

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM**



BÀI TẬP LỚN

TÊN HỌC PHẦN: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

ĐỀ TÀI: QUẢN LÝ BỆNH VIỆN

Giảng viên: ThS. Nguyễn Ngọc An

Sinh viên thực hiện:

STT	Mã sinh viên	Họ và tên	Ngày sinh	Lớp
1	1871070010	Tạ Ánh My	01/02/2006	HTTT 18 -01
2	1871070013	Nguyễn Anh Thư	11/03/2006	HTTT 18-01

Hà Nội, năm 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM



BÀI TẬP LỚN

TÊN HỌC PHẦN: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

ĐỀ TÀI: QUẢN LÝ BỆNH VIỆN

STT	Mã sinh viên	Họ và tên	Ngày sinh	Điểm	
				Bằng số	Bằng chữ
1	1871070010	Tạ Ánh My	01/02/2006		
2	1871070013	Nguyễn Anh Thư	11/03/2006		

CÁN BỘ CHẤM THI 1

CÁN BỘ CHẤM THI 2

Hà Nội, năm 2025

MỤC LỤC

MỤC LỤC	3
LỜI MỞ ĐẦU	6
BẢNG CÁC TỪ VIẾT TẮT	8
CHƯƠNG 1: GIỚI THIỆU	10
1.1. Tóm tắt về cơ sở dữ liệu:	10
1.1.1. Khái niệm Cơ sở dữ liệu (Database):	10
1.1.2. Khái niệm Hệ quản trị Cơ sở dữ liệu (DBMS):	10
1.1.3. Vai trò của CSDL và HQTCSDL trong đề tài "Quản lý bệnh viện":	11
1.2. Công cụ và công nghệ sử dụng:.....	12
1.3. Bảng phân công:	12
CHƯƠNG 2: PHÂN TÍCH YÊU CẦU HỆ THỐNG	13
2.1. Phân tích và yêu cầu cơ sở dữ liệu:	13
2.1.1. Mục tiêu của hệ thống:	13
2.1.2. Phân tích yêu cầu dữ liệu (nghiệp vụ hệ thống):	13
2.2. Các chức năng chính của hệ thống:	14
2.2.1. Chức năng quản lý người dùng hệ thống:	14
2.2.2. Chức năng quản lý bệnh nhân:	14
2.2.3. Chức năng quản lý bác sĩ:	15
2.2.4. Chức năng quản lý khoa:	15
2.2.5. Chức năng quản lý lịch khám:.....	15
2.2.6. Chức năng quản lý hóa đơn:.....	15
2.2.7. Chức năng báo cáo – thống kê:	15
2.3. Mô tả mối quan hệ giữa các thực thể.....	16

CHƯƠNG 3: THIẾT KẾ CƠ SỞ DỮ LIỆU.....	17
3.1. Thiết kế mô hình thực thể quan hệ (ER)	17
3.2. Chuyển đổi mô hình ER thành lược đồ quan hệ:	18
3.3 Cấu trúc các bảng dữ liệu:	18
CHƯƠNG 4: CÀI ĐẶT CƠ SỞ DỮ LIỆU	21
4.1. Tạo bảng:	21
4.2. Sơ đồ cơ sở dữ liệu:	23
CHƯƠNG 5: CÁC ĐÓI TƯỢNG CƠ SỞ DỮ LIỆU NÂNG CAO	24
5.1. Cài đặt các chỉ mục (Indexes)	24
5.2. Views	24
5.3. Thủ tục	26
5.4. Hàm người dùng định nghĩa	30
5.5. Trigger	33
CHƯƠNG 6: BẢO MẬT VÀ QUẢN TRỊ CƠ SỞ DỮ LIỆU.....	41
6.1. Quản lý người dùng và quyền truy cập	41
6.2. Bảo mật cơ sở dữ liệu	42
6.2.1. Mã hóa dữ liệu (Data Encryption)	42
6.2.2. Bảo mật kết nối (Connection Security)	43
6.2.3. Giới hạn truy cập dữ liệu bằng View	44
6.2.4. Phân quyền truy cập và tổ chức dữ liệu theo Schema	44
6.2.5. Kết hợp nhiều lớp bảo mật (Defense in Depth).....	46
6.3. Quản lý sao lưu và phục hồi	47
6.4. Quản lý hiệu suất cơ sở dữ liệu	49
CHƯƠNG 7: KẾT LUẬN VÀ KHUYẾN NGHỊ.....	51

7.1. Tóm tắt liên quan	51
7.2. Khuyên nghị	51
KẾT LUẬN	53
DANH MỤC TÀI LIỆU THAM KHẢO	54

LỜI MỞ ĐẦU

Trong bối cảnh xã hội hiện đại, y tế là một trong những ngành dịch vụ trọng yếu, có ảnh hưởng trực tiếp đến sức khỏe và chất lượng cuộc sống của con người. Cùng với sự phát triển của công nghệ thông tin, việc ứng dụng tin học hóa vào công tác quản lý tại các cơ sở y tế, đặc biệt là bệnh viện, đã trở thành một xu thế tất yếu và cấp thiết.

Hiện nay, các bệnh viện phải đổi mới với việc quản lý một khối lượng thông tin khổng lồ và vô cùng phức tạp. Dữ liệu này không chỉ bao gồm thông tin hành chính của hàng ngàn Bệnh nhân, hồ sơ lý lịch của đội ngũ Bác sĩ, mà còn là toàn bộ lịch sử khám chữa bệnh, các chỉ định y khoa, lịch hẹn, thông tin về các Khoa phòng, và các giao dịch tài chính liên quan đến Hóa đơn, viện phí.

Việc quản lý các nghiệp vụ này bằng phương pháp thủ công (sổ sách, hồ sơ giấy) hoặc các công cụ văn phòng cơ bản (như Excel) bộc lộ rất nhiều bất cập nghiêm trọng:

1. Tính trùng lặp và thiếu nhất quán: Thông tin của một bệnh nhân có thể phải nhập lại nhiều lần ở các khoa phòng khác nhau, dẫn đến sai lệch (ví dụ: sai địa chỉ, số điện thoại) và lãng phí tài nguyên lưu trữ.
2. Khó khăn trong tra cứu và tổng hợp: Việc tìm kiếm lịch sử khám bệnh của một bệnh nhân, thống kê số ca khám trong ngày của một bác sĩ, hay lập báo cáo doanh thu theo khoa phòng tốn rất nhiều thời gian và công sức, làm giảm hiệu suất làm việc.
3. Rủi ro về an ninh và bảo mật: Hồ sơ bệnh án giấy rất dễ bị thất lạc, hư hỏng, hoặc bị truy cập trái phép bởi những người không có thẩm quyền, làm lộ thông tin nhạy cảm của bệnh nhân.
4. Không thể xử lý các nghiệp vụ đồng thời: Khi nhiều nhân viên (lễ tân, bác sĩ, y tá, kế toán) cùng lúc muốn truy cập hoặc cập nhật thông tin, hệ thống thủ công sẽ không thể đáp ứng, gây ra xung đột và ách tắc.

Trước những thách thức đó, việc xây dựng một Hệ quản trị cơ sở dữ liệu (HQTCSDL) là giải pháp căn cơ và hiệu quả. Một CSDL được thiết kế tốt sẽ giải quyết triệt để các vấn đề

trên bằng cách cung cấp một nền tảng lưu trữ dữ liệu tập trung, an toàn, nhất quán và có khả năng truy vấn mạnh mẽ.

Mục tiêu của đề tài "Quản lý bệnh viện" là tiến hành phân tích, thiết kế và cài đặt một cơ sở dữ liệu quan hệ, cho phép quản lý hiệu quả các nghiệp vụ cốt lõi của bệnh viện, bao gồm: quản lý thông tin bệnh nhân, quản lý bác sĩ và khoa, tổ chức và sắp xếp lịch khám, và lập hóa đơn thanh toán. Hệ thống này sẽ là nền tảng vững chắc để bệnh viện tối ưu hóa quy trình vận hành, nâng cao chất lượng dịch vụ khám chữa bệnh và đảm bảo tính minh bạch, an toàn cho toàn bộ dữ liệu y tế.

BẢNG CÁC TỪ VIẾT TẮT

STT	TỪ VIẾT TẮT	VIẾT ĐẦY ĐỦ
1	CSDL	Cơ sở dữ liệu
2	HQTCSDL	Hệ quản trị cơ sở dữ liệu
3	DDL	Data Definition Language
4	DML	Data Manipulation Language
5	DCL	Data Control Language
6	SQL	Structured Query Language
7	PK	Primary Key
8	FK	Foreign Key
9	RDBMS	Relational Database Management System
10	SSMS	SQL Server Management Studio
11	IDE	Integrated Development Environment (Môi trường phát triển tích hợp)
12	ER	Entity–Relationship
13	TDE	Transparent Data Encryption (Mã hóa dữ liệu toàn bộ ở mức lưu trữ)
14	SSL	Secure Sockets Layer
15	TLS	Transport Layer Security
16	T-SQL	Transact-SQL
17	DMVs	Dynamic Management Views (Các khung nhìn quản lý động)

18	ISO 27001	International Standard for Information Security Management (Tiêu chuẩn quốc tế về quản lý an toàn thông tin)
19	GDPR	General Data Protection Regulation (Quy định bảo vệ dữ liệu chung – EU)

CHƯƠNG 1: GIỚI THIỆU

1.1. Tóm tắt về cơ sở dữ liệu:

1.1.1. Khái niệm Cơ sở dữ liệu (Database):

CSDL là một bộ sưu tập có tổ chức của các dữ liệu có liên quan logic với nhau, được lưu trữ và truy xuất bằng hệ thống máy tính. Thay vì lưu trữ dữ liệu một cách rời rạc trong các tệp tin riêng lẻ, CSDL gom chúng lại vào một kho lưu trữ chung, được thiết kế để loại bỏ sự trùng lặp và tối ưu hóa cho việc truy vấn. Trong đề tài này, CSDL sẽ chứa các bảng dữ liệu như BenhNhan, BacSi, Khoa, LichKham, HoaDon.

1.1.2. Khái niệm Hệ quản trị Cơ sở dữ liệu (DBMS):

HQTCSQL là một hệ thống phần mềm chuyên dụng, đóng vai trò như một "người gác cổng" và "quản lý" cho CSDL. Nó cung cấp một giao diện cho người dùng và ứng dụng để tương tác với dữ liệu mà không cần quan tâm đến chi tiết lưu trữ vật lý. Các chức năng cốt lõi của một HQTCSQL bao gồm:

- **Ngôn ngữ Định nghĩa Dữ liệu (DDL - Data Definition Language):** Cung cấp các lệnh cho phép định nghĩa cấu trúc logic của CSDL, bao gồm tạo, sửa đổi và xóa các bảng, cũng như thiết lập các ràng buộc (như khóa chính, khóa ngoại) để đảm bảo tính đúng đắn của dữ liệu.
- **Ngôn ngữ Thao tác Dữ liệu (DML - Data Manipulation Language):** Cung cấp các lệnh (chủ yếu là SQL - Structured Query Language) để thực hiện các thao tác cơ bản: truy vấn (SELECT), chèn (INSERT), cập nhật (UPDATE) và xóa (DELETE) dữ liệu.
- **Ngôn ngữ Điều khiển Dữ liệu (DCL - Data Control Language):** Quản lý quyền truy cập và bảo mật. Nó cho phép cấp (GRANT) hoặc thu hồi (REVOKE) quyền của người dùng đối với các hành động cụ thể trên các đối tượng dữ liệu, đảm bảo rằng chỉ những người có thẩm quyền mới được xem hoặc sửa đổi thông tin nhạy cảm.
- **Quản lý giao dịch (Transaction Management):** Đảm bảo tính toàn vẹn của dữ liệu ngay cả khi có lỗi hệ thống hoặc nhiều người dùng truy cập cùng lúc. Nó đảm bảo các thuộc tính ACID (Atomicity, Consistency, Isolation, Durability).

