

TRẦN THIÊN THÀNH (chủ biên)  
NGUYỄN THỊ TUYẾT, NGUYỄN THỊ KIM PHƯỢNG, PHAN ĐÌNH SINH

**GIÁO TRÌNH**  
**HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

**Quy Nhơn, 2019**

## MỤC LỤC

LỜI NÓI ĐẦU .....	1
CHƯƠNG 1. KHÁI QUÁT VỀ CƠ SỞ DỮ LIỆU VÀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU .....	9
1.1. Khái niệm cơ sở dữ liệu .....	9
1.1.1. Cơ sở dữ liệu là gì? .....	9
1.1.2. Sự cần thiết của các hệ cơ sở dữ liệu .....	10
1.1.3. Mục tiêu của các hệ cơ sở dữ liệu .....	11
1.2. Một số mô hình cơ sở dữ liệu.....	12
1.2.1. Mô hình ngoài .....	13
1.2.2. Mô hình dữ liệu .....	13
1.2.3. Mô hình trong.....	15
1.3. Hệ quản trị cơ sở dữ liệu .....	16
CÂU HỎI ÔN TẬP CHƯƠNG 1 .....	18
CHƯƠNG 2. TỔNG QUAN VỀ SQL SERVER .....	19
2.1. Giới thiệu hệ quản trị cơ sở dữ liệu quan hệ (Relational DataBase Management System- RDBMS) và SQL server .....	19
2.1.1. Hệ quản trị cơ sở dữ liệu quan hệ là gì?.. <b>Error! Bookmark not defined.</b>	
2.1.2. SQL Server .....	19
2.1.2.1. SQL là gì? .....	19
2.1.2.2. SQL Server là gì? .....	20
2.1.3. Vai trò của SQL .....	21
2.2. Các thành phần của SQL server.....	22
2.1. Mô hình chung SQL Server trên mạng. ....	23
2.2. Mô hình Desktop.....	24
2.3. Mô hình Client/Server.....	25
2.4. Mô hình kết nối ứng dụng trên mạng Internet .....	26
2.5. SQL Server Management Studio .....	27
2.6. Tạo cơ sở dữ liệu bằng giao diện của SQL Server Management Studio .....	28
2.7. Cấu trúc cơ sở dữ liệu. ....	29
CÂU HỎI ÔN TẬP CHƯƠNG 2 .....	30
CHƯƠNG 3. BẢNG DỮ LIỆU – TABLE .....	32
3.1. Khái niệm bảng dữ liệu .....	32
3.2. Kiểu dữ liệu .....	33
3.3. Ràng buộc dữ liệu .....	36
3.3.1. Khóa chính (Primary Key).....	38
3.3.2. Khóa ngoài (Foreign Key). ....	38
3.3.3. Ràng buộc Unique.....	39
3.3.4. Ràng buộc Check. ....	39
3.3.5. Ràng buộc giá trị mặc định (Default).....	40
3.4. Tạo bảng dữ liệu.....	40
3.4.1. Tạo ràng buộc PRIMARY KEY .....	42
3.4.2. Tạo ràng buộc FOREIGN KEY .....	43
3.4.3. Tạo ràng buộc CHECK .....	44
3.4.4. Tạo ràng buộc UNIQUE .....	45

3.5. Sửa, xóa cấu trúc bảng .....	46
3.5.1. Sửa cấu trúc bảng .....	46
3.5.2. Xóa bảng .....	48
3.6. Nhập dữ liệu vào bảng .....	49
3.7. Cập nhật dữ liệu .....	50
3.8. Xóa dữ liệu .....	51
3.9. Tạo chỉ mục (Index) .....	53
3.9.1. Chỉ mục là gì? .....	53
3.9.2. Các loại chỉ mục .....	54
3.9.3. Tạo chỉ mục .....	54
3.9.4. Xóa chỉ mục: .....	55
CHƯƠNG 4. TRUY VẤN DỮ LIỆU – QUERY .....	56
4.1. Câu lệnh truy vấn dạng tổng quát .....	56
4.1.1. Câu lệnh SELECT với mệnh đề FROM .....	57
4.1.2. Câu lệnh SELECT với mệnh đề WHERE .....	58
4.1.3. Câu lệnh SELECT với mệnh đề ORDER BY .....	60
4.1.4. Câu lệnh SELECT với mệnh đề GROUP BY .....	61
4.1.5. Câu lệnh SELECT với mệnh đề INTO .....	61
4.1.6. Câu lệnh Select với mệnh đề Having .....	62
4.1.7. Câu lệnh Select với mệnh đề Compute .....	62
4.2. Truy vấn dữ liệu trên nhiều bảng .....	64
4.3. Các loại phép kết nối trong SQL Server: .....	66
4.3.1. Phép kết nối trong (Inner Join) .....	66
4.3.2. Phép kết nối ngoài (Outer Join) .....	68
4.3.3. Phép kết nối tích hợp (CROSS JOIN) .....	71
4.3.4. Phép tự kết kết nối (Self Join) .....	71
4.4. Truy vấn con .....	66
4.5. Tối ưu hóa truy vấn .....	73
4.6. Hợp các câu lệnh truy vấn .....	75
4.6.1. Cú pháp câu lệnh hợp .....	75
4.6.2. Các nguyên tắc khi sử dụng UNION .....	76
CHƯƠNG 5. KHUNG NHÌN - VIEW .....	78
5.1. Khái niệm .....	78
5.2. Tạo khung nhìn .....	80
5.2.1. Tạo khung nhìn bằng câu lệnh .....	80
5.2.2. Tạo khung nhìn bằng chức năng có sẵn của Microsoft SQL Server .....	80
5.2.3. Thực thi khung nhìn .....	81
5.3. Một số quy định khi sử dụng khung nhìn .....	81
5.4. Cập nhật, bổ sung và xóa dữ liệu thông qua khung nhìn .....	82
5.5. Sửa đổi khung nhìn .....	82
5.6. Xóa khung nhìn .....	83
CHƯƠNG 6. THỦ TỤC THƯỜNG TRÚ – STORED PROCEDURE .....	84
6.1. Định nghĩa .....	84
6.2. Ưu điểm khi quản lý dữ liệu bằng thủ tục lưu trữ .....	85
6.3. Phân loại thủ tục lưu trữ .....	86
6.4. Tạo thủ tục lưu trữ .....	87

6.5. Lệnh gọi thủ tục lưu trữ .....	88
6.6. Sửa/xóa thủ tục lưu trữ .....	89
6.7. Sử dụng biến trong thủ tục .....	89
6.8. Các phát biểu điều khiển .....	92
6.9. Sửa, xóa thủ tục .....	94
CÂU HỎI ÔN TẬP CHƯƠNG 6 .....	94
CHƯƠNG 7. HÀM DO NGƯỜI DÙNG ĐỊNH NGHĨA – USER DEFINED FUNCTION.....	95
7.1. Khái niệm .....	95
7.2. Những hạn chế khi sử dụng hàm do người dùng định nghĩa .....	95
7.3. Phân loại .....	95
7.3.1. Hàm vô hướng .....	96
7.3.2. Hàm nội tuyến .....	97
7.3.3. Hàm bao gồm nhiều câu lệnh bên trong .....	98
CÂU HỎI ÔN TẬP CHƯƠNG 7 .....	99
CHƯƠNG 8. TRIGGER .....	100
8.1. Giao tác .....	100
8.1.1. Giới thiệu.....	100
8.1.2. Khái niệm .....	100
8.2. Trigger .....	101
8.2.1. Khái niệm .....	101
8.2.2. Khi nào dùng trigger? .....	103
8.2.3. Tạo trigger .....	104
8.2.4 Các kiểu Trigger.....	105
8.2.5 Sửa đổi Trigger .....	106
8.2.6 Xóa Trigger .....	107
CÂU HỎI ÔN TẬP CHƯƠNG 8 .....	107
CHƯƠNG 9. BẢO MẬT VÀ PHÂN QUYỀN NGƯỜI DÙNG.....	108
9.1. Khái niệm .....	108
9.2. Quản lý người dùng.....	109
9.3. Các cơ chế bảo mật .....	109
9.4. Các mức bảo mật.....	110
9.5. Tạo người dùng .....	112
9.5.1. Tạo Login .....	112
9.5.2. Tạo USER .....	114
9.5.3. Cấp phát quyền cho người dùng .....	115
CÂU HỎI ÔN TẬP CHƯƠNG 9 .....	115
CHƯƠNG 10. IMPORT/EXPORT DATA VÀ BACK UP/RESTORE DATA ....	116
10.1. Back up Database .....	116
10.1.1. Các kiểu Back up .....	116
10.1.2. Back up Database.....	118
10.2. Restore Database .....	119
10.3. IMPORT DATA.....	120
10.4. EXPORT DATA .....	122
CÂU HỎI ÔN TẬP CHƯƠNG 10 .....	124
BÀI THỰC HÀNH .....	125

TÀI LIỆU THAM KHẢO.....	130
PHỤ LỤC.....	<b>ERROR! BOOKMARK NOT DEFINED.</b>



## LỜI NÓI ĐẦU

Hệ quản trị cơ sở dữ liệu là môn học bắt buộc trong chương trình đào tạo ngành Công nghệ thông tin và một số ngành khác có lưu trữ và xử lý dữ liệu.

Microsoft SQL Server là một hệ quản trị cơ sở dữ liệu mô hình quan hệ (Relational Database Management System – RDBMS) được phát triển bởi Microsoft dùng để quản lý và lưu trữ dữ liệu theo mô hình Client/Server. Đây là một trong những hệ quản trị cơ sở dữ liệu có đầy đủ các tính năng của một hệ quản trị cơ sở dữ liệu cơ bản, được nhiều nhà phát triển ứng dụng lựa chọn như một giải pháp lưu trữ dữ liệu cho các ứng dụng.

Giáo trình Hệ quản trị cơ sở dữ liệu là những nội dung của các giảng viên Khoa Công nghệ thông tin – Trường Đại học Quy Nhơn, giảng dạy trong nhiều năm qua. Trong giáo trình này, chúng tôi không có tham vọng đề cập đến mọi khía cạnh của SQL Server mà chỉ đặt mục tiêu là tài liệu cho sinh viên và những người tự học nắm được các khái niệm và ý nghĩa của hệ quản trị cơ sở dữ liệu, làm quen và thực hành trên một hệ quản trị cơ sở dữ liệu cụ thể là Microsoft SQL Server.

Giáo trình được chia thành 10 chương với nội dung như sau:

Chương 1: Khái quát về cơ sở dữ liệu và hệ quản trị cơ sở dữ liệu

Chương 2: Tổng quan về SQL Server

Chương 3: Bảng dữ liệu - Table

Chương 4: Truy vấn dữ liệu - Query

Chương 5: Khung nhìn - View

Chương 6: Thủ tục thường trú - Stored Procedure

Chương 7: Hàm do người dùng định nghĩa - User Defined Function

Chương 8: Trigger

Chương 9: Quản lý người dùng và bảo mật trong SQL Server

Chương 10: Backup và Restore Data; Import và Export Data

Mặc dù đã có nhiều cố gắng trong việc biên soạn và cập nhật cho phù hợp với thực tế nhưng giáo trình chắc chắn vẫn còn những hạn chế và thiếu sót. Nhóm biên soạn mong nhận được những ý kiến đóng góp từ đồng nghiệp và độc giả để giáo trình ngày được hoàn thiện hơn với mục tiêu đáp ứng tốt nhất cho người học.

Mọi đóng góp xin gửi về: Khoa CNTT, Trường Đại học Quy Nhơn, 170 An Dương Vương, Tp. Quy Nhơn, Tỉnh Bình Định.

**NHÓM BIÊN SOẠN**



# **CHƯƠNG 1. KHÁI QUÁT VỀ CƠ SỞ DỮ LIỆU VÀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

Trong chương này trình bày những khái niệm cơ bản về các hệ cơ sở dữ liệu theo mô hình quan hệ do E.F Codd đề xuất. Những khái niệm này bao gồm mục tiêu của một hệ cơ sở dữ liệu; Sự cần thiết phải tổ chức dữ liệu dưới dạng hệ cơ sở dữ liệu; Tính độc lập của dữ liệu thể hiện mô hình hình kiến trúc 3 mức. Qua đó có thể thấy cơ sở dữ liệu phản ánh tính trung thực, khách quan của thế giới dữ liệu; không dư thừa thông tin và cũng không thiếu thông tin.

Nội dung của chương bao gồm các phần:

- Cơ sở dữ liệu là gì?
- Sự cần thiết của các hệ cơ sở dữ liệu.
- Mô hình kiến trúc 3 mức cơ sở dữ liệu.
- Mục tiêu của các hệ cơ sở dữ liệu.
- Hệ quản trị cơ sở dữ liệu.

## **1.1. Khái niệm cơ sở dữ liệu**

### ***1.1.1. Cơ sở dữ liệu là gì?***

Cơ sở dữ liệu (CSDL) là một bộ sưu tập về các loại dữ liệu tác nghiệp, bao gồm các loại dữ liệu âm thanh, tiếng nói, chữ viết, văn bản, đồ họa, hình ảnh tĩnh hay hình ảnh động....được mã hoá dưới dạng các chuỗi nhị phân và được lưu trữ dưới dạng tệp tin dữ liệu trong các bộ nhớ của máy tính. Cấu trúc lưu trữ dữ liệu tuân theo các quy tắc dựa trên lý thuyết toán học. Cơ sở dữ liệu phản ánh trung thực thế giới hiện thực khách quan.

Cơ sở dữ liệu là tài nguyên thông tin chung cho nhiều người cùng sử dụng. Bất kỳ người sử dụng nào trên mạng máy tính, tại các thiết bị đầu cuối, về nguyên tắc có quyền truy nhập khai thác toàn bộ hay một phần dữ liệu theo chế độ trực tuyến hay tương tác mà không phụ thuộc vào vị trí địa lý của người sử dụng với các tài nguyên đó.

Cơ sở dữ liệu được các hệ ứng dụng khai thác bằng ngôn ngữ truy xuất dữ liệu hoặc bằng các chương trình ứng dụng để xử lý, tìm kiếm, tra cứu, sửa đổi, bổ sung hay loại bỏ dữ liệu. Tìm kiếm và tra cứu thông tin là một trong những chức năng quan trọng và phổ biến nhất của các ứng dụng khai thác cơ sở dữ liệu.

### ***1.1.2. Sự cần thiết của các hệ cơ sở dữ liệu***

Cơ sở dữ liệu được tổ chức có những ưu điểm:

*Giảm bớt dư thừa dữ liệu trong lưu trữ:* Trong các ứng dụng lập trình truyền thống, phương pháp tổ chức lưu trữ dữ liệu vừa tốn kém; lãng phí bộ nhớ và các thiết bị lưu trữ; vừa dư thừa thông tin lưu trữ. Nhiều chương trình ứng dụng khác nhau cùng xử lý trên các dữ liệu như nhau, dẫn đến sự dư thừa đáng kể về dữ liệu. Nếu tổ chức lưu trữ theo lý thuyết CSDL thì có thể hợp nhất các tệp lưu trữ dữ liệu để các chương trình ứng dụng có thể cùng chia sẻ tài nguyên trên cùng một hệ cơ sở dữ liệu.

*Tránh được sự không nhất quán trong lưu trữ dữ liệu và bảo đảm được tính toàn vẹn của dữ liệu:* Nếu một thuộc tính được mô tả trong nhiều tệp dữ liệu khác nhau và lặp lại nhiều lần trong các bản ghi, khi thực hiện việc cập nhật, sửa đổi, bổ sung sẽ không sửa hết nội dung các mục đó. Nếu dữ liệu càng nhiều thì sự sai sót khi cập nhật, bổ sung càng lớn. Khả năng xuất hiện mâu thuẫn, không nhất quán thông tin càng nhiều, dẫn đến không nhất quán dữ liệu trong lưu trữ. Tất yếu kéo theo sự dị thường thông tin, thừa, thiếu và mâu thuẫn thông tin.

Thông thường, trong một thực thể, giữa các thuộc tính có mối quan hệ ràng buộc lẫn nhau, tác động ảnh hưởng lẫn nhau. Như vậy, có thể khẳng định, nếu dữ liệu không tổ chức tốt tất yếu không thể phản ánh đúng thế giới hiện thực dữ liệu, không phản ánh đúng bản chất vận động của dữ liệu. Sự không nhất quán dữ liệu trong lưu trữ làm cho dữ liệu mất đi tính toàn vẹn của nó. Tính toàn vẹn dữ liệu đảm bảo cho sự lưu trữ dữ liệu luôn luôn đúng.

*Có thể triển khai đồng thời nhiều ứng dụng trên cùng một CSDL:* Điều này có nghĩa là các ứng dụng không chỉ chia sẻ chung tài nguyên dữ liệu mà còn trên cùng một CSDL có thể triển khai đồng thời nhiều ứng dụng khác nhau tại các thiết bị đầu cuối khác nhau. Tổ chức dữ liệu theo lý thuyết cơ sở dữ liệu sẽ thống nhất các tiêu chuẩn, thủ tục và các biện pháp bảo vệ, an toàn dữ liệu. Các hệ CSDL sẽ được quản lý tập trung bởi một người hay một nhóm người quản trị CSDL. Bằng các hệ quản trị CSDL, người quản trị CSDL có thể áp dụng thống nhất các tiêu chuẩn, quy định, thủ tục chung như quy định thống nhất về mẫu biểu báo cáo, thời gian bổ sung, cập nhật dữ liệu. Điều này làm dễ dàng cho công việc bảo trì dữ liệu. Người quản trị CSDL có thể bảo đảm việc truy nhập tới CSDL, có thể kiểm tra, kiểm soát các quyền truy nhập của người sử dụng. Ngăn chặn các truy nhập trái phép, sai quy định từ trong ra hoặc từ ngoài vào.

### ***1.1.3. Mục tiêu của các hệ cơ sở dữ liệu***

Người sử dụng khi thao tác trên các cơ sở dữ liệu không được làm thay đổi cấu trúc lưu trữ dữ liệu và chiến lược truy nhập tới các hệ cơ sở dữ liệu. Dữ liệu chỉ được biểu diễn, mô tả một cách duy nhất. Cấu trúc lưu trữ dữ liệu và các chương trình ứng dụng trên các hệ CSDL hoàn toàn độc lập với nhau, không phụ thuộc lẫn nhau. Vì vậy bảo đảm tính độc lập dữ liệu là mục tiêu quan trọng của các hệ cơ sở dữ liệu. Có thể định nghĩa tính độc lập dữ liệu là “Tính bất biến của các ứng dụng đối với sự thay đổi trong cấu trúc lưu trữ và chiến lược truy nhập dữ liệu”.

Khi thay đổi cấu trúc lưu trữ và các chiến lược truy nhập dữ liệu không kéo theo thay đổi nội dung của các chương trình ứng dụng và ngược lại, khi các chương trình thay đổi cũng không làm ảnh hưởng đến cấu trúc lưu trữ và chiến lược truy nhập của dữ liệu. Tính độc lập của dữ liệu bảo đảm cho việc biểu diễn nội dung thông tin cho các thực thể là duy nhất và bảo đảm tính toàn vẹn và nhất quán dữ liệu trong lưu trữ.

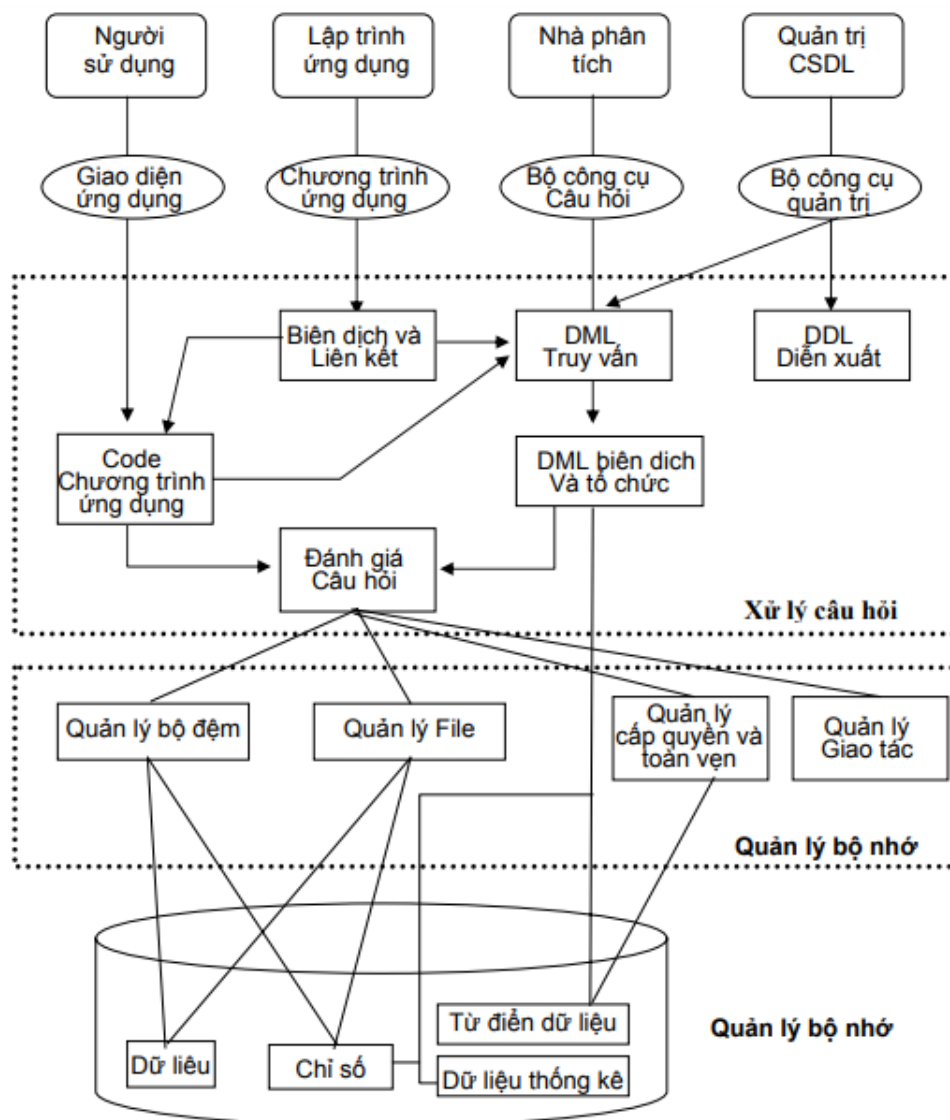
Trong các mô hình dữ liệu, mô hình quan hệ là mô hình có cơ sở lý thuyết tốt nên được các nhà phát triển hệ quản trị cơ sở dữ liệu lựa chọn. Mô hình quan hệ mô tả thế giới hiện thực một cách chính xác, khách quan, phù hợp với cách nhìn và sử dụng của người dùng. Vì vậy tính độc lập dữ liệu trong các hệ cơ sở dữ liệu quan hệ cao.

Trong kiến trúc hệ cơ sở dữ liệu (hình 1.1) tính độc lập dữ liệu được thể hiện:

Có rất nhiều cách nhìn dữ liệu ở mô hình ngoài, người sử dụng khác nhau có cách nhìn dữ liệu khác nhau và các ứng dụng khác nhau có những cách nhìn dữ liệu cũng khác nhau, nhưng chỉ có duy nhất một cách nhìn dữ liệu ở mức quan niệm, biểu diễn toàn bộ thông tin trong CSDL đó là cách nhìn dữ liệu tổng quát của người sử dụng. Và cũng chỉ có duy nhất một và chỉ một cách biểu diễn CSDL dưới dạng lưu trữ vật lý.

Ánh xạ trong xác định giữa mô hình trong và mô hình dữ liệu, nhằm bảo đảm được tính độc lập của dữ liệu, nghĩa là nếu cấu trúc lưu trữ của CSDL thay đổi, tức là thay đổi định nghĩa về cấu trúc lưu trữ dữ liệu thì ánh xạ này phải cũng phải thay đổi tương ứng sao cho sơ đồ quan niệm (mô hình dữ liệu) không được thay đổi. Tương tự ánh xạ ngoài xác định tương ứng giữa một mô hình của người sử dụng nào đó với mô hình dữ liệu. Nó chuyển đổi dạng biểu diễn dữ liệu lưu trữ sang dạng biểu diễn dữ liệu mà các ứng dụng cần đến.

Các ứng dụng khác nhau có nhiều khung nhìn khác nhau với dữ liệu như nhau. Các ứng dụng độc lập với cấu trúc lưu trữ và chiến lược truy nhập. Giữa chúng không có sự ràng buộc lẫn với nhau. Điều này có nghĩa là các ứng dụng hoàn toàn độc lập với bất cứ một cấu trúc lưu trữ và chiến lược truy nhập dữ liệu cụ thể nào. Ngược lại cấu trúc lưu trữ và chiến lược truy nhập dữ liệu không phụ thuộc vào bất kỳ ứng dụng cụ thể nào. Người quản trị CSDL phải có khả năng đáp ứng với mọi sự thay đổi về cấu trúc lưu trữ và các chiến lược truy nhập mà không cần biết tới có những ứng dụng nào trên CSDL.



**Hình 1.1. Sơ đồ kiến trúc hệ thống cơ sở dữ liệu**

## 1.2. Một số mô hình cơ sở dữ liệu

Mô hình kiến trúc 3 mức của hệ CSDL gồm: Mức trong, mức mô hình dữ liệu (mức quan niệm) và mức ngoài. Giữa các mức tồn tại các ánh xạ quan niệm trong và ánh xạ quan niệm ngoài. Trung tâm của hệ thống là mức quan niệm, tức là mức

mô hình dữ liệu. Ngoài ra còn có khái niệm người sử dụng, hệ quản trị CSDL và người quản trị CSDL. Người sử dụng là những người tại thiết bị đầu cuối truy nhập vào các hệ CSDL theo chế độ trực tuyến hay tương tác bằng các chương trình ứng dụng hay bằng các ngôn ngữ truy xuất dữ liệu.

### ***1.2.1. Mô hình ngoài***

Người sử dụng có thể truy nhập toàn bộ hay một phần CSDL mà họ quan tâm, phụ thuộc vào quyền truy nhập của họ. Cách nhìn CSDL của người sử dụng nói chung là trừu tượng. Họ nhìn CSDL bằng mô hình ngoài, gọi là mô hình con dữ liệu. Chẳng hạn người sử dụng là một nhân viên của phòng kế toán tài chính, chỉ nhìn thấy tập các xuất hiện kiểu bản ghi ngoài về doanh thu, sản lượng trong tháng, không thể nhìn thấy các xuất hiện kiểu bản ghi lưu trữ về các chỉ tiêu kỹ thuật của hệ thống.

Mô hình ngoài là nội dung thông tin của CSDL dưới cách nhìn của người sử dụng. Nó là nội dung thông tin của một phần dữ liệu tác nghiệp được một người hoặc một nhóm người sử dụng. Nói cách khác, mô hình ngoài mô tả cách nhìn dữ liệu của người sử dụng và mỗi người sử dụng có cách nhìn dữ liệu khác nhau. Nhiều mô hình ngoài khác nhau có thể cùng tồn tại trong một hệ CSDL, nghĩa là có nhiều người sử dụng chia sẻ chung cùng một cơ sở dữ liệu. Hơn nữa, có thể mô hình ngoài quan hệ, mô hình ngoài phân cấp hay mô hình ngoài kiểu mạng cũng có thể tồn tại trong một cơ sở dữ liệu.

Mô hình ngoài gồm nhiều xuất hiện kiểu bản ghi ngoài, nghĩa là mỗi một người sử dụng có một sơ đồ dữ liệu riêng, một khung nhìn dữ liệu riêng. Bản ghi ngoài của người sử dụng có thể khác với bản ghi lưu trữ và bản ghi quan niệm. Mô hình ngoài được xác định bởi một sơ đồ ngoài bao gồm các mô tả về kiểu bản ghi ngoài như tên các trường, kiểu dữ liệu các trường, độ rộng của trường....

- Ngôn ngữ truy xuất dữ liệu của người sử dụng thao tác trên các bản ghi ngoài.
- Người sử dụng khác nhau có khung nhìn dữ liệu khác nhau.
- Người sử dụng đầu cuối có thể là các ứng dụng hay thao tác trực tiếp bằng ngôn ngữ thao tác, truy vấn dữ liệu.

### ***1.2.2. Mô hình dữ liệu***

Mô hình dữ liệu (mô hình quan niệm) là cách nhìn dữ liệu một cách tổng quát của người sử dụng. Nghĩa là có rất nhiều cách nhìn dữ liệu ở mô hình ngoài, nhưng

chỉ có duy nhất một cách nhìn dữ liệu ở mức quan niệm. Biểu diễn toàn bộ thông tin trong CSDL là duy nhất.

Mô hình dữ liệu gồm nhiều xuất hiện của nhiều kiểu bản ghi dữ liệu. Ví dụ kiểu xuất hiện bản ghi về nhân sự, kiểu xuất hiện bản ghi về doanh thu, sản lượng, kiểu xuất hiện bản ghi về cước đàm thoại...

Mô hình dữ liệu được xác định bởi một sơ đồ dữ liệu mô tả của nhiều kiểu thực thể, chẳng hạn như mô tả thực thể tuyến cáp, các loại cáp, thầy giáo, học sinh... Sơ đồ dữ liệu bao gồm các định nghĩa về các kiểu bản ghi, đó là các ràng buộc cho quyền và tính toàn vẹn thích hợp. Những ràng buộc này chính là các tính chất của dữ liệu, tính liên kết các thuộc tính cùng một kiểu dữ liệu. Các định nghĩa này không bao hàm về cấu trúc lưu trữ, cũng như về chiến lược truy nhập, chúng chỉ là các định nghĩa về nội dung thông tin, về tính độc lập của dữ liệu trong mô hình quan niệm.

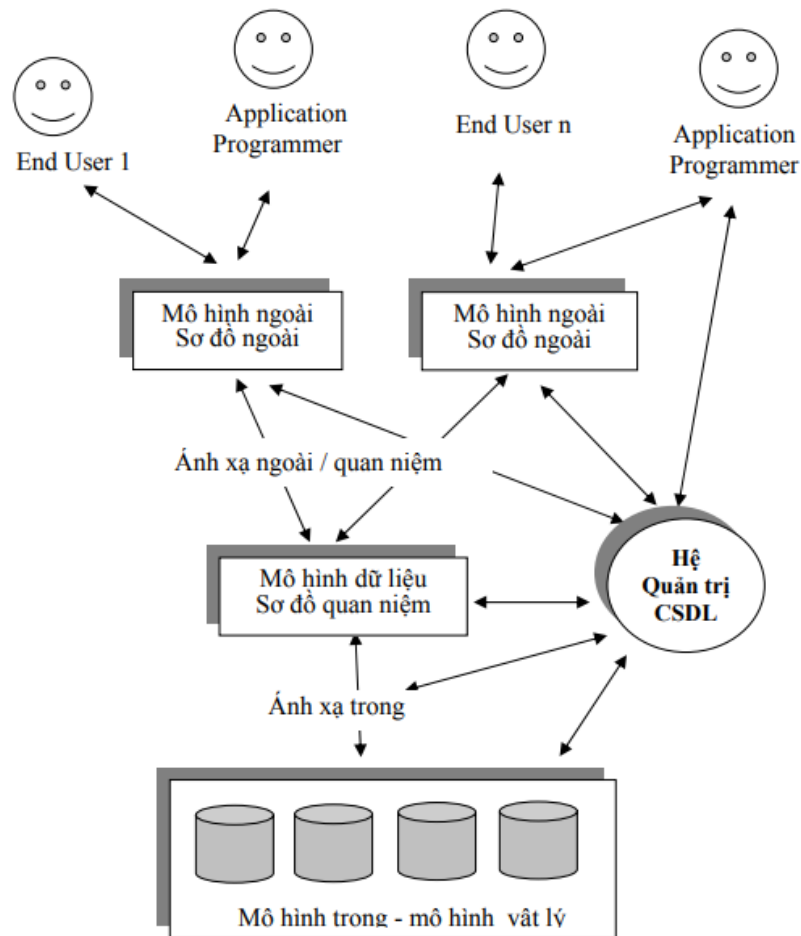
Sơ đồ quan niệm luôn luôn ổn định, nghĩa là nếu mô tả thêm một kiểu thực thể đặc biệt sát nhập vào sơ đồ dữ liệu, không được làm thay đổi sơ đồ dữ liệu cũ. Nếu sơ đồ dữ liệu không ổn định thì các ứng dụng và mô hình ngoài cũng không ổn định. Sơ đồ dữ liệu chỉ được thay đổi khi có sự điều chỉnh trong thế giới thực, đòi hỏi điều chỉnh lại định nghĩa sao cho nó phản ánh thế giới hiện thực khách quan hơn, chân lý hơn.

Thiết kế mô hình dữ liệu là giai đoạn quan trọng và quyết định trong việc thiết kế và cài đặt các hệ cơ sở dữ liệu. Quá trình thiết kế không phụ thuộc quá nhiều vào cấu trúc lưu trữ vật lý và chiến lược truy nhập của dữ liệu. Như vậy việc thiết kế sơ đồ dữ liệu phải được tiến hành độc lập với việc thiết kế sơ đồ trong và các sơ đồ ngoài liên kết, vì nếu không việc thiết kế sẽ không ổn định và phải xem xét lại tác động thường xuyên đến nhiều thành phần khác của hệ thống.

Với cách thiết kế truyền thống hiện nay, người thiết kế chỉ cung cấp một số sơ đồ trong và một tập các sơ đồ ngoài và họ coi đó là sơ đồ dữ liệu, là mô hình dữ liệu. Vì vậy tính không ổn định hệ thống, tính không phù hợp với các ứng dụng nảy sinh sau một thời gian hoạt động; mâu thuẫn và dị thường thông tin sẽ xảy ra; vi phạm tính toàn vẹn của dữ liệu.

Ngoài các định nghĩa về xuất hiện nhiều kiểu bản ghi quan niệm, sơ đồ dữ liệu còn chứa các định nghĩa về quyền truy nhập của người sử dụng, các thủ tục kiểm tra tính đúng đắn của dữ liệu nhằm bảo đảm tính toàn vẹn của CSDL, các luồng lưu chuyển thông tin và quy định cách thức sử dụng thông tin. Như vậy mô hình dữ liệu

là cách nhìn toàn bộ nội dung thông tin của CSDL, sơ đồ quan niệm là định nghĩa của cách nhìn ấy; là bước đi đầu tiên, quan trọng trong việc thiết kế và cài đặt các hệ cơ sở dữ liệu.



**Hình 1.2. Mô hình kiến trúc cơ sở dữ liệu**

### **1.2.3. Mô hình trong**

Mô hình trong là mô hình lưu trữ vật lý dữ liệu. Chỉ có duy nhất một và chỉ một cách biểu diễn CSDL dưới dạng lưu trữ vật lý. Mô hình trong là cách biểu diễn cơ sở dữ liệu trừu tượng ở mức thấp nhất.

- Mô hình trong gồm nhiều xuất hiện của nhiều kiểu bản ghi lưu trữ được xác định bởi một sơ đồ trong. Thông tin biểu diễn trong mô hình trong là duy nhất.

- Sơ đồ trong bao gồm các định nghĩa mô hình trong. Không chỉ xác định các kiểu khác nhau của bản ghi lưu trữ mà còn xác định rõ sự tồn tại của các chỉ dẫn, cách sắp xếp các bản ghi theo thứ tự nào. Nó xác định dữ liệu lưu trữ và truy nhập như thế nào thông qua các đường dẫn truy nhập tới dữ liệu.

Ánh xạ quan niệm trong được xác định giữa mô hình trong và mô hình dữ liệu nhằm bảo đảm tính độc lập của dữ liệu. Nếu cấu trúc lưu trữ của CSDL thay đổi, nghĩa là thay đổi định nghĩa về cấu trúc lưu trữ dữ liệu thì ánh xạ này phải cũng phải thay đổi tương ứng sao cho sơ đồ quan niệm (mô hình dữ liệu) không thay đổi.

Ánh xạ quan niệm ngoài là ánh xạ được xác định tương ứng một-một giữa mô hình ngoài của người sử dụng với mô hình dữ liệu.

### **1.3. Hệ quản trị cơ sở dữ liệu**

Hệ quản trị cơ sở dữ liệu (DataBase Management System - DBMS) là phần mềm điều khiển các chiến lược truy nhập CSDL. Khi người sử dụng đưa ra yêu cầu truy nhập bằng một ngôn ngữ truy xuất dữ liệu nào đó, hệ quản trị CSDL tiếp nhận và thực hiện các thao tác trên CSDL lưu trữ. Đối tượng nghiên cứu của CSDL là các thực thể và mối quan hệ giữa các thực thể. Mối quan hệ giữa các thực thể cũng là một loại thực thể đặc biệt. Trong cách tiếp cận CSDL theo mô hình quan hệ, dựa trên cơ sở lý thuyết đại số quan hệ để xây dựng các quan hệ chuẩn, khi kết nối không tổn thất thông tin và khi biểu diễn dữ liệu là duy nhất. Dữ liệu được lưu trữ trong bộ nhớ của máy tính không những phải tính đến yếu tố về tối ưu không gian lưu trữ, mà phải đảm bảo tính khách quan, trung thực của dữ liệu hiện thực. Nghĩa là phải đảm bảo tính nhất quán của dữ liệu và giữ được sự toàn vẹn của dữ liệu.

Các chức năng chủ yếu của một hệ quản trị cơ sở dữ liệu:

- Mô tả dữ liệu tạo lập và duy trì sự tồn tại của CSDL.
- Cho phép truy xuất vào CSDL theo thẩm quyền đã được cấp.
- Cập nhật, chèn thêm, loại bỏ hay sửa đổi dữ liệu.
- Đảm bảo an toàn, bảo mật dữ liệu và tính toàn vẹn dữ liệu.
- Tạo cấu trúc dữ liệu tương ứng với mô hình dữ liệu.
- Đảm bảo tính độc lập dữ liệu. Tức là cấu trúc lưu trữ dữ liệu độc lập với các trình ứng dụng dữ liệu.
- Tạo mối liên kết giữa các thực thể.
- Cung cấp các phương tiện sao lưu, phục hồi (backup, restore).
- Điều khiển tương tranh.

Các bước thực hiện của hệ quản trị CSDL có thể tóm tắt như sau:

- Người sử dụng đưa ra yêu cầu truy nhập bằng ngôn ngữ truy xuất dữ liệu.



- Hệ quản trị cơ sở dữ liệu sẽ tiếp nhận và phân tích yêu cầu.
- Hệ quản trị cơ sở dữ liệu xem xét sơ đồ ngoài, ánh xạ ngoài, sơ đồ quan niệm, ánh xạ trong,...
- Thực hiện các thao tác trên CSDL lưu trữ.

