số tiền chưa thanh toán được lưu vào tham số đầu ra `p_TotalUnpaid`.

- **Câu lệnh:**

```
CREATE PROCEDURE GetUnpaidBills (
    OUT p_TotalUnpaid DECIMAL(10, 2)
)
BEGIN
    SELECT SUM(TotalAmount)
    INTO p_TotalUnpaid
    FROM Bills
    WHERE PaidStatus = 'Unpaid';
END $$
DELIMITER ;
```

- **Ý nghĩa**:
  Thủ tục này rất hữu ích cho việc quản lý tài chính trong bệnh viện hoặc cơ sở y tế, cho phép nhanh chóng tính toán và theo dõi số tiền mà bệnh viện vẫn chưa thu được từ các hóa đơn chưa thanh toán. Điều này giúp bộ phận kế toán hoặc quản lý dễ dàng nhận diện và theo dõi các khoản nợ.

4. Thủ tục CreateBill

Mục tiêu:

Thủ tục `CreateBill` được thiết kế để tạo một hóa đơn mới cho bệnh nhân, đồng thời thêm các chi tiết hóa đơn liên quan (mô tả và số tiền) vào bảng `BillDetails`. Thủ tục này giúp quy trình tạo hóa đơn và chi tiết hóa đơn trở nên tự động, dễ dàng và chính xác.

Chi tiết kỹ thuật:

- **Tham số đầu vào**:
  - `p_PatientID`: ID bệnh nhân (kiểu dữ liệu `INT`).
  - `p_TotalAmount`: Tổng số tiền của hóa đơn (kiểu dữ liệu `DECIMAL(10, 2)`).

- `p_IssueDate`: Ngày phát hành hóa đơn (kiểu dữ liệu `DATE`).
- `p_PaidStatus`: Trạng thái thanh toán của hóa đơn, có thể là 'Paid' hoặc 'Unpaid' (kiểu dữ liệu `ENUM`).
- `p_Description1`: Mô tả chi tiết hóa đơn thứ nhất (kiểu dữ liệu `VARCHAR(255)`).
- `p_Amount1`: Số tiền của chi tiết hóa đơn thứ nhất (kiểu dữ liệu `DECIMAL(10, 2)`).
- `p_Description2`: Mô tả chi tiết hóa đơn thứ hai (kiểu dữ liệu `VARCHAR(255)`).
- `p_Amount2`: Số tiền của chi tiết hóa đơn thứ hai (kiểu dữ liệu `DECIMAL(10, 2)`).

- **Quá trình thực hiện**:
  - Thủ tục này thực hiện việc chèn thông tin hóa đơn vào bảng `Bills` trước tiên, với các tham số bệnh nhân, tổng số tiền, ngày phát hành và trạng thái thanh toán.
  - Sau khi thêm hóa đơn vào bảng `Bills`, thủ tục lấy ID của hóa đơn mới tạo (`LAST_INSERT_ID()`) và lưu vào biến `newBillID`.
  - Tiếp theo, thủ tục chèn các chi tiết hóa đơn vào bảng `BillDetails` với `BillID` là ID của hóa đơn vừa được tạo, cùng với mô tả và số tiền của các chi tiết hóa đơn.
- **Câu lệnh:**

```
CREATE PROCEDURE CreateBill(
    IN p_PatientID INT,
    IN p_TotalAmount DECIMAL(10, 2),
    IN p_IssueDate DATE,
    IN p_PaidStatus ENUM('Paid', 'Unpaid'),
    IN p_Description1 VARCHAR(255),
    IN p_Amount1 DECIMAL(10, 2),
    IN p_Description2 VARCHAR(255),
    IN p_Amount2 DECIMAL(10, 2)
)
BEGIN
    DECLARE newBillID INT;

    INSERT INTO Bills (PatientID, TotalAmount, IssueDate, PaidStatus)
    VALUES (p_PatientID, p_TotalAmount, p_IssueDate, p_PaidStatus);

    SET newBillID = LAST_INSERT_ID();
    INSERT INTO BillDetails (BillID, Description, Amount)
    VALUES (newBillID, p_Description1, p_Amount1),
           (newBillID, p_Description2, p_Amount2);

END $$
```

- **Ý nghĩa**:
  Thủ tục
  `CreateBill` giúp tự động hóa quá trình tạo hóa đơn và chi tiết hóa đơn cho bệnh nhân, giảm thiểu việc nhập liệu thủ công và đảm bảo tính nhất quán trong cơ sở dữ liệu. Bằng cách này, khi bệnh nhân nhận được hóa đơn, các thông tin về số tiền, mô tả và trạng thái thanh toán sẽ được quản lý một cách chính xác và hiệu quả.

Kết luận chung:

Các thủ tục trên giúp tự động hóa và tối ưu hóa quy trình nhập liệu và tính toán trong cơ sở dữ liệu. Bằng việc sử dụng các thủ tục này, hệ thống sẽ trở nên hiệu quả hơn trong việc quản lý thông tin bác sĩ, bệnh nhân, cũng như theo dõi các hóa đơn chưa thanh toán, giúp giảm thiểu sai sót và tiết kiệm thời gian quản lý.

XII. Kết luận

Chương trình quản lý bệnh viện giúp tự động hóa các quy trình trong bệnh viện cũng như việc quản lý nhập, xuất thông tin bệnh nhân và dữ liệu y tế một cách chính xác, khoa học, làm tăng năng suất và hiệu quả công việc. Trên cơ sở hỗ trợ lãnh đạo trong việc quản lý các hoạt động của bệnh viện một cách tự động hóa, chương trình giúp tránh được những sai sót và hạn chế mà phương pháp quản lý thủ công gây ra, đáp ứng kịp thời yêu cầu lãnh đạo, chỉ đạo của cấp trên trong mọi tình huống.

Chương trình quản lý bệnh viện hoàn thành tương đối tốt nhiệm vụ đáp ứng nhu cầu cập nhật, truy vấn thông tin nhanh chóng và giải quyết việc tổ chức nhất quán cơ sở dữ liệu trong lưu trữ thông tin y tế. Tuy nhiên, chương trình vẫn còn một số nhược điểm, thiếu sót cần được khắc phục và hoàn thiện trong thời gian tới.