

## Chương 8

---

# TRIGGERS

# Nội dung

---

- Định nghĩa trigger
- Sử dụng trigger
- Các loại triggers
- INSERT triggers
- UPDATE triggers
- DELETE triggers

# Định nghĩa

---

- A trigger là một loại Procedure đặc biệt, nó được định nghĩa để tự động thực thi khi có một câu lệnh Insert, Update, Delete được sử dụng.
- Trigger dùng để ràng buộc các qui tắc quản lý, kiểm soát tính toàn vẹn dữ liệu một cách tự động khi dữ liệu bị hiệu chỉnh.
- Trigger tự động thực thi, không gọi trigger thi hành một cách trực tiếp.

# Sử dụng Triggers

---

- Trigger được định nghĩa trên 1 table cụ thể, nhưng không thể tạo trigger trên temporary hay system table.
- Không gọi trực tiếp hoặc truyền nhận tham số đối với trigger
- Có thể được kích hoạt bởi nhiều hơn 1 event
- Được sử dụng hầu hết các phát biểu T\_SQL để viết trigger All CREATE, ALTER, DROP, GRANT, REVOKE, DENY ,LOAD, RESTORE ,RECONFIGURE TRUNCATE TABLE ,UPDATE STATISTICS, SELECT INTO

# Sử dụng Triggers

---

- Xử lý hành động trên nhiều dòng.
- Đọc dữ liệu từ các Table khác trong CSDL khác.
- Không ngăn ngừa thay đổi cấu trúc, mà quan tâm đến sự thay đổi hay xóa dữ liệu trong các bảng có quan hệ với nhau.

# Sử dụng Triggers

---

- Các Constraint kiểm tra trước, sau đó mới tới Trigger.
- Không nên dùng quá nhiều trigger trong một table.
- Không thể tạo trigger trên các đối tượng ở Temporary.
- Không nên thiết kế Trigger trả về tập kết quả để đảm bảo tính chất chuyển tác giữa các user và lập trình.

# Creating Triggers

## Syntax

```
CREATE TRIGGER trigger_name  
ON table [WITH ENCRYPTION]  
{FOR | AFTER| INSTEAD OF}  
{[INSERT][,][UPDATE][,][DELETE] }  
[WITH APPEND] [NOT FOR REPLICATION]  
AS  
sql_statement [ . . . n ]
```

**Xem thông tin về trigger : lưu trong table sysobjects  
và syscomments**

```
Sp_helptext Trigger_Name  
Sp_helptrigger Table_Name  
Sp_depends Table_Name
```

# Hành động kích hoạt trigger

---

- **[DELETE] [ , ] [INSERT] [ , ] [UPDATE]** : thao tác mà khi thực hiện thì trigger tự động thực thi.
  - **Khi Insert** mẫu tin mới vào bảng thì mẫu tin mới đó cũng lưu trong bảng **INSERTED**
  - **Khi Delete** mẫu tin trong bảng thì mẫu tin bị xóa được di chuyển sang bảng **DELETED**.
  - **Khi Update** mẫu tin trong bảng thì bảng được cập nhật và bảng **INSERTED** chứa mẫu tin mới, còn **DELETED** chứa mẫu tin cũ.



# Thuộc tính của trigger

---

- **WITH APPEND**

- Chỉ định thêm một trigger
- WITH APPEND không được dùng với INSTEAD OF triggers.

- **NOT FOR REPLICATION**

- Trigger sẽ không thực hiện khi bảng có liên quan đến kỹ thuật sao chép nhân bản (rellication)
- ***sql\_statement***: câu lệnh SQL chứa điều kiện và hành động của trigger.

