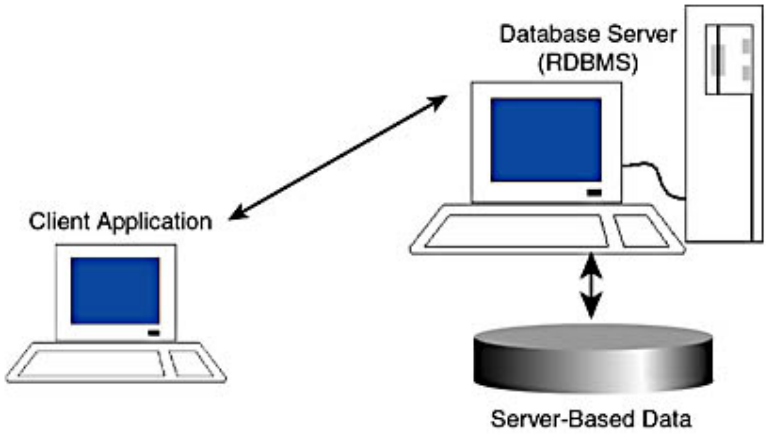
CHƯƠNG 1: TỔNG QUAN VỀ SQL SERVER VÀ CSDL QUAN HỆ

I- GIỚI THIỆU SQL SERVER

SQL là một hệ thống quản trị CSDLQH (RDBMS) nhiều người dùng kiểu Client/Server. Đây là hệ thống cơ bản dùng lưu trữ dữ liệu cho hầu hết các ứng dụng lớn hiện nay.

- **Ứng dụng kiểu Client/Server:**

Một ứng dụng kiểu Client/Server bao gồm 2 phần: Một phần chạy trên Server(máy chủ) và phần khác chạy trên các workstations (máy trạm).



Phần Server: chứa các CSDL, cung cấp các chức năng phục vụ cho việc tổ chức và quản lý CSDL, cho phép nhiều người sử dụng cùng truy cập dữ liệu. Điều này không chỉ tiết kiệm mà còn thể hiện tính nhất quán về mặt dữ liệu. Tất cả dữ liệu đều được truy xuất thông qua server, không được truy xuất trực tiếp. Do đó, có độ bảo mật cao, tính năng chịu lỗi, chạy đồng thời, sao lưu dự phòng…

Phần Client (Ứng dụng khách): Là các phần mềm chạy trên máy trạm cho phép người sử dụng giao tiếp CSDL trên Server.

SQL Server sử dụng ngôn ngữ lập trình và truy vấn CSDL **Transact-SQL**, một version của Structured Query Language. Với Transact-SQL, bạn có thể truy xuất dữ liệu, cập nhật và quản lý hệ thống CSDL quan hệ.

Với mỗi Máy chủ bạn chỉ có một hệ thống QTCSDL SQL Server. Nếu muốn có nhiều hệ thống QTCSDL bạn cần có nhiều máy chủ tương ứng.

- **Các phiên bản SQL Server 2000 :**

a) Phiên bản chuẩn (Standard Edition): là phiên bản cung cấp toàn bộ chức năng và được thiết kế nhằm chạy trên máy tính với HĐH Windows NT hoặc Windows 2000 Server.

b) Phiên bản Personal Engine: Chạy trên máy đơn với HĐH Windows NT; Windows 2000 Server và cả Windows 9x. Nó hỗ trợ hầu hết các tính năng của SQL Server. Do đó, bạn có thể xây dựng CSDL với phiên bản này, sau đó triển khai trên các phiên bản khác.

Ngoài ra còn các phiên bản khác chạy trên Window NT hoặc 2000 Server như :

Enterprise Edition: dùng cho xí nghiệp

Developer Edition : dùng cho các doanh nghiệp nhỏ

Desktop Engine

II- Các Thành Phần Của Một CSDL Trong SQL Server :

Mỗi CSDL có các đối tượng sau:

1- Tables :

Table là đối tượng chính của CSDL dùng lưu trữ dữ liệu cần quản lý. Mỗi table có 1 hay nhiều Field. Mỗi Field ứng với một loại dữ liệu cần lưu trữ.

Table còn có các thành phần liên quan như :

a) Constraint – Ràng buột:

Constraint là các chỉ định ràng buột dữ liệu trong bảng hoặc các bảng khác nhau theo một quy tắc nào đó.

b) Triggers – Bẫy Lỗi:

Trigger thường chứa các mã lệnh kiểm tra dữ liệu, có tính năng tự động thực hiện khi có hành động nào đó xảy ra đối với dữ liệu trong Table như Insert, Update, Delete.

c) Indexs – Chỉ mục :

Hổ trợ cho việc sắp xếp và tìm kiếm nhanh thông tin trên table.

2- Diagram – Sơ đồ quan hệ:

Thể hiện mối quan hệ dữ liệu giữa các table.

3- Views – Khung nhìn hay table ảo:

Là đối tượng dùng hiển thị dữ liệu được rút trích, tính toán từ các Table theo nhu cầu của người dùng.

4- Stored Procedure – Thủ tục nội:

Chứa các lệnh T-SQL dùng thực hiện một số tác vụ nào đó. Stored Proc có thể nhận và truyền tham số. Stored Proc được biên dịch trước, do đó thời gian thực hiện nhanh khi được gọi. Có nhiều Stored Proc hệ thống được định nghĩa với tiền tố "sp_" có nhiệm vụ thu thập thông tin từ các bảng hệ thống và rất có ích cho việc quản trị.

5- User Defined Function :

Hàm do người dùng định nghĩa

6- Users :

Chứa danh sách User sử dụng CSDL. Người quản trị hệ thống cao nhất có User Name là **dbo**, tên đăng nhập (Login Name) hệ thống mặc định là **sa.** Tài khoản **sa** luôn tồn tại và không thể bỏ đi. Để thay đổi mật khẩu của **sa**, cách nhanh nhất là:

Mở trình Query Analyzer

Thực hiện thủ tục hệ thống : EXEC **SP_PASSWORD** NULL, <NewPass>

7- Roles :

Các qui định vai trò và chức năng của User trong hệ thống SQL Server

8- Rules :

Các qui tắc ràng buộc dữ liệu được lưu trữ trên Table

9- Defaults :

Các khai báo giá trị mặc định

10-User Defined Data Type :

Kiểu dữ liệu do người dùng tự định nghĩa

11-Full Text Catalogs :

Tập phân loại dữ liệu Text.

III- CÁC CSDL HỆ THỐNG CỦA SQL SERVER:

Sau khi cài đặt, SQL Server có 4 CSDL hệ thống và 2 CSDL ví dụ sau:

1- **Master:** là CSDL kiểm soát tất cả các hoạt động trên SQL Server, chứa thông tin về hệ thống SQL Server : Các tài khoản đăng nhập, cấu hình hệ thống, thông tin về các CSDL đã tạo, các thủ tục hệ thống thực hiện các tác vụ quản trị hệ thống, các thủ tục của người dùng tạo thêm…

Ví dụ: khi User tạo CSDL mới, thêm hay xóa Stored Procedure, các thông tin này đều được lưu trữ trong CSDL master.

Chú ý: Cần Backup CSDL Master mỗi khi bạn sửa đổi hệ thống.

2- **Model :** Chứa các template dùng làm mẫu để tạo CSDL mới. Khi bạn tạo CSDL thì SQL Server lấy tất cả các mẫu (bao gồm Tables, Views,…) từ CSDL này.

3- **MSDB:** dùng hổ trợ dịch vụ SQL Server Agent, bao gồm sắp xếp thông tin về các công việc theo lịch biểu, các cảnh báo lỗi, các sự kiện, nhân bản. Lịch sử về các hoạt động Backup đều được lưu trong CSDL này.

Ví dụ: Khi bạn tạo một lịch trình cho việc backup dữ liệu hay lịch trình để thực hiện Stored Procedure, tất cả các tác vụ này đều lưu trong CSDL này.

Nếu xóa CSDL này, bạn phải cài đặt lại nó khi cần dùng hoặc khi hệ thống yêu cầu.

4- **Tempdb:** là nơi lưu trữ các thông tin tạm thời của các hoạt động trên SQL, ví dụ như các table tạm phục vụ cho việc sắp xếp dữ liệu. CSDL tempdb tự khởi tạo lại mỗi khi SQL Server được khởi động lại.

5- **Pubs:** là CSDL mẫu về một nhà xuất bản, bao gồm các tác giả, các cuốn sách, và thông tin về việc bán sách. Hấu hết các tính năng CSDL đều được thể hiện trong CSDL này.

6- **NorthWind:** Là CSDL hổ trợ cho việc học tập SQL Server đối với những người sử dụng MS Access

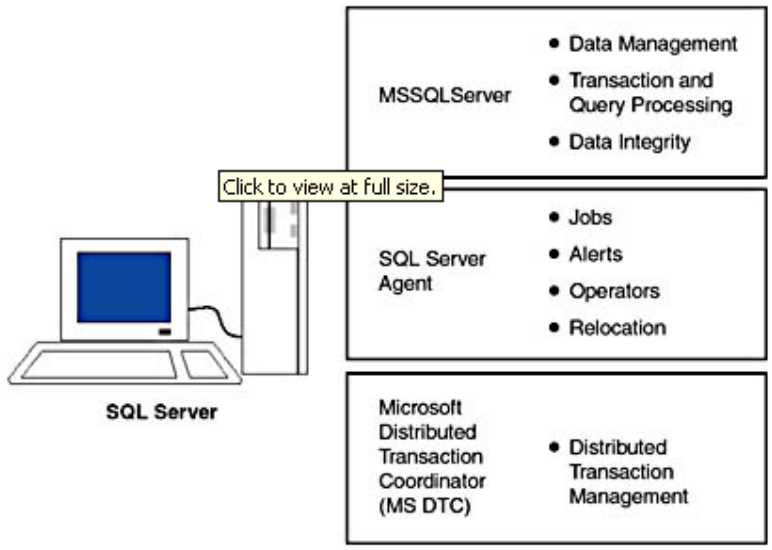
IV- Các Công Cụ Của SQL Server:

SS chứa các công cụ hổ trợ cho việc quản lý và truy cập CSDL :

1- Service Manager: Các dịch vụ của SQL Server:

Đây là trình quản lý các dịch vụ trên SQL Server như: MSSQL Server, SQL Server Agent, Microsoft Distrinuted Transaction Coordinator (MS DTC).

- MSSQL Server : là RDBMS, xử lý các phát biểu Transact-SQLvà quản lý các file lưu trữ các CSDL
- SQL Server Agent : dùng lập lịch thực hiện tự động các công việc như Backup dữ liệu, Replication, …
- **Microsoft Distributed Transaction Coordinator (**MSDTC): Hổ trợ cho các ứng dụng Client làm việc với các dữ liệu được phân bổ trên nhiều máy chủ.



- **Combo Server:** Chứa danh sách các SQL Server có khả năng truy xuất từ máy của bạn. Tuy nhiên, bạn có thể gỏ tên máy chủ cần kết nối không có trong danh sách và click nút mũi tên trên hộp Services, Tiện ích Services Manager sẽ cố gắng kết nối tới máy chủ ở xa.
- **Combo Services:** Chứa tên các dịch vụ được cung cấp bởi SQL Server đã chọn.

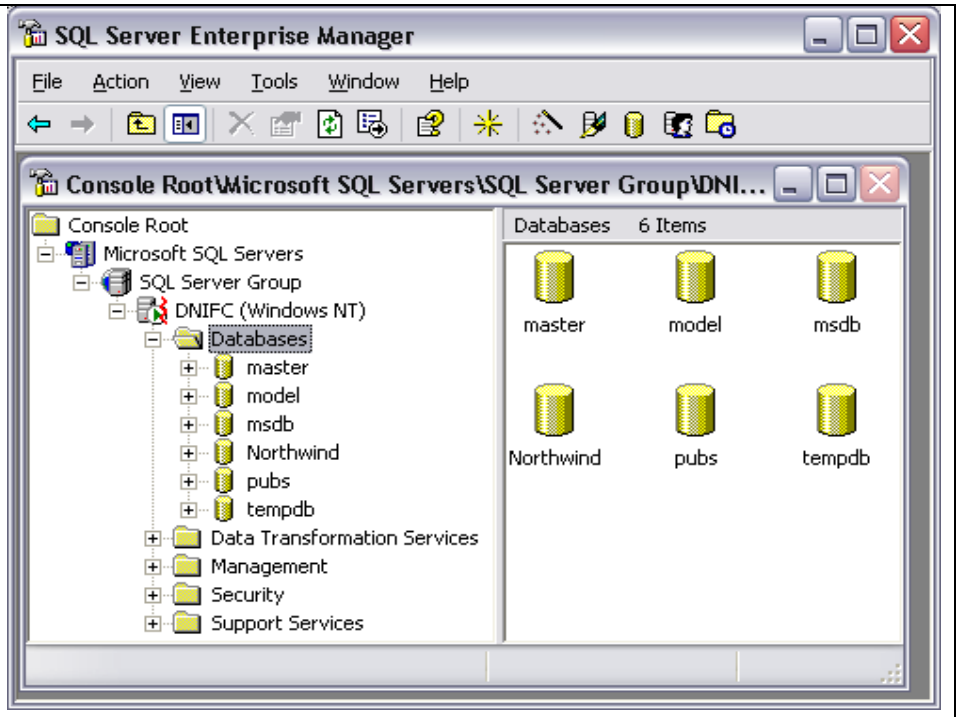
Mỗi dịch vụ sẽ ở 1 trong 3 trạng thái. Một số dịch vụ không cung cấp chức năng tạm dừng.

Ví dụ MSSQLServer vẫn tiếp tục hoạt động khi nó tạm dừng, nhưng các kết nối mới thì không được phép.

2- Trình Enterprise Manager:

Enterprise Manager cung cấp các chức năng phát triển và quản trị SS bằng giao diện đồ họa. Các tính năng của EM:

- Tạo, xóa, cập nhật CSDL và các đối tượng của nó.
- Quản lý lịch trình Backup dữ liệu
- Quản lý người dùng đang truy cập CSDL trên SS.
- Tạo, xóa cập nhật quyền Login User
- Định cấu hình cho Server
- Tạo và quản lý tìm kiếm.

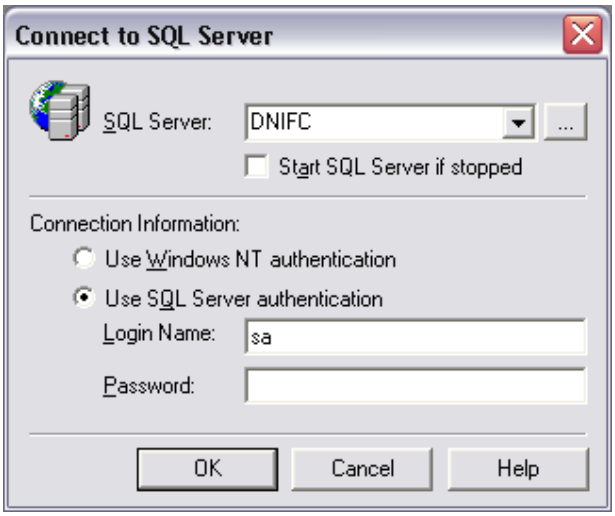


- **Databases** : chứa các CSDL được cài đặt trên máy chủ.
- **Data Transform Service (DTS)** : Cung cấp phương tiện chuyển đổi dữ liệu từ nhiều nguồn khác nhau vào SQL Server hoặc từ SQL Server sang các nguồn khác. Nó cũng có giao diện cho phép lập trình xây dựng những gói dịch vụ chuyển đổi dữ liệu.
- **Management**: Cho phép bạn thực hiện các tác vụ quản trị CSDL, xem nhật ký hoạt động của Server, quản lý SQL Server Agent.
- **Replication:** cho phép phân bổ dữ liệu và các đối tượng Database từ một CSDL đến một database đến một database khác. Typically, replication is used between physically distributed servers.
- **Security** : chứa các chức năng điều khiển tất cả các hoạt động liên quan đến việc bảo mật của SQL Server.
- **Support Services**: cho phép điều khiển các dịch vụ khác có quan hệ với SQL Server như : Distributed Transaction Coordinator, Full Text Search và SQL Mail cho phép nhận e-mail từ SQL Server.
- **Metadata Services** enables you to manage metadata, or data about data.

3- Công Cụ Lập Trình - Query Analyzer (ISQLW):

Là giao diện chính để chạy các truy vấn Transact-SQL hoặc thủ tục lưu trữ.

Khi khởi động Query Analyzer sẽ xuất hiện hộp thoại Connection to SQL Server:

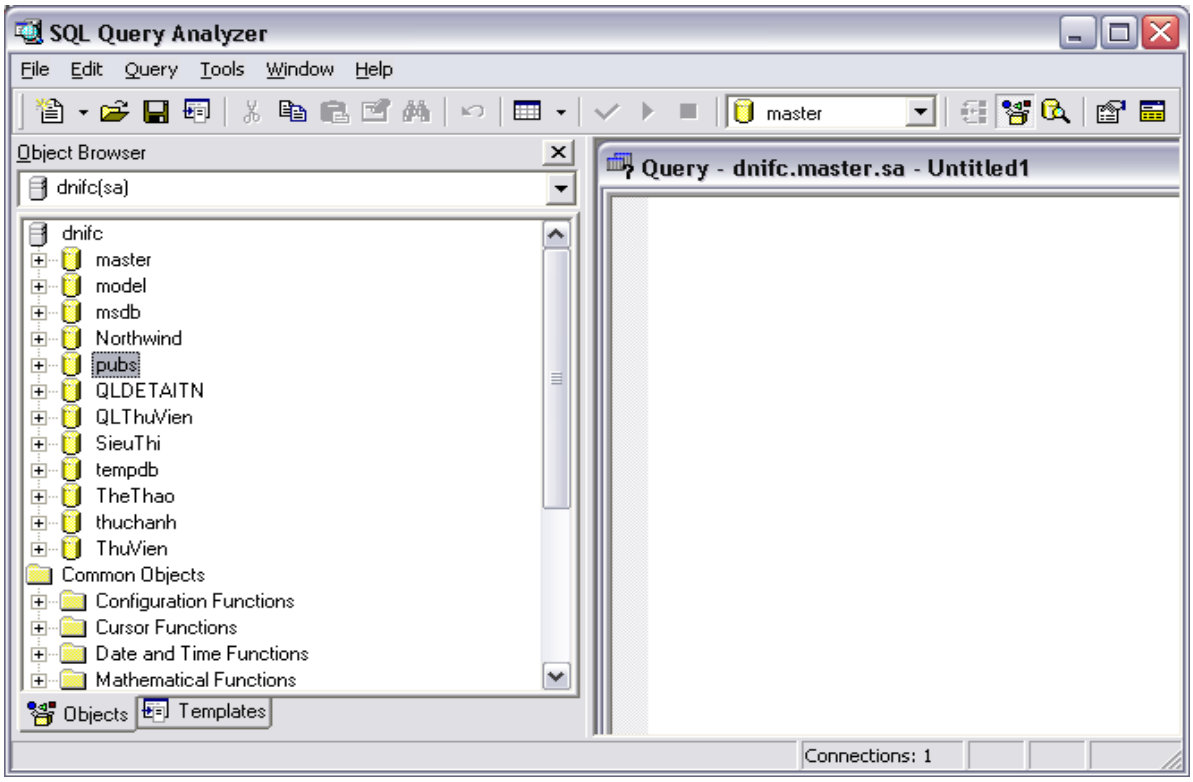


- Combo SQL Server: chọn tên máy chủ chứa SQL Server hoặc bạn có thể nhập "(local)" hay dấu "." để kết nối với bản SQL Server cục bộ. Bạn cũng có thể để trống tên máy chủ, Query Analyzer sẽ hiểu và kết nối với SQL Server của bạn.
- Check Box "Start SQL Server if stopped" để yêu cầu SQL Server tự khởi động, nếu SQL Server chưa được khởi động.