1.1.3. Vai trò của CSDL và HQTCSDL trong đề tài "Quản lý bệnh viện":

Trong phạm vi đề tài, việc áp dụng CSDL và HQTCSDL mang lại những lợi ích then chốt để giải quyết các vấn đề đã nêu ở phần mở đầu:

- Loại bỏ trùng lặp và đảm bảo tính nhất quán (Consistency):
 - Thay vì nhập tên, tuổi, địa chỉ bệnh nhân nhiều lần, hệ thống chỉ lưu thông tin này một lần duy nhất trong bảng BenhNhan với một MaBenzNhan (PK) duy nhất.
 - Các bảng khác như LichKham hay HoaDon khi cần thông tin bệnh nhân sẽ chỉ cần tham chiếu đến MaBenzNhan này (qua khóa ngoại - FK). Khi thông tin bệnh nhân (ví dụ: số điện thoại) thay đổi, chỉ cần cập nhật tại một nơi duy nhất.
- Đảm bảo tính toàn vẹn dữ liệu (Integrity):
 - HQTCSDL cho phép định nghĩa các ràng buộc chặt chẽ. Ví dụ:
 - Toàn vẹn thực thể: MaBenzNhan trong bảng BenhNhan phải là duy nhất và không được để trống (PK).
 - Toàn vẹn tham chiếu: Hệ thống sẽ không cho phép tạo một LichKham với MaBacSi không tồn tại trong bảng BacSi, hoặc xóa một Khoa nếu vẫn còn BacSi đang làm việc tại khoa đó (nhờ ràng buộc khóa ngoại FK).
 - Toàn vẹn miền: Đảm bảo dữ liệu nhập vào là hợp lệ (ví dụ: cột GioiTinh chỉ nhận giá trị 'Nam' hoặc 'Nữ').
 - Truy xuất dữ liệu hiệu quả và linh hoạt:
 - Ngôn ngữ SQL cho phép thực hiện các truy vấn phức tạp một cách nhanh chóng mà hồ sơ giấy không thể làm được.
 - Ví dụ nghiệp vụ:
 - (Lễ tân) "Tìm tất cả các khung giờ còn trống của bác sĩ 'Nguyễn Văn A' chuyên khoa 'Tim mạch' vào tuần tới."
 - (Bác sĩ) "Hiển thị toàn bộ lịch sử khám bệnh và các hóa đơn trước đây của bệnh nhân có mã 'BN001'."
 - (Quản lý) "Thống kê tổng doanh thu của bệnh viện trong tháng 10, nhóm theo từng khoa."

- An ninh và Bảo mật (Security):
- HQTCSDL cho phép phân quyền chi tiết. Ví dụ:
 - Nhân viên lễ tân: Chỉ có quyền xem và tạo mới LichKham.
 - Bác sĩ: Có quyền xem LichKham của mình và xem/cập nhật hồ sơ bệnh án của bệnh nhân mình phụ trách.
 - Nhân viên kế toán: Chỉ có quyền xem và tạo HoaDon.
 - Giám đốc: Có quyền xem tất cả các báo cáo tổng hợp.

Điều này đảm bảo thông tin y tế nhạy cảm của bệnh nhân được bảo vệ nghiêm ngặt.

1.2. Công cụ và công nghệ sử dụng:

- Hệ quản trị CSDL: Microsoft SQL Server Đây là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) được chọn để lưu trữ, xử lý và quản lý toàn bộ dữ liệu của bệnh viện.
- Công cụ quản lý và phát triển: SQL Server Management Studio (SSMS) Đây là môi trường tích hợp (IDE) chính, được sử dụng để thiết kế, viết lệnh T-SQL (để tạo bảng, ràng buộc, chèn dữ liệu) và quản trị cơ sở dữ liệu SQL Server.

1.3. Bảng phân công:

STT	Tên công việc	Người thực hiện
1	Lời nói đầu	Tạ Ánh My
2	Chương 1: Giới thiệu	Tạ Ánh My
3	Chương 2: Phân tích yêu cầu hệ thống	Tạ Ánh My
4	Chương 3: Thiết kế cơ sở dữ liệu	Tạ Ánh My
5	Chương 4: Cài đặt cơ sở dữ liệu	Nguyễn Anh Thư
6	Chương 5: Các đối tượng cơ sở dữ liệu nâng cao	Nguyễn Anh Thư
7	Chương 6: Bảo mật và quản trị cơ sở dữ liệu	Nguyễn Anh Thư
8	Chương 7: Kết luận và khuyến nghị	Nguyễn Anh Thư

CHƯƠNG 2: PHÂN TÍCH YÊU CẦU HỆ THỐNG

Chương này tập trung vào việc xác định các yêu cầu nghiệp vụ cơ bản mà hệ thống cơ sở dữ liệu cần phải đáp ứng, cũng như liệt kê các chức năng chính mà hệ thống sẽ hỗ trợ.

2.1. Phân tích và yêu cầu cơ sở dữ liệu:

2.1.1. Mục tiêu của hệ thống:

Hệ thống Quản lý bệnh viện được xây dựng với mục tiêu lưu trữ, xử lý và quản lý toàn bộ thông tin liên quan đến hoạt động khám chữa bệnh của bệnh viện. Cơ sở dữ liệu được thiết kế nhằm:

- Quản lý thông tin bệnh nhân một cách đầy đủ, chính xác và nhát quán.
- Quản lý bác sĩ, khoa chuyên môn, và mối quan hệ giữa bác sĩ với khoa.
- Theo dõi lịch khám bệnh của từng bác sĩ và bệnh nhân, tránh trùng lặp thời gian khám.
- Quản lý và tra cứu hóa đơn khám chữa bệnh, phục vụ thống kê doanh thu và báo cáo.
- Hỗ trợ bệnh viện tra cứu, cập nhật và báo cáo dữ liệu một cách nhanh chóng, chính xác.

2.1.2. Phân tích yêu cầu dữ liệu (nghiệp vụ hệ thống):

Căn cứ vào mục tiêu và đề tài, cơ sở dữ liệu cần hỗ trợ các yêu cầu nghiệp vụ chính sau:

- Quản lý thông tin bệnh nhân:

- Lưu trữ thông tin cá nhân: mã bệnh nhân, họ tên, ngày sinh, giới tính, địa chỉ, số điện thoại.
- Cho phép thêm mới, sửa đổi, xóa hoặc tìm kiếm thông tin bệnh nhân theo mã hoặc tên.
- Đảm bảo mỗi bệnh nhân có mã duy nhất (Primary Key).

- Quản lý thông tin bác sĩ:

- Lưu trữ thông tin bác sĩ gồm mã, tên, khoa làm việc và chuyên khoa.
- Một bác sĩ chỉ thuộc về một khoa duy nhất, đảm bảo mỗi quan hệ 1–N (Khoa – Bác sĩ).
- Có thể tra cứu danh sách bác sĩ theo khoa hoặc chuyên khoa.

- Quản lý thông tin khoa:
 - Lưu trữ danh sách các khoa trong bệnh viện, gồm mã khoa và tên khoa.
 - Là bảng dùng để tham chiếu khóa ngoại (FK) cho bảng Bác sĩ.
 - Hỗ trợ thêm, sửa, xóa, thống kê số lượng bác sĩ thuộc từng khoa.
- Quản lý lịch khám bệnh:
 - Ghi nhận thông tin mỗi lần khám bệnh: mã lịch khám, mã bác sĩ, mã bệnh nhân, ngày khám, giờ khám.
 - Cho phép sắp xếp lịch khám, tránh trùng lịch giữa bác sĩ và bệnh nhân.
 - Có thể tìm kiếm lịch khám theo ngày, theo bác sĩ hoặc theo bệnh nhân.
- Quản lý hóa đơn khám chữa bệnh:
 - Lưu trữ mã hóa đơn, mã bệnh nhân, ngày lập hóa đơn và tổng tiền.
 - Cho phép truy xuất hóa đơn theo bệnh nhân hoặc theo ngày.
 - Hỗ trợ thống kê doanh thu theo khoảng thời gian (tháng/quý/năm).

2.2. Các chức năng chính của hệ thống:

Hệ thống cơ sở dữ liệu quản lý bệnh viện cần hỗ trợ các chức năng chính sau:

2.2.1. Chức năng quản lý người dùng hệ thống:

- Cho phép quản trị viên (Admin) đăng nhập, thêm, sửa, xóa và tra cứu dữ liệu trong các bảng.
- Các người dùng khác (ví dụ nhân viên y tế, bác sĩ) có thể được cấp quyền đọc dữ liệu.

2.2.2. Chức năng quản lý bệnh nhân:

- Thêm bệnh nhân mới với đầy đủ thông tin: mã bệnh nhân, tên, ngày sinh, giới tính, địa chỉ, số điện thoại.
- Cập nhật thông tin khi bệnh nhân thay đổi địa chỉ hoặc số điện thoại.
- Tra cứu bệnh nhân theo mã, tên hoặc ngày sinh.

- Xóa thông tin bệnh nhân khi không còn trong hệ thống.

2.2.3. *Chức năng quản lý bác sĩ:*

- Thêm mới bác sĩ kèm thông tin về mã khoa và chuyên khoa.
- Cập nhật thông tin chuyên khoa hoặc chuyển khoa công tác.
- Tra cứu danh sách bác sĩ theo khoa, hoặc theo tên bác sĩ.
- Thông kê số lượng bác sĩ theo từng khoa.

2.2.4. *Chức năng quản lý khoa:*

- Thêm mới, chỉnh sửa hoặc xóa khoa trong bệnh viện.
- Xem danh sách các khoa hiện có.
- Liệt kê các bác sĩ trực thuộc từng khoa.

2.2.5. *Chức năng quản lý lịch khám:*

- Tạo lịch khám mới, xác định bác sĩ phụ trách và bệnh nhân tương ứng.
- Kiểm tra để tránh trùng lịch (một bác sĩ không thể khám hai bệnh nhân cùng thời điểm).
- Cập nhật hoặc hủy lịch khám khi có thay đổi.
- Tra cứu lịch khám theo ngày, theo bác sĩ, hoặc theo bệnh nhân.

2.2.6. *Chức năng quản lý hóa đơn:*

- Lập hóa đơn cho từng bệnh nhân sau khi hoàn tất khám chữa bệnh.
- Tính tổng tiền và lưu trữ ngày lập hóa đơn.
- Tra cứu hóa đơn theo mã bệnh nhân, theo ngày hoặc theo khoảng thời gian.
- Thông kê doanh thu theo từng tháng hoặc theo khoa điều trị.

2.2.7. *Chức năng báo cáo – thống kê:*

- Thông kê tổng số bệnh nhân đã khám theo từng tháng.
- Thông kê số lượng bác sĩ của từng khoa.
- Báo cáo doanh thu tổng hợp theo ngày/tháng/năm.

2.3. Mô tả mối quan hệ giữa các thực thể

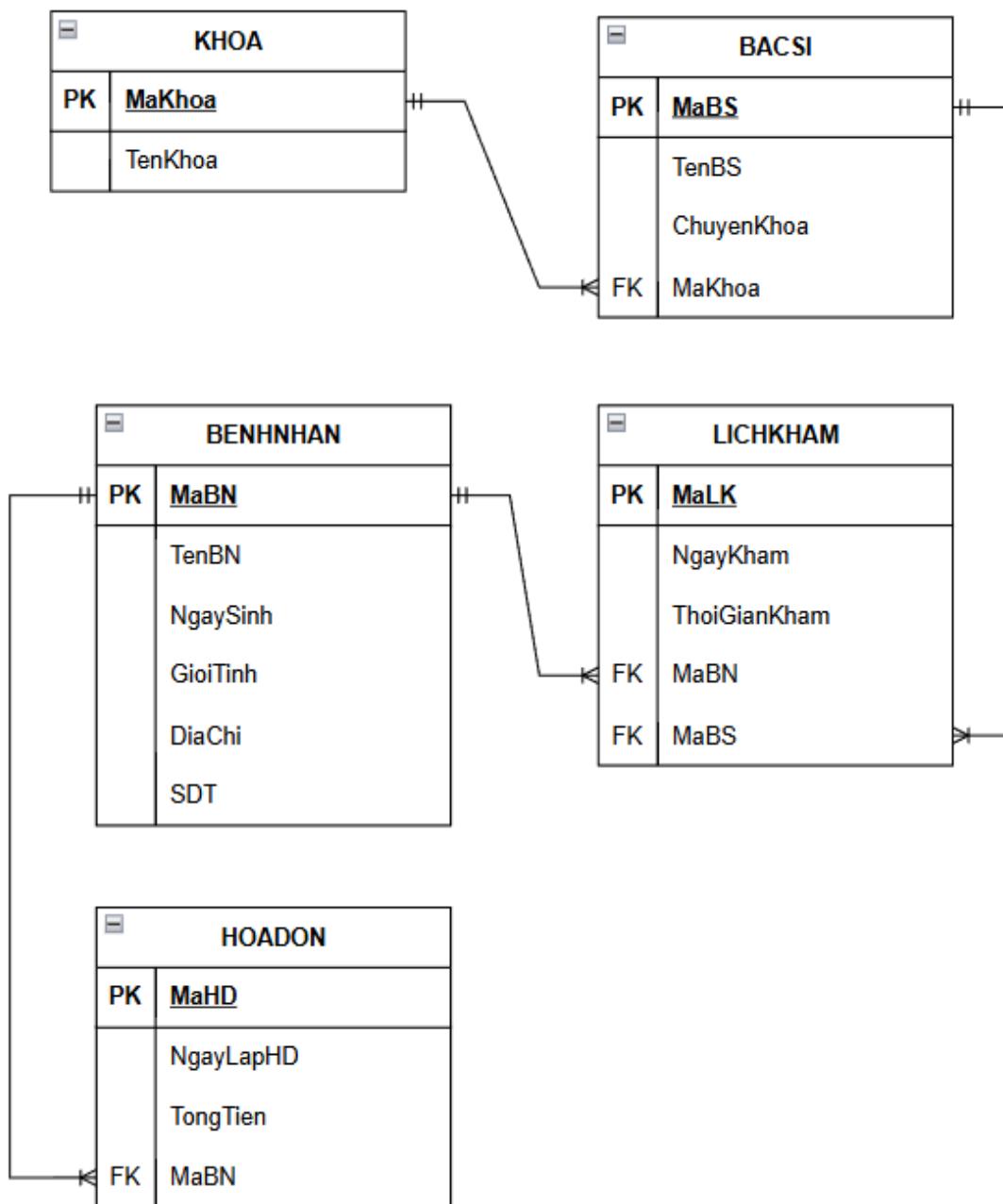
Để hệ thống hoạt động logic và đảm bảo toàn vẹn dữ liệu, các bảng có mối quan hệ như sau:

Thực thể	Mối quan hệ	Thực thể liên quan	Mô tả
Khoa	1 - N	Bác sĩ	Một khoa có nhiều bác sĩ, mỗi bác sĩ chỉ thuộc một khoa
Bác sĩ	1 - N	Lịch khám	Một bác sĩ có thể có nhiều lịch khám
Bệnh nhân	1 - N	Lịch khám	Một bệnh nhân có thể có nhiều lịch khám
Bệnh nhân	1 - N	Hóa đơn	Một bệnh nhân có thể có nhiều hóa đơn khám bệnh

CHƯƠNG 3: THIẾT KẾ CƠ SỞ DỮ LIỆU

3.1. Thiết kế mô hình thực thể quan hệ (ER)

Dựa trên yêu cầu nghiệp vụ và các chức năng chính của hệ thống, ta vẽ được mô hình ER như sau:



3.2. Chuyển đổi mô hình ER thành lược đồ quan hệ:

- BENHNHAN (MaBN, TenBN, NgaySinh, GioiTinh, DiaChi, SDT)
- BACSI (MaBS, TenBS, ChuyenKhoa, MaKhoa)
- KHOA (MaKhoa, TenKhoa)
- HOADON (MaHD, NgayLapHD, TongTien, MaBN)
- LICHKHAM (MaLK, NgayKham, ThoiGianKham, MaBS, MaBN)

3.3 Cấu trúc các bảng dữ liệu:

- **Bảng BENHNHAN:**

STT	Tên Trường	Kiểu dữ liệu	Độ rộng	Ràng buộc	Giải thích
1	MaBN	Varchar	10	PK	Khóa chính
2	TenBN	Nvarchar	50	Not Null	Không rỗng, Tên bệnh nhân
3	NgaySinh	Date		Not Null	Không rỗng, Ngày sinh bệnh nhân
4	GioiTinh	Nvarchar	5	Not Null, Check	Không rỗng, Giới tính (Nam/Nữ)
5	DiaChi	Nvarchar	100		Địa chỉ cư trú
6	SDT	Varchar	15	Not Null	Không rỗng, Số điện thoại liên hệ

- **Bảng BACSI:**

STT	Tên Trường	Kiểu dữ liệu	Độ rộng	Ràng buộc	Giải thích
1	MaBS	Varchar	10	PK	Khóa chính
2	TenBS	Nvarchar	50	Not Null	Không rỗng, Tên bác sĩ
3	ChuyenKhoa	Nvarchar	100	Not Null	Không rỗng, Tên khoa của bác sĩ
4	MaKhoa	Varchar	10	FK	Khóa ngoại liên kết đến mã khoa

- **Bảng KHOA:**

STT	Tên Trường	Kiểu dữ liệu	Độ rộng	Ràng buộc	Giải thích
1	MaKhoa	Varchar	10	PK	Khóa chính
2	TenKhoa	Nvarchar	50	Not Null	Không rỗng, Tên khoa trong bệnh viện

- **Bảng HOADON:**

STT	Tên Trường	Kiểu dữ liệu	Độ rộng	Ràng buộc	Giải thích
1	MaHD	Varchar	10	PK	Khóa chính
2	NgayLapHD	Date		Not Null	Không rỗng, Ngày lập hóa đơn
3	TongTien	Decimal	(18, 2)	Not Null Check (≥ 0)	Không rỗng, Không âm
4	MaBN	Varchar	10	FK	Khóa ngoại liên kết đến mã bệnh nhân tương ứng

- **Bảng LICHKHAM:**

STT	Tên Trường	Kiểu dữ liệu	Độ rộng	Ràng buộc	Giải thích
1	MaLK	Varchar	10	PK	Khóa chính
2	NgayKham	Date		Not Null	Không rỗng
3	ThoiGianKham	Time			Thời gian khám
4	MaBN	Varchar	10	FK	Khóa ngoại liên kết đến mã bệnh nhân tương ứng
5	MaBS	Varchar	10	FK	Khóa ngoại liên kết đến mã bác sĩ thực hiện khám

CHƯƠNG 4: CÀI ĐẶT CƠ SỞ DỮ LIỆU

4.1. Tạo bảng:

```
USE MASTER;
GO
ALTER DATABASE QLBENHVIEN SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO
IF EXISTS (SELECT * FROM SYS.DATABASES WHERE NAME = 'QLBENHVIEN')
    DROP DATABASE QLBENHVIEN;
GO
CREATE DATABASE QLBENHVIEN;
GO
USE QLBENHVIEN;
GO
/*Tạo các bảng trên và khai báo khóa chính, khóa ngoại.*/
CREATE TABLE BENHNHAN (
    MaBN VARCHAR(10) PRIMARY KEY,
    TenBN NVARCHAR(50) NOT NULL,
    NgaySinh DATE NOT NULL,
    GioiTinh NVARCHAR(5) NOT NULL CHECK (GioiTinh IN (N'Nam',
    N'Nữ')),
    DiaChi NVARCHAR(100),
    SDT VARCHAR(15)
);
GO
CREATE TABLE KHOA (
    MaKhoa VARCHAR(10) PRIMARY KEY,
    TenKhoa NVARCHAR(50) NOT NULL
);
GO
CREATE TABLE BACSI (
    MaBS VARCHAR(10) PRIMARY KEY,
    TenBS NVARCHAR(50) NOT NULL,
    ChuyenKhoa NVARCHAR(100) NOT NULL,
    MaKhoa VARCHAR(10) References KHOA(MaKhoa)
);
GO
CREATE TABLE HOADON (
    MaHD VARCHAR(10) PRIMARY KEY,
    MaBN VARCHAR(10) NOT NULL,
```

```

        NgayLapHD DATE NOT NULL DEFAULT GETDATE(),
        TongTien DECIMAL(18,2) NOT NULL CHECK (TongTien >= 0),
        CONSTRAINT FK_HOADON_BENHNHAN FOREIGN KEY (MaBN) REFERENCES
BENHNHAN(MaBN)
);

GO
CREATE TABLE LICHKHAM (
    MaLK VARCHAR(10) PRIMARY KEY,
    NgayKham DATE NOT NULL DEFAULT GETDATE(),
    ThoiGianKham TIME NOT NULL,
    MaBS VARCHAR(10) NOT NULL,
    MaBN VARCHAR(10) NOT NULL,
    CONSTRAINT FK_LICHKHAM_BACSI FOREIGN KEY (MaBS) REFERENCES
BACSI(MaBS),
    CONSTRAINT FK_LICHKHAM_BENHNHAN FOREIGN KEY (MaBN) REFERENCES
BENHNHAN(MaBN)
);
GO

/*--- Dữ liệu KHOA ---*/
INSERT INTO KHOA VALUES
('K01', N'Nội Tổng Hợp'),
('K02', N'Ngoại Chấn Thương'),
('K03', N'Nhi Khoa'),
('K04', N'Tim Mạch'),
('K05', N'Tai Mũi Họng');

/*--- Dữ liệu BỆNH NHÂN ---*/
INSERT INTO BENHNHAN VALUES
('BN01', N'Nguyễn Văn Tuấn', '1990-03-15', N'Nam', N'Hà Nội',
'0912345678'),
('BN02', N'Trần Thị Bích', '1985-07-22', N'Nữ', N'Hải Phòng',
'0987654321'),
('BN03', N'Lê Văn Chiến', '2000-12-05', N'Nam', N'Nam Định',
'0909123456'),
('BN04', N'Phạm Thị Dung', '1995-05-10', N'Nữ', N'Ninh Bình',
'0934567890'),
('BN05', N'Đỗ Văn Huy', '1988-09-19', N'Nam', N'Hà Nam',
'0978123456');

/*--- Dữ liệu BÁC SĨ ---*/
INSERT INTO BACSI VALUES
('BS01', N'Nguyễn Hữu Minh', N'Nội tổng hợp', 'K01'),

```

```

('BS02', N'Trần Văn Thắng', N'Ngoại tổng hợp', 'K02'),
('BS03', N'Lê Thị Hồng', N'Nhi khoa', 'K03'),
('BS04', N'Phạm Văn Nam', N'Tim mạch', 'K04'),
('BS05', N'Hoàng Thị Lan', N'Tai Mũi Họng', 'K05');

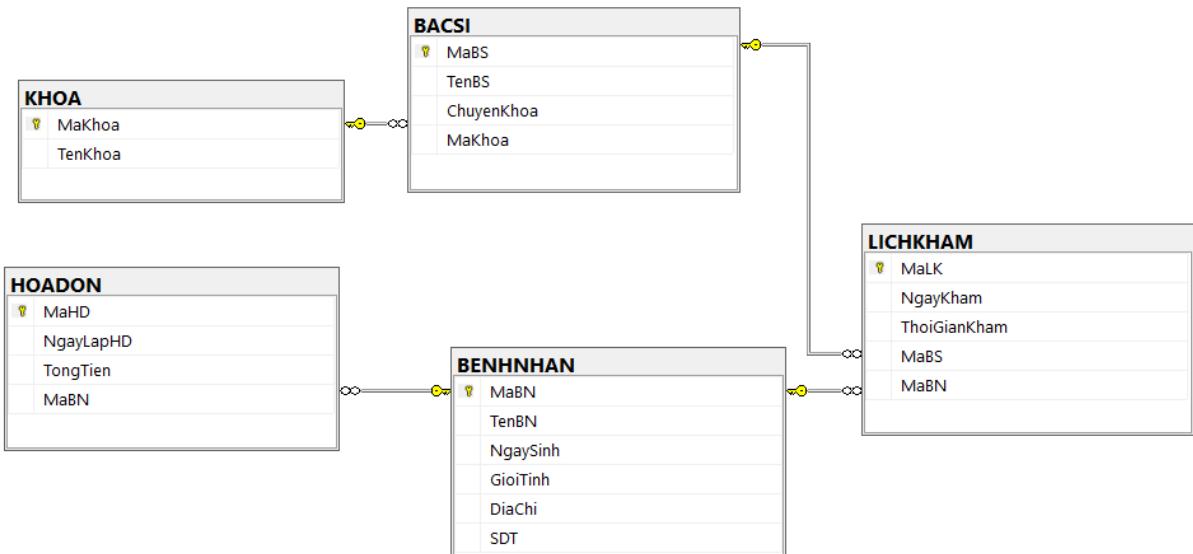
/*--- Dữ liệu LỊCH KHÁM ---*/
INSERT INTO LICHKHAM VALUES
('LK01', CAST(GETDATE() AS DATE), '08:30', 'BS01', 'BN01'),
('LK02', '2025-11-03', '09:00', 'BS02', 'BN02'),
('LK03', '2025-11-04', '10:15', 'BS03', 'BN03'),
('LK04', CAST(GETDATE() AS DATE), '13:45', 'BS04', 'BN04'),
('LK05', '2025-11-05', '15:30', 'BS05', 'BN05');

/*--- Dữ liệu HÓA ĐƠN ---*/
INSERT INTO HOADON VALUES
('HD01', 'BN01', '2025-11-02', 250000),
('HD02', 'BN02', '2025-11-03', 350000),
('HD03', 'BN03', '2025-11-04', 150000),
('HD04', 'BN04', '2025-11-04', 500000),
('HD05', 'BN05', '2025-11-05', 420000);

