

# Chương 3

## Lập trình T-SQL

### Phần 3 - Trigger

**Giáo trình & Tài liệu tham khảo:**

1. **Microsoft SQL Server 2008 R2 Unleashed**, Ray Rankins, Paul Bertucci, Chris Gallelli, Alex T. Silverstein, 2011, Pearson Education, Inc
2. **MS SQL Server 2012 T-SQL fundamentals**, Tizik Ben-Gan
3. <https://docs.microsoft.com/>

# Nội dung

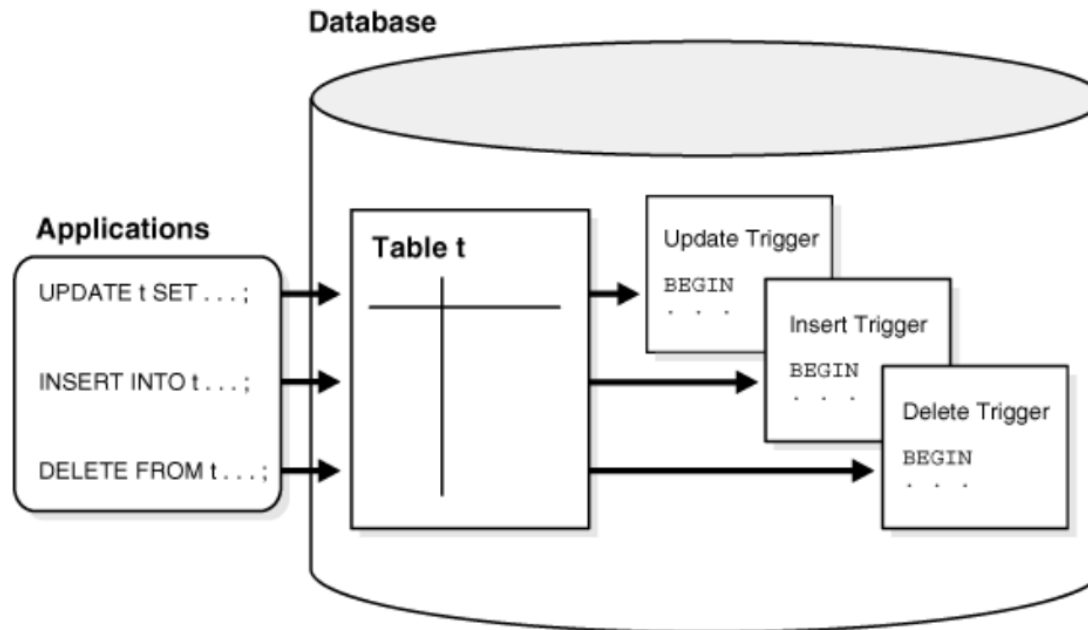
- Khái niệm
- Hai loại trigger và cơ chế hoạt động
- Các ứng dụng

# Khái niệm

- **Trigger là một loại stored procedure đặc biệt, tự động chạy khi một event xảy ra trong database hay server**
- **Event ?**
  - DML events : thêm , xóa, sửa dữ liệu
  - DDL events : tạo/sửa một object trong database
  - System events : startup, shutdown, error messages
  - User events : logon và logoff

# Khái niệm

Các trigger chạy tự động khi user thực hiện các lệnh thay đổi dữ liệu trên table t



# Khái niệm

## Ứng dụng của Trigger

- Tự động phát sinh giá trị cho các cột tính toán
- Đảm bảo các ràng buộc mà không cài đặt được bằng data types, constraints, defaults, rules
- Nhập/Chỉnh sửa dữ liệu trong table khi user thực hiện thao tác trên views
- Đọc/Chỉnh sửa dữ liệu trong table , bao gồm table trong DB khác
- Hiển thị thông tin về các DB events, user events ...
- ...

# Khái niệm

## Ứng dụng của Trigger

Trigger cho phép đảm bảo các ràng buộc mà không cài đặt được bằng data types, constraints, defaults, or rules.

- Với những db giảm dạng chuẩn, tồn tại dư thừa dữ liệu => cần duy trì sự toàn vẹn dữ liệu bằng sử dụng trigger
- Với dạng ràng buộc trên column mà dựa trên các rows trong cùng table hoặc các row trong table khác => dùng trigger để đảm bảo ràng buộc
- Sử dụng trigger để phát sinh giá trị default dựa trên data trong column, row hay table khác
- Sử dụng trigger để đảm bảo các ràng buộc toàn vẹn giữa các table trong các database khác nhau

# Khái niệm

## Phân loại

- Phân loại trigger theo loại action mà trigger thực hiện
  - **DML trigger** : trigger thực hiện các thao tác trên dữ liệu như insert , update , delete dữ liệu
  - DDL trigger : trigger thực hiện thay đổi các object trong DB
  - ...

