

# **Transact-SQL nâng cao**

# Nội dung

- Khai báo và sử dụng biến
- Các lệnh điều khiển
- Cursor
- Stored procedure
- Function
- Sử dụng Trigger cài đặt RBTV

# Biến cục bộ

- Là một đối tượng có thể chứa giá trị thuộc một kiểu dữ liệu nhất định
- Tên biến:
  - Bắt đầu bằng một ký tự @
- Tầm vực của biến:
  - Biến cục bộ có giá trị trong một *query batch* hoặc trong một stored procedure/ function

# Biến cục bộ - Khai báo

- Khai báo biến cục bộ bằng lệnh declare

- Cung cấp tên biến và kiểu dữ liệu

**Declare** *tên\_biến* *Kiểu\_dữ\_liệu*

- Ví dụ:

**Declare** @MaSinhVien *char*(10)

**Declare** @HoTen *nvarchar*(30)

**Declare** @Sum *float*, @Count *int*

# Biến cục bộ - Gán giá trị

- Dùng lệnh set để gán giá trị cho biến
  - Giá trị gán cho biến phải phù hợp với kiểu dữ liệu của biến

**Set** *tên\_biến = giá\_trị*

**Set** *tên\_biến = tên\_biến*

**Set** *tên\_biến = biểu\_thức*

**Set** *tên\_biến = kết\_quả\_truy\_vấn*

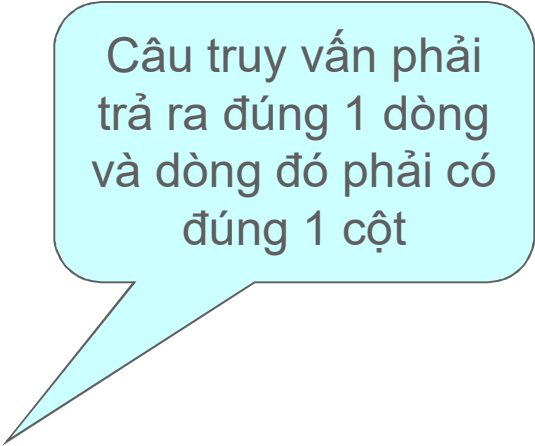
# Gán giá trị (tt)

- Ví dụ

Set @MaLop = 'TH2001'

Set @SoSV = (select count (\*) from SinhVien)

Set @MaLop = 'TH'+Year(@NgayTuyenSinh)



Câu truy vấn phải  
trả ra đúng 1 dòng  
và dòng đó phải có  
đúng 1 cột

## Biến cục bộ - Gán giá trị (tt)

- Đưa kết quả truy vấn vào biến:

Ví dụ :

SV(MaSV: int; HoTen: nvarchar(30), Tuoi int)

*Select @Var1 = HoTen, @Var2 = Tuoi from SV  
where MaSV = 1*

- Lưu ý: nếu câu truy vấn trả về nhiều dòng, các biến chỉ nhận giá trị tương ứng của dòng đầu tiên

# Biến toàn cục

- Là các biến hệ thống do SQL Server cung cấp
  - Tên biến bắt đầu bằng 2 ký tự @
  - SQL tự cập nhật giá trị cho các biến này, NSD không thể gán giá trị trực tiếp



- Một số biến hệ thống thường dùng
  - @@error
  - @@rowcount
  - @@trancount
  - @@fetch\_status

# Nội dung

- Khai báo và sử dụng biến
- Các lệnh điều khiển
- Cursor
- Stored procedure
- Function
- Sử dụng Trigger cài đặt RBTV

## If...else

- Xét điều kiện để quyết định những lệnh T-SQL nào sẽ được thực hiện
- Cú pháp:

*If biểu\_thức\_điều\_kiện*

*Lệnh| Khối\_lệnh*

*[Else Lệnh| Khối\_lệnh]*

*Khối lệnh là một hoặc nhiều lệnh nằm trong cặp từ khóa begin...end*

## If...else (tt)

- Ví dụ

**HocPhan(MaHP, TenHP, SiSo)**

**DangKy(MaSV, MaHP)**

*Viết lệnh để thêm một đăng ký mới cho sinh viên có mã số 001 vào học phần HP01 (giả sử học phần này đã tồn tại trong bảng HocPhan)*

# If...else (tt)

- Ví dụ

Declare @SiSo int

select @SiSo = SiSo from HocPhan where MaHP= 'HP01'

if @SiSo < 50

**Begin**

insert into DANG\_KY(MaSV, MaHP)

values('001', 'HP01')

print N'Đăng ký thành công'

**End**

**Else**

print N'Học phần đã đủ SV'