Bạn cần cung cấp thông tin kết nối:

- Nếu đang làm việc trên máy Windows 9x, bạn chỉ có thể dùng tùy chọn "Use SQL Server authentication" với Login Name là **sa.**
- Nếu đang làm việc trên máy Windows NT hay đang kết nối với SQL Server chạy trên Windows NT, bạn có thể chọn Option "Use Windows NT authentication" hay "Use SQL Server authentication" để kết nối tùy thuộc vào cách cài đặt chế độc bảo mật của người quản trị SQL Server.

Nếu kết nối SS thành công, sẽ hiển thị màn hình làm việc của QA.



QA cho phép thực hiện 32 kết nối riêng rẽ cùng một lúc. Mỗi kết nối có một thanh tiêu đề nhận dạng các yếu tố sau:

- Máy tính được đăng nhập
- CSDL đang sử dụng
- Thông tin đăng nhập
- Tên File truy vấn đang mở
- Số của cửa sổ được hiển thị

Query Analyzer Icons

Use this	To do this
	Mở cửa sổ truy vấn mới
	Mở một file truy vấn (.sql)
	Lưu các lệnh vào file.
	Mở một file truy vấn mẫu
	Xóa nội dung cửa sổ
	Cách hiển thị kết quả truy vấn: Result to text, result to grid, result to file.
	Kiểm tra cú pháp.
	Thực hiện truy vấn.
	Kết thúc truy vấn.
	Hiển thị sơ đồ đánh giá tốc độ thực hiện truy vấn.
	Hiện ẩn khung liệt kê các đối tượng CSDL (Objects Browser)
	Mở hộp thoại khai báo thuộc tính kết nối.
	Hiện ẩn khung chứa kết quả truy vấn

Bạn có thể chọn CSDL cần truy cập từ DB Conmbo box trên thanh công cụ hoặc có thể sử dụng lệnh: Use <database name> .

- Transact-SQL Script là tập hợp các lệnh được lưu trữ và thi hành cùng lúc. Lệnh File\Open và File\Save cho phép mở và lưu một truy vấn (hoặc tập hợp các truy vấn). Theo mặc định, các Sript có phần mở rộng là ".SQL".
- HighLight từ khóa bằng chuột và ấn Shift-F1 để mở phần trợ giúp liên quan.
- Bạn có thể chạy một lệnh trong cửa sổ Query bằng cách Highlight câu lệnh bằng chuột và sau đó cho thi hành.

4- Tiện ích mạng Client / Server Network:

Cung cấp các thư viện nghi thức kết nối mạng (Netword-Libraries) cho phép các máy trạm có thể truy cập CSDL trên máy Server: *Named Pipes; TCP/IP; Multiprotocol; NW Link IPX/SPX*

5- Books Online:

Sách hướng dẫn trực tuyến được lưu dưới dạng HTML đã được biên dịch, nên có thể xem chúng bằng các trình duyệt Web.

V- Tính Bảo Mật Trên SQL Server :

SQL Server kiểm tra User ở 2 mức :

- Mức đăng nhập vào SQL Server
- Mức sử dụng các đối tượng trên SQL Server.

1- Mức đăng nhập vào SQL Server :

Để kết nối với SS, người sử dụng phải có một tài khoản đăng nhập(Login Account) được cung cấp bởi người quản trị hệ thống.

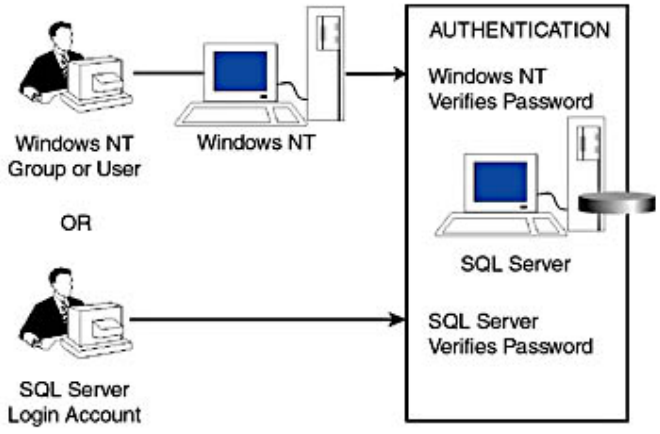
Khi SQL Server chạy trên Windows NT, Người Quản trị hệ thống có thể chỉ định nó chạy ở 1 trong 2 chế độ xác nhận :

a- Chế độ xác nhận Windows NT (Windows NT Authentication Mode):

Khi đó User chỉ cần sử dụng tài khoản đăng nhập của Windows NT truy cập vào mạng là có thể kết nối tới SS và các tài nguyên khác trên mạng. Người quản trị hệ thống phải gán quyền truy xuất SQL Server cho mỗi tài khoản người dùng hoặc tài khoản nhóm người dùng trên mạng. SQL Server cài đặt trên Windows 9x không hỗ trợ chế độ này.

b- Chế độ hỗn hợp (Mixed Mode):

Một số User có thể kết nối với SQL Server với xác nhận Windows NT. Một số User khác chỉ sử dụng xác nhận của SQL Server (SQL Server Authentication) dựa trên Login Name và Password do người quản trị CSDL cấp.



2- Quyền thao tác trên SQL Server:

Tùy theo yêu cầu, mỗi người dùng có thể được gán hoặc không gán các quyền như :

Quyền sử dụng các ứng dụng CSDL trong SQL Server

Quyền tạo và sửa đổi cấu trúc các đối tượng trong SQL Server

Quyền truy cập và xử lý dữ liệu.

Chương 2 : Các Phát Biểu Cơ Bản Của Transact-SQL

I- GIỚI THIỆU NGÔN NGỮ TRANSACT-SQL :

T-SQL is ngôn ngữ thủ tục thế hệ thứ 3. Không giống như những NNLT khác, bạn không thể dùng nó để tạo ra các chương trình ứng dụng độc lập. Các phát biểu của nó chỉ được thực hiện trong môi trường SQL Server với mục đích truy vấn và hiệu chỉnh dữ liệu trong CSDL quan hệ.

T-SQL có các phát biểu được phân loại như sau :

- **Data Control Language (DCL):** Chứa các lệnh điều khiển, phân quyền truy xuất dữ liệu.
- **Data Definition Language (DDL):** Dùng tạo, sửa xóa các đối tượng trong CSDL – như Database, table, Index, Default, Procedure, Function, Schema, View, và Trigger,
- **Data Manipulation Language (DML)**: Chứa các lệnh thêm, sửa, xoá dữ liệu
- **Data Query Language (DQL) :** Chỉ chứa 1 phát biểu SELECT dùng truy vấn dữ liệu
- Các thành phần khác của ngôn ngữ như kiểu dữ liệu, biến, toán tử, hàm, các cấu trúc điều khiển và chú thích.

CSDL sử dụng trong chương này:

a. KHUVUC(MaKV, TenKV, MaNVQL)
b. NHANVIEN(MaNV, HoTenNV, Phai, LuongCB, CongViec, #MaKV)
c. LOAIHANG(MaLH, TenLH)
d. MATHANG(MaMH, TenMH, DVT, DonGia, SoTon, MaLH)
e. HOADON(SoHD, NgayHD, MaNV)
f. CTHD(MaHD, MaMH, SL, DGBan)

II- Kiểu dữ liệu:

Các kiểu dữ liệu trong SQL gồm có các loại sau:

	Trực hằng
Số Chính Xác - Exact Numerics **Số nguyên:** **Bigint** (8 bytes) giá trị từ -2^63 đến 2^63-1 **Int** :(4 bytes) giá trị từ -2^31 đến 2^31 - 1. **SmallInt** : (2 bytes) giá trị từ 2^15 đến 2^15 - 1. **Tinyint** : (1 byte) giá trị từ 0 đến 255. **Luận lý** **Bit** : có giá trị 0, 1 hoặc NULL. **Số thực** **Decimal**(n, d) : -10^38 +1 đến 10^38 –1. n <= 38; d mặc định bằng 0 **Numeric**(n, d) : Giống như decimal. **Tiền tệ :** **Money** : -2^63 đến 2^63 - 1, độ chính xác đến 2 chữ số sau dấu thập phân.. **SmallMoney :** -214,748.3648 đến +214,748.3647,với độ chính xác đến 2 chữ số sau dấu thập phân .	
Số gần đúng - Approximate Numerics **Float**[(n)] : -1.79E + 308 đến 1.79E + 308. **n** có giá trị từ 1 – 53, là số bít dùng lưu phần định trị **Real** : -3.40E + 38 đến 3.40E + 38.	
Ngày giờ	Bao trong dấu

Datetime : 1-1- 1753 đến 31-12- 9999, độ chính xác 3/100 giây hay 3.33 milliseconds. **Smalldatetime** : 1-1- 1900 đến 6-6- 2079, với độ chính xác là 1 phút.	nháy đơn.
Chuỗi ký tự (không theo Unicode) - Character Strings **Char**(n) : độ dài cố định, tối đa là 8000 ký tự. **Varchar**(n): độ dài không cố định, tối đa là 8000 ký tự. **Text** : độ dài không cố định, tối đa là 2^31 – 1 ký tự.	Bao trong dấu nháy đơn
Chuỗi ký tự Unicode - Unicode Character Strings **nChar**(n) : độ dài cố định, tối đa là 4000 ký tự. **nVarchar**(n) : độ dài không cố định, tối đa là 4000 ký tự. **nText** : độ dài không cố định, tối đa là 2^30 – 1 ký tự.	Bao trong dấu nháy đơn và phải bắt đầu bằng chữ N: **N'sssss'**
Số nhị phân - Binary Strings **Binary**(n) :độ dài cố định (tối đa 8000 bytes). **Varbinary(n) :** độ dài thay đổi (tối đa 8000 bytes). **Image** : độ dài thay đổi (tối đa 2^31-1 bytes).	**0Xnnnn**
Other Data Types **Cursor** : kiểu con trỏ **Sql_Variant** : Nhận giá trị của nhiều kiểu dữ liệu khác nhau trong SQL Server ngoại trừ các kiểu text, ntext, timestamp, và sql_variant. **Table** : dùng lưu trữ các tập dữ liệu cho lần xử lý sau. **Timestamp** : kiểu số (binary(8) hay varbinary(8)). Cột khai báo kiểu này sẽ được tự động cập nhật với giá trị phân biệt mỗi khi thêm một mẫu tin mới.. **UniqueIdentifier** : A globally unique identifier (GUID).	

Chú ý: Kiểu Text, nText, và Image không dùng cho biến cục bộ.

Trực hằng (*Literals*): bao gồm hằng số (*Number - Ví dụ. 1234.56 1234.56*), hằng văn bản (*Text*) và ngày giờ trong cặp dấu nháy đơn (*Ví dụ.* 'Nguyễn Hồng Anh') và hằng lôgic (*True* hay *False*)

III- TRUY XUẤT DỮ LIỆU : (*DATA QUERY LANGUAGE*) :

SELECT <*danh sách các cột*>
[INTO *new_table* **]**
[FROM <*bảng nguồn*>**]**
[WHERE <*điều kiện chọn dòng*> **]**
[GROUP BY <*danh sách cột khóa phân nhóm*>**]**
[HAVING <*điều kiện chọn nhóm*> **]**
[ORDER BY <*danh sách cột khóa sắp xếp*> **[**ASC | DESC **]]**

Cú pháp đầy đủ của phát biểu Select khá phức tạp. Chúng ta sẽ lần lượt làm rõ từng phần của phát biểu này.

1- FROM CLAUSE :

Chỉ định nguồn dữ liệu. Để truy vấn thông tin từ nhiều bảng, Sử dụng phép kết giữa các bảng trong mệnh đề FROM: (theo ANSI)

<*table_source*> <*join_type*> <*table_source*> **ON** <*search_condition*>

<*join_type*>: Gồm các phép kết :

INNER JOIN, LEFT [OUTER] JOIN; RIGHT [OUTER] JOIN;
FULL [OUTER] JOIN : kết hợp Left Join và Right Join
CROSS JOIN : không có mệnh đề ON và là phép tích Cartesian.

<*search_condition*>: chỉ định điều kiện liên kết giữa 2 bảng.

2- SELECT CLAUSE :

SELECT [DISTINCT] [TOP n [PERCENT] [WITH TIES]] <column_list>
[FROM <*table_list*>]

- **DISTINCT** : Chỉ hiện những dòng có dữ liệu phân biệt.Mặc định các dòng trùng dữ liệu đều được xuất hiện trong kết quả.
- **TOP n [PERCENT] :** chỉ hiện **n** dòng hoặc n% dòng đầu tiên. **n** là số nguyên tứ 0 đến 100.
- **WITH TIES:** hiện luôn những dòng có cùng giá trị của những cột khóa sắp xếp trong mệnh đề ORDER BY.
- **<column_list>** ::= { * | { table_name | view_name | table_alias }.*
  | column_name [[AS] column_alias]
  | expression [[AS] column_alias]
  | column_alias = expression } [,...n]
  - Dấu * : hiển thị tất cả các cột của các table
  - { TableName | ViewName | TableAlias }.* : Hiện tất cả các cột của bảng chỉ định.
  - ColumnName [AS] ColunmAlias : Đổi tên cột trên bảng nguồn. Nếu bí danh có dấu cách hoặc trùng với từ khoá của SS, bạn phải ghi bí danh trong dấu nháy đơn hoặc dấu ngoặc vuông ([…])
  - **column_alias = expression** hoặc ***expression [AS] column_alias*:** Tạo cột tính toán. Nếu không chỉ định Column Alias thì cột không có tên.

    expression là dãy các toán hạng (*Operand*) nối với nhau bởi các phép toán (*Operator*):

a- Các phép toán có thể là:

Các phép toán số học: * (*nhân*), / (*chia*), % (*phần dư*); + (*cộng*), - (*trừ*). Thứ tự ưu tiên cao nhất theo 3 cụm từ trái qua phải.

Phép nối chuỗi : (+)

b- Toán hạng có thể là:

Tên thuộc tính (có thể kèm theo tên bảng và dấu chấm đứng trước).

Hàm (*function*).

Trực hằng (*Literals*): bao gồm hằng số (*Number - Ví dụ. 1234.56 1234.56*), hằng văn bản (*Text*) và ngày giờ trong cặp dấu nháy đơn (*Ví dụ.* 'Nguyễn Hồng Anh') và hằng lôgic (*True* hay *False*); Giá trị NULL.

Tên biến (*Variable*)

c- Các hàm toán học:

ABS(*x*)	: Trị tuyệt đối của x
SQRT(*x*)	: Căn bậc hai của x
SQUARE(x)	: x^2
POWER(y, x)	: y^x
LOG(*x*)	: Logarit tự nhiên của x
EXP(*x*)	: Hàm mũ cơ số e của x: e^x.
SIGN(*x*)	: Lấy dấu của số x (-1: x<0, 0: x=0, +1: x>0)
ROUND(*x,n*)	: Làm tròn tới n số lẻ.
CEILING(x)	: Số nguyên nhỏ nhất nhưng lớn hơn x
FLOOR(X)	: Số nguyên lớn nhất nhưng nhỏ hơn x

... và các hàm lượng giác: SIN, COS, TAN, ASIN, ACOS, ATAN ...

d- Các hàm xử lý chuỗi ký tự:

ACSII(ch) : Mã ASCII của ký tự **ch**

CHAR(n) : Ký tự có mã ASCII là **n**

LOWER(str) : Trả về chuỗi chữ thường
UPPER(str) : Trả về chuỗi chữ hoa
LTRIM(str) : Trả về chuỗi không có dấu cách bên trái
RTRIM(str) : Trả về chuỗi không có dấu cách bên phải
LEFT*(str,n)*: Lấy *n* ký tự phía trái của dãy *str*.
RIGHT(*str,n*): Lấy *n* ký tự phía phải của dãy *str*.
SUBSTRING(*str, start, n*): Lấy *n* ký tự của dãy str kể từ vị trí start trong dãy.
REPLACE(str1, str2, str3): thay thế tất cả str2 trong str1 bằng str3.
STUFF(*str1, start, n, str2*): Thay thế n ký tự trong str1 từ vị trí start bằng chuỗi str2.
STR(x, len [, Dec]): Chuyển số x thành chuỗi.

e- Các hàm xử lý ngày tháng và thời gian:

GETDATE(): Cho ngày tháng năm hiện tại (Oracle: SYSDATE)
DAY(*dd*): Cho số thứ tự ngày trong tháng của biểu thức ngày *dd*.
MONTH(*dd*): Cho số thứ tự tháng trong năm của biểu thức ngày *dd*.
YEAR(*dd*): Cho năm của biểu thức ngày *dd*.

DATEPART(datepart, date) DATEADD(datepart,number, date) DATEDIFF(datepart, date1, date2)	**Datepart**	**Abbreviations**
	Year	yy, yyyy
	Quarter	qq, q
	Month	mm, m
	Day of year	dy, y
	Day of Month	dd, d
	Week of year	wk, ww
	Weekday	dw
	Hour	hh
	Minute	mi, n
	Second	ss, s
	Millisecond	Ms

f- Các hàm chuyển đổi kiểu giá trị:

CAST(expression AS data_type[(length)])

CONVERT (data_type[(length)], expression [, style])

Style : Dạng thức kiểu ngày mà bạn muốn khi chuyển đổi dữ liệu kiểu datetime hoặc smalldatetime tới kiểu ký tự (nchar, nvarchar, char, varchar, nchar, or nvarchar), Hoặc dạng chuỗi mà bạn muốn khi chuyển dữ liệu kiểu số (float, real, money, or smallmoney) sang kiểu ký tự(nchar, nvarchar, char, varchar, nchar, or nvarchar).

Trong bảng, 2 cột bên trái biểu diễn dạng giá trị datetime hoặc smalldatetime chuyển sang character. Cộng thêm 100 cho giá trị style để được dạng năm 4 chữ số.

Without century (yy)	**With century (yyyy)**	**Standard**	**Input/Output****
-	0 or 100 (*)	Default	mon dd yyyy hh:miAM (or PM)
1	101	USA	mm/dd/yy
2	102	ANSI	yy.mm.dd
3	103	British/French	dd/mm/yy
4	104	German	dd.mm.yy
5	105	Italian	dd-mm-yy
6	106	-	dd mon yy

7	107	-	mon dd, yy
8	108	-	hh:mm:ss
-	9 or 109 (*)	Default+milliseconds	mon dd yyyy hh:mi:ss:mmmAM (or PM)
10	110	USA	mm-dd-yy

Ví dụ: Select Conver(Char(8), GetDate(), 2) -- *kết quả : 04.09.16*

Giá trị Style để chuyển kiểu **float** hay **real** sang kiểu ký tự.

