ĐẠI HỌC BÁCH KHOA HÀ NỘI

Bài tập lớn

Thực hành Cơ sở dữ liệu

Thiết kế và xây dựng hệ thống quản lí bệnh án, bệnh nhân trong bệnh viện

Giảng viên hướng dẫn : TS. Trần Văn Đặng

Khoa : Khoa học máy tính

Nhóm sinh viên thực hiện : Trần Thị Phương Thảo - 20225766

(Nhóm 13) Lý Công Tiến - 20225934

Nguyễn Trọng Hinh - 20225842

Trường : Công nghệ Thông tin và Truyền thông

HÀ NỘI, 06/2022

MỤC LỤC

DANH MỤC HÌNH VẼ

CHƯƠNG 1. Giới thiệu về dự án

1.1 Giới thiệu về đề tài

1.2 Mở đầu

Trong bối cảnh hiện đại hóa và số hóa toàn cầu, ngành y tế không ngừng nỗ lực cải tiến để nâng cao chất lượng dịch vụ chăm sóc sức khỏe. Việc quản lý bệnh án và thông tin bệnh nhân một cách hiệu quả là một yếu tố then chốt giúp cải thiện hiệu quả điều trị và chăm sóc bệnh nhân. Tuy nhiên, nhiều bệnh viện và cơ sở y tế vẫn đang sử dụng các phương pháp quản lý truyền thống, dựa trên giấy tờ hoặc các hệ thống thông tin rời rạc, thiếu tính đồng bộ. Điều này không chỉ gây ra lãng phí nguồn lực mà còn tiềm ẩn nhiều rủi ro liên quan đến sai sót dữ liệu và bảo mật thông tin.

Đề tài này nhằm thiết kế và xây dựng một cơ sở dữ liệu (CSDL) hiện đại và toàn diện, phục vụ cho việc quản lý bệnh án, thông tin bệnh nhân và hoạt động của bệnh viện. Hệ thống CSDL này sẽ giúp lưu trữ, quản lý và truy xuất thông tin một cách nhanh chóng, chính xác và an toàn, từ đó hỗ trợ công tác điều trị, nghiên cứu y khoa và quản lý bệnh viện.

Nghiên cứu và phát triển hệ thống CSDL này sẽ tập trung vào các chức năng chính bao gồm quản lý hồ sơ bệnh án, thông tin bệnh nhân, quản lý lịch sử khám chữa bệnh, quản lý nhân sự y tế và các tài nguyên của bệnh viện. Hệ thống sẽ được thiết kế để áp dụng cho các bệnh viện và cơ sở y tế đa khoa. Phương pháp nghiên cứu bao gồm khảo sát các nhu cầu và quy trình hiện tại tại các bệnh viện, phân tích yêu cầu hệ thống, thiết kế mô hình dữ liệu, và triển khai Cơ sở dữ liệu. Ngoài ra, nghiên cứu sẽ áp dụng các biện pháp bảo mật dữ liệu để đảm bảo an toàn thông tin bệnh nhân.

Chương 1 của tài liệu sẽ giới thiệu các khái niệm và cơ sở lý thuyết về cơ sở dữ liệu và quản lý thông tin y tế. Chương 2 sẽ phân tích yêu cầu hệ thống, mục đích, tình huống sử dụng. Chương 3 là phần thiết kế mô hình dữ liệu chi tiết. Chương 4 trình bày quá trình triển khai hệ thống. Chương 5 sẽ thử nghiệm và đánh giá hệ thống, tăng hiệu suất. Phần cuối cùng sẽ đưa ra các kết luận và đề xuất hướng phát triển trong tương lai.

CHƯƠNG 2. Các khái niệm, cơ sở lý thuyết về cơ sở dữ liệu và quản lý thông tin y tế

2.1 Khái niệm về cơ sở dữ liệu (CSDL)

CSDL (Database) là một tập hợp có cấu trúc của dữ liệu được tổ chức và lưu trữ để có thể dễ dàng truy cập, quản lý và cập nhật. CSDL được sử dụng trong nhiều lĩnh vực khác nhau để lưu trữ thông tin và hỗ trợ các hoạt động quản lý, phân tích dữ liệu. Có nhiều loại CSDL khác nhau, bao gồm:

- **CSDL quan hệ (Relational Database)**: Dữ liệu được lưu trữ trong các bảng (tables) có mối quan hệ với nhau. Hệ quản trị CSDL quan hệ (RDBMS) như MySQL, PostgreSQL và Oracle là các hệ thống phổ biến sử dụng loại CSDL này.
- **CSDL phi quan hệ (NoSQL Database)**: Được thiết kế để lưu trữ dữ liệu không có cấu trúc hoặc bán cấu trúc, chẳng hạn như MongoDB, Cassandra và Redis.

2.2 Các thành phần cơ bản của một hệ quản trị cơ sở dữ liệu (DBMS)

Hệ quản trị CSDL (Database Management System - DBMS) là phần mềm cho phép người dùng định nghĩa, tạo lập, bảo trì và điều khiển truy cập đến CSDL. Các thành phần chính của DBMS bao gồm:

- **Dữ liệu (Data)**: Thông tin cần quản lý, được tổ chức thành các bảng trong trường hợp CSDL quan hệ.
- **Người dùng (Users)**: Bao gồm người quản trị CSDL (DBA), lập trình viên và người dùng cuối.
- **Phần mềm (Software)**: Bao gồm hệ quản trị, công cụ phát triển và các ứng dụng hỗ trợ khác.
- **Phần cứng (Hardware)**: Máy chủ và thiết bị lưu trữ dữ liệu.

2.3 Quản lý thông tin y tế

Quản lý thông tin y tế (Healthcare Information Management) là việc thu thập, lưu trữ, phân tích và bảo mật thông tin liên quan đến sức khỏe của bệnh nhân và các hoạt động của cơ sở y tế. Các khía cạnh chính bao gồm:

- **Hồ sơ y tế điện tử (Electronic Health Records - EHR)**: Hệ thống kỹ thuật số lưu trữ hồ sơ y tế của bệnh nhân, bao gồm lịch sử bệnh, chẩn đoán, kết quả xét nghiệm và kế hoạch điều trị.

- **Hệ thống thông tin bệnh viện (Hospital Information System - HIS)**: Hệ thống quản lý toàn bộ hoạt động của bệnh viện, từ quản lý bệnh nhân, nhân sự, tài chính đến lịch trình làm việc và kho dược.
- **Bảo mật thông tin y tế**: Đảm bảo an toàn và bảo mật cho dữ liệu y tế của bệnh nhân, bao gồm việc tuân thủ các quy định pháp luật như HIPAA (Health Insurance Portability and Accountability Act).

2.4 Lợi ích của việc sử dụng CSDL trong quản lý thông tin y tế.

- ***Tăng cường hiệu quả quản lý***: Giúp tự động hóa các quy trình quản lý, giảm thiểu sai sót và tăng tốc độ truy xuất thông tin.
- ***Cải thiện chất lượng chăm sóc***: Cung cấp thông tin chính xác và kịp thời cho các chuyên gia y tế, hỗ trợ việc ra quyết định điều trị.
- ***Bảo mật và tuân thủ pháp luật***: Đảm bảo bảo mật thông tin bệnh nhân và tuân thủ các quy định về bảo vệ dữ liệu.
- ***Hỗ trợ nghiên cứu và phân tích***: Cung cấp dữ liệu cho các nghiên cứu y khoa và phân tích xu hướng sức khỏe cộng đồng.

2.5 Các thách thức trong việc quản lý thông tin y tế

- ***Bảo mật và quyền riêng tư***: Đảm bảo an toàn cho thông tin y tế nhạy cảm của bệnh nhân.
- ***Tích hợp dữ liệu***: Đảm bảo sự nhất quán và tích hợp giữa các hệ thống thông tin khác nhau trong bệnh viện.
- ***Chi phí triển khai***: Đầu tư vào hạ tầng công nghệ và đào tạo nhân viên sử dụng hệ thống.

2.6 Các công nghệ và phương pháp tiếp cận.

- ***Hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS)***: Sử dụng các hệ thống như MySQL, PostgreSQL để quản lý dữ liệu cấu trúc.
- ***Hệ thống quản lý cơ sở dữ liệu phi quan hệ (NoSQL)***: Sử dụng các hệ thống như MongoDB, CouchDB để quản lý dữ liệu không cấu trúc.
- ***Blockchain***: Áp dụng công nghệ blockchain để tăng cường bảo mật và minh bạch trong quản lý thông tin y tế.
- ***AI và Machine Learning***: Sử dụng trí tuệ nhân tạo để phân tích dữ liệu và hỗ trợ ra quyết định trong chăm sóc sức khỏe.

Kết luận

Việc thiết kế và xây dựng một hệ thống cơ sở dữ liệu hiện đại và hiệu quả để quản

lý bệnh án, thông tin bệnh nhân và hoạt động bệnh viện là một yếu tố quan trọng giúp nâng cao chất lượng dịch vụ y tế. Sự kết hợp giữa công nghệ tiên tiến và quản lý thông tin y tế chuyên nghiệp sẽ giúp các cơ sở y tế cải thiện hiệu quả hoạt động, đảm bảo an toàn thông tin và hỗ trợ các hoạt động nghiên cứu y khoa.

