TRƯỜNG CAO ĐẲNG NGHỀ CÔNG NGHIỆP HÀ NỘI
Chủ biên: Nguyễn Thị Nhung
Đồng tác giả: Vũ Thị Kim Phượng

GIÁO TRÌNH
QUẢN TRỊ CƠ SỞ DỮ LIỆU NÂNG CAO MS SQL SERVER
(Lưu hành nội bộ)

Hà Nội năm 2013

Tuyên bố bản quyền

Giáo trình này sử dụng làm tài liệu giảng dạy nội bộ trong trường cao đẳng nghề Công nghiệp Hà Nội

Trường Cao đẳng nghề Công nghiệp Hà Nội không sử dụng và không cho phép bất kỳ cá nhân hay tổ chức nào sử dụng giáo trình này với mục đích kinh doanh.

Mọi trích dẫn, sử dụng giáo trình này với mục đích khác hay ở nơi khác đều phải được sự đồng ý bằng văn bản của trường Cao đẳng nghề Công nghiệp Hà Nội

CHƯƠNG I: GIỚI THIỆU LỊCH SỬ PHÁT TRIỂN

I. Giới thiệu chung về lịch sử phát triển của SQL Server .Sự quan trọng và cần thiết của chúng trong lập trình ứng dụng

1.1. Định nghĩa hệ quản trị cơ sở dữ liệu (HQTCSDL)

HQTCSDL(Database management system) là tập hợp các phần mềm cho phép định nghĩa các cấu trúc để lưu trữ thông tin trên máy, nhập dữ liệu, thao tác trên các dữ liệu đảm bảo sự an toàn và bí mật của dữ liệu.

Định nghĩa cấu trúc: Định nghĩa cấu trúc CSDL bao gồm việc xác định kiểu dữ liệu, cấu trúc và những ràng buộc cho dữ liệu được lưu trữ trong CSDL.

Nhập dữ liệu: Là việc lưu trữ dữ liệu vào các thiết bị lưu trữ trung gian được điều khiển bằng HQTCSDL.

Thao tác dữ liệu: thao tác trên CSDL bao gồm những chức năng như truy xuất cơ sở dữ liệu để tìm kiếm thông tin cần thiết, cập nhật cơ sở dữ liệu và tổng hợp những báo cáo từ dữ liệu.

1.2 Lịch sử phát triển

Tháng 6 năm 1970 Mô hình quan hệ cho dữ liệu dùng trong ngân hàng dữ liệu chia sẻ có khối lượng lớn của tiến sĩ Edgar F.Codd, một mô hình đã được chấp nhận rộng rãi là mô hình tiêu chuẩn dùng cho hệ thống quản lý cơ sở dữ liệu quan hệ.

Giữa những năm 1970, một nhóm các nhà phát triển tại trung tâm nghiên cứu của IBM tại San Jose phát triển hệ thống cơ sở dữ liệu "Hệ thống R" dựa trên mô hình của Codd viết tắt là "SEQUEL" (Structured English Query Language – Ngôn ngữ truy vấn tiếng anh có cấu trúc). Sau này, tên viết tắt SEQUEL được rút gọn thành SQL để tránh việc tranh chấp nhãn hiệu (từ SEQUEL đã được một công ty máy bay của UK là Hawker-Siddeley đăng ký). Tuy SQL bị ảnh hưởng bởi công trình của tiến sĩ Codd nhưng nó không do tiến sĩ Codd thiết kế mà lại do Donald D.Chamberlin và Raymond F. Boyce tại IBM thiết kế.

Đến năm 1974 phiên bản CSDL phi thương mại, không hỗ trợ SQL đầu tiên được ra đời.

Năm 1978, IBM đã tổ chức cuộc thử nghiệm và đã chứng minh được sự có ích và tính thực tiễn của hệ thống (điều này còn chứng minh được sự thành công của IBM). Dựa vào kết quả đó IBM bắt đầu phát triển các sản phẩm thương mại bổ sung thêm SQL dựa trên nguyên mẫu Hệ thống R bao gồm:

+ System/38: được công bố năm 1978 và được thương mại hóa tháng 8/1979.
+ SQL/DS: được giới thiệu vào năm 1981
+ DB2: năm 1983.

Đến năm 1979 Relational Software, Inc (bây giờ là Oracle Corporation) giới thiệu Oracle V2 (Phiên bản 2), phiên bản thương mại đầu tiên hỗ trợ SQL cho máy tính VAX.

(*Chú ý*:Trong Oracle tất cả các chương trình và người sử dụng phải sử dụng SQL để truy nhập vào dữ liệu trong CSDL của Oracle. Các chương trình ứng dụng và các công cụ Oracle cho phép người sử dụng truy nhập tới CSDL mà không cần sử dụng trực tiếp SQL. Nhưng những ứng dụng đó khi chạy phải sử dụng SQL.)

Năm 1986, SQL được thừa nhận là tiêu chuẩn của ANSI (American National Standards Institute) và năm 1987 SQL được công nhận là chuẩn ISO. Có rất nhiều phiên bản khác nhau của SQL được dùng cho các hệ thống cơ sở dữ liệu hiện nay.

SQL Server của Microsoft đang dùng phiên bản Transact-SQL hay T-SQL.

Microsoft bắt đầu xây dựng SQL Server (một sản phẩm cơ sở dữ liệu sử dụng ngôn ngữ SQL) với Sybase để dùng cho OS/2. Khi Microsoft rời bỏ OS/2 để quan tâm đến hệ điều hành mạng mới của hang, Windows NT Server, thì họ quyết định tiếp tục phát triển bộ Engine SQL Server dùng cho Windows NT Server. Sản phẩm đạt được là Microsoft SQL 4.2, sau được nâng cấp thành 4.21.

Sau khi Microsoft và Sybase tách riêng thì Microsoft phát triển SQL server 6.0, tiếp đó là 6.5. Sau đó Microsoft đã cải tiến và hầu như viết lại một engine mới cho SQL Server 7.0. Cho nên có thể nói từ version 6.5 lên 7.0 là một bước nhảy vọt. Có một số đặc tính của SQL Server 7.0 không tương thích với version 6.5. Trong khi đó từ version 7.0 lên SQL Server 2000 thì những cải tiến chủ yếu là mở rộng các tính năng về Web và làm cho SQL Server 2000 đáng tin cậy hơn. Và chính SQL Server 2000 là phiên bản đánh dấu tầm quan trọng của SQL Server Tiếp đó là phiên bản SQL Server 2005 và mới nhất là SQL Server 2008.

1.3.Giới thiệu SQL Server 2008

Microsoft SQL Server 2008 là một hệ quản trị CSDL quan hệ (RDBMS), cung cấp cách tổ chức dữ liệu bằng cách lưu chúng vào các bảng.

1.3.1 Các phiên bản của SQL Server 2008

SQL Server 2008 có các phiên bản khác nhau, tùy thuộc vào yêu cầu thực thi và cấu hình tại chế độ chạy thực. SQL Server chia thành các phiên bản sau:

- SQL Server 2008 Enterprise Edition: chứa đầy đủ các đặc trưng của SQL Server và có thể chạy tốt trên hệ thống lên đến 32 CPUs và 64 GB RAM. Thêm vào đó nó có các dịch vụ giúp cho việc phân tích dữ liệu rất hiệu quả (Analysis Services).
- SQL Server 2008 Standard Edition: Rất thích hợp cho các công ty vừa và nhỏ vì giá thành rẻ hơn nhiều so với Enterprise Edition, nhưng lại bị giới hạn một số chức năng cao cấp khác, edition này có thể chạy tốt trên hệ thống lên đến 4 CPU và 2 GB RAM.
- SQL Server 20008Personal Edition:vđược tối ưu hóa để chạy trên PC nên có thể cài đặt trên hầu hết các phiên bản của windows, kể cả Windows 98.
- SQL Server Developer Edition: Có đầy đủ các tính năng của Enterprise Edition nhưng được chế tạo đặc biệt như giới hạn số lượng người kết nối vào Server cùng một

lúc.... Đây là edition mà các bạn muốn học SQL Server cần có. Edition này có thể cài trên Windows 2000 Professional hay Win NT Workstation.

- SQL Server 2008 Desktop Engine: Đây chỉ là một engine chạy trên desktop và không có user interface (giao diện). Thích hợp cho việc triển khai ứng dụng ở máy client. Kích thước cơ sở dữ liệu bị giới hạn khoảng 2 GB.
- SQL Server 2008 Windows CE Edition: Dùng cho các ứng dụng chạy trên Windows CE

Trong tất cả các phiên bản trên, bản SQL Server 2000 Enterprise Edition được ứng dụng rộng rãi do hỗ trợ đầy đủ và mạnh mẽ về khả năng đáp ứng và độ tin cậy.

1.3.2 Tìm hiểu các đặc trưng của SQL Server 2008

SQL Server 2008 bao gồm một số đặc trưng tạo nên một Hệ quản trị CSDL đáp ứng được yêu cầu rất cao trong thực thi CSDL.

Dễ cài đặt (Easy Installation): SQL Server cung cấp các công cụ quản trị và phát triển để cho người sử dụng dễ dàng cài đặt, sử dụng và quản lý hệ thống.

Tích hợp với Internet (Integration with Internet): SQL Server 2008 database engine hỗ trợ XML. Nó được tối ưu để có thể chạy trên môi trường cơ sở dữ liệu rất lớn (Very Large Database Environment) lên đến Tera-Byte và có thể phục vụ cùng lúc cho hàng ngàn user. Mô hình lập trình (programming model) SQL Server 2008 được tích hợp với kiến trúc Windows DNA trợ giúp cho phát triển ứng dụng Web. Nó cũng hỗ trợ một số đặc tính khác như English Query để người phát triển hệ thống có thể truy vấn dữ liệu thân thiện hơn. Và Microsoft Search Services cung cấp khả năng tìm kiếm rất mạnh, đặc biệt thích hợp cho phát triển ứng dụng Web.

Hỗ trợ kiến trúc Client/Server(Supports Client/Server model): Ứng dụng có thể chạy trên Client, truy cập dữ liệu được lưu trữ trên Server. Server có nhiệm vụ xử lý các yêu cầu và trả lại kết quả cho Client.

Tương thích với nhiều hệ điều hành (Operating System Compatibility):
Có thể cài đặt trên hầu hết các hệ điều hành của Microsoft (danh sách chi tiết kèm theo). Chú ý khi cài đặt trên Windows NT Server 4, bạn phải chạy thêm Service Pack 5(SP5).

Mềm dẻo và khả năng dễ dùng: Đây là phiên bản cơ sở dữ liệu có thể làm việc trên nhiều hệ thống khác nhau từ máy tính xách tay cài đặt hệ điều hành Windows 98 đến máy tính server cài đặt phiên bản Windows 2008 Data Center.

Thích hợp với nhiều giao thức: SQL Server 2008 hỗ trợ hầu hết những giao thức thông dụng như AppleTalk, TCP/IP.

Hỗ trợ việc nhân bản dữ liệu (Data Replication Support): tức là có hai hay nhiều bản sao của CSDL được đồng bộ để những thay đổi trên một bản sẽ được cập nhật vào các bản khác.

Tìm kiếm (Full-Text): tìm kiếm Full-Text cho phép tìm kiếm theo các kí tự. Nó cũng có thể tìm kiếm theo từ hoàn chỉnh hay cụm từ. Indexing wizard tạo index trên một bảng nhất định. Wizard này có thể tìm thấy trong Enterprise Manager. Nó chứa tất cả các dữ liệu cần thiết để tìm kiếm từ/cụm từ.

Sách hướng dẫn trực tuyến (Books Online): books online là một thành phần thêm vào, nó tốn không gian trên server. Trợ giúp dưới dạng một quyển sách giúp cho việc tìm kiếm theo bất kì chủ đề nào rất dễ dàng.

Kho dữ liệu (Data Warehousing): SQL server cung cấp một vài công cụ để xây dựng kho dữ liệu. Sử dụng DTS designer, bạn có thể định nghĩa các bước thực hiện, luồng công việc và chuyển đổi dữ liệu để xây dựng kho dữ liệu từ nhiều nguồn dữ liệu khác nhau. SqL server cũng cung cấp nhiều công cụ để phân tích dữ liệu dựa trên các câu hỏi bằng tiếng anh. Dữ liệu được lấy ra và phân tích được dùng trong quá trình phân tích dữ liệu trực tuyến.

Cài đặt SQL Server 2008

Trước khi cài đặt, bạn phải đảm bảo các yêu cầu về phần mềm và phần cứng được đáp ứng.

- Yêu cầu cấu hình phần cứng để cài đặt SQL Server 2008

Thành phần phần cứng	**Yêu cầu**
Bộ vi xử lý	Intel compatible 32bit CPU(166MHZ hoặc cao hơn)
Dung lượng đĩa	Cài đặt tối thiểu 95MB, cài đặt đầy đủ 270MB
Card mạng	Cần thiết nếu máy trạm cần kết nối tới máy chủ
RAM	128 MB
CD-ROM	Cần thiết để cài đặt từ CD

- Yêu cầu phần mềm cài đặt SQL Server 2008

+ Windows 98, Windows NT Server 4.0 với Server Pack 5 hoặc phiên bản sau, Windows NT.
+ Workstation 4.0 với Server Pack 5 hoặc phiên bản sau hoặc windows 2000
+ Internet Explorer 5.0 hoặc phiên bản sau.

Lợi ích của việc sử dụng HQTCSDL

- Hạn chế dư thừa dữ liệu.
- Ngăn cản truy cập dữ liệu bất hợp pháp (bảo mật và phân quyền sử dụng).
- Cung cấp khả năng lưu trữ lâu dài cho các đối tượng và cấu trúc dữ liệu.
- Cho phép suy dẫn dữ liệu (từ dữ liệu này suy ra dữ liệu khác) sử dụng Rules.
- Cung cấp giao diện đa người dùng.
- Cho phép biểu diễn mối quan hệ phức tạp giữa các dữ liệu.
- Đảm bảo ràng buộc toàn vẹn dữ liệu (Enforcing Integrity Constraints).
- Cung cấp thủ tục sao lưu và phục hồi (backup và recovery)

Kết luận: SQL Server 2000 là một hệ thống quản lý CSDL quan hệ với nhiều tính năng cho phép bạn cấu hình hệ thống thỏa mãn nhu cầu giao dịch của bạn dù là công ty có quy mô nhỏ hay lớn các giao dịch thương mại điện tử.

II. Các kiểu dữ liệu trong SQL Server

Kiểu dữ liệu được dùng để xác định kiểu thông tin (số, ký tự…) và cần bao nhiêu không gian để chứa thông tin trong một cột. Bạn phải chọn kiểu dữ liệu một cách cẩn thận, vì chúng không dễ dàng thay đổi khi bảng được tạo xong và khi đã nhập dữ liệu.

Một số kiểu dữ liệu có chiều dài thay đổi, trong khi một số khác có chiều dài cố định.

Cho phép Null: Tính chất cho phép Null (Nullability) của một cột có nghĩa là không bắt buộc phải nhập dữ liệu cho cột đó. Nếu bạn muốn cho phép một cột rỗng, chỉ cần đặc tả NULL. Nếu bạn muốn trong mỗi hàng đều phải nhập dữ liệu cho một nào đó, hãy xác định NOT NULL. Nếu bạn không xác định NULL hay NOT NULL, giá trị mặc định cho cơ sở dữ liệu sẽ được dùng. (Khi mới cài SQL Server, mặc định của hệ thống sẽ là NOT NULL – Nhưng mặc định này có thể sửa đổi được).

2.1. Số nguyên (Integers)

Có một số kiểu dữ liệu số nguyên: int, smallint, tinyint, bigint dùng để lưu trữ các giá trị vô hướng, chính xác. Sự khác nhau giữa các kiểu dữ liệu số nguyên là kích thước không gian lưu trữ mà chúng yêu cầu và phạm vi giá trị chúng có thể lưu trữ được.

Bigint: Là kiểu dữ liệu Integer 8 byte có miền giá trị tử -2^{63} đến $2^{63} – 1$.

Int: Là kiểu dữ liệu Integer 4 byte có miền từ -2^{31} đến $2^{31} – 1$.

Smallint: Là kiểu dữ liệu Integer 2 byte có miền từ -2^{15} đến $2^{15} – 1$.

Tinyint: Là kiểu dữ liệu Integer 1 byte có miền giá trị từ 0 đến 255.

Bit: Dữ liệu nguyên có giá trị 0 hoặc 1 hoặc NULL

Ví dụ: một số khai báo mẫu:

EmployeeAge tinyint NULL

EmployeeID smallint NOT NULL

CustomerID int NOT NULL

Chú ý: Lưu trữ, truy tìm và các phép toán thực hiện trên kiểu dữ liệu số nguyên luôn luôn tốt hơn bất kỳ kiểu dữ liệu nào khác. Hãy dùng kiểu dữ liệu số nguyên bất kỳ khi nào có thể.

2.2. Kiểu chuỗi ký tự không hỗ trợ Unicode (Character Strings)

Chuỗi có thể chứa dữ liệu ký tự bao gồm: chữ cái, số và các ký hiệu. Bạn có thể lưu trữ ký tự với chiều dài cố định hoặc chiều dài thay đổi bằng các dùng từ khóa char(n) hay varchar(n).

Char(n): Kiểu dữ liệu ký tự có chiều dài n ký tự và không theo mã Unicode, có khả năng lưu trữ tối đa 8000 ký tự.

Khi tạo một trường có chiều dài cố định, và đang đặc tả là trường này luôn chứa n byte thông tin.

+ Nếu dữ liệu nhập vào ít hơn n byte, nó sẽ được đệm vào khoảng trắng sao cho nó luôn chiếm n byte.

+ Nếu cố nhập nhiều hơn n byte dữ liệu thì nó sẽ bị cắt bỏ.

Ví dụ: Bảng 2.1 chỉ ra một số ví dụ về việc nhập dữ liệu vào một trường được khai báo là Fname char(8) (ký hiệu * biểu thị một khoảng trắng trong ví dụ này.

Dữ liệu được nhập	Trường Fname chứa
Lawrence	Lawrence
Mark Anthony	Mark Ant
Peter	Peter***

Bảng 2.1 Các trường ký tự có chiều dài cố định

Varchar: Kiểu dữ liệu ký tự có độ dài thay đổi và không theo mã Unicode, có khả năng lưu trữ tối đa 8000 characters.

Khi sử dụng trường có chiều dài thay đổi, thì phải đặc tả chiều dại cực đại mà trường có thể chứa. Trường có chiều dài thay đổi sẽ không được đệm bởi các khoảng trắng. Điều này làm cho cơ sở dữ liệu của bạn hiệu quả hơn về bộ nhớ, nhưng sẽ gặp khó khăn trong việc thực thi.

Khi một trường được khai báo có chiều dài biến đổi, SQL Server sẽ phải xác định nơi nào trường phải dừng lại và bắt đầu trường kế tiếp. Sẽ có một số byte "phụ phí" để hỗ trợ cho kiểu dữ liệu chuỗi có chiều dài thay đổi. Varchar hữu ích trong trường hợp dữ liệu có chiều dài khác nhau nhiều, và cho phép giá trị NULL trong trường của bạn.

Khi gõ dữ liệu kiểu ký tự vào SQL Server, nên bao dữ liệu trong cặp dấu nháy đơn (hoặc dấu nháy kép). Dấu nháy đơn được ưa dùng hơn do không sợ nhầm lẫn giữa các hằng chuỗi và các định danh trong SQL Server. Để gõ giá trị NULL vào một trường trong SQL Server hãy dùng từ khóa NULL không có dấu nháy.

2.3. Kiểu chuỗi ký tự có hỗ trợ Unicode (Unicode Character Strings).

Nchar: Kiểu dữ liệu ký tự có độ dài xác định n ký tự và theo mã Unicode, có khả năng lưu trữ tối đa 4000 ký tự.

Nvarchar: Kiểu dữ liệu ký tự có độ dài thay đổi với n ký tự và theo mã Unicode, có khả năng lưu trữ tối đa 4000 ký tự.

2.4. Dữ liệu nhị phân (Binary Data)

Kiểu dữ liệu nhị phân được dùng để lưu trữ dữ liệu nhị phân. Dữ liệu nhị phân được lưu trữ như là một chuỗi các số 0 và 1, biểu thị khi nhập xuất là các cặp số thập lục phân. Các cặp số thập lục phân này bao gồm các ký số từ 0 đến 9 và các ký tự từ A đến F.

Ví dụ: Tạo một trường SomeData kiểu binary(20) thì sẽ có 20 byte dữ liệu.

Cũng như với kiểu chuỗi có thể xác định tối đa là 8000 byte cho cả hai kiểu dữ liệu binary(n) và varbinary(n).

Để nhập dữ liệu vào kiểu dữ liệu nhị phân, thì phải đặt trước dữ liệu chuỗi 0x. Ví dụ để nhập giá trị 10 vào một trường binary thì phải gõ: 0×10.

Ví dụ: MyIcons varbinary (255), MyCursors binary (200)

2.5.Kiểu dữ liệu số gần đúng (Approximate Numerics)

Các kiểu dữ liệu số gần đúng là float(n) và real. Các số được lưu trong các kiểu dữ liệu này bao gồm hai phần: phần định trị và phần số mũ. Giải thuật được dùng để tạo ra hai phần này thực sự không thể chính xác (Nói cách khác bạn không thể nhận lại chính xác cái mà bạn nhập vào.

float (n): Dữ liệu số động có phạm vi từ -1.79E + 308 đến 1.79E + 308. Kích thước lưu từ 4 đến 8 byte. Độ chính xác cho phép đối với các số dấu phảy động lên đến 38 chữ số. Nếu không định trước giá trị cho nó thì theo mặc định một số float có độ chính xác là 15 chữ số.

Real: Dữ liệu số động có phạm vi từ - 3.40E + 38 đến 3.40E + 38. Kích thước lưu trữ là 4 byte. Các số thực có độ chính xác là 7 chữ số.

Ví dụ: tạo kiểu dữ liệu như sau: SomeVal real .

Thì có thể lưu trữ các số 188445.2 hay 1884.452 nhưng không thể lưu giá trị 188445.27 hay số 1884.4527. Để lưu trữ được các số lớn hơn thì phải tạo ra biến float với độ chính xác đủ lớn để lưu trữ tất cả các chữ số.

Lưu ý: Float và real thường được dùng cho dữ liệu khoa học và thống kê khi mà độ chính xác tuyệt đối có thể không quan trọng, nhưng dãy dữ liệu biến thiên từ các số cực nhỏ đến các số cực lơn.

2.6. Kiểu dữ liệu số chính xác (Decimal và Numeric)

Độ chính xác sẽ được giữ đến chữ số có nghĩa nhỏ nhất. Khi khai báo một kiểu dữ liệu chính xác thì phải xác định cả độ chính xác (precision) và tỉ lệ (scale). Nếu không xác định thì SQL Server sẽ dùng các giá trị mặc định là 18 và 0 (tương đương với việc tạo dữ liệu kiểu integer.

Decimal: độ chính xác được xác định và miền giá trị từ $- 10^{38} + 1$ đến $10^{38} – 1$.

Numeric: Chức năng tương tự như decimal.

Ví dụ: sử dụng kiểu dữ liệu số

Weight float (8, 4)

Density numeric (5,4).

2.7. Kiểu dữ liệu đặc biệt

Có một số kiểu dữ liệu không thuộc hẳn về một chủng loại nào. Chúng được xếp vào kiểu dữ liệu đặc biệt.

Bit: là kiểu dữ liệu logic và được dùng để lưu trữ thông tin Boolean. Kiểu dữ liệu Boolean được dùng ở hai trạng thái như on/off, true/false, yes /no hay 0/1.

Các cột bit không cho phép giá trị Null, và không thể tạo chỉ mục được. Các kiểu dữ liệu bit chỉ yêu cầu 1 byte bộ nhớ.

Ví dụ: Gender bit NOT NULL

Paid bit NOT NULL

Printed bit NOT NULL

Text và Image

Được dùng khi yêu cầu lưu trữ vượt quá giới hạn 8000 ký tự. Kiểu dữ liệu này có thể lưu trữ lên đến 2GB dữ liệu kiểu nhị phân cho một khai báo. Những kiểu dữ liệu này thường được nhắc đến như các đối tượng Blobs.

Khi khai báo một kiểu dữ liệu này, con trỏ 16-byte sẽ được bổ sung vào hàng. Con trỏ 16-byte này sẽ trỏ đến một trang dữ liệu 8KB nằm ngoài nơi dữ liệu của bạn được lưu trữ. Nếu dữ liệu vượt quá trang dữ liệu 8KB đầu tiên thì một con trỏ 16-byte sẽ được phát sinh để trỏ tới các trang BLOB.

Việc lưu trữ và hiển thị dữ liệu text và image có thể làm giảm đáng kể tốc độ thực thi trên cơ sở dữ liệu, vì sẽ có một số lượng lớn dữ liệu được đẩy vào nhật ký giao tác (transaction log) trong suốt qúa trình thêm, cập nhật, xóa, Để tránh có thể sử dụng lệnh WRITETEXT –lệnh sẽ cập nhật các thay đổi vào dữ liệu mà không tạo ra một mục tương ứng vào transaction log.

Ví dụ: EmployeePhoto image , ScannedContracts image

Description text, Conments text

2.8. Kiểu tiền tệ: Có hai kiểu dữ liệu tiền tệ: Money và Smallmoney

Money: Là kiểu dữ liệu tiền tệ có miền giá trị từ - 2^{63} đến 2^{63} – 1. Kích thước là 8 byte

Smallmoney: Là kiểu dữ liệu tiền tệ có miền giá trị từ - 2^{31} đến 2^{31} – 1. Kích thước là 4 byte.

Ví dụ: AccountsReceivable money

AccountsPayable smallmoney

2.9. Kiểu dữ liệu ngày tháng (Datetime và Smalldatetime)

Datetime: Kiểu dữ liệu ngày/tháng từ January 1, 1753, tới December 31, 9999, với độ chính xác là 3/100 của second hoặc 3.33 milliseconds. Kích thước là 8 byte.

Smalldatetime: Kiểu dữ liệu ngày/tháng từ January 1, 1900, tới December 31, 2079, với độ chính xác là 1 phút. Kích thước là 4 byte.

2.10. Các kiểu dữ liệu khác

Cursor: Là một tham chiếu tới một con trỏ. Chỉ dùng cho các biến và tham số trong stored procedure. Không có kích thước

Sql_variant: Là kiểu dữ liệu có khả năng lưu trữ rất nhiều kiểu dữ liệu khác nhau của SQL SERVER, ngoại trừ text, Ntext, timestamp, and sql_variant. Kích thước thay đổi

Table: là kiểu dữ liệu đặc biệt được sử dụng để lưu trữ tập kết quả của một quá trình xử lý. Tương tự như bảng tạm, khai báo bao gồm danh sách cột và các kiểu dữ liệu. Kích thước thay đổi theo định nghĩa bảng.

Uniqueidentifier: Là kiểu dữ liệu có khả năng tự động cập nhật giá trị khi có 1 bản ghi được thêm mới (tương tự như kiểu dữ liệu Autonumber của Microsoft Access).

Kiểu dữ liệu Timestamp: Mỗi khi có dòng được thêm hoặc cập nhật vào một bảng có cột kiểu timestamp thì giá trị thời gian sẽ được tự động cập nhật. Kích thước là 8 byte.

Ví dụ: PhoneCall timestamp NOT NULL

LastModified timestamp NOT NULL

2.11. Kiểu dữ liệu do người dùng định nghĩa

Người dùng có thể định nghĩa kiểu dữ liệu riêng, dùng trong một cơ sở dữ liệu cụ thể và đưa vào trong cơ sở dữ liệu Model để dùng cho hàng loạt các cơ sở dữ liệu mới được tạo sau đó.

Để tạo một kiểu dữ liệu do người dùng định nghĩa thì phải tạo nó dựa vào các kiểu dữ liệu được cung cấp sẵn của hệ thống.

Phải tạo kiểu dữ liệu của người dùng trước khi sử dụng nó trong một bảng nào đó.

Kiểu dữ liệu do người dùng định nghĩa là kiểu dữ liệu hệ thống đã được tùy chỉnh. Tùy chỉnh một kiểu dữ liệu hữu dụng khi bạn có một số bảng phải lưu cùng một kiểu dữ liệu trong cột và bạn muốn đảm bảo sử dụng thống nhất kiểu dữ liệu cho các cột tương ứng đó trong mỗi bảng (có chính xác cùng kiểu, chiều dài).

Ví dụ: trong bảng Sach có cột MaNXB và trong bảng NhaXuatBan cũng có cột MaNXB, để đảm bảo cột MaNXB trong hai bảng có cùng kiểu dữ liệu, cùng chiều dài ta có thể định nghĩa một manxb_type rồi sau đó gán kiểu dữ liệu này cho cả hai cột trong hai bảng.

+ Tạo kiểu dữ liệu do người dùng định nghĩa bạn có thể sử dụng Enterprise Manager thực hiện theo các bước sau:

1). Mở SQL Server Enterprise Manager.

2). Mở cơ sở dữ liệu Pubs của bạn

3. Mở danh mục "User Defined Data Types"

4). Nhấn chuột phải vào danh mục và chọn New User Defined Data Type từ menu pop – up.

5). Điền tên, kiểu dữ liệu, độ dài, và bất kỳ quy luật hay mặc định nào bạn cần.

6). Nhấn OK khi đã hoàn thành .

Ví dụ: Tạo kiểu dữ liệu manxb-type bằng Enterprise Manager

B1: Khởi chạy Enterprise Manager, mở rộng nhóm MyGroup, mở rộng tên server cục bộ, mở rộng danh mục Database, mở rộng CSDL MyDB.

B2: Chọn danh mục User Defined Data Types, bấm phải chuột rồi chọn New User Defined Data Type từ trình đơn tắt. Cửa xổ User –Defined Data Type Properties xuất hiện.

B3. Trong trường Name nhập vào manxb-type, sau đó chọn kiểu char trong danh mục thả xuống Data type, trong trường Length nhập vào gái trị 4, không chọn hộp kiểm Allow Nulls, Rule là none, Default là none. Bấm OK để lưu kiểu dữ liệu mới này. Như vậy là bạn đã tạo thành công kiểu dữ liệu manxb-type.

Bài tập: Tạo kiểu dữ liệu matacgia-type có thuộc tính sau:

Name: matacgia-type	Data type: char
Length: 10	Allow NULLs: không chọn
Rule: none	Default: none.

+ Xóa kiểu dữ liệu do người dùng định nghĩa

1). Mở rộng cơ sở dữ liệu của bạn và chọn danh mục User Defined Data Types.

2). Bấm chuột phải vào kiểu dữ liệu người dùng định nghĩa rồi chọn Delete.

Ví dụ: Để xóa kiểu dữ liệu do người dùng định nghĩa trong Enterprise Manager, giả sử xóa matacgia-type ta thực hiện các bước sau:

B1: Chọn danh mục User Defined Data Typed trong CSDL MyDB, Bên vùng cửa sổ bên trái của Enterprise Manager chọn matacgia-type, bấm phải chuột rồi chọn Delete từ trình đơn tắt.

B2: Hộp thoại Drop Objects xuất hiện, bấm Drop All

Chú ý: Nếu bạn muốn xóa kiểu dữ liệu đã tạo và kiểu dữ liệu này đang được dùng bởi bảng nào đó thì bạn sẽ không thể xóa nó được. Bạn nhận được một thông điệp lỗi. Bấm nút Show Dependencies để xem những bảo nào đang dùng kiểu dữ liệu này.

Câu hỏi cuối chương

1. Trình bày lịch sử phát triển của MS SQL Server.
2. Nêu tầm quan trọng của sql trong lập trình ứng dụng
3. Trình bày các kiểu dữ liệu trong SQL SERVER

Chương II. CÁC THÀNH PHẦN CỦA SQL SERVER

2. 1 Mô hình quan hệ

Mô hình quan hệ được Ted Codd đưa ra đầu tiên vào năm 1970 và gây được sự chú ý ngay tức khắc vì tính đơn giản và các cơ sở toán học của nó. Mô hình quan hệ sử dụng khái niệm quan hệ toán học như khối xây dựng cơ sở và có cơ sở lý thuyết của nó trong lý thuyết tập hợp và logic vị từ bậc nhất.

Ngày nay, hầu hết các tổ chức đã áp dụng CSDL quan hệ để quản lý dữ liệu trong đơn vị mình.

Mô hình cơ sở dữ liệu quan hệ là cách thức biểu diễn dữ liệu dưới dạng bảng hay còn gọi là quan hệ, mô hình được xây dựng dựa trên cơ sở lý thuyết đại số quan hệ.

Cấu trúc dữ liệu: dữ liệu được tổ chức dưới dạng quan hệ hay còn gọi là bảng.

Thao tác dữ liệu: sử dụng những phép toán mạnh (bằng ngôn ngữ SQL)

Mô hình quan hệ biểu thị cơ sở dữ liệu như một tập các quan hệ.

Trong mô hình quan hệ người ta đưa ra một số khái niệm sau.

2.1.1. Thuộc tính: là một tính chất riêng biệt mô tả một thông tin nào đó của một đối tượng trong CSDL.

Chẳng hạn với bài toán quản lý sinh viên, đối tượng sinh viên cần phải chú ý đến các đặc trưng riêng như: Họ tên, Mã SV, Ngày sinh, Giới tính, Địa chỉ. Các đặc trưng này là các thuộc tính.

- Mỗi một thuộc tính được đặc trưng bởi ba thành phần:

+ *Tên thuộc tính*: Trong cùng một đối tượng không có hai thuộc tính cùng tên.

+ *Kiểu dữ liệu*: Các thuộc tính phải thuộc vào một kiểu dữ liệu nhất định (số , chuỗi, ngày tháng, logic, hình ảnh…). Kiểu dữ liệu ở đây là kiểu đơn.

+ *Miền giá trị*: Thông thường mỗi thuộc tính chỉ chọn lấy giá trị trong một tập con của kiểu dữ liệu và tập hợp con đó gọi là miền giá trị của thuộc tính đó.

Ví dụ: Thuộc tính Ngày sinh thì có kiểu dữ liệu là Datetime

Thường người ta dùng các chữ cái hoa A, B, C …để biểu diễn các thuộc tính hoặc A_1, A_2…A_n để biểu diễn một số lượng lớn các thuộc tính.

2.1.2. Quan hệ

Lược đồ quan hệ (Relation Schema)

Tập tất cả các thuộc tính cần quản lý của một đối tượng cùng với mối liên hệ giữa chúng được gọi là lược đồ quan hệ. Lược đồ quan hệ Q với tập thuộc tính { A_1, A_2…A_n} được viết là Q(A_1, A_2…A_n). Tập các thuộc tính của Q được ký hiệu là Q^+

Ví dụ: Lược đồ quan hệ sinh viên với các thuộc tính như là:

SinhViên (Họ tên, Mã SV, Ngày sinh, Giới tính, Địa chỉ)

Nhiều lược đồ quan hệ cùng nằm trong một hệ thống quản lý được gọi là một lược đồ cơ sở dữ liệu.

Ví dụ: lược đồ cơ sở dữ liệu để quản lý điểm thi của sinh viên có thể gồm các lược đồ quan hệ sau:

SinhViên (Họ tên, Mã SV, Ngày sinh, Giới tính, Địa chỉ)

Điểm (Mã SV, Điểm thi).

Quan hệ (Relation): Sự thể hiện của lược đồ quan hệ Q ở một thời điểm nào đó được gọi là quan hệ.

Một quan hệ là một bảng dữ liệu 2 chiều (cột và dòng), mô tả một thực thể. Mỗi cột tương ứng với một thuộc tính của thực thể. Mỗi dòng chứa các giá trị dữ liệu của một đối tượng cụ thể thuộc thực thể.

Rõ ràng là trên một lược đồ quan hệ có thể định nghĩa rất nhiều quan hệ.

Thường ta dùng các ký hiệu như R, S, Q để chỉ các lược đồ quan hệ, còn quan hệ được định nghĩa trên nó tương ứng được ký hiệu là r, s q.

2.1.3. Bộ

Bộ là tập mỗi giá trị liên quan của tất cả các thuộc tính của một lược đồ quan hệ. Thường người ta dùng các chữ cái thương như t, p, ..để biểu diễn các bộ. Chẳng hạn để nói bộ t thuộc quan hệ r ta viết t r.

Về trực quan thì mỗi quan hệ xem như một bảng, trong đó mỗi cột là một thông tin về một thuộc tính, mỗi dòng là thông tin về một bộ.

Họ tên	Mã SV	Ngày sinh	Giới tính	Địa chỉ
Lê Vân	4515202	12/09/84	Nữ	Hà Nội
Hoàng Tùng	4516802	21/03/84	Nam	Bắc Ninh
Trương Định	4620503	15/05/85	Nam	Hà Nam
Phạm An	4612203	16/04/85	Nam	Nam Đinh
Đỗ Cung	4521402	20/01/84	Nam	Nghệ An

Bảng1. Quan hệ SinhViên

Bảng một chỉ ra một ví dụ của quan hệ SinhViên tương ứng với lược đồ SinhViên ở trên. Mỗi bộ trong quan hệ biểu diễn một đối tượng sinh viên cụ thể.

Như vậy một quan hệ r của $R(A_1, A_2 \ldots A_n)$, ký hiệu là r(R) là một tập m_bộ $r = \{t_1, t_2 \ldots, t_m\}$ trong đó:

Mỗi $t_i = <v1,\ldots,vn>$, v_i $dom(A_i)$

r(R) $dom(A_1)x\ldots xdom(A_n)$. ta có A_i là các thuộc tính và miền giá trị của A_i là $D_i = dom(A_i)$ với i = 1..n

Chú ý: - Các tập $(D_1, D_2 \ldots, D_n)$ là tập các miền giá trị của R

- n được gọi là bậc của quan hệ r
- m được gọi là lực lượng của r
- Quan hệ bậc 1 là quan hệ nhất nguyên, bậc 2 là quan hệ nhị nguyên, bậc n là quan hệ n nguyên.

Ví dụ: Quan hệ EMPLOYEE trên tập các thuộc tính R={SSN, Name, BDate, Address, Salary} là một quan hệ 5 ngôi.

SSN	Name	BDate	Address	Salary	
001	Đỗ Hoàng Minh	1960	Hà nội	425	t1
002	Đỗ Như Mai	1970	Hải Phòng	390	t2
003	Đặng Hoàng Nam	1973	Hà nội	200	t3

t1(001, ‘Đỗ Hoàng Minh’, 1960, ‘Hà nội’ , 425) = t1(R) là một bộ của quan hệ EMPLOYE

2.1.4 Tính chất của quan hệ

- Giá trị đưa vào cột là đơn nhất
- Các giá trị trong cùng một cột phải thuộc cùng một miền giá trị (cùng kiểu)
- Thứ tự dòng cột tùy ý

2.1.5. Khóa

Cho lược đồ quan hệ R, S R^+. S được gọi là một siêu khóa (Superkey) của lược đồ quan hệ R nếu với hai bộ tuy ý trong quan hệ R thì giá trị của các thuộc tính trong S là khác nhau.

Một lược đồ quan hệ có thể có nhiều siêu khóa. Siêu khóa chứa ít thuộc tính nhất được gọi là khóa chỉ định, trong trường hợp lược đồ quan hệ có nhiều khóa chỉ định, thì khóa được chọn để cài đặt gọi là khóa chính (Primary key)(gọi tắt là khóa).

Khóa chính là một (hoặc một tập) các thuộc tính đóng vai trò là nguồn của một phụ thuộc hàm mà đích lần lượt là các thuộc tính còn lại.

Ví dụ: R={SSN, Name, BDate, Address, Salary}

SSN→ Name, BDate, Address, Salary

(Nguồn)→ (Đích)

Ta thấy, từ SSN ta có thể suy ra toàn bộ các thuộc tính ứng. Vậy SSN được gọi là khóa chính.

Một số gợi ý khi chọn khóa:

- Khóa không nên là tập hợp của quá nhiều thuộc tính. Trong trường hợp khóa có nhiều thuộc tính, có thể thêm một thuộc tính “nhân tạo” thay chúng làm khóa chính cho quan hệ.
- Nếu khóa chính được cấu thành từ một số thuộc tính, thì các thành phần nên tránh sử dụng thuộc tính có giá trị thay đổi theo thời gian: như tên địa danh, phân loại.

Các thuộc tính tham gia một khóa được gọi là thuộc tính khóa (prime key), ngược lại được gọi là thuộc tính không khóa (non prime key).

Một thuộc tính được gọi là khóa ngoại nếu nó là thuộc tính của một lược đồ quan hệ này nhưng lại là khóa chính của lược đồ quan hệ khác.

Khóa phụ (second key): đóng vai trò khi ta muốn sắp xếp lại dữ liệu trong bảng.

Ví dụ: Ta có bảng SINHVIEN (MaSV, Hoten, GioiTinh, Diem).

Muốn sắp xếp lại danh sách sinhviên theo thứ tự a, b, c.. của Họ tên. Khi đó thuộc tính Hoten được gọi là khóa phụ.

Ví dụ: Ta hãy xem lược đồ quan hệ sau:

Xe(SODANGBO, QUICACH, NHDANG, MAUSAC, SOSUON, SOMAY, MAXE, QUOCGIA)

Siêu khóa: (SOSUON, QUICACH),...

Khóa chỉ đinh: (SODANGBO,QUOCGIA), (SOSUON), (SOMAY), (MAXE)

Khóa chính: MAXE
Thuộc tính khóa: SODANGBO,QUOCGIA, SOSUON, SOMAY, MAXE
Thuộc tính không khóa: QUICACH, HINHDANG, MAUSAC
Khóa của SinhViên là (MãSV)…..

2.1.6 Các phép trên mô hình quan hệ

2.1.6.1 Phép toán cập nhật

a. Phép chèn (Insert): là phép bổ xung thêm một bộ vào quan hệ r cho trước.

+ Biểu diễn: INSERT(r; A1=d1,A2=d2,...,An=dn) với Ai là thuộc tính, di thuộc dom(Ai), i=1,..,n.

Nếu thứ tự các trường là cố định, có thể biểu diễn phép chèn dưới dạng không tường minh INSERT(r; d1,d2,..., dn).

+ Ví dụ : Chèn thêm một bộ t4=('004', 'Hoàng Thanh Vân',1969, 'Hà nội', 235) vào quan hệ EMPLOYEE(SSN, Name, BDate, Address, Salary) ta có thể viết:

INSERT(EMPLOYEE; SSN= '004', Name= 'Hoàng Thanh Vân', BDate=1969, Address= 'Hà nội', Salary=235).