0 (the default): Tối đa 6 chữ số, dùng trong ký hiệu khoa học.

1 luôn luôn 8 chữ số. Luôn dùng trong ký hiệu khoa học.

2 luôn luôn 16 chữ số. Luôn dùng trong ký hiệu khoa học..

In the following table, the column on the left represents the style value for **money** or **smallmoney** conversion to character data.

0 (the default) Không dấu phẩy phân cách hàng ngàn; có 2 chữ số thập phân.

Ví dụ: 4235.98.

1 Có dấu phân cách hàng ngàn và 2 chữ số thập phân;

Ví dụ: 3,510.92.

2 Không dấu phẩy phân cách hàng ngàn, ; có 4 chữ số thập phân.

Ví dụ: 4235.9819.

g- Một số hàm hệ thống:

ISDATE(variable | column name): Kiểm tra dạng ngày hợp lệ. Trả về 1 nếu hợp lệ và 0 nếu không hợp lệ.

ISNUMERIC(variable | column name): Kiểm tra dạng số hợp lệ. Trả về 1 nếu hợp lệ và 0 nếu không hợp lệ.

ISNULL(expression, value) : Trả về giá trị value nếu expression có giá trị NULL, ngược lại trả về giá trị của expression. Giá trị trả về cùng kiểu với exoression.

NULLIF(exp1, exp2) : Trả về giá trị NULL nếu exp1 = exp2.

COALESCE(exp1, exp2, .., expN) : trả về biểu thức khác NULL đầu tiên.

@@ROWCOUNT: Trả về số dòng (kiểu integer) trả về bởi phát biểu cuối cùng.

Ví dụ: cập nhật dữ liệu với UPDATE và dùng @@ROWCOUNT để xác định số dòng đã được thay đổi.

```
UPDATE authors SET au_lname = 'Jones'
    WHERE au_id = '999-888-7777'
IF @@ROWCOUNT = 0
    print 'Warning: No rows were updated'
```

@@ERROR : Trả về mã lỗi (integer) nếu có của phát biểu T-SQL cuối cùng. Trả về số 0 nếu không có lỗi.

Ví dụ: Dùng @@ERROR để kiểm tra vi phạm ràng buộc (error #547) trong phát biểu cập nhật.

```
USE pubs
GO
UPDATE authors SET au_id = '172 32 1176'
WHERE au_id = "172-32-1176"
IF @@ERROR = 547
    print "Vi phạm ràng buộc"
```

3- WHERE CLAUSE:

Chọn lựa những mẫu tin theo điều kiện.

Syntax: WHERE <search_condition> | <old_outer_join>

<old_outer_join> ::= column_name { *= | =* } column_name

3.1 **Các phép toán so sánh**: <, <=, !<, >, >=, !>, =, <> hay !=.

Kết quả phép so sánh là giá trị lôgíc (*True* hoặc *False*).

3.2 **Các phép toán lôgic**: NOT, AND (*conjunction*), OR (*disjunction*).

Kết quả các phép toán lôgíc là một giá trị lôgíc.

3.3 Các phép toán phạm vi:

IS [NOT] NULL

[NOT] IN (<*danh sách giá trị*>)

[NOT] BETWEEN <*Min*> AND <*Max*>

[NOT] LIKE '*Mẫu v.bản' --Dùng ký hiệu thay thế là dấu % và dấu (_)*

[NOT] EXISTS(SubQuery) : Trả về True nếu tồn tại ít nhất 1 mẫu tin.

3.4 <*phép so sánh*> [<*lượng từ*>] (*SubQuery*):

- <*Phép so sánh*> có thể là các phép so sánh số học (>, >=, <, <=, <>, =) hoặc phép toán tập hợp IN, LIKE hoặc NOT LIKE.
- <*Lượng từ*> có thể là ALL, ANY (hoặc SOME). Phép so sánh = ANY có thể được thay tương đương bằng phép toán IN; phép so sánh <> ALL có thể thay tương đương bằng phép toán NOT IN.

3.5 Phép kết ngoài (outer join): *column_name* { *= | =* } *column_name*

Trong đó, phép toán (*=) Left outer join và phép toán (=*) Right outer join.

Chú ý: Để định lại thứ tự các phần của kiểu ngày phù hợp với hằng kiểu ngày trong biểu thức điều kiện. Sử dụng lệnh : SET DATEFORMAT mdy | dmy | ymd | ydm | myd | dym

4- ORDER BY CLAUSE:

Mệnh đề ORDER BY dùng sắp xếp kết quả tìm được. Cú pháp mệnh đề này là:

ORDER BY <*têncột*>|<*biểuthức*> **[**ASC | DESC**]**, <*têncột*>|<*biểuthức*> **[**ASC | DESC**]**, ...

5- Tổng hợp dữ liệu:

5.1 Các hàm tổng hợp:

AVG([Distinct] Column_name**)**

Count(*)

Count([Distinct] Column_name**)**

Max(Column_name**)**

Min(Column_name**)**

Sum([Distinct] Column_name**)**

Ví dụ: Cho biết số mặt hàng đã bán trong ngày cuối cùng:

```
SELECT COUNT( DISTINCT  MaMH) FROM HoaDon, CTHD
WHERE HoaDon.NgayHD IN (SELECT MAX(NgayHD) FROM HoaDon)
      AND HoaDon.MaHD = CTHD.MaHD
```

5.2 GROUP BY CLAUSE:

Cú pháp: GROUP BY [ALL] <*danh sách cột khóa phân nhóm*>

Phân nhóm mẫu tin theo giá trị của các cột làm chuẩn phân nhóm, Mỗi nhóm dữ liệu trả về một dòng tổng hợp.

Mệnh đề GROUP BY ALL trả về tất cả các nhóm, kể cả những nhóm không thỏa mãn điều kiện của mệnh đề WHERE.

Chú ý: *Mệnh đề GROUP BY phải chứa tất cả các cột không tổng hợp có trong mệnh đề SELECT.*

5.3 HAVING CLAUSE:

Cú pháp: **HAVING** <*điều kiện chọn nhóm*>

Dùng chỉ định những dòng tổng hợp xuất hiện phải thỏa mãn điều kiện chỉ định.

IV- TOÁN TỬ UNION:

Dùng kết hợp các kết quả của 2 hay nhiều truy vấn vào cùng một kết quả.

SELECT <*danh sách các cột*>
FROM <*bảng nguồn*>
[WHERE <*điều kiện chọn dòng*> **]**
[GROUP BY <*danh sách cột khóa phân nhóm*>**]**
[HAVING <*điều kiện chọn nhóm*> **]**
UNION [ALL]
SELECT <*danh sách các cột*>
FROM <*bảng nguồn*>
[WHERE <*điều kiện chọn dòng*> **]**
[GROUP BY <*danh sách cột khóa phân nhóm*>**]**
[HAVING <*điều kiện chọn nhóm*> **]**
[ORDER BY <*danh sách cột khóa sắp xếp*> **[**ASC | DESC **]]**

Chú ý :

- Các kết quả truy vấn phải cùng số cột, cùng thứ tự và cùng kiểu dữ liệu tương ứng từng cột.
- Bảng kết quả có tên cột được tạo từ Select đầu tiên.
- Mệnh đề ORDER BY chỉ cho phép đứng cuối trong lệnh UNION
- Từ khóa ALL : dùng chỉ định hiển thị cả những dòng trùng dữ liệu. Nếu không có từ khóa ALL thì chỉ hiện các dòng phân biệt.

V- DATA MANIPULATION LANGUAGE (DML)

1- CHÈN MẪU TIN MỚI :

a- Chèn trực tiếp một mẫu tin mới:

Cú pháp: **INSERT INTO** <table name> **[**(column list)**]** **VALUES** (value list)

b- Chèn dữ liệu từ các bảng :

Cú pháp: **INSERT INTO** <table name> **[**(column list)**]** **<**SELECT Statement>

2- SỬA DỮ LIỆU:

Cú pháp 1:

UPDATE <table name> SET {<column name> = <value>} [,...n]
[FROM { < table_source > } [**,**...*n*] **]**
[WHERE <conditions>**];**

Cú pháp 2:

UPDATE <table name> SET {<column name> = <Select statement>}[,…n]
[FROM { < table_source > } [**,**...*n*] **]**

[WHERE <conditions>**];**

Ví dụ :

- Cập nhật số tồn và gấp đôi đơn giá của các mặt hàng có mã loại hàng bằng 1

```
UPDATE MatHang SET SoTon = IsNull(SoTon, 0) + 100 , DonGia = DonGia * 2
        WHERE MALH = 1
```

- Cập nhật số tồn của các mặt hàng đã bán trong ngày

```
UPDATE MatHang SET SoTon = SoTon - SL
    FROM CTHD, HoaDon , MatHang
      WHERE CTHD.SoHD = HoaDon.SoHD
            And CTHD.MaMH = MatHang.MaMH
            AND HoaDon.NgayHD = (SELECT MAX(HoaDon.NgayHD) FROM
                                 HoaDon)
```

- Cập nhật tiền Hóa đơn của các hóa đơn bán trong ngày

```
UPDATE HoaDon
    SET TienHD = (SELECT SUM(SL*DGBan) FROM CTHD
          WHERE HoaDon.MaHD = CTHD.MaHD)
    Where HoaDon.NgayHD IN (SELECT MAX(NgayHD) FROM HoaDon)
```

- Tăng đơn giá của 10 mặt hàng có đơn giá thấp.

```
UPDATE MatHang SET DonGia  =  DonGia * 1.1
FROM MatHang,
          (SELECT TOP 1 * FROM MatHang ORDER BY DonGia) AS t1
WHERE MatHang.MaMH = t1.MaMH
```

Hay có thể bỏ tên table MatHang trong mệnh đề FROM

```
UPDATE MatHang SET DonGia  = DonGia * 1.1
      FROM (SELECT TOP 1 * FROM MatHang ORDER BY DonGia)
            AS t1 WHERE MatHang.MaMH = t1.MaMH
```

3- XÓA MẪU TIN:

Cú pháp 1: **DELETE FROM** <Table Name> **[WHERE** <conditions>**];**

Cú pháp 2: **DELETE <Table Name> FROM** <Join Table> **[WHERE** <conditions>**];**

Ví dụ :

- Xóa hóa đơn có mã hóa đơn bằng 1

```
Delete From CTHD WHERE MAHD = 1 And MaMH = 5
```

- Xóa các CTHD của các hóa đơn đã bán trong ngày

```
Delete CTHD
    FROM CTHD, HoaDon
      WHERE CTHD.SoHD = HoaDon.SoHD
      AND HoaDon.NgayHD = (SELECT MAX(HoaDon.NgayHD)
            FROM HoaDon)
```

- Xóa MatHang đơn giá thấp.

```
DELETE MatHang
FROM MatHang,
      (SELECT TOP 1 * FROM MatHang ORDER BY DonGia) AS t1
      WHERE MatHang.MaMH = t1.MaMH
```

Hay có thể bỏ tên table MatHang trong mệnh đề FROM

```
DELETE MatHang
    FROM (SELECT TOP 1 * FROM MatHang ORDER BY DonGia)
        AS t1 WHERE MatHang.MaMH = t1.MaMH
```

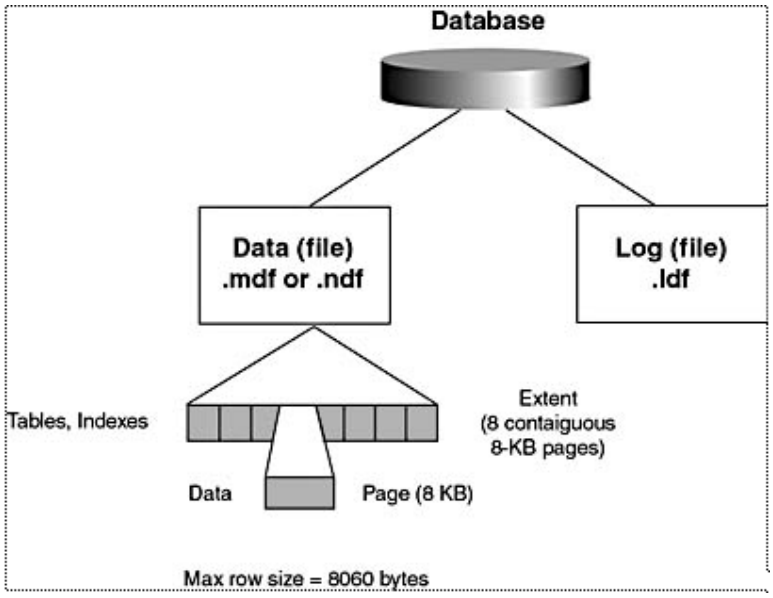
4- Tạo mới một bảng với các bộ giá trị lấy từ CSDL:

Cú pháp: SELECT <Column list> **INTO <new table name>** ….

Chương 3 : TẠO LẬP CSDL TRÊN SQL SERVER

I- Các Loại File Lưu Trữ CSDL :

Khi tạo một CSDL, SQL Server sẽ tạo những file lưu trữ. Có 2 loại file như sau:



- File dữ liệu: bao gồm
  - File chính (Primary data file): Mỗi CSDL chỉ có 1 file dữ liệu chính có phần mở rộng là MDF.
  - Các file dữ liệu phụ (Secondary data files): Các file này chứa các dữ liệu và đối tượng không nằm vừa trong Primary file. Một số CSDL có thể lớn đến nỗi phải cần nhiều file dữ liệu phụ hay cần sử dụng các file phụ trên các ổ đĩa riêng để phân dữ liệu qua nhiều đĩa. Các file dữ liệu phụ tiếp theo nên có phần mở rộng là NDF.
- Các file nhật ký (Log files): lưu trữ nhật ký giao tác (LDF) (Transaction log) thực hiện trên CSDL, nhằm mục đích phục hồi CSDL khi có sự cố.

Khi sử dụng nhiều file dữ liệu, SQL Server tự động trải dữ liệu qua tất cả các file dữ liệu. Điều này làm giảm tranh chấp và các điểm nóng (hotspot) trong dữ liệu.

Tuy nhiên, đối với file nhật ký, SQL không trải thông tin trên các file nhật ký. Khi 1 file nhật ký đầy, thông tin sẽ được ghi tiếp vào file khác.

II- Tạo CSDL:

Để tạo CSDL bạn có thể dùng câu lệnh Create Database trong Query Analyzer hoặc sử dụng tiện ích Enterprise Manager.

1- Bằng Lệnh CREATE DATABASE:

Cú pháp : **CREATE DATABASE** *DatabaseName*
[ON [PRIMARY] (<*Thông tin File Dữ Liệu***>) ,…]**
[LOG ON (<*Thông tin File Log***>) ,…]**

Trong đó :

<Thông tin File> = (FILENAME = *'d:\Path\FileName'*
[, NAME = *LogicalName* **]**
[, SIZE = *<Size MB or KB>***]**
[, MAXSIZE = *< MaxSize >* **]**
[, FILEGROWTH = *<No of KyloByte or Percentage>***])**

Arguments

- **Database_name :** Tên Database phải được phân biệt trên cùng server (tối đa 128 ký tự)
- **ON :** Khai báo các file chứa CSDL
- **PRIMARY:** Dùng chỉ định file chính của CSDL. Nếu không chỉ định Primary, file đầu tiên được liệt kê trong phát biểu Create Database trở thành primary file.

- **NAME = 'LogicalName' :** Tên luận lý của File lưu trữ CSDL. Tên này được sử dụng trong các phát biểu của T-SQL. Yêu cầu phân biệt.
- **FILENAME = 'FileName' :** Tên lưu trên đĩa. Bao gồm cả ổ đĩa, thư mục
- **SIZE = <size MB or KB> :** Kích thước File theo đơn vị MB (mặc định) hoặc KB. Thấp nhất 512 KB, mặc định 1 MB.
- **MAXSIZE = max_size :** Chỉ định kích thước tối đa mà file có thể tăng. Nếu không chỉ định, kích thước file sẽ tăng cho đến khi đĩa đầy.
- **FILEGROWTH :** Khai báo số gia khi tăng kích thước File, không được lớn hơn MaxSize. Mặc định là 10% và giá trị nhỏ nhất là 64 KB.
- **LOG ON :** Khai báo các file dùng lưu trữ nhật ký thao tác trên database. Nếu không chỉ định LOG ON, SQL tự tạo một file nhật ký có size bằng 25 percent của tổng kích thước của tất cả các data files trên database.