Một hệ quản trị cơ sở dữ liệu thông thường có các thành phần chính như sau:

- Ngôn ngữ định nghĩa dữ liệu (Data Definition Language).
- Ngôn ngữ thao tác dữ liệu (Data Manipulation Language).
- Ngôn ngữ truy vấn dữ liệu (Query Language).
- Bộ báo cáo (Report Write).
- Bộ đồ họa (Graphics Generator).
- Bộ giao tiếp ngôn ngữ chủ (Host Language Interface).
- Ngôn ngữ thủ tục (Procedure Language)
- Từ điển dữ liệu (Data Dictionary).

Để quản lý một CSDL cần có người quản trị CSDL. Người quản trị CSDL là một người hay một nhóm người có khả năng chuyên môn cao về công nghệ thông tin, có trách nhiệm quản lý và điều khiển toàn bộ hoạt động của các hệ CSDL. Vì vậy người quản trị CSDL cần phải đặt ra các hình thức, quy định cho người sử dụng nhằm ngăn chặn việc truy nhập trái phép vào các hệ CSDL. Người quản trị CSDL có thể cho phép người sử dụng những quyền truy nhập như chỉ được phép đọc, đọc một phần, có thể sửa, bổ sung một phần, v.v...

Người quản trị CSDL có một số nhiệm vụ chính:

- Xác định thực thể và nội dung thông tin cần lưu trữ. Xác định sơ đồ quan niệm đáp ứng yêu cầu truy nhập của người sử dụng.
- Quyết định cấu trúc lưu trữ và chiến lược truy nhập: Người quản trị CSDL phải xác định cách thức biểu diễn dữ liệu như mô tả cấu trúc lưu trữ trong, mô tả cấu trúc lưu trữ vật lý. Xác định mô hình dữ liệu, định nghĩa ánh xạ giữa cấu trúc lưu trữ và sơ đồ ngoài. Thực hiện các chiến lược lưu trữ, quản lý hệ thống.
- Người quản trị CSDL phải tạo môi trường giao tiếp giữa người sử dụng với các hệ CSDL, vì sơ đồ ngoài cho người sử dụng cách nhìn dữ liệu tương ứng với ngôn ngữ truy xuất dữ liệu thích hợp, nên người quản trị CSDL phải cung cấp sơ đồ quan

niệm, các ánh xạ, và cấu trúc lưu trữ. Kiểm soát thẩm quyền truy nhập của người sử dụng và bảo đảm quyền truy nhập của họ.

- Duy trì các tiêu chuẩn thống nhất về các thủ tục lưu trữ và cấu trúc lưu trữ, biểu diễn thông tin và các chiến lược truy nhập. Kiểm soát và kiểm tra tính đúng đắn của dữ liệu; áp dụng các biện pháp an toàn, an ninh dữ liệu.

- Xác định chiến lược lưu trữ, sao chép, phục hồi...trong các trường hợp hư hỏng do sai sót, hoặc trục trặc kỹ thuật.

## **CÂU HỎI ÔN TẬP CHƯƠNG 1**

1. Cơ sở dữ liệu là gì? Bạn hiểu thế nào là một hệ cơ sở dữ liệu tác nghiệp?
2. Sự cần thiết tổ chức lưu trữ dữ liệu theo lý thuyết cơ sở dữ liệu?
3. Cho ví dụ minh họa về giảm bớt dư thừa dữ liệu trong lưu trữ và không nhất quán dữ liệu trong lưu trữ làm cho dữ liệu mất đi tính toàn vẹn.
4. Trình bày tổng quát kiến trúc mô hình hệ cơ sở dữ liệu 3 lớp.
5. Trình bày và phân tích tính ổn định trong mô hình quan niệm.
6. Vai trò và chức năng của ánh xạ quan niệm trong và ánh xạ quan niệm ngoài.
7. Mục tiêu của các hệ cơ sở dữ liệu? Ví dụ minh họa.
8. Chứng minh rằng kiến trúc mô hình cơ sở dữ liệu 3 lớp đảm bảo được tính độc lập dữ liệu và tính ổn định cao.
9. Tại sao nói, mô hình dữ liệu là cách nhìn toàn bộ nội dung thông tin của CSDL, sơ đồ quan niệm là định nghĩa của cách nhìn ấy? Ví dụ minh họa.
10. Hiểu thế nào về khái niệm “tính toàn vẹn dữ liệu” và “tham chiếu toàn vẹn”.

## CHƯƠNG 2. TỔNG QUAN VỀ SQL SERVER

Trong chương này, trình bày một cách khái quát về hệ quản trị cơ sở dữ liệu Microsoft SQL Server. Các chức năng mà một hệ quản trị cơ sở dữ liệu cần phải có.

Nội dung chính của chương này gồm:

- Giới thiệu hệ quản trị cơ sở dữ liệu Microsoft SQL Server.
- Các thành phần của Microsoft SQL Server
- Mô hình Microsoft SQL Server
- Giới thiệu công cụ Enterprise Manager

### 2.1. Giới thiệu hệ quản trị cơ sở dữ liệu SQL server

#### 2.1.1. SQL Server

##### 2.1.1.1. SQL là gì?

SQL, viết tắt của *Structured Query Language* (ngôn ngữ hỏi có cấu trúc), là công cụ sử dụng để tổ chức, quản lý và truy xuất dữ liệu được lưu trữ trong các cơ sở dữ liệu. SQL là một hệ thống ngôn ngữ bao gồm tập các câu lệnh sử dụng để tương tác với cơ sở dữ liệu quan hệ.

SQL được phát triển từ ngôn ngữ SEQUEL bởi IBM theo mô hình E.F.Codd tại trung tâm nghiên cứu của IBM ở California vào những năm 70 cho hệ quản trị CSDL lớn. Đầu tiên SQL được sử dụng trong các ngôn ngữ quản lý CSDL và chạy trên các máy đơn lẻ. Song do sự phát triển nhanh chóng của nhu cầu xây dựng những CSDL lớn theo mô hình khách chủ: trong mô hình này toàn bộ CSDL được tập trung trên máy chủ (Server), mọi thao tác xử lý dữ liệu được thực hiện trên máy chủ bằng các lệnh SQL máy trạm chỉ dùng để cập nhập hoặc lấy thông tin từ máy chủ.

Tên gọi *ngôn ngữ hỏi có cấu trúc* phần nào làm chúng ta liên tưởng đến một công cụ (ngôn ngữ) dùng để truy xuất dữ liệu trong các cơ sở dữ liệu. Hơn thế nữa, SQL được sử dụng để điều khiển tất cả các chức năng mà một hệ quản trị cơ sở dữ liệu cung cấp cho người dùng bao gồm:

- **Định nghĩa dữ liệu:** SQL cung cấp khả năng định nghĩa các cơ sở dữ liệu, các cấu trúc lưu trữ và tổ chức dữ liệu đồng thời tạo mối quan hệ giữa các thành phần dữ liệu.
- **Truy xuất và thao tác dữ liệu:** Với SQL, người dùng có thể dễ dàng thực

hiện các thao tác truy xuất, bổ sung, cập nhật và loại bỏ dữ liệu trong các cơ sở dữ liệu.

- **Điều khiển truy cập:** SQL có thể được sử dụng để tạo và cấp phát quyền người dùng đồng thời kiểm soát các thao tác của người dùng trên dữ liệu, đảm bảo sự an toàn cho cơ sở dữ liệu.
- **Đảm bảo toàn vẹn dữ liệu:** SQL định nghĩa các ràng buộc toàn vẹn trong cơ sở dữ liệu nhờ đó đảm bảo tính hợp lệ và chính xác của dữ liệu trước các thao tác cập nhật cũng như các lỗi của hệ thống.

Như vậy, có thể nói rằng SQL là một ngôn ngữ hoàn thiện được sử dụng trong các hệ thống cơ sở dữ liệu và là một thành phần không thể thiếu trong các hệ quản trị cơ sở dữ liệu. Mặc dù SQL không phải là một ngôn ngữ lập trình như C, C++, Java,... song các câu lệnh mà SQL cung cấp có thể được nhúng vào trong các ngôn ngữ lập trình nhằm xây dựng các ứng dụng tương tác với cơ sở dữ liệu.

Khác với các ngôn ngữ lập trình quen thuộc như C, C++, Java,... SQL là ngôn ngữ có tính khai báo. Với SQL, người dùng chỉ cần mô tả các yêu cầu cần phải thực hiện trên cơ sở dữ liệu mà không cần phải chỉ ra cách thức thực hiện các yêu cầu như thế nào. Chính vì vậy, SQL là ngôn ngữ dễ tiếp cận và dễ sử dụng. Ngày nay trong các ngôn ngữ lập trình bậc cao đều có sự trợ giúp của SQL. Nhất là trong lĩnh vực phát triển của Internet ngôn ngữ SQL càng đóng vai trò quan trọng hơn, nó được sử dụng để nhanh chóng tạo các trang web động.

#### *2.1.1.2. SQL Server là gì?*

Microsoft SQL Server (hay còn gọi SQL Server) là một hệ thống quản lý cơ sở dữ liệu quan hệ được phát triển bởi Microsoft, có chức năng chính là lưu trữ, truy xuất dữ liệu và thông qua câu lệnh SQL nó có thể trao đổi dữ liệu với các ứng dụng trên máy Client. Trong môi trường Client/Server cơ sở dữ liệu được lưu trên máy Server, người dùng truy cập dữ liệu từ các máy Client qua một hệ thống mạng.

SQL Server hỗ trợ nhiều chức năng quản trị dữ liệu trên môi trường nhiều người dùng theo kiểu Client/Server. SQL Server chịu trách nhiệm đáp ứng các yêu cầu truy cập dữ liệu từ các máy Client, xử lý và chuyển kết quả về cho máy Server. SQL Server có một hệ thống bảo mật nhiều cấp giúp cho việc quản lý và bảo mật dữ liệu thuận tiện và chặt chẽ.

SQL Server có một số đặc tính sau:

- Cho phép quản trị một hệ CSDL lớn, có tốc độ xử lý dữ liệu nhanh đáp ứng yêu cầu về thời gian.
- Cho phép nhiều người cùng khai thác trong một thời điểm đối với một CSDL (tối đa 30000 user).
- Có hệ thống phân quyền bảo mật tương thích với hệ thống bảo mật của công nghệ NT (Network Technology), tích hợp với hệ thống bảo mật của Windows NT hoặc sử dụng hệ thống bảo mật độc lập của SQL Server.
- Hỗ trợ trong việc triển khai CSDL phân tán và phát triển ứng dụng trên Internet.
- Cho phép lập trình kết nối với nhiều ngôn ngữ lập trình khác dùng để xây dựng các ứng dụng (Visual Basic, C, C++, ASP, ASP.NET, XML,...).
- Sử dụng câu lệnh truy vấn dữ liệu Transact-SQL: là ngôn ngữ SQL mở rộng dựa trên SQL chuẩn của ISO (International Organization for Standardization) và ANSI (American National Standards Institute).

### ***2.1.2. Vai trò của SQL***

Bản thân SQL không phải là một hệ quản trị cơ sở dữ liệu, nó không thể tồn tại độc lập. SQL thực sự là một phần của hệ quản trị cơ sở dữ liệu, nó xuất hiện trong các hệ quản trị cơ sở dữ liệu với vai trò ngôn ngữ và là công cụ giao tiếp giữa người sử dụng và hệ quản trị cơ sở dữ liệu.

Trong hầu hết các hệ quản trị cơ sở dữ liệu quan hệ, SQL có những vai trò như sau:

- **SQL là ngôn ngữ hỏi có tính tương tác:** Người sử dụng có thể dễ dàng thông qua các trình tiện ích để gửi các yêu cầu dưới dạng các câu lệnh SQL đến cơ sở dữ liệu và nhận kết quả trả về từ cơ sở dữ liệu.
- **SQL là ngôn ngữ lập trình cơ sở dữ liệu:** Các lập trình viên có thể nhúng các câu lệnh SQL vào trong các ngôn ngữ lập trình để xây dựng nên các chương trình ứng dụng giao tiếp với cơ sở dữ liệu.
- **SQL là ngôn ngữ quản trị cơ sở dữ liệu:** Thông qua SQL, người quản trị cơ sở dữ liệu có thể quản lý được cơ sở dữ liệu, định nghĩa các cấu trúc lưu trữ dữ liệu, điều khiển truy cập cơ sở dữ liệu,...
- **SQL là ngôn ngữ cho các hệ thống client/server:** Trong các hệ thống cơ sở dữ liệu client/server, SQL được sử dụng như là công cụ để giao tiếp giữa các trình ứng dụng phía máy khách với máy chủ cơ sở dữ liệu.

- **SQL là ngôn ngữ truy cập dữ liệu trên Internet:** Cho đến nay, hầu hết các máy chủ Web cũng như các máy chủ trên Internet sử dụng SQL với vai trò là ngôn ngữ để tương tác với dữ liệu trong các cơ sở dữ liệu.
- **SQL là ngôn ngữ cơ sở dữ liệu phân tán:** Đối với các hệ quản trị cơ sở dữ liệu phân tán, mỗi một hệ thống sử dụng SQL để giao tiếp với các hệ thống khác trên mạng, gửi và nhận các yêu cầu truy xuất dữ liệu với nhau.
- **SQL là ngôn ngữ sử dụng cho các cổng giao tiếp cơ sở dữ liệu:** Trong một hệ thống mạng máy tính với nhiều hệ quản trị cơ sở dữ liệu khác nhau, SQL thường được sử dụng như là một chuẩn ngôn ngữ để giao tiếp giữa các hệ quản trị cơ sở dữ liệu.

## 2.2. Các thành phần của SQL Server

Các thành cơ bản trong SQL Server gồm có: Reporting Services, Database Engine, Integration Services, Notification Services, Full Text Search Service,... Tất cả kết hợp với nhau tạo thành một giải pháp hoàn chỉnh giúp cho việc phân tích và lưu trữ dữ liệu trở nên dễ dàng hơn.

+ **Database Engine:** Đây là phần lõi của SQL Server nó có chức năng chứa dữ liệu dưới dạng các bảng và hỗ trợ các kết nối đến CSDL. Ngoài ra, nó còn có khả năng tự điều chỉnh ví dụ: trả lại tài nguyên cho hệ điều hành khi một user log off và yêu cầu thêm các tài nguyên của máy khi cần.

+ **Integration Services:** là tập hợp các đối tượng lập trình và các công cụ đồ họa cho việc sao chép, di chuyển và chuyển đổi dữ liệu. Khi bạn làm việc trong một công ty lớn thì dữ liệu được lưu trữ ở nhiều nơi khác nhau như được chứa trong: Oracle, SQL Server, DB2, Microsoft Access,... và bạn chắc chắn sẽ có nhu cầu di chuyển dữ liệu giữa các server này. Ngoài ra, bạn còn muốn định dạng dữ liệu trước khi lưu vào database, Integration Services sẽ giúp bạn giải quyết được công việc này dễ dàng.

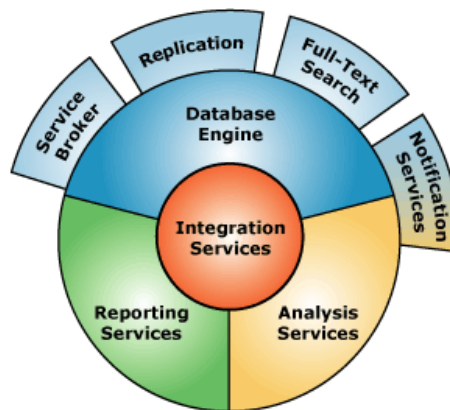
+ **Analysis Services:** Đây là một dịch vụ phân tích dữ liệu của Microsoft. Công cụ này giúp bạn trong việc phân tích dữ liệu một cách hiệu quả và dễ dàng bằng cách dùng kỹ thuật khai thác dữ liệu – datamining và khái niệm hình khối nhiều chiều – multi dimendion cubes.

+ **Notification Services:** Dịch vụ thông báo này là nền tảng cho sự phát triển và triển khai các ứng dụng soạn và gửi thông báo đến hàng ngàn người đăng ký sử dụng trên nhiều loại thiết bị khác nhau.

+ **Reporting Services**: là một công cụ tạo, quản lý và triển khai báo cáo trên server và client. Ngoài ra, nó còn là nền tảng cho việc phát triển và xây dựng các ứng dụng báo cáo.

+ **Full Text Search Service**: là một thành phần đặc biệt trong việc truy vấn và đánh chỉ mục dữ liệu văn bản không cấu trúc được lưu trữ trong các cơ sở dữ liệu SQL Server.

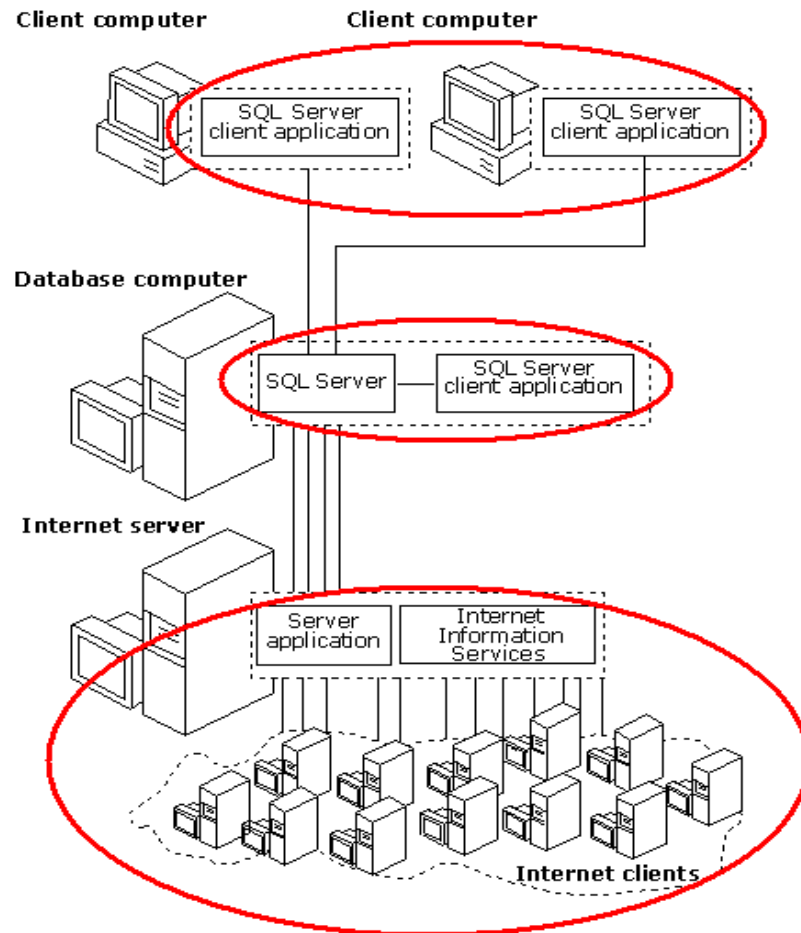
+ **Service Broker**: là dịch vụ cân bằng tải cho SQL Server. Nó giúp cho CSDL thực hiện được nhiều thao tác với dữ liệu bằng cách dùng hàng đợi lưu dữ liệu tạm thời.



*Hình 2.1. Các thành phần của SQL Server*

## 2.1. Mô hình chung SQL Server

SQL Server là hệ quản trị CSDL hoạt động theo mô hình Client/Server, có thể thực hiện trao đổi dữ liệu theo nhiều mô hình mạng khác nhau, nhiều giao thức và phương thức truyền tin khác nhau.



**Hình 2.2. Mô hình SQL Server trên hệ thống mạng**

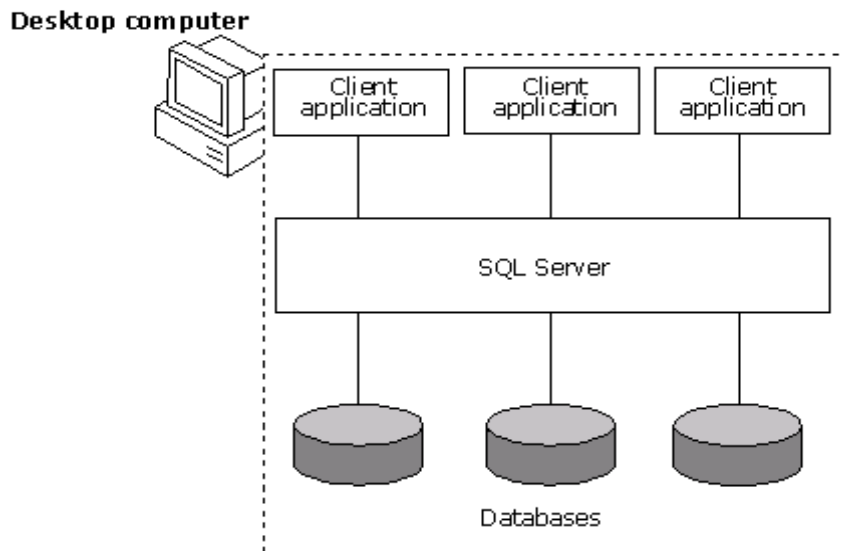
Trong sơ đồ trên thể hiện ba kiểu kết nối ứng dụng đến SQL Server:

- Kết nối trên Desktop: Có thể kết nối trên cùng máy chủ chứa CSDL SQL Server hoặc kết nối qua mạng nội bộ.
- Kết nối qua mạng diện rộng: Thông qua đường truyền mạng diện rộng kết nối đến máy chủ CSDL SQL Server.
- Kết nối qua mạng Internet: Các ứng dụng kết nối thông qua Internet, sử dụng dịch vụ webserver IIS thực hiện ứng dụng trên Internet (ASP, JSP, ASP.net,...)

## **2.2. Mô hình Desktop**

Trên Desktop mô hình SQL Server được thể hiện như sau:



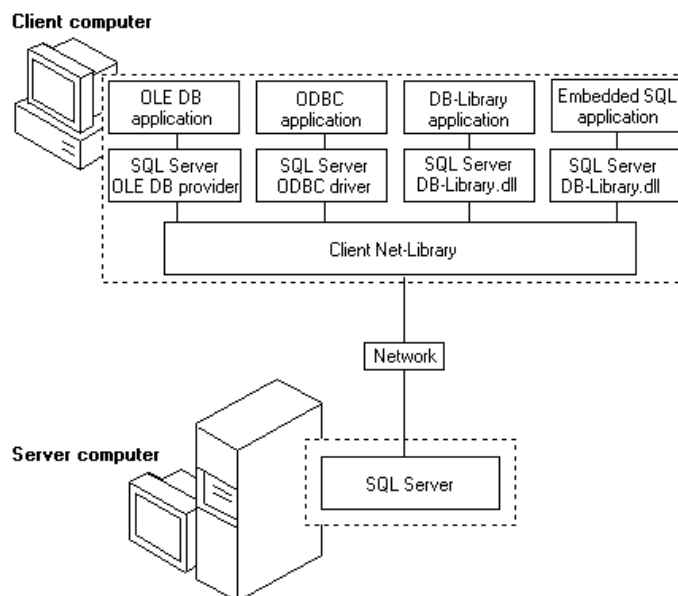


**Hình 2.3. Mô hình SQL Server trên Desktop**

Trên một Desktop có thể có nhiều ứng dụng, mỗi ứng dụng có thể thực hiện thao tác với nhiều CSDL.

### 2.3. Mô hình Client/Server

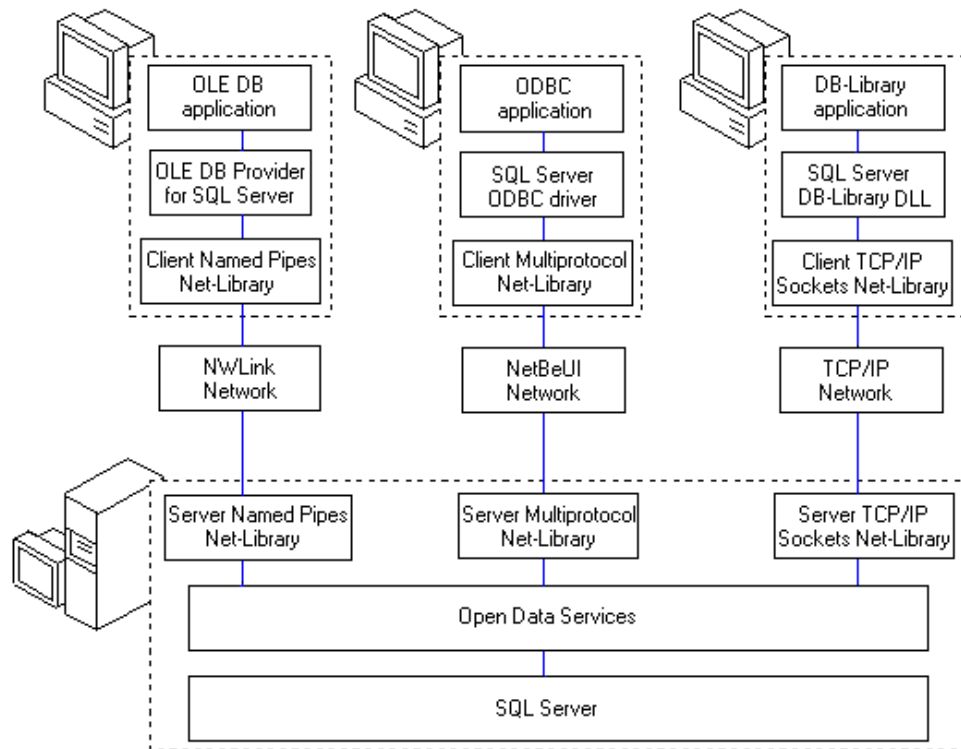
Trong mô hình client/server, ứng dụng trao đổi với SQL Server theo sơ đồ sau:



**Hình 2.4. Mô hình Client/Server của SQL Server**

Như sơ đồ trên nhận thấy SQL Server cho phép các ứng dụng kết nối theo các phương thức sau: OLE DB, ODBC, DB-Library, Embedded SQL, đây là các phương thức kết nối hữu ích cho những nhà phát triển ứng dụng.

Chi tiết hơn ta có sơ đồ sau:

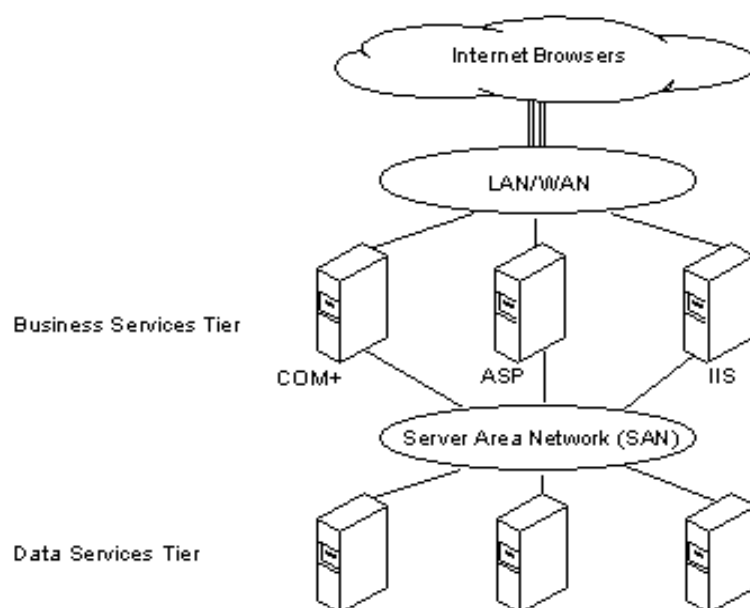


**Hình 2.5. Mô hình trao đổi dữ liệu với các ứng dụng**

Trong sơ đồ trên cho thấy, SQL Server có thể thực hiện trao đổi dữ liệu với các ứng dụng theo nhiều giao thức truyền tin khác nhau (TCP/IP, NetBEUI, Names Pipes,...), các ứng dụng có thể sử dụng nhiều phương thức kết nối khác nhau (OLE DB, ODBC, DB-Library).

#### 2.4. Mô hình kết nối ứng dụng trên mạng Internet

Các ứng dụng kết nối với SQL Server trên mạng Internet, các máy chủ SQL Server sẽ được quản lý thông qua các hệ thống máy chủ mạng, hệ điều hành mạng, các ứng dụng (COM+, ASP, IIS) sẽ thông qua máy chủ mạng kết nối đến SQL Server, mô hình này có thể áp dụng cho các mạng nội bộ, diện rộng, ứng dụng được khai thác trên trình duyệt Internet. Xem xét mô hình dưới đây:



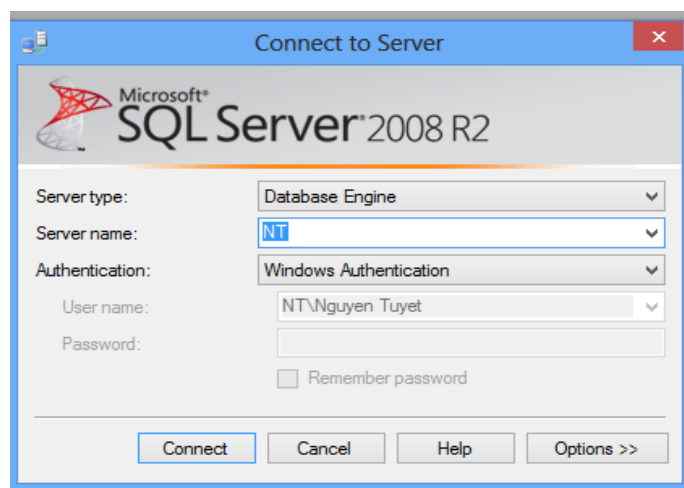
**Hình 2.6. Mô hình kết nối các ứng dụng**

## 2.5. SQL Server Management Studio

SQL Server được kết hợp bởi nhiều thành phần riêng biệt. Các thành phần này khi phối hợp với nhau sẽ tạo thành một giải pháp hoàn chỉnh giúp cho việc lưu trữ và phân tích dữ liệu trở nên dễ dàng hơn. Một trong những công cụ quan trọng của SQL Server là SQL Server Management Studio.

Với SQL Server Management Studio chúng ta có thể thực hiện được các thao tác với CSDL bằng câu lệnh hoặc trên giao diện người dùng. SQL Server Management Studio được thiết kế đơn giản và dễ sử dụng.

Để sử dụng Microsoft SQL Server Management Studio, đầu tiên cần phải đăng nhập vào CSDL SQL Server.

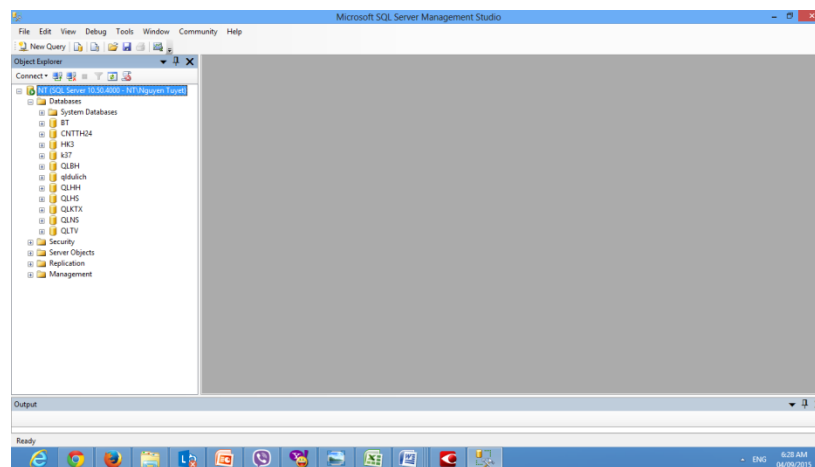


**Hình 2.7. Màn hình đăng nhập SQL Server**

Trong đó:

- Server Type: thường chọn là Database Engine. Các tùy chọn khác là kiểu kết nối khác đến máy chủ SQL Server.
- Server Name: chọn tên máy chủ SQL Server sẽ kết nối.
- Authentication: xác định cách xác thực đăng nhập. Khi cài đặt ở phần chọn quyền login, ta chọn Mixed Mode để có thể cho phép login bằng cả 2 quyền đó là Windows và SQL Server.

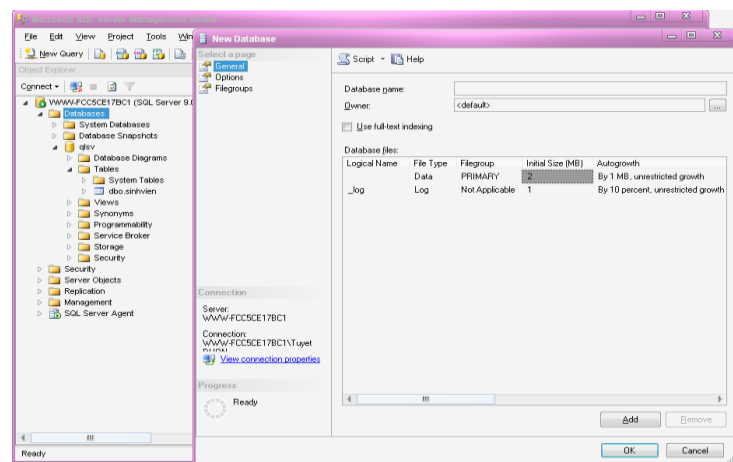
Sau khi nhấn nút Connect sẽ xuất hiện màn hình sau:



**Hình 2.8. Màn hình làm việc của SQL Server**

## 2.6. Tạo cơ sở dữ liệu bằng giao diện của SQL Server Management Studio

Bước 1. Chọn Database -> Nhấp chuột phải -> Chọn New Database.



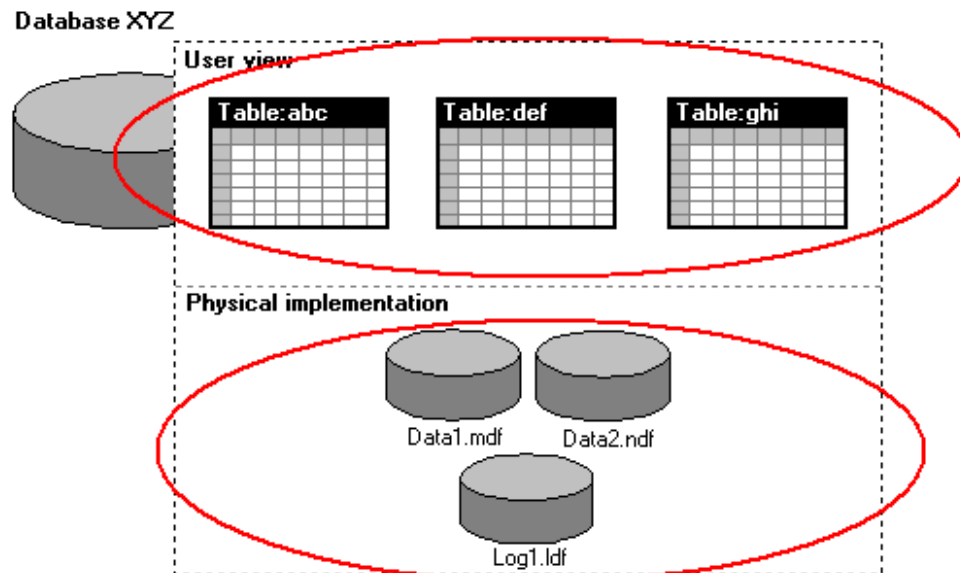
**Hình 2.9. Màn hình tạo database**

Bước 2: Nhập tên Database vào **Database Name**.

Lưu ý: Mỗi Database chỉ tồn tại với một **tên duy nhất**, không trùng lặp với bất kỳ tên các Database nào đã có.

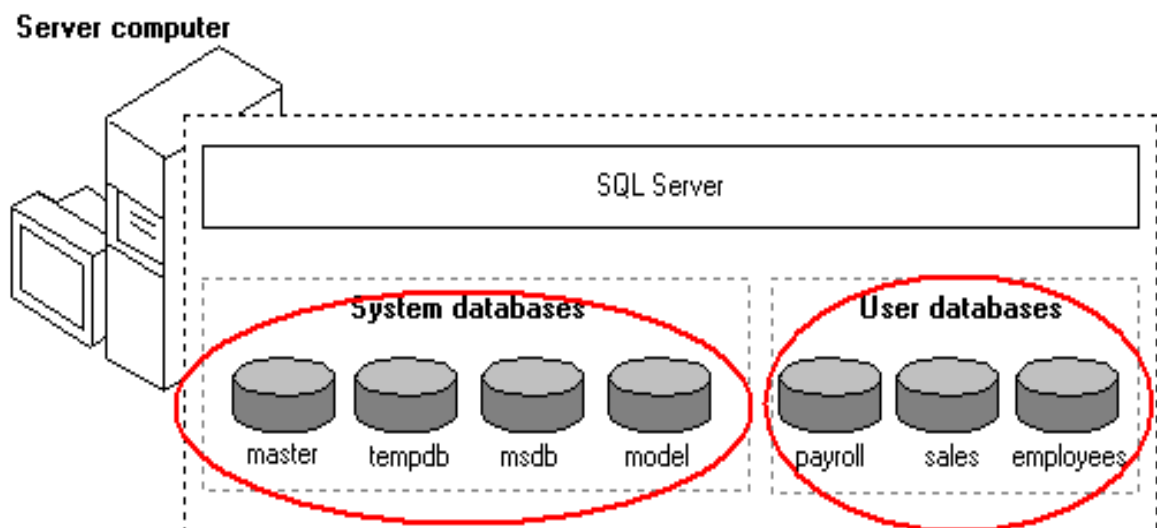
## 2.7. Cấu trúc lưu trữ cơ sở dữ liệu

Cơ sở dữ liệu trong SQL Server lưu trữ theo 2 phần: phần dữ liệu (gồm một tập tin chính \*.mdf và tập tin phụ \*.ndf) và phần nhật ký (\*.ldf).



*Hình 2.10. Sơ đồ lưu trữ cơ sở dữ liệu của SQL Server.*

Cơ sở dữ liệu trong SQL Server chia thành 2 loại: Cơ sở dữ liệu hệ thống (do SQL Server sinh ra khi cài đặt) và cơ sở dữ liệu người dùng (do người dùng tạo ra).



*Hình 2.11. Sơ đồ các cơ sở dữ liệu trong SQL Server*

CSDL hệ thống gồm:

- Master: Lưu trữ các thông tin login account, cấu hình hệ thống, thông tin quản trị các CSDL, là CSDL quan trọng nên thường được sao lưu để bảo đảm an toàn cho hệ thống.