# Khái niệm

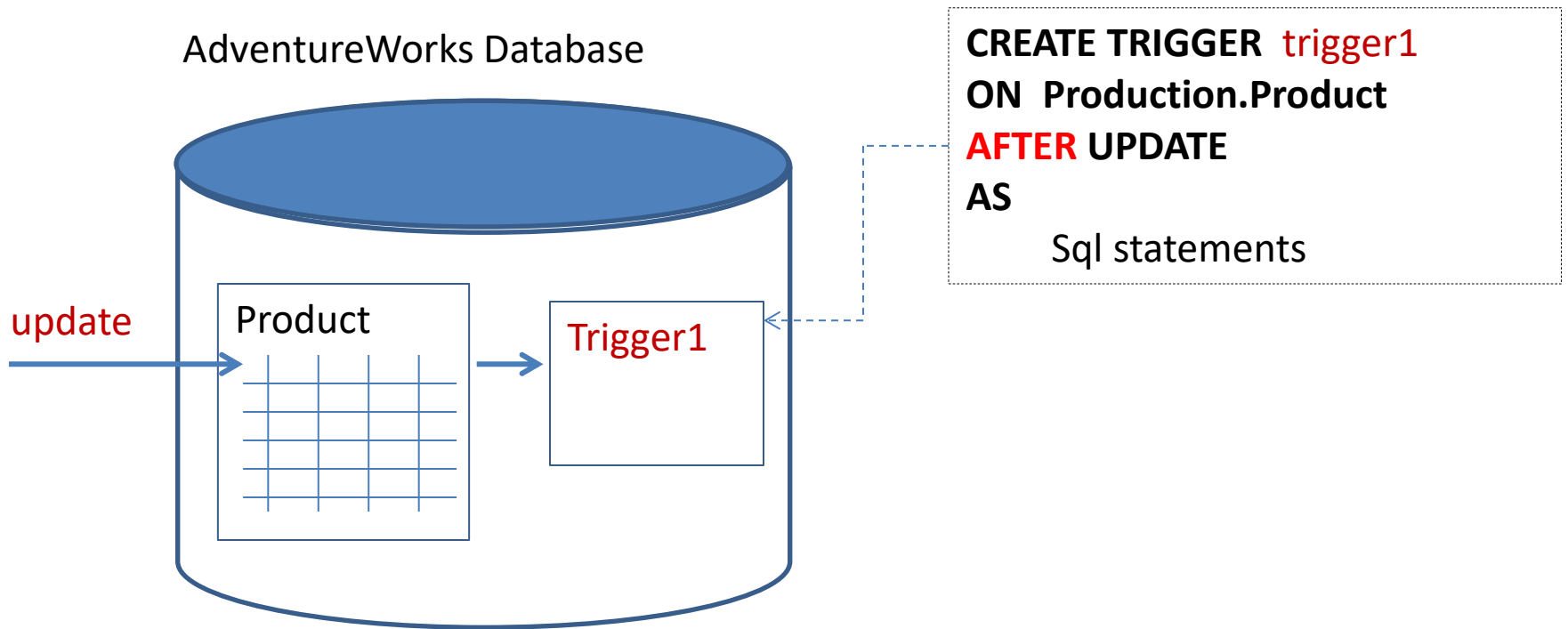
## Phân loại

- Phân loại theo cơ chế hoạt động => Hai loại trigger
  - AFTER trigger
  - INSTEAD OF trigger