# Creating Triggers

---

- **AFTER triggers (mặc định):** khi delete hay insert một dòng vào Table, sau khi thực hiện thì Trigger mới tự động thực thi gọi là reactive.
- **INSTEAD OF triggers:** kiểm tra trước khi Insert/Delete. Không xây dựng được trên table có áp dụng cascade delete/ update
- **Nested trigger:** table 1 có trigger1, table 2 có trigger 2, nếu thao tác trên Table 1 mà có liên quan đến Table 2 thì Trigger 2 sẽ thực thi còn gọi là lồng Trigger

# Creating Triggers

---

For/After	Instead of
- Chỉ áp dụng cho table	- Áp dụng cho table, view
- Có thể định nghĩa nhiều trigger trên một hành động I/U/D	- Chỉ định nghĩa một Trigger trên một hành động I/U/D
Thực thi sau khi : <ul style="list-style-type: none"><li>+ Xử lý ràng buộc</li><li>+ Thực hiện xong hành động I/U/D phát sinh trigger</li></ul>	- Thi hành trước khi: <ul style="list-style-type: none"><li>+ Xử lý ràng buộc</li><li>+ Thay thế hành động phát sinh trigger</li></ul>
	- Không xây dựng được trên table có áp dụng cascade D/U

# Creating Triggers

	AFTER trigger	INSTEAD OF trigger
Thực thi	<ul style="list-style-type: none"><li>- Thực thi các RBTV</li><li>• Bật cờ sự kiện</li><li>• Tạo table inserted và deleted.</li><li>• Thực thi sự kiện bất trigger</li><li>• Thực thi code trong trigger</li></ul>	<ul style="list-style-type: none"><li>Tạo table inserted và deleted.</li><li>• Thực thi code trong trigger</li><li>• Thực thi các RBTV bỏ qua sự kiện bất trigger</li></ul>
Ràng buộc tham chiếu	Không giới hạn	Không cho phép
Table inserted và deleted	Không cho phép column có kiểu dữ liệu text, ntext, image	Cho phép column có kiểu dữ liệu text, ntext, image

# Creating triggers

---

- Example

```
CREATE TRIGGER ThemxoaCTHD ON [Order Details]
FOR INSERT, UPDATE
AS
    Raiserror ('Co %d dòng đã được hiệu
    chỉnh',0,1,@ @rowcount)
RETURN
```

# Creating triggers

---

Ví dụ:

```
CREATE TRIGGER NoDelete
ON Product
FOR DELETE AS
IF(SELECT ProductID FROM Deleted )=12
BEGIN
    Print 'You cannot delete the Productid=12'
    RollBack transaction
END
```

# Creating triggers

---

Ví dụ:

```
CREATE TRIGGER NoUpdate
ON Product
FOR Update
IF Update(ProductID)
    BEGIN
        PRINT 'You cannot update Productid'
        RollBack Transaction
    END
```

# Hiệu chỉnh Triggers

---

- Cú pháp:

```
ALTER TRIGGER [ schema_name . ] trigger_name  
ON { table | view }  
[ WITH ENCRYPTION ]  
{  
    { { FOR | AFTER | INSTEAD OF }  
      { [DELETE] [, ] [INSERT] [, ] [UPDATE] }  
      [ WITH APPEND ]  
      [ NOT FOR REPLICATION ]  
      AS  
      sql_statement [ ...n ]  
    } }  
}
```



## Ví dụ: Đảm bảo ràng buộc toàn vẹn dữ liệu

---

```
CREATE TRIGGER trDelINV
ON NHANVIEN
FOR DELETE
AS
RAISERROR('%d hàng bị xóa trong bảng NHANVIEN', 0,
          1, @@rowcount)
-----
CREATE TRIGGER trDelPhong
ON PHONG
FOR DELETE
AS
DELETE NHANVIEN FROM NHANVIEN ,DELETED WHERE
      DELETED.MAPHONG =NHANVIEN.MAPHONG
```

# Một số chú ý khi dùng trigger

---

- Một bảng có nhiều trigger
- Mỗi một trigger có tên duy nhất
- Trong trigger thường dùng mệnh đề **IF EXISTS**

