TÀI LIỆU HƯỚNG DẪN THỰC HÀNH MÔN HỌC
IT004 - CƠ SỞ DỮ LIỆU (DATABASE)

NỘI DUNG THỰC HÀNH TUẦN 5

Hướng dẫn thực hành

Lê Võ Đình Kha - khalvd@uit.edu.vn

GIỚI THIỆU NỘI DUNG THỰC HÀNH TUẦN 5

NỘI DUNG

1. Tổng quan về Trigger trong SQL.
2. Cú pháp và cấu trúc Trigger.
3. Bảng Inserted và Deleted trong Trigger.
4. Ví dụ tham khảo.
5. Bài tập thực hành và hỏi đáp.

1

TỔNG QUAN VỀ TRIGGER TRONG SQL

TỔNG QUAN VỀ TRIGGER TRONG SQL

Phân loại ràng buộc toàn vẹn trong SQL

- **Việc ràng buộc toàn vẹn trong SQL Server được chia làm 2 loại chính:**
  - **Ràng buộc đơn giản (Simple constraints):** Sử dụng CONSTRAINT để mô tả các quy tắc toàn vẹn.
  - **Ràng buộc phức tạp (Complex constraints):** Sử dụng **TRIGGER** để thực hiện các quy tắc toàn vẹn phức tạp hơn mà CONSTRAINT không hỗ trợ.

TỔNG QUAN VỀ TRIGGER TRONG SQL

Các ràng buộc đơn giản thường sử dụng trong SQL Server

- **PRIMARY KEY:** Ràng buộc khóa chính, kết hợp giữa **NOT NULL** và **UNIQUE**. Xác định duy nhất mỗi bản ghi trong bảng.
- **FOREIGN KEY**: Thiết lập mối quan hệ giữa các bảng, ràng buộc khóa ngoại.
- **NOT NULL:** Đảm bảo dữ liệu trong cột không được để trống.
- **UNIQUE:** Đảm bảo dữ liệu trong cột là duy nhất, không bị trùng lặp.
- **CHECK:** Kiểm tra dữ liệu nhập vào phải theo định dạng hoặc điều kiện nhất định.
- **DEFAULT**: Đặt giá trị mặc định cho cột nếu không có giá trị được cung cấp.

RÀNG BUỘC TOÀN VẸN TRONG SQL

Các ràng buộc phức tạp thường sử dụng trong SQL Server

Loại ràng buộc phức tạp	Mô tả
Ràng buộc tham chiếu nhiều quan hệ	Đảm bảo tính toàn vẹn tham chiếu giữa nhiều bảng có liên quan.
Ràng buộc liên bộ, liên quan hệ	Đảm bảo mối quan hệ giữa các bảng không chỉ trên một thuộc tính mà có thể trên nhiều thuộc tính.
Ràng buộc liên thuộc tính	Kiểm tra ràng buộc dựa trên mối quan hệ giữa các thuộc tính khác nhau trong bảng.
Ràng buộc thuộc tính tổng hợp	Đảm bảo giá trị tổng hợp (như tổng, trung bình) thỏa mãn một điều kiện cụ thể.
Ràng buộc chu trình (Cyclic Constraints)	Đảm bảo không có chu trình trong cấu trúc tham chiếu của dữ liệu.

TỔNG QUAN VỀ TRIGGER TRONG SQL

Định nghĩa về Trigger

- **Trigger** là một tập hợp các câu lệnh SQL được lưu trữ và tự động thực thi khi một sự kiện thay đổi dữ liệu xảy ra trên một bảng hoặc một view.
- Trigger thường được dùng để thực hiện các kiểm tra ràng buộc phức tạp, bảo toàn dữ liệu và tự động hóa các hành động trong cơ sở dữ liệu.
- **Trigger được liên kết với các thao tác DML như:**
  - **INSERT:** Thêm dữ liệu vào bảng.
  - **UPDATE:** Cập nhật dữ liệu trên bảng.
  - **DELETE:** Xóa dữ liệu khỏi bảng.