# AFTER trigger

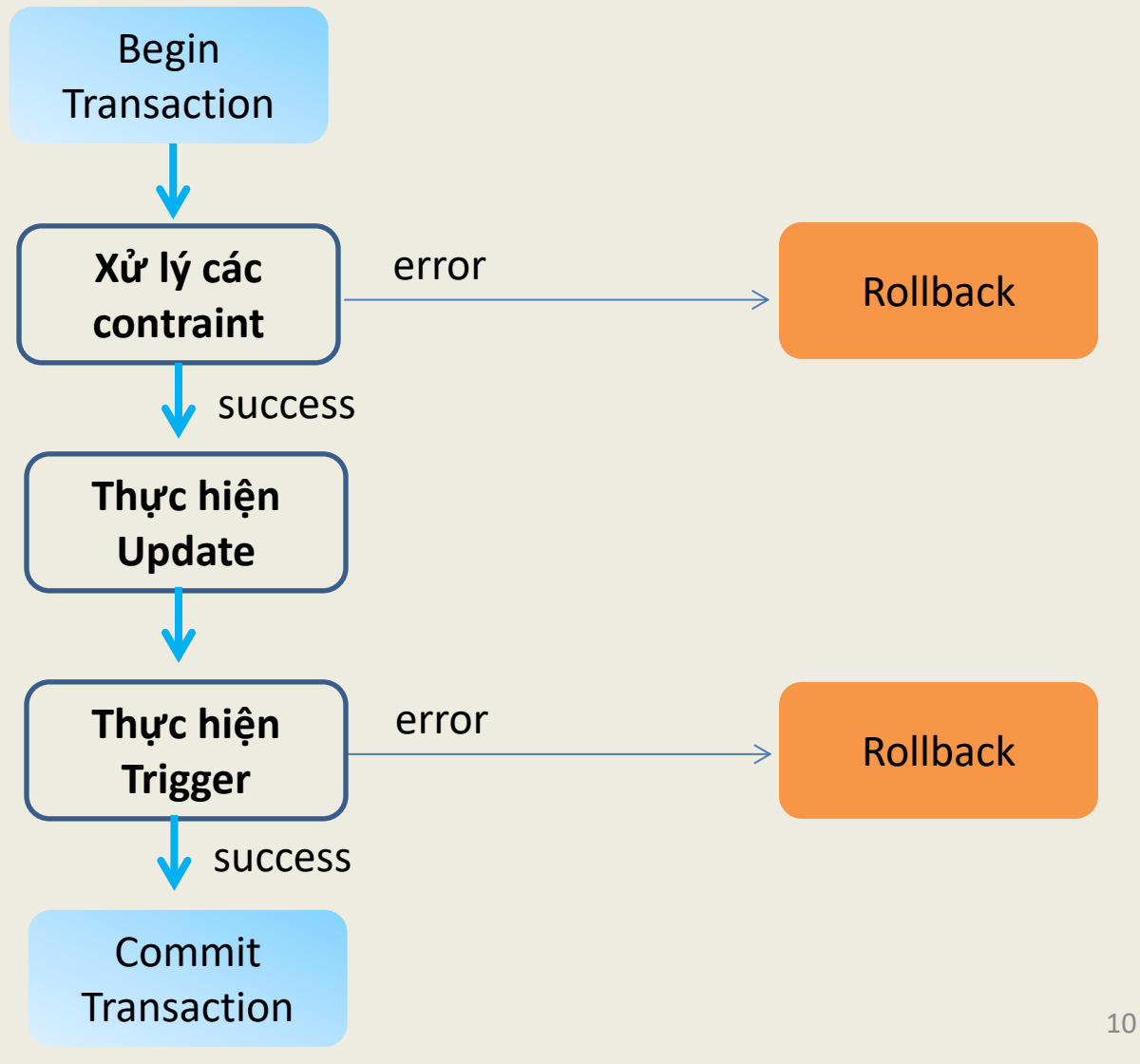
## Cơ chế hoạt động



# AFTER trigger

## Cơ chế hoạt động

Thực thi  
lệnh **update**  
trên  
table  
product



# AFTER trigger

## Cú pháp

```
CREATE TRIGGER trigger_name
```

```
ON table_name
```

```
AFTER { INSERT | UPDATE | DELETE }
```

```
AS
```

```
SQL statements
```

← Tên trigger

← Trigger phát sinh  
khi sự kiện nào  
xảy ra trên table  
nào

← Nội dung

→ Loại  
Trigger

# AFTER trigger

## Cú pháp

```
CREATE TRIGGER test  
ON Production.Product  
AFTER UPDATE ,INSERT, DELETE  
AS  
print 'trigger vidu1b'
```

- Tạo trigger tên 'test'
- Trigger tự động chạy khi xảy ra event = thực hiện thao tác UPDATE ,hoặc INSERT, hoặc DELETE trên bảng Product
- Trigger thực hiện in ra 1 message

# AFTER trigger

## Ví dụ 1

```
CREATE TRIGGER vidu1  
ON Production.Product  
AFTER UPDATE  
AS  
print 'trigger vidu1'
```

- *Tạo trigger tên 'vidu1'*
- *Trigger tự động chạy khi xảy ra event = thực hiện thao tác update trên bảng Product*
- *Trigger thực hiện in ra 1 message*

# AFTER trigger

## Ví dụ 1

User thực thi câu lệnh update

```
UPDATE Production.Product
SET SafetyStockLevel = 1200
WHERE ProductID = 1
```

Begin  
Transaction

Xử lý các  
constraint

success

Thực hiện  
Update

Cập nhật  
SafetyStockLevel  
một record trong  
Product

Thực hiện  
Trigger

In ra message

success

Commit  
Transaction

# Sử dụng **Inserted** và **Deleted** tables

- Là các temporary memory-resident tables, chứa những dòng data được tác động bởi lệnh insert, update, delete trên table
- Các table này chỉ được sử dụng trong trigger
- Dữ liệu trong Inserted và Deleted, cụ thể là

<b>Statement</b>	<b>Contents of inserted</b>	<b>Contents of deleted</b>
INSERT	Rows added	Empty
UPDATE	New rows	Old rows
DELETE	Empty	Rows deleted

# Sử dụng Inserted và Deleted tables

## Ví dụ 2

-- Tạo trigger Hiển thị nội dung Inserted và Deleted tables

Create trigger vidu2  
on Production.Product  
after update  
as

print 'Inserted table : '

*select \* , 'Inserted row's here' from inserted*

print 'Deleted table : '