## Ví dụ

*Tạo một trigger kiểm tra khóa ngoại **Manhom** khi nhập dữ liệu vào bảng **Danhmucsach***

```
create trigger ktkhoangoai
on dmsach for insert
as
    if exists(select * from inserted
              where manhom not in(select manhom from nhomsach))
begin
    print 'Ma nhom khong ton tai'
    rollback transaction
end
```

# Hiệu chỉnh Triggers

---

Ví dụ:

```
ALTER TABLE [Order Details]
```

```
DISABLE TRIGGER ALL
```

- **Xóa trigger:**
  - **DROP TRIGGER Trigger\_Name**

Chú ý: IF UPDATE không sử dụng được đối với câu lệnh DELETE.

# Các loại Triggers

---

*Insert Triggers*

*Update Triggers*

*Delete Triggers*

# *Insert Triggers*

---

- **Insert trigger:** Trigger sẽ được thực thi khi có mẫu tin chèn vào bảng, SQL server tạo ra bảng mang tên INSERTED để lưu các mẫu tin chèn, trong Trigger ta có thể tham khảo đến mẫu tin này.
- Các bước thực hiện
  - Step 1  
INSERT statement to a table with an INSERT trigger defined
  - Step 2  
INSERT Statement Logged
  - Step 3  
Trigger Actions Executed

# Insert Triggers

## TRIGGER Actions Execute

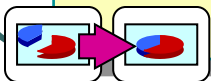
```
Trigger Code:  
USE Northwind  
CREATE TRIGGER OrdDet_Insert  
ON [Order Details]  
FOR INSERT  
AS  
UPDATE P SET  
UnitsInStock = (P.UnitsInStock - I.Quantity)  
FROM Products AS P INNER JOIN Inserted AS I  
ON P.ProductID = I.ProductID
```

**Order Details**

OrderID	ProductID	UnitPrice	Quantity	Discount
10522	10	31.00	7	0.2
10523	41	9.65	9	0.15
10524	7	30.00	1	0.0
10523	2	19.00	5	0.2

**Products**

ProductID	UnitsInStock	...	...
1	15		
2	10		
3	65		
4	20		



# *Insert Triggers*

---

**Example:**

**CREATE TRIGGER Trg\_NgayLap\_NgayGiaoHD**

**ON Orders AFTER INSERT**

**AS**

**DECLARE @NgayLapHD DateTime, @NgayGiao DateTime**

**SELECT @NgayLapHD=hd.Orderdate, @NgayGiao=hd.Required**

**FROM Orders hd INNER JOIN Inserted i ON hd.Orderid=i.orderid**

**If @NgayGiao<@NgayLapHD**

**BEGIN**

**RAISERROR(500103,10,1)**

**ROLLBACK TRANSACTION**

**END**

**INSERT HoaDon VALUES (1003,'1/1/2004','N','TP. HCM',111,'12/24/2003')**



# Insert Triggers

MaHD	NgayLapHD	LoaiHD	NoiChuyen	NgayGiao	MaKH
1001	12/23/2003	N	Tp. HCM	12/28/2003	CDCN4
1002	01/01/2004	X	Can Tho	01/05/2004	DHCT
1003	01/01/2004	N	Tp. HCM	03/03/2004	CDCN4

1

**INSERT HoaDon**

**VALUES (1003,'1/1/2004','N','TP. HCM',111,'12/24/2004')**

3

**INSERT Trigger**

MaHD	NgayLapHD	LoaiHD	NoiChuyen	NgayGiao	MaKH
1003	01/01/2004	N	Tp. HCM	03/03/2004	CDCN4

2

**Inserted Table**

# *Insert Triggers*

---

## **Example: kiểm tra khóa chính**

```
CREATE TRIGGER ktTonTai ON [Orders] FOR INSERT AS
IF EXISTS (SELECT * FROM INSERTED I inner join Orders o
           ON i.Orderid = o.Orderid
BEGIN
    printf “Đã có hóa đơn này rồi, nhập lại”
    ROLLBACK TRANSACTION
END
```