TỔNG QUAN VỀ TRIGGER TRONG SQL

Các loại Trigger trong SQL

- SQL Server hỗ trợ hai loại Trigger chính: **AFTER Trigger và INSTEAD OF Trigger**. Trong các hệ quản trị khác như MySQL và PostgreSQL, có thêm loại **BEFORE Trigger.**

Loại Trigger	Mô tả	Thời điểm kích hoạt
AFTER Trigger	Được kích hoạt **sau** khi thao tác INSERT, UPDATE, DELETE đã hoàn tất và không có lỗi xảy ra.	Sau khi thực hiện thành công thao tác DML.
INSTEAD OF Trigger	**Thay thế** thao tác INSERT, UPDATE, DELETE bằng một hành động khác được định nghĩa trong Trigger.	Thay vì thực hiện thao tác gốc, trigger sẽ được kích hoạt.
BEFORE Trigger (*)	Được kích hoạt trước khi thao tác INSERT, UPDATE, DELETE diễn ra, để kiểm tra hoặc ngăn chặn.	Trước khi thao tác chính được thực hiện.

** BEFORE Trigger **không hỗ trợ** (trong SQL Server)*

2

CÚ PHÁP VÀ CẤU TRÚC TRIGGER

CÚ PHÁP VÀ CẤU TRÚC TRIGGER

Cú pháp tạo Trigger

- Cú pháp chung để tạo một trigger:

```
CREATE TRIGGER <Tên trigger>
ON <Tên bảng>
AFTER | INSTEAD OF (INSERT), (UPDATE), (DELETE)
AS
BEGIN
        ---Nội dung thực thi/câu lệnh
END;
```

** Có thể dùng **FOR** thay thế cho **AFTER**, vì **FOR** và **AFTER** hoạt động giống nhau*

CÚ PHÁP VÀ CẤU TRÚC TRIGGER

Giải thích các từ khóa trong cấu trúc tạo Trigger

Từ khóa	Ý nghĩa
CREATE TRIGGER	Dùng để khai báo một trigger mới.
<Tên Trigger>	Tên của trigger, thường đặt theo quy tắc dễ hiểu để nhận diện.
ON <Tên bảng>	Chỉ định bảng mà trigger sẽ được gắn vào và giám sát các thao tác trên bảng này.
AFTER/FOR	Trigger sẽ được kích hoạt **sau khi** thao tác INSERT, UPDATE, DELETE hoàn tất.
INSTEAD OF	Trigger sẽ **thay thế** thao tác INSERT, UPDATE, DELETE, thường dùng trên view hoặc bảng phức tạp.
INSERT, UPDATE, DELETE	Các thao tác sẽ kích hoạt trigger, có thể chọn một hoặc nhiều thao tác.
AS	Bắt đầu phần thân của trigger, nơi định nghĩa các lệnh sẽ được thực thi khi trigger được kích hoạt.
BEGIN...END	Khối lệnh thực hiện khi trigger được kích hoạt. Có thể chứa các câu lệnh kiểm tra và thao tác SQL.

CÚ PHÁP VÀ CẤU TRÚC TRIGGER

Ví dụ 1: Trigger thông báo khi thêm mới khách hàng

```
CREATE TRIGGER trg_AfterInsert_KhachHang
ON KHACHHANG
AFTER INSERT
AS
BEGIN
    PRINT 'Khach hang moi da duoc them vao KHACHHANG.'
END;
```

CÚ PHÁP VÀ CẤU TRÚC TRIGGER

Ví dụ 2: Trigger ràng buộc cấm xóa khách hàng

```
CREATE TRIGGER trg_PreventDelete_KhachHang
ON KHACHHANG
INSTEAD OF  DELETE
AS
BEGIN
      PRINT 'Khong duoc xoa khach hang.'
END;
```