CHƯƠNG 3. Phân tích yêu cầu hệ thống

3.1 Phân tích yêu cầu hệ thống

3.1.1 Yêu cầu về chức năng(Functional Requirements)

1. Quản lý thông tin bệnh nhân:
    - Lưu trữ và cập nhật thông tin cá nhân của bệnh nhân bao gồm họ tên, ngày sinh, giới tính, địa chỉ, và số điện thoại.
    - Theo dõi lịch sử khám bệnh, chẩn đoán, và quá trình điều trị của bệnh nhân.
    - Quản lý các kết quả xét nghiệm, hình ảnh y khoa và các tài liệu liên quan khác.
2. Quản lý hồ sơ bệnh án:
    - Lưu trữ và truy xuất hồ sơ bệnh án điện tử (EHR) của bệnh nhân.
    - Quản lý các đơn thuốc, phác đồ điều trị.
    - Ghi nhận và theo dõi tình trạng sức khỏe, diễn biến bệnh lý của bệnh nhân.
3. Quản lý nhân sự y tế:
    - Lưu trữ thông tin về nhân viên y tế bao gồm bác sĩ, y tá, và nhân viên hành chính.
    - Theo dõi lịch làm việc, chuyên môn của các nhân viên y tế.
    - Quản lý phân công công việc và lịch trực của nhân viên y tế.
4. Quản lý tài nguyên bệnh viện:
    - Quản lý phòng khám, giường bệnh, thiết bị y tế, và các tài nguyên khác.
    - Theo dõi tình trạng sử dụng và bảo trì của các thiết bị y tế.
    - Quản lý kho dược và vật tư y tế, bao gồm nhập, xuất và tồn kho.
5. Bảo mật và quyền truy cập:
    - Phân quyền truy cập cho các nhóm người dùng khác nhau (nhân viên y tế, bệnh nhân, người kiểm duyệt).
    - Đảm bảo an toàn và bảo mật thông tin bệnh nhân, tuân thủ các quy định về bảo vệ dữ liệu.

3.1.2 Yêu cầu phi chức năng (Non-functional Requirements)

1. Hiệu suất: Hệ thống phải có khả năng xử lý và truy xuất dữ liệu nhanh chóng, đảm bảo hiệu quả trong quá trình hoạt động.
2. Tính sẵn sàng: Hệ thống phải hoạt động ổn định và có khả năng phục hồi sau sự cố, đảm bảo tính liên tục trong quá trình sử dụng.
3. Khả năng mở rộng: Hệ thống phải dễ dàng mở rộng để đáp ứng nhu cầu gia tăng về dữ liệu và người dùng trong tương lai.
4. Bảo mật: Đảm bảo an toàn cho dữ liệu y tế nhạy cảm, bao gồm mã hóa dữ liệu và quản lý quyền truy cập.

3.2 Mục đích

1. Cải thiện hiệu quả quản lý bệnh viện.
2. Nâng cao chất lượng chăm sóc sức khỏe.
3. Tăng cường bảo mật và tuân thủ quy định pháp luật.
4. Hỗ trợ ra quyết định và nghiên cứu y khoa.
5. Thống kê và nghiên cứu y tế.

3.3 Tình huống sử dụng

1. Tiếp nhận bệnh nhân mới
2. Sửa đổi, bổ sung thông tin
3. Lưu trữ thông tin và kết quả xét nghiệm, phẫu thuật, lịch sử sử dụng thuốc
4. Chẩn đoán và điều trị
5. Theo dõi và điều chỉnh
6. Chia sẻ thông tin với bộ phận chăm sóc sau ra viện.
7. Xóa hồ sơ tồn tại quá thời gian cho phép
8. Kiểm soát số lượng, ca làm việc của các bộ phận nhân viên y tế.
9. Kiểm soát, quản lí kho dược của bệnh viện.

CHƯƠNG 4. Xây dựng cơ sở dữ liệu chi tiết

4.1 Thực thể và các thuộc tính

1. personal_info: identifier, name, dob, gender, phone, address
2. medical_record: medical_record_id, identifier, doctor_id, admission, discharge, diagnosis, results
3. doctor: dr_id, dr_name, dr_position, department_id, dob, gender, phone
4. department: department_id, department_name
5. surgery: surgery_id, surgery_name, department_id, price
6. test: test_id, test_name, department_id, price
7. medicine: medicine_id, medicine_name, dosage_form, price
8. prescription: prescription_id, medicine_id, date, medical_record_id, dr_id, dosage, doses
9. surgery_line: surgery_line_id, surgery_id, medical_record_id, referring, attending, date, results
10. test_line: test_line_id, test_id, medical_record_id, referring, attending, date, results
11. bill_prescription: bill_id, prescription_id, total
12. bill_test: bill_id, test_line_id, total
13. bill_surgery: bill_id, surgery_id, total

4.2 Mô tả chi tiết các bảng và mối quan hệ

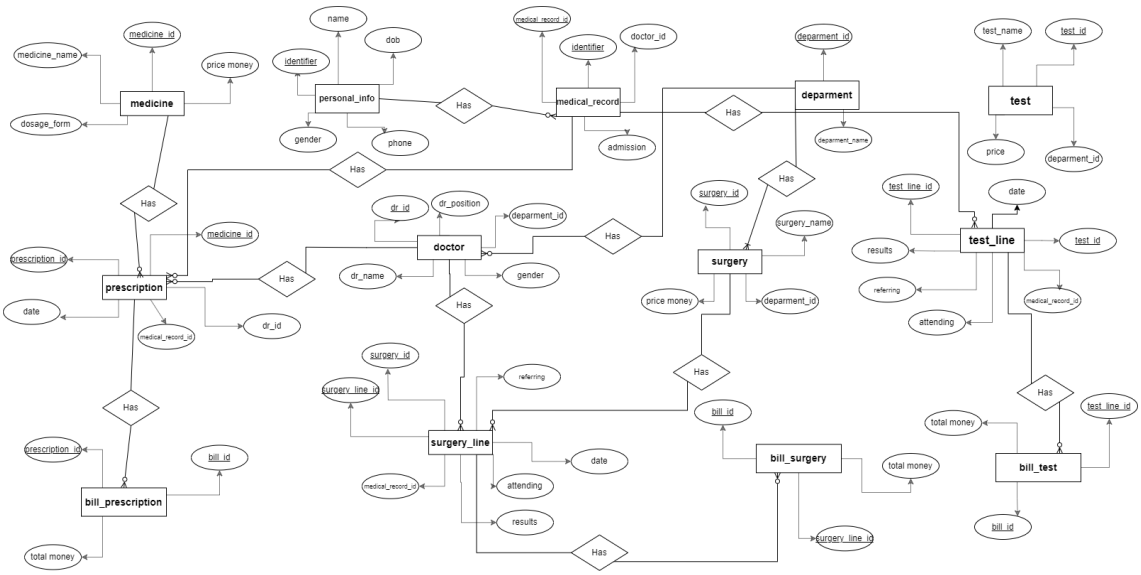
- Personal_infor: Chứa thông tin cá nhân của bệnh nhân. Mỗi bệnh nhân có một mã định danh duy nhất (identifier).
- Department: Chứa thông tin của các khoa khám bệnh trong bệnh viện bao gồm tên và một mã định danh duy nhất.
- Doctor: Chứa thông tin cá nhân của các bác sĩ, cũng có một mã định danh duy nhất trong bệnh viện. Liên kết với một khoa thông qua khóa ngoại (department_id)
- Medical_record: Lưu trữ các hồ sơ bệnh án của bệnh nhân. Mỗi hồ sơ bệnh án liên kết với một bệnh nhân qua khóa ngoại (identifier). Liên kết với 1 bác sĩ thông qua khóa ngoại (dr_id)

- Test, Surgery, Medicine: Lần lượt chứa thông tin các loại xét nghiệm, phẫu thuật, các loại thuốc mà bệnh viện đang có. Bảng Test, Surgery sẽ liên kết với 1 bảng Department thông qua khóa ngoại (department_id)
- Test_line, Surgery_line, Prescription: Lần lượt chứa thông tin chi tiết của các xét nghiệm mà từng bệnh nhân đã làm. Liên kết với bảng bác sĩ qua 2 khóa ngoại (dr_id) gồm bác sĩ chỉ định và bác sĩ thực hiện. Liên kết với 1 bệnh án thông quá khóa ngoại (medical_record_id)
- bill_test, bill_surgery, bill_prescription: Chứa thông tin hóa đơn của xét nghiệm, phẫu thuật, đơn thuốc. Liên kết thông qua các khóa ngoại với các bảng Test_line, Surgery_line, Prescription

4.3 Các liên kết

- medicine(n) – prescription (n)
- prescription (1) – bill_prescription (1)
- doctor (1) – prescription (n)
- doctor (n) – surgery_line (n)
- doctor (n) – deparment (1)
- surgery (n) – surgery_line (n)
- deparment (1) – surgery (n)
- surgery_line (1) – bill_surgery (1)
- medical_record (1) – prescription (1)
- test (1) – bill_test (1)
- test(n) – test_line(n)
- test_line (1) – bill_test (1)

4.4 Lược đồ quan hệ (ERD - Entity-Relationship Diagram)



Hình 4.1: Lược đồ quan hệ ERD

CHƯƠNG 5. Trình bày quá trình triển khai hệ thống

5.1 Triển khai hệ thống

5.1.1 Tạo các bảng

```sql
CREATE TABLE personal_infor (
    identifier VARCHAR(25) NOT NULL,
    name VARCHAR(100) NOT NULL,
    dob date NOT NULL,
    gender VARCHAR(50) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    address VARCHAR(255) NOT NULL,

    PRIMARY KEY(identifier)
);
```

Hình 5.1: Tạo bảng Personal_infor

```sql
CREATE TABLE department (
    department_id VARCHAR(25) NOT NULL,
    department_name VARCHAR(100) NOT NULL,

    PRIMARY KEY(department_id)
);
```

