HỌC VIỆN KỸ THUẬT MẬT MÃ
KHOA CÔNG NGHỆ THÔNG TIN

Bài giảng: Lập trình trong T-SQL (Hàm và thủ tục)

"Tri thức là chìa khóa mở cánh cửa tương lai"

Tổng quan về T-SQL

- ***T-SQL*** là một ngôn ngữ lập trình database hướng thủ tục được sử dụng trong SQL Server
- ***T-SQL*** tổ chức theo từng khối lệnh, một khối lệnh có thể lồng trong một khối lệnh khác.
  - Một khối lệnh được đặt nằm trong cụm BEGIN …END
  - Bên trong khối lệnh có nhiều lệnh
- ***Cấu trúc khối lệnh:***

```
BEGIN
    --Khai báo biến
    --Các câu lệnh
END
```

Biến trong T-SQL

➢ ***Ý nghĩa***

Variable (biến) là một đối tượng trong CSDL dùng để lưu dữ liệu tạm thời.

Biến hệ thống (Global Variable)	Biến do người dùng định nghĩa (Local Variable)

Biến hệ thống

➢ ***Ý nghĩa***

Biến hệ thống cung cấp các thông tin hệ thống

Chú ý Tên biến bắt đầu bằng chữ @@

➢ ***Ví dụ***

`@@VERSION`	Thông tin phiên bản Microsoft SQL Server
`@@ROWCOUNT`	Số dòng bị tác động bởi câu lệnh gần nhất

Biến do người dùng tự định nghĩa

- ***Ý nghĩa***
  - ✓ Biến do người dùng tự định nghĩa để phục vụ nhu cầu riêng
  - ✓ Sử dụng các biến nhằm lưu giá trị tính toán được hoặc truy xuất được từ cơ sở dữ liệu

Chú ý Tên biến bắt đầu bằng chữ @

- ***Cú pháp***
  - ✓ Khai báo biến `DECLARE TÊN_BIẾN KIỂU_DỮ_LIỆU`
  - ✓ Gán giá trị cho biến `SET hoặc SELECT`
  - ✓ Xuất kết quả `PRINT hoặc SELECT`

Ví dụ 1: gán giá trị hai biến và in kết quả ra màn hình

```
Begin
    --Khai báo hai biến a, b
    declare @a  int, @b int;
    --Gán giá trị cho hai biến
    set @a=4;
    select @b=8;
    --In giá trị ra màn hình, sử dụng hàm cast để ép kiểu int về kiểu chuỗi
    print 'a='+ cast(@a as varchar(20));
    print 'b='+ cast(@b as varchar(20));
end
```

Ví dụ 2: gán giá trị cho biến bằng giá trị trong câu lệnh truy vấn

```
BEGIN
    USE QL_SV;
    Declare @S int;
    set @S = (select count(masv) from sinhvien)
    print N'Số Sv là:'+cast(@S as varchar(10))
END


BEGIN
    USE QL_SV;
    Declare @S int;
    select @S=count(masv) from sinhvien
    print N'Số Sv là:'+cast(@S as varchar(10))
END
```

Các cấu trúc lệnh trong T-SQL

Cấu trúc rẽ nhánh IF…ELSEIF… ELSE

- ***Cú pháp***

```
IF (điều kiện 1)
   Khối lệnh 1;
[ELSEIF (điều kiện 2)
   Khối lệnh 2;
]
…
[ELSE
   Khối lệnh n+1;
]
```

- ***Ví dụ***

```
BEGIN
    DECLARE @a int, @b int;
    SET @a=1;
    SET @b=2;
    IF (@a<@b) PRINT N'Số a nhỏ hơn số b';
    ELSE PRINT N'Số a lớn hơn số b’;

END
```

Cấu trúc điều khiển trong T-SQL

Câu lệnh CASE

➢ ***Cú pháp***

```
CASE <biểu thức lựa chọn>

   WHEN gt1 THEN <thực hiện khi giá trị 1>
   WHEN gt2  THEN <thực hiện khi giá trị 2>
   ...
   WHEN gtn THEN <thực hiện khi giá trị n>

   ELSE <thực hiện khi giá trị khác>

END
```