# *Insert Triggers*

---

## **Example: kiểm tra tồn tại khóa ngoại**

```
CREATE TRIGGER ktTonTai ON [Order details] FOR INSERT AS
IF NOT EXISTS (SELECT * FROM INSERTED I inner join Orders o
                ON i.Orderid = o.Orderid
BEGIN
    Raiserror(60000,16,1,'Orderid','Order
details',Orderid','Orders')
    ROLLBACK TRANSACTION
END
```

# *Delete Triggers*

---

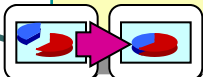
- Delete trigger: Trigger sẽ được thực thi khi có mẫu tin xóa khỏi bảng, SQL server tạo ra bảng mang tên DELETE để lưu các mẫu tin bị xóa, trong Trigger ta có thể tham khảo đến mẫu tin này.
- Có 3 cách ràng buộc khi sử dụng DELETE trigger.
  - The Cascade method
  - The Restrict method
  - The Nullify method

# Delete Triggers

## Trigger Actions Execute

```
USE Northwind
CREATE TRIGGER Category_Delete
  ON Categories
  FOR DELETE
AS
  UPDATE P SET Discontinued = 1
  FROM Products AS P INNER JOIN deleted AS d
  ON P.CategoryID = d.CategoryID
```

Products			
ProductID	Discontinued	...	...
1	0		
2	1		
3	0		
4	0		



# Delete Triggers

---

## Example:

```
ALTER TRIGGER Trg_Xoa_HD
ON Orders AFTER DELETE
AS
SET NOCOUNT ON
IF EXISTS (SELECT * FROM Deleted)
BEGIN
    DELETE [Order Details] WHERE [Order details].Orderid
    IN (SELECT hd.Orderid FROM orders hd
        INNER JOIN Deleted d ON hd.Orderid=d.Orderid)
    RAISERROR('Cac chi tiet HD da bi xoa',10,1)
END
SET NOCOUNT ON
-----
DELETE Orders WHERE Orderid=10178
```

# Delete Triggers

MaHD	NgayLapHD	LoaiHD	NoiChuyen	NgayGiao	MaKH
1001	12/23/2003	N	Tp. HCM	12/28/2003	CDCN4
1002	01/01/2004	X	Can Tho	01/05/2004	DHCT
1003	01/01/2004	N	Tp. HCM	03/03/2004	CDCN4

1

DELETED from HoaDon WHERE MaHD=1003

2

Deleted row

3

DELETED Trigger

MaHD	NgayLapHD	LoaiHD	NoiChuyen	NgayGiao	MaKH
1003	01/01/2004	N	Tp. HCM	03/03/2004	CDCN4

# *Update Triggers*

---

- **Update trigger:** mỗi khi có mẫu tin nào đó được cập nhập, giá trị những cột liên quan đến trigger sẽ được kiểm tra trước khi cập nhập. Mẫu tin bị cập nhập sẽ được sao lưu vào bảng insert (chứa giá trị mới) và bảng Delete(chứa giá trị cũ).
- **Các bước thực hiện**
  - Step 1:  
DELETE Statement to a Table with a DELETE Statement Defined
  - Step 2  
DELETE Statement Logged
  - Step 3  
Trigger Actions Executed



# Update Triggers

## TRIGGER Actions Execute

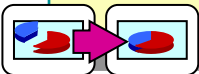
```
USE Northwind
GO
CREATE TRIGGER Employee_Update
  ON Employees
  FOR UPDATE
AS
IF UPDATE (EmployeeID)
BEGIN TRANSACTION
  RAISERROR ('Transaction cannot be processed.\
***** Employee ID number cannot be modified.', 10, 1)
ROLLBACK TRANSACTION
```



