CHƯƠNG 3

LẬP TRÌNH TRÊN SQL-SERVER

NỘI DUNG

3.1. Các kiểu DL trong SQL Server

3.2. Các hàm trong SQL Server

3.3 T-SQL Programming

3.4. Thủ tục và hàm người dung

3.5. Trigger

3.6. Transactions và Locks

3.7 Lập trình ứng dụng trên SQL Server

3.3 T-SQL PROGRAMMING

CẤU TRÚC ĐIỀU KHIỂN

Cấu trúc rẽ nhánh IF … ELSE

- **Cú pháp:**

IF biểu_thức_điều_kiện

Lệnh | khối_lệnh khi điều kiện đúng

[ELSE

Lệnh | khối lệnh khi điều kiện sai

]

Với khối lệnh gồm nhiều câu, cần đặt giữa cặp từ khóa BEGIN … END

BEGIN

Câu lệnh 1

…

Câu lệnh n

END

Ví dụ cấu trúc IF … ELSE

- **VD1**: Từ bảng SinhVien và bảng Diemthi, tính điểm trung bình của sinh viên có tên Nguyễn Ngọc Anh và hiển thị 'Đạt' nếu điểm trung bình lớn hơn hoặc bằng 3.5

```
Declare @DiemTB as real
Select @DiemTB = avg(diemthi.diem)
From diemthi, sinhvien
where diemthi.masv=sinhvien.masv and sinhvien.ten='Anh' and
sinhvien.hodem=N'Nguyễn Ngọc‘

if (@DiemTB>=3.5) print N'Sinh viên Anh có kết quả: Đạt'
else print N'Sinh viên Anh có kết quả: Không Đạt'
```

Ví dụ cấu trúc IF … ELSE

Các cấu trúc IF … ELSE có thể lồng nhau

- VD2: Xếp loại điểm thi môn học

```
Declare @DiemTB as real;
Select @DiemTB = avg(diemthi.diem)
From diemthi, sinhvien
where diemthi.masv=sinhvien.masv and sinhvien.ten='Anh';
print N'Điểm trung bình là: ' + cast(@DiemTB as char(4));
if (@DiemTB<3.5) print N'Xếp loại: Không Đạt'
else if (@DiemTB<5) print N'Xếp loại: D'
else if (@DiemTB<6.5) print N'Xếp loại: C'
else if (@DiemTB<8) print N'Xếp loại: B'
else print N'Xếp loại: A'
```

Cấu trúc lựa chọn CASE

- **CASE:** để đánh giá một danh sách các điều kiện và trả về 1 trong các biểu thức kết quả thỏa mãn điều kiện đánh giá
- **CASE** có 2 định dạng:
  - CASE đơn giản (Simple CASE)
  - CASE tìm kiếm (Searched CASE)

Simple CASE

Cú pháp

```
CASE biểu_thức_đầu_vào
  WHEN biểu_thức_1 THEN biểu_thức_kết_quả_1
  WHEN biểu_thức_2 THEN biểu_thức_kết_quả_2
  …
  WHEN biểu_thức_n THEN biểu_thức_kết_quả_n
  ELSE biểu_thức_kết_quả_khác
END
```

→ Một biểu thức sẽ được dung để so sánh với một tập các giá trị để xác định kết quả

Simple CASE

- **VD3**: hiện ra màn hình tên tháng hiện tại

```
Declare @thanghientai nvarchar(20);
select @thanghientai=
(
 Case month(getdate())
when 1 then N'Tháng một'
when 2 then N'Tháng hai‘
…
when 11 then N'Tháng mười một'
else  N'Tháng mười hai'
end
)
print @thanghientai
```

Cú pháp Searched CASE

Cú pháp:

```
CASE
  WHEN biểu_thức_điều_kiện_1 THEN biểu_thức_kết_quả_1
  WHEN biểu_thức_điều_kiện_2 THEN biểu_thức_kết_quả_2
  …
  WHEN biểu_thức_điều_kiện_n THEN biểu_thức_kết_quả_n
  ELSE biểu_thức_kết_quả_khác
END
```