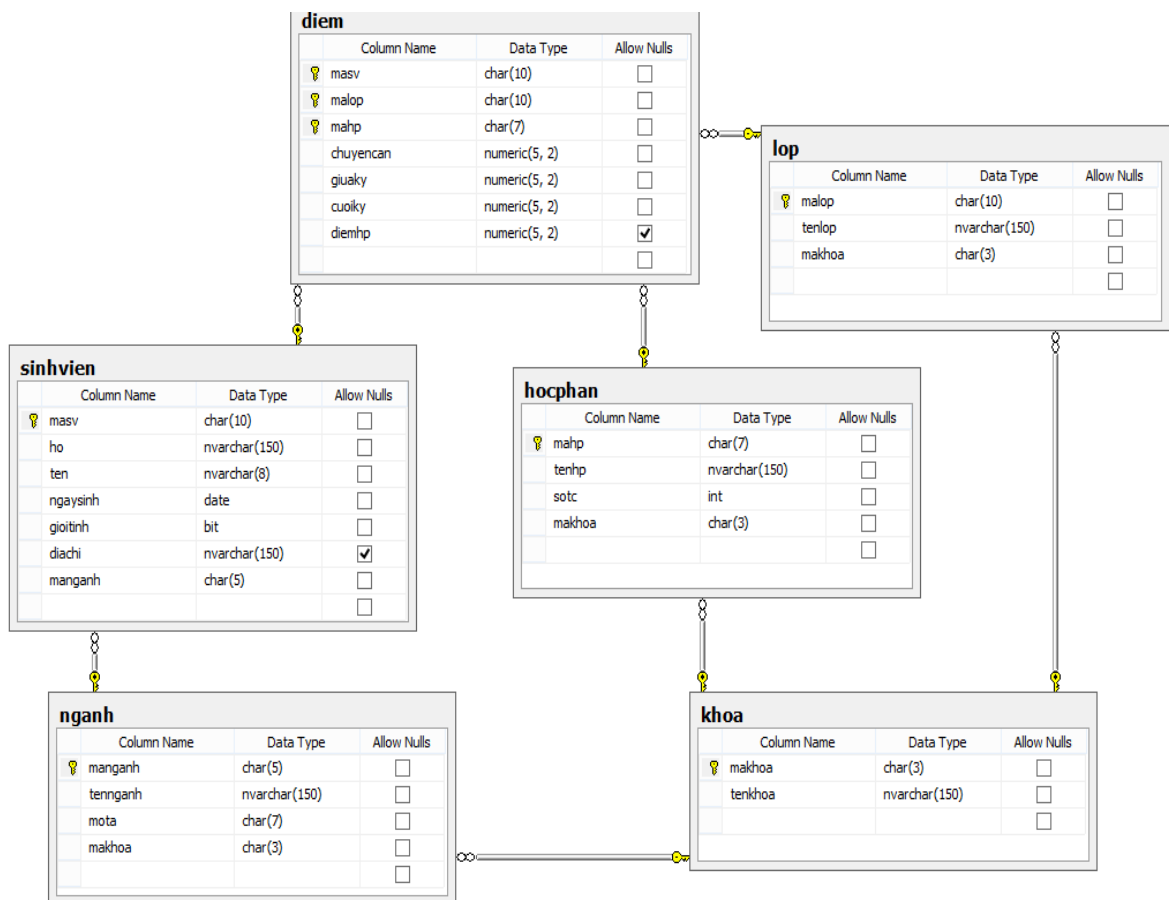
- Tempdb: Chứa các bảng và các thủ tục được lưu trữ tạm thời. Các bảng và thủ tục nói trên được lưu trữ trong CSDL này để phục vụ cho các user.

- Model: Database này đóng vai trò như một template cho các database khác. Nghĩa là khi một user database được tạo ra thì SQL Server sẽ copy toàn bộ các system objects (tables, stored procedures...) từ Model database sang database mới vừa tạo.

- Msdb: Database này được SQL Server Agent sử dụng để hoạch định các báo động và các công việc cần làm (schedule alerts and jobs).

## 2.8. Cơ sở dữ liệu minh họa

Cơ sở dữ liệu QLSINHVIEN- quản lý điểm sinh viên của Trường ĐH Quy Nhơn được sử dụng làm ví dụ xuyên suốt trong giáo trình.



Trong đó:

- Bảng KHOA lưu trữ danh mục Khoa.
- Bảng NGANH lưu trữ thông tin về các ngành đào tạo.
- Bảng LOP lưu trữ thông tin về các lớp học phần.
- Bảng SINHVIEN có dữ liệu là các thông tin sinh viên.
- Bảng HOCPHAN lưu trữ các học phần trong chương trình đào tạo.
- Bảng DIEM lưu trữ điểm các học phần của sinh viên.

## **CÂU HỎI ÔN TẬP CHƯƠNG 2**

1. Hệ quản trị CSDL quan hệ là gì?
2. SQL Server là gì? các thành phần trong SQL Server?
3. Kiến trúc SQL Server trong hệ thống client/server?

## CHƯƠNG 3. BẢNG DỮ LIỆU – TABLE

Trong chương này chúng ta sẽ tìm hiểu những kiến thức liên quan đến bảng dữ liệu, như:

- Các kiểu dữ liệu;
- Các ràng buộc dữ liệu;
- Tạo, sửa, xóa cấu trúc bảng;
- Nhập dữ liệu vào bảng;
- Cập nhật, xóa dữ liệu;
- Tạo chỉ mục.

### 3.1. Khái niệm bảng dữ liệu

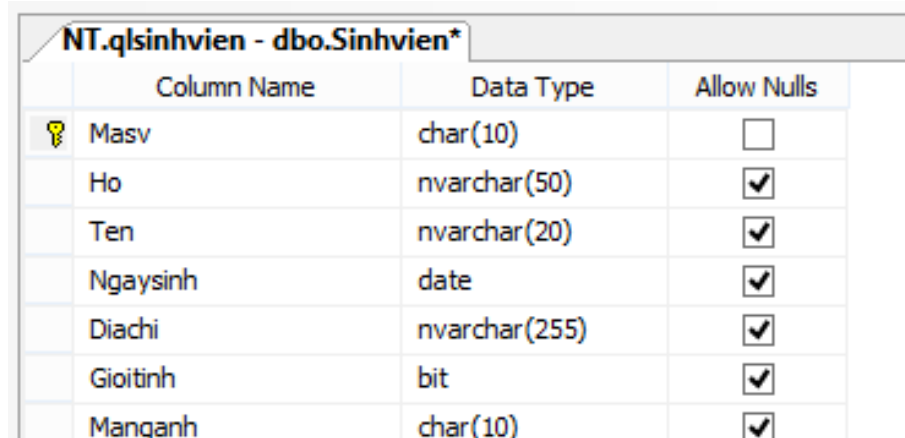
Bảng dữ liệu là một đối tượng của cơ sở dữ liệu, bảng dùng để lưu trữ dữ liệu, mỗi bảng gồm nhiều cột (còn gọi là trường - Field) và nhiều hàng (còn gọi là bản ghi - Record). Mỗi cột ứng với một loại dữ liệu, dữ liệu nhập vào có thể chấp nhận hoặc từ chối phụ thuộc vào nguyên tắc ràng buộc dữ liệu hoặc loại dữ liệu tương thích do người dùng định nghĩa.


Trường dữ liệu: là cột (hay gọi là thuộc tính) của bảng nhằm mô tả các đặc trưng riêng của đối tượng. Một trường dữ liệu cần phải có: tên trường, kiểu dữ liệu của trường, chiều dài của trường (hay gọi là độ rộng của trường),...

Bản ghi là một hàng dữ liệu của bảng trong CSDL, một bản ghi chứa thông tin mô tả các thuộc tính của đối tượng trong bảng.

Ví dụ sau mô tả bảng dữ liệu *Sinhvien* chứa thông tin sinh viên gồm: Mã sinh viên, Họ, Tên, Ngày sinh, Địa chỉ, Giới tính, Mã ngành

- Cấu trúc bảng:



	Column Name	Data Type	Allow Nulls
	Masv	char(10)	<input type="checkbox"/>
	Ho	nvarchar(50)	<input checked="" type="checkbox"/>
	Ten	nvarchar(20)	<input checked="" type="checkbox"/>
	Ngaysinh	date	<input checked="" type="checkbox"/>
	Diachi	nvarchar(255)	<input checked="" type="checkbox"/>
	Gioitinh	bit	<input checked="" type="checkbox"/>
	Manganh	char(10)	<input checked="" type="checkbox"/>

**Hình 3.1. Tạo cấu trúc bảng trong SQL Server**



- Dữ liệu đưa vào bảng

NT.qlsinhvien - dbo.Sinhvien		NT.qlsinhvien - dbo.Sinhvien*					
Masv	Ho	Ten	Ngaysinh	Diachi	Gioitinh	Manganh	
3651080041	Tô Nhật	Tịnh	1995-08-07...	H. Phú Mỹ, T. Bình Định	0	11615	
3651080042	Đào Duy	Toàn	1995-09-11...	H. Nghĩa Hành, T. Quảng Ngãi	0	11615	
3651080044	Ngô	Tồn	1995-08-07...	Tp. Quy Nhơn, T. Bình Định	0	11615	
3651080058	Trần Văn	Trình	1993-02-10...	H. Phú Mỹ, T. Bình Định	0	11615	
3651080045	Võ Phan Thanh	Tú	1995-03-21...	H. An Nhơn, T. Bình Định	0	11615	
3651080061	Trần Thế	Vinh	1995-02-02...	H. Ninh Phước, T. Ninh Thuận	0	11615	
3651080063	Phan Tấn	Vỹ	1994-05-02...	H. Phú Hoà, T. Phú Yên	0	11615	
3651080064	Nguyễn Trương Thảo	Xuyên	1995-08-31...	H. Hoài Nhơn, T. Bình Định	1	11615	
3653030001	Nguyễn Văn	Ấn	1995-08-14...	H. Đức Phổ, T. Quảng Ngãi	0	10411	
3653030002	Nguyễn Tấn	Bửu	1995-11-28...	Tp. KonTum, T. KonTum	0	10411	
3653030003	Trương Thị Mỹ	Diên	1995-10-06...	Tp. Quy Nhơn, T. Bình Định	1	10411	
3653030004	Trương Thị Thủy	Dung	1995-09-02...	H. Phú Hoà, T. Phú Yên	1	10411	

**Hình 3.2. Dữ liệu bảng Sinhvien**

Khi định nghĩa bảng dữ liệu, ngoài việc xác định tên bảng, tên trường,... cần phải xác định các thành phần khác như: kiểu dữ liệu, các ràng buộc, các khóa, ... Các thành phần này nhằm tạo ra các ràng buộc toàn vẹn dữ liệu của các bảng trong cơ sở dữ liệu.

### 3.2. Kiểu dữ liệu

Kiểu dữ liệu (Data Type) là một quy định về cấu trúc, miền giá trị của dữ liệu có thể nhập vào và tập các phép toán có thể thao tác trên miền giá trị đó.

Mỗi một cột (trường) của bảng phải thuộc một kiểu dữ liệu nhất định đã được định nghĩa trước. Mỗi kiểu dữ liệu quy định các giá trị dữ liệu cho phép đối với cột đó.

Dưới đây là một số kiểu dữ liệu chuẩn do SQL Server cung cấp:

*Nhóm kiểu dữ liệu Binary:* là các kiểu dữ liệu chứa giá trị nhị phân dùng để lưu trữ hình ảnh, âm thanh và video.

**Bảng 3.1. Các kiểu dữ liệu thuộc nhóm Binary**

Kiểu dữ liệu	Phạm vi biểu diễn
Binary (n)	Độ dài cố định n byte, trong đó n có giá trị từ 1 đến 8000.
Varbinary (n)	Độ dài tối đa n byte, độ dài đó có thể thay đổi theo dữ liệu thực tế, n có giá trị từ 1 đến 8000.
Image	Độ dài tối đa 2.147.483.647 bytes, độ dài thay đổi theo dữ liệu thực tế.

*Nhóm kiểu dữ liệu Text:* là các kiểu dữ liệu dùng để lưu trữ xâu ký tự, bao gồm:

**Bảng 3.2. Các kiểu dữ liệu thuộc nhóm kiểu Text**

<b>Kiểu dữ liệu</b>	<b>Phạm vi biểu diễn</b>
Char (n)	Xâu ký tự có độ dài cố định n ký tự, n từ 1 đến 8000, lưu trữ các ký tự có trong bảng mã ASCII.
Nchar (n)	Xâu ký tự có độ dài cố định n ký tự, n từ 1 đến 4000, lưu trữ các ký tự có trong bảng mã Unicode.
Varchar (n)	Xâu ký tự có độ dài tối đa n ký tự, n từ 1 đến 8000, độ dài này sẽ thay đổi theo dữ liệu thực tế, lưu trữ các ký tự có trong bảng mã ASCII.
Nvarchar (n)	Xâu ký tự có độ dài tối đa n ký tự, n từ 1 đến 4000, độ dài này sẽ thay đổi theo dữ liệu thực tế, lưu trữ các ký tự có trong bảng mã Unicode.
Text (n)	Dùng để lưu trữ dữ liệu dạng văn bản, chứa cả ký tự xuống dòng, độ dài tối đa là 2.147.483.647 ký tự, cơ chế quản lý dữ liệu theo dạng con trỏ, lưu trữ các ký tự có trong bảng mã ASCII.
Ntext (n)	Tương tự như Text(n), tối đa 1.073.741.823 ký tự, lưu trữ các ký tự có trong bảng mã Unicode.

*Nhóm kiểu dữ liệu Data/Time:* là các kiểu dữ liệu dùng để lưu trữ các giá trị ngày và thời gian của một ngày.

**Bảng 3.3. Các kiểu dữ liệu thuộc nhóm dữ liệu Date/Time**

<b>Kiểu dữ liệu</b>	<b>Phạm vi biểu diễn</b>
Datetime	Jan 1, 1753 00:00:0000 ÷ Dec 31, 9999 00:00:0000
Smalldatetime	Jan 1, 1900 00:00:0000 ÷ Jun 6, 2079 00:00:0000
Date	Lưu trữ một kiểu ngày dưới dạng June 30, 1991
Time	Lưu trữ một kiểu giờ dưới dạng 12:30 PM

*Nhóm kiểu dữ liệu Numeric:* là các kiểu dữ liệu dùng để lưu trữ dữ liệu số, bao gồm:

+ Số nguyên

**Bảng 3.4. Các kiểu dữ liệu số nguyên**

Kiểu dữ liệu	Phạm vi biểu diễn
Bigint	$-9,223,372,036,854,775,808 \div 9,223,372,036,854,775,807$
Int	$-2,147,483,648 \div 2,147,483,647$
Smallint	$-32,768 \div 32,767$
Tinyint	$0 \div 255$

+ Số thực

**Bảng 3.5. Các kiểu dữ liệu số thực**

Kiểu dữ liệu	Phạm vi biểu diễn
Decimal	$-10^{38} + 1 \div 10^{38} - 1$
Numeric	$-10^{38} + 1 \div 10^{38} - 1$
Float	$-1.79E + 308 \div 1.79E + 308$
Real	$-3.40E + 38 \div 3.40E + 38$

*Nhóm kiểu dữ liệu Monetary:* là các kiểu dữ liệu dùng để lưu trữ giá trị tiền tệ.

**Bảng 3.6. Các kiểu dữ liệu tiền tệ**

Kiểu dữ liệu	Phạm vi biểu diễn
Money	$-922,337,203,685,477.5808 \div 922,337,203,685,477.5807$
Smallmoney	$-214,748.3648 \div 214,748.3647$

*Bit:* là kiểu dữ liệu mà một trường khai báo kiểu này chỉ nhận một trong hai giá trị 0 hoặc 1.

*Các kiểu dữ liệu khác*

**Bảng 3.7. Các kiểu dữ liệu mở rộng**

Kiểu dữ liệu	Phạm vi biểu diễn
Sql_variant	Là kiểu dữ liệu có thể lưu trữ nhiều loại dữ liệu có kiểu khác nhau trong cùng một cột của một bảng. Ví

	dù có thể lưu trữ nhiều loại dữ liệu kiểu int, binary, char, nhưng không chứa dữ liệu kiểu text, ntext, image, timestamp.
Timestamp	Là kiểu dữ liệu có kích thước 8 bytes, lưu trữ dạng số nhị phân do hệ thống tự sinh ra, mỗi giá trị thuộc kiểu timestamp trong CSDL là duy nhất.
Uniqueidentifier	Là kiểu dữ liệu dùng để lưu trữ các giá trị nhị phân, kích thước 16 bytes. Một GUID (Globally Unique Identifier) là một số nhị phân duy nhất; vì thế GUID được sử dụng cho mục đích là khóa chính trong các cơ sở dữ liệu lớn.
XML	Là kiểu dữ liệu dùng để lưu trữ dữ liệu XML.
Geometry	Là kiểu dữ liệu không gian được sử dụng để biểu diễn các đối tượng trong hệ tọa độ Euclide.
Geography	Là kiểu dữ liệu không gian được sử dụng để biểu diễn các đối tượng trong hệ tọa độ trái đất.

### 3.3. Ràng buộc dữ liệu

Ràng buộc dữ liệu là các quy tắc trong một cơ sở dữ liệu nhằm kiểm tra tính đúng đắn và hợp lệ của dữ liệu trước khi lưu trữ.

- Một ràng buộc luôn gắn với một bảng;
- Nếu không đặt tên cho ràng buộc thì hệ thống sẽ tự động phát sinh tên cho ràng buộc;
- Có thể tạo ràng buộc cùng với thời điểm tạo bảng hoặc sau khi đã tạo bảng xong.

*Các loại ràng buộc trong SQL Server:*

- Toàn vẹn thực thể (Entry integrity): Mỗi thực thể đều được xác định theo một khóa, khi biết khóa ta hoàn toàn có thể xác định được thực thể tương ứng. Ví dụ ràng buộc khóa chính là loại ràng buộc toàn vẹn thực thể.
- Toàn vẹn theo miền (Domain integrity): Là loại toàn vẹn có tác dụng với các cột dữ liệu trong một phạm vi nào đó. Ràng buộc toàn vẹn theo miền để bảo đảm tính hợp lệ của dữ liệu trong một thuộc tính. Ví dụ ràng buộc check là loại ràng buộc toàn vẹn theo miền.

- Toàn vẹn dạng tham chiếu (Referential integrity): Khi một bảng có quan hệ với một bảng khác theo một mối quan hệ, trong mối quan hệ đó khóa ngoài sẽ là khóa tham chiếu của khóa chính, giá trị của khóa ngoài sẽ thuộc tập các giá trị của khóa chính hoặc giá trị NULL. Ví dụ ràng buộc kiểu quan hệ (Relationship) là ràng buộc toàn vẹn tham chiếu.
- Toàn vẹn do người dùng định nghĩa (User-defined integrity): Là các quy định dữ liệu nhập theo một số qui tắc được định nghĩa bởi người dùng. Các qui tắc này là khác với các loại toàn vẹn dữ liệu trên.

Bốn loại ràng buộc toàn vẹn nói trên ta có thể thống kê tương ứng với các khóa, quy tắc ràng buộc trong SQL Server như sau:

**Bảng 3.8. Các loại ràng buộc**

Kiểu toàn vẹn	Công cụ trong SQL Server
Entry integrity	<ul style="list-style-type: none"> <li>- Ràng buộc Primary key</li> <li>- Ràng buộc Unique</li> <li>- Cột Identity</li> </ul>
Domain integrity	<ul style="list-style-type: none"> <li>- Giá trị mặc định Default</li> <li>- Ràng buộc khóa ngoài Foreign Key</li> <li>- Ràng buộc Check</li> <li>- Thuộc tính NOT NULL</li> </ul>
Referential integrity	<ul style="list-style-type: none"> <li>- Ràng buộc khóa ngoài Foreign Key</li> <li>- Ràng buộc Check</li> </ul>
User-defined integrity	<ul style="list-style-type: none"> <li>- Rules</li> <li>- Stored procedures</li> <li>- Triggers</li> </ul>

Có thể phân các ràng buộc trên thành 2 loại:

- Loại đơn giản: sử dụng CONSTRAINT để mô tả;
- Loại phức tạp: sử dụng TRIGGER để thực hiện (trình bày ở chương 8).

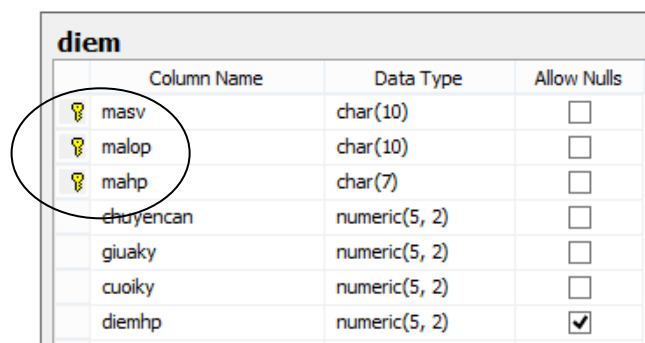
Các loại ràng buộc đơn giản gồm: Kiểm tra duy nhất PRIMARY KEY, UNIQUE; Kiểm tra khác rỗng NOT NULL; Kiểm tra tồn tại FOREIGN KEY; Kiểm tra miền giá trị CHECK, DEFAULT.

### 3.3.1. Khóa chính (Primary Key)

Khóa chính là một cột hoặc tổ hợp nhiều cột được dùng để xác định duy nhất mỗi bản ghi trong một bảng dữ liệu. Ngoài ra, nó còn dùng để thiết lập quan hệ (hay ràng buộc tham chiếu) giữa các bảng trong cơ sở dữ liệu. Nếu một cột được định nghĩa khóa chính thì cột đó không nhận giá trị NULL.

Dưới đây là một số điểm lưu ý khi sử dụng khóa chính:

- Một bảng dữ liệu chỉ được phép có duy nhất một khóa chính
- Muốn đặt khóa chính trên cột thì cột đó không được chứa các giá trị giống nhau và phải là cột NOT NULL. Ví dụ: yêu cầu các sinh viên khác nhau có mã khác nhau; để thỏa mãn điều kiện này, ta đặt khóa chính trên cột mã sinh viên (*masv*).
- Muốn đặt khóa chính trên một tổ hợp nhiều cột thì tổ hợp cột đó không được chứa tập giá trị giống nhau và mỗi cột của tổ hợp cột đó phải NOT NULL. Ví dụ có thể đặt khóa chính cho bảng *Diem* từ tổ hợp ba cột: mã sinh viên (*masv*), mã lớp học phần (*malop*) và mã học phần (*mahp*) như sau, khi đó sẽ không có bộ giá trị nào được phép trùng nhau trên ba cột đó:



	Column Name	Data Type	Allow Nulls
🔑	masv	char(10)	<input type="checkbox"/>
🔑	malop	char(10)	<input type="checkbox"/>
🔑	mahp	char(7)	<input type="checkbox"/>
	chuyencan	numeric(5, 2)	<input type="checkbox"/>
	giuaky	numeric(5, 2)	<input type="checkbox"/>
	cuoiky	numeric(5, 2)	<input type="checkbox"/>
	diemhp	numeric(5, 2)	<input checked="" type="checkbox"/>

Hình 3.3. Tổ hợp 3 cột làm khóa chính trong bảng *Diem*

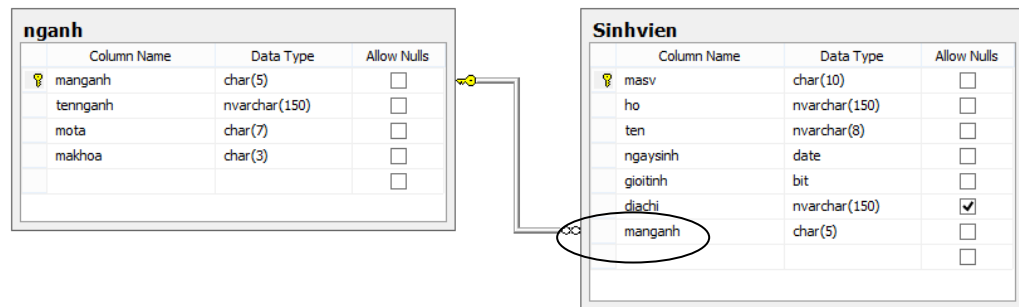
### 3.3.2. Khóa ngoài (Foreign Key)

Các bảng dữ liệu trong một cơ sở dữ liệu thường có mối quan hệ với nhau. Những mối quan hệ này biểu diễn cho sự quan hệ giữa các đối tượng trong thế giới thực. Về mặt dữ liệu, những mối quan hệ được đảm bảo thông qua việc đòi hỏi sự có mặt của một giá trị dữ liệu trong bảng này phải phụ thuộc vào sự tồn tại của giá trị dữ liệu đó ở trong một bảng khác.

Ràng buộc khóa ngoài được sử dụng để liên kết các bảng dữ liệu trong một CSDL. Khóa ngoài được đặt trên một cột, nó tham chiếu đến khóa chính trong một

bảng khác. Khóa ngoài và khóa chính phải cùng kiểu dữ liệu và cùng kích thước. Giá trị của khóa ngoài luôn nằm trong tập giá trị của khóa chính trong mối quan hệ đã thiết lập.

Ví dụ: Muốn biết thông tin sinh viên học ngành nào, thì bảng *Sinhvien* cần định nghĩa cột *manganh* làm khóa ngoài và tham chiếu đến *manganh* của bảng *nganh*.



**Hình 3.4. Liên kết khóa ngoài với khóa chính**

### 3.3.3. Ràng buộc Unique

Trong một bảng chỉ có duy nhất một khóa chính nhưng có thể có nhiều cột có tính chất như khóa chính (tức là giá trị trong cột đó là duy nhất). Nếu những cột đó không được chọn làm khóa chính thì chúng phải được thiết lập ràng buộc Unique. Như vậy, ràng buộc Unique được sử dụng để đảm bảo rằng các giá trị được nhập vào trong cột đó là duy nhất.

Ví dụ, mỗi sinh viên có một email riêng không trùng với bất kỳ email của sinh viên nào khác. Để đảm bảo điều kiện này, thuộc tính *email* của bảng *Sinhvien* nên đặt ràng buộc Unique.

Một bảng có thể có nhiều cột được ràng buộc Unique. Cột đặt ràng buộc Unique có thể lưu trữ giá trị NULL, nhưng chỉ lưu trữ được duy nhất một lần.

### 3.3.4. Ràng buộc Check

Ràng buộc Check cho phép kiểm tra giá trị dữ liệu nhập vào cột thông qua một biểu thức điều kiện. Nếu biểu thức điều kiện nhận giá trị False (với bộ dữ liệu đang nhập), thì dữ liệu vi phạm ràng buộc này và không được nhập vào trong bảng.

Ví dụ: Cột *diem* của bảng *Sinhvien* chỉ lưu trữ các điểm số từ 0 đến 10. Để đảm bảo điều kiện này, chúng ta sử dụng ràng buộc Check để giới hạn giá trị nhập vào cột điểm của sinh viên chỉ trong đoạn [0-10], người dùng sẽ không thể nhập những giá trị nằm ngoài đoạn [0-10].

### 3.3.5. Ràng buộc giá trị mặc định (Default).

Ràng buộc này được dùng để gán giá trị mặc định cho cột nếu giá trị của cột đó không được nhập, nếu nhập giá trị khác thì nó sẽ nhận giá trị mới nhập này.

Mỗi bảng có thể có nhiều cột được đặt ràng buộc Default. Mỗi cột của bảng chỉ có một giá trị mặc định.

### 3.4. Tạo bảng dữ liệu

Khi tạo bảng, chúng ta cần xác định:

- Bảng mới được tạo ra sử dụng với mục đích gì và có vai trò như thế nào trong cơ sở dữ liệu?

- Cấu trúc của bảng bao gồm những cột nào? Mỗi một cột có ý nghĩa như thế nào trong việc biểu diễn dữ liệu? Kiểu dữ liệu của mỗi cột là gì? Cột đó có cho phép nhận giá trị NULL hay không?

- Những cột nào sẽ tham gia vào khóa chính của bảng? Bảng có quan hệ với những bảng khác hay không và nếu có thì quan hệ như thế nào?

- Trên các cột của bảng có tồn tại những ràng buộc về khuôn dạng, điều kiện hợp lệ của dữ liệu hay không; nếu có thì sử dụng ở đâu và như thế nào?

Để tạo bảng trong cơ sở dữ liệu ta dùng câu lệnh CREATE TABLE. Cú pháp như sau:

*CREATE TABLE tên\_bảng*

*(tên\_cột kiểu\_dữ\_liệu\_cột các\_ràng\_buộc [, ...,*

*tên\_cột kiểu\_dữ\_liệu\_cột các\_ràng\_buộc] [,các\_ràng\_buộc\_trên\_bảng])*

**Ví dụ 1:** Câu lệnh dưới đây định nghĩa bảng *Nganh* gồm các cột *manganh* (mã ngành), *tennganh* (tên ngành), *chitieu* (chỉ tiêu tuyển sinh của ngành học). Ràng buộc cột *manganh* làm khóa chính.

```
CREATE TABLE Nganh
```

```
(manganh          CHAR(5)          NOT NULL
```

```
                CONSTRAINT pk_manganh PRIMARY KEY,
```

```
tennganh          NVARCHAR(50)     NOT NULL,
```

```
chitieu           int               NOT NULL)
```



**Ví dụ 2:** Câu lệnh dưới đây định nghĩa bảng *Sinhvien* với các cột *masv* (mã sinh viên), *ho* (họ), *ten* (tên), *gioitinh* (giới tính), *ngaysinh* (ngày sinh của sinh viên), *dienthoai* (điện thoại), *diachi* (địa chỉ liên hệ), *manganh* (mã ngành sinh viên theo học). Ràng buộc cột *masv* làm khóa chính, *manganh* là khóa ngoài (tham chiếu đến *manganh* của bảng *nganh*), *ngaysinh* chỉ nhập sinh viên từ 18 tuổi trở lên.

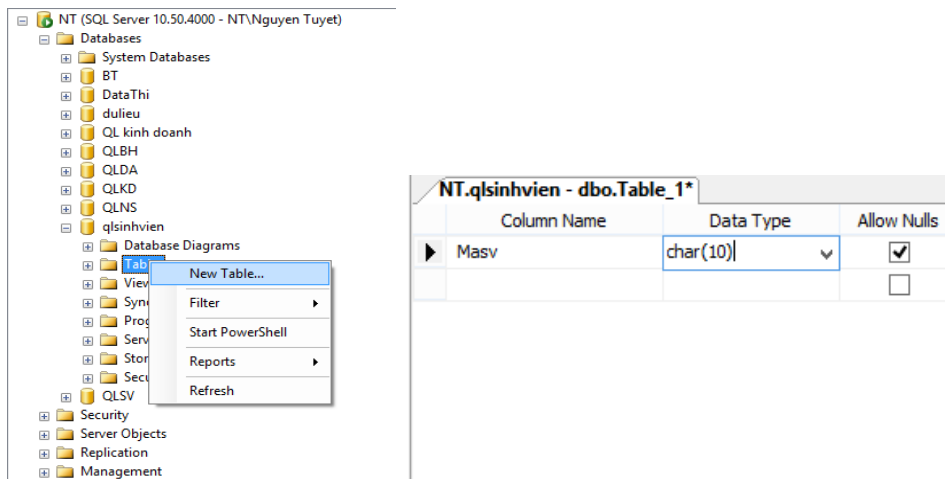
```
CREATE TABLE Sinhvien
(masv          CHAR(5)          NOT NULL
                        CONSTRAINT pk_masv PRIMARY KEY,
ho             NVARCHAR(150) NOT NULL,
ten            NVARCHAR(8)   NOT NULL,
gioitinh       BIT           NULL,
ngaysinh       DATE          NULL
                        CONSTRAINT chk_ngaysinh
CHECK (YEAR(GETDATE())-YEAR(ngaysinh)>=18),
dienthoai      CHAR(11)      NULL,
diachi         NVARCHAR (50) NULL,
manganh        CHAR(5)       NOT NULL
                        CONSTRAINT fk_manganh FOREIGN KEY(manganh)
REFERENCES nganh(manganh)
ON DELETE CASCADE ON UPDATE CASCADE)
```

Trong đó:

Getdate() là hàm lấy ngày tháng năm hiện tại của hệ thống

Year(D) là hàm lấy ra năm của ngày D

Ngoài ra, có thể tạo bảng bằng chức năng New Table có sẵn của Microsoft SQL Server:



**Hình 3.5. Tạo bảng bằng chức năng New Table của Microsoft SQL Server**

### 3.4.1. Tạo ràng buộc PRIMARY KEY

Để khai báo một ràng buộc PRIMARY KEY, ta sử dụng cú pháp:

*[CONSTRAINT tên\_ràng\_buộc] PRIMARY KEY [(danh\_sách\_cột)]*

Trong đó: *tên\_ràng\_buộc* do người dùng tự đặt, *danh\_sách\_cột* (nếu có) liệt kê tổ hợp các cột làm khóa chính.

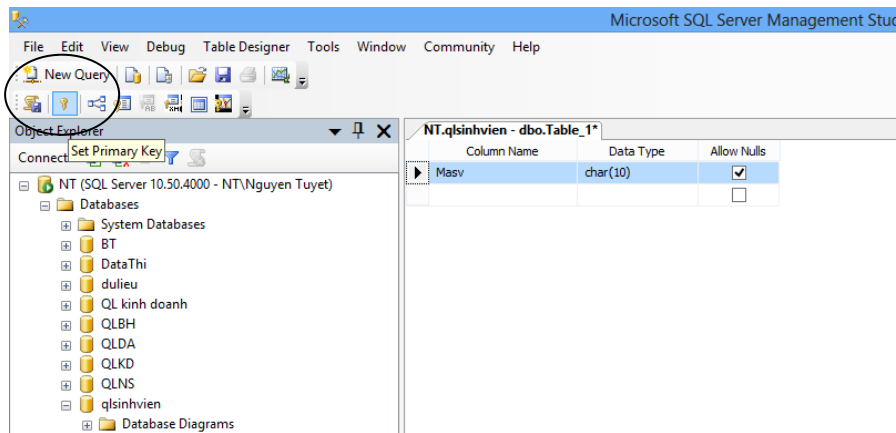
Ví dụ: Xem lại ví dụ 2

```
CREATE TABLE Sinhvien
(
    masv          CHAR(10)
    CONSTRAINT pk_masv PRIMARY KEY,
    ho            NVARCHAR(150) NOT NULL,
    ten           NVARCHAR(8)   NOT NULL,
    gioitinh      BIT           NULL,
    ngaysinh      DATE          NULL,
    dienthoai     CHAR(11)      NULL,
    diachi        NVARCHAR(50)  NULL,
    manganh       CHAR(5)       NOT NULL)

```

Trong đó: *pk\_masv* là tên ràng buộc.

Bạn cũng có thể tạo khóa chính bằng cách nhấn chuột vào biểu tượng chiếc chìa khóa trên thanh công cụ như hình bên dưới:



**Hình 3.6. Biểu tượng khóa chính trên màn hình SQL Server**

### 3.4.2. Tạo ràng buộc FOREIGN KEY

Để khai báo một ràng buộc FOREIGN KEY, ta sử dụng cú pháp:

```
[CONSTRAINT tên_ràng_buộc] FOREIGN KEY [(danh_sách_cột)]
REFERENCES tên_bảng_tham_chiếu(danh_sách_cột_tham_chiếu)
[ON DELETE CASCADE / NO ACTION / SET NULL / SET DEFAULT]
[ON UPDATE CASCADE / NO ACTION / SET NULL / SET DEFAULT]
```

Trong đó:

- *tên\_ràng\_buộc*: do người dùng tự đặt (theo quy tắc định danh).
- *ON DELETE CASCADE/ON UPDATE CASCADE*: Tự động xóa/cập nhật nếu bản ghi được tham chiếu bị xóa/cập nhật.
- *NO ACTION*: (Mặc định) Nếu bản ghi trong bảng tham chiếu đang được tham chiếu bởi một bản ghi bất kỳ trong bảng được định nghĩa thì bản ghi đó không được phép xóa hoặc cập nhật (đối với cột được tham chiếu).
- *SET NULL*: Cập nhật lại khóa ngoài của bản ghi thành giá trị NULL (nếu cột cho phép nhận giá trị NULL).
- *SET DEFAULT*: Cập nhật lại khóa ngoài của bản ghi nhận giá trị mặc định (nếu cột có qui định giá trị mặc định).

Ví dụ: Xem lại ví dụ 2 và thêm ràng buộc FOREIGN KEY

```
CREATE TABLE Sinhvien
(
    masv          CHAR(10)
    CONSTRAINT pk_masv PRIMARY KEY,
```

```

ho            NVARCHAR(150) NOT NULL,
ten           NVARCHAR(8)   NOT NULL,
gioitinh      BIT           NULL,
ngaysinh      DATE          NULL,
dienthoai     CHAR(11)      NULL,
diachi        NVARCHAR (50) NULL,
manganh       CHAR(5)       NOT NULL

```

```

CONSTRAINT fk_manganh FOREIGN KEY(manganh)
REFERENCES nganh(manganh)
ON DELETE CASCADE ON UPDATE CASCADE)

```

Trong đó: `fk_manganh` là tên ràng buộc.

Bảng tham chiếu phải được định nghĩa trước. Do đó, nếu các bảng có mối quan hệ vòng, ta có thể không thể định nghĩa ràng buộc `FOREIGN KEY` ngay trong câu lệnh `CREATE TABLE` mà phải định nghĩa thông qua lệnh `ALTER TABLE`.

### 3.4.3. Tạo ràng buộc *CHECK*

Để khai báo một ràng buộc `CHECK`, ta sử dụng cú pháp:

*[CONSTRAINT tên\_ràng\_buộc] CHECK (điều\_kiện)*

Trong đó, *điều\_kiện* là một biểu thức logic tác động lên cột nhằm qui định giá trị hoặc khuôn dạng dữ liệu được cho phép nhập vào cột đó.

Ví dụ: xem lại ví dụ 2 và bổ sung ràng buộc `CHECK`

```

CREATE TABLE Sinhvien
(masv          CHAR(10)
                CONSTRAINT pk_masv PRIMARY KEY,
ho            NVARCHAR(150) NOT NULL,
ten           NVARCHAR(8)   NOT NULL,
gioitinh      BIT           NULL,
ngaysinh      DATE

```

```

CONSTRAINT chk_ngaysinh
CHECK (YEAR(GETDATE())-YEAR(ngaysinh)>=18),
dienthoai CHAR(11) NULL,
diachi NVARCHAR (50) NULL,
manganh CHAR(5) NOT NULL
CONSTRAINT fk_manganh FOREIGN KEY (manganh)
REFERENCES nganh (manganh)
ON DELETE CASCADE ON UPDATE CASCADE)

```

Trong đó:

- `chk_ngaysinh` là tên ràng buộc
- `GETDATE()`: hàm cho kết quả là ngày tháng năm hiện tại của hệ thống
- `YEAR(kiểu ngày)`: hàm cho kết quả là năm từ kiểu ngày.

#### 3.4.4. Tạo ràng buộc **UNIQUE**

Để khai báo một ràng buộc **UNIQUE** ta sử dụng cú pháp:

*[CONSTRAINT tên\_ràng\_buộc] UNIQUE [(danh\_sách\_cột)]*

Ví dụ: Xem lại ví dụ 2, ràng buộc dữ liệu cột *dienthoai* không được phép trùng dữ liệu.

```

CREATE TABLE Sinhvien
(masv CHAR(10)
CONSTRAINT pk_masv PRIMARY KEY,
ho NVARCHAR(150) NOT NULL,
ten NVARCHAR(8) NOT NULL,
gioitinh BIT NULL,
ngaysinh DATE
CONSTRAINT chk_ngaysinh
CHECK (YEAR(GETDATE())-YEAR(ngaysinh)>22),
dienthoai CHAR(11) NULL
CONSTRAINT uk_dienthoai UNIQUE(dienthoai),

```

```

diachi          NVARCHAR (50) NULL,
manganh         CHAR(5)          NOT NULL
                CONSTRAINT fk_manganh FOREIGN KEY (manganh)
                REFERENCES nganh (manganh)
                ON DELETE CASCADE ON UPDATE CASCADE)

```

Trong đó: uk\_dienthoai là tên ràng buộc.