# While

- Thực hiện lặp lại một đoạn lệnh T-SQL khi điều kiện còn đúng
- Cú pháp

**While** *biểu\_thức\_điều\_kiện*

*Lệnh* | *Khởi lệnh*

- Có thể sử dụng *Break* và *Continue* trong khối lệnh của while
  - ✓ Break: thoát khỏi vòng while hiện hành
  - ✓ Continue : trở lại đầu vòng while, bỏ qua các lệnh sau đó

# While (tt)

- Ví dụ

**SinhVien(MaSV: int, HoTen: nvarchar(30))**

*Viết lệnh xác định một mã sinh viên mới theo qui định: mã sinh viên tăng dần, nếu có chỗ trống thì mã mới xác định sẽ chèn vào chỗ trống đó*

Vd: 1,2,3,7 → mã sinh viên mới: 4

## While (tt)

- Ví dụ:

*Declare @STT int*

*Set @STT=1*

**While** *exists(select \* from SV where MaSV = @STT)*

*set @STT = @STT+1*

*Insert into SV(MaSV, HoTen)*

*values(@STT, 'Nguyen Van A')*



## Case

- Kiểm tra một dãy các điều kiện và trả về kết quả phù hợp với điều kiện đúng
- Được sử dụng như một hàm trong câu select

## Case (tt)

- **Cú pháp: Có hai dạng**

- Dạng 1 (simple case):

**Case** *Biểu\_thức\_đầu\_vào*

**When** *Giá\_trị* **then** *kết\_quả*

[...n]

[ **Else** *kết\_quả\_khác* ]

**End**

## Case (tt)

- Dạng 2 (searched case):

**Case**

**When** *biểu\_thức\_điều\_kiện* **then** *kết\_quả*

[...n]

[ **Else** *kết\_quả\_khác* ]

**End**

## Case (tt)

- Ví dụ:

**NHAN\_VIEN(MaNV, HoTen, NgaySinh, CapBac,  
Phai)**

- Cho biết những nhân viên đến tuổi về hưu(tuổi về hưu của nam là 60, của nữ là 55)

## Case - ví dụ (tt)

```
select * from NHAN_VIEN  
where datediff(yy, NgaySinh, getdate()) > =
```

**Case** Phai

**when** 'Nam' **then** 60

**when** 'Nu' **then** 55

**End**

## Case \_ VD (tt)

- Cho biết mã NV, họ tên và loại nhân viên (cấp bậc <=3: bình thường, cấp bậc = null: chưa xếp loại, còn lại: cấp cao)

```
Select MaNV, HoTen, 'Loai' = Case  
    when CapBac<=3 then 'Binh Thuong'  
    when CapBac is null then 'Chua xep loai'  
    else 'Cap Cao' End
```

```
From NhanVien
```

# Nội dung

- Khai báo và sử dụng biến
- Các lệnh điều khiển
- Cursor
- Stored procedure
- Function
- Sử dụng Trigger cài đặt RBTV

## Cursor - Khái niệm

- Là một cấu trúc dữ liệu ánh xạ đến một tập các dòng dữ liệu là kết quả của một câu truy vấn (select)
- Cho phép duyệt tuần tự qua tập các dòng dữ liệu và đọc giá trị từng dòng.



## Cursor - Khái niệm (tt)

- Vị trí hiện hành của cursor có thể được dùng như điều kiện trong mệnh đề where của lệnh update hoặc delete
  - cho phép cập nhật/xoá dữ liệu (dữ liệu thật sự trong CSDL) tương ứng với vị trí hiện hành của cursor

# Cursor - Khai báo

- Có thể sử dụng cú pháp chuẩn SQL 92 hoặc cú pháp T\_SQL mở rộng
  - Cú pháp SQL 92 chuẩn:

**Declare** *cur\_name* [**Insensitive**] [**Scroll**] **Cursor**

**For** *select\_statement*

[ **For** {**Read only**| **Update** [**of** *column\_name* [,...n] ] } ]

# Cursor – Khai báo (tt)

– Cú pháp T\_SQL mở rộng

**Declare** cursor\_name **Cursor**

[ **Local** | **Global** ]

[ **Forward\_only** | **Scroll** ]

[ **Static** | **Dynamic** ]

[ **Read\_only** ]

**For** select\_statement

[ **For Update** [ **of** column\_name [,...n] ] ]

Chú ý: Tên cursor trong các cách khai báo không bắt đầu bằng ký tự “@.

## Cursor – Khai báo (tt)

- Ý nghĩa các tham số tùy chọn:
  - ***Insensitive/ static***: nội dung của cursor không thay đổi trong suốt thời gian tồn tại, trong trường hợp này cursor chỉ là read only
  - ***Dynamic***: trong thời gian tồn tại, nội dung của cursor có thể thay đổi nếu dữ liệu trong các bảng liên quan có thay đổi.