→ Đánh giá tập các biểu thức Boolean để xác định kết quả

Ví dụ: Searched CASE

- **VD4**: Viết lại ví dụ xếp loại SV Nguyễn Ngọc Anh

```
Declare @DiemTB as real;
Select @DiemTB = avg(diemthi.diem)
From diemthi, sinhvien
where diemthi.masv=sinhvien.masv and sinhvien.ten='Anh'and
       sinhvien.hodem=N'Nguyễn Ngọc';
print N'Điểm trung bình là: ' + cast(@DiemTB as char(4));
 Declare @Xeploai nvarchar(15);
 Select @Xeploai =
    (CASE
       when @DiemTB<3.5 then N'Xếp loại: Không Đạt'
       when (@DiemTB<5) then N'Xếp loại: D'
       when (@DiemTB<6.5) then N'Xếp loại: C'
       when (@DiemTB<8) then N'Xếp loại: B'
       else  N'Xếp loại: A'
     END)
  Print @Xeploai
```

Cấu trúc lặp WHILE

- **Cú pháp:**

```
WHILE
biểu_thức_điều_kiện
    BEGIN
        Khối lệnh 1
        [BREAK]
        Khối lệnh 2
        [CONTINUE]
        Khối lệnh 3
        …
    END
```

- **BREAK:** thoát khỏi vòng lặp WHILE, tất cả các lệnh sau từ khóa BREAK và trước từ khóa END sẽ bị bỏ qua.
- **CONTINUE**: thực hiện restart lại vòng lặp hiện tại tại vị trí bắt đầu. Tất cả các câu lệnh sau từ khóa CONTINUE và trước từ khóa END sẽ bị bỏ qua.

Ví dụ cấu trúc WHILE

- **VD5**: Hiển thị các số từ 1 đến 9

```
declare @biendem int;
set @biendem=1
while (@biendem<10)
    begin
      Print N' Số hiện tại: ' + cast (@biendem as
char(2))
      set @biendem=@biendem+1
    end
```

MỘT SỐ TOÁN TỬ ĐẶC BIỆT

Một số toán tử đặc biệt

- **Một số toán tử đặc biệt dùng trong các biểu thức điều kiện:**

Toán tử	Ý nghĩa	Ví dụ
ALL	Tất cả	3.5 <= ALL (SELECT Diem from KETQUA)
ANY	Một vài (ít nhất 1)	3.5 > ANY (SELECT Diem from KETQUA)
SOME	Tương tự ANY	3.5 > SOME (SELECT Diem from KETQUA)
BETWEEN	Nằm giữa phạm vi	@Diem BETWEEN (3 and 5)
EXISTS	Tồn tại	EXISTS (SELECT Diem from KETQUA)
IN	Kiểm tra xem một giá trị có tồn tại trong một tập cho trước không	@GT in (N'Nam', N'Nữ')

Ví dụ các toán tử đặc biệt - Exists

- **VD6: H**iển thị MaSV, HoTen, KetQua của tất cả các sinh viên trong bảng SinhVien với KetQua = 'Còn nợ môn' với sinh viên có môn thi chưa đạt và 'Đã qua hết' với sinh viên đã qua hết các môn

```
select masv, hodem+ten as Hoten,  KetQua=
(
   Case
      when exists (select * from diemthi
                   where diemthi.masv=sinhvien.masv
and diemthi.diem<3.5)
      then N'Còn nợ môn' else N'Đã qua hết'
 END
)
from sinhvien
```

Ví dụ các toán tử đặc biệt

- **Ví dụ:** cách khác dùng ALL

```
select Sinhvien.masv, hodem+ten as Hoten,  KetQua=
(
   Case
      when 3.5<=all (select diem from diemthi
                     where diemthi.masv=sinhvien.masv)
     then  N'Đã qua hết' else N'Còn nợ môn'
 END
)
from sinhvien
```

Ví dụ các toán tử đặc biệt

- **Ví dụ:** cách khác dùng ANY

```
select Sinhvien.masv, hodem+ten as Hoten, KetQua=
(
  Case
      when 3.5>any (select diem from diemthi
          where diemthi.masv=sinhvien.masv)

      then N'Còn nợ môn'
      else N'Đã qua hết'
 END
)
from sinhvien
```

Ví dụ các toán tử đặc biệt

- **Ví dụ:** cách khác dùng IN

```
select Sinhvien.masv, hodem+ten as Hoten, KetQua=
(
  Case
      when masv in (select masv from diemthi
                    where diemthi.masv=sinhvien.masv
and diemthi.diem<3.5)
      then N'Còn nợ môn'
      else N'Đã qua hết'
 END
)
from sinhvien
```

