

Transact-SQL nâng cao

Nội dung

- 1. Khai báo và sử dụng biến**
- 2. Các lệnh điều khiển**
- 3. Stored procedure**
- 4. Function**
- 5. Sử dụng Trigger cài đặt RBTV**

1. 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, Người sử dụng không thể gán giá trị trực tiếp

- Một số biến hệ thống thường dùng
 - @@error
 - @@rowcount
 - @@trancount
 - @@fetch_status

Nội dung

1. Khai báo và sử dụng biến
2. Các lệnh điều khiển
3. Stored procedure
4. Function
5. 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 | *Khối 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

1. Khai báo và sử dụng biến
2. Các lệnh điều khiển
3. Stored procedure
4. Function
5. 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*

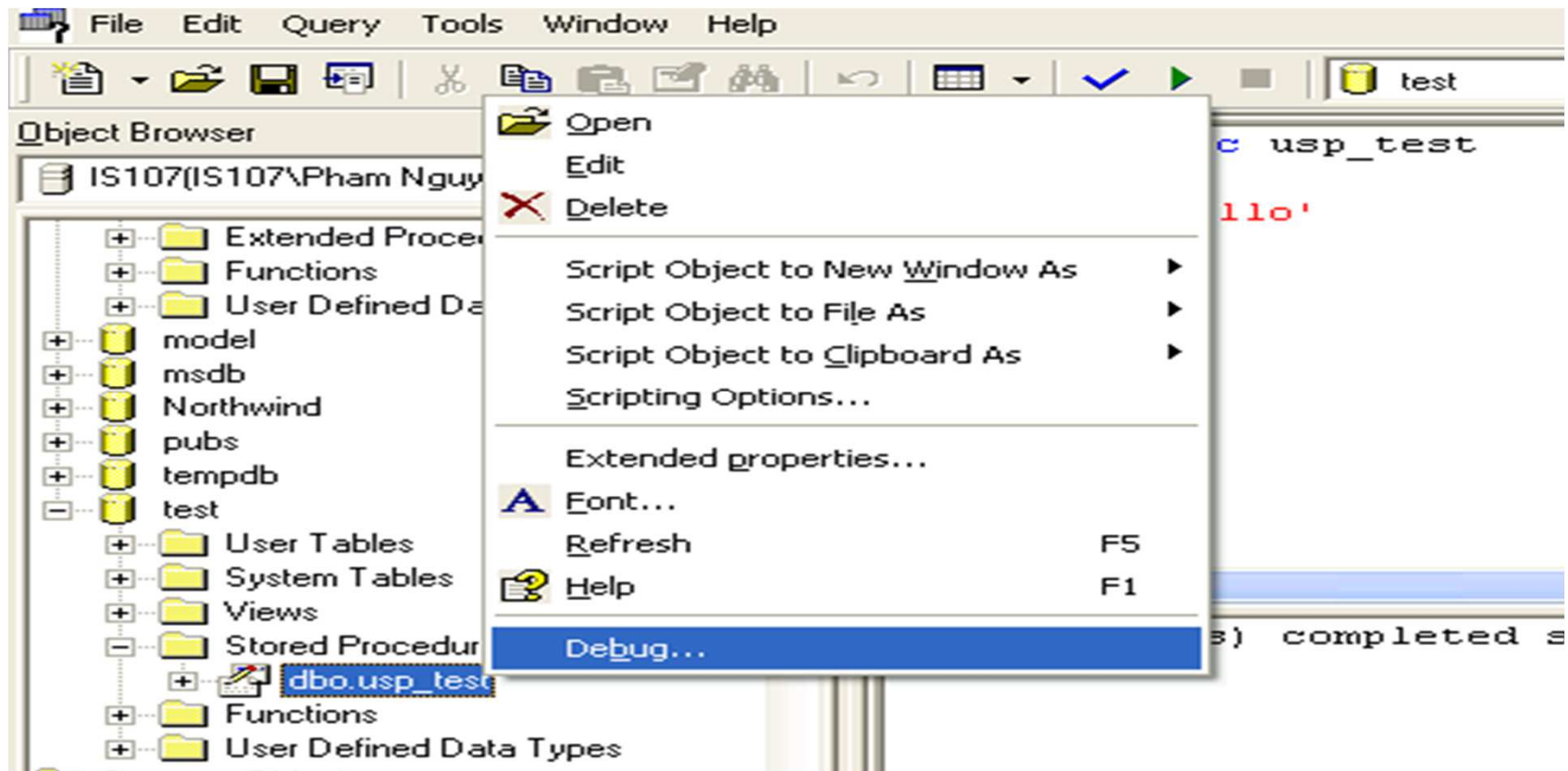
Ví dụ:

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

1. Khai báo và sử dụng biến
2. Các lệnh điều khiển
3. Stored procedure
4. Function
5. 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*