Cấu trúc điều khiển trong T-SQL

➢ ***Ví dụ***

```
SELECT MaSV, HotenSV,
                        (CASE Gioitinh
                        WHEN N'Nữ' THEN N'Đây là sinh viên nữ'
                        WHEN N'Nam' THEN N'Đây là sinh viên nam'
                        ELSE N'Chưa phân biệt được giới tính'
                        END) AS N'Thông báo'
FROM Sinhvien
```

➢ ***Kết quả***

Results | Messages

	MaSV	HotenSV	Thông báo
1	AT1	Cao Thu Huyền	Đây là sinh viên nữ
2	AT2	Nguyễn Thị Hải	Đây là sinh viên nữ
3	CN1	Trần Mạnh Cường	Đây là sinh viên nam
4	CN2	Lê Văn Minh	Đây là sinh viên nam
5	DT1	Nguyễn Bảo lâm	Đây là sinh viên nam
6	DT3	Vũ Tuấn Đạt	Đây là sinh viên nam

Cấu trúc điều khiển trong T-SQL

Câu lệnh WHILE

- ***Cú pháp***

```
WHILE (điều kiện )
 BEGIN
        Khối lệnh;
 END;
```

- ***Ví dụ***

```
DECLARE @A INT = 0
WHILE @A < 5
BEGIN
   PRINT N'Xin chào'
   SET @A = @A + 1
END
PRINT N'Kết thúc vòng lặp While’
```

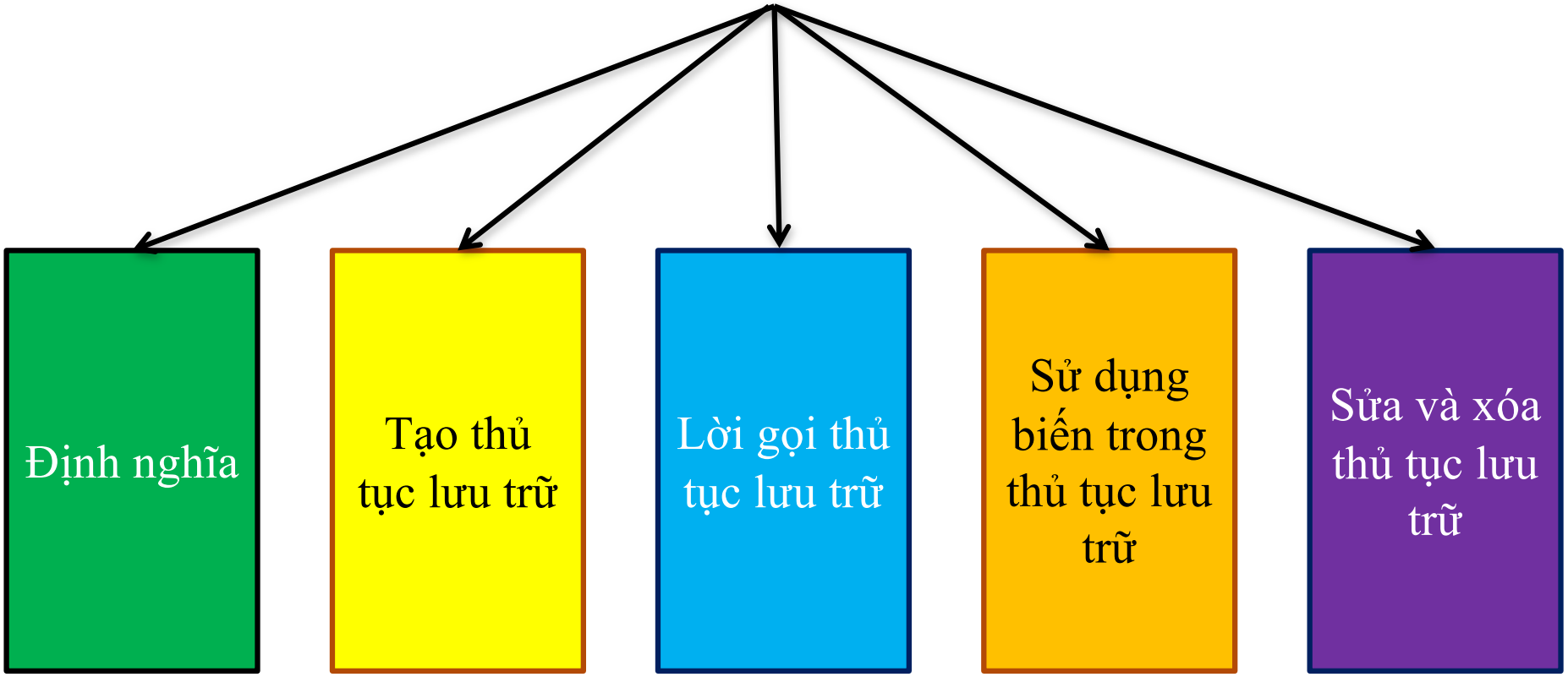
- ***Chú ý:*** có thể sử dụng BREAK để thoát khỏi vòng lặp hoặc CONTINUE để bỏ qua vòng lặp hiện tại, tiếp tục thực hiện vòng lặp mới.