CÚ PHÁP VÀ CẤU TRÚC TRIGGER

Ví dụ 3: Số lượng sản phẩm trong bảng **SANPHAM** không được nhỏ hơn 0.

```
CREATE TRIGGER trg_CheckQuantity_Sanpham
ON SANPHAM
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (SELECT * FROM SANPHAM WHERE SOLUONG < 0)
    BEGIN
        RAISERROR ('Số lượng không thể âm.', 16, 1);
        ROLLBACK;
    END
END;
```

CÚ PHÁP VÀ CẤU TRÚC TRIGGER

So sánh PRINT và RAISERROR

Tiêu chí	PRINT	RAISERROR
Mục đích	Hiển thị thông báo	Hiển thị lỗi và có thể dừng quá trình
Ảnh hưởng	Không ảnh hưởng đến quá trình	Có thể dừng và **ROLLBACK** thao tác
Mức độ nghiêm trọng	Không có mức độ nghiêm trọng	Có thể đặt từ 0 đến 25
Ứng dụng	Gỡ lỗi, thông báo thông tin	Kiểm tra ràng buộc, thông báo lỗi

- **Dùng PRINT** khi chỉ muốn thông báo thông tin cho người dùng mà không cần dừng quá trình.
- **Dùng RAISERROR** khi cần báo lỗi và ngăn chặn thao tác sai.

CÚ PHÁP VÀ CẤU TRÚC TRIGGER

Mức độ nghiêm trọng (Severity Level) trong RAISERROR

- **Dưới đây là ý nghĩa của một số mức độ thường gặp:**

Mức độ nghiêm trọng	Ý nghĩa
0 - 10	Thông báo thông tin hoặc cảnh báo nhẹ, không phải là lỗi nghiêm trọng.
11 - 16	Lỗi do người dùng gây ra (User Error). Người dùng có thể khắc phục.
17 - 19	Lỗi hệ thống (System Error), có thể cần can thiệp của quản trị viên.
20 - 25	Lỗi nghiêm trọng (Fatal Error), thường dẫn đến ngắt kết nối hoặc dừng hệ thống.

- Mức độ 16 là mức thường dùng nhất trong các Trigger hoặc thủ tục, chỉ ra rằng đây là lỗi do người dùng gây ra, chẳng hạn như nhập dữ liệu sai hoặc vi phạm quy tắc.

CÚ PHÁP VÀ CẤU TRÚC TRIGGER

Các câu lệnh kiểm tra và quản lý Trigger trong SQL

Tên câu lệnh	Cú pháp tổng quát
Hiển thị Trigger	**SHOW TRIGGERS;**
Tạo Trigger	**CREATE TRIGGER** <...>
Sửa Trigger	**ALTER TRIGGER** <Tên Trigger> ...
Xóa Trigger	**DROP TRIGGER** <Tên Trigger>;
Tắt Trigger	**DISABLE TRIGGER** <Tên Trigger> **ON** <Tên bảng>;
Bật Trigger	**ENABLE TRIGGER** <Tên Trigger> **ON** <Tên bảng>;
Hiển thị Trigger trong thông tin	**SELECT * FROM INFORMATION_SCHEMA.TRIGGERS;**
Hiển thị cấu trúc Trigger	**SHOW CREATE TRIGGER** <Tên Trigger>;

3

BẢNG INSERTED VÀ DELETED TRONG TRIGGER

BẢNG INSERTED VÀ DELETED TRONG TRIGGER

Khái niệm về bảng Inserted và Deleted

- Khi thực hiện một Trigger trong SQL Server, hai bảng đặc biệt là **INSERTED** và **DELETED** được sử dụng để lưu trữ các bản ghi dữ liệu trước và sau khi thực hiện các thao tác INSERT, UPDATE, hoặc DELETE.
- Cả hai bảng này là các **bảng ảo**, tức là chúng không thực sự lưu trữ dữ liệu trong cơ sở dữ liệu mà chỉ tồn tại trong phạm vi của Trigger.