+ Chú ý : Kết quả của phép chèn có thể gây ra một số sai sót là :

- Bộ mới được thêm không phù hợp với lược đồ quan hệ cho trước
- Một số giá trị của một số thuộc tính nằm ngoài miền giá trị của thuộc tính đó.
- Giá trị khoá của bộ mới có thể là giá trị đã có trong quan hệ đang lưu trữ.

b. Phép loại bỏ (DEL): Là phép xoá một bộ ra khỏi một quan hệ cho trước.

- Biểu diễn : DEL(r; A1=d1,A2=d2,...,An=dn) hay DEL((r, d1,d2,..., dn).

Nếu K=(E1,E2,...,Em) là khoá thì có thể viết DEL(r; E1=e1,E2=e2,...,Em=em)

- Ví dụ :

+ Để xoá bộ t1 ra khỏi quan hệ r:

DEL(EMPLOYEE; SSN= '004', Name= 'Hoàng Thanh Vân', BDate=1969, Address= 'Hà nội', Salary=235).

+ Cần loại bỏ một nhân viên trong quan hệ EMPLOYEE mà biết SSN đó là '004' thì chỉ cần viết: DEL(EMPLOYEE; SSN= '004')

c. Phép cập nhật (UPDATE): Là phép tính dùng để sửa đổi một số giá trị nào đó tại một số thuộc tính.

+ Biểu diễn :

UPD (r; A1=d1,A2=d2,...,An=dn; B1=b1,B2=b2,...,Bk=bk)

Với {B1,B2,...,Bk} là tập các thuộc tính mà tại đó các giá trị của bộ cần thay đổi. {B1,B2,...,Bk} ứng với tập thuộc tính {A1,A2,...,An}.

Hay UPD(r; E1=e1,E2=e2,...,Em=e; B1=b1,B2=b2,...,Bk=bk) với K = (E1,E2,...,Em) là khoá.

+ Ví dụ : Để thay đổi tên nhân viên có SSN= '003' trong quan hệ EMPLOYEE thành Nguyễn Thanh Mai ta có thể viết :

CH (EMPLOYEE; SSN= '03'; Name= 'Nguyễn Thanh Mai')

2.1.6.2. Các phép toán đại số quan hệ

a. Các phép toán tập hợp

i). Phép hợp (Union operation)

Cho hai lược đồ quan hệ Q_1 và Q_2 có cùng tập thuộc tính {A1,A2,..,An}. r_1 và r_2 lần lượt là hai quan hệ trên Q1 vaø Q2. Pheùp hôïp cuûa hai löôïc ñoà quan heä Q1 vaø Q2 seõ taïo thaønh moät löôïc ñoà quan heä Q3. Q3 ñöôïc xaùc ñònh nhö sau:

Q3+= {A1,A2,..,An}

r3= r1+r2 = {t | t ∈ r1 hoaëc t ∈ r2}

Ví duï:

r1

MASV	MAMH	DIEMTHI
99001	CSDL	5.0
99002	CTDL	2.0
99003	MANG	8.0

r2

MASV	MAMH	DIEMTHI
99002	CTDL	2.0
99001	TTNT	5.0
99003	CSDL	6.0

r3 = r1 + r2

MASV	MAMH	DIEMTHI
99001	CSDL	5.0
99002	CTDL	2.0
99003	MANG	8.0
99001	TTNT	5.0
99003	CSDL	6.0

ii). Phép giao(Intersection):

Cho hai löôïc ñoà quan heä Q1 vaø Q2 coù cuøng taäp thuoäc tính {A1,A2,..,An}. **r**1 vaø r2 laàn löôït laø hai quan heä treân Q1 vaø Q2. Pheùp giao cuûa hai löôïc ñoà quan heä Q1 vaø Q2 seõ taïo thaønh moät löôïc ñoà quan heä Q3 nhö sau:

Q3+={A1,A2,..,An}

r3 = r1*r2= {t | t ∈ r1 vaø t ∈ r2}

Ví duï:

r1

MASV	MAMH	DIEMTHI
99001	CSDL	5.0
99002	CTDL	2.0
99003	MANG	8.0

r2

MASV	MAMH	DIEMTHI
99002	CTDL	2.0
99001	TTNT	5.0
99003	CSDL	6.0

r3 = r1*r2

MASV	MAMH	DIEMTHI
99002	*CTDL*	*2.0*

iii) Phép trừ (Minus, difference)

Cho hai löôïc ñoà quan heä Q1 vaø Q2 coù cuøng taäp thuoäc tính {A1,A2,..,An}. r1 vaø r2 laàn löôït laø hai quan heä treân Q1 vaø Q2. Pheùp tröø löôïc ñoà quan heä Q1 cho Q2 seõ taïo thaønh moät löôïc ñoà quan heä Q3 nhö sau:

Q3+={A1,A2,..,An}

r3 = r1 - r2= {t | t ∈ r1 vaø t ∉ r2}

Ví duï:

r1 **r**2 **r**3 = **r**1 - **r**2

MASV	MAMH	DIEMTHI
99001	CSDL	5.0
99002	CTDL	2.0
99003	MANG	8.0

MASV	MAMH	DIEMTHI
99002	CTDL	2.0
99001	TTNT	5.0
99003	CSDL	6.0

MASV	MAMH	DIEMTHI
99001	CSDL	5.0
99003	MANG	8.0

iv Tích Descartes (Cartesian Product, product)

Cho hai löôïc ñoà quan heä Q1(A1,A2,..,An), Q2(B1,B2,..,Bm). r1 vaø r2 laàn löôït laø hai quan heä treân Q1 vaø Q2. Tích Descartes cuûa hai löôïc ñoà quan heä Q1 vaø Q2 seõ taïo thaønh moät löôïc ñoà quan heä Q3 nhö sau:

Q3+ = Q1+ ∪ Q2+= {A1,..., B1,...}

r3 = r1 x r2 = {(t1,t2)| t1 ∈ r1 vaø t2 ∈ r2}

Ví duï:

$\mathbf{r}_1$

MASV	MAMH	DIEMTHI
99001	CSDL	5.0
99002	CTDL	2.0
99003	MANG	8.0

$\mathbf{r}_2$

MAMH	TENMH
CSDL	CO SO DU LIEU
FOX	FOXPRO

$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$

MASV	MAMH	DIEMTHI	MAMH	TENMH
99001	CSDL	5.0	CSDL	CO SO DU LIEU
99001	CSDL	5.0	FOX	FOXPRO
99002	*CTDL*	*2.0*	CSDL	CO SO DU LIEU
99002	*CTDL*	*2.0*	FOX	FOXPRO
99003	MANG	8.0	CSDL	CO SO DU LIEU
99003	MANG	8.0	FOX	FOXPRO

b. Các phép toán quan hệ

i) Phép chọn (Select)

Phép chọn được sử dụng để chọn một tập hợp các bộ thoả mãn điều kiện chọn từ một quan hệ. Ta có thể xem phép chọn như một bộ lọc, nó chỉ giữ lại các bộ thoả mãn điều kiện đặt ra.

Kí hiệu phép chọn δ<điều kiện chọn>(R)

Điều kiện chọn là một biểu thức logic được chỉ ra trên các thuộc tính của R.

Ví dụ: Đưa ra những nhân viên thuộc đơn vị có mã là 4 từ bảng Nhanvien

δ<madv = 4>(Nhanvien)

Biểu thức logic chỉ ra trong điều kiện chọn được tạo nên từ một số hạng mục có dạng

<tên thuộc tính> <phép so sánh><giá trị bằng>

Hoặc <tên thuộc tính> <phép so sánh> <tên thuộc tính>

+ Các phép toán: >, <, >=, <=, <>

+ Giá trị hằng là một giá trị hằng từ miền giá trị của thuộc tính.

Các hạng mục còn có thể được nối với nhau bằng các phép toán logic như and, or, not để tạo ra một điều kiện chọn chung.

ii) Phép chiếu (Projection)

Phép chiếu là phép chọn một số cột của bảng

Cho moät löôïc ñoà quan heä Q(A1,A2,..,An). r laø quan heä treân Q. X ⊆ Q+. Pheùp chieáu cuûa Q leân taäp thuoäc tính X seõ taïo thaønh löôïc ñoà quan heä Q'= Q[X], trong ñoù Q'+ chính laø X vaø r' chính laø r nhöng chæ laáy caùc thuoäc tính cuûa X. Q'+ = X

r' = **r**[X] = r.X = {t'|∃ t∈r vaø t.X = t[X] = t'}

pheùp chieáu chính laø pheùp ruùt trích döõ lieäu theo coät (chieàu doïc)

r

MASV	MAMH	DIEMTHI
99001	CSDL	5.0
99002	CTDL	2.0
99003	MANG	8.0

r' = r.{MAMH}

MAMH
CSDL
CTDL
MANG

r

MASV	MAMH	DIEMTHI
99001	CSDL	5.0
99002	CTDL	2.0
99003	MANG	8.0

r'= r:DIEMTHI >= 5

MASV	MAMH	DIEMTHI
99001	CSDL	5.0
99003	MANG	8.0

ii) Phép nối, phép nối tự nhiên

Cho hai löôïc ñoà quan heä Q1(A1,A2,..,An), Q2(B1,B2,..,Bm).

r1 vaø r2 laàn löôït laø hai quan heä treân Q1 vaø Q2.

Ai vaø Bj laàn löôït laø caùc thuoäc tính cuûa Q1 vaø Q2 sao cho MGT(AI) = MGT(BJ) (MGT: mieàn giaù trò).

θ laø moät pheùp so saùnh treân MGT(AI).

Pheùp nối giöõa Q1 vaø Q2 seõ taïo thaønh moät löôïc ñoà quan heä Q3 nhö sau:

Q3+ = Q1+ ∪ Q2+

r3=r1

*Ai*θ*Bj*

|><| r2 = {t12|∃t1∈r1,∃t2∈r2 sao cho

t12.Q1+ = t1

t12.Q2+ = t2

t1.Ai θ t2.Bj}

Ta ruùt ra caùc böôùc cuï theå ñeå thöïc hieän pheùp nối keát nhö sau:

- Taïo tích descartes
- Thöïc hieän pheùp choïn theo ñieàu kieän E=Ai θ Bj

Ví duï:

Ai là thuộc tính B, Bj laø thuoäc tính F vaø θ laø pheùp so saùnh ">=". Ta ñöôïc keát quaû laø quan heä sau:

Neáu θ ñöôïc söû duïng trong pheùp keát laø pheùp so saùnh baèng (=) thì ta goïi laø pheùp keát baèng. Hôn nöõa neáu AI ≡ Bj thì pheùp keát baèng naøy ñöôïc goïi laø phép nối tự nhiên. Phép nối tự nhiên là một phép nối thường dùng nhất trong thực tế.

Ví duï: Vôùi Ai ≡ Bj = MAMH

r_1

A	B	C
6	**5**	4
7	**5**	5
4	**2**	6

r_2

E	F	H
1	**5**	9
4	**6**	8
7	**5**	3

$r_3 = r_1 \overset{B \geq F}{|><|} r_2$

A	B	C	E	F	H
6	5	4	1	5	9
6	5	4	7	5	3
7	5	5	1	5	9
7	5	5	7	5	3

r_1

MASV	MAMH	DIEMTHI
99001	CSDL	5.0
99002	CTDL	2.0
99003	MANG	8.0

r_2

MAMH	TENMH
CSDL	CO SO DU LIEU
CTDL	CAU TRUC DU LIEU

$r_3 = r_1 \overset{MAMH}{|><|} r_2$

MASV	MAMH	DIEMTHI	TENMH
99001	CSDL	5.0	CO SO DU LIEU
99002	CTDL	2.0	CAU TRUC DU LIEU

iii). Phép chia(division):

Cho hai lược đồ quan hệ Q_1 (A_1, A_2,…,A_n), Q_2(B_1, B_2…,B_m). r_1 và r_2 lần lượt là hai quan hệ của Q_1 và Q_2. A_i, B_j lần lượt là các thuộc tính của Q_1 và Q_2 sao cho n >m. Phép chia Q_1 và Q_2 sẽ tạo thành một lược đồ quan hệ Q_3 như sau:

Q3+ = {A1,...,An-m}

r3 = r1÷r2 = {t3|∀t2∈r2, ∃t1∈r1 t3=t1.{A1,...,An-m}

t2=t1.{An-m+1,...,An}}

Ví duï:

r_1				
A_1	A_2	A_3	A_4	A_5
a	b	d	c	g
a	b	d	e	f
b	c	e	e	f
e	g	c	c	g
e	g	c	e	f
a	b	e	g	e

r_2	
B_1	B_2
c	g
e	f

$r_3 = r_1 \div r_2$		
A_1	A_2	A_3
a	b	d
e	g	c

Khóa của Sv là MASV, khóa của Mh là MAMH, khóa của Kh là MAKHOA, khóa của Kq là (MASV, MAMH), khóa của Lop là MALOP, trong Lop thuộc tính MAKHOA là khóa ngoại.

2. 2.Cấu trúc và vai trò của các CSDL

Sau khi cài đặt SQL Server, các cơ sở dữ liệu sau sẽ được cài đặt: master, model, msdb, tempdb,pubs và Northwind. Bạn cũng có thể thêm cơ sở dữ liệu của mình về sau (đây là các CSDL người dùng) , nhưng các cơ sở dữ liệu này phải đảm bảo có ở đó.

Một vài trong số các cơ sở dữ liệu này (như master, model, tempdb và msdb) là các cơ sở dữ liệu hệ thống. Cơ sở dữ liệu hệ thống chứa thông tin về SQL Server. SQL Server sử dụng CSDL hệ thống để vận hành và quản lý các CSDL người dùng. CSDL người dùng được tạo bởi người sử dụng. Cả hai kiểu CSDL này đều lưu trữ dữ liệu và bạn không thể bỏ chúng mà không gây tác hại cho SQL Server.

Chú ý: Các CSDL hệ thống không nên thay đổi, việc thay đổi chúng có thể làm máy chủ ngừng hoạt động.

Hai cơ sở dữ liệu còn lại là pubs và Northwind là các cơ sở dữ liệu mẫu đơn giản giúp bạn tìm hiểu SQL Server. Bạn có thể bỏ đi các cơ sở dữ liệu này mà không nguy hại gì.

Để nhập dữ liệu vào trong các bảng của bất kỳ một CSDL mẫu nào, bạn phải chỉ ra CSDL tương ứng, lựa chọn bảng và kích chuột phải lên nó để hiển thị menu 'shortcut'. Từ menu shortcut, chọn **Open Table** và **Return all Rows**. Sau đó nhập dữ liệu thích hợp vào bảng.

2.2.1Cơ sở dữ liệu master

Là một cơ sở dữ liệu chính để chạy SQL Server. Ghi nhận thông tin cấp hệ thống, thông tin khởi tạo SQL Server và các thiết lập cấu hình SQL Server. CSDL này cũng ghi nhận tất cả tài khoản đăng nhập, sự tồn tại của các CSDL khác, vị trí của tập tin chính cho tất cả CSDL người dùng. Hãy luôn giữ bản sao lưu mới nhất của CSDL master. Các cơ sở dữ liệu master chỉ có thể được khôi phục lại khi gặp tình huống tai họa nhờ các kỹ thuật đặc biệt.

2.2.2 Cơ sở dữ liệu model

Là khuôn mẫu cho tất cả CSDL khác được tạo trên hệ thống, kể cả tempdb. CSDL model phải tồn tại trên hệ thống bởi vì nó được dùng để tạo lại tempdb mỗi khi SQL Server được khởi động.

Mỗi khi bạn tạo một cơ sở dữ liệu mới, thì cơ sở dữ liệu model sẽ được sao chép, sau đó, các yêu cầu của bạn về kích thước và các thay đổi khác cho cơ sở dữ liệu sẽ được áp dụng. Do đó, mọi đối tượng có trong cơ sở dữ liệu này sẽ được sao chép vào cơ sở dữ liệu mới như nó được tạo ra ở đó.

Ví dụ: Có thể đặt bảng và tên người sử dụng vào cơ sở dữ liệu này ngay sau khi cài đặt SQL Server. Mỗi khi có một cơ sở dữ liệu được tạo sau đó, bảng và tên người sử dụng sẽ xuất hiện trong mọi cơ sở dữ liệu.

Cơ sở dữ liệu model có kích thước khoảng 1.5MB sau khi cài đặt. Vì cơ sở dữ liệu được sao chép để tạo dựng cơ sở dữ liệu mới nên không có cơ sở dữ liệu nào nhỏ hơn cơ sở dữ liệu model.

2.2.3 Cơ sở dữ liệu tempdb

Cơ sở dữ liệu tempdb là nơi các sắp xếp, kết nối và các hoạt động khác đòi hỏi vị trí tạm thời được thực hiện. Cơ sở dữ liệu này có kích thước sau khi cài đặt là 2.5MB. Nhưng như các trường hợp với các cơ sở dữ liệu khác trong SQL Server theo mặc định, nó có thể tăng thêm kích thước khi bạn cần thêm khoảng trống. Cơ sở dữ liệu tempdb sẽ được khởi tạo lại mỗi SQL Server (dịch vụ MSSQL Server) được khởi động lại.

2.2.4 Cơ sở dữ liệu msdb

Cơ sở dữ liệu msdb hỗ trợ dịch vụ SQL Server Agent bao gồm sắp xếp thông tin về các công việc, các cảnh báo lỗi, các sự kiện, và nhân bản. Lịch sử về tất cả các hoạt động sao lưu, lưu trữ được giữ trong cơ sở dữ liệu này. Cơ sở dữ liệu msdb có kích thước mặc định khoảng 8.5MB.

2.2.5 Cơ sở dữ liệu Pubs

Cơ sở dữ liệu này có ý nghĩa như một công cụ học tập. Nó chứa cơ sở dữ liệu mẫu về một nhà xuất bản, bao gồm tác giả, các cuốn sách và việc bán sách. Hầu hết các tính năng cơ sở dữ liệu đều được nêu bật qua việc cài đặt chúng vào cơ sở dữ liệu Pubs. Cơ sở dữ liệu này có kích thước dưới 2MB sau khi cài đặt.

2.2.6 Cơ sở dữ liệu Northwind

Đây là một cơ sở dữ liệu để học tập khác. Cơ sở dữ liệu Northwind là cơ sở dữ liệu mẫu được hỗ trợ cho Microsoft Access. Vì ngày càng có nhiều người sử dụng Microsoft Access di trú vào SQL Server, nên cơ sở dữ liệu Northwind được cung cấp để hỗ trợ họ tìm hiểu các tính năng của sản phẩm này với cơ sở dữ liệu tương tự. Theo mặc định Northwind có kích thước khoảng 4MB.

Câu hỏi cuối chương

1. Trình bày cấu trúc của từng loại csdl.

2. Nêu vài trò của từng loại csdl

3. Cho các lược đồ cơ sở dữ liệu sau

a. Lược đồ cơ sở dữ liệu Công Ty gồm các quan hệ sau: (trang 63 GTNMCSDL)

NhanVien (HoTen, Ten, MaNV, NgaySinh, ĐiaChi, GioiTinh, Luong, MaNGS, MaĐV)

ĐonVi (TenĐV, MaĐV, MaNQL, NgayBatĐau)

ĐiaĐiemĐonVi (MaĐV, ĐiaĐiemĐV)

DuAn (TenDA, MaDA, ĐiaDiemDA, MaĐV)

NhanVienDuAn (MaNV,MaDA, SoGio)

b. Lược đồ cơ sở dữ liệu quản lý điểm sinh viên gồm các lược đồ quan hệ sau:

Sv(MASV, HOSV,TENSV, GIOITINH, NGAYSINH, MALOP, TINH)

Lop(MALOP,TENLOP,SISO,MAKHOA)

Kh(MAKHOA,TENKHOA, SOCBGD)

Mh(MAMH, TENMH, SOTIET)

Kq(MASV, MAMH, DIEMTHI)

- Xác định khóa chính và khóa ngoài của các quan hệ trên

Chương III. GIỚI THIỆU MỘT SỐ CÔNG CỤ SQL SERVER

I. Sử dụng công cụ Enterprise manager để tạo ra các CSDL, các đối tượng trong CSDL.

Muốn tạo một cơ sở dữ liệu trong SQL Server cho riêng mình nhằm lưu trữ các dữ liệu riêng biệt và đưa vào khai thác các dữ liệu đó thì có thể sử dụng Enterprise Manager.

Chỉ những người với vai trò là **sysadmin** và **dbcreator** thì mới có thể tạo lập cơ sở dữ liệu. Do đó, bạn có thể đăng nhập vào với tên tài khoản người dùng là **SA** để thực hiện việc tạo cơ sở dữ liệu mới cho ứng dụng của mình.

1. Các thuộc tính của một cơ sở dữ liệu trong **Microsoft SQL Server**

+ Tên cơ sở dữ liệu (Database name)

Là duy nhất trong một **Microsoft SQL Server**, độ dài tối đa là 123 ký tự. Bạn nên đặt tên cơ sở dữ liệu gợi nhớ như QLBanhang (Quản lý bán hàng), QLHocsinh (Quản lý học sinh), …

+ Vị trí tập tin (File location)

Là tên và đường dẫn vật lý của các loại tập tin dữ liệu dùng để lưu trữ cơ sở dữ liệu của **Microsoft SQL Server**. Thông thường, các tập tin này sẽ được lưu tại **C:\Program Files\Microsoft SQL Server\MSSQL\Data**.

+ Tên tập tin (File name)

Là tên luận lý của mỗi loại tập tin dữ liệu tương ứng mà hệ thống **Microsoft SQL Server** dùng để quản lý bên trong. Tương ứng mỗi loại tập tin dữ liệu sẽ có một tên tập tin riêng biệt.

+ Kích thước ban đầu (Initial size)

Là kích thước khởi tạo của tập tin dữ liệu khi cơ sở dữ liệu mới được tạo lập. Đơn vị tính là **MegaByte**MB). Thông thường kích thước ban đầu của một cơ sở dữ liệu mới tối thiểu phải bằng kích thước của cơ sở dữ liệu **Model**, bởi vì **Microsoft SQL Server** sẽ lấy cơ sở dữ liệu **Model** làm khuôn dạng mẫu khi tạo lập một cơ sở dữ liệu mới.

+ Việc tăng trưởng kích thước tập tin dữ liệu (File growth)

Là các quy định cho việc tăng trưởng tự động kích thước tập tin dữ liệu, bởi vì các dữ liệu sẽ được người dùng nạp vào để lưu trữ ngày càng nhiều hơn so với kích thước ban đầu khi tạo lập. Việc tăng trưởng sẽ tự động làm tăng kích thước tập tin dữ liệu liên quan theo từng **MB** (**in megabytes**) hoặc theo tỷ lệ phần trăm (**by percent**) của kích thước tập tin hiện hành khi các dữ liệu bên trong **Microsoft SQL Server** lưu trữ gần đây so với kích thước tập tin vật lý hiện thời. Mặc định kích thước tập tin dữ liệu sẽ được tăng tự động **10%** khi dữ liệu lưu trữ gần đây bị đầy.

+ Kích thước tối đa tập tin dữ liệu (Maximum file size)

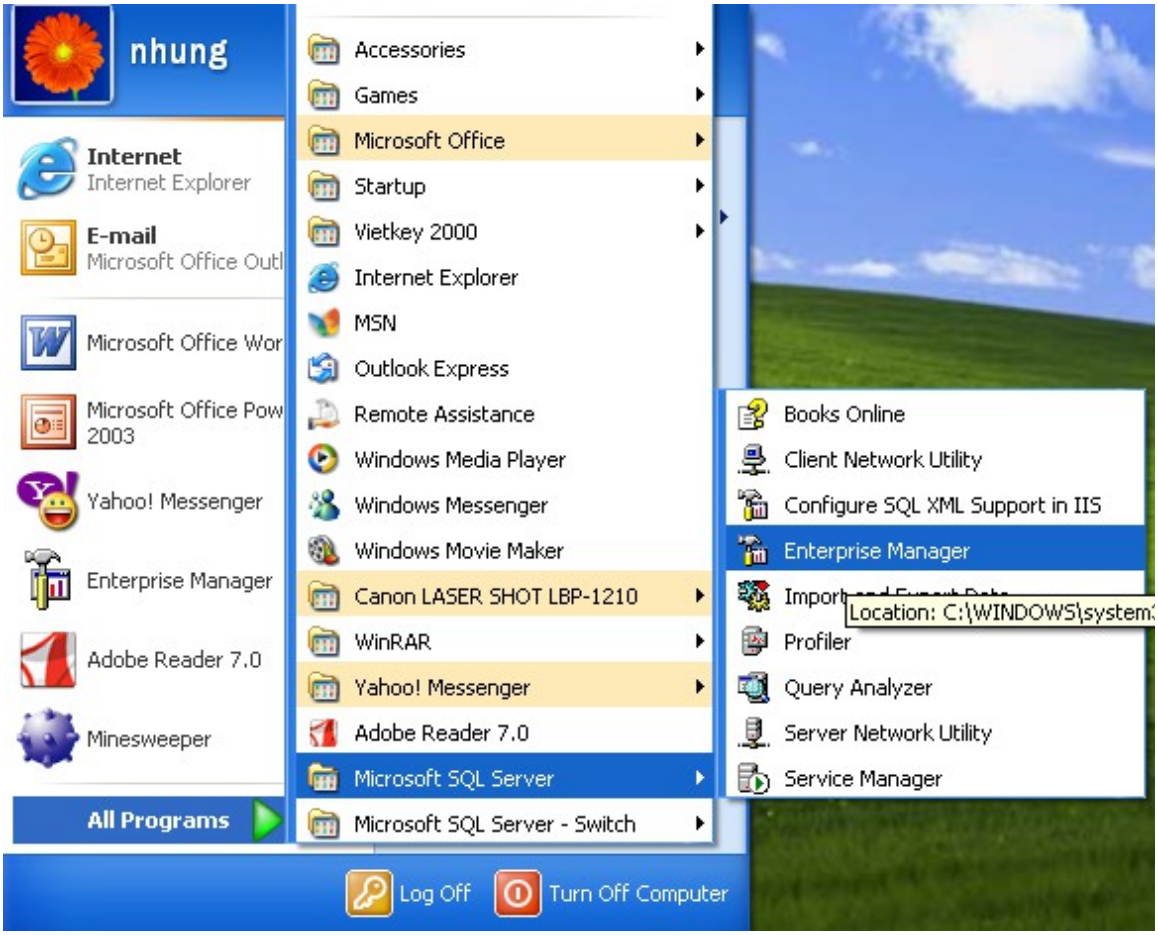
Là việc quy định sự tăng trưởng tự động kích thước của các tập tin dữ liệu nhưng có giới hạn (**restrict file growth**) đến một số **MB** nào đó hoặc là không có giới hạn (**un-restrict file growth**). Trong trường hợp nếu bạn chọn có giới hạn kích thước của tập tin dữ liệu, bạn phải biết tự thêm vào các tập tin dữ liệu mới khi dữ liệu lưu trữ đã bằng với kích thước tối đa của tập tin dữ liệu. Các tập tin dữ liệu mới này chính là loại tập tin thứ yếu (**Secondary data file**) và bạn có thể lưu trữ các tập tin vật lý này tại các đĩa cứng khác có bên trong **Microsoft SQL Server**. Đây cũng là một nét đặc trưng cửa mô hình cơ sở dữ liệu phân tán (**distributed database**) mà **Microsoft SQL Server** hỗ trợ.

2. Tạo CSDL bằng Enterprise Manager

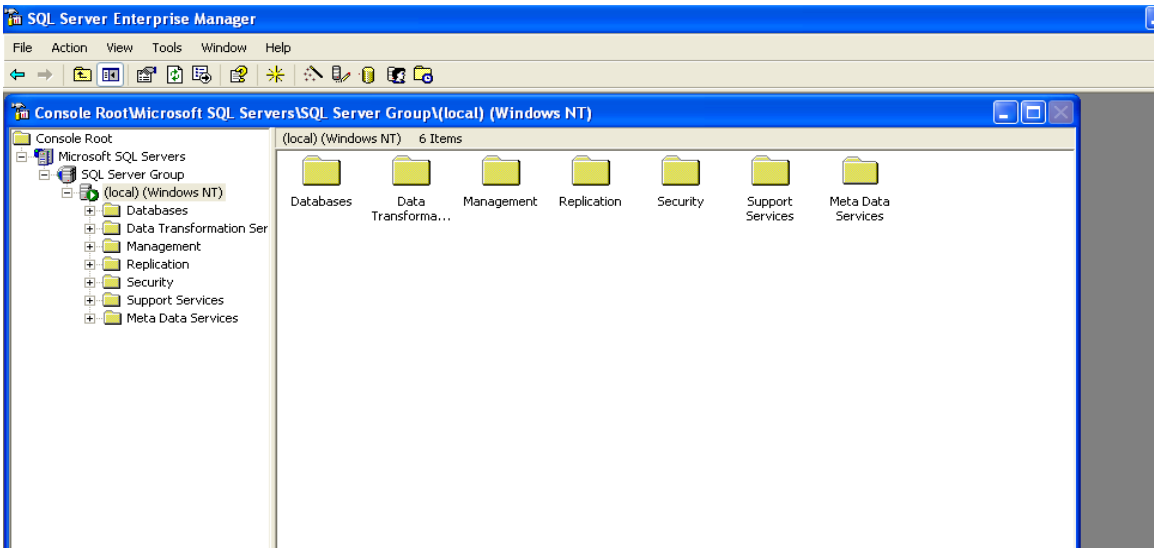
2.1Tạo CSDL

Để tạo CSDL bằng **Enterprise Manager** chúng ta phải làm theo các bước sau:

Bước 1: Khởi động SQL Server Enterprise Manager bằng cách chọn nút Start/ Programs/ Microsoft SQL Server/ Enterprise Manager.



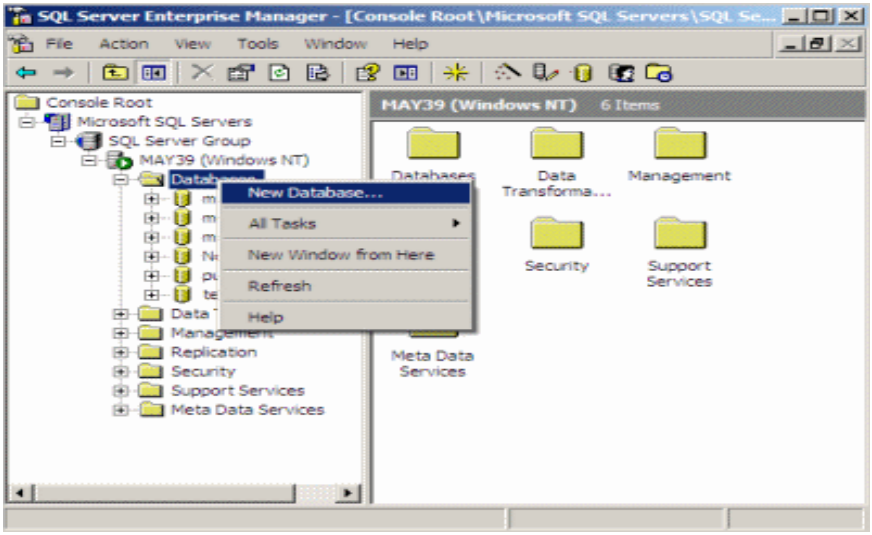
Hình 3.1a Khởi tạo Enterprise Manager



Hình 3.1b Khởi tạo Enterprise Manager

Bước 2: Chọn tên Server cục bộ của máy bạn trong nhóm MyGroup.Mở rộng tên server, (Nháy đúp chuột lên thư mục Databases. Một danh sách của CSDL được tạo lập trong nhóm SQL Server được hiển thị.)

Kích chuột phải lên đối tượng Database, chọn New Database



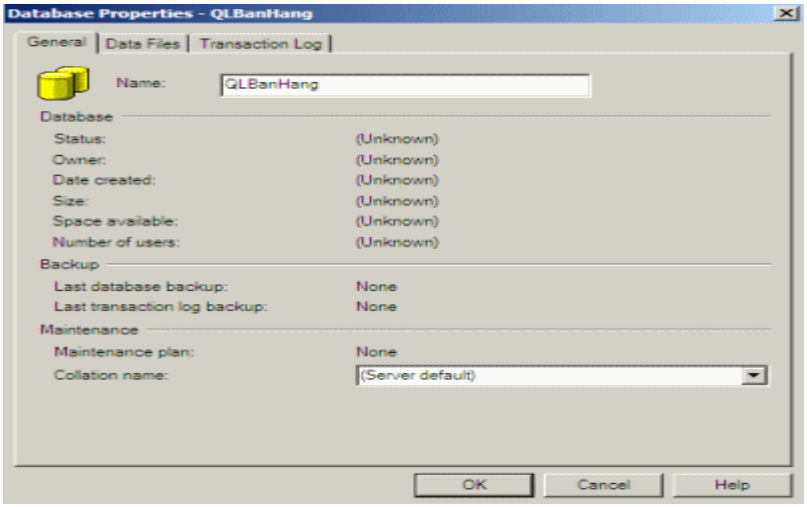
Hình 3.2 Tùy chọn New Database

(Bước 3: Hoặc có thể chọn vào mục Action trên thanh menu: một trình đơn sẽ xuất hiện. Chọn tùy chọn New Database từ trình đơn vừa xuất hiện.)

Bước 3:Một cửa sổ Database Properties xuất hiện.

- Chọn tab General, trong trường Name ta gõ tên CSDL mới cần tạo vào.

Ví dụ như **QLBanHang**, chỉ định kích thước ban đầu khởi tạo của tập dữ liệu chính, thay đổi các thuộc tính khác (nếu cần).



Hình 3.3 Cửa sổ Data Properties và thẻ General

- Bấm **OK** là cơ sở dữ liệu mới đã được tạo

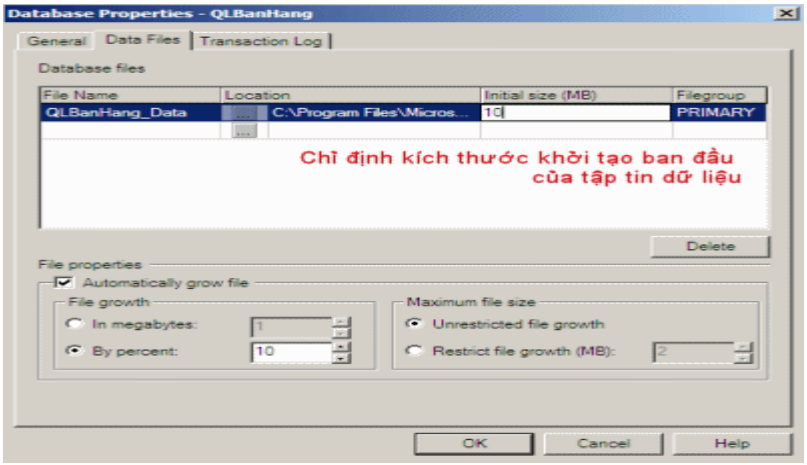
2.2 Thay đổi các thuộc tính của tệp tin

Nếu muốn thay đổi các thuộc tính của tệp tin thì ta thực hiện như sau:

Chọn tab **Transaction Log** để thay đổi các thuộc tính của tập tin lưu vết.

- Chọn tab **Data Files** để chỉ định kích thước ban đầu khi khởi tạo của tập tin dữ liệu chính, thay đổi các thuộc tính khác (nếu cần).

Khi chọn thẻ **Data Files**, **Enterprise Manager** tự động tạo tập tin dữ liệu chính QLBanHang_ Data (có thể thay đổi tên tệp tin này, đường dẫn vật lý của nó). Tập tin chính thuộc nhóm tập tin PRIMARY không thể thay đổi. Nếu bạn muốn xóa bỏ tập tin vừa nhập chỉ việc chọn nó rồi bấm phím Delete.

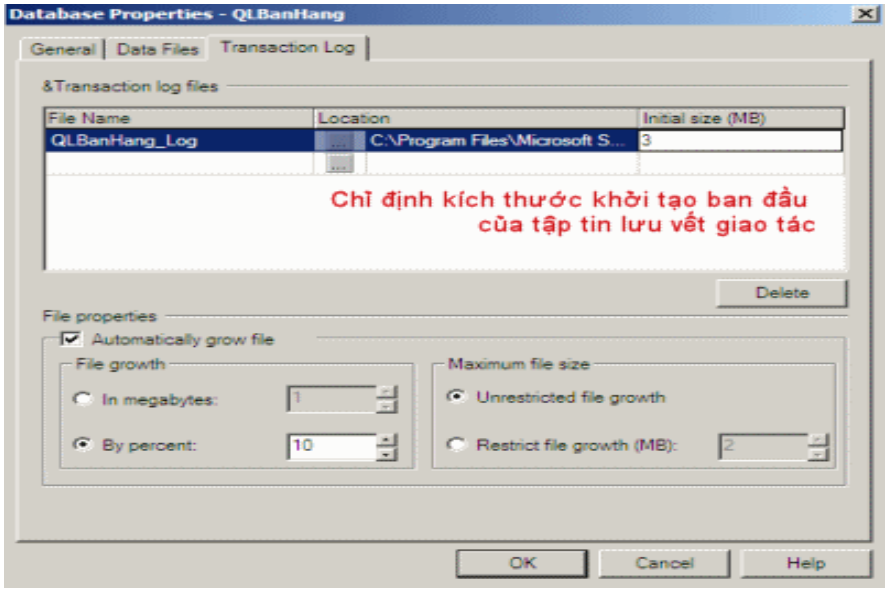


Hình 3.4 Các thuộc tính cơ sở dữ liệu - tab Data Files

Trong vùng **File Properties** bên dưới (nhìn hình 3.4), thấy hộp kiểm **Automatically grow file** được chọn, nghĩa là dung lượng tập tin tự động gia tăng 10% vì tùy chọn **By percent** được chọn và kích thước tối đa của tập tin không bị giới hạn do **Unrestricted file growth** được chọn. Bạn có thể thiết lập các tùy chọn này cho từng tập tin dữ liệu riêng biệt. Chọn tập tin QLBanHang_Data hãy để các tùy chọn **File properties** mặc định.

Chọn tab **Transaction Log** để thay đổi kích thước và các thuộc tính của tập tin lưu vết giao tác.

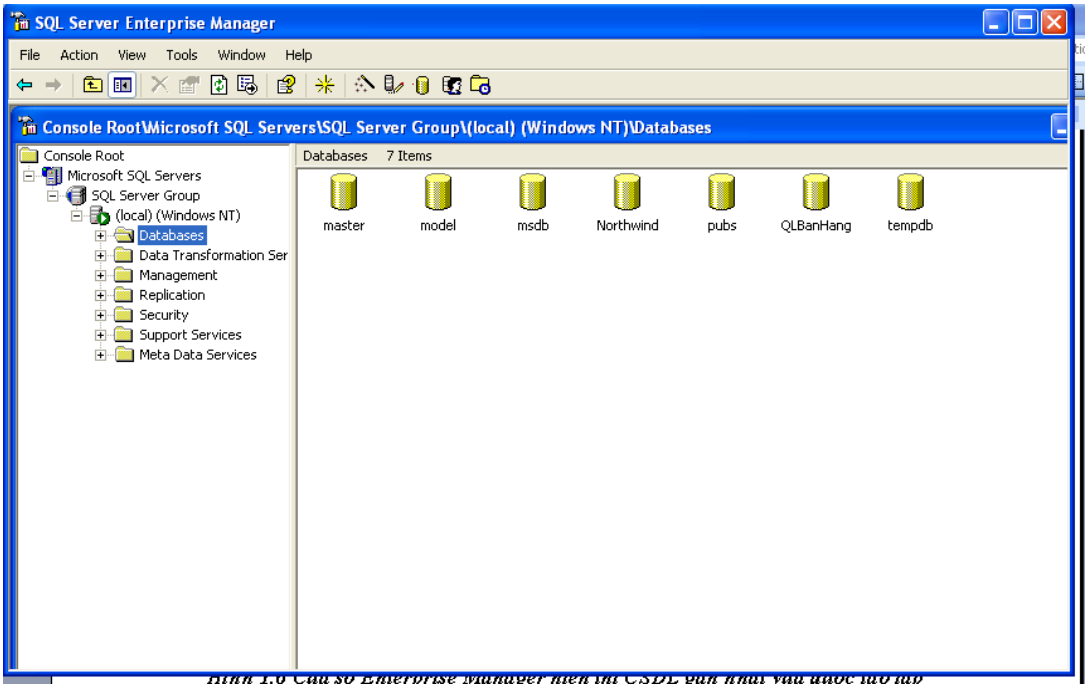
- Tại tab **Transaction Log** chỉ định kích thước ban đầu khi khởi tại của tập tin lưu vết giao tác, thay đổi các thuộc tính khác (nếu cần).



Hình 3.5 Các thuộc tính cơ sở dữ liệu - tab Transaction Log

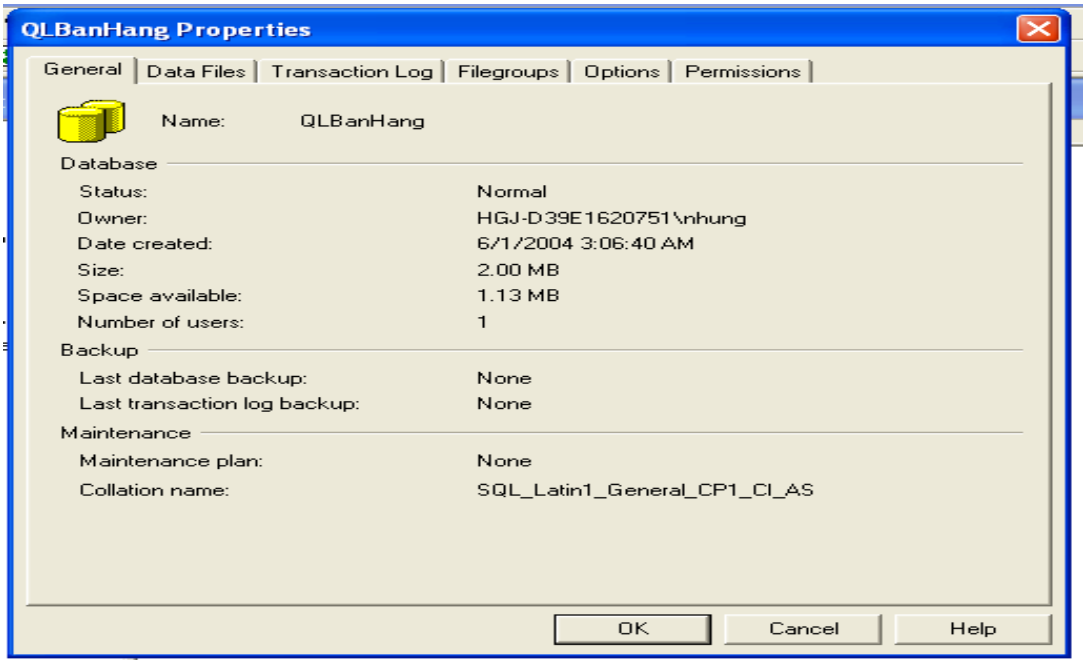
Nhấn **OK** để hệ thống bắt đầu thực hiện việc khởi tạo cơ sở dữ liệu đã chỉ định.