Hình 5.2: Tạo bảng Department

```
CREATE TABLE doctor (
    dr_id VARCHAR(25) NOT NULL,
    dr_name VARCHAR(100) NOT NULL,
    dr_position VARCHAR(100) NOT NULL,
    department_id VARCHAR(25) NOT NULL,
    dob date NOT NULL,
    gender VARCHAR(50) NOT NULL,
    phone VARCHAR(15) NOT NULL,

    PRIMARY KEY(dr_id),
    FOREIGN KEY(department_id) ON department(department_id)
);
```

Hình 5.3: Tạo bảng Doctor

```
CREATE TABLE medical_record (
    medical_record_id VARCHAR(25) UNIQUE NOT NULL,
    identifier VARCHAR(25) NOT NULL,
    dr_id VARCHAR(25) NOT NULL,
    admission date NOT NULL,
    discharge date NOT NULL,
    diagnosis VARCHAR(500) NOT NULL,
    results VARCHAR(255) NOT NULL,

    PRIMARY KEY(medical_record_id, identifier),
    FOREIGN KEY(identifier) ON personal_infor(identifier),
    FOREIGN KEY(dr_id) ON doctor(dr_id)
);
```

Hình 5.4: Tạo bảng Medical_record

```
CREATE TABLE test (
    test_id VARCHAR(25) NOT NULL,
    test_name VARCHAR(100) NOT NULL,
    department_id VARCHAR(25) NOT NULL,
    price MONEY NOT NULL

    PRIMARY KEY(test_id),
    FOREIGN KEY(department_id) ON department(department_id)
);
```

Hình 5.5: Tạo bảng Test

```
CREATE TABLE test_line (
    test_line_id VARCHAR(25) NOT NULL,
    test_id VARCHAR(25) NOT NULL,
    medical_record_id VARCHAR(25) NOT NULL,
    referring VARCHAR(25) NOT NULL,
    attending VARCHAR(25) NOT NULL,
    date DATE NOT NULL,
    results VARCHAR(500) NOT NULL

    PRIMARY KEY(test_line_id,test_id),
    FOREIGN KEY(test_id) ON test(test_id),
    FOREIGN KEY(medical_record_id) ON medical_record(medical_record_id),
    FOREIGN KEY(referring) ON doctor(dr_id),
    FOREIGN KEY(attending) ON doctor(dr_id)
);
```

Hình 5.6: Tạo bảng Test_line

```
CREATE TABLE bill_test (
    bill_id VARCHAR(25) NOT NULL,
    test_line_id VARCHAR(25) NOT NULL,
    total MONEY

    PRIMARY KEY(bill_id),
    FOREIGN KEY(test_line_id) ON test_line(test_line_id)
);
```

Hình 5.7: Tạo bảng Bill_test

```
CREATE TABLE surgery (
    surgery_id VARCHAR(25) NOT NULL,
    surgery_name VARCHAR(100) NOT NULL,
    department_id VARCHAR(25) NOT NULL,
    price MONEY NOT NULL

    PRIMARY KEY(surgery_id),
    FOREIGN KEY(department_id) ON department(department_id)
);
```

Hình 5.8: Tạo bảng Surgery

```
CREATE TABLE surgery_line (
    surgery_line_id VARCHAR(25) NOT NULL,
    surgery_id VARCHAR(25) NOT NULL,
    medical_record_id VARCHAR(25) NOT NULL,
    referring VARCHAR(25) NOT NULL,
    attending VARCHAR(25) NOT NULL,
    date date NOT NULL,
    results VARCHAR(500) NOT NULL

    PRIMARY KEY(surgery_line_id,surgery_id),
    FOREIGN KEY(surgery_id) ON surgery(surgery_id),
    FOREIGN KEY(medical_record_id) ON medical_record(medical_record_id),
    FOREIGN KEY(referring) ON doctor(dr_id),
    FOREIGN KEY(attending) ON doctor(dr_id)
);
```

Hình 5.9: Tạo bảng Surgery_line

```
CREATE TABLE bill_surgery (
    bill_id VARCHAR(25) NOT NULL,
    surgery_line_id VARCHAR(25) NOT NULL,
    total MONEY

    PRIMARY KEY(bill_id),
    FOREIGN KEY(surgery_line_id) ON surgery_line(surgery_line_id)
);
```

Hình 5.10: Tạo bảng Bill_surgery

```
CREATE TABLE medicine (
    medicine_id VARCHAR(25) NOT NULL,
    medicine_name VARCHAR(500) NOT NULL,
    dosage_form VARCHAR(25) NOT NULL,
    price MONEY NOT NULL,

    PRIMARY KEY(medicine_id)
);
```

Hình 5.11: Tạo bảng Medicine

```
CREATE TABLE prescription (
    prescription_id VARCHAR(25) NOT NULL,
    medicine_id VARCHAR(100) NOT NULL,
    date DATE NOT NULL,
    medical_record_id VARCHAR(25) NOT NULL,
    dr_id VARCHAR(25) NOT NULL,
    dosage VARCHAR(100) NOT NULL,
    doses NUMERIC NOT NULL,

    PRIMARY KEY(presciption_id, medicine_id),
    FOREIGN KEY(medicine_id) ON medicine(medicine_id),
    FOREIGN KEY(medical_record_id) ON medical_record(medical_record_id),
    FOREIGN KEY(dr_id) ON doctor(dr_id)
);
```

Hình 5.12: Tạo bảng Presciption

```
CREATE TABLE bill_prescription (
    bill_id VARCHAR(25) NOT NULL,
    prescription_id VARCHAR(25) NOT NULL,
    total MONEY

    PRIMARY KEY(bill_id),
    FOREIGN KEY(prescription_id) ON prescription(prescription_id)
);
```

Hình 5.13: Tạo bảng Bill_presciption

5.1.2 Các câu truy vấn trong hệ thống

a, Query

1. Phân cấp người dùng thành các nhóm: Bệnh nhân, nhân viên y tế, quản trị viên -> Đảm bảo tính bảo mật, phân quyền cho từng nhóm người dùng.

```sql
-- Tạo các vai trò
CREATE ROLE patient_role;
CREATE ROLE doctor_role;
CREATE ROLE admin_role;

-- Tạo các người dùng và gán vai trò
CREATE ROLE patient_1 LOGIN PASSWORD '00000000';
GRANT patient_role TO patient_1;

CREATE ROLE doctor_1 LOGIN PASSWORD '00007737';
GRANT doctor_role TO doctor_1;

CREATE ROLE admin_1 LOGIN PASSWORD '11110000';
GRANT admin_role TO admin_1;

-- Gán quyền cho vai trò
-- Bệnh nhân chỉ có quyền SELECT
GRANT SELECT ON Medical_Record TO patient_role;

-- Bác sĩ có quyền SELECT, INSERT
GRANT SELECT, INSERT ON Medical_Record TO doctor_role;

-- Quản trị hệ thống có quyền SELECT, INSERT, UPDATE, DELETE
GRANT ALL ON Medical_Record TO admin_role;

```

Hình 5.14: Phân cấp người dùng

2. Lấy danh sách id của các bác sĩ có số bệnh nhân họ đã khám từ 7 bệnh nhân trở lên -> Tính hiệu suất làm việc -> Đề xuất khen thưởng, thăng chức.

```sql
SELECT
    med.dr_id,
    dr.dr_name,
    COUNT(med.*) AS Num_of_patients
FROM
    medical_record med
NATURAL JOIN
    doctor dr
GROUP BY
    med.dr_id,
    dr.dr_name
HAVING
    COUNT(med.*) >= 7
ORDER BY
    Num_of_patients DESC;
```

Hình 5.15: Lấy id các bác sĩ > 7 bệnh nhân

3. Biết rằng độ tuổi nghỉ hưu của các bác sĩ được quy định như sau: 61 tuổi đối với nam, 56 tuổi đối với nữ. Hãy in ra thông tin các bác sĩ đã đến tuổi nghỉ hưu. -> Kiểm soát số lượng nhân sự của bệnh viện.

```
SELECT
    *
FROM
    doctor dr
WHERE
    (gender = 'Male' AND DATE_PART('year', AGE(current_date, dob)) >= 61)
    OR
    (gender = 'Female'  AND DATE_PART('year', AGE(current_date, dob)) >= 56)
```

Hình 5.16: Hiển thị thông tin các bác sĩ nghỉ hưu

4. Truy vấn để lấy danh sách bác sĩ và số bệnh nhân mà họ đã điều trị.

```
SELECT
    d.dr_id,
    d.dr_name,
    COUNT(med.medical_record_id) As PatientCount
FROM
    doctor d
JOIN
    medical_record med ON d.dr_id = med.dr_id
GROUP BY
    d.dr_id,
    d.dr_name
ORDER BY
    PatientCount DESC;
```

Hình 5.17: Lấy danh sách bác sĩ và số bệnh nhân họ đã điều trị

5. Truy vấn để lấy danh sách thuốc và số lượng thuốc đã được kê đơn cho mỗi bệnh nhân.

```sql
SELECT
    pre.medical_record_id,
    p.identifier,
    p.name,
    m.medicine_name,
    SUM(pre.doses) AS TotalDoses
FROM
    medicine m
JOIN
    prescription pre ON m.medicine_id = pre.medicine_id
JOIN
    medical_record med ON med.medical_record_id = pre.medical_record_id
JOIN
    personal_infor p On p.identifier = med.identifier
GROUP BY
    m.medicine_name,
    p.identifier,
    pre.medical_record_id
ORDER BY
    medical_record_id DESC;
```