Nội dung bài học

Thủ tục lưu trữ
(Stored procedure)

Hàm
(Function)

Thủ tục lưu trữ



Thế nào là một thủ tục lưu trữ ?

➢ ***Định nghĩa***

Một thủ tục là một đối tượng trong CSDL bao gồm nhiều câu lệnh T-SQL được nhóm lại với nhau thành một nhóm có những khả năng sau:

- ✓ *Thực hiện một công việc hay tác vụ nào đó, có thể tái sử dụng nhiều lần*
- ✓ *Các cấu trúc điều khiển có thể được sử dụng trong thủ tục*
- ✓ *Bên trong thủ tục có thể sử dụng các biến như trong ngôn ngữ lập trình*

Tạo thủ tục lưu trữ

- ***Cú pháp***

```
CREATE PROCEDURE <tên thủ tục>
 [(<DSách tham số>) ]
[WITH RECOMPILE|ENCRYPTION|RECOMPILE, ENCRYPTION]
AS
<Các câu lệnh của thủ tục>
```

Tạo thủ tục lưu trữ

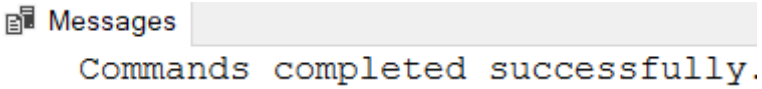
Trong đó

- ✓ ***Tên của thủ tục lưu trữ*** phải tuân theo quy tắc định danh và không vượt quá 128 ký tự
- ✓ ***Danh sách tham số:*** Khai báo của mỗi một tham số tối thiểu phải bao gồm hai phần:
  - tên tham số được bắt đầu bởi dấu @
  - kiểu dữ liệu của tham số
- ✓ ***RECOMPILE:*** Nếu tuỳ chọn WITH RECOMPILE được chỉ định, thủ tục sẽ được dịch lại mỗi khi được gọi.
- ✓ ***ENCRYPTION:*** nếu tuỳ chọn WITH ENCRYPTION, thủ tục sẽ được mã hóa
- ✓ ***Tập hợp các câu lệnh sử dụng trong nội dung thủ tục***: Các câu lệnh này có thể đặt trong cặp từ khoá BEGIN … END hoặc không

Tạo thủ tục lưu trữ

➢ *Ví dụ 1* Tạo thủ tục cho biết danh sách sinh viên theo Lớp

```
CREATE PROCEDURE Thongtinsinhvien
(@lop nvarchar(5))
 AS
SELECT MaSV, HotenSV, Gioitinh, Ngaysinh, Lop
FROM Sinhvien
WHERE lop = @lop
```