**Transaction cannot be processed.**  
**\*\*\*\*\* Member number cannot be modified**

### Employees

EmployeeID	LastName	FirstName	Title	HireDate
1	Davolio	Nancy	Sales Rep	~~~
2	Fuller	Andrew	Vice Pres.	~~~
3	Leverling	Janet	Sales Rep	~~~
4	Peacock	Margare	Sales Rep	~~~



# Update Triggers

MaHD	NgayLapHD	LoaiHD	NoiChuyen	NgayGiao	MaKH
1001	12/23/2003	N	Tp. HCM	12/28/2003	CDCN4
1002	01/01/2004	X	Can Tho	01/05/2004	DHCT
1003	01/01/2004	N	Tp. HCM	03/03/2004	CDCN4

**1** UPDATED HoaDon SET MaKH='TH3' WHERE MaHD=1003

**3** UPDATED Trigger

**2** UPDATE ROW

MaHD	NgayLapHD	LoaiHD	NoiChuyen	NgayGiao	MaKH
1003	01/01/2004	N	Tp. HCM	03/03/2004	TH3

# Update Triggers

MaHD	NgayLapHD	LoaiHD	NoiChuyen	NgayGiao	MaKH
1001	12/23/2003	N	Tp. HCM	12/28/2003	CDCN4
1002	01/01/2004	X	Can Tho	01/05/2004	DHCT
1003	01/01/2004	N	Tp. HCM	03/03/2004	th3

1

INSERTED Table

MaHD	NgayLapHD	LoaiHD	NoiChuyen	NgayGiao	MaKH
1003	01/01/2004	N	Tp. HCM	03/03/2004	th3

2

DELETED Table

MaHD	NgayLapHD	LoaiHD	NoiChuyen	NgayGiao	MaKH
1003	01/01/2004	N	Tp. HCM	03/03/2004	CDCN4

3  
UPDATED Trigger

# *Update Triggers*

---

## **Example**

```
CREATE TRIGGER NoUpdateProduct
ON Orders
FOR UPDATE AS
IF (SELECT Orderdate FROM inserted)>= getdate()
BEGIN
    PRINT 'Ngày lap phai lon hon hay bang
        ngay hom nay'
    ROLLBACK TRANSACTION
END
```

# *Update Triggers*

---

## **Bài tập**

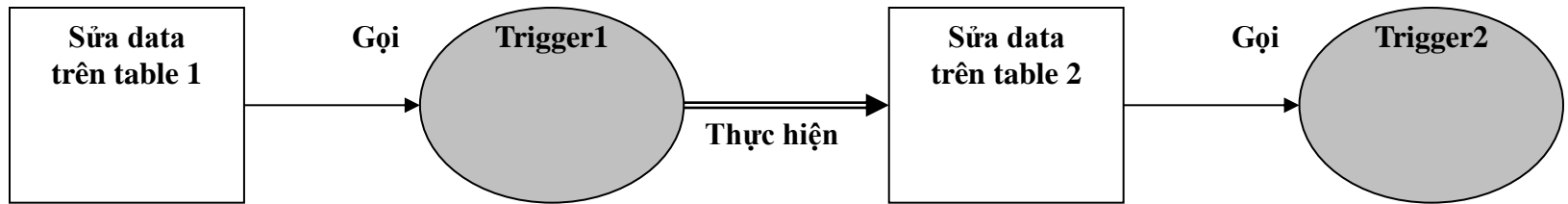
- 1) Tạo trigger khi thêm vào bảng chi tiết hóa đơn thì cập nhập lại tổng tiền của hóa đơn đó trong bảng hóa đơn
- 2) Tạo trigger khi thêm vào bảng chi tiết hóa đơn thì cập nhập lại số lượng tồn của sản phẩm trong bảng sản phẩm