## Cursor – Khai báo (tt)

- **Local:** cursor cục bộ, chỉ có thể sử dụng trong phạm vi một khối (query batch) hoặc một thủ tục/hàm
- **Global:** cursor toàn cục (tồn tại trong suốt connection hoặc đến khi bị hủy tường minh)

- ***Forward\_only***: cursor chỉ có thể duyệt một chiều từ đầu đến cuối
- ***Scroll***: có thể duyệt lên xuống cursor tùy ý
- ***Read only***: chỉ có thể đọc từ cursor, không thể sử dụng cursor để update dữ liệu trong các bảng liên quan (ngược lại với “for update...” )

# Cursor – Khai báo (tt)

## Mặc định:

- Global
- Forward\_only
- Read only hay “for update” tùy thuộc vào câu truy vấn
- Dynamic

# Duyệt cursor

- Dùng lệnh Fetch để duyệt tuần tự qua cursor

**Fetch**

[ [Next| Prior| First| Last| Absolute n| Relative n]

**From** ] Tên\_cursor

[Into Tên\_biến [,...n] ]



## Duyệt cursor (tt)

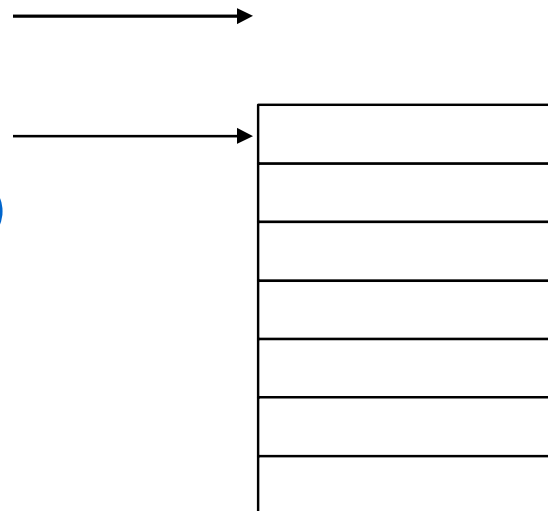
- Mặc định : fetch next
- Đối với cursor dạng forward\_only, chỉ có thể fetch next
- Biến hệ thống @@fetch\_status cho biết lệnh fetch vừa thực hiện có thành công hay không
  - ✓ Là cơ sở để biết đã duyệt đến cuối cursor hay chưa

# Duyệt cursor (tt)

Trước lệnh fetch đầu tiên:  
@@fetch\_status không xác định  
Fetch next lần đầu tiên:  
@@fetch\_status = 0 (thành công)

...

@@ fetch\_status <> 0



# Cursor – Trình tự sử dụng

- Khai báo cursor
- “Mở” cursor bằng lệnh **Open**

**Open** tên\_cursor

- Fetch (next,...) cursor để chuyển đến vị trí phù hợp
  - ✓ Có thể đưa các giá trị của dòng hiện hành vào các biến thông qua mệnh đề into của lệnh fetch
  - ✓ Nếu không có mệnh đề into, các giá trị của dòng hiện hành sẽ được hiển thị ra cửa sổ kết quả (result pane) sau lệnh fetch
  - ✓ Có thể sử dụng vị trí hiện tại như là điều kiện cho mệnh đề where của câu delete/ update (nếu cursor không là read\_only)

## Cursor - Trình tự sử dụng (tt)

- Lặp lại việc duyệt và sử dụng cursor, có thể sử dụng biến @@fetch\_status để biết đã duyệt qua hết cursor hay chưa.

- Đóng cursor bằng lệnh **Close**

**Close** Tên\_cursor

- ✓ Sau khi đóng, vẫn có thể mở lại nếu cursor chưa bị hủy

- Hủy cursor bằng lệnh **deallocate**

**Deallocate** Tên\_cursor

## Cursor – Ví dụ

SINHVIEN (MaSV, HoTen, MaKhoa)

KHOA(MaKhoa, TenKhoa)

- Ví dụ 1: Duyệt và đọc giá trị từ cursor

Cập nhật lại giá trị MaSV = Viết tắt tên Khoa + MaSV hiện tại cho tất cả sinh viên

## Cursor – Ví dụ (tt)

```
declare cur_DSKhoa cursor  
for select MaKhoa, TenKhoa from Khoa  
open cur_DSKhoa  
declare @MaKhoa int,  
        @TenKhoa varchar(30), @TenTat varchar(5)  
fetch next from cur_DSKhoa into @MaKhoa,  
@TenKhoa
```

## Cursor – ví dụ (tt)

```
while @@fetch_status = 0
```

```
begin
```

```
-- xác định tên tắt của Khoa dựa vào @TenKhoa...
```

```
update SinhVien set MaSV = @TenTat+MaSV
```

```
Where MaKhoa = @MaKhoa
```

```
    fetch next from cur_DSKhoa into @MaKhoa,  
    @TenKhoa
```

```
end
```

```
Close cur_DSKhoa
```

```
Deallocate cur_DSKhoa
```