CẤU TRÚC TRY..CATCH

Khối lệnh TRY … CATCH

- Hoạt động: Gồm khối TRY và khối CATCH.
  - Khi một điều kiện lỗi được dò thấy ở khối TRY, điều khiển được chuyển sang khối CATCH để xử lý. Sau khi khối CATCH điều khiển ngoại lệ, điều khiển được chuyển cho câu lệnh Transact-SQL ngay sau lệnh END CATCH
  - Nếu không lỗi trong khối TRY, điều khiển được chuyển cho câu lệnh sau END CATCH.
- Cú pháp:

```
BEGIN TRY
        { các câu lệnh }
END TRY
BEGIN CATCH
        { các câu lệnh}
END CATCH
```

Khối lệnh TRY … CATCH

- Lưu ý :
  - TRY và CATCH phải cùng lô lệnh (nằm giữa hai từ khóa GO GO)
  - Sau khối TRY phải là khối CATCH
  - Có thể lồng nhiều cấp

Khối lệnh TRY … CATCH

- VD7:

```
Declare @i float
begin try
    set @i=2/0
end try
begin catch
    select ERROR_NUMBER() as errornumber,
           ERROR_MESSAGE() as errormessage
end catch
```

- Trả về

	eromumber	eromessage
1	8134	Divide by zero error encountered.

Một số hàm ERROR thường dùng

ERROR_NUMBER() : Trả về mã số của lỗi

ERROR_MESSAGE() Trả về chuỗi lỗi

ERROR_SEVERITY() returns the error severity.

ERROR_STATE() returns the error state number.

ERROR_LINE() : Trả về dòng gây ra lỗi

ERROR_PROCEDURE() Trả về tên thủ tục/ trigger gây ra lỗi

Thủ tục RAISERROR

- Ý nghĩa : Trả về thông báo lỗi cho ứng dụng
- **Cú pháp**:

  Raiserror(tbao_lỗi, mức_độ, trạng_thái [, các_tham_số])
  - Tbao_lỗi:
    - Chuỗi thông báo lỗi bất kỳ
    - Mã thông báo lỗi do người dùng định nghĩa trước bằng sp_addmessage và được lưu trong sys.messages. Giá trị phải > 50000
  - Mức_độ: từ 0 đến 25 thể hiện mức độ nghiêm trọng của lỗi
  - Trạng_thái: số từ 1-127 để xác định vị trí lỗi khi sử dụng cùng một thông báo lỗi tại nhiều điểm khác nhau
  - Các_tham_số: hỗ trợ các thông báo lỗi cần tham số

Thủ tục RAISERROR (Ví dụ)

VD8:

```
declare @phantram float
set @phantram = 101

declare @ErrMsg nvarchar(50)
if (@phantram not between 0 and 100)
  begin
    set @ErrMsg = N'Tỷ lệ phần trăm phải nằm trong
khoảng [0,100]'
    Raiserror (@ErrMsg,16,1)
  end
```

3.4. THỦ TỤC LƯU TRỮ, HÀM

THỦ TỤC LƯU TRỮ (Stored Procedure)

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.

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)

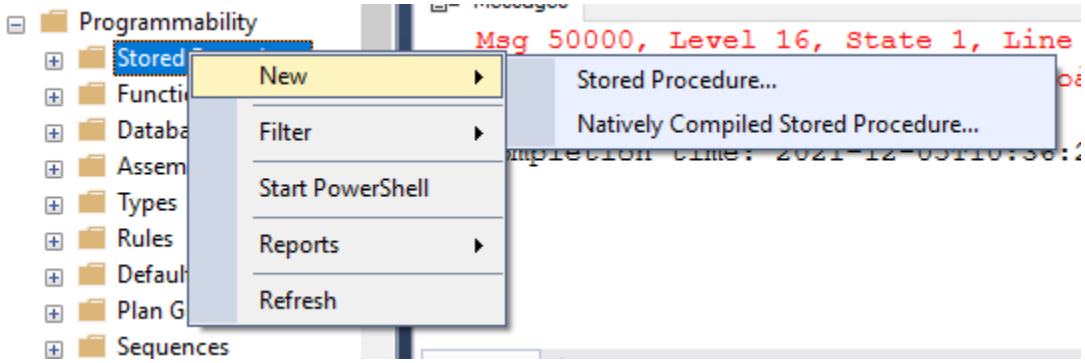
- **Local stored procedure:**
  - Nằm trong CSDL do người dùng tạo ra, thực hiện một công việc nào đó.
  - Có thể được tạo ra trong CSDL master
- **Remote stored procedure**:
  - Thủ tục sử dụng thủ tục của một server khác