```

GO

4.2. Sơ đồ cơ sở dữ liệu:



CHƯƠNG 5: CÁC ĐỐI TƯỢNG CƠ SỞ DỮ LIỆU NÂNG CAO

5.1. Cài đặt các chỉ mục (Indexes)

- IDX_BENHNHAN_TenBN: Tăng tốc độ tìm kiếm bệnh nhân theo tên
 - Câu lệnh SQL tạo chỉ mục:
`CREATE INDEX IDX_BENHNHAN_TenBN ON BENHNHAN(TenBN);`
- IDX_BENHNHAN_SDT: Tăng tốc độ truy vấn khi tra cứu bệnh nhân theo số điện thoại.
 - Câu lệnh SQL tạo chỉ mục:
`CREATE INDEX IDX_BENHNHAN_SDT ON BENHNHAN(SDT);`
- IDX_LICHKHAM_NgayKham: Tối ưu các truy vấn lọc danh sách bác sĩ theo khoa
 - Câu lệnh SQL tạo chỉ mục
`CREATE INDEX IDX_LICHKHAM_NgayKham ON LICHKHAM(NgayKham);`
- IDX_BACSI_MaKhoa: Tối ưu các truy vấn lọc danh sách bác sĩ theo khoa
 - Câu lệnh SQL tạo chỉ mục
`CREATE INDEX IDX_BACSI_MaKhoa ON BACSI(MaKhoa);`
- IDX_HOADON_MaBN: Tăng tốc độ truy vấn hóa đơn theo mã bệnh nhân.
 - Câu lệnh SQL tạo chỉ mục
`CREATE INDEX IDX_HOADON_MaBN ON HOADON(MaBN);`

5.2. Views

- VW_ThongTinKhamBenh: Hiển thị thông tin khám bệnh chi tiết (bệnh nhân, bác sĩ, khoa, ngày khám).
 - Câu lệnh tạo View:
`CREATE VIEW VW_ThongTinKhamBenh AS
SELECT LK.MaLK, BN.TenBN, BS.TenBS, K.TenKhoa, LK.NgayKham,
LK.ThoiGianKham
FROM LICHKHAM LK
JOIN BENHNHAN BN ON LK.MaBN = BN.MaBN
JOIN BACSI BS ON LK.MaBS = BS.MaBS
JOIN KHOA K ON BS.MaKhoa = K.MaKhoa;`

- Kết quả của view

	MaLK	TenBN	TenBS	TenKhoa	NgayKham	ThoiGianKham
1	LK01	Nguyễn Văn Tuân	Nguyễn Hữu Minh	Nội Tổng Hợp	2025-11-02	08:30:00.0000000
2	LK02	Trần Thị Bích	Trần Văn Thắng	Ngoại Chấn Thương	2025-11-03	09:00:00.0000000
3	LK03	Lê Văn Chiến	Lê Thị Hồng	Nhi Khoa	2025-11-04	10:15:00.0000000
4	LK04	Phạm Thị Dung	Phạm Văn Nam	Tim Mạch	2025-11-04	13:45:00.0000000
5	LK05	Đỗ Văn Huy	Hoàng Thị Lan	Tai Mũi Họng	2025-11-05	15:30:00.0000000

- VW_HoaDonBenhNhan: Hiển thị tổng tiền hóa đơn của từng bệnh nhân.

- Câu lệnh tạo view

```
CREATE VIEW VW_HoaDonBenhNhan AS
SELECT HD.MaHD, BN.TenBN, HD.NgayLapHD, HD.TongTien
FROM HOADON HD
JOIN BENHNHAN BN ON HD.MaBN = BN.MaBN;
```

- Kết quả của view

	MaHD	TenBN	NgayLapHD	TongTien
1	HD01	Nguyễn Văn Tuân	2025-11-02	250000.00
2	HD02	Trần Thị Bích	2025-11-03	350000.00
3	HD03	Lê Văn Chiến	2025-11-04	150000.00
4	HD04	Phạm Thị Dung	2025-11-04	500000.00
5	HD05	Đỗ Văn Huy	2025-11-05	420000.00

- VW_BacSiKhoa: Hiển thị danh sách bác sĩ theo từng khoa.

- Câu lệnh tạo view:

```
CREATE VIEW VW_BacSiKhoa AS
SELECT BS.MaBS, BS.TenBS, BS.ChuyenKhoa, K.TenKhoa
FROM BACSI BS JOIN KHOA K ON BS.MaKhoa = K.MaKhoa;
```

- Kết quả của view:

	MaBS	TenBS	ChuyenKhoa	TenKhoa
1	BS01	Nguyễn Hữu Minh	Nội tổng hợp	Nội Tổng Hợp
2	BS02	Trần Văn Thắng	Ngoại tổng hợp	Ngoại Chấn Thương
3	BS03	Lê Thị Hồng	Nhi khoa	Nhi Khoa
4	BS04	Phạm Văn Nam	Tim mạch	Tim Mạch
5	BS05	Hoàng Thị Lan	Tai Mũi Họng	Tai Mũi Họng

- VW_LichKhamHomNay: Hiển thị danh sách lịch khám trong ngày hiện tại.

- Câu lệnh tạo view:

```
CREATE VIEW VW_LichKhamHomNay AS
SELECT LK.MaLK, BN.TenBN, BS.TenBS, LK.ThoiGianKham
```

```

        FROM LICHKHAM LK
        JOIN BENHNHAN BN ON LK.MaBN = BN.MaBN
        JOIN BACSI BS ON LK.MaBS = BS.MaBS
        WHERE LK.NgayKham = CAST(GETDATE() AS DATE);
    
```

- Kết quả của view:

	MaLK	TenBN	TenBS	ThoiGianKham
1	LK01	Nguyễn Văn Tuấn	Nguyễn Hữu Minh	08:30:00.0000000
2	LK04	Phạm Thị Dung	Phạm Văn Nam	13:45:00.0000000

- VW_SoLanKham: Hiển thị danh sách bệnh nhân và số lần khám

- Câu lệnh tạo view:

```

CREATE VIEW VW_SoLanKham AS
SELECT BN.TenBN, COUNT(LK.MaLK) AS SoLanKham
FROM BENHNHAN BN LEFT JOIN LICHKHAM LK ON BN.MaBN = LK.MaBN
GROUP BY BN.TenBN;
    
```

- Kết quả của view:

	TenBN	SoLanKham
1	Đỗ Văn Huy	1
2	Lê Văn Chiến	1
3	Nguyễn Văn Tuấn	1
4	Phạm Thị Dung	1
5	Trần Thị Bích	1

5.3. Thủ tục

- sp_DemSoLichKham_BacSi: Đếm số lịch khám của từng bác sĩ.

- Câu lệnh tạo thủ tục:

```

CREATE PROCEDURE sp_DemSoLichKham_BacSi
AS
SELECT BS.TenBS, COUNT(LK.MaLK) AS SoLichKham
FROM BACSI BS LEFT JOIN LICHKHAM LK ON BS.MaBS = LK.MaBS
GROUP BY BS.TenBS;
    
```

- Câu lệnh gọi thủ tục:

```
EXEC sp_DemSoLichKham_BacSi
```

- Kết quả của thủ tục:

TenBS	SoLichKham
1 Hoàng Thị Lan	1
2 Lê Thị Hồng	1
3 Nguyễn Hữu Minh	1
4 Phạm Văn Nam	1
5 Trần Văn Thắng	1

- sp_TimBenhNhanTheoMa: Tìm kiếm bệnh nhân theo mã

- Câu lệnh tạo thủ tục:

```
CREATE PROCEDURE sp_TimBenhNhanTheoMa @MaBN VARCHAR(10)
AS SELECT * FROM BENHNHAN WHERE MaBN = @MaBN;
```

- Câu lệnh gọi thủ tục:

```
EXEC sp_TimBenhNhanTheoMa @MaBN = 'BN01';
```

- Kết quả của thủ tục:

	MaBN	TenBN	NgaySinh	GioiTinh	DiaChi	SDT
1	BN01	Nguyễn Văn Tuấn	1990-03-15	Nam	Hà Nội	0912345678

- sp_TinhTongTienBenhNhan: Tính tổng tiền của tất cả các hóa đơn mà bệnh nhân đã thanh toán.

- Câu lệnh tạo thủ tục:

```
CREATE PROCEDURE sp_TinhTongTienBenhNhan @MaBN VARCHAR(10)
AS
SELECT BN.TenBN, SUM(TongTien) AS TongTien
FROM HOADON HD JOIN BENHNHAN BN ON HD.MaBN = BN.MaBN
WHERE HD.MaBN = @MaBN GROUP BY BN.TenBN;
```

- Câu lệnh gọi thủ tục:

```
EXEC sp_TinhTongTienBenhNhan @MaBN = 'BN03';
```

- Kết quả của thủ tục:

	TenBN	TongTien
1	Lê Văn Chiến	150000.00

- sp_ThemBenhNhan: Thêm bệnh nhân mới

- Câu lệnh tạo thủ tục:

```
CREATE PROCEDURE sp_ThemBenhNhan
    @MaBN VARCHAR(10), @TenBN NVARCHAR(50), @NgaySinh DATE,
    @GioiTinh NVARCHAR(5), @DiaChi NVARCHAR(100), @SDT
    VARCHAR(15)
AS
    INSERT INTO BENHNHAN VALUES (@MaBN, @TenBN, @NgaySinh,
    @GioiTinh, @DiaChi, @SDT);
```

- Câu lệnh gọi thủ tục:

```
EXEC sp_ThemBenhNhan
    @MaBN = 'BN10',
    @TenBN = N'Lê Văn Nam',
    @NgaySinh = '1995-10-12',
    @GioiTinh = N'Nam',
    @DiaChi = N'Hà Nội',
    @SDT = '0912345678';
```

- Kết quả của thủ tục:

	MaBN	TenBN	NgaySinh	GioiTinh	DiaChi	SDT
1	BN01	Nguyễn Văn Tuấn	1990-03-15	Nam	Hà Nội	0912345678
2	BN02	Trần Thị Bích	1985-07-22	Nữ	Hải Phòng	0987654321
3	BN03	Lê Văn Chiến	2000-12-05	Nam	Nam Định	0909123456
4	BN04	Phạm Thị Dung	1995-05-10	Nữ	Ninh Bình	0934567890
5	BN05	Đỗ Văn Huy	1988-09-19	Nam	Hà Nam	0978123456
6	BN10	Lê Văn Nam	1995-10-12	Nam	Hà Nội	0912345678

- sp_CapNhatDiaChiBenhNhan: Cập nhật địa chỉ mới của bệnh nhân từng đến khám

- Câu lệnh tạo thủ tục:

```
CREATE PROCEDURE sp_CapNhatDiaChiBenhNhan @MaBN VARCHAR(10),
    @DiaChi NVARCHAR(100)
AS
    UPDATE BENHNHAN SET DiaChi = @DiaChi WHERE MaBN = @MaBN;
```

- Câu lệnh gọi thủ tục:

```
EXEC sp_CapNhatDiaChiBenhNhan
    @MaBN = 'BN10',
    @DiaChi = N'Hồ Chí Minh';
SELECT * FROM BENHNHAN WHERE MaBN = 'BN10';
```

- Kết quả của thủ tục:

	MaBN	TenBN	NgaySinh	GioiTinh	DiaChi	SDT
1	BN10	Lê Văn Nam	1995-10-12	Nam	Hồ Chí Minh	0912345678

- sp_XoaBenhNhanTheoMa: Xóa thông tin của bệnh nhân khi không còn cần thiết hoặc nhập sai dữ liệu.

- Câu lệnh tạo thủ tục:

```

CREATE PROCEDURE sp_XoaBenhNhanTheoMa
    @MaBN VARCHAR(10)
AS
BEGIN
    IF EXISTS (SELECT 1 FROM BENHNHAN WHERE MaBN = @MaBN)
    BEGIN
        DELETE FROM BENHNHAN WHERE MaBN = @MaBN;
        PRINT N'Dã xóa bệnh nhân có mã ' + @MaBN;
    END
    ELSE
    BEGIN
        PRINT N'Không tìm thấy bệnh nhân có mã ' + @MaBN;
    END
END;

```

- Câu lệnh gọi thủ tục:

```
EXEC sp_XoaBenhNhanTheoMa @MaBN = 'BN10';
```

- Kết quả của thủ tục:

	MaBN	TenBN	NgaySinh	GioiTinh	DiaChi	SDT
1	BN01	Nguyễn Văn Tuấn	1990-03-15	Nam	Hà Nội	0912345678
2	BN02	Trần Thị Bích	1985-07-22	Nữ	Hải Phòng	0987654321
3	BN03	Lê Văn Chiến	2000-12-05	Nam	Nam Định	0909123456
4	BN04	Phạm Thị Dung	1995-05-10	Nữ	Ninh Bình	0934567890
5	BN05	Đỗ Văn Huy	1988-09-19	Nam	Hà Nam	0978123456

(0 rows affected)

(0 rows affected)

(1 row affected)

(1 row affected)

Đã xóa bệnh nhân có mã BN10

5.4. Hàm người dùng định nghĩa

- fn_TimBenhNhanGanDung: Tìm kiếm nhanh khi người nhập chỉ nhớ một phần tên.

