# TRIGGER

Sau bài học này, sinh viên có thể:

- Hiểu được trigger là gì, công dụng của nó
- Tạo trigger.
- Xoá trigger
- Thay đổi trigger

# Nội dung chi tiết

- **Giới thiệu**
- Tạo Trigger
- Hiệu chỉnh Trigger
- Xóa Trigger
- Ví dụ

# Giới thiệu

- TRIGGER là một Stored procedure đặc biệt, tự động được chạy mỗi khi có một hành động nào đó xảy ra có liên quan đến nó.
- Các hoạt động xảy ra giúp TRIGGER hoạt động: Insert, Delete, Update (DML). TRIGGER không được thực thi một cách tường minh nên cần thận trọng khi dùng TRIGGER.

# Giới thiệu

- TRIGGER hoạt động gần giống các hàm bắt sự kiện trong javascript. Nó chỉ được kích hoạt khi xảy ra một hành động mà cụ thể là các thao tác Insert, Delete, Update.
- Nó hoạt động một cách thụ động nên người sử dụng không thể biết trigger thực thi khi nào.

# Sử dụng trigger để làm gì?

- Đảm bảo tính ràng buộc toàn vẹn cho CSDL.
- Kiểm soát dữ liệu hiện có trong CSDL khi có thay đổi giá trị của một mẫu tin trong bảng.
- Kiểm tra dữ liệu mới nhập vào có thỏa mãn điều kiện không.
- Yêu cầu xác nhận khi xóa mẫu tin trong bảng.
- Tự động cập nhật dữ liệu cho bảng B khi dữ liệu bảng A thay đổi (khi 2 bảng có quan hệ với nhau).

# Triggers hoạt động như thế nào?

- Triggers được thực hiện tự động sau khi lệnh INSERT, UPDATE, hoặc DELETE được thực hiện trên một table mà trigger đó được định nghĩa. Còn các constraints và INSTEAD OF trigger sẽ được kiểm tra trước khi lệnh INSERT, UPDATE, hoặc DELETE thực hiện.
- **Constraints sẽ được kiểm tra trước trigger**.
- Một table có thể có nhiều Triggers cho một action. Một trigger có thể được định nghĩa cho nhiều action.
- Thứ tự thi hành sẽ là: trigger INSTEAD OF, các constraints,và sau cùng là triggerAFTER (hay FOR).

# Triggers hoạt động như thế nào?

- Khi có nhiều trigger trong một table, thì table owner có thể dùng procedure hệ thống sp_settriggerorder để chỉ định trigger đầu và trigger cuối để thực thi. Thứ tự của các trigger còn lại không thể sắp xếp được.
- User phải có quyền để thực hiện tất cả các lệnh mà được định nghĩa trong Triggers
- Table Owners không thể tạo ra các Triggers trên Views hoặc Temporary Tables nhưng có thể tham chiếu đến view và temporary.

# Triggers hoạt động như thế nào

- Triggers không trả kết quả về.
- Triggers có thể điều khiển multi-row actions: một hành động INSERT, UPDATE, hoặc DELETE gọi một trigger có thể ảnh hưởng lên nhiều dòng dữ liệu, Ta có thể chọn:
  - Xử lý tất cả các dòng cùng với  nhau trong trường hợp các dòng ảnh hưởng phải thỏa điều kiện của trigger.
  - Xử lý từng dòng thỏa điều kiện.

# Insert TRIGGER

- Được thực hiện mỗi khi mẫu tin mới được chèn vào bảng
- Một bảng tạm Inserted sẽ được sinh ra để chứa mẫu tin cần chèn

# Delete TRIGGER

- Được thực hiện mỗi khi các mẫu tin trong bảng bị xóa.
- Một bảng tạm Deleted được sinh ra để lưu các mẫu tin bị xóa.

# Update TRIGGER

- Được thực hiện khi các bản các mẫu tin của bảng được cập nhật
- Hai bảng Inserted và Deleted sẽ được sinh ra.
- Bảng Inserted sẽ lưu thông tin các mẫu tin mới được sửa, bảng Deleted sẽ lưu thông tin các mẫu tin cũ.

# Instead of TRIGGER

- Trigger cho phép cập nhật dữ liệu các bảng thông qua view có liên kết nhiều bảng
- Hoặc kiểm tra trước khi các constraint có tác dụng.

# Một số chú ý:

- Một bảng có nhiều trigger
- Mỗi một trigger có tên duy nhất
- Trong trigger thường dùng mệnh đề IF EXISTS
- Sử dụng trigger trong toàn vẹn dữ liệu
- Sử dụng trigger trong ràng buộc tham chiếu.

