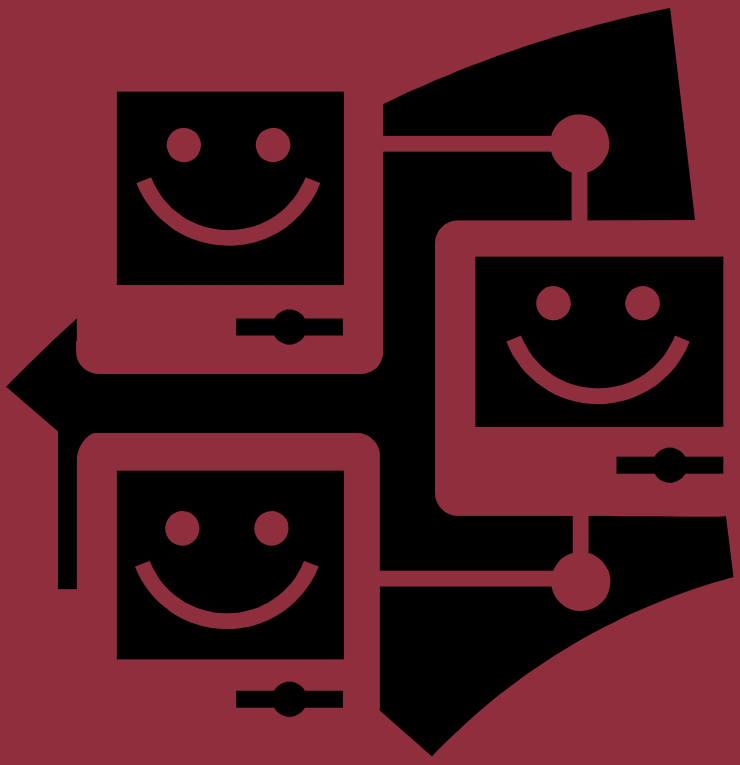
Trường Trung Học Giao Thông Công Chánh
Khoa Công Nghệ Thông Tin

Giáo trình:

SQL Server



Giáo viên biên soạn: Lưu Hữu Phước
Khoa Công Nghệ Thông Tin

Năm 2008

Chương 1 TỔNG QUAN VỀ SQL SERVER và CƠ SỞ DỮ LIỆU VỚI MÔ HÌNH QUAN HỆ

I. LỊCH SỬ PHÁT TRIỂN CỦA SQL SERVER:

Vào những năm 1970 hãng IBM đã đưa ra ngôn ngữ máy tính được thiết kế đặc biệt dùng để làm truy vấn CSDL gọi là: SEQUEL (Struct English Query Language). Theo thời gian thì ngôn ngữ này được bổ sung dẫn đến mức nó không chỉ là ngôn ngữ dùng cho truy vấn dữ liệu mà còn có thể tạo ra các CSDL và cho phép quản lý các tính năng bảo mật của hệ thống. IBM đã công bố ngôn ngữ này rộng rãi và nó được biết đến với tên SQL (Structure Query Language). Có nhiều phiên bản khác nhau của SQL được dùng cho các hệ thống CSDL hiện nay. Phần mềm SQL của Microsoft sử dụng phiên bản: Transact – SQL (T-SQL)

Microsoft bắt đầu xây dựng SQL cùng với hãng phần mềm Sybase để dùng cho hệ điều hành OS/2. Sau khi tách riêng ra thì Microsoft tiếp tục phát triển phần mềm SQL dùng cho hệ điều hành Window NT và luôn luôn có các phiên bản ra đời.

II. CƠ SỞ DỮ LIỆU SỬ DỤNG TRONG SQL SERVER

SQL Server dùng loại CSDL được gọi là CSDL quan hệ (Relational Database). Đây là CSDL mà dữ liệu bên trong tổ chức thành các bảng chứa các cột, dòng dữ liệu có liên quan đến nhau. Trong SQL Server, một CSDL không chỉ chứa dữ liệu thô mà còn chứa các thông tin có liên quan đến CSDL như các bảng ảo, các Store Procedure, Trigger, ...

III. CÁC ĐỐI TƯỢNG CỦA CƠ SỞ DỮ LIỆU QUAN HỆ:

1) Bảng (Table)

- Là đối tượng quan trọng nhất của một CSDL, đối tượng này chứa các cột với các kiểu dữ liệu khác nhau và các dữ liệu thô thực sự thì được đặt trên các dòng.

2) Thủ tục được lưu (Store Procedure)

- Đối tượng này chứa các đoạn mã SQL có thể viết và lưu trữ trong CSDL. Khi Store Procedure này được gọi thi hành thì các đoạn mã SQL trong đó sẽ được thi hành một cách tự động.

Ví dụ: để thêm một mẫu tin vào bảng Dm_MonHoc, ta viết một Store mà trong đó dùng câu lệnh SQL: Insert Into, và Store này có các tham số chính là dữ liệu của mẫu tin cần thêm.

3) Trigger:

- Thực chất là một Store Procedure, nó sẽ tự động được thi hành khi có sự cập nhật dữ liệu như thêm, xoá, sửa từ Table tương ứng có khai báo Trigger. Trigger được thực hiện để bảo đảm các quy luật nghiệp vụ được áp dụng cho CSDL là đúng đắn.

Ví dụ: Sau khi thêm, sửa, hay xoá một sinh viên trong bảng Dm_SinhVien thì phải tự động cập nhật sĩ số của lớp học đó trong bảng Dm_LopHoc.

4) Quy luật (Rules)

- Một Rules khi được gán cho một cột nào đó sẽ bảo đảm dữ liệu nhập vào phải phù hợp với tiêu chuẩn do người dùng đưa ra. Rule được hiểu như ràng buộc về miền giá trị trên cột đó.

Ví dụ:

- *Cột SoLuong phải là số dương*
- *Cột DiemThi phải có giá trị từ 0 đến 10*

5) Khoá chính (Primary key)

- Mặc dù không phải là một đối tượng trong CSDL, nhưng khoá rất cần cho CSDL quan hệ. Khoá chính bắt buộc phải duy nhất trên các dòng trong các Table.

6) Khoá ngoại (Foreign key)

- Cũng không phải là một đối tượng trong CSDL, nó chính là sự tham chiếu của các cột từ một bảng này đến khoá chính hay các ràng buộc duy nhất của một bảng khác. SQL Server sử dụng khoá chính và khoá ngoại để liên hệ dữ liệu với nhau từ các bảng riêng biệt khi thực hiện một câu truy vấn từ nhiều bảng.

7) Ràng buộc (Constraints)

- Là cơ chế bảo đảm tính toàn vẹn dữ liệu.

8) Giá trị mặc định (Default)

- Một Default có thể xác lập trên một trường sao cho khi không có dữ liệu được đưa vào thì giá trị mặc định sẽ được dùng đến.

Ví dụ: giá trị mặc định của Phai là 1

9) Bảng ảo

- Là một đối tượng dùng để lưu trữ các câu truy vấn trong CSDL, nó được tạo ra và lưu trữ lại sao cho có thể dùng dễ dàng về sau. Một bảng ảo thường chứa một số cột được lấy từ một bảng hay từ một liên kết hai hay nhiều bảng.

IV. SQL SERVER VÀ MÔ HÌNH CLIENT / SERVER

- Phần mềm SQL Server của Microsoft là một mô hình CSDL Client/Server.

Một ứng dụng theo kiểu Client/Server có thể được chia thành hai phần: Một phần chạy trên Server và một phần khác chạy trên WorkStation (trạm làm việc).

- SQL Server là phần Server của chương trình với nhiều Client có thể kết nối vào Server.

V. CÁC CƠ SỞ DỮ LIỆU MẶC ĐỊNH TRÊN SQL SERVER:

1) Cơ sở dữ liệu Master:

- Đây là một CSDL chính để chạy SQL Server. CSDL chứa một con trỏ đến một tập tin dữ liệu cơ sở chứa thông tin về các cơ sở dữ liệu khác được cài đặt trong hệ thống cũng như các thông tin về các dịch vụ.

2) *Cơ sở dữ liệu Model:*

- Đây là CSDL mẫu. Mỗi khi có một CSDL mới được tạo ra thì CSDL Model sẽ được sao chép sang. Sau đó các yêu cầu về kích thước và các thay đổi nếu muốn sẽ được áp dụng trên CSDL mới.

3) *Cơ sở dữ liệu TempDB:*

- Là nơi sắp xếp, các kết nối và các hoạt động khác đòi hỏi vị trí tạm thời được thực hiện.

VI. CÁC CÔNG CỤ CHÍNH CỦA SQL SERVER:

1) *SQL Server Books Online:*

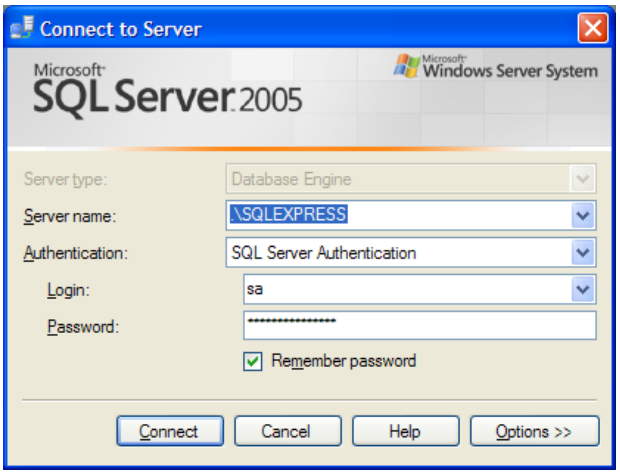
- Là một công cụ trợ giúp trực tuyến giúp người làm việc truy tìm sự giúp đở về những vấn đề có liên quan đến SQL Server.

2) *Tiện ích (SQL Server configuration manager)*

- Để các máy PC có thể kết nối và dùng CSDL trên SQL Server thì ta cần cấu hình các tiện ích trên máy trạm giống như cấu hình của Server với tiện ích SQL Server configuration manager. Giao thức (protocol) thường dùng nhất là TCP/IP/

3) *SQL Server Management Studio:*

- Đây là tiện ích cung cấp cho người quản trị nhiều chức năng quản lý SQL Server với giao diện đồ hoạ như tạo, cập nhật, xoá CSDL. Ngoài ra, tiện ích này còn cho phép soạn thảo và thi hành các câu lệnh SQL, tạo các đối tượng Store Procedure, Trigger, Bảng ảo, ... Khi kích hoạt tiện ích này thì cần cung cấp tên hay địa chỉ IP của Server cần kết nối. Đồng thời cung cấp tên người dùng và mật khẩu nếu cần.