Ví dụ 1: Tạo CSDL BanHang, bắt đầu có kích thước 20MB – trong đó, 15MB dành cho file dữ liệu và 5MB dành cho file nhật ký.

```
CREATE DATABASE BanHang
    ON  ( NAME = Sales_dat,
          FILENAME = 'c:\mssql7\data\saledat.mdf',
          SIZE = 15MB,   MAXSIZE = 50MB,
          FILEGROWTH = 20% )
    LOG ON ( NAME = 'Sales_log',
          FILENAME = 'c:\mssql7\data\salelog.ldf',
          SIZE = 5MB,   MAXSIZE = 20MB,
          FILEGROWTH = 1MB )
```

Ví dụ 2: Tạo CSDL lưu ở nhiều file. Theo Microsoft, File dữ liệu đầu tiên có phần mở rộng là MDF, các file dữ liệu còn lại có phần mở rộng là .NDF. Các file nhật ký có phần mở rộng là LDF.

```
CREATE DATABASE BanHang
    ON ( NAME = Sales_dat1,
        FILENAME = 'c:\data\sale.mdf',
         SIZE = 10,
         MAXSIZE = 50,
         FILEGROWTH = 5 ),
        ( NAME = Sales_dat2,
         FILENAME = 'c:\data\sale1.ndf',
         SIZE = 10,     MAXSIZE = 50,
         FILEGROWTH = 5 )
    LOG ON
        ( NAME = 'Sales_log1',
         FILENAME = 'c:\data\sale.ldf',
         SIZE = 5,   MAXSIZE = 20,
         FILEGROWTH = 5 ),
        ( NAME = 'Sales_log2',
         FILENAME = 'c:\data\sale1.ldf',
         SIZE = 5,   MAXSIZE = 20,
         FILEGROWTH = 5 )
```

Do không dùng từ khóa PRIMARY nên mặc định file đầu tiên (Sale_dat1) là file chính.

```
CREATE DATABASE BanHang
    ON ( NAME = Sales_dat1,
        FILENAME = 'c:\data\sale.mdf',
        SIZE = 10,
        MAXSIZE = 50,
        FILEGROWTH = 5 ),
        PRIMARY ( NAME = Sales_dat2,
        FILENAME ='c:\data\sale1.ndf',
         SIZE = 10,   MAXSIZE = 50,
         FILEGROWTH = 5 )
    LOG ON
        ( NAME = 'Sales_log1',
         FILENAME = 'c:\data\sale.ldf',
         SIZE = 5,   MAXSIZE = 20,
         FILEGROWTH = 5 ),
        ( NAME = 'Sales_log2',
         FILENAME = 'c:\data\sale1.ldf',
         SIZE = 5,   MAXSIZE = 20,
         FILEGROWTH = 5 )
```

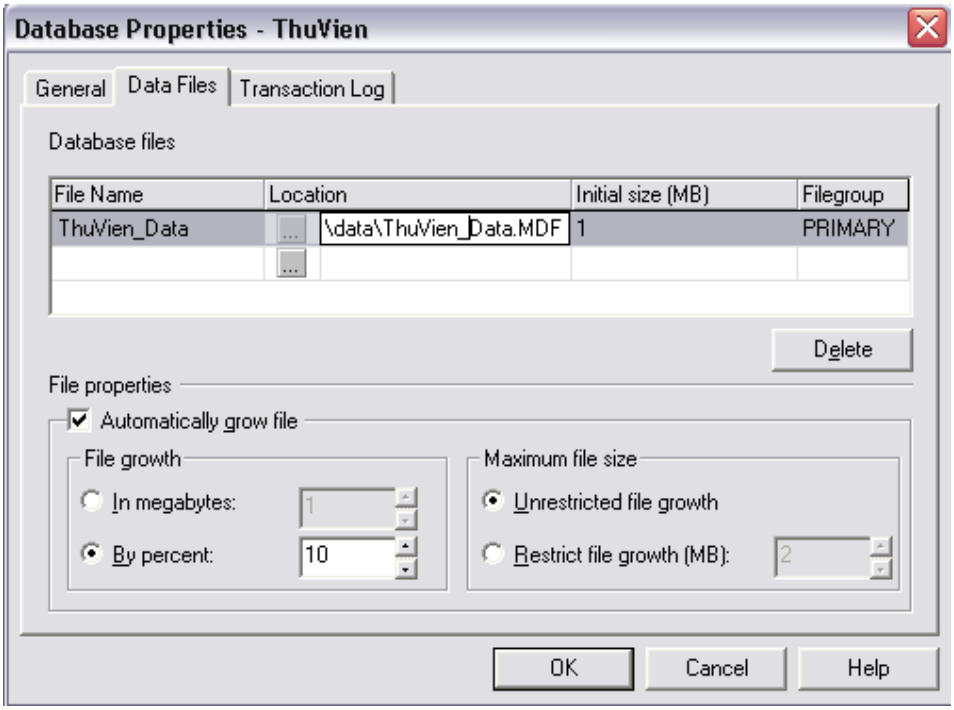
Do dùng từ khóa PRIMARY nên file (Sale_dat2) là file chính.

Chú thích:

- CSDL mới được tạo là bản sao của CSDL Model, nên mọi thứ trong CSDL Model sẽ có trong CSDL mới. Mặc định, các thành viên có vai trò *sysadmin* – System Administrater và *dbcreator* – Database Creators mới có quyền tạo Database mới.

2- Bằng Enterprise Manager:

Click phải vào Folder Databases hoặc khoảng trống trên khung bên trái, chọn New Database, cửa sổ tạo CSDL được hiển thị:



Tab General: Nhập tên Database trong ô Name, ví dụ: TheThao.

Tab Data Fiels : Database files: hiện dòng chứa tên file dữ liệu, ví dụ: Thethao_Data, với kích thước ban đầu là 1MB trong thư mục mặc định …\Data.

File properties:

Check Box Automatically grow file: được chọn để cho phép tăng kích cỡ file. Kích cỡ tối đa, được chỉ định không hạn chế (Unrestricted filegrowth) hoặc hạn chế (Restrict filegrowth (MB)).

Tab Transaction Log: tên file nhật ký mặc định là TheThao_Log.LDF

III- Xóa CSDL

1- Bằng lệnh DROP DATABASE:

Cú pháp: **DROP DATABASE** <DatabaseName> [, …]

Ví dụ: DROP DATABASE mydb1, mydb2

Chú ý:

- CSDL Master có Table **SYSDATABASES** chứa thông tin như Name, ID,… của các database trên Server

If Exists(Select 'True' From master..SysDatabases Where Name = 'Thuvien')

Drop Database ThuVien

- Bạn phải có vai trò db_owner trên CSDL.
- Không ai đang làm việc với CSDL

2- Bằng Enterprise Manager:

Click phải vào tên Database trên khung bên trái, chọn **Delete**.

IV- Sửa Đổi CSDL

1- Bằng lệnh ALTER DATABASE:

Để thêm hay xóa file và nhóm file hoặc thay đổi các thuộc tính của file và nhóm file, như thay đổi tên và dung lượng của file sử dụng cú pháp:

ALTER DATABASE databasename
ADD FILE <Thông tin File Dữ Liệu> [,...n]
| **ADD LOG FILE** <Thông tin file Log > [,...n]
| **REMOVE FILE** <Tên logic>
| **MODIFY FILE** <Thông tin file>

Các ví dụ:

A. Thêm 1 file chứa dữ liệu cho database

```
CREATE DATABASE Test
ON   (  FILENAME = 'C:\data\Testdat1.ndf',
        NAME = Testdat1, SIZE = 5MB,  MAXSIZE = 100MB,
        FILEGROWTH = 5MB
        )
GO
ALTER DATABASE Test
        ADD FILE (    NAME = Testdat2,
                      FILENAME = 'c:\mssql7\data\Testdat2.ndf',
                      SIZE = 5MB,  MAXSIZE = 100MB,
                      FILEGROWTH = 5MB
                )
```

C. Thêm 2 file Log kích thước 5-MB cho Database

```
ALTER DATABASE Test
        ADD LOG FILE
                ( NAME = Testlog2,
                  FILENAME = 'C:\Data\Testlog2.ldf',
                  SIZE = 5MB,   MAXSIZE = 100MB,
                  FILEGROWTH = 5MB),
                ( NAME = Testlog3,
                  FILENAME = 'c:\Data\Testlog3.ldf',
                  SIZE = 5MB,   MAXSIZE = 100MB,  FILEGROWTH = 5MB)
```

D. Xoá file của database

```
ALTER DATABASE Test REMOVE FILE Testdat2
```

E. Sửa file : Tăng kích thước file cho Test database in Example B.

```
ALTER DATABASE Test
        MODIFY FILE (NAME = Testdat1,  SIZE = 20MB)
```

2- Bằng Enterprise Manager:

Click phải vào tên Database trên khung bên trái, chọn **Properties**.

V- Đổi Tên CSDL:

EXEC SP_RENAMEDB *'OldName'*, *'NewName'*

Chú ý:

☑ Bạn phải là có quyền Admin

- ☑ CSDL phải ở chế độ SingleUser
- ☑ Tên các file và nhóm file không bị ảnh hưởng bởi sự thay đổi.

VI- Xem Thông Tin CSDL:

1- Trong Query Analyzer Bằng Thủ Tục Lưu Trữ Hệ Thống:

Thông tin của các Database được lưu trữ trong table hệ thống SYSDATABASES của CSDL Master. Bạn có thể liệt kê bằng lệnh:

```
Select * From Master.dbo.SysDatabases
```

Ngoài ra, bạn có thể dùng các system stored procedures để hiển thị thông tin about databases and database options:

System stored procedure	Description
SP_HELPDB	Hiển thị thông tin của tất cả Databases trên Server gồm: name, size, owner, ID, creation date, and status information.
SP_HELPDB *database_name*	Hiển thị thông tin của database được chỉ định, gồm : name, size, owner, ID, creation date, and status information. Ngoài ra còn cho biết chi tiết các file dữ liệu và log file.
SP_SPACEUSED [*objname*]	Kích thước của current database hoặc table trong current database.

Ví dụ 1: Xem thông tin của CSDL TheThao

```
EXEC SP_HELPDB TheThao
```

Ví dụ 2: Xem kích thước CSDL TheThao.

```
USE TheThao
Go
EXEC SP_SPACEUSED
```

Ví dụ 3: Xem kích thước table tblCLB

```
USE TheThao
Go
EXEC SP_SPACEUSED tblCLB
```

VII- TẠO TABLE :

Khi tạo table, bạn cần quan tâm đến các yêu tố trên các Field được tạo như :

Key : Field đó là khóa hay không

ID : Field đó có thuộc tính Identity hay không

Column Name : Tên Field

Data Type : Kiểu dữ liệu của Field

Size : Kích thước lưu trữ

Allow Null : Cho phép Null

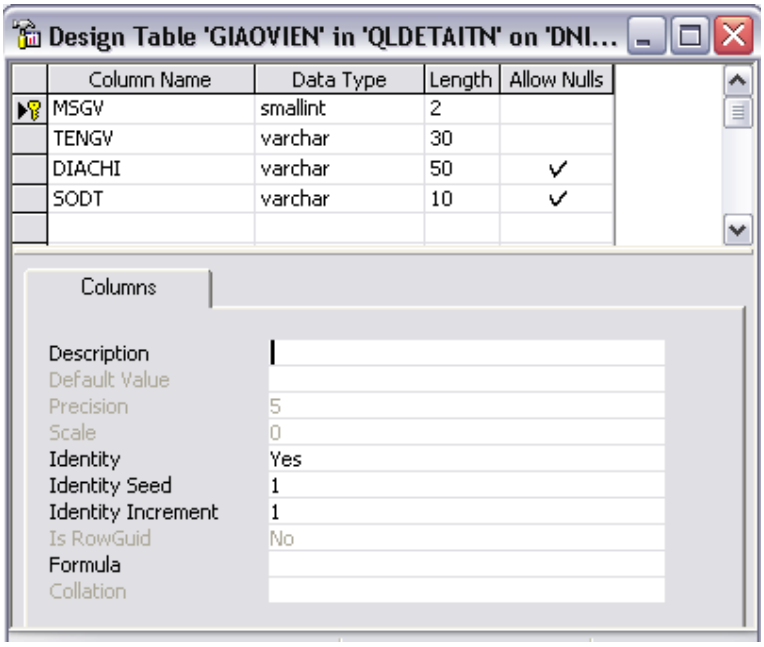
Default : Giá trị mặc định

Identity : Field số có giá trị tăng tự động, với giá trị ban đầu là bao nhiêu (Identity Seed) và số gia(Identity Increament) là bao nhiêu.

Tên Table bạn nên đặt với tiếp đầu ngữ là "tbl".

1- BẰNG ENTERPRISE MANAGER:

Click phải vào mục Tables trên khung trái và chọn New Table…



2- Bằng lệnh Create Table:

Cú pháp :

CREATE TABLE [DatabaseName.[Owner].| Owner.] TableName

(<ColumnDefinition> | ColumnName AS <Expression> | <TableConstraint> [,...n])

a) Khai Báo Cấu Trúc :

Mỗi cột bạn cần chỉ định: <TênCột> <DataType>

Example

```
CREATE TABLE customer
(cust_id uniqueidentifier ROWGUIDCOL NOT NULL DEFAULT NEWID(),
cust_name char(30) NOT NULL)
```

b) Khai báo ràng buộc :

Có 2 loại khai báo RB:

☑ **<u>Khai báo RBTV trên mỗi cột:</u> (Column Constraint)**

Sau lời khai báo tên và kiểu của cột bạn có thể khai báo tiếp các RB dữ liệu trên cột đó.

Các RB có thể là :

- NULL hoặc NOT NULL
- PRIMARY KEY hoặc UNIQUE [CLUSTERED | NONCLUSTERED]
  - Clustered : Một Table chỉ có 1 chỉ mục loại này, xác định thứ tự vật lý của table. Được dùng cho các thuộc tính thường phải tìm kiếm giá trị trên đó
  - Non-Clustered: Sắp xếp dữ liệu theo các Field chỉ định. Một Table có thể có nhiều chỉ mục loại này. Dữ liệu và Index được lưu ở 2 nơi khác nhau .
- REFERENCES ref_table [(ref_column)]
  [ON DELETE { CASCADE | NO ACTION}]
  [ON UPDATE { CASCADE | NO ACTION }]
- CHECK (logical_expression)
- DEFAULT <Exp> : Giá trị mặc định của cột trên mẫu tin mới
- IDENTITY [(seed, increment)] : Thường được dùng khai báo cho cột kiểu số nguyên, là khoá chính của table. Chỉ định giá trị được ghi trên mẫu tin mới là một số phân biệt với các mẫu tin khác. Trong đó : Seed -giá trị khởi đầu và Increment - số gia.

Ví dụ: CREATE TABLE [dbo].[SanPham] (
MaSP SmallInt Identity (1,1) NOT NULL ,
TenSP varchar(30) NOT NULL,
DonGia Money Default 0,
SoTon Real Default 0) ON PRIMARY

Chú ý:

- Mỗi table chỉ có 1 cột mang thuộc tính IDENTITY.
- *Hàm IDENT_SEED('tablename')* : trả về giá trị Seed, nếu table không có cột khai báo Identity sẽ cho giá trị NULL.
- *Hàm IDENT_INCR('tablename')* : trả về giá trị Increment.
- Hàm @@IDENTITY dùng nhận giá trị identity của mẫu tin được chèn cuối cùng.

SQL Server tự động đặt tên cho mỗi RB. Tuy nhiên, Bạn có thể đặt tên RB bằng cú pháp :

[CONSTRAINT <Tên RB>]

```
/********************** table CongViec************************** */
CREATE TABLE CONGVIEC
( MACV smallint IDENTITY(1,1) CONSTRAINT PK_CV PRIMARY KEY CLUSTERED,
  MOTA varchar(50) CONSTRAINT NN_MOTA NOT NULL DEFAULT 'Cong Viec Moi',
  LCBmin tinyint CONSTRAINT NN_LCBmin NOT NULL
                CONSTRAINT CHK_LCBmin CHECK (LCBmin >= 10),
  LCBmax tinyint CONSTRAINT NN_LCBmax NOT NULL
                CONSTRAINT CHK_LCBmax CHECK (LCBmax <= 250)
)
/* ***************** publishers table ******************** */
      CREATE TABLE publishers
      ( pub_id  char(4) NOT NULL
        CONSTRAINT UPKCL_pubind PRIMARY KEY CLUSTERED
```

```
    CHECK (pub_id IN ('1389', '0736', '0877', '1622', '1756')
        OR pub_id LIKE '99[0-9][0-9]'),
  pub_name varchar(40) NULL, city varchar(20) NULL,
  state  char(2) NULL, country varchar(30) NULL DEFAULT('USA')
)
```

☑ **Khai báo ràng buộc trên nhiều cột của Table: (Table Constraint)**

Được khai báo riêng, nằm sau các khai báo cột. Bao gồm các ràng buộc:

- ☑ [CONSTRAINT *constraint_name*]
- ☑ PRIMARY KEY | UNIQUE **(** *column* [ASC | DESC] [**,**...*n*] **)**
- ☑ FOREIGN KEY **(** *column* [**,**...] **)** REFERENCES *ref_table* [**(** *ref_column* [**,**...] **)**]
  [ON DELETE { CASCADE | NO ACTION }]
  [ON UPDATE { CASCADE | NO ACTION }]
- ☑ CHECK **(** *logical_expression* **)**

Ví dụ:

```
CREATE TABLE DDH
(  MADH  SmallInt IDENTITY(1,1) CONSTRAINT PK_DDH PRIMARY KEY CLUSTERED,
   MaNCC SmallInt NOT NULL CONSTRAINT FK_DHH_NCC REFERENCES NHACC(MaNCC),
   NgayDH SmallDateTime DEFAULT GetDate() NOT NULL,
   NgayYCGiao SmallDateTime DEFAULT GetDate() NOT NULL,
   CONSTRAINT CHK_NgayYCGiao CHECK (NgayDH <= NgayYCGiao)
)
Go
EXEC SP_HELP DDH          /*Xem thông tin Table
```

c) Khai báo cột biểu thức:

Cột biểu thức, còn gọi là cột ảo, có dữ liệu lấy từ những cột khác bằng những biểu thức. Không dùng biểu thức là truy vấn con. Cột ảo không tham gia vào khóa của bảng.

Cú pháp: ColumnName AS <Expression>

Ví dụ :

```
CREATE TABLE mytable
    (   low int, high int,
        myavg AS (low + high)/2
    )
EXEC sp_help mytable      /*Xem thông tin Table
```

d) Tạo Table Tạm :

Table được tạo bằng các phát biểu trên sẽ được lưu trữ trong CSDL đang mở. Nhưng đôi khi bạn cần tạo một Table mà bạn sẽ xóa ngay. Khi đó bạn có thể tạo table tạm lưu trữ bên trong CSDL TempDB bằng cú pháp trên nhưng sử dụng ký hiệu # (Pound) kề trước tên Table.

```
CREATE TABLE #MyTable (
    Field1 int PRIMARY KEY,
    Field2 char(10) NOT NULL,
    Field3 datetime )
```

Table tạm có thể cục bộ (local) hoặc toàn cục phụ thuộc vào phạm vi hoạt động:

- ☑ Table cục bộ, được khai báo với 1 dấu #, chỉ được truy xuất bởi kết nối vừa tạo ra nó.
- ☑ Table toàn cục, được khai báo với 2 dấu #, được truy xuất bởi các kết nối hiện hành

VIII- THAY ĐỔI CẤU TRÚC BẢNG – ALTER TABLE:

Các trường hợp thay đổi cấu trúc bảng, đó là: Bổ sung thêm cột mới (ADD *Column*); Xóa cột; Sửa đổi định nghĩa của cột (MODIFY *Column*) và lệnh hủy bỏ RBTV trên cột hay trên cả bảng.

Cú pháp chung:

ALTER TABLE table_name

ALTER COLUMN column_name {data_type [(p[, s])] [NULL|NOT NULL]}
| **ADD** {[<column_definition>] | ColName AS Expression },...
| **DROP COLUMN** column [,...n]
| **[WITH CHECK | WITH NOCHECK] ADD** {<table_constraint> }[,...n]
| **DROP** [CONSTRAINT] constraint_name
| **{CHECK | NOCHECK} CONSTRAINT** {ALL | constraint_name[,...n]}
| **{ENABLE | DISABLE} TRIGGER** {ALL | trigger_name[,...n]}

1- Sửa đổi kiểu dữ liệu hoặc kích thước của cột :

ALTER TABLE table
ALTER COLUMN ColName DataType [(p [, s])] [NULL | NOT NULL]

Ví dụ: Sửa lại kích thước cột tên nhân viên (**Name**) thành 25 ký tự:

ALTER TABLE NhanVien ALTER COLUMN HoTenNV CHAR (25)

Chú ý: Không thể sửa cột :

- ☑ Có kiểu **text**, **image**, **ntext**, or **timestamp**
- ☑ Cột tính toán hoặc đã dùng trong cột tính toán
- ☑ Đã sử dụng trong các ràng buộc ngoại trừ chỉ thay đổi kích thước.
- ☑ Không được phép sửa đổi kích thước của cột cho nhỏ lại, và cũng không được phép thay đổi kiểu dữ liệu của cột, trừ trường hợp cột đó chưa có dữ liệu gì.
- ☑ Không thể sửa đổi cột hiện chứa giá trị NULL từ thuộc tính NULL thành NOT NULL.

2- Thêm cột

ALTER TABLE table_name **ADD** <Định nghĩa cột> [,...]