BẢNG INSERTED VÀ DELETED TRONG TRIGGER

Khái niệm về bảng Inserted và Deleted

Bảng	Mô tả	Thao tác	Dữ liệu chứa trong bảng
INSERTED	Chứa dữ liệu mới được thêm vào bảng hoặc dữ liệu đã bị cập nhật.	**INSERT**	Các bản ghi mới được thêm vào bảng.
		UPDATE	Các giá trị mới của các bản ghi đã được cập nhật.
DELETED	Chứa dữ liệu trước khi bị xóa hoặc cập nhật.	**DELETE**	Các bản ghi đã bị xóa khỏi bảng.
		UPDATE	Các giá trị cũ của các bản ghi trước khi cập nhật.

BẢNG INSERTED VÀ DELETED TRONG TRIGGER

Sử dụng bảng INSERTED cho thao tác INSERT

- Khi thêm dữ liệu vào bảng, Trigger sẽ lấy các bản ghi mới từ bảng INSERTED.

```
CREATE TRIGGER trg_AfterInsert_KhachHang
ON KHACHHANG
AFTER INSERT
AS
BEGIN
    -- Lấy dữ liệu mới từ bảng Inserted và thực hiện hành động
    SELECT * FROM INSERTED;
END;
```

BẢNG INSERTED VÀ DELETED TRONG TRIGGER

Sử dụng bảng DELETED cho thao tác DELETE

- Khi một bản ghi bị xóa, Trigger sẽ lấy các bản ghi cũ từ bảng DELETED.

```
CREATE TRIGGER trg_AfterDelete_KhachHang
ON KHACHHANG
AFTER DELETE
AS
BEGIN
    -- Lấy dữ liệu bị xóa từ bảng Deleted và thực hiện hành động
    SELECT * FROM DELETED;
END;
```

BẢNG INSERTED VÀ DELETED TRONG TRIGGER

Sử dụng kết hợp INSERTED và DELETED cho thao tác UPDATE

- Đối với thao tác UPDATE, Trigger sẽ sử dụng cả bảng INSERTED (dữ liệu mới) và bảng DELETED (dữ liệu cũ) để so sánh và xử lý.

```
CREATE TRIGGER trg_AfterUpdate_KhachHang
ON KHACHHANG
AFTER UPDATE
AS
BEGIN
    -- Lấy dữ liệu cũ từ Deleted và dữ liệu mới từ Inserted
    SELECT * FROM INSERTED;
    SELECT * FROM DELETED;
END;
```

BẢNG INSERTED VÀ DELETED TRONG TRIGGER

Ví dụ sử dụng bảng INSERTED cho thao tác INSERT

- Ví dụ 4: Tự động thông báo khi có khách hàng mới được thêm vào.

```
CREATE TRIGGER trg_AfterInsert_KhachHang
ON KHACHHANG
AFTER INSERT
AS
BEGIN
    DECLARE @MAKH char(4);
    SELECT @MAKH = MAKH FROM INSERTED;
    PRINT 'Khách hàng mới được thêm vào với MAKH: ' + CAST(@MAKH AS VARCHAR);
END;
```

BẢNG INSERTED VÀ DELETED TRONG TRIGGER

Ví dụ sử dụng bảng DELETED cho thao tác DELETE

- Ví dụ 5: Ghi lại thông tin khách hàng bị xóa khỏi bảng KhachHang.

```
CREATE TRIGGER trg_AfterDelete_KhachHang
ON KHACHHANG
INSTEAD OF DELETE
AS
BEGIN
    DECLARE @MAKH char(4);
    SELECT @MAKH = MAKH FROM DELETED;
    PRINT 'Khách hàng với MAKH: ' + CAST(@MAKH AS VARCHAR) + ' đã bị xóa.';
END;
```