# Nested Triggers

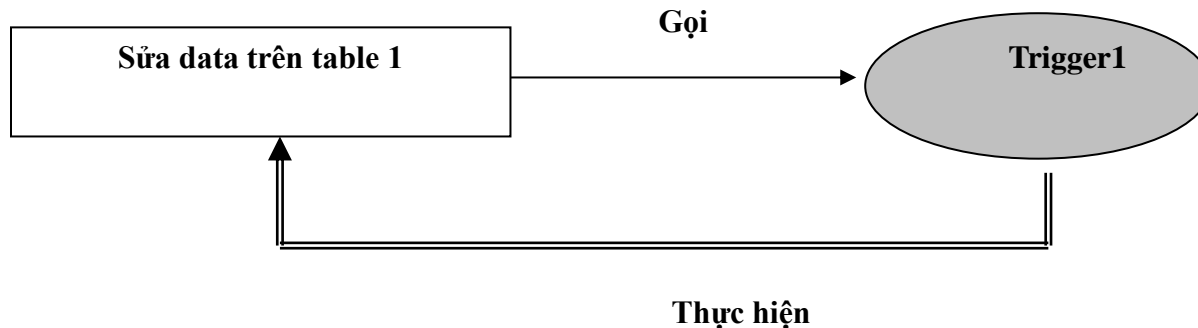
---

- Thao tác của một trigger kéo theo việc thi hành một trigger khác, các trigger này được gọi là trigger lồng nhau.
- Có thể lồng tối đa 32 cấp.
- Các trigger được xem như một đơn vị thi hành transaction. Do vậy, một trigger trong dãy trigger lồng nhau bị lỗi, thì SQL Server sẽ rollback tất cả các action đã thực hiện bởi các trigger.

# Nested Triggers

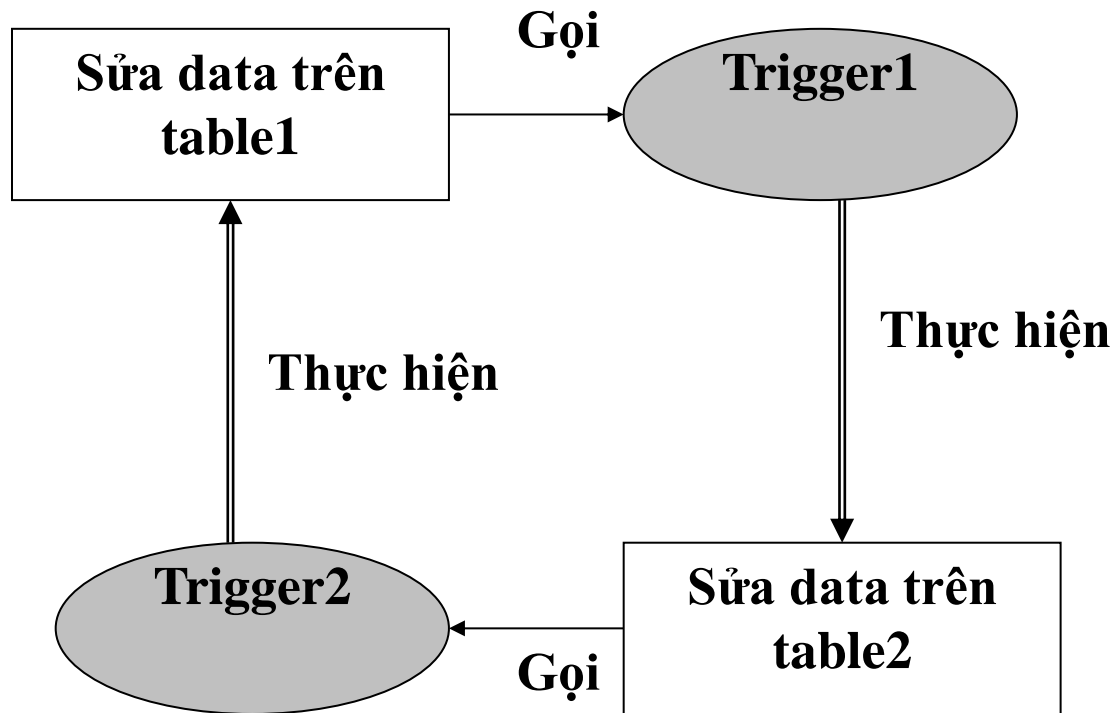


- **Trigger gọi chính nó (recursive trigger) :**  
Để tạo trigger dạng này phải bật option của database:  
`sp_dboption database_name, 'recursive triggers', True`