Hình 5.18: Danh sách thuốc và số lượng thuốc cho mỗi bệnh nhân

6. Truy vấn ra thông tin bệnh nhân và số lần nhập viện.

```sql
SELECT
    p.*,
    COUNT(med.identifier) AS AdmissionNums
FROM
    personal_infor p
JOIN
    medical_record med ON p.identifier = med.identifier
GROUP BY
    med.identifier,
    p.identifier
ORDER BY
    AdmissionNums DESC;
```

Hình 5.19: Thông tin và số lần nhập viện của mỗi bệnh nhân

7. Đối với các ca phẫu thuật có tình trạng nguy kịch và cần phải phẫu thuật gấp, viết truy vấn để đếm số lượng các ca phẫu thuật đó và đã có kết quả tốt vào cuối tuần -> Tăng lương cho bác sĩ chịu trách nhiệm ca phẫu thuật đó.

```
SELECT
    attending,
    dr_name,
    COUNT(surgery_line_id) AS Surgery_Weekend_Success
FROM
    surgery_line
NATURAL JOIN
    doctor
WHERE
    results = 'Complete Success' AND EXTRACT(DOW FROM DATE) IN (6,0)
GROUP BY
    attending, dr_name
ORDER BY
    COUNT(surgery_line_id) DESC
-- 6:Sat / 0:Sun
```

Hình 5.20: Số ca phẫu thuật mà mỗi bác sĩ đã thực hiện thành công vào cuối tuần

8. Truy vấn để hiển thị thông tin các bác sĩ (còn đang công tác) ở các khoa.

```
SELECT
    dr.dr_id,
    dr.dr_name,
    dr.dr_position,
    d.department_name,
    dr.gender,
    DATE_PART('year', AGE(current_date, dr.dob)) AS Age
FROM
    doctor dr
JOIN
    department d ON dr.department_id = d.department_id
WHERE
    (dr.gender = 'Male' AND DATE_PART('year', AGE(current_date, dr.dob)) <= 61)
    OR
    (dr.gender = 'Female' AND DATE_PART('year', AGE(current_date, dr.dob)) <= 54)
ORDER BY
    department_name,
    dr_id
```

Hình 5.21: Thông tin các bác sĩ ở các khoa

9. Đối với các ca phẫu thuật cho trẻ sơ sinh (dưới 28 ngày tuổi) điều quan trọng nhất là cần phải có một số loại xét nghiệm nhất định để kiểm tra tình trạng của trẻ điển hình là xét nghiệm máu. Viết truy vấn đưa ra thông tin của trẻ sơ sinh đã được xét nghiệm máu (Blood tests).

```sql
SELECT p.*
FROM personal_infor p
JOIN medical_record med ON p.identifier = med.identifier
JOIN test_line tl ON med.medical_record_id = tl.medical_record_id
JOIN test t ON tl.test_id = t.test_id
WHERE t.test_name = 'Blood tests'
AND AGE(tl.date,p.dob) <= INTERVAL '28 days'
GROUP BY p.identifier
```

Hình 5.22: Thông tin trẻ sơ sinh được xét nghiệm máu

10. Hiển thị phần trăm của kết quả khám chữa bệnh ở bệnh viện -> Đánh giá tổng quan chất lượng khám, chữa bệnh ở bệnh viện -> Nâng cao chất lượng.

```sql
SELECT
        results,
        COUNT(results),
        TO_CHAR(COUNT(results) * 100.0 / (SELECT COUNT(*) FROM medical_record), 'FM999990.00') || '%' AS percentage
FROM
        medical_record
GROUP BY
        results;
```

Hình 5.23: Phần trăm kết quả khám, chữa bệnh

b, Function

1. Tìm các bệnh án mắc cùng loại bệnh => Tra cứu lịch sử, phác đồ điều trị của một số căn bệnh => Phục vụ học tập, nghiên cứu.

```
CREATE FUNCTION public.find_by_disease(value character varying)
RETURNS TABLE(
    mid character varying,
    patient_name character varying,
    admission date,
    discharge date,
    diagnosis character varying,
    results character varying
) AS $$
BEGIN
    RETURN QUERY
        SELECT
            med.medical_record_id,
            p.name,
            med.admission,
            med.discharge,
            med.diagnosis,
            med.results
        FROM
            personal_infor p
        NATURAL JOIN
            medical_record med
        WHERE
                med.diagnosis = value;
END;
$$ LANGUAGE plpgsql;
```

Hình 5.24: Tìm các bệnh án có cùng loại bệnh

2. Tra cứu bệnh án dựa vào mã định danh của bệnh nhân => Tra cứu lịch sử thăm khám.

```
CREATE FUNCTION find_by_id(value character varying)
RETURNS TABLE(
    mid character varying,
    patient_name character varying,
    admission date,
    discharge date,
    diagnosis character varying,
    results character varying
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        med.medical_record_id,
        p.name,
        med.admission,
        med.discharge,
        med.diagnosis,
        med.results
    FROM
        personal_infor p
    JOIN
        medical_record med ON p.identifier = med.identifier
    WHERE
        med.diagnosis = value;
END;
$$ LANGUAGE plpgsql
```

Hình 5.25: Tra cứu bệnh án dựa vào mã định danh

3. Hiện nay số lượng các ca phẫu thuật do tai nạn giao thông đã tăng lên, hãy viết hàm để khi bệnh nhân làm phẫu thuật với mức giá loại phẫu thuật trên 35$ thì được giảm 15% loại phẫu thuật đó rồi tính tổng chi phí các loại phẫu thuật của bệnh nhân.

```
CREATE OR REPLACE FUNCTION discounted_surgery_cost(value VARCHAR(25))
RETURNS NUMERIC AS $$
DECLARE
    total_cost NUMERIC := 0;
    surgery_cost NUMERIC;
    discounted_cost NUMERIC;
BEGIN
    FOR surgery_cost IN
        SELECT
            s.price
        FROM
            surgery s
        JOIN
            surgery_line sl ON s.surgery_id = sl.surgery_id
        JOIN
            medical_record med ON sl.medical_record_id = med.medical_record_id
        JOIN
            personal_infor p ON med.identifier = p.identifier
        WHERE
            p.identifier = value
        LOOP
            IF surgery_cost > 35
                THEN discounted_cost := surgery_cost * 0.85;
            ELSE
                discounted_cost := surgery_cost;
            END IF;
        total_cost := total_cost + discounted_cost;
        END LOOP;
    RETURN total_cost;
END;
$$ LANGUAGE plpgsql;
SELECT discounted_surgery_cost('851-17-6955');
```

Hình 5.26: Giảm giá phẫu thuật

4. Viết hàm nhận đầu vào là identifier của bệnh nhân và trả về các loại xét nghiệm đã thực hiện.

```
CREATE OR REPLACE FUNCTION Patients_Tested(value varchar(255))
RETURNS TABLE(
    test_line_id VARCHAR,
    test_id VARCHAR,
    test_name VARCHAR,
    date DATE,
    results VARCHAR
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        tl.test_line_id,
        tl.test_id,
        t.test_name,
        tl.date,
        tl.results
    FROM
        test_line tl
    NATURAL JOIN
        test t
    JOIN
        medical_record med ON tl.medical_record_id = med.medical_record_id
    WHERE
        med.identifier = value;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM Patients_Tested('785-64-6151')
(Có bệnh nhân xét nghiệm và cả bệnh nhân không xét nghiệm gì)
```

Hình 5.27: Trả lại các xét nghiệm bệnh nhân đã thực hiện

5. Viết hàm nhận đầu vào là identifier của bệnh nhân và trả về các loại phẫu thuật đã thực hiện.

```
CREATE OR REPLACE FUNCTION patients_surgeries(value VARCHAR(255))
RETURNS TABLE(
    surgery_line_id VARCHAR,
    surgery_id VARCHAR,
    surgery_name VARCHAR,
    date DATE,
    results VARCHAR
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        sl.surgery_line_id,
        sl.surgery_id,
        s.surgery_name,
        sl.date, sl.results
    FROM
        surgery_line sl
    NATURAL JOIN
        surgery s
    JOIN
        medical_record med ON sl.medical_record_id = med.medical_record_id
    WHERE
        med.identifier = value;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM patients_surgeries('785-64-6151');
```

Hình 5.28: Trả lại các phẫu thuật bệnh nhân đã thực hiện

6. Hiển thị các loại thuốc mà bệnh nhân này đã sử dụng trong quá trình điều trị ở bệnh viện -> Hỗ trợ điều trị, tùy thuộc vào các loại thuốc đã từng sử dụng để xây dựng phác đồ điều trị hợp lí, tránh xung đột.

```
CREATE OR REPLACE FUNCTION medicine_usage_history(id VARCHAR(20))
RETURNS TABLE(
    Medicine_name VARCHAR(500)
)
AS $$
BEGIN
    RETURN QUERY
    SELECT
        m.medicine_name
    FROM
        medicine m
    JOIN
        prescription pre ON m.medicine_id = pre.medicine_id
    JOIN
        medical_record med ON med.medical_record_id = pre.medical_record_id
    WHERE
        med.identifier = id;
 END;
 $$ LANGUAGE plpgsql;
SELECT * FROM medicine_usage_history('420-42-9114')
```

Hình 5.29: Trả lại lịch sử sử dụng thuốc

7. Tính giá trị hóa đơn xét nghiệm.

```
CREATE OR REPLACE FUNCTION test_bill(id VARCHAR(20))
RETURNS TABLE(
    Name VARCHAR(50),
    Medical_record_id VARCHAR(20),
    Total MONEY
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        p.name,
        med.medical_record_id,
        SUM(bt.total) AS total
    FROM
        bill_test bt
    JOIN
        test_line tl ON tl.test_line_id = bt.test_line_id
    JOIN
        medical_record med ON med.medical_record_id = tl.medical_record_id
    JOIN
        personal_infor p ON p.identifier = med.identifier
    WHERE
        med.identifier = id
    GROUP BY
        p.name,
        med.medical_record_id,
        med.identifier
    ORDER BY
        med.medical_record_id;
END;
$$ LANGUAGE plpgsql;
```