BẢNG INSERTED VÀ DELETED TRONG TRIGGER

Ví dụ sử dụng cả bảng INSERTED và DELETED cho thao tác UPDATE

- Ví dụ 6: Theo dõi khi có sự thay đổi tên của khách hàng trong bảng.

```
CREATE TRIGGER trg_AfterUpdate_KhachHang
ON KHACHHANG
AFTER UPDATE
AS
BEGIN
    DECLARE @OldName VARCHAR(40), @NewName VARCHAR(40);
    SELECT @OldName = HOTEN FROM DELETED;
    SELECT @NewName = HOTEN FROM INSERTED;
    PRINT 'Tên khách hàng đã thay đổi từ "' + @OldName + '" sang "' + @NewName + '".';
END;
```

4

VÍ DỤ THAM KHẢO VỀ TRIGGER

VÍ DỤ THAM KHẢO VỀ TRIGGER

Ví dụ 7: Kiểm tra giá trị doanh số của khách hàng không âm

```
CREATE TRIGGER trg_check_doanhso_KhachHang
ON KHACHHANG
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (SELECT * FROM inserted WHERE DOANHSO < 0)
    BEGIN
        RAISERROR('Doanh số không thể nhỏ hơn 0!', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
```

VÍ DỤ THAM KHẢO VỀ TRIGGER

Ví dụ 8: Khi thêm một chi tiết hóa đơn (CTHD), tự động cập nhật TRIGIA của hóa đơn tương ứng trong bảng HOADON.

```
CREATE TRIGGER trg_update_trigia_CTHD
ON CTHD
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    DECLARE @SOHD INT;

    SELECT @SOHD = SOHD FROM inserted;

    UPDATE HOADON
    SET TRIGIA = (SELECT SUM(C.SL * S.GIA)
                  FROM CTHD C
                  JOIN SANPHAM S ON C.MASP = S.MASP
                  WHERE C.SOHD = @SOHD)
    WHERE SOHD = @SOHD;
END;
```

VÍ DỤ THAM KHẢO VỀ TRIGGER

Ví dụ 9: Không cho phép thêm hóa đơn nếu mã khách hàng (MAKH) không tồn tại trong bảng KHACHHANG (hay khách hàng chưa đăng ký).

```
CREATE TRIGGER trg_check_makh_HoaDon
ON HOADON
AFTER INSERT
AS
BEGIN
    IF EXISTS (SELECT * FROM inserted
                WHERE MAKH IS NOT NULL
                        AND MAKH NOT IN (SELECT MAKH
                                         FROM KHACHHANG))
    BEGIN
        RAISERROR('Khách hàng không tồn tại!', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
```

VÍ DỤ THAM KHẢO VỀ TRIGGER

Ví dụ 10: Không cho phép xóa nhân viên nếu nhân viên đó đã tham gia bán hàng (có trong bảng HOADON).

```
CREATE TRIGGER trg_prevent_delete_nhanvien
ON NHANVIEN
INSTEAD OF DELETE
AS
BEGIN
    IF EXISTS (SELECT * FROM deleted WHERE MANV IN (SELECT MANV FROM HOADON))
    BEGIN
        RAISERROR('Không thể xóa nhân viên đã có hóa đơn!', 16, 1);
        RETURN;
    END

    DELETE FROM NHANVIEN WHERE MANV IN (SELECT MANV FROM deleted);
END;
```

VÍ DỤ THAM KHẢO VỀ TRIGGER

Ví dụ 11: Khi thêm khách hàng mới, nếu không nhập ngày đăng ký (NGDK), hệ thống tự động cập nhật là ngày hiện tại.

```
CREATE TRIGGER trg_set_ngdk_KhachHang
ON KHACHHANG
AFTER INSERT
AS
BEGIN
    UPDATE KHACHHANG
    SET NGDK = GETDATE()
    WHERE NGDK IS NULL
            AND MAKH IN (SELECT MAKH FROM inserted);
END;
```

VÍ DỤ THAM KHẢO VỀ TRIGGER

Ví dụ 12: Khi thêm hoặc cập nhật hóa đơn, tự động cập nhật doanh số của khách hàng.

```
CREATE TRIGGER trg_update_doanhso_HoaDon
ON HOADON
AFTER INSERT, UPDATE
AS
BEGIN
    DECLARE @MAKH CHAR(4);
    SELECT @MAKH = MAKH FROM inserted;
    UPDATE KHACHHANG
    SET DOANHSO = (SELECT SUM(TRIGIA) FROM HOADON WHERE MAKH = @MAKH)
    WHERE MAKH = @MAKH;
END;
```

NHẬN XÉT VỀ PHƯƠNG PHÁP VIẾT TRIGGER

Các phương pháp dùng biến ảo, IF EXISTS, và IF NOT EXISTS.