- Câu lệnh tạo hàm:

```
CREATE FUNCTION fn_TimBenhNhanGanDung(@Ten NVARCHAR(50))
RETURNS TABLE
AS
RETURN
(
    SELECT MaBN, TenBN, DiaChi, SDT
    FROM BENHNHAN
    WHERE TenBN LIKE '%' + @Ten + '%'
);
```

- Câu lệnh gọi hàm:

```
SELECT * FROM dbo.fn_TimBenhNhanGanDung(N'Nguyễn');
```

- Kết quả của hàm:

	MaBN	TenBN	DiaChi	SDT
1	BN01	Nguyễn Văn Tuấn	Hà Nội	0912345678

- fn_BacSiPhuTrach: Hiển thị mã và tên bác sĩ gần nhất điều trị bệnh nhân.

- Câu lệnh tạo hàm:

```
CREATE FUNCTION fn_BacSiPhuTrach(@MaBN VARCHAR(10))
RETURNS TABLE
AS
RETURN
(
    SELECT TOP 1
        BS.MaBS,
        BS.TenBS
    FROM LICHKHAM LK
    JOIN BACSI BS ON LK.MaBS = BS.MaBS
    WHERE LK.MaBN = @MaBN
    ORDER BY LK.NgayKham DESC
);
```

- Câu lệnh gọi hàm:

```
SELECT * FROM dbo.fn_BacSiPhuTrach('BN05');
```

- Kết quả của hàm:

Results		Messages
	MaBN	TenBN
1	BS05	Hoàng Thị Lan

- fn_CoLichKhamHomNay: Để tra cứu bệnh nhân nào đang có hẹn trong ngày.
 - Câu lệnh tạo hàm:


```
CREATE FUNCTION fn_CoLichKhamHomNay(@MaBN VARCHAR(10))
RETURNS BIT
AS
BEGIN
    DECLARE @kq BIT = 0;
    IF EXISTS (SELECT * FROM LICHKHAM WHERE MaBN = @MaBN AND
NgayKham = CAST(GETDATE() AS DATE))
        SET @kq = 1;
    RETURN @kq;
END;
```
 - Câu lệnh gọi hàm:


```
SELECT dbo.fn_CoLichKhamHomNay('BN03') AS CoHenHomNay;
```

Results		Messages
	CoHenHomNay	
1	0	

- fn_CoHoaDon: Dùng để kiểm tra bệnh nhân có hóa đơn hay chưa
 - Câu lệnh tạo hàm:


```
CREATE FUNCTION fn_CoHoaDon(@MaBN VARCHAR(10))
RETURNS BIT
AS
BEGIN
    DECLARE @kq BIT = 0;
    IF EXISTS (SELECT * FROM HOADON WHERE MaBN = @MaBN)
        SET @kq = 1;
    RETURN @kq;
END;
```
 - Câu lệnh gọi hàm:


```
SELECT dbo.fn_CoHoaDon('BN01') AS CoHoaDon;
```

- Kết quả của hàm:

Results		Messages
CoHoaDon		
1		1

- fn_SoNgayTuLanKhamGanNhat: Xem bệnh nhân đã bao lâu chưa tái khám.

- Câu lệnh tạo hàm:

```
CREATE FUNCTION fn_SoNgayTuLanKhamGanNhat(@MaBN
VARCHAR(10))
RETURNS INT
AS
BEGIN
    DECLARE @NgayCuoI DATE;
    SELECT @NgayCuoI = MAX(NgayKham) FROM LICHKHAM WHERE MaBN
= @MaBN;
    RETURN DATEDIFF(DAY, @NgayCuoI, GETDATE());
END;
```

- Câu lệnh gọi hàm:

```
SELECT dbo.fn_SoNgayTuLanKhamGanNhat('BN05') AS SoNgay;
```

- Kết quả của hàm:

Results		Messages
SoNgay		
1		2

- fn_BacSiDaCoLichHomNay: Dùng trong hệ thống đặt lịch, tránh trùng lịch bác sĩ.

- Câu lệnh tạo hàm:

```
CREATE FUNCTION fn_BacSiDaCoLichHomNay(@MaBS VARCHAR(10))
RETURNS BIT
AS
BEGIN
    DECLARE @kq BIT = 0;
    IF EXISTS (SELECT * FROM LICHKHAM WHERE MaBS = @MaBS AND
NgayKham = CAST(GETDATE() AS DATE))
        SET @kq = 1;
    RETURN @kq;
END;
```

- Câu lệnh gọi hàm:

```
SELECT dbo.fn_BacSiDaCoLichHomNay('BS01') AS DaCoLich;
```

- Kết quả của hàm:

DaCoLich	
1	0

- fn_DanhSachBenhNhanTheoBacSi: Xem bác sĩ đang phụ trách những bệnh nhân nào.

- Câu lệnh tạo hàm:

```
CREATE FUNCTION fn_DanhSachBenhNhanTheoBacSi(@MaBS
VARCHAR(10))
RETURNS TABLE
AS
RETURN
(
    SELECT DISTINCT BN.MaBN, BN.TenBN, BN.GioiTinh,
BN.NgaySinh
    FROM BENHNHAN BN
    JOIN LICHKHAM LK ON BN.MaBN = LK.MaBN
    WHERE LK.MaBS = @MaBS
);
```

- Câu lệnh gọi hàm:

```
SELECT * FROM dbo.fn_DanhSachBenhNhanTheoBacSi('BS02');
```

- Kết quả của hàm:

Results			
	MaBN	TenBN	GioiTinh
1	BN02	Trần Thị Bích	Nữ

NgaySinh

5.5. Trigger

- trg_KhongXoaBacSiConLich: Đảm bảo toàn vẹn dữ liệu giữa bảng BACSI và LICHKHAM.

- Câu lệnh tạo trigger:

```
CREATE TRIGGER trg_KhongXoaBacSiConLich
ON BACSI
INSTEAD OF DELETE
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM LICHKHAM LK
        JOIN DELETED D ON LK.MaBS = D.MaBS
    )
)
```

```

        BEGIN
            RAISERROR(N'Không thể xóa bác sĩ vì còn lịch khám
liên quan!',16,1);
            ROLLBACK TRANSACTION;
            RETURN;
        END
    ELSE
        DELETE FROM BACSI WHERE MaBS IN (SELECT MaBS FROM
DELETED);
    END;

```

- Câu lệnh gọi trigger:
-- Giả sử bác sĩ BS01 có lịch khám
`DELETE FROM BACSI WHERE MaBS = 'BS01';`
- Kết quả của trigger:

 Messages

```

Msg 50000, Level 16, State 1, Procedure trg_KhongXoaBacSiConLich, Line 11 [Batch Start Line 374]
Không thể xóa bác sĩ vì còn lịch khám liên quan!
Msg 3609, Level 16, State 1, Line 375
The transaction ended in the trigger. The batch has been aborted.

```

- trg_LogBenhNhan: Tự động ghi lại nhật ký (log) khi thêm mới bệnh nhân vào bảng BENHNHAN.

- Câu lệnh tạo trigger:
`IF OBJECT_ID('LogBenhNhan', 'U') IS NOT NULL
 DROP TABLE LogBenhNhan;
GO
CREATE TABLE LogBenhNhan (
 MaBN VARCHAR(10),
 NgayThem DATETIME
);
GO
IF EXISTS (SELECT * FROM sys.triggers WHERE name =
'trg_LogBenhNhan')
 DROP TRIGGER trg_LogBenhNhan;
GO
CREATE TRIGGER trg_LogBenhNhan
ON BENHNHAN
AFTER INSERT
AS
 INSERT INTO LogBenhNhan
 SELECT MaBN, GETDATE() FROM inserted;`

- Câu lệnh gọi trigger:

```
INSERT INTO BENHNHAN VALUES
('BN100', N'Nguyễn Minh An', '2000-05-12', N'Nam', N'Hà Nội',
'0909123456');
```

```
SELECT * FROM LogBenhNhan;
```

- Kết quả của trigger:

	MaBN	NgayThem
1	BN10	2025-11-06 22:30:53.573
2	BN100	2025-11-07 18:14:24.443

- trg_KiemTraTrungLichKham: Kiểm tra xem bác sĩ có bị trùng lịch khám hay không.

- Câu lệnh tạo trigger:

```
CREATE TRIGGER trg_KiemTraTrungLichKham
ON LICHKHAM
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM LICHKHAM L
        JOIN INSERTED I ON L.MaBS = I.MaBS
        AND L.NgayKham = I.NgayKham
        AND L.ThoiGianKham = I.ThoiGianKham
    )
    BEGIN
        RAISERROR(N'Bác sĩ đã có lịch khám tại thời điểm
này!',16,1);
        ROLLBACK TRANSACTION;
        RETURN;
    END
    ELSE
        INSERT INTO LICHKHAM(MaLK, NgayKham, ThoiGianKham,
        MaBS, MaBN)
        SELECT MaLK, NgayKham, ThoiGianKham, MaBS, MaBN FROM
        INSERTED;
    END;
```

- Câu lệnh gọi trigger:

```
INSERT INTO LICHKHAM VALUES
('LK10', '2025-11-03', '09:00', 'BS02', 'BN10');
```

- o Kết quả của trigger:

Messages

```

Msg 50000, Level 16, State 1, Procedure trg_KiemTraTrungLichKham, Line 14 [Batch Start Line 425]
Bác sĩ đã có lịch khám tại thời điểm này!
Msg 3609, Level 16, State 1, Line 426
The transaction ended in the trigger. The batch has been aborted.

```

- trg_XoaBenhNhan: Khi xóa bệnh nhân, hệ thống sẽ tự động xóa các lịch khám và hóa đơn liên quan.

- o Câu lệnh tạo trigger:

```

IF EXISTS (SELECT * FROM sys.triggers WHERE name = 'trg_XoaBenhNhan')
    DROP TRIGGER trg_XoaBenhNhan;
GO
CREATE TRIGGER trg_XoaBenhNhan
ON BENHNHAN
INSTEAD OF DELETE
AS
BEGIN
    DELETE FROM LICHKHAM WHERE MaBN IN (SELECT MaBN FROM deleted);
    DELETE FROM HOADON WHERE MaBN IN (SELECT MaBN FROM deleted);
    DELETE FROM BENHNHAN WHERE MaBN IN (SELECT MaBN FROM deleted);
END;

```

- o Câu lệnh gọi trigger:

```

DELETE FROM BENHNHAN WHERE MaBN = 'BN05';
SELECT * FROM LICHKHAM WHERE MaBN = 'BN05';
SELECT * FROM HOADON WHERE MaBN = 'BN05';

```

- o Kết quả của trigger:

Messages		Results		Messages		
(1 row affected)						
(1 row affected)						
(1 row affected)						
(1 row affected)						
		MaLK	NgayKham	ThoiGianKham	MaBS	MaBN
1	LK01	2025-11-06	08:30:00.0000000	BS01	BN01	
2	LK02	2025-11-03	09:00:00.0000000	BS02	BN02	
3	LK03	2025-11-04	10:15:00.0000000	BS03	BN03	
4	LK04	2025-11-06	13:45:00.0000000	BS04	BN04	
5	LK06	2025-11-07	08:30:00.0000000	BS05	BN06	
		MaHD	MaBN	NgayLapHD	TongTien	
1	HD01	BN01	2025-11-02	250000.00		
2	HD02	BN02	2025-11-03	350000.00		
3	HD03	BN03	2025-11-04	150000.00		
4	HD04	BN04	2025-11-04	500000.00		
5	HD06	BN06	2025-11-07	250000.00		

- trg_ThongBaoCapNhatBS: Thông báo ra màn hình khi một bác sĩ được chuyển sang khoa khác.

- Câu lệnh tạo trigger:

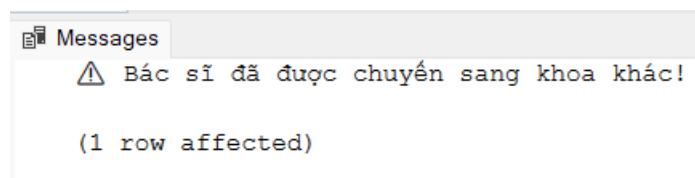
```

IF EXISTS (SELECT * FROM sys.triggers WHERE name =
'trg_ThongBaoCapNhatBS')
DROP TRIGGER trg_ThongBaoCapNhatBS;
GO
CREATE TRIGGER trg_ThongBaoCapNhatBS
ON BACSI
AFTER UPDATE
AS
IF UPDATE(MaKhoa)
    PRINT N'⚠️ Bác sĩ đã được chuyển sang khoa khác!';
  
```

- Câu lệnh gọi trigger:

```
UPDATE BACSI SET MaKhoa = 'K02' WHERE MaBS = 'BS01';
```

- Kết quả của trigger:



- trg_CapNhatTongTien_BenhNhan: Tự động cộng dồn chi phí cho bệnh nhân mỗi khi thêm hóa đơn mới.