Hình 5.30: Tính hóa đơn xét nghiệm

8. Tính giá trị hóa đơn phẫu thuật.

```
CREATE OR REPLACE FUNCTION surgery_bill(id VARCHAR(20))
RETURNS TABLE(
    Name VARCHAR(50),
    Medical_record_id VARCHAR(20),
    Total MONEY
 ) AS $$
BEGIN
    RETURN QUERY
    SELECT
        p.name,
        med.medical_record_id,
        SUM(bs.total) AS total
    FROM
        bill_surgery bs
    JOIN
        surgery_line sl ON sl.surgery_line_id = bs.surgery_line_id
    JOIN
        medical_record med ON med.medical_record_id = sl.medical_record_id
    JOIN
        personal_infor p ON p.identifier = med.identifier
    WHERE
        med.identifier = id
    GROUP BY
        p.name,
        med.medical_record_id,
        med.identifier
    ORDER BY
        med.medical_record_id;
END;
$$ LANGUAGE plpgsql;
```

Hình 5.31: Tính hóa đơn phẫu thuật

9. Tính giá trị hóa đơn đơn thuốc.

```
CREATE OR REPLACE FUNCTION prescription_bill(id VARCHAR(20))
RETURNS TABLE(
    Name VARCHAR(50),
    Medical_record_id VARCHAR(20),
    Total MONEY
 ) AS $$
BEGIN
    RETURN QUERY
    SELECT
        p.name,
        med.medical_record_id,
        SUM(bp.total) AS total
    FROM
        bill_prescription bp
    JOIN
        prescription pre ON pre.prescription_id = bp.prescription_id
    JOIN
        medical_record med ON med.medical_record_id = pre.medical_record_id
    JOIN
        personal_infor p ON p.identifier = med.identifier
    WHERE
        med.identifier = id
    GROUP BY
        p.name,
        med.medical_record_id,
        med.identifier
    ORDER BY
        med.medical_record_id;
END;
$$ LANGUAGE plpgsql;
```

Hình 5.32: Tính hóa đơn đơn thuốc

10. Tính tổng chi phí dịch vụ của mỗi bệnh nhân khi khám bệnh (bao gồm đơn thuốc, xét nghiệm và phẫu thuật nếu có)

```
CREATE OR REPLACE FUNCTION bill(id VARCHAR(20), type_of_bill TEXT)
RETURNS TABLE(
    Name VARCHAR(50),
    Medical_record_id VARCHAR(20),
    Total MONEY
 ) AS $$
 BEGIN
    IF type_of_bill = 'test'
        THEN RETURN QUERY
            SELECT
                tb.name,
                tb.medical_record_id,
                tb.total
            FROM
                test_bill(id) AS tb;

    ELSIF type_of_bill = 'surgery'
        THEN RETURN QUERY
            SELECT
                sb.name,
                sb.medical_record_id,
                sb.total
            FROM
                surgery_bill(id) AS sb;

    ELSIF type_of_bill = 'prescription'
        THEN RETURN QUERY
            SELECT
                pb.name,
                pb.medical_record_id,
                pb.total
            FROM
                prescription_bill(id) AS pb;
```

```
    ELSIF type_of_bill = 'all'
        THEN RETURN QUERY
            SELECT
                combined.name,
                combined.medical_record_id,
                SUM(combined.total) AS total
            FROM (
                SELECT
                    tb.name,
                    tb.medical_record_id,
                    tb.total
                FROM
                    test_bill(id) AS tb

                UNION ALL

                SELECT
                    sb.name,
                    sb.medical_record_id,
                    sb.total
                FROM
                    surgery_bill(id) AS sb

                UNION ALL

                SELECT
                    pb.name,
                    pb.medical_record_id,
                    pb.total
                FROM
                    prescription_bill(id) AS pb) AS combined
                GROUP BY
                    combined.name,
                    combined.medical_record_id
                ORDER BY
                    combined.medical_record_id;
```

```
            SELECT
                tb.name,
                tb.medical_record_id,
                tb.total
            FROM
                test_bill(id) AS tb

            UNION ALL

            SELECT
                sb.name,
                sb.medical_record_id,
                sb.total
            FROM
                surgery_bill(id) AS sb

            UNION ALL

            SELECT
                pb.name,
                pb.medical_record_id,
                pb.total
            FROM
                prescription_bill(id) AS pb) AS combined
            GROUP BY
                combined.name,
                combined.medical_record_id
            ORDER BY
                combined.medical_record_id;

    ELSE
        RAISE EXCEPTION 'Invalid type_of_bill value. Allowed values are: test, surgery, prescription, all';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Hình 5.33: Tính toán các chi phí

11. Hiển thị bệnh án gần đây nhất của bệnh nhân.

```
CREATE OR REPLACE FUNCTION find_recently_med(value VARCHAR(15))
RETURNS TABLE(
    Patient_name VARCHAR(35),
    Medical_record_id VARCHAR(20),
    Doctor_name VARCHAR(20),
    Admission DATE,
    Discharge DATE,
    Diagnosis VARCHAR(500),
    Result VARCHAR(255)
 ) AS $$
BEGIN
    RETURN QUERY
    SELECT
        p.name,
        med.medical_record_id,
        dr.dr_name,
        med.admission,
        med.discharge,
        med.diagnosis,
        med.results
    FROM
        medical_record med
NATURAL JOIN
        personal_infor p
JOIN
        doctor dr ON dr.dr_id = med.dr_id
WHERE
        med.identifier = value
ORDER BY
        med.admission DESC
LIMIT 1;
END;
$$ LANGUAGE plpgsql;
```

Hình 5.34: Hiển thị bệnh án gần nhất

12. Hiển thị các bệnh nhân được điều trị bởi 1 bác sĩ cụ thể.

```
CREATE OR REPLACE FUNCTION get_patients_by_doctor(value VARCHAR(25))
RETURNS TABLE(
    identifier VARCHAR(25),
    name VARCHAR(100),
    dob DATE,
    gender VARCHAR(50),
    phone VARCHAR(15),
    address VARCHAR(255)
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        p.identifier,
        p.name,
        p.dob,
        p.gender,
        p.phone,
        p.address
    FROM
        personal_infor p
    JOIN
        medical_record med ON p.identifier = med.identifier
    WHERE
        med.doctor_id = value;
END;
$$ LANGUAGE plpgsql;
```

Hình 5.35: Hiển thị các bệnh nhân được điều trị bởi 1 bác sĩ cụ thể

13. Biến đầu vào là 1 khoảng thời gian -> Đưa ra tổng số xét nghiệm, phẫu thuật của từng khoa đã thực hiện -> Xem xét nâng cấp các khoa đó (csvc, tuyển thêm bác sĩ).

```
CREATE OR REPLACE FUNCTION upgrade_department(from_year INTEGER, to_year INTEGER)
RETURNS TABLE(
    department_name VARCHAR(255),
    quantity INTEGER
) AS $$
DECLARE
    total INTEGER;
    cnt_test INTEGER := 0;
    cnt_surgery INTEGER := 0;
    k RECORD;
    fromdate DATE;
    todate DATE;
BEGIN
    fromdate := to_date(from_year::text, 'YYYY');
    todate := to_date(to_year::text, 'YYYY');

    FOR k IN
        SELECT
            d.department_id,
            d.department_name
        FROM
            department d
        LOOP
            SELECT
                COUNT(*) INTO cnt_test
            FROM
                test t
            JOIN
                test_line tl ON t.test_id = tl.test_id
            WHERE
                t.department_id = k.department_id AND tl.date BETWEEN fromdate AND todate;
```

```
            SELECT
                COUNT(*) INTO cnt_surgery
            FROM
                surgery s
            JOIN
                surgery_line sl ON s.surgery_id = sl.surgery_id
            WHERE
                s.department_id = k.department_id AND sl.date BETWEEN fromdate AND todate;
            total := cnt_test + cnt_surgery;
                IF total > 0 THEN
                    department_name := k.department_name;
                    quantity := total;
                    RETURN NEXT;
                END IF;
        END LOOP;
END;
$$ LANGUAGE plpgsql;
```

Hình 5.36: Đưa ra tổng số xét nghiệm, phẫu thuật của từng khoa đã thực hiện

14. Hiển thị ra các dạng thuốc của một loại thuốc -> tùy thuộc vào thể trạng của bệnh nhân để sử dụng các dạng thuốc khác nhau.

```
CREATE OR REPLACE FUNCTION form(value VARCHAR(500))
RETURNS TABLE(
    dosage_form VARCHAR(35)
)
AS $$
BEGIN
    RETURN QUERY
    SELECT
        m.dosage_form
    FROM
        medicine m
    WHERE
        m.medicine_name = value;
END;
$$ LANGUAGE plpgsql;
```

Hình 5.37: Hiển thị ra các dạng thuốc của một loại thuốc

c, Trigger và Trigger Function

1. Viết trigger cho việc khi cập nhật số lượng bác sĩ thì thông báo ra hay không lượng bác sĩ cần phải tuyển thêm khi số lượng bác sĩ nghỉ hưu trên 200 người.

