Lập trình trên SQL Server

LẠI HIỀN PHƯƠNG

BỘ MÔN HTTT – KHOA CNTT

EMAIL: LHPHUONG@TLU.EDU.VN

Thủ tục lưu trữ (Stored Procedures)

Khái niệm

- Thủ tục lưu trữ (Stored Procedure) là một đối tượng trong CSDL bao gồm một tập nhiều câu lệnh SQL được nhóm lại với nhau thành một nhóm.

Đặc điểm

- Có thể nhận tham số truyền vào
- Có thể gọi thủ tục khác
- Trả về các giá trị thông qua các tham số
- Chuyển giá trị tham số cho các thủ tục được gọi
- Trả về giá trị trạng thái thủ tục là thành công hay không thành công

Ưu điểm

- **Lập trình theo module**: thủ tục được xây dựng một lần trong CSDL, có thể được gọi nhiều lần bởi một hay nhiều ứng dụng.
- **Thực hiện nhanh hơn**: thực hiện một thủ tục lưu trữ nhanh hơn thực hiện một lượng lớn các câu lệnh T-SQL vì khi máy chủ nhận được mỗi câu lệnh đều phải kiểm tra tính hợp lệ quyền của tài khoản từ máy khách.

Ưu điểm (tiếp)

- **Làm giảm lưu lượng trên mạng**: do chỉ cần gửi một câu lệnh gọi thủ tục thay vì phải gửi một tập các dòng lệnh từ ứng dụng đến máy chủ.
- **An ninh bảo mật hơn**: thay vì cấp phát quyền trực tiếp cho người sử dụng trên các câu lệnh SQL và trên các đối tượng CSDL, ta có thể cấp quyền cho người sử dụng thông qua thủ tục lưu trữ. Việc gán quyền như trên giúp cho vấn đề an ninh bảo mật trong CSDL tốt hơn.

Phân loại thủ tục lưu trữ

- **System stored procedure:**
  - Thủ tục được lưu trữ trong CSDL Master
  - Bắt đầu bằng chữ **sp_**
  - Thường được sử dụng trong quản trị CSDL và an ninh bảo mật.
  - Ví dụ: Muốn biết tất cả các tiến trình đang được thực hiện bởi user ‘sa’

```
sp_who @loginame = ‘sa’
```

Phân loại thủ tục lưu trữ (tiếp)

- **Extended stored procedure:**
  - Thủ tục sử dụng chương trình ngoại vi đã được biên dịch thành DLL
  - Bắt đầu bằng chữ **xp_**
  - Ví dụ:
    - Xp_sendmail dùng gửi mail
    - Xp_cmdshell dùng thực hiện lệnh của DOS
      - xp_cmdshell ‘dir c:\’