*select \* , 'Deleted row's here' from deleted*

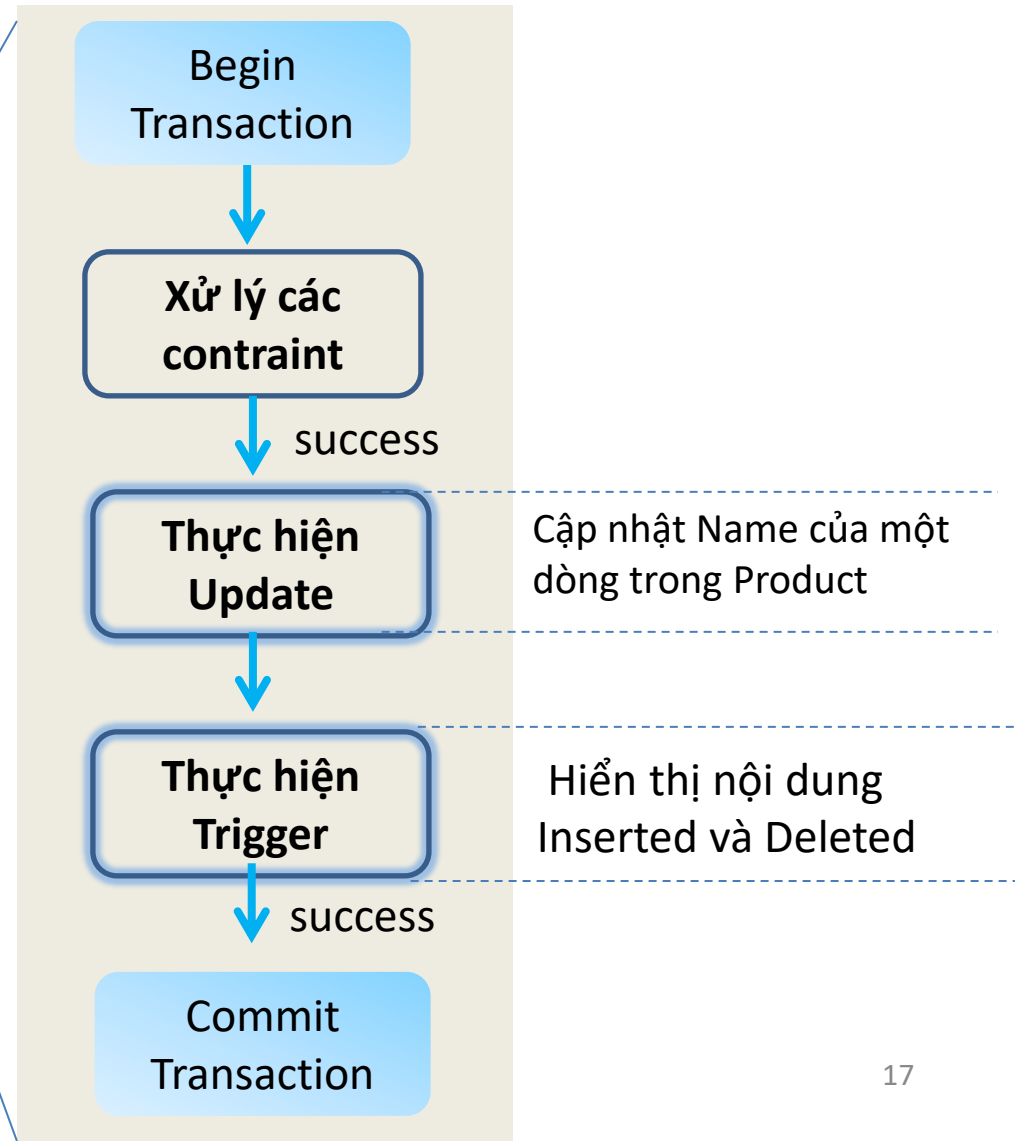


# Sử dụng Inserted và Deleted tables

## Ví dụ 2

-- Thực thi lệnh update  
trên table Product  
để test trigger


Update Production.Product  
Set name = 'Steel' + name  
Where ProductID = 2



Lệnh update thay đổi 1 dòng trong Product table => Sự thay đổi được lưu lại trong Deleted và Inserted Table

```
Update Production.Product  
Set name = 'Steel' + name  
Where ProductID = 2
```

**Deleted table** chứa Old row



ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost
2	Bearing Ball	BA-8327	0	0	NULL	1000	750	0.00

**Inserted table** chứa New row

ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost
2	SteelBearing Ball	BA-8327	0	0	NULL	1000	750	0.00

# Sử dụng Inserted và Deleted tables

## Ví dụ 3

-- Tạo trigger : khi user update cột **UnitPrice** trên bảng **Purchasing.PurchaseOrderDetail** thì thay đổi tương ứng cột **StandardCost** trong **Production.Product**

### Create trigger vidu3

```
on Purchasing.PurchaseOrderDetail
after update
as
If Update (UnitPrice)
Begin
    update Production.Product
    set StandardCost = Unitprice
    From Production.Product p join Inserted I
    on p.ProductID = i.ProductID
End
```

# Sử dụng Inserted và Deleted tables

## Ví dụ 3

-- thực hiện update trên bảng *Purchasing.PurchaseOrderDetail*  
=> Quan sát kết quả trên 2 table:  
*Purchasing.PurchaseOrderDetail* và *Production.Product* ?