Phân loại thủ tục lưu trữ (tiếp)

- 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

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)

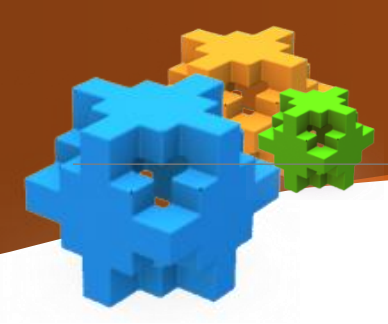
```
-- Template generated from Template Explorer using:
-- Create Procedure (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the procedure.
-- ============================================
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- ============================================
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
    -- Add the parameters for the stored procedure here
    <@Param1, sysname, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
    <@Param2, sysname, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
```

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
```

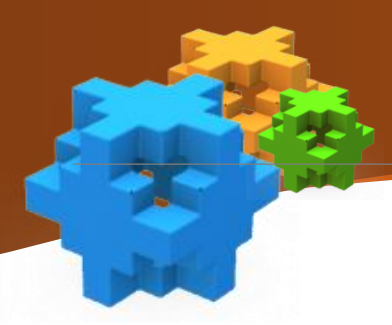
- **Chú ý**:
  - Có thể viết tắt là CREATE PROC
  - Cặp từ khóa BEGIN … END không bắt buộc



Tạo thủ tục bằng T-SQL (tiếp)

- **Tên thủ tục:** tuân theo quy tắc định danh
- **Danh sách tham số:**
  - Khai báo ngay sau tên thủ tục, các tham số cách nhau bởi dấu phẩy.
  - Mỗi tham số khai báo gồm 2 phần:
    - Tên tham số: được bắt đầu bởi @tên_tham_số
    - Kiểu_dữ_liệu của tham số

    VD: @maMH nvarchar(10)



Tạo thủ tục bằng T-SQL (tiếp)

- **Tùy chọn:** các tùy chọn cách nhau bởi dấu phẩy
  - 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 được 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 bằng T-SQL (tiếp)

- **VD9 :** Hãy viết thủ tục hiển thị MaSV, HoTen, TenMon, Diem của tất cả sinh viên.

```
Create procedure sp_SV_Ketqua
as
Begin
  select sinhvien.masv, sinhvien.hodem + sinhvien.ten
as Hoten, monhoc.tenmonhoc, diemthi.diem
  from sinhvien inner join diemthi on
      sinhvien.masv=diemthi.masv
inner join monhoc on diemthi.mamonhoc=monhoc.mamonhoc
end
```

Tạo thủ tục bằng T-SQL (tiếp)

- VD10: Giả sử ta cần thực hiện một chuỗi các thao tác như sau trên cơ sở dữ liệu:

1. Bổ sung thêm môn học *Lập trình web* có mã *Web* và số đơn vị học trình là *2* vào bảng MONHOC

2. Lên danh sách nhập điểm thi môn *Lập trình web* cho các sinh viên học lớp hệ thống thông tin k60 có mã 60HT

(tức là bổ sung thêm vào bảng DIEMTHI các bản ghi với cột MAMONHOC nhận giá trị *web,* cột MASV nhận giá trị lần lượt là mã các sinh viên học lớp có mã 60HT và các cột điểm là NULL).

```
CREATE PROC sp_LenDanhSachDiem(
    @mamonhoc NVARCHAR(10), @tenmonhoc NVARCHAR(50),
       @sodvht SMALLINT, @malop NVARCHAR(10))
AS
BEGIN
      INSERT INTO monhoc
      VALUES(@mamonhoc,@tenmonhoc,@sodvht)
      INSERT INTO diemthi(mamonhoc,masv)
      SELECT @mamonhoc,masv
      FROM sinhvien
      WHERE malop=@malop
END

sp_LenDanhSachDiem 'Web','Lập trình Web',2,'60HT'
```

Xem thông tin về thủ tục

- Xem nội dung thủ tục

  sp_helptext tên_thủ_tục

  - Chú ý: 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

- Thực thi thủ tục: **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 tham số khi định nghĩa thủ tục nếu tất cả các đối số được viết:

    @tên_tham_số = giá_trị