Chương 2 TẠO LẬP CSDL TRÊN SQL SERVER

I. TẠO CSDL:

- Để tạo một Cơ sở dữ liệu trên SQL Server thì người ta thường dùng một trong hai phương pháp sau:
  - *Sử dụng SQL Server Management Studio\Object Explorer*
  - *Sử dụng câu lệnh SQL*

1) *Sử dụng câu lệnh SQL*

a) Cú pháp:

```
CREATE DATABASE <Database_Name>
ON [Primary]
( Name = <Logical_Name>,
FileName = <'Physical_name'>,
Size = <Size>, MaxSize=<Size>,
Filegrowth = <Size> )
LOG ON
( Name = <Logical_Name>,
FileName = <'Physical_name'>,
Size = <Size>,
MaxSize = <Size>,
Filegrowth = <Size> )
```

b) Ý nghĩa:

- **<Database_Name>**: là đại diện cho CSDL đang tạo.
- **<On Primary>**: đặc tả File CSDL thuộc về nhóm File nào, nhóm File mặc định là Primary.
- **<Name>**: là tên Logic của File dữ liệu thường được dùng khi Backup, Export, Import … dữ liệu để phân biệt ta thêm chuỗi Data ở phía sau.
- **<File_Name>**: chứa đường dẫn và ổ đĩa nơi chứa tập tin dữ liệu vật lý. Nếu một CSDL có nhiều File dữ liệu thì File dữ liệu đầu tiên là .MDF, các File dữ liệu tiếp theo là .NDF. Trường hợp nếu là File nhật ký giao tác thì tên tập tin là .LDF.
- **<Size>:** là kích thước khi khởi tạo CSDL
- **<Max size>**: Là kích thước tối đa mà CSDL có thể đạt đến.
- **<File Growth>**: là kích thước gia tăng mỗi khi CSDL có dữ liệu được thêm vào vượt quá kích thước hiện hành.
- **<Log On>**: Là mô tả về File nhật ký giao tác. Các tham số tương tự như trên.

c) Ví dụ minh họa:

Ví dụ 1: Tạo CSDL với tên là Vidu1_xx với:

- File dữ liệu
  - *Name: Vidu1_xx_Data*
  - *File Name: 'Vidu1_xx_data.Mdf'*
  - *Size: 2MB*
  - *Max size: 5MB*
  - *File Growth: 1MB*
- File Log (nhật ký)
  - *Name: Vidu1_xx_Log*
  - *File Name: 'Vidu1_xx_data.Ldf'*
  - *Size: 2MB*
  - *Max size: 5MB*
  - *File Growth: 1MB*

Bài giải:

```
CREATE DATABASE Vidu1_xx
ON Primary
(Name=Vidu1_xx_data,
FileName='E:\Data\LTxx\Vidu1_xx_data.MDF',
Size=2MB, Maxsize=5MB, FileGrowth=1MB)
LOG ON
(Name=Vidu1_xx_log,
FileName='E:\Data\LTxx\Vidu1_xx_log.Ldf',
Size=2MB, Maxsize=5MB, FileGrowth=1MB)
```

Ví dụ 2: Tạo CSDL với tên là Vidu2_xx có:

- 2 File dữ liệu: Size=2, MAxsize=5, FileGrowth=1
- 2 File Log: kích thước như trên

Bài giải:

```
CREATE DATABASE Vidu2_xx
ON Primary
(Name=Vidu2_xx_data1,
FileName='E:\Data\LTxx\Vidu2_xx_data1.MDF', Size=2MB,
Maxsize=5MB, FileGrowth=1MB),
(Name=Vidu2_xx_data2,
FileName='E:\Data\LTxx\Vidu2_xx_data2.MDF', Size=2MB,
Maxsize=5MB, FileGrowth=1MB)
LOG ON
(Name=Vidu2_xx_log1,
FileName='E:\Data\LTxx\Vidu2_xx_log1.Ldf',
Size=2MB, Maxsize=5MB, FileGrowth=1MB),
```

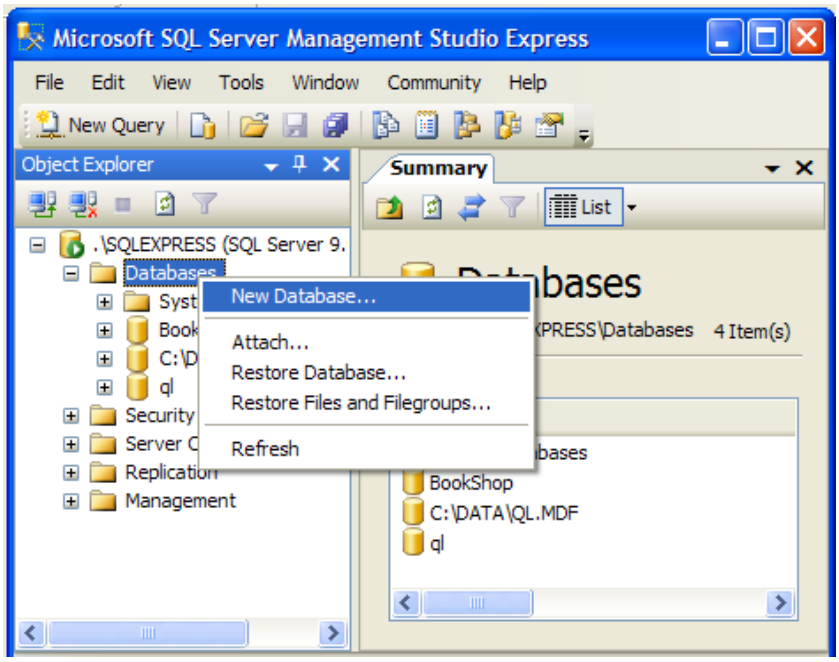
```
(Name=Vidu2_xx_log2,
FileName='E:\Data\LTxx\Vidu2_xx_log2.Ldf',
Size=2MB, Maxsize=5MB,  FileGrowth=1MB)
```

Lưu ý: Nếu ta xây dựng một CSDL như Vidu2_xx thì:

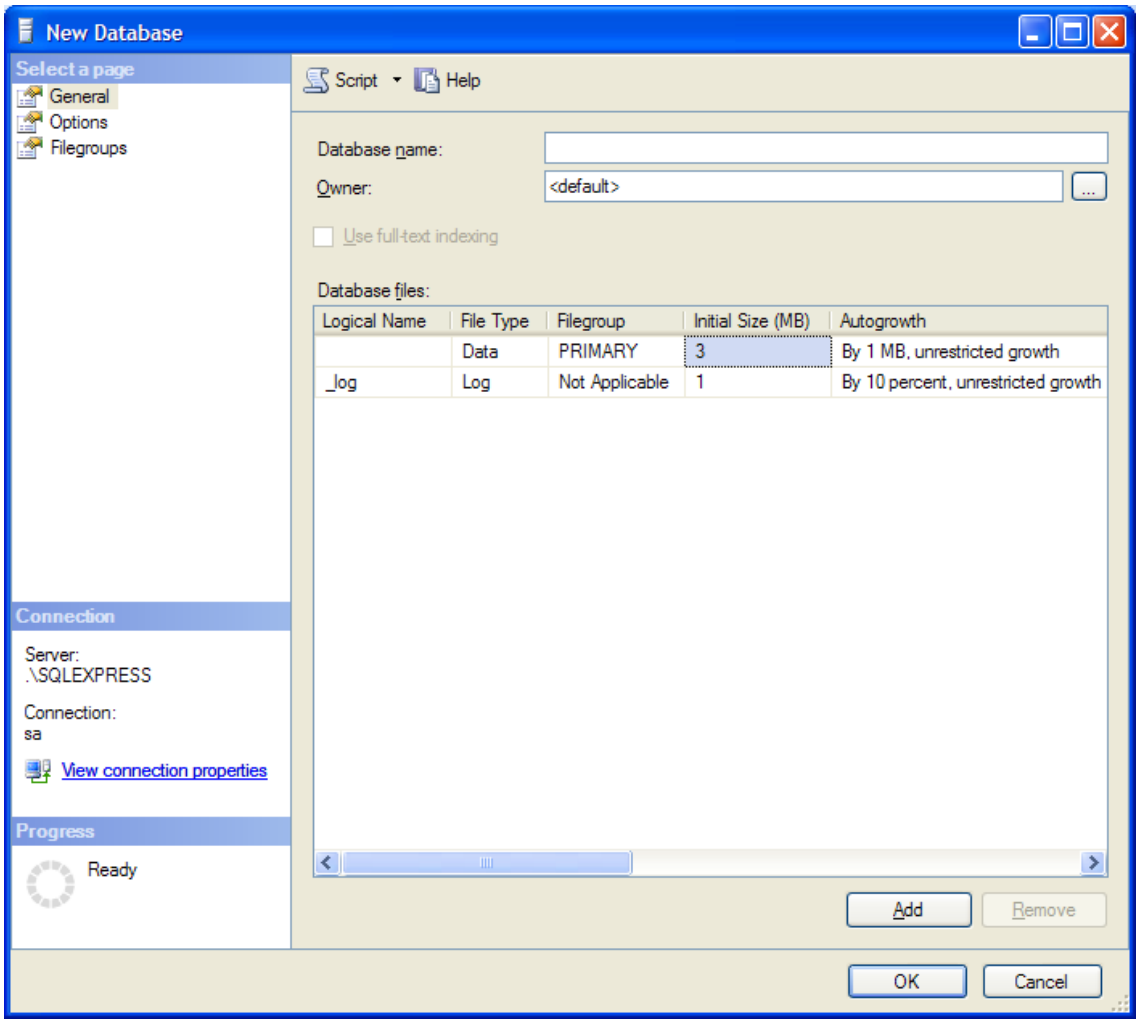
- Thông tin trên CSDL sẽ tự động trải qua các File dữ liệu dẫn đến làm giảm tranh chấp khi có nhiều người cùng truy xuất vào CSDL
- Đối với File nhật ký thì việc tự động trải thông tin không xảy ra: khi File nhật ký này đầy thì dữ liệu mới chuyển vào File nhật ký kế tiếp.

2) *Sử dụng SQL Server Management Studio\Object Explorer:*

- Khởi động SQL Server Management Studio
- Kết nối với SQL Server (Tên Server theo tên máy)



- Ở cửa sổ bên trái chọn Folder Database, Click chuột phải và chọn New Database.



- Đặt tên CSDL tại ***Database Name***.
- Click tại (…) của cột Path trên lưới để xác định đường dẫn của tập tin CSDL và tập tin nhật ký trên đĩa cứng.
  - ***Initial Size (MB)****: là kích thước File khởi tạo (tương ứng với thành phần lệnh Size)*
  - ***Auto Growth (MB)****: là kích thước lớn nhất của File CSDL (tương ứng với thành phần lệnh Maxsize).*
- Click OK để hoàn tất.

II. XEM THÔNG TIN VỀ CSDL:

1) Dùng câu lệnh:

Cú pháp:

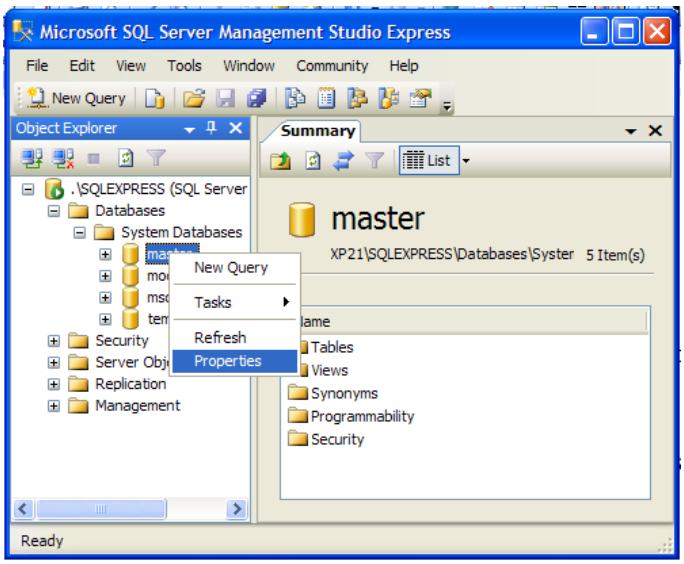
```
EXEC Sp_HelpDb <Database_Name>
```

Ví dụ:

- Exec Sp_HelpDb (thi hành bấm F5, kiểm tra lại lỗi trước khi thi hành là CTRL-F5)
- Exec Sp_HelpDb Vidu2_xx.