### 3.5. Sửa, xóa cấu trúc bảng

#### 3.5.1. Sửa cấu trúc bảng

Khi ta cần thay đổi một số thuộc tính cũng như cấu trúc của bảng, tuyệt đối không nên xóa bảng đó và tạo lại cấu trúc mới bởi vì làm như vậy sẽ mất đi tất cả các dữ liệu trong bảng. Thay vào đó, hãy sử dụng câu lệnh **ALTER TABLE** để thực hiện việc thay đổi cấu trúc bảng.

Câu lệnh sửa cấu trúc bảng dạng tổng quát:

*ALTER TABLE tên\_bảng*

*ADD định\_nghĩa\_cột /*

*ALTER COLUMN tên\_cột kiểu\_dữ\_liệu [NULL / NOT NULL] / DROP COLUMN tên\_cột /*

*ADD CONSTRAINT tên\_ràng\_buộc định\_nghĩa\_ràng\_buộc / DROP CONSTRAINT tên\_ràng\_buộc / NOCHECK CONSTRAINT tên\_ràng\_buộc*

Các công việc thường dùng trong sửa cấu trúc bảng:

#### - Thêm một cột

*ALTER TABLE tên\_bảng*

*ADD định\_nghĩa\_cột*

#### - Sửa một cột

*ALTER TABLE tên\_bảng*

*ALTER COLUMN tên\_cột kiểu\_dữ\_liệu [NULL / NOT NULL]*

#### - Xóa một cột

*ALTER TABLE tên\_bảng*

*DROP COLUMN tên\_cột*

**- Thêm ràng buộc Check**

*ALTER TABLE tên\_bảng*

*[CONSTRAINT tên\_ràng\_buộc] CHECK (điều\_kiện)*

**- Thêm ràng buộc Khóa chính**

*ALTER TABLE tên\_bảng*

*[CONSTRAINT tên\_ràng\_buộc] PRIMARY KEY [(danh\_sách\_cột)]*

**- Thêm ràng buộc Unique**

*ALTER TABLE tên\_bảng*

*[CONSTRAINT tên\_ràng\_buộc] UNIQUE [(danh\_sách\_cột)]*

**- Thêm ràng buộc Default**

*ALTER TABLE tên\_bảng*

*[CONSTRAINT tên\_ràng\_buộc] DEFAULT (giá\_trị) FOR tên\_cột*

**- Thêm ràng buộc FOREIGN KEY**

*ALTER TABLE tên\_bảng*

*[CONSTRAINT tên\_ràng\_buộc] FOREIGN KEY [(danh\_sách\_cột)]*

*REFERENCES tên\_bảng\_tham\_chiếu (danh\_sách\_cột\_tham\_chiếu)*

*[ON DELETE CASCADE / NO ACTION / SET NULL / SET DEFAULT]*

*[ON UPDATE CASCADE / NO ACTION / SET NULL / SET DEFAULT]*

**- Vô hiệu hóa ràng buộc**

*ALTER TABLE tên\_bảng*

*NOCHECK CONSTRAINT tên\_ràng\_buộc*

Ví dụ: Thêm cột *email* vào bảng *Sinhvien*

*ALTER TABLE Sinhvien*

*ADD email VARCHAR(20)*

Ví dụ: Xóa bỏ ràng buộc CHECK trên cột *ngaysinh* khỏi bảng *Sinhvien*:

*ALTER TABLE Sinhvien*

*DROP CONSTRAINT ck\_ngaysinh*

Ví dụ: Định nghĩa lại kiểu dữ liệu của cột *Ho* (Nvarchar(50)) trong bảng *Sinhvien* và cho phép cột này chấp nhận giá trị NULL:

```
ALTER TABLE Sinhvien  
  
ALTER COLUMN ho NVARCHAR(50) NULL
```

### 3.5.2. Xóa bảng

Lệnh **DROP TABLE** trong SQL được sử dụng để xóa một bảng và tất cả dữ liệu, chỉ mục, trigger, ràng buộc và quyền được trao cho bảng đó. Bảng bị xóa thì tất cả thông tin có sẵn trong bảng đó cũng sẽ bị xóa vĩnh viễn.

Cú pháp câu lệnh xóa bảng:

**DROP TABLE *tên\_bảng***

Nếu bảng có định nghĩa khóa chính và khóa chính này được một khóa ngoài của bảng khác (trong một mối quan hệ) tham chiếu thì phải xóa ràng buộc này trước khi xóa bảng đó.

Ví dụ: Xóa bảng *Nganh*

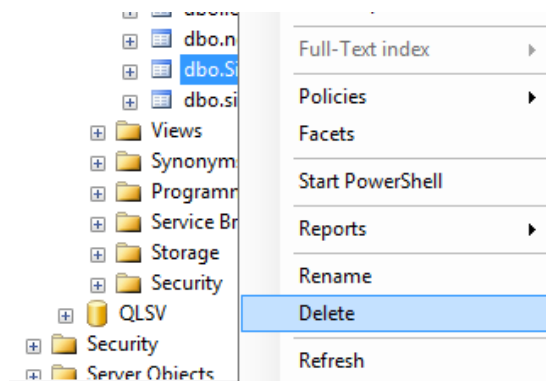
- Vì *manganh* của bảng *Sinhvien* là khóa ngoài tham chiếu đến *manganh* của bảng *Nganh*, nên phải xóa ràng buộc này trước rồi mới xóa được bảng *Nganh* (xem lại ví dụ 1 và 2).

```
ALTER TABLE Sinhvien  
  
DROP CONSTRAINT fk_manganh
```

- Xóa bảng *Nganh*

```
DROP TABLE Nganh
```

Ngoài ra, chúng ta có thể thực hiện đơn giản hơn bằng cách nhấn chuột phải vào bảng cần xóa, chọn Delete:



**Hình 3.7. Thao tác xóa bảng bằng lệnh có sẵn của SQL Server**



### 3.6. Nhập dữ liệu vào bảng

Lệnh **INSERT INTO** trong SQL Server được sử dụng để thêm các hàng dữ liệu mới vào một bảng trong Database.

Cú pháp câu lệnh thêm dữ liệu mới vào bảng:

*INSERT INTO tên\_bảng [(danh\_sách\_cột)]*

*VALUES (danh\_sách\_giá\_trị)*

- Nếu nhập *danh\_sách\_giá\_trị* đầy đủ và theo thứ tự cột trong bảng thì không cần ghi *danh\_sách\_cột* sau *tên\_bảng*.
- Nếu cột nhận dữ liệu có trong bảng mã Unicode (đã khai báo khi định nghĩa bảng) thì trước nhập giá trị vào cột đó bạn phải thêm ký tự N.

Ví dụ: Nhập dữ liệu bảng *Khoa* (có cấu trúc như hình bên)

NT.qlsinhvien - dbo.khoa		
Column Name	Data Type	Allow Nulls
makhoa	char(3)	<input type="checkbox"/>
tenkhoa	nvarchar(150)	<input type="checkbox"/>
		<input type="checkbox"/>

```
INSERT INTO Khoa VALUES ('101', N'Toán')
```

```
INSERT INTO Khoa VALUES ('103', N'Hóa')
```

Ví dụ: Nhập dữ liệu bảng *Nganh* (có cấu trúc như hình bên)

NT.qlsinhvien - dbo.nganh		
Column Name	Data Type	Allow Nulls
manganh	char(5)	<input type="checkbox"/>
tennganh	nvarchar(150)	<input type="checkbox"/>
mota	char(7)	<input type="checkbox"/>
makhoa	char(3)	<input type="checkbox"/>
		<input type="checkbox"/>

```
INSERT INTO Nganh
```

```
VALUES ('10101',N'Sư phạm Toán học','101-K33','101')
```

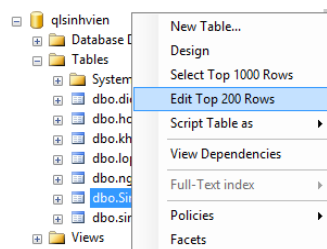
```
INSERT INTO Nganh
```

```
VALUES ('10102',N'Toán học','104-K33','101')
```

```
INSERT INTO Nganh
```

```
VALUES ('10103',N'Sư phạm Toán học','101-K34','101')
```

Ngoài ra có thể dùng cách khác để nhập dữ liệu cho bảng: nhấn chuột phải vào tên bảng, chọn Edit Top 200 Rows:



**Hình 3.8. Thao tác nhập dữ liệu cho bảng bằng chức năng có sẵn của SQL Server**

**Đưa dữ liệu vào bảng mà dữ liệu đó được lấy từ dữ liệu của các bảng khác:**

```
INSERT INTO tên_bảng[(danh_sách_cột)]  
SELECT [danh_sách_cột], danh_sách_giá_trị  
FROM tên_bảng  
WHERE điều_kiện
```

Các giá trị dữ liệu được đưa vào bảng không được chỉ định tường minh mà thay vào đó là một câu lệnh SELECT, truy vấn dữ liệu từ bảng khác (nội dung câu lệnh truy vấn SELECT được trình bày ở chương 4)

Ví dụ: Giả sử bảng *Sinhvien\_hue* đã được định nghĩa gồm các trường: *ho* (họ), *ten* (tên), *ngaysinh* (ngày sinh). Hãy nhập dữ liệu vào bảng *Sinhvien\_hue* dữ liệu lấy từ bảng *Sinhvien*, gồm những sinh viên có địa chỉ ở Huế

```
INSERT INTO Sinhvien_hue  
SELECT ho,ten,ngaysinh  
FROM Sinhvien  
WHERE diachi like '%Huế%'
```

Kết quả của câu lệnh SELECT phải có số cột bằng với số cột có trong bảng nhập và phải tương thích về kiểu dữ liệu đã định nghĩa trong bảng.

### 3.7. Cập nhật dữ liệu

Câu lệnh UPDATE được sử dụng để cập nhật/sửa đổi dữ liệu đã có trong bảng. Giá trị được cập nhật trên cột nào đều phải thỏa mãn các ràng buộc đã được thiết lập trên cột đó.

Cú pháp câu lệnh cập nhật dữ liệu:

```
UPDATE tên_bảng  
SET tên_cột = giá_trị/biểu_thức [..., tên_cột = biểu_thức]  
[FROM danh_sách_bảng]  
[WHERE điều_kiện]
```

Nếu trong *biểu\_thức* hoặc *điều\_kiện* cập nhật có sử dụng các trường ở các bảng khác *tên\_bảng* cập nhật thì phải ghi *danh\_sách\_bảng* có liên quan ở mệnh đề *FROM*.

Ví dụ: Cập nhật lại số tín chỉ (*sotc*) của các môn học có số tín chỉ nhỏ hơn 2 thành 2

```

UPDATE Hocphan
SET sotc = 2
WHERE sotc < 2

```

Ví dụ: Cập nhật lại số tín chỉ của các môn học có số tín chỉ lớn hơn hoặc bằng 5 thành 4, số tín chỉ bằng 4 thành 3, còn lại là 2.

Vì giá trị cập nhật được phân chia thành 3 trường hợp, nên ta phải sử dụng cấu trúc điều khiển CASE WHEN...THEN....

```

UPDATE Hocphan
SET sotc= CASE
                WHEN sotc >=5 THEN 4
                WHEN sotc =4 THEN 3
                ELSE 2
            END

```

Ví dụ: Cập nhật lại số tín chỉ là 2 cho tất cả các học phần của Khoa Giáo dục thể chất – Quốc phòng

```

UPDATE Hocphan
SET sotc = 2
FROM Khoa
WHERE Khoa.makhoa = Hocphan.makhoa
AND tenkhoa=N'Giáo dục thể chất - Quốc phòng'

```

### 3.8. Xóa dữ liệu

SQL Server cung cấp 2 câu lệnh để xóa dữ liệu, DELETE và TRUNCATE. Cú pháp của hai lệnh này như sau:

```

DELETE FROM tên_bảng
[FROM danh_sách_bảng]
[WHERE điều_kiện]

```

hoặc *TRUNCATE TABLE tên\_bảng*

Những điểm khác nhau cơ bản của hai lệnh này:

DELETE cung cấp các lựa chọn để xóa những dòng dữ liệu thỏa mãn các điều kiện nhất định, như WHERE hoặc JOIN với các bảng khác. TRUNCATE không có lựa chọn nào, mà luôn xóa bỏ toàn bộ dữ liệu của bảng. Nói cách khác, ta không thể dùng TRUNCATE để xóa một phần dữ liệu của bảng.

Quá trình thực hiện DELETE: tìm các bản ghi thỏa mãn điều kiện của câu lệnh và xóa các bản ghi này. Việc tìm các bản ghi cần xóa được thực hiện giống hệt như một câu lệnh SELECT, cũng tối ưu hóa, lựa chọn giữa các phương án thực hiện khác nhau và chọn ra phương án tối ưu. TRUNCATE thì chỉ có một phương án thực hiện duy nhất, đó là xóa bỏ tất cả các dòng dữ liệu của bảng.

Với DELETE, các bản ghi bị xóa sẽ được kiểm tra xem có vi phạm ràng buộc FOREIGN KEY không. Nếu trước đó, khi ta định nghĩa ràng buộc FOREIGN KEY mà có lựa chọn CASCADE DELETE, thì thay vì báo lỗi SQL Server sẽ đồng thời xóa hết các bản ghi trong cả bảng quan hệ. TRUNCATE thì không kiểm tra như vậy, nếu bảng có ràng buộc FOREIGN KEY, SQL Server sẽ báo lỗi và không cho thực hiện. Do vậy, lựa chọn CASCADE DELETE trong khai báo FOREIGN KEY chỉ ảnh hưởng đến lệnh DELETE chứ không tác dụng đối với TRUNCATE.

DELETE thực ra chỉ đánh dấu xóa các bản ghi chứ ngay sau đó dữ liệu của các bản ghi bị xóa vẫn nằm còn lưu. Khi ta INSERT thêm dữ liệu vào bảng thì các bản ghi mới sẽ ghi đè lên các vùng lưu trữ đó. Ta có thể kiểm tra để thấy kích thước bảng không thay đổi ngay cả sau khi chạy DELETE. TRUNCATE thì xóa hết dữ liệu đồng thời giải phóng vùng lưu trữ dành cho bảng, trả lại cho SQL Server.

DELETE cho phép áp dụng đối với bảng ở server khác được nối qua linked server. TRUNCATE không cho phép điều này, bạn chỉ có thể TRUNCATE bảng nằm trên cùng server.

Vì vậy, dùng DELETE luôn luôn chậm hơn TRUNCATE. Càng có nhiều bản ghi DELETE càng chậm, còn TRUNCATE thì không phụ thuộc vào lượng dữ liệu. DELETE có phạm vi ứng dụng rộng hơn; còn TRUNCATE thì thực hiện nhanh hơn.

Ví dụ: Câu lệnh dưới đây xóa khỏi bảng *Sinhvien* những sinh viên có địa chỉ ở Huế

```
DELETE FROM Sinhvien  
WHERE diachi LIKE N'%Huế%'
```

Ví dụ: Xóa khỏi bảng *Sinhvien* những sinh viên có địa chỉ ở Huế và có điểm học phần bằng 0

```
DELETE FROM Sinhvien
```

```
FROM Diem

WHERE Sinhvien.masv=Diem.masv AND diachi
LIKE N'%Huế%' and diemhp=0
```

Ví dụ: Xóa khỏi bảng *Nganh* những ngành không có sinh viên nào học

```
DELETE FROM Nganh

WHERE manganh NOT IN (SELECT DISTINCT
manganh FROM Sinhvien)
```

Ví dụ: Xóa toàn bộ dữ liệu trong bảng *Diem*

```
DELETE FROM Diem
```

có tác dụng tương tự với câu lệnh

```
TRUNCATE TABLE Diem
```

### 3.9. Tạo chỉ mục (Index)

#### 3.9.1. Chỉ mục là gì?

Chỉ mục là một phần quan trọng đối với CSDL đặc biệt là cơ sở dữ liệu lớn. Chỉ mục được thiết lập từ một hoặc nhiều cột của bảng dữ liệu, các giá trị của khóa chỉ mục sẽ được sắp xếp và lưu trữ theo một danh sách. Mỗi giá trị trong khóa chỉ mục là duy nhất trong danh sách này và nó liên kết đến giá trị trong bảng dữ liệu (liên kết dạng con trỏ). Việc lưu trữ dữ liệu của bảng có khóa chỉ mục được thực hiện theo cấu trúc cây cân bằng (B-Tree index) nhằm tăng tốc độ truy xuất dữ liệu.

Chỉ mục là một trong những yếu tố quan trọng nhất góp phần vào việc nâng cao hiệu suất của cơ sở dữ liệu. Chỉ mục làm tăng tốc độ của quá trình truy vấn dữ liệu bằng cách cung cấp phương pháp truy xuất nhanh chóng tới các dòng trong các bảng, tương tự như mục lục của một cuốn sách giúp bạn nhanh chóng tìm đến một trang bất kỳ mà bạn muốn trong cuốn sách đó.

Ví dụ, nếu bạn muốn tham chiếu tất cả các trang trong một cuốn sách về một chủ đề nào đó, đầu tiên bạn nghĩ ngay đến mục lục của nó, nơi mà liệt kê tất cả các chương, chủ đề theo thứ tự và sau đó được tham chiếu tới một hoặc nhiều trang cụ thể.

Chỉ mục được tạo ra trên các cột trong bảng hoặc khung nhìn. Chúng cung cấp một phương pháp giúp bạn nhanh chóng tìm kiếm dữ liệu dựa trên các giá trị trong các cột. Ví dụ, nếu bạn tạo ra một chỉ mục trên cột khóa chính và sau đó tìm kiếm một dòng dữ liệu dựa trên một trong các giá trị của cột này, đầu tiên SQL

Server sẽ tìm giá trị này trong chỉ mục, sau đó nó sử dụng chỉ mục để nhanh chóng xác định vị trí của dòng dữ liệu bạn cần tìm. Nếu không có chỉ mục, SQL Server sẽ thực hiện động tác quét qua toàn bộ bảng (table scan) để xác định vị trí dòng cần tìm, mà table scan là một trong những động tác có hại nhất cho hiệu suất của SQL Server.

Chỉ mục có thể tạo trên hầu hết các cột trong bảng hoặc khung nhìn. Ngoại trừ các cột dùng để lưu trữ các đối tượng dữ liệu lớn như kiểu Image, Text, varchar(max)...

### ***3.9.2. Các loại chỉ mục***

Có 2 loại chỉ mục chính:

**Clustered Index:** là một dạng cấu trúc dùng để lưu trữ và sắp xếp dữ liệu vật lý trong bảng hoặc khung nhìn dựa trên các giá trị khóa của chúng. Mặc định, khi bạn tạo khóa chính (Primary key) cho một bảng nào đó, tức bạn đã tạo một Clustered Index. Khi dữ liệu trong bảng hoặc khung nhìn cần được lưu trữ và sắp xếp theo một thứ tự nhất định chính là lúc cần dùng đến Clustered Index.

**Nonclustered Index:** có một cấu trúc tách biệt với bản ghi trong bảng hoặc khung nhìn. Mỗi một chỉ mục loại này chứa các giá trị của các cột khóa trong khai báo của chỉ mục, và mỗi một bản ghi giá trị của khóa trong chỉ mục này chứa một con trỏ tới dòng dữ liệu tương ứng của nó trong bảng.

Ngoài ra, còn có thể mở rộng các kiểu chỉ mục:

**Composite index:** Là kiểu chỉ mục có nhiều hơn một cột. Trong SQL Server 2005 và 2008, có thể tạo đồng thời tối đa 16 cột trong một chỉ mục, với yêu cầu kích thước của chỉ mục không vượt quá giới hạn 900 byte. Cả hai kiểu chỉ mục cơ sở là Clustered Index và Non Clustered Index cũng có thể đồng thời là kiểu Composite index.

**Unique Index:** Là kiểu chỉ mục dùng để đảm bảo tính duy nhất trong các cột được tạo chỉ mục. Nếu chỉ mục loại này được tạo dựa trên nhiều cột, thì tính duy nhất của giá trị được tính trên tất cả các cột đó, không chỉ riêng rẽ từng cột. Ví dụ, nếu bạn đã tạo ra một chỉ mục trên các cột Holot và Ten trong một bảng, thì giá trị của hai cột này kết hợp với nhau phải là duy nhất, nhưng riêng rẽ từng cột thì giá trị vẫn có thể trùng nhau.

### ***3.9.3. Tạo chỉ mục***

*CREATE [UNIQUE] INDEX tên\_chỉ\_mục*

*ON tên\_bảng(tên\_cột, [tên\_cột])*

Trong đó :

Unique index được sử dụng không chỉ để tăng hiệu suất, mà còn cho mục đích toàn vẹn dữ liệu. Một Unique index không cho phép bất kỳ bản sao giá trị nào được chèn vào trong bảng.

Việc tạo một Single-column index hoặc một Composite index tùy thuộc vào sử dụng cột nào thường xuyên trong mệnh đề WHERE của một truy vấn như là các điều kiện lọc. Nếu chỉ có một cột được sử dụng, thì lựa chọn tốt nhất là Single-column index. Nếu có hai hoặc nhiều cột được sử dụng thường xuyên trong mệnh đề WHERE như là các bộ lọc, thì dạng chỉ mục Composite index là lựa chọn tối ưu.

Ví dụ: Tạo chỉ mục có tên id\_masv cho cột masv trên bảng Sinhvien

**CREATE INDEX** id\_masv

**ON** Sinhvien (masv)

#### **3.9.4. Xóa chỉ mục**

Một chỉ mục có thể bị xóa bằng lệnh DROP INDEX trong SQL. Cần cẩn thận trong khi xóa một chỉ mục, bởi vì khi đó hiệu suất có thể chậm hơn hoặc không được cải thiện.

*DROP INDEX tên\_chỉ\_mục*

*ON tên\_bảng*

### **CÂU HỎI ÔN TẬP CHƯƠNG 3**

1. Các kiểu dữ liệu trong SQL Server?
2. Các kiểu ràng buộc trong SQL Server?
3. Sự khác nhau giữa Primary key constraint và Unique constraint là gì?
4. Tìm sự khác nhau giữa hai lệnh xóa dữ liệu DELETE và TRUNCATE?

## CHƯƠNG 4. TRUY VẤN DỮ LIỆU – QUERY

Truy vấn là thao tác truy xuất dữ liệu lưu trữ trong các bảng nhằm phục vụ nhu cầu khai thác thông tin. Nội dung chính của chương này gồm:

- Giới thiệu câu lệnh truy vấn dạng tổng quát;
- Chức năng của từng mệnh đề trong câu lệnh truy vấn;
- Các phép kết nối trong truy vấn;
- Truy vấn con;
- Hợp các kết quả truy vấn.

Kết thúc chương này bạn có thể:

- Thực hiện được các thao tác truy vấn dữ liệu;
- Truy vấn trên nhiều bảng với các loại phép kết nối;
- Sử dụng truy vấn con để truy vấn dữ liệu;

### 4.1. Câu lệnh truy vấn dạng tổng quát

Câu lệnh SELECT được sử dụng để truy xuất dữ liệu từ một hay nhiều bảng và kết quả khi thực hiện câu lệnh cũng được hiển thị dưới dạng bảng. Câu lệnh này có thể dùng để thực hiện phép chọn (truy xuất một tập con các dòng trong một hay nhiều bảng), phép chiếu (truy xuất một tập con các cột trong một hay nhiều bảng) và phép kết nối (liên kết các dòng trong hai hay nhiều bảng để truy xuất dữ liệu). Ngoài ra, câu lệnh này còn cung cấp khả năng thực hiện các thao tác truy vấn và thống kê dữ liệu phức tạp khác.

Cú pháp câu lệnh truy vấn:

```
SELECT [ALL / DISTINCT] / [TOP n] <danhsách_cột> / [biểu_thức]
[AS <tên_cột_mới>]
[INTO <tên_bảng_mới>]
FROM <danhsách_bảng/khung_nhìn>
[WHERE <các_điều_kiện_truy_vấn>]
[GROUP BY <danhsách_cột>]
[HAVING <điều_kiện_dựa_trên_GROUP_BY>]
[ORDER BY <cột_sắp_xếp>]
```



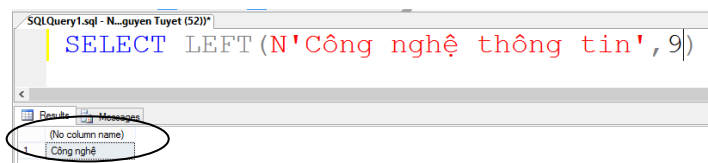
*[COMPUTE các\_hàm\_thống\_kê [BY danh\_sách\_cột]]*

### Một số lưu ý khi viết câu lệnh:

- Câu lệnh SQL không phân biệt chữ hoa, chữ thường.
- Các từ khóa không được viết tắt hoặc tách ra thành nhiều dòng.
- Các mệnh đề khác nhau nên đặt trên những dòng khác nhau.
- Một câu lệnh có thể viết trên nhiều dòng, nhiều câu lệnh có thể viết trên một dòng.
- Trong câu lệnh trên, thành phần đặt trong cặp dấu [ ] là không bắt buộc lúc nào cũng có. Thành phần nào không có thì bỏ đi, còn nếu có thì phải xuất hiện theo trật tự này.
- Thực hiện câu lệnh bằng cách bôi đen rồi nhấn F5 hoặc nhấn Execute trên thanh công cụ và phải chú ý câu lệnh đó đang thực hiện trên Database nào.
- SQL Server sử dụng dấu -- để đánh dấu phần chú thích cho câu lệnh đơn và cặp dấu /\*...\*/ để chú thích cho một nhóm lệnh ghép.
- Trong trường hợp đặc biệt, SELECT không có mệnh đề FROM, dùng để hiển thị thông tin không lấy trong cơ sở dữ liệu.

Ví dụ: Hiển thị 9 ký tự đầu của chữ Công nghệ thông tin:

```
SELECT LEFT(N' Công nghệ thông tin', 9)
```



**Hình 4.1. Kết quả khi thực thi câu lệnh**

#### 4.1.1. Câu lệnh SELECT với mệnh đề FROM

Trường hợp đơn giản nhất câu lệnh SELECT gồm:

```
SELECT danh_sách_cột
```

```
FROM danh_sách_bảng
```

Truy xuất dữ liệu của *danh\_sách\_cột* từ *danh\_sách\_bảng*.

Khi cần hiển thị tất cả các cột của bảng ta dùng dấu \*. Dùng dấu này như một cách viết tắt để liệt kê toàn bộ các cột của bảng nằm trong mệnh đề FROM.

Ví dụ: Muốn lấy toàn bộ thông tin của bảng *Sinhvien*, ta viết:

```
SELECT * FROM Sinhvien
```

Hoặc 

```
SELECT Sinhvien.* FROM Sinhvien
```

Khi cần chọn ra  $n$  dòng dữ liệu đầu tiên ta sử dụng từ khóa *TOP  $n$*  trước *danh\_sách\_cột*. Nếu có thêm từ khóa *PERCENT* đi kèm theo sau thì truy vấn chỉ chọn ra  $n$  phần trăm mẫu tin đầu tiên.

Ví dụ: Hiển thị họ tên và ngày sinh của 5 sinh viên đầu tiên trong danh sách

```
SELECT TOP 5 ho,ten,ngaysinh  
FROM Sinhvien
```

Ví dụ: Hiển thị họ tên và ngày sinh của 10% số lượng sinh viên hiện có trong bảng *Sinhvien*

```
SELECT TOP 10 PERCENT ho,ten,ngaysinh  
FROM Sinhvien
```

Khi truy vấn có thể sẽ xảy ra nhiều mẫu tin trùng lặp trong kết quả. Nếu chỉ cần hiển thị một mẫu tin trong những mẫu tin đó bạn sử dụng *DISTINCT* trong mệnh đề *SELECT*.

Ví dụ:

```
SELECT DISTINCT ho,ten,ngaysinh  
FROM Sinhvien
```

#### **4.1.2. Câu lệnh *SELECT* với mệnh đề *WHERE***

Dùng *WHERE* để tạo điều kiện cần lọc mẫu tin.

```
SELECT danh_sách_cột  
FROM danh_sách_bảng  
WHERE biểu_thức_điều_kiện
```

Các phép toán và ký tự đại diện dùng trong *biểu\_thức\_điều\_kiện*:

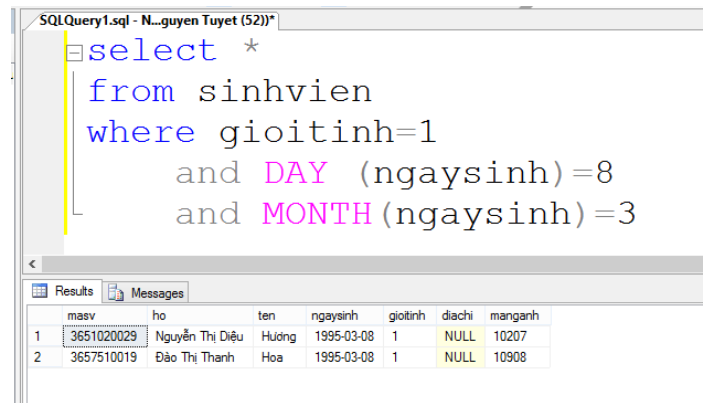
- Các phép toán so sánh:  $>$ ,  $<$ ,  $>=$ ,  $<=$ ,  $=$ ,  $!=$ ,  $<>$ ,  $!>$ ,  $!<$ ;

Ví dụ: Hiển thị những học phần có số tín chỉ lớn hơn 3

```
SELECT * FROM Hocphan  
WHERE sotc>3
```

- Các phép toán logic: *or*, *and*, *not*;

Ví dụ: Xem danh sách sinh viên nữ sinh ngày 8/3



**Hình 4.2. Câu lệnh và kết quả khi thực thi câu lệnh**

- Kiểm tra giới hạn dữ liệu: between...and...

Ví dụ: Xem danh sách điểm cuối kỳ nằm trong khoảng 5 đến 7 điểm

```
SELECT *
FROM diem
WHERE cuoiky between 5 and 7
```

- Làm việc trên tập hợp:

Ví dụ: Liệt kê danh sách sinh viên học 2 ngành 10510 và 11307

```
SELECT *
FROM sinhvien
WHERE manganh in ('10510','11307')
```

- Kiểm tra khuôn dạng dữ liệu: Từ khoá LIKE (NOT LIKE) sử dụng trong câu lệnh SELECT nhằm mô tả khuôn dạng của dữ liệu cần tìm kiếm. Chúng thường được kết hợp với các ký tự đại diện sau đây:

**Bảng 4.1. Các ký tự đại diện dùng trong câu lệnh Select**

Ký tự đại diện	ý nghĩa
%	Chuỗi ký tự bất kỳ gồm không hoặc nhiều ký tự
_	Ký tự đơn bất kỳ
[]	Ký tự đơn bất kỳ trong giới hạn được chỉ định (ví dụ [a-f]) hay một tập (ví dụ [abcdef])
[^]	Ký tự đơn bất kỳ không nằm trong giới hạn được chỉ định ( ví dụ [^a-f] hay một tập (ví dụ [^abcdef])).

Ví dụ: Lọc ra những sinh viên có họ bắt đầu bằng Lê và tên bắt đầu bằng chữ A hoặc chữ B

```
SELECT ho,tên FROM Sinhvien
WHERE ho LIKE N'Lê%' AND tên LIKE N'[AB]%'
```

- Giá trị NULL:

Trong mệnh đề WHERE, để kiểm tra giá trị của một cột có giá trị NULL hay không, ta sử dụng cách viết IS NULL/ IS NOT NULL như sau:

```
SELECT danh_sách_cột
FROM danh_sách_bảng
WHERE tên_cột IS [NOT] NULL
```

Ví dụ: Kiểm tra sinh viên nào chưa có điểm cuối kỳ

```
SELECT * FROM Diem
WHERE cuoiky IS NULL
```

#### **4.1.3. Câu lệnh SELECT với mệnh đề ORDER BY**

Để kết quả truy vấn được sắp xếp theo tiêu chí nào đó ta sử dụng ORDER BY.

```
SELECT danh_sách_cột
FROM danh_sách_bảng
ORDER BY tên_cột ASC / DESC
```

- Số cột sắp xếp tối đa là 16, thứ tự sắp xếp từ trái qua phải. Mặc định là sắp xếp tăng dần, khi cần sắp xếp giảm ta sử dụng từ khóa DESC phía sau tên cột.

- Có thể chỉ định số thứ tự của cột cần sắp xếp trong *danh\_sách\_cột*.

Ví dụ: Hiển thị danh sách những sinh viên tên Bình, sắp xếp theo giới tính và tuổi

```
SELECT ho,tên,gioitinh,
YEAR(GETDATE())-YEAR(ngaysinh) AS tuoi
FROM Sinhvien
WHERE tên=N'Bình'
ORDER BY gioitinh,tuoi
```

Hoặc:

```
SELECT ho,tên,gioitinh,
YEAR(GETDATE())-YEAR(ngaysinh) AS tuoi
FROM Sinhvien
```

```
WHERE ten=N'Bình'

ORDER BY 3,4
```

	ho	ten	gioitinh	tuoi
1	Lý Thanh	Bình	0	22
2	Nguyễn Văn	Bình	0	23
3	Lê Thanh	Bình	0	24
4	Đoàn Thị Thanh	Bình	1	22
5	Võ Đăng	Bình	1	22

**Hình 4.3. Kết quả thực thi câu lệnh của ví dụ trên**

#### **4.1.4. Câu lệnh SELECT với mệnh đề GROUP BY**

Mệnh đề GROUP BY dùng để phân nhóm dữ liệu phục vụ tính toán thống kê.

Ví dụ: Thống kê tổng số sinh viên của từng mã ngành

```
SELECT manganh, count(masv) AS sosv
FROM Sinhvien
GROUP BY manganh
```

Sử dụng AS trong câu lệnh SELECT dùng để đặt bí danh cho các cột. Bí danh được đặt tùy ý, nếu đặt bí danh có sử dụng dấu cách và dấu Tiếng Việt thì phải đặt trong cặp dấu nháy đơn.

Có 3 cách đặt bí danh cho cột trong truy vấn:

```
SELECT bí_danh_1 = cột_1, cột_2 bí_danh_2, cột_3 AS bí_danh_3
FROM tên_bảng
```

Ví dụ: Trong bảng Lóp hãy đặt bí danh cho cột *malop* là *Mã lớp*, cột *tenlop* là *Tên lớp*, cột *khoa* là *Khóa*

```
SELECT 'Mã lớp' = malop, tenlop 'Tên lớp', khoa AS 'Khóa'
FROM Lop
```

#### **4.1.5. Câu lệnh SELECT với mệnh đề INTO**

Câu lệnh SELECT ... INTO có tác dụng tạo một bảng mới có cấu trúc và dữ liệu được xác định từ kết quả của truy vấn. Bảng mới được tạo ra sẽ có số cột bằng số cột được chỉ định trong danh sách chọn và số dòng sẽ là số dòng kết quả của truy vấn.

Ví dụ: Câu lệnh dưới đây truy vấn dữ liệu tạo bảng *Hocbong* bao gồm các cột *ho, ten, diemhp*, dữ liệu lấy từ bảng *Sinhvien*, gồm những sinh viên có điểm học phần tất cả các môn  $\geq 8$

```
SELECT ho,ten,diemhp
INTO Hocbong
FROM Sinhvien, Diem
WHERE Sinhvien.masv=Diem.masv and diemhp>=8
```

Vì dữ liệu liên quan đến truy vấn được lấy từ 2 bảng *Sinhvien*, *Diem* nên mệnh đề FROM phải liệt kê 2 bảng và mệnh đề WHERE phải thêm biểu thức kết nối 2 bảng. (Nội dung chi tiết xin xem mục 4.2. Truy vấn dựa trên nhiều bảng)

#### 4.1.6. Câu lệnh Select với mệnh đề Having

Với mệnh đề SELECT... FROM bên dưới kết hợp mệnh đề HAVING cho phép bạn có thể lọc lại dữ liệu sau khi đã nhóm dữ liệu của các dòng trong một bảng. Khác với mệnh đề WHERE dùng để lọc các dòng dữ liệu hiện đang có trong bảng, mệnh đề HAVING chỉ được phép sử dụng đi kèm theo mệnh đề GROUP BY dùng để lọc lại dữ liệu sau khi đã nhóm. Điều này có nghĩa là mệnh đề HAVING chỉ được dùng kèm với mệnh đề GROUP BY.

Ví dụ: Thống kê những ngành chỉ có dưới 10 sinh viên theo học

```
SELECT Sinhvien.manganh, tennganh, count(masv)
as 'so sv'
FROM Sinhvien,Nganh
WHERE Sinhvien.manganh=Nganh.manganh
GROUP BY Sinhvien.manganh,tennganh
HAVING count(masv)<10
```

#### 4.1.7. Câu lệnh Select với mệnh đề Compute

Với cú pháp SELECT ... FROM bên dưới kết hợp mệnh đề COMPUTE cho phép bạn có thể tạo ra dòng thống kê dữ liệu ở cuối kết quả truy vấn.

Ví dụ:

```
SELECT makhoa, SUM(sotc) AS 'tong so tc'
FROM Hocphan
```

	makhoa	tong so tc
1	101	907
2	102	766
3	103	1102
4	104	903
5	105	593
6	106	684
7	107	684
8	108	849
9	109	680
10	110	741
11	111	571
12	112	595
13	113	523
14	114	656

GROUP BY makhoa

ORDER BY makhoa

COMPUTE COUNT(makhoa), SUM(SUM(sotc))

Khi cần thống kê theo từng nhóm ta dùng COMPUTE...BY, ví dụ:

SELECT makhoa, sotc

FROM Hocphan

ORDER BY makhoa

COMPUTE SUM(sotc) BY makhoa

	makhoa	sotc
1	101	3
2	101	3
3	101	3
4	101	3
5	101	3
6	101	3
7	101	3
8	101	3
sum		
1		907

	makhoa	sotc
1	102	2
2	102	3
3	102	2
4	102	4
5	102	5
6	102	4
7	102	3
8	102	2
sum		
1		766

**Hình 4.4. Kết quả thực thi câu lệnh Compute...by**

Khi cần thay đổi kết quả của truy vấn tùy thuộc vào các trường hợp khác nhau ta sử dụng cấu trúc CASE trong danh sách chọn. Cú pháp như sau:

*CASE*

*WHEN điều\_kiện THEN kết\_quả*

*[ ... ]*

*[ELSE kết\_quả\_của\_else]*

*END*

Ví dụ: Để hiển thị giới tính là Nam hay Nữ trong kết quả truy vấn, ta viết:

SELECT masv, ho, ten,

```

CASE
    WHEN gioitinh=1 THEN 'Nam'
    ELSE N'Nữ'
END AS gioitinh
FROM Sinhvien

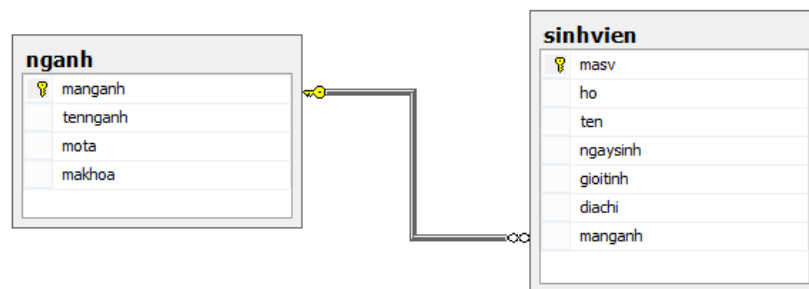
```

#### 4.2. Truy vấn dữ liệu trên nhiều bảng

Khi cần thực hiện truy vấn dữ liệu từ hai hay nhiều bảng ta phải sử dụng phép kết nối.