```
VD: sp_LenDanhSachDiem @malop='60HT',
                       @mamonhoc='web',
    @tenmonhoc='Lập trình web', @sodvht=2
```

Thực thi (gọi) thủ tục

- 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

- Có thể định nghĩa lại thủ tục 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: 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:

```
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

- **VD 11**: Viết thủ tục không có tham số hiển thị MaSV, HoTen, Diem của những sinh viên có điểm cao nhất môn Cơ sở dữ liệu

```
create procedure sp_SV_KQtotnhat
as
begin
	declare @mamon nvarchar(10)
	select @mamon=monhoc.mamonhoc from monhoc
		where  monhoc.tenmonhoc = N'cơ sở dữ liệu'
	declare @maxdiem float
	select @maxdiem=max(diem) from diemthi
		where diemthi.mamonhoc=@mamon
 Select sinhvien.masv, sinhvien.hodem+' '+ sinhvien.ten as Hoten,
diemthi.diem
from sinhvien,diemthi
where sinhvien.masv=diemthi.masv and diemthi.diem=@maxdiem and
diemthi.mamonhoc=@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)

- **VD 12:** 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_KQtotnhat2 @tenmon nvarchar(30)
as
begin
	declare @mamon nvarchar(10)
	select @mamon=monhoc.mamonhoc from monhoc
		where monhoc.tenmonhoc = N'cơ sở dữ liệu'
	declare @maxdiem NUMERIC(5, 2)
	select @maxdiem=max(diem) from diemthi
		where diemthi.mamonhoc=@mamon
Select sinhvien.masv, sinhvien.hodem+' '+ sinhvien.ten as
Hoten, diemthi.diem
from sinhvien,diemthi
where sinhvien.masv=diemthi.masv and diemthi.diem=@maxdiem and
diemthi.mamonhoc=@mamon
End
```

Thủ tục có tham số vào (tiếp)

VD 13: Xây dựng thủ tục hiển thị danh sách sinh viên theo mã lớp

```
CREATE PROCEDURE p_DSSV
AS
SELECT MaSV, Hodem + ' '+Ten as Hoten, Ngaysinh, MaLop
From sinhvien
-- ==> thực thi: p_DSSV


CREATE PROCEDURE p_DSSV @parMaLop varchar(10)='61TH%'
AS
SELECT MaSV, Hodem + ' '+Ten as Hoten, Ngaysinh
From sinhvien Where MaLop like @parMaLop
GO
-- ==> thực thi: p_DSSV '61TH2'
```

Thủ tục có tham số vào (tiếp)

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

```
sp_SV_KQtotnhat2 N'cơ sở dữ liệu'
```

  - 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_KQtotnhat2 @tenmon=N'cơ sở dữ liệu'
```

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

Thủ tục có tham 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 có tham trị ra (tiếp)

Vận hết

VD14: Xây dựng thủ tục pp_Siso để xuất giá trị sĩ số của một lớp theo tham số mã lớp. Mã lớp được truyền vào khi thủ tục được thực hiện

```
CREATE PROCEDURE pp_Siso @parMaLop Char(10),
                         @parSiso Int OUTPUT
AS
        SELECT @parSiso=count(*)
        From sinhvien Where MaLop=@parMaLop
        GO

DECLARE @siso int
exec pp_Siso '61TH2',@parSiso=@siso OUTPUT
Print 'Si so lop 61TH2 là :'+ convert(varchar(3),@siso)
Go
```

Thủ tục có tham trị ra (tiếp)

VD14:

```
CREATE PROCEDURE pp_Siso @parMaLop Char(10),
                         @parSiso Int OUTPUT
AS
       SELECT @parSiso=count(*)
       From sinhvien
       Where MaLop=@parMaLop
GO