Lời gọi thủ tục lưu trữ

- Thủ tục lưu trữ được gọi theo cấu trúc:

  `Tên_thủ_tục  [danh_sách_tham_số]`

- Nếu thủ tục được gọi từ một thủ tục khác, thực hiện bên trong một trigger hay phối hợp với câu lệnh SELECT, cấu trúc như sau;

  Execute Tên_thủ_tục_lưu _trữ [danh_sách_tham_số]

- Danh sách tham số truyền vào trong lời gọi phải theo đúng thứ tự khai báo các tham số trong thủ tục lưu trữ

- Thứ tự các đối số có thể không cần tuân theo thứ tự của các tham số như khi định nghĩa

  @tên_tham_số = giá_trị

➢ *Ví dụ 1* Tạo thủ tục cho biết danh sách sinh viên theo Lớp

```
CREATE PROCEDURE Thongtinsinhvien
(@lop nvarchar(5))
 AS
SELECT MaSV, HotenSV, Gioitinh, Ngaysinh, Lop
FROM Sinhvien
WHERE lop = @lop
```

Messages

```
Commands completed successfully.
```

Thực hiện lời gọi: `Thongtinsinhvien 'CT4A'`

Kết quả

Results | Messages

	MaSV	HotenSV	Gioitinh	Ngaysinh	Lop
1	CN1	Trần Mạnh Cường	Nam	1998-12-01	CT4A
2	CN2	Lê Văn Minh	Nam	1998-09-20	CT4A

- ***Ví dụ 2*** Tạo một thủ tục để lưu trữ danh sách sinh viên học một lớp cụ thể và đến từ một thành phố cụ thể

```
CREATE PROCEDURE Chon_SV
(@Lop char(10), @Quequan nvarchar(50))
AS
SELECT * FROM Sinhvien WHERE Lop = @Lop and Quequan = @Quequan
```

Messages

```
Commands completed successfully.
```

```
Chon_SV 'AT16A', N'Hà Nội'
```

- ***Kết quả***

Results | Messages

	masv	hotensv	gioitinh	lop	quequan	ngaysinh	sdt	email
1	AT1	Cao Thu Huyền	Nữ	AT16A	Hà Nội	1998-10-10	0987878909	abc@gmail.com
2	AT3	Hoàng Minh Chiến	Nam	AT16A	Hà Nội	1998-11-10	0967878909	ktm@gmail.com

Sử dụng biến trong thủ tục lưu trữ

➢ ***Ví dụ 3*** Kiểm tra hai bạn sinh viên có cùng lớp hay không

```
create proc kiemtralop
(@maSV1 varchar(10),
 @maSV2 varchar(10))
as
declare @lop1 varchar(10),@lop2 varchar(10)
set @lop1 = (select lop from Sinhvien where MaSV = @maSV1)
set @lop2 = (select lop from Sinhvien where MaSV = @maSV2)
if @lop1 = @lop2
print N'Hai bạn sinh viên có mã '+ @masv1 + N' và '+ @masv2 +
N' học cùng lớp ' + @lop1
else
print N'Hai bạn sinh viên có  mã '+  @masv1 + N' và '+ @masv2 +
N' không học cùng lớp'

 kiemtralop 'AT1','AT3'
```

SELECT

➢ ***Kết quả***

Messages

Hai bạn sinh viên có mã AT1 và AT3 học cùng lớp AT16A

Sử dụng biến trong thủ tục lưu trữ

➢ ***Ví dụ 4*** Kiểm tra hai bạn sinh viên có cùng lớp hay không

```
create proc kiemtralop
(@maSV1 int,
 @maSV2 int)
as
declare @lop1 varchar(10),@lop2 varchar(10)
set @lop1 = (select lop from Sinhvien where MaSV = @maSV1)
set @lop2 = (select lop from Sinhvien where MaSV = @maSV2)
if @lop1 = @lop2
print  N'Hai bạn sinh viên có mã '+ str(@masv1,2) + N' và '+
str(@masv2,2) + N' học cùng lớp ' + @lop1
else
print N'Hai bạn sinh viên có  mã '+ str(@masv1,2)+N' và ' +
str(@masv2,2) + N' không học cùng lớp'

 kiemtralop 1,3
```