- Câu lệnh tạo trigger:

```

--Thêm cột nếu chưa có
IF COL_LENGTH('BENHNHAN', 'TongChiPhi') IS NULL
BEGIN
    ALTER TABLE BENHNHAN ADD TongChiPhi DECIMAL(18,2) DEFAULT
    0;
END
GO
  
```

```

-- Cập nhật tổng chi phí cho tất cả bệnh nhân dựa trên hóa
đơn đã có
UPDATE BN
SET BN.TongChiPhi = ISNULL(Tong.HoaDonTong, 0)
FROM BENHNHAN BN
LEFT JOIN (
    SELECT MaBN, SUM(TongTien) AS HoaDonTong
    FROM HOADON
  
```

```

        GROUP BY MaBN
    ) Tong ON BN.MaBN = Tong.MaBN;
GO

-- Xóa trigger cũ nếu có
IF OBJECT_ID('trg_CapNhatTongTien_BenhNhan', 'TR') IS NOT
NULL
    DROP TRIGGER trg_CapNhatTongTien_BenhNhan;
GO

-- Tạo lại trigger tự động cộng dồn cho hóa đơn mới
CREATE TRIGGER trg_CapNhatTongTien_BenhNhan
ON HOADON
AFTER INSERT
AS
BEGIN
    -- Cập nhật tổng chi phí khi có hóa đơn mới
    UPDATE BN
    SET BN.TongChiPhi = ISNULL(BN.TongChiPhi, 0) + I.TongTien
    FROM BENHNHAN BN
    JOIN INSERTED I ON BN.MaBN = I.MaBN;
END;

```

- Câu lệnh gọi trigger:

```

INSERT INTO HOADON VALUES ('HD07', 'BN06', GETDATE(), 500000);
INSERT INTO HOADON VALUES ('HD08', 'BN06', GETDATE(), 300000);

```

```

SELECT MaBN, TongChiPhi FROM BENHNHAN WHERE MaBN = 'BN06';

```

- Kết quả của trigger:

	MaBN	TongChiPhi
1	BN06	1050000.00

- trg_TuDongSinhMaHoaDon: Tự động sinh mã hóa đơn (MaHD) dạng "HD01", "HD02", ... khi người dùng không nhập mã trong lúc thêm mới.

- Câu lệnh tạo trigger:

```

IF EXISTS (SELECT * FROM sys.triggers WHERE name =
'trg_TuDongSinhMaHoaDon')
    DROP TRIGGER trg_TuDongSinhMaHoaDon;
GO

```

```

CREATE TRIGGER trg_TuDongSinhMaHoaDon
ON HOADON
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @MaxSo INT, @MaMoi VARCHAR(10);

    -- Lấy số lớn nhất trong mã HD hiện có (VD: HD01 → 1, HD120 → 120)
    SELECT @MaxSo = ISNULL(MAX(CAST(SUBSTRING(MaHD, 3, LEN(MaHD)) AS INT)), 0)
    FROM HOADON;

    -- Duyệt tất cả dòng được chèn (có thể INSERT nhiều dòng 1 lúc)
    INSERT INTO HOADON (MaHD, MaBN, NgayLapHD, TongTien)
    SELECT
        -- Nếu người dùng không nhập MaHD, hệ thống tự sinh
        CASE
            WHEN MaHD IS NULL OR MaHD = ''
                THEN 'HD' + RIGHT(REPLICATE('0', 2) +
CAST(@MaxSo + ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS VARCHAR(10)),
CASE
                WHEN @MaxSo + ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) < 100 THEN 2
                ELSE LEN(CAST(@MaxSo + ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS VARCHAR(10)))
            END)
            ELSE MaHD
        END AS MaHD,
        MaBN,
        ISNULL(NgayLapHD, GETDATE()),
        ISNULL(TongTien, 0)
    FROM inserted;
END;

```

- o Câu lệnh gọi trigger:

```

INSERT INTO HOADON (MaBN, TongTien) VALUES ('BN02', 250000);
SELECT * FROM HOADON;

```

- Kết quả của trigger:

	MaHD	MaBN	NgayLapHD	TongTien
1	HD01	BN01	2025-11-02	250000.00
2	HD02	BN02	2025-11-03	350000.00
3	HD03	BN03	2025-11-04	150000.00
4	HD04	BN04	2025-11-04	500000.00
5	HD06	BN06	2025-11-07	250000.00
6	HD07	BN06	2025-11-07	500000.00
7	HD08	BN06	2025-11-07	300000.00
8	HD09	BN02	2025-11-07	250000.00

- trg_LogXoaBenhNhan

- Câu lệnh tạo trigger:

```
CREATE TABLE LOG_XOABENHNHAN (
    MaBN VARCHAR(10),
    TenBN NVARCHAR(50),
    NgayXoa DATETIME DEFAULT GETDATE(),
    NguoiThucHien NVARCHAR(50) DEFAULT SUSER_NAME()
);
GO
```

```
CREATE TRIGGER trg_LogXoaBenhNhan
ON BENHNHAN
AFTER DELETE
AS
BEGIN
    INSERT INTO LOG_XOABENHNHAN(MaBN, TenBN)
    SELECT MaBN, TenBN FROM DELETED;
END;
```

- Câu lệnh gọi trigger:

```
DELETE FROM BENHNHAN WHERE MaBN = 'BN11';
```

```
SELECT * FROM LOG_XOABENHNHAN;
```

- Kết quả của trigger:

Messages	Results	Messages		
	MaBN	TenBN	NgayXoa	NguoiThucHien
(1 row affected)	1	BN11	Nguyễn Đức An	2025-11-07 21:40:11.693
(1 row affected)				DESKTOP-MKPOLRC\Bac Viet
(1 row affected)				
(1 row affected)				

CHƯƠNG 6: BẢO MẬT VÀ QUẢN TRỊ CƠ SỞ DỮ LIỆU

6.1. Quản lý người dùng và quyền truy cập

Trong quá trình vận hành hệ thống cơ sở dữ liệu, việc đảm bảo an toàn thông tin và giới hạn quyền truy cập là yếu tố đặc biệt quan trọng. Hệ thống quản trị cơ sở dữ liệu SQL Server cung cấp cơ chế quản lý người dùng và phân quyền chặt chẽ nhằm kiểm soát hoạt động truy cập, sửa đổi và thao tác dữ liệu. Mục tiêu của việc quản lý này là giúp ngăn chặn truy cập trái phép, đảm bảo tính toàn vẹn và bảo mật của dữ liệu trong toàn hệ thống.

Trong SQL Server, mỗi cá nhân hoặc nhóm người dùng được định danh thông qua LOGIN (tài khoản đăng nhập) và USER (người dùng trong cơ sở dữ liệu). Quản trị viên có thể tạo các tài khoản này để giới hạn phạm vi truy cập cho từng người dùng theo đúng vai trò của họ trong hệ thống. SQL Server cho phép cấp quyền (GRANT) để cho phép người dùng thực hiện một số thao tác như xem, thêm, sửa hoặc xóa dữ liệu, và thu hồi quyền (REVOKE) khi không còn cần thiết. Ngoài ra, cơ chế ROLE giúp nhóm các quyền lại với nhau, sau đó gán cho nhiều người dùng cùng lúc, giúp việc quản lý trở nên dễ dàng, đồng thời giảm rủi ro cấp sai quyền cho cá nhân.

Việc phân quyền hợp lý không chỉ đảm bảo hoạt động của hệ thống diễn ra an toàn mà còn giúp duy trì nguyên tắc “người dùng chỉ được phép thao tác trong phạm vi nhiệm vụ được giao”, góp phần quan trọng trong việc bảo mật tổng thể của cơ sở dữ liệu.

Ví dụ minh họa cho việc tạo và cấp quyền trong SQL Server:

```
CREATE LOGIN UserA WITH PASSWORD = '123456';
CREATE USER UserA FOR LOGIN UserA;
GRANT SELECT, INSERT ON BENHNHAN TO UserA;
```

Trong ví dụ trên, tài khoản UserA được tạo mới với mật khẩu riêng, sau đó được cấp quyền xem (SELECT) và thêm dữ liệu (INSERT) trong bảng BENHNHAN. Điều này giúp người dùng có thể truy cập và thao tác đúng phạm vi cho phép mà không ảnh hưởng đến các phần dữ liệu khác của hệ thống.

Quản lý người dùng và quyền truy cập đóng vai trò cốt lõi trong công tác bảo mật cơ sở dữ liệu. Việc thiết lập cơ chế phân quyền rõ ràng, hợp lý và có kiểm soát không chỉ giúp bảo vệ dữ liệu khỏi truy cập trái phép, mà còn đảm bảo tính minh bạch và an toàn cho toàn bộ

hệ thống. Đây là nền tảng cần thiết để các hoạt động khai thác dữ liệu trong doanh nghiệp diễn ra ổn định, tin cậy và tuân thủ quy định bảo mật thông tin.

6.2. Bảo mật cơ sở dữ liệu

Trong thời đại số hiện nay, dữ liệu được ví như “tài sản vô hình” có giá trị cao nhất của mọi tổ chức. Dữ liệu không chỉ phản ánh hoạt động kinh doanh, hành chính, y tế hay giáo dục, mà còn chứa đựng những thông tin nhạy cảm, riêng tư của cá nhân và doanh nghiệp. Vì vậy, bảo mật cơ sở dữ liệu (Database Security) trở thành một yêu cầu thiết yếu nhằm đảm bảo tính bí mật (Confidentiality), toàn vẹn (Integrity) và sẵn sàng (Availability) của thông tin.

SQL Server – một hệ quản trị cơ sở dữ liệu mạnh mẽ của Microsoft – cung cấp một hệ thống bảo mật đa tầng, giúp kiểm soát truy cập, ngăn chặn tấn công, mã hóa dữ liệu và giám sát hoạt động người dùng. Việc triển khai đúng các cơ chế bảo mật không chỉ giúp phòng ngừa rò rỉ dữ liệu, ngăn chặn truy cập trái phép, mà còn góp phần nâng cao uy tín, tuân thủ quy định pháp lý và đảm bảo vận hành an toàn cho toàn bộ hệ thống thông tin.

6.2.1. Mã hóa dữ liệu (Data Encryption)

Mã hóa là một trong những phương pháp quan trọng nhất để bảo vệ dữ liệu nhạy cảm. Cơ chế này giúp biến đổi dữ liệu gốc thành dạng không thể đọc được nếu không có khóa giải mã, từ đó bảo vệ thông tin ngay cả khi dữ liệu bị truy cập trái phép. SQL Server hỗ trợ nhiều kỹ thuật mã hóa khác nhau, gồm:

- Mã hóa đối xứng (Symmetric Encryption): Sử dụng cùng một khóa để mã hóa và giải mã. Ví dụ: AES_256, Triple DES.
- Mã hóa bất đối xứng (Asymmetric Encryption): Sử dụng cặp khóa công khai và khóa riêng (RSA).
- Transparent Data Encryption (TDE): Mã hóa toàn bộ cơ sở dữ liệu ở mức lưu trữ, bảo vệ dữ liệu khi bị sao chép ra ngoài.

Ví dụ minh họa: Tạo một khóa mã hóa đối xứng dùng để bảo vệ dữ liệu bệnh nhân:

```
CREATE SYMMETRIC KEY KhoaBenhNhan  
WITH ALGORITHM = AES_256  
ENCRYPTION BY PASSWORD = 'MatKhau!@#';
```

Sau khi tạo khóa, ta có thể mở khóa và sử dụng để mã hóa hoặc giải mã dữ liệu:

```
-- Mở khóa
OPEN SYMMETRIC KEY KhoaBenhNhan
DECRYPTION BY PASSWORD = 'MatKhau!@#';

-- Mã hóa dữ liệu
UPDATE BENHNHAN
SET CMND = EncryptByKey(Key_GUID('KhoaBenhNhan'), CMND);

-- Giải mã dữ liệu
SELECT MaBN, TenBN,
       CONVERT(NVARCHAR(20), DecryptByKey(CMND)) AS
CMND_GiaiMa
FROM BENHNHAN;

-- Đóng khóa
CLOSE SYMMETRIC KEY KhoaBenhNhan;
```

Cách làm này đảm bảo dữ liệu nhạy cảm (như CMND, mật khẩu, thông tin tài chính) được bảo vệ chặt chẽ và không thể đọc được nếu không có khóa giải mã.

6.2.2. Bảo mật kết nối (Connection Security)

Ngoài việc bảo vệ dữ liệu lưu trữ, cần đảm bảo an toàn trong quá trình truyền tải dữ liệu giữa máy chủ SQL Server và ứng dụng hoặc máy khách. SQL Server hỗ trợ các giao thức bảo mật như SSL (Secure Sockets Layer) và TLS (Transport Layer Security), giúp mã hóa luồng dữ liệu khi di chuyển trên mạng.