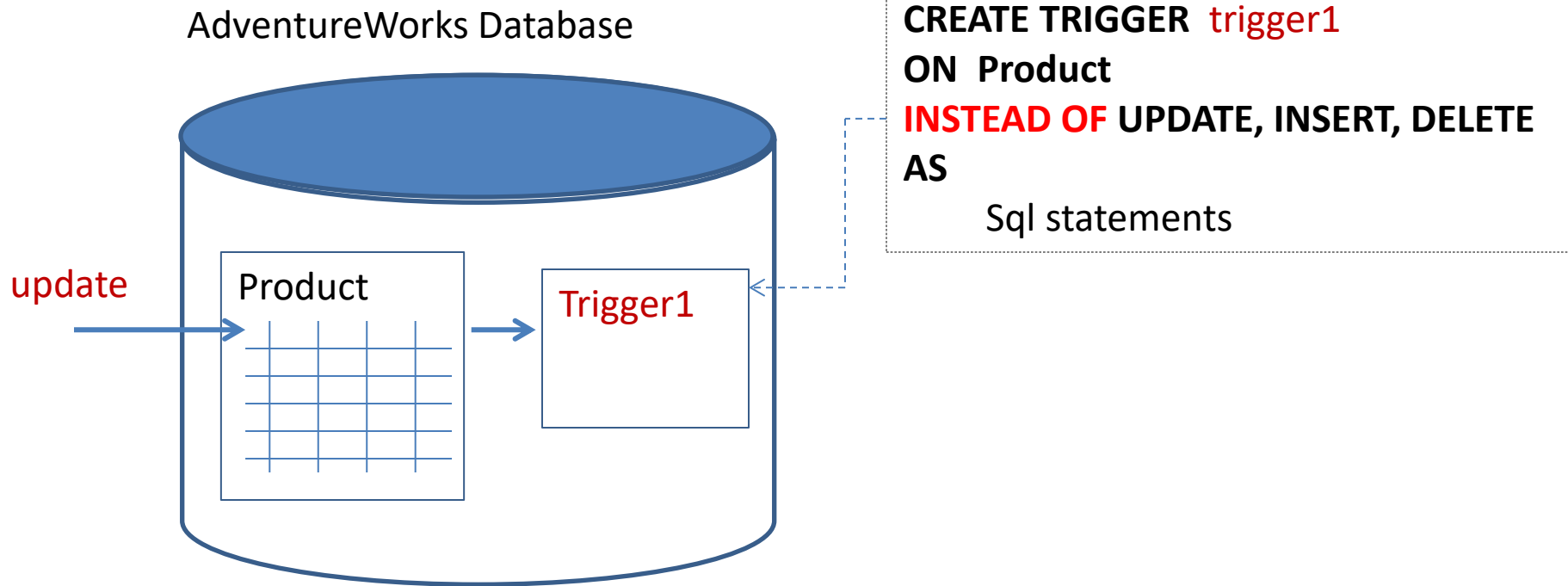
Update *Purchasing.PurchaseOrderDetail*  
set UnitPrice = 14  
where ProductID = 707

GO

Update *Purchasing.PurchaseOrderDetail*  
set UnitPrice = 13.0863  
where ProductID = 707

# INSTEAD OF trigger

## Cơ chế hoạt động



# INSTEAD OF trigger

## Cơ chế hoạt động

Thực thi  
lệnh  
**update**  
trên table  
Product

Begin  
Transaction

Thực hiện  
Trigger  
(trigger thực  
hiện *thay thế*  
lệnh Update )

error

Rollback

success

Commit  
Transaction

Error ?

*Khi tham chiếu object không tồn tại  
Hoặc vi phạm các ràng buộc*

# INSTEAD OF trigger

## Cú pháp

```
CREATE TRIGGER trigger_name
```

```
ON table_name
```

```
INSTEAD OF { INSERT | UPDATE | DELETE }
```

```
AS
```

```
SQL statements
```

← Tên trigger

← Trigger phát sinh  
khi sự kiện nào  
xảy ra trên table  
nào

← Nội dung

Loại  
Trigger

# INSTEAD OF so sánh với AFTER trigger

- Thực hiện tạo Instead of trigger tương tự ví dụ 1

```
CREATE TRIGGER triggerInsteadof  
ON Production.Product  
INSTEAD OF UPDATE  
AS  
print 'Insteadof trigger'
```