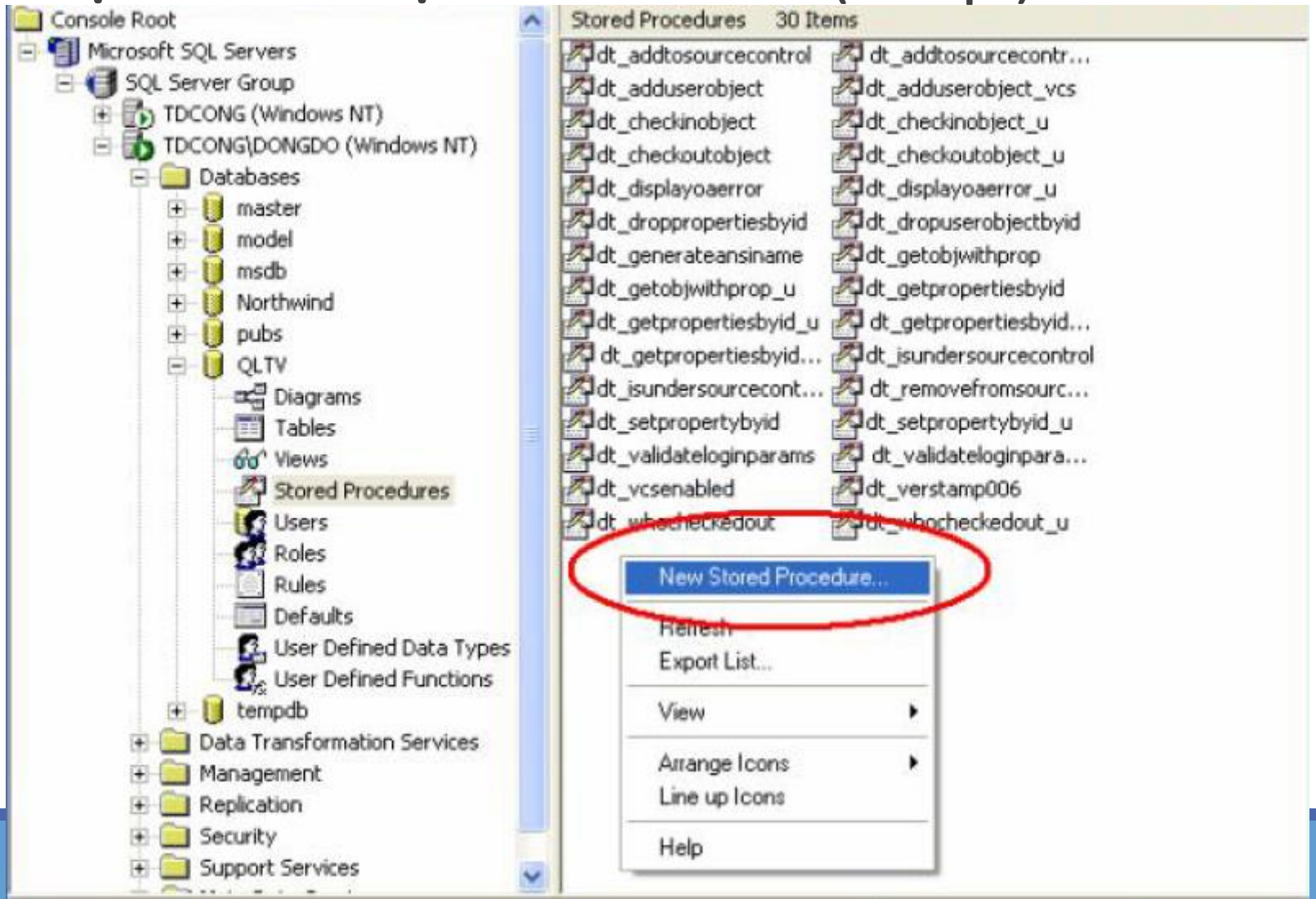
Phân loại thủ tục lưu trữ (tiếp)

- **Local stored procedure:**
  - Nằm trong CSDL do người dùng tạo ra, thực hiện một công việc nào đó.
  - Có thể được tạo ra trong CSDL master
- **Temporary stored procedure**:
  - Tương tự như local store procedure nhưng được tạo ra trên CSDL TempDB
  - Thủ tục tự hủy khi kết nối tạo ra nó ngắt hoặc SQL Server ngưng hoạt động
- **Remote stored procedure**:
  - Thủ tục sử dụng thủ tục của một server khác

Tạo thủ tục lưu trữ

- **Bằng SQL Server Management Studio:**
  - Chọn CSDL cần tạo thủ tục
  - Chọn Stored Procedures, kích chuột phải chọn New Stored Procedure
  - Đặt tên thủ tục, xác định role người khai thác và soạn kịch bản câu lệnh

Tạo thủ tục lưu trữ (tiếp)



Tạo thủ tục lưu trữ bằng T-SQL

- **Cú pháp**

```
CREATE PROCEDURE Tên_thủ_tục [(danh_sách_tham_số)]
[WITH các_tùy_chọn]
AS
BEGIN
        Các_câu_lệnh_của_thủ_tục
END
```

- **Chú ý**:
  - Có thể viết tắt là CREATE PROC
  - Cặp từ khóa BEGIN … END không bắt buộc

Tạo thủ tục lưu trữ bằng T-SQL (tiếp)

- **Tên thủ tục:** tuân theo quy tắc định danh và không vượt quá 128 ký tự
- **Danh sách tham số:** các tham số được khai báo ngay sau tên thủ tục, cách nhau bởi dấu phảy. Khai báo mỗi tham số:

  @tên_tham_số kiểu_dữ_liệu

  VD: @maMH nvarchar(10)
- **Tùy chọn:** các tùy chọn cách nhau bởi dấu phảy, có các loại:
  - RECOMPILE: thông thường, thủ tục sẽ được dịch sẵn ở lần gọi đầu tiên. Nếu có tùy chọn RECOMPILE, thủ tục sẽ được dịch lại mỗi khi gọi
  - ENCRIPTION: yêu cầu mã hóa thủ tục. Nếu thủ tục đã được mã hóa, ta không thể xem được nội dung của thủ tục

Tạo thủ tục lưu trữ bằng T-SQL (tiếp)

- **Ví dụ:** Từ CSDL QLSV với các bảng MonHoc(**MaMon**,TenMon,MoTa), SinhVien(**MaSV**, HoTen, GioiTinh, DiaChi, Email), KETQUA(**MaSV, MaMon**, Diem), viết thủ tục hiển thị MaSV, HoTen, TenMon, Diem của tất cả sinh viên.

```
create procedure sp_SV_KetQua
as
Begin
select SinhVien.MaSV, HoTen, MonHoc.TenMon,Diem from SinhVien
inner join KETQUA on SinhVien.MaSV = KETQUA.MaSV
inner join MonHoc on KETQUA.MaMon = MonHoc.MaMon
End
```

Xem một số thông tin về thủ tục

- Xem nội dung thủ tục

  sp_helptext tên_thủ_tục
  - Nội dung thủ tục không được hiển thị trong trường hợp thủ tục được tạo với tùy chọn ENCRYPTION
- Xem thông tin về người tạo, ngày giờ tạo

  sp_help tên_thủ_tục
- Xem các đối tượng mà các lệnh trong thủ tục tham chiếu đến:

  sp_depends tên_thủ_tục
- Liệt kê tất cả các thủ tục trong CSDL:

  sp_stored_procedures

Thực thi (gọi) thủ tục lưu trữ

- Yêu cầu HQT CSDL thực thi thủ tục bằng lời gọi có dạng:

  tên_thủ_tục [danh_sách_các_đối_số]

  - Số lượng và thứ tự các đối số phải tương ứng với số lượng và thứ tự của các tham số khi định nghĩa thủ tục
  - Thứ tự của các đối số có thể không cần tuân theo thứ tự của tham số như khi định nghĩa thủ tục nếu đối số được viết dưới dạng:

    @tên_tham_số = giá_trị

- Gọi thủ tục bên trong một thủ tục khác, bên trong một trigger hay kết hợp với các câu lệnh SQL khác

  EXECUTE tên_thủ_tục [danh_sách_các_đối_số]

- Nếu gọi thủ tục trong CSDL khác, tên_thủ_tục phải viết đầy đủ:

  tên_CSDL.tên_người_tạo.tên_thủ_tục

Sửa/Xóa thủ tục lưu trữ

- Khi một thủ tục được tạo ra, ta có thể tiến hành định nghĩa lại thủ tục bằng câu lệnh ALTER PROCEDURE, cú pháp như sau:

```
ALTER PROCEDURE Tên_thủ_tục [(danh_sách_tham_số)]
 [WITH các_tùy_chọn]
 AS
 BEGIN
        Các_câu_lệnh_của_thủ_tục
 END
```

- Xóa một thủ tục đã có bằng DROP PROCEDURE:

```
DROP PROCEDURE Tên_thủ_tục
```

Sử dụng biến trong thủ tục

- Bên trong thủ tục có thể sử dụng các biến để lưu trữ các giá trị tính toán được hoặc truy xuất được từ CSDL
- Khai báo biến bằng từ khóa DECLARE như thông thường

  DECLARE @Tên_biến kiểu_dữ_liệu
- Biến được khai báo bên trong thủ tục chỉ được sử dụng bên trong thủ tục

Sử dụng biến trong thủ tục (tiếp)

- **Ví dụ**: Viết thủ tục không có tham số hiển thị MaSV, HoTen, Diem của những sinh viên có điểm cao nhất môn Hệ Quản trị CSDL

```
create procedure sp_SV_KetQuaTotNhat
as
Begin
declare @maMon int
select @maMon = MonHoc.MaMon from MonHoc
where MonHoc.TenMon = N'Hệ Quản trị CSDL'
declare @maxDiem float
select @maxDiem = max(Diem) from KETQUA
where KETQUA.MaMon = @maMon

select SinhVien.MaSV, HoTen, Diem from SinhVien,KETQUA
where SinhVien.MaSV = KETQUA.MaSV
and KETQUA.Diem = @maxDiem
and KETQUA.MaMon = @maMon
End
```