# Cursor – Ví dụ (tt)

- Ví dụ 2: dùng cursor để xác định dòng cập nhật

```
declare cur_DSKhoa cursor scroll  
for select MaKhoa, TenKhoa from Khoa  
open cur_DSKhoa  
fetch absolute 2 from cur_DSKhoa  
if (@@fetch_status = 0)  
    update Khoa  
        set TenKhoa = 'aaa'  
        where current of cur_DSKhoa  
Close cur_DSKhoa  
Deallocate cur_DSKhoa
```



# Biến cursor

- Ta có thể khai báo một biến kiểu cursor và gán cho nó tham chiếu đến một cursor đang tồn tại.
- Biến cursor có thể được xem như là con trỏ cursor
- Biến cursor là một biến cục bộ
- Biến cursor sau khi gán giá trị được sử dụng như một cursor thông thường.

# Biến cursor (tt)

- Ví dụ :

**Declare @cur\_var cursor**

**set @cur\_var = my\_cur** -- *my\_cur là một cursor đang tồn tại*

*Hoặc:*

**Declare @cur\_var cursor**

**set @cur\_var = cursor for select\_statement**

# Nội dung

- Khai báo và sử dụng biến
- Các lệnh điều khiển
- Cursor
- Stored procedure
- Function
- Sử dụng Trigger cài đặt RBTV

# Stored procedure – Khái niệm

- Thủ tục “nội” | thủ tục thường trú
  - Thủ tục:
    - ✓ Chứa các lệnh T\_SQL
    - ✓ Tương tự như một thủ tục trong các ngôn ngữ lập trình: có thể truyền tham số, có tính tái sử dụng
  - Nội, thường trú: được dịch và lưu trữ thành một đối tượng trong CSDL

# Stored procedure – Khái niệm (tt)

- Ý nghĩa:
  - Tính tái sử dụng.
  - Các lệnh trong stored procedure được tối ưu hóa một lần khi dịch → tiết kiệm thời gian khi thực thi.
  - Giảm khối lượng thông tin trao đổi khi ứng dụng gửi yêu cầu thực hiện công việc về database server
  - Hỗ trợ tốt hơn cho việc đảm bảo an toàn (security) cho CSDL

# Stored procedure (tt)

- Cú pháp

**Create** {**proc** | **procedure**} *procedure\_name*

{*Parameter\_name* *DataType* [=default] [**output**] }[,...n]

**As**

...

[**return** [*return\_value*] ]

**Go**

**Lưu ý:**

- ✓ Tên tham số đặt theo qui tắc như tên biến cục bộ
- ✓ Chỉ có thể trả về giá trị kiểu int

# Stored procedure (tt)

- Ví dụ:

- Viết thủ tục thêm một đăng ký của sinh viên vào một học phần (tổng quát ví dụ trong phần If ...else)

**Create procedure** usp\_ThemDangKy

    @MaSV char(5),

    @MaHP char(5),

    @SiSo int = 0 **output**

**As**

# Stored procedure (tt)

```
select @SiSo = SiSo from HocPhan where MaHP= @MaHP
if @SiSo < 50
    Begin
        insert into DANG_KY(MaSV, MaHP)
            values(@MaSV, @MaHP)
        set @SiSo = @SiSo+1
        return 1
    End
return 0
Go
```



# Stored proc – gọi thực hiện

- Cú pháp:

**EXEC| EXECUTE**

{ [ @return\_status = ] *procedure\_name*

{ [ @parameter\_name = ] *value* [ **OUTPUT** ] } [ ,...*n* ]

## Lưu ý:

- ✓ Có thể truyền giá trị cho tham số input là một hằng hoặc một biến đã gán giá trị, không truyền được một biểu thức.
- ✓ Để nhận được giá trị output, truyền vào một biến cho một tham số output.

# Stored proc – gọi thực hiện (tt)

- Ví dụ:

- **Exec** usp\_ThemDangKy '001', 'HP01'
- **Exec** usp\_ThemDangKy @MaHP = 'HP01', @MaSV = '001'
- **Declare** @SiSo int  
**Exec** usp\_ThemDangKy '001','HP01', @SiSo **output**
- **Declare** @SiSo int, @KetQua int  
**Exec** @KetQua = usp\_ThemDangKy '001','HP01', @SiSo **output**