# INSTEAD OF so sánh với AFTER trigger

User thực thi câu lệnh update

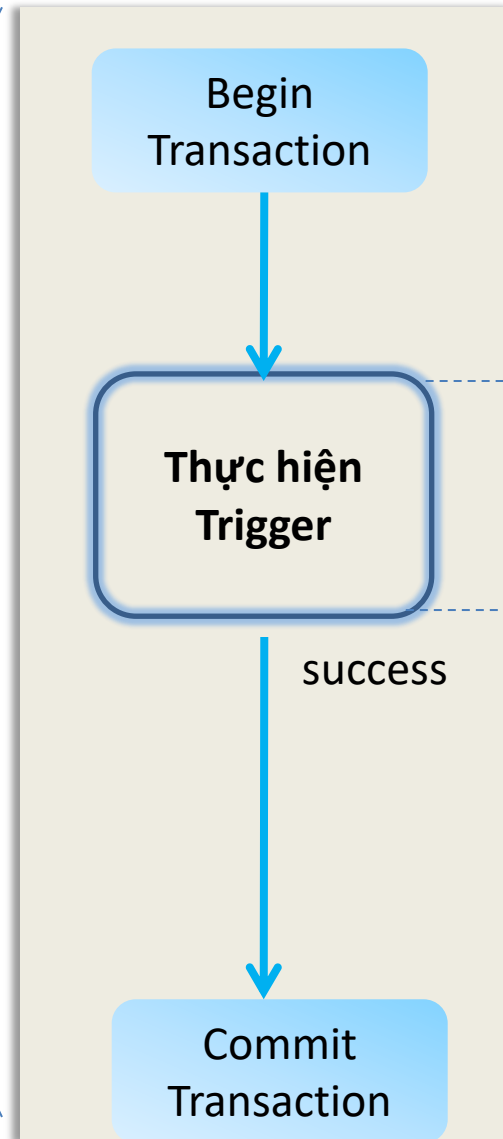
**UPDATE**

Production.Product

**SET**

SafetyStockLevel = 1500

**WHERE** ProductID = 1



In ra Message

# INSTEAD OF trigger

## Ví dụ 4

*-- Tạo trigger dạng instead of cho thao tác update trên bảng Product. Trigger thực hiện update dữ liệu và in ra message .*

```
CREATE TRIGGER vidu4
ON Production.Product
INSTEAD OF UPDATE
AS
PRINT 'trigger thực thi update ...'
UPDATE Production.Product
SET SafetyStockLevel = i.SafetyStockLevel
FROM Production.Product p join Inserted I
    on p.ProductID = i.ProductID
```

# INSTEAD OF trigger

## Ví dụ 4

-- Cách 2

**CREATE TRIGGER** vidu4

**ON** Production.Product

**INSTEAD OF UPDATE**

**AS**

**PRINT** 'trigger thực thi update ...'

**Declare** @stocklevel int, @proID int

**Select** @stocklevel = SafetyStockLevel , @proID = ProductID  
from Inserted

**UPDATE** Production.Product

**SET** SafetyStockLevel = @stocklevel

**WHERE** ProductID = @proID

# INSTEAD OF trigger

## Ví dụ 4

User thực thi câu lệnh update

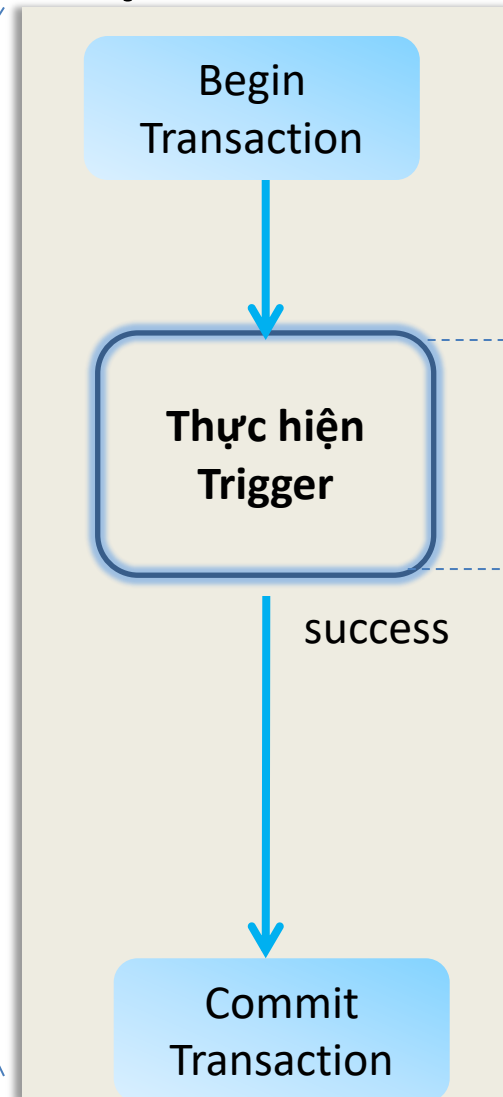
**UPDATE**

Production.Product

**SET**

SafetyStockLevel = 1500

**WHERE** ProductID = 1



In ra Message  
và  
Thực thi lệnh Update

# INSTEAD OF trigger

## Ví dụ 4\_2

Create view vw\_product

as

```
select Productid, p.name, ListPrice, psc.Name as SubCatName,  
psc.ProductSubcategoryID  
from Production.Product p join Production.ProductSubcategory psc  
on p.ProductSubcategoryID = psc.ProductSubcategoryID  
where psc.ProductSubcategoryID = 1
```

*---- update qua view không được*

Update vw\_product

```
set Listprice = listprice + 1 , SubCatName = 'L1 ' + SubCatName
```

# Instead of trigger

## Ví dụ 4\_2

*-- Tạo trigger cho thao tác update trên view vw\_product. Trigger thực hiện update trong các table tương ứng*

### Create trigger cau4\_2

On vw\_product

Instead of update

As

**Update** Production.ProductSubcategory

Set Name = I.SubCatName

From Production.ProductSubcategory psc join Inserted I

on psc.ProductSubcategoryID = I.ProductSubcategoryID

**Update** Production.Product

Set ListPrice = I.ListPrice

From Production.Product p join Inserted I on p.productID=I.productID

# Instead of trigger

## Ví dụ 4\_2

*-- cách viết khác*

Create trigger cau4\_2b

On vw\_product

Instead of update

As

**Declare** @name nvarchar(5), @price = float , @proID int, @proSubCat int

**Select** @name = subCatName , @proSubCat = ProductSubcategoryID ,

@price = ListPrice , @proID = productID

from Inserted

**Update** Production.ProductSubcategory

Set Name = @Name

Where ProductSubcategoryID = @proSubCat

**Update** Production.Product

Set ListPrice = @price

Where ProductID = @proID

# Ví dụ 5

- Hai lệnh sau cho thông tin về mã khuyến mãi có mã là 2

```
Select * from sales.SpecialOffer  
where SpecialOfferID = 2
```