CSDL mới QLBanHang được hiển thị chi tiết trong cửa sổ trình quản lý (Enterprise Manager Window) như hình 3.6



Hình 3.6 Cửa sổ Enterprise Manager hiển thị CSDL gần nhất vừa được tạo lập

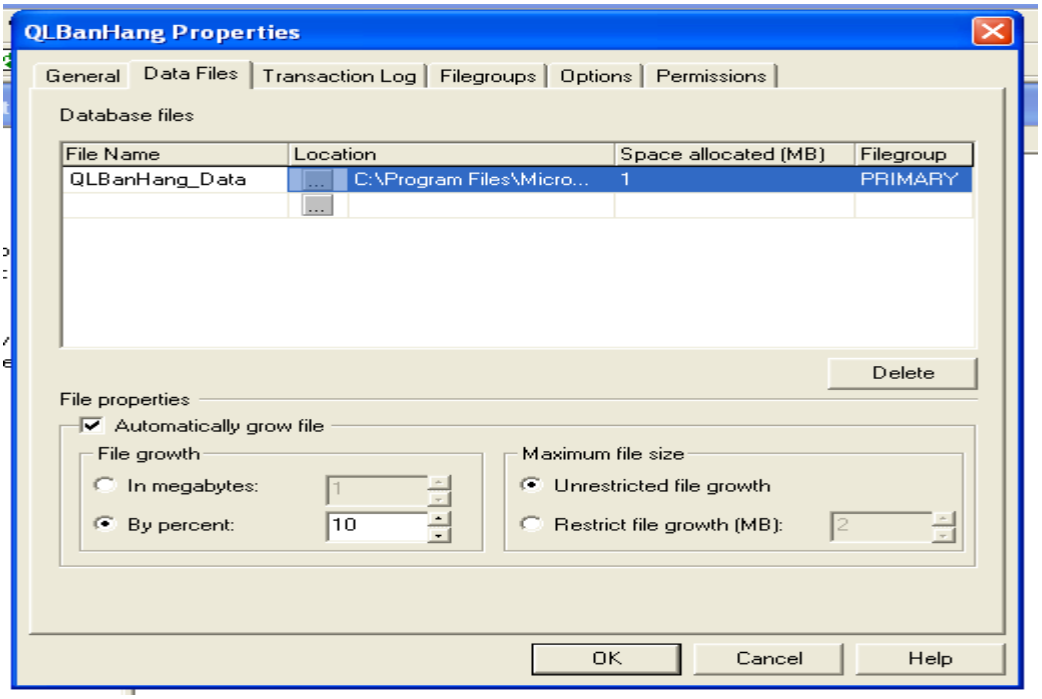
Nháy đúp chuột vào CSDL QLBanHang để hiển thị các thuộc tính của CSDL như hình 3.7



Hình3.7 Các thuộc tính của CSDL .

Như vậy bạn đã tạo thành công CSDL QLBanHang.

Chúng ta có thể kiểm tra chi tiết tệp dữ liệu của CSDL QLBanHang giống hình 3.8 bằng cách chọn vào tab Data Files trong của sổ Database Properties.

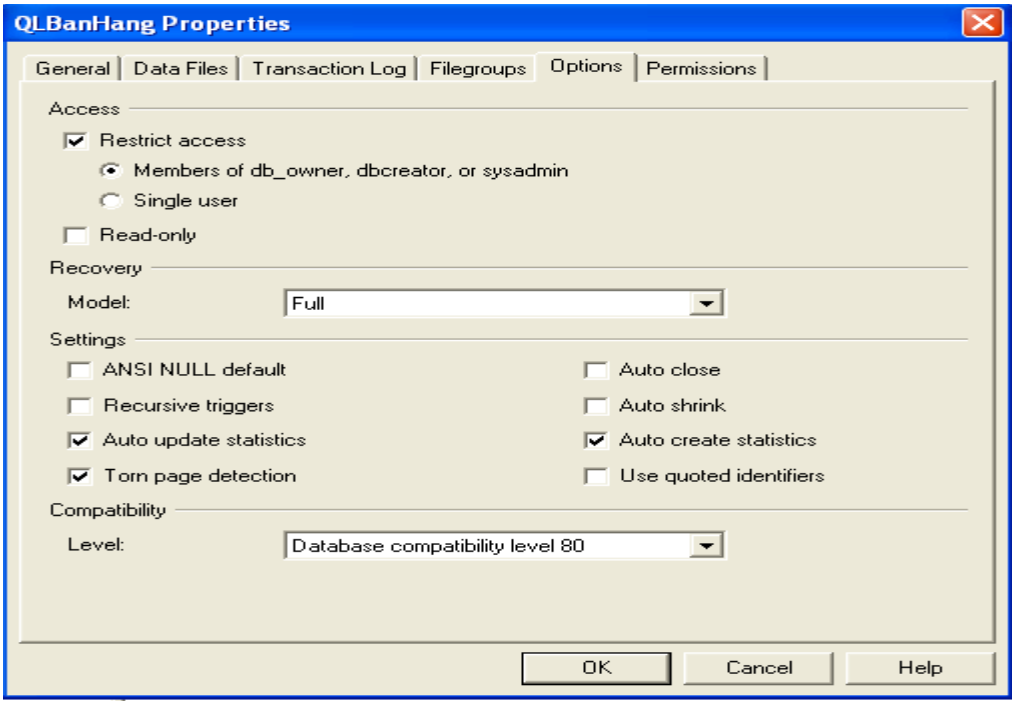


Hình 3.8 Kiểm tra các thuộc tính vật lý cua tệp tin

2.3 Cấu hình các tùy chọn SQL Server cho một CSDL

Chúng ta có thể Cấu hình các tùy chọn SQL Server cho một CSDL như sau:

B1: Trong cửa sổ các thuộc tính CSDL, chọn mục **Options**



Hình 3.9 Cài đặt các thuộc tính truy cập

B2: Chọn tùy chọn Restrict access và kiểm tra nó

B3. Chọn tùy chọn single user để cho chép chỉ một người sử dụng truy cập vào CSDL tại một thời điểm hoặc chọn tùy chọn members of db-owner, dbcreator, orsysadmin nếu quyền truy cập chỉ được gán cho các thành viên trong nhóm này.

B4: chọn OK

Chúng ta cũng có thể thiết lập cá thuộc tính khác như read only và autoshrink.

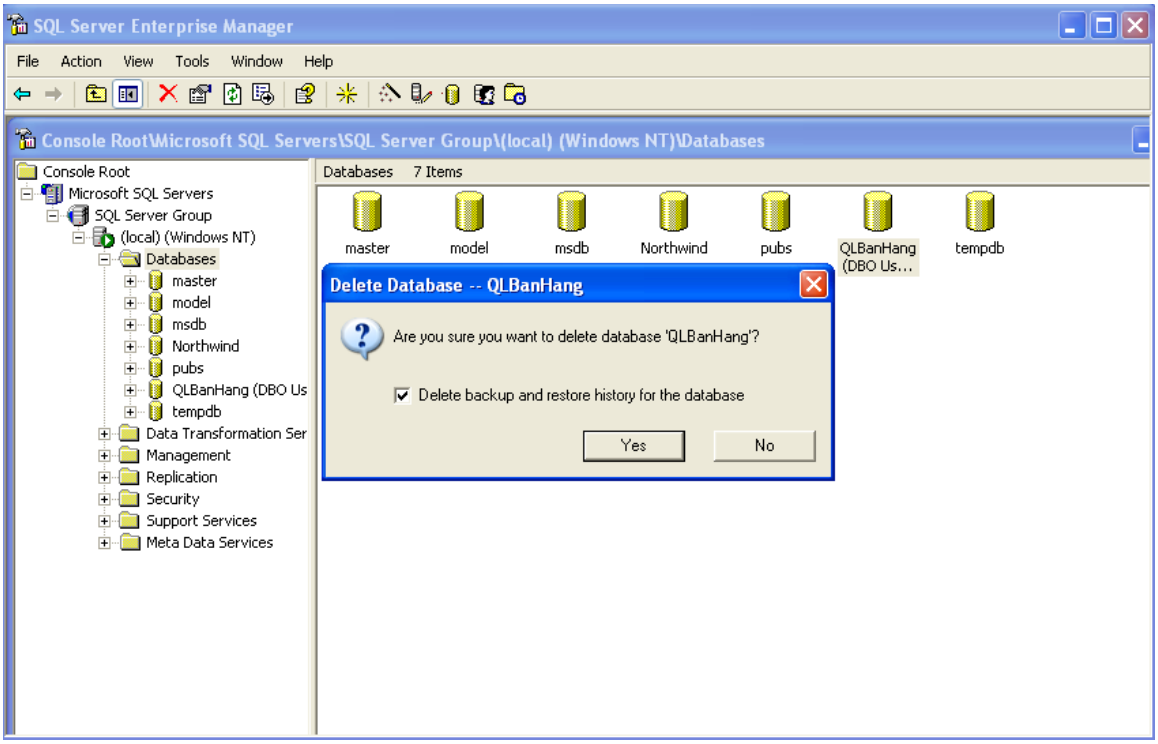
Chọn read only để kích hoạt tùy chọn. Nếu một CSDL được thiết lập read only, người sử dụng không thể sửa chữa nó nhưng có thể đọc dữ liệu của CSDL.

Chọn tùy chọn autohrinh để kích hoạt nó. Khi tùy chọn này được kích hoạt, các tệp tin của CSDL được tự động điều chỉnh theo chu kỳ, kích cỡ của các tệp dữ liệu và các tệp tin nhật ký có thể được điều chỉnh lại một cách tự động bởi SQL Server.

Sau khi tạo CSDL có thể dùng SQL Server Enterprise Manager để xem CSDL mới tạo với đối tượng trong nó.

+ Khởi động Enterprise Manager, mở rộng nhóm Mygroup, mở rộng tên server cục bộ của máy bạn, mở rộng danh mục Database, sau đó chọn CSDL QLBanHang mà vừa tạo.

Xóa CSDL: Khởi động Enterprise Manager, mở rộng nhóm Mygroup, mở rộng tên server cục bộ của máy bạn, mở rộng danh mục Database, sau đó chọn CSDL QLBanHang mà vừa tạo, bấm phải chuột rồi chọn Delete từ trình đơn tắt. Một thông điệp Delete Database xuất hiện yêu cầu bạn xác nhận. Bấm yes để đồng ý xóa CSDL và các thông tin sao lưu, khôi phục nó như trong hình:



Hình 3.10 Hộp thông điệp Delete Database.

II. Dùng công cụ Query Analyzer để tạo các CSDl, các đối tượng trong CSDL.

Bạn có thể tạo một cơ sở dữ liệu mới bằng câu lệnh CREATE DATABASE được thực hiện trong tiện ích Query Analyzer.

Lệnh **CREATE DATABASE** được sử dụng để tạo một CSDL mới.Nhưng như đã nói ở trên người dùng phải có quyền thích hợp để tạo lập một CSDL mới. Người quản trị hệ thống gán quyền này cho người sử dụng.

Mẫu đơn giản của câu lệnh tạo CSDL là:

CREATE DATABASE DATABASE_NAME
ON [PRIMARY]
(
NAME = logical_file_name,
FILENAME = 'os_file_name',
SIZE = size (in MB or KB),
MAXSIZE = maximum_size (in MB or KB) or UNLIMITET (fill all available space),
FILEGROWTH = growth_increment (in MB or KB)
)
LOG ON
(
NAME = logical_file_name,
FILENAME = 'os_file_name',
SIZE = size (in MB or KB),
MAXSIZE = maximum_size (in MB or KB) or UNLIMITET, FILEGROWTH = growth_increment (in MB or KB)
)
[FOR LOAD | FOR ATTACH].

Tùy chọn **LOG ON** lưu trữ trạng thái của tệp (log files) là các tập tin nhật ký của CSDL vào ổ đĩa.

Ví dụ1: CSDL Passenger được tạo bằng Query Analyzer được mô tả như sau:

CREATE DATABASE PassengerAnotherdatabase
ON [PRIMARY]
(
NAME = MYDAT,
FILENAME = 'C:\ Program Files\Microsoft SQL Server\MSSQL\Data\ MyDat.mdf',
SIZE = 10, MAXSIZE = 50, FILEGROWTH = 5
)
LOG ON
(

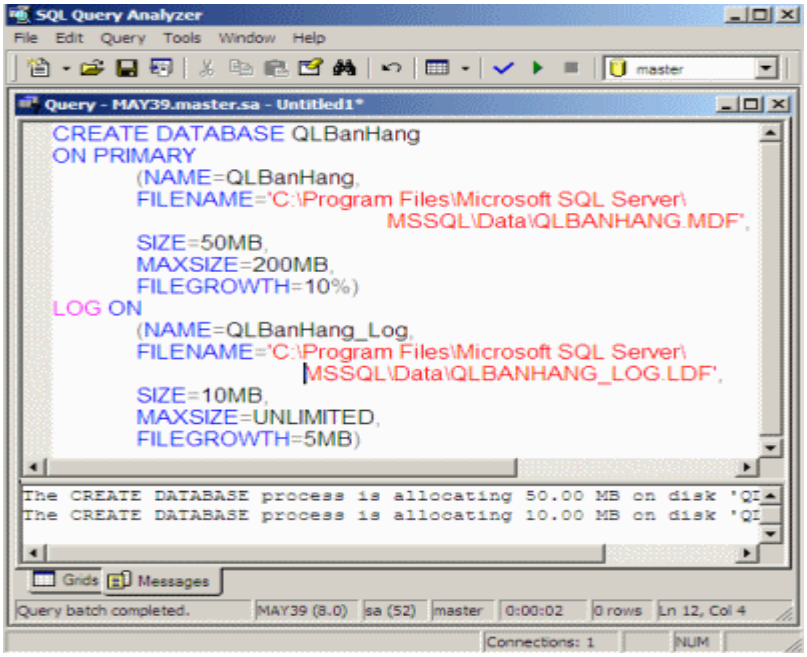
NAME = MYDAT_LOG,

FILENAME = 'C:\ Program Files\Microsoft SQL Server\MSSQL\Data\ MyDat.ldf',

SIZE = 5MB, MAXSIZE = 25MB, FILEGROWTH = 5MB)

GO.

Ví dụ2:Tạo một cơ sở dữ liệu có tên là QLBanHang với kích thước ban đầu lúc khởi tạo của tập tin dữ liệu chính là 50MB, tự động tăng kích thước lên 10% khi dữ liệu bị đầy, kích thước tăng trưởng tập tin dữ liệu tối đa không quá 200MB, tập tin lưu vết với kích thước ban đầu lúc khởi tạo là 10MB, tự động tăng kích thước tập tin lên 5MB khi dữ liệu bị đầy, kích thước tăng trưởng tập tin không giới hạn. Bạn thực hiện câu lệnh như sau :



Hình 3.12Tạo cơ sở dữ liệu QLBanHang bằng tiện ích Query Analyzer

Để tạo CSDL mới trên, bạn cần thực hiện các bước sau:

Bấm nút Start > Programs > Microsoft SQL Server > Query Analyzer. Hộp thoại kết nối với SQL Server xuất hiện, trong danh sách SQL Server chọn tên server cục bộ của máy bạn (mặc định). Phần Connect using chọn Windows authentication rồi bấm OK. Ứng dụng SQL Server Query Analyzer xuất hiện. Viết đoạn script vào vùng cửa sổ bên phải của Query Analyzer. Nhấn phím F5 hoặc nút Execute Query (hình tam giác mầu xanh) để thực thi các lệnh tạo CSDL.Như vậy là bạn đã tạo thành công CSDL mới.

Như vậy, bạn có hai cách để thực hiện các hành động trong **Microsoft SQL Server** : hoặc thực hiện các thao tác trong tịch ích **Enterprise Manager** hoặc thực hiện các câu lệnh **T-SQL** (**Transaction SQL**) trong tiện ích **Query Analyzer** để tạo ra các đối tượng bên trong **Microsoft SQL Server**. Để môi trường làm việc trở nên sinh động hơn, bạn nên phối hợp sử dụng cả hai tiện ích nêu trên

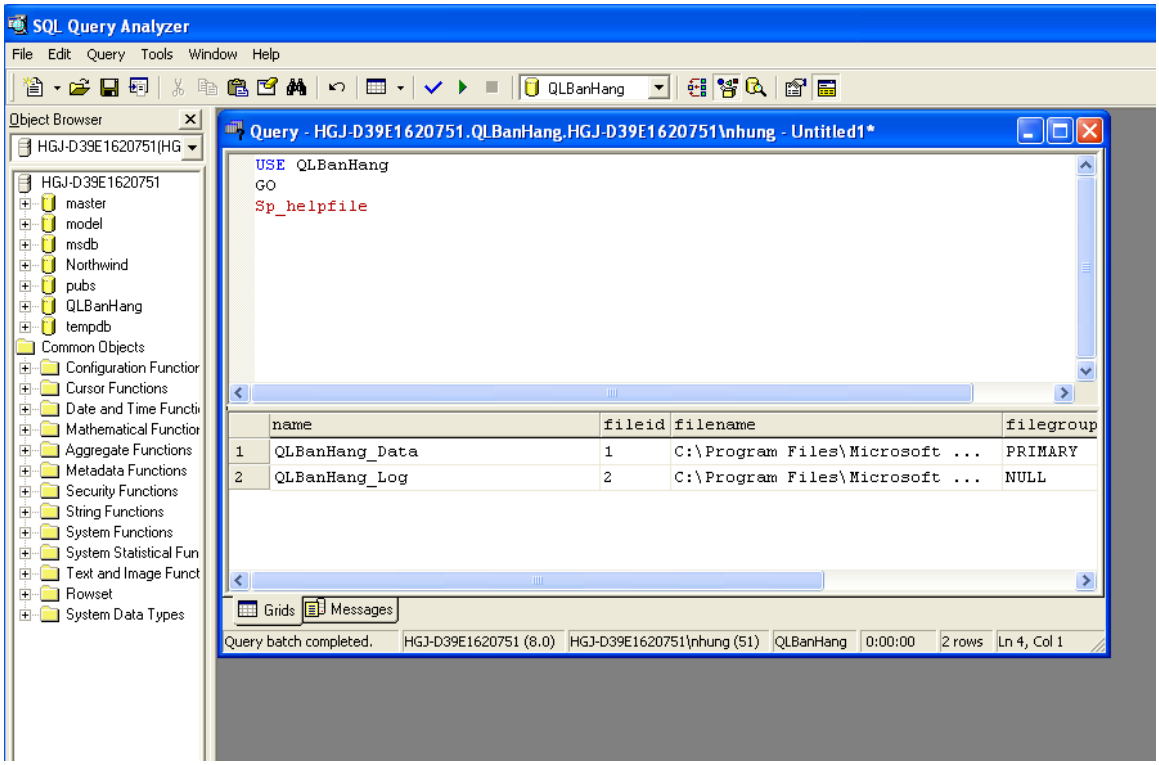
Sử dụng SQL Query Analyzer để xem một số thông tin về CSDL.

- Trong vùng cửa sổ bên phải nhập các lệnh sau rồi nhấn phím F5.
- Để hiển thị thông tin của tất cả các tập tin trong CSDL QLBanHang:

  USE QLBanHang

  GO

  Sp_helpfile



  Hình 3.13 Thông tin các tệp tin trong CSDL QLBanHang

- Để hiển thị thông tin của tất cả các tập tin trong CSDL TestDB và không gian đã cấp phát:

Sp_helpdb TestDB

GO.

- Để xem thông tin của tất cả CSDL trong SQL Server:

Sp_helpdb

GO.

Xóa CSDL: khởi động SQL Query Analyzer, nhập lệnh sau để xóa CSDL TestDB:

USE master

GO

DROP DATABASE TestDB

GO.

III.Thiết lập cấu hình về Client network utility, Sever network utility.

3.1 Client network utility

The Client Network Utility is a configuration tool that tells ADO which network protocols to use when connecting to SQL Server and MSDE. The utility can be very useful especially if the network connectivity is limited to certain protocols, like TCP/IP when connecting to the Internet.

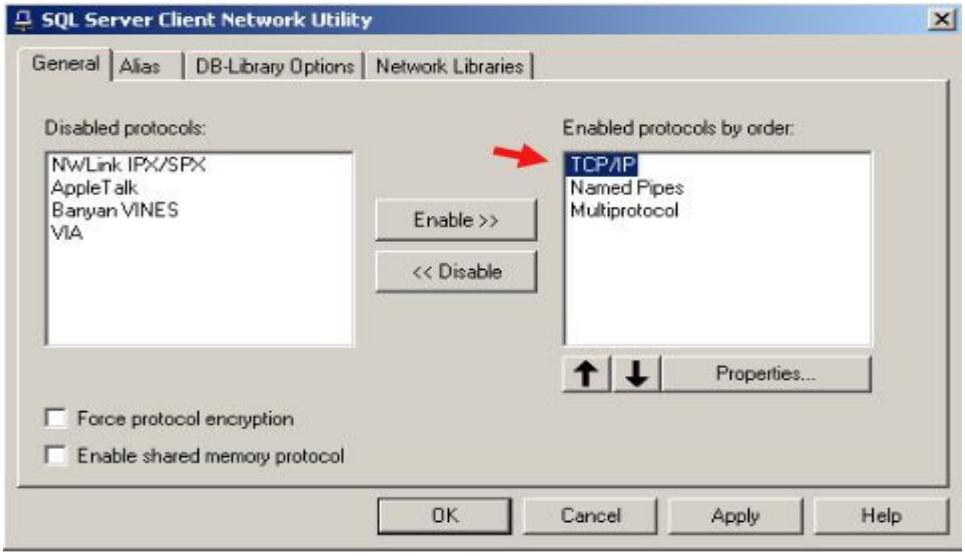
Client Network Utility là một công cụ cấu hình cho các giao thức mạng ADO đó để sử dụng khi kết nối với SQL Server và MSDE. Các tiện ích có thể rất hữu ích đặc biệt là nếu kết nối mạng được giới hạn trong các giao thức nhất định, như giao thức TCP / IP khi kết nối với Internet.

SQL Server installation, creates a shortcut for the Client Network Utility in the Microsoft SQL Server group in the Start menu. The MSDE installation on the other hand, does not create the shortcut leading many to believe that the Client Network Utility does not ship with MSDE. As a matter of fact, The utility is part of the Microsoft Data Access Components that ship with Windows and can be downloaded from the Microsoft Website.

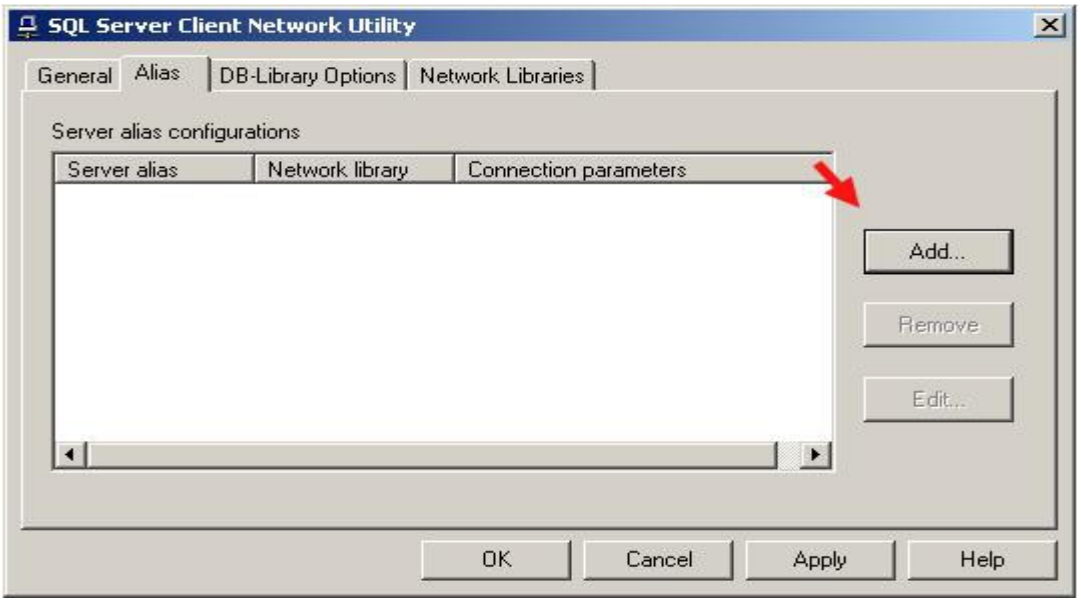
You can use The Client Network Utility to configure the protocols you wish to support. You can also create aliases that can act like server names. All aliases created in the Client Network Utility can be registered in Teratrax Database Manager and accessed like any other server name.

How to set up Microsoft SQL 2000 Client Network Utility

In the General information tab Enable TCP/IP and move to the top of the list.

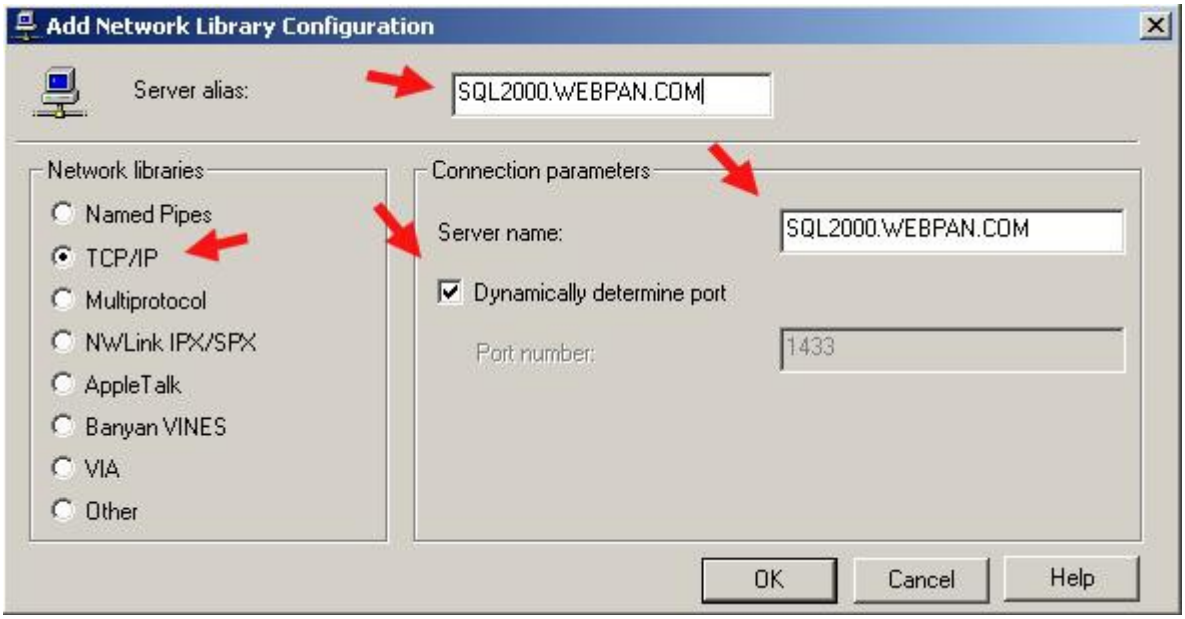


You must create a **Server Alias** using **Client Network Utility** before creating your SQL Database connection. When you open Client Network Utility, select the **Alias** tab, as shown below...

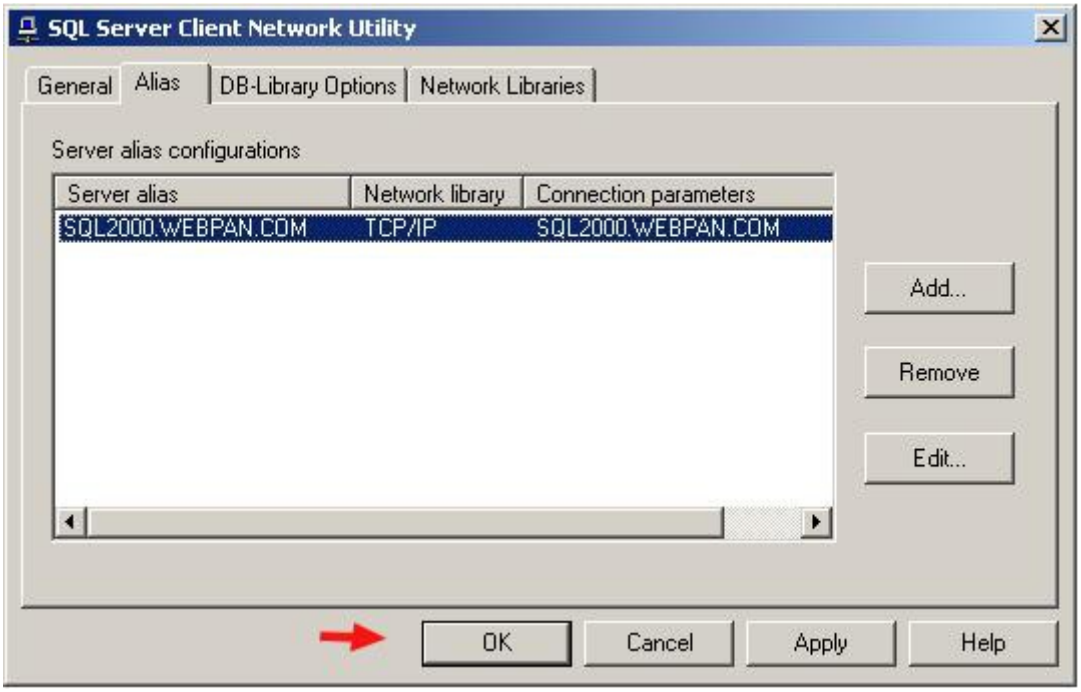


Press **Add...** and then select the **TCP/IP** radio button, as shown below...

Enter the name of your Database in the **Server alias:** text box. Under the **Connection parameters** section, enter the IP Address of the server where your SQL Database is located in the **Server name:** text box. Make sure you have selected the checkbox for **Dynamically determine port**. See the example below for further details...



After you have done entered the appropriate information, press **OK** and the window should now look like this...



Press **Apply**, then **OK**, and you're now ready to setup the connection to your SQL Server Database through Enterprise Manager!

3.2. Server network utility

Thông tin mới - SQL Server 2000 SP3.

In most cases, you do not need to reconfigure Microsoft® SQL Server™ to listen on additional server Net-Libraries. Trong hầu hết trường hợp, bạn không cần phải cấu hình lại Microsoft ® SQL Server ™ để nghe trên máy chủ thêm Net-Thư viện. However, if your server uses a network protocol on which SQL Server, by default, is not listening (for example, if your server is using NWLink IPX/SPX), and the SQL Server server Net-Library for that protocol is not activated to listen for SQL Server clients, you must use SQL Server Network Utility. Tuy nhiên, nếu máy chủ của bạn sử dụng một giao thức mạng trên đó SQL Server, theo mặc định, không được nghe (ví dụ, nếu máy chủ của bạn đang sử dụng NWLink IPX / SPX), và máy chủ SQL Server Net-Thư viện cho các giao thức đó là không được kích hoạt để lắng nghe khách hàng với SQL Server, bạn phải sử dụng SQL Server Network Utility.

Although no server Net-Library configuration actions are necessary to enable SQL Server applications to connect to any instance of SQL Server, you can do the following: Mặc dù không có máy chủ Net-Thư viện hành động cấu hình là cần thiết để cho phép các ứng dụng SQL Server để kết nối với bất kỳ trường hợp của SQL Server, bạn có thể làm như sau:

Manage the server Net-Library properties for each instance of SQL Server on a database computer. Quản lý các máy chủ Net-Thư viện thuộc tính cho mỗi trường hợp của SQL Server trên một máy tính cơ sở dữ liệu.

Enable the server protocols on which the instance of SQL Server will listen. Cho phép các giao thức máy chủ mà trên đó các ví dụ của SQL Server sẽ lắng nghe. For example,

enable the protocol for VIA (Virtual Interface Architecture). Ví dụ, cho phép giao thức cho VIA (Virtual diện Kiến trúc). This protocol provides highly reliable and efficient data transfer, when used with specific hardware. Giao thức truyền dữ liệu này cung cấp độ tin cậy và hiệu quả, khi được sử dụng với phần cứng cụ thể. For VIA to work, you must use the supported hardware. Đối với VIA để làm việc, bạn phải sử dụng phần cứng được hỗ trợ. VIA is not available for systems running Microsoft Windows® 98. VIA không sẵn sàng cho hệ thống chạy Windows ® 98. For more information about VIA, see VIA Clients . Để biết thêm thông tin về VIA, xem VIA khách hàng.

Disable a server protocol that is no longer needed. Vô hiệu hóa một giao thức máy chủ mà không còn cần thiết.

Specify or change the network address on which each enabled protocol will listen. Chỉ định hoặc thay đổi địa chỉ mạng mà mỗi giao thức được kích hoạt sẽ lắng nghe.

When you are entering network addresses manually on a computer running multiple instances of SQL Server, you must not duplicate network addresses between instances. Khi bạn nhập vào địa chỉ mạng bằng tay trên một máy tính chạy nhiều trường hợp của SQL Server, bạn không được trùng lặp địa chỉ mạng giữa các trường. You can specify a comma-separated list of port addresses for the TCP/IP protocol. Bạn có thể chỉ định một dấu phẩy-danh sách tách các địa chỉ cổng TCP / IP. If you specify a list of port addresses, the instance of SQL Server will listen on those ports on each IP address available on the computer running the instance. Nếu bạn chỉ định một danh sách các địa chỉ cổng, các ví dụ của SQL Server sẽ lắng nghe trên các cổng trên mỗi địa chỉ IP có sẵn trên máy tính đang chạy các ví dụ.

If the instance is running on a SQL Server 2000 failover cluster, it will listen on those ports on each IP address selected for SQL Server during SQL Server setup. Nếu ví dụ là

CHƯƠNG 4. PHÁT BIỂU CƠ BẢN T-SQL

T-SQL là sự nâng cao của ngôn ngữ SQL chuẩn. Nó là ngôn ngữ cơ bản dùng để giao tiếp giữa ứng dụng và SQL Server. T-SQL cung cấp khả năng DDL (ngôn ngữ định nghĩa dữ liệu) và DML (ngôn ngữ tương tác dữ liệu) của SQL chuẩn.

Ngôn ngữ DML (Data Manipulation Language) được dùng để tương tác dữ liệu chứa trong các đối tượng của CSDL dùng các phát biểu INSERT, SELECT, UPDATE và DELETE. Những phát biểu này cho phép thêm dòng dữ liệu mới, chọn dòng dữ liệu bằng cách thực hiện truy vấn, cập nhật dòng dữ liệu đang tồn tại và xóa dòng dữ liệu không dùng nữa.

I. CÂU LỆNH TRUY VẤN SELECT

Như chúng ta đã biết dữ liệu chỉ có ý nghĩa khi chúng ta truy xuất và thao tác với chúng. Chúng ta đã quen thuộc với các truy vấn dữ liệu từ một bảng. Câu lệnh dùng để viết truy vấn là **SELECT**. Khi thực hiện **SELECT**, thông tin lưu trữ trong bảng được hiển thị.

Chúng ta có thể mở rộng câu lệnh này để tạo ra các câu truy vấn rất phức tạp và nhiều thành phần.

SELECT là câu lệnh SQL quan trọng nhất. Dùng **SELECT** chúng ta có thể:

+ Hiển thị một số hay tất cả các trường trong bảng

+ Hiển thị một số hay tất cả các bản ghi trong bảng.

+ Hiển thị các thông tin tính toán của dữ liệu trong bảng như giá trị trung bình hoặc tổng của các giá trị trong trường.

+ Liên kết thông tin từ hai hoặc nhiều bảng.

1.1 Cú pháp

SELECT *[DISTINCT]|Columns_list|Expression_list|**

FROM *<Tables_list>*

WHERE *<Conditions>*

GROUP BY *<Columns>*

HAVING *<Conditions_for_group>*

ORDER BY *[ACS| DESC]*

Trong đó:

- Sau **SELECT**: Các thông tin cần đưa ra, đó chính là danh sách các thuộc tính

- Sau **FROM:** Danh sách các tên bảng, từ đó thông tin được lấy ra.

- Sau **WHERE**: Các biểu thức logic, chỉ ra thông tin được lấy ra từ hàm nào hoặc điều kiện nối giữa các bảng.

- Sau **GROUP BY**: Các cột mà trong đó được tính theo từng nhóm.

- Sau **HAVING**: Biểu thức logic chỉ ra thông tin được lấy ra từ nhóm nào.

- Sau **ORDER BY**: Chỉ ra các cột mà trong đó thông tin được sắp xếp theo thứ tự.

o ASC: thông tin được sắp xếp theo chiều tăng dần (ASCendent)

o DESC: thông tin được sắp xếp theo chiều giảm dần(DESCendent)

1.2 Ví dụ:

Xét bảng: **NHANVIEN**

NHANVIEN				
MaNV	HoTen	CongViec	Luong	MaDV
NV001	Phạm Thị Nhàn	Thư ký	500	0001
NV002	Hoàng Thanh Vân	Giáo viên	600	0001
NV003	Hoàng Thị Lan	Giáo viên	200	0002
NV004	Đỗ Trung Dũng	Thư ký	700	0003
...	...	....	...	...

1.3 Đưa ra các cột

1.3.1 Đưa tất cả các cột

Ví dụ: Đưa tất cả các thông tin về nhân viên

SELECT *

FROM NHANVIEN

Kết quả: Toàn bộ bảng trên.

1.3.2 Đưa một số các cột

Ví dụ: Đưa ra Hoten, Luong của các nhân viên

SELECT Hoten, Luong

FROM NHANVIEN

Kết quả:

sl_NV_some_col	
Hoten	Luong
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Hoàng Thị Lan	200
Đỗ Trung Dũng	700

1.3.3 Tránh các giá trị trùng lặp (DISTINCT)

Ví dụ: Đưa ra các công việc khác nhau trong bảng NHANVIEN

SELECT DISTINCT Congviec

FROM NHANVIEN

Kết quả: - Nếu không có lệnh DISTINCT và có DISTINCT:

Congviec
Thư ký
Giáo viên
Giáo viên
Thư ký

CongViec
Giáo viên
Thư ký

1.3.4 Đưa ra các giá trị của các biểu thức

Ví dụ: Đưa ra Hoten, Luongnam (Lương *12) của tất cả các nhân viên

SELECT Hoten, Luong*12

FROM NHANVIEN

Kết quả:

sl_bieuthuc

Hoten	Expr1001
Phạm Thị Nhàn	6000
Hoàng Thanh Vân	7200
Hoàng Thị Lan	2400
Đỗ Trung Dũng	8400

1.3.5 Sử dụng bí danh cột

SELECT Hoten, Luong*12 AS Luongnam

FROM NHANVIEN

Kết quả:

Hoten	LuongNam
Phạm Thị Nhàn	6000
Hoàng Thanh Vân	7200
Hoàng Thị Lan	2400
Đỗ Trung Dũng	8400

1.3.6 Sắp xếp thứ tự (ORDER BY)

Ví dụ: Đưa ra Hoten, Luong sắp xếp theo thứ tự tăng dần/ giảm dần của Luong.

SELECT Hoten, Luong

FROM NHANVIEN

ORDER BY Luong [ASC/ DESC]

Kết quả:

Hoten	Luong
Hoàng Thị Lan	200
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Đỗ Trung Dũng	700

- Trong đó ASC(ascendent) là tăng dần, DESC(descendent) là giảm dần.

- Nếu để giá trị mặc định thì sẽ sắp xếp theo chiều tăng dần.

1.4 Đưa ra các hàng

Lệnh có dạng:

SELECT [DISTINCT]|Columns_list|Expression_list|*

FROM <Tables_list>

WHERE <Conditions>

Điều kiện sau mệnh để Where là một biểu thức lôgic, sử dụng các phép toán sau:

1.4.1 Sử dụng các phép so sánh

= : Toán tử bằng hay tương đương

!= : Toán tử khác hay không tương đương

> : Toán tử lớn hơn

< : Toán tử nhỏ hơn

>= : Toán tử lớn hơn hoặc bằng

<= : Toán tử nhỏ hơn hoặc bằng

Ví dụ: Đưa ra Hoten, Luong của các nhân viên có Luong>300

Hoten	Luong
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Đỗ Trung Dũng	700

1.4.2 Sử dụng các phép logic: AND, OR, NOT

Ví dụ: Đưa ra Hoten, Luong của những nhân viên có công việc là Giáo viên và mức lương >300.

SELECT HoTen, Luong

FROM NHANVIEN

WHERE (Luong>300) AND (Congviec='Giáo viên')

Kết quả:

HoTen	Luong
Hoàng Thanh Vân	600

- Phân tích ví dụ sau:

SELECT HoTen, Luong

FROM NHANVIEN

WHERE (((Luong)>400) AND (Not(CongViec)=('Thư ký'))

OR (Congviec=('Thư ký')))

Kết quả:

HoTen	Luong
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Đỗ Trung Dũng	700

1.4.3 Các toán tử của SQL

- **[NOT] BETWEEN x AND y:** [Không] nằm giữa giá trị X và Y
- **IN (danh sách):** thuộc bất kỳ giá trị nào trong danh sách
- **x [NOT] LIKE y**: Đúng nếu x [không] giống khung mẫu y.

Các ký tự dùng trong khuôn mẫu:

Dấu gạch dưới (_) : Chỉ một ký tự bất kỳ

Dấu phần trăm (%) : Chỉ một nhóm ký tự bất kỳ

- **IS [NOT] NULL:** kiểm tra giá trị rỗng
- **EXISTS**: Trả về TRUE nếu có tồn tại.
- **Phép BETWEEN … AND …**

Ví dụ: Đưa ra những nhân viên có Lương trong khoảng 300 đến 600.

