



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THỐNG THÔNG TIN

MÔN CƠ SỞ DỮ LIỆU - IT004

CHƯƠNG 5 (tt): TRIGGER

ThS. TẠ VIỆT PHƯƠNG
phuongtv@uit.edu.vn

Khối lệnh

Khối lệnh T-SQL (T-SQL Block) là một nhóm các câu lệnh T-SQL được đặt trong cặp từ khóa BEGIN và END. Khối lệnh cho phép nhóm các câu lệnh có liên quan về mặt logic để thực hiện một tác vụ cụ thể.

Đặc điểm:

- **Phân tách logic:** Giúp tổ chức và phân chia code thành các phần logic riêng biệt, giúp code dễ đọc và bảo trì hơn.
- **Phạm vi biến:** Biến được khai báo bên trong khối lệnh chỉ có hiệu lực trong phạm vi khối lệnh đó.
- **Xử lý luồng:** Cho phép sử dụng các cấu trúc điều khiển (IF-ELSE, WHILE, CASE) để điều khiển luồng thực thi các câu lệnh.
- **Xử lý lỗi:** Cho phép sử dụng khối TRY...CATCH để xử lý lỗi phát sinh trong khối lệnh.

Khối lệnh

Ví dụ:

```
DECLARE @SoLuongTon INT;  
SELECT @SoLuongTon = SoLuongTon FROM SanPham WHERE MaSP = 1;  
IF @SoLuongTon > 0  
BEGIN  
    UPDATE SanPham SET SoLuongTon = SoLuongTon - 1 WHERE MaSP = 1;  
    PRINT 'Đã bán 1 sản phẩm.';  
END  
ELSE  
BEGIN  
    PRINT 'Sản phẩm đã hết hàng.';  
END;
```

Biến

Biến là một đối tượng có thể lưu giữ một giá trị dữ liệu. Dữ liệu có thể được chuyển đến câu lệnh T-SQL bằng việc sử dụng biến cục bộ. Biến có thể được phân thành 2 loại: biến cục bộ và toàn cục:

Biến

Biến cục bộ:

- Trong T-SQL biến cục bộ được tạo và được sử dụng cho việc lưu trữ tạm thời trong khi câu lệnh SQL được thực hiện. Tên của biến cục bộ phải bắt đầu với dấu '@'

Declare tên_biến Kiểu_dữ_liệu

tên_biến: xác định tên của biến, tên của biến phải bắt đầu với 1 dấu '@'.

Kiểu_dữ_liệu: là kiểu dữ liệu được định nghĩa bởi người sử dụng hoặc hệ thống.

Ví dụ:

Declare @MaSinhVien char(10)

Declare @HoTen nvarchar(30)

Declare @Sum float, @Count int

Biến

Câu lệnh SET hoặc SELECT được sử dụng để gán giá trị đến cho biến cục bộ

```
SET tên_biến = giá_trị  
SET tên_biến = tên_biến  
SET tên_biến = biểu_thức  
SET tên_biến = kết_quả_truy_vấn
```

Ví dụ:

```
Declare @STT int
```

```
SET @STT = 1
```

```
SELECT HoTen FROM SinhVien
```

```
WHERE MaSinhVien=@STT
```

Ví dụ:

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

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

Biến

Đưa kết quả truy vấn vào biến:

Ví dụ: SinhVien(MaSV: int; HoTen: nvarchar(30), Tuoi int)

```
Select @Var1 = HoTen, @Var2 = Tuoi from SinhVien
```

```
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

Biến toàn cục: Biến toàn cục là biến được định nghĩa và xử lý bởi **hệ thống**. Biến toàn cục trong SQL Server được bắt đầu với 2 dấu '@'. Giá trị của các biến này có thể được truy lục bằng câu truy vấn SELECT đơn giản.

Ví dụ:

```
SELECT @@VERSION AS sqlServerVersionDetails
```

SQL tự cập nhật giá trị cho các biến này, user không thể gán giá trị trực tiếp

Bản chất là 1 hàm (function)

Một số biến thông dụng:

- @@error
- @@rowcount
- @@trancount
- @@fetch_status
- @@SERVERNAME: Trả về tên của server hiện tại.

Biến

Gán giá trị: Sử dụng toán tử = hoặc :=

Ví dụ:

```
SET @HoTen = 'Nguyễn Văn A';
```

```
SET @Tuoi = 20;
```

Sử dụng biến trong câu lệnh SQL:

```
SELECT * FROM SinhVien WHERE HoTen = @HoTen;
```

Stored procedure

Stored Procedure là đoạn chương trình kịch bản (programming scripts) với các câu lệnh SQL nhúng (embedded SQL) được lưu dưới dạng đã được biên dịch và thi hành trực tiếp bởi MS SQL server.

Stored Procedure: Đóng gói logic phức tạp

Còn gọi là: **Thủ tục “nội”, thủ tục nội tại, thủ tục thường trú**

Một stored procedure là một mã SQL đã được chuẩn bị sẵn mà có thể lưu, vì thế mã này có thể được sử dụng lại nhiều lần.

Stored procedure

Cú pháp:

```
CREATE PROCEDURE tên procedure  
{Parameter_name DataType [=default] [output] }  
AS  
Câu lệnh SQL  
GO;
```

Câu lệnh thực thi một Stored Procedure:
EXEC tên procedure;

Lưu ý:

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

Stored procedure

Ví dụ:

```
CREATE PROCEDURE DanhSachNV  
AS  
BEGIN  
    SELECT HoTen, NTNS  
    FROM NhanVien  
    ORDER BY HoTen;  
END;
```

Stored procedure

Ví dụ:

```
CREATE PROCEDURE DanhSachNV1 @max_luong INT
AS
BEGIN
    SELECT HoTen, NTNS, Luong
    FROM NhanVien WHERE Luong< @max_luong
    ORDER BY HoTen;
END;
```

Stored procedure

Ví dụ:

```
CREATE PROCEDURE usp_TinhTong
@So1 INT,          -- Tham số input
@So2 INT,          -- Tham số input
@Tong INT OUTPUT   -- Tham số output
AS
BEGIN
    SET @Tong = @So1 + @So2;
END;
```

Tham số input: Dùng để truyền dữ liệu vào stored procedure. Mặc định, mọi tham số được khai báo đều là tham số input.

Tham số output: Dùng để stored procedure trả về dữ liệu cho chương trình gọi nó. Cần thêm từ khóa OUTPUT sau kiểu dữ liệu.

Review Chương 4, phần 2.2

- ❖ **Ràng buộc toàn vẹn trong SQL được chia làm 2 loại chính:**
 - **Loại đơn giản:** Sử dụng **Constraint** để mô tả
 - **Loại phức tạp:** Sử dụng **Trigger** để thực hiện
- ❖ Có thể khai báo RBTV ở mức cột hoặc mức bảng

Trigger

Trigger: Tự động hóa tác vụ

Trigger là một loại stored procedure đặc biệt được thực thi (execute) một cách tự động khi có một sự kiện thay đổi dữ liệu (data modification) xảy ra như Update, Insert hoặc Delete.

Trigger được dùng để đảm bảo tính toàn vẹn dữ liệu (Data Integrity) hoặc thực hiện các quy tắc nghiệp vụ (business rules) nào đó.

Trigger

Phân loại Trigger

Nội dung:

- DML trigger: Kích hoạt bởi các sự kiện DML (INSERT, UPDATE, DELETE).
 - After trigger: Thực thi sau khi thao tác DML hoàn thành. Có thể quay lui thao tác đã thực hiện bằng lệnh rollback transaction.
 - Instead of trigger: Thay thế cho thao tác DML, thường được dùng để xử lý cập nhật trên view
- DDL trigger: Kích hoạt bởi các sự kiện DDL (CREATE, ALTER, DROP).
- Logon trigger: Kích hoạt khi người dùng đăng nhập vào SQL Server hoặc các sự kiện đặc biệt (LOGON, LOGOFF, STARTUP, SHUTDOWN)

Trigger

Trigger và Constraint

- Trigger chỉ áp dụng cho dữ liệu mới. Constraint có thể áp dụng cho dữ liệu cũ và dữ liệu mới.
- Trigger có thể tuân theo các quy tắc phức tạp mà Constraint không thể.

Trigger

Khi trigger được thực thi, SQL tự động tạo ra 2 bảng tạm với cùng cấu trúc với bảng mà trigger được định nghĩa trên đó.

Bảng INSERTED chứa dữ liệu mới khi thực thi câu lệnh Insert hoặc câu lệnh Update.

Bảng DELETED chứa dữ liệu bị xóa khi thực thi câu lệnh Delete hoặc chứa dữ liệu cũ (old) khi thực thi câu lệnh Update.

Hai bảng này chỉ tồn tại trong thời gian trigger xử lý và cục bộ cho mỗi trigger.

Hoạt động	Bảng INSERTED	Bảng DELETED
INSERT	dữ liệu mới được insert	không có dữ liệu
DELETE	không có dữ liệu	chứa dữ liệu bị xóa
UPDATE	chứa dữ liệu sau khi được cập nhật	chứa dữ liệu trước khi cập nhật

Trigger

Đối với thao tác insert:

Insert into HOADON values (1004, '01/09/2006', 'KH02', 180000)

HOADON			
SOHD	NGHD	MAKH	TRIGIA
1001	23/07/2006	KH01	320,000
1002	12/08/2006	KH01	840,000
1003	23/08/2006	KH02	100,000
1004	01/09/2006	KH02	180,000

INSERTED			
SOHD	NGHD	MAKH	TRIGIA
1004	01/09/2006	KH02	180,000

DELETED			
SOHD	NGHD	MAKH	TRIGIA

1004	01/09/2006	KH02	180,000
------	------------	------	---------

Trigger

Đối với thao tác delete

Delete from HOADON where soh=1004

HOADON			
SOHD	NGHD	MAKH	TRIGIA
1001	23/07/2006	KH01	320,000
1002	12/08/2006	KH01	840,000
1003	23/08/2006	KH02	100,000
1004	01/09/2006	KH02	180,000

1004	01/09/2006	KH02	180,000
------	------------	------	---------

INSERTED			
SOHD	NGHD	MAKH	TRIGIA

DELETED			
SOHD	NGHD	MAKH	TRIGIA
1004	01/09/2006	KH02	180,000

Trigger

Đối với thao tác update:

Update HOADON set makh='kh07', trigia=300000 Where sohd=1004

HOADON			
SOHD	NGHD	MAKH	TRIGIA
1001	23/07/2006	KH01	320,000
1002	12/08/2006	KH01	840,000
1003	23/08/2006	KH02	100,000
1004	01/09/2006	KH07	300,000

INSERTED			
SOHD	NGHD	MAKH	TRIGIA
1004	01/09/2006	KH07	300,000
DELETED			
SOHD	NGHD	MAKH	TRIGIA
1004	01/09/2006	KH02	180,000

Trigger

Tạo trigger: cú pháp

```
CREATE [OR ALTER] TRIGGER Tên_Trigger  
ON Tên_Table  
AFTER (FOR) | INSTEAD OF INSERT, DELETE, UPDATE  
AS
```

Các _lệnh_của_Trigger

AFTER | FOR | INSTEAD OF: Xác định thời điểm trigger được kích hoạt.

Xóa trigger: cú pháp

```
DROP TRIGGER Tên_Trigger
```

Trigger

Có 2 loại triggers: INSTEAD OF và AFTER (FOR).

INSTEAD OF:

- Trigger được gọi thực hiện thay cho thao tác delete/insert/update tương ứng.
- Trigger instead of thường được dùng để xử lý cập nhật trên view.

AFTER (FOR):

- Trigger được gọi thực hiện sau khi thao tác delete/ insert/ update tương ứng đã được thực hiện thành công.
- Có thể quay lui thao tác đã thực hiện bằng lệnh rollback transaction.

Trigger

Ví dụ **Instead Of Trigger**

-- Tạo view chỉ hiển thị sản phẩm có giá lớn hơn 100