GO

```
Select * from sales.SpecialOfferProduct  
where SpecialOfferID = 2
```

Mã khuyến mãi này áp dụng cho mặt hàng 707 , thời gian áp dụng từ 2005-07-01 đến 2008-06-30, với số lượng mua từ 11 -> 14

=> **Hãy viết một trigger đảm bảo những điều kiện của mã khuyến mãi 2 ?**

=> viết trigger cho thao tác update cột OrderDate trên SalesOrderHeader, nếu OrderDate nằm ngoài khoảng thời gian trên đối với hóa đơn đang áp dụng mã khuyến mãi 2, thì không cho phép sửa và xuất ra thông báo



# Ví dụ 5

-- cách 1 : dùng After trigger

**Create trigger vd5\_1**

On sales.SalesOrderHeader

**After** Update

As

If exists (select \*

from Inserted i join Sales.SalesOrderDetail sod on  
i.SalesOrderID = sod.SalesOrderID

where SpecialOfferID = 2 and

(OrderDate < '2005-07-01' or OrderDate > '2008-06-30') )

Begin

print N'Thay đổi OrderDate không hợp lệ do mặt hàng đang áp dụng  
mã khuyến mãi 2 '

rollback

End

# Ví dụ 5

-- cách 2 : dùng instead of trigger

**Create trigger vd5\_2**

On sales.SalesOrderHeader

**Instead of** Update

As

**If exists** (select \* from Inserted i join Sales.SalesOrderDetail sod  
on i.SalesOrderID = sod.SalesOrderID  
where SpecialOfferID = 2 and  
(OrderDate < '2005-07-01' or OrderDate > '2008-06-30') )

Print N'Không được thay đổi OrderDate ngoài khoảng thời gian áp dụng mã khuyến mãi 2 '

**Else if exists** (select \* from Inserted i join Sales.SalesOrderDetail sod  
on i.SalesOrderID = sod.SalesOrderID  
where SpecialOfferID = 2 )

Update sales.SalesOrderHeader

Set soh.OrderDate = i.OrderDate

From sales.SalesOrderHeader soh join Inserted I  
on soh.SalesOrderID = i.SalesOrderID

# Ví dụ 6

-- Nhận diện thao tác user vừa thực hiện trên table Product ?

Create trigger vidu6

on Production.Product

after insert, update, delete

as

If not exists (select \* from Deleted)

print 'user đã thực hiện insert dữ liệu'

Else if not exists (select \* from Inserted)

print 'user đã thực hiện delete dữ liệu'

Else

print 'user đã thực hiện update dữ liệu'

# Ví dụ 6

*-- thực hiện lệnh Insert, update , delete 1 dòng trên Product để test trigger*

# Ví dụ 7

-- Tạo trigger (after hoặc Instead of) cho thao tác **update** trên các table *Purchasing.PurchaseOrderDetail* pod và *[Production].Product* p để đảm bảo ràng buộc :  $pod.Unitprice = p.StandardCost$  với những mặt hàng mua từ nhà cung cấp để Resale

-- tạo trigger trên *Purchasing.PurchaseOrderDetail*

**CREATE TRIGGER** vidu71

**ON** *Purchasing.PurchaseOrderDetail*

**AFTER UPDATE**

**AS**

**If update(UnitPrice)**

**If exists** ( select \* from Inserted a join *[Production].Product* b on  
a.SalesOrderID = b.SalesOrderID  
where a.Unitprice <> b.StandardCost and  
MakeFlag = 0 and FinishedGoodsFlag = 1 )

Begin

print 'Đơn giá nhập UnitPrice phải bằng với StandardCost của mặt hàng'

rollback

end

# Ví dụ 7

*-- tạo trigger trên Production.Product*

**CREATE TRIGGER** vidu72

**ON** *Production.Product*

**AFTER UPDATE**

**AS**

**If update(StandardCost)**

**If exists** ( select \*

from Inserted a join Purchasing.PurchaseOrderDetail b

on a.SalesOrderID = b.SalesOrderID

where a.StandardCost <> b.Unitprice and

MakeFlag = 0 and FinishedGoodsFlag = 1 )

Begin

print N'Đơn giá nhập UnitPrice phải bằng với StandardCost của mặt hàng'

rollback

end

# Ví dụ 8

-- Dùng mệnh đề **IF UPDATE(column)** kiểm tra lệnh update thực hiện chỉnh sửa trên cột nào ?

*Tạo trigger cho thao tác update trên bảng Product, không cho phép chỉnh sửa cột StandardCost và Name*

Create trigger vidu8

on Production.Product

after update

as

If (Update(StandardCost) or Update(Name))