Sửa, xóa thủ tục lưu trữ

1. *Sửa thủ tục lưu trữ*

```
ALTER PROCEDURE <tên thủ tục>
 [(<DSách tham số>) ]
[WITH RECOMPILE|ENCRYPTION|RECOMPILE, ENCRYPTION]
AS
<Các câu lệnh của thủ tục>
```

2. Xóa thủ thục lưu trữ

```
DROP PROCEDURE <tên thủ tục>
```

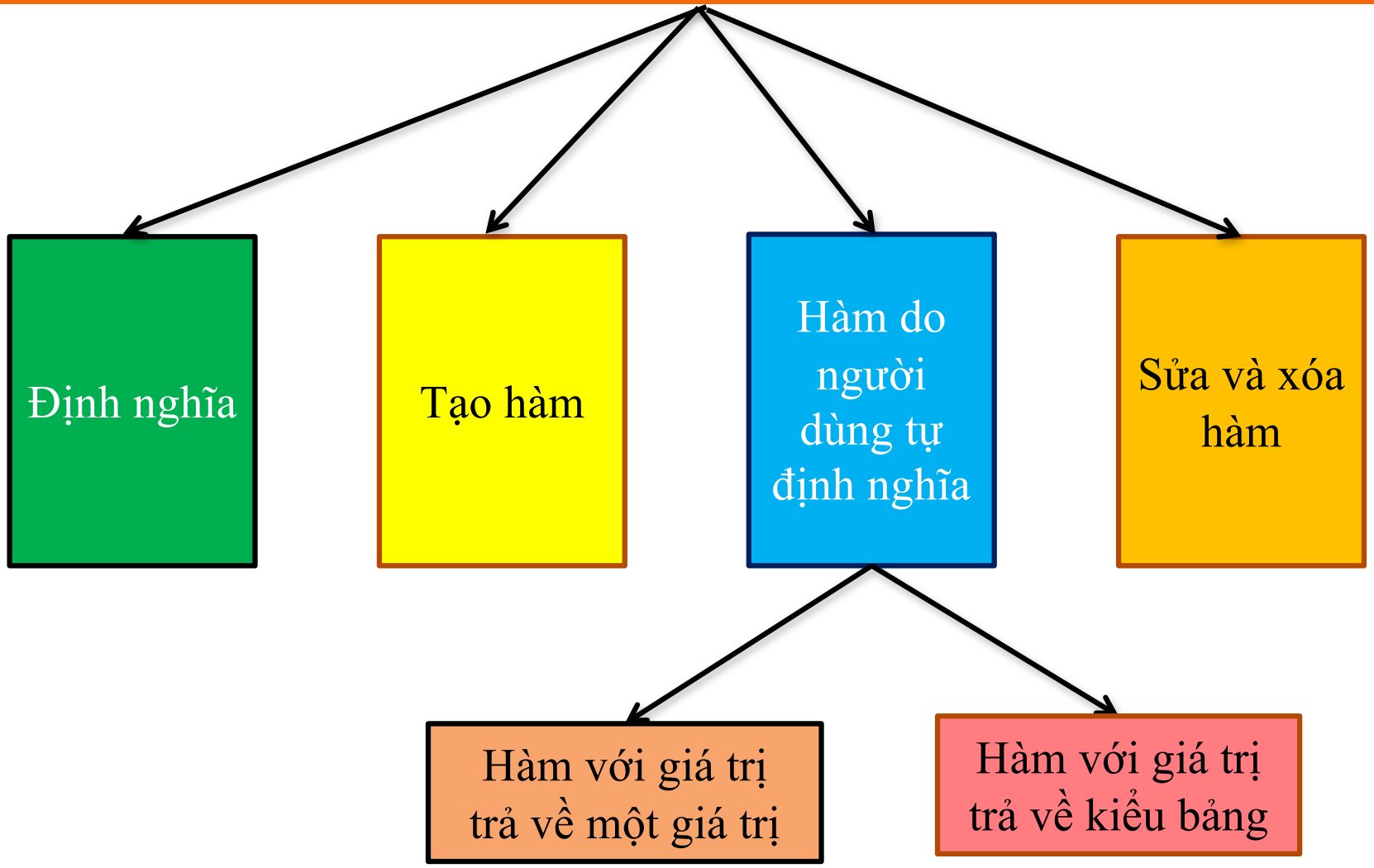
Lợi ích khi sử dụng thủ tục lưu trữ

- ✓ Tăng tốc độ thực hiện
- ✓ Tốc độ truy cập dữ liệu nhanh hơn
- ✓ Chương trình được modul hóa
- ✓ Giảm thiểu sự lưu thông trên mạng
- ✓ Nâng cao khả năng bảo mật

Thủ tục lưu trữ
(Stored procedure)

Hàm
(Function)

Hàm


Định nghĩa
Tạo hàm
Hàm do người dùng tự định nghĩa
Sửa và xóa hàm
Hàm với giá trị trả về một giá trị
Hàm với giá trị trả về kiểu bảng

Thế nào là hàm?

➢ ***Ý nghĩa***

- ✓ Hàm là một đối tượng trong cơ sở dữ liệu tương tự như thủ tục
- ✓ Điểm khác giữa hàm và thủ tục là hàm trả về giá trị thông qua tên hàm còn thủ tục thì không
- ✓ Có thể sử dụng hàm như một thành phần của một biểu thức
- ✓ Có hàm do HQT CSDL cung cấp sẵn