```
CREATE OR REPLACE FUNCTION check_retired_doctors()
RETURNS TRIGGER AS $$
DECLARE
    retired_doc_count int;
BEGIN
    SELECT
        COUNT(*) INTO retired_doc_count
    FROM
        doctor
    WHERE
        (gender = 'Male' AND DATE_PART('year', AGE(current_date, dob)) >= 61)
        OR
        (gender = 'Female' AND DATE_PART('year', AGE(current_date, dob)) >= 56);
    IF retired_doc_count > 200 THEN
        RAISE NOTICE 'Number of retired doctors is %: Need to hire more doctors!',retired_doc_count;
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER Notice_doc
AFTER INSERT OR DELETE OR UPDATE ON doctor
FOR EACH ROW
EXECUTE FUNCTION check_retired_doctors()
```

Hình 5.38: Trigger cập nhật số lượng bác sĩ và hiển thị thông báo

2. Trigger để kiểm tra khi số lượng bệnh án đang trong quá trình hoặc liên quan đến vấn đề về việc điều trị quá 1000 thì dừng việc tiếp nhận thêm bệnh án.

```
CREATE OR REPLACE FUNCTION check_medical_record_count ()
RETURNS TRIGGER AS $$
BEGIN
    IF(Select Count(*) From medical_record
       WHERE medical_record.results In('Under treatment')) >= 1000
    Then
    RAISE EXCEPTION 'The number of patients is overloaded, unable to accept more patients!';
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER Trigger_medical_record_count
BEFORE INSERT ON medical_record
FOR EACH ROW
EXECUTE FUNCTION check_medical_record_count();
```

Hình 5.39: Trigger thông báo quá tải

3. Trigger để mỗi khi thêm một loại phẫu thuật mới liên quan về tim vào thì chi phí không được dưới 40 USD do những ca phẫu thuật tim đòi hỏi sự phức tạp.

```
CREATE OR REPLACE FUNCTION check_Heart_money()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.surgery_name ILike '%heart%' Then
        IF NEW.price ::Numeric < 40 Then
            RAISE EXCEPTION 'The cost for heart-related surgery should not be less than 40 USD';
        END IF;
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER Heart_money
AFTER INSERT OR UPDATE ON surgery
FOR EACH ROW
EXECUTE FUNCTION check_Heart_money();
```

Hình 5.40: Trigger thêm giá phẫu thuật tim

4. Viết trigger để Update phần Total của Bill_Surgery (nếu thực hiện vào thứ 7, chủ nhật thì total = total * 1.5)

```
-- Tạo hàm để tính toán total
CREATE OR REPLACE FUNCTION calculate_total(p_surgery_id VARCHAR, p_surgery_date DATE)
RETURNS MONEY AS $$
DECLARE
    surgery_price MONEY;
    calculated_total MONEY;
BEGIN
    -- Lấy giá trị Price từ bảng surgery
    SELECT Price INTO surgery_price
    FROM surgery
    WHERE surgery.surgery_id = p_surgery_id;
    -- Tính toán total
    IF EXTRACT(DOW FROM p_surgery_date) IN (0, 6) THEN
        -- Nếu là thứ Bảy (6) hoặc Chủ Nhật (0), nhân đôi giá trị
        calculated_total := surgery_price * 1.5;
    ELSE
        calculated_total := surgery_price;
    END IF;
    RETURN calculated_total;
END;
$$ LANGUAGE plpgsql;
-- Tạo trigger để cập nhật total trong bảng bill_surgery
CREATE OR REPLACE FUNCTION UPDATE_bill_total()
RETURNS TRIGGER AS $$
BEGIN
    -- Chèn giá trị total mới vào bảng bill_surgery
    INSERT INTO bill_surgery (bill_id, surgery_line_id, total)
    VALUES (NEW.surgery_line_id, NEW.surgery_line_id,
            calculate_total(NEW.surgery_id, NEW.date));
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
-- Tạo trigger để gọi hàm UPDATE_bill_total sau khi thêm bản ghi vào bảng surgery_line
CREATE TRIGGER trg_UPDATE_bill_total
AFTER INSERT ON surgery_line
FOR EACH ROW
EXECUTE FUNCTION UPDATE_bill_total();
```

Hình 5.41: Trigger update total bill surgery

5. Viết trigger để Update phần Total của Bill_Test (nếu thực hiện vào thứ 7, chủ nhật thì total = total * 1.5)

```
CREATE OR REPLACE FUNCTION calculate_prescription_total(p_prescription_id VARCHAR, p_prescription_date DATE)
RETURNS MONEY AS $$
DECLARE
    medicine_price MONEY;
    calculated_total MONEY;
BEGIN
    SELECT price INTO medicine_price
    FROM medicine
    WHERE medicine.medicine_id = p_prescription_id;
    IF EXTRACT(DOW FROM p_prescription_date) IN (0, 6) THEN
        -- Nếu là thứ Bảy (6) hoặc Chủ Nhật (0), nhân đôi giá trị
        calculated_total := medicine_price * 1.5;
    ELSE
        calculated_total := medicine_price;
    END IF;
    RETURN calculated_total;
END;
$$ LANGUAGE plpgsql;
CREATE OR REPLACE FUNCTION UPDATE_bill_prescription_total()
RETURNS TRIGGER AS $$
BEGIN
    -- Chèn giá trị total mới vào bảng bill_prescription
    INSERT INTO bill_prescription (bill_id, prescription_id, total)
    VALUES (NEW.prescription_id, NEW.prescription_id,
            calculate_prescription_total(NEW.medicine_id, NEW.date));

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_UPDATE_bill_prescription_total
AFTER INSERT ON prescription
FOR EACH ROW
EXECUTE FUNCTION UPDATE_bill_prescription_total();
```

Hình 5.42: Trigger update total bill test

6. Viết trigger để Update phần Total của Bill_Prescription.

```
CREATE OR REPLACE FUNCTION calculate_prescription_total(p_prescription_id VARCHAR, p_prescription_date DATE)
RETURNS MONEY AS $$
DECLARE
    medicine_price MONEY;
    calculated_total MONEY;
BEGIN
    SELECT price INTO medicine_price
    FROM medicine
    WHERE medicine.medicine_id = p_prescription_id;
    IF EXTRACT(DOW FROM p_prescription_date) IN (0, 6) THEN
        -- Nếu là thứ Bảy (6) hoặc Chủ Nhật (0), nhân đôi giá trị
        calculated_total := medicine_price * 1.5;
    ELSE
        calculated_total := medicine_price;
    END IF;
    RETURN calculated_total;
END;
$$ LANGUAGE plpgsql;
CREATE OR REPLACE FUNCTION UPDATE_bill_prescription_total()
RETURNS TRIGGER AS $$
BEGIN
    -- Chèn giá trị total mới vào bảng bill_prescription
    INSERT INTO bill_prescription (bill_id, prescription_id, total)
    VALUES (NEW.prescription_id, NEW.prescription_id,
            calculate_prescription_total(NEW.medicine_id, NEW.date));

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_UPDATE_bill_prescription_total
AFTER INSERT ON prescription
FOR EACH ROW
EXECUTE FUNCTION UPDATE_bill_prescription_total();
```

Hình 5.43: Trigger update total bill presciption

7. Trigger để kiểm tra id của 1 loại thuốc thì có tồn tại ở trong đơn thuốc hay

không?

```
CREATE OR REPLACE FUNCTION check_med_exists()
RETURNS TRIGGER AS $$
Declare
    med_exists Boolean;
BEGIN
    --check IF medicine_id exists in medicine
    Select Exists(
        Select 1 From medicine
        WHERE medicine_id = NEW.medicine_id) Into med_exists;
    -- RAISE EXCEPTION IF not exists
    IF Not med_exists Then
        RAISE EXCEPTION 'medicine_id Number % does not exists here', NEW.medicine_id;
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
CREATE TRIGGER check_med_before_insert
BEFORE INSERT OR UPDATE ON medicine
FOR EACH ROW
EXECUTE FUNCTION check_med_exists();
```

Hình 5.44: Kiểm tra đơn thuốc

CHƯƠNG 6. Thử nghiệm và đánh giá hệ thống

6.1 Tối ưu hóa hệ thống

- Với câu truy vấn số 2

  Tạo index: CREATE INDEX index1 ON medical_record(dr_id);

	dr_id character varying (25)	dr_name character varying (100)	num_of_patients bigint
1	Dr-206-834-622	Royall Nevin	25
2	Dr-206-245-678	Brigg Bebbington	23
3	Dr-205-791-428	Merrielle Penney	23
4	Dr-209-486-490	Zorana Jewers	22
5	Dr-208-435-284	Corinne Filintsev	21
6	Dr-203-512-184	Romonda Keating	21
7	Dr-202-135-299	Bobette Gostling	20

Total rows: 296 of 296 Query complete 00:00:00.071

(a) Không dùng index

	dr_id character varying (25)	dr_name character varying (100)	num_of_patients bigint
1	Dr-206-834-622	Royall Nevin	25
2	Dr-206-245-678	Brigg Bebbington	23
3	Dr-205-791-428	Merrielle Penney	23
4	Dr-209-486-490	Zorana Jewers	22
5	Dr-208-435-284	Corinne Filintsev	21
6	Dr-203-512-184	Romonda Keating	21
7	Dr-202-135-299	Bobette Gostling	20

Total rows: 296 of 296 Query complete 00:00:00.048

(b) Có dùng index

Hình 6.1: So sánh hiệu quả khi dùng index

Hiệu suất tăng 32,39%

- Với câu truy vấn số 3

Data Output Messages Notifications

	dr_id [PK] character varying (25)	dr_name character varying (100)	dr_position character varying (100)	department_id character varying (25)	dob date	gender character varying (50)	phone character varying (15)
1	Dr-194-179-479	Eal Petracci	Attending Physician	K738	1954-07-28	Male	547-309-1904
2	Dr-191-223-640	Gaelan Harron	Attending Physician	K156	1924-07-30	Male	992-461-6717
3	Dr-198-892-828	Janie Upsale	Attending Physician	K894	1936-03-06	Female	542-153-2113
4	Dr-191-171-750	Giacinta Alker	Attending Physician	K448	1935-05-02	Female	197-129-4068
5	Dr-193-441-923	Isabelle Molnar	Attending Physician	K780	1950-08-30	Female	177-136-3597
6	Dr-191-471-488	Babette Narrie	Attending Physician	K011	1959-01-14	Female	654-397-4803
7	Dr-195-071-586	Sindee Yuryaev	Attending Physician	K791	1941-03-18	Female	795-911-2482

Total rows: 534 of 534 Query complete 00:00:00.383

Hình 6.2: Chưa tối ưu hóa