- Điều này đặc biệt quan trọng khi hệ thống được triển khai trên môi trường Internet hoặc mạng nội bộ có nhiều điểm truy cập.
- Mọi thông tin gửi đi (bao gồm tên đăng nhập, mật khẩu, dữ liệu truy vấn) đều được mã hóa end-to-end, ngăn chặn hành vi nghe lén (sniffing) hoặc tấn công trung gian (Man-in-the-Middle Attack).

Ví dụ, khi cấu hình kết nối từ ứng dụng tới SQL Server, ta có thể bật tùy chọn:

```
Encrypt = True; TrustServerCertificate = False;
```

Nhờ đó, dữ liệu truyền đi giữa client và server được mã hóa hoàn toàn.

6.2.3. Giới hạn truy cập dữ liệu bằng View

Trong một hệ thống quản lý bệnh viện, dữ liệu của bệnh nhân là thông tin quan trọng và nhạy cảm, bao gồm mã bệnh nhân, địa chỉ, số điện thoại, tiền sử bệnh,... Nếu cấp quyền truy cập trực tiếp vào bảng gốc, nguy cơ rò rỉ hoặc truy cập trái phép dữ liệu là rất cao. Vì vậy, View (khung nhìn) là một công cụ quan trọng giúp đảm bảo an toàn dữ liệu và kiểm soát quyền truy cập ở mức độ chi tiết hơn.

View cho phép tạo ra một "lớp nhìn" ảo từ một hoặc nhiều bảng, trong đó người thiết kế có thể giới hạn cột, giới hạn hàng, hoặc lọc dữ liệu theo điều kiện cụ thể. Nhờ đó, người dùng chỉ nhìn thấy đúng phạm vi dữ liệu mà họ được phép xem, trong khi thông tin nhạy cảm vẫn được bảo vệ trong bảng gốc. Ví dụ:

```
CREATE VIEW VW_ThongTinBenhNhanAnToan AS  
SELECT TenBN, NgaySinh, GioiTinh  
FROM BENHNHAN;
```

View này chỉ hiển thị các thông tin cơ bản như tên bệnh nhân, ngày sinh và giới tính, nhằm phục vụ cho các công việc thống kê, tiếp nhận hoặc hỗ trợ khám bệnh. Những thông tin riêng tư hơn như địa chỉ, số điện thoại, mã bệnh nhân sẽ không được hiển thị. Khi muốn cấp quyền truy cập, thay vì cấp trực tiếp vào bảng BENHNHAN, ta chỉ cần cấp quyền cho View:

```
GRANT SELECT ON VW_ThongTinBenhNhanAnToan TO NhanVienPhongKham;
```

Nhờ cách này, nhân viên phòng khám chỉ có thể xem những thông tin được phép trong View, và không thể truy cập hoặc khai thác dữ liệu đầy đủ từ bảng gốc. Điều này giúp đảm bảo an toàn thông tin, tuân thủ các quy tắc bảo mật dữ liệu và giảm nguy cơ lộ thông tin trong mô hình quản lý bệnh viện.

6.2.4. Phân quyền truy cập và tổ chức dữ liệu theo Schema

Trong SQL Server, Schema được hiểu như một vùng chứa (namespace) dùng để tổ chức và phân nhóm các đối tượng cơ sở dữ liệu như bảng, view, thủ tục và hàm theo từng chức năng. Việc phân loại dữ liệu bằng Schema mang lại nhiều lợi ích trong quản trị hệ thống, đặc biệt là trong các môi trường lớn như hệ thống quản lý bệnh viện.

Khi hệ thống có nhiều bộ phận như bác sĩ, điều dưỡng, kế toán, quản trị hệ thống,... mỗi nhóm người dùng chỉ cần truy cập vào những dữ liệu phù hợp với công việc của họ. Việc

đặt các bảng liên quan vào từng Schema giúp quản trị viên quản lý quyền truy cập dễ dàng hơn, vì có thể gán quyền trực tiếp cho cả một Schema thay vì phải cấp quyền cho từng bảng.

Ví dụ:

```
CREATE SCHEMA YKHOA AUTHORIZATION dbo;  
  
ALTER SCHEMA YKHOA TRANSFER dbo.BENHNHAN;  
ALTER SCHEMA YKHOA TRANSFER dbo.LICKHAM;
```

Ở đây, các bảng BENHNHAN và LICKHAM được chuyển vào schema YKHOA để thể hiện rằng chúng thuộc bộ phận chuyên môn y khoa. Sau khi tổ chức dữ liệu, ta có thể cấp quyền cho nhóm bác sĩ:

```
GRANT SELECT, INSERT ON SCHEMA::YKHOA TO BacSi;  
  
DENY DELETE ON SCHEMA::YKHOA TO BacSi;
```

Nhờ đó:

- Nhóm BacSi có quyền xem dữ liệu (SELECT) và thêm dữ liệu mới (INSERT) trong toàn bộ Schema YKHOA.
- Tuy nhiên, họ không được phép xóa dữ liệu (DELETE), giúp tránh mất mát dữ liệu quan trọng.

Việc phân tách dữ liệu theo Schema đem lại các lợi ích lớn:

- Tách biệt dữ liệu giữa các bộ phận như y khoa, kế toán, nhân sự,... giúp dữ liệu rõ ràng và dễ quản lý hơn.
- Tối ưu hóa phân quyền, tránh phải cấu hình từng bảng riêng lẻ.
- Giảm nguy cơ truy cập nhầm hoặc sai lệch, bảo vệ dữ liệu nhạy cảm trong môi trường bệnh viện.

Các điều trên cho thấy áp dụng Schema là một trong những phương pháp quan trọng giúp tăng cường bảo mật và quản lý dữ liệu hiệu quả trong các hệ thống lớn.

6.2.5. Kết hợp nhiều lớp bảo mật (Defense in Depth)

Trong một hệ thống cơ sở dữ liệu, việc bảo mật không thể chỉ dựa vào một cơ chế duy nhất. Mỗi lớp bảo vệ đều có thể bị vượt qua nếu tồn tại lỗ hổng, vì vậy SQL Server áp dụng mô hình “Defense in Depth” – tức bảo vệ dữ liệu bằng nhiều lớp bảo mật hoạt động đồng thời. Cách tiếp cận này giúp gia tăng mức độ an toàn và giảm thiểu rủi ro ngay cả khi một lớp bị tấn công.

Lớp bảo mật đầu tiên là xác thực (Authentication), nơi hệ thống kiểm tra danh tính người dùng trước khi cho phép đăng nhập. SQL Server hỗ trợ nhiều phương thức xác thực như Windows Authentication – tin cậy theo tài khoản hệ điều hành, hoặc SQL Login – đăng nhập trực tiếp bằng tài khoản SQL. Điều này đảm bảo rằng chỉ những người dùng hợp lệ mới có thể truy cập vào hệ thống.

Sau khi xác thực, hệ thống áp dụng lớp thứ hai: phân quyền (Authorization). Đây là cơ chế xác định người dùng được phép làm gì trên các đối tượng dữ liệu. SQL Server sử dụng các lệnh như GRANT, DENY và REVOKE để thiết lập quyền truy cập, giúp đảm bảo người dùng chỉ thao tác đúng phạm vi công việc của họ và không thể truy cập trái phép vào dữ liệu nhạy cảm.

Tiếp theo là lớp mã hóa (Encryption), đảm nhiệm việc bảo vệ dữ liệu trong quá trình lưu trữ và truyền tải. Các kỹ thuật mã hóa như Transparent Data Encryption (TDE) hoặc mã hóa cột (Column Encryption) giúp ngăn chặn việc đọc trộm dữ liệu, ngay cả khi kẻ tấn công lấy được tệp cơ sở dữ liệu.

Bên cạnh đó, lớp kiểm toán (Auditing) đóng vai trò theo dõi và ghi lại tất cả các hành động liên quan đến truy cập và thao tác dữ liệu. Nhờ hệ thống audit log, quản trị viên có thể phát hiện hành vi bất thường, truy cứu nguồn gốc sự cố và tăng cường khả năng giám sát.

Cuối cùng, SQL Server còn hỗ trợ quản lý kết nối bằng cách giám sát, giới hạn địa chỉ IP, số lượng kết nối hoặc khung thời gian truy cập. Điều này giúp ngăn chặn tấn công từ các nguồn lạ hoặc tránh việc hệ thống bị quá tải bởi những truy cập không mong muốn. Nhờ sự kết hợp của nhiều lớp bảo vệ này, mô hình Defense in Depth mang lại khả năng phòng thủ toàn diện. Dù một lớp bị xâm nhập, những lớp còn lại vẫn tiếp tục hoạt động, giúp bảo vệ dữ liệu và giảm thiểu thiệt hại cho hệ thống.

Bảo mật cơ sở dữ liệu giữ vai trò then chốt trong việc vận hành và phát triển các hệ thống thông tin hiện đại. SQL Server cung cấp một hệ sinh thái bảo mật toàn diện, bao phủ từ mức người dùng, mức đối tượng, đến mức truyền tải và dữ liệu vật lý. Khi triển khai đồng bộ các cơ chế như mã hóa dữ liệu (Encryption), bảo mật kết nối bằng SSL/TLS, phân quyền chi tiết thông qua Authorization, cùng với việc sử dụng View và Schema để che giấu hoặc tách biệt dữ liệu nhạy cảm, hệ thống không chỉ trở nên an toàn hơn mà còn đáp ứng tốt các tiêu chuẩn quốc tế như ISO 27001 hay GDPR. Tóm lại, bảo mật cơ sở dữ liệu không chỉ là trách nhiệm riêng của quản trị viên mà là cam kết chiến lược của toàn bộ tổ chức nhằm bảo vệ tài sản thông tin – yếu tố cốt lõi trong kỷ nguyên số ngày nay.

6.3. Quản lý sao lưu và phục hồi

Trong hệ thống cơ sở dữ liệu, rủi ro mất mát dữ liệu có thể xảy ra bất cứ lúc nào do lỗi phần cứng, tấn công mạng, lỗi phần mềm hoặc thao tác nhầm của người dùng. Vì vậy, sao lưu (backup) và phục hồi (restore) là hai hoạt động quan trọng nhằm đảm bảo dữ liệu luôn an toàn và có thể khôi phục khi xảy ra sự cố. SQL Server cung cấp nhiều cơ chế sao lưu linh hoạt, cho phép tổ chức xây dựng chiến lược bảo vệ dữ liệu phù hợp với mô hình vận hành và mức độ quan trọng của thông tin. SQL Server hỗ trợ nhiều loại sao lưu nhằm đáp ứng các mức độ bảo vệ khác nhau:

a) Full Backup (Sao lưu toàn bộ):

Đây là hình thức sao lưu toàn diện nhất, ghi lại toàn bộ dữ liệu trong cơ sở dữ liệu tại thời điểm sao lưu. Full Backup đóng vai trò nền tảng để thực hiện các loại sao lưu khác.

b) Differential Backup (Sao lưu phần thay đổi):

Ghi lại các dữ liệu đã thay đổi kể từ lần Full Backup gần nhất. Cơ chế này giúp tiết kiệm thời gian và dung lượng, đồng thời vẫn bảo đảm khả năng phục hồi nhanh chóng.

c) Transaction Log Backup (Sao lưu nhật ký giao dịch):

Sao lưu toàn bộ các giao dịch đã được ghi vào Transaction Log. Loại sao lưu này cho phép phục hồi cơ sở dữ liệu đến một thời điểm chính xác (point-in-time recovery), rất quan trọng trong các hệ thống yêu cầu tính nhất quán cao.

d) Lập lịch sao lưu tự động:

SQL Server Agent cho phép tạo tác vụ (Job) để chạy sao lưu theo lịch định sẵn: theo ngày, theo tuần hoặc theo giờ. Việc tự động hóa giúp giảm thiểu sai sót do con người và bảo đảm dữ liệu luôn được lưu trữ kịp thời.

e) Phục hồi dữ liệu (Restore Database):

Khi xảy ra sự cố, SQL Server hỗ trợ phục hồi từ nhiều tệp sao lưu khác nhau, tùy thuộc vào chiến lược đã áp dụng. Quá trình phục hồi có thể bao gồm:

- Phục hồi từ Full Backup
- Phục hồi từ Differential Backup
- Phục hồi thêm Transaction Log để đưa dữ liệu về đúng thời điểm mong muốn

Ví dụ minh họa:

```
BACKUP DATABASE QLBENHVIEN  
TO DISK = 'D:\Backup\QLBENHVIEN_FULL.bak';  
  
RESTORE DATABASE QLBENHVIEN  
FROM DISK = 'D:\Backup\QLBENHVIEN_FULL.bak';
```

f) Chiến lược sao lưu:

Việc thiết kế chiến lược sao lưu hợp lý giúp đảm bảo:

- Tính toàn vẹn của dữ liệu
- Khả năng phục hồi nhanh chóng
- Tối ưu dung lượng lưu trữ
- Đảm bảo hệ thống sẵn sàng hoạt động trở lại trong thời gian ngắn nhất

Quản lý sao lưu và phục hồi là một trong những nhiệm vụ quan trọng của quản trị cơ sở dữ liệu nhằm đảm bảo hệ thống vận hành ổn định và an toàn. SQL Server cung cấp bộ công cụ mạnh mẽ cho phép thực hiện sao lưu toàn phần, sao lưu vi sai và sao lưu nhật ký giao dịch, đồng thời hỗ trợ phục hồi ở nhiều mức độ chi tiết. Việc xây dựng chiến lược sao lưu phù hợp giúp tổ chức giảm thiểu tối đa rủi ro mất dữ liệu và tăng khả năng đảm bảo tính sẵn sàng cho hệ thống trong mọi tình huống.