# Nội dung chi tiết

- Giới thiệu
- **Tạo Trigger**
- Hiệu chỉnh Trigger
- Xóa Trigger
- Ví dụ

# Tạo Trigger

```
CREATE TRIGGER trigger_name
   ON { table | view }
   [ WITH ENCRYPTION ]
   {
      {{ FOR | AFTER | INSTEAD OF }
        { [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE] }
           [ WITH APPEND ]
      AS
         [ { IF UPDATE ( column )
            [ { AND | OR } UPDATE ( column ) ]  [ ...n ]
          } ]
         sql_statement [ ...n ]
      }
   }
```

# Tham số (1)

- *Table* | *view* : tên view/table mà tigger được thực hiện khi có action tương ứng.

- WITH ENCRYPTION: mã hoá nột dung text của lệnh create trigger trong table **syscomments**.

- AFTER: Trigger sẽ được gọi chỉ khi tất cả các hành động đã thực hiện xong. Các kiểm tra constrain sẽ được kiểm tra hoàn thành trước khi trigger thực hiện. Default là AFTER nếu chỉ có từ khoá FOR được chỉ định. AFTER trigger không thể định nghĩa trên view.

- INSTEAD OF: chỉ định trigger đựoc thực hiện thay cho action của trigger. INSTEAD OF triggers không cho phép cập nhật dữ liệu trên view có WITH CHECK OPTION.

# Tham số (2)

- { [DELETE] [,] [INSERT] [,] [UPDATE] } : chỉ định action gắn với trigger. Đối với INSTEAD OF triggers, action DELETE không cho phép trên table mà có relationship mà chỉ định CASCADE ON DELETE. Tương tự, action UPDATE không cho phép trên table có relationships mà CASCADE ON UPDATE.

- Table **deleted** và **inserted** là logical tables. Chúng có cấu trúc giống với table mà trigger được định nghĩa, chứa các dòng giá trị cũ hoặc mới mà có thể thay đổi bởi action của user. Ta có truy xuất dữ liệu trong 2 table này trong định nghĩa trigger.

# Tham số (3)

- Các giá trị kiểu **text**, **ntext**, hoặc **image** trong table **inserted** và **deleted** không truy xuất được.

- IF UPDATE (*column*)**: kiểm tra action update trên cột được chỉ định, không dùng cho action delete.**

- **With Append:** Chèn thêm trigger này vào các trigger đã có trước đó

# Tạo Trigger (tt)

- Ví dụ: ràng buộc lương nhân viên phải là số dương

```
CREATE TRIGGER LUONG_DUONG ON NHANVIEN
FOR INSERT, UPDATE
AS
BEGIN
 IF (SELECT COUNT(*) FROM INSERTED WHERE LUONG < 0) > 0
 BEGIN
        Print N'Lương phải là số dương'
        Rollback Tran
 END
END
```

# Nội dung chi tiết

- Giới thiệu
- Tạo Trigger
- **Hiệu chỉnh Trigger**
- Xóa Trigger
- Ví dụ

# Hiệu chỉnh Trigger

- Cú pháp

```
ALTER TRIGGER <Ten_ Trigger> ON <Ten_bang>
FOR <[INSERT] [,] [UPDATE] [,] [DELETE]>
AS
<Cac_cau_lenh_SQL>
```

# Hiệu chỉnh Trigger (tt)

- Ví dụ: ràng buộc 0 <= LUONG <= 100000

```
ALTER TRIGGER LUONG_DUONG ON NHANVIEN
FOR INSERT, UPDATE
AS
BEGIN
 IF (SELECT COUNT(*) FROM INSERTED
      WHERE LUONG < 0 OR LUONG > 100000) > 0
   BEGIN
          Print '0 <= LUONG <= 100000'
          Rollback Tran
   END
END
```

# Nội dung chi tiết

- Giới thiệu
- Tạo Trigger
- Hiệu chỉnh Trigger
- **Xóa Trigger**
- Ví dụ

# Xóa Trigger

- Cú pháp

```
DROP TRIGGER <Ten_ Trigger>
```

- Ví dụ

```
DROP TRIGGER LUONG_DUONG
```

# Nội dung chi tiết

- Giới thiệu
- Tạo Trigger
- Hiệu chỉnh Trigger
- Xóa Trigger
- **Ví dụ**

# Ví dụ 1

- Ràng buộc lương nhân viên phải tăng (không bằng hoặc nhỏ hơn lương cũ)

```
CREATE TRIGGER LUONG_TANG ON NHANVIEN
FOR UPDATE
AS
IF UPDATE(LUONG)
BEGIN
        IF (SELECT COUNT(*) FROM INSERTED I, DELETED D
         WHERE D.LUONG >= I.LUONG AND I.MANV=D.MANV) > 0
        BEGIN
                Print N'Lương nhân viên phải tăng'
                Rollback Tran
        END
END
```