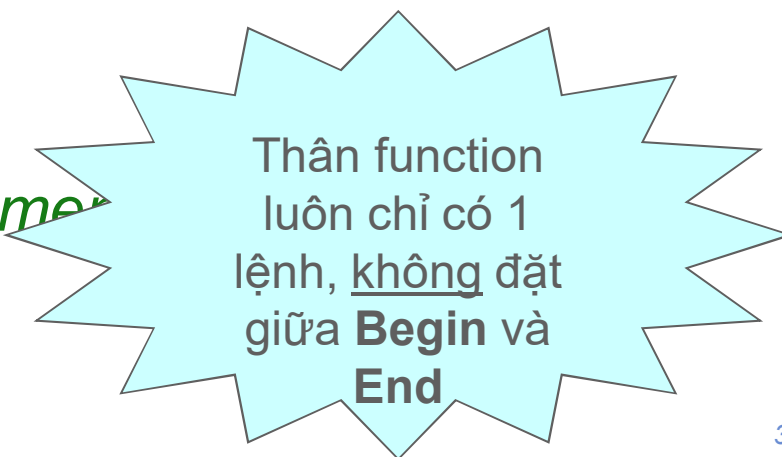
({*parameter_name* *DataType* [= *default*] } [...n])

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

1. Khai báo và sử dụng biến
2. Các lệnh điều khiển
3. Stored procedure
4. Function
5. 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

Khái 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)

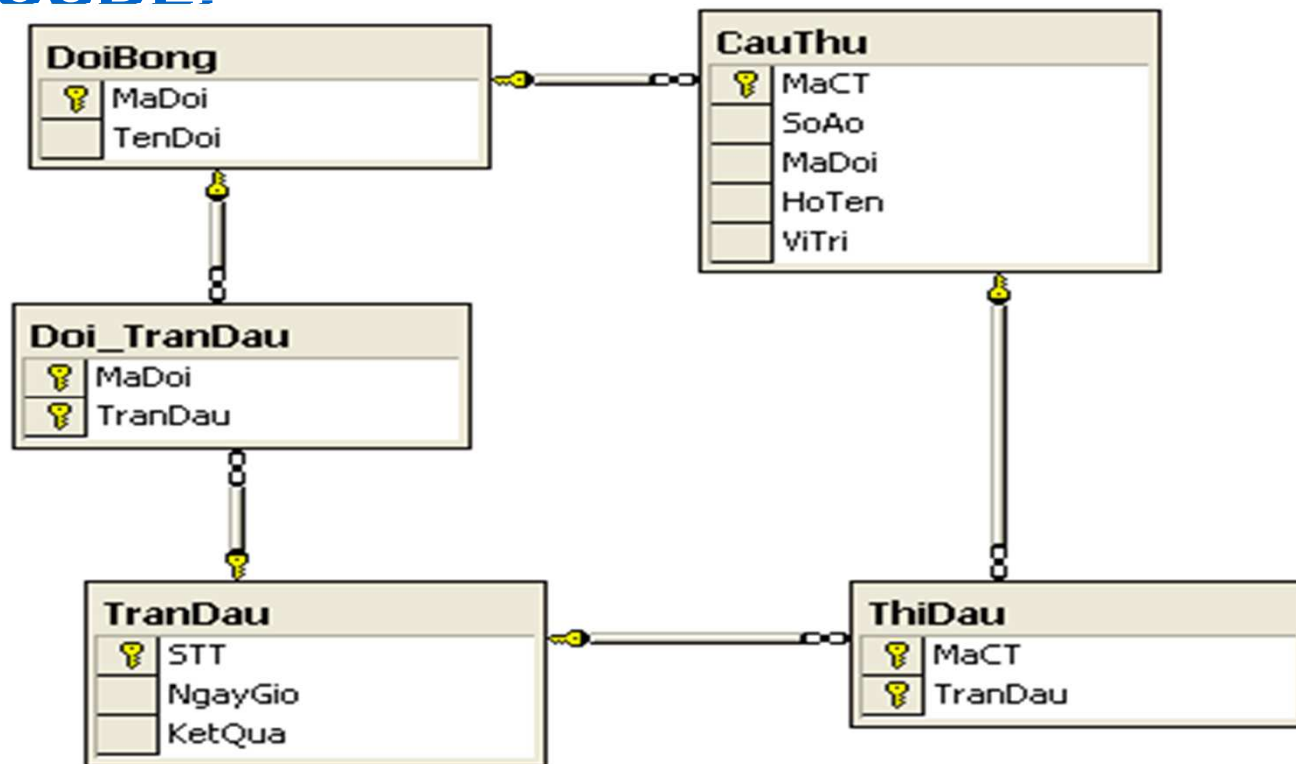
```
if update(MaPDH) or update (NgayGH)
    if exists(select * from inserted i, DatHang d
        where i.MaPDH = d.MaPDH
            and i.NgayGH<d.NgayDH)
        begin
            raiserror (N'Ngày GH không thể nhỏ hơn ngày ĐH,15,1)
            rollback tran
        end
    end
go
```

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**