2) Dùng Object Explorer:

Click phải chuột vào CSDL muốn xem thông tin ở cửa sổ bên trái chọn Properties.



III. SỬA ĐỔI CSDL:

1) *Cú pháp chung:*

Cú pháp:

```
ALTER DATABASE <Database_name>
Add File <F*> To FileGroup <FileGroup_Name> |
Add Log File <F*> [, . . . , n] |
Remove File <Logical_Name> |
Add FileGroup <FileGroup_Name> |
Remove FileGroup <FileGroup_Name> |
Modify File <F*>
Modify FileGroup <FileGroup_Name>
<FileGroup_Property>
Modify Name = <Tên CSDL mới>
```

Với <F*> là

(Name=<Logical_name>,
FileName=<'Physical_Name'>,
Size=<Size>,
MaxSize=>Size>,
FileGrowth=<Size>)

- **AddFile**: thêm một file dữ liệu vào
- **Add Log File**: là thêm một File nhật ký vào
- **Remove File**: là xoá File khỏi CSDL, chú ý là File phải rỗng trước khi xoá
- **Add File Group**: là thêm nhóm File mới
- **Remove File Group**: là xoá nhóm File
- **Modify File**: cho phép sửa đổi các thuộc tính của File
- **Modify File Group**: cho phép sửa đổi thuộc tính của nhóm File
- **Modify Name**: cho phép sửa đổi tên CSDL.

2) Mở rộng CSDL:

Để mở rộng CSDL thì ta có thể thêm file dữ liệu hay file nhật ký vào CSDL. Việc mở rộng này có thể thực hiện theo hai cách:

a) Sử dụng lệnh thêm vào CSDL các File dữ liệu cũng như File nhật ký.

Ví dụ: Thêm một File dữ liệu vào CSDL Vidu2.

```
Alter Database Vidu2_xx
Add File
(Name=Vidu2_xx_data3,
FileName='E:\Data\LTxx\Vidu2_xx_Data3.ndf', Size= 2MB,
Maxsize=5MB)
```

b) Sử dung SQL Server Management Studio:

- Click chuột phải tại tên Database, chọn Properties xuất hiện cửa sổ Database Properties.
- Click nút ***Add*** để thêm file CSDL

3) Thu nhỏ CSDL

Ta có thể thu nho lại CSDL bằng cách xóa bớt file dữ liệu, file nhật ký hoặc thu nhỏ kích thước ban đầu của CSDL.

Cú pháp:

```
DBCC Shrink Database (<Database_Name>,Percent_Target)
```

- **Percent_Target**: dung lượng của CSDL còn lại sau khi thu.

Chú ý:

- Không thể thu nhỏ hơn kích thước cần thiết của việc lưu trữ dữ liệu.
- Không thể thu nhỏ hơn kích thước của CSDL mẫu.

4) Thu nhỏ File:

Cho phép thu nhỏ kích thước của các File

Cú pháp:

```
DBCC ShrinkFile (<Logical_Name>,Target_Size[EmptyFile])
```

Vídụ: Xoá trống một File trong CSDL Vidu2.

```
Use Vidu2_xx
Go
DBCC ShrinkFile (Vidu2_xx_Data2, Empty File)  => Xoá nội
dung (xoá trống) tập tin
Go
Alter Database Vidu2_xx
Remove File Vidu2_xx_data2
```

5) Đổi tên CSDL:

Ta có thể đổi tên CSDL theo hai cách như sau:

Cú pháp 1:

```
EXEC Sp_RenameDB <'Tên cũ'> , <'Tên mới'>
```

Ví dụ:

```
EXEC Sp_RenameDB  'OldTest' ,  'NewTest'
```

Cú pháp 2:

```
ALTER DATABASE <Database_name>
Modify Name = <Tên CSDL mới>
```

Ví dụ:

```
Alter Database VD1_xx
Modify Name=VD2_xx
```

Chú ý:

- CSDL phải ở chế độ một người dùng. Cú pháp chuyển một CSDL về chế độ một người dùng:

```
EXEC Sp_DbOption <Database_Name>,'Single User',True
```

- Phải đứng tại CSDL Master để đổi tên
- Sau đó ta phải đổi lại chế độ nhiều người dùng theo cú pháp sau:

```
EXEC Sp_DbOption <Database_Name>,'Single User',False
```

Ví dụ: Đổi tên CSDL Vidu2 thành VD2.

```
Use Master
Go
Exec Sp_DbOption Vidu2_xx, 'Single User', True
Go
Exec Sp_RenameDb 'Vidu2_xx' , 'VD2_xx'
Go
Exec Sp_DbOption VD2_xx, 'Single User', False
```

6) Thay đổi thông số của các File trong CDSL:

```
ALTER DATABASE <Database_name>
Modify File
(Name=<Tên Logic>, NewName=<Tên Logic mới>,
Size=<K.thước mới>, ...)
```

7) Xoá CSDL:

Cú pháp:

```
DROP DATABASE <Database_Name1> [,<Database_Name2>, ….]
```

Chú ý:

- Khi xoá thì phải đứng tại CSDL Master và thực hiện theo cú pháp trên.

Ví dụ: Xoá CSDL Vidu2

```
Use Master
Go
Drop Database Vidu2_xx
```

Chương 3 TẠO VÀ QUẢN LÝ BẢNG TRONG SQL SERVER

I. TẠO BẢNG:

1) *Khái niệm:*

Bảng dùng để lưu dữ liệu trong một CSDL, được tổ chức thành các cột, dòng. Trên mỗi cột của bảng chỉ lưu một loại dữ liệu nhất định được đặc trưng bởi kiểu dữ liệu.

2) *Kiểu dữ liệu:*

a) Giá trị Null:

- Khi một cột có giá trị là Null thì không nhất thiết phải nhập dữ liệu cho cột đó.

b) Kiểu chuỗi Char(n), VarChar(n)

- *Char(n) (10) : Chiều dài cố định Vd: Char(10) => 'TH'*
- *Varchar(n) (10) : chiều dài thay đổi Vd: Varchar(10) => 'TH'*
- *Nchar(n) hay NvarChar(n): hỗ trợ dữ liệu Unicode*

c) Kiểu số nguyên:

- *TinyInt (Byte) : 0 – 255*
- *SmallInt (Integer) : từ -32.768 đến +-32.768*
- *Int (Long) : từ -2 tỷ đến +2 Tỷ*

d) Kiểu số thực:

- *Real (Single) : có độ chính xác < 7 chữ số thập phân*
- *Float (Double) :*
- *Numeric :* } *độ chính xác > 7 chữ số thập phân*
- *Decimal :*

e) Kiểu Bit (Yes/No):

Thường dùng để biểu diễn loại dữ liệu chỉ nhận một trong hai giá trị. Miền giá trị của dữ liệu Bit là {0, 1} và không cho phép nhận giá trị Null.

f) Kiểu ngày giờ:

- *SmallDateTime : Có miền giá trị từ 01/01/1900 → 06/06/2079*
- *DateTime : Có miền giá trị từ 01/01/1753 → 31/12/9999*

3) *Cột tính toán:*

- Lưu trữ biểu thức tính toán dữ liệu, các cột sử dụng trong biểu thức ở trong cùng một bảng

Cú pháp:

```
<Tên cột> As <Biểu thức>
```

4) *Cột kiểu nguyên với thuộc tính Indentity:*

- Trường hợp này được sử dụng khi ta muốn giá trị của cột tự động tăng. Cú pháp sử dụng thuộc tính này như sau:

```
<Tên cột> Int Identity(n, m)
```

  - *n là số bắt đầu*
  - *m là bước tăng.*

5) *Tạo bảng:*

a) Sử dụng SQL:

Cú pháp:

```
CREATE TABLE <Tên bảng>
(<Tên cột> <Kiểu dữ liệu>[Primary key][identity|NULL|NOT NULL],..)
```

Để tạo khoá chính, có 3 cách:

- Cách 1: như cú pháp trên, là khai báo từ khoá Primary Key ngay sau tên Field cần làm khoá chính. Cách này không tiện lợi vì không thể tạo khoá chính cho nhiều Field và không thể thay đổi khoá chính.

Ví dụ: Tạo bảng Monhoc(MaMH, TenMH, Sotiet) như sau:

```
Use QLSV
GO
Create Table Monhoc
(MaMH char(2) Primary key, TenMH varchar(20), Sotiet
TinyInt Not NULL)
GO
```

- Cách 2: Sau khi đã khai báo các Field thì cuối lệnh ta dùng mệnh đề Primary Key (<Danh sách các Field làm khoá chính>) theo cú pháp sau:

```
CREATE TABLE <Tên bảng>
(<Tên cột 1> <Kiểu dữ liệu>,<Tên cột 2> <Kiểu dữ liệu> ,...,
Primary Key (<Tên cột 1> [,<Tên cột 2>,...]))
```

Ví dụ: Bảng CTHD(SoHD, MaHH, Soluong, Dongia) được tạo như sau:

```
Use QLSV
GO
Create Table CTHD
(SoHD char(2), MaHH char(3),Soluong SmallInt, Dongia
Float, Primary Key (SoHD,MaHH))
GO
```

- Cách 3: (ưu việt nhất) vì ta có thể xoá hay điều chỉnh bộ khoá chính cho bảng, dùng mệnh đề Constraint, theo cú pháp:

```
CREATE TABLE <Tên bảng>
(<Tên cột 1> <Kiểu dữ liệu>,<Tên cột 2> <Kiểu dữ liệu>,...,
Constraint <Tên khoá chính>
Primary Key (<Tên cột 1> [,<Tên cột 2>,...])
```

Ví dụ: Bảng CTHD(SoHD, MaHH, Soluong, Dongia) được tạo như sau:

```
Use QLSV
GO
Create Table CTHD
(..., Constraint PK_CTHD Primary Key (SoHD,MaHH))
GO
```

Đối với bảng đầu nhiều, có khoá chính và khoá ngoại, cú pháp tạo bảng vẫn như trên, nhưng thêm các ràng buộc khoá ngoại theo cú pháp như sau:

```
...
Constraint <Tên ràng buộc khoá ngoại 1> Foreign Key(<Tên
Field khoá ngoại>)
References <Tên bảng chứa khoá chính cần quan hệ>(<Tên Field
khoá chính>),
Constraint <Tên ràng buộc khoá ngoại 1> Foreign Key(<Tên
Field khoá ngoại>)
References <Tên bảng chứa khoá chính cần quan hệ>(<Tên Field
khoá chính>),...
```

Ví dụ Cho các bảng sau:

```
Khoa(MaKH, TenKH) và SinhVien(MaSV, HoSV, TenSV, MaKH)
Bảng SinhVien được tạo như sau:
Use QLSV
GO
Create Table SinhVien(MaSV char(3), HoSV Varchar(50),
TenSV Varchar(10), Constraint PKSinhvien Primary
Key(MaSV), Constraint FK_Kh_SV Foreign Key(MaKH)
References Khoa(MaKH))
GO
```

b) Sử dụng Object Explorer:

- Chọn Database \ chọn Tables \ Click chuột phải \ chọn New Table Tạo Table

6) *Xoá bảng:*

```
DROP TABLE <Table_Name>
```

Ví dụ:

```
Drop Table Monhoc
```

II. SỬA ĐỔI CẤU TRÚC BẢNG:

1) *Thay đổi kiểu dữ liệu cho cột trong Bảng:*

Cú pháp:

```
ALTER TABLE <Table_name>
ALTER COLUMN <Column_Name> <Kiểu dữ liệu mới> [Null | Not Null]
```

Chú ý:

- Nếu trường hợp muốn đổi tên cho một cột, thì thực hiện hai thao tác: Xoá cột cũ và thêm cột mới

Ví dụ: Sửa đổi kiểu dữ liệu của cột SoTiet trong bảng thành SmallInt

```
Alter Table Monhoc
Alter Column SoTiet SmallInt
```

2) *Thêm cột vào bảng:*

Cú pháp:

```
ALTER TABLE <Table_name>
ADD <Column_Name> <Kiểu dữ liệu mới> [Null | Not Null]
```

Ví dụ: Thêm cột TongMT vào bảng SinhVien

```
Alter Table SinhVien
Add TongMT TinyInt
```

3) *Thêm ràng buộc vào bảng:*

Cú pháp:

```
ALTER TABLE <Table_name>
ADD CONSTRAINT . . . ., CONSTRAINT . . . . . .
```

Ví dụ: Thêm ràng buộc khóa ngoại vào bảng SinhVien như sau:

```
Alter Table SinhVien
Add Constraint FK_Kh_SV Foreign Key(MaKH) References
Khoa(MaKH)
```

4) *Xoá ràng buộc*

Cú pháp:

```
ALTER TABLE <Table_name>
DROP CONSTRAINT . . . .
```

Ví dụ: Xóa ràng buộc khóa ngoại trong bảng SinhVien:

```
Alter Table SinhVien
Drop Constraint FK_Kh_SV
```

5) *Xoá cột:*

Cú pháp:

```
ALTER TABLE <Table_name>
DROP COLUMN <Column_Name>
```

Ví dụ: Xóa cột TongMT trong bảng SinhVien

```
Alter Table SinhVien
Drop Column TongMT
```

6) *Trường hợp hiệu chỉnh Field là khoá chính:*

Phải thực hiện các bước sau:

- Xoá dây quan hệ, bằng lệnh.
- Xoá bỏ ràng buộc khoá chính cho Field đó.
- Xoá Field khoá chính.
- Thêm các Field vào bảng.

a) Xoá dây quan hệ:

- Chỉ xoá từng dây, không cho phép xoá nhiều dây quan hệ.

Cú pháp:

```
ALTER TABLE <Tên bảng>
DROP CONSTRAINT <Tên quan hệ khoá ngoại>
```

Ví dụ:

```
Alter Table KetQuaThi Drop Constraint FK_KQ_SV 'Dây quan hệ với bảng Dm_SinhVien
Alter Table KetQuaThi Drop Constraint FK_KQ_MH 'Dây quan hệ với bảng Dm_MonHoc
```

b) Xoá bỏ ràng buộc khoá chính:

```
ALTER TABLE <Tên bảng>
DROP CONSTRAINT <Tên ràng buộc khoá chính>
```

Ví dụ:

```
Alter Table KetQuaThi Drop Constraint PK_KetQua
```

c) Xoá Field khoá chính đó:

- Cho phép xoá nhiều cột cùng một lúc.

```
ALTER TABLE <Tên bảng>
DROP COLUMN <Field 1 tham gia khoá chính> [,<Field 2 tham gia khoá chính>,...]
```

Ví dụ:

```
Alter Table KetQuaThi Drop Column MaMon, MaSv
```

d) Thêm các Field vào bảng bằng lệnh:

```
ALTER TABLE <Tên bảng>
ADD COLUMN <Tên Field khoá chính><Kiểu Dữ liệu>,...
```

Ví dụ:

```
Alter Table KetQuaThi
Add Column Masv VarChar(3), MaMon VarChar(4),
Constraint K_Chinh Primary Key (Masv, MaMon)
```

III. TẠO VÀ HIỆU CHỈNH CÁC RÀNG BUỘC DỮ LIỆU TRÊN BẢNG:

1) *Ràng buộc duy nhất:*

Là ràng buộc sao cho dữ liệu trên cột này không được trùng, dù không phải là khoá chính của bảng.

Ví dụ: cột Số CMND của nhân viên thì không được trùng, mặc dù khoá chính là MaNV.

Cú pháp:

```
CREATE TABLE <Table_Name> ( ...
<Column_Name> <Kiểu dữ liệu> Not Null | Unique )
ALTER TABLE <Table_Name>
Add Constraint <Constraint_Name> Unique (<Tên cột>)
```

Ví dụ: Tạo ràng buộc duy nhất cho cột TenMH trong bảng MonHoc

```
Alter Table Monhoc
Add Constraint UNI_TenMH Unique(TenMH)
```

2) *Ràng buộc kiểm tra:*

Là ràng buộc miền giá trị trên cột.

Cú pháp:

```
ALTER TABLE <Table_Name>
ADD Constraint <Constraint_Name> Check (<Biểu thức>)
```

Ví dụ:

```
Alter Table KetQuaThi
Add Constraint KT_DiemDau Check (DiemDau Between 15 and
22)
```

3) *Ràng buộc giá trị mặc định:*

Cú pháp:

```
ALTER TABLE <Table_Name>
ADD Constraint <Constraint_Name> Default (<Giá trị>) For <Tên
cột>
```

Ví dụ: Tạo ràng buộc Default cho cột Sotiet trong bảng Monhoc là 60

```
Alter Table Monhoc
Add Constraint Def_Sotiet 60 For Sotiet
```

IV. CÁC HÀM THƯỜNG DÙNG

1) *Hàm thống kê:*

Gồm các hàm Min, Max, Count …

2) *Hàm chuỗi:*

Gồm các hàm Upper, Lower, Len, Left, Right ...

3) *Hàm thời gian:*

- Để lấy ngày hiện hành: GetDate()
- Day, Month, Year: được dùng để lấy ngày, tháng, năm của dữ liệu kiểu ngày tháng.
- **Hàm DatePart**: trả về giá trị là thời gian tùy theo tham số DatePart.
  - *Cú pháp:*

```
DatePart(<DatePart>,<Ngày tháng>)
```

  - *Tham số Datapart: yy, mm, dd*
  - *Ví dụ:*

```
DatePart(yy, '05/12/2005')
Kết quả nhận được là: 2005
```

- **Hàm DateDiff**: trả về giá trị là hiệu số khoảng thời gian giữa ngày bắt đầu và ngày kết thúc tùy theo tham số DatePart
  - *Cú pháp:*

```
DateDiff(<DatePart>,<Ngày bắt đầu>,<Ngày kết thúc>)
```

  - *Ví dụ:*

```
DateDiff(yy,'05/12/2005','05/16/2005')
Kết quả nhận được là: 4
```

- **Hàm DateAdd**: trả về giá trị là thời gian cộng thêm vào một ngày tháng tùy theo tham số DatePart.
  - *Cú pháp:*

```
DateAdd(<datePart>,<Số>, <Ngày tháng>)
```

  - *Ví dụ:*

```
DateAdd(dd, 2,'12/10/2004')
Kết quả nhận được là: 12/12/2004
```

- **Hàm DateName**: trả về giá trị là thứ trong tuần.
  - *Cú pháp:*

```
DateName(<DatePart>,<Ngày tháng>)
```

- *Ví dụ:*

```
DateName(dw, '7/11/2007') => Kết quả là: WednesDay.
```

4) *Các hàm chuyển kiểu dữ liệu:*

- **Hàm Convert:** thường được dùng để chuyển đổi dữ liệu kiểu số hay dữ liệu kiểu ngày tháng sang chuỗi.
  - *Cú pháp:*

```
Convert(<Kiểu dữ liệu>,<Biểu thức>,<mã định dạng>)
```

  - *Ví dụ:*

```
Convert(char(10), '05/17/1998', 103)
Kết quả là chuỗi: '17/05/1998'
```

- **Hàm Cast:** thường dùng để chuyển dữ liệu kiểu số sang chuỗi.
  - *Cú pháp:*

```
Cast(<Biểu thức> As <Kiểu dữ liệu>)
```

  - *Ví dụ:*

```
Cast(12 As varchar(10))
```

- **Hàm IsNull:** được dùng để thay thế giá trị Null trong biểu thức bằng giá trị khác.
  - *Cú pháp:*

```
IsNull(<Biểu thức>,<Giá trị thay thế>)
```

  - *Ví dụ:*

```
IsNull(Sum(soluong),0)
```

5) *Cấu trúc Case:*

Cú pháp 1:

```
Case <Biểu thức>
When <Giá trị 1> Then <Kết quả 1>
When <Giá trị 2> Then <Kết quả 2>
...
Else
<Kết quả N>
End
```

Ví dụ:

```
Select ..., PhaiSv = Case  Phai When 1 Then 'Nam'
Else 'Nữ'
End
```

V. CÁC TRUY VẤN CƠ BẢN VỚI T-SQL:

1) *Khái niệm về SELECT:*

Cú pháp:

```
SELECT <*|<Danh sách các cột, biểu thức>
FROM <Danh sách các Table>
WHERE <Điều kiện>
GROUP BY <Danh sách các cột, biểu thức làm tiêu chuẩn gom nhóm>
HAVING <Điều kiện trên nhóm>
ORDER BY <Danh sách các cột, biểu thức làm tiêu chuẩn sắp xếp>
```

a) Các phép toán thường dùng:

- Phép toán so sánh:
  - != (Hay <>): So sánh khác
  - !> (hay <=): So sánh bé hơn hay bằng.
  - !< (hay >=): So sánh lớn hơn hay bằng.
  - =: So sánh bằng
- Phép toán nối chuỗi: sử dụng phép toán +
- Phép toán tập hợp: In, Not In dùng để so sánh thuộc hay không thuộc về một tập hợp.
- Phép toán Like: So sánh gần đúng một chuỗi. Ký tự đại diện sử dụng "%"
- Phép toán Exists: Đây là một phép toán Logic:
  - *Trả về giá trị TRUE nếu có ít nhất một dòng tồn tại trong câu SQL là tham số của phép toán này.*
  - *Trả về FALSE trong trường hợp ngược lại*
- Phép toán BETWEEN . . . AND: So sánh thuộc về một khoảng đóng.