- ✓ Có hàm do người dùng định nghĩa để phục vụ cho những mục đích riêng của mình

Cách tạo hàm

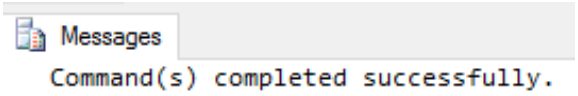
➢ ***Cú pháp***

```
CREATE FUNCTION tên_hàm ([danh_sách_tham_số])
RETURNS (kiểu_trả_về_của_hàm)
AS
BEGIN
    các_câu_lệnh_của_hàm
    RETURN giá trị
END
```

Hàm với giá trị trả về là một giá trị

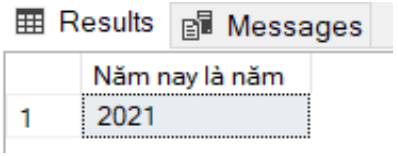
➢ ***Ví dụ 1*** Hàm lấy ra giá trị là năm hiện hành

```
CREATE FUNCTION GetCurrYear1()
RETURNS int
AS
BEGIN
 RETURN   YEAR(getdate())
END
```




```
SELECT dbo.GetCurrYear1() 'Năm nay là năm'
```

➢ ***Kết quả***



Hàm với giá trị trả về là một giá trị

➢ ***Ví dụ 2*** Viết hàm xác định thứ trong tuần của một giá trị kiểu ngày

DATEPART(dangthoigian, thoigian)

Giá trị	Giải thích
year, yyyy, yy	Năm
quarter, qq, q	Quý
month, mm, m	Tháng
dayofyear	Ngày trong năm
day, dy, y	Ngày
week, ww, wk	Tuần
weekday, dw, w	Ngày trong tuần
hour, hh	Giờ
minute, mi, n	Phút
second, ss, s	Giây
millisecond, ms	Milli giây

```
select DATEPART(dw,'2021/09/20') 'Ngày thứ'
```