Begin

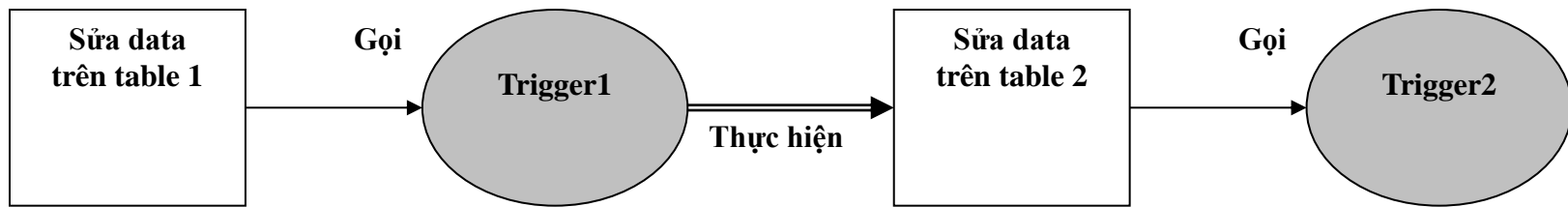
print N'Không được phép hiệu chỉnh cột StandardCost và cột Name'

rollback

End

# Chuỗi các trigger

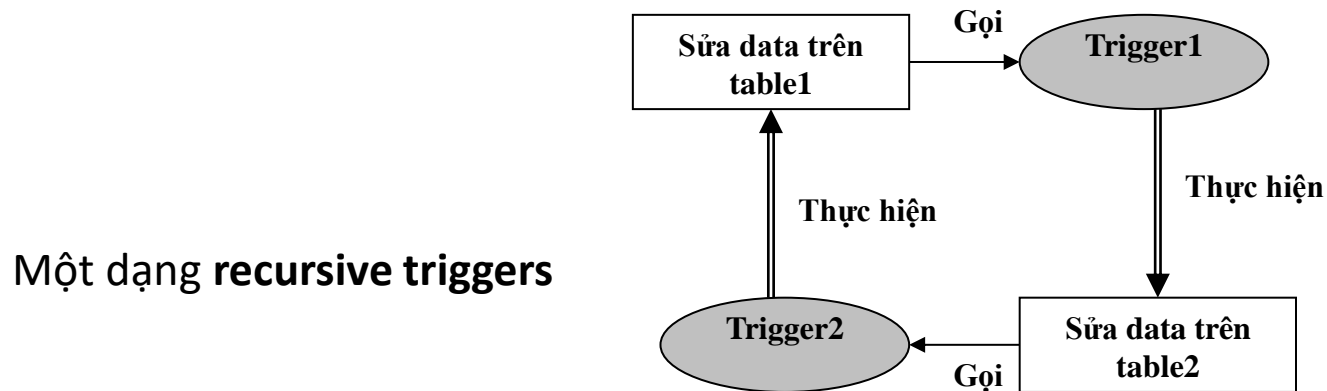
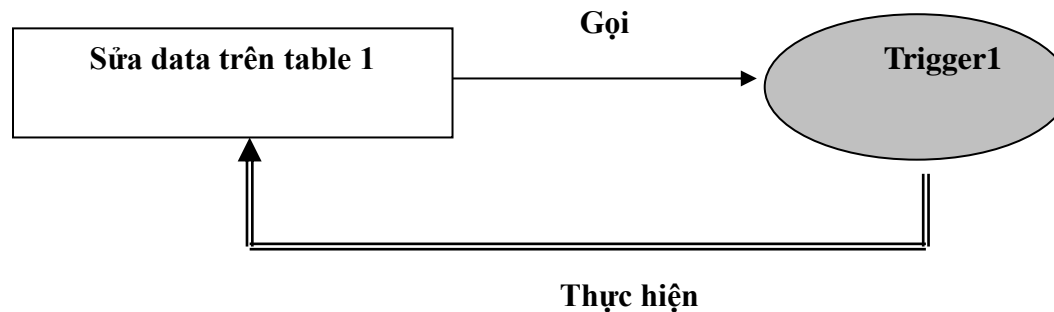
- Thao tác của một trigger kéo theo việc thi hành một trigger khác, các trigger này được gọi là **trigger lồng nhau** (Nested Trigger)
- Có thể lồng tối đa 32 cấp.
- Các trigger được xem như một đơn vị thi hành (một transaction). Do vậy, một trigger trong dãy trigger lồng nhau bị lỗi, thì SQL Server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





# Chuỗi các trigger

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



# Chuỗi các trigger

## Ví dụ

**CREATE DATABASE Showroom**

**GO**

Use Showroom

**CREATE TABLE Car**

(

CarId int identity(1,1) primary key,

Name varchar(100),

Make varchar(100),

Model int ,

Price int ,

Type varchar(20)

)

**CREATE TABLE CarLog**

(

LogId int identity(1,1) primary key,

CarId int ,

CarName varchar(100),

)

**Yêu cầu :** Khi data được nhập vào Car table thì cũng được nhập tự động vào CarLog table . Data không được nhập trực tiếp vào CarLog table

⇒ tạo 2 trigger

⇒ Sử dụng biến @@NESTLEVEL để xác định nguồn gọi trigger

Nếu trigger được gọi trực tiếp thì

@@NESTLEVEL= 1

Nếu trigger được gọi từ một trigger khác thì

@@NESTLEVEL= 2

....

# Chuỗi các trigger

## Ví dụ

```
CREATE TRIGGER [dbo].[CAR_INSERT]
    ON [dbo].[Car]
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @car_id INT, @car_name VARCHAR(50)

    SELECT @car_id = INSERTED.CarId, @car_name = INSERTED.name
    FROM INSERTED

    INSERT INTO CarLog
    VALUES (@car_id, @car_name)
END
```

# Chuỗi các trigger

## Ví dụ

```
CREATE TRIGGER