Thủ tục có tham số vào

- Tham số vào dùng để truyền giá trị vào trong thủ tục
- **Cú pháp**

```
CREATE PROCEDURE Tên_thủ_tục
@tên_tham_số_1 kiểu_dữ_liệu
[, @tên_tham_số_2 kiểu_dữ_liệu, … ]
[WITH các_tùy_chọn]
AS
BEGIN
        Các_câu_lệnh_của_thủ_tục
END
```

- **Gọi thủ tục**

```
[EXECUTE|EXEC] @tên_thủ_tục giá_trị_tham_số_1[,giá_trị_tham_số_2,…]
[EXECUTE|EXEC] @tên_thủ_tục tham_số_1 = giá_trị_1
                                [, tham_số_2 = giá _trị_2,…]
```

Thủ tục có tham số vào (tiếp)

- **Ví dụ**: Viết thủ tục hiển thị MaSV, HoTen, Diem của những sinh viên có điểm cao nhất một môn học với tên môn là tham số truyền vào

```
create procedure sp_SV_KetQuaTotNhat @tenMon nvarchar(30)
as
Begin
declare @maMon int
select @maMon = MonHoc.MaMon from MonHoc
where MonHoc.TenMon = @tenMon
declare @maxDiem float
select @maxDiem = max(Diem) from KETQUA
where KETQUA.MaMon = @maMon

select SinhVien.MaSV, HoTen, Diem from SinhVien,KETQUA
where SinhVien.MaSV = KETQUA.MaSV
and KETQUA.Diem = @maxDiem
and KETQUA.MaMon = @maMon
End
```

Thủ tục có tham số vào (tiếp)

- Cách truyền tham số
  - Gán giá trị theo thứ tự

```
sp_SV_KetQuaTotNhat N'Hệ Quản trị CSDL'
```

    - Trong trường hợp có nhiều tham số đầu vào, số lượng và thứ tự giá trị các tham số phải giống như khi định nghĩa

  - Gán giá trị theo tên biến

```
sp_SV_KetQuaTotNhat @tenMon = N'Hệ Quản trị CSDL'
```

    - Thứ tự các tham số truyền vào không cần giống như khi định nghĩa

Thủ tục sử dụng tham số lấy giá trị ra (tham trị)

- Sử dụng từ khóa OUTPUT để chỉ tham số lấy giá trị ra
- **Cú pháp**

```
CREATE PROCEDURE Tên_thủ_tục
@tên_tham_số_vào_1 kiểu_dữ_liệu
[, @tên_tham_số_vào_2 kiểu_dữ_liệu, … ]
@tên_tham_số_ra_1 kiểu_dữ_liệu OUTPUT
[, @tên_tham_số_vào_2 kiểu_dữ_liệu OUTPUT, … ]

[WITH các_tùy_chọn]
AS
BEGIN
        Các_câu_lệnh_của_thủ_tục
END
```

Thủ tục sử dụng tham số lấy giá trị ra (tham trị) (tiếp)

- **Ví dụ**: viết thủ tục trả về điểm cao nhất và tên sinh viên đạt điểm cao nhất với môn thi được truyền vào qua tham số

```
create procedure sp_SV_KetQuaTotNhat
@tenMon nvarchar(30),
@maxDiem float OUTPUT,
@tenSV nvarchar(30) OUTPUT
as
Begin
declare @maMon int
select @maMon = MonHoc.MaMon from MonHoc where MonHoc.TenMon = @tenMon
select @maxDiem = max(Diem) from KETQUA where KETQUA.MaMon = @maMon

select @tenSV = SinhVien.HoTen from SinhVien,KETQUA
where SinhVien.MaSV = KETQUA.MaSV
and KETQUA.Diem = @maxDiem and KETQUA.MaMon = @maMon
End
```

Thủ tục sử dụng tham số lấy giá trị ra (tham trị)

- Cách gọi thủ tục:
  - Phải khai báo biến để lưu các giá trị trả về

```
declare @tenSV nvarchar(30), @diemCaoNhat float
execute sp_SV_KetQuaTotNhat N'Hệ Quản trị CSDL',
        @diemCaoNhat OUTPUT,
        @tenSV OUTPUT
print @diemCaoNhat
print @tenSV
```