# Ví dụ 2

- Ràng buộc nhân viên phải từ 18 tuổi trở lên

```
CREATE TRIGGER TUOI_18_TROLEN ON NHANVIEN
FOR INSERT, UPDATE
AS
IF EXISTS (SELECT * FROM INSERTED
            WHERE DATEADD(YY,18,NGSINH) > GETDATE())
 BEGIN
        Print N'Nhân viên phải 18 tuổi trở lên'
        Rollback Tran
 END
```

# Ví dụ 3

- Ràng buộc không cho phép thêm, sửa bảng NHANVIEN nếu mã phòng (PHG) không có trong bảng PHONGBAN

```
CREATE TRIGGER TONTAI_PHONG ON NHANVIEN
FOR INSERT, UPDATE
AS
IF NOT EXISTS (SELECT * FROM INSERTED , PHONGBAN
                    WHERE PHG = MAPHG)
 BEGIN
        Print N'Mã phòng chưa tồn tại'
        Rollback Tran
 END
```

# Ví dụ 4

- Ràng buộc không cho sửa MAPHG trong bảng PHONGBAN

```
CREATE TRIGGER KHONG_SUA_KHOACHINH ON PHONGBAN
FOR UPDATE
AS
IF UPDATE(MAPHG)
BEGIN
        Print N'Không được sửa khóa chính'
        Rollback Tran
END
```

# Ví dụ 5

- Sửa MAPHG trong bảng PHONGBAN thì phải sửa luôn những mẫu tin có liên quan trong bảng NHANVIEN

```
CREATE TRIGGER SUA_DAYCHUYEN ON PHONGBAN
FOR UPDATE
AS
IF UPDATE(MAPHG)
BEGIN
        UPDATE NHANVIEN
        SET PHG = I.MAPHG
        FROM NHANVIEN, DELETED D, INSERTED I
        WHERE PHG = D.MAPHG
END
```

# Ví dụ 6

- Không cho xóa NHANVIEN nếu nhân viên đó có trong bảng PHANCONG

```
CREATE TRIGGER KHONG_XOA_NV_CO_PHANCONG ON NHANVIEN
FOR DELETE
AS
IF (SELECT COUNT(*) FROM DELETED , PHANCONG
        WHERE MANV = MA_NVIEN) > 0
 BEGIN
        Print N'Nhân viên có trong phân công'
        Rollback Tran
 END
```

# Bài tập áp dụng (QLVT)

- Tạo trigger để khi insert một record vào trong table CHITIETHOADON, thì cập nhật lại SLTON của vật tư đó trong table VATTU
- Tạo trigger để không cho phép một hoá đơn có nhiều hơn 4 chi tiết hoá đơn
- Tạo trigger không cho phép hai vật tư trùng tên
- Tạo trigger để không cho phép xoá cùng lúc nhiều hơn một khách hàng

- Chỉ cho phép mua các mặt hàng có số lượng tồn lớn hơn hoặc bằng số lượng cần mua và tính lại số lượng tồn mỗi khi có chi tiet hoa đơn.
- Không cho phép user xoá một lúc nhiều hơn một vật tư.
- Chỉ bán mặt hàng GẠCH (các loai gach) với số lượng là bội số của 100.

```
create trigger t1  on chitiethoadon
for insert
as
declare @sl int, @mavt varchar(10)
select @sl = sl, @mavt = mavt from inserted
update vattu set slt=slt- @sl where mavt =@mavt
```

```
create trigger T6 on chitiethoadon
for insert
as
declare @sl int, @mavt varchar(10)
select @sl = sl, @mavt = mavt from inserted
if (select count(*) from vattu where mavt=@mavt and SLTON<@sl)>0
 begin
   print N'số lượng tồn không đủ bán'
   rollback Tran
 end
update vattu set SLTON=SLTON- @sl where mavt =@mavt
```

```
insert into CHITIETHOADON values ('HD002','VT07',500,null,30000)
```

```
alter trigger T1 on chitiethoadon
for insert,update
as
begin
   update CHITIETHOADON set KHUYENMAI= case
              when a.sl>100 then a.sl*a.GIABAN*10/100
   when a.sl<=100 and a.sl>50 then a.sl*a.GIABAN*10/50
   else 0
   end
from inserted a where CHITIETHOADON.MAHD= a.mahd and
    CHITIETHOADON.MAVT=a.MAVT
end
insert into CHITIETHOADON values ('HD007','VT01',200,null,1000)
```