Ví dụ: Xuất ra các khoa không có sinh viên

```
SELECT Dm_Khoa.Makh,Dm_Khoa.Tenkh
FROM Dm_Khoa
WHERE Not Exists (SELECT * From DMSV Where
Dm_khoa.Makh=DmSV.Makh)
```

2) *Các phát biểu SELECT khác:*

a) Select với AS:

- Sử dụng từ khóa AS để đặt tên cho cột mới hay biểu thức.

```
SELECT <Danh sách cột>, <Biểu thức> AS <Tên cột mới>
```

b) Select với Top N:

- Cho phép lấy ra N mẫu tin đầu tiên (có thể không sử dụng mệnh đề Order By)

Cú pháp:

```
SELECT TOP N <Danh sách cột>
SELECT TOP N [With Ties] From ... Order By
```

Ví dụ: Xuất ra 5 sinh viên có học bổng cao nhất

```
Select Top 5 With Ties MaSv, HoSV, TenSv
From SinhVien
Order By Hocbong Desc
```

Chú ý: *Nếu dùng Top N With Ties: kết quả là các dòng tương ứng với N giá trị đầu tiên theo mệnh đề Order By*

c) Select với Distinct:

- Dùng để loại bỏ các mẫu tin trùng lắp nhau

Cú pháp:

```
SELECT Distinct <Danh sách các cột>
```

Ví dụ: Xuất ra các sinh viên có điểm từ 9 trở lên

```
Select Distinct Kq.MaSv, HoSV, TenSv from SinhVien Sv
Inner Join Ketqua Kq
On Sv.MaSv=Kq.MaSv where Diem>=9
```

3) Cập nhật dữ liệu cho bảng:

a) Thêm dữ liệu vào bảng với giá trị cụ thể:

Cú pháp:

```
INSERT INTO <Table_name>[(<D_sách cột>)]VALUES(<Giá trị các cột>)
```

Ví dụ: Thêm một dòng dữ liệu vào bảng Khoa như sau:

```
Insert Into Khoa(Makh,TenKhoa) Values('MT','Môi trường')
```

Chú ý:

- Trường hợp thêm dữ liệu với tất cả các cột thì ta không cần chỉ ra danh sách tất cả các cột theo cú pháp trên.

  Thêm dữ liệu tiếng Việt Unicode, trước dữ liệu có kí tự N. Vd: `N'Môi trường'`

Ví dụ: Thêm một dòng dữ liệu vào bảng Khoa như sau:

```
Insert Into Khoa Values('MT','Môi trường')
```

b) Thêm dữ liệu vào bảng từ bảng khác:

Cú pháp:

```
INSERT INTO <Table_Name> [<Danh sách cột>]
SELECT [<Danh sách cột>]
FROM <Table_Name>
WHERE <Điều kiện>
```

Ví dụ: Thêm các sinh vien thuộc khoa Tin học vào bảng Sinhvien_TH đã có:

```
Insert Into Sinhvien_TH
Select * From Sinhvien Where MaKH='TH'
```

c) Cập nhật dữ liệu trong bảng:

Cú pháp:

```
UPDATE <Table_Name>
SET <Column_Name> = <Giá trị / Biểu thức>
WHERE <Điều kiện>
```

Ví dụ: Tăng học bổng sinh viên khoa Anh văn thêm 10000

```
Update SinhVien
Set HocBong=HocBong+10000
Where MaKH='AV'
```

d) Tạo bảng mới với dữ liệu từ bảng có sẵn:

Cú pháp:

```
SELECT <Danh sách cột> INTO <Table_Name>
FROM <Table_Name>
WHERE <Điều kiện>
```

Ví dụ: Tạo bảng HocBong_Khoa chứa tổng học bổng của từng Khoa:

```
Select SV.MaKh, TenKH, Sum(HocBong) As TongHB Into
HocBong_Khoa
From SinhVien SV Inner Join Khoa KH on KH.MaKh=SV.MaKH
Group By SV.MaKh, TenKH
```

e) Xoá dữ liệu:

Cú pháp:

```
DELETE FROM <Table_Name>
WHERE <Điều kiện>
```

Ví dụ: Xóa các sinh viên chưa thi môn nào

```
Delete From SinhVien
where Not Exists (Select * From Ketqua where
MaSV=Sinhvien.MaSV)
```

Cú pháp 2:

```
Case
When (<Biểu thức>) Then <Kết quả 1>
When (<Biểu thức>) Then <Kết quả 2>
...
Else
<Kết quả N>
End
```

Ví dụ:

```
Select ..., GhiChu = Case
When HocBong > 50 Then 'Giỏi'
When HocBong > 40 Then 'Khá'
Else  'Không xét'
End
```

Chương 4 BẢNG ẢO (VIEW) – THỦ TỤC NỘI TẠI (STORE PROCEDURE) TRONG SQL SERVER

I. BẢNG ẢO (VIEW):

1) *Khái niệm:*

- Trong quá trình xử lý dữ liệu, có khi cần kết nối những bảng có quan hệ với nhau để kết xuất dữ liệu theo ý muốn. Để thực hiện công việc này ta có thể dùng các phát biểu SQL, nhưng nếu dùng như thế thì cấu trúc của phát biểu SQL không cho phép lưu trữ như một đối tượng của SQL Server.
- Nếu ta dùng bảng ảo thì kết xuất cũng tương tự như phát biểu SQL, nhưng bảng ảo được xem là một đối tượng của SQL Server nên ta có thể quản trị, xử lý dễ dàng theo yêu cầu của người dùng.

Tạo một bảng ảo có những lợi ích như sau:

- Kiểm soát được những gì người dùng có thể thấy.
- Đơn giản hóa giao diện người dùng bằng cách tạo ra các bảng ảo với các truy vấn.
- Đảm bảo an toàn người dùng khi thao tác với các dòng, cột theo yêu cầu.

2) *Tạo bảng ảo:*

Cú pháp:

```
CREATE VIEW <View_Name>
[With Encription]
as <Phát biểu Select>
[With Check Option]
```

- **With EnCription**: nếu dùng thì bảng ảo sẽ được mã hoá.
- **With Check Option**: Nếu dùng thì khi thao tác trên bảng ảo như thêm dữ liệu, cập nhật dữ liệu, phải tuân theo giới hạn câu Select.

Ví dụ:

```
Create View Add_MonHoc
As Select Mamon,...,SoTiet,...
From Dm_MonHoc
Where SoTiet > 20
With Check Option
```

Nếu sau câu lệnh này, mà ta thêm dữ liệu vào bảng Dm_MonHoc bằng tên bảng ảo MonHoc mà cột SoTiet không thoả điều kiện ở trên => Báo lỗi.

Ví dụ:

```
Insert Into Add_MonHoc Values ('TS', 'Tau song', 10, 1)
```

3) *Sử dụng bảng ảo để thay đổi dữ liệu:*

Khi sửa đổi dữ liệu trong một bảng ảo thì dữ liệu của một bảng cơ sở bên dưới sẽ bị thay đổi. Sửa đổi dữ liệu với bảng ảo phải tuân theo các quy tắc sau:

- Sự sửa đổi không để ảnh hưởng nhiều hơn một bảng bên dưới. Nếu bảng ảo kết hợp nhiều bảng thì ta có thể sửa dữ liệu của một trong các bảng.
- Nếu dùng tùy chọn With Check thì mọi thay đổi trên bảng ảo phải tuân theo giới hạn của câu Select.

4) *Thay đổi bảng ảo:*

Cú pháp:

```
ALTER VIEW <View_Name>
<...>
```

Ví dụ: Sửa lại bảng ảo trong Ví dụ trên.

```
Alter View Add_MonHoc As
Select * From Dm_MonHoc Where SoTiet > 30
With Check Option
```

5) *Xoá bảng ảo:*

Cú pháp:

```
DROP VIEW <View_Name>
```

Ví dụ: Xóa bảng ảo Vw_ThemMH

```
Drop View Vw_ThemMH
```

6) *Xem thông tin về bảng ảo:*

Cú pháp:

```
EXEC Sp_HelpText <View_Name>
```

Ví dụ: Xem thông tin nội dung của bảng ảo

```
EXEC Sp_helpText Vw_XoaMH
```

7) *Các mối liên kết thường dùng với bảng ảo:*

a) Liên kết tương đương:

- Là mối liên kết mà bảng kết quả sẽ bao gồm tất cả các dòng có giá trị của trường liên kết xuất hiện ở cả hai bảng.

Cú pháp:

```
Select ... from <Tên Table 1> Inner Join <Tên Table 2>
On <Tên Table 1>.<Trường liên kết>=<Tên Table
2>.<Trường liên kết>
...
```

b) Liên kết ưu tiên một phía:

- Là mối liên kết mà bảng kết quả sẽ bao gồm tất cả các dòng bên bảng được ưu tiên, tại những trường bên bảng còn lại không có liên kết thì sẽ nhận giá trị là Null.

Cú pháp:

```
Select ... from <Tên Table 1> Left[Right] Join <Tên Table 2>
On <Tên Table 1>.<Trường liên kết>=<Tên Table 2>.<Trường liên kết>
```

II. KỊCH BẢN (SCRIPT) VÀ LÔ (BATCH):

1) *Khái niệm Script:*

Script là một tập hợp một hay nhiều câu lệnh SQL được lưu dưới dạng các tập tin có phần mở rộng là .SQL.

2) *Khai báo biến trong Sql Server:*

Khi có nhu cầu cần tính toán, ta có thể khai báo biến theo cú pháp như sau:

```
Declare <@Tên biến> <Kiểu dữ liệu> [,...]
```

Ví dụ:

```
Declare @masv varchar (3)
```

Để gán giá trị cho biến ta sử dụng cú pháp sau:

```
Set <Tên biến > = <giá trị>
```

Ví dụ:

```
Set @masv = "A05"
```

Ghi chú:

- Ta có thể gán cho một biến là giá trị trả về của một câu Select. Giá trị trả về của câu Select phải là duy nhất một giá trị

Ví dụ:

```
Declare @ sosv Smallint
Set @Sosv = (Select Count(*) from Dm_SinhVien Where Makh='TH')
Print @Sosv
```

3) *Lô (Batch):*

a) Khái niệm:

Lô là tập hợp các câu lệnh SQL được gửi đến Server để xử lý cùng một lúc.

b) Một số nguyên tắc:

- Khi trong lô có lệnh gây ra lỗi cú pháp(Syntax Error) thì SQL Server sẽ không biên dịch được và kết quả là không có lệnh nào được thi hành.
- Khi trong lô có lệnh gây ra lỗi thi hành (run-time error) thì các lệnh trước lệnh gây ra lỗi không bị ảnh hưởng, còn từ lệnh đó trở về sau thì có khả năng không thực hiện được.
- Khi sử dụng các lệnh Create View, Create Store Procedure, Create Trigger, Alter column, ... thì các lệnh này không thể kết hợp với các lệnh khác trong cùng một lô. Nói cách khác thì các lệnh này phải đặt trong các lô khác nhau.

Ví dụ:

```
Use QL_SinhVien
Go
Select * From Dm_SinhVien
Go
Select * From Dm_SinhVien
Go
```

Ví dụ:

```
Create Table TTT ……
Go
Alter Table TTT
Add N VarChar(5)
Go
```

4) *SQL động:*

- Là câu SQL không được định nghĩa trước, chẳng hạn tên bảng được đưa từ bên ngoài vào hay có được từ một phép toán gán nào đó
- Để thực hiện câu SQL động, ta thực hiện cú pháp sau:

```
EXEC (<Chuỗi lệnh SQL>)
```

Ví dụ:

```
Declare @TblName VarChar(20), @MyStr VarChar(200)
Set @TblName='Dm_SinhVien'
Set @MyStr= 'Select * From ' + @TblName
Set @MyStr = @MyStr + '  Where hocBong>5000 '
Set @MyStr = @MyStr + '  And Makh="TH" '
Exec (@MyStr)
```

III. CẤU TRÚC ĐIỀU KHIỂN TRONG SQL SERVER:

1) *Cấu trúc IF:*

Cú pháp:

```
IF (<Biểu thức điều kiện>)
[Begin]
<Câu lệnh | Khối lệnh>
[End]
[Else
[Begin]
<Câu lệnh | Khối lệnh>
[End]]
```

Ví dụ:

Nếu biến @Flag = 1 thì thêm một dòng vào Dm_MonHoc, ngược lại thêm một dòng vào Dm_Khoa

```
Declare @Flag  Bit
Set @Flag = 1
If @Flag = 1
Insert Into Dm_MonHoc Values (@f1, @f2, @f3)
Else
Insert Into Dm_Khoa Values (@f1, @f2)
```

2) *Cấu trúc While:*

Cú pháp:

```
While <Biểu thức điều kiện>
[Begin]
<Câu lệnh SQL | Khối lệnh>
...
[Break]
...
[Continue]
[End]
```

Ví dụ:

```
Declare @i Int
Set @i = 1
While (@i <=10)
Begin
Print @i
Set @i = @i + 1
End
```

IV. STORE PROCEDURE (THỦ TỤC NỘI TẠI):

1) *Khái niệm:*

- Trong SQL Server khi thực hiện các tác vụ trong CSDL thì người ta thường dùng đối tượng Store Procedure.
- Store Procedure được lưu trữ như một phần của CSDL và ta có thể gọi thực hiện như các thủ tục khác.