Results | Messages

	Ngày thứ
1	2

```
select DATEPART(dw,'2021/09/25') 'Ngày thứ'
```

Results | Messages

	Ngày thứ
1	7

```
select DATEPART(dw,'2021/09/26') 'Ngày thứ'
```

Results | Messages

	Ngày thứ
1	1

Hàm với giá trị trả về là một giá trị

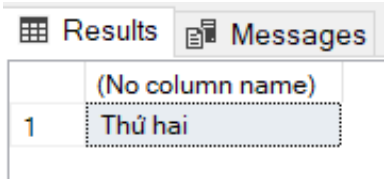
➢ ***Ví dụ 3*** Viết hàm xác định thứ trong tuần của một giá trị kiểu ngày

```
CREATE FUNCTION thu(@ngay DATE)
RETURNS NVARCHAR(10)
AS
BEGIN
 DECLARE @st NVARCHAR(10)
 SELECT @st=CASE DATEPART(DW,@ngay)
    WHEN 1 THEN N'Chủ 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
```

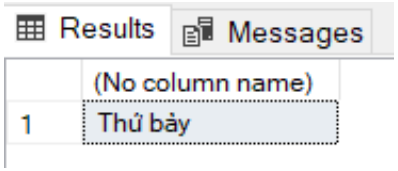
Hàm với giá trị trả về một giá trị

➢ ***Kết quả***

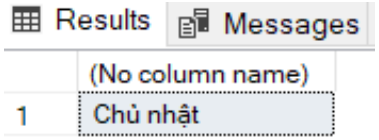
```
set dateformat dmy
select dbo.thu('20/09/2021')
```




```
select dbo.thu('25/09/2021')
```




```
select dbo.thu('26/09/2021')
```



Cách sử dụng hàm

- ***Ví dụ 4*** Xem danh sách sinh viên sinh vào thứ mấy trong tuần

```
SELECT MaSV, HotenSV, Ngaysinh, dbo.thu(ngaysinh) AS 'Thứ trong tuần'
FROM Sinhvien
```

- ***Kết quả***

Results | Messages

	MaSV	HotenSV	Ngaysinh	Thứ trong tuần
1	AT1	Cao Thu Huyền	1998-10-10	Thứ bảy
2	AT2	Nguyễn Thị Hải	1998-11-15	Chủ nhật
3	AT3	Hoàng Minh Chiến	1998-11-10	Thứ ba
4	CN1	Trần Mạnh Cường	1998-12-01	Thứ ba
5	CN2	Lê Văn Minh	1998-09-20	Chủ nhật
6	DT1	Nguyễn Bảo lâm	1998-05-12	Thứ ba
7	DT3	Vũ Tuấn Đạt	1998-09-13	Chủ nhật

Hàm với giá trị trả về kiểu bảng

➢ ***Định nghĩa***

Hàm kiểu bảng (table-valued function) là một loại hàm do người dùng định nghĩa, trong đó kết quả trả về là một cấu trúc kiểu bảng. Có thể sử dụng như một bảng thông thường, như `SELECT` hay `JOIN` với nó.

Hàm kiểu bảng được chia làm **hai** loại:

1. *Hàm In-line*
2. *Hàm Multi-statement*

Hàm nội tuyến (Inline)

➢ ***Cú pháp***

```
CREATE FUNCTION tên_hàm ([danh_sách_tham_số])
RETURNS TABLE
AS
RETURN (câu_lệnh_select)
```

➢ ***Lưu ý***

1. Kiểu trả về của hàm phải được chỉ định bởi mệnh đề RETURNS TABLE.
2. Trong phần thân của hàm chỉ có duy nhất một câu lệnh RETURN xác định giá trị trả về của hàm thông qua duy nhất một câu lệnh SELECT.