Ví dụ: Thêm vài cột có ràng buộc:

```
CREATE TABLE ViDu ( CotA INT CONSTRAINT CotA_un UNIQUE)
GO
ALTER TABLE  ViDu ADD
        /* Thêm cột khóa chính */
    CotB INT IDENTITY CONSTRAINT CotB_pk PRIMARY KEY,
        /* Thêm cột tham chiếu với cột khác trên cùng table*/
    CotC INT NULL
        CONSTRAINT CotC_fk REFERENCES ViDu(CotA),
        /* Thêm cột với ràng buộc dạng thức của dữ liệu  */
    CotD VARCHAR(16) NULL
        CONSTRAINT CotD_chk
            CHECK (CotD IS NULL  OR CotD LIKE "[0-9][0-9][0-9]-[0-
              9][0-9][0-9][0-9]"  OR CotD LIKE "([0-9][0-9][0-9]) [0-9][0-
              9][0-9]-[0-9][0-9][0-9][0-9]"),
```

```
        /* Thêm cột với giá trị mặc định */
        CotE DECIMAL(3,3) CONSTRAINT CotE_defa DEFAULT .081
    GO
    EXEC SP_HELP ViDu
```

Chú ý: Trong trường hợp Table đã có dữ liệu, nếu bạn cần:

- **Thêm cột NOT NULL**: Phải thực hiện qua 3 bước: (*i*) thêm cột với thuộc tính NULL, (*ii*) điền đầy đủ các giá trị cho cột, (*iii*) đổi lại thuộc tính của cột thành NOT NULL.
- **Thêm cột mới và điền giá trị Default cho các dòng đang tồn tại trong bảng:**

  Dùng DEFAULT với thuộc tính **WITH VALUES** để cung cấp giá trị cho mỗi dòng đang tồn tại trong bảng.

  ALTER TABLE MyTable **ADD** AddDate smalldatetime **NOT NULL**
  CONSTRAINT AddDateDflt DEFAULT GetDate() WITH VALUES

3- Xóa cột :

ALTER TABLE <*tablename*> **DROP COLUMN** ColName [,...n]

Ví dụ :
```
CREATE TABLE ViDuXoaCot ( CotA INT, CotB VARCHAR(20) NULL)
GO
ALTER TABLE ViDuXoaCot DROP COLUMN CotB
GO
EXEC sp_help ViDuXoaCot
GO
```

Chú ý: Không thể xóa những cột:

- *Đang dùng trong một Index.*
- *Có ràng buộc CHECK, FOREIGN KEY, UNIQUE, or PRIMARY KEY.*
- *Có chỉ định giá trị DEFAULT.*
- *Có chỉ định rule.*

4- Thêm RBTV cho bảng :

ALTER TABLE TableName
[WITH CHECK | WITH NOCHECK] ADD { <TableConstraint> }[,...n]

- <TableConstraint>::= [CONSTRAINT *constraint_name*]
  PRIMARY KEY(*Col*1 [**,***...n*] **)**
  UNIQUE (*Col1* [**,***...n*] **)**
  FOREIGN KEY (*Col1* [**,***...n*] **) REFERENCES** *ref_table* (*ref_col* [**,***...n*] **)**
  [ON DELETE { CASCADE | NO ACTION }]
  [ON UPDATE { CASCADE | NO ACTION }]
  DEFAULT <Exp> **FOR** *Col* [**WITH VALUES**]
  CHECK(*BT Điều kiện* **)**

Ví dụ:
```
ALTER TABLE SanPham
    ADD CONSTRAINT SoTon_defa DEFAULT 0 FOR SoTon
```

- **WITH NOCHECK:** Không kiểm tra ràng buộc đối với những dữ liệu đang có trên cột

```
ALTER TABLE ViDu WITH NOCHECK
    ADD CONSTRAINT CotA_chk CHECK (CotA > 1)
GO
ALTER TABLE CTDH WITH NOCHECK
```

ADD CONSTRAINT SoLuong_chk CHECK (SoLuong >=0)

Chú ý: Để sửa đổi RBTV, trước hết cần loại bỏ (DROP) chúng rồi sau đó bổ sung RBTV mới.

5- Hủy bỏ RBTV khỏi bảng.

ALTER TABLE table **DROP CONSTRAINT** <T*ên RBTV*>

6- Kích hoạt hay tạm ngưng kiểm tra các ràng buộc Foreign key và Check:

ALTER TABLE table

{CHECK | NOCHECK} CONSTRAINT {ALL | ConstraintName[,...n]}

Ví dụ :-- *Disable the constraint and try again.*

ALTER TABLE ViDu NOCHECK CONSTRAINT CotA_chk

-- Reenable the constraint and try another insert, will fail.

ALTER TABLE ViDu CHECK CONSTRAINT CotA_chk

7- Đổi tên cột :

SP_RENAME 'TableName.OldColName', 'NewColName', **'COLUMN'**

Ví dụ: SP_RENAME 'customers.[contact title]', 'title', **'COLUMN'**

IX- XÓA TABLE :

Cú pháp:: **DROP TABLE** <*tên bảng*> [, …]

Ví dụ:

A. Xóa table trong database hiện hành: DROP TABLE titles1

B. Xóa table trong database khác : DROP TABLE pubs.dbo.authors2

Chú ý: Không thể xóa Table cha được tham chiếu bởi Table khác.

X- ĐỔI TÊN BẢNG:

Cú pháp: EXEC **SP_RENAME** 'OldName', 'NewName'

Ví dụ: Đổi tên table **customers** thành **custs**.

EXEC sp_rename 'customers', 'custs'

XI- XÓA CÁC DÒNG TRÊN TABLE VÀ GIẢI PHÓNG VÙNG NHỚ:

Cú pháp: **TRUNCATE TABLE <*TableName*>**

Ngữ nghĩa: Khác với DELETE FROM <*tên bảng*> được sử dụng để xóa bản ghi khỏi bảng nhưng vùng nhớ trên bộ nhớ thứ cấp (đĩa từ hoặc vật mang tin từ tính khác) vẫn không được giải phóng để dùng lại. Lệnh TRUNCATE sau khi đã xóa bỏ các bản ghi khỏi bảng thì vùng nhớ của các bản ghi này sẽ được thu hồi và cho phép các bảng khác sử dụng.

Ghi chú: Chỉ có người tạo ra bảng hoặc những người quản trị CSDL mới có quyền hạn TRUNCATE bảng.

Chương IV : TẠO VIEW (BẢNG ẢO)

I- Khái niệm :

View là đối tượng dùng truy xuất dữ liệu trên các Table tạo ra bảng ảo chứa dữ liệu được yêu cầu. Thông qua

II- SỬ DỤNG T-SQL:

1- Tạo View bằng T-SQL:

```
CREATE VIEW ViewName [(ColName1, ...)]
        [WITH ENCRYPTION]
AS <Phát biểu select>
        [WITH CHECK OPTION]
```

Ví dụ:

```
USE BanHang
CREATE VIEW vwTienHD (MaHD, NgayLap, TienHD)
        AS  SELECT a.MaHD, NgayHD, SUM(SL*DG)
                FROM HoaDon a, CTHD b
                GROUP BY a.MaHD, NgayHD
                HAVING SUM(SL*DG) > 0
Go
EXEC sp_helptext vwTienHD
SELECT * FROM vwTienHD
```

```
Ví dụ:      CREATE VIEW vwTienHD (MaHD, NgayLap, TienHD)
            AS
            SELECT a.MaHD, NgayHD, SUM(SL*DG)
                    FROM HoaDon a, CTHD b WHERE Year(NgayHD) =2004
                    GROUP BY a.MaHD, NgayHD HAVING SUM(SL*DG) > 0
            Union
            SELECT a.MaHD, NgayHD, SUM(SL*DG)
                    FROM HoaDon a, CTHD b WHERE Year(NgayHD) =2005
                    GROUP BY a.MaHD, NgayHD HAVING SUM(SL*DG) > 0
        Go
        EXEC SP_HelpText vwTienHD
```

2- Các hạn chế khi tạo Views

- Views không thực hiện trên các bảng tạm.
- Lệnh CREATE VIEW không thể kết hợp với các phát biểu T-SQL khác trong một gói
- Views không có hơn 1024 columns.
- Không chứa mệnh đề INTO.

3- Mã hóa View:

Để tránh trường sao chép định nghĩa View, bạn có thể mã hóa View bằng cách thêm từ khóa WITH ENCRIPTION trước từ khóa AS.

```
Ví dụ:      CREATE VIEW vwTienHD (MaHD, NgayLap, TienHD)
            WITH ENCRIPTION
```

```
AS
SELECT a.MaHD, NgayHD, SUM(SL*DG)
        FROM HoaDon a, CTHD b
        GROUP BY a.MaHD, NgayHD
        HAVING SUM(SL*DG) > 0
GO
EXEC SP_HELPTEXT vwTienHD
```

Kết quả là: *The Object comments have been encription.*

Chú ý: View đã mã hóa vẫn có thể sửa hoặc xóa nó.

4- Sử Dụng VIEW Để Thay Đổi Dữ Liệu:

Bạn có thể sử dụng View trong các phát biểu Insert, Update, Delete để thay đổi dữ liệu trên Table nguồn của View.

- **Các hạn chế :**
  - Không thể sử dụng lệnh INSERT hay DELETE trên View có sử dụng phát biểu INNER JOIN. (Trừ khi có sử dụng INSTEAD OF Trigger)
  - Không thể sử dụng lệnh INSERT nếu View có khai báo cột tính toán hoặc chứa những cột có ràng buộc NOT NULL mà không khai báo giá trị mặc định

Ví dụ :

```
CREATE VIEW vwCTHD (MaHD, MaMH, SoLuong)
        AS
        SELECT MaHD, MaSP, SL
        FROM CTHD
        Where  MaHD = 10248
Go
Insert vwCTHD(MaHD,MaMH,SoLuong)  Values(10248,1,20)
Insert vwCTHD(MaHD,MaMH,SoLuong)  Values(10249,1,20)
```

- Bạn có thể ràng buộc dữ liệu được cập nhật phải thỏa mãn điều kiện Where của View bằng cách khai báo thêm Từ khóa : **WITH CHECK OPTION**

```
CREATE VIEW vwCTHD (MaHD, MaMH, SoLuong)
        AS
        SELECT MaHD, MaSP, SL
        FROM CTHD
        Where  MaHD = 10248
        WITH CHECK OPTION
Go
UPDATE vwCTHD SET soluong = soluong + 10
INSERT vwCTHD(MaHD,MaMH,SoLuong)  VALUES(10248,1,20)
/* lệnh chèn sau đây sẽ bị lỗi : The attempted insert or update failed because the
        target view either specifies WITH CHECK OPTION or spans a view that
        specifies WITH CHECK OPTION and one or more rows resulting from
        the operation did not qualify under the CHECK OPTION constraint.
INSERT vwCTHD(MaHD,MaMH,SoLuong)  VALUES(10249,1,20)
```

5- Thay đổi định nghĩa View:

Sử dụng phát biểu ALTER VIEW để thay đổi định nghĩa của View nhưng vẫn duy trì giấy phép cho View. Nếu bạn xóa View, sau đó tạo lại nó, bạn phải cấp lại các giấy phép cho nó.

Syntax

ALTER VIEW V*iewName* [(*column* **,** ...)]
[WITH ENCRYPTION]
AS
select_statement
[WITH CHECK OPTION]

6- Xóa View:

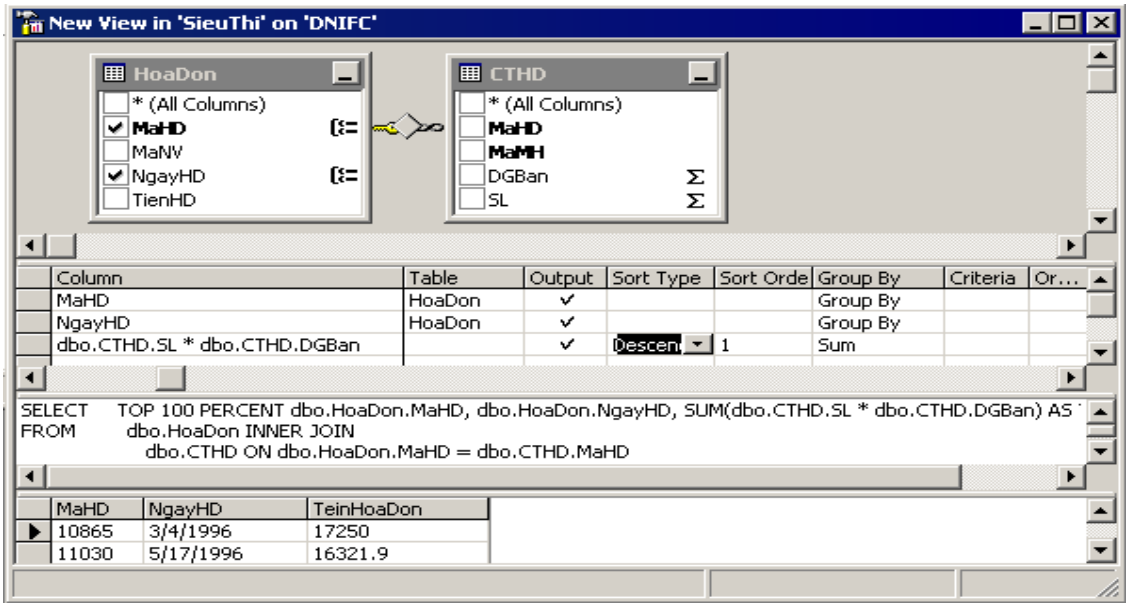
DROP VIEW <view_name>

Ví dụ:

```
USE SieuThi
Go
IF EXISTS (SELECT TABLE_NAME FROM INFORMATION_SCHEMA.VIEWS
        WHERE TABLE_NAME = 'vwCTHD')
    DROP VIEW vwCTHD
GO
CREATE VIEW vwCTHD (MaHD, MaMH, SoLuong)
    AS
    SELECT MaHD, MaSP, SL
        FROM CTHD
        Where  MaHD = 10248
```

III- SỬ DỤNG ENTERPRISE MANAGER

Click phải vào đối tượng **Views** trên khung trái và chọn **New View…**



Click nút để chọn Table|View nguồn.

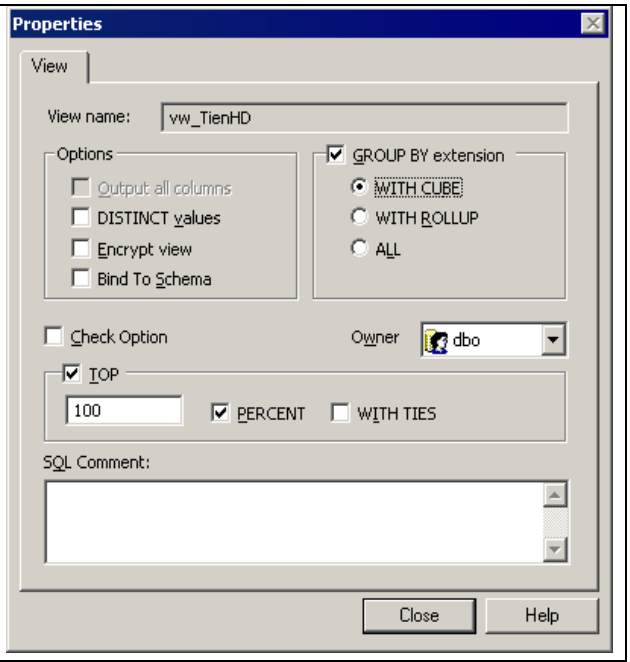
Click nút để thêm cột Group By

Tạo và khai báo thuộc tính cho các Field

Click nút để thực hiện truy vấn

Click nút để lưu và đặt tên View

Click nút để mở cửa sổ thuộc tính của View



b) WITH CUBE

Tổng hợp theo từng nhóm các cột trong GROUP BY. Đồng thời tổng hợp trên tất cả mẫu tin.

Ví dụ:

```
SELECT MaNV, NgayHD, SUM(SL * DGBan) AS TienHD,
        COUNT(HoaDon.MaHD) AS SoHD
        FROM        HoaDon INNER JOIN CTHD ON HoaDon.MaHD = CTHD.MaHD
        WHERE     (NgayHD = CONVERT(DATETIME, '1995-09-25 00:00:00', 102))
        GROUP BY MaNV, NgayHD WITH CUBE
        ORDER BY MaNV, NgayHD
```

MaNV	NgayHD	TienHoaDon	SoHD
	<NULL>	2508	6
<NULL	9/25/1995	2508	6
3	<NULL>	1422	3
3	9/25/1995	1422	3
6	<NULL>	1086	3
6	9/25/1995	1086	3

c) WITH ROLLUP

WITH ROLLUP tương tự WITH CUBE, nhưng chỉ tổng hợp từng nhóm các cột từ trái sang phải không tổng hợp chéo các cột.

Ví dụ:

```
SELECT  MaNV, NgayHD, SUM(SL * DGBan) AS TienHD,
        COUNT(HoaDon.MaHD) AS SoHD
        FROM HoaDon INNER JOIN CTHD ON HoaDon.MaHD = CTHD.MaHD
        WHERE (NgayHD = CONVERT(DATETIME, '1995-09-25 00:00:00', 102))
        GROUP BY MaNV, NgayHD WITH ROLLUP
        ORDER BY MaNV, NgayHD
```

MaNV	NgayHD	TienHoaDon	SoHD
<NULL	<NULL>	2508	6
3	<NULL>	1422	3
3	9/25/1995	1422	3
6	<NULL>	1086	3
6	9/25/1995	1086	3

CHƯƠNG V: STORED PROCEDURES

I- Khái niệm:

Stored Procedure là một đối tượng được xây dựng bởi những phát biểu của SQL Server, và được lưu trữ trong CSDL với 1 tên phân biệt. Mỗi Stored Procedure có thể chứa nhiều câu lệnh SQL.

Stored Procedure tương tự như các thủ tục trong các NNLT khác vì chúng có thể:

- Chứa các phát biểu và có thể gọi các thủ tục khác (Nested) hoặc đệ qui (Recursive).
- Có thể nhận các tham số .
- Có thể trả về các giá trị tính toán được trong thủ tục.

Khi muốn thực hiện các phát biểu đã lưu trữ trong thủ tục ta chỉ cần gọi tên thủ tục.

Chú ý: Không sử dụng phát biểu CREATE tạo đối tượng bên trong Stored Procedure.

II- Tạo Stored procedure Bằng Lệnh CREATE:

Cú pháp:

```
CREATE PROC[EDURE] <TênTT> [; number]
        [Danh Sách tham số]
        [WITH   RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION]
        AS
                [Danh Sách biến cục bộ]
                <sql_statement>
        GO
```

1- Biến cục bộ :

a) Khai báo biến cục bộ:

Trong T-SQL, biến chứa giá trị trung gian phải được khai báo trước khi sử dụng:

DECLARE @V*arName data_type* [,*...n*]

- Tất cả các biến cục bộ phải bắt đầu bằng ký hiệu @ trước tên của nó.
- **Phạm vi biến**: Biến khai báo trong một strored procedure hoặc một gói lệnh chỉ khả dụng bên trong vùng đã khai báo nó.
- Trong T-SQL không có khái niệm biến toàn cục.