```
SELECT
    *
FROM
    doctor dr
WHERE
    gender = 'Male' AND EXTRACT(year FROM AGE(current_date, dob)) >= 61

UNION

SELECT
    *
FROM
    doctor dr
WHERE
    gender = 'Female' AND EXTRACT(year FROM AGE(current_date, dob)) >= 56;
CREATE INDEX index2 ON doctor(gender,dob)
```

Hình 6.3: Tối ưu hóa

	dr_id character varying (25)	dr_name character varying (100)	dr_position character varying (100)	department_id character varying (25)	dob date	gender character varying (50)	phone character varying (15)
1	Dr-196-653-756	Jared Stit	Specialist	K051	1922-04-22	Male	597-853-1821
2	Dr-191-099-109	Paten Spriddle	Attending Physician	K051	1921-04-08	Male	428-562-7702
3	Dr-191-707-988	Fabe McLanaghan	Resident Physician	K717	1956-10-20	Male	963-995-1515
4	Dr-198-025-970	Timoteo Bedrosian	Consultant	K967	1929-08-21	Male	921-247-8411
5	Dr-195-533-693	Daisie McTague	Resident Physician	K977	1920-09-24	Female	719-626-8215
6	Dr-195-364-002	Marven Dyshart	Resident Physician	K907	1927-04-17	Male	206-933-6872
7	Dr-190-977-104	Eugine Trimming	Attending Physician	K099	1920-08-19	Female	834-700-2753

Total rows: 534 of 534 Query complete 00:00:00.087

Hình 6.4: Sử dụng index

Hiệu suất tăng 77.28%

- Với câu truy vấn số 4

  Tạo index: CREATE INDEX index1 ON medical_record(dr_id)

	dr_id character varying (25)	dr_name character varying (100)	num_of_patients bigint
1	Dr-206-834-622	Royall Nevin	25
2	Dr-206-245-678	Brigg Bebbington	23
3	Dr-205-791-428	Merrielle Penney	23
4	Dr-209-486-490	Zorana Jewers	22
5	Dr-208-435-284	Corinne Filintsev	21
6	Dr-203-512-184	Romonda Keating	21
7	Dr-202-135-299	Bobette Gostling	20

Total rows: 296 of 296 Query complete 00:00:00.071

(a) Không dùng index

	dr_id character varying (25)	dr_name character varying (100)	num_of_patients bigint
1	Dr-206-834-622	Royall Nevin	25
2	Dr-206-245-678	Brigg Bebbington	23
3	Dr-205-791-428	Merrielle Penney	23
4	Dr-209-486-490	Zorana Jewers	22
5	Dr-208-435-284	Corinne Filintsev	21
6	Dr-203-512-184	Romonda Keating	21
7	Dr-202-135-299	Bobette Gostling	20

Total rows: 296 of 296 Query complete 00:00:00.048

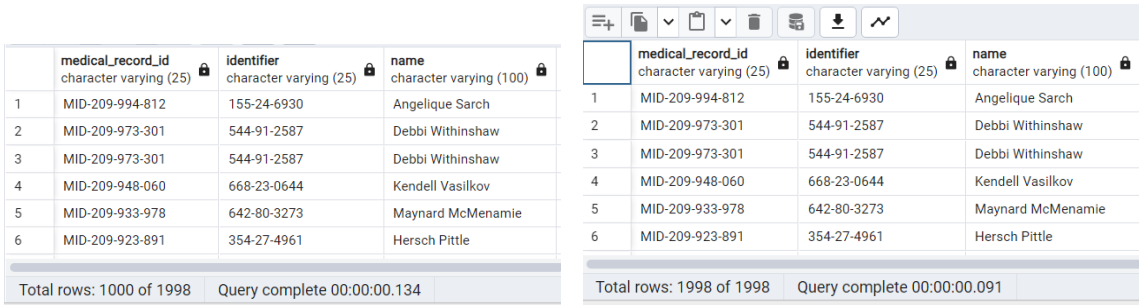
(b) Có dùng index

Hình 6.5: So sánh hiệu quả khi dùng index

Hiệu suất tăng 54.94%

- Với câu truy vấn số 5

  Tạo index: CREATE INDEX index3 ON prescription(medical_record_id)



	medical_record_id character varying (25)	identifier character varying (25)	name character varying (100)
1	MID-209-994-812	155-24-6930	Angelique Sarch
2	MID-209-973-301	544-91-2587	Debbi Withinshaw
3	MID-209-973-301	544-91-2587	Debbi Withinshaw
4	MID-209-948-060	668-23-0644	Kendell Vasilkov
5	MID-209-933-978	642-80-3273	Maynard McMenamie
6	MID-209-923-891	354-27-4961	Hersch Pittle

Total rows: 1000 of 1998 Query complete 00:00:00.134

(a) Không dùng index

	medical_record_id character varying (25)	identifier character varying (25)	name character varying (100)
1	MID-209-994-812	155-24-6930	Angelique Sarch
2	MID-209-973-301	544-91-2587	Debbi Withinshaw
3	MID-209-973-301	544-91-2587	Debbi Withinshaw
4	MID-209-948-060	668-23-0644	Kendell Vasilkov
5	MID-209-933-978	642-80-3273	Maynard McMenamie
6	MID-209-923-891	354-27-4961	Hersch Pittle

Total rows: 1998 of 1998 Query complete 00:00:00.091

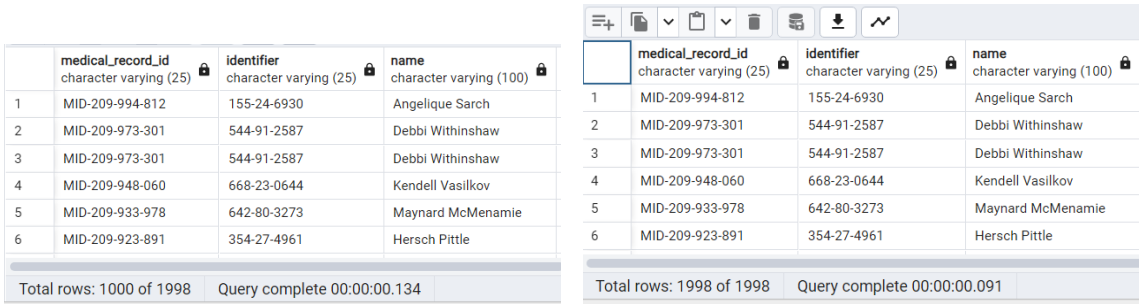
(b) Có dùng index

Hình 6.6: So sánh hiệu quả khi dùng index

Hiệu suất tăng 32,08%

- Với câu truy vấn 6

  Tạo index: CREATE INDEX index4 ON medical_record(identifier)



	medical_record_id character varying (25)	identifier character varying (25)	name character varying (100)
1	MID-209-994-812	155-24-6930	Angelique Sarch
2	MID-209-973-301	544-91-2587	Debbi Withinshaw
3	MID-209-973-301	544-91-2587	Debbi Withinshaw
4	MID-209-948-060	668-23-0644	Kendell Vasilkov
5	MID-209-933-978	642-80-3273	Maynard McMenamie
6	MID-209-923-891	354-27-4961	Hersch Pittle

Total rows: 1000 of 1998 Query complete 00:00:00.134

(a) Không dùng index

	medical_record_id character varying (25)	identifier character varying (25)	name character varying (100)
1	MID-209-994-812	155-24-6930	Angelique Sarch
2	MID-209-973-301	544-91-2587	Debbi Withinshaw
3	MID-209-973-301	544-91-2587	Debbi Withinshaw
4	MID-209-948-060	668-23-0644	Kendell Vasilkov
5	MID-209-933-978	642-80-3273	Maynard McMenamie
6	MID-209-923-891	354-27-4961	Hersch Pittle

Total rows: 1998 of 1998 Query complete 00:00:00.091

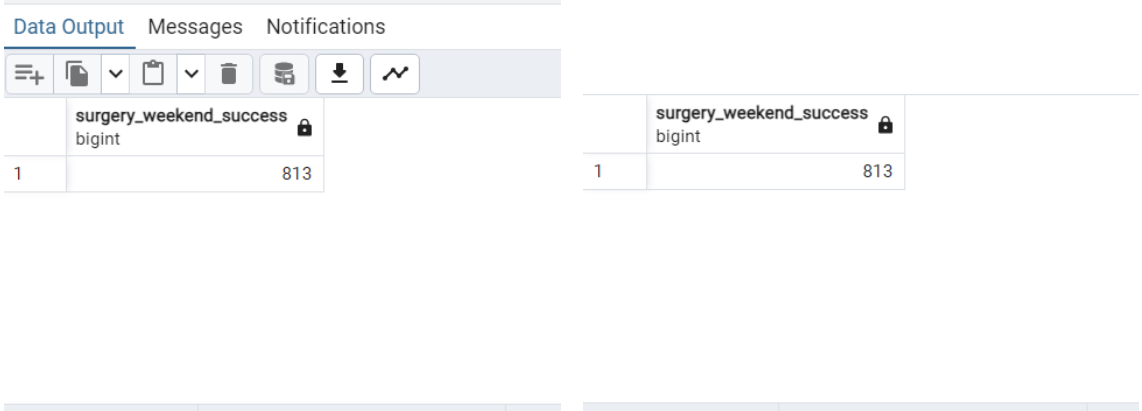
(b) Có dùng index

Hình 6.7: So sánh hiệu quả khi dùng index

Hiệu suất tăng 22,49%

- Với câu truy vấn 7

  Tạo index: CREATE INDEX index5 ON Surgery_line(Results)



Data Output Messages Notifications

	surgery_weekend_success bigint
1	813

Total rows: 1 of 1 Query complete 00:00:00.069

(a) Không dùng index

	surgery_weekend_success bigint
1	813

Total rows: 1 of 1 Query complete 00:00:00.058

(b) Có dùng index

Hình 6.8: So sánh hiệu quả khi dùng index

Hiệu suất tăng 15,94%

- Với câu truy vấn 8

	dr_id character varying (25)	dr_name character varying (100)	dr_position character varying (100)
1	Dr-203-343-257	Kaye Lowdyane	Attending Physician
2	Dr-203-394-479	Joann Liddington	Resident Physician
3	Dr-208-012-310	Vikky Drohan	Head of Department
4	Dr-209-577-864	Eleanora Martinot	Chief Medical Officer
5	Dr-206-363-230	Karna Burniston	Specialist
6	Dr-208-248-916	Zia Dahl	Head of Department

Total rows: 213 of 213 Query complete 00:00:00.103

Hình 6.9: Không sử dụng index