# Stored procedure (tt)

- Sửa thủ tục

Thay từ khóa **Create** trong lệnh tạo thủ tục bằng từ khóa **Alter**

- Xóa thủ tục

**Drop {procedure|proc} *procedure\_name***

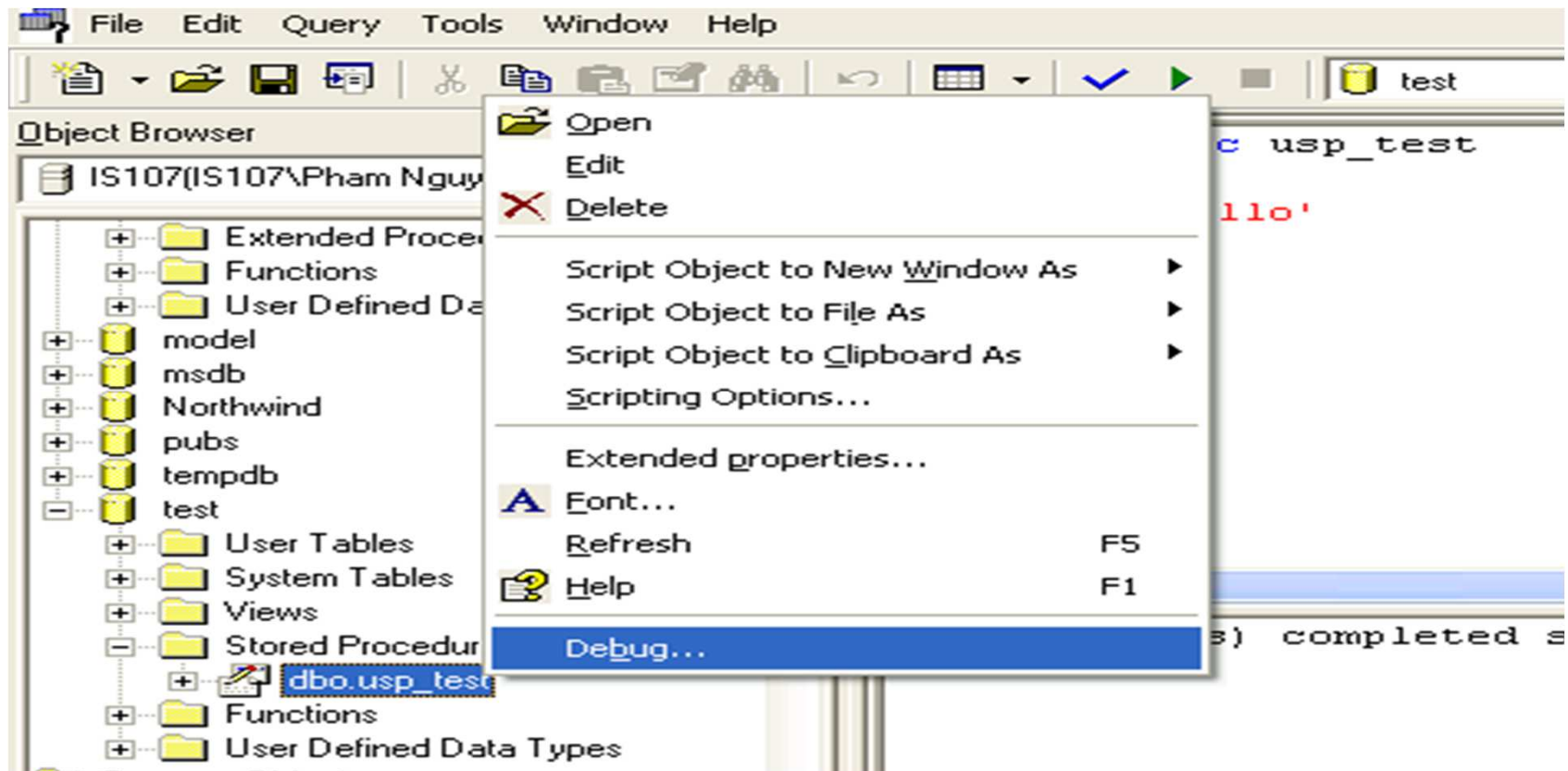
Ví dụ:

**Drop procedure usp\_ThemDangKy**

## Stored procedure hệ thống

- SQL Server cung cấp sẵn nhiều thủ tục thực hiện các công việc: quản lý CSDL, quản lý người dùng, cấu hình CSDL,...
- Các thủ tục này có tên bắt đầu bằng “sp\_”
  - → Khi xây dựng thủ tục, tránh đặt tên thủ tục với “sp\_” ở đầu.

# Debug stored procedure



# Nội dung

- Khai báo và sử dụng biến
- Các lệnh điều khiển
- Cursor
- Stored procedure
- Function
- Sử dụng Trigger cài đặt RBTV

# Hàm người dùng (user function) – khái niệm

- **Giống stored procedure:**
  - Là mã lệnh có thể tái sử dụng
  - Chấp nhận các tham số input
  - Dịch một lần và từ đó có thể gọi khi cần
- **Khác stored procedure**
  - Chấp nhận nhiều kiểu giá trị trả về (chỉ một giá trị trả về)
  - Không chấp nhận tham số out put
  - Khác về cách gọi thực hiện

## Hàm người dùng – khái niệm (tt)

- Có thể xem hàm người dùng thuộc về 3 loại tùy theo giá trị trả về của nó :
  - Giá trị trả về là kiểu dữ liệu cơ sở (int, varchar, float, datetime...)
  - Giá trị trả về là Table có được từ một câu truy vấn
  - Giá trị trả về là table mà dữ liệu có được nhờ tích lũy dần sau một chuỗi thao tác xử lý và insert.