2) *Xây dựng Store Procedure đơn giản:*

Cú pháp:

```
CREATE PROC <Procedure_Name>
With ReCompile
As
<Câu lệnh SQL>
```

Ví dụ: Xây dựng Store Procedure để tạo bảng Monhoc

```
Create Proc sp_ThemMH
As
Insert Into MonHoc Values('05' , 'Toán', 60)
```

3) *Thay đổi Store Procedure:*

Cú pháp:

```
Alter proc <Proc Name>
As . . .
```

Ví dụ:

```
Alter Proc sp_ThemMH
As Insert Into MonHoc Values('06' , 'Hóa', 60)
```

4) *Xóa Store Procedure:*

Cú pháp:

```
Drop Proc <Proc Name>
```

Ví dụ:

```
Drop Proc sp_themMH
```

5) *Xem nội dung của Store Procedure:*

Cú pháp:

```
Exec Sp_HelpText <Proc Name>
```

Ví dụ:

```
Exec Sp_HelpText sp_ThemMH
```

6) *Thi hành một Store Procedure:*

Cú pháp:

```
Exec <Proc Name>
```

Ví dụ:

```
Exec sp_ThemMH
```

7) *Sử dụng Store Procedure với tham số:*

Trong các Store Procedure, ta có thể dùng cho các tham số do người dùng truyền vào, hay do một Store khác gởi đến. Có hai loại tham số:

- Tham số nhập.
- Tham số xuất (OutPut) là dạng tham biến.

a) Khai báo tham số trong Store Procedure:

- *Cú pháp:*

```
Create Proc <Tên Store Procedure >
@ Tham số 1 < Kiểu dữ liệu>,
@ Tham số 2 < Kiểu dữ liệu>,
...,
@ Tham số n < Kiểu dữ liệu> OUTPUT
As <Nội dung Store Procedure >
```

- ***Ví dụ:*** *Tạo một Store Procedure để thêm một môn học vào danh mục môn học với tham số.*

```
Create Proc Sp_Them_Mh
@Mmh VarChar(2),
@Tenmh VarChar(20),
@SoTiet SmallInt,
@MonThi Bit
As
Insert Into Dm_monHoc Values (@Mmh, @Tenmh, @Sotiet,
@MonThi)
Go
Exec Sp_Them_Mh '09', 'Quân sự', 60, 0
Go
Select * From Dm_MonHoc
```

b) Dùng giá trị mặc nhiên trong Store Procedure:

- Khi khai báo Store Procedure ta có thể dùng giá trị mặc nhiên cho tham số. Khi gọi thi hành Store Procedure này, nếu tham số không được truyền giá trị thì giá trị mặc nhiên sẽ được dùng

- *Cú pháp:*

```
CREATE PROC <Tên thủ tục>
@<Tên tham số> <Kiểu dữ liệu>=<Giá trị>,...
```

- *Chú ý: Khi truyền tham số thì ta có thể dùng cú pháp như trên hoặc dùng cú pháp sau:*

```
<Tên Tham số> = <Giá trị>
```

- *Ví dụ:*

```
Use Ql_SinhVien
Go
Exec Sp_Them_Mh @Mmh='09', @Tenmh='Quân sự', @SoTiet=60,
@MonThi=0
```

```
Use Ql_SinhVien
Go
Exec Sp_Them_Mh @Tenmh='Quân sự', @Mmh='09', @MonThi=0,
@SoTiet=60
```

```
Use Ql_SinhVien
Go
Exec Sp_Them_Mh @Mmh='09', @SoTiet=60
```

```
Use Ql_SinhVien
Go
Exec Sp_Them_Mh @Mmh='09', 'Quân sự', 60
```

8) *Dùng tham số OUTPUT trong Store Procedure:*

Tham số này được dùng để xuất giá trị khi Store Procedure thực hiện xong, hoặc dùng giá trị này làm tham số cho các Store khác.

Cú pháp:

```
<Tên tham số> <Kiểu dữ liệu> Output
```

Ví dụ: Tạo procedure Tinh thực hiện tính tổng 2 số

```
Create Proc Tinh
@So1 smallint
@So2 smallint
@So smallint Output
```

```
As Select @So= @So1 + @So2
```

Ví dụ: Gọi thi hành

```
Declare @MySo SmallInt
Exec Tinh 2,3, @MySo Output
Print 'Số được tính kết quả là: ' + @MySo
```

Lệnh Print trên sẽ gây lỗi, vì @MySo là kiểu số, do vậy ta phải dùng các cách sau:

- **Hàm Convert**: Convert (VarChar(10), @<Tên biến>)
- **Hàm Str**: Str (@<Tên biến>[, <Độ rộng>])
- **Hàm RTrim**: RTrim (@<Tên biến>).

9) Dùng giá trị trả về RETURN:

Bên trong một Store Procedure ta có thể gán giá trị trả về là một số nguyên nào đó.

Ví dụ:

```
Create Proc MyReturn
@So1 smallint
@So2 smallint
@So smallint Output
Set @So = @So1 + @So2
Return @So
```

Ví dụ: Gán Store cho biến.

```
Declare @Bien SmallInt
Exec @Bien = MyReturn 9 ,9 ,0
Print @Bien
```

10) Dùng Store Procedure với từ khoá With Recompile:

Khi dùng tuỳ chọn này, thì toàn bộ Store Procedure sẽ được biên dịch lại mỗi khi thi hành

Ví dụ:

```
Create Proc MyCompile
@Bien1 SmallInt,...
With ReCompile
As
     Select ...
```

11) Dùng Store Procedure với từ khoá With Encription:

Dùng để mã khoá cho một Store Procedure

Ví dụ:

```
Create Proc MyCompile
@Bien1 SmallInt,...
...
With EnCription
As
     Select ...
```

Ghi chú: Ta có thể dùng cú pháp sau để biên dịch lại Store Procedure cho riêng lần thi hành.

```
EXEC <Tên Store Procedure > With ReCompile
```

Chương 5 TRIGGER & BIẾN CURSOR VÀ HÀM DO NGƯỜI DÙNG ĐỊNH NGHĨA

I. TRIGGER:

1) *Khái niệm:*

- Là một đối tượng được lưu trữ trong CSDL, thường được dùng để kiểm tra tính hợp lý, tính đồng bộ khi dữ liệu có sự thay đổi.
- Trigger cũng tương tự như các Store Procedure, bên trong cũng chứa các câu lệnh SQL dùng để thực hiện một số hành động nào đó.
- Điểm khác biệt giữa Trigger và Store Procedure là Trigger không có tham số và Trigger sẽ tự động được thi hành khi dữ liệu trong các bảng có sự thay đổi dữ liệu.
- Trong quá trình dữ liệu bị thay đổi, ta có thể thực hiện lệnh quay lui (ROLLBACK TRAN) trong Trigger để hủy bỏ kết quả của các hành động thêm, xoá, sửa.

2) *Một số ứng dụng của Trigger:*

- Cập nhật tự động dữ liệu giữa các bảng có liên quan: Hành động này nhằm đảm bảo sự thay đổi dữ liệu giữa các bảng có liên quan được đồng bộ.
  - ***Ví dụ**: Khi xóa một khoa trong bảng Khoa thì phải xóa luôn các sinh viên thuộc về Khoa đó trong bảng Sinhvien.*
- Kiểm tra các ràng buộc dữ liệu phức tạp: Khác với đối tượng Constraint chỉ kiểm tra dữ liệu trên một bảng, Trigger có thể kiểm tra dữ liệu của các trường trong một hay nhiều bảng khác nhau.
- Các bảng Logic INSERTED và DELETED:
  - *Khi thêm mới một mẫu tin, dữ liệu thêm mới sẽ được đặt trong bảng INSERTED.*
  - *Khi xóa một mẫu tin, dữ liệu xóa sẽ được đặt trong bảng DELETED.*
  - *Khi sửa một mẫu tin, dữ liệu mới sẽ được đặt trong bảng INSERTED, còn dữ liệu cũ được đặt trong bảng DELETED.*

3) *Các thao tác trên Trigger:*

a) Tạo Trigger:

Cú pháp:

```
CREATE TRIGGER <Trigger_Name>
ON <Table_Name>
{FOR | AFTER} | INSTEAD OF INSERT[ DELETE , UPDATE]
AS
<Lệnh>
Go
```

- **Table_Name**: là tên bảng mà khi có sự cập nhật dữ liệu thì Trigger sẽ được thi hành.

- **{FOR | AFTER}**: Đây là loại Trigger được thực hiện sau khi việc cập nhật dữ liệu đã hoàn tất xong (kể cả việc kiểm tra các ràng buộc do đối tượng Constraint thực hiện). Trigger dạng này thường được sử dụng khi muốn đồng bộ dữ liệu trong các bảng có liên quan.
- **INSTEAD OF:** Đây là loại Trigger có thể áp dụng được cả trên bảng và bảng ảo.

Lưu ý:

- Đối với các bảng hay bảng ảo sử dụng Trigger dạng này thi khi có hành động cập nhật lại dữ liệu trên bảng hay bảng ảo, các hành động này sẽ không được thực hiện mà Trigger sẽ được thực hiện thay thế.
- Đối với các bảng mà tại sự kiện ON DELETE và ON UPDATE có giá trị là CASCADE thì ta không thể sử dụng Trigger loại này cho các thao tác Update và Delete.
- INSERT, UPDATE, DELETE: là các từ khóa cho biết Trigger sẽ được thực hiện tương ứng với hành động Thêm (Insert), Xóa (Delete), Sửa (Update).