Câu lệnh sử dụng phép kết nối thực hiện lấy các dòng dữ liệu trong các bảng tham gia truy vấn, so sánh giá trị của các dòng này trên một hoặc nhiều cột được chỉ định trong điều kiện kết nối và kết hợp các dòng thỏa mãn điều kiện thành những dòng kết quả truy vấn.

Ví dụ: Cần lấy danh sách những sinh viên học ngành *Công nghệ thông tin*



**Hình 4.5. Hai bảng trích trong cơ sở dữ liệu**

```

SELECT ho, ten, tennganh
FROM Sinhvien,Nganh
WHERE Sinhvien.manganh = Nganh.manganh AND
    tennganh=N'Công nghệ Thông tin'

```



	manganh	tennganh		mota	makhoa
66	10503	Công nghệ thông tin		C03-K34	105
67	10504	Công nghệ thông tin		105-K34	105
68	10505	Sư phạm Tin học		113-K34	105
69	10506	Công nghệ thông tin		105-K35	105
70	10507	Sư phạm Tin học		113-K35	105
71	10508	Công nghệ thông tin		C03-K35	105
72	10509	Sư phạm Tin học		113-K36	105
73	10510	Công nghệ thông tin		105-K36	105
74	10511	Sư phạm Tin học		113-K37	105
75	10512	Công nghệ thông tin		105-K37	105
76	10513	Công nghệ thông tin		C03-K37	105
77	10514	Công nghệ thông tin		113-K38	105

	ho	ten	tennganh
1	Trịnh Xuân	Cường	Công nghệ thông tin
2	Trương Văn	Cường	Công nghệ thông tin
3	Phạm Xuân	Dữ	Công nghệ thông tin
4	Hồ Đức	Duy	Công nghệ thông tin
5	Đặng Kỳ	Duyên	Công nghệ thông tin
6	Phạm Thị	Duyên	Công nghệ thông tin
7	Quách Đình	Dự	Công nghệ thông tin
8	Đỗ Anh	Đạt	Công nghệ thông tin
9	Hồ Tấn	Đạt	Công nghệ thông tin
10	Nguyễn Thành	Đạt	Công nghệ thông tin
11	Đinh Ngọc	Điệp	Công nghệ thông tin
12	Trần Văn	Đạt	Công nghệ thông tin

	masv	ho	ten	ngaysinh	giotinh	diachi	manganh
1	3651050001	Trịnh Xuân	Cường	1995-06-04	0	NULL	10510
2	3651050002	Trương Văn	Cường	1995-02-02	0	NULL	10510
3	3651050003	Phạm Xuân	Dữ	1995-08-08	0	NULL	10510
4	3651050004	Hồ Đức	Duy	1995-08-17	0	NULL	10510
5	3651050005	Đặng Kỳ	Duyên	1994-09-20	1	NULL	10510
6	3651050006	Phạm Thị	Duyên	1995-06-16	1	NULL	10510
7	3651050007	Quách Đình	Dự	1994-05-06	0	NULL	10510
8	3651050008	Đỗ Anh	Đạt	1994-02-02	0	NULL	10510
9	3651050009	Hồ Tấn	Đạt	1995-01-05	0	NULL	10510
10	3651050010	Nguyễn Thành	Đạt	1995-03-05	0	NULL	10510
11	3651050011	Đinh Ngọc	Điệp	1990-08-03	0	NULL	10510
12	3651050012	Trần Văn	Đạt	1994-10-25	0	NULL	10510

**Hình 4.6. Kết quả của truy vấn nhiều bảng**

Khi truy vấn dữ liệu trên nhiều bảng cần phải xác định các yếu tố sau:

- Những cột nào cần hiển thị trong kết quả truy vấn?
- Những bảng nào có tham gia vào truy vấn?
- Điều kiện để thực hiện phép kết nối giữa các bảng dữ liệu là gì?

Trong các yếu tố kể trên, việc xác định chính xác điều kiện để thực hiện phép kết nối giữa các bảng đóng vai trò quan trọng. Trong đa số các trường hợp, điều kiện của phép kết nối được xác định nhờ vào mối quan hệ giữa các bảng tham gia truy vấn. Thông thường, đó là điều kiện bằng nhau giữa khóa chính và khóa ngoài của hai bảng có mối quan hệ với nhau. Như vậy, để có thể đưa ra một câu lệnh kết nối thực hiện chính xác yêu cầu truy vấn dữ liệu đòi hỏi phải hiểu được mối quan hệ cũng như ý nghĩa của chúng giữa các bảng dữ liệu.

Ngoài ra, khi truy vấn dữ liệu trên nhiều bảng chúng ta cần xác định lại:

- **Danh sách chọn trong phép kết nối**

Việc sử dụng tên các cột trong danh sách chọn có thể là:

- + Tên của một số cột nào đó trong các bảng có tham gia vào truy vấn. Nếu tên cột trong các bảng trùng tên nhau thì tên cột phải được viết dưới dạng:

tên\_bảng.tên\_cột

- + Dấu sao (\*) được sử dụng trong danh sách chọn khi cần hiển thị tất cả các cột của các bảng tham gia truy vấn, ta viết: tên\_bảng.\*

#### - Mệnh đề **FROM** trong phép kết nối

Sau mệnh đề **FROM** của câu lệnh sử dụng phép kết nối là danh sách tên các bảng tham gia vào truy vấn. Nếu ta sử dụng dấu \* trong danh sách chọn thì thứ tự của các bảng liệt kê sau **FROM** sẽ ảnh hưởng đến thứ tự các cột được hiển thị trong kết quả truy vấn.

#### - Mệnh đề **WHERE** trong phép kết nối

Khi hai hay nhiều bảng được kết nối với nhau, ta phải chỉ định điều kiện để thực hiện phép kết nối ngay sau mệnh đề **WHERE**. Điều kiện kết nối được biểu diễn dưới dạng biểu thức logic, so sánh giá trị dữ liệu giữa các cột của các bảng tham gia truy vấn.

Ví dụ: Câu lệnh dưới đây hiển thị danh sách các sinh viên với các thông tin: mã sinh viên, họ và tên, mã lớp, tên lớp và tên khoa

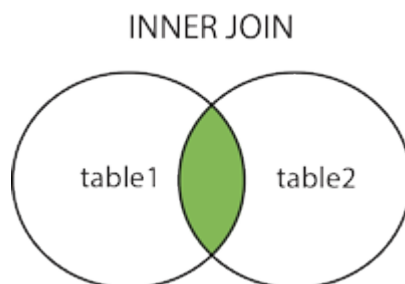
```
SELECT masv,ho,ten,sinhvien.malop,tenlop,tenkhoa
FROM Sinhvien,Lop,Khoa
WHERE Sinhvien.malop = Lop.malop AND
      Lop.makhoa=Khoa.makhoa
```

### 4.3. Các loại phép kết nối trong SQL Server:

- Kết nối trong (Inner Join): gồm Equi Join và Natural Join;
- Kết nối ngoài (Outer Join): gồm Left Outer Join, Right Outer Join và Full Outer Join;
- Kết nối tích hợp (Cross Join);
- Tự kết nối (Self Join).

#### 4.3.1. Phép kết nối trong (Inner Join)

Sử dụng **INNER JOIN** để trả về kết quả là các bản ghi mà cột được kết nối ở hai bảng bằng giá trị nhau, các bản ghi chỉ xuất hiện ở một trong hai bảng sẽ bị loại.



**Hình 4.7. Kết nối INNER JOIN**

Ví dụ: Câu lệnh sau hiển thị thông tin sinh viên và ngành học của sinh viên

```
SELECT Nghanh.manganh, tennganh, Sinhvien.manganh, ten
FROM Nghanh INNER JOIN Sinhvien
ON Nghanh.manganh=Sinhvien.manganh
```

manganh	tennganh	manganh	ten
11205	Giáo dục Thể chất	11205	Văn
11205	Giáo dục Thể chất	11205	Văn
11205	Giáo dục Thể chất	11205	Vi
11205	Giáo dục Thể chất	11205	Việt
11205	Giáo dục Thể chất	11205	Vòng
11205	Giáo dục Thể chất	11205	Ý
11205	Giáo dục Thể chất	11205	Yan
11108	Giáo dục Mầm non	11108	Anh
11108	Giáo dục Mầm non	11108	Bé
11108	Giáo dục Mầm non	11108	Châu
11108	Giáo dục Mầm non	11108	Diễm
11108	Giáo dục Mầm non	11108	Đuẩn

**Hình 4.8. Kết quả thực thi câu lệnh Select sử dụng kết nối Inner join**

SQL Server cung cấp các loại phép kết nối trong sau đây:

- *Phép kết nối bằng* (Equi Join) là một phép kết nối trong đó giá trị của các cột sử dụng để kết nối được so sánh với nhau dựa trên tiêu chuẩn bằng và tất cả các cột trong các bảng tham gia kết nối đều được đưa ra trong kết quả.

Ví dụ: Câu lệnh dưới đây thực hiện phép kết nối bằng giữa hai bảng LOP và KHOA

```
SELECT *
FROM Lop, Khoa
WHERE Lop.makhoa=Khoa.makhoa
```

	malop	tenlop	makhoa	makhoa	tenkhoa
1	1010111011	SP Toán K33	101	101	Toán
2	1010211011	TH Toán K33	101	101	Toán
3	1010311111	SP Toán K36	101	101	Toán
4	1010411111	TH TOÁN - K36	101	101	Toán
5	1020111011	SP Lý K33	102	102	Lý-Kỹ thuật công nghiệp
6	1020311011	TH Lý K33	102	102	Lý-Kỹ thuật công nghiệp
7	1020411111	SP VẬT LÝ - K36	102	102	Lý-Kỹ thuật công nghiệp
8	1020511111	TH LÝ - K36	102	102	Lý-Kỹ thuật công nghiệp
9	1030211011	SP Hóa K33	103	103	Hóa
10	1030311011	TH Hóa K33	103	103	Hóa
11	1030411011	CN Hóa K33	103	103	Hóa

**Hình 4.9. Kết quả thực thi câu lệnh Select sử dụng kết nối bằng**

Trong kết quả của câu lệnh trên, cột *makhoa* (mã khoa) sẽ xuất hiện hai lần trong kết quả phép kết nối (cột *makhoa* của bảng *Khoa* và cột *makhoa* của bảng *Lop*) và như vậy là không cần thiết. Ta có thể loại bỏ bớt đi những cột trùng tên trong kết quả truy vấn bằng cách chỉ định danh sách cột cần được hiển thị trong danh sách chọn của câu lệnh.

```
SELECT malop,ten lop,Lop.makhoa,tenkhoa
FROM Lop,Khoa
WHERE Lop.makhoa=Khoa.makhoa
```

- *Phép kết nối tự nhiên* (Natural Join) là một phép kết nối trong đó điều kiện kết nối giữa hai bảng chính là điều kiện bằng giữa khóa ngoài và khóa chính của hai bảng; và trong danh sách chọn của câu lệnh chỉ giữ lại một trong hai cột tham gia vào điều kiện của phép kết nối.

Ví dụ: Để thực hiện phép kết nối tự nhiên, câu lệnh trong ví dụ trên được viết lại như sau:

```
SELECT Lop.*,tenkhoa
FROM Lop,Khoa
WHERE Lop.makhoa=Khoa.makhoa
```

Trong các câu lệnh kết nối, ngoài điều kiện của phép kết nối được chỉ định trong mệnh đề WHERE còn có thể chỉ định các điều kiện tìm kiếm dữ liệu khác (điều kiện chọn). Thông thường, các điều kiện này được kết hợp với điều kiện kết nối thông qua toán tử AND.

Ví dụ: Câu lệnh dưới đây hiển thị họ tên và ngày sinh của các sinh viên *Khoa Công nghệ Thông tin*

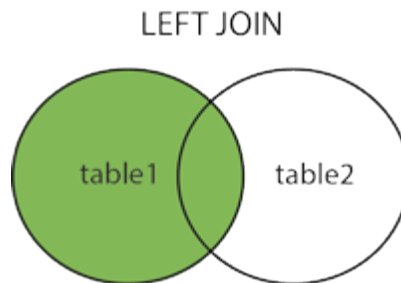
```
SELECT ho,ten,ngaysinh
FROM Sinhvien,Lop,Khoa
WHERE tenkhoa=N'Khoa Công nghệ Thông tin' AND
Sinhvien.malop = Lop.malop AND
Lop.makhoa = Khoa.makhoa
```

#### **4.3.2. Phép kết nối ngoài (Outer Join)**

Trong các phép kết nối đã đề cập ở trên, chỉ những dòng có giá trị trong các cột được chỉ định thỏa mãn điều kiện kết nối mới được hiển thị trong kết quả truy vấn và được gọi là phép kết nối trong. Theo một nghĩa nào đó, những phép kết nối này loại bỏ thông tin chứa trong những dòng không thỏa mãn điều kiện kết nối. Tuy nhiên, đôi khi ta cũng cần giữ lại những thông tin này bằng cách cho phép những dòng không thỏa mãn điều kiện kết nối có mặt trong kết quả của phép kết nối. Để làm điều này, ta có thể sử dụng *phép kết nối ngoài*.

SQL Server cung cấp các loại phép kết nối ngoài sau đây:

- *Phép kết nối ngoài trái* (Left Outer Join, ký hiệu: \*=): Phép kết nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên trái trong điều kiện kết nối cho dù những dòng này không thỏa mãn điều kiện của phép kết nối



**Hình 4.10. Kết nối Left join**

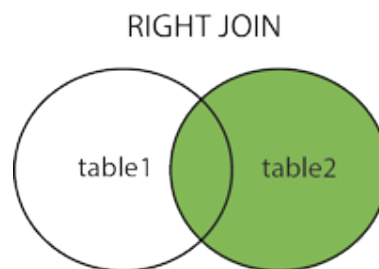
Ví dụ: Hiển thị tất cả các ngành mà không có sinh viên đăng ký học

```
SELECT Ngành.manganh, tennganh, Sinhvien.manganh, ten
FROM Ngành LEFT JOIN Sinhvien
ON Ngành.manganh=Sinhvien.manganh
where sinhvien.manganh is null
```

manganh	tennganh	manganh	ten
11310	Quản lý nhà nước	NULL	NULL
11519	Quản trị kinh doanh	NULL	NULL
10417	Nông học	NULL	NULL
10802	Sư phạm Địa lý	NULL	NULL
11416	Kinh tế	NULL	NULL
11406	Kế toán	NULL	NULL
10304	Công nghệ Hóa học	NULL	NULL
11303	Giáo dục chính trị	NULL	NULL
10201	Sư phạm Vật Lý	NULL	NULL

**Hình 4.11. Kết quả thực hiện câu lệnh sử dụng kết nối Left join**

- *Phép kết nối ngoài phải* (Right Outer Join, ký hiệu: =\*): Phép kết nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên phải trong điều kiện kết nối cho dù những dòng này không thỏa điều kiện của phép kết nối.



**Hình 4.12. Kết nối Right join**

Ví dụ: Trường hợp sau cho thấy sinh viên tên Phú đăng ký ngành học không có trong danh sách các Ngành.

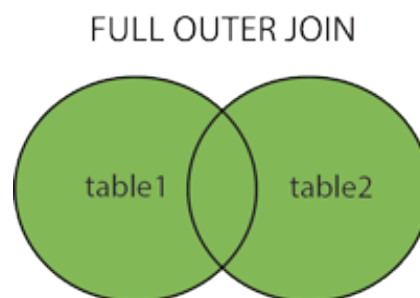
```
SELECT Ngành.manganh, tennganh, Sinhvien.manganh, ten
FROM Ngành RIGHT JOIN Sinhvien
ON Ngành.manganh=Sinhvien.manganh
```

	manganh	tennganh	manganh	ten
1	NULL	NULL	15423	Phú
2	11614	Kỹ thuật điện, điện tử	11614	Phúc
3	11614	Kỹ thuật điện, điện tử	11614	Phúc
4	11614	Kỹ thuật điện, điện tử	11614	Phương
5	11614	Kỹ thuật điện, điện tử	11614	Phước
6	11614	Kỹ thuật điện, điện tử	11614	Quý
7	11614	Kỹ thuật điện, điện tử	11614	Sang
8	11614	Kỹ thuật điện, điện tử	11614	Sĩ
9	11614	Kỹ thuật điện, điện tử	11614	Tài

**Hình 4.13. Kết quả thực hiện câu lệnh sử dụng kết nối Right join**

Nếu trong các cột của các bảng tham gia vào điều kiện của phép kết nối có các giá trị NULL thì các giá trị NULL được xem như là không bằng nhau.

- *Phép kết nối ngoài đầy đủ* (Full Outer Join): Phép kết nối này hiển thị trong kết quả gồm tất cả các bản ghi của cả hai bảng. Với các bản ghi chỉ xuất hiện trong một bảng thì các cột dữ liệu từ bảng kia được điền giá trị NULL.



**Hình 4.14. Kết nối Full join**

Ví dụ: Câu lệnh sau hiển thị những ngành không có sinh viên đăng ký học và ngược lại những sinh viên đăng ký ngành không có trong danh sách ngành.

```
SELECT Ngành.manganh, tennganh, Sinhvien.manganh, ten
FROM Ngành FULL JOIN Sinhvien
ON Ngành.manganh=Sinhvien.manganh
where ngành.manganh is null or sinhvien.Manganh is
null
```

	manganh	tenmanganh	manganh	ten
1	NULL	NULL	15423	Phú
2	10605	Văn học	NULL	NULL
3	11604	Công nghệ kỹ thuật Điện tử, truyền thống	NULL	NULL
4	10502	Sư phạm Tin học	NULL	NULL
5	11106	Giáo dục Mầm non	NULL	NULL
6	10711	Sư phạm Lịch sử	NULL	NULL
7	10608	Văn học	NULL	NULL
8	10817	Quản lý đất đai	NULL	NULL
9	11202	Giáo dục Thể chất - Quốc phòng	NULL	NULL
10	10110	Toán học	NULL	NULL
11	10319	Sư phạm Hóa học	NULL	NULL
12	10704	Lịch sử	NULL	NULL
13	10601	Sư phạm Ngữ văn	NULL	NULL
14	10820	Quản lý đất đai	NULL	NULL
15	10810	Quản lý đất đai	NULL	NULL
16	10103	Sư phạm Toán học	NULL	NULL
17	11414	Kinh tế	NULL	NULL

**Hình 4.15. Kết quả thực hiện câu lệnh sử dụng kết nối Full join**

### 4.3.3. Phép kết nối tích hợp (CROSS JOIN)

Dùng Cross Join để ghép dữ liệu từ hai bảng trong đó số dòng thu được bằng với số dòng của bảng thứ nhất nhân với số dòng của bảng thứ hai.

Ví dụ:

```
SELECT TOP 10 *
FROM Nganh CROSS JOIN Sinhvien
```

Lưu ý : trong câu lệnh này không có từ khóa "ON".

### 4.3.4. Phép tự kết nối (Self Join)

Phép tự kết nối là phép kết nối mà trong đó điều kiện kết nối được chỉ định liên quan đến các cột của cùng một bảng. Trong trường hợp này, sẽ có sự xuất hiện tên của cùng một bảng nhiều lần trong mệnh đề FROM và do đó các bảng cần phải được đặt bí danh.

Ví dụ: Để biết được họ tên và ngày sinh của các sinh viên có cùng ngày sinh với nhau, ta phải thực hiện phép tự kết nối ngay trên chính bảng *Sinhvien*. Trong câu lệnh kết nối, bảng *Sinhvien* xuất hiện trong mệnh đề FROM với bí danh là *a* và *b*. Câu lệnh được viết như sau:

```
SELECT b.ho,b.ten,b.ngaysinh
FROM Sinhvien a, Sinhvien b
WHERE a.ngaysinh=b.ngaysinh AND a.masv<>b.masv
```

## 4.4. Truy vấn con

Truy vấn con là một câu lệnh truy vấn được lồng vào các câu lệnh truy vấn khác nhằm thực hiện các truy vấn tính toán phức tạp. Truy vấn con được đặt trong mệnh đề WHERE, HAVING, FROM, ORDER BY hoặc SELECT.

Ví dụ: Xem danh sách sinh viên có *diemhp* cao nhất

```
SELECT ho, ten, diemhp
FROM Sinhvien, Diem
WHERE Sinhvien.masv=Diem.masv
AND diemhp=(SELECT max(diemhp) FROM Diem)
```

Như vậy truy vấn con tìm *diemhp* cao nhất sẽ thay thế cho một giá trị trong mệnh đề WHERE của câu lệnh SELECT.

Khi sử dụng đến truy vấn con, cần chú ý đến một vài yếu tố sau:

- Cần mở và đóng ngoặc đơn cho câu lệnh truy vấn con.
- Chỉ được phép tham chiếu đến tên một cột hoặc một biểu thức sẽ trả về giá trị trong truy vấn con.
- Kết quả của truy vấn con có thể trả về là một giá trị đơn lẻ hoặc một danh sách (một cột) các giá trị.
- Cấp độ lồng nhau của các truy vấn con bên trong Microsoft SQL Server là 32 mức.

Ví dụ: Liệt kê những ngành không có sinh viên nào đăng ký học.

```
SELECT *
FROM ngành
WHERE manganh NOT IN (select distinct manganh from
sinhvien)
```

Các toán tử thường dùng khi kết hợp với truy vấn con:

- Truy vấn con trả về duy nhất một dòng: Sử dụng các toán tử so sánh một dòng như: =, >, >=, <, <=, <>
- Truy vấn con trả về nhiều dòng: Sử dụng các toán tử so sánh nhiều dòng như:
  - + IN: bằng một trong các giá trị trả về của truy vấn con.
  - + NOT IN: không bằng giá trị nào trong các giá trị trả về của truy vấn con.
  - + ANY: chỉ cần thỏa một trong các giá trị trả về của truy vấn con.



- + ALL: phải thỏa tất cả các giá trị trả về của truy vấn con.
- + Hàm EXISTS kiểm tra tồn tại kết quả trả về từ câu truy vấn con. Nếu tìm thấy một dòng trong truy vấn con thì hàm trả về kết quả TRUE và kết thúc tìm kiếm trong truy vấn con. Nếu chưa tìm thấy dòng nào thì hàm trả về FALSE, tiếp tục tìm kiếm trong truy vấn con. Những câu truy vấn có ANY hay IN đều có thể chuyển thành câu truy vấn có EXISTS

Ví dụ: Chọn những sinh viên có tất cả các môn thi đều có điểm cuối kỳ từ 5 trở lên (nghĩa là không tồn tại một môn nào điểm cuối kỳ dưới 5)

```
SELECT DISTINCT masv
FROM diem d1
WHERE NOT EXISTS
    (SELECT *
     FROM diem d2
     WHERE d2.masv = d1.masv AND d2.cuoiky < 5)
```

#### 4.5. Tối ưu hóa truy vấn

Hiệu năng ứng dụng phụ thuộc vào rất nhiều yếu tố, trong đó có một yếu tố rất quan trọng đó là thời gian để máy chủ xử lý câu lệnh T-SQL. Đôi khi thiết kế cơ sở dữ liệu và các yêu cầu truy vấn phức tạp làm cản trở tốc độ thực thi của các câu lệnh T-SQL.

Cách viết code từng câu lệnh T-SQL cũng có thể khiến máy chủ SQL phải làm việc nhiều hơn để xử lý truy vấn. Vậy làm thế nào để viết các câu truy vấn tối ưu, tận dụng tốt tài nguyên máy chủ SQL.

##### a. Ghi rõ tên cột trong câu lệnh SELECT

Việc sử dụng dấu sao (\*) sau SELECT trong câu lệnh truy vấn để trả về tất cả các cột của các bảng khai báo trong mệnh đề FROM sẽ làm máy chủ SQL quét toàn bộ bảng dữ liệu, trả về dữ liệu trùng lặp, tiêu tốn nhiều tài nguyên.

```
SELECT *
FROM Sinhvien
```

Thay vì sử dụng dấu (\*), bạn nên khai báo rõ ràng tên các cột trong câu lệnh SELECT. Câu lệnh trên được viết lại như sau:

```
SELECT masv, ho, ten, diachi, manganh  
FROM Sinhvien
```

*b. Ghi rõ tên cột trong câu lệnh INSERT*

Tương tự như vậy, bạn cũng nên chỉ rõ tên từng cột bạn muốn chèn dữ liệu vào trong câu lệnh INSERT.

```
INSERT INTO Sinhvien(masv, ho, ten)  
VALUES ('3656110003', N 'Trần Minh', N 'Cảnh')
```

Bằng cách viết trên, khi thêm mới một cột vào bảng sinhvien, câu lệnh INSERT vẫn tiếp tục làm việc với điều kiện cột đó được tạo với giá trị mặc định DEFAULT hoặc cho phép NULL.

*c. Thêm tiền tố điều kiện cho ký tự đại diện để tăng tốc tìm kiếm*

Sử dụng các ký tự đại diện thích hợp có thể cải thiện hiệu suất câu truy vấn.

Ví dụ: tìm kiếm tất cả họ tên của những sinh viên có lót chữ 'Mai'.

```
SELECT DISTINCT ho, ten  
FROM Sinhvien  
WHERE ho LIKE N'%Mai%'
```

Câu lệnh sử dụng ký tự phần trăm (%) để thay thế cho một dãy ký tự bất kỳ trong chuỗi có chứa chữ 'Mai' của cột *ho*. Điều này khiến máy chủ SQL thực hiện thao tác quét chỉ mục nhằm tìm kiếm tất cả các họ có chứa chữ 'Mai' để giải quyết câu truy vấn. Kết quả có thể trả về nhanh hơn nếu nó không phải đọc toàn bộ chỉ mục, bằng cách đặt thêm tiền tố điều kiện trước ký tự thay thế (%).

Ví dụ: tìm kiếm tất cả các sinh viên có họ bắt đầu bằng 'Nguyễn' và có chứa chữ 'Mai'

```
SELECT DISTINCT ho, ten  
FROM Sinhvien  
WHERE ho LIKE N'Nguyễn%Mai%'
```

Bằng cách đặt chữ 'Nguyễn' phía trước dấu phần trăm (%) trong câu lệnh tìm kiếm, bạn đã cho máy chủ SQL biết rằng nó có thể sử dụng một thao tác tìm kiếm chỉ mục để giải quyết câu truy vấn. Một khi máy chủ SQL đọc tới bản ghi cuối cùng có họ bắt đầu bằng 'Nguyễn', nó biết rằng không còn bản ghi nào có họ bắt đầu bằng 'Nguyễn' nữa và sẽ dừng lại.

Ví dụ: tìm kiếm tất cả các bản ghi có họ bắt đầu bằng một ký tự bất kỳ trong khoảng từ 'A' đến 'M' và có chứa chữ 'Mai'.

```
SELECT DISTINCT ho,ten
FROM Sinhvien
WHERE ho LIKE N'[A-M]%Mai%'
```

*d. Chỉ dùng DISTINCT khi cần*

Đặt từ khóa DISTINCT trong câu lệnh SELECT sẽ loại bỏ các kết quả trùng lặp trong số những kết quả trả về của câu truy vấn. Nó khiến máy chủ SQL phải thực hiện thêm thao tác SORT để sắp xếp dữ liệu nhằm nhận biết và loại bỏ các bản ghi trùng lặp. Vì thế, nếu bạn biết trước các kết quả trả về sẽ không trùng lặp thì không nên dùng từ khóa DISTINCT trong câu lệnh T-SQL.

*e. Chỉ dùng UNION khi cần*

Cũng giống như trường hợp từ khóa DISTINCT, toán tử UNION đòi hỏi thêm thao tác SORT để máy chủ SQL có thể loại bỏ những kết quả trùng lặp. Khi cần dùng toán tử UNION để kết nối hai tập hợp bản ghi với nhau, trong đó các bản ghi là độc nhất không trùng lặp, tốt hơn bạn nên dùng toán tử UNION ALL.

#### **4.6. Hợp các câu lệnh truy vấn**

Phép hợp được sử dụng trong trường hợp ta cần gộp kết quả của hai hay nhiều truy vấn thành một tập kết quả duy nhất. SQL cung cấp toán tử UNION để thực hiện phép hợp.

**4.6.1. Cú pháp câu lệnh hợp**

```
Câu_lệnh_select_1
UNION [ALL] Câu_lệnh_select_2
[UNION [ALL] Câu_lệnh__select_3]
...
[UNION [ALL] Câu_lệnh_select_n]
[ORDER BY cột_sắp_xếp]
[COMPUTE danh_sách_hàm_gộp [BY danh_sách_cột]]
```

Việc kết hợp dữ liệu của hai truy vấn SELECT ... FROM bằng mệnh đề UNION cho phép bạn có thể tạo ra một tập hợp các mẫu tin từ các mẫu tin có trong

câu lệnh SELECT ... FROM thứ nhất và các mẫu tin có trong câu lệnh SELECT ... FROM thứ hai. Khác với việc liên kết dữ liệu bằng mệnh đề JOIN, mệnh đề UNION thực ra chỉ thực hiện việc thêm vào các dòng dữ liệu trong câu lệnh SELECT ... FROM thứ nhất vào cuối các dòng dữ liệu trong câu lệnh SELECT ... FROM thứ hai.

Ví dụ: Hiển thị những sinh viên có địa chỉ Quy Nhơn và sinh sau năm 1986.

```
SELECT masv, ho, ten
FROM Sinhvien
WHERE diachi=N'Quy Nhơn'
UNION
SELECT masv, ho, ten
FROM Sinhvien
WHERE YEAR(ngaysinh)>1986
```

Trong ví dụ trên, câu lệnh SELECT thứ nhất chọn ra những *sinh viên có địa chỉ Quy Nhơn*. Câu lệnh SELECT thứ hai chọn ra những *sinh viên sinh sau năm 1986*. Hợp kết quả của hai câu lệnh này ta được kết quả là những *sinh viên có địa chỉ Quy Nhơn hoặc sinh viên sinh sau năm 1986*.

Mặc định kết quả sẽ bỏ đi những dòng dữ liệu giống nhau, nếu muốn giữ lại những dòng dữ liệu giống nhau ta dùng UNION ALL.

#### **4.6.2. Các nguyên tắc khi sử dụng UNION**

- Danh sách cột trong các truy vấn thành phần phải có cùng số lượng.
- Các cột tương ứng trong tất cả các bảng, hoặc tập con bất kỳ các cột được sử dụng trong mỗi truy vấn thành phần phải cùng kiểu dữ liệu.
- Các cột tương ứng trong từng truy vấn thành phần của một câu lệnh UNION phải xuất hiện theo thứ tự như nhau.
- Khi các kiểu dữ liệu khác nhau được kết hợp với nhau trong câu lệnh UNION, chúng sẽ được chuyển sang kiểu dữ liệu tương thích (nếu có thể chuyển đổi kiểu dữ liệu được).
- Tiêu đề cột trong kết quả của phép hợp sẽ là tiêu đề cột được chỉ định trong truy vấn đầu tiên.

- Truy vấn thành phần đầu tiên có thể có INTO để tạo mới một bảng từ kết quả của chính phép hợp.
- Mệnh đề ORDER BY và COMPUTE dùng để sắp xếp kết quả truy vấn hoặc tính toán các giá trị thống kê chỉ được sử dụng ở cuối câu lệnh UNION. Chúng không được sử dụng trong bất kỳ truy vấn thành phần nào.
- Mệnh đề GROUP BY và HAVING chỉ có thể được sử dụng trong bản thân từng truy vấn thành phần. Chúng không được phép sử dụng để tác động lên kết quả chung của phép hợp.
- Phép toán UNION có thể được sử dụng bên trong câu lệnh INSERT.
- Phép toán UNION không được sử dụng trong câu lệnh CREATE VIEW.

### **CÂU HỎI ÔN TẬP CHƯƠNG 4**

1. Các loại kết nối trong truy vấn?
2. Khi nào thì truy vấn sử dụng mệnh đề Group by?
3. Truy vấn con có đặc điểm gì?
4. Tại sao phải tối ưu hóa truy vấn?
5. Hợp các kết quả truy vấn cần tuân thủ những nguyên tắc nào?

## CHƯƠNG 5. KHUNG NHÌN - VIEW

Ở chương trước, chúng ta biết truy vấn có thể phức tạp, đặc biệt khi chúng sử dụng nhiều phép nối hoặc truy vấn con. Vì vậy, đôi khi cần lưu lại những truy vấn đó để sử dụng thường xuyên. Việc đó được thực hiện bằng cách lưu trữ câu lệnh vào file hoặc tạo khung nhìn. Khác với bảng, khung nhìn được lưu trữ như một phần của cơ sở dữ liệu. Khung nhìn có thể được sử dụng không chỉ bởi các lập trình viên mà còn bởi cả người dùng và các chương trình ứng dụng truy cập vào cơ sở dữ liệu. Điều này đem lại một số lợi ích so với việc sử dụng bảng một cách trực tiếp.

Để hiểu hơn về khung nhìn, trong chương này giới thiệu:

- Khung nhìn là gì?
- Tại sao phải sử dụng khung nhìn?
- Tạo, sửa, xóa và sử dụng khung nhìn như thế nào?

### 5.1. Khái niệm

Khung nhìn trong SQL Server không chứa bất kỳ dữ liệu nào; nó được lưu trữ dưới dạng lệnh SELECT trong cơ sở dữ liệu. Vì kết quả của câu lệnh SELECT được thể hiện dưới dạng một bảng, nên ta có thể xem khung nhìn như là một bảng ảo, có nội dung được định nghĩa thông qua một câu lệnh SELECT.

Thông qua khung nhìn chúng ta có thể xem được một phần dữ liệu từ một hoặc nhiều bảng. Khung nhìn được sử dụng để hạn chế truy cập cơ sở dữ liệu hoặc để ẩn dữ liệu phức tạp.

Một khung nhìn có thể có tối đa 1024 cột. Điểm khác biệt giữa khung nhìn và bảng là khung nhìn không được xem là một cấu trúc lưu trữ dữ liệu tồn tại trong cơ sở dữ liệu. *Thực chất dữ liệu trong khung nhìn được lấy từ các bảng thông qua câu lệnh SELECT.*

Có ba loại khung nhìn cơ bản:

- Standard View: khung nhìn được tạo bao gồm các cột là các cột của bảng hoặc các khung nhìn khác.
- Indexed View: khung nhìn được tạo và được đặt chỉ mục Unique Clustered Index.
- Partitioned View: khung nhìn được tạo bao gồm các dữ liệu được phân cụm ngang từ một hoặc nhiều bảng.

Những **ưu điểm** khi quản lý dữ liệu bằng khung nhìn:

- **Bảo mật dữ liệu:** Người sử dụng chỉ được phép thao tác trên phần dữ liệu đã cấp phát quyền, điều này làm hạn chế được phần nào việc người sử dụng truy cập trực tiếp dữ liệu.

- **Đơn giản hoá các thao tác truy vấn dữ liệu:** Dữ liệu của khung nhìn được tập hợp từ nhiều bảng khác nhau, vì vậy người sử dụng có thể thực hiện các yêu cầu truy vấn dữ liệu một cách đơn giản từ khung nhìn thay vì phải đưa ra những câu truy vấn phức tạp.

- **Tập trung và đơn giản hóa dữ liệu:** Thông qua khung nhìn ta có thể cung cấp cho người sử dụng những cấu trúc đơn giản, dễ hiểu hơn về dữ liệu trong cơ sở dữ liệu đồng thời giúp cho người sử dụng tập trung hơn trên những phần dữ liệu cần thiết.

- **Độc lập dữ liệu:** Một khung nhìn có thể cho phép người sử dụng có được cái nhìn về dữ liệu độc lập với cấu trúc của các bảng trong cơ sở dữ liệu cho dù các bảng cơ sở có bị thay đổi phần nào về cấu trúc.

Tuy nhiên, việc sử dụng khung nhìn cũng tồn tại một số **nhược điểm** sau:

- Do hệ quản trị cơ sở dữ liệu thực hiện việc chuyển đổi các truy vấn trên khung nhìn thành những truy vấn trên các bảng cơ sở nên nếu một khung nhìn được định nghĩa bởi một truy vấn phức tạp thì sẽ dẫn đến chi phí lớn về mặt thời gian khi thực hiện truy vấn liên quan đến khung nhìn.

- Mặc dù thông qua khung nhìn có thể thực hiện được thao tác bổ sung và cập nhật dữ liệu cho bảng cơ sở nhưng chỉ hạn chế đối với những khung nhìn đơn giản. Đối với những khung nhìn phức tạp thì thường không thực hiện được; hay nói cách khác dữ liệu trong khung nhìn là chỉ đọc.

Tóm lại:

- Mỗi khung nhìn chứa một câu lệnh SELECT được lưu trữ như một đối tượng trong cơ sở dữ liệu. Bảng được viết trong câu lệnh SELECT tạo khung nhìn được gọi là *bảng cơ sở* của khung nhìn.

- Truy vấn tạo khung nhìn được tối ưu bởi SQL Server trước khi lưu vào cơ sở dữ liệu. Sau đó, trong bất kỳ câu lệnh thao tác dữ liệu nào như SELECT, INSERT, UPDATE và DELETE, bạn có thể gọi tên khung nhìn ở bất cứ nơi nào mà bạn thường sử dụng bảng.

- Dù vận hành như một bảng ảo, khung nhìn không lưu trữ bất kỳ dữ liệu nào. Do khung nhìn tham chiếu trở lại các bảng cơ sở của nó, khung nhìn luôn trả về dữ

liệu hiện thời.