SELECT HoTen, Luong

FROM NHANVIEN

WHERE Luong BETWEEN 300 AND 600

Kết quả:

HoTen	Luong
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600

- **Phép IN (Một tập hợp);**

Ví dụ: Đưa ra những nhân viên có lương hoặc 200, 300, 600.

SELECT HoTen, Luong

FROM NHANVIEN

WHERE Luong IN (200,500,600)

Kết quả:

HoTen	Luong
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Hoàng Thị Lan	200

- **Phép LIKE**

- Ký tự thay thế '%' đại diện cho một nhóm các ký tự chưa biết (trong Access là: *).

- Ký tự thay thế '_' đại diện cho một ký tự chưa biết (trong Access là:?).

- Ví dụ: Đưa ra Hoten, Congviec của các nhân viên có Họ tên bắt đầu bằng chữ 'Hoàng'.

SELECT HoTen, Congviec

FROM NHANVIEN

WHERE Hoten LIKE 'Hoàng*'

Kết quả:

HoTen	Congviec
Hoàng Thanh Vân	Giáo viên
Hoàng Thị Lan	Giáo viên

Ví dụ:

SELECT HoTen, Congviec

FROM NHANVIEN

WHERE Hoten LIKE 'Hoàng Thanh Vân'

- Phép IS [NOT] NULL

Ví dụ:

SELECT * FROM NHANVIEN WHERE Diachi IS NULL

1.5 Sử dụng các hàm

Các HQTCSDL đưa ra các hàm khác nhau, vì thế khi làm việc với HQTCSDL nào chúng ta nên tìm hiểu các hàm và cách sử dụng chúng đối với HQTCSDL đó. Sau đây là một số các loại hàm thường dùng.

1.5.1 Hàm số học

Đầu vào và đầu ra là các giá trị kiểu số.

ROUND(n[,m]): Cho giá trị làm tròn của n (đến cấp m, mặc nhiên m=0)

TRUNC(n[,m]): Cho giá trị n lấy m chữ số tính từ chấm thập phân.

CEIL(n): Cho số nguyên nhỏ nhất lớn hơn hoặc bằng n.

FLOOR(n): Cho số nguyên lớn nhất bằng hoặc nhỏ hơn n.

POWER(m,n): Cho lũy thừa bậc n của m.

EXP(n): Cho giá trị của en

SQRT(n): Cho căn bậc 2 của n, n>=0

SIGN(n): Cho dấu của n.

n<0 có SIGN(n)= -1

n=0 có SIGN(n)= 0

n>0 có SIGN(n)= 1

ABS(n): Cho giá trị tuyệt đối

MOD(m,n): Cho phần dư của phép chia m cho n

1.5.2 Một số hàm kiểu số tham khảo khác:

LOG(m,n) cho logarit cơ số m của n

SIN(n) cosin của n (n tính bằng radian)

COS(n) cho cosin của n (n tính bằng radian)

TAN(n) cotang của n (n tính bằng radian)

1.5.3 Hàm nhóm

o **COUNT():** Đếm số lần xuất hiện của thuộc tính.

o **SUM(colume):** Tính tổng các giá trị của thuộc tính (thuộc loại số học)

o **AVG(colume):** Tính giá trị trung bình các giá trị của thuộc tính (thuộc loại số học)

o **MAX(colume):** Tìm giá trị cực đại của thuộc tính

o **MIN(colume):** Tìm giá trị cực tiểu của thuộc tính.

1.5.4 Sử dụng hàm nhóm

Đối số của các hàm nhóm là tên của thuộc tính mà hàm phải tính toán.

Ví dụ:

Đưa ra lương trung bình, lương lớn nhất, nhỏ nhất của tất cả các nhân viên trong bảng NHANVIEN.

SELECT Avg(Luong) AS LuongTB,

Max(Luong) AS LuongCN,

Min(Luong) AS LuongTN,

COUNT(MaNV) AS TongNV

FROM NHANVIEN

Kết quả:

LuongTB	LuongCN	LuongTN	TongNV
500	700	200	4

1.5.5 Mệnh đề GROUP BY

Mệnh đề **GROUP BY** <các cột> cho phép đưa ra thông tin theo từng nhóm.

Ví dụ: Đưa ra Côngviệc, Lương trung bình của từng loại công việc.

SELECT CongViec, AVG(Luong) AS LuongTB

FROM NHANVIEN

GROUP BY CongViec

Kết quả:

CongViec	LuongTB
Giáo viên	400
Thư ký	600

Có thể thêm vào một mệnh đề WHERE để đưa vào một tiêu chuẩn chọn lựa các dòng. SQL thực hiện cùng một cách xử lý, đầu tiên là loại bỏ các dòng không đáp ứng tiêu chuẩn đã được xác định trong mệnh đề WHERE.

Ví dụ:

SELECT CongViec, AVG(Luong) AS LuongTB
FROM NHANVIEN
WHERE Luong>200
GROUP BY CongViec

Kết quả:

CongViec	LuongTB
Giáo viên	600
Thư ký	600

- Sử dụng mệnh đề GROUP BY để đưa ra các thông tin về các nhóm con trong các nhóm lớn.

Ví dụ: Đưa ra tổng lương của từng nhóm công việc trong từng đơn vị.

SELECT MaDV, CongViec, SUM(Luong) AS TongLuong
FROM NHANVIEN
GROUP BY MaDV, CongViec

Kết quả:

MaDV	CongViec	TongLuong
0001	Giáo viên	600
0001	Thư ký	500
0002	Giáo viên	200
0003	Thư ký	700

Chú ý: Nếu tên các cột ghi sau SELECT không phải là đối số của các hàm nhóm thì phải đưa vào mệnh đề GROUP BY.

Ví dụ:

TongLuong
1100
200
700

1.5.6 Mệnh đề HAVING

Muốn đưa ra các nhóm trên cơ sở thông tin nhóm thì điều kiện phải được viết trong mệnh đề HAVING (Không viết trong mệnh đề WHERE).

Ví dụ: Đưa ra những Congviec và trung bình lương của các công việc có trung bình lương >=300.

SELECT CongViec, Avg(Luong) AS TBLuong
FROM NHANVIEN
GROUP BY CongViec
HAVING (Avg(Luong)>300)
Kết quả:

CongViec	TBLuong
Giáo viên	400
Thư ký	600

Ví dụ: Đưa ra những đơn vị và lương lớn nhất của các đơn vị có lương lớn nhất >=300.

SELECT MaDV, Max(Luong) AS MaxLuong
FROM NHANVIEN
GROUP BY MaDV
HAVING Max(Luong)>300
Kết quả:

MaDV	MaxLuong
0001	600
0003	700

Ghi chú: Mệnh để HAVING là mệnh đề tương đương với WHERE áp dụng cho các nhóm. Nói chung, mệnh đề này chỉ sử dụng nếu đã có chỉ thị một mệnh đề GROUP BY.

1.6 Lấy thông tin từ nhiều bảng

Muốn lấy thông tin từ nhiều bảng ta cần phải thực hiện nối các bảng, điều kiện nối phải được thiết đặt đầu tiên trong mệnh đề Where.

1.6.1 Nối bằng (Equi-Join)

Điều kiện nối là một đẳng thức.

Ví dụ: Đưa ra Hoten, Congviec, TenDV của tất cả nhân viên.

SELECT HoTen, CongViec, TenDV
FROM NHANVIEN, DONVI
WHERE NHANVIEN.MaDV= DONVI.MaDV
Kết quả:

HoTen	CongViec	TenDV
Phạm Thị Nhàn	Thư ký	KHTN

HoTen	CongViec	TenDV
Hoàng Thanh Vân	Giáo viên	KHTN
Hoàng Thị Lan	Giáo viên	DHTL
Đỗ Trung Dũng	Thư ký	DHQG

1.6.2 Bí danh bảng

Được viết ngay bên phải tên bảng trong mệnh đề FROM.

Ví dụ:

SELECT HoTen, CongViec, TenDV
FROM NHANVIEN NV, DONVI DV
WHERE NV.MaDV= DV.MaDV

5.6.3 Nối không bằng (Non Equi-Join)

Ví dụ: Đưa ra Hoten, Congviec, MaBac của tất cả nhân viên

SELECT HoTen, CongViec, MaBac
FROM NHANVIEN NV, BACLUONG BL
WHERE NV.Luong BETWEEN BL.BacThap AND BL.BacCao

Kết quả:

sl_non_equi

HoTen	CongViec	MaBac
Phạm Thị Nhàn	Thư ký	1
Hoàng Thanh Vân	Giáo viên	2
Đỗ Trung Dũng	Thư ký	3

Chú ý: Nếu ngoài các điều kiện nối còn có thêm các điều kiện khác thì điều kiện nối phải được viết trước.

Ví du: Đưa ra HoTen, Congviec, TenDV, Luong của những nhân viên có Luong>=500.

SELECT HoTen, CongViec, TenDV, Luong
FROM NHANVIEN AS NV, DONVI AS DV
WHERE (NV.MaDV=DV.MaDV) AND (Luong>=500);

Kết quả:

HoTen	CongViec	TenDV	Luong
Phạm Thị Nhàn	Thư ký	KHTN	500

HoTen	CongViec	TenDV	Luong
Hoàng Thanh Vân	Giáo viên	KHTN	600
Đỗ Trung Dũng	Thư ký	DHQG	700

1.6.4 Nối bảng với chính nó

Giả sử trong bảng NHANVIEN ta thêm 1 thuộc tính (cột) là MaPT (Mã phụ trách) để lưu mã của nhân viên phụ trách trực tiếp 1 nhân viên khác. Cụ thể như sau:

SELECT NV.MaNV, NV.Hoten, PT.MaNV, PT.Hoten

FROM NHANVIEN NV, NHANVIEN PT
WHERE (NV.MaNV=PT.MaPT) AND (NV.Luong>PT.Luong)
Kết quả:

NV.MaNV	NV.Hoten	PT.MaNV	PT.Hoten
NV002	Hoàng Thanh Vân	NV001	Phạm Thị Nhàn
NV002	Hoàng Thanh Vân	NV003	Hoàng Thị Lan

1.6.5 Thực hiện kết nối thông qua từ khóa Join

Ta có thể thực hiện lấy dữ liệu từ hai bảng thông qua từ khóa JOIN.

INNER JOIN (nối trong)

Cú pháp:

SELECT field1, field2, field3
FROM table1
INNER JOIN table2
ON table1.keyfield=table2.foreign_keyfield;

Ví dụ: Giả sử có hai bảng:

KHACHHANG:

MaKH	TenKH
01	Hoàng Thanh Vân
02	Lê Thị Nhàn
03	Phan Thanh Hòa
04	Phạm Hồng Thanh

DONHANG:

MaSP	TenSP	MaKH
H102	Máy in	01
H106	Bàn	03
H301	Ghế	03

Yêu cầu: Đưa ra tên khách hàng và tên sản phẩm khách hàng đó mua.

SELECT KHACHHANG.TenKH, DONHANG.TenSP
FROM KHACHHANG
INNER JOIN DONHANG
ON KHACHHANG.MaKH=DONHANG.MaKH
Kết quả:

TenKH	TenSP
Hoàng Thanh Vân	Máy in
Phan Thanh Hòa	Bàn
Phan Thanh Hòa	Ghế

INNER JOIN trả về tất cả các dòng từ hai bảng thỏa mãn điều kiện. Nếu

những dòng dữ liệu có bên table1 mà không có trong table2 thì sẽ không được hiển thị (khác với …)

LEFT JOIN

Cú pháp:

```
SELECT field1, field2, field3
FROM table1
LEFT JOIN table2
ON table1.keyfield = table2.foreign_keyfield
```

Ví dụ:

```
SELECT KHACHHANG.TenKH, DONHANG.TenSP
FROM KHACHHANG
LEFT JOIN DONHANG
ON KHACHHANG.MaKH=DONHANG.MaKH
```

Kết quả:

TenKH	TenSP
Hoàng Thanh Vân	Máy in
Lê Thị Nhàn	
Phan Thanh Hòa	Bàn
Phan Thanh Hòa	Ghế
Phạm Hồng Thanh	

LEFT JOIN trả về tất cả các dòng có ở bảng thứ nhất, mặc dù ở bảng thứ hai không thỏa mãn phép toán. Nếu dữ liệu có ở bảng thứ nhất mà không có ở bảng thứ hai thì dữ liệu vẫn hiển thị.

RIGHT JOIN

Cú pháp

```
SELECT field1, field2, field3
FROM table1
RIGHT JOIN table2
ON table1.keyfield =
table2.foreign_keyfield
```

Ví dụ

```
SELECT KHACHHANG.TenKH, DONHANG.TenSP
FROM KHACHHANG
RIGHT JOIN DONHANG
ON KHACHHANG.MaKH=DONHANG.MaKH
```

Kết quả:

TenKH	TenSP
Hoàng Thanh Vân	Máy in
Phan Thanh Hòa	Bàn
Phan Thanh Hòa	Ghế

RIGHT JOIN trả về tất cả các dòng có ở bảng 2, mặc dù bảng 1 không thỏa mãn phép toán. Nếu dữ liệu có ở bảng 2 mà không có ở bảng 1 thì vẫn được hiển thị.

1.7 Thực hiện các phép toán trên tập hợp

Các phép toán trên tập hợp gồm: Hợp (UNION) hoặc UNION ALL, Giao (INTERSECT), Trừ (MINUS)

Điều kiện thực hiện các phép toán trên tập hợp: Các bảng tham gia vào phép toán phải có cùng số cột như nhau.

- Phép UNION.

Ví dụ: Đưa ra những công việc trong đơn vị 1 có MaDV là 0001 và đơn vị 2 có MaDV là 0002.

NHANVIEN

NHANVIEN					
MaNV	HoTen	CongViec	Luong	MaDV	MaPT
NV001	Phạm Thị Nhàn	Thư ký	500	0001	NV002
NV002	Hoàng Thanh Vân	Giáo viên	600	0001	NV003
NV003	Hoàng Thị Lan	Giáo viên	200	0002	NV002
NV004	Đỗ Trung Dũng	Thư ký	700	0003	NV002
NV005	Đỗ Văn Hải	Bảo vệ	100	0001	NV002
NV006	Nguyễn Nam Hải	Giám đốc	1000	0001	

```
SELECT CongViec
FROM NHANVIEN
WHERE MaDV='0001'
UNION
SELECT CongViec
FROM NHANVIEN
WHERE MaDV='0002'
```

Kết quả:

CongViec
Bảo vệ
Giám đốc
Giáo viên
Thư ký

- Phép INTERSECT: Nếu thay UNION bằng INTERSECT thì kết quả sẽ đưa ra những công việc vừa có trong đơn vị 1, vừa có trong đơn vị 2.

- Phép MINUS: Nếu thay UNION bằng MINUS thì kết quả sẽ đưa ra những công việc chỉ có trong đơn vị 1, mà không có trong đơn vị 2.

1.8 Các câu hỏi lồng nhau

- Là các lệnh SELECT trong đó có chứa các lệnh SELECT khác.
- Các câu lệnh SELECT bên trong nằm sau mệnh đề WHERE hoặc HAVING của SELECT bên ngoài.

- Cách thực hiện của câu lệnh SELECT lồng nhau:

- Thực hiện lệnh SELECT bên trong.
- Sử dụng kết quả của lệnh SELECT bên trong để thực hiện lệnh SELECT bên ngoài.
- Số các lệnh SELECT lồng nhau được phép là 255.

5.8.1 Lệnh SELECT bên trong cho kết quả là 1 hàng

Xét bảng NHANVIEN trên.

Ví dụ: Đưa ra Hoten, TenDV, Congviec, Luong của những người có lương lớn hơn lương trung bình của toàn bộ nhân viên.

Đối với yêu cầu này ta cần làm những việc sau:

- Đưa ra trung bình lương của tất cả các nhân viên.
- Đưa ra những nhân viên thỏa mãn yêu cầu.

```
SELECT Hoten, TenDV, Congviec, Luong
FROM NHANVIEN AS NV, DONVI AS DV
WHERE (NV.MaDV= DV.MaDV)
AND (Luong> ( SELECT AVG(Luong)
FROM NHANVIEN ))
```

Kết quả:

Hoten	TenDV	Congviec	Luong
Nguyễn Nam Hải	KHTN	Giám đốc	1000
Hoàng Thanh Vân	KHTN	Giáo viên	600
Đỗ Trung Dũng	DHQG	Thư ký	700

Ví dụ 2:Đưa ra những nhân viên có lương lớn hơn người có lương lớn nhất trong đơn vị có tên là DHTL.

Công việc:

- Tìm MaDV có tên đơn vị là DHTL.
- Tìm mức lương lớn nhất trong đơn vị này.
- Tìm những nhân viên có lương thỏa mãn yêu cầu.

```
SELECT Hoten, TenDV, Congviec, Luong
FROM NHANVIEN AS NV, DONVI AS DV
WHERE (NV.MaDV= DV.MaDV)
AND (Luong> ( SELECT MAX(Luong)
```

FROM NHANVIEN
WHERE MaDV =
SELECT MaDV
FROM DONVI
WHERE TenDV='DHTL')))
Kết quả:

Hoten	TenDV	Congviec	Luong
Nguyễn Nam Hải	KHTN	Giám đốc	1000
Phạm Thị Nhàn	KHTN	Thư ký	500
Hoàng Thanh Vân	KHTN	Giáo viên	600
Đỗ Trung Dũng	DHQG	Thư ký	700

1.8.2 Lệnh SELECT bên trong cho kết quả là nhiều hàng

Giả sử lệnh SELECT bên trong có dạng:
SELECT MaDV,MAX(Luong) AS LuongLN,MIN(Luong) AS LuongNN
FROM NHANVIEN
GROUP BY MaDV
Kết quả:

MaDV	LuongLN	LuongNN
0001	1000	100
0002	200	200
0003	700	700

Như vậy, kết quả của câu lệnh SELECT bên trong cho kết quả là một tập giá trị, thì ta phải sử dụng các phép toán so sánh với tập hợp, không sử dụng được các phép toán so sánh như (>, <, =, ….).

Toán tử SOME/ANY/ALL/NOT IN/EXITS

[NOT] IN : Không thuộc

ANY và SOME : So sánh một giá trị với mỗi giá trị trong một danh sách hay trong kết quả trả về của câu hỏi con, phải sau toán tử =

ALL : So sánh một giá trị với mọi giá trị trong danh sách hay trong kết quả trả về của câu hỏi con.

EXISTS : Trả về TRUE nếu có tồn tại.

- Phép toán IN:

Ta có biểu thức: <Giá trị> IN {Tập hợp} trả lại kết quả = TRUE nếu tập hợp các giá trị nằm trong tập hợp đứng sau IN.

Bảng NHANVIEN:

NHANVIEN

NHANVIEN					
MaNV	HoTen	CongViec	Luong	MaDV	MaPT
NV001	Phạm Thị Nhàn	Thư ký	500	0001	NV002
NV002	Hoàng Thanh Vân	Giáo viên	600	0001	NV003
NV003	Hoàng Thị Lan	Giáo viên	200	0002	NV002
NV004	Đỗ Trung Dũng	Thư ký	700	0003	NV002
NV005	Đỗ Văn Hải	Bảo vệ	100	0001	NV002
NV006	Nguyễn Nam Hải	Giám đốc	1000	0001	
NV007	Nguyễn Hoàng Lan	Giáo viên	500	0001	NV006
NV008	Nguyễn Thanh Ngọc	Giáo viên	700	0002	

Ví dụ 1: Đưa ra Hoten, MaDV, Luong của các nhân viên có Luong=Luong thấp nhất trong đơn vị của họ.

Công việc:

- Tính lương thấp nhất cho từng đơn vị
- So sánh (MaDV, Luong) của tất cả nhân viên với tập hợp đó.

```
SELECT Hoten, MaDV, Luong
FROM NHANVIEN
WHERE (MaDV, Luong) IN (Select MaDV, Min(Luong)
From NHANVIEN
Group by MaDV)
```

Đối với một vài HQTCSDL, tập hợp trong phép toán IN chỉ bao gồm 1 giá trị. Ví dụ không thể so sánh (MaDV, Luong), chỉ được phép so sánh MaDV hoặc Luong.

Ví dụ 2: Đưa ra Hoten, MaDV, Luong của các nhân viên có Luong=Luong thấp nhất trong một đơn vị nào đó.

```
SELECT NHANVIEN.MaNV, NHANVIEN.Hoten, NHANVIEN.Luong
FROM NHANVIEN
WHERE NHANVIEN.Luong IN (
SELECT Min(NHANVIEN.Luong) AS MinOfLuong
FROM NHANVIEN
GROUP BY NHANVIEN.MaDV)
```

Kết quả:

MaNV	Hoten	Luong
NV003	Hoàng Thị Lan	200
NV004	Đỗ Trung Dũng	700
NV005	Đỗ Văn Hải	100
NV008	Nguyễn Thanh Ngọc	700

Phép toán ALL

Kết hợp với các phép so sánh thông thường để so sánh một giá trị với 1 tập hợp.

Giá_trị > ALL{Tập hợp}: Biểu thức TRUE nếu giá trị so sánh > tất cả các giá trị trong tập hợp.

Ví dụ:

5> ALL(2,3,4): TRUE

5> ALL(2,4,6): FALSE

Phép toán ANY

Giá_trị> ANY{Tập hợp}: Biểu thức TRUE nếu giá trị so sánh > một giá trị nào đó trong tập hợp.

Ví dụ: 5> ANY(2,4,6): TRUE

Ví dụ: Đưa ra Hoten, Luong của các nhân viên có Luong lớn nhất của đơn vị có mã đơn vị là 0002.

```
SELECT Hoten, Luong
FROM NHANVIEN
WHERE Luong> ALL(
Select Luong
From NHANVIEN
Where MaDV ='0002')
```

Kết quả select trong là:

Luong
200
700

Kết quả của cả câu lệnh:

Hoten	Luong
Nguyễn Nam Hải	1000

Nếu thay ALL = ANY thì kết quả:

Hoten	Luong
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Đỗ Trung Dũng	700
Nguyễn Nam Hải	1000
Nguyễn Hoàng Lan	500
Nguyễn Thanh Ngọc	700

1.8.3 Mệnh đề HAVING trong SELECT lồng nhau.

Mệnh đề HAVING được sử dụng khi có điều kiện nhóm

Ví dụ: Đưa ra MaDV, AVG(Luong) của đơn vị có trung bình lương lớn hơn lương nhỏ nhất của đơn vị có mã đơn vị là 0003.

- Tính lương lớn nhất của đơn vị có mã đơn vị là 0003
- Đưa ra những đơn vị có TBLương > Lương nhỏ nhất vừa tính được

```
SELECT MaDV, Avg(Luong) AS AvgOfLuong
FROM NHANVIEN
GROUP BY NHANVIEN.MaDV
HAVING AVG(Luong)>
Select Min(Luong)
From NHANVIEN
Where MaDV='0002')
```

1.8.4 Mệnh đề ORDER BY trong SELECT lồng nhau

Mỗi lệnh SELECT chỉ có 1 mệnh đề ORDER By duy nhất. Một lệnh SELECT lồng nhau được coi là một lệnh SELECT. Vì vậy, nếu muốn sắp xếp dữ liệu thì mệnh để ORDER BY phải là mệnh đề cuối cùng của lệnh SELECT ngoài cùng, các lệnh SELECT bên trong không có ORDER BY.

1.9 Các lệnh lồng nhau liên kết

Các lệnh liên kết cũng là các lệnh SELECT lồng nhau nhưng nó có cách thực hiện khác các lệnh lồng nhau thông thường.

Các bước thực hiện:

- Xét 1 hàng của bảng
- Sử dụng dữ liệu của hàng đó để thực hiện lệnh SELECT bên trong.
- Sử dụng kết quả của SELECT bên trong để thực hiện SELECT bên ngoài
- Lặp lại các bước trên cho đến khi hết các hàng được xét.

Ví dụ: Có bảng NHANVIEN

NHANVIEN					
MaNV	HoTen	CongViec	Luong	MaDV	MaPT
NV001	Phạm Thị Nhàn	Thư ký	500	0001	NV002
NV002	Hoàng Thanh Vân	Giáo viên	600	0001	NV003
NV003	Hoàng Thị Lan	Giáo viên	200	0002	NV002
NV004	Đỗ Trung Dũng	Thư ký	700	0003	NV002
NV005	Đỗ Văn Hải	Bảo vệ	100	0001	NV002
NV006	Nguyễn Nam Hải	Giám đốc	1000	0001	
NV007	Nguyễn Hoàng Lan	Giáo viên	500	0001	NV006
NV008	Nguyễn Thanh Ngọc	Giáo viên	700	0002	

Đưa ra Hoten, MaDV, Luong của những nhân viên có Luong> LuongTB của đơn vị của họ.

```
SELECT NHANVIEN.HoTen, NHANVIEN.MaDV, NHANVIEN.Luong
FROM NHANVIEN
WHERE ((NHANVIEN.Luong)> (Select AVG(Luong)
From NHANVIEN NV1
```

Where NV1.MaDV= NHANVIEN.MaDV))

II. Các lệnh INSERT, UPDATE,DELETE

Trong SQL, người ta dung ba câu lệnh INSERT, UPDATE, DELETE để thao tác trên dữ liệu.

2.1 Lệnh INSERT

Cú pháp:

INSERT [INTO] <TableName> (Column1, Column2, …, Columnn)

VALUES (Values1, Values2,…., Valuesn)

Lệnh này được dùng để xen thêm một hoặc nhiều dòng (bản ghi) mới vào một bảng. Dạng đơn giản nhất của lệnh này là thêm mỗi lần 1 dòng. Nó đòi hỏi phải nên tên của bảng, tên các thuộc tính và giá trị cần gán cho chúng. Nếu không nêu tên các thuộc tính thì điều đó có nghĩa là tất cả các thuộc tính trong bảng đều cần được thêm giá trị theo thứ tự từ trái sang phải.

Ví dụ 1: Giả sử ta đã có cấu trúc bảng NHANVIEN(MaNV, TenNV, Diachi, Tuoi)

- Thêm bản ghi mới có tất cả các trường cho bảng NHANVIEN. Vì tất cả các thuộc tính trong bảng đều được thêm giá trị nên ta không cần có danh sách các thuộc tính ngay sau tên bảng NHANVIEN.

```
INSERT INTO NHANVIEN
VALUES('DHTL05','Nguyễn Công Thành', 'KhoaCNTT',22 )
```

- Thêm bản ghi mới vào bảng có hạn chế của các trường thì cần chỉ ra tường minh danh sách các trường tương ứng.

Ví dụ: Thêm nhân viên vào bảng nhân viên nhưng không có thông tin về tuổi.

```
INSERT INTO DOCGIA(MaDG,TenDG,DiaChi)
VALUES('DHTL06','Nguyễn Phương Lan', 'Khoa May' )
```

Chú ý: các thuộc tính không được thiết lập giá trị trong câu lệnh trên sẽ lấy giá trị mặc định hoặc không xác định (Null).

- Ngoài ra chúng ta còn có thể them dữ liệu cho bảng từ giá trị của bảng khác.

Cú pháp:

INSERT [INTO] <TableName> (Column1, Column2, …, Columnn)

SELECT Select_list FROM <Tables>

Ví dụ: Insert into NHANVIEN_tam (TenNV, Tuoi)

Select TenNV, Tuoi from NHANVIEN where Tuoi >20.

- Câu lệnh Insert sẽ gặp lỗi trong trường hợp một trường nào đó có rang buộc NOT NULL, nhưng lại không được thiết lập giá trị khi chèn một bản ghi mới.

Ví dụ: Insert into NHANVIEN (HoTen, Ten, MaDV)

Values ('Hoàng', 'Hải', 5);

Lệnh trên sẽ gặp lỗi, bởi vì thuộc tính MaDV có ràng buộc NOT NULL nhưng không được thiết lập.

2.2 Lệnh UPDATE

Lệnh này được dung để sửa đổi giá trị các trường của các bản ghi trong bảng.

Cú pháp:

UPDATE <Table _name>

SET (Colunm_name = <new value>)

WHERE <Condition>

Trong đó: Mệnh đề **SET** dùng để xác định giá trị cập nhật cho các trường.

Mệnh đề **WHERE** dùng để lựa chọn các bản ghi cần cập nhật dữ liệu

Ví dụ: để thay đổi địa điểm và mã số đơn vị của dự án số 10 thành 'Hà Đông' và 5 chúng ta sử dụng câu lệnh sau:

```
UPDATE DuAn
SET     DiaDiemDA= 'Hà Đông', MaDV = 5
WHERE MaDA = 10;
```

- Nhiều bản ghi có thể bị sửa đổi trong một lệnh UPDATE.

Ví dụ: Tăng 10% lương cho tất cả các nhân viên trong đơn vị 'Nghiên Cứu'

```
UPDATE NhanVien
SET Luong = Luong*1.1
WHERE MaDV in (SELECT MaDV
                    FROM DonVi
                     WHERE TenDV = 'Nghiên Cứu');
```

Chú ý: Mỗi lệnh UPDATE tại một thời điểm chỉ thao tác được trên một trường duy nhất mà thôi. Muốn thực hiện cập nhật trên nhiều trường hãy sử dụng nhiều câu lệnh

2.3 Lệnh DELETE

Lệnh này dùng để xóa các bản ghi trong bảng.

Cú pháp:

DELETE FROM <Table_name> WHERE <Conditions>

Lệnh này gồm 1 mệnh đề DELETE FROM để chỉ ra tên gọi của bảng được xét, và một mệnh để WHERE để chỉ ra các dòng cần phải xóa. Như vậy, ta có thể cùng lúc xóa được nhiều dòng nếu dòng đó thỏa mãn điều kiện. Muốn xóa mọi dòng của một bảng thì không cần đưa vào mệnh đề WHERE.

Ví dụ: Xóa một bản ghi (dòng) có MaDG = 'DHTL01' trong bảng DocGia

```
DELETE FROM DocGia WHERE MaDG = 'DHTL01'
```

- Xóa những độc giả có địa chỉ là 41NC có trong bảng DocGia

```
DELETE FROM DocGia WHERE DiaChi = '41NC'
```

CHƯƠNG 5. TẠO VÀ SỬA ĐỔI BẢNG DỮ LIỆU

T-SQL là sự nâng cao của ngôn ngữ SQL chuẩn. Nó là ngôn ngữ cơ bản dùng để giao tiếp giữa ứng dụng và SQL Server. T-SQL cung cấp khả năng DDL (**Data Definition Language**:ngôn ngữ định nghĩa dữ liệu) và DML (**Data Manipulation Language**: ngôn ngữ tương tác dữ liệu). của SQL chuẩn.

DDL – Data Definition Language, được dùng để định nghĩa và quản lý các đối tượng CSDL như CSDL, Bảng và View. Các phát biểu DDL thường bao gồm các lệnh CREATE, ALTER và DROP cho tường đối tượng.

Để chạy ta dùng công cụ Query Analyzer.

5. 1. CREATE TABLE (19)

Create table được dùng để tạo bảng với một số thông số như tên bảng, thuộc tính và các ràng buộc. Bảng được tạo là bảng rỗng cho đến khi nó được thêm dữ liệu vào.

5.1.1. Cú pháp để tạo một bảng sử dụng T-SQL

CREAT TABLE <tên bảng>

(<thuộc tính 1> <kiểu dữ liệu> <cỡ> <ràng buộc>,

<thuộc tính 2> <kiểu dữ liệu> <cỡ> <ràng buộc>,

……

<thuộc tính n> <kiểu dữ liệu> <cỡ> <ràng buộc>,

[constraint mệnh đề])

Trong đó, mệnh đề constraint cho phép ta khai báo các ràng buộc dữ liệu.

Ví dụ 1: Tạo bảng Hocsinh (MaHS, TenHS, Ngaysinh, Gioitinh, Diachi):

Create table HocSinh

```
( MaHS          int          Not NULL,
  TenHS         varchar(30)  NOT NULL,
  NgaySinh      Date time,
  Gioitinh      bit,
  Diachi        varchar (50) )
```

Bảng này sẽ được nhận một tên gọi và một cấu trúc (danh sách tên các thuộc tính và một vài đặc trưng). Khi mới được tạo, bảng chưa có dữ liệu, chỉ là một cấu trúc logic có thể tiếp nhận các dữ liệu.

a). Tên của bảng

Tên của bảng được xác định ngay sau lệnh Create table.

Mỗi hệ quản trị csdl có một quy tắc đặt tên riêng.

+ Tên bảng phải bắt đầu bằng một chữ cái, dưới 30 kí tự (chữ cái, chữ số, và dấu '_').

+ Tên bảng phải khác tên gọi khác của bảng hay của khung nhìn và với tên gọi đã dành riêng của SQL.

+ Không phân biệt hoa, thường.

b). Xác định các thuộc tính

Trong lệnh tạo bảng ta phải xác định cấu trúc của bảng. Cần phải xác định mỗi thuộc tính của một định nghĩa kết thúc bằng dấu ',' và gồm:

+ tên thuộc tính

+ Loại dữ liệu và độ dài

+ Các ràng buộc có liên quan

c). Các loại dữ liệu

5.1.2 Các loại ràng buộc trong bảng dữ liệu

Các dạng constraint gồm:

- NOT NULL
- UNIQUE
- PRIMARY KEY

- FOREIGN KEY (REFERENTIAL)
- CHECK
- DEFAULT
- IDENTITY

a). NOT NULL _không rỗng

Khi có mệnh đề NOT NULL có trong định nghĩa của một cột thì ta bắt buộc thuộc tính này phải có giá trị. Nếu ta không chỉ thị gì trong định nghĩa của thuộc tính thì nó có thể có hoặc không có giá trị.

Ví dụ 2: Tạo bảng Lop (MaLop, TenLop)

Create table Lop

(MaLop int NOT NULL, TenLop char (30) NOT NULL)

b) UNIQUE – Duy nhất

Ràng buộc duy nhất (UNIQUE constraint) yêu cầu tất cả các giá trị trong trường phải là duy nhất, giá trị của trường phải khác nhau trên tất cả các bản ghi của bảng.

Một bảng có thể có nhiều ràng buộc duy nhất, và tất cả các bản ghi phải thoả mãn phải thoả mãn các ràng buộc đó. Ràng buộc duy nhất đảm bảo tính toàn vẹn thực thể vì tất cả các bản ghi trong bảng bao giờ cũng khác nhau.

Chúng ta có thể sử dụng ràng buộc duy nhất để đảm bảo các giá trị trùng lặp không được nhập vào một trường nào đó. Khi đó chức năng của ràng buộc duy nhất giống với chức năng của ràng buộc khoá chính ngoại trừ nó cho phép các giá trị null. Cả hai ràng buộc đó đều đảm bảo tính duy nhất, nhưng ràng buộc duy nhất sẽ được sử dụng thay vì ràng buộc khoá chính khi chúng ta muốn đảm bảo tính nhất quán của:

+ Một trường hay một tổ hợp các trường không phải khoá chính: có những trường hợp chúng ta cần nhiều trường phải có các giá trị duy nhất. Khi đó, chúng ta không thể thiết lập các trường đó là khoá chính được vì trong một bảng chỉ có duy nhất một ràng buộc khoá chính. Cho nên ta có thể đặt ràng buộc duy nhất trên các trường đó vì một bảng cho phép có nhiều ràng buộc duy nhất.

+ Một trường cho phép nhận giá trị null: Các ràng buộc duy nhất có thể được định nghĩa trên các trường cho phép các giá trị null, trong khi ràng buộc khoá chính chỉ có thể được định nghĩa trên những trường không cho phép các giá trị null.

+ Một ràng buộc duy nhất cũng có thể được tham chiếu bởi một ràng buộc khoá ngoài.

Ví dụ 3: Tạo bảng Lop (MaLop, TenLop)

Create table Lop

(MaLop int NOT NULL, TenLop char (30) NOT NULL

CONSTRAINT UNQ_TenLop UNIQUE(TenLop))

c). PRIMARY KEY – khoá chính

Chỉ ra ràng buộc duy nhất (giống UNIQUE) tuy nhiên khoá là dạng khoá UNIQUE cấp cao nhất. Một bảng chỉ có thể có một khoá chính. Các giá trị trong khoá chính phải not null.

Cú pháp

[CONSTRAINT constraint_name]
PRIMARY KEY [CLUSTERED|NONCLUSTERED]

[(colname [,colname2 [...,colname16]])]

Ví dụ 4: Tạo bảng KHACH_HANG:

USE QLBanHang

GO

CREATE TABLE KHACH _HANG

(MA-KH int NOT NULL,

HO_KH nvarchar (50),

TEN_KH nvarchar (20),

DIEN_THOAI char (20),

Constraint MA_KH_PK PRIMARY KEY (MA_KH))

GO.

Hoặc có thể viết câu lệnh như sau:

CREATE TABLE KHACH _HANG

(MA-KH int NOT NULL primary key,

HO_KH nvarchar (50),

TEN_KH nvarchar (20),

DIEN_THOAI char (20)

Chú ý: *bắt đầu mỗi phát biểu T_SQL có thể thêm dòng để chọn CSDL*

USE < CSDL _Name>

GO

d) FOREIGN KEY – khoá ngoại

Chỉ ra mối liên hệ ràng buộc tham chiếu giữa bảng này với bảng khác.

Từ khoá ON DELETE CASCADE được chỉ định trong dạng khoá này để khi dữ liệu cho bị xóa thì dữ liệu con cũng tự động vị xoá theo.

Cú pháp:

[CONSTRAINT constraint_name]
[FOREIGN KEY (colname [,colname2 [...,colname16]])]
REFERENCES reference_table [(ref_colname[,ref_colname2[...,ref_colname 16]])]

Ví dụ 5: Hai bảng DONVI và bảng NHANVIEN có mối quan hệ cha – con (1_N). Thuộc tính MaDV trong bảng NHANVIEN(bảng con) là khoá ngoại, được tham chiếu từ thuộc tính MaDV của bảng DONVI(bảng cha).

Tạo 2 bảng như sau:

CREATE TABLE DONVI
(MaDV char(2) primary key,
TenDV char(20) not null)
CREATE TABLE NHANVIEN
(MaNV char(10) primary key,
TenNV char(30) not null,
Diachi char(50),
madv char(2)
CONSTRAINT k_n_madv FOREIGN KEY(madv) REFERENCES
DONVI(MaDV))

e) CHECK- Ràng buộc kiểm tra giá trị

Ràng buộc CHECK được sử dụng để yêu cầu các giá trị trong cột, hoặc khuôn dạng dữ liệu trong cột phải theo một quy tắc nào đó. Trên một cột có thể có nhiều ràng buộc này. Để khai báo một rang buộc CHECK cho một cột nào đó ta dùng cú pháp sau.

Cú pháp:

[CONSTRAINT constraint_name]

CHECK (expression)

Trong đó, expression là một biểu thức logic. Sau khi có ràng buộc này, giá trị nhập vào cho cột phải thoả mãn điều kiện mới được chấp nhận.

Ví dụ: CREATE TABLE NHANVIEN

(MaNV CHAR(10) NOT NULL PRIMARY KEY,

TenNV CHAR(30),

Luong NUMBER(10,2)

CONSTRAINT CK_SAL CHECK(SAL>500))

f) DEFAULT-Mặc định

Ràng buộc **DEFAULT** được sử dụng để quy định giá trị mặc định cho một cột. Giá trị này sẽ tự động gán cho cột nếu người sử dụng không nhập vào khi bổ sung bản ghi.

Cú pháp:

[CONSTRAINT constraint_name]

DEFAULT {const_expression/nonarguments_function/NULL}

Ví dụ:

CREATE TABLE NHANVIEN

(MaNV char(10) primary key,

TenNV char(30) not null,

Gioitinh char (3) default ‘nam’)

g) Thuộc tính IDENTITY

Phương pháp thứ 3 để thiết lập ràng buộc thực thể là áp dụng thuộc tính IDENTITY cho một trường. Thuộc tính này có thể được áp dụng cho trường có kiểu dữ liệu decimal, int, smallint, numeric. Nó sinh ra giá trị duy nhất trong bảng.

Theo mặc định, giá trị bắt đầu thiết lập bởi thuộc tính này là 1. Thuộc tính này được thiết lập khi tạo bảng.

Cú pháp:

Column_name Data type IDENTITY (seed, increment)

Trong đó: Column_name: tên trường được gán thuộc tính Identity.

Seed: giá trị khởi tạo của cột identity

Increment: bước nhảy được sử dụng để sinh ra giá trị tiếp theo cho cột. Nó cũng có thể là số âm

Ví dụ: để sinh số thứ tự cho trường STT trong bảng học sinh

Create table hocsinh

(tenhs char(20) not null,

STT int identity (1,1))

Giá trị bắt đầu của trường STT được đặt là 1, và mỗi khi thêm một bản ghi, nó tự động tăng thêm 1.

5. 2. CREATE VIEW (7)

Là lệnh để tạo khung nhìn. Một khung nhìn được xác định bởi tên, danh sách các thuộc tính. Trong trường hợp không chỉ ra danh sách thuộc tính, CREATE VIEW sẽ mặc định lấy tất cả các trường hợp trong các bảng tiên đề.

Cú pháp:

CREATE VIEW <Viewname> [WITH SCHEMABINDING]
AS <Select_Statement>
[WITH CHECK OPTION]

Trong đó:

+ Viewname: là tên của view cần tạo

+ WITH SCHEMABINDING: Đảm bảo rằng tất cả các đối tượng có trong câu lệnh tạo View không thể được xoá khi View đang tồn tại.