6.4. Quản lý hiệu suất cơ sở dữ liệu

Hiệu suất của cơ sở dữ liệu giữ vai trò quan trọng trong việc đảm bảo hệ thống vận hành ổn định, đáp ứng nhanh và xử lý khối lượng lớn truy vấn trong thời gian ngắn. Khi cơ sở dữ liệu ngày càng mở rộng, việc tối ưu hóa không chỉ giúp cải thiện tốc độ phản hồi của hệ thống mà còn giảm tải cho phần cứng, tăng tuổi thọ hệ thống và nâng cao trải nghiệm người dùng. Do đó, quản lý hiệu suất là công việc cần thiết và thường xuyên của người quản trị cơ sở dữ liệu. SQL Server cung cấp nhiều công cụ và kỹ thuật hỗ trợ tối ưu hóa hiệu suất, trong đó đặc biệt quan trọng gồm:

- Tối ưu hóa thông qua chỉ mục (Indexing):

Chỉ mục giúp tăng tốc độ truy vấn bằng cách tạo cấu trúc dữ liệu hỗ trợ tìm kiếm nhanh hơn. Việc tạo chỉ mục phù hợp cho các cột thường xuyên được tìm kiếm, lọc hoặc sắp xếp sẽ giảm đáng kể thời gian thực thi truy vấn.

Ví dụ:

```
CREATE INDEX IX_BENHNHAN_TenBN ON BENHNHAN(TenBN);
```

- Phân tích Execution Plan:

Execution Plan cho phép người quản trị đánh giá cách SQL Server thực thi một câu lệnh SQL. Qua đó có thể phát hiện các điểm nghẽn như quét bảng toàn bộ (Table Scan), sử dụng JOIN không tối ưu, hoặc thiếu chỉ mục. Việc phân tích kế hoạch thực thi là bước quan trọng để điều chỉnh và viết lại câu truy vấn hiệu quả hơn.

- Tối ưu hóa câu lệnh SQL:

Một số kỹ thuật phổ biến bao gồm:

- Giảm truy vấn con lồng nhau (nested subqueries)
- Tránh sử dụng SELECT *
- Sử dụng JOIN hợp lý
- Hạn chế phép tính trên cột trong mệnh đề WHERE

Những tối ưu này giúp SQL Server dễ dàng lựa chọn kế hoạch thực thi tối ưu, đồng thời giảm tài nguyên xử lý.

d) Tái tổ chức và tái tạo chỉ mục (Index Reorganize / Rebuild):

Sau thời gian hoạt động, chỉ mục có thể bị phân mảnh, làm giảm hiệu suất truy vấn. SQL Server hỗ trợ:

- REORGANIZE: Sắp xếp lại các trang dữ liệu, giảm mức phân mảnh nhẹ.
- REBUILD: Tạo lại toàn bộ chỉ mục, áp dụng khi phân mảnh lớn.

Ví dụ:

```
ALTER INDEX IX_BENHNHAN_TenBN ON BENHNHAN REORGANIZE;
```

e) Giám sát tài nguyên hệ thống:

Việc theo dõi CPU, RAM, I/O, dung lượng đĩa và tốc độ đọc/ghi giúp quản trị viên phát hiện kịp thời các vấn đề về tài nguyên ảnh hưởng đến hiệu suất cơ sở dữ liệu. SQL Server cung cấp các công cụ như Activity Monitor, SQL Server Profiler, và Dynamic Management Views (DMVs) để hỗ trợ giám sát.

Quản lý hiệu suất cơ sở dữ liệu là một quá trình liên tục và đòi hỏi sự kết hợp giữa nhiều kỹ thuật khác nhau. Từ việc tạo và bảo trì chỉ mục, phân tích kế hoạch thực thi, tối ưu câu lệnh SQL, đến giám sát tài nguyên hệ thống—tất cả đều nhằm đảm bảo cơ sở dữ liệu hoạt động ổn định, nhanh chóng và hiệu quả. Một chiến lược tối ưu hóa tốt sẽ giúp giảm chi phí phần cứng, tránh quá tải và nâng cao hiệu suất tổng thể của hệ thống.

CHƯƠNG 7: KẾT LUẬN VÀ KHUYẾN NGHỊ

7.1. Tóm tắt liên quan

Trong suốt quá trình thực hiện đề tài xây dựng hệ thống cơ sở dữ liệu quản lý bệnh viện, nhóm đã hoàn thành toàn bộ yêu cầu đề ra từ khâu phân tích, thiết kế đến cài đặt và thử nghiệm. Trước hết, nhóm đã xây dựng được mô hình dữ liệu đầy đủ gồm các bảng: BENHNHAN, BACSI, KHOA, HOADON, LICHKHAM, cùng với các ràng buộc quan trọng như khóa chính, khóa ngoại, CHECK, DEFAULT và quy định tính toàn vẹn dữ liệu.

Tiếp theo, hệ thống đã được mở rộng bằng các thành phần nâng cao như chỉ mục, view, stored procedures, functions và trigger, giúp tăng hiệu suất truy vấn, tự động hóa các thao tác, và tăng cường an toàn dữ liệu. Các kỹ thuật bảo mật như phân quyền, tạo LOGIN/USER, mã hóa dữ liệu, tổ chức schema và sao lưu – phục hồi cũng đã được áp dụng hiệu quả nhằm đảm bảo tính an toàn và ổn định của hệ thống.

Cuối cùng, nhóm đã xây dựng được một báo cáo hoàn chỉnh, có cấu trúc rõ ràng, phản ánh đầy đủ quy trình triển khai hệ thống cơ sở dữ liệu theo đúng chuẩn học thuật. Các kết quả đạt được không chỉ thỏa mãn yêu cầu môn học mà còn góp phần cung cấp kiến thức về thiết kế và quản trị cơ sở dữ liệu trong thực tiễn.

7.2. Khuyến nghị

Mặc dù hệ thống đã hoạt động ổn định và đáp ứng các chức năng cơ bản của một hệ thống quản lý bệnh viện, vẫn còn nhiều hướng phát triển và cải tiến trong tương lai. Thứ nhất, hệ thống có thể được mở rộng để tích hợp giao diện ứng dụng thực tế (web hoặc desktop) nhằm hỗ trợ người dùng cuối truy cập và thao tác trực quan hơn. Việc kết nối CSDL với ứng dụng giúp quản lý dữ liệu diễn ra tự động, giảm thiểu sai sót và nâng cao trải nghiệm người dùng.

Thứ hai, cần nghiên cứu thêm các cơ chế bảo mật nâng cao như mã hóa cột (Column Encryption), bảo mật cấp dòng (Row-Level Security) hoặc sử dụng Always Encrypted để nâng cao mức độ an toàn cho dữ liệu nhạy cảm. Ngoài ra, việc triển khai các công cụ giám sát hiệu suất (Performance Monitoring) và tự động tối ưu chỉ mục theo thời gian thực cũng giúp hệ thống hoạt động hiệu quả hơn.

Cuối cùng, nhóm đề xuất mở rộng hệ thống theo hướng tích hợp thêm các phân hệ lớn như quản lý thuốc, quản lý ca trực bác sĩ, quản lý xét nghiệm hoặc lưu trữ hồ sơ bệnh án điện tử (EMR). Đây là các hướng nghiên cứu quan trọng và có tính ứng dụng cao, giúp hệ thống trở thành một giải pháp quản lý tổng thể cho bệnh viện trong bối cảnh chuyển đổi số.

KẾT LUẬN

Dự án xây dựng cơ sở dữ liệu quản lý bệnh viện đã đạt được nhiều ưu điểm nổi bật, thể hiện qua việc thiết kế mô hình dữ liệu chặt chẽ, đảm bảo tính toàn vẹn và đáp ứng đúng yêu cầu của hệ thống thực tế. Các bảng dữ liệu được xây dựng với đầy đủ khóa chính, khóa ngoại, các ràng buộc CHECK và DEFAULT, giúp hạn chế sai sót trong quá trình nhập liệu. Đồng thời, việc triển khai các đối tượng nâng cao như chỉ mục (Index), view, stored procedures, functions và trigger đã giúp tối ưu hiệu suất, tăng tính tự động và nâng cao mức độ an toàn của hệ thống. Các kỹ thuật bảo mật và phân quyền người dùng cũng góp phần đảm bảo dữ liệu được bảo vệ toàn diện.

Bên cạnh các ưu điểm đạt được, hệ thống vẫn tồn tại một số hạn chế nhất định. Thứ nhất, cơ sở dữ liệu mới chỉ mô phỏng các nghiệp vụ cơ bản của bệnh viện, chưa bao quát đầy đủ toàn bộ quy trình thực tế như quản lý thuốc, xét nghiệm, hồ sơ bệnh án điện tử hoặc lịch trực của bác sĩ. Thứ hai, hệ thống chưa được tích hợp với giao diện ứng dụng thực tế, nên việc thao tác vẫn cần thực hiện trực tiếp bằng câu lệnh SQL, gây khó khăn cho người dùng không chuyên. Ngoài ra, hệ thống cũng chưa áp dụng các giải pháp bảo mật mức cao như mã hóa cột, phân quyền theo hàng dữ liệu hoặc quản lý truy cập động theo vai trò.

Trong tương lai, hệ thống có thể được phát triển theo nhiều hướng nhằm nâng cao tính ứng dụng và mức độ hoàn thiện. Việc xây dựng giao diện phần mềm (web hoặc desktop) kết nối trực tiếp với cơ sở dữ liệu sẽ giúp hệ thống trở nên thân thiện và dễ sử dụng hơn. Bên cạnh đó, có thể mở rộng cơ sở dữ liệu để tích hợp thêm các phân hệ quản lý như thuốc, vật tư y tế, hồ sơ bệnh án điện tử (EMR), lịch trực bác sĩ hoặc quản lý phòng bệnh. Cuối cùng, nghiên cứu và áp dụng các công nghệ bảo mật nâng cao như Always Encrypted, Row-Level Security hay Transparent Data Encryption (TDE) sẽ giúp hệ thống an toàn, hiện đại và đáp ứng yêu cầu thực tế trong môi trường bệnh viện số.

DANH MỤC TÀI LIỆU THAM KHẢO

[1]. Nguyễn Thị Tính (2024), *Giáo trình Hệ Quản Trị Cơ Sở Dữ Liệu SQL Server*, NXB Xây Dựng.

[2]. Nguyễn Thái Nghe (2014), *Giáo trình Hệ quản trị cơ sở dữ liệu*, Tài liệu PDF trực tuyến, Slideshare.

Truy cập tại: www.slideshare.net

[3]. Trường Cao đẳng Công nghệ Thông tin – Truyền thông (2022), *Giáo trình Quản Trị Cơ Sở Dữ Liệu với SQL Server*, Studocu.

Truy cập tại: [Giáo trình Quản trị Cơ sở Dữ liệu với SQL Server - Nghề CNTT Trung cấp - Studocu](https://studocu.com/doc/giao-trinh-quan-tri-co-so-du-lieu-voi-sql-server-nghiep-cntt-trung-cap-studocu)

[4]. Đại học Huế (2014), *Giáo trình Hệ Quản Trị Cơ Sở Dữ Liệu*, Tài liệu học PDF, Hue University Repository.

Truy cập tại: [Microsoft Word - SachCSDL_bannop.tif](https://repository.hue.ac.vn/handle/123456789/10000)

[5]. *Tự học MS SQL Server* (n.d.), Thư Viện Lập Trình – Tài liệu tiếng Việt.

[6]. Louis Davidson & Jessica Moss (2021), *Pro SQL Server Relational Database Design and Implementation: Database Design Delivering Business Results*, Apress.

[7]. Markus Winand (2018), *SQL Indexing and Tuning e-Book for Developers* (Tài liệu trực tuyến).

Truy cập tại: use-the-index-luke.com

[8]. J. Strate (2012), *Expert Performance Indexing for SQL Server 2012*, Apress.

Truy cập tại: https://link.springer.com/book/10.1007/978-1-4302-3742-6?utm_source=.com