Ví dụ: Tạo Trigger thực hiện kiểm tra khi thêm mới một môn học

```
Create Trigger Tg_ThemMH
On MonHoc
InStead Of Insert
As
Declare @ mmh char(2), @tmh varchar(50), @st SmallInt
Select @mmh=MaMH, @tmh=TenMH, @st=Sotiet From Inserted
If Exists(Select * from Monhoc where MaMH=@mmh)
Begin
     Print 'Mã môn học đã tồn tại'
     Return
End
Insert Into Monhoc Values(@mmh, @tmh, @st)
GO
```

b) Sửa Trigger:

Cú pháp:

```
ALTER TRIGGER ...
```

Ví dụ: Ở ví dụ trên, có thể sửa lại nội dung bên trong và thay từ khóa Create thành Alter.

c) Xoá Trigger:

Cú pháp:

```
DROP TRIGGER <Trigger_Name>
```

Ví dụ:

```
Drop Trigger Tg_ThemMH
```

II. KIỂU DỮ LIỆU CURSOR:

1) *Khái niệm:*

- Cursor là một kiểu dữ liệu cho phép tham chiếu và duyệt lần lượt qua một tập hợp các mẫu tin trong khi xử lý các thao tác. Cursor còn được sử dụng làm tham số xuất trong các Store Procedure.
- Người ta thường dùng kiểu dữ liệu này khi có nhu cầu thao tác trên từng mẫu tin hoặc xử lý một công việc nào đó trên tập hợp các mẫu tin.

2) *Các thao tác trên biến Cursor:*

a) Khai báo biến Cursor:

Cú pháp:

```
Declare <Tên biến> Cursor [Scroll] For <Câu Select>
```

Ví dụ:

```
Declare Cur_KH Cursor Scroll For Select * From Dm_Khoa
```

b) Mở biến Cursor:

```
OPEN <Tên biến Cursor>
```

Ví dụ:

```
Open Cur_MH
```

c) Lấy kết quả của một mẫu tin đưa vào các biến:

Cú pháp:

```
FETCH <Tham số> From <Biến
Cursor> INTO <D_sách biến>
```

- Với tham số có các giá trị sau:
  - *Next: lấy kết quả của mẫu tin kế tiếp.*
  - *Prior: lấy kết quả của mẫu tin phía trước.*
  - *Last: lấy kết quả của mẫu tin cuối cùng.*
  - *First: lấy kết quả của mẫu tin đầu tiên.*

Ví dụ:

```
Declare @mmh char(2), .....
Fetch Next From Cur_MH Into @mmh, .....
```

d) Đóng biến Cursor:

Cú pháp:

```
CLOSE <Tên biến Cursor>
```

Ví dụ:

```
Close Cur_MH
```

e) Giải phóng biến Cursor:

Cú pháp:

```
DEALLOCATE <Tên biến Cursor>
```

Ví dụ:

```
DeAllocate Cur_MH
```

Chú ý:

- <Tên biến Cursor> không có dấu hiệu @
- Khi khai báo SCROLL thì có thể di chuyển theo cả hai chiều.
- Để kiểm tra việc đọc dữ liệu từ biến Cursor có thành công hay không, ta sử dụng biến hệ thống @@FETCH_STATUS. Nếu giá trị của biến là 0 thì việc đọc thành công, ngược lại là thất bại.

Ví dụ minh họa:

```
Declare Cur_MH For Select * from Monhoc
Declare @mmh char(2), @tmh varchar(50), @st SmallInt
Open Cur_MH
Fetch Next From Cur_MH Into @mmh, @tmh, @st
While @@Fetch_Status =0
Begin
     Print @mmh+@tmh+Cast(@st As Char(10))
     Fetch Next From Cur_MH Into @mmh, @tmh, @st
End
Close Cur_MH
DeAllocate Cur_MH
```

III. BIẾN KIỂU BẢNG:

1) *Khái niệm:*

Trong các phiên bản SQL Server sau này thì có một kiểu dữ liệu mới được bổ sung là kiểu dữ liệu bảng dùng để lưu trử tạm thời một tập hợp các mẫu tin khi thực hiện các thao tác. Kiểu dữ liệu này thường được dùng khi xây dựng các hàm tạo bảng.

2) *Cú pháp khai báo:*

```
Declare @<Tên biến> TABLE (<Danh sách các cột>)
```

3) Ví dụ minh họa:

```
Declare @Tbl_MH Table(Mamh char(2), TenMH Varchar(50))
Insert Into @Tbl_MH
Select MaMh, TenMH from Monhoc where Sotiet>60
Select * From @Tbl_MH
```

IV. HÀM DO NGƯỜI DÙNG ĐỊNH NGHĨA (USER DEFINE FUNCTION):

1) Hàm đơn trị (Scalar Function):

Là hàm sẽ trả về một giá trị có kiểu là kiểu dữ liệu cơ sở.

Cú pháp:

```
CREATE FUNCTION <Tên hàm>(<Danh sách các tham số>)
RETURNS <Kiểu dữ liệu>
AS
     BEGIN
          <Thân hàm>
     RETURN <Giá trị trả về>
END
```

Ví dụ minh họa:

```
Create Function Tinhtoan(@so Int)
Returns Int
As
Begin
     Declare @m Int
     Set @m=@so + 1
     Return @m
End
Print DBO.Tinhtoan(3)
```

2) Hàm đọc bảng:

Hàm trả về một tập hợp các mẫu tin lấy từ bảng nào đó.

Cú pháp:

```
CREATE FUNCTION <Tên hàm>(<Danh sách tham số>)
RETURNS TABLE
AS
     RETURN (<Câu lệnh Select>)
```

Ví dụ:

```
Create Function Fn_Doc_MH(@st SmallInt)
Returns Table
```

```
As
Return(Select * from Monhoc where Sotiet>=@st)
Select * from Fn_Doc_MH(30)
```

3) *Hàm tạo bảng:*

Hàm trả về một biến kiểu bảng.

Cú pháp:

```
CREATE FUNCTION <Tên hàm>(<Danh sách các tham số>)
RETURNS @<Tên biến kiểu Table> TABLE (<Danh sách
các cột>)
AS
     BEGIN
          <Thân hàm>
     RETURN
END
```

Ví dụ:

```
Create Function Fn_Tao_Mh(@st SmallInt)
Returns @Tbl_TaoMH Table(MaMH char(2), TenMH Varchar(50))
As
Begin
Insert Into @Tbl_Tao_Mh
Select MaMH, TenMH from Monhoc Where sotiet>=@st
Return
End
Select * From Fn_Tao_Mh(30)
```

Chương 6 BẢO MẬT TRONG SQL SERVER

I. KHÁI NIỆM:

Nhằm tăng cường tính năng bảo mật dữ liệu . SQL Server đưa ra một số cơ chế bảo vệ CSDL trong môi trường nhiều người dùng. Một số cơ chế bảo mật thường dùng như:

- Xây dựng các tài khoản & nhóm người dùng trong hệ thống và CSDL.
- Cấp quyền người dùng trên CSDL trong hệ thống.
- Cấp quyền tạo và sửa đổi cấu trúc các đối tượng trong CSDL.

Khi đăng nhập vào một hệ thống CSDL đa người dùng , người dùng phải cung cấp tài khoản đăng nhập và mật khẩu . Dựa trên các yếu tố này , hệ thống sẽ kiểm tra tính hợp lệ của tất cả các thành viên người dùng trên hệ thống và cho phép truy xuất CSDL thích hợp.

II. CÁC CƠ CHẾ BẢO MẬT

Trong SQL Server thì tồn tại hai loại tài khoản là tài khoản đăng nhập và tài khoản người dùng trên CSDL cụ thể.

- **Tài khoản đăng nhập** : là tài khoản tạo ra trên hệ thống để cho phép người dùng có thể kết nối đến SQL Server
- **Tài khoản người dùng trên CSDL** : là tài khoản được tạo ra trên CSDL cụ thể cho phép người dùng có thể thao tác trên CSDL vói quyền hạn thích hợp . Tài khoản này thường gắn liền với tài khoản đăng nhập.

Khi tạo tài khoản đăng nhập trên SQL Server, có thể dùng các cơ chế sau:

- **Giao tiếp với hệ điều hành**: Cơ chế này cho phép sử dụng các tài khoản người dùng của hệ điều hành để đăng nhập vào SQL Server. Với cách này thì người dùng vừa có thể dùng các CSDL trong SQL Server vừa có thể truy cập đến các tài nguyên trên mạng.
- **Bảo mật chuẩn**: Cơ chế này sử dụng các tài khoản do SQL Server tạo ra và các tài khoản này độc lập với hệ điều hành mạng.
- **Cơ chế thứ 3**: là tổng hợp hai cơ chế trên.

III. BẢO MẬT VỚI CƠ CHẾ XÁC NHẬN CỦA SQL SERVER:

1) *Tạo mới một tài khoản đăng nhập:*

Cú pháp :

- Sử dụng trên SQL Server 2000

```
EXEC Sp_AddLogin <'username'>,<'password'>,<'database'>
```

- Sử dụng trên SQL Server 2005

```
CREATE LOGIN <username>
[WITH PASSWORD='password', DEFAULT_DATABASE = Database]
```

- Database là tên CSDL mặc định khi người dùng đăng nhập vào hệ t hống. Nếu bỏ trống thì CSDL mặc định là Master (Trường hợp muốn gán quyền tạo CSDL thì phải chọn CSDL mặc định là Master).

Ví dụ:

- Sử dụng trên SQL Server 2000

```
Exec Sp_AddLogin 'HuuPhuoc', 'aa', 'QL_SinhVien_xx'
```

- Sử dụng trên SQL Server 2005

```
CREATE LOGIN HuuPhuoc WITH PASSWORD='aa',
DEFAULT_DATABASE=QLSV
```

2) *Thay đổi Password:*

Cú pháp :

- Sử dụng trên SQL Server 2000

```
Sp_PassWord <'password cũ'>, <'password mới'>,
<'username'>
```

- Sử dụng trên SQL Server 2005

```
ALTER LOGIN  <username> WITH PASSWORD = 'password
mới'
```

Ví dụ:

- Sử dụng trên SQL Server 2000

```
Exec Sp_Password 'aa', 'xx', 'QL_SinhVien_xx'
```

- Sử dụng trên SQL Server 2000

```
Alter login  qlsv_xx  with password = 'xx'
```

3) *Thay đổi Cơ sở dữ liệu mặc định:*

Cú pháp :

- Sử dụng trên SQL Server 2000

```
Exec Sp_DefaultDB <'username'>, <'database'>
```

- Sử dụng trên SQL Server 2005

```
Alter Login <username> With Default_Database = database
```

Ví dụ:

- Sử dụng trên SQL Server 2000

```
Exec Sp_DefaultDB 'HuuPhuoc', 'QL_HoaDon_xx'
```

- Sử dụng trên SQL Server 2000

```
Alter Login HuuPhuoc With Default_Database = QL_HoaDon_xx
```

4) *Thay đổi tên đăng nhập (Chỉ áp dụng cho SQL Server 2005):*

Cú pháp:

```
Alter Login <username cũ> With Name = <username mới>
```

Ví dụ:

```
Alter Login HuuPhuoc with Name=phuocLuu
```

5) ***Vô hiệu hóa tài khoản đăng nhập(Chỉ áp dụng cho SQL Server 2005):***

Cú pháp:

```
Alter Login <username> {Enable|Disable}
```

Ví dụ:

```
Alter Login Phuocluu Disable
```

6) ***Xem thông tin về tài khoản được tạo:***

Cú pháp:

```
Exec Sp_HelpLogins <'username'>
```

Ví dụ:

```
Exec Sp_HelpLogin 'LT_01'
```

7) ***Xoá tài khoản:***

Cú pháp :

- Sử dụng trên SQL Server 2000

```
Exec Sp_DropLogin <'username'>
```

- Sử dụng trên SQL Server 2005

```
Drop Login <username>
```

Chú ý:

- Tạo tài khoản đăng nhập với cú pháp nào thì xóa với cú pháp tương ứng (nghĩa là sử dụng cú pháp SQL Server 2000 hay 2005 cho từng cặp lệnh tương ứng).