Hàm nội tuyến (Inline)

➢ ***Ví dụ 1*** Tạo hàm xem điểm của sinh viên theo lớp

```
CREATE FUNCTION Diem_Lop
(@lop varchar(10)
)
RETURNS TABLE
AS
RETURN
(
SELECT SV.MaSV, HotenSV, TenMH, Diem
FROM Sinhvien SV, Monhoc MH, Ketqua KQ
WHERE SV.MaSV = KQ.MaSV and KQ.MaMH = MH.MaMH and Lop = @lop
)
```

Hàm nội tuyến (Inline)

Sau khi khởi tạo thành công, ta có thể thực thi hàm bằng cách:

```
SELECT * FROM dbo.Diem_Lop ('AT16A')
```

➢ ***Kết quả***

Results | Messages

	MaSV	HotenSV	Lop	TenMH	Diem
1	AT1	Cao Thu Huyền	AT16A	Cơ sở dữ liệu	8
2	AT1	Cao Thu Huyền	AT16A	Hệ thống máy tính	9
3	AT1	Cao Thu Huyền	AT16A	Mạng máy tính	7
4	AT1	Cao Thu Huyền	AT16A	Tin học cơ sở	9

Hàm bao gồm nhiều câu lệnh bên trong (Multi-statement)

Trong trường hợp cần sử dụng nhiều câu lệnh trong phần thân hàm, ta sử dụng cú pháp như sau:

```
CREATE FUNCTION tên_hàm([danh_sách_tham_số])
RETURNS @biến_bảng TABLE định_nghĩa_bảng
AS
BEGIN
     các_câu_lệnh_trong_thân_hàm
 RETURN
END
```

Hàm bao gồm nhiều câu lệnh bên trong (Multi-statement)

➢ ***Ví dụ*** Tạo hàm cho biết số sinh viên của mỗi lớp đã thi môn nhập vào

```
CREATE FUNCTION Func_TongSV
(@mamon char(5) )
RETURNS @bangthongke TABLE
(
lop varchar(10),
tongsosv int
)
AS
BEGIN
INSERT INTO @bangthongke
SELECT Lop, count(SV.MaSV)
FROM Sinhvien SV, Ketqua KQ , Monhoc MH
WHERE SV.MaSV = KQ.MaSV and Mh.MaMH = KQ.MaMH
      and Mh.MaMH = @mamon
GROUP BY Lop
RETURN
END
```

Hàm bao gồm nhiều câu lệnh bên trong (Multi-statement)

Sau khi khởi tạo thành công, ta có thể thực thi hàm bằng cách:

```
SELECT * FROM Func_TongSV('CSDL')
```

➢ ***Kết quả***

Results | Messages

	lop	tongsosv
1	AT16A	1
2	AT16K	1
3	CT4A	2

Sửa và xóa hàm

1. *Sửa hàm*

```
ALTER FUNCTION tên_hàm ([danh_sách_tham_số])
RETURNS (kiểu_trả_về_của_hàm)
AS
BEGIN
    các_câu_lệnh_của_hàm
END
```

2. *Xóa hàm*

```
DROP FUNCTION tên_hàm
```

Ôn tập phần thủ tục

Câu 1: Viết một thủ tục đưa ra các sinh viên có năm sinh bằng với năm sinh được nhập vào (lấy năm sinh bằng hàm datepart(yyyy,ngaysinh))

Câu 2: Viết thủ tục xóa dữ liệu của sinh viên theo mã sinh viên

Câu 3: Viết thủ tục sửa tên lớp của sinh viên theo mã sinh viên

Câu 4: Viết một thủ tục kiểm tra xem hai sinh viên có cùng năm sinh hay không

Câu 5: Viết thủ tục so sánh 2 sinh viên có mã được nhập vào xem sinh viên nào được sinh trước.

Ôn tập phần hàm

Câu 1: Viết hàm đưa ra ngày hiện tại

Câu 2: Viết một hàm đưa ra quý sinh. Áp dụng để đưa ra danh sách các bạn sinh viên sinh quý 2

Câu 3: Viết hàm chỉ ra năm nhuận hay không nhuận của một năm nhập vào

Câu 4: Viết hàm đưa ra các sinh viên có năm sinh bằng với năm sinh được nhập vào

Câu 5: Viết hàm kiểm tra xem hai sinh viên có cùng năm sinh hay không

Câu 6: Viết hàm so sánh 2 sinh viên có mã được nhập vào xem sinh viên nào được sinh trước.