```
CREATE VIEW v_SanPhamGiaCao
```

```
AS
```

```
SELECT * FROM SanPham WHERE Gia > 100;
```

-- Tạo **INSTEAD OF TRIGGER** cho view

```
CREATE TRIGGER trg_v_SanPhamGiaCao_Insert
```

```
ON v_SanPhamGiaCao
```

```
INSTEAD OF INSERT
```

```
AS
```

```
BEGIN
```

Trigger

-- Kiểm tra giá sản phẩm trong bảng ảo inserted

```
IF EXISTS (SELECT 1 FROM inserted WHERE Gia <= 100)
```

```
BEGIN
```

```
    RAISERROR('Giá sản phẩm phải lớn hơn 100!', 16, 1);
```

```
    ROLLBACK TRANSACTION;
```

```
END
```

```
ELSE
```

```
BEGIN
```

```
    -- Thêm sản phẩm vào bảng SanPham
```

```
    INSERT INTO SanPham (MaSP, TenSP, Gia)
```

```
    SELECT MaSP, TenSP, Gia FROM inserted;
```

```
END
```

```
END;
```

Trigger

Trigger `trg_v_SanPhamGiaCao_Insert` kích hoạt thay thế cho thao tác `INSERT` trên view `v_SanPhamGiaCao`.

Trigger kiểm tra giá sản phẩm, nếu giá nhỏ hơn hoặc bằng 100, trigger sẽ báo lỗi và rollback.

Nếu giá lớn hơn 100, trigger sẽ chèn dữ liệu vào bảng gốc `SanPham`.

Trigger

KHACHHANG (MAKH, HOTEN, NGSINH, NGDK)

HOADON (SOHD, NGHD, MAKH, TRIGIA)

Ngày mua hàng (NGHD) của một khách hàng thành viên sẽ lớn hơn hoặc bằng ngày khách hàng đó đăng ký thành viên (NGDK).

	THÊM	XÓA	SỬA
KHACHHANG	-	-	+ (NGDK)
HOADON	+	-	+ (NGHD, MAKH)

Trigger

Sinh viên tìm hiểu các cấu trúc sau:

- + Khai báo biến: DECLARE
- + Xuất thông tin: PRINT, RAISERROR
- + Cấu trúc điều kiện: IF
- + Cấu trúc lặp: FOR
- + Con trỏ: CURSOR
- + Hủy cập nhật dữ liệu vào hệ thống bộ nhớ: ROLLBACK
TRAN

Trigger

```
CREATE TRIGGER nghd_hoadon_insert
ON hoadon
AFTER INSERT
AS
    DECLARE @ng_muahang smalldatetime
    DECLARE @ng_dangky smalldatetime
    SELECT @ng_muahang=nghd, @ng_dangky=ngdk
    FROM khachhang, inserted
    WHERE khachhang.makh=inserted.makh
    IF @ng_muahang< @ng_dangky
    BEGIN
        rollback transaction
        print 'ngay mua hang phai lon hon ngay dang ky'
    END
```

Trigger

HOẶC:

```
CREATE TRIGGER nghd_hoadon_insert
ON hoadon
AFTER INSERT
AS
    IF (EXISTS (SELECT *
                FROM inserted i JOIN khachhang kh ON i.makh=kh.makh
                WHERE i.nghd<kh.ngdk))
BEGIN
    rollback transaction
    print 'ngày mua hàng phải lớn hơn ngày đăng ký'
END
```

Trigger

```
CREATE TRIGGER nghd_hoadon_update
ON hoadon
AFTER UPDATE
AS
IF (UPDATE (makh) OR UPDATE (nghd))
BEGIN
    DECLARE @ng_muahang smalldatetime
    DECLARE @ng_dangky smalldatetime
    SELECT @ng_muahang=nghd, @ng_dangky=ngdk
    FROM khachhang, inserted
    WHERE khachhang.makh=inserted.makh
    IF @ng_muahang< @ng_dangky
    BEGIN
        rollback transaction
        print 'ngay mua hang phai lon hon ngay dang ky'
    END
END
END
```


Trigger

HOẶC:

```
CREATE TRIGGER nghd_hoadon_update
ON hoadon
AFTER UPDATE
AS
IF (UPDATE (makh) OR UPDATE (nghd)
BEGIN
    IF (EXISTS (SELECT *
                FROM inserted i JOIN khachhang kh ON i.makh=kh.makh
                WHERE i.nghd<kh.ngdk))
    BEGIN
        rollback transaction
        print 'ngày mua hàng phải lớn hơn ngày đang ký'
    END
END
END
```

Trigger

```
CREATE TRIGGER nghd_khachhang_update  
ON KHACHHANG  
AFTER UPDATE  
AS
```

```
    DECLARE @ng_dangky smalldatetime, @makhang char(4)  
    SELECT @ng_dangky=ngdk, @makhang=makh  
    FROM inserted  
    IF (UPDATE (ngdk))  
    BEGIN  
    IF (EXISTS (SELECT *  
                FROM hoadon  
                WHERE makh=@makhang AND nghd<@ng_dangky))  
    BEGIN  
        rollback transaction  
    END  
    END
```

Trigger

HOẶC:

```
CREATE TRIGGER nghd_khachhang_update
ON KHACHHANG
AFTER UPDATE
AS
    IF (UPDATE (ngdk))
    BEGIN
        IF (EXISTS (SELECT *
                    FROM hoadon hd JOIN inserted i ON hd.makh=i.makh
                    WHERE hd.nghd<kh.ngdk))
        BEGIN
            rollback transaction
        END
    END;
END;
```

Trigger

Trường hợp thì chỉ muốn ngưng kiểm tra trigger thì ta sử dụng lệnh sau:

```
ALTER TABLE Tên_bảng  
DISABLE/ENABLE TRIGGER Tên_trigger (/ALL)
```

Trigger

Sử dụng quá mức trigger có thể ảnh hưởng đến hiệu suất hệ thống. **Ví dụ:** Nếu một bảng có trigger thực hiện nhiều kiểm tra phức tạp sau mỗi lần **INSERT**, việc thêm dữ liệu vào bảng này sẽ chậm hơn so với bảng không có trigger.

Không nên sử dụng trigger cho các thao tác xảy ra thường xuyên với dữ liệu lớn vì nó sẽ làm giảm hiệu suất tổng thể. Khuyến khích chỉ sử dụng trigger khi cần thiết. Các giải pháp thay thế có thể là kiểm tra tính toàn vẹn của dữ liệu bằng CHECK Constraints hoặc Stored Procedures.

Lược đồ CSDL quản lý giáo vụ

HOCVIEN (MAHV, HO, TEN, NGSINH, GIOITINH, NOISINH, CMND, MALOP)

LOP (MALOP, TENLOP, TRGLOP, SISO, MAGVCN)

KHOA (MAKHOA, TENKHOA, NGTLAP, TRGKHOA)

MONHOC (MAMH, TENMH, TCLT, TCTH, MAKHOA)

DIEUKIEN (MAMH, MAMH_TRUOC)


GIAOVIEN (MAGV, HOTEN, HOCVI, HOCHAM, GIOITINH, NGSINH, NGVL, HESO, MUCLUONG, MAKHOA)

GIANGDAY (MALOP, MAMH, MAGV, HOCKY, NAM, TUNGAY, DENNGAY)

KETQUATHI (MAHV, MAMH, LANTHI, NGTHI, DIEM, KQUA)

Bài tập

1. Trưởng lớp của một lớp phải là học viên của lớp đó.
2. Học viên chỉ được thi lại một môn (lần thi >1) khi điểm của lần thi trước đó dưới 5.
3. Ngày thi của lần thi sau phải lớn hơn ngày thi của lần thi trước (cùng học viên, cùng môn học).
4. Mỗi năm học chỉ được giảng dạy tối đa 10 lớp
5. Mỗi học kỳ của một năm học, một lớp chỉ được dạy tối đa 3 môn.



THANK YOU!

Q & A

ThS. TẠ VIỆT PHƯƠNG
phuongtv@uit.edu.vn