DECLARE @siso int
exec pp_Siso '61TH2',@parSiso=@siso OUTPUT
SELECT 'Si so lop 61TH2 là :'= @siso
Go
```

Thủ tục có tham trị ra (tiếp)

- **VD 15**: 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_KQtotnhat3 @tenmon nvarchar(30),
            @maxdiem float OUTPUT,@TenSV nvarchar (30) OUTPUT
as
begin
      declare @mamon nvarchar(10)
      select @mamon=monhoc.mamonhoc from monhoc
            where monhoc.tenmonhoc = @tenmon
      select @maxdiem=max(diem) from diemthi
            where diemthi.mamonhoc=@mamon
      Select @tenSV= sinhvien.hodem+' '+ sinhvien.ten
      from sinhvien,diemthi
      where sinhvien.masv=diemthi.masv and
      diemthi.diem=@maxdiem and diemthi.mamonhoc=@mamon
End
```

Thủ tục có tham trị ra (tiếp)

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

```
declare @tensv nvarchar(30), @diemcaonhat float
execute sp_SV_KQtotnhat3 N'cơ sở dữ liệu',
      @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

@tên_tham_số kiểu_dữ_liệu = giá_trị_mặc_định

Tham số với giá trị mặc định (tiếp)

- VD16: 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 noisinh 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

- VD17: 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  bit, @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

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

```
declare @mycursor CURSOR
exec thutucCursor 0, @dssv = @mycursor OUTPUT
fetch next from @mycursor
while (@@FETCH_STATUS =0)
      fetch next from @mycursor
close @mycursor
deallocate @mycursor
```

HÀM DO NGƯỜI DÙNG ĐỊNH NGHĨA (USER-DEFINED FUNCTION)

Khái niệm

- 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

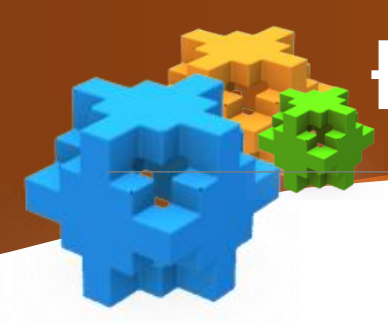
Các loại hàm

- 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

VD18: Viết hàm tính số lượng Sinh viên thi môn ‘Cơ sở dữ liệu’

```
create function SlSV_mon (@tenmon nvarchar(30))
returns int
as
begin
      declare @sl int
      select @sl=count(masv) from diemthi,monhoc
            where diemthi.mamonhoc=monhoc.mamonhoc
                  and monhoc.tenmonhoc=@tenmon
      return @sl
end
```

Cách sử dụng hàm vô hướng

- Khi thực hiện hàm, chú ý cần dùng tên đầy đủ.

```
print dbo.slSV_mon (N'Cơ sở dữ liệu')
select dbo.slSV_mon (N'Cơ sở dữ liệu')
print N'Số lượng sinh viên thi môn học là:'+ cast (dbo.slSV_mon
(N'Cơ sở dữ liệu') as char(3))
```

- 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 tenmonhoc from MonHoc
Where dbo.slSV_mon(tenmonhoc) >= dbo.slSV_mon(N'Hệ Quản trị cơ
sở dữ liệu')
```

Ví dụ hàm vô hướng

VD19: tính ngày trong tuần (thứ trong tuần):

```
CREATE FUNCTION thu(@ngay DATETIME)
RETURNS NVARCHAR(10)
AS
BEGIN
        DECLARE @st NVARCHAR(10)
        SELECT @st=CASE DATEPART(DW,@ngay)
        WHEN 1 THEN N'Chu nhật'
        WHEN 2 THEN N'Thứ hai'
        WHEN 3 THEN N'Thứ ba'
        WHEN 4 THEN N'Thứ tư'
        WHEN 5 THEN N'Thứ năm'
        WHEN 6 THEN N'Thứ sáu'
        ELSE N'Thứ bảy'
END
RETURN (@st) /* Trị trả về của hàm */
END
```

```
SELECT masv,hodem,ten,dbo.thu(ngaysinh),ngaysinh
FROM sinhvien WHERE malop='61TH1'
```

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
```

Hàm trả về kết quả là một bảng

VD20: 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 char(5), Hoten nvarchar(30))
as
begin
      insert into @bien
      select sinhvien.masv, sinhvien.hodem+'
                              '+sinhvien.ten as Hoten
      from sinhvien,diemthi,monhoc
      where sinhvien.masv=diemthi.masv and
            diemthi.mamonhoc=monhoc.mamonhoc and
            monhoc.tenmonhoc=@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'Cơ sở dữ liệu')
```

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

VD21: 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('9/06/2002')
```

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.