# Hàm người dùng – khai báo

- Loại 1: Giá trị trả về là kiểu dữ liệu cơ sở

Create function *func\_name*

( {*parameter\_name* *DataType* [= *default* ] } [...n])

returns *DataType*

Dù không có tham số cũng phải ghi cặp ngoặc rỗng

As

Begin

Dù thân function chỉ có 1 lệnh cũng phải đặt giữa **Begin** và **End**

...

Return {*value* | *variable* | *expression*}

End

# Hàm người dùng – khai báo (tt)

– Ví dụ:

**Create function** SoLonNhat

(@a int,@b int,@c int) **returns** int

**As**

**Begin**

declare @max int

set @max = @a

if @b > @max set @max = @b

if @c > @max set @max = @c

**return** @max

**End**

# Hàm người dùng – khai báo (tt)

- Loại 2: Giá trị trả về là Table có được từ một câu truy vấn

Create function *func\_name*

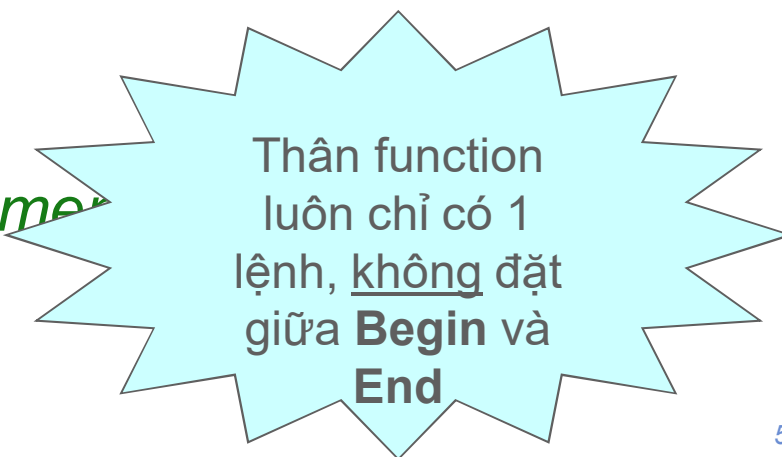
( {*parameter\_name* *DataType* [= *default* ] } [,...n])

returns Table

As

Return [ ( ]*select\_statement*

Go



Thân function  
luôn chỉ có 1  
lệnh, không đặt  
giữa **Begin** và  
**End**

# Hàm người dùng – khai báo (tt)

– Ví dụ

**Create function** DanhSachMatHang

( @MaDonHang varchar(10) ) **returns Table**

**As**

**Return**

(Select MH.TenHang,MH.DonGia  
From ChiTietDH CT, MatHang MH  
Where CT.MaDH = @MaDonHang  
and CT.MaMH = MH.MaMH)

**Go**

# Hàm người dùng – khai báo (tt)

- Loại 3: Giá trị trả về là table mà dữ liệu có được nhờ tích lũy dần sau một chuỗi thao tác xử lý và insert.

**Create function** func\_name

( {parameter\_name DataType [= default ] } [...n])

**returns** TempTab\_name **Table**(Table\_definition)

**As**

**Begin**

...

**Return**

**End**

# Hàm người dùng – khai báo (tt)

– Ví dụ:

```
Create function DanhSachLop ()  
returns @DS Table(@MaLop varchar(10),@SoSV int)  
As  
    --các xử lý insert dữ liệu vào bảng DS  
    return  
Go
```

## Hàm người dùng – Khai báo (tt)

- Lưu ý : Trong thân hàm không được sử dụng các hàm hệ thống bất định (Built-in nondeterministic functions ), bao gồm :
  - GETDATE
  - GETUTCDATE
  - NEWID
  - RAND
  - TEXTPTR
  - @@TOTAL\_ERRORS, @@CPU\_BUSY, @@TOTAL\_READ, @@IDLE, @@TOTAL\_WRITE, @@CONNECTIONS ...