## 5.2. Tạo khung nhìn

### 5.2.1. Tạo khung nhìn bằng câu lệnh

Câu lệnh CREATE VIEW được sử dụng để tạo ra khung nhìn và có cú pháp như sau:

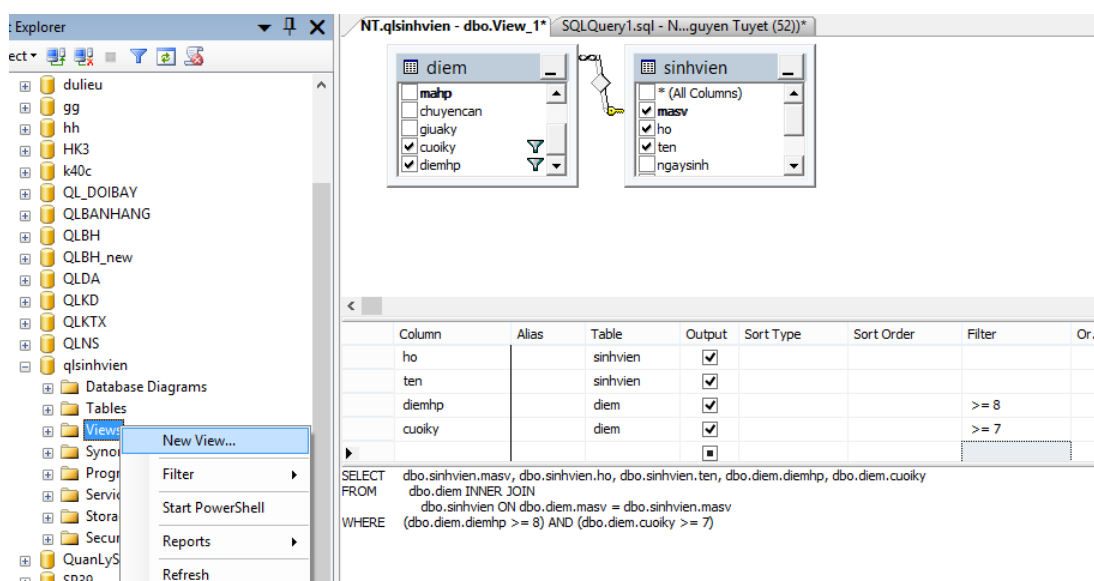
```
CREATE VIEW tên_khung_nhìn[(danh_sách_tên_cột)]
AS
[WITH {ENCRYPTION / SCHEMABINDING / VIEW_METADATA}]
câu_lệnh_SELECT
```

Ví dụ: Tạo khung nhìn lưu trữ những sinh viên có điểm học phần (diemhp) từ 8 điểm trở lên

```
CREATE VIEW ds_sinh_vien_gioi AS
SELECT sinhvien.masv, ho, ten, diemhp
FROM sinhvien INNER JOIN diem
ON sinhvien.Masv = diem.Masv
WHERE diemhp >= 8
```

### 5.2.2. Tạo khung nhìn bằng chức năng có sẵn của Microsoft SQL Server

Quan sát hình dưới đây:



Hình 5.1. Tạo khung nhìn bằng chức năng có sẵn của MS SQL Server



- Nhấn chuột phải tại View của database, chọn New View;
- Add các bảng tham gia tạo View
- Nhấn chọn các trường và thêm điều kiện theo yêu cầu của View ở cửa sổ bên phải.

### 5.2.3. Thực thi khung nhìn

Vì khung nhìn là một bảng ảo nên việc thực thi khung nhìn tương tự như việc sử dụng câu lệnh SELECT đối với các bảng. Mỗi lần thực thi khung nhìn dữ liệu được cập nhật lại

Ví dụ: muốn thực thi khung nhìn `ds_sinhvien_gioi` ở trên, ta viết:

```
SELECT * FROM ds_sinhvien_gioi
```

### 5.3. Một số quy định khi sử dụng khung nhìn

- Đặt tên khung nhìn không được trùng với tên của bất kỳ bảng hoặc khung nhìn nào đang tồn tại.

- Câu lệnh SELECT tạo khung nhìn có thể gọi đến 256 bảng và có thể sử dụng mọi kết hợp giữa các phép nối, hợp, hoặc truy vấn con hợp lệ.

- Có thể tạo khung nhìn dựa trên một khung nhìn khác thay vì bảng. Khung nhìn này được gọi là *khung nhìn lồng nhau*. Các khung nhìn trong SQL Server có thể được lồng vào nhau tới 32 mức.

- Câu lệnh SELECT tạo khung nhìn không thể bao gồm mệnh đề INTO và không thể bao gồm mệnh đề ORDER BY trừ khi từ khóa TOP cũng được sử dụng. Để sắp xếp các hàng trong một khung nhìn, bạn phải bao gồm mệnh đề ORDER BY trong câu lệnh SELECT lấy dữ liệu từ khung nhìn đó.

- Có thể đặt tên cột trong mỗi khung nhìn bằng cách viết một danh sách tên nằm trong dấu ngoặc đơn ngay sau tên khung nhìn, hoặc bằng cách viết các tên mới trong mệnh đề SELECT. Phải đặt tên cho các cột của khung nhìn trong các trường hợp sau đây:

+ Trong kết quả của câu lệnh SELECT có ít nhất một cột được sinh ra bởi một biểu thức (tức không phải là một tên cột trong bảng cơ sở) và cột đó chưa được đặt tiêu đề.

+ Tồn tại hai cột trong kết quả của câu lệnh SELECT có cùng tiêu đề cột.

- Có thể sử dụng mệnh đề WITH ENCRYPTION để ngăn người dùng không cho xem mã SQL định nghĩa khung nhìn.

- Có thể sử dụng mệnh đề **WITH SCHEMABINDING** để liên kết một khung nhìn với *schema cơ sở dữ liệu*. Sau đó, bạn không thể xóa các bảng mà khung nhìn dựa trên hoặc chỉnh sửa những bảng gây ảnh hưởng tới khung nhìn.

- Không thể qui định ràng buộc và tạo chỉ mục cho khung nhìn.

- Câu lệnh **SELECT** với mệnh đề **COMPUTE ... BY** không được sử dụng để định nghĩa khung nhìn.

#### 5.4. Cập nhật, bổ sung và xóa dữ liệu thông qua khung nhìn

Đối với một số khung nhìn, ta có thể tiến hành thực hiện các thao tác cập nhật, bổ sung và xóa dữ liệu. Thực chất, những thao tác này sẽ được chuyển thành những *thao tác trên các bảng cơ sở và có tác động đến những bảng cơ sở*.

Về mặt lý thuyết, để có thể thực hiện thao tác bổ sung, cập nhật và xóa, một khung nhìn trước tiên phải thỏa mãn các điều kiện sau đây:

- Trong câu lệnh **SELECT** định nghĩa khung nhìn không được sử dụng từ khóa **DISTINCT**, **TOP**, **GROUP BY** và **UNION**.

- Các thành phần xuất hiện trong danh sách chọn của câu lệnh **SELECT** phải là các cột trong các bảng cơ sở. Trong danh sách chọn không được chứa các biểu thức tính toán, các hàm gộp.

Ngoài những điều kiện trên, các thao tác thay đổi đến dữ liệu thông qua khung nhìn còn phải đảm bảo thỏa mãn các ràng buộc trên các bảng cơ sở, tức là vẫn đảm bảo tính toàn vẹn dữ liệu.

#### 5.5. Sửa đổi khung nhìn

Câu lệnh **ALTER VIEW** được sử dụng để định nghĩa lại khung nhìn nhưng không làm thay đổi các quyền đã được cấp phát cho người sử dụng trước đó. Thực chất của việc sửa đổi khung nhìn là sửa lại câu truy vấn **SELECT**.

Cú pháp như sau:

*ALTER VIEW tên\_khung\_nhìn [(danh\_sách\_tên\_cột)]*

*AS*

*Câu\_lệnh\_SELECT*

Ví dụ: Sửa lại khung nhìn **ds\_sinhvien\_gioi** lưu trữ những sinh viên có điểm học phần (diemhp) từ 8 điểm trở lên và điểm cuối kỳ từ 7 điểm trở lên

**ALTER VIEW ds\_sinh\_vien\_gioi AS**

```
SELECT sinhvien.masv, ho, ten, cuoiky, diemhp
FROM sinhvien INNER JOIN diem
ON sinhvien.Masv = diem.Masv
WHERE diemhp >= 8 and cuoiky>=7
```

## 5.6. Xóa khung nhìn

Khi một khung nhìn không còn sử dụng, ta có thể xóa nó ra khỏi cơ sở dữ liệu thông qua câu lệnh:

*DROP VIEW tên\_khung\_nhìn*

Nếu một khung nhìn bị xóa, toàn bộ những quyền đã cấp phát cho người sử dụng trên khung nhìn cũng đồng thời bị xóa. Do đó, nếu ta tạo lại khung nhìn thì phải tiến hành cấp phát lại quyền cho người sử dụng.

Ví dụ: Câu lệnh dưới đây xóa khung nhìn ds\_sinhvien\_gioi ra khỏi cơ sở dữ liệu

DROP VIEW ds\_sinhvien\_gioi

## CÂU HỎI ÔN TẬP CHƯƠNG 5

1. Khung nhìn là gì? Cho ví dụ minh họa.
2. Tại sao phải sử dụng khung nhìn?
3. So sánh sự giống và khác nhau giữa khung nhìn và bảng trong SQL Server?
4. Tạo, sửa, xóa và sử dụng khung nhìn như thế nào? Thực hiện các thao tác này qua một ví dụ cụ thể?

## CHƯƠNG 6. THỦ TỤC THƯỜNG TRỮ – STORED PROCEDURE

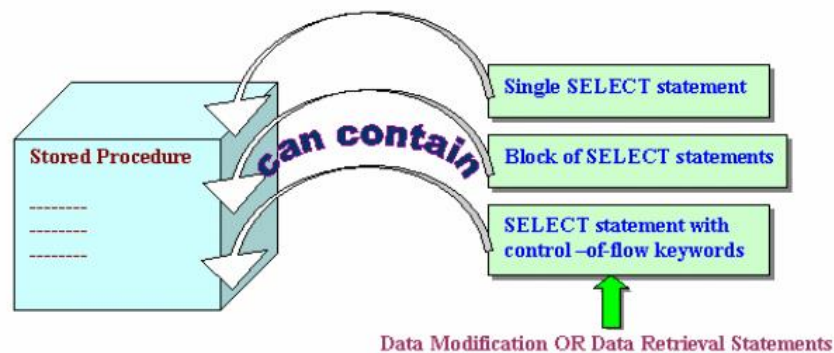
Trong các chương trước, khi truy vấn dữ liệu xong chúng ta có thể lưu các nhóm câu lệnh SQL vào một file văn bản (dạng \*.sql) để khi cần có thể sử dụng lại. Tuy nhiên thay vì lưu file văn bản ta có thể lưu vào trong SQL Server dưới dạng thủ tục lưu trữ (Stored Procedure). Đối với một lập trình viên đã có nhiều kinh nghiệm thường sẽ sử dụng thủ tục lưu trữ để thực hiện thao tác dữ liệu thay vì viết câu lệnh trực tiếp (dạng văn bản). Trong chương này chúng tôi sẽ giới thiệu về thủ tục thường trú – một phần khá quan trọng trong lập trình với cơ sở dữ liệu. Nội dung trình bày trong chương này gồm:

- Thủ tục lưu trữ là gì? Tại sao phải sử dụng thủ tục lưu trữ?
- Tìm hiểu các loại thủ tục lưu trữ của hệ thống;
- Cách tạo và sử dụng thủ tục lưu trữ như thế nào?
- Cách khai báo biến và sử dụng biến trong thủ tục lưu trữ.

### 6.1. Định nghĩa

Thủ tục lưu trữ là một nhóm câu lệnh Transact-SQL đã được biên dịch, lưu trữ trong SQL Server dưới một tên nào đó và được xử lý như một khối lệnh thống nhất (chứ không phải thực hiện nhiều câu SQL riêng lẻ).

Các thành phần của thủ tục lưu trữ:



**Hình 6.1. Các thành phần của thủ tục lưu trữ**

- Các cấu trúc điều khiển (IF, WHILE, FOR) có thể được sử dụng trong thủ tục.
- Bên trong thủ tục lưu trữ có thể sử dụng các biến như trong ngôn ngữ lập trình nhằm lưu giữ các giá trị tính toán được và lưu các giá trị được truy xuất từ cơ sở dữ liệu.
- Một thủ tục có thể nhận các tham số truyền vào cũng như có thể trả về các giá trị thông qua các tham số.

Khi một thủ tục lưu trữ đã được định nghĩa, nó có thể được gọi thông qua tên thủ tục, nhận các tham số truyền vào, thực thi các câu lệnh SQL bên trong thủ tục và có thể trả về các giá trị sau khi thực hiện xong.

## 6.2. Ưu điểm khi quản lý dữ liệu bằng thủ tục lưu trữ

- **Tăng tốc độ thực hiện:** Một trong những lợi ích lớn nhất khi sử dụng thủ tục lưu trữ đó là tốc độ. Thủ tục lưu trữ được tối ưu hoá trong ngay ở lần biên dịch đầu tiên, điều này cho phép chúng có thể thực hiện nhanh hơn rất nhiều lần so với các câu lệnh Transact-SQL thông thường.

- **Tốc độ truy cập dữ liệu nhanh hơn:** Khi thực thi một câu lệnh SQL thì SQL Server phải kiểm tra quyền (permission) xem user gửi câu lệnh đó có được phép thực hiện hay không đồng thời kiểm tra cú pháp rồi mới tạo ra một kế hoạch thực thi và thực thi. Nếu có nhiều câu lệnh như vậy gửi qua mạng có thể làm giảm đi tốc độ làm việc của server. SQL Server sẽ làm việc hiệu quả hơn nếu dùng thủ tục lưu trữ vì người gửi chỉ gửi một câu lệnh đơn và SQL Server chỉ kiểm tra một lần sau đó tạo ra một kế hoạch và thực thi. Nếu thủ tục lưu trữ được gọi nhiều lần thì kế hoạch thực thi có thể được sử dụng lại nên sẽ làm việc nhanh hơn. Ngoài ra cú pháp của các câu lệnh SQL đã được SQL Sever kiểm tra trước khi lưu nên nó không cần kiểm lại khi thực thi.

- **Chương trình được modul hoá:** Một khi thủ tục lưu trữ được tạo ra nó có thể được sử dụng lại. Điều này sẽ làm cho việc bảo trì dễ dàng hơn do việc tách rời giữa những logic thể hiện bên trong thủ tục lưu trữ với cơ sở dữ liệu. Ví dụ nếu có một sự thay đổi nào đó về mặt logic thì ta chỉ việc thay đổi mã lệnh bên trong thủ tục lưu trữ mà thôi. Những ứng dụng dùng thủ tục lưu trữ này có thể sẽ không cần phải thay đổi mà vẫn tương thích với những thay đổi mới về mặt logic của thủ tục.

- **Nhất quán:** Lợi ích nữa của thủ tục lưu trữ là thiết lập các ràng buộc dữ liệu để đảm bảo tính nhất quán. Người sử dụng không thể tùy tiện thao tác với dữ liệu để làm mất tính nhất quán của dữ liệu.

- **Nâng cao khả năng bảo mật dữ liệu:** Giả sử chúng ta muốn giới hạn việc truy xuất dữ liệu trực tiếp của một user nào đó trên một số bảng, ta có thể viết một thủ tục lưu trữ để truy xuất dữ liệu và chỉ cho phép user đó được sử dụng thủ tục lưu trữ đã viết sẵn mà thôi chứ không thể thao tác trực tiếp trên các bảng đó. Ví dụ, ta có thể tạo ra thủ tục lưu trữ, phân quyền EXECUTE cho những thủ tục lưu trữ này, và khi đó những user khác không được phép trực tiếp gọi thủ tục để làm việc với dữ

liệu. Ngoài ra thủ tục lưu trữ có thể được mã hóa (encrypt) để tăng cường tính bảo mật.

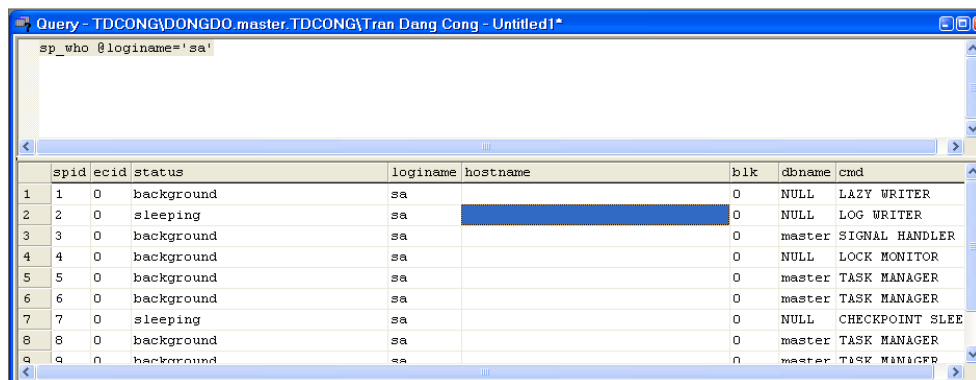
### 6.3. Phân loại thủ tục lưu trữ

#### System Stored Procedure:

Đây là thủ tục được lưu trữ trong CSDL Master, nó bắt đầu bằng **sp\_** được sử dụng trong quản trị CSDL và an ninh bảo mật.

Ví dụ: Muốn biết tất cả các tiến trình đang thực hiện bởi user nào:

```
sp_who @loginame='sa'
```



	spid	ecid	status	loginame	hostname	blk	dbname	cmd
1	1	0	background	sa		0	NULL	LAZY WRITER
2	2	0	sleeping	sa		0	NULL	LOG WRITER
3	3	0	background	sa		0	master	SIGNAL HANDLER
4	4	0	background	sa		0	NULL	LOCK MONITOR
5	5	0	background	sa		0	master	TASK MANAGER
6	6	0	background	sa		0	master	TASK MANAGER
7	7	0	sleeping	sa		0	NULL	CHECKPOINT SLEEPER
8	8	0	background	sa		0	master	TASK MANAGER
9	9	0	background	sa		0	master	TASK MANAGER

*Hình 6.2. Kết quả khi thực thi thủ tục sp\_who*

#### Local Stored Procedure:

Đây là loại thủ tục lưu trữ thường dùng nhất, nằm trong CSDL do người dùng tạo ra, thực hiện một công việc nào đó. Thủ tục loại này thường được tạo bởi DBA (Database Administrator) hoặc người lập trình.

#### Temporary Stored Procedure:

Đây là loại thủ tục có chức năng tương tự như Local Stored Procedure nhưng thủ tục loại này tự hủy khi kết nối tạo ra nó bị ngắt hoặc SQL Server ngưng hoạt động. Nó được tạo ra trên CSDL Tempdb.

#### Extended Stored Procedure:

Đây là loại thủ tục sử dụng chương trình ngoại vi đã được biên dịch thành thư viện liên kết động DLL (Dynamic Link Library). Tên thủ tục được bắt đầu bằng **xp\_**. Ví dụ thủ tục xp\_sendmail dùng để gửi mail, thủ tục xp\_cmdshell dùng để thực hiện lệnh của DOS (xp\_cmdshell 'dir c:\').

#### Remote Stored Procedure:

Đây là loại thủ tục sử dụng (gọi) thủ tục của một server khác.

#### 6.4. Tạo thủ tục lưu trữ

```
CREATE PROCEDURE / PROC tên_thủ_tục [(danh_sách_tham_số)]  
[WITH RECOMPILE / ENCRYPTION / RECOMPILE / ENCRYPTION]  
AS  
  
[BEGIN]  
  
    [DECLARE Khai_báo_các_biến_cục_bộ_của_thủ_tục]  
  
    Các_câu_lệnh_của_thủ_tục  
  
[END]
```

Trong đó:

- *tên\_thủ\_tục*: Tên của thủ tục lưu trữ (sau đây gọi tắt là thủ tục) do người dùng tự đặt (đặt theo qui tắc định danh, không được vượt quá 128 ký tự).
- *danh\_sách\_tham\_số*: Các tham số của thủ tục được khai báo ngay sau tên thủ tục và nếu thủ tục có nhiều tham số thì các khai báo cách nhau bởi dấu phẩy. Có thể sử dụng tối đa 2100 tham số trong *danh\_sách\_tham\_số* đối với mỗi thủ tục. Dung lượng tối đa có mỗi thủ tục là 128 MB.

Khai báo mỗi tham số bao gồm hai phần: *@tên\_tham\_số kiểu\_dữ\_liệu*

- *tên\_tham\_số* được bắt đầu bởi dấu @.
- *kiểu\_dữ\_liệu* của tham số

Ví dụ: @manganh char(5)

- RECOMPILE: Thông thường, thủ tục sẽ được phân tích, tối ưu và dịch sẵn ở lần gọi đầu tiên. Nếu tùy chọn WITH RECOMPILE được chỉ định, thủ tục sẽ được dịch lại mỗi khi được gọi.
- ENCRYPTION: Thủ tục sẽ được mã hóa nếu tùy chọn WITH ENCRYPTION được chỉ định. Nếu thủ tục đã được mã hóa, ta không thể xem được nội dung của thủ tục.
- DECLARE: khai báo biến phụ (nếu cần) để sử dụng trong thân thủ tục.
- các\_câu\_lệnh\_của\_thủ\_tục: Tập hợp các câu lệnh sử dụng trong nội dung thủ tục. Các câu lệnh này có thể đặt trong cặp từ khóa BEGIN...END hoặc có thể không.

Ví dụ: Hãy thực hiện các thao tác như sau trên cơ sở dữ liệu

1. Bổ sung thêm học phần *cơ sở dữ liệu* có mã *TI-005*, số tín chỉ là 3, mã

khoa quản lý học phần là *105* vào bảng HOCPHAN

2. Lên danh sách nhập điểm thi học phần *cơ sở dữ liệu* cho các sinh viên học lớp có mã *1050324102* (tức là bổ sung vào bảng DIEM các bản ghi với cột MAHP nhận giá trị *TI-005*, cột MASV nhận giá trị lần lượt là mã các sinh viên học lớp có mã *1050324102*, các cột điểm là *NULL*).

```
CREATE PROC sp_LenDanhSachDiem(
    @mahp          VARCHAR(10) ,
    @tenhp          NVARCHAR(50) ,
    @sotc           SMALLINT ,
    @makhoa         VARCHAR(10) ,
    @malop          VARCHAR(10) )
AS
BEGIN
    INSERT INTO hocphan
    VALUES (@mahp, @tenhp, @sotc, @makhoa)
    INSERT INTO diem(mahp, masv)
    SELECT @mahp, masv
    FROM sinhvien
    WHERE malop=@malop
END
```

### 6.5. Lệnh gọi thủ tục lưu trữ

Khi một thủ tục lưu trữ được tạo ra, ta có thể yêu cầu hệ quản trị cơ sở dữ liệu thực thi thủ tục bằng lệnh gọi thủ tục có dạng:

*tên\_thủ\_tục*      [*danh\_sách\_các\_đối\_số*]

*Danh\_sách\_các\_đối\_số* (tham số thực sự) phải phù hợp với số lượng và thứ tự của các tham số hình thức khi định nghĩa thủ tục.

Trong trường hợp lệnh gọi thủ tục được thực hiện bên trong một thủ tục khác, bên trong một trigger, thủ tục có giá trị trả về hay kết hợp với các câu lệnh SQL khác, ta sử dụng cú pháp như sau:



*EXECUTE tên\_thủ\_tục [danh\_sách\_các\_đối\_số]*

Nếu thứ tự của các đối số được truyền cho thủ tục (tham số thực sự) không theo thứ tự của các tham số hình thức như khi định nghĩa thủ tục thì tất cả các đối số phải được viết dưới dạng:

*@tên\_tham\_số = giá\_trị*

Ví dụ: Lời gọi thủ tục ở ví dụ trên có thể viết như sau:

```
sp_LenDanhSachDiem@malop='1050324102',  
@tenhp='Cơ sở dữ liệu',  
@mahp='TI-005',  
@sotc=3
```

#### 6.6. Sửa/xóa thủ tục lưu trữ

Khi cần thay đổi thủ tục lưu trữ ta sử dụng cú pháp:

```
ALTER PROCEDURE / PROC tên_thủ_tục [(danh_sách_tham_số)]  
[WITH RECOMPILE / ENCRYPTION / RECOMPILE / ENCRYPTION]  
AS  
[BEGIN]  
  
[Khai_báo_các_biến_cục_bộ_của_thủ_tục]  
  
Các_câu_lệnh_của_thủ_tục  
  
[END]
```

Xóa một thủ tục lưu trữ bằng cú pháp:

```
DROP PROC tên_thủ_tục
```

#### 6.7. Sử dụng biến trong thủ tục

Ngoài những tham số được truyền cho thủ tục, bên trong thủ tục còn có thể sử dụng các biến nhằm lưu trữ các giá trị tính toán được hoặc truy xuất được từ cơ sở dữ liệu. Các biến trong thủ tục được khai báo bằng từ khóa DECLARE theo cú pháp như sau:

```
DECLARE @tên_biến kiểu_dữ_liệu
```

*tên\_biến* phải bắt đầu bởi ký tự @ và tuân theo qui tắc về định danh.

Ví dụ: Trong định nghĩa của thủ tục dưới đây sử dụng các biến để chứa các giá trị

truy xuất được từ cơ sở dữ liệu.

```
CREATE PROCEDURE sp_Vidu (
    @malop1 VARCHAR(10),
    @malop2 VARCHAR(10))
AS
BEGIN
    DECLARE @tenlop1 NVARCHAR(30)
    DECLARE @makhoa1 VARCHAR(10)
    DECLARE @tenlop2 NVARCHAR(30)
    DECLARE @makhoa2 VARCHAR(10)

    SELECT @tenlop1=tenlop,@makhoa1=makhoa
    FROM lop WHERE malop=@malop1
    SELECT @tenlop2=tenlop,@makhoa2=makhoa
    FROM lop WHERE malop=@malop2

    PRINT @tenlop1+'thuoc khoa'+str(@makhoa1)
    PRINT @tenlop2+' thuoc khoa'+str(@makhoa2)
    IF @makhoa1=@makhoa2
        PRINT N'Hai lớp cùng thuộc một khoa'
    ELSE
        PRINT N'Hai lớp khác khoa'
END
```

Có 2 loại tham số hình thức:

- tham số nhập
- tham số xuất: có thêm từ khóa DECLARE trước tên biến trong lời gọi và thêm chữ OUTPUT sau tên kiểu trong thủ tục.

Ví dụ:

```
CREATE PROCEDURE t_cong (
    @a int,
```

```
@b int,  
@c int out)
```

```
AS Select @c=@a+@b
```

Lời gọi

```
declare @tong int  
select @tong=0  
execute t_cong 100,200, @tong out  
select @tong
```

Có thể cung cấp giá trị mặc định cho tham số: khi khai báo nó ta cho nó “= giá trị” và khi gọi thủ tục ta bỏ qua các giá trị cho các tham số này.

Ví dụ:

```
CREATE PROC t_macdinh(  
    @tenlop NVARCHAR(30)=NULL,  
    @noisinh NVARCHAR(100)=N'Huế'  
  
AS  
  
    BEGIN  
  
    IF @tenlop IS NULL  
        SELECT hodem,ten  
        FROM sinhvien INNER JOIN lop  
            ON sinhvien.malop=lop.malop  
        WHERE noisinh=@noisinh  
    ELSE  
  
        SELECT hodem,ten  
        FROM sinhvien INNER JOIN lop  
            ON sinhvien.malop=lop.malop  
        WHERE noisinh=@noisinh AND  
            tenlop=@tenlop  
  
    END
```

Khi đó lời gọi t\_macdinh tìm những sinh viên sinh ở Huế,

t\_macdinh @tenlop='Tin K29' tìm những sinh viên Tink29 sinh ở Huế.

## 6.8. Các phát biểu điều khiển

- Khởi lệnh : *BEGIN...END*

Cú pháp:

*BEGIN*

*Câu lệnh hoặc nhóm lệnh được thực thi*

*END*

- IF ... ELSE

*IF <BIỂU THỨC LOGIC>*

*CÂU LỆNH SQL/BEGIN CÂU LỆNH END*

*ELSE*

*CÂU LỆNH SQL/BEGIN CÂU LỆNH END*

- CASE: cho phép nhận một giá trị từ nhiều lựa chọn

*CASE <Biểu thức logic>*

*WHEN <Biểu thức > THEN <Câu lệnh>*

*...*

*[ELSE <Câu lệnh>]*

*END*

Ví dụ:

```
Select top 10
      Masv, tensv, ngaysinhnhath, thu=
Case datepart(w,ngaysinhnhath)
      When 1 then 'Sunday'
      When 2 then 'Monday'
      When 3 then 'Tuesday'
      When 4 then 'Wednesday'
      When 5 then 'Thursday'
```

```

        When 6 then 'Friday'

        When 7 then 'Saturday'

    End

    From sinhvien

```

Trong đó: hàm datepart(w, ngaysinhnhath) trả về 1 số nguyên cho biết ngaysinhnhath là thứ mấy trong w (tuần)

- WHILE

*WHILE <Biểu thức logic>*

*Câu lệnh SQL*

Hoặc *BEGIN*

*Câu lệnh SQL*

*[BREAK]*

*[CONTINUE]*

*END*

Ví dụ:

```

DECLARE @Number As int --khai báo biến Number

SET @Number = 1 -- gán giá trị 1 cho biến Number

WHILE @Number< 5 -- Trong khi biến Number còn nhỏ hơn 5
    thì còn thực hiện...

    BEGIN

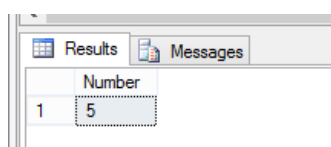
        SET @Number =@Number + 1 --Tăng giá trị biến
                                Number lên 1

    END

    SELECT @Number AS Number  --In giá trị biến
                                Number ra màn hình

```

Kết quả:



Number
1

Trong cấu trúc WHILE có thể dùng WAITFOR DELAY/TIME <TIME> để tạm dừng một thời gian TIME trước khi xử lý tiếp các phát biểu sau đó.

### 6.9. Sửa, xóa thủ tục

#### – Sử dụng câu lệnh.

Sửa: Alter Procedure

*ALTER PROCEDURE tên\_thủ\_tục[ @biến kiểu\_dữ\_liệu ]*

*AS*

*Begin*

*[Khai\_báo\_biến\_phụ]*

*Câu lệnh Transact-SQL*

*End*

Xóa: *DROP PROCEDURE tên\_thủ\_tục*

## CÂU HỎI ÔN TẬP CHƯƠNG 6

1. Thủ tục là gì? Tại sao phải viết thủ tục?
2. Có mấy loại thủ tục lưu trữ?
3. Có mấy loại tham số hình thức? Cho ví dụ.
4. Local Stored Procedure (thủ tục lưu trữ người dùng tạo ra) có mấy dạng?  
Cho ví dụ.

## **CHƯƠNG 7. HÀM DO NGƯỜI DÙNG ĐỊNH NGHĨA – USER DEFINED FUNCTION**

Trong SQL Server ngoài những hàm hệ thống có sẵn như các nhóm hàm: String Functions, Data and Time Functions, Mathematical Functions v.v...người ta cũng có thể tự xây dựng các hàm để phục vụ nhu cầu tính toán, phát triển ứng dụng hoặc các mục đích xử lý khác nhằm mang lại tiện lợi hơn cho người sử dụng.

Chương này tập trung giới thiệu các nội dung chính sau:

- Trình bày một cách khái quát hàm do người dùng định nghĩa, so sánh sự khác nhau giữa hàm và thủ tục;
- Phân loại hàm do người dùng định nghĩa và cách tạo chúng như thế nào?
- Cách sử dụng hàm trong CSDL.

### **7.1. Khái niệm**

Hàm do người dùng định nghĩa về mặt nào đó thì nó giống như thủ tục ở chương trước mà giáo trình đã giới thiệu. Hàm là một đối tượng trong cơ sở dữ liệu bao gồm một tập nhiều câu lệnh SQL được nhóm lại với nhau thành một nhóm. Điểm khác biệt giữa hàm và thủ tục là hàm trả về một giá trị thông qua tên hàm còn thủ tục thì không chính vì vậy mà trong thủ tục có tham số OUTPUT để lấy kết quả trả về. Điều này cho phép ta sử dụng hàm như là một thành phần của một biểu thức chẳng hạn như trong các câu lệnh truy vấn hay các câu lệnh thực hiện cập nhật dữ liệu

### **7.2. Những hạn chế khi sử dụng hàm do người dùng định nghĩa**

- Các hàm do người dùng định nghĩa không thể được sử dụng để thực hiện các hành động sửa đổi trạng thái cơ sở dữ liệu.
- Các hàm do người dùng định nghĩa không thể trả về nhiều tập kết quả. Nếu bạn cần trả về nhiều bộ kết quả thì hãy sử dụng thủ tục lưu trữ.
- Các hàm do người dùng định nghĩa không thể gọi thủ tục lưu trữ.
- Các hàm do người dùng định nghĩa có thể được lồng nhau; nghĩa là, một hàm do người dùng định nghĩa có thể gọi hàm khác. Mức lồng nhau được tăng lên khi hàm được gọi bắt đầu thực thi và giảm dần khi hàm được gọi kết thúc thực hiện.

### **7.3. Phân loại**

Hàm do người dùng định nghĩa được chia làm 3 loại:

- Scalar (hàm vô hướng): được sử dụng để trả về một giá trị duy nhất dựa trên tham số truyền vào.
- Inline table-valued (hàm nội tuyến): hàm trả về một bảng dựa trên một câu lệnh SQL duy nhất định nghĩa các dòng và các cột trả về.
- Multi -statement table-valued (hàm bao gồm nhiều câu lệnh SQL bên trong, trả về giá trị dạng bảng): hàm cũng trả về kết quả là một tập hợp giá trị dạng bảng dựa trên nhiều câu lệnh SQL.

### 7.3.1. Hàm vô hướng

```
CREATE FUNCTION tên_hàm ([danh_sách_tham_số])
RETURNS (kiểu_trả_về_của_hàm)
AS
BEGIN
    các_câu_lệnh_của_hàm
END
```

**Ví dụ 1:** (Trường hợp hàm không có *Danh\_Sach\_Cac\_Tham\_So*) hàm trả về giá trị là năm hiện hành

```
CREATE FUNCTION getYear ()
RETURNS int
AS
BEGIN
    RETURN YEAR(getdate())
END
```

**Ví dụ 2:** (Trường hợp hàm có sử dụng tham số trong *Danh\_Sach\_Cac\_Tham\_So*). Hàm sẽ trả về số ngày của tháng, năm do người dùng truyền vào. Với năm nhuận thì tháng 2 có 29 ngày, các năm khác có 28 ngày.

```
CREATE FUNCTION getnumday ( @Thang Int, @Nam Int)
RETURNS int
AS
BEGIN
    DECLARE @Ngay Int
```



```

IF @Thang = 2
BEGIN
    IF ((@Nam % 4 = 0 AND @Nam % 100 <> 0)
        OR (@Nam % 400 = 0))
        SET @Ngay = 29
    ELSE
        SET @Ngay = 28
END
ELSE
SELECT @Ngay =
CASE @Thang
    WHEN 1 THEN 31
    WHEN 3 THEN 31
    WHEN 5 THEN 31
    WHEN 7 THEN 31
    WHEN 8 THEN 31
    WHEN 10 THEN 31
    WHEN 12 THEN 31
    WHEN 4 THEN 30
    WHEN 6 THEN 30
    WHEN 9 THEN 30
    WHEN 11 THEN 30
END
RETURN @Ngay
END

```

### ***7.3.2. Hàm nội tuyến***

```

CREATE FUNCTION tên_hàm ([danh_sách_tham_số])
RETURNS TABLE

```

AS

*RETURN (1 câu lệnh\_select)*

Chú ý: Trong phần thân của hàm chỉ có duy nhất một câu lệnh RETURN xác định giá trị trả về của hàm thông qua duy nhất một câu lệnh SELECT. Ngoài ra, không sử dụng bất kỳ câu lệnh nào khác trong phần thân của hàm.

Ví dụ: Tạo hàm để xem danh sách sinh viên của một **Khóa** nào đó?

```
CREATE FUNCTION func_XemSV (@makhoa SMALLINT)
```

```
RETURNS TABLE
```

AS

```
RETURN (SELECT masv, hodem, ten, ngaysinh  
        FROM sinhvien INNER JOIN lop  
        ON sinhvien.malop=lop.malop  
        WHERE khoa=@makhoa)
```

Thực thi hàm        `SELECT * FROM func_XemSV(40)`

### **7.3.3. Hàm bao gồm nhiều câu lệnh bên trong**

Đối với hàm nội tuyến, phần thân của hàm chỉ cho phép sự xuất hiện duy nhất của câu lệnh RETURN. Trong trường hợp cần phải sử dụng đến nhiều câu lệnh trong phần thân của hàm, ta sử dụng cú pháp như sau để định nghĩa hàm:

*CREATE FUNCTION tên\_hàm ([danh\_sách\_tham\_số])*

*RETURNS @biến\_bảng TABLE định\_nghĩa\_bảng*

*AS*

*BEGIN*

*các\_câu\_lệnh\_trong\_thân\_hàm*

*RETURN*

*END*

Lưu ý:

- Sau từ khóa RETURNS là một biến bảng phải được định nghĩa. Và sau từ khóa RETURN ở cuối hàm không có tham số nào đi kèm.

- Cấu trúc của bảng trả về bởi hàm được xác định dựa vào định nghĩa của bảng trong mệnh đề RETURNS. Biến *@biến\_bảng* trong mệnh đề RETURNS có phạm vi sử dụng trong hàm và được sử dụng như là một tên bảng.

- Câu lệnh RETURN trong thân hàm không chỉ định giá trị trả về. Giá trị trả về của hàm chính là các dòng dữ liệu trong bảng có tên là *@biếnbảng* được định nghĩa trong mệnh đề RETURNS

Ví dụ: Nếu người dùng cho biết mã ngành thì lấy danh sách sinh viên thuộc mã ngành đó. Ngược lại, nếu người dùng không biết mã ngành thì hiển thị danh sách tất cả sinh viên. Viết hàm để thực hiện yêu cầu trên?

```
CREATE FUNCTION func_DSSV(@manganh char(4))
RETURNS @bang table (masv char(10), ho nvarchar(50), ten
nvarchar(30))
AS
BEGIN
    if @manganh=' '
        insert into @bang
        select masv,ho,ten from sinhvien
    else
        insert into @bang
        select masv, ho, ten
        from sinhvien
        where manganh=@manganh
RETURN
END
```

## CÂU HỎI ÔN TẬP CHƯƠNG 7

1. Hàm do người dùng định nghĩa là gì? Phân biệt hàm với thủ tục?
2. Có mấy loại hàm do người dùng định nghĩa? Cho ví dụ?

## CHƯƠNG 8. TRIGGER

Trong chương này chúng ta sẽ tìm hiểu một chức năng rất đặc biệt trong hệ quản trị CSDL SQL Server đó là Trigger. Trigger được xem như là một chức năng quan trọng giúp bảo đảm các ràng buộc toàn vẹn. Vậy nó là gì? nó có tác dụng như thế nào? và sử dụng nó ra sao?