+ WITH CHECK OPTION: Đảm bảo rằng nếu bạn muốn sửa hoặc thêm dữ liệu thông qua View thì những dữ liệu đó phải thoả mãn tất cả các điều kiện trong câu lệnh Select

Sau đây là các ví dụ:

Ví dụ 1: Tạo view KHACH_HANG_VW trên bảng KHACH_HANG chỉ có thể truy cập cột HO_KH và TEN_KH thì thực hiện như sau:

```
USE QLBanHang
GO
CREATE VIEW KHACH_HANG_VW
AS
   SELECT HO_KH,  TEN_KH
FROM KHACH_HANG
```

GO

Ví dụ 2: tạo view MatHang_VW trên bảng MatHang chỉ chứa những dòng dữ liệu có MaLoaiHang là CPU bạn thực hiện như sau:

```
USE QLBanHang
GO
CREATE VIEW MatHang _VW
AS
    SELECT *
    FROM MatHang
    WHERE MaLoaiHang = 'CPU'
GO
```

Ví dụ 3.tạo view TONG _GIA_CPU_VW chứa tổng giá trị của các loại mặt hàng là CPU như sau, đặt tên cho cột SUM (DonGia) là TONG:

```
CREATE VIEW TONG_GIA_CPU_VW
AS
    SELECT MaLoaiHang, SUM(DonGia) TONG
    From MatHang
    Where MaLoaiHang = 'CPU'
GO.
```

Ví dụ 4: tạo view TONG_GIA_VW chứa tổng giá trị của từng loại mặt hàng.

```
CREATE VIEW TONG_GIA_VW
AS
    SELECT MaLoaiHang, SUM (DonGia) TONG
    FROM MatHang
    GROUP BY MaLoaiHang
GO.
```

Ví dụ 5: **CREATE VIEW NhanVienDuAn1**

AS SELECT HoTen, Ten, TenDA, SoGio

FROM NhanVien, DuAn, NhanVienDuAn

WHERE NhanVien.MaNV = NhanVienDuAn.MaVN AND DuAn.MaDA = NhanVienDuAn.MaDA;

Vì trong ví dụ này không chỉ ra danh sách các thuộc tính cho NhanVienDuAn1 nên khung nhìn này lấy toàn bộ các thuộc tính sau mệnh đề SELECT.

Ví dụ 6: **CREATE VIEW DonVi_info**

(TenDV varchar(15),

SoNhanVien integer,

TongLuong integer);

AS SELECT TenDV, COUNT(*), SUM (SALARY)

FROM (DonVi JOIN NhanVin ON DonVi.MaDV = NhanVien.MaDV)

GROUP BY TenDV;

Trong ví dụ này chúng ta chỉ ra một cách tường minh tên các trường, nên các thuộc tính của khung nhìn DonVi_info không lấy tên mặc định của các thuộc tính của khung nhìn select nữa.

5. 3.ALTER TABLE.(19)

Cấu trúc của bảng cơ sở dữ liệu đang tồn tại có thể được thay đổi bởi câu lệnh ALTER TABLE. Chúng ta có thể thêm một thuộc tính (cột) mới, thay đổi cấu trúc của một thuộc tính (cột) đang có, bổ sung khoá, bổ sung ràng buộc.

Cú pháp:

ALTER TABLE < Table_Name>

[ALTER COLUMN <Column_name> <New _data_type>]

ADD [<Column_name> <Data_Type>]/[constraint <constraint name><type of constraint>]

{DROP Column <column_name>}

Trong đó:

+ <Table_Name> là tên của bảng cần sửa.

+ ALTER COLUMN xác định những trường nào cần sửa.

+ <Column_name> là tên của trường cần được sửa, xóa hay thêm.

+ <New _data_type> là kiểu dữ liệu mới của trường được sửa.

+ ADD xác định trường sẽ thêm vào bảng, hoặc tạo một ràng buộc trên bảng đã có.

+ DROP Column xác định trường sẽ bị xóa khỏi bảng.

Thay đổi kiểu dữ liệu của một thuộc tính

Cú pháp:

ALTER TABLE <Tên_bảng>
ALTER (Tên_cột, Kiểu_mới)

Hoặc *ALTER TABLE <Tên_bảng>*
ALTER COLUMN Tên_cột , Kiểu_cột_mới[(size)]

Để thay đổi kiểu dữ liệu của cột DienThoai từ varchar (20) thành char (10) và có thể NULL trong bảng NhaXuatBan thực hiện như sau:

ALTER TABLE NhaXuatBan

ALTER COLUMN DienThoai char (10) NULL

Thêm một ràng buộc

Ví dụ: thêm ràng buộc check cho bảng đơn vị

ALTER TABLE DONVI

ADD CONSTRAINT check_madv

Check (madv like '[0-9] [0-9]')

Thêm một thuộc tính

Cú pháp:

ALTER TABLE <Tên_bảng>
ADD COLUMN Tên_cột , Kiểu_cột[(size)])

Ví dụ 1: thêm thuộc tính ghi chú vào bảng đơn vị

Alter table donvi

ADD (Ghichu, varchar (255))

Ví dụ 2: Thêm thuộc tính QueQuan với kiểu dữ liệu varchar (255) và ràng buộc null

```
ALTER TABLE HocSinh
ADD QueQuan varchar (255) NULL
```

Ví dụ 3: Để thêm cột FAX có kiểu char (10) và được phép NULL và bảng NhaXuatBan:

```
ALTER TABLE NhaXuatBan
ADD FAX char (10)  NULL
```

Xóa một thuộc tính

Cú pháp:

Alter table <tên bảng>

Drop <tên thuộc tính>

Ví dụ : Loại bỏ thuộc tính địa chi Email trong bảng nhà xuất bản.

```
ALTER TABLE NhaXuatBan
DROP COLUMN Email
```

Thay đổi kiểu ràng buộc của thuộc tính

Ví dụ 1: Để thay đổi kiểu dữ liệu của cột MaNXB từ char (4) thành int với thuộc tính IDENTITY (1,1):

```
ALTER TABLE  NhaXuatBan
DROP COLUMN MaNXB
GO
ALTER TABLE NhaXuatBan
ADD MaNXB       int NOT NULL    IDENTITY (1,1)
GO.
```

Ví dụ 2: Thay đổi thuộc tính NULL hay NOT NULL

Để thay đổi cột DiaChi trong bảng NhaXuatBan từ NOT NULL cho phép NULL:

```
ALTER TABLE NhaXuatBan
ALTER COLUMN DiaChi    nvarchar (255)  NULL
```

Thực hành: Để tạo và sửa đổi bảng dữ liệu cần sử dụng Query Analyzer

Để sử dụng Query Analyzer thực hiện các bước sau:

1. Khởi động Query Analyzer bằng một trong 3 cách sau:

+ Nhập isqlw tại dấu nhắc DOS: C:\>isqlw

+ Mở Enterprise Manager và chọn SQL Query Analyzer từ trình đơn Tools

+ Bấm > Programs > Microsoft SQL Server > Query Analyzer

2. Hộp thoại kết nối với SQL Server xuất hiện, trong danh sách SQL Server

chọn tên server cục bộ của máy bạn (mặc định). Phần Connect using chọn Windows authentication rồi bấm OK. Ứng dụng SQL Server Query Analyzer xuất hiện. Viết đoạn script vào vùng cửa sổ bên phải của Query Analyzer. Nhấn phím F5 hoặc nút Execute Query (hình tam giác mầu xanh) để thực thi các lệnh.

Bài tập (trang121 csdl1)

Chương 6. KHÓA VÀ RÀNG BUỘC DỮ LIỆU

Trong phần này chúng ta thảo luận về các hạn chế trên các dữ liệu trong một lược đồ cơ sở dữ liệu quan hệ. Các hạn chế đó gọi là ràng buộc dữ liệu.

6.1 Khái niệm cơ bản về ràng buộc dữ liệu

6.1.1 Định nghĩa ràng buộc

Ràng buộc: là những quy tắc được áp đặt lên trên dữ liệu đảm bảo *tính tin cậy* và *độ chính xác* của dữ liệu. Các luật toàn vẹn được thiết kế để giữ cho dữ liệu phù hợp và đúng đắn.

Ràng buộc dùng để kiểm tra khi có sự biến đổi dữ liệu như thêm vào, xóa, cập nhật từ bất kỳ các nguồn khác nhau truy cập đến CSDL.

Nếu dữ liệu thêm vào, xóa, hay cập nhật không thỏa mãn các điều kiện hoặc quy luật đã định, tùy vào nhóm phân lỗi mà SQL sinh ra ngoại lệnh nhằm thông báo cho người dùng biết. Dữ liệu khi đó sẽ không được phép cặp nhật hay thay đổi trong CSDL.

6.1.2 Các loại ràng buộc dữ liệu

Như chúng ta đã biết, thực hiện những ràng buộc dữ liệu giúp tất cả các giá trị của dữ liệu được lưu trữ đúng đắn. Tất cả dữ liệu được thêm vào cơ sở dữ liệu đều phải thỏa mãn các ràng buộc. Sau đây là một số loại ràng buộc dữ liệu: ràng buộc dữ liệu nhập vào, ràng buộc miền, ràng buộc trọn vẹn

6.2 Ràng buộc dữ liệu nhập vào

6.2.1 Ràng buộc khóa và ràng buộc trên các giá trị không xác định null

Một quan hệ được định nghĩa như một tập hợp các bộ. Theo định nghĩa, các phần tử của một tập hợp là khác nhau. Vì vậy, mọi bộ trong quan hệ khác nhau. Điều đó có nghĩa là không có hai bộ có cùng một tổ hợp giá trị cho tất cả các thuộc tính của chúng.

Thông thường, có tồn tại các tập con của các thuộc tính của một lược đồ quan hệ có tính chất là không có hai bộ nào ở trong mọi trạng thái quan hệ r của R có cùng một tổ hợp giá trị cho các thuộc tính của nó. Giả sử chúng ta ký hiệu một tập con như vậy là SK, khi đó với hai bộ khác nhau bất kỳ t_1 và t_2 trong một trạng thái quan hệ r của R chúng ta có ràng buộc là $t_1[SK] \neq t_2[SK]$. Tập hợp thuộc tính SK như vậy được gọi là một siêu khóa (superkey) của lược đồ quan hệ R. Một siêu khóa SK xác định rõ một ràng buộc về tính duy nhất. Phát biểu rằng, không có hai bộ khác nhau trong một trạng thái r của R có cùng một giá trị cho SK. Mỗi quan hệ có ít nhất một siêu khóa mặc định, đó là tập hợp tất cả các thuộc tính của nó.

Mỗi khóa K của một lược đồ quan hệ R là một siêu khóa của R với tính chất là nếu bỏ đi bất kỳ thuộc tính A nào ra khỏi K thì sẽ còn lại một tập K không phải là siêu khóa của R. Như vậy, một khóa là một siêu khóa tối thiểu, nghĩa là đó là một siêu khóa mà ta không thể vứt bỏ thuộc tính nào ra khỏi nó mà vẫn giữ được ràng buộc về tính duy nhất.

Ví dụ, xét quan hệ SINH_VIÊN với các thuộc tính Mã_số, Họ_tên, Ngày_sinh, Giới_tính, Địa_chỉ. Thuộc tính Mã_số là một khóa của SINH_VIÊN, bởi vì không có hai bộ sinh viên có cùng một giá trị cho Mã_số. Mọi tập hợp thuộc tính có chứa Mã_số, ví dụ {Mã_số, Họ_tên, Ngày_sinh} đều là một siêu khóa. Tuy nhiên, siêu khóa {Mã_số, Họ_tên, Ngày_sinh} không phải là khóa, bởi vì nếu bỏ đi thuộc tính Họ_tên hoặc Ngày_sinh, hoặc cả hai thì nó vẫn còn là một siêu khóa.

Giá trị của một thuộc tính khóa có thể được sử dụng để xác định một cách duy nhất mỗi bộ trong một quan hệ. Ví dụ, giá trị 4515202 của Mã_số xác định một cách duy nhất bộ

giá trị tương ứng với sinh viên "Lê Vân" trong quan hệ SINH_VIÊN (hình 3.1). Chú ý rằng, một tập hợp thuộc tính tạo nên một khóa là một tính chất của lược đồ quan hệ. Điều ràng buộc là tính chất đó phải thỏa mãn trên mọi trạng thái của lược đồ. Một khóa được xác định từ ý nghĩa của các thuộc tính và tính chất là bất biến; tính chất đó phải thỏa mãn khi chúng ta chèn thêm các bộ mới vào quan hệ. Ví dụ, ta không thể và không được chỉ định thuộc tính Họ_tên của quan hệ SINH_VIÊN là khóa, bởi vì không có gì đảm bảo rằng không tồn tại hai sinh viên có cùng họ tên.

Nói chung, một lược đồ quan hệ có thể có nhiều hơn một khóa. Trong trường hợp đó, mỗi một khóa được gọi làm một khóa dự tuyển. Thông thường phải chỉ định một trong các khóa dự tuyển làm khóa chính của quan hệ. Khóa chính là một khóa dự tuyển mà các giá trị của chúng được dùng để xác định các bộ trong quan hệ. Ta quy ước các thuộc tính tạo nên khóa chính của một lược đồ quan hệ được gạch dưới.

Ví dụ: SINH_VIÊN(<u>Mã_số</u>, Họ_tên, Ngày_sinh, Giới_tính, Địa_chỉ)

Chú ý rằng, khi một lược đồ quan hệ có nhiều khóa dự tuyển, việc lựa chọn một khóa dự tuyển để làm khóa chính là tùy ý. Tuy nhiên, tốt nhất là chọn khóa chính gồm một thuộc tính hoặc có số các thuộc tính là ít nhất.

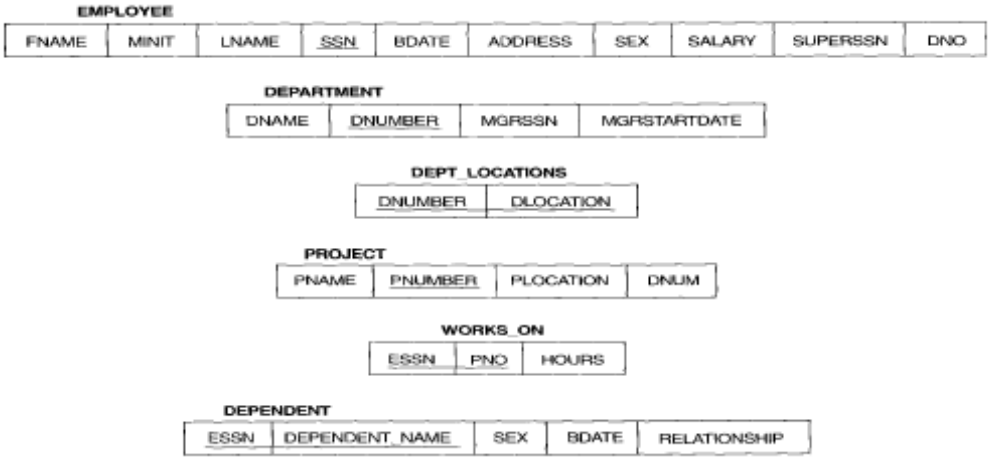
Một ràng buộc khác trên các thuộc tính chỉ rõ khi nào thì cho phép các giá trị null. Những thuộc tính luôn luôn phải có một giá trị xác định và hợp lệ thì bị ràng buộc là NOT null.

6.2.2 Ràng buộc toàn vẹn thực thể

Ràng buộc toàn vẹn thực thể được phát biểu như sau: Khóa chính phải luôn luôn có giá trị xác định, nghĩa là không được phép có giá trị null. Sở dĩ có điều đó là do giá trị của khóa chính được sử dụng để xác định các bộ riêng biệt trong một quan hệ (tức một dòng là duy nhất trong bảng). Việc có giá trị null cho khóa chính kéo theo việc chúng ta không thể xác định được một số bộ giá trị. Ví dụ nếu có hai hay nhiều hơn các bộ giá trị có giá trị null cho khóa chính thì chúng ta không thể có khả năng phân biệt chúng.

Do đó mỗi một lược đồ quan hệ R, chúng ta phải xác định khóa chính của nó. Khóa chính trong lược đồ quan hệ được gạch chân ở phía dưới của thuộc tính.

Sau đây là danh sách các lược đồ quan hệ trong cơ sở dữ liệu Company sau khi xác định ràng buộc thực thể:



Hình 6.1 Lược đồ cơ sở dữ liệu Company

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

Hình 6.2 Một thể hiện của cơ sở dữ liệu Company

Lưu ý: Ràng buộc khóa và ràng buộc thực thể được xác định cho từng quan hệ

SQL Server có một số công cụ thực hiện ràng buộc toàn vẹn thực thực thể như sau:

- PRIMARY KEY constraint
- UNIQUE constraint
- IDENTITY property

6.3 Ràng buộc miền (Domains constraints)

Định nghĩa:Ràng buộc miền là một hợp các kiểu dữ liệu và những giá trị giới hạn mà thuộc tính có thể nhận được. Thông thường việc xác định miền giá trị của các thuộc tính bao gồm một số các yêu cầu sau: ***Tên thuộc tính, Kiểu dữ liệu, Độ dài dữ liệu, khuôn dạng của dữ liệu,các giá trị giới hạn cho phép, ý nghĩa, có duy nhất hay không, có cho phép giá trị rỗng hay không.***

Các ràng buộc miền chỉ ra rằng, giá trị của mỗi thuộc tính A phải là một giá trị nguyên tử thuộc miền giá trị dom(A). Các kiểu dữ liệu liên kết các miền bao gồm: các kiểu dữ liệu số chuẩn cho các số nguyên (short integer, integer, long integer), các số thực (float, double precision float). Ngoài ra còn các kiểu dữ liệu ký tự (dãy ký tự với độ dài cố định, dãy ký tự với độ dài thay đổi), ngày, thời gian và tiền tệ. Các loại miền khác có thể là các miền con của

một kiểu dữ liệu, hoặc một kiểu dữ liệu đếm được, trong đó mọi giá trị có thể được liệt kê rõ ràng.

Như vậy ràng buộc miền liên quan đến một hay nhiều cột của bảng. Ứng với mỗi cột cụ thể có các quy luật hay tiêu chuẩn. Khi thêm hay cập nhật bản ghi mà không quan tâm đến sự liên quan đến các bản ghi trong bảng.

Công cụ thực hiện:

- DEFAULT definition
- FOREIGN KEY constraint
- CHECK constraint
- NOT NULL property
- Rules

6.4 Ràng buộc trọn vẹn

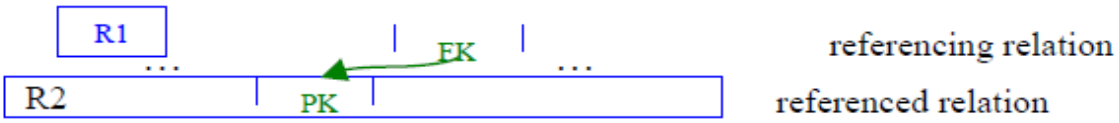
6.4.1 Ràng buộc toàn vẹn thực thể

6.4. 2 Ràng buộc toàn vẹn tham chiếu

Ràng buộc toàn vẹn tham chiếu được chỉ ra giữa hai quan hệ để duy trì sự tương ứng giữa các bộ của hai quan hệ.

Ràng buộc toàn vẹn tham chiếu được phát biểu là: Một bộ giá trị trong một quan hệ tham chiếu đến một bộ giá trị đã tồn tại trong một quan hệ khác.

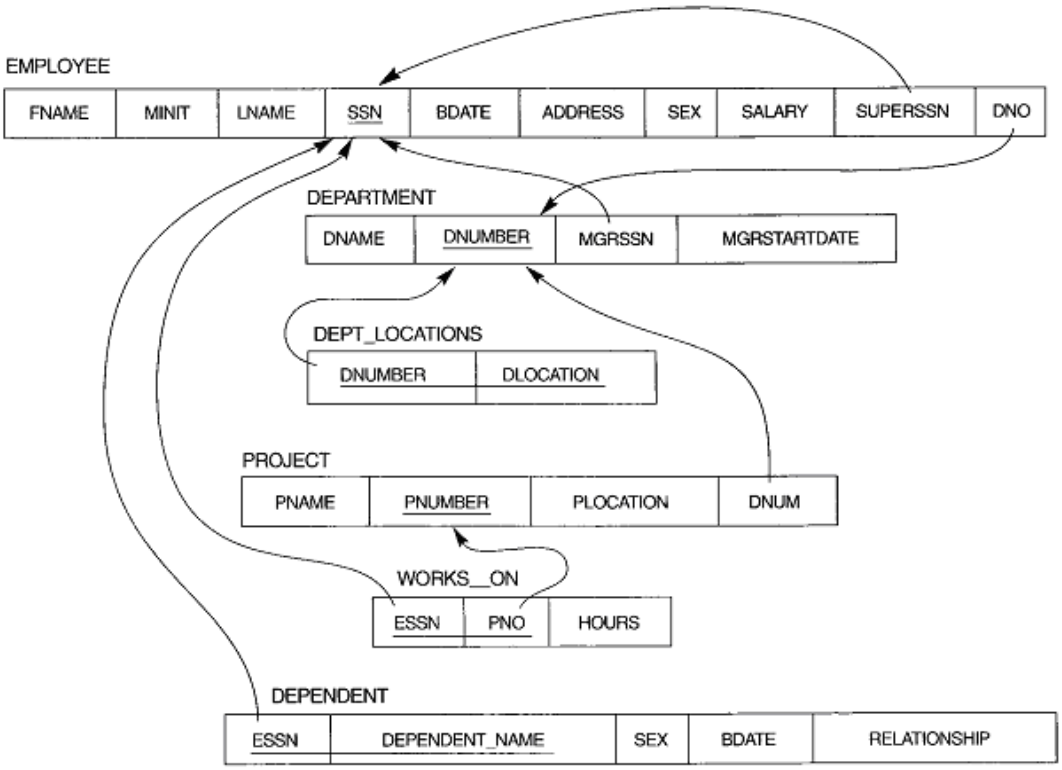
Ràng buộc toàn vẹn tham chiếu phải xác định trên 2 quan hệ: quan hệ tham chiếu (referencing relation) và quan hệ được tham chiếu (referenced relation).



Ràng buộc toàn vẹn tham chiếu còn được gọi là ràng buộc khóa ngoài.

+ **Khóa ngoài**: Một thuộc tính được gọi là khóa ngoại (Foreign key _FK) nếu nó là thuộc tính của một lược đồ quan hệ này nhưng lại là khóa chính của lược đồ quan hệ khác.

Ví dụ: Thuộc tính DNO của quan hệ EMPLOYEE tham chiếu tới thuộc tính DNumber của quan hệ DEPARTMET.



Hình 6.3 Các ràng buộc tham chiếu trong cơ sở dữ liệu Company

6.5 Giá trị mặc định và quy luật

6.5.1 Giá trị mặc định (Defaults)

Giống như các ràng buộc khác, nó giúp định nghĩa của bảng có dữ liệu phù hợp ngay cả khi không có người dùng nhập vào.

Căn cứ vào giá trị mặc định được khai báo trong cột của bảng, khi dữ liệu thêm vào, nếu bạn cung cấp giá trị ứng với cột đó rỗng, giá trị mặc định sẽ được sử dụng.

Những đặc điểm chính của ràng buộc Defaults

+ Giá trị mặc định chỉ dùng cho trường hợp thêm bản ghi mới.
+ Không quan tâm đến các hành động cập nhật hay xóa.
+ Nếu giá trị đưa vào là khác rỗng, giá trị mặc định sẽ không được sử dụng
+ Nếu giá trị đưa vào là rỗng, giá trị mặc định sẽ được sử dụng.

Giá trị mặc định được dùng để thêm dữ liệu vào cho một cột khi bạn không muốn chèn trực tiếp vào cột đó. Thiết lập giá trị mặc định cho một cột nào đó, nếu người dùng không nhập giá trị cho cột này thì nó sẽ nhận giá trị mặc định.

Có thể thiết lập giá trị mặc định cho một trường nào đó ngay khi tạo bảng bằng mệnh đề DEFAULT <value>.

Cú pháp:

CREATE TABLE Table_name

(Column_name Data_Type DEFAULT default_value)

Ví dụ 1:

CREATE TABLE employee

(employee_cd char(4),

employee_nm varchar(50),

grade char(2),

hra varchar(10) default 'N.A.')

6.5.2 Quy luật (Rules)

Các quy luật củng cố thêm tính toàn vẹn trên miền bằng cách kiểm tra chặt chẽ hơn tính hợp lệ của các giá trị. Quy luật thường được dùng để đảm bảo những giá trị như:

+ Tương hợp như một mẫu (như mệnh đề like)

+ Tương hợp với danh sách các giá trị (như mệnh đề in)

+ Thuộc về một khoảng giá trị nào đó (như mệnh đề between)

Quy luật là những đối tượng độc lập đòi hỏi bạn phải được phép tạo chúng. Bạn phải thuộc về role db_owner hoặc sysadmin. Quy luật được lưu trong bảng hệ thống sysobjects và syscomments ở mỗi cơ sở dữ liệu. Quy luật được kiểm tra khi có những tác vụ chèn và cập nhật dữ liệu (khi có liên quan đến cột được áp đặt quy luật).

Bạn có thể tạo quy luật bằng câu lệnh CREATE RULE

Cú pháp:

CREATE RULE rule_name AS condition_expression

:

SP_BINDRULE rule_name, table_name.column_name

Trong đó:

+ rule_name là tên hợp lệ và duy nhất trong cơ sở dữ liệu nó được tạo ra.

+ Condition_expression là biểu thức điều kiện có dạng @variable_name <where clause>.

<Where clause> có thể là mệnh đề where hợp lệ nào đó, bao gồm các toán tử số học between, in, like, and, or, và not…Một quy luật không thể tham chiếu đến các giá trị biến hay những cột khác trong cơ sở dữ liệu. Để thực hiện bạn phải kiểm tra ràng buộc hay trigger.

Ví dụ:

CREATE RULE check_PNR

AS @pnr BETWEEN 1 AND 500

:

SP_BINDRULE check_PNR, Reservation.PNR_no

<u>Thực hành</u>: ***Thực hiện các ràng buộc bằng T-SQL***

- Được thiết đặt trên một hoặc một tập hợp các cột của bảng.

- Nhằm thiết đặt những giới hạn cho việc nhập giá trị cho cột dữ liệu.
- Có thể được định nghĩa ngay khi tạo bảng hoặc sửa cấu trúc bảng.

+ PRIMARY KEY Constraint

Thiết đặt một hoặc tập hợp các cột làm khoá chính của bảng.

Cú pháp:

CREATE TABLE Table_name
(<Column_definition> PRIMARY KEY)

Ví dụ CREATE TABLE Reservation_copy
(PNR_no int PRIMARY KEY)

+ UNIQUE Constraint

Quy định cột này phải có giá trị khác nhau trên mỗi dòng

Cú pháp:

CREATE TABLE Table_name
(<Column_definition> UNIQUE)

Ví dụ:

CREATE TABLE passenger_copy
([PP no] VARCHAR(20) UNIQUE

+ IDENTITY Property

Quy định giá trị của một cột nào đó trong bảng là tự động
- seed_value: giá trị ban đầu
- increment_value: giá trị tăng

Cú pháp:

CREATE TABLE Table_name
(Column_name Data_Type IDENTITY
[(<seed_value>, increment_value>)])

Ví dụ:

CREATE TABLE Reservation_Copy
(ticket_no INT IDENTITY(1,1))

+ FOREIGN Key Constraint

Chỉ ra một cột làm khoá ngoại của bảng (nhằm liên kết dữ liệu trong hai bảng)

Cú pháp:

CREATE TABLE Table_name
Column_name Data_Type, ……

FOREIGN KEY (Column_name) REFERENCES Primarykey_Tablename)

Ví dụ:

CREATE TABLE Passenger

(PNR_no int, ticket_no int, name varchar(15),

……

FOREIGN KEY (PNR_no)

REFERENCES Reservation)

+ CHECK Constraint

Giới hạn dữ liệu được lưu trữ trong cột.

Cú pháp:

CREATE TABLE Table_name

(Column_name Data_Type CHECK (value1, value2, ..), ..)

Ví dụ:

CREATE TABLE Reservation

(.., class_code char(3) CHECK('EX', 'FC', 'E'), ..)

+ NOT NULL Constraint

Nếu một trường nào được quy định là NOT NULL, tức là không rỗng thì người sử dụng bắt buộc phải nhập dữ liệu cho trường này.

Cú pháp:

CREATE TABLE Table_name

(Column_name Data_Type NOT NULL, ……)

Ví dụ:

CREATE TABLE Passenger

(……, name varchar(15) NOT NULL, ……)

Chương 7.CHUẨN HÓA QUAN HỆ

Trong chương này, chúng ta sẽ thảo luận về một số vấn đề đã được phát triển nhằm mục đích chọn được lược đồ quan hệ "tốt" nghĩa là đo đạc một cách hình thức để biết vì sao tập hợp các thuộc tính này nhóm vào trong các lược đồ quan hệ thì tốt hơn nhóm kia.

7.1 Khái niệm về chuẩn hóa

7.1.1 Dư thừa dữ liệu (Redundancy)

Một mục tiêu của thiết kế lược đồ là làm tối thiểu không gian lưu trữ các quan hệ cơ sở. Các thuộc tính được nhóm vào trong các lược đồ quan hệ có một ảnh hưởng đáng kể đến không gian lưu trữ. Nếu cùng một thông tin được lưu trữ nhiều lần trong cơ sở dữ liệu thì ta gọi đó là dư thừa dữ liệu và điều này sẽ làm lãng phí không gian nhớ.

Ví dụ: RESULT(StNo, StName, SubNo,SubName, Credit, Mark)

Quan hệ RESULT(Kết quả học tập) có các thuộc tính: StNo(Mã sinh viên), StName(Tên sinh viên), SubNo(Mã môn học), SubName(Tên môn học), Credit (Số đơn vị học trình) và Mark (điểm thi của sinh viên trong môn học).

StNo	StName	SubNo	SubName	Credit	Mark
St01	Mai	Sub04	CSDL	3	9
St01	Mai	Sub01	TRR	5	10
St01	Mai	Sub02	PPS	4	8
St02	Vân	Sub04	CSDL	3	10
St02	Vân	Sub01	TRR	5	9
St03	Thanh	Sub07	Tiếng Anh	4	8

Hình 7.1 Minh họa dữ liệu của quan hệ RESULT

Ở đây có sự dữ thừa dữ liệu: Thông tin về sinh viên và môn học bị lặp lại nhiều lần. Nếu sinh viên có mã St01 thi 10 môn học thì thông tin về sinh viên này bị lặp lại 10 lần, tương

tự đối với môn học có mã Sub04, nếu có 1000 sinh viên thi thì thông tin về môn học cũng lặp lại 1000 lần.

Việc dữ thừa dữ liệu ngoài việc lãng phi không gian nhớ, nó còn dẫn đến một vấn đề nghiêm trọng hơn là sự dị thường cập nhật dữ liệu.

7.1.2 Các dị thường cập nhật dữ liệu

Dị thường cập nhật bao gồm: dị thường chèn, dị thường xóa, dị thường sửa đổi. Những dị thường cập nhật này sẽ đưa vào cơ sở dữ liệu những thông tin "lạ" và làm cho cơ sở dữ liệu mất tính đúng đắn.

+ Dị thường chèn (Insertion anomalies): Gây ra khó khăn khi chèn các bộ giá trị vào bảng hoặc dẫn đến vi phạm ràng buộc.

Ví dụ: Nếu muốn thêm thông tin một sinh viên mới nhập trường (chưa có điểm môn học nào) vào quan hệ thì không được vì khóa chính của quan hệ trên gồm 2 thuộc tính StNo và SubNo.

+ Dị thường xóa (Deletion anomalies): Gây ra việc mất thông tin xóa

Ví dụ: Giả sử xóa đi bản ghi cuối cùng thì thông tin về môn học có mã môn học là SubNo=Sub07 cũng mất.(do là người cuối cùng học môn học có mã là sub07)

+ Dị thường sửa đổi (tính không nhất quán): Gây ra việc sửa đổi hàng loạt khi ta muốn sửa đổi một giá trị trong một bộ nào đó.

Ví dụ: Giả sử ta sửa bản ghi thứ nhất, tên sinh viên được chữa thành Nga thì dữ liệu này lại không nhất quán với bản ghi thứ 2 và thứ 3 (vẫn có tên là Mai) điều này dẫn đến việc phải sửa đổi hàng loạt tất cả những sinh viên có tên là Mai thành Nga.

Nhận xét: Qua phân tích trên, ta thấy chúng ta nên tìm cách tách quan hệ trên thành các quan hệ nhỏ hơn.

7.1.3 Khái niệm về chuẩn hóa

Chuẩn hóa dữ liệu: có thể được xem như một quá trình phân tích các lược đồ quan hệ cho trước dựa trên các phụ thuộc hàm và các khóa chính của chúng để đạt đến các tính chất mong muốn:

i) Cực tiểu dư thừa và

ii) Cực tiểu các phép dị thường cập nhật dữ liệu.

7.2 Cấu trúc phụ thuộc dữ liệu

Việc quan trọng nhất khi thiết kế cơ sở dữ liệu quan hệ là ta phải chọn ra tập các lược đồ quan hệ tốt nhất dựa trên một số tiêu chí nào đó. Và để có được lựa chọn tốt, thì chúng ta cần đặc biệt quan tâm đến mối ràng buộc giữa các dữ liệu trong quan hệ, đó chính là các phụ thuộc hàm.

7.2.1 Phụ thuộc hàm (Functional Dependencies)

- Phụ thuộc hàm (FDs) được sử dụng làm thước đo để đánh giá một quan hệ tốt.
- FDs và khoá được sử dụng để định nghĩa các dạng chuẩn của quan hệ.
- FDs là những ràng buộc dữ liệu được suy ra từ ý nghĩa và các mối liên quan giữa các thuộc tính.

7.2.1.1 Định nghĩa phụ thuộc hàm

Một phụ thuộc hàm là một ràng buộc giữa hai nhóm thuộc tính của một cơ sở dữ liệu. Giả sử rằng, lược đồ cơ sở dữ liệu của ta có n thuộc tính A_1, A_2….A_n. Toàn bộ cơ sở dữ liệu được mô tả bằng một lược đồ quan hệ chung R(U), U={ A_1, A_2….A_n}. r là một trạng thái quan hệ của R.

Định nghĩa phụ thuộc hàm: Cho r(U), với r là quan hệ và U là tập thuộc tính.

Cho X, Y là tập con của U, phụ thuộc thàm X Y (đọc là X xác định Y) được định nghĩa là: với hai bộ giá trị t_1 và t_2 bất kỳ trong r

nếu có $t_1[X] = t_2[X]$ thì $t_1[Y] = t_2[Y]$ (tức là nếu hai bộ có cùng giá trị X thì có cùng giá trị Y).

Phụ thuộc hàm được viết tắt là FD. Tập các thuộc tính của X được gọi là vế trái của FD, tập các thuộc tính của Y được gọi là vế phải của D. Như vậy, X xác định Y trong lược đồ quan hệ R khi và chỉ khi nếu hai bộ của r(R) bằng nhau trên các giá trị của X thì chúng nhất thiết phải bằng nhau trên các giá trị của Y.

Phụ thuộc hàm được suy ra từ những quy tắc dữ liệu khi ta khảo sát yêu cầu của bài toán.

Chú ý: Nếu X Y thì không thể nó gì về Y X.

Ví dụ: Từ mã số bảo hiểm xã hội, ta có thể suy ra được tên của nhân viên. Từ mã dự án, ta có thể suy ra tên và vị trí của dự án.

Ví dụ: Lược đồ quan hệ

Muon (sothe,Masach, Tennguoimuon,tensach,ngaymuon)

Với các phụ thuộc hàm:

Sothe Tennguoimuon

Masach tensach

Sothe,masach ngaymuon

Ví dụ: Cho quan hệ DAY

Giaovien	Monhoc	Tailieu
AA	Mon 1	XX
AA	Mon 2	YY
BB	Mon 3	ZZ
CC	Mon 4	TT

Nhìn vào bảng ta có thể nói có một phụ thuộc hàm Tailieu →Monhoc. Tuy nhiên, chúng ta không thể khẳng định được vì điều đó chỉ đúng với trạng thái quan hệ này, biết đâu trong trạng thái quan hệ khác có thể có hai môn học có cùng tài liệu tham khảo. Trong ví dụ này không có phụ thuộc hàm giữa Giaovien và Monhoc vì AA dạy hai môn học: mon 1 và mon 2.

7.2.1.2 Hệ tiên đề Amstrong

Cho lược đồ quan hệ R(U), U là tập thuộc tính, F là tập các phụ thuộc hàm được xác định trên lược đồ quan hệ R(U).

X, Y là hai tập con của U. Một phụ thuộc hàm X →Y được gọi là suy diễn logic từ F nếu X → Y đúng trong mỗi trạng thái quan hệ r là mở rộng hợp pháp của R. Nghĩa là ta có phụ thuộc hàm X →Y được suy diễn logic từ F nếu r trên U thỏa các phụ thuộc hàm trong F thì cũng thỏa phụ thuộc hàm X →Y.

Ví dụ: Tập phụ thuộc hàm: F = { A → B, B → C}

Ta có phụ thuộc hàm A → C là phụ thuộc hàm được suy từ F.

Hệ tiên đề Armstrong được sử dụng để tìm ra các phụ thuộc hàm suy diễn từ F.

Hệ tiên đề Armstrong bao gồm 6 quy tắc;

1. Phản xạ: Nếu X Y thì X → Y

2. Tăng trưởng: Nếu Z U và X → Y thì XZ → YZ (Ký hiệuXZ là X Z)

3. Bắc cầu: Nếu X → Y và Y → Z thì X → Z

4. Giả bắc cầu:(tựa bắc cầu) Nếu X → Y và WY → Z thì XW → Z

5. Luật hợp: Nếu X → Y và X → Z thì X →YZ

6. Luật phân rã: Nếu X → Y và Z Y thì X → Z (Quy tắc chiếu X →YZ thì X → Y và X→Z.

Trong sáu luật trên thì luật 4, 5, 6 được suy từ luật 1, 2, 3.

7.2.1.3 Bao đóng của tập phụ thuộc hàm

- Ta gọi f là một phụ thuộc hàm được suy diễn từ F, ký hiệu là F⊨ f nếu tồn tại một chuỗi phụ thuộc hàm: $f_1, f_2, \ldots, f_n$ sao cho f_n=f và mỗi f_i là một thành viên của F hay được suy dẫn từ những phụ thuộc hàm f_j (j=1,…,i-1) trước đó nhờ vào luật suy diễn .

- **Bao đóng của F:** ký hiệu là F^+ là tập tất cả các phụ thuộc hàm được suy diễn từ F nhờ vào hệ tiên đề Armstrong. F^+ được định nghĩa:

$$F^+ = \{ X \rightarrow Y \mid F \models X \rightarrow Y \}$$

Ví dụ: F = {X →Y; Y→ T}

F^+ = {F {X→T, X →YT}

7.2.1.4 Bao đóng của tập thuộc tính X dưới một tập phụ thuộc hàm F

X là tập thuộc tính xuất hiện ở về trái của một phụ thuộc hàm nào đấy trong F.

Bao đóng của tập thuộc tính X xác định trên tập phụ thuộc hàm F ký hiệu là X^+ : là tập hợp tất cả các thuộc tính có thể suy ra từ X. Ký hiệu:

$$X^+ = \{ Y \mid F \models X \rightarrow Y \} = \{Y \mid X \quad Y \quad F^+\}$$

X^+ có thể được tính toán thông qua việc lặp đi lặp lại các quy tắc 1, 2, 3 của hệ tiên đề Armstrong.

Bổ đề: X →Y được suy diễn từ tập phụ thuộc hàm F bằng các quy tắc suy diễn Amstrong khi và chỉ khi Y X^+

Thật vậy, giả sử X→Y được suy diễn từ tập phụ thuộc hàm F bằng các quy tắc suy diễn Amstrong và $Y = A_1A_2\ldots A_n$ với $A_1, A_2,\ldots,A_n$ là các thuộc tính. Như vậy, theo quy tắc chiếu (quy tắc phân rã) ta có $X \rightarrow A_1$, $X \rightarrow A_2 \ldots X \rightarrow A_n$. Theo định nghĩa của X^+, thì A_i X^+ với i= 1,2…,n. Như vậy Y X^+.

Ngược lại, giả sử Y X^+ và $Y= A_1A_2\ldots A_n$. Theo định nghĩa X^+ ta có

$X \rightarrow A_i$ với i = 1, 2…,n. Theo quy tắc hợp ta có X →Y.

Thuật toán xác định bao đóng của tập thuộc tính X (Xác định X^+)

```
X+ := X;
Repeat.
        Old X+ := X+;
        for (mỗi phụ thuộc hàm Y →Z trong F) do
        if Y  X+ then X+ = X+  Z;
until  ( X+ = Old X+ );
```

Ví dụ 1: Cho tập phụ thuộc hàm:

F={MabaohiemXH→TenNhanvien,MaDA→{TenDA,VitriDA},

MabaohiemXH, MaDA} → Sogio}

Suy ra:

{MabaohiemXH}$^+$ = {MabaohiemXH, Tennhanvien}

{MaDA}$^+$ = {MaDA,TenDA, VitriDA}

{MabaohiemXH, MaDA}$^+$ = {MabaohiemXH, MaDA, TenNhanvien, TenDA, VitriDA, Sogio}

Như vậy, tập thuộc tính {SSN, PNUMBER} là khoá của quan hệ.

7.2.1.5 Khoá của quan hệ

Cho quan hệ R(U), tập K U được gọi là khóa của quan hệ R nếu: K^+=U và nếu bớt một phần tử khỏi K thì bao đóng của nó sẽ khác U.

Như thế tập K U là khoá của quan hệ nếu K^+=U và (K \A)$^+$ ≠U , A U.

Ví dụ: Xét lược đồ quan hệ R(A, B, C, D, E, F) và tập phụ thuộc hàm

F = {A,B→F; A→C,D; B→E}

Ta có $\{A,B\}^+ = (A, B, C, D, E, F)$, $A^+ = \{A, C, D\}$, $B^+ = \{B, E\}$ Vậy K={A, B} là khoá của quan hệ.

Ví dụ: Cho R = { A, B, C, D, E, G } và tập phụ thuộc hàm:

F= { AB → C , D → EG , BE → C , BC → D , CG → BD, ACD → B, CE → AG}

Ta sẽ thấy các tập thuộc tính:

K_1 = { A, B } , K_2 = {B,E} , K_3={C,G} , K_4={C,E} , K_5 = {C,D}, K_6={B,C}

đều là khóa của quan hệ.

Như vậy, một quan hệ có thể có nhiều khóa.

Thuật toán tìm khoá: Tìm một khóa K của R(U) dựa trên tập F các phụ thuộc hàm.

Ý tưởng: Bắt đầu từ tập U vì Closure(U^+,F) = U. Sau đó ta bớt dần các phần tử của U để nhận được tập bé nhất mà bao đóng của nó vẫn bằng U.

Thuật toán:

Input: Lược đồ quan hệ R(U), tập phụ thuộc hàm F.

Output: Khoá K

Bước 1**:** Gán K = U

Buớc 2**:** Lặp lại các bước sau với mỗi thuộc tính A trong K.