# Hàm người dùng – sử dụng

- Các hàm người dùng được sử dụng trong câu truy vấn, trong biểu thức... phù hợp kiểu dữ liệu trả về của nó
- Ví dụ:
  - Select **dbo.SoLonNhat(3,5,7)**
  - Select \* from **DanhSachLop()**



## Hàm người dùng – sử dụng (tt)

- Lưu ý:

- Nếu dùng giá trị mặc định của tham số, phải dùng từ khóa **default**
- Khi gọi hàm loại 1 (trả về giá trị cơ bản), phải có tên owner của hàm đi kèm (ví dụ `dbo.SoLonNhat`)

## Hàm người dùng (tt)

- Thay đổi hàm người dùng

Thay từ khóa **create** trong các lệnh tạo hàm bằng từ khóa **alter**

- Xóa hàm người dùng

**Drop function** *tên\_hàm\_cần\_xóa*

– Ví dụ :

**Drop function** DanhSachMatHang

# Các hàm hệ thống

- Ngoài các hàm do người dùng định nghĩa, SQL Server còn cung cấp các hàm xây dựng sẵn của hệ thống
- Các hàm này cung cấp tiện ích như xử lý chuỗi, xử lý thời gian, xử lý số học...
- Sinh viên tìm hiểu thêm về các hàm này trong Books on-line và các tài liệu tham khảo

# Nội dung

- Khai báo và sử dụng biến
- Các lệnh điều khiển
- Cursor
- Stored procedure
- Function
- Sử dụng Trigger cài đặt RBTV

# Trigger -Giới thiệu

- **Là một loại stored procedure đặc biệt**
  - Tự động thực hiện khi có thao tác insert, delete hoặc update trên dữ liệu
  - Thường dùng để kiểm tra các ràng buộc toàn vẹn của CSDL hoặc các quy tắc nghiệp vụ.
  - Một trigger được định nghĩa trên **một** bảng, nhưng các xử lý trong trigger có thể sử dụng nhiều bảng khác.

## Trigger – giới thiệu (tt)

- **Xử lý của trigger thường cần sử dụng đến hai bảng tạm:**
  - **Inserted:** chứa các dòng vừa mới được thao tác insert/ update thêm vào bảng.
  - **Deleted:** chứa các dòng vừa mới bị xóa khỏi bảng bởi thao tác update/delete.

*(update = delete dòng chứa giá trị cũ+ insert dòng chứa giá trị mới)*

## Trigger – giới thiệu

- **Inserted và deleted là các bảng trong bộ nhớ chính**
  - Cục bộ cho mỗi trigger
  - Có cấu trúc giống như bảng (table) mà trigger định nghĩa trên đó
  - Chỉ tồn tại trong thời gian trigger đang xử lý

## Trigger – giới thiệu(tt)

- Nếu thao tác insert/ delete/ update thực hiện trên nhiều dòng, trigger cũng chỉ được gọi một lần
  - Bảng inserted/ deleted có thể chứa nhiều dòng



# Khai báo trigger

- Cú pháp:

**Create trigger** *tên\_trigger*

**On** {*tên\_bảng*|*tên\_view*}

{**For**| **After**| **Instead of** } { [**delete**] [,] [**insert**] [,] [**update**] }

**As**

{ *các lệnh T-sql* }

**go**

# Khai báo trigger (tt)

- **For | After:**

- Trigger được gọi thực hiện sau khi thao tác delete/ insert/ update tương ứng đã được thực hiện thành công
  - ✓ Các dòng mới được thêm chứa **đồng thời** trong bảng dữ liệu và bảng inserted
  - ✓ Các dòng bị xoá chỉ nằm trong bảng deleted (đã bị xoá khỏi bảng dữ liệu)
- Có thể xử lý **quay lui** thao tác đã thực hiện bằng lệnh ***rollback transaction***

# Khai báo trigger (tt)

- **Instead of:**

- Trigger được gọi thực hiện thay cho thao tác delete/ insert/ update tương ứng
  - ✓ Các dòng mới được thêm **chỉ** chứa trong bảng inserted
  - ✓ Các dòng bị chỉ định xoá nằm đồng thời trong bảng deleted và bảng dữ liệu (dữ liệu không bị xoá).
- Trigger *Instead of* thường được dùng để xử lý cập nhật trên khung nhìn (view).

# Khai báo trigger (tt)

- Lưu ý:

- Lệnh tạo trigger phải là lệnh đầu tiên trong một *query batch*
- Trên một bảng có thể định nghĩa nhiều trigger *for/after* cho mỗi thao tác...
- ...nhưng chỉ có thể định nghĩa một trigger *instead of* cho mỗi thao tác

## Khai báo trigger (tt)

- Không thể định nghĩa trigger *instead of update/delete* trên bảng có cài đặt khoá ngoại dạng *update cascade/delete cascade*
- Trong thân trigger, có thể sử dụng hàm *Update(tên\_cột)* để kiểm tra xem việc cập nhật/insert được thực hiện trên cột nào.
  - ✓ *Update(tên\_cột) = true* : có thực hiện cập nhật trên cột *tên\_cột*

## Trigger -Ví dụ 1

- Cho CSDL:

DatHang(MaPDT, NgayDH,...)

GiaoHang(MaPGH, MaPDH, NgayGH,...)