```
(SELECT
    dr.dr_id,
    dr.dr_name,
    dr.dr_position,
    d.department_name,
    dr.gender,
    DATE_PART('year', AGE(current_date, dr.dob)) AS Age
FROM
    doctor dr
JOIN
    department d ON dr.department_id = d.department_id
WHERE
    dr.gender LIKE 'M%' AND DATE_PART('year', AGE(current_date, dr.dob)) <= 61
)
UNION
(SELECT
    dr.dr_id,
    dr.dr_name,
    dr.dr_position,
    d.department_name,
    dr.gender,
    DATE_PART('year', AGE(current_date, dr.dob)) AS Age
FROM
    doctor dr
JOIN
    department d ON dr.department_id = d.department_id
WHERE
    dr.gender LIKE 'F%' AND DATE_PART('year', AGE(current_date, dr.dob)) <= 54
)
ORDER BY department_name, dr_id;
CREATE INDEX index6 ON doctor(gender,dob,department_id)
```

Total rows: 213 of 213 Query complete 00:00:00.046

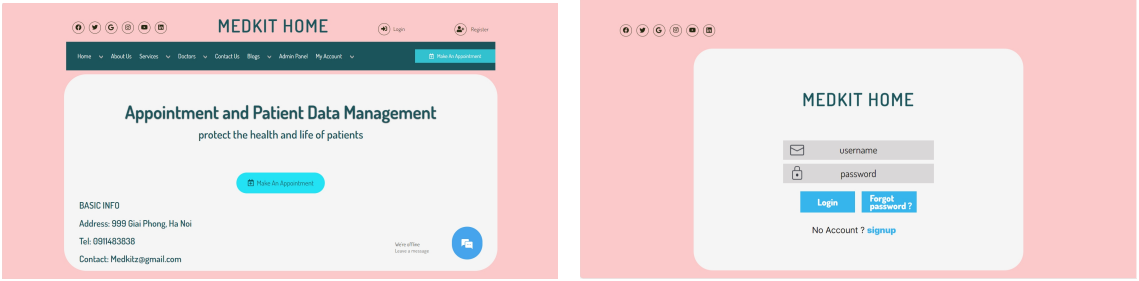
Hình 6.10: Tối ưu hóa câu truy vấn và sử dụng index

- Tương tự với các câu truy vấn khác:

  - Câu 9
    CREATE INDEX index7 ON personal_infor(dob)
    CREATE INDEX index8 ON test(test_name)
  - Câu 10
    CREATE INDEX percentage ON medical_record(results)
  - Câu 1
    CREATE INDEX index9 ON medical_record(diagnosis)
  - Câu 2
    CREATE INDEX index4 ON medical_record(identifier)
  - Câu 3
    CREATE INDEX index10 ON surgery(price)
    CREATE INDEX index4 ON medical_record(identifier)
  - Câu 4, 5, 6, 7, 8, 9
    CREATE INDEX index4 ON medical_record(identifier)
  - Câu 10
    CREATE INDEX index4 ON medical_record(identifier)
    CREATE INDEX index11 ON surgery_line(medical_record_id) CREATE INDEX index12 ON test_line(medical_record_id) CREATE INDEX index13 ON prescription_line(medical_record_id)
  - Câu 11
    CREATE INDEX index14 ON medical_record(identifier,admission)
  - Câu 12
    CREATE INDEX index1 ON medical_record(dr_id)
  - Câu 13
    CREATE INDEX index15 ON test(department_id) CREATE INDEX index16 ON surgery(department_id)
  - Câu 14

CREATE INDEX index15 ON medicine(medicine_name)

6.2 Thiết kế giao diện



(a) Trang chủ của trang web **(b)** Trang đăng nhập của trang web

Hình 6.11: Giao diện web hồ sơ bệnh án online

CHƯƠNG 7. Kết luận và đề xuất cải tiến

7.1 Kết luận và đề xuất hướng phát triển trong tương lai

7.1.1 Kết luận

Đề tài này đã tập trung nghiên cứu và phát triển một hệ thống cơ sở dữ liệu (CSDL) hiện đại và toàn diện phục vụ cho việc quản lý bệnh án, thông tin bệnh nhân và các hoạt động của bệnh viện. Qua quá trình nghiên cứu, khảo sát nhu cầu thực tế và phân tích các yêu cầu hệ thống, chúng tôi đã thiết kế và triển khai một mô hình dữ liệu hợp lý, đáp ứng được các tiêu chí về tốc độ, độ chính xác và tính an toàn trong việc quản lý thông tin y tế.

Hệ thống CSDL được xây dựng đã chứng minh hiệu quả trong việc lưu trữ, quản lý và truy xuất thông tin nhanh chóng, chính xác và an toàn, hỗ trợ đắc lực cho công tác điều trị, nghiên cứu y khoa và quản lý bệnh viện. Việc ứng dụng hệ thống này giúp giảm thiểu các sai sót do phương pháp quản lý truyền thống, tiết kiệm nguồn lực và nâng cao chất lượng dịch vụ y tế.

7.1.2 Đề xuất hướng phát triển trong tương lai

- **Nâng cao tính linh hoạt và mở rộng hệ thống:** Nghiên cứu phát triển các tính năng linh hoạt hơn để hệ thống có thể dễ dàng tích hợp với các phần mềm và hệ thống khác trong bệnh viện. Điều này giúp đồng bộ hóa dữ liệu và cải thiện hiệu quả quản lý tổng thể.
- **Ứng dụng trí tuệ nhân tạo (AI) và học máy (ML):** Áp dụng các kỹ thuật AI và ML để phân tích dữ liệu bệnh nhân, dự đoán xu hướng bệnh tật và hỗ trợ quyết định lâm sàng. Điều này có thể cải thiện đáng kể chất lượng chẩn đoán và điều trị.
- **Phát triển giao diện người dùng (UI/UX) thân thiện hơn:** Nghiên cứu và cải tiến giao diện người dùng để đảm bảo tính thân thiện, dễ sử dụng và tối ưu hóa trải nghiệm người dùng, đặc biệt là cho các nhân viên y tế.
- **Cải tiến bảo mật và quyền riêng tư:** Nghiên cứu các phương pháp bảo mật tiên tiến để bảo vệ dữ liệu bệnh nhân khỏi các mối đe dọa an ninh mạng. Điều này bao gồm việc triển khai các biện pháp mã hóa mạnh mẽ và quản lý quyền truy cập hiệu quả.
- **Đánh giá và cải tiến hiệu suất hệ thống:** Liên tục theo dõi, đánh giá hiệu suất của hệ thống và thực hiện các biện pháp tối ưu hóa để đảm bảo hệ thống hoạt động ổn định, nhanh chóng và hiệu quả.

- **Hỗ trợ đào tạo và nâng cao kỹ năng cho nhân viên y tế:** Cung cấp các chương trình đào tạo và tài liệu hướng dẫn để giúp nhân viên y tế sử dụng hệ thống một cách hiệu quả, từ đó nâng cao chất lượng quản lý và chăm sóc bệnh nhân.
- **Nghiên cứu khả năng mở rộng quốc tế:** Xem xét việc triển khai hệ thống ở quy mô quốc tế, nghiên cứu các yêu cầu và quy định khác nhau ở từng quốc gia để đảm bảo tính tương thích và hiệu quả khi áp dụng ở các môi trường y tế khác nhau.

Với những định hướng phát triển này, hệ thống cơ sở dữ liệu quản lý thông tin y tế sẽ không chỉ đáp ứng tốt các yêu cầu hiện tại mà còn có khả năng thích ứng và phát triển bền vững trong tương lai, đóng góp tích cực vào sự nghiệp chăm sóc sức khỏe cộng đồng.

SOURCE CODE

1. Cơ sở dữ liệu: Click here
2. Các câu truy vấn, hàm, trigger: Click here
3. Giao diện: Click here