Ví dụ: DECLARE @Counter int

DECLARE @FirstName varchar(25)

Hay khai báo trên cùng dòng:

DECLARE @FirstName varchar(25), @Counter int

- Sau khi khai báo biến có giá trị NULL

b) Gán giá trị cho biến cục bộ:

Cách 1: **SET** @V*arName* = <E*xpression*>

Ví dụ: Declare @Ngay DateTime

```
set @Ngay ='2005/5/15'
print @Ngay
Set @Ngay = @Ngay + 1
print @Ngay
```

Cách 2: Dùng câu lệnh Select:

SELECT @V*arName* = <Expression> [FROM …]

Nếu query trả về nhiều record thì giá trị trên record cuối cùng sẽ được gán cho biến.

```
DECLARE @intMaNV int
SELECT @intMaNV = 5 + 6
SELECT @intMaNV = MAX(MaNV) FROM SieuThi..NhanVien
```

- *Nếu SELECT trả về nhiều dòng, thì biến được gán giá tri từ dòng cuối cùng trả về bởi Select.*
- *T-SQL có tính năng tự động chuyển kiểu. Do đó, có khả năng bị mất dữ liệu. Bạn có thể dùng hàm để chuyển đổi kiểu:*

```
CAST ( expression AS data_type )
CONVERT ( data_type [ ( length ) ] , expression [ , style ] )
```

Ví dụ: Tạo thủ tục thực hiện truy vấn MaHD, tiền bán của từng hóa đơn.

2- Có 2 loại tham số : Input và Output

Cú pháp : **@TênTS datatype [= default][OUTPUT]**

- Có thể khai báo 1024 tham số. Các tham số cách nhau bởi dấu phẩy. Không dùng Kiểu hình ảnh.
- Nếu cung cấp giá trị mặc định cho tham số, thì có thể thực hiện thủ tục mà không chỉ định giá trị cho tham số.

Ví dụ: Tạo Stored Procedure Chọn 10 mặt hàng có đơn giá cao nhất gồm các cột: Tên và đơn giá.

```
Use SieuThi
Go
CREATE PROCEDURE MuoiMHCaoNhat
AS
        SET ROWCOUNT 10
        SELECT MatHang.TenMH , MatHang.DonGia FROM MatHang
                ORDER BY MatHang.DonGia DESC
Go
```

- **SET ROWCOUNT { n | *@number_var* }**

Chỉ chọn n dòng đầu tiên khi rút trích dữ liệu..

Để bỏ chỉ định này và cho hiện hết các dòng : SET ROWCOUNT 0

Ví dụ: Xây dựng thủ tục trả về n mặt hàng đầu tiên có đơn giá cao nhất:

```
CREATE PROCEDURE MuoiMHCaoNhat
        @n smallint = 10
AS
        SET ROWCOUNT @n
        SELECT MatHang.TenMH , MatHang.DonGia
                FROM MatHang
                ORDER BY MatHang.DonGia DESC
```

Hay

```
CREATE PROCEDURE MuoiMHCaoNhat
@n smallint = 10
AS
        Exec ('SELECT Top ' + @n + ' MatHang.TenMH , MatHang.DonGia
        FROM MatHang
        ORDER BY MatHang.DonGia DESC')
```

Chú ý: Sử dụng 2 dấu nháy đơn để bao các biến chuỗi hoặc ngày

Ví dụ: Declare @str Char(10)

Set @str = 'Chao Ban'

Exec ('Select ''' + @str + '''')

3- ;number:

Dùng nhóm các thủ tục có cùng tên và phân biệt nhau bởi số thứ tự *number*. Khi đó, tên của thủ tục có thêm số thứ tự đã khai báo: **orderproc**;1, **orderproc**;2.

Để xóa nhóm thủ tục này bạn chỉ cần dùng 1 lệnh xóa với tên của nhóm.

III- THI HÀNH STORED PROCEDURE:

1- Thi hành stored procedure không có tham số:

Sau khi tạo xong, bạn có thể gọi thủ tục trong QA như sau:

MuoiMHCaoNhat

hoặc EXEC MuoiMHCaoNhat

2- Thực hiện Stored Procedures có Tham số

Có 2 các truyền giá trị cho tham số: Truyền theo vị trí và truyền theo tên tham số.

Cách 1: Truyền trị theo vị trí :

Phải liệt kê theo thứ tự từ trái sang phải. Chỉ có thể bỏ qua những tham số đã gán giá trị mặc định cuối cùng. Ví dụ, thủ tục có 5 tham số đã gán mặc định, Bạn có thể bỏ qua tham số 4 và 5 nhưng không thể bỏ qua 4 và gán trị cho 5.

[EXEC[UTE]] TênTT giátrị [OUTPUT] [,…*n*]

Example

EXEC ThemHD 10, '2005/05/15

Cách 2: Truyền theo tên tham số:

[EXEC[UTE]] *procedure_name* @TênTS = *value [OUTPUT]* [,…*n*]

3- Trả về giá trị với tham số Output

```
CREATE PROC spTBCong
        @Avg smallint  Output,
        @A smallint=3, @B smallint = 2,
AS
        Select @Avg = (@A + @B) / 2
Go
```

Để nhận giá trị Avg: đầu tiên bạn phải khai báo một biến, sau đó chạy thủ tục.

Ví dụ: Declare @KQ smallint

EXEC spTBCong @KQ OUTPUT, 10

Select 'The Average Score is : ', @KQ

Hay

Declare @KQ smallint

EXEC spTBCong @A = 10, @B = 9, @AVG = @KQ **OUTPUT**

Select 'The Average Score is : ', @KQ

IV- Các Phát Biểu Điều Khiển:

1- Phát biểu RETURN [<integer value to result>]:

Dùng kết thúc thủ tục và trả về nơi gọi một số nguyên nào đó, mặc định là 0.

Ví dụ:
```
CREATE PROCEDURE MHCaoNhat AS
SELECT Top 1 With Ties MatHang.TenMH , MatHang.DonGia
    FROM MatHang
    ORDER BY MatHang.DonGia DESC
Return @@RowCount
```

Để nhận giá trị Return: đầu tiên bạn phải khai báo một biến và sau đó chạy thủ tục.

```
Declare @SoMT smallint
EXEC @SoMT = MHCaoNhat
Select 'So Mat Hang : ', @SoMT
Go
```

Chú ý: Khi thực hiện 1 câu lệnh SQL hay Stored Proc có tham số, tất cả các dấu nháy đơn (') dùng để đóng và mở cho một chuỗi hoặc ngày tháng đều bắt buộc phải có.

2- Phát biểu If .. Else:

```
IF <Boolean Expression>
    BEGIN <Các lệnh> END
ELSE
    BEGIN <Các lệnh> END
```

Ví dụ: Tạo thủ tục liệt kê hóa đơn theo tháng và năm.

```
CREATE PROC spHD  ( @Thang SmallInt, @Nam SmallInt )
As
    If (@Thang <1 Or @Thang>12)
    begin
        Print 'Thang Sai'
        return 0
    end
    Select * From Orders
    Where Year(NgayLap) = @Nam And Month(NgayLap)= @Thang
Go
```

3- Hàm CASE:

Trả về giá trị chọn lựa từ nhiều điều kiện khác nhau

CASE có 2 dạng :

- Hàm CASE đơn giản so sánh 1 biểu thức với tập các biểu thức đơn giản để xác định kết quả.

```
CASE input_expression
   WHEN when_expression THEN result_expression
      [ ...n ]
   [ELSE else_result_expression  ]
END
```

Ví dụ: Simple Case

```
Create Proc spr_ThuTrongTuan
(@D As SmallDateTime, @Thu VarChar(10) OUTPUT)
As
    Set @Thu = Case Datepart(w, @D)
        When 1 Then 'Chu Nhat'
```

```
            When 2 Then 'Thu Hai'
            When 3 Then 'Thu Ba'
            When 4 Then ‘Thu Tu’
                    When 5 Then ‘Thu Nam’
                    When 6 Then ‘Thu Sau’
            Else 'Thu Bay'
        End
    Go
```

- Hàm CASE tìm kiếm (searched CASE) kiểm tra các biểu thức điều kiện để xác định kết quả.

```
CASE
   WHEN Boolean_expression THEN result_expression
     [ ...n ]
   [ELSE else_result_expression ]
END
```

Ví dụ: Xây dựng thủ tục trả về số ngày trong tháng khi biết tháng và năm

```
Create Proc spr_Days (@Thang Int, @Nam Int)
As
Declare @SN Int
Set @SN = Case
            When @Thang In (1,3,5,7,8,10,12) Then 31
            When @Thang In (4,6,9,11) Then 30
            When @Nam % 4 = 0 Then 29
            Else 28
        End
Return @SN
Go
```

- Có thể lồng Case:

```
Create Proc spr_NgayTrongThang (@Thang Int, @Nam Int)
As
Declare @SN Int
Set @SN = Case
            When @Thang In (1,3,5,7,8,10,12) Then 31
            When @Thang In (4,6,9,11) Then 30
            Else Case
                    When @Nam % 4 = 0 Then 29
                    Else 28
                 End
        End
Return @SN
Go
```

Thực hiện thủ tục:

```
Declare @SN int
Exec @SN = spr_ngaytrongthang 2,2000
```

```
print 'So ngay trong thang 2/2002 la ' + Cast(@SN As Char)
```

4- Phát biểu WHILE :

WHILE được dùng nhiều trong kiểu dữ liệu CURSOR. Thông thường WHILE thường sử dụng để duyệt từ mẫu tin đầu tiên đến mẫu tin cuối cùng hoặc ngược lại.

```
WHILE <boolean expression>
BEGIN
    <Các phát biểu T-SQL>
    [CONTINUE] [BREAK]
END
```

Ví dụ: Tạo thủ tục tính tổng S = 1 + 3 + 5 +… + (2N-1)

```
Create Proc TongLe ( @N int )
AS
    DECLARE @I int, @S int
    SET @I = 1
    SET @S = 0
    WHILE @I <= 2*@N -1
    BEGIN
    --WAITFOR DELAY '00:00:00.001'
        Set @S = @S + @I
        Set @I = @I + 1
    END
    Return @S
```

Gọi thực hiện:

```
declare @s int
Exec @s = tongle 3
print ‘Tong la : ‘ + @s
```

Phát biểu WAITFOR : WAITFOR DELAY <’time’> | TIME <’time’>

Khi sử dụng WHILE, có thể dùng thêm lệnh WAITFOR với thời gian cho trước.

Tham số DELAY dùng chỉ thị khoản thời gian chờ nhất định. Thời gian tính bằng ‘h:m:s’, lớn nhất là 24 giờ.

Tham số TIME: chỉ thị hệ thống chờ cho đến thời gian được chỉ định.

V- Sửa Xóa Thủ Tục Lưu Trữ:

1- Thu thập thông tin về Thủ tục được lưu:

Để xem nội dung câu lệnh Create Proc của thủ tục được lưu, bạn có thể dùng EM hay chạy thủ tục hệ thống **sp_helftext** với tên thủ tục được lưu.

sp_helptext *spAuthors*

2- Đổi Tên TTLT: sp_Rename <OldName>, <NewName>

3- Xóa Thủ tục được lưu:

Giống như xóa các đối tượng khác, chúng ta lại sử dụng phát biểu Drop.

DROP PROC[EDURE] <tên Proc>

4- Thay đổi thủ tục lưu trữ:

ALTER PROC[EDURE] procedure_name [;number]

```
[Danh sách tham số]
[WITH RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION]
AS
        sql_statement
GO
```

VI- Xử lý lỗi trong SP:

Để tăng hiệu năng của stored procedures, Bạn nên chứa mã trả về trạng thái thành công hay thất bại của thủ tục. Nếu thất bại thì thông báo về lỗi đã xảy ra.

Để kiểm tra lỗi bạn sử dụng hàm @@Error. Để trả về thông tin về lỗi bạn có thể sử dụng lệnh Return hoặc lệnh Raiserror.

1- Hàm @@ERROR

Hàm *@@ERROR* chứa mã lỗi cho phát biểu vừa thực hiện. Nó tự động xóa và đặt lại mã lỗi với mỗi phát biểu được thực hiện. Giá trị 0 nếu phát biểu thành công.

Bạn có thể dùng hàm @@ERROR để xác định mã lỗi hoặc làm điều kiện kết thúc thủ tục.

Ví dụ: Chèn Hoá đơn mới, nếu ngày lập là chủ nhật thì chuyển thành ngày thứ 2

```
CREATE PROC spThemHD  ( @MaHD SmallInt, @NgayLap SmallDateTime )
    As
        If (IsDate(@NgayLap) = 0)
            Return 0
        If DatePart(w, @NgayLap) = 1
        Set @NgayLap = @NgayLap + 1
        Insert Into HoaDon(MaHD, NgayLap) Values(@MaHD, @NgayLap)
        If (@Error <> 0)
        Begin
            RaiseError('Loi them hoa don moi',10,1)
            Return  0
        End
```

2- Lệnh RAISERROR

Phát biểu *RAISERROR* dùng xuất thông báo lỗi lấy từ table **sysmessages** hoặc một thông báo với nội dung nào đó

RAISERROR ({ *msg_id* | *msg_str* } { **,** *severity* **,** *state* } [**,** *argument* [**,***...n*]] **)**
[WITH *option* [**,***...n*]]

Arguments

- *msg_id:* Mã số > 50000 của dòng thông báo do người dùng tạo trên table **sysmessages** bằng thủ tục sp_AddMessage
- *msg_str:* Nội dung thông báo lỗi cần xuất. .
- *severity:* Mã số chỉ định mức độ nghiên trọng của lỗi.

  Mức độ từ 0-19 được xem như là thông tin phản hồi. Mức từ 0 đến 16: lỗi phát sinh do dữ liệu. Mức 10 là lbáo cáo trạng thái. Mức từ 11 - 16 là những lỗi có thể được hiệu chỉnh bởi user.

  Mức từ 20 - 25 chỉ định các lỗi tác hại từ ứng dụng không thể phục hồi. Nếu thông báo mức này, kết nối từ Client đến Server sẽ bị ngắt.

- *State:* là số nguyên từ 1 đến 127 biểu diễn thông tin về trạng thái thỉnh cầu của lỗi., nhằm chỉ rõ lỗi thuộc nhóm nào trong hệ thống lỗi. Giá trị phủ nhận của state là 1.
- *Argument: Là tham số dùng thay thế vào những biến được sử dụng trong chuỗi msg_str* hoặc thông báo tương ứng với *msg_id*. Mỗi tham số thay thế có thể là 1 biến cục bộ hoặc giá trị thuộc kiểu: **int1**, **int2**, **int4**, **char**, **varchar**, **binary**, or **varbinary**.

  RAISERROR ('The level for job_id:%d should be between %d and %d.',
  16, 1, @@JOB_ID, @@MIN_LVL, @@MAX_LVL)
- *Option:* có thể chứa các giá trị sau:

Value	Description
LOG	Ghi lỗi vào file nhật ký.
NOWAIT	Gởi thông báo ngay lập tức đến client.
SETERROR	Đặt giá trị msg_id hay 50000 cho hàm **@@**ERROR.

Ví dụ: RAISERROR('Invalid member number', 10, 1) WITH LOG

Khi xuất hiện lỗi, mã lỗi được đặt trong hàm @@ERROR. Mặc định @@ERROR được đặt bằng 0 cho các lỗi có severity từ 1 đến 10.

3- Thêm Thông báo lỗi vào bảng sysmessages:

a- Dùng thủ tục sp_addmessage để thêm vào bảng sysmessages. .

sp_AddMessage @msgnum = *msg_id* **,**
@severity = *severity* **,**
@msgtext = '*msg***'**
[, @lang = '*language***']**
[, @with_log = '*True* | *False***']**
[, @replace = 'REPLACE']

msg_id: phải có giá trị từ **50.001** trở lên.

'REPLACE': dùng chỉ định lệnh sẽ thay thế msg_id nếu msg_id đã có trong bảng.

Example: USE master

EXEC sp_addmessage @msgnum = 60000, @severity = 16,
@msgtext = 'Kiểu dữ liệu không hợp lệ' , @lang = 'us_english', @with_log = 'true'

b- Để xóa thống báo lỗi : sử dụng sp_dropmessage

sp_DropMessage @msgnum = *msg_id*

Chú thích – Comments

Có 2 cách chú thích: chú thích trên cùng dòng lệnh hoặc chú thích trên nhiều dòng.

Chú thích trên cùng dòng với một phát biểu - In-Line Comments

Sử dụng 2 dấu trừ (--) trước lời chú thích.

Chú thích trên nhiều dòng – Block Comments

Bắt đầu bởi /* và kết thúc bởi dấu */

Chương VI: Chuyển Tác Và Bẫy Lỗi

I- Các Phát Biểu Chuyển Tác – Transactions:

1- Khái niệm :

Transactions dùng đảm bảo rằng các lệnh thay đổi dữ liệu được xử lý trọn vẹn. Nếu có một lệnh nào đó trong Transaction bị lỗi thì phải bãi bỏ các lệnh trong Transaction và phục hồi lại toàn bộ dữ liệu đã bị thay đổi bởi các lệnh trong Transaction trước đó.

Có 2 cách khai báo một Transaction : Khai báo rõ ràng hoặc khai báo ngầm định

2- Các Phát Biểu đóng gói một Transaction :

a- Bắt đầu một chuyển tác:

Syntax : BEGIN TRAN[SACTION] [*transaction_name*]

Có thể không cần đặt tên Transaction. Tuy nhiên nên đặt tên Transaction để dễ đọc.

b- Xác Nhận Kết Thúc Thành Công Một Chuyển Tác:

Syntax: COMMIT TRAN[SACTION] [*transaction_name*]

c- Đánh dấu vị trí trong chuyển tác:

Lưu vị trí chuyển tác

Syntax: SAVE TRAN[SACTION] <save_*name*>

d- Kết thúc không thành công một chuyển tác:

Khi gặp lệnh này, tất cả những lệnh được thực hiện trong Transaction bắt đầu từ lệnh Begin Tran hoặc từ vị trí đánh dấu trong chuyển tác sẽ bị bãi bỏ

Syntax: ROLLBACK TRAN[SACTION] [*transaction_name*] | [save_name]

Ví dụ : Tạo SP thêm một sinh viên mới trong table SinhVien và thêm vào table SV_Detai MSDT mà sinh viên thực hiện.