Tham số với giá trị mặc định

- Các tham số được khai báo trong thủ tục có thể nhận các giá trị mặc định
- Giá trị mặc định sẽ được gán cho tham số trong trường hợp không truyền đối số cho tham số khi có lời gọi đến thủ tục
- Các tham số với giá trị mặc định được khai báo như sau khi tạo thủ tục

@tên_tham_số kiểu_dữ_liệu = giá_trị_mặc_định

Tham số với giá trị mặc định (tiếp)

- Ví dụ: Hiển thị danh sách tất cả các sinh viên có địa chỉ tại một tỉnh nào đó. Tên tỉnh được truyền vào qua tham số, mặc định là tỉnh Hà Nội

```
create procedure dsSV_theoTinh @tenTinh nvarchar(30) = N'%Hà Nội%'
as
begin
    select * from SinhVien
    where DiaChi like @tenTinh
end
```

Tham số với giá trị mặc định (tiếp)

- Gọi thủ tục
  - Sử dụng giá trị mặc định (tỉnh Hà Nội)

```
execute dsSV_theoTinh
```

  - Sử dụng giá trị tham số truyền vào

```
execute dsSV_theoTinh N'%Hải Phòng%'
```

Thủ tục trả về biến kiểu con trỏ

- Thủ tục trả về biến con trỏ quản lý một bảng dữ liệu được truy vấn bằng câu lệnh select
- **Cú pháp**

```
CREATE PROCEDURE Tên_thủ_tục
@tên_tham_số_vào_1 kiểu_dữ_liệu
[, @tên_tham_số_vào_2 kiểu_dữ_liệu, … ]
@tên_con_trỏ1 CURSOR VARYING OUTPUT
[, @tên_con_trỏ2 CURSOR VARYING OUTPUT, … ]
[WITH các_tùy_chọn]
AS
BEGIN
        set @tên_con_tror1 = CURSOR for Câu_lệnh_SQL
        Open @tên_con_trỏ1
        ….
END
```

Thủ tục trả về biến kiểu con trỏ (tiếp)

- Ví dụ: viết thủ tục trả về biến kiểu con trỏ chứa danh sách các sinh viên có giới tính được truyền vào qua tham số

```
create procedure thuTucCursor
@GioiTinh nvarchar(3),
@dsSV CURSOR VARYING OUTPUT
AS
Begin
SET @dsSV = CURSOR
FOR
Select * from SinhVien where GioiTinh = @GioiTinh
Open @dsSV
end
```

Thủ tục trả về biến kiểu con trỏ (tiếp)

- Gọi thủ tục: cần khai báo biến kiểu con trỏ. Sau đấy sử dụng như bình thường

```
declare @myCURSOR CURSOR
exec thuTucCursor N'Nữ',@dsSV = @myCURSOR OUTPUT
FETCH NEXT from @myCURSOR
while (@@FETCH_STATUS=0)
    FETCH NEXT from @myCURSOR
close @myCURSOR
deallocate @myCURSOR
```

Một số bài tập

- **Bài tập 1**: Viết thủ tục Sp_Update_SV có tham số dùng để cập nhật dữ liệu mới cho một sinh viên khi biết MaSV trong bảng SinhVien
- **Bài tập 2**: Viết thủ tục dùng để lấy về điểm trung bình một môn học của cả lớp, của các SV nữ, của các sinh viên Nam với Tên môn học là tham số truyền vào.

- Chú ý: các bài tập này thực hiện trên CSDL QLSV với
  - SinhVien(**MaSV**, HoTen, GioiTinh, DiaChi,Email)
  - MonHoc(**MaMon**, TenMon, MoTa)
  - KETQUA(**MaSV, MaMon**, Diem)

Hàm do người dùng định nghĩa (User-defined function)

Khái niệm hàm

- Hàm là đối tượng CSDL tương tự như thủ tục
- Hàm trả về một giá trị thông qua tên hàm
- Có thể sử dụng hàm như là một thành phần của một biểu thức
- Có 2 loại hàm:
  - Hàm do hệ quản trị CSDL cung cấp sẵn (đã học)
  - Hàm do người dùng định nghĩa nhằm phục vụ cho mục đích riêng của mình