- Tính $(K-A)^+$ đối với F
- Nếu $(K-A)^+$ =U thì loại A ra khỏi K tức K:= K-A

Nhận xét:

- Thuật toán trên chỉ tìm được một khóa. Nếu cần tìm nhiều khóa, ta thay đổi trật tự loại bỏ các phần tử của K.
- Chúng ta có thể cải thiện tốc độ thực hiện thuật toán trên bằng cách: Trong bước 1 ta chỉ gán K=Left (là tập các phần tử có bên tay trái của các phụ thuộc hàm)

Ví dụ:

Cho lược đồ quan hệ R = { A,B,C,D,E,G,H,I} và tập phụ thuộc hàm:

F= { AC → B, BI → ACD, ABC → D , H → I , ACE → BCG , CG → AE }

Tìm khoá K?

Ta có Left={A,B,C,H,E,G}

Bước 1: K=Left={A,B,C,H,E,G}

Bước 2:

Tập thuộc tính	A	B	C	D	E	G	H	I	Ghi chú
ABCHEG	x	x	x	x	x	x	x	x	
BCHEG	x	x	x	x	x	x	x	x	Loại A

CHEG x x x x x x x x Loại B

CHG x x x x x x x x Loại E

Như vậy, {C,H,G} là một khoá của R.

Nếu muốn tìm tất cả các khoá của R, ta cần thay đổi trật tự loại bỏ phần tử của khoá K.

7.2.1.6 Tập phụ thuộc hàm tương đương

Hai tập phụ thuộc hàm **F và G là tương đương** nếu:

- Tất cả các phụ thuộc hàm trong F có thể được suy ra từ G, và
- Tất cả các phụ thuộc hàm trong G có thể suy ra từ F.

Vì thế, F và G là tương đương nếu $F^+ = G^+$

Nếu F và G là tương đương thì ta nói **F phủ G hay G phủ F**.

Vì thế, thuật toán sau đây sẽ kiểm tra sự tương đương của hai tập phụ thuộc hàm:

- F phủ E: X Y E, tính X^+ từ F, sau đó kiểm tra xem Y X^+
- E phủ F: X Y F, tính X^+ từ E, sau đó kiểm tra xem Y X^+

Ví dụ: Xét hai tập phụ thuộc hàm

F = {A→C, AC→D, E→AD, E→H}

E = {A→CD, E→AH}

Chứng minh hai tập phụ thuộc hàm trên là tương đương

Chứng minh F phủ E

Tìm bao đóng của các vế trái của các phụ thuộc hàm trong E theo F

$\{A\}^+ = \{A, C, D\}$

$\{E\}^+ = \{E, A, D, H\}$

Ta thấy các bao đóng này chứa các vế phải tương ứng. từ đó suy ra F phủ E

Chứng minh E phủ F

Tìm các bao đóng của vế trái của các phụ thuộc hàm trong F theo E. Ta có

$\{A\}^+ = \{A, C, D\}$

$\{AC\}^+ = \{A, C, D\}$

$\{E\}^+ = \{E, A, H\}$

Ta thấy các bao đóng này chứa các vế phải tương ứng. Từ đó suy ra E phủ F

Như vậy E tương đương với F.

7.2.1.7 Tập phụ thuộc hàm tối thiểu

Tập phụ thuộc hàm là tối thiểu nếu nó thoả mãn các điều kiện sau:

1. Chỉ có một thuộc tính nằm ở phía bên phải của tất cả các phụ thuộc hàm trong F.

2. Không thể bỏ đi bất kỳ một phụ thuộc hàm nào trong F mà vẫn có được một tập phụ thuộc hàm tương đương với F (tức là, không có phụ thuộc hàm dư thừa).

3. Không thể thay thế bất kỳ phụ thuộc hàm X→A nào trong F bằng phụ thuộc hàm Y→A, với Y X mà vẫn có được một tập phụ thuộc hàm tương đương với F (tức là, không có thuộc tính dư thừa trong phụ thuộc hàm)

Nhận xét:

- Tất cả các tập phụ thuộc hàm đều có phụ thuộc hàm tối thiểu tương đương với nó.
- Có thể có nhiều phụ thuộc hàm tối thiểu

Thuật toán: Tìm tập phụ thuộc hàm tối thiểu G của F

1. Đặt G:=F.
2. Thay thế tất cả các phụ thuộc hàm X→{A_1,A_2,…,A_n} trong G bằng n phụ thuộc hàm: X →A_1, X →A_2,…, X →A_n.
3. Với mỗi phụ thuộc hàm X → A trong G, với mỗi thuộc tính B trong X nếu ((G -{X → A}) {(X - {B}) →A}) là tương đương với G, thì thay thế X→ A bằng (X - {B}) → A trong G. *(Loại bỏ thuộc tính dư thừa trong phụ thuộc hàm)*
4. Với mỗi phụ thuộc hàm X → A trong G, nếu (G- {X → A}) tương đương với G, thì loại bỏ phụ thuộc hàm X → A ra khỏi G.*(Loại bỏ phụ thuộc hàm dư thừa)*

Ví dụ: Tìm tập phụ thuộc hàm tối thiểu G của F (hay tìm phủ tối thiểu G của F)

F = {A→BC, B→AC, C→AB}

Bước 1: G = F = {A→BC, B→AC, C→AB}

Bước 1. G = {A→B, A→C, B→A, B→C, C→A, C→B}

Bước 3. Do các phụ thuộc hàm trong G đều có vế trái gồm một thuộc tính nên G vẫn giữ nguyên.

Bước 4. Loại bỏ phụ thuộc hàm thừa.

1. Do A→B và B→C nên A→C là thừa. Do C→B, B→A nên C→ A là thừa. Bỏ đi những phụ thuộc hàm thừa ta được một tập phụ thuộc hàm tối thiểu

G = {A→B, B→A, B→C, C→B}

2. Do A→B và B→C nên A→C là thừa. Do B→C và C→A nên B→A là thừa. do C→A, A→B nên C→B là thừa. Bỏ đi những phụ thuộc hàm thừa ta nhận được tập phụ thuộc hàm tối thiểu: G = { A→B, B→C, C→A}

7.2.2 Phụ thuộc đa trị

Trong phần này chúng ta thảo luận khái niệm phụ thuộc hàm đa trị . Các phụ thuộc hàm đa trị hệ quả của dạng chuẩn 1 không cho phép một thuộc tính của một bộ có một tập giá trị (nghĩa là các thuộc tính đa trị). Nếu chúng có hai hoặc nhiều hơn các thuộc tính độc lập và đa trị trong cùng một lược đồ quan hệ thì phải lặp lại mỗi một giá trị của một trong các

thuộc tính với mỗi giá trị của thuộc tính khác, để giữ cho trạng thái quan hệ nhất quán và duy trì tính độc lập giữa các thuộc tính. Ràng buộc đó được chỉ ra bằng một phụ thuộc hàm đa trị.

7.2.2.1Định nghĩa phụ thuộc hàm đa trị

Giả thiết có một lược đồ quan hệ R(U), X và Y là hai tập con của U. Một phụ thuộc hàm đa trị ký hiệu là X →→Y chỉ ra ràng buộc sau đây trên một trạng thái quan hệ bất kỳ r của R: Nếu hai bộ t_1 và t_2 tồn tại trong R sao cho $t_1[X] = t_2[X]$ thì hai bộ t_3 và t_4 cũng tồn tại trong R với các tính chất sau:

$t_3[X] = t_4[X] = t_1[X] = t_2[X]$

$t_3[Y] = t_1[Y]$ và $t_4[Y] = t_2[Y]$

$t_3[Z] = t_2[Z]$ và $t_4[Z] = t_1[Z]$ với Z = (R – (X Y))

Khi X→→Y thỏa mãn ta nói rằng X đã xác định Y. Bởi do tính đối xứng trong định nghĩa khi X →→Y thỏa mãn trong R thì X →→Z cũng thỏa mãn trong R. Như vậy X →→Y kéo theo X →→Z vì thế đôi khi nó được viết là : X →→ Y□Z.

Định nghĩa hình thức chỉ ra rằng, cho trước một bộ giá trị của X, tập hợp các giá trị của Y được xác định bởi giá trị này của X là được xác định hoàn toàn bởi một mình X và không phụ thuộc vào các giá trị của các thuộc tính còn lại Z trong R. Như vậy, mỗi khi hai bộ tồn tại có các giá trị khác nhau của Y, nhưng cùng một giá trị X thì các giá trị này của Y phải được lặp lại trong các bộ riêng rẽ với mỗi giá trị khác nhau của Z có mặt với cùng giá trị của X. Điều đó tương ứng một cách không hình thức với Y là một thuộc tính đa trị của các thực thể được biểu diễn bằng các bộ trong R.

Ví dụ: Cho bảng nhân viên

TenNV	TenDA	TenconNV
Nam	DA01	Lan
Nam	DA02	Hoa
Nam	DA01	Hoa
Nam	DA02	Lan

Trong bảng trên có hai phụ thuộc đa trị là TenNV →→TenDA

TenNV→→TenconNV

Một phụ thuộc hàm đa trị X→→Y được gọi là phụ thuộc hàm đa trị tầm thường nếu:

+ Y là tập con của X;

+ Hoặc X Y = R

Một phụ thuộc hàm đa trị không thỏa mãn một trong hai điều kiện trên gọi là một phụ thuộc hàm đa trị không tầm thường.

7.2.2.2Các quy tắc suy diễn đối với các phụ thuộc hàm và phụ thuộc đa trị.

Các quy tắc từ 1 đến 8 sau đây tạo nên một tập hợp đúng đắn và đầy đủ cho việc suy diễn các phụ thuộc hàm và phụ thuộc đa trị từ một tập các phụ thuộc hàm cho trước. Giả thiết rằng, tất cả các thuộc tính được chứa trong một lược đồ quan hệ R $\{A_1, A_2,...,A_n\}$ và X, Y, Z là tập con của R (FD là phụ thuộc hàm, MVM ký hiệu phụ thuộc đa trị).

Quy tắc 1 (quy tắc phản xạ cho FD): Nếu X Y thì X → Y

Quy tắc 2 (quy tắc tăng cho FD):

Nếu Z U và X → Y thì XZ → YZ (Ký hiệuXZ là X Z)

Quy tắc 3(Quy tắc bắc cầu cho FD): Nếu X → Y và Y → Z thì X → Z

Quy tắc 4 (quy tắc bù cho MVD):

{X→→Y}|={X→→(R – (*X* *Y*))}

Quy tắc 5 (quy tắc tăng cho MVD)

Nếu X→→Y và W Z thì WX →→YZ

Quy tắc 6 (quy tắc bắc cầu cho MVD):

{X→→Y, Y→→Z} |= X→→(Z-Y)

Quy tắc 7 (quy tắc tái tạo cho FD và MVD):

{X→Y}|=X→Y

Quy tắc 8 (quy tắc liên hợp cho FD và MVD):

Nếu X→Y và có tồn tại W với các tính chất:

a) W ∩Y=∅,

b)W →Z và

c) *Y* *Z* *Y* *Z*

thì X →Z.

Từ quy tắc 1 đến quy tắc 3 là các suy diễn Amstrong đối với phụ thuộc hàm. Quy tắc 4 đến 6 là các quy tắc suy diễn chỉ liên quan đến các phụ thuộc đa trị. Quy tắc 8 liên kết các phụ thuộc hàm và các phụ thuộc đa trị. Đặc biệt quy tắc 7 nói rằng, một phụ thuộc hàm là một trường hợp đặc biệt của một phụ thuộc hàm đa trị. Điều đó có nghĩa là mỗi phụ thuộc hàm cũng là một phụ thuộc hàm đa trị, bởi vì nó thỏa mãn định nghĩa hình thức của phụ thuộc hàm đa trị. Về cơ bản, một phụ thuộc hàm X→Y là một phụ thuộc hàm đa trị X→→Y với một hạn chế phụ là: có nhiều nhất một giá trị của Y được kết hợp với mỗi giá trị của X. Cho trước một tập hợp các phụ thuộc hàm và phụ thuộc đa trị chỉ ra trên R $\{A_1, A_2,...,A_n\}$, ta có thể sử dụng các quy tắc đã cho để suy ra tập hợp đầy đủ các phụ thuộc (hàm và đa trị) F^+ đúng trong mọi trạng thái quan hệ r của R thỏa mãn F. (F^+ là bao đóng của F).

7.3. Chuẩn hóa lược đồ quan hệ

Quá trình chuẩn hóa (do Codd đề nghị năm 1972) là xét một lược đồ quan hệ và thực hiện một loạt các kiểm tra để xác định nó có thỏa mãn một dạng chuẩn nào đó hay không. Quá trình này được thực hiện bằng việc đánh giá mỗi quan hệ với tiêu chuẩn của các dạng chuẩn và tách các quan hệ nếu cần. Quá trình này có thể xem như là việc thiết kế quan hệ bằng việc phân tích. Lúc đầu, Codd đề nghị ba dạng chuẩn gọi là dạng chuẩn 1, dạng chuẩn 2, và dạng chuẩn 3. Một định nghĩa mạnh hơn của dạng chuẩn 3 gọi là dạng chuẩn Boyce Codd Do Boyce và Codd đề nghị muộn hơn. Tất cả các dạng chuẩn này dựa trên các phụ thuộc hàm giữa các thuộc tính của một quan hệ. Sau đó là dạng chuẩn 4 (4NF) và dạng chuẩn 5 (5NF) được đề nghị dựa trên các phụ thuộc hàm đa trị và phụ thuộc hàm nối.

Các lược đồ quan hệ không thỏa mãn các kiểm tra dạng chuẩn sẽ được tách ra thành các lược đồ quan hệ nhỏ hơn thỏa mãn các kiểm tra và có các tính chất mong muốn.Như vậy, thủ tục chuẩn hóa cung cấp cho những người thiết kế cơ sở dữ liệu:

a). Một cơ cấu hình thức để phân tích các lược đồ quan hệ dựa trên các khóa của nó và các phụ thuộc hàm giữa các thuộc tính của nó.

b) Một loạt các kiểm tra dạng chuẩn có thể thực hiện trên các lược đồ quan hệ riêng rẽ, sao cho cơ sở dữ liệu quan hệ có thể được chuẩn hóa đến một mức cần thiết.

Dạng chuẩn của một quan hệ liên quan đến điều kiện dạng chuẩn cao nhất mà nó thỏa mãn. Các dạng chuẩn khi được xem xét độc lập với các sự kiện khác không đảm bảo một thiết kế cơ sở dữ liệu tốt. Nói chung, việc xác minh riêng biệt từng lược đồ quan hệ ở dạng chuẩn này hay dạng chuẩn khác là chưa đủ. Tốt hơn là quá trình chuẩn hóa thông qua phép tách phải khẳng định một vài tính chất hỗ trợ mà tất cả các lược đồ quan hệ phải có. Chúng gồm hai tính chất sau:

- Tính chất nối không mất mát (hoặc nối không phụ thêm), nó đảm bảo rằng vấn đề tạo ra các bộ giả không xuất hiện đối với các lược đồ quan hệ được tạo ra sau tách.
- Tính chất bảo toàn sự phụ thuộc: nó đảm bảo rằng từng phụ thuộc hàm sẽ được biểu hiện trong các quan hệ riêng rẽ nhận được sau khi tách.

Trước khi định nghĩa các dạng chuẩn cần xem xét lại định nghĩa khóa của quan hệ.

7.3.1 Dạng chuẩn 1 (1NF – First Normal Form)

Định nghĩa: Một quan hệ ở dạng chuẩn 1 nếu các giá trị của tất cả thuộc tính trong quan hệ là **nguyên tử (**đơn, không phân chia được -tức là chỉ có 1 giá trị tại một thời điểm**).**

Như vậy 1NF không cho phép quan hệ có các thuộc tính đa trị hoặc có các nhóm thuộc tính lặp (quan hệ trong quan hệ).

Ví dụ: xét quan hệ Donvi và Nhanvienduan như bảng sau:

Donvi	MaDV	TenDV	MaNQL	DiaDiem

5	Nghiên cứu	NV002	Nam Định, Hà Nội, Bắc Ninh
4	Hành chính	NV014	Hà Nội
1	Lãnh đạo	NV061	Hà Nội

Nhanviendua n	MaDA	TenDA	TenNV	Sogio
	1	DA01	Van Nam	15 20
	2	DA02	Nam Thanh Bằng	10 12 28
	3	DA03	Thanh	20

Các quan hệ này không thỏa mãn điều kiện 1NF. Quan hệ Donvi chứa thuộc tính đa trị Diadiem, quan hệ Nhanvienduan chứa nhóm các thuộc tính lặp TenNV và Sogio.

Để đạt đến dạng chuẩn 1NF đối với các quan hệ trên, ta dùng phương pháp sau:

Loại bỏ các thuộc tính vi phạm dạng chuẩn 1 và đặt chúng vào một bảng riêng cùng với khóa chính của quan hệ ban đầu. Khóa chính của bảng này là một tổ hợp khóa chính của quan hệ ban đầu và thuộc tính đa trị hoặc khóa bộ phận của nhóm lặp.

Các thuộc tính còn lại lập thành một quan hệ với khóa chính là khóa chính ban đầu.

Áp dụng: Lược đồ quan hệ Donvi ở trên sẽ được tách thành hai:

Donvi(MaDV, TenDV,MaNQL)

DonviDiaDiem(MaDV,Diadiem)

Lược đồ quan hệ Nhanvienduan cũng được tách thành hai:

DuAn(MaDA,TenDA)

Nhanvienduan(MaDA,Tennhanvien,sogio)

7.3.2 Dạng chuẩn 2 (2NF – Second Normal Form)

Dạng chuẩn 2NF dựa trên khái niệm phụ thuộc hàm đầy đủ. Một phụ thuộc hàm X→Y là một phụ thuộc hàm đầy đủ nếu loại bỏ bất kỳ thuộc tính A nào ra khỏi X thì phụ thuộc hàm không còn đúng nữa. Điều đó có nghĩa là: Với thuộc tính A bất kỳ, A X, (X-{A} Y.

Một phụ thuộc hàm X→Y là phụ thuộc hàm bộ phận nếu có thể bỏ một thuộc tính A X ra khỏi X mà phụ thuộc hàm này vẫn đúng có nghĩa là với

A X, (X-{A}) Y

Ví dụ: xét lược đồ quan hệ

Nhanvienduan(MaNV,MaDA,sogio,TenNV,TenDA,DiadiemDA)

Khi đó:

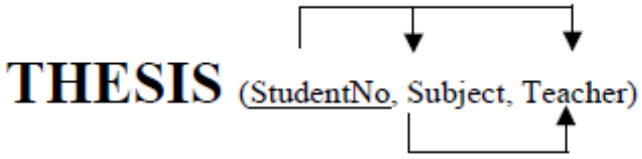
MaNV,MaDA →sogio là phụ thuộc hàm đầy đủ.

MaNV,MaDA→TenNV là phụ thuộc hàm bộ phận bởi vì có phụ thuộc hàm MaNV→TenNV.

Định nghĩa: Một quan hệ ở dạng chuẩn 2 nếu:

- Quan hệ đó ở dạng chuẩn 1
- Tất cả các thuộc tính không phải là khóa **phụ thuộc đầy đủ** vào khóa.

Ví dụ: Quan hệ sau đây **ở dạng 2NF:**



StudentNo	Subject	Teacher
SV01	1	Nguyễn Văn Hiệu
SV02	2	Ngô Lan Phương
SV03	1	Nguyễn Văn Hiệu
SV04	1	Nguyễn Văn Hiệu

Nếu một quan hệ không thỏa mãn điều kiện 2NF ta có thể chuẩn hóa nó để có các quan hệ 2NF như sau: Loại bỏ các thuộc tính không khóa phụ thuộc vào một bộ phận khóa chính và tách thành một bảng riêng, khóa chính của bảng này là bộ phận khóa mà chúng phụ thuộc vào. Các thuộc tính còn lại lập thành một quan hệ, khóa chính của nó là khóa chính ban đầu.

Ví dụ: Xét lược đồ quan hệ

Nhanvienduan(MaNV,MaDA,sogio,TenNV,TenDA,DiadiemDA)

Với các phụ thuộc hàm:

MaNV,MaDA→sogio

MaNV→TenNV

MaDA→TenDA,DiadiemDA.

Ta nhận thấy ở đây có những thuộc tính không khóa phụ thuộc vào bộ phận của khóa chính . Như vậy, nó không thỏa mãn điều kiện 2NF.

Áp dụng phương pháp chuẩn hóa trên, lược đồ quan hệ được tách thành các lược đồ sau:

N_D1 (MaDA,TenDA,DiadiemDA);

N_D2 (MaNV,TenNV);

N_D3 (MaNV,MaDA,sogio).

7.3.3 Dạng chuẩn 3 (3NF- Third Normal Form)

Dạng chuẩn 3NF dựa trên khái niệm phụ thuộc bắc cầu. Một phụ thuộc hàm X→Y trong một lược đồ quan hệ R là một phụ thuộc hàm bắc cầu nếu có một tập hợp thuộc tính Z không phải là một khóa dự tuyển, cũng không phải là tập con của một khóa nào, và cả X→Z, Z→Y đều đúng.

Định nghĩa dạng chuẩn 3NF: Một quan hệ ở dạng chuẩn 3NF nếu:

- Quan hệ ở dạng chuẩn 2
- Và không có chứa các phụ thuộc hàm **phụ thuộc bắc cầu vào khoá** (không có thuộc tính không khóa nào của R phụ thuộc bắc cầu vào khóa chính)

Nếu một lược đồ quan hệ không thỏa mãn điều kiện 3NF ta có thể chuẩn hóa nó để nó có được các lược đồ 3NF như sau: Loại bỏ các thuộc tính phụ thuộc bắc cầu ra khỏi quan hệ và tách chúng thành một quan hệ riêng có khóa chính là thuộc tính bắc cầu. Các thuộc tính còn lại lập thành một quan hệ có khóa chính là của quan hệ ban đầu.

Ví dụ: xét lược đồ quan hệ:

Nhanviendonvi(TenNV,MaNV,Ngaysinh,Diachi,MaDV,TenDV,MaNQL)

Với các phụ thuộc hàm:

MaVN→TẹnNV,Ngaysinh,Diachi,MaDV,TenDV,MaNQL;

MaDV→TenDV,MaNQL

Các thuộc tính TenDV,MaNQL phụ thuộc bắc cầu vào khóa chính, do đó lược đồ quan hệ trên không thỏa mãn điều kiện 3NF.

Áp dụng phương pháp chuẩn hóa ở trên, lược đồ quan hệ tách thành như sau:

NV_DV1 (TenNV, MaNV, Ngaysinh, Diachi, MaDV)

NV_DV2(MaDV,TenDV, MaNQL)

7.3.4 Dạng chuẩn BCNF (Boyce-Codd Normal Form)

Định nghĩa: Một lược đồ quan hệ R được gọi là có dạng chuẩn BCNF nếu nó ở dạng chuẩn 3NF và không có các thuộc tính khóa phụ thuộc hàm vào thuộc tính không khóa.

Hay nói cách khác: Quan hệ R ở dạng chuẩn BCNF khi tất cả các phụ thuộc hàm X→A trong R đều phải có X là khóa của R.

Ví dụ: lược đồ R(A, B, C, D, E) với các phụ thuộc hàm

A, B→C, D, E ;

D → B.

Quan hệ này vi phạm dạng chuẩn BCNF vì thuộc tính khóa B phụ thuộc hàm vào thuộc tính không khóa D.

Nếu một lược đồ quan hệ không thỏa mãn điều kiện BCNF, ta có thể chuẩn hóa nó để nó có được các lược đồ BCNF như sau: Loại bỏ các thuộc tính khóa phụ thuộc hàm vào thuộc

tính không khóa ra khỏi quan hệ và tách chúng thành một quan hệ riêng có khóa chính là thuộc tính không khóa gây ra phụ thuộc.

Áp dụng phương pháp chuẩn hóa ở trên, lược đồ quan hệ tách ra như sau:

R_1(D, B)

R_2(A, D, C, E).

Ví dụ: Cho lược đồ quan hệ R = (A, B, C, D, E, F, G, H, I, J) có khóa chính là A, B.

Với tập các phụ thuộc hàm như sau;

A, B → C, D, E, F, G, H, I, J

A→E, F, G, H, I, J

F→I, J

D→B.

Do có phụ thuộc hàm A→E, F, G, H, I, J mà A là bộ phận của khóa chính nên quan hệ R là vi phạm chuẩn 2NF. Ta tách R ra:

R_1(A, E, F, G, H, I, J)

R_2(A, B, C, D)

Trong R_1 do có phụ thuộc hàm F→I, J nên ta có I, J phụ thuộc bắc cầu vào khóa chính, R_1 vi phạm chuẩn 3NF. Trong đó R_2 ta có phụ thuộc hàm D→B trong đó B là một thuộc tính khóa, R_2 vi phạm chuẩn BCNF. Tách R_1 và R_2 ta có;

R_{11}(F,I, J); R_{12}(**A**, E, F, G, H); R_{21}(**D**,B); R_{22}(**A**, **D**, C).

Chương 8. BẢNG ẢO

Trong chương này chúng ta sẽ tìm hiểu về một cấu trúc dữ liệu phụ đó là view. View tồn tại riêng biệt với dữ liệu nhưng có liên quan chặt chẽ với dữ liệu đó. Một view được dùng để lọc hoặc xử lý dữ liệu trước khi người dùng truy cập nó.

8.1 Khái niệm về view

Trong phần này chúng ta sẽ tìm hiểu xem view là gì? Các kiểu view, thuận lợi của việc sử dụng view, và các hạn chế SQL Server trong sử dụng view.

8.1.1 View

Truy vấn thường được sử dụng để xuất dữ liệu ra từ bảng. Nó được thực hiện trên dữ liệu (actual data) của bảng. Thay vì việc truy vấn và thực hiện trực tiếp trên dữ liệu thực (ví dụ như Queries trong Microsoft Access), SQL Server đã hỗ trợ một khái niệm mới, đó là View. View là một bảng tương tự như bảng chứa dữ liệu thực, nhưng nó chỉ là bảng logic (không phải là bảng vật lý), có nghĩa là nó không có vị trí lưu trữ vật lý của dữ liệu. Vì thế, View thường được gọi như một bảng ảo (Vitual table).

View là một cách khác để nhìn vào dữ liệu, và nó có thể nhìn thấy dữ liệu từ một hay nhiều bảng. Tuy nhiên, View không tồn tại như một tập dữ liệu được lưu trữ trên đĩa, mà nó chỉ là một tham chiếu tới những bảng dữ liệu vật lý.

View hoạt động như một bộ lọc dữ liệu từ cơ sở dữ liệu vì thực chất View là kết quả của câu lệnh truy vấn Select. Câu lệnh này có thể lấy dữ liệu từ nhiều bảng và nhiều cơ sở dữ liệu đồng thời.

Thuận lợi của view là view với thuộc tính khác nhau có thể được tạo có dữ liệu không bị trùng lặp.

Như vậy, Views thường được sử dụng để:

- Lọc những bản ghi từ bảng theo yêu cầu.
- Bảo vệ dữ liệu từ những người dùng không có quyền
- Giảm độ phức tạp của dữ liệu
- Tóm lược nhiều cơ sở dữ liệu vật lý vào một cơ sở dữ liệu logic

8.1.2 Các kiểu view

Một số kiểu view có thể được tạo, mỗi kiểu có những thuận lợi trong một số trường hợp cụ thể. Kiểu view bạn muốn tạo phụ thuộc vào điều bạn muốn sử dụng view. Vì vậy có một số kiểu view như sau:

+ **Tập con các cột của bảng**: Một view có thể bao gồm một hoặc nhiều cột của bảng, đây là kiểu view phổ biến nhất được dùng để đơn giản dữ liệu hoặc mục đích bảo mật.

+ **Tập con các dòng của bảng**: Một view có thể gồm một hoặc nhiều dòng dữ liệu. Kiểu view này cũng hữu ích cho mục đích bảo mật.

+ **Liên kết 2 hoặc nhiều bảng**: Bạn có thể tạo view bằng cách sử dụng hoạt động liên kết. Hoạt động liên kết phức tạp có thể được đơn giản khi view được dùng.

+ **Thông tin tập hợp**: Bạn có thể tạo view chứa dữ liệu tập hợp. Kiểu view này cũng được dùng để đơn giản hóa các hoạt động phức tạp.

8.1.3 Thuận lợi của view

Một thuận lợi của việc sử dụng view là nó luôn cung cấp dữ liệu cập nhật nhất. Phát biểu SELECT định nghĩa view chỉ được thực thi khi view được truy cập, vì thế tất cả những thay đổi dữ liệu trong bảng được phản ánh trong view.

Một thuận lợi khác là view có thể cấp bảo mật khác bảo mật ở bảng. Truy vấn định nghĩa view được thực thi dưới cấp độ bảo mật của người dùng đã tạo view. Như vậy bạn có thể dùng view để che phần dữ liệu bạn không muốn các lớp người dùng cụ thể nhìn thấy.

Lợi ích của view đối với người sử dụng:

Đối với người sử dụng cuối - End Users

- Dễ dàng để hiểu kết quả: Trong khi tạo ra các view, tên cột có thể được thay đổi sao cho có nghĩa hơn, vì vậy nó làm cho người sử dụng có thể dễ dàng hiểu được cột này biểu diễn cái gì. Việc thay đổi tên cột trong view không tác động đến tên cột trong bảng mà view tham chiếu đến.

- Dễ hơn để thực hiện dữ liệu: có nhiều người biết ít về SQL, các câu lệnh SQL trở nên khó khăn đối với họ khi họ muốn tạo ra truy vấn phức tạp từ nhiều bảng khác nhau. Bởi

vậy, view được tạo tra cho việc truy cập dữ liệu từ nhiều bảng khác nhau, nó giúp người sử dụng dễ dàng trong việc truy cập cơ sở dữ liệu.

Đối với người phát triển hệ thống- Developers

- **Dễ dàng để truy cập dữ liệu (dễ dàng hạn chế việc mất mát dữ liệu):** Một nhà phát triển có thể muốn giấu những thông tin trong một số cột hoặc một số dòng nào đó. Bằng việc dùng view, người sử dụng có thể được phép truy cập linh hoạt tới những dữ liệu mà họ muốn, trong khi vẫn duy trì được bảo mật đối với những dữ liệu khác trong cùng một bảng hoặc trong các bảng khác nhau. Để làm được điều này, view được thiết lập ngăn chặn việc truy cập các cột không được phép, các cột này sẽ bị ẩn đối với người sử dụng.

- Dễ dàng để bảo trì ứng dụng: Chúng ta dễ dàng soát lỗi của View hơn là soát lỗi của những truy vấn. Dò tìm lỗi trong từng bước của mỗi một quá trình trong một view là dễ dàng bởi tất cả các dòng đều là một phần của view.

8.1.4 Các hạn chế SQL Server trong sử dụng view

SQL Server có một số hạn chế khi tạo view và sử dụng view là:

+ **Giới hạn cột**: Một view có thể tham khảo tối đa 1024 cột trong bảng.

+ **Giới hạn CSDL**: Một view chỉ có thể được tạo trên bảng trong CSDL mà người tạo view có thể truy cập.

+ **Giới hạn bảo mật**: Người tạo view phải có quyền truy cập tất cả các cột được thao khảo trong view.

+ **Các quy tắc toàn vẹn dữ liệu:** Bất kỳ cập nhật, sửa đổi trên view không thể vi phạm các quy tắc toàn vẹn.

+ **Giới hạn số cấp view lồng nhau**: Một view có thể được tạo từ các view khác.

+ **Giới hạn ở phát biểu SELECT**: Phát biểu SELECT của view không thể chứa phát biểu ORDER BY, INTO, COMPUTER và COMPUTER BY.

8.2 Cách dùng view để lọc dữ liệu

8.2.1 Thủ tục lưu trữ

Trong phần này chúng ta sẽ tìm hiểu về thủ tục lưu trữ (Stored procedure-SPs). SPs là công cụ cần thiết cho bất kỳ hệ quản trị cơ sở dữ liệu nào. Người phát triển hoặc người quản trị viết SPs để thực hiện những công việc quản trị hoặc các quy tắc dữ liệu phức tạp. SPs có thể chứa những câu lệnh thực hiện dữ liệu (DML) hoặc những câu lệnh truy vấn dữ liệu(SELECT). Sử dụng SELECT để trả về giá trị.

Định nghĩa: SPs là tập hợp của các câu lệnh T-SQL được biên dịch trước **(pre_compiled).** SPs được đặt tên và được xử lý như một khối lệnh thống nhất (chứ không phải thực hiện rời rạc các câu lệnh).

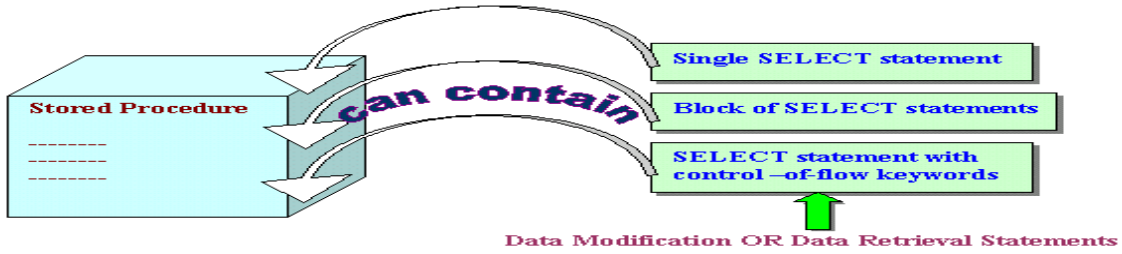
SQL Server cung cấp một số các thủ tục được lưu trữ sẵn trong hệ thống giúp thực hiện một số công việc thường xuyên. Nó được gọi là thủ tục hệ thống –System stored procedures. Còn những thủ tục do người sử dụng tự viết gọi là User stored procedures.

SPs trong SQL Server cũng tương tự như khái niệm về thủ tục trong các ngôn ngữ lập trình khác, bởi vì:

Chấp nhận biến đầu vào và trả lại kết quả khi thực hiện.

Chứa những câu lệnh dùng trong lập trình có thể thao tác với cơ sở dữ liệu và có thể gọi đến các thủ tục khác.

Trả lại giá trị trạng thái khi thủ tục được gọi để xác định việc thực hiện thủ tục thành công hay thất bại.



Hình 8.1 Các thành phần của SPs

Lợi ích khi quản lý dữ liệu bằng SPs

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

Tốc độ truy cập dữ liệu nhanh hơn: Khi thực thi một câu lệnh SQL thì SQL Server phải kiểm tra permission xem user gửi câu lệnh đó có được phép thực hiện câu lệnh hay không đồng thời kiểm tra cú pháp rồi mới tạo ra một execute plan và thực thi. Nếu có nhiều câu lệnh như vậy gửi qua network có thể làm giảm đi tốc độ làm việc của server. SQL Server sẽ làm việc hiệu quả hơn nếu dùng stored procedure vì người gửi chỉ gửi một câu lệnh đơn và SQL Server chỉ kiểm tra một lần sau đó tạo ra một execute plan và thực thi. Nếu stored procedure được gọi nhiều lần thì execute plan 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 stored procedure được tạo ra nó có thể được sử dụng lại. Điều này sẽ làm cho việc bảo trì (maintainability) dễ dàng hơn do việc tách rời giữa

business rules (tức là những logic thể hiện bên trong stored procedure) và cơ sở dữ liệu. Ví dụ nếu có một sự thay đổi nào đó về mặt logic thì ta chỉ việc thay đổi code bên trong stored procedure mà thôi. Những ứng dụng dùng stored procedure này có thể sẽ không cần phải thay đổi mà vẫn tương thích với business rule mới.

Nhất quán: Lợi ích nữa của SPs là thiết đặt được ràng buộc dữ liệu để đảm bảo tính nhất quán. Người sử dụng không thể thực hiện tuỳ tiện dữ liệu để làm mất tính đúng đắn 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 vào một số bảng, ta có thể viết một stored procedure để truy xuất dữ liệu và chỉ cho phép user đó được sử dụng stored procedure đã viết sẵn mà thôi chứ không thể thao tác trực tiếp trên các bảng đó. Ví dụ, ta có thể tạo ra SPs để ta làm chủ và chỉ cung cấp quyền EXCUTE cho những SPs này, vì thế những người sử dụng khác không được phép trực tiếp làm việc với dữ liệu.

Ngoài ra stored procedure có thể được encrypt (mã hóa) để tăng cường tính bảo mật.

SPs chia làm 2 loại:

System stored procedures: Thủ tục mà những người sử dụng chỉ có quyền thực hiện, không được phép thay đổi.

User stored procedures: Thủ tục do người sử dụng tạo và thực hiện

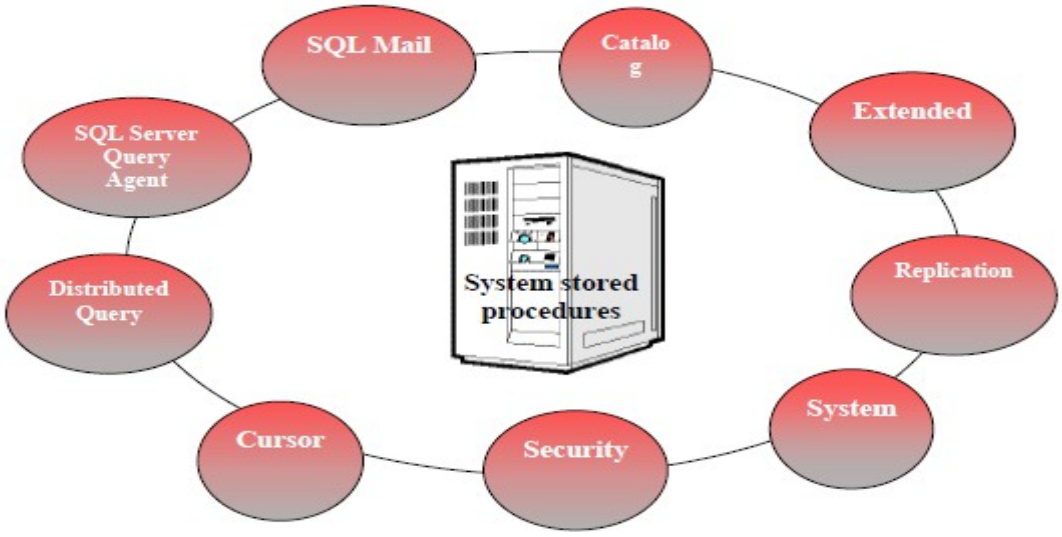
8.2.2. System stored procedures

Là những stored procedure chứa trong Master Database và thường bắt đầu bằng tiếp đầu ngữ **sp_** .

Các stored procedure này thuộc loại built-in và chủ yếu dùng trong việc quản lý cơ sở dữ liệu (administration) và bảo mật (security). Ví dụ bạn có thể kiểm tra tất cả các processes đang được sử dụng bởi user DomainName\Administrators bạn có thể dùng :

sp_who @loginame='DomainName\Administrators'

Người ta có thể chia các System stored procedures thành các loại sau:



Hình 8.2. Các loại của System stored procedures

Có hàng trăm system stored procedure trong SQL Server. Bạn có thể xem chi tiết phân loại và nội dung của từng thủ tục trong SQL Server Books Online.

Sau đây là một số thủ tục hệ thống thường sử dụng

System stored procedure	Chức năng
sp_Databases	Danh sách những Database có thể (avaiable) trên Server (Danh sách này sẽ là khác nhau tuỳ thuộc vào quyền của người sử dụng)
sp_server_info	Chi tiết những thông tin về Server, ví dụ như tập các đặc tính, phiên bản…
sp_stored_procedures	Danh sách tất cả các thủ tục có thể trên môi trường hiện tại
sp_tables	Danh sách tất các bảng có thể trên môi trường hiện tại
sp_start_job	Khởi động tất cả các automated task ngay lập tức
sp_stop_job	Ngừng lại tất cả các automated task đang chạy
sp_password	Thay đổi password cho login account
sp_configure	Thay đổi lựa chọn cấu hình chung của SQL SERVER. Khi người sử dụng không lựa chọn thì hệ thống sẽ hiển thị cấu hình mặc định.
sp_help	Hiển thị thông tin về bất kỳ đối tượng nào trong Database
sp_helptext	Hiển thị nội dung (text) của các đối tượng

8.2.3 User-defined Stored Procedures

Cú pháp

Người sử dụng có thể sử dụng câu lệnh CREATE PROCEDURE để tạo thủ tục trong CSDL hiện tại.

Database owner mặc định có quyền sử dụng câu lệnh CREATE PROCEDURE.

Cú pháp: ***CREATE PROC[EDURE] procedure_name***

Ví dụ:

```
CREATE PROCEDURE London_Flights AS
PRINT 'This code displays the details of flights to London'
SELECT * FROM flight WHERE destination='Lon'
```

Các chỉ dẫn

Tên thủ tục phải tuân theo quy tắc đặt tên

Tất cả các đối tượng của cơ sở dữ liệu có thể được tạo trong SPs, trừ những đối tượng: defaults, rules, triggers, procedures, và views.

Những đối tượng đã được tạo có thể được tham chiếu đến ngay khi nó được tạo.

Stored procedures có thể tham chiếu tới những bảng phụ (temporary tables). Có thể có 2100 biến trong stored procedure.

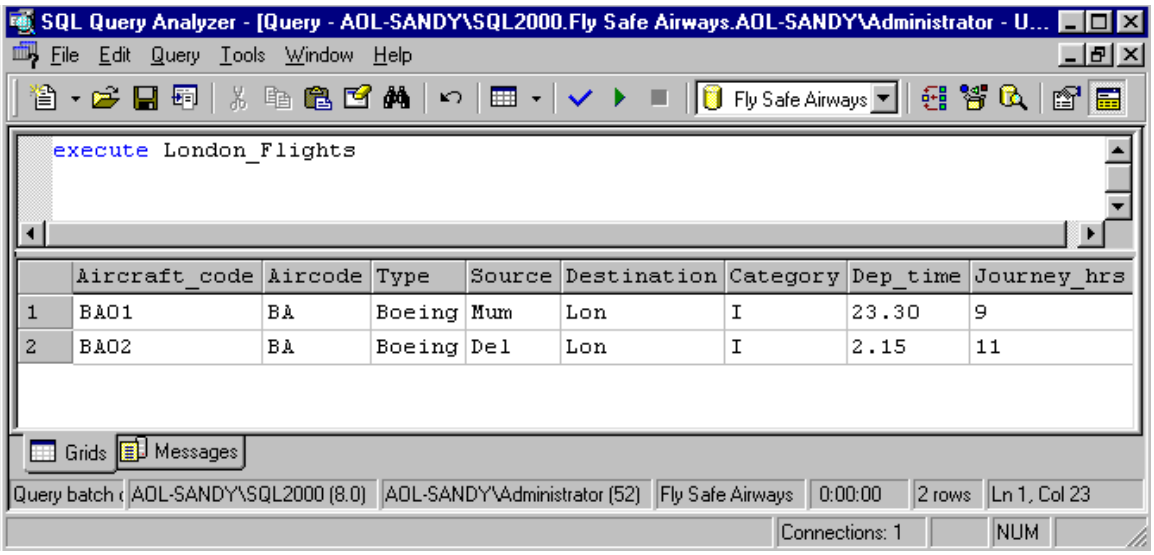
Chúng ta có thể tạo nhiều biến địa phương trong stored procedure nếu bộ nhớ cho phép.