# Nested Triggers

---





# INSTEAD OF Trigger on View

---

## View definition

```
CREATE VIEW service_view
AS
    SELECT o.Productid as ma1, p.Productid as ma2,
           ProductName , orderid
    FROM Products p JOIN [order details] o
    ON p.productid =o.productid
```

# INSTEAD OF Trigger on view (contd.)

---

## Trigger Definition

```
CREATE TRIGGER del_service  
ON service_view  
INSTEAD OF DELETE  
AS
```

```
    DELETE Products WHERE Productid IN  
        (SELECT ma1 FROM DELETED)  
    DELETE [order details] WHERE productid IN  
        (SELECT ma2 FROM DELETED)
```

# INSTEAD OF TRIGGERS

---

- ***INSTEAD OF* Triggers**
  - Trigger này sẽ thi hành thay cho các câu lệnh Insert, Delete, Update. Khi tạo trigger kiểu này bạn phải viết lại các lệnh Insert, Delete, Update đối với dữ liệu.
  - Có thể áp dụng cho cả View và Table.
  - Không cho phép áp dụng với các View có lựa chọn With Check Option

# INSTEAD OF TRIGGERS

---

**Example: tests the quantity of a product in stock before accepting an order**

```
CREATE TRIGGER InsOrdDet ON [Order Details]
INSTEAD OF INSERT
AS
DECLARE @qty int
SELECT @qty=quantity FROM Inserted
IF @qty<= (SELECT UnitsInStock FROM Products P JOIN Inserted I ON
           P.ProductID = I.ProductID)
    INSERT INTO [Order Details]
    SELECT * FROM Inserted
ELSE
    RAISERROR('Not enough products in stock', 16, 1)
```

# Enforcing Data Integrity

```
CREATE TRIGGER BackOrderList_Delete
ON Products FOR UPDATE
AS
IF (SELECT BO.ProductID FROM BackOrders AS BO JOIN
    Inserted AS I ON BO.ProductID = I.Product_ID
    ) > 0
BEGIN
    DELETE BO FROM BackOrders AS BO
    INNER JOIN Inserted AS I
    ON BO.ProductID = I.ProductID
END
```

<i>Products</i>			
<i>ProductID</i>	<i>UnitsInStock</i>	<i>...</i>	<i>...</i>
1	15		
2	15		
3	65		
4	20		



Trigger Deletes Row

<i>BackOrders</i>		
<i>ProductID</i>	<i>UnitsOnOrder</i>	<i>...</i>
1	15	
12	10	
3	65	
2	15	

# Enforcing Business Rules

## Products with Outstanding Orders Cannot Be Deleted

```
IF (Select Count (*)  
    FROM [Order Details] INNER JOIN deleted  
    ON [Order Details].ProductID = deleted.ProductID  
    ) > 0  
ROLLBACK TRANSACTION
```

DELETE statement executed on  
Product table

<i>Products</i>			
<i>ProductID</i>	<i>UnitsInStock</i>	<i>...</i>	<i>...</i>
1	15		
2	0		
3	65		
4	20		

Trigger code  
checks the Order Details  
table

<i>Order Details</i>				
<i>OrderID</i>	<i>ProductID</i>	<i>UnitPrice</i>	<i>Quantity</i>	<i>Discount</i>
10522	10	31.00	7	0.2
10523	2	19.00	9	0.15
10524	41	9.65	24	0.0
10525	7	30.00		