Các loại hàm

- Scalar: trả về một giá trị
- Inline Table-valued: Sử dụng một câu lệnh select để trả về một tập row
- Multi-statement Table-valued: sử dụng nhiều câu lệnh để trả về một tập row

Định nghĩa hàm vô hướng

- **Cú pháp**

```
CREATE FUNCTION [Tên_người_tạo.] Tên_hàm
([danh_sách_tham_số])
RETURNS kiểu_dữ_liệu_trả_về_của_hàm
[WITH các_tùy_chọn]
 AS
 BEGIN
        Các_câu_lệnh_của_hàm
 END
```

Định nghĩa hàm vô hướng (tiếp)

- Danh sách tham số là danh sách các tham số đầu vào của hàm, mỗi tham số được khai báo như sau:

@tên_tham_số kiểu_dữ_liệu [= giá_trị_mặc_định]

- Kiểu dữ liệu trả về của hàm là kiểu dữ liệu vô hướng
- Các tùy chọn tương tự như với thủ tục

Ví dụ hàm vô hướng

- Viết hàm tính số lượng Sinh viên thi môn 'Hệ Quản trị CSDL'

```
create function slSV_mon(@tenMon nvarchar(30))
returns int
as
begin
    declare @SL int;
    select @SL = count(MaSV) from KETQUA, MonHoc
    where KETQUA.MaMon = MonHoc.MaMon
    and MonHoc.TenMon = @tenMon
    return @SL
end
```

Cách sử dụng hàm

- Khi thi hành hàm, chú ý cần dùng tên đầy đủ.

```
print dbo.slSV_mon(N'Hệ Quản trị CSDL')
select dbo.slSV_mon(N'Hệ Quản trị CSDL')
```

- Có thể sử dụng hàm trong mệnh đề Where:
  - Ví dụ: Hiển thị tên các môn học có số lượng người thi nhiều hơn hoặc bằng môn 'Hệ Quản trị CSDL'

```
Select TenMon from MonHoc
Where dbo.slSV_mon(TenMon) >= dbo.slSV_mon(N'Hệ Quản trị CSDL')
```

Định nghĩa hàm trả về kết quả là một bảng

- **Cú pháp**

```
CREATE FUNCTION [Tên_người_tạo.] Tên_hàm
([danh_sách_tham_số])
RETURNS @bien TABLE(danh_sách_cột)
[WITH các_tùy_chọn]
 AS
 BEGIN
        Các_câu_lệnh_của_hàm
 END
```

Ví dụ hàm trả về bảng

- Viết hàm trả về danh sách sinh viên thi môn học nào đó, tên môn được truyền vào qua tham số

```
create function dsSV_mon(@tenMon nvarchar(30))
returns @bien TABLE(MaSV int, HoTen nvarchar(30))
as
begin
    insert into @bien
    select SinhVien.MaSV, SinhVien.HoTen
    from SinhVien, KETQUA, MonHoc
    where SinhVien.MaSV = KETQUA.MaSV
    and KETQUA.MaMon = MonHoc.MaMon
    and MonHoc.TenMon = @tenMon
    return
end
```

Gọi hàm trả về bảng

- Sử dụng hàm trả về kết quả là một bảng như là Table

```
select * from dsSV_mon(N'Hệ Quản trị CSDL')
```

Định nghĩa hàm trả về kết quả là một bảng

- **Cú pháp 2**

```
CREATE FUNCTION [Tên_người_tạo.] Tên_hàm
([danh_sách_tham_số])
RETURNS TABLE
 AS
 return (Câu_lệnh_SELECT)
```

Ví dụ với cú pháp 2

Ví dụ: Viết hàm trả về danh sách các sinh viên sinh sau ngày nào đó, ngày được truyền vào qua tham số

```
create function BangSV(@NgaySinh date)
returns Table
as
return (select * from SinhVien where SinhVien.NgaySinh>@NgaySinh)
```

Gọi hàm:

```
select * from BangSV('04/09/1985')
```

Bài tập

Bài 1: Viết hàm tính độ tuổi trung bình của Sinh Viên trong bảng SinhVien

Bài 2: Viết hàm trả về danh sách các môn thi của một sinh viên có điểm cao hơn điểm trung bình tất cả các môn của sinh viên đó.