Nếu MSDT không tồn tại chỉ bỏ thao tác chèn mẫu tin mới cho Table SV_DeTai.

```
CREATE PROCEDURE ThemSVDT
   @MSDT Char(6), @MSSV  char(6),
   @TenSV VarChar(30), @Lop char(6),
   @SoDT VarChar(10)= NULL, @DiaChi Char(10)=NULL
AS
   BEGIN TRANSACTION VT1
   INSERT SinhVien (MSSV,TenSV,SoDT, Lop, DiaChi)
   VALUES (@MSSV, @TenSV, @SoDT, @Lop, @DiaChi)
      IF @@ERROR <> 0
         BEGIN
            ROLLBACK TRAN VT1
            RETURN 0
         END

   SAVE TRAN VT2
   INSERT SV_DETAI(MSSV, MSDT)
   VALUES (@MSSV, @MSDT )
```

```
        IF (@@ERROR <> 0)  ROLLBACK TRAN VT2
    COMMIT TRANSACTION VT1
GO
```

3- Chuyển tác ngầm định: (Implicit Transactions)

Bất kỳ transaction nào mà bắt đầu, kết thúc hoặc roll back bằng lệnh BEGIN TRANSACTION, COMMIT TRANSACTION, or ROLLBACK TRANSACTION đều là *Explicit Transaction*. Bạn có thể thực hiện một Implicit Transaction bằng cách khai báo :

Syntax: SET IMPLICIT_TRANSACTIONS {ON | OFF}

Khi đặt chế độ **Implicit Transaction** là ON, Các phát biểu sau đây sẽ tự động bắt đầu một Transaction:

- SELECT, INSERT, UPDATE, DELETE
- ALTER TABLE
- TRUNCATE TABLE
- OPEN, FETCH
- GRANT, REVOKE

Khi đặt ON, thì ở cuối các Tracsaction cần phải có lệnh Commit hoặc Roll Back. Nếu không thì các lệnh trong Transaction và tất cả dữ liệu đã thay đổi sẽ bị bỏ qua khi người dùng kết thúc kết nối.

Nếu cài đặt là OFF (default). Mỗi phát biểu tự động xác nhận nếu không bị lỗi.

II- Bẫy Lỗi – TRIGGERS

1- Khái niệm:

Trigger là dạng đặt biệt của SP, dùng khai báo ràng buộc dữ liệu cho một table, View và tự động thực hiện khi một trong 3 phát biểu Insert, Update, Delete thay đổi dữ liệu trên table đó. Trigger không được gọi trực tiếp như SP, không có tham số và giá trị trả về như SP.

Trigger chỉ được thực hiện khi phát biểu cập nhật đã thoả mãn các ràng buộc đã khai báo trên Table. Lợi ích chính của triggers là chúng có thể chứa các xử lý phức tạp trên các table có dữ liệu liên quan với table đang cập nhật.

Trigger có thể chứa phát biểu ROLLBACK TRAN ngay cả khi không có phát biểu BEGIN TRAN. Trong trường hợp phát biểu ROLLBACK TRANSACTION bên trong 1 Trigger được thực hiện:

- ☑ Nếu trigger này được kích hoạt bởi 1 phát biểu cập nhật từ bên trong một transaction khác, thì toàn bộ Transaction đó bị bãi bỏ.
- ☑ Nếu trigger được kích hoạt bởi 1 phát biểu cập nhật từ bên trong một gói, thì sẽ bãi bỏ toàn bộ gói.

Dựa vào ứng dụng của Trigger, có 3 loại Trigger như sau: Insert Trigger; Update Trigger; Delete Trigger

2- Tạo Trigger cho Table:

Cú pháp Tạo Trigger:

```
CREATE TRIGGER <trigger_name> ON <table name>
        [WITH ENCRYPTION]
        AFTER | FOR DELETE, INSERT, UPDATE
AS   <Các phát biểu T-sql>
```

- trigger_name : Tên Trigger phải phân biệt.
- ON <tablename> : tên table mà Trigger sẽ thực hiện. Không sử dụng Trigger cho View.

- WITH ENCRYPTION : Mã hóa Trigger, không cho xem và sửa đổi câu lệnh tạo Trigger..
- FOR DELETE, INSERT, UPDATE

  Dùng chỉ định những phát biểu cập nhật nào nào trên Table sẽ kích hoạt Trigger.

  Khi thực hiện Trigger, SQL sẽ tạo các bảng tạm: INSERTED và DELETED
  - **Khi Insert** mẫu tin mới vào Table thì mẫu tin mới đó cũng lưu trong table INSERTED
  - **Khi Delete** mẫu tin trong table: Thì các mẫu tin bị xoá đó được di chuyển sang table Deleted.
  - **Khi Update** mẫu tin trong table: thì table đó và table Inserted đều chứa mẫu tin có nội dung mới, còn Deleted chứa mẫu tin có nội dung cũ.

  Bạn không thể thay đổi dữ liệu trên các table DELETED VÀ INSERTED. Nhưng bạn có thể dùng 2 table này để xử lý các mẫu tin trên các table liên quan. Ngoài ra, trong trigger Insert và Update, bạn có thể thay đổi nội dung của các mẫu tin mới bằng lệnh Update trên table có trigger.
- AS : Từ khóa bắt đầu các hành động bên trong Trigger. Trigger có thể chứa hầu hết các lệnh của T-SQL ngoại trừ một số lệnh sau:
  - Các lệnh CREATE, ALTER, and DROP.
  - TRUNCATE TABLE
  - SELECT INTO (because it creates a table)

Chú ý:

- *Chủ của table và những thành viên có Role db_owner, db_ddladmin, và sysadmin có thể tạo và xoá triggers. Các permissions không thể sang nhượng. Hơn nữa, người tạo Trigger phải có quyền thực hiện tất cả phát biểu trên các tables.*
- *Triggers không thể tạo view và table tạm (temporary tables), nhưng chúng có thể tham chiếu đến các views và temporary tables.*
- *Các lệnh INSERT, UPDATE, hoặc DELETE có thể tác động trên nhiều dòng. Để biết được số dòng bị tác động, sử dụng hàm @@ROWCOUNT bên trong Trigger*

a- Sử dụng Trigger để ràng buộc toàn vẹn dữ liệu:

Ví dụ: Khi xóa hay thay đổi MSGV trong GIAOVIEN_HD, nếu giáo viên đó là chủ tịch hội đồng thì báo lỗi kết thúc.

```
Create Trigger trg_XoaHoiDongGV On HoiDong_GV
For Delete, Update
As
If Exists(Select 1 From HoiDong a, Deleted b
        Where a.MSHD = b.MSHD And a.MSGVCTHD = b.MSGV)
Begin
        Raiserror('Khong xoa hay thay doi giao vien CTHD',16,1)
        RollBack Tran
End
```

Kiểm tra Trigger:

```
Delete From HoiDong_GV Where MsHD = 1 And MsGV = 1
Go
Update HoiDong_GV Set MSGV = 5 From HoiDong_GV a, HoiDong b
        Where a.MSHD = b.MSHD And a.MSGV = b.MSGVCTHD And b.MSHD = 1
```

```
Go
```

b- Sử dụng trigger để kiểm tra RB giá trị

Ví dụ: Kiểm tra RB : Một hội đồng không có quá 10 đề tài.

```
Create Trigger trg_ThemHDDT On  Hoidong_DT
        For Insert, Update
As
        If (Select Count(a.mshd) From HoiDong_DT a, INSERTED b
                Where a.mshd = b.mshd) > 10
        Begin
                RaisError ('Khong the > 10', 16, 1)
                RollBack Tran
        End
        Go
```

Ví dụ: 2 Hội đồng trong cùng 1 ngày không thể trùng phòng

```
Create Trigger trg_HoiDong On HoiDong
For Insert, Update
As
If Exists(Select 1 From HoiDong a, Inserted b
        Where a.MSHD = b.MSHD
        And a.NgayHD = b.NgayHD And a.Phong = b.Phong)
Begin
        Raiserror('Hai hoi dong cung 1 ngay khong trung phong',16,1)
        RollBack Tran
End
Go
Update HoiDong Set NgayHD = '2001/10/30', Phong =2 Where MSHD = 4
```

Ví dụ: Một giáo viên không thể vừa là giáo viên phản biện vừa là giáo viên hướng dẫn đề tài.

Chú ý: Bạn có thể định nghĩa nhiều Trigger (khác tên nhau) cho cùng một hành động. Khi đó thứ tự thực hiện các trigger đó được xác định dựa trên thứ tự tạo ra chúng. Bạn có thể thay đổi thứ tự thực hiện mặc định này bằng SP: **Sp_SetTriggerOrder**

sp_SetTriggerOrder trg_UpdateAction2, **First**, 'Update'

sp_SetTriggerOrder trg_UpdateAction1, **Last**, 'Update'

SP này chỉ có thể chỉ định trigger nào được thực hiện đầu tiên và Trigger nào được thực hiện cuối cùng. Các Trigger còn lại sẽ thực hiện theo thứ tự tạo ra chúng.

3- Tạo Trigger cho View:

```
CREATE TRIGGER <trigger_name> ON <View name>
        [WITH ENCRYPTION]
        INSTEAD OF DELETE | INSERT | UPDATE
        AS   <Các phát biểu T-sql>
```

Trong các phiên bản trước phiên bản 2000, bạn không thể dùng các lệnh Insert, Update, Delete để cập nhật dữ liệu trên View.

Trong phiên bản 2000, bạn có thể cập nhật dữ liệu trên một bảng nguồn của View bằng tên của View. Trường hợp bạn muốn dùng 1 lệnh, cập nhật trên nhiều bảng nguồn của View thì phải tạo Trigger INSTEAD OF cho các View đó.

Không giống như AFTER triggers, Chỉ có thể INSTEAD OF cho mỗi lệnh hoặc INSERT, hoặc UPDATE, hoặc DELETE. Các Trigger INSTEAD OF sẽ thực hiện trước các AFTER triggers đã cài đặt cho các Table

Ví dụ: Tạo View liệt kê danh sách đề tài và sinh viên thực hiện đề tài đó

```
CREATE VIEW vwSVDeTai
AS
SELECT A.MSDT, A.TENDT, B.MSSV, NULLIF(C.TENSV, NULL) As TENSV
FROM  DETAI A INNER JOIN SV_DETAI B ON A.MSDT = B.MSDT
      INNER JOIN SINHVIEN C ON B.MSSV = C.MSSV
```

Để thêm đề tài mới trong table DeTai và sinh viên thực hiện đề tài đó trong table SV_DeTai, thay vì thực hiện 2 lệnh Insert trên từng Table, bạn có thể thực hiện 1 lệnh thông qua đối tượng View đã tạo.

```
INSERT vwSVDeTai(msdt, Tendt, mssv)
      Values( 97014, 'Ma Hoa Du Lieu', '01th01')
```

Để làm được điều này, bạn phải tạo Trigger Instead Of Insert cho View

```
CREATE TRIGGER tgvwSVDeTai ON vwSVDeTai
      INSTEAD OF INSERT
AS
      If (Select Count(*) From Inserted) > 0
      Begin
            Insert detai(msdt, Tendt) Select A.msdt,A.tendt  From Inserted A
            Insert SV_detai(mssv,msdt) Select A.mssv,A.msdt From Inserted A
      End
```

Chú ý: Việc sử dụng hàm NULLIF(C.TENSV, NULL) As TENSV để tránh trường hợp lỗi TenSV không được NULL vì lúc này TenSV là một Field tính toán.

4- Kiểm Tra Cột Được Cập Nhật :

Hàm **Update(<Column Name>)** : Dùng kiểm tra <Column Name> có được cập nhật dữ liệu hay không.

Hàm **Columns_Update():** Trả về các Byte cho biết những cột nào đã được cập nhật. Mỗi Bit trong các Byte này tương ứng với một cột trong Table theo thứ tự từ trái qua phải. Cột nào được cập nhật thì Bit tương ứng có giá trị 1. Sử dụng các toán tử Bitwise để kiểm tra cột nào được cập nhật.

^ (Bitwise Exclusive OR), & (Bitwise AND), | (Bitwise OR)

5- Disabling or Enabling a Trigger:

ALTER TABLE *table* ENABLE | DISABLE TRIGGER ALL | *trigger_name*[,…*n*]

6- Hiệu chỉnh Trigger:

Bạn có thể thay đổi các lệnh cần thực hiện cũng như hành động cập nhật mà Trigger sẽ được gọi thực hiện.

ALTER TRIGGER *trigger_name* …

7- Xóa Trigger:

DROP TRIGGER {*trigger*} [,…*n*]

Nếu xóa một table thì tất cả Triggers của nó cũng bị xóa.

Quyền xóa :sysadmin, db_owner and db_ddladmin roles.

Chương 5 : Kiểu Con Trỏ (SQL Cursor)

I- Tổng Quan:

1- Khái niệm:

Cursor là kiểu dữ liệu cho phép truy xuất đến trên từng mẫu tin trong tập kết quả trả về bởi câu lệnh Select. Ngoài ra, bạn có thể sử dụng các phát biểu Update hoặc Delete để cập nhật hay xóa mẫu tin hiện hành trên các bảng cơ sở của Select bằng mệnh đề WHERE CURRENT OF <Tên Cursor> .

2- Các thao tác chung trên Cursor:

Khai báo cursor : DECLARE <cursor_name> CURSOR FOR <lệnh Select>

Mở cursor : OPEN <cursor_name>

Sau lệnh mở cursor, con trỏ mẫu tin hiện hành nằm ở vùng BOF.

Xử lý mẫu tin trên cursor:

Di chuyển mẫu tin hiện hành: FETCH NEXT FROM *cursor_name*

Sử dụng phát biểu Update hoặc Delete để cập nhật hay xóa mẫu tin hiện hành

Đóng cursor: CLOSE <tên cursor>

Hủy bỏ cursor: DEALLOCATE <TÊN CURSOR>

Ví dụ : Điền số báo danh

```
Create Proc Sc_DienSBD
As
    Declare @I Int
    --Khai báo biến Con trỏ
    Declare Cursv Cursor For
        Select Sbd, Tensv From Sinhvien Order By Tensv
    Open Cursv   --Mở con trỏ
    --Xử lý mẫu tin trên con trỏ
    Set @I = 1
    Fetch Next From Cursv
    While @@Fetch_Status = 0
    Begin
        Update Sinhvien Set Sbd = @I Where Current Of Cursv
        Fetch Next From Cursv
        Set @I = @I + 1
    End
    Close Cursv  --Đóng con trỏ
    Deallocate Cursv     --Giải phóng con trỏ
Go
```

II- Khai báo Cursor:

```
DECLARE <CursorName> CURSOR
    [ LOCAL | GLOBAL ] -- Phạm vi hoạt động
    [ FORWARD_ONLY | SCROLL ] -- Phương thức di chuyển
    [ STATIC | KEYSET | DYNAMIC ] -- Loại Cursor
    [ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ] –Xử lý đồng thời
    [ TYPE_WARNING]
    FOR <lệnh Select>
    [ FOR UPDATE [ OF ColumnName [, …n] ] ]
```

1- Phạm vi hoạt động của Cursor:

Mặc định, cursor có phạm vi Global trên kết nối mà nó đã được tạo. Nghĩa là, bạn có thể sử dụng cursor trên các gói thực hiện trên kết nối đó, trừ phi bạn đóng và giải phóng Cursor. Nếu bạn mở Cursor chưa đóng thì sẽ bị lỗi và có khi bị treo cho đến khi đóng kết nối. Với lý do đó, khi không sử dụng Cursor Global, bạn nên đóng và giải phóng Cursor.

Nếu bạn muốn tạo cursor Local bạn phải chỉ định rõ ràng trong khai báo cursor:

```
Declare Cursv Cursor  Local For
        Select Sbd, Tensv From Sinhvien Order By Tensv
```

Cursor Local có phạm vi hoạt động bên trong gói đã tạo nó. Và tự giải phóng khi kết thúc gói.

2- Phương Thức Di Chuyển Trên Cursor:

Có 2 phương thức di chuyển MTHH:

- FORWARD_ONLY : là phương thức mặc định, chỉ cho phép di chuyển sang mẫu tin kế tiếp.
- SCROLL : Cho phép di chuyển lên xuống trong tập mẫu tin.

3- Các Loại Cursor:

Có 3 loại Cursor:

☑ STATIC : có thuộc tính READ ONLY, do đó không thể cập nhật các bảng nền thông qua Cursor này. Khi tạo Cursor Static, dữ liệu từ các bảng gốc sẽ được Copy sang một bảng tạm trong CSDL **tempdb.** Do đó, Nếu các table nguồn của Cursor bị thay đổi dữ liệu thì các dữ liệu không xuất hiện trên Cursor.

```
Server: Msg 16929, Level 16, State 1, Procedure SC_DIENSBD, Line 14
The cursor is READ ONLY. The statement has been terminated.
```

☑ DYNAMIC: Cho phép cập nhật dữ liệu trên các table nguồn (dùng mệnh đề WHERE CURRENT OF <tênCursor> trong các phát biểu UPDATE or DELETE), và tự động hiển thị tất cả những thay đổi từ table nguồn. Tuy nhiên, dữ liệu và thứ tự của các mẫu tin trong tập mẫu tin có thể bị thay đổi.

☑ KEYSET : Giống như cursor Dynamic. Nhưng nó chỉ được tạo khi bảng nguồn có khai báo khóa, nếu không thì SQL tự động chuyển sang loại STATIC. Khi tạo Cursor KEYSET, Tập các khóa của bảng nguồn được lưu trên một table của CSDL tempdb. Do đó, việc xóa mẫu tin hoặc thay đổi giá trị khóa trên các bảng nguồn không thông qua Cursor sẽ không phản hồi trên tập mẫu tin.

Cursor kiểu STATIC, KEYSET, và DYNAMIC mặc định dùng phương thức SCROLL.

TYPE_WARNING : Gởi thông báo chú ý về client nếu Cursor thực hiện chuyển đổi ngầm định từ kiểu yêu cầu sang một kiểu khác.

4- Xử lý đồng thời:

Trong môi trường nhiều người dùng cùng làm việc trên cùng tập dữ liệu, Làm thế nào để các Users chắc chắn rằng những thay đổi của họ không bị thay đổi bởi người dùng khác? Phụ thuộc vào kiểu Cursor mà bạn đã sử dụng, bạn không thể nhận thấy được những thay đổi cho đến khi bạn đóng Cursor và mở lại nó.

Trừ phi sử dụng cursor trong một transaction, nếu không các table nguồn của cursor không tự động khóa dữ liệu. SQL Server 2000 có 4 chọn lựa cho phép ngăn cản việc sửa đổi mẫu tin cho tới khi thực hiện xong hoặc bằng cách khóa các table nguồn của cursor để bảo vệ các thay đổi của bạn.

☑ **READ_ONLY** : Dùng khi chỉ truy xuất dữ liệu mà không sửa đổi dữ liệu.

☑ **SCROLL_LOCKS** : Khoá các dòng đã được đọc vào Cursor đối với các User khác.

☑ **OPTIMISTIC WITH VALUES**: Chỉ khóa các giá trị mà bạn vừa thay đổi. Nếu người dùng khác thay đổi các giá trị đó sẽ nhận được thông báo lỗi.

☑ **OPTIMISTIC WITH ROW VERSIONING** —Khi muốn cả dòng được cập nhật, không chỉ một vài Fields trong nó.

5- Khai báo cột trong Cursor được phép cập nhật:

UPDATE [OF *column_name* [**,**...*n*]]

☑ Nếu chỉ định **OF *column_name* [,...*n*]** chỉ những cột liệt kê mới được sửa đổi.

☑ Nếu chỉ định UPDATE mà không chỉ định danh sách cột, thì tất cả các cột đều có khả năng cập nhật trừ phi chỉ định READ_ONLY.