IV. BẢO MẬT VỚI CƠ CHẾ XÁC NHẬN CỦA WINDOWS:

Cho phép các tài khoản người dùng, tài khoản nhóm của Windows kết nối với SQL Server. Cơ chế này cho phép người dùng vừa có thể truy cập vào SQL Server, vừa có thể truy cập các tài nguyên trên mạng.

1) ***Gán tài khoản người dùng, tài khoản nhóm kết nối SQL Server:***

Cú pháp :

- Sử dụng trên SQL Server 2000

```
SP_GrantLogin [{Domain Name|ComputerName}\LoginName]
```

- Sử dụng trên SQL Server 2005

```
Create Login [<Domain|Computer>\<LoginName>] From
Windows [With Default_Database = Database]
```

Ví dụ:

```
Create Login [CNTT1\LT50] From Windows With
Default_Database=Sv
```

2) Xoá toàn bộ thông tin đăng nhập:

Cú pháp :

- Sử dụng trên SQL Server 2000

```
SP_RevokeLogin [@LoginName=]'username'
```

- Sử dụng trên SQL Server 2005

```
Drop Login <username>
```

3) Ngăn cản sự kết nối đến SQL Server:

Cú pháp :

- Sử dụng trên SQL Server 2000

```
SP_DenyLogin [@LoginName=] 'username'
```

- Sử dụng trên SQL Server 2005

```
Alter Login  <username>  Disable
```

V. NHÓM (ROLE) TRONG SQL SERVER VÀ CƠ SỞ DỮ LIỆU:

1) Nhóm hệ thống (Role Server) trên SQL Server:

Thường dùng với các nhóm sau:

- **SysAdmin**: Thành viên của nhóm này có toàn quyền trên SQL Server.
- **SecurityAdmin**: Thành viên của nhóm có thể Thêm, xoá, quyền truy cập của người dùng, tạo CSDL, thay đổi CSDL, đăng nhập máy chủ, …….
- **DbCreator**: thành viên của nhóm này có thể tạo CSDL.

2) Các lệnh liên quan đến Role Server:

- Gán một người dùng cho một Role Server:

```
Sp_AddSrvRoleMember <'username'>,'Role Server'>
```

Ví dụ:

```
Exec Sp_AddSrvRoleMember 'LT', 'DbCreator'
```

- Loại bỏ một người dùng từ một Role Server:

```
Sp_DropSrvRoleMember 'username','Role Server'
```

Ví dụ:

```
Exec Sp_DropSrvRoleMember 'LT', 'SysAdmin'
```

- Xem thông tin về nhóm hệ thống:

```
Sp_helpSrvRole [<Tên Role Server>]
```

3) *Nhóm hệ thống trên Database:*

Thường dùng các nhóm sau:

- **Db_Owner**: thành viên nhóm này có toàn quyền trên CSDL
- **Db_AccessAdmin**: thành viên nhóm này có thể thêm, xoá quyền truy cập của người dùng, thêm, xoá User ……
- **Db_DataReader**: thành viên nhóm này có thể Select trên bảng, bảng ảo
- **Db_DataWriter**: thành viên nhóm này có thể Insert, Update, Delete các bảng, bảng ảo.

4) *Nhóm người dùng định nghĩa trên Database:*

Các lệnh trên nhóm sử dụng chung cho nhóm hệ thống trên CSDL và nhóm người dùng định nghĩa.

5) *Tạo Role:*

Cú pháp :

- Sử dụng trên SQL Server 2000

```
Exec Sp_AddRole <'Tên Role'>[,<'username sở hữu Role'>]
```

- Sử dụng trên SQL Server 2005

```
Create Role <Tên Role> [Authorization <username|Role sở Hữu>]
```

Lưu ý:

- User sở hữu nhóm phải là user có trong CSDL mà ta tạo Role.
- Nếu không khai báo user sở hữu Role thì user tạo Role mặc định sẽ sở hữu Role.

Ví dụ:

```
Exec Sp_AddLogin 'Hs1', 'aa', 'Ql_SinhVien_xx'
Use Ql_SinhVien_xx
Exec Sp_GrantDBAccess 'hs1', 'Hs1_Ql_SinhVien_xx'
Exec Sp_AddRole 'NhanVien', 'Hs1_Ql_SinhVien_xx'
```

6) *Xoá Role:*

Cú pháp :

- Sử dụng trên SQL Server 2000

```
Exec Sp_DropRole   [@Role_Name = <'Tên Role'>
```

- Sử dụng trên SQL Server 2005

```
Drop Role <Tên Role>
```

Chú ý:

- Khi xoá một Role thì Role phải rỗng. Mặt khác, tạo role theo cú pháp nào thì nên xóa role theo cú pháp đó.

7) Đổi tên Role (Chỉ áp dụng cho SQL Server 2005):

Cú pháp :

```
Alter Role <Tên Role cũ> With Name = <Tên Role mới>
```

8) Thêm một thành viên vào Role:

Cú pháp :

```
Exec Sp_AddRoleMember <'Tên Role>, <'member'>
```

Chú ý:

- Member có thể là SQL user, Windows User, Database Role, Database User, ...

Ví dụ:

```
Exec Sp_AddRoleMember 'NhanVien', 'Hs1_Ql_SinhVien_xx'
Exec Sp_AddRoleMember 'NhanVien', 'Hs2_Ql_SinhVien_xx'
```

9) Xoá một thành viên của một Role:

Cú pháp :

```
Sp_DropRoleMember  <'Tên Role'>,<'member'>
```

10) Xem thông tin về Role trên Database:

Cú pháp :

```
Sp_helpRole [<Tên Role>]
```

VI. QUYỀN NGƯỜI DÙNG:

Quyền người dùng được định nghĩa, mức độ và người dùng có thể thực hiện thao tác trên CSDL. Ta có thể phân quyền người dùng thành các loại như sau:

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

1) Quyền truy cập vào CSDL:

Cú pháp :

- Sử dụng trên SQL Server 2000:

```
Use <Tên CSDL muốn gán quyền truy cập>
Sp_GrantDBAccess <'LoginName>, <'username'>
```

- Sử dụng trên SQL Server 2005:

```
Create User <username> [For Login <LoginName>]
```

Chú ý:

- Nếu bỏ qua For Login thì tên username xem như trùng với LoginName

Ví dụ:

```
Use QL_SinhVien_xx
Exec Sp_GrantdbAccess 'HuuPhuoc',
'HuuPhuoc_QL_SinhVien_xx' >.  Hay:
Create user 'huuPhuoc_QL_Sinhvien_xx' for login HuuPhuoc
```

2) *Xoá quyền truy cập vào CSDL:*

Cú pháp :

- Sử dụng trên SQL Server 2000:

```
EXEC Sp_RevokeDbAccess <'username'>
```

- Sử dụng trên SQL Server 2005:

```
Drop user <username>
```

Ví dụ:

- Sử dụng trên SQL Server 2000:

```
Exec Sp_RevokeDbAccess 'HuuPhuoc_QL_SinhVien_xx'
```

- Sử dụng trên SQL Server 2005:

```
Drop user HuuPhuoc_QL_SinhVien_xx
```

3) *Đổi tên người dùng (chỉ áp dụng cho SQL Server 2005):*

Cú pháp :

```
Alter User <username Cũ> With <username mới>
```

4) *Xem thông tin về người dùng trong CSDL:*

Cú pháp :

```
Sp_HelpUser [<username>]
```

5) *Cấp phát quyền thực thi trên CSDL:*

Sau khi cấp quyền cho người dùng truy cập vào một CSDL, tiếp theo ta phải gán quyền trên các đối tượng đó. Danh sách các quyền dùng để xử lý dữ liệu trên đối tượng:

- **Select**: Có thể xem dữ liệu trên bảng, bảng ảo, Cột.
- **Insert**: Thêm dữ liệu vào bảng, bảng ảo
- **Update**: Sửa dữ liệu có sẵn trên bảng, bảng ảo, Cột.
- **Delete**: Xoá dữ liệu trong bảng, bảng ảo
- **Execute**: thi hành một Store Procedure
- **References**: tham khảo khoá ngoại đến một bảng.

Chú ý:

- Nếu người dùng không có quyền thêm hay xoá trong bảng nhưng người dùng này lại có quyền Execute đối với một Store Procedure mà thực chất Store Procedure này chứa các lệnh thêm hay xoá dữ liệu trong bảng => khi thi hành Store Procedure thì dữ liệu vẫn được thêm, xoá một cách gián tiếp thông qua Store Procedure.

6) *Cấp quyền:*

Cú pháp:

```
GRANT { ALL | <Danh sách quyền>} ON
{<Table>|<View>| <Table>(<Các cột>)} TO
{<UserName_In_DB> | ROLE} [<With Grant Option>]
```

Chú ý:

- Khi có từ khoá With Grant Option, thì user này có thể gán quyền cho các user khác.

Ví dụ:

```
Grant Select, Insert On Dm_SinhVien To
HuuPhuoc_QL_SinhVien_xx
```

7) *Loại bỏ quyền:*

Cú pháp:

```
REVOKE {ALL |<D/sách quyền>}
ON { VIEW | STORE PROC | TABLE[(<D/sách cột>)]} TO
<UserName_In_DB>
```

Ví dụ:

```
Revoke Select  On SinhVien To HuuPhuoc_SinhVien_xx
Grant Select, Update On SinhVien(Masv,Hosv,Ten) To
HuuPhuoc_SinhVien_xx
```

8) *Từ chối quyền truy cập:*

Cú pháp:

```
DENY { ALL | <D/sách quyền>} ON {VIEW | STORE PROC
| TABLE(<D/sách cột>) TO <Name_In_DB>
```

Ví dụ:

Trong trường hợp một user thuộc về nhiều nhóm, ví dụ như user LT_ 01 thuộc 2 nhóm:

- Nhóm 1: có quyền Select, Update trên bảng Sinhvien
- Nhóm 2: có quyền Insert, Delete trên bảng Sinhvien

Để LT_01 không có quyền Delete trên bảng Sinhvien, ta dùng lệnh Deny như sau:

```
Deny Delete On SinhVien To LT_01
```

VII. QUYỀN TẠO ĐỐI TƯỢNG TRONG CƠ SỞ DỮ LIỆU:

1) *Quyền tạo CSDL:*

Khi gán quyền tạo CSDL thì ta phải đứng tại CSDL hiện hành là Master

```
Use Master
Exec Sp_AddLogin 'HuuPhuoc', 'aa'
Exec Sp_GrantDBAccess 'HuuPhuoc', 'Gv'
Grant Create Database To GV (dùng đăng nhập trong SQL)
```

2) *Gán quyền tạo bảng, tạo bảng ảo, tạo Store Procedure:*

Cú pháp:

```
GRANT { ALL | Statement } TO <username>
```

Ví dụ:

```
Use QL
Grant Create Table To HuuPhuoc_QL_SinhVien_xx
```

3) *Loại bỏ quyền:*

```
REVOKE <Danh sách quyền> TO <username>
```

```
Use QL
Revoke Create Table To HuuPhuoc_QL_SinhVien_xx
```

4) *Từ chối quyền truy cập*

```
DENY <Danh sách quyền> TO <User_Name>
```

```
Use QL
Deny Create Table To HuuPhuoc_QL_SinhVien_xx
```