Kích thước tối đa cho stored procedure là 128 MB.

Thực hiện User-defined Stored Procedures (Lời gọi thủ tục)

Cú pháp: ***EXEC[UTE] procedure_name***

Ví dụ:



Hình 8.3.Thực hiện User-defined Stored Procedures

Sử dụng biến trong Stored Procedures

Biến có thể được sử dụng để nhập dữ liệu vào (INPUT) hoặc xuất dữ liệu ra ngoài (OUTPUT)

Cú pháp:

CREATE PROCEDURE procedure_name

@Parameter_name data_type

AS

:

Ví dụ:

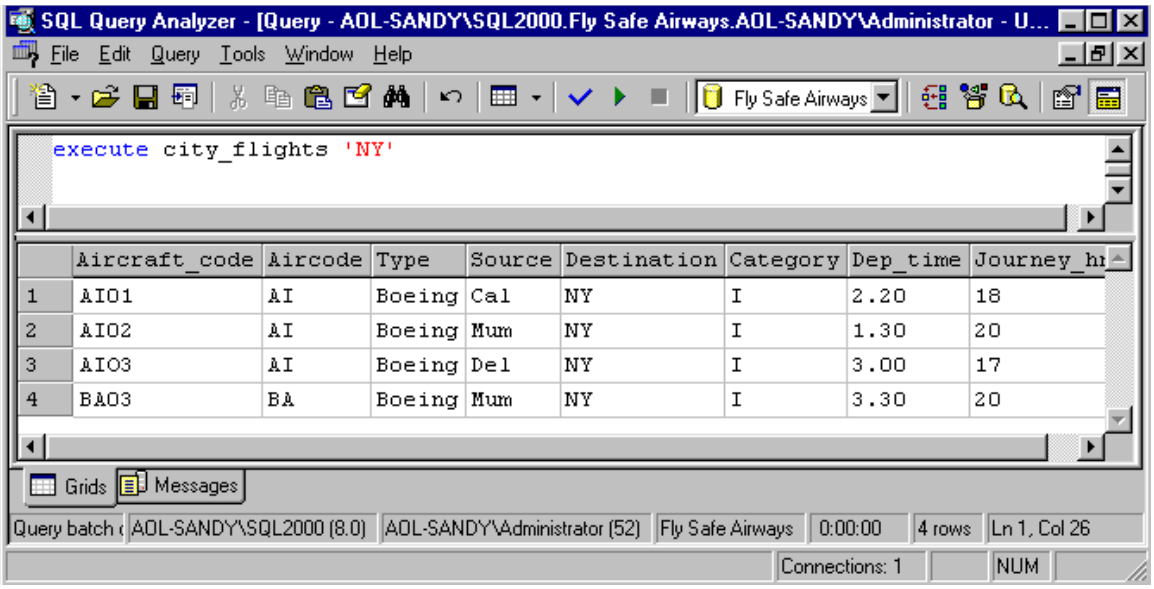
CREATE PROCEDURE city_flights

@v_city varchar(15)

AS

SELECT * FROM flight WHERE destination=@v_city

Thực hiện thủ tục có biến:



Hình 8.4. Thực hiện User-defined Stored Procedures có biến

Nếu có nhiều biến trong thủ tục thì khi thực hiện ta liệt kê theo thứ tự các biến và phải cách nhau bằng dấu phẩy.

Biên dịch lại - Re-compiling Stored Procedures

Khi người sử dụng làm thay đổi tới những index của bảng. Stored procedures phải được biên dịch lại (recompiled) để chấp nhận những thay đổi đó.

Có 3 cách để biên dịch lại procedures:

Sử dụng sp_recompile system stored procedure: Bạn có thể sử dụng cách này để biên dịch lại thủ tục ở lần chạy kế tiếp của nó.

Cú pháp: ***sp_recompile [@objectname=] 'object'***

Chỉ ra WITH RECOMPILE trong câu lệnh CREATE PROCEDURE: SQL Server sẽ biên dịch lại thủ tục ở mỗi lần nó thực hiện.

Cú pháp:

CREATE PROCEDURE procedure_name

@Parameter_name data_type

WITH RECOMPILE

AS

:

Chỉ ra WITH RECOMPILE trong câu lệnh EXECUTE:

Lời gọi thủ tục (Biên dịch lại ngay ở lần thực hiện này.)

Cú pháp: ***EXEC[UTE] procedure_name WITH RECOMPILE***

Sửa cấu trúc của Stored Procedures

Câu lệnh ALTER PROCEDURE được sử dụng để sửa SP.

Cú pháp tương tự CREATE PROCEDURE chỉ thay từ CREATE bằng ALTER.

Việc sửa chữa vẫn lưu lại quyền của người sử dụng (user permissions)

Thông báo lỗi

Trả về mã lỗi (Code) hoặc câu lệnh RAISERROR có thể được sử dụng để nhắc người sử dụng về lỗi.

Mã lỗi trả về là số nguyên.

Câu lệnh RAISERROR giải thích lỗi và chỉ ra mức độ lỗi.

Return Codes

Return codes là số nguyên, giá trị mặc định là 0.

Giá trị của **Return codes** phải được trả về vào một biến

Cú pháp:

DECLARE @return_variable_name data_type

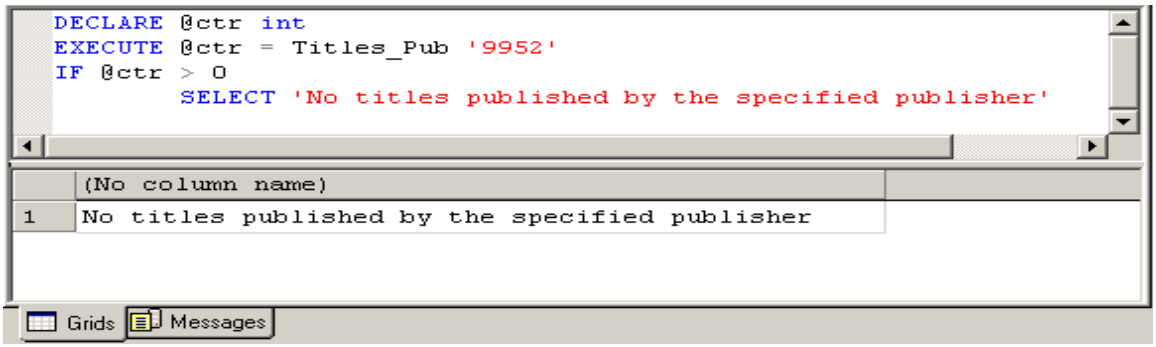
EXECUTE @return_variable_name = procedure_name

Ví dụ:

```
ALTER PROCEDURE Titles_Pub
@v_pubid char(4)
AS
DECLARE @v_return int
SELECT @v_return=COUNT(*)
FROM titles WHERE pub_id = @v_pubid
IF @v_return>0
SELECT * FROM titles WHERE pub_id = @v_pubid
ELSE
```

RETURN @v_return+1

Kết quả thực hiện:



Câu lệnh RAISERROR

Trong SPs, chúng ta có thể sử dụng câu lệnh PRINT để hiển thị thông báo lỗi cho người sử dụng. Tuy nhiên, những lời nhắc này chỉ là tạm thời và chỉ hiển thị cho người sử dụng chúng ta cần sử dụng câu lệnh RAISERROR để ghi lại những lỗi này và gán cho nó mức severity.

Cú pháp:

RAISERROR ({msg_id | msg_str}{,severity, state}

[WITH option[...n]]

Ví dụ:

```
WHILE @v_ctr > 0
BEGIN
SELECT @v_ctr * @v_ctr
SELECT @v_ctr = @v_ctr – 1
IF @v_ctr = 2
BEGIN
RAISERROR('Counter has fallen below 3', 1, 2)
BREAK
END
END
```

Kết quả:

25

16

9

Nội dung thông báo:

(1 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)

Msg 50000, Level 1, State 50000

Counter has fallen below 3

Sử dụng SELECT để trả về giá trị

Bạn có thể trả về giá trị bằng phát biểu SELECT trong thủ tục hoặc trả về kết quả được thiết lập từ truy vấn SELECT

Ví dụ: sau trả về giá trị bằng phát biểu SELECT trong thủ tục

```
USE MyDB
GO
CREATE PROCEDURE InDienThoai @ma_kh int
AS
SELECT HO_KH, Ten_KH, DienThoai
FROM KHACHHANG
WHERE Ma_KH = @ma_kh
GO
```

Gọi thủ tục với mã khách hàng là 777:

```
EXEC InDienThoai 777
```

Kết qủa như sau:

HO_KH	TEN_KH	DienThoai
Nguyen thi	Minh Khai	9145662

(1 row(s) affected)

Ví dụ sau trả về giá trị biến bằng phát biểu SELECT. Giá trị trả cho biến kq bằng lệnh SELECT như sau:

```
USE MyDB
GO
IF EXISTS (SELECT name
        FROM sysobjects
        WHERE name = 'KiemTraDT' AND type = 'P')
DROP PROCEDURE KiemTraDT
GO
CREATE PROCEDURE KiemTraDT @ma_kh INT
AS
DECLARE @kq varchar (50)
IF (SELECT DienThoai
        FROM KHACHHANG
```

WHERE Ma_KH = @ma_kh) IS NOT NULL

SET @kq = ‘da luu so dien thoai’

ELSE

SET @kq = ‘chua luu so dien thoai’

SELECT “ketqua” = @kq

PRINT ‘tra ve gia tri cho bien kq bang select’

GO

Gọi thủ tục với mã khách hàng là 779

Exec KiemTraDT 779

Kết quả trả về là

Ketqua

…….

Chua luu so dien thoai

(1 row(s) affected)

Tra ve gia tri cho bien kq bang select.

8.3 Cách cập nhật dữ liệu vào view (trang 147 csdl II)

View có thể sửa chữa dữ liệu được xây dựng trên bảng:

- View chứa đựng ít nhất một bảng được định nghĩa sau mệnh đề FROM.

- Không chứa những hàm nhóm hoặc mệnh đề GROUP BY, UNION, DISTINCT, hoặc TOP.

- View không chứa những cột được suy ra từ những cột khác

Các chức năng có thể thực hiện trên View tương tự như đối với bảng. Chúng ta có thể thực hiện các câu lệnh INSERT, UPDATE, và DELETE trên View.

Khi chúng ta thay đổi dữ liệu thông qua View, đồng nghĩa với việc chúng ta thay đổi dữ liệu trên các bảng mà View đó đang tham chiếu. Tuy nhiên, nên thực hiện một số các quy luật sau khi thực hiện sửa chữa dữ liệu thông qua View.

Câu lệnh SELECT trong định nghĩa View không nên chứa:

Các hàm nhóm dữ liệu (Aggregate functions)

Các mệnh đề TOP, GROUP BY. UNION, hoặc DISTINCT.

Cột có giá trị được suy ra từ các cột khác(derived columns)

Sau mệnh đề FROM trong câu lệnh SELECT nên có ít nhất một bảng. Ví dụ, View sau đây không thể cập nhật dữ liệu:

CREATE VIEW NoTable AS

SELECT Getdate() AS CurrentDate

@@LANGUAGE AS CurrentLanguage

Chúng ta chỉ có thể cập nhật và thêm dữ liệu vào 1 bảng đứng sau mệnh đề FROM của View. Nếu muốn cập nhật dữ liệu trên nhiều bảng, chúng ta phải sử dụng INSTEAD OF trigger.

Nếu như bảng được tham chiếu trong View chứa cột có ràng buộc NOT NULL không phải là một phần của View thì chúng ta phải gán giá trị mặc định cho cột này để có thể thêm dữ liệu cho bản ghi.

Nếu định nghĩa View có chứa lựa chọn WITH CHECK, tất cả các cột được sửa chữa phải thoả mãn điều kiện trong câu lệnh SELECT. Ví dụ, nếu câu lệnh SELECT có chứa mệnh đề WHERE emp_id<=500, thì chúng ta không thể sửa lại dữ liệu trong cột emp_id có giá trị lớn hơn 500.

Chúng ta có thể xoá dữ liệu nếu như View chỉ tham chiếu đến 1 bảng. Để xoá dữ liệu trên View có tham chiếu sang nhiều bảng, chúng ta phải sử dụng INSTEAD OF trigger.

Chúng ta cũng có thể sử dụng DPVs để cập nhật dữ liệu cho các bảng tham chiếu.

Sửa cấu trúc Views

Chúng ta có thể sử dụng câu lệnh ALTER VIEW để thực hiện sửa cấu trúc của View. Cú pháp của nó tương tự như cú pháp của lệnh CREATE VIEW, chỉ cần thay thế từ khóa CREATE bằng từ khoá ALTER.

Cú pháp:

ALTER VIEW <Viewname> [WITH SCHEMABINDING]

AS <Select_Statement>

[WITH CHECK OPTION]

Ví dụ:

```
ALTER VIEW Try AS
SELECT flight.aircraft_code,
airlines_master.airline_name,
flight.source, flight.destination, flight.dep_time,
flight.journey_hrs
FROM airlines_master INNER JOIN Flight
ON airlines_master.aircode = flight.aircode
```

Xoá Views

Khi một View nào đó không còn cần thiết nữa, chúng ta có thể xoá nó.

Cú pháp: ***DROP VIEW <Viewname>***

Ví dụ:DROP VIEW Try

Chương 9. THIẾT KẾ CƠ SỞ DỮ LIỆU

9.1 Khái niệm về mô hình quan hệ (Ôn lại)

Mô hình cơ sở dữ liệu quan hệ là cách thức biểu diễn dữ liệu dưới dạng bảng hay còn gọi là quan hệ, mô hình được xây dựng dựa trên cơ sở lý thuyết đại số quan hệ.

Cấu trúc dữ liệu: dữ liệu được tổ chức dưới dạng quan hệ hay còn gọi là bảng.

Thao tác dữ liệu: sử dụng những phép toán mạnh (bằng ngôn ngữ SQL).

9.1.1. Thuộc tính: là một tính chất riêng biệt mô tả một thông tin nào đó của một đối tượng trong CSDL.

Chẳng hạn với bài toán quản lý sinh viên, đối tượng sinh viên cần phải chú ý đến các đặc trưng riêng như: Họ tên, Mã SV, Ngày sinh, Giới tính, Địa chỉ. Các đặc trưng này là các thuộc tính.

- Mỗi một thuộc tính được đặc trưng bởi ba thành phần:

+ *Tên thuộc tính*: Trong cùng một đối tượng không có hai thuộc tính cùng tên.

+ *Kiểu dữ liệu*: Các thuộc tính phải thuộc vào một kiểu dữ liệu nhất định (số , chuỗi, ngày tháng, logic, hình ảnh…). Kiểu dữ liệu ở đây là kiểu đơn.

+ *Miền giá trị*: Thông thường mỗi thuộc tính chỉ chọn lấy giá trị trong một tập con của kiểu dữ liệu và tập hợp con đó gọi là miền giá trị của thuộc tính đó.

Ví dụ: Thuộc tính Ngày sinh thì có kiểu dữ liệu là Datetime

Thường người ta dùng các chữ cái hoa A, B, C …để biểu diễn các thuộc tính hoặc A_1, A_2…A_n để biểu diễn một số lượng lớn các thuộc tính.

9.1.2. Quan hệ

Lược đồ quan hệ (Relation Schema)

Tập tất cả các thuộc tính cần quản lý của một đối tượng cùng với mối liên hệ giữa chúng được gọi là lược đồ quan hệ. Lược đồ quan hệ Q với tập thuộc tính { A_1, A_2…A_n} được viết là Q(A_1, A_2…A_n). Tập các thuộc tính của Q được ký hiệu là Q^+

Ví dụ: Lược đồ quan hệ sinh viên với các thuộc tính như là:

SinhViên (Họ tên, Mã SV, Ngày sinh, Giới tính, Địa chỉ)

Nhiều lược đồ quan hệ cùng nằm trong một hệ thống quản lý được gọi là một lược đồ cơ sở dữ liệu.

Ví dụ: lược đồ cơ sở dữ liệu để quản lý điểm thi của sinh viên có thể gồm các lược đồ quan hệ sau:

SinhViên (Họ tên, Mã SV, Ngày sinh, Giới tính, Địa chỉ)

Điểm (Mã SV, Điểm thi).

Quan hệ (Relation): Sự thể hiện của lược đồ quan hệ Q ở một thời điểm nào đó được gọi là quan hệ.

9.1.3. Bộ

Bộ là tập mỗi giá trị liên quan của tất cả các thuộc tính của một lược đồ quan hệ. Thường người ta dùng các chữ cái thương như t, p, ..để biểu diễn các bộ. Chẳng hạn để nói bộ t thuộc quan hệ r ta viết t r.

Về trực quan thì mỗi quan hệ xem như một bảng, trong đó mỗi cột là một thông tin về một thuộc tính, mỗi dòng là thông tin về một bộ.

Họ tên	Mã SV	Ngày sinh	Giới tính	Địa chỉ
Lê Vân	4515202	12/09/84	Nữ	Hà Nội
Hoàng Tùng	4516802	21/03/84	Nam	Bắc Ninh
Trương Định	4620503	15/05/85	Nam	Hà Nam
Phạm An	4612203	16/04/85	Nam	Nam Đinh
Đỗ Cung	4521402	20/01/84	Nam	Nghệ An

Bảng1. Quan hệ SinhViên

Bảng một chỉ ra một ví dụ của quan hệ SinhViên tương ứng với lược đồ SinhViên ở trên. Mỗi bộ trong quan hệ biểu diễn một đối tượng sinh viên cụ thể.

9.1.4. Khóa

Cho lược đồ quan hệ R, S R^+. S được gọi là một siêu khóa (Superkey) của lược đồ quan hệ R nếu với hai bộ tuy ý trong quan hệ R thì giá trị của các thuộc tính trong S là khác nhau.

Một lược đồ quan hệ có thể có nhiều siêu khóa. Siêu khóa chứa ít thuộc tính nhất được gọi là khóa chỉ định, trong trường hợp lược đồ quan hệ có nhiều khóa chỉ định, thì khóa được chọn để cài đặt gọi là khóa chính (Primary key)(gọi tắt là khóa).

Các thuộc tính tham gia một khóa được gọi là thuộc tính khóa (prime key), ngược lại được gọi là thuộc tính không khóa (non prime key).

Một thuộc tính được gọi là khóa ngoại nếu nó là thuộc tính của một lược đồ quan hệ này nhưng lại là khóa chính của lược đồ quan hệ khác.

Khóa phụ (second key): đóng vai trò khi ta muốn sắp xếp lại dữ liệu trong bảng.

Ví dụ: Ta có bảng SINHVIEN (MaSV, Hoten, GioiTinh, Diem).

Muốn sắp xếp lại danh sách sinhviên theo thứ tự a, b, c.. của Họ tên. Khi đó thuộc tính Hoten được gọi là khóa phụ.

9.2 Hiểu và phân biệt được mô hình logic và mô hình vật lý.

9.2.1

9.2.2 Cấu trúc vật lý của CSDL

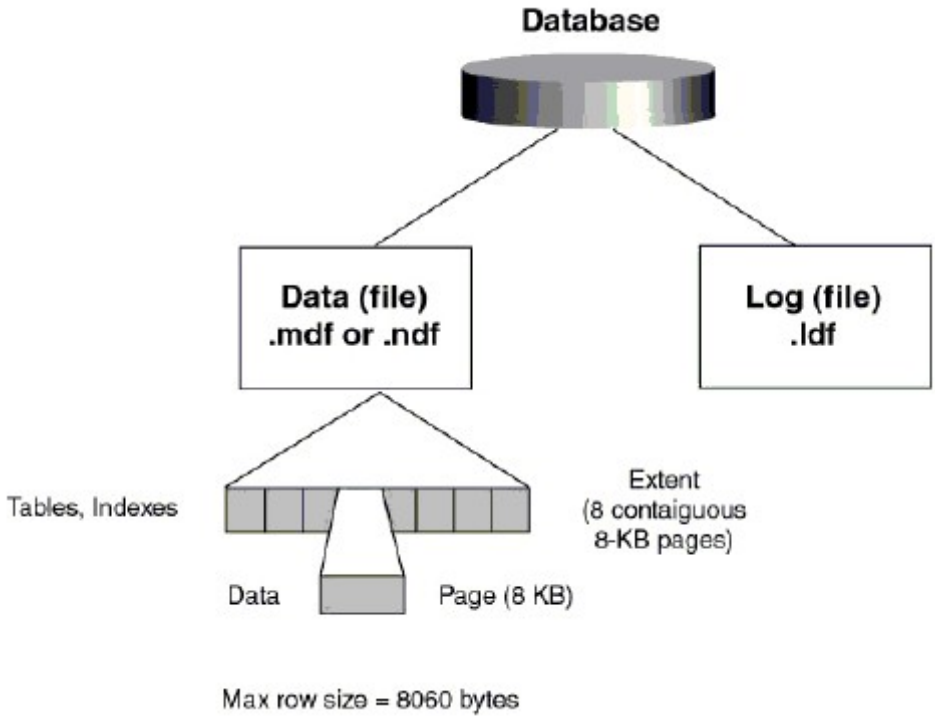
Mỗi một cơ sở dữ liệu trong SQL Server đều chứa ít nhất một data file chính (primary), có thể có thêm một hay nhiều data file phụ (Secondary) và một transaction log file.

Primary data file (thường có phần mở rộng **.mdf**): đây là file chính chứa data và những system tables.

Secondary data file (thường có phần mở rộng **.ndf**): đây là file phụ thường chỉ sử dụng khi cơ sở dữ liệu được phân chia để chứa trên nhiều dĩa.

Transaction log file (thường có phần mở rộng **.ldf**): đây là file ghi lại tất cả những thay đổi diễn ra trong một cơ sở dữ liệu và chứa đầy đủ thông tin để có thể roll back hay roll forward khi cần.

Data trong SQL Server được chia thành từng **Page** 8KB và 8 page liên tục tạo thành một **Extent** như hình vẽ dưới đây:



Hình 9.1 Cấu trúc vật lý của một cơ sở dữ liệu trong SQL Server.

Trước khi SQL Server muốn lưu dữ liệu vào một table nó cần phải dành riêng một khoảng trống trong data file cho table đó. Những khoảng trống đó chính là các extents.

Có 2 loại Extents:

Mixed Extents (loại hỗn hợp) dùng để chứa dữ liệu của nhiều tables trong cùng một Extent.

Uniform Extent (loại thuần nhất) dùng để chứa dữ liệu của một table. Đầu tiên SQL Server dành các Page trong Mixed Extent để chứa dữ liệu cho một table sau đó khi dữ liệu tăng trưởng thì SQL dành hẳn một Uniform Extent cho table đó.

9.2.3 Cấu trúc logic của một cơ sở dữ liệu

Hầu như mọi thứ trong SQL Server được tổ chức thành những objects ví dụ như tables, views, stored procedures, indexes, constraints.... Những system objects trong SQL Server thường có bắt đầu bằng chữ *sys* hay *sp*. Các objects trên sẽ được nghiên cứu lần lượt trong các bài sau.

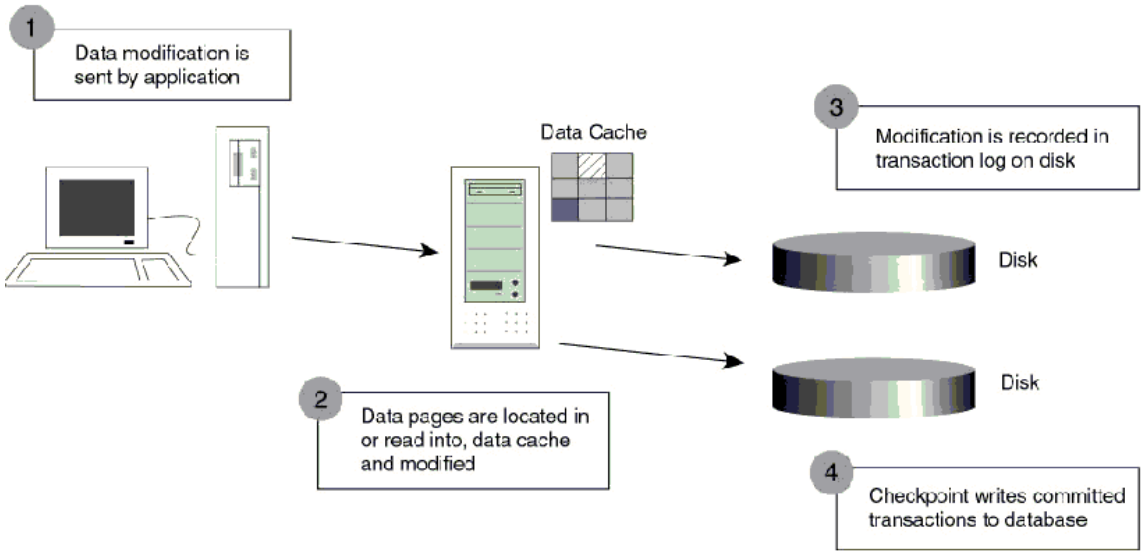
9.2.4 Nguyên tắc hoạt động của Transaction log file.

Transaction log file trong SQL Server dùng để ghi lại các thay đổi xảy ra trong cơ sở dữ liệu.

Quá trình này diễn ra như sau:

Đầu tiên khi có một sự thay đổi dữ liệu như Insert, Update, Delete được yêu cầu từ các ứng dụng, SQL Server sẽ tải (load) data page tương ứng lên memory (vùng bộ nhớ này gọi là data cache), sau đó dữ liệu trong data cache được thay đổi(những trang bị thay đổi còn gọi là *dirty-page*).

Tiếp theo mọi sự thay đổi đều được ghi vào transaction log file cho nên người ta gọi là *write-ahead* log. Cuối cùng thì một quá trình gọi là **Check Point Process** sẽ kiểm tra và viết tất cả những transaction đã được commited (hoàn tất) vào đĩa cứng (flushing the page).



Hình 9.1. Quá trình hoạt động của Transaction

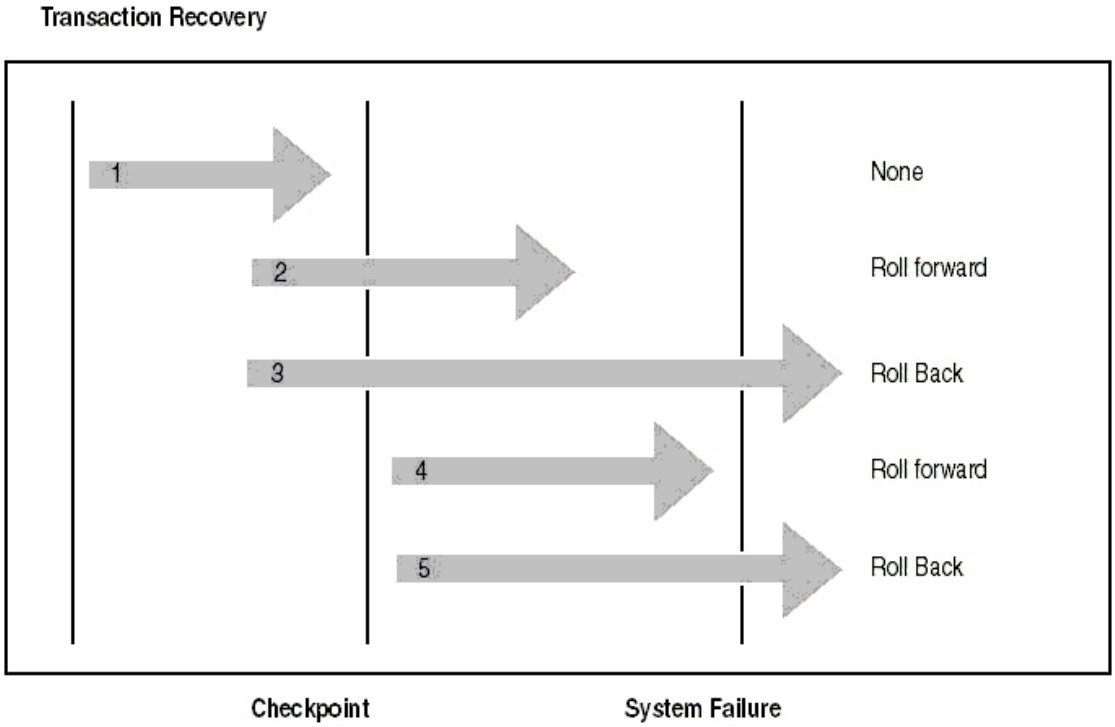
Ngoài **Check Point Process** những dirty-page còn được đưa vào đĩa bởi một **Lazy writer**. Đây là một thành phần làm nhiệm vụ quét qua phần data cache theo một chu kỳ nhất định sau đó lại dừng để chờ lần quét tới.

Check Point Process hoạt động như thế nào để có thể đảm bảo một transaction được thực thi mà không gây ra dữ liệu "bẩn"-dirty data.

Trong hình vẽ bên dưới (Transaction Recovery), một transaction được biểu diễn bằng một mũi tên. Trục nằm ngang là trục thời gian. Giả sử một Check Point được đánh dấu vào thời điểm giữa transaction 2 và 3 như hình vẽ và sau đó sự cố xảy ra trước khi gặp một Check point kế tiếp. Như vậy khi SQL Server được restart nó sẽ dựa trên những gì ghi trong transaction log file để phục hồi dữ liệu (xem hình vẽ).

Điều đó có nghĩa là SQL Server sẽ không cần làm gì cả đối với transaction 1 vì tại thời điểm Check point data đã được lưu vào đĩa rồi. Trong khi đó transaction 2 và 4 sẽ được Roll Forward vì tuy đã được commited nhưng do sự cố xảy ra trước thời điểm check point kế tiếp nên dữ liệu chưa kịp lưu vào đĩa. Tức là dựa trên những thông tin được ghi trên log file SQL Server hoàn toàn có đầy đủ cơ sở để viết vào đĩa cứng. Còn transaction 3 và 5 thì chưa được

commited (do bị down bất ngờ) cho nên SQL Server sẽ Roll Back hai transaction này dựa trên những gì được ghi trên log file.



Hình 9.2. Khôi phục Transaction

9.3 Cách backup và restore dữ liệu (sao lưu và phục hồi dữ liệu)

9.3.1 Sao lưu dữ liệu (Backup dữ liệu)

Sao lưu dữ liệu là một trong những tác vụ quan trọng nhất của NQTCSDL. Lưu trữ các tập tin sao lưu và có kế hoạch cẩn thận để phụ hồi cho phép NQTCSDL khôi phụ hệ thống khi sự cố hư hỏng xảy ra. Trách nhiệm của NQTCSDL là giữ hệ thống hoạt động ổn định càng lâu càng tốt, trong trường hợp hệ thống hư hỏng thì công việc phục hồi phải nhanh nhất, giúp giảm thời gian chết, cả sự bất tiện và chi phí. Các công nghệ hệ thống đĩa liên cung và khả năng chịu lỗi có thể giúp ích nhưng chúng không thể tay thế cho kế hoạch sao lưu tốt và phương pháp sao lưu tin cậy.

Những nguyên nhân gây ra mất dữ liệu:

- Đĩa cứng hư
- Vô ý hay cố ý sửa đổi dữ liệu như xóa hay thay đổi dữ liệu.
- Trộm cắp
- Virus

Để tránh việc mất dữ liệu, chúng ta nên thường xuyên sao lưu cơ sở dữ liệu. Nếu như dữ liệu hay cơ sở dữ liệu bị hư thì ta có thể dùng bản sao lưu (backup) này để khôi phục lại cơ sở dữ liệu bị mất.

Sao lưu-backup một cơ sở dữ liệu (CSDL:) là tạo một bản sao CSDL, ta có thể dùng bản sao để khôi phục lại CSDL nếu CSDL bị mất. Bản sao gồm tất cả những file có trong CSDL kể cả transaction log.

Transaction log (hay log file) chứa những dữ liệu thay đổi trong CSDL (Ví dụ như khi ta thực hiện các lệnh INSERT, UPDATE, DELETE). Transaction log được sử dụng trong suốt quá trình khôi phục để **roll forward** những transaction hoàn thành và **roll back** những transaction chưa hoàn thành.

Roll back là hủy bỏ giao dịch chưa hoàn thành khi hệ thống xảy ra sự cố,…
(hoặc trong trường hợp sao lưu, khi đã thực hiện xong việc sao lưu mà giao dịch
chưa hoàn thành) (xem chi tiết ở phần Transaction).

Roll forward là khôi phục tất cả giao dịch đã hoàn thành khi hệ thống xảy ra sự cố,… (hoặc trong trường hợp sao lưu, những giao dịch đã hoàn thành khi đã thực hiện xong việc sao lưu) (xem chi tiết ở phần Transaction).

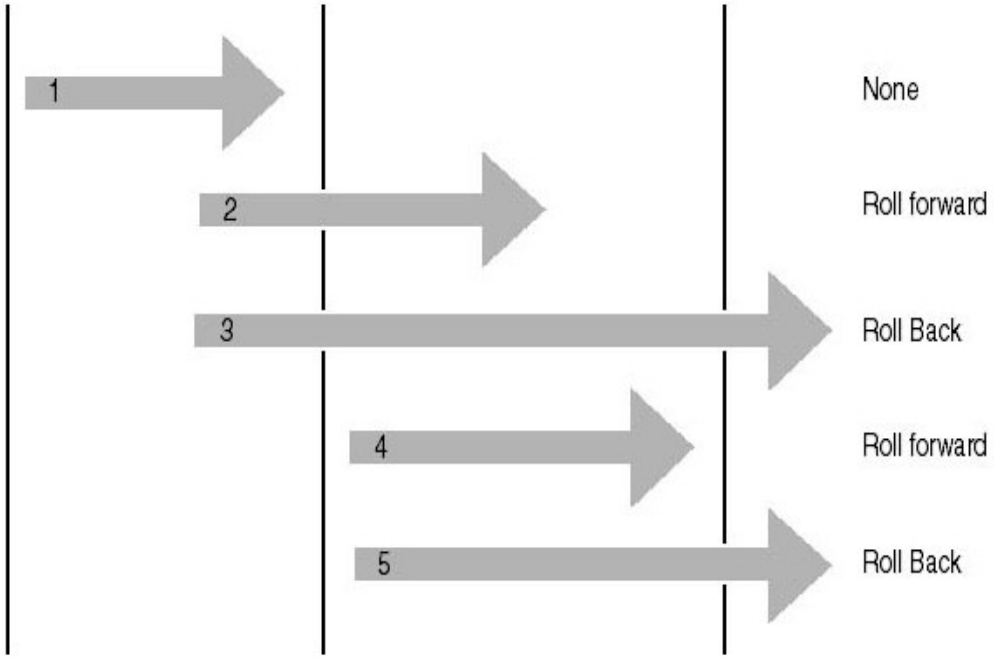
Checkpoint là thời điểm ghi lại tất cả những trang dữ liệu thay đổi lên đĩa.

Hoạt động sao lưu liên quan đến việc lưu trữ dữ liệu từ CSDL để dùng lại nó, nó tượng tự như hoạt động sao lưu được thực hiện bởi hệ điều hành. Trong khi sao lưu, dữ liệu được phép sao chép từ CSDL và lưu ở một nơi khác, sự khác nhau giữa sao lưu cấp hệ điều hành và sao lưu cấp CSDL là sao lưu cấp hệ điều hành có thể lưu từng tập tin riêng, nhưng ngược lại sao lưu mức CSDL thì sao lưu toàn bộ CSDL.

Ví dụ minh họa roll back và roll forward:

- Giao dịch 1 commit trước khi checkpoint, không làm gì cả vì dữ liệu đã được thay đổi trong CSDL (ứng với số 1 trong hình).
- Giao dịch 2 và 4 commit sau khi checkpoint nhưng trước khi hệ thống xảy ra sự cố, do đó những giao dịch này được tạo lại từ log file. Điều này gọi là roll forward (ứng với số 2 và 4 trong hình).
- Giao dịch 3 và 5 chưa commit khi hệ thống xảy ra sự cố, do đó những giao dịch này không được thực hiện và trả về CSDL khi chưa xảy ra giao dịch. Điều này gọi là roll back (ứng với số 3 và 5 trong hình).

Hình minh họa quá trình khôi phục giao dịch:



Hình 9.3 .Khôi phục giao dịch

Sao lưu một transaction log là chỉ sao lưu những thay đổi xảy ra trong transaction log kể từ lần sao lưu transaction log cuối cùng.

Sao lưu một CSDL ghi lại toàn bộ trạng thái của dữ liệu tại thời điểm thực hiện xong sao lưu.

Trong thời gian sao lưu, SQL Server 2000 cho phép thực hiện việc giao dịch (transaction), tức là quá trình sao lưu có thể được thực hiện trong khi SQL Server 2000 đang thực thi.

❖ **Các phương pháp sao lưu**

Có vài phương pháp sao lưu CSDL: sao lưu đầy đủ, sao lưu chỉ những thay đổi, sao lưu tập tin log giao dịch, sao lưu nhóm tập tin và tập tin dữ liệu. Mỗi cái có chế độ hoạt động và tính năng riêng. Sao lưu đầy đủ (Full backup) bao gồm tất cả dữ liệu trong CSDL, nhóm tập tin hoặc tập tin dữ liệu. Sao lưu thay đổi (differential backup) chỉ sao lưu dữ liệu được thay đổi kể từ lần sao lưu gần nhất. Sao lưu tập tin log giao dịch (transaction log) được dùng để sao lưu và cắt tập tin log giao dịch. Sao lưu nhóm tập tin và tập tin dữ liệu (Filegroup và data file) được dùng để sao lưu nhóm tập tin hoặc tập tin dữ liệu xác định.

Full Database Backups: là sao lưu toàn bộ CSDL, tất cả nhóm tập tin và tập tin dữ liệu là một phần của CSDL đều được sao lưu. Đây là kỹ thuật phổ biến dùng cho các CSDL có kích thước vừa và nhỏ.

Copy tất cả data files, user data và database objects như system tables, indexes, user-defined tables trong một database.

Ta nên dùng Full database backup nếu hệ thống có những đặc điểm sau:

- Dữ liệu ít quan trọng và những thay đổi của CSDL có thể tạo lại bằng tay tốt hơn là dùng transaction log.
- CSDL ít thay đổi, như CSDL chỉ đọc.
- Sao lưu 1 CSDL là sao lưu toàn bộ CSDL mà không để ý đến nó có thay đổi so với lần sao lưu cuối cùng không. Điều này có nghĩa là sẽ mất nhiều vùng nhớ cho 1 bản sao và tốn nhiều thời gian để thực hiện sao lưu so với việc dùng transaction log backup và differential backup.

Differential Database Backups: Cho phép bạn sao lưu chỉ những dữ liệu thay đổi kể từ lần sao lưu gần nhất. Kỹ thuật này nhanh hơn và ít tốn không gian lưu trữ hơn sao lưu đầy đủ. Tuy nhiên phương pháp khó khăn và tốn nhiều thời gian để khôi phục dữ liệu hơn.

Differential backup chỉ ghi lại những trang thay đổi ngay sau khi thực hiện sao lưu full database lần cuối cùng. Do đó, sẽ nhanh hơn thực hiện full database backup rất nhiều.

Không như transaction log backup, differential backup không tạo lại CSDL chính xác tại thời điểm xảy ra sự cố, nó cũng như full database backup, chỉ tạo lại CSDL tại thời điểm backup cuối cùng. Vì thế, differential backup thường được bổ sung bằng cách tạo transaction log sau mỗi differential backup. Sử dụng kết hợp database backup, differential backup, và transaction log backup ta có thể giảm tối thiểu khả năng mất dữ liệu và thời gian khôi phục dữ liệu.

Ta nên dùng differential backup nếu hệ thống có những đặc điểm sau:

- Dữ liệu ít quan trọng và những thay đổi của CSDL có thể tạo lại bằng tay tốt hơn là dùng transaction log.
- Tài nguyên để thực hiện database backup giới hạn như thiếu vùng lưu trữ hoặc thời gian thực hiện sao lưu. Ví dụ: CSDL 10 terabyte đòi hỏi nhiều thời gian và vùng lưu trữ để thực hiện sao lưu.
- Tối thiểu hóa thời gian khôi phục và giảm việc mất những giao dịch bằng cách kết hợp differential backup với full database backup và transaction log backup.

Xem xét vídụ sau:

24:00 (Thứ 3)	Full Database backup (1)
6:00 (Thứ 4)	Differential database backup (2)
12:00 (Thứ 4)	Differential database backup (3)
18:00 (Thứ 4)	Differential database backup (4)
24:00 (Thứ 4)	Full Database backup (5)
6:00 (Thứ 5)	Differential database backup (6)
12:00 (Thứ 5)	Differential database backup (7)

Differential backup tạo vào lúc 6:00 ngày thứ tư (2) chứa tất cả những thay đổi của database backup tạo từ lúc 24:00 ngày thứ ba(1).

Differential backup tạo vào lúc 6:00 ngày thứ năm(6) chứa tất cả những thay đổi của database backup tạo từ lúc 24:00 ngày thứ tư (5).

Nếu có khôi phục CSDL đến trạng thái vào 12:00 ngày thứ năm, ta thực hiện những bước sau:

- Khôi phục database tạo lúc 24:00 ngày thứ tư.

Khôi phục differential backup tạo lúc 12:00 ngày thứ năm.

Bất kỳ thay đổi nào sau trưa thứ năm đều bị mất trừ khi có khôi phục transaction log backup.

Chú ý: Các tạo Differential backup và khôi phục cũng tương tự như thực hiện với Full database backup.