Ràng buộc: Ngày giao hàng không thể nhỏ hơn ngày đặt hàng tương ứng

# Trigger -Ví dụ 1

- **Bảng tầm ảnh hưởng:**

	Thêm	Xoá	Sửa
DatHang	-	-(*)	+(NgayDH)
GiaoHang	+	-	+(NgayGH, MaPDH)

→ Cần cài đặt trigger cho thao tác sửa trên đặt hàng, và thêm/sửa trên giao hàng

## Trigger -Ví dụ 1 (tt)

- Trigger cho thao tác thêm và sửa trên giao hàng:

Create trigger tr\_GH\_ins\_upd\_NgayGH

On GIAOHANG for insert, update

As



## Trigger -Ví dụ 1 (tt)

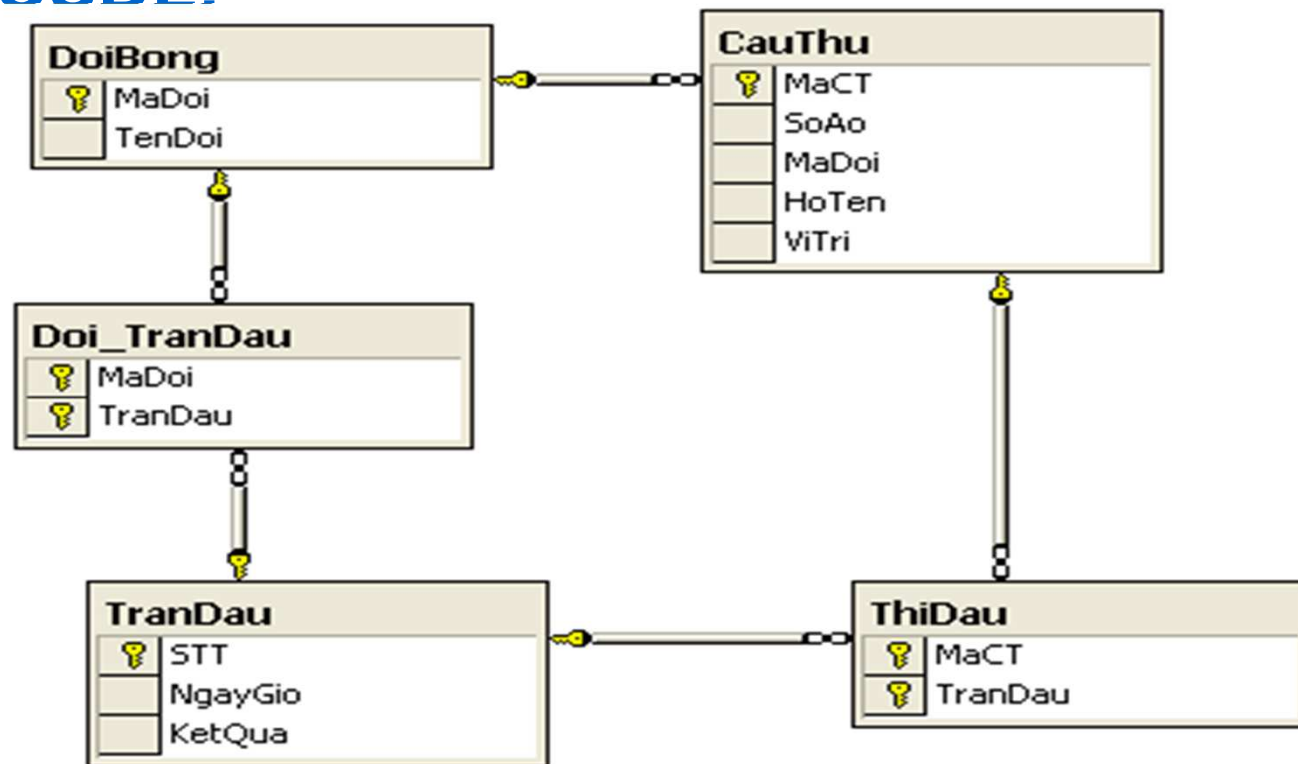
```
if update(MaPDH) or update (NgayGH)
    if exists(select * from inserted i, DatHang d
        where i.MaPDH = d.MaPDH
            and i.NgayGH<d.NgayDH)
        begin
            raiserror (N'Ngày GH không thể nhỏ hơn ngày ĐH,15,1)
            rollback tran
        end
    end
go
```

## Trigger -Ví dụ 1 (tt)

- Trigger cho thao tác sửa trên đặt hàng?

# Trigger - Ví dụ 2

- Cho CSDL:



## Trigger - Ví dụ 2 (tt)

- **Qui định:**
  - Các trận đấu của cùng một đội bóng phải cách nhau tối thiểu 24 giờ
  - Cầu thủ chỉ có thể thi đấu trong một trận đấu mà đội của cầu thủ đó tham gia
- **Cài đặt các trigger để kiểm tra các ràng buộc trên**