Phương pháp	Giải thích
Biến ảo (@Variable)	Dùng để lưu trữ dữ liệu tạm thời lấy từ **bảng inserted** hoặc **deleted** trước khi thực hiện xử lý.
IF EXISTS	Dùng để kiểm tra xem có dữ liệu nào thỏa điều kiện trong bảng inserted hoặc deleted không.
IF NOT EXISTS	Dùng để kiểm tra xem không có dữ liệu nào thỏa điều kiện trong bảng inserted hoặc deleted.

NHẬN XÉT VỀ PHƯƠNG PHÁP VIẾT TRIGGER

Sự khác biệt giữa ROLLBACK, ROLLBACK TRANSACTION, và RETURN trong SQL Server

Câu lệnh	Mô tả
ROLLBACK	Hủy toàn bộ giao dịch hiện tại và đưa cơ sở dữ liệu trở lại trạng thái trước khi bắt đầu giao dịch.
ROLLBACK TRANSACTION	Hủy một giao dịch cụ thể, hoặc nếu không có tên giao dịch, sẽ hủy giao dịch hiện tại.
RETURN	Kết thúc ngay lập tức một thủ tục hoặc trigger và trả về một giá trị số nguyên (tùy chọn).

NHẬN XÉT VỀ PHƯƠNG PHÁP VIẾT TRIGGER

Khi nào nên & không nên dùng Trigger

- **Nên dùng Trigger khi:**
  - Tự động cập nhật, tính toán hoặc kiểm tra ngay trong CSDL.
  - Áp dụng quy tắc phức tạp không viết được bằng CONSTRAINT.
  - Ghi log, kiểm tra gian lận, audit.
- **Không nên dùng Trigger khi:**
  - Hệ thống yêu cầu hiệu suất cao, dữ liệu lớn.
  - Logic dễ viết hơn bằng Stored Procedure hoặc bên ứng dụng.
  - Trigger chồng lắp dễ gây lỗi logic và khó bảo trì.

5

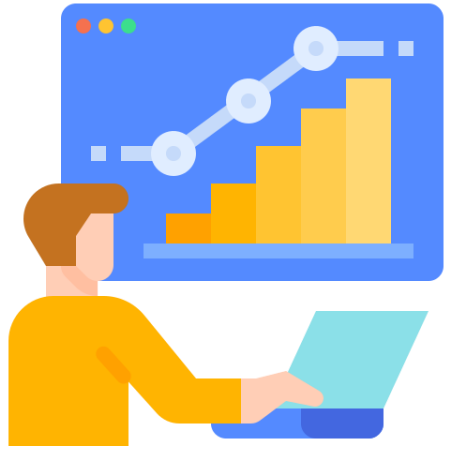
BÀI TẬP THỰC HÀNH VÀ HỎI ĐÁP

BÀI TẬP THỰC HÀNH VÀ HỎI ĐÁP

Yêu cầu: (Xem thêm đề bài tại file Bài tập thực hành tuần 5)

Phần bài làm lưu trữ trên một file script có tên **<MSSV>_<HoVaTen>_BTTH5.sql** (trong đó **MSSV** là mã số sinh viên, **HoVaTen** là họ và tên của sinh viên).

Sử dụng các câu lệnh SQL trong công cụ **SQL Server Management Studio** để thực hiện yêu cầu. Bài tập **Quản lý giáo vụ** và bài tập **Quản lý bán hàng**.



HỎI ĐÁP

Liên hệ hỗ trợ

Lê Võ Đình Kha - khalvd@uit.edu.vn