Transaction  
rolled back

'Transaction cannot be processed'  
'This product has order history'

- VD: cập nhật điểm môn học -> tự động cập nhật gpa

```
CREATE TRIGGER auto_updateGPA ON enroll  
FOR UPDATE, DELETE  
AS
```

---

```
UPDATE Student  
SET GPA = agv(mark)  
FROM Student s INNER JOIN enroll e ON s.SID = e.SID  
WHERE e.SID in (SELECT SID FROM deleted)
```

- VD: một sinh viên không được học quá 10 môn

```
CREATE TRIGGER overTotalcCourse ON enroll  
FOR INSERT  
AS
```

```
IF EXISTS (SELECT 1 FROM enroll WHERE SID in  
(SELECT SID FROM inserted)  
GROUP BY SID  
HAVING COUNT(CID) > 10
```

```
)
```

```
ROLLBACK TRAN
```

# Triggers

---

- Cập nhật điểm môn học thông qua view

```
CREATE VIEW V_enroll  
AS  
SELECT * FROM enroll
```

```
CREATE TRIGGER update_mark ON V_enroll  
INSTEAD OF UPDATE  
AS
```

```
    UPDATE enroll  
    SET mark = v.mark  
    FROM enroll e INNER JOIN inserted I on e.SID= i.SID AND  
    e.CID = s.SID
```



## Sử dụng mệnh đề **IF UPDATE** trong trigger

---

- Ví dụ 5.13: Xét lại ví dụ với hai bảng MATHANG và NHATKYBANHANG, trigger dưới đây được kích hoạt khi ta tiến hành cập nhật cột SOLUONG cho một bản ghi của bảng NHATKYBANHANG (lưu ý là chỉ cập nhật đúng một bản ghi)

## Sử dụng mệnh đề **IF UPDATE** trong trigger

---

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong
ON nhatkybanhang
FOR UPDATE
AS
IF UPDATE(soluong)
UPDATE mathang
SET mathang.soluong = mathang.soluong –
    (inserted.soluong-deleted.soluong)
FROM (deleted INNER JOIN inserted ON
    deleted.stt = inserted.stt) INNER JOIN mathang
    ON mathang.mahang = deleted.mahang
```

# Sử dụng mệnh đề **IF UPDATE** trong trigger

---

- Với trigger ở ví dụ trên, câu lệnh:  
UPDATE nhattybanhang  
SET soluong=soluong+20  
WHERE stt=1
- sẽ kích hoạt trigger ứng với mệnh đề IF UPDATE (soluong) và câu lệnh UPDATE trong trigger sẽ được thực thi.
- Tuy nhiên câu lệnh sau lại không kích hoạt trigger này  
UPDATE nhattybanhang  
SET nguoiimua='Mai Hữu Toàn'  
WHERE stt=3

# Sử dụng mệnh đề **IF UPDATE** trong trigger

---

- Ví dụ 5.14: Giả sử ta định nghĩa bảng R như sau:

```
CREATE TABLE R
(  A INT,
   B INT,
   C INT
)
```

và trigger trg\_R\_update cho bảng R:

```
CREATE TRIGGER trg_R_test
ON R
FOR UPDATE
AS
  IF UPDATE(A)
    Print 'A updated'
  IF UPDATE(C)
    Print 'C updated'
```

# Sử dụng mệnh đề **IF UPDATE** trong trigger

---

Câu lệnh:

```
UPDATE R SET A=100 WHERE A=1
```

sẽ kích hoạt trigger và cho kết quả là: A updated  
và câu lệnh:

```
UPDATE R SET C=100 WHERE C=2
```

cũng kích hoạt trigger và cho kết quả là: C updated  
còn câu lệnh:

```
UPDATE R SET B=100 WHERE B=3
```

hiển nhiên sẽ không kích hoạt trigger