### 8.1. Giao tác

#### 8.1.1. Giới thiệu

- *Ví dụ 1:* Khách hàng A chuyển số tiền X từ tài khoản ATM sang tài khoản tiết kiệm.

Giao dịch này gồm 2 thao tác:

+ Rút số tiền X trong tài khoản ATM của khách hàng A:  $SoDu = SoDu - X$

+ Cộng số tiền X vào tài khoản tiết kiệm của khách hàng A:  $SoDu = SoDu + X$

Trong khi thực hiện giao dịch trên, nếu 1 trong 2 thao tác trên gặp sự cố thì cả giao dịch phải bị hủy để đảm bảo nhất quán dữ liệu.

- *Ví dụ 2:* Thêm thông tin của một sinh viên mới vào danh sách lớp nếu lớp đó có dưới 40 sinh viên.

Việc này được thực hiện bởi hai bước:

+ Thêm thông tin của một sinh viên mới vào bảng danh sách lớp (Chèn dữ liệu)

+ Cập nhật số lượng sinh viên dựa vào danh sách lớp. Nếu số lượng dưới 40 thì cho phép thêm thông tin sinh viên, ngược lại giao dịch phải bị hủy để đảm bảo nhất quán dữ liệu. (Cập nhật dữ liệu)

Tóm lại, khi thực hiện một giao dịch gồm một dãy các thao tác, để đảm bảo tính nhất quán dữ liệu thì hoặc là tất cả các thao tác đều thực hiện thành công, hoặc là hủy tất cả các thao tác để phục hồi dữ liệu về trạng thái ban đầu. Trường hợp đó người ta gọi là giao tác.

#### 8.1.2. Khái niệm

Giao tác cơ sở dữ liệu (database transaction) là đơn vị tương tác của một hệ quản trị cơ sở dữ liệu, mỗi giao tác được xử lý một cách nhất quán và tin cậy mà không phụ thuộc vào các giao tác khác. Một hệ cơ sở dữ liệu lý tưởng sẽ phải đảm bảo toàn bộ các tính chất ACID cho mỗi giao tác. (Wikipedia)

Giao tác là một tiến trình xử lý có xác định điểm đầu và điểm cuối, được chia nhỏ thành các thao tác, tiến trình được thực thi một cách tuần tự và độc lập các thao tác đó theo nguyên tắc hoặc tất cả đều thành công hoặc một thao tác thất bại thì toàn bộ tiến trình thất bại. Nếu việc thực thi một thao tác nào đó bị hỏng đồng nghĩa với việc dữ liệu phải rollback (trở lại) trạng thái ban đầu.

Các tính chất ACID của giao tác:

ACID là từ viết tắt các chữ cái đầu của bốn từ tiếng Anh: Atomicity, Consistency, Isolation, Durability. Chúng được coi là 4 thuộc tính quan trọng của một hệ quản trị cơ sở dữ liệu khi xử lý bất kỳ giao tác nào. Nếu thiếu một trong những thuộc tính này thì tính toàn vẹn của cơ sở dữ liệu khó có thể đảm bảo. Các tính chất ACID trong trường hợp này sẽ đảm bảo các giao dịch được thực hiện một cách đáng tin cậy (Wikipedia):

– Tính Nguyên tử (Atomicity) Một giao dịch có nhiều thao tác khác biệt thì hoặc là toàn bộ dãy các thao tác được thực hiện trên CSDL hoặc không thao tác nào trong dãy thao tác đó được thực hiện.

– Tính nhất quán (Consistency) Một giao tác hoặc là sẽ tạo ra một trạng thái mới và hợp lệ cho dữ liệu, hoặc trong trường hợp có lỗi sẽ chuyển toàn bộ dữ liệu về trạng thái trước khi thực hiện giao tác.

– Tính cô lập (Isolation) Giao tác khi thực hiện không làm ảnh hưởng và không bị ảnh hưởng bởi các giao tác khác thực hiện đồng thời.

– Tính bền vững (Durability): Dữ liệu sau khi giao tác hoàn thành sẽ được hệ thống lưu lại sao cho ngay cả trong trường hợp hỏng hóc hoặc có lỗi hệ thống, dữ liệu vẫn đảm bảo trong trạng thái chuẩn xác.

Giao tác được định nghĩa bắt đầu bằng BEGIN TRAN và kết thúc bởi hành động COMMIT TRAN hoặc ROLLBACK TRAN, trong thân giao tác có thể sử dụng SELECT, DELECT, UPDATE, INSERT.

- COMMIT TRAN xác định kết thúc hay hoàn tất giao tác.
- ROLLBACK TRAN khi gặp chuyển tác này tất cả những phát biểu từ khi gặp BEGIN sẽ bị huỷ bỏ.
- SAVE TRAN để lưu lại vị trí một chuyển tác.

## 8.2. Trigger

### 8.2.1. Khái niệm

Trigger là một thủ tục được thực hiện tự động để đáp ứng các sự kiện nhất định trên một bảng hay một khung nhìn cụ thể trong cơ sở dữ liệu. Trigger chủ yếu được sử dụng để duy trì tính toàn vẹn của thông tin trên cơ sở dữ liệu (Wikipedia).

Mỗi một trigger được tạo ra và được gắn liền với một bảng nào đó trong cơ sở dữ liệu. Khi dữ liệu trong bảng bị thay đổi (tức là bảng chịu tác động của các câu lệnh INSERT, UPDATE hay DELETE) thì trigger sẽ được tự động kích hoạt.

Khi trigger được kích hoạt, SQL Server tạo ra một giao tác theo dõi những thay đổi do thao tác kích hoạt trigger hoặc trigger gây ra và giao tác này cho phép dữ liệu trở về trạng thái trước đó.

SQL định nghĩa hai bảng logic INSERTED và DELETED để sử dụng trong các trigger. Cấu trúc của hai bảng này tương tự như cấu trúc của bảng mà trigger tác động. Dữ liệu trong hai bảng này tùy thuộc vào câu lệnh tác động lên bảng làm kích hoạt trigger; cụ thể trong các trường hợp sau:

- Khi câu lệnh DELETE được thực thi trên bảng, các dòng dữ liệu bị xóa sẽ được sao chép vào trong bảng DELETED. Bảng INSERTED trong trường hợp này không có dữ liệu.

- Dữ liệu trong bảng INSERTED là dòng dữ liệu được bổ sung vào bảng gây nên sự kích hoạt đối với trigger bằng câu lệnh INSERT. Bảng DELETED trong trường hợp này không có dữ liệu.

- Khi câu lệnh UPDATE được thực thi trên bảng, các dòng dữ liệu cũ chịu sự tác động của câu lệnh sẽ được sao chép vào bảng DELETED, còn trong bảng INSERTED sẽ là các dòng sau khi đã được cập nhật.

Trigger chia thành 2 loại: INSTEAD OF và AFTER:

- INSTEAD OF: là trigger mà thao tác kích hoạt trigger sẽ bị bỏ qua và thay vào đó là các lệnh trong trigger được thực hiện.

- AFTER: là loại trigger mặc định, trigger này thực hiện các lệnh bên trong trigger sau khi đã thực hiện thao tác kích hoạt trigger.

*Ví dụ:* Khi thêm thông tin (ngày nhập học) sinh viên mới hoặc chỉnh sửa ngày nhập học của sinh viên, thì ngày nhập học phải trước ngày hiện tại. Tạo trigger để kiểm soát công việc này như sau:

```
CREATE TRIGGER Insert_NamNH  
ON SinhVien  
FOR INSERT, UPDATE
```

AS

```
If (select NgayNH from Inserted) > getdate()  
Begin  
    Print 'Ngày nhập học không hợp lý'  
    ROLLBACK TRAN  
End
```

Trong đó: Insert\_NamNH là tên trigger do người dùng tự đặt

Getdate() là hàm lấy ngày tháng năm hiện tại của hệ thống

Rollback tran là lệnh hủy tất cả những thay đổi về dữ liệu

### 8.2.2. Khi nào dùng trigger?

Trigger rất hữu ích khi:

- Dùng trigger để kiểm soát, ngăn chặn và có thể hủy bỏ những thao tác không đúng quy tắc trong việc thay đổi dữ liệu hoặc nhằm đảm bảo tính toàn vẹn cho liệu và cho ra những câu thông báo thích hợp với người dùng.

- Dùng trigger để tự động thực hiện hàng loạt các thao tác khác trên CSDL ứng với một số thao tác thay đổi dữ liệu nào đó (UPDATE, INSERT hoặc DELETE) trên một bảng hay nhiều bảng.

- Dùng trigger để kiểm soát các quy tắc phức tạp giữa các bảng trong CSDL mà các Check Constraints (Database Tools) không thực hiện được.

Tuy nhiên, trigger vẫn tồn tại một số hạn chế:

- Trigger không được kích hoạt bởi một bảng tạm hay bảng hệ thống (temporary, các table có tên chứa #, ##, system table), nhưng nó có thể tham chiếu đến nội dung bên trong các bảng tạm, bảng hệ thống.

- Các trigger INSTEAD OF DELETE, INSTEAD OF UPDATE không thể định nghĩa trên các table có chứa khóa ngoại (đặc tính Cascade Delete Related Records của quan hệ khóa ngoại đó được thiết lập). Nhưng với Truncate Table, Writetext không kích hoạt được trigger định nghĩa cho thao tác Delete.

*Ví dụ:* Tạo trigger để thực hiện công việc sau: Khi một sinh viên đăng ký môn học mới (chưa có trong danh sách) vào danh sách lớp (bảng SinhVien) thì trường SiSo trong bảng Lop sẽ tự động cập.

```
CREATE TRIGGER SisoLop  
ON DSdangkyHP
```

```

FOR INSERT
AS
Begin
    Declare @maHP_moi char(10) = (select MaHP from
inserted)
    Declare @SiSo_moi float = (select count (*) from
DSdangkyHP where MaHP=@maHP_moi group by MaHP)
    Update hocphan
    SET SiSoLop= @SiSo_moi
    WHERE MaHP =@maHP_moi
End

```

### 8.2.3. Tạo trigger

Câu lệnh sau được sử dụng để định nghĩa trigger:

```

CREATE TRIGGER <tên_trigger>
ON <tên_bảng> /<tên_view>
    [WITH ENCRYPTION]
    {FOR | AFTER | INSTEAD OF } {[INSERT][,][UPDATE][,][DELETE]}
    [NOT FOR REPLICATION]
AS
    [IF UPDATE <tên_cột>
    [AND UPDATE <tên_cột>|OR UPDATE <tên_cột>]...]
    các_câu_lệnh_của_trigger

```

Trong đó:

- WITH ENCRYPTION: ngăn chặn người dùng khác xem nội dung của trigger.
- AFTER: Trigger được thực thi sau khi tất cả các câu lệnh trong thân trigger đã thực thi thành công. AFTER là kiểu mặc định nếu trigger có dùng từ khóa FOR. Không định nghĩa trigger AFTER cho View.
- INSTEAD OF: Trigger được thực thi thay cho các câu lệnh SQL gây ra trigger. INSTEAD OF dùng được cho View.
- {[INSERT][,][UPDATE][,][DELETE]}: Chỉ rõ thao tác mà trigger thực thi trên bảng hoặc View.
- NOT FOR REPLICATION: Trigger sẽ không được thực hiện khi việc đồng bộ hóa tạo sự thay đổi trên các bảng, các View có liên quan.



- IF UPDATE (tên\_cột) dùng để chỉ định trigger được kích hoạt khi có sự thay đổi dữ liệu trên các cột được chỉ ra (tên\_cột). IF UPDATE chỉ kiểm tra thao tác Insert, Update (không dùng cho Delete).

#### 8.2.4 Các kiểu Trigger

##### - INSERT TRIGGER:

Ví dụ: Khi một sinh viên đăng ký môn học mới vào bảng DsdangkyHP, thì trigger sẽ kiểm tra xem sinh viên đó đã đăng ký bao nhiêu học phần; Nếu chưa đủ 10 học phần thì cho phép đăng ký bình thường còn không thì thông báo đã đủ không cho phép đăng ký. (Giả sử quy định chỉ cho phép đăng ký tối đa 10 học phần).

```
CREATE TRIGGER SoHP
ON DSdangkyHP
FOR INSERT
AS
Begin
    Declare @MasvDK char (10) = (select Masv from
inserted)
    Declare @SoHP float = (select count (*) from
DSdangkyHP where Masv=@MasvDK group by Masv)
    If (@SoHP > 10)
    Begin
        Print 'Ban da dang ky 10 HP, khong duoc DK nua'
        ROLLBACK TRAN
    End
End
```

##### - UPDATE TRIGGER:

Ví dụ: Khi cập nhật điểm HP thì trigger tính lại điểm HP theo công thức qui định.

```
CREATE TRIGGER DiemHP
ON Diem
FOR UPDATE
AS
Begin
    Declare @MasvUP char (10) = (select Masv from
inserted)
```

```

        Declare @DiemHP float = (select (chuyencan * 10 +
giuaky * 20 + cuoiky * 70)/100 from Diem where
Masv=@MasvUP)
        update Diem
        set Dhp= @DiemHP
        where Masv=@MasvUP
End

```

### **- DELETE TRIGGER**

Ví dụ: Viết trigger sao cho khi xóa 1 sinh viên thì bảng dsdangkyHP cũng sẽ xóa theo.

```

create trigger Delete_MH
ON sinhVien
FOR DELETE
AS
Begin
        Declare @Masv_Delete char (10) = (select Masv from
deleted)
        Delete DSDangkyHP where masv = @Masv_Delete
End

```

### **- Kết hợp INSERT và UPDATE**

Ví dụ: Viết trigger thực hiện việc nếu thay đổi (thêm hoặc sửa) điểm HP thì đều phải tính lại điểm HP theo công thức quy định.

```

CREATE TRIGGER DiemHP_IU
ON Diem
FOR INSERT, UPDATE
AS
Begin
        Declare @MasvUP char (10) = (select Masv from
inserted)
        Declare @DiemHP float = (select (chuyencan * 10 +
giuaky * 20 + cuoiky * 70)/100 from Diem where
Masv=@MasvUP)
        update Diem set Dhp= @DiemHP where Masv=@MasvUP
End

```

#### ***8.2.5 Sửa đổi Trigger***

Thực hiện sửa đổi trigger đã tồn tại, ta sử dụng cú pháp:

```
ALTER TRIGGER tên_trigger  
ON <tên_bảng> / <tên_View>  
[WITH ENCRYPTION]  
(FOR / AFTER / INSTEAD OF)  
{[DELETE][,][INSERT][,][UPDATE]  
[NOT FOR REPLICATION]  
AS  
[các_câu_lệnh_của_trigger]
```

Ví dụ: Trong trigger DiemHP\_IU ở trên, nhưng quy định về tính điểm thay đổi vì vậy trigger cập nhật điểm được sửa đổi như sau.

```
ALTER TRIGGER DiemHP_IU  
ON Diem  
FOR INSERT, UPDATE  
AS  
Begin  
    Declare @MasvUP char (10) = (select Masv from  
inserted)  
    Declare @DiemHP float = (select (chuyencan * 20 +  
giuaky * 30 + cuoiky * 50)/100 from Diem where  
Masv=@MasvUP)  
    update Diem set Dhp= @DiemHP where Masv=@MasvUP  
End
```

### 8.2.6 Xóa Trigger

Cú pháp:

```
DROP TRIGGER <tên_trigger>
```

Ví dụ: Xóa trigger DiemHP và trigger SoHP

```
Drop Trigger DiemHP  
Drop Trigger SoHP
```

## CÂU HỎI ÔN TẬP CHƯƠNG 8

1. Khi nào dùng trigger?
2. So sánh trigger với procedure, function.
3. Có các kiểu trigger nào?

## CHƯƠNG 9. BẢO MẬT VÀ PHÂN QUYỀN NGƯỜI DÙNG

SQL Server chỉ cho phép bạn truy cập vào hệ thống qua xác thực đăng nhập. Chỉ khi bạn có quyền hạn ở mức độ nhất định thì bạn mới có thể tạo thêm những login khác. Trong SQL Server Management Studio bạn có thể xem tất cả các login bằng cách mở rộng những node Security/Logins. Những Login này chỉ có quyền truy cập vào Server chứ chưa thể vào được database (cơ sở dữ liệu) ở bên trong.

Với mỗi cơ sở dữ liệu lại duy trì một danh sách những người dùng, các người dùng này luôn luôn gắn với một login ở mức độ Server. Nếu bạn đăng nhập vào SQL Server qua login này, bạn sẽ có quyền truy cập vào cơ sở dữ liệu theo quyền hạn mà người dùng tương ứng với nó được cung cấp. Bạn có thể xem những người dùng này bằng cách mở rộng Security/Users của cơ sở dữ liệu tương ứng.

Trong chương này sẽ cho chúng ta biết bảo mật dữ liệu SQL Server là gì? Cách bảo mật và quản trị người dùng thực hiện như thế nào?

### 9.1. Khái niệm

Bảo mật thông tin là bảo vệ thông tin, dữ liệu chỉ cho phép những cá nhân, tổ chức có nhiệm vụ, quyền sử dụng được phép truy cập, không cho phép những người không được cho phép đánh cắp, thay đổi thông tin, dữ liệu.

Bảo mật thông tin là duy trì tính bảo mật, tính toàn vẹn dữ liệu và tính sẵn sàng cho toàn bộ thông tin.

- Tính bảo mật: đảm bảo thông tin đó là duy nhất, những người muốn truy cập phải được cấp phát quyền truy cập.
- Tính toàn vẹn, chính xác: bảo vệ sự hoàn chỉnh toàn diện, chính xác thông tin của hệ thống thông tin.
- Tính sẵn sàng: việc bảo mật phải luôn sẵn sàng, được thực hiện bất kì lúc nào, ở đâu.

Bảo mật cơ sở dữ liệu trong SQL Server cho phép những người quản trị cơ sở dữ liệu thiết lập quyền hạn cho người dùng hoặc nhóm người dùng khai thác CSDL. Người dùng hoặc nhóm người dùng sau khi được cấp quyền, có thể đăng nhập vào hệ thống và thực hiện quyền hạn mà mình được cấp.

Thông qua tính năng bảo mật, SQL Server cho phép xác định người sử dụng nào đang truy cập dữ liệu. Mỗi cơ sở dữ liệu có một tập hợp người sử dụng riêng,

mỗi người có quyền hạn riêng để truy cập CSDL. Các quyền này cho phép người sử dụng được truy cập, thay đổi dữ liệu hoặc xóa các đối tượng dữ liệu.

## 9.2. Quản lý người dùng

Một trong những lý do quan trọng để sử dụng SQL Server là khả năng quản lý nhiều người sử dụng truy cập đến cùng CSDL tại một thời điểm. Có nhiều vấn đề mà hệ quản trị CSDL phải thực hiện khi có nhiều người truy cập đồng thời như: quyền ưu tiên truy cập dữ liệu của người sử dụng, hai người cùng cập nhật một mẫu tin tại một thời điểm,... SQL Server đã hỗ trợ nhiều chức năng cho phép giải quyết đa số các vấn đề phức tạp này.

SQL Server cho phép tạo và quản lý hai loại người dùng:

- Login là người dùng hệ thống (System User). Mỗi Login dùng để truy cập vào hệ thống SQL Server, các loại Login chỉ có quyền truy cập vào Server chứ chưa hẳn có quyền truy cập vào các Database trên Server, các quyền truy cập vào Database thường được gắn liền với các người dùng CSDL (Users).

- User là người dùng CSDL (Database User). Mỗi Database có một danh sách người dùng được phép truy cập CSDL của mình, mỗi user luôn được gắn với một Login ở mức Server. Ví dụ, khi đăng nhập vào SQL Server thông qua Login có tên Login1, khi đó User có tên Ulogin1 tương ứng có quyền truy cập vào Database theo quyền hạn mà Ulogin1 được cấp. Mỗi Login có thể gắn với một hoặc nhiều User với quyền hạn khác nhau trên các Database.

Khi cài đặt SQL Server, hệ quản trị mặc định tạo sẵn một account SA - System Administrator. SA có toàn quyền trên hệ quản trị CSDL như account, user, back up và khôi phục CSDL, cấp phát, thu hồi quyền,... SA là account của quản trị viên hệ thống SQL Server.

## 9.3. Các cơ chế bảo mật

SQL Server áp dụng ba cơ chế bảo mật: Windows Authentication và SQL server Authentication và cơ chế hỗn hợp (cơ chế gồm cả Windows và SQL)

Cơ chế này quy định từ khi tạo Login trong SQL server. Khi tạo một Login mới, SQL Server sẽ yêu cầu bạn chọn cơ chế xác thực Window Authentication hay SQL server Authentication cho Login đó.

- Nếu chọn Window Authentication, hệ thống yêu cầu cung cấp một Window Account và SQL Server chỉ lưu tên của Login đó trong danh sách Login.

- Nếu chọn SQL Server Authentication, hệ thống yêu cầu cung cấp Login Name và Password và cả hai thông tin được lưu trong SQL Server.

Khi đăng nhập chúng ta phải chọn một trong hai cơ chế xác thực. Nếu chọn Windows chính Account hiện đang đăng nhập vào Windows được dùng, và nếu chọn cơ chế SQL Server Authentication thì hệ thống sẽ kiểm tra xem Login và Password có trong danh sách hay không mới cho phép đăng nhập hoặc từ chối.

#### 9.4. Các mức bảo mật

Role là tập hợp các nhóm quyền. Server Role là nhóm các quyền ở mức Server mà khi Login được cấp, nó có thể thực hiện một số thao tác nhất định đó ở mức Server.

Ví dụ quyền Role Sysadmin là nhóm quyền cao nhất, nó có toàn bộ các quyền hoạt động trong Server (như tạo database, được phép truy cập vào tất cả các database,...)

Login khi mới được tạo sẽ có quyền Role Public, quyền này được phép đăng nhập vào Server.

Database Role tập hợp các quyền truy cập vào Database thành từng nhóm để dễ tạo lập và sửa đổi. Các User mặc định đều có Role Public. Khi đó User chỉ có thể nhìn thấy tên Database mà chưa có quyền gì khác.

Một số Database Role như: Role db\_datareader có quyền đọc dữ liệu ở các bảng; Role db\_datawriter có quyền ghi vào các bảng; Role db\_owner có quyền cao nhất đối với Database (tạo, chỉnh sửa, xóa bảng, tạo thủ tục, trigger,...)

SQL Server có 4 mức bảo mật khác nhau tùy theo mức độ cho phép người dùng ở mức nào:

- Mức Hệ điều hành: người dùng được phép đăng nhập vào hệ thống và được sử dụng một số quyền thuộc nhóm quyền Server Role.

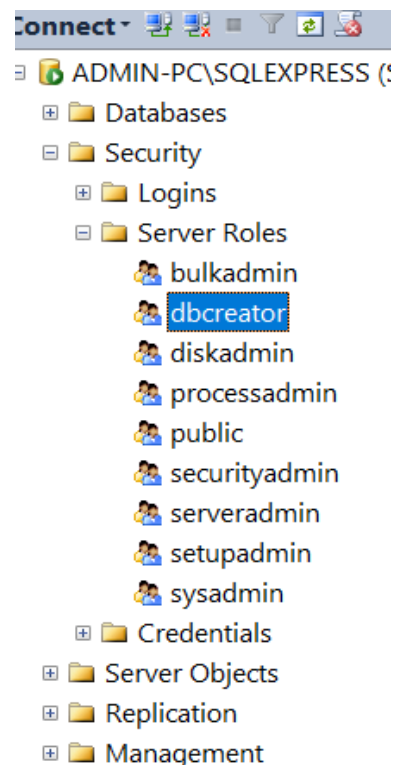
- Mức SQL Server: người dùng có thể được cấp phát quyền ở mức SQL. Trong đó quyền mặc định Login nào cũng có là Role Public, hoặc có thể được cấp quyền cao nhất Sysadmin.

- Mức Database: người dùng có thể truy cập hoặc xử lý CSDL nếu người dùng đó được cấp phát quyền thao tác trên CSDL đó.

- Mức đối tượng (bảng, view, store procedure,...): để truy xuất vào đối tượng nếu người dùng được cấp phát quyền trên đối tượng đó.

Server Role: Nhóm các quyền thực hiện quản trị hệ thống, gồm các nhóm sau:

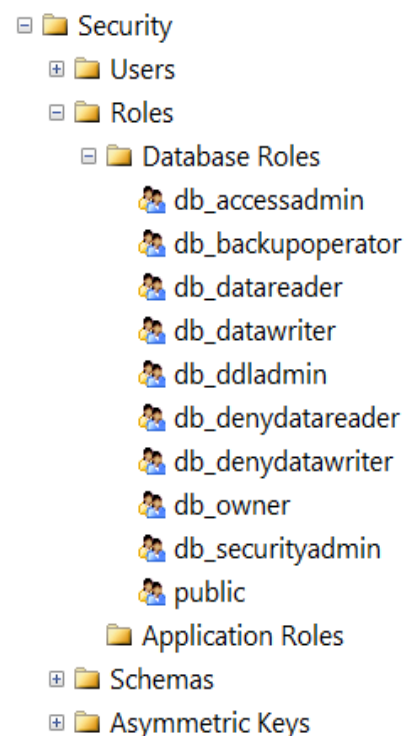
- + Bulk Insert Administrators: Được phép thực hiện Bulk Insert.
- + Database Creators: Được phép tạo và sửa đổi cấu trúc Database.
- + Disk Administrators: Có thể quản trị các file trên đĩa.
- + Process Administrator: Quản trị các dịch vụ đang chạy của SQL Server.
- + Public: Account có quyền mặc định của SQL Server.
- + Security Administrators: Quản trị hệ thống bảo mật như quản lý account, cấp quyền, thu hồi quyền, Re-set mật khẩu cho account.
- + Setup Administrators: Quản trị các thủ tục mở rộng như thêm hoặc xóa các các linked Server hoặc thực thi một số restore procedure của hệ thống.
- + System Administrators: Account là thành viên của Role này có toàn quyền trong hệ thống SQL Server.



**Hình 9.1. Các nhóm quyền Server Role**

Database Role: nhóm các quyền thực hiện trên Database:

- + Db\_accessadmin: quyền quản trị truy cập CSDL.
- + Db\_backupoperator: quyền sao lưu dữ liệu.
- + Db\_datareader: quyền đọc dữ liệu trong database.
- + Db\_datawriter: quyền ghi, sửa, xóa dữ liệu lên database.



**Hình 9.2. Các nhóm quyền Database Role**

- + Db\_ddladmin: có quyền thực hiện câu lệnh DDL (Data Definition Language) trong database.
- + Db\_denydatareader: không thể đọc dữ liệu trong database.
- + Db\_denydatawriter: không thể ghi dữ liệu vào database.
- + Db\_owner: quyền sở hữu database tức là có toàn quyền trên database.
- + Db\_securityadmin: quyền được phép cấp quyền cho các account khác trong phạm vi database của account đó.
- + Public: quyền chung cho tất cả các người dùng CSDL, đây là quyền tối thiểu để truy cập database.

## 9.5. Tạo người dùng

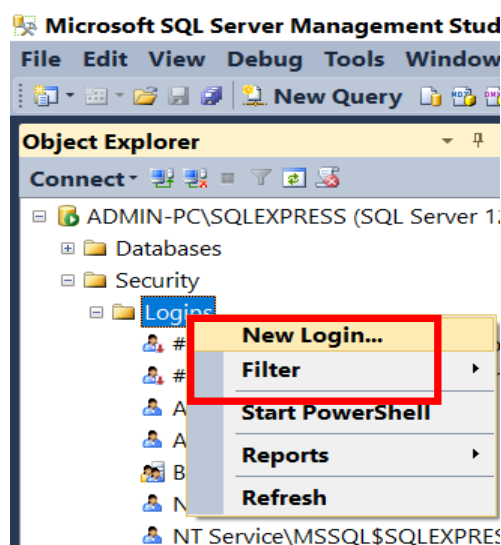
### 9.5.1. Tạo Login

- Tạo Login với cơ chế bảo mật Windows

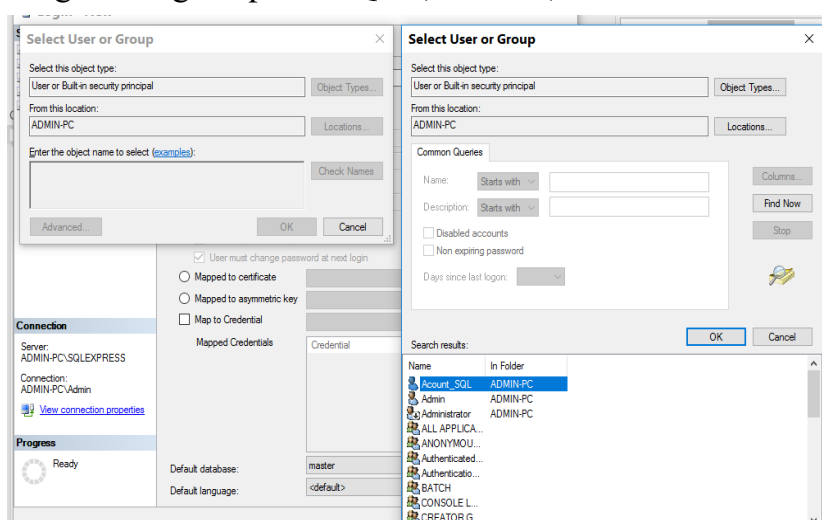
Bước 1: Mở cửa sổ Object Explorer, Chọn mục Security\Login\New Login (Hình 9.3)

Bước 2: Ở mục Login Name, chọn nút lệnh Search → Hộp thoại Select User or Group, chọn nút Advanced và tiếp tục chọn Find

Now → Danh sách các Account trong Windows hiện ra, và chọn một Account muốn tạo dùng để đăng nhập vào SQL. (Hình 9.4)



Hình 9.3. Tạo Login



Hình 9.4. Chọn Account

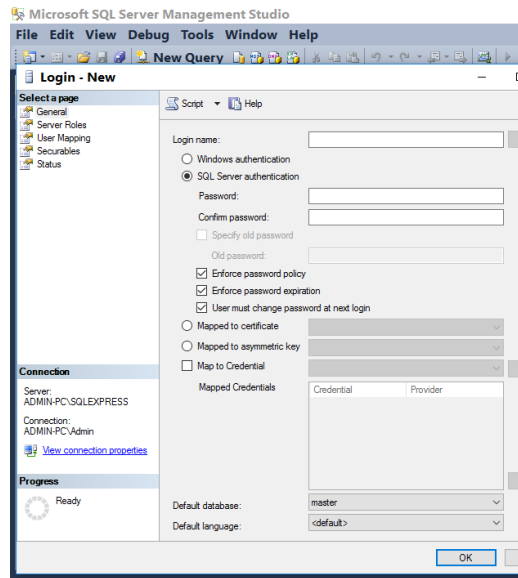


Bước 3: Chọn nút lệnh OK

- *Tạo Login với cơ chế bảo mật SQL Server Authentication:*

Bước 1: Mở cửa sổ Object Explorer, Chọn mục Security\Login\New Login (Hình 9.3)

Bước 2: Ở mục Login Name, gõ tên Login muốn đặt. (Hình 9.5)



**Hình 9.5. Đặt tên Login Name**

Bước 3: Chọn option SQL Server Authentication

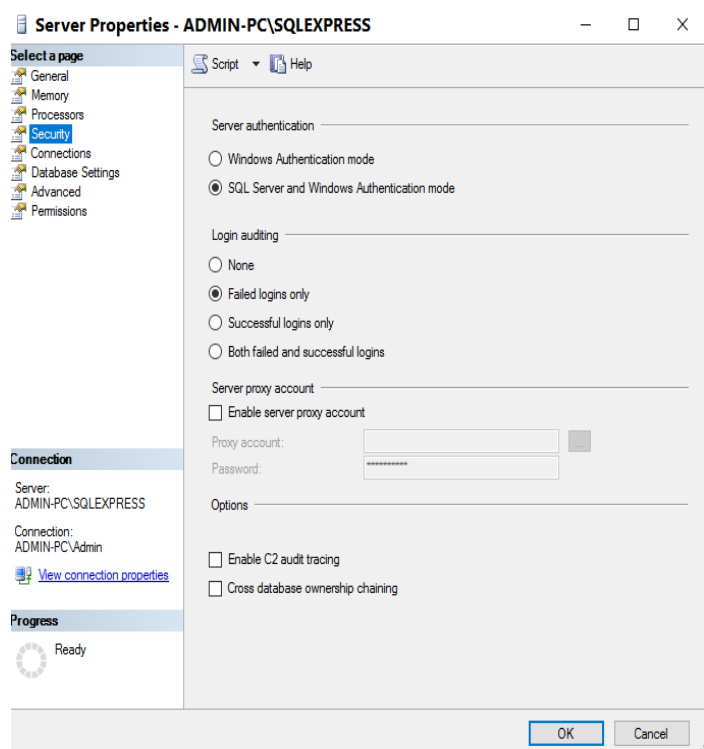
Bước 4: Ở mục Password đặt Password đăng nhập

Bước 5: Ở mục Default Database chọn Database cho phép Login truy cập.

Bước 6: Chọn nút lệnh OK

Chúng ta muốn Server cho phép đăng nhập Login với cơ chế Windows hoặc cả hai cơ chế, chúng ta thiết lập ở thuộc tính của Server như sau (Hình 9.6)

Ở mục Server Authentication:



**Hình 9.6. Thiết lập hai cơ chế bảo mật**

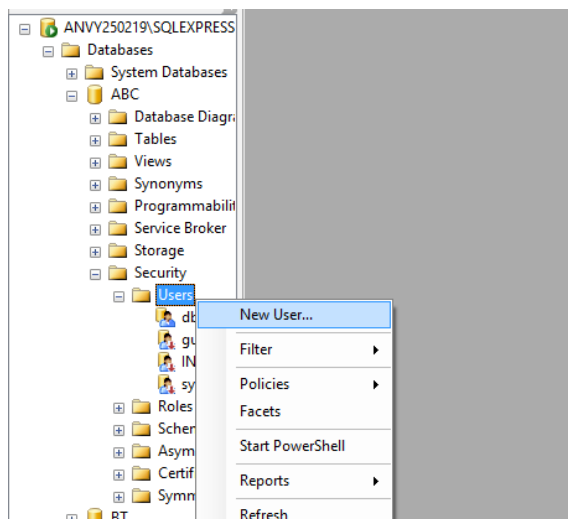
- Chọn Windows Authentication mode: Server chỉ cho phép Login đăng nhập cơ chế Windows.
- Chọn SQL Server and Windows Authentication mode: Server cho phép Login đăng nhập bằng một trong hai cơ chế Windows hoặc SQL.

### 9.5.2. Tạo USER

Khi tạo User, User được tạo sẽ làm việc với một Database cụ thể. Các bước tạo User như sau:

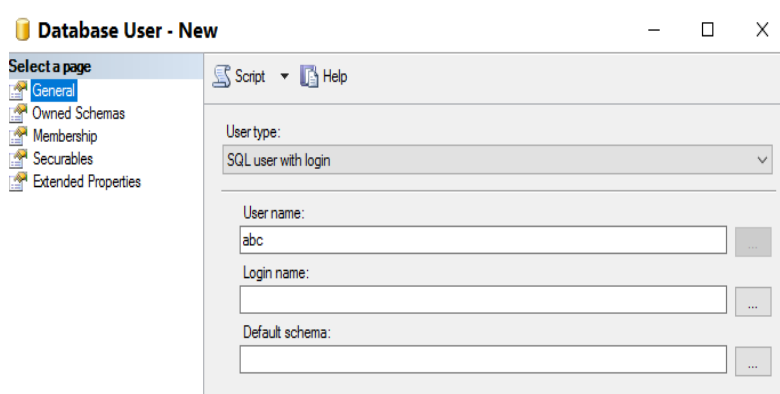
Bước 1: Vào Database muốn tạo User, mở mục Security\ Users, nhấp chuột phải lên mục Users, chọn New Users (Hình 9.7)

Xuất hiện cửa sổ Database Users



**Hình 9.7. Tạo User**

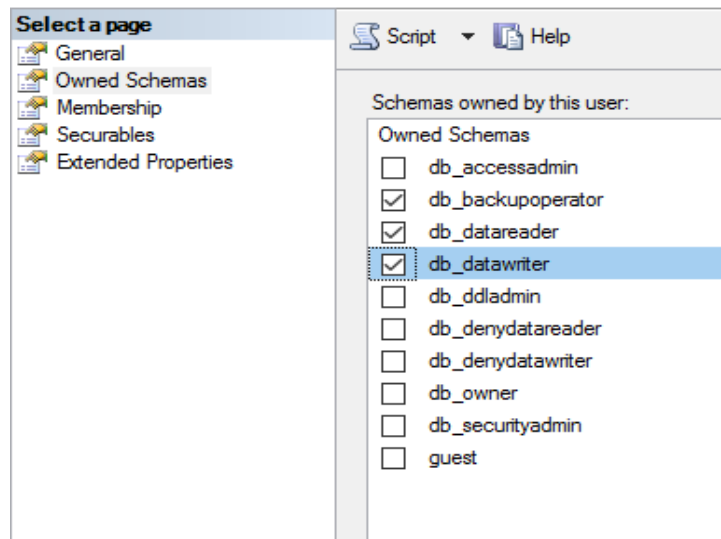
Bước 2: Ở mục General, tại text box User name, điền tên User. (Hình 9.8)



**Hình 9.8. Đặt tên User**

Bước 3: Mở mục Owned Schemas, cấp các quyền thao tác cho User trên Database đó. (Hình 9.9)

Bước 4: Nhấn chọn OK  **Database User - New**



**Hình 9.9. Cấp quyền cho các User**

### **9.5.3. Cấp phát quyền cho người dùng**

#### **Dùng T-SQL:**

*USE <DB name>*

*GRAN <permission name> ON <Object name> TO <User name>*

Giải thích: Lệnh này thực hiện phân quyền *permission name* cho người dùng *User name* trên đối tượng *Object name* thuộc *DB name*.

Ví dụ: Phân quyền db\_owner cho người dùng Troly trên bảng tbl\_diem của database QL\_sinhvien

*USE <QL\_sinhvien>*

*GRAN <db\_owner> ON <tbl\_Diem> TO <Troly>*

Trong quản lý dữ liệu, bảo mật là một việc không thể thiếu, nó ảnh hưởng đến sự sống còn của dữ liệu. SQL Server đã hỗ trợ tốt việc này thông qua các mức bảo mật và công cụ cấp phát quyền người dùng.

## **CÂU HỎI ÔN TẬP CHƯƠNG 9**

1. SQL Server có các mức bảo mật nào, chúng khác nhau như thế nào?
2. Server Role, Database Role là gì?
3. Hãy nêu các bước tạo người dùng.
4. Người dùng có thể có các quyền gì trên cơ sở dữ liệu.

## CHƯƠNG 10. IMPORT/EXPORT DATA VÀ BACK UP/RESTORE DATA

Trong SQL Server, việc sao lưu (Backup database) và phục hồi dữ liệu (Restore database) là một trong những thao tác quan trọng mà người quản trị cơ sở dữ liệu phải thực hiện. Nếu như thao tác sao lưu được thực hiện để lưu dữ liệu và được thực hiện thường xuyên thì thao tác phục hồi dữ liệu chỉ được thực hiện khi nào máy chủ bị sự cố như hư ổ cứng hoặc dữ liệu bị mất do người dùng vô tình hoặc cố ý xoá.

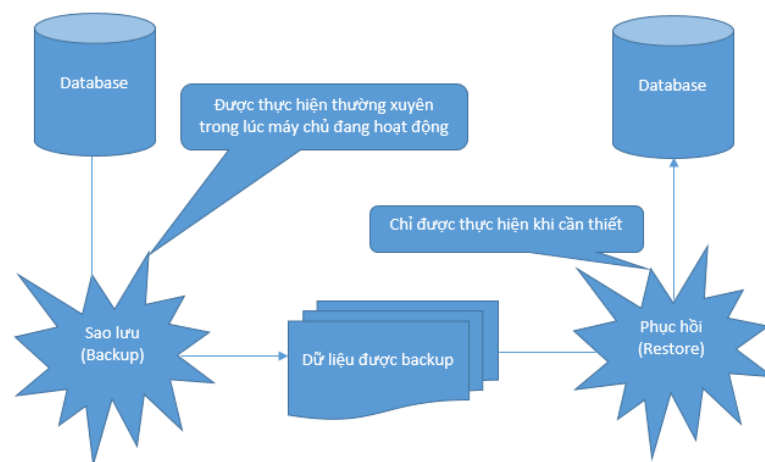
Trong chương này chúng tôi sẽ trình bày chi tiết về kỹ thuật sao lưu và phục hồi dữ liệu. Đồng thời giới thiệu tiện ích Import hay Export dữ liệu để trao đổi dữ liệu giữa các loại cơ sở dữ liệu với nhau trong SQL Server.

### 10.1. Back up Database

#### 10.1.1. Các kiểu Back up

SQL Server cung cấp 3 loại Back up:

- **Full Backup:** Kiểu Backup toàn bộ dữ liệu tại thời điểm thực hiện (thường dùng nhất) cả những user data và database objects như System tables, indexes, user defined tables.
- **Differential Backup:** Kiểu Backup các dữ liệu mới được cập nhật so với lần **full backup** trước đó.
- **Transaction Log Backup:** Backup các *log recor* hiện có trong log file. Nói cách khác, thực hiện sao lưu các *hành động* (các thao tác xảy ra đối với database), không sao lưu dữ liệu. Đồng thời nó cũng cắt bỏ (truncate) log file, loại bỏ các log record vừa được backup ra khỏi log file.



Hình 10.1. Tổng quan về sao lưu và phục hồi dữ liệu

Để giảm thiểu mất dữ liệu khi có sự cố chúng ta cần tăng số lần backup. Tuy nhiên với một database có dung lượng lớn và được cập nhật liên tục, thì việc thực hiện full backup với tần suất cao thì không khả thi, vì như vậy tốn rất nhiều tài nguyên (CPU, I/O). Với kiểu **differential backup** và **transaction log backup**, chúng ta có thể tạo lập các phương án sao lưu thích hợp, đảm bảo dữ liệu được backup thường xuyên hơn mà không chiếm nhiều tài nguyên của hệ thống.

Ví dụ: Kế hoạch chi tiết về sao lưu và khôi phục dữ liệu của người quản trị cơ sở dữ liệu như sau:

- **Full backup:** Một lần mỗi ngày vào 0h sáng.

- **Differential backup:** vào các thời điểm 7h, 12h, 14h, 18h, 22h (5 lần/ngày).

*Differential backup* luôn sao lưu các dữ liệu đã thay đổi kể từ lần **full backup** trước (trong ví dụ trên là các dữ liệu đã thay đổi kể từ 0h), chứ không phải từ lần differential backup trước đó. Vì thế bản backup lúc 12h sẽ bao gồm các dữ liệu đã được sao lưu trong bản backup lúc 7h, bản backup lúc 14h gồm các dữ liệu đã có trong bản 12h,...

- **Transaction log backup:** 15 phút một lần vào các thời điểm 5', 20', 35', và 50' của mỗi giờ (4 lần/giờ).

**Transaction log backup** chỉ sao lưu các log record kể từ lần **transaction log backup** trước đó.

*Giả sử Database bị hỏng vào thời điểm 12h55', bạn cần khôi phục lại database theo trình tự sau:*

**Bước 1:** Khôi phục từ bản full backup gần với thời điểm có sự cố nhất (bản full backup lúc 0h).

**Bước 2:** Khôi phục từ bản differential backup gần với thời điểm có sự cố nhất (bản lúc 12h).

**Bước 3:** Khôi phục tất cả các transaction log backup kể từ sau lần differential backup gần nhất, lần lượt theo trình tự thời gian. Đó là các bản tại các thời điểm 12h5', 12h20', 12h35', và 12h50'.

Bước 1 và 2 đưa database trở lại trạng thái giống như lúc 12h. Ở bước 3, với mỗi lần khôi phục transaction log thì các thao tác chứa trong đó được đem ra thực hiện lại trên database (gọi là log forwarding) và do đó đưa nó về trạng thái gần hơn thời điểm xảy ra sự cố. Như vậy sau khi hoàn tất khôi phục bốn bản transaction log

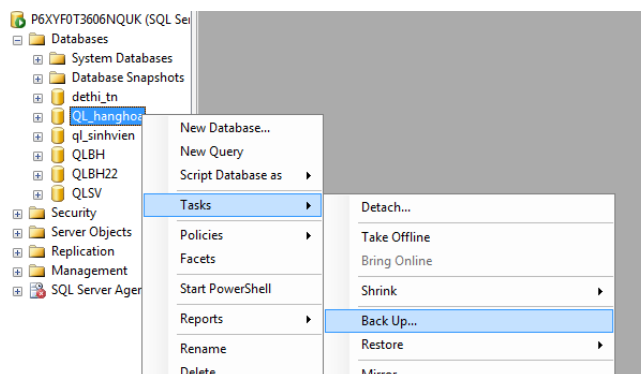
backup thì database sẽ ở vào trạng thái giống như lúc 20h50'. Vì vậy, các thay đổi xảy ra trong 5 phút sau đó (từ 12h50' đến 12h55') đã bị mất.

### 10.1.2. Back up Database

Sao lưu dữ liệu được thực hiện để lưu dữ liệu thêm một bản và đặt ở đâu đó.

*Các bước thực hiện Back up Database bằng giao diện của Microsoft SQL Server:*

**Bước 1:** Trong cửa sổ Object Explore, nhấn chuột phải vào Database muốn Back up, chọn mục Task, chọn tiếp mục Back up (hình 10.2)

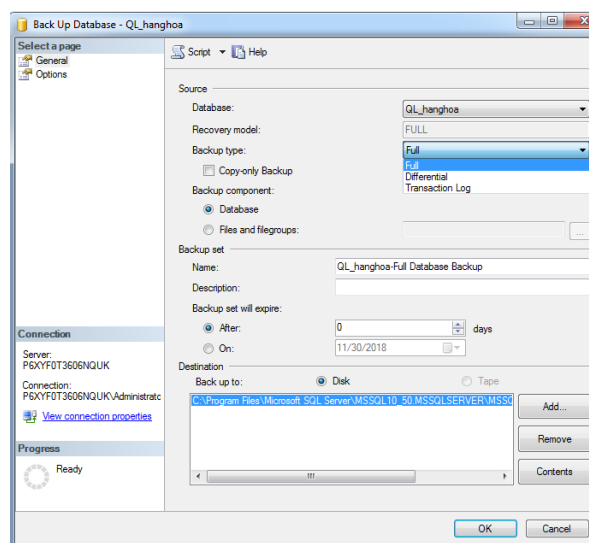


**Hình 10.2. Back up database trên giao diện**

**Bước 2:** Trong hộp thoại Back up Database:

- Chọn tên Database cần Backup
- Chọn kiểu Backup (Backup type): Full/ Differential/ Transaction Log
- Chỉ ra đường dẫn đến thư mục chứa file Backup

**Bước 3:** Nhấn chọn nút lệnh OK (Hình 10.3)



**Hình 10.3 Giao diện thực hiện Back up database**

*Các bước thực hiện Back up Database bằng câu lệnh:*

- Full/ Database:

*Backup Database <DB name> To Disk = 'Path + File name'*

- Differential Database:

*Backup Database <DB name> To Disk = 'Path + File name' With Different*

- Transactional Log

*Backup Log <DB name> To Disk = 'Path + File name'*

Ví dụ: Sao lưu database QL\_sinhvien vào đĩa D:

Backup Database QL\_Sinhvien To Disk = 'D:\QLSV\_full.bak'

Backup Database QL\_Sinhvien To Disk = 'D:\QLSV\_diff.bak'  
With different

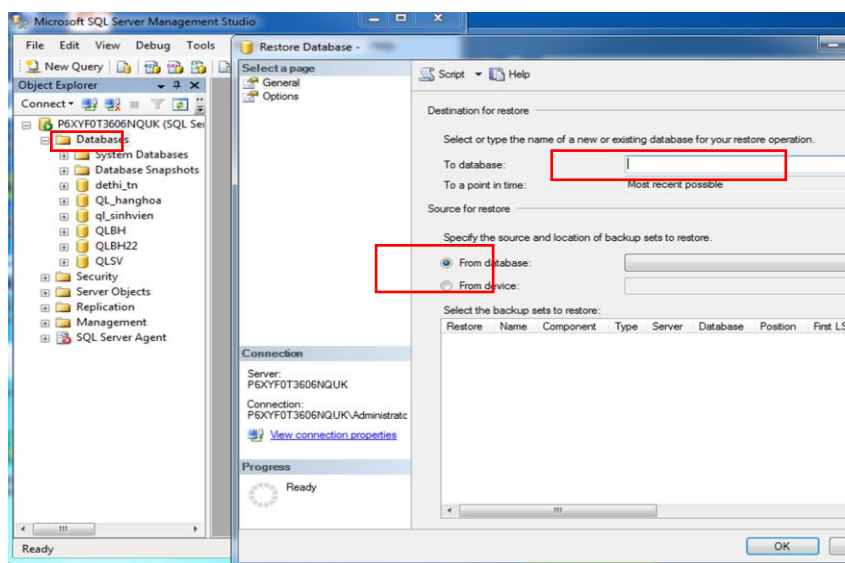
Backup Log QL\_Sinhvien To Disk = 'D:\QLSV\_log.trn'

## 10.2. Restore Database

Sao lưu dữ liệu là việc được thực hiện thường xuyên, còn việc phục hồi dữ liệu là thao tác chỉ được thực hiện khi dữ liệu bị mất do người dùng vô tình hoặc cố ý xóa, hoặc máy chủ gặp sự cố, ổ đĩa bị hỏng,... Ngoài ra, việc phục hồi dữ liệu còn được thực hiện khi có nhu cầu sao chép dữ liệu từ máy này sang máy khác.

*Các bước Restore Database:*

**Bước 1:** Trong cửa sổ Object Explore, nhấn chuột phải vào mục Database, chọn mục Restore Database nhận được hộp thoại như hình 10.4



**Hình 10.4. Restore database từ cửa sổ Explore**

**Bước 2:** Trong hộp text To Database điền tên Database (tên mới) hoặc chọn tên đã có để phục hồi.

**Bước 3:** Ghi rõ đường dẫn đến file chứa Database đã back up vào hộp text From Database hoặc thiết bị chứa file đó.

**Bước 4:** Nhấn chọn OK

Hoặc dùng T-SQL để Restore Database

```
RESTORE DATABASE <DB Name> FROM DISK= 'Path + file name'  
[ WITH RECOVERY / NORECOVERY / STANDBY]
```

Trong đó:

- Recovery là tùy chọn mặc định của câu lệnh Restore, nếu không chỉ rõ NoRecovery hay Standby thì mặc định Recovery sẽ được thực hiện.
- Restore với tùy chọn NoRecovery sẽ cho Database không thể sử dụng sau khi phục hồi, không thực hiện Rollback các Uncommitted transaction và cho phép phục hồi bằng các bản sao lưu kế tiếp.
- Restore với tùy chọn Standby sẽ cho Database có thể dùng sau khi phục hồi nhưng chỉ ở chế độ chỉ đọc (ReadOnly), Rollback các Uncommitted transaction và cho phép phục hồi bằng các bản sao lưu kế tiếp.

Tóm lại, sao lưu và phục hồi là một việc cần thiết trong quá trình quản lý dữ liệu nhằm đảm bảo an toàn cho dữ liệu.

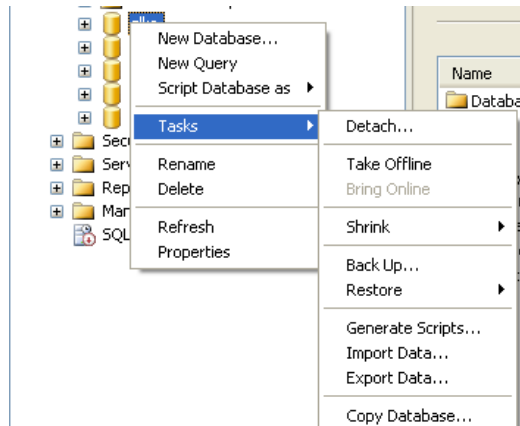
### 10.3. IMPORT DATA

Một trong những cách trao đổi dữ liệu giữa các loại cơ sở dữ liệu với nhau là sử dụng tiện ích Import hay Export dữ liệu từ định dạng này sang định dạng khác. SQL Server cung cấp hai chức năng chính là Import dùng để nhập dữ liệu vào cơ sở dữ liệu SQL Server các loại cơ sở dữ liệu khác và Export dùng để xuất dữ liệu từ SQL Server ra các loại cơ sở dữ liệu khác.

Việc đầu tiên ta phải xác định cơ sở dữ liệu nguồn thuộc loại cơ sở dữ liệu nào để chọn và dĩ nhiên cơ sở dữ liệu đích chính là SQL Server 2008

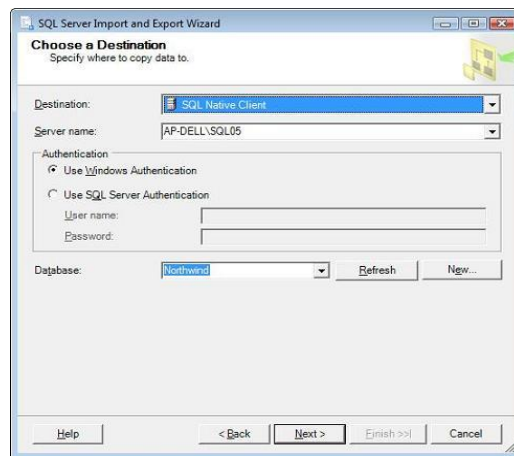
- Nhấp chuột phải tại Database chọn Tasks -> Import data (Hình 10.5)





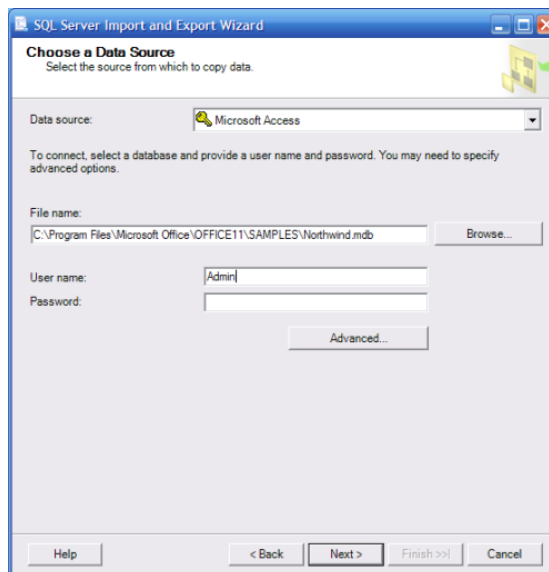
**Hình 10.5. Chức năng Import Data trong giao diện**

- Chọn Database -> Next

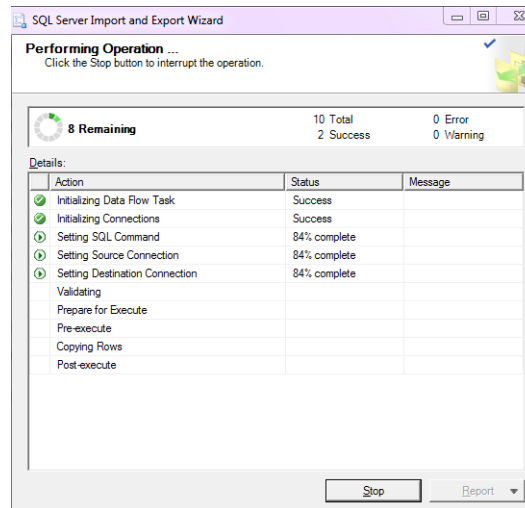


**Hình 10.6. Chọn CSDL muốn Import**

- Chọn Data Source -> Next



**Hình 10.7. Chọn CSDL nguồn**

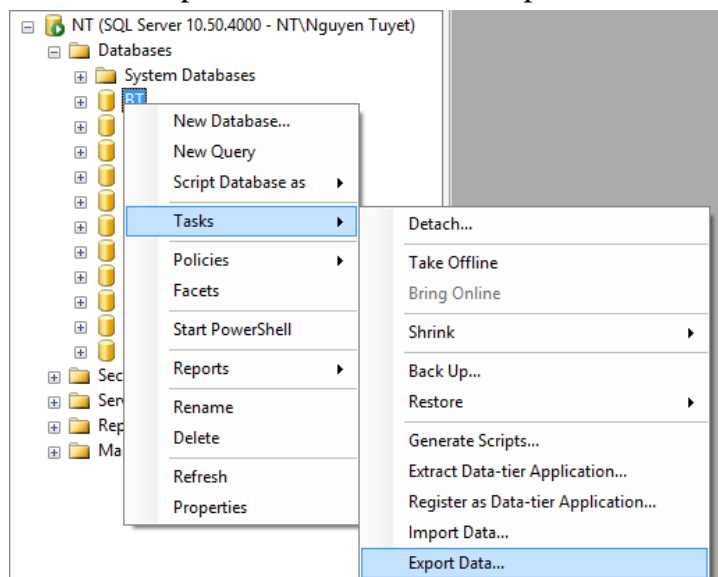


**Hình 10.8. Giao diện hoàn thành việc Import Data**

#### 10.4. EXPORT DATA

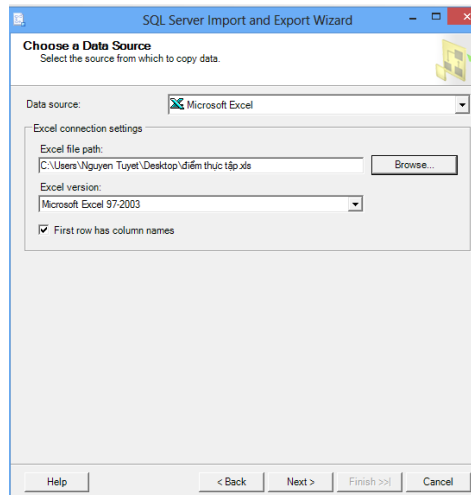
Quá trình phát triển ứng dụng có kết nối với SQL Server, đôi khi chúng ta cần 1 database thao tác trên nhiều máy. Việc tạo lại database hết sức vất vả và mất nhiều thời gian. SQL Server có hỗ trợ chức năng xuất database dưới dạng file cơ sở dữ liệu mong muốn.

Bước 1: Nhấn chuột phải tại Database cần Export -> Tasks -> Export Data



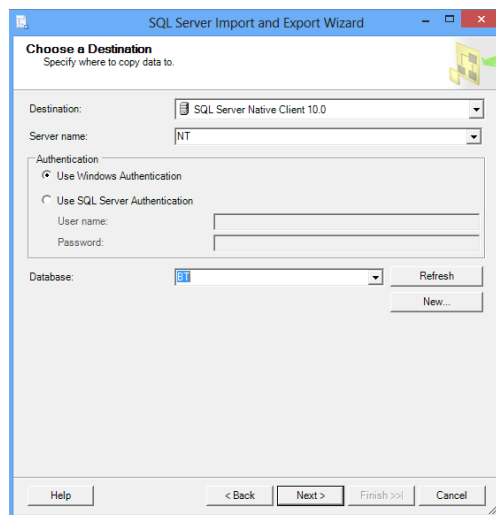
**Hình 10.9. Chức năng Export Data trong giao diện**

Bước 2: Chọn Data Source -> Next



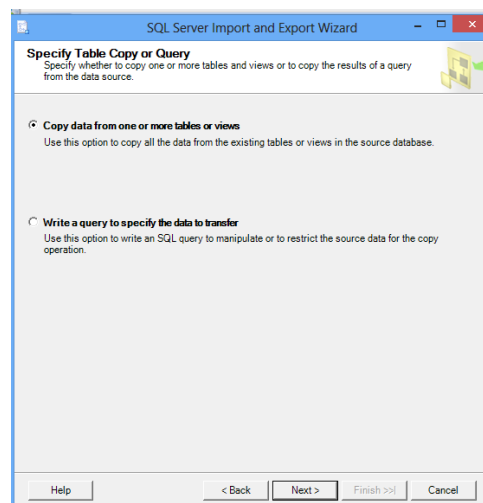
**Hình 10.10. Chọn Data Source**

Bước 3: Chọn Database muốn kết xuất -> Next



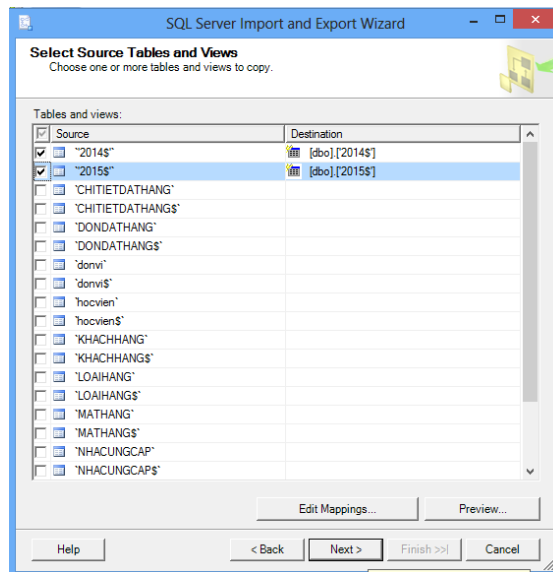
**Hình 10.11. Chọn Database trong SQL Server**

Bước 4. Chọn Copy data từ Table hoặc View -> Next



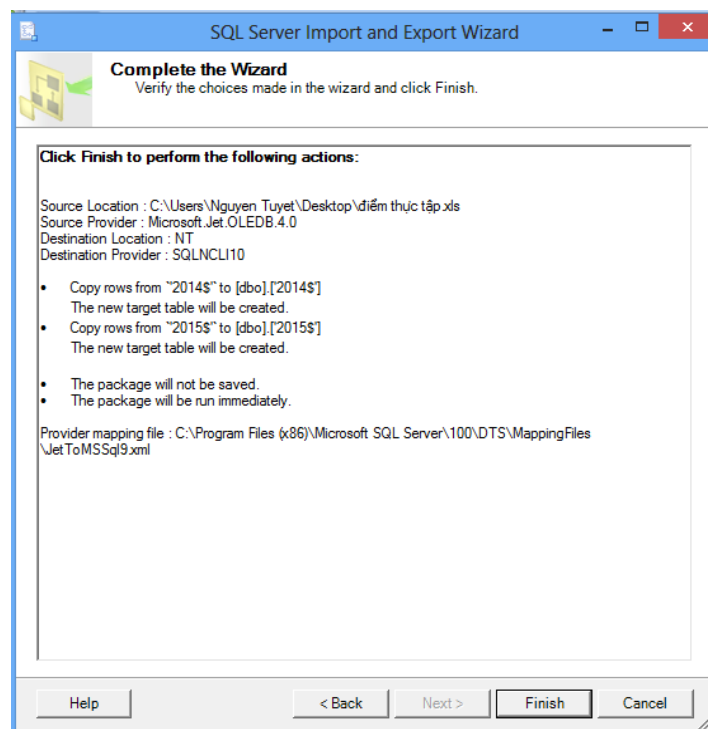
**Hình 10.12. Chọn xuất ra file từ Table hoặc View**

Bước 5: Lựa chọn Table hoặc View muốn kết xuất -> Next



**Hình 10.13.** Chọn Table hoặc View

Bước 6: Nhấn Finish để hoàn thành thao tác



**Hình 10.14.** Giao diện hoàn thành thao tác Export Data

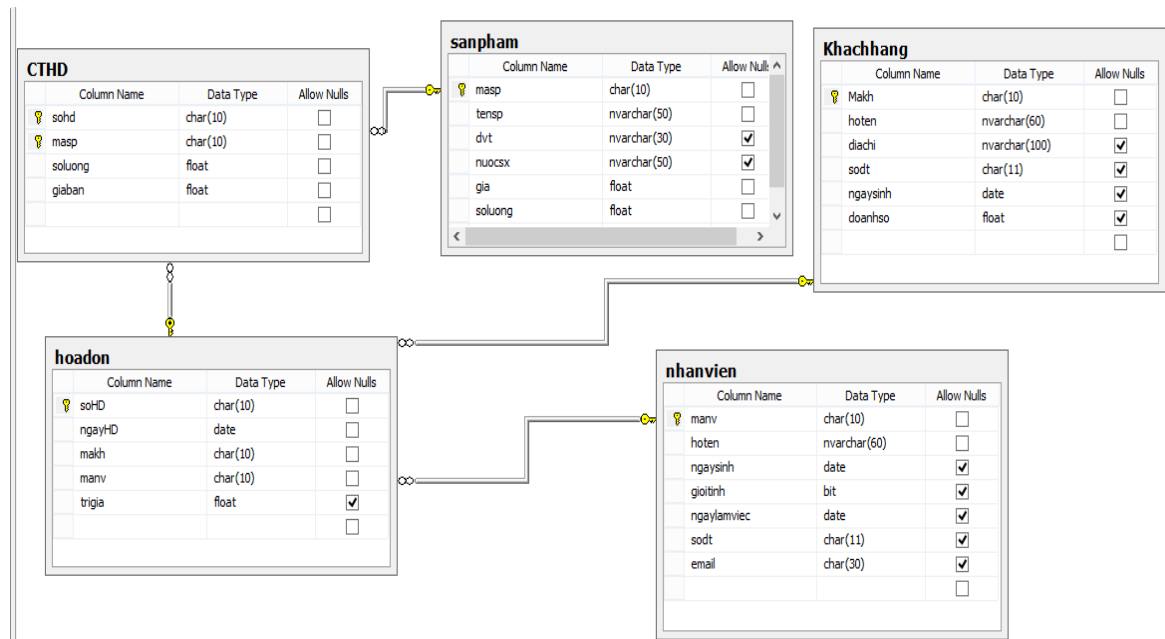
## CÂU HỎI ÔN TẬP CHƯƠNG 10

1. Sao lưu và phục hồi nhằm mục đích gì? Khi nào cần sao lưu?
2. Có các kiểu Sao lưu dữ liệu nào?
3. Nêu các bước sao lưu, phục hồi dữ liệu?

## BÀI THỰC HÀNH

### Bài thực hành số 1: Tạo cơ sở dữ liệu (3 tiết)

Thực hành **tạo cơ sở dữ liệu** QLBANHANG sau được sử dụng để quản lý công tác bán hàng trong một công ty nhỏ.



trong đó:

- Bảng SANPHAM lưu trữ dữ liệu về các mặt hàng hiện có trong công ty.
- Bảng NHANVIEN có dữ liệu là các thông tin về nhân viên làm việc trong công ty
- Bảng KHACHHANG được sử dụng để lưu trữ các thông tin về khách hàng của công ty.
- Bảng HOADON: Mỗi một hóa đơn phải do một nhân viên của công ty lập
- Thông tin chi tiết của hóa đơn (đặt mua mặt hàng gì, số lượng, giá cả,...) được lưu trữ trong bảng CTHD.

#### 1. Thêm các ràng buộc:

- a. Bảng Nhanvien: ngaysinh chỉ nhận nhân viên trên 18 tuổi trở lên.
- b. Bảng Khachhang: trường dienthoai, email không được trùng dữ liệu
- c. Bảng Sanpham: trường soluong  $\geq 50$

#### 2. Nhập dữ liệu đầy đủ cho các bảng trên

## **Bài thực hành số 2: Các thao tác trên CSDL (3 tiết)**

Yêu cầu: Thực hành *truy vấn dữ liệu*

Sử dụng CSDL QLBAHANG để thực hiện các truy vấn sau:

1. Mã sản phẩm, tên sản phẩm và số lượng sản phẩm hiện có trong công ty
2. Họ tên, địa chỉ và ngày bắt đầu làm việc của các nhân viên trong cty
3. Địa chỉ, điện thoại của các khách hàng ở *Quy Nhơn*.
4. Mã và tên sản phẩm có giá trị lớn hơn 100000 và số lượng hiện có ít hơn 50
5. Cho biết danh sách khách hàng chưa từng mua hàng của công ty
6. Những sản phẩm nào được sản xuất ở *Nhật Bản*.
7. Những khách hàng nào đã mua *sữa hộp* của công ty.
8. Đơn hàng số 1 do ai đặt và do nhân viên nào lập, trị giá là bao nhiêu?
9. Hãy cho biết có những nhân viên nào lại chính là khách hàng của công ty (có cùng sodt)
10. Liệt kê danh sách những nhân viên có cùng ngày sinh.

## **Bài thực hành số 3: Câu lệnh SQL (3 tiết)**

Yêu cầu: Thực hành *truy vấn dữ liệu*

Sử dụng CSDL QLBAHANG để thực hiện các truy vấn sau:

1. Những sản phẩm nào chưa từng được khách hàng mua tính đến thời điểm này.
2. Những nhân viên nào chưa từng lập 1 hóa đơn tính đến thời điểm này.
3. Những nhân viên nào của công ty có thâm niên cao nhất
4. Tổng số tiền mà khách hàng phải trả cho mỗi hóa đơn là bao nhiêu
5. Trong năm 2018 những sản phẩm nào đặt mua đúng một lần.
6. Tính trị giá của mỗi hóa đơn.
7. Tính doanh số mua hàng của mỗi khách hàng.
8. Mỗi nhân viên của công ty đã lập bao nhiêu hóa đơn (nếu chưa từng lập hóa đơn nào thì cho kết quả là 0)
9. Thống kê số lượng hàng tồn của công ty
10. Tính tổng số tiền lời mà công ty thu được từ mỗi mặt hàng trong năm 2018

#### **Bài thực hành số 4: Câu lệnh SQL (tiếp theo – 3 tiết)**

Yêu cầu: Thực hành truy vấn nâng cao.

1. Nhân viên nào của công ty bán được số lượng hàng nhiều nhất và số lượng hàng bán được của những nhân viên này là bao nhiêu
2. Đơn đặt hàng nào có số lượng hàng được đặt mua ít nhất
3. Số tiền nhiều nhất mà khách hàng đã từng bỏ ra để đặt hàng trong các đơn đặt hàng là bao nhiêu
4. Mỗi đơn đặt hàng đặt mua những mặt hàng nào và tổng số tiền của đơn đặt hàng
5. Mỗi một loại hàng bao gồm những mặt hàng nào, tổng số lượng của mỗi loại và tổng số lượng của tất cả các mặt hàng hiện có trong công ty

#### **Bài thực hành số 5: View (3 tiết)**

Yêu cầu: Thực hành tạo, sửa, xóa khung nhìn.

1. *Nhật Bản* đã cung cấp những mặt hàng nào
2. Sản phẩm *Điện thoại di động* do nước nào sản xuất
3. Những khách hàng nào có doanh số mua hàng cao nhất
4. Đơn hàng số 1 do ai đặt và do nhân viên nào lập, thời gian và trị giá bao nhiêu
5. Hãy cho biết tổng số lượng sản phẩm do Việt Nam sản xuất
6. Trong đơn hàng số 3 đặt mua những mặt hàng nào và tổng số tiền là bao nhiêu
7. Hãy cho biết có những khách hàng nào chưa có số điện thoại
8. Trong công ty có những nhân viên nào có cùng ngày sinh
9. Những nhân viên nào chưa xuất được hóa đơn trong 30 ngày qua
10. Cho biết những sản phẩm có số lượng ít hơn 10

#### **Bài thực hành số 6: Thủ tục (3 tiết)**

Yêu cầu: Thực hành tạo thủ tục thường trú

1. Viết thủ tục để xem doanh số mua hàng của khách hàng có mã X nào đó (X là tham số của thủ tục)
2. Viết thủ tục để xem masp và tensp của các sản phẩm có giá trị lớn hơn x và số lượng hiện có ít hơn y (x,y là tham số)
3. Viết thủ tục cho biết nhân viên X đã lập bao nhiêu hóa đơn (tham số x là manv)

- Viết thủ tục cho biết hóa đơn  $X$  do nhân viên nào lập, có bao nhiêu sản phẩm trên hóa đơn đó (tham số  $X$  là maHD)
- Viết thủ tục để xem những sản phẩm nào đã được mua với số lượng nhiều nhất

### **Bài thực hành số 7: Hàm (3 tiết)**

Yêu cầu: Thực hành tạo hàm trả về giá trị, hàm trả về biến bảng. Vận dụng gọi hàm trong thao tác với CSDL.

- Viết hàm trả về tổng tiền (trị giá) mà khách hàng phải trả trên mỗi hóa đơn
- Viết hàm trả về tình trạng bán của các sản phẩm. Nếu soluong các sản phẩm (trong bảng CTHD)  $>100$  thì bán chạy, ngược lại là bán chậm.
- Viết hàm trả về một bảng gồm các thông tin Masp, tensp và soluong của các sản phẩm hiện có trong công ty.
- Viết hàm trả về một bảng gồm danh sách khách hàng có doanh số mua hàng cao nhất tính đến thời điểm này.
- Viết hàm trả về số lượng sản phẩm bán chạy nhất trong tháng 6/2019

### **Bài thực hành số 8: Trigger(3 tiết)**

Yêu cầu: Thực hành tạo Trigger

- Tạo Trigger ràng buộc soluong trong bảng Sanpham  $\geq$  soluong nhập vào bảng CTHD (cùng mã hàng)
- Tạo Trigger không cho phép nhập vào bảng Sanpham những mặt hàng có số lượng  $\leq 10$
- Viết trigger khi sửa giá cho một (hoặc nhiều) masp thì in ra:
  - Danh sách các masp vừa được sửa giá.
  - Danh sách các masp vừa được sửa cùng giá cũ và giá mới
- Tạo Trigger không cho phép xóa những nhân viên có thâm niên trên 10 năm
- Viết trigger sao cho khi nhập soluong mua hàng ở bảng CTHD thì Trigia ở bảng Hoadon và doanhso ở bảng Khachhang tự động cập nhật theo.
- Tạo Trigger có nhiệm vụ sửa lại **giaban** trong bảng CTHD sao cho **giaban=1.3\*gia** (gia trong bảng Sanpham)

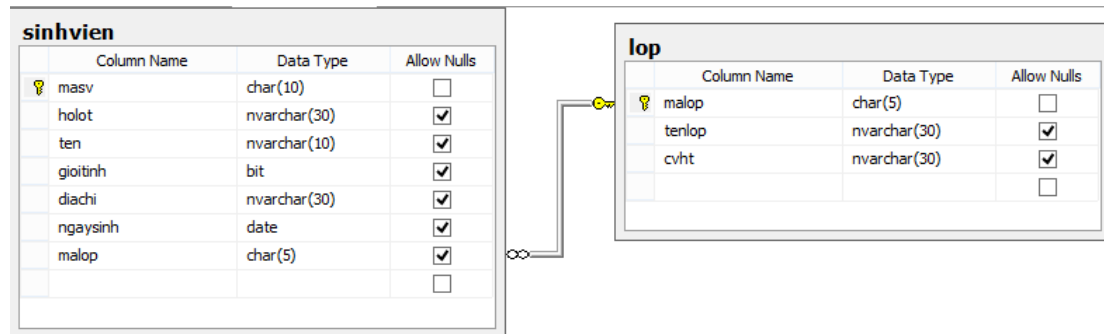
### **Bài thực hành số 9: Bào mật và phân quyền người dùng (3 tiết)**

Tạo và phân quyền cho người dùng. Đăng nhập SQL từ người dùng vừa tạo.



## Bài thực hành số 10: Bài tập tổng hợp (3 tiết)

### 1. Tạo CSDL QLSV



malop	tenlop	cvht
10537	CNTT37	Nguyễn Thị Bình
11339	SP Tin39	Trần Văn Tâm
10536	CNTT36	Lê Ngọc Nữ
10538	CNTT38	Hồ Kim Hùng
11237	TH Toán37	Nguyễn Văn Dũng

masv	holot	ten	gioitinh	diachi	ngaysinh	malop
sv1	Hồ Văn	Nam	True	Quy Nhơn	1998-04-12	10537
sv2	Lê Thị	Nguyệt	False	Quảng Ngãi	1990-02-11	NULL
sv3	Nguyễn Minh	Tùng	True	Quy Nhơn	1990-03-04	10538
sv4	Trần Hạ	Huyền	False	Gia Lai	1991-06-07	11339
sv5	Nguyễn Thị	Lan	False	Quy Nhơn	1990-04-09	11237

☞ Trường NGAYSINH chỉ nhập những sinh viên từ 18 tuổi trở lên.

☞ Tạo liên kết khóa ngoài(MALOP của bảng SINHVIEN tham chiếu đến bảng MALOP của bảng LOP)

☞ Nhập đầy đủ dữ liệu cho các bảng.

### 2. Dùng câu lệnh SQL để tạo các truy vấn sau:

- Xem danh sách học sinh học lớp CNTT38, có địa chỉ Quy Nhơn.
- Thống kê số sinh viên theo từng lớp.
- Tạo view SVCNTT chứa danh sách sinh viên học các lớp CNTT (CNTT37, CNTT36,...)

3. Viết thủ tục để xem danh sách sinh viên có độ tuổi trong một khoảng [a...b] nào đó (a,b là tham số)

4. Viết trigger không cho phép xóa những sinh viên học CNTT mà có địa chỉ Quy Nhơn.

## TÀI LIỆU THAM KHẢO

1. Nguyễn Kim Anh, Nguyên lý của các hệ cơ sở dữ liệu, NXB ĐHQG HN, 2004.
2. Nguyễn Xuân Huy, Lê Hoài Bắc, Bài tập Cơ sở dữ liệu, NXB Thống kê, 2003.
3. Jeffrey D. Ullman, The principles of database and knowledge base system Vol. 1, 2, Computer Science Press, 1989.
4. Vũ Tuyết Trinh, *Microsoft SQL Server 2008*, ĐH Bách Khoa Hà Nội