Sự khác nhau giữa differential backup và transaction log backup:

- **Giống nhau:** tối thiểu hóa thời gian sao lưu.
- **Khác nhau:**

o Differential backup: chỉ lưu lần thay đổi cuối cùng

o Transaction log backup: chứa tất cả những thay đổi kể từ lần sao lưu full database backup cuối cùng.

Do differential backup lưu những trang thay đổi, gồm những trang dữ liệu và cả trang transaction log thay đổi. Vì sao lưu differential backup có kích thước lớn hơn sao lưu transaction backup nên ta ít sao lưu differential backup thường xuyên so với sao lưu transaction backup. Do đó ta không thể khôi phục CSDL đến thời điểm xảy ra sự cố khi sao lưu differential backup và không thể khôi phục CSDL đến thời điểm mà ta mong muốn

File or File Group Backups : Copy một data file đơn hay một file group. Dùng phương pháp này để sao lưu nhóm tập tin riêng biệt tùy thuộc cách hệ thống được cấu hình. Đồng thời phương pháp này kết hợp với khả năng của SQL Server 2000 để phục hồi một tập tin dữ liệu

đơn. Phương pháp sao lưu tập tin hữu dụng khi bạn không đủ thời gian để sao lưu toàn bộ nhóm tập tin.

Chỉ sao lưu những file CSDL chỉ định. File hoặc file group backup thường được sử dụng chỉ khi không có đủ thời gian để sao lưu toàn bộ CSDL.

Sử dụng file hoặc file group backup có thể tăng tốc độ khôi phục bằng cách chỉ khôi phục những file hoặc filegroup bị hư. Khi sao lưu file hoặc file group thì SQL Server không có sao lưu file transaction log do đó ta phải tạo transaction log backup sau khi sao lưu file hoặc file group.

Ví dụ: Một CSDL có 2 filegroup filegroup_a và filegroup_b nhưng chỉ có đủ thời gian để sao lưu 1 nửa filegroup, do đó:

- Sao lưu filegroup_a vào các ngày thứ hai, tư, sáu.
- Sao lưu transaction log ngay sau khi sao lưu filegroup.
- Sao lưu filegroup_b vào các ngày thứ năm, sáu ,bảy.
- Sao lưu transaction log ngay sau khi sao lưu filegroup

Transaction Log Backups:

Chỉ ghi lại những thay đổi trong transaction log. Transaction log backup chỉ chép lại log file. Nếu chỉ có bản sao log file thì không thể khôi phục lại được CSDL. Nó được sử dụng sau khi CSDL đã được khôi phục lại.

Sao lưu transaction log định kỳ để tạo ra 1 chuỗi transaction log backup cho phép user linh động lựa chọn để khôi phục CSDL. Tạo transaction log backup làm cho CSDL có thể khôi phục đến thời điểm xảy ra sự cố.

Khi tạo transaction log backup, điểm bắt đầu backup là:

- Điểm kết thúc của transaction log backup trước đó (nếu có một transaction log backup tạo ra trước đó).
- Transaction log backup như là một phần cuối của database backup hoặc differential backup mới nhất nếu không có transaction log backup nào được tạo ra trước đó (database backup hoặc differential backup chứa một bản sao vùng tích cực của transaction log).

Cho phép sao lưu transaction log, sao lưu này rất quan trọng cho phục hồi CSDL.Ghi nhận một cách thứ tự tất cả các transactions chứa trong transaction log file kể từ lần transaction log backup gần nhất. Loại backup này cho phép ta phục hồi dữ liệu trở ngược lại vào một thời điểm nào đó trong quá khứ mà vẫn đảm bảo tính nhất quán.

Ta nên dùng transaction log backup nếu hệ thống có những đặc điểm sau:

- Tài nguyên để thực hiện database backup giới hạn như thiếu vùng lưu trữ hoặc thời gian thực hiện backup. Ví dụ: CSDL 10 terabyte đòi hỏi nhiều thời gian và vùng lưu trữ để backup.

- Bất kỳ việc mất những thay đổi sau lần database backup cuối cùng là không thể chấp nhận được. Ví dụ : hệ thống CSDL kinh doanh tài chính, nó không thể chấp nhận mất bất kỳ giao dịch nào.
- Mong muốn trả về CSDL tại thời điểm xảy ra sự cố. Ví dụ muốn khôi phục lại CSDL trước khi xảy ra sự cố 10 phút.
- CSDL thay đổi thường xuyên.

Vì transaction log backup thường sử dụng tài nguyên ít hơn nên chúng được backup thường xuyên hơn. Điều này giảm khả năng mất dữ liệu hoàn toàn.

Ít gặp trường hợp transaction log backup lớn hơn database backup. Ví dụ CSDL có tỉ lệ giao dịch cao và những giao dịch ảnh hưởng đến phần lớn CSDL gây ra transaction log tăng nhanh hoặc ít sao lưu transaction log. Trong trường hợp này tạo transaction log backup thướng xuyên hơn.

Khôi phục CSDL và áp dụng transaction log backup:

- Sao lưu transaction log hiện hành nếu sự cố xảy ra (trừ khi đĩa chứa file transaction log bị hư).
- Khôi phục database backup mới nhất.
- Áp dụng tất cả các transaction log backup được tạo ra sau khi thực hiện full database backup.
- Áp dụng transaction log backup cuối cùng được tạo ra ở bước 1 để khôi phục lại CSDL đến thời điểm xảy ra sự cố.

Vì thế, mặc dù sử dụng transaction log backup tăng khả năng khôi phục, nhưng tạo và áp dụng chúng cũng phức tạp hơn dùng Full Database backup. Khôi

phục CSDL sử dụng cả full database backup và transaction log backup chỉ khi ta có chuỗi transaction log backup liên tục. SQL Server 2000 không cho phép lưu transaction log trong cùng file lưu CSDL. Vì nếu file này hư thì ta không thể sử dụng nó để khôi phục tất cả những thay đổi kể từ lần sao lưu full database backup cuối cùng.

Cắt (truncate) transaction log

Khi SQL Server sao lưu xong transaction log, nó cắt phần không tích cực của transaction log. SQL Server sử dụng lại phần cắt này. Phần không tích cực là phần của transaction log không còn sử dụng nữa trong quá trình khôi phục CSDL vì tất cả giao dịch trong phần này đã hoàn tất. Ngược lại, phần tích cực của transaction log chứa những giao dịch đang chạy và chưa hoàn thành.

Điểm kết thúc phần không tích cực của transaction log, điểm cắt, là điểm đầu tiên của những sự kiện sau:

- Checkpoint gần nhất tương ứng với điểm đầu tiên mà tại đó SQL Server sẽ roll forward những giao dịch trong quá trình khôi phục.

- Bắt đầu của giao dịch tích cực cũ nhất; 1 giao dịch chưa commit hoặc roll back. Tương ứng với điểm đầu tiên mà SQL Server roll back giao dịch trong suốt quá trình khôi phục.

Điều kiện transaction log backups

Transaction log không nên sao lưu:

- Nếu CSDL thiết lập **trunc. log on chkpt** (truncate log on checkpoint) là TRUE (thì không thể tạo ra log record dùng để roll forward); tạo database backup hoặc differential backup thay thế.

- Nếu bất kỳ thao tác nonlogged nào xảy ra trong CSDL kể từ khi thực hiện sao lưu full database backup lần cuối cùng; tạo full database backup hoặc differential backup thay thế.

- Cho đến khi thực hiện sao lưu full database backup vì transaction log backup chứa những thay đổi của database backup.

- Nếu transaction log bị cắt, trừ khi database backup hoặc differential backup được tạo ra sau khi cắt transaction log .

- Nếu bất kỳ file nào được thêm vào hay xóa khỏi CSDL; database backup nên tạo ra thay thế ngay lúc đó.

❖ **Thiết bị backup dữ liệu**

Hoạt động sao lưu có thể hướng đến thiết bị vật lý hoặc thiết bị logic.

+ Thiết bị vật lý là đĩa cứng, bằng từ….

+ Thiết bị logic chỉ tồn tại trong SQL Server và chỉ được dùng cho SQL Server thực hiện sao lưu. Để sao lưu tới thiết bị logic, bạn phải tạo thiết bị trước. Để tạo thiết bị sao lưu logic bạn có thể tạo bằng Enterprise Manager hoặc T-SQL.

Dùng Enterprise Manager

Để tạo thiết bị sao lưu bằng Enterprise Manager, thực hiện các bước sau:

B1: trong Enterprise, mở rộng server bạn muốn thực hiện sao lưu, giả sử chọn server cục bộ của máy bạn, mở rộng danh mục Management.

B2. Bấm chuột lên danh mục backup và chọn New Backup Device từ trình đơn tắt vừa xuất hiện để hiển thị cửa sổ Backup Device Properties.

B3: Nhập tên mô tả cho thiết bị sao lưu trong hộp Name, ví dụ thiếtbị_saoluu_1.Hộp File Name được điền tự động. Để thay đổi tên tập tin hoặc đường dẫn mới, bấm nút […] để mở hộp thoại Backup Device Location. Ví dụ như chọn thư mục C:\Backup,xem hình 9.2. Lưu ý là bạn phải tạo thư mục Backup ngoài hệ điều hành trước. Bấm OK để tạo thiết bị mới.

Hình 9.4 Cửa sổ Backup Device Properties

Dùng T-SQL trong Query Analyzer.

Để tạo thiết bị sao lưu bằng T-SQL phạn phải sử dụng thủ tục sp_addumpdevice. Cú pháp như sau:

Sp_addumpdevice device_type, logical_name, Physical_name

+device_type; kiểu thiết bị có thể là disk cho ổ đĩa cứng, tape cho ổ bằng từ..

+Logical_name:tên logic của thiết bị

+physical_name: tên tập tin.

Ví dụ: để tạo thiết bị trên đĩa cứng với tên thiết bị la thietbi_saoluu_2 trong thư mục C:\Backup như sau:

Exec sp_addumpdevice 'disk',

'thietbi_saoluu_2',

'C:\Backup\thietbi_saoluu_2.BAK'

Sau khi thiết bị được tạo nó sẵn sang để dùng

❖ **Thực hiện backup dữ liệu**

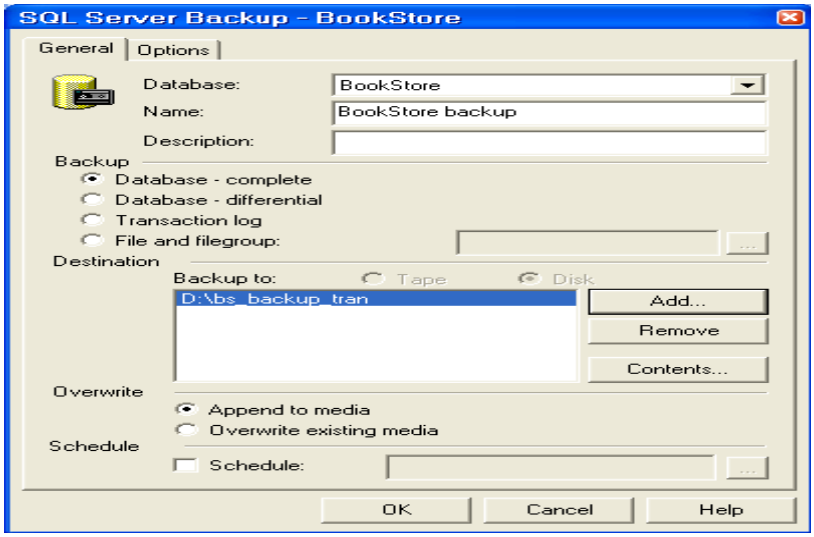
Bạn có thể thực hiện sao lưu CSDL bằng Enterprise Manager hoặc T-SQL. Phương pháp Enterprise Manager rất dễ dùng, tuy nhiên bạn nên sử dụng phương pháp nào phù hợp với bạn nhất.

Sao lưu dùng Enterprise Manager

Để thực hiện sao lưu bằng Enterprise Manager bạn thực hiện các bước sau:

B1. Kích vào server group, và kích vào server chứa Database muốn backup.

B2. Kích **Databases**, kích phải chuột vào database, trỏ chuột vào **All Tasks**, sau đó kích **Backup Database**.



Hình 9.5 Cửa sổ SQL Server Backup

B3. Trong hộp danh sách Database chọn CSDL bạn muốn sao lưu ví dụ dư BookStore. Trong **Name** box, tên sao lưu dựa vào tên CSDL và được điền tự động, cũng có thể nhập tên khác.. Trong **Description**, có thể soạn chú thích cho backup set này.

B4. Dưới mục Backup chỉ ra kiểu sao lưu.Những tùy chọn này có sẵn tùy theo CSDL.

+ **Database - Complete**: Thực hiện sao lưu đầy đủ CSDL.

+ **Database – Differential**: Thực hiện sao lưu phần thay đổi.

+ **Transaction Log**: thực hiện sao lưu tập tin log giao dịch.

+ **file and filegroup**: thực hiện sao lưu tập tin và nhóm tập tin.

Trong hình 9.3 ta chọn **Database - complete**.

B5. Dưới mục **Destination**, kích **Tape** hoặc **Disk (tuỳ thuộc bạn muốn backup vào loại thiết bị nào: tape cho ổ băng từ., disk cho ổ đĩa cứng).** Sau đó chỉ ra đường dẫn chứa tập tin backup ví dụ như là D:\bs_backup_tran. (Bấm Add hộp thoại Select Backup Destination xuất hiện)

Hình 9.6 Cửa sổ Select Backup Destination

Bấm kích Add hộp thoại Select Backup Destination xuất hiện. Chọn tùy chọn Backup Device, trong danh sách chọn thietbi_saoluu_1 chẳng hạn.. Bấm OK để trở về hộp thoại SQL Server Backup. Nút Remove để loại bỏ thiết bị khỏi Destination.

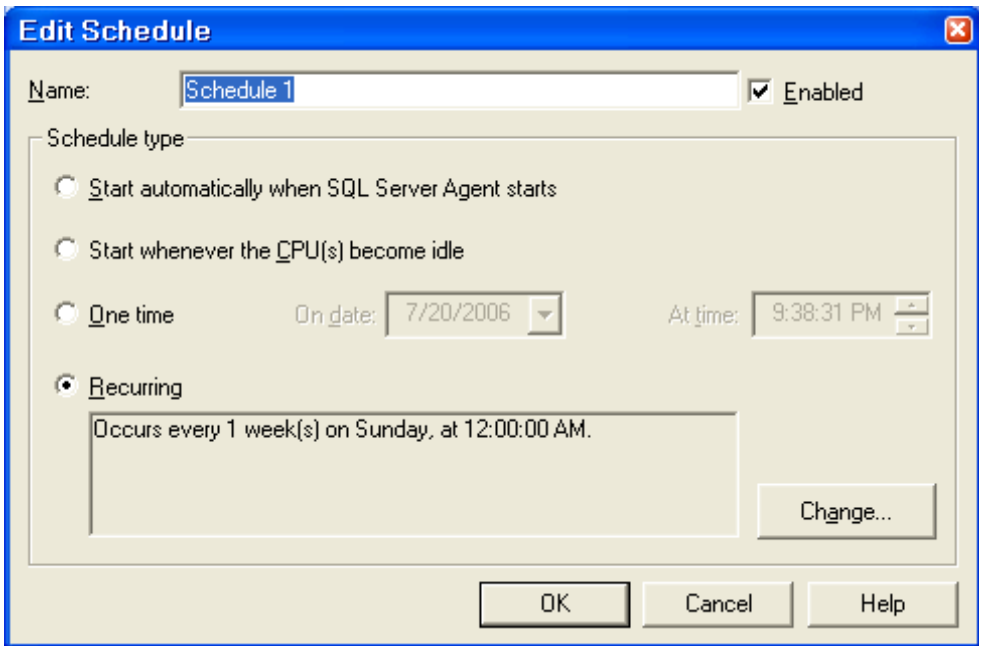
Lưu ý: Nếu bạn không muốn lưu vào thiết bị logic mà bạn muốn lưu trực tiếp thành một tập tin xuống hệ điều hành (thiết bị vật lý). Chọn tùy chọn File Name, bấm nút […], hộp thoại Backup Device Location xuất hiện, chọn thư mục bạn muốn sao lưu, trong hộp File Name: nhập tên tập tin sao lưu rồi bấm OK. Ví dụ chọn thư mục C:\Backup\và tên tập tin.

B6. Dưới Overwite, thực hiện như sau:

- **Kích Append to media** để thêm một tệp backup mới

- Kích **Overwrite existing media** để ghi đè lên tệp đang tồn tại.

B7. [Chức năng không bắt buộc] Chọn **Schedule** check box để xếp lịch cho việc backup (backup operation). Ví dụ bạn muốn tự động backup hàng tuần vào 12h00 ngày chủ nhật, hoặc… thì bạn có thể sử dụng chức năng này.



Hình 9.7 cửa sổ Edit Schedule

B8: Sau khi hoàn tất các thiết lập, bấm OK để thực hiện sao lưu.

(**Cách tạo transaction log backup bằng EM:** Cách làm tương tự như đối với tạo Full Database backup, tuy nhiên ở bước 4 ta phải chọn **Transaction log**.

Lưu ý: Nếu lựa chọn **Transaction Log** không được phép thì ta phải kiểm tra lại **recovery model** để thiết đặt là **Full** hoặc **Bulk-Logged**. Bởi vì chỉ những model này mới hỗ trợ **transaction log backup**.)

Sao lưu dùng T-SQL

Để sao lưu dùng T-SQL, bạn sử dụng lệnh Backup.

BACKUP DATABASE: dùng để sao lưu toàn bộ CSDL hoặc tập tin hay nhóm tập tin.

BACKUP LOG: dùng để sao lưu tập tin log giao dịch.

Ví dụ: Sao lưu cơ sở dữ liệu Northwind vào thietbi_saoluu_1 bạn thực hiện như sau:

BACKUP DATABASE Northwind to thietbi_saoluu_1

Lệnh trên sao lưu dữ liệu nối tiếp vào thiết bị, vì vậy kích thước tập tin thietbi_saoluu_1.BAK lớn dần lên mỗi khi bạn thực hiện lệnh trên.

Để sao lưu ghi đè, bạn dùng với tham số WITH INIT thực hiện như sau:

BACKUP DATABASE Northwind to thietbi_saoluu_1

WITH INIT

Ví dụ sao lưu CSDL MyDB trực tiếp vào đĩa (thiết bị vật lý) với tên tập tin là MyDB_020809 bạn thực hiện như sau:

BACKUP DATABASE MyDB

TO DISK = N'C:\Backup\MyDB_020809'

WITH INIT

Ví dụ: sao lưu tập tin log giao dịch của CSDL MyDB vào thietbi_saoluu_2 thực hiện như sau**:**

BACKUP LOG MyDB to thietbi_saoluu_2

9.3.2 Khôi phụ dữ liệu

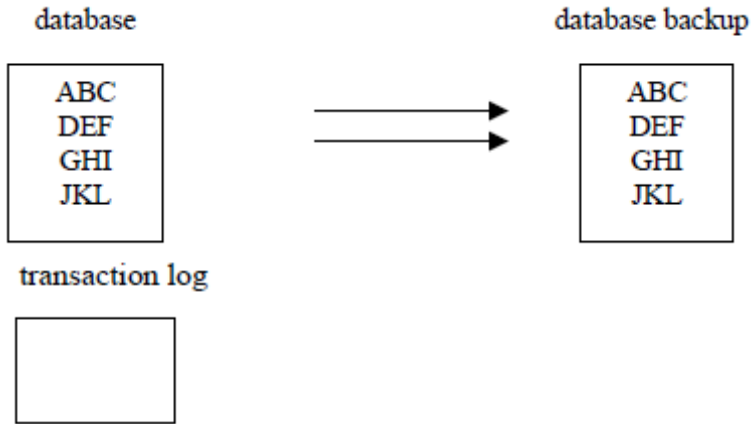
Trong khi khôi phụ, dữ liệu sao lưu được sao chép trở lại CSDL, không giống như quá trình sao lưu, quá trình khôi phục không thể được thực hiện trong khi SQL Server đang thực thi. Ngoài ra bảng không thể được khôi phục riêng biệt. Nếu một người dùng bị mất dữ liệu, dữ liệu bị mất không dễ dàng được khôi phục bởi vì hoạt động khôi phục sẽ khôi phục toàn bộ CSDL hoặc một phần của nó. Sự phân biệt dữ liệu của người dùng đơn với tất cả dữ liệu trong CSDL có thể rất khó.

Việc khôi phục một bản sao lưu CSDL sẽ trả về CSDL cùng trạng thái của CSDL khi ta thực hiện việc sao lưu. Giao dịch (transaction) nào không hoàn thành trong khi sao lưu (backup) CSDL được roll back để đảm bảo tính nhất quán CSDL.

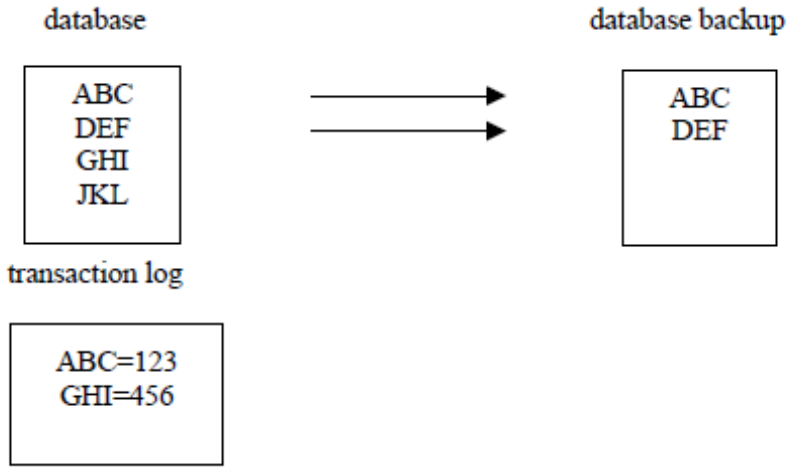
Khôi phục một bản sao lưu transaction log là áp dụng lại tất cả giao dịch (transaction) hoàn thành trong transaction log đối với CSDL. Khi áp dụng bản sao lưu transaction log, SQL Server đọc trước transaction log, roll forward tất cả các transaction . Khi đến cuối bản sao lưu transaction log, SQL Server roll back tất cả transaction mà không hoàn thành khi ta bắt đầu thực hiện sao lưu, tạo lại trạng thái chính xác của CSDL tại thời điểm bắt đầu thực hiện sao lưu.

Ví dụ minh họa sao lưu (backup) và khôi phục (restore) một CSDL có xảy ra giao dịch (transaction) khi thực hiện sao lưu:

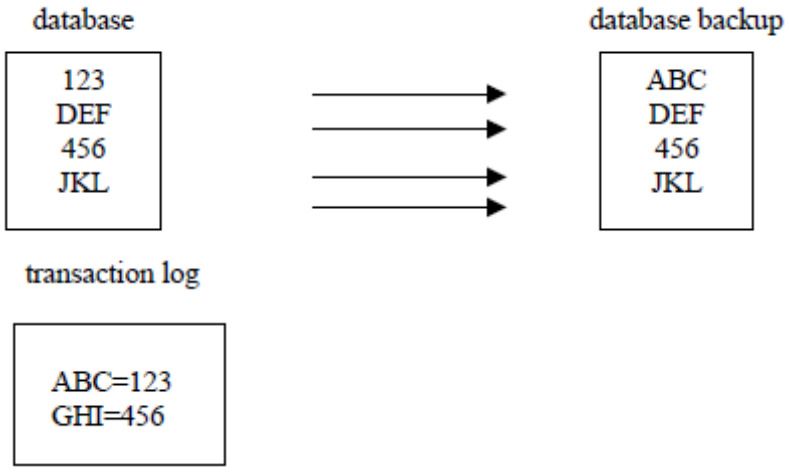
1. Bắt đầu backup: Giả sử CSDL gồm có các dữ liệu ABC, DEF, GHI, JKL, transaction log file không có dữ liệu vì không có giao dịch nào xảy ra. Khi đang thực hiện sao lưu (backup) được một phần dữ liệu thì xảy ra giao dịch, SQL Server 2000 sẽ ưu tiên cho việc giao dịch trước, việc sao lưu (backup) tạm thời dừng lại.



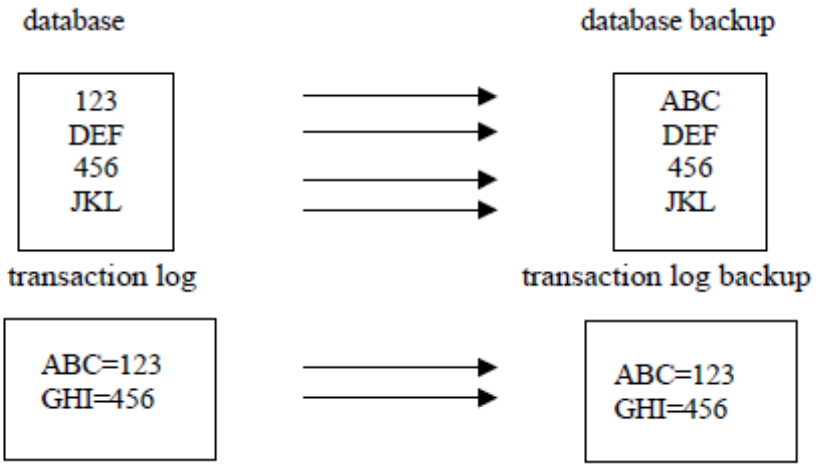
2. Xảy ra giao dịch (transaction), dữ liệu ABC được thay bằng 123, GHI được thay bằng 456.



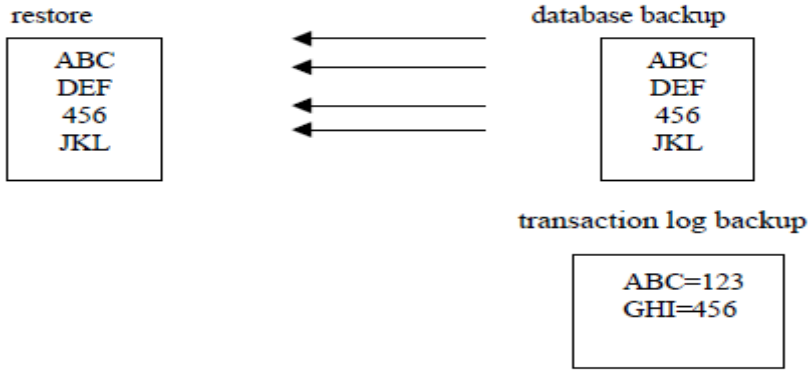
3. Khi thực hiện giao dịch (transaction) xong, SQL Server thực hiện tiếp việc sao lưu (backup), sẽ chép phần còn lại của dữ liệu nhưng dữ liệu đã thay đổi do xảy ra giao dịch.



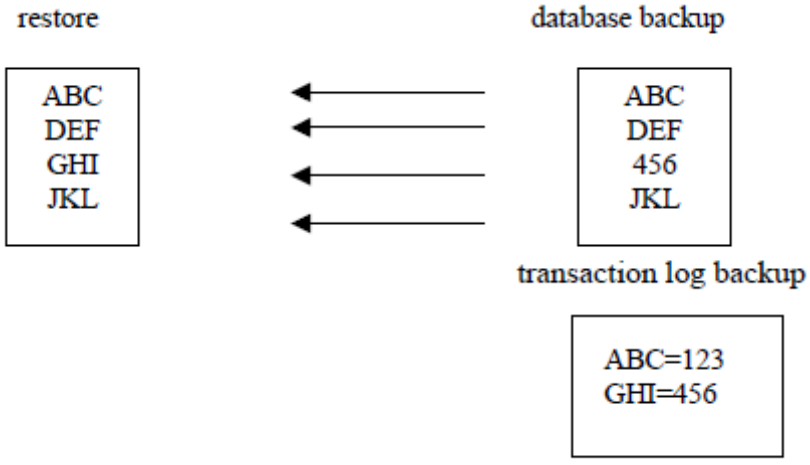
4. Khi sao lưu xong phần dữ liệu thì sẽ chép tiếp phần transaction log



5. Khi có yêu cầu khôi phục (restore) CSDL , CSDL được khôi phục trước, chép lại toàn bộ CSDL của bản sao lưu CSDL đó.



6. Sau đó SQL Server sẽ khôi phục tiếp phần transaction log. Trước tiên sẽ roll forward nhưng khi đọc đến dữ liệu thứ ba thì nó thấy dữ liệu này đã được thay đổi rồi do đó nó sẽ roll back (trả về dữ liệu ban đầu khi chưa thực hiện giao dịch) để nhất quán dữ liệu.



❖ **Các phương pháp khôi phục dữ liệu**

Loại sao lưu dữ liệu được thực hiện ảnh hưởng đến cách khôi phục.

+ Khôi phục từ sao lưu đầy đủ:khôi phục từ sao lưu đầy đủ là một quá trình khá đơn giản: bạn chỉ việc khôi phục các tập tin sao lưu bằng Enterprise hoặc T-SQL.

+ Khôi phục từ sao lưu những thay đổi: Để khôi phục từ sao lưu những thay đổi, trước hết bạn phải khôi phục từ sao lưu đầy đủ, sau đó khôi phục tất cả những sao lưu những thay đổi đã được tạo từ lần sao lưu đầy đủ gần nhất. Nên nhớ sao lưu những thay đổi được dùng để sao lưu dữ liệu thay đổi từ lần sao lưu đầy đủ gần nhất hoặc sao lưu những thay đổi gần nhất. Phải dùng tùy chọn NORECOVERY trừ phi bạn đang thực hiện khôi phục từ sao lưu đầy đủ gần nhất. Nếu bạn đang khôi phục từ tập tin log giao dịch tiếp tục từ khi sao lưu những thay đổi, bạn cũng phải sao lưu tập tin log hiện thời và áp dụng tất cả những tập tin log thay đổi.

+ Khôi phục từ sao lưu tập tin log giao dịch

Để thực hiện phụ hồi CSDL đưa về trạng thái trước khi bị hư hỏng, trước hết bạn phải khôi phục các tập tin dữ liệu từ sao lưu đầy đủ mới nhất và sau đó khôi phục từ sao lưu những thay đổi từ lần sao lưu đầy đủ đó. Bạn sao lưu những thay đổi này bằng cách khôi phục tất cả những sao lưu tập tin log giao dịch đã xảy ra từ trước khi hư hỏng hệ thống.

Để đảm bảo bạn không bị mất bất kỳ giao dịch mới nhất nào khi bạn khôi phục tập tin log, bạn phải lưu log hiện thời. Nếu bạn quên sao lưu log hiện thời, bạn sẽ mất những thay đổi mới nhất đã được ghi trong log bởi vì hoạt động khôi phục sẽ ghi đè tập tin log giao dịch.

Để sử dụng tập tin log giao dịch khôi phục CSDL về trạng thái trước khi nó xảy ra hư hỏng, thực hiện các bước sau:

1. Sao lưu tập tin log giao dịch hiện đang hoạt động, sử dụng tùy chọn NO_TRUNCATE.

2. Khôi phục sao lưu đầy đủ mới nhất

3. Khôi phục từ bất kỳ sao lưu những thay đổi nào để đưa CSDL về trạng thái mà khi sao lưu mới nhất được thực hiện.

4. Khôi phục tất cả tập tin log giao dịch từ sao lưu những thay đổi mới nhất để thực hiện lại bất kỳ giao dịch đã xảy ra từ lần sao lưu gần nhất.

5. Khôi phục sao lưu log giao dịch đã thực hiện ở bước 1 để đưa CSDL về trạng thái trước khi xảy ra hư hỏng

❖ **Thực hiện khôi phục dữ liệu**

Bạn có thể khôi phục dữ liệu bằng cách sử dụng Enterprise Manager hoặc R-SQL, cả 2 phương pháp đều cho cùng kết quả.

Dùng Enterprise Manager

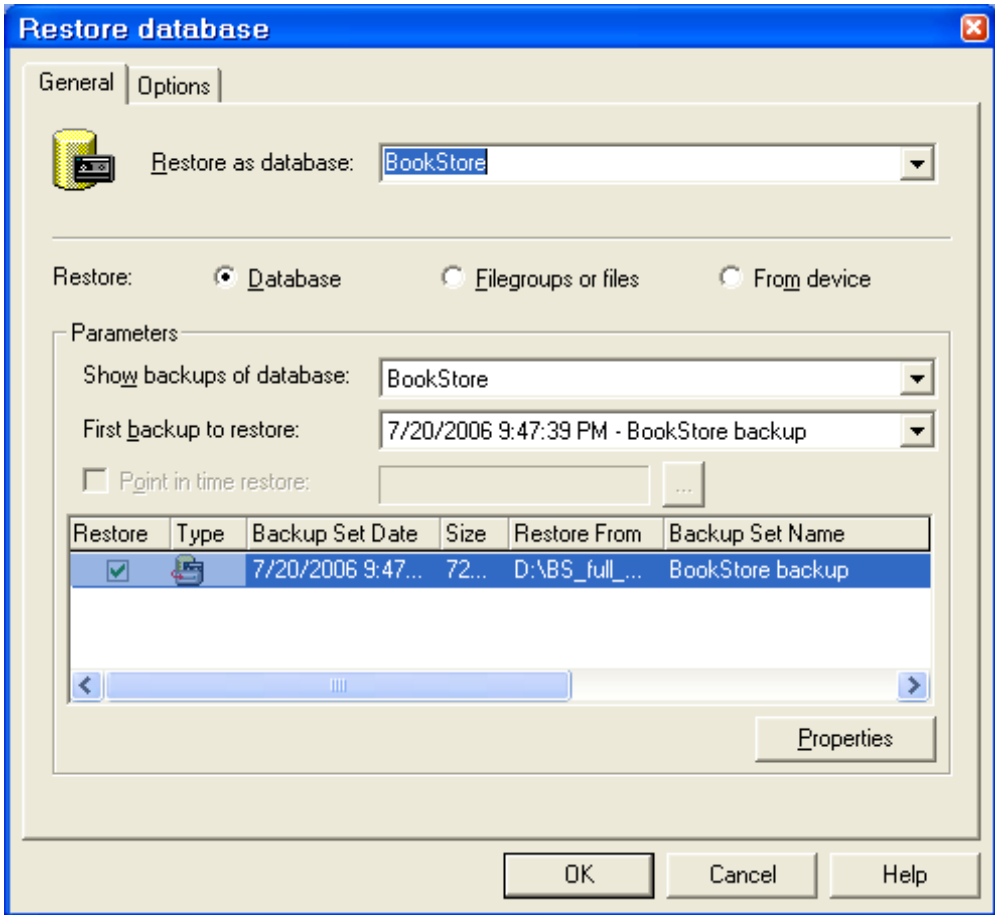
Để khôi phục dữ liệu bằng Enterprise Manager bạn thực hiện các bước sau:

Khôi phục Full database backup bằng EM

Thực hiện các bước sau:

1. Kích vào server group, và kích vào server chứa Database muốn backup.

2. Kích **Databases**, kích phải chuột vào database, trỏ chuột vào **All Tasks**, sau đó kích **Restore Database**.



Hình 9.8 Cửa sổ Restore database

3. Trong **Restore as database** box, soạn thảo hoặc chọn tên database nếu muốn thay đổi tên Database mặc định không.

4. Kích **Database**.

5. Trong danh sách **First backup to restore**, kích vào bản sao lưu muốn được phục hồi(vì có thể có nhiều bản đã được Backup).

6. Trong danh sách **Restore**, kích vào database muốn được phục hồi.

Mô tả quá trình thực hiện:

Khôi phục lại database backup là trả về trạng thái của CSDL khi lệnh backup được thực thi. SQL Server tạo lại CSDL theo các bước sau:

o Chép tất cả dữ liệu trong bản sao vào CSDL khôi phục.

o Bất kỳ giao dịch nào không hoàn thành trong database backup thì được roll back để bảo đảm tính nhất quán dữ liệu.

Quá trình này bảo đảm CSDL sau khi khôi phục là một bản sao của CSDL khi thực hiện sao lưu, trừ những giao dịch không hoàn thành được roll back. Điều này đảm bảo tính toàn vẹn dữ liệu.

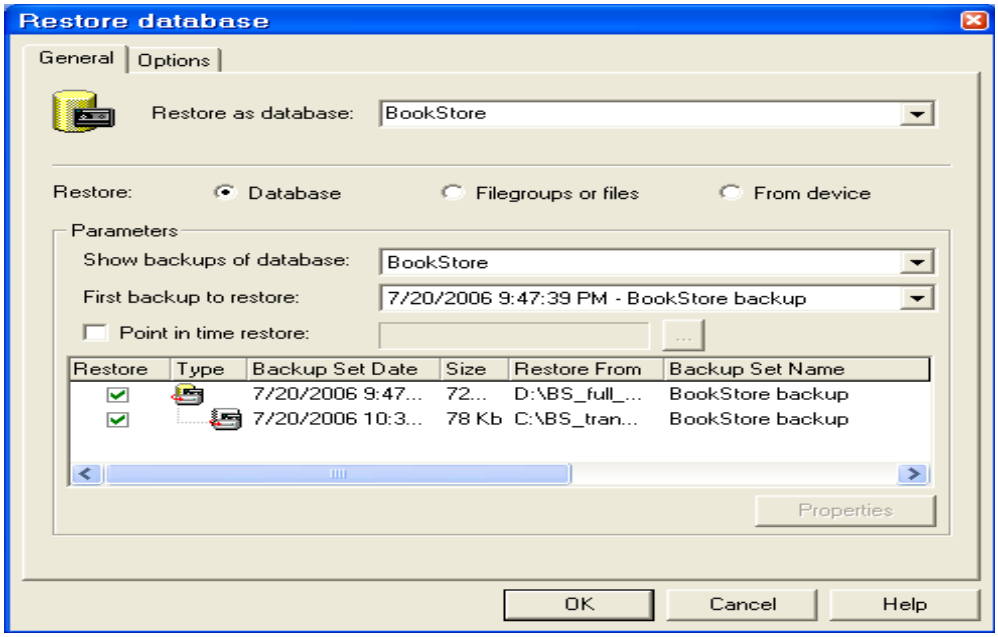
Ngoài ra, để tránh việc cố tình viết đè lên CSDL đã tồn tại, quá trình khôi phục thực hiện kiểm tra an toàn một cách tự động. Quá trình khôi phục không thực hiện nếu:

- Tên CSDL khôi phục đã tồn tại trên server và tên CSDL cần khôi phục không tương ứng với tên CSDL ghi trong backup set.
- Tên CSDL khôi phục đã tồn tại trên server nhưng dữ liệu bên trong không giống với dữ liệu bản sao database backup. Ví dụ: CSDL khôi phục có cùng tên với CSDL đã có trong SQL Server nhưng dữ liệu thì khác ví dụ như có những bảng dữ liệu khác.
- Một hoặc nhiều file yêu cầu tạo tự động bằng thao tác khôi phục (không để ý đến CSDL đó tồn tại hay chưa) nhưng những file này có cùng tên với CSDL đã tồn tại rồi.

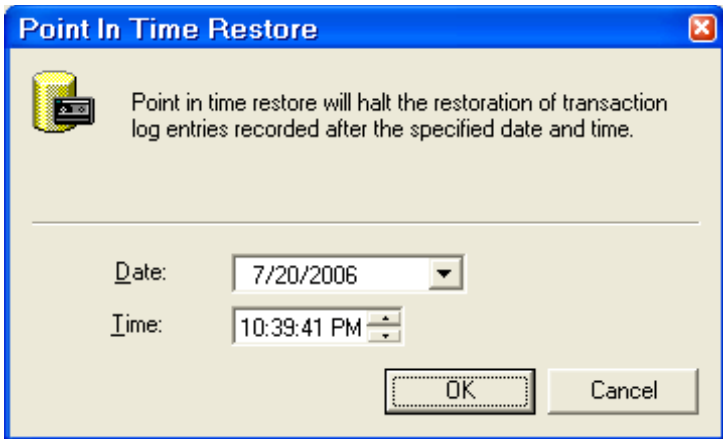
Tuy nhiên việc kiểm tra an toàn có thể không có tác dụng nếu có mục đích viết đè.

Khôi phục transaction log backup bằng EM

Thực hiện tương tự như cách **Khôi phục Full database backup.**



Tuy nhiên, ta có thể xác định được thời điểm nào đó muốn khôi phục dữ liệu trong quá khứ.



Khôi phục file hoặc filegroup backup:

File hoặc file group có thể được khôi phục từ database backup hoặc file hoặc file group. Ta không cần thiết backup transaction log nếu không có thay đổi từ khi sao lưu file hoặc file group.

Ví dụ: Nếu filegroup_b cần khôi phục vì 1 bảng trong filegroup bị hư, ta sẽ:

- Khôi phục filegroup_b backup tạo vào ngày thứ năm.
- Áp dụng transaction log backup của filegroup_b.

Dùng T-SQL

Để khôi phục CSDL dùng T-SQL, bạn dùng lệnh RESTORE, tương tự như lệnh BACKUP. Nó có thể khó dùng lúc đầu nhưng bạn có thể đưa các thủ tục quản trị vào các script để có thể dùng lại nhiều lần sau đó.

Ví dụ để khôi phục CSDL Northwind từ thietbi_saoluu_1 bạn đã thực hiện ở trên, bạn thực hiện như sau:

RESTORE DATABASE Northwind

FROM thietbi_saoluu_1

Nếu bạn khôi phục CSDL đang tồn tại, bạn có thể dùng tùy tồn WITH REPLACE như sau:

RESTORE DATABASE Northwind

FROM thietbi_saoluu_1

WITH REPLACE

Ví dụ để khôi phục CSDL MyDB từ tập tin sao lưu MyDB_010809 bạn thực hiện như sau:

RESTORE DATABASE MyDB

FROM DISK = 'C:\Backup\MyDB_020809'

Kết chương

Trong chương này bạn đã hiểu về các thuật ngữ sao lưu và khôi phục. Bạn cũng đã học phương pháp thực hiện sao lưu và phục hồi CSDL của SQL Server. Bạn biết được sự khác nhau giữa sao lưu đầy đủ và sao lưu chỉ những thay đổi của dữ liệu. Cách sao lưu CSDL và tập tin log giao dịch. Trong quá trình đó bạn cũng đã học cách lập biểu thời gian cho sao lưu tự động. Ngoài ra bạn biết cách khôi phục CSDL từ tập tin sao lưu hoặc thiết bị sao lưu bằng cả hai phương pháp Enterprise Manager và T-SQL. Bạn cũng biết cách khôi phục lại CSDL với tên mới hoặc ghi đè lên CSDL đang tồn tại.