III- Truy xuất dữ liệu trên Cursor:

FETCH [NEXT | PRIOR | FIRST | LAST
| ABSOLUTE { *n* | **@***nvar* } | RELATIVE { *n* | **@***nvar* }]
FROM [GLOBAL] *cursor_name*
[INTO **@***variable_name* [**,**...*n*]]

- **NEXT** : Chuyển sang mẫu tin kế tiếp.
- **PRIOR** : Chuyển về mẫu tin trước đó.
- **FIRST** : Chuyển về mẫu tin đầu tiên.
- **LAST** : Chuyển đến mẫu tin cuối cùng.
- **ABSOLUTE {*n* | *@nvar*}** : Nếu *n* or *@nvar* > 0, tìm đến dòng thứ n tính từ dòng đầu tiên đếm xuống trong tập mẫu tin. Nếu *n* or *@nvar* < 0, tìm đến dòng thứ n tính từ dòng cuối cùng đếm lên. Nếu *n* or *@nvar* = 0, chuyển đến vùng BOF và không có giá trị trả về. Hằng số *n* phải là số nguyên và biến *@nvar* phải thuộc kiểu **smallint**, **tinyint**, hoặc **int**. Không sử dụng phương thức ABSOLUTE cho kiểu DYNAMIC.
- **RELATIVE {*n* | *@nvar*}** : Nếu *n* hoặc *@nvar* > 0, chuyển xuống n dòng tính từ dòng kề dưới dòng hiện hành. Nếu *n* or *@nvar* < 0, Chuyển lên n dòng trước dòng hiện hành. Nếu *n* or *@nvar* = 0, trả về dòng hiện hành.
- ***cursor_name*:** Tên cursor đang mở. Nếu tồn tại cursor cục bộ và cursor toàn cục có cùng tên thì tên cursor được sử dụng sẽ là cursor cục bộ nếu không có từ khóa GLOBAL.
- **INTO *@varname*[,...*n*]** : Danh sách biến cục bộ nhận dữ liệu tương ứng từ các cột trên mẫu tin hiện hành, theo thứ tự từ trái sang phải. Số biến phải bằng số cột đã liệt kê trong câu lệnh Select khi tạo Cursor. Kiểu dữ liệu của mỗi biến phải tương thích với kiểu dữ liệu của cột hoặc được hỗ trợ chuyển kiểu ngầm định theo kiểu của cột.

Kiểm tra kết quả của lệnh FETCH: Sử dụng hàm @@FETCH_STATUS sau lệnh FETCH. Hàm trả về một trong 3 giá trị:

0	Nếu lệnh FETCH chuyển đến 1 mẫu tin trong danh sách.
-1	Nếu lệnh FETCH chuyển đến vùng BOF hoặc EOF
-2	Nếu chuyển đến 1 dòng đã bị xóa trên Server (Keyset).

Ví dụ:

```
CREATE proc sprcur
As
        Declare @ms char(6), @ten varchar(30)
        Declare a cursor SCROLL for  select mssv, tensv from sinhvien
        Open a
        Fetch next from  a into @ms, @ten
        While @@fetch_status = 0
        Begin
                Print @ms + '-' + @ten
                Fetch Next from  a into @ms, @ten
        End
        Close a
        Deallocate a
```

```
GO
```

CHƯƠNG 6 : BẢO MẬT (SECURITY)

I- Khái Niệm:

Mỗi CSDL có 1 hay nhiều users được chỉ định quyền truy xuất dữ liệu. Người quản trị có thể cấp quyền truy xuất CSDL bằng cách tạo một tài khoản đăng nhập (login) SQL Server cho User, thêm User vào CSDL và gán quyền cho User trên CSDL đó. Bao gồm các loại quyền:

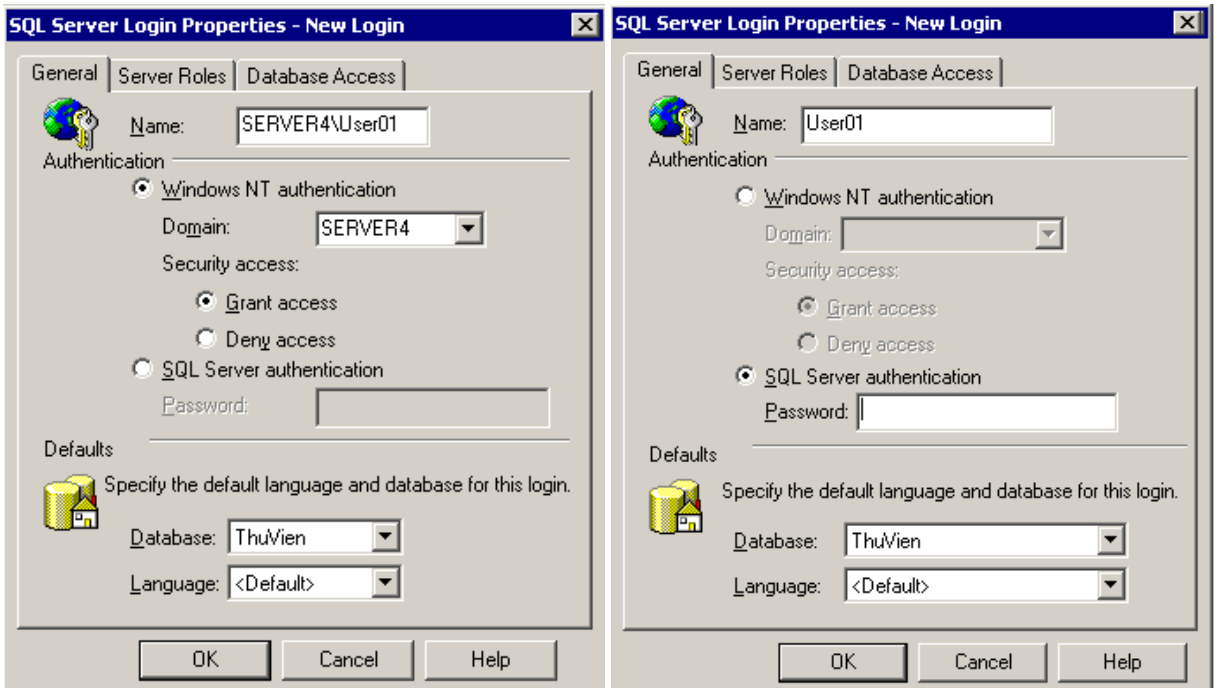
- Quyền truy cập vào SQL Server
- Quyền truy xuất CSDL
- Quyền thực hiện trên các đối tượng của CSDL
- Quyền xử lý dữ liệu

II- Sử dụng EM:

1- Tạo Tài Khoản Đăng Nhập (Login Acount):

(1) Mở mục Security, click phải mục Login và chọn New Login…

(2) Trên trang General, nhập tên đăng nhập, chế độ xác nhận, CSDL mặc định.



Chú ý: Đăng nhập với SQL Server thường dùng cho việc kết nối quay số và mạng peer-to-peer.

(3) **Trang Server Roles:** Chọn vai trò quản trị mức Server cho tài khoản đăng nhập

System Administrators	Đặc quyền cao nhất; cho phép thực hiện mọi tác vụ trên SQL
Security Administrators	Quản lý các server logins.
Server Administrators	Cho phép bạn định cấu hình những cài đặt server-wide.
Setup Administrators	Cho phép thêm và xóa các linked servers, và truy xuất vài SP
Database Creator	Tạo và hiệu chỉnh databases.
Disk Administrators	Quản lý các files trên đĩa.
Process Administrators	Quản lý tiến trình đang chạy trong một thể hiện của SQL Server.
Bulk Administrators	Thực hiện phát biểu BULK INSERT.

Chú ý :Bất kỳ Users của Windows NT thuộc nhómBUILTIN\Administrators đều có vai trò **sysadmin**.

(4) **Trang Database Access:** Chọn CSDL được phép truy xuất và vai trò của nó trong từng CSDL được chọn.

Fixed Database	Role Description
Public	Vai trò chung cho tất cả người dùng.
db_owner	Quyền cao nhất trong database.
db_accessadmin	Điều khiển truy xuất, cài đặt hoặc xóa user accounts.
db_datareader	Đọc tất cả dữ liệu trên database.
db_datawriter	Thêm, sửa, xóa dữ liệu trên các tables người dùng trong database.
db_ddladmin	Thêm, sửa, xóa các đối tượng objects (runs all DDLs).
db_securityadmin	Quản lý các roles, các thành viên của role, giấy phép trong database.
db_backupoperator	Cho phép back up database.
db_denydatareader	Từ chối quyền truy vấy dữ liệu trong database.
db_denydatawriter	Từ chối quyền thay đổi dữ liệu trong database.

Sau khi tạo login, nó tự động nhập vào tập Users của mỗi database được chọn, với tên User trùng với tên Login. Bạn có thể thay đổi thu hồi vai trò của nó trên từng CSDL bằng cách sửa đổi thuộc tính của Login, hoặc chuyển đến tập Users của database và thay đổi thuộc tính hoặc xoá user nào mà bạn không muốn cho truy xuất data của bạn.

Các Login được lưu trong table SysLogins của CSDL Master:

```
If Exists( Select 1 From Master..SysLogins Where Name = 'Login')
```

Các User trong một CSDL được lưu trong table SysUsers của CSDL đó

```
If Exists( Select 1 From SysUsers Where Name = 'User01')
```

2- Thay đổi thuộc tính cho Login:

Bấm đúp vào tên Login hoặc click phải và chọn mục Properties

3- Cấp Quyền Thực Thi Trên Mỗi CSDL:

Chọn database, trong mục Users bấm đúp vào tên User cần hiệu chỉnh (Login-ID).

Click nút Permission để chỉ định quyền truy cập dữ liệu trên từng Table, View. Quyền kiểm tra RB tham chiếu (DRI - Declarative Referential Integrity). Quyền thực hiện các thủ tục lưu trử.

Chú thích: Quyền kiểm tra RB tham chiếu được sử dụng khi Table A được cấp quyền Update hoặc Insert. Table A có RB FOREIGN KEY với table B, mà table B không được cấp quyền SELECT.

4- Cấp Quyền Tạo Đối Tượng Trên CSDL:

- Click phải vào tên CSDL, chọn Properties
- Trong HT Properties, chọn trang Permissions
- Đánh dấu chọn các phát biểu được cần cấp quyền thực hiện cho các User.

5- Xóa User trên một CSDL:

Click phải vào tên user và chọn Delete.

6- Tạo User với Login đã có:

Click phải vào mục **Users** và chọn **New Database User...**

Chú ý: Bạn có thể thay đổi tên User bằng cách xóa và tạo lại với login cũ.

7- Xóa Tài Khoản Đăng Nhập:

Click phải vào tên Login và chọn Delete

Với cách này sẽ xóa bỏ các user trong các CSDL đã chọn cho tài khoản này.

III- Sử Dụng QA:

1- Tạo Tài Khoản Đăng Nhập (Login Acount):

a- Thêm Tài khoản với chế độ xác nhận SQL Server:

SP_ADDLOGIN [@login =] <'Tênđăngnhập'>
[, @password =] <'password'>
[, @defdb =] <'TênCSDL'>
[, @deflanguage =] <'Ngônngữ'>
[, @sid =] <'Mã nhận dạng Login'>
[, @encriptopt = 'skip_encription']

- @defdb : tên CSDL mặc định được mở khi đăng nhập. Nếu không chỉ định mặc định là CSDL **Master.**
- @deflanguage : Ngôn ngữ mặc định
- @sid : varbinary(16) : Mã nhận dạng của hệ thống. Nếu không chỉ định, hệ thống tự tạo một mã số mới.
- @encriptopt varchar(20): Mặc định Password sẽ được mã hóa khi lưu trong các table hệ thống. Ngoại trù bạn gán giá trị **skip_encryption** cho tham số này.
- Thủ tục trả về giá trị 0 nếu thành công, ngược lại trả về giá trị 1.

Ví dụ: Tạo tài khoản tên 'myname', mật khẩu 'mypwd' vào CSDL 'QLDeTai'

Sp_AddLogin 'user01', '01', 'QLDeTai'

Kết quả trả về : New loigin created

b- Cấp quyền kết nối cho User hay nhóm User của Windows kết nối đến SQL Server:

SP_GRANTLOGIN [@loginame =] 'login'

'Login': <Domain>|<Computer name>\<tênnhóm>

Ví dụ: Thêm tài khoản cho user Windows NT [Server4\User01] kết nối đến SQL Server.

EXEC sp_grantlogin ' Server4\User01'

Hay EXEC sp_grantlogin [Server4\User01]

*Chú ý: Chỉ được thức hiện bởi những thành viên có vai trò **sysadmin** và **securityadmin**.*

2- Thay đổi mật khẩu:

Sp_Password [@old =] <'oldPW'>

[, @new =] <'newPW'>
[, @LoginName =] <'login'>

3- Cấp quyền Truy xuất CSDL Hiện Hành cho Login:

Sp_GrantDBAccess [@loginname =] 'Tênđăngnhập'
[[, @name_in_db =] 'TênUser']

Ví dụ: Use QLDeTai
Go
sp_GrantDBAccess 'user01', 'Anh'

4- Xóa quyền truy xuất CSDL hiện hành:

Sp_RevokeDBAccess [@name_in_db =] 'TênUser'

Ví dụ: Sp_RevokeDBAccess 'Anh'

5- Cấp quyền thực thi trên CSDL:

Bao gồm các quyền: Select, Insert, Update, Delete, Reference, Excecute.

GRANT ALL | <quyền> [,…]
ON <TênTable|View>[(<têncột>,..)] | <tênSP>
TO <tên Login hoặc Role> [,…]
[WITH GRANT OPTION]
[**AS** <tên Role>]

- **All** : Cấp tất cả các quyền thực thi Select, Insert, Update, Delete hay Reference trên table hay view; quyền Excecute cho SP.
- **TO** <tên Login hoặc Role> : Khi cấp quyền cho nhóm hay user của Windows NT, phải chỉ định: <Domain>|<Computer name>\<tênnhóm>.

  Để cấp quyền cho nhóm cục bộ Windows NT built-in, dùng từ khóa BUILTIN thay thế tên domain hoặc computer name.

  Quyền thực thi đã cấp cho role **Public** được áp dụng cho tất cả users trong CSDL. Quyền thực thi đã cấp cho user **Guest** được sử dụng cho tất cả Users không được phân quyền truy xuất trên CSDL.

Ví dụ: USE QLDeTai
Grant All On DeTai To User01
GO
GRANT SELECT ON DeTai TO public
GO
GRANT INSERT, UPDATE, DELETE ON DeTai TO Mary, [Corporate\BobJ]
GO

- **WITH GRANT OPTION :** Cho phép Login cấp quyền đã chỉ định trên đối tượng cho Login khác.
- **AS {group | role}** : Được dùng khi quyền thực thi trên một đối tượng đã cấp cho nhóm hoặc role, và một User của nhóm hoặc Role muốn cấp quyền thực thi cho User khác không là thành viên của nhóm hoặc Role.

Ví dụ: table NhatKy được tạo bởi user Lac. Lac cấp quyền SELECT table NhatKy cho Role BanBe với mênh đề WITH GRANT OPTION để các user thành viên của Role BanBe có thể nhường quyền này cho các user khác không thuộc Role BanBe.

User Hong, là thành viên của BanBe, muốn nhường quyền SELECT table NhatKy cho user Khoa, không là thành viên của role BanBe.

```
/* User Lac */
GRANT SELECT ON NhatKy TO BanBe WITH GRANT OPTION
/* User Hong */
GRANT SELECT ON NhatKy TO Khoa AS BanBe
```

6- Từ Chối quyền thực thi trên CSDL:

DENY ALL | <quyền> [,…]
[**(** *column* [**,**...*n*] **)**] ON { *table* | *view* }
| ON *table* | *view* [**(** *column* [**,**...*n*] **)**]
| ON *stored_procedure*
TO <tên Login hoặc Role> [,…] [Cascade]

Ví dụ: USE QLDeTai
GO
GRANT SELECT ON DeTai TO public
GO
DENY SELECT, INSERT, UPDATE, DELETE
ON DeTai TO Mary, John, Tom

7- Xóa bỏ quyền thực thi đã cấp hoặc từ chối trên CSDL:

REVOKE [GRANT OPTION FOR]
{ ALL | *permission* [**,**...*n*] }
[**(** *column* [**,**...*n*] **)**] ON { *table* | *view* }
| ON { *table* | *view* } [**(** *column* [**,**...*n*] **)**]
| ON { *stored_procedure* | *extended_procedure* }
| ON { *user_defined_function* }
TO | FROM *security_account* [**,**...*n*]
[CASCADE]
[AS { *group* | *role* }]

Ví dụ : Xóa bỏ quyền Select đã từ chối cho User Mary trên table DeTai.
REVOKE SELECT ON DeTai TO Mary

8- Cấp Quyền tạo đối tượng trong CSDL:

GRANT ALL | <lệnh> [,…]
TO <tên Login hoặc Role> [,…]

Bao gồm các lệnh: CREATE DATABASE; CREATE DEFAULT;
CREATE PROCEDURE; CREATE RULE; CREATE TABLE; CREATE VIEW;
BACKUP DATABASE; BACKUP LOG

Ví dụ: GRANT CREATE DATABASE, CREATE TABLE
TO Mary, John, [Corporate\BobJ]

9- Từ Chối quyền tạo đối tượng trên CSDL:

DENY { ALL | *statement* [**,**...*n*] } **FROM** *security_account* [**,**...*n*]

Ví dụ: DENY CREATE TABLE FROM Joe, [Corporate\BobJ]

10- Xóa bỏ quyền tạo đối tượng đã cấp hoặc từ chối trên CSDL:

REVOKE { ALL | *statement* [**,**...*n*] } **FROM** *security_account* [**,**...*n*]

Ví dụ : REVOKE CREATE TABLE FROM Joe, [Corporate\BobJ]

IV- Vai Trò Của User Trong SQL Server

SQL Server hổ trợ 2 nhóm roles:

- Vai trò trên SQL Server Chứa các quyền quản trị SQL Server
- Vai trò trên Database: Chứa các quyền quản lý và thực thi trên các đối tượng của CSDL. Ngoài các vai trò được cung cấp bởi SQL Server có thể tạo thêm vai trò khác.

1- Tạo Vai trò trên CSDL:

☑ **Trong EM:** Mở CSDL, Click phải vào mục Roles và chọn New Database Role…, Nhập tên Role mới và click nút Add để thêm User

☑ **Trong QA: SP_ADDROLE** [@rolename =] 'role' [, [@ownername =] 'owner']

Ví dụ: Trong CSDL QLDeTai thêm role 'QuanLy'

```
SP_ADDROLE 'QuanLy'
Go
GRANT SELECT ON DeTai TO QuanLy
```

2- Thêm User vào Role:

SP_ADDROLEMEMBER [@rolename =] 'role',
[@membername =] 'UserName'

Ví dụ: Cấp quyền truy cập CSDL QLDeTai cho User và thêm vai trò ‘QuanLy’ cho User.

```
USE QLDeTai
GO
EXEC SP_GRANTDBACCESS 'Server4\User01', 'Hong'
GO
EXEC SP_ADDROLEMEMBER 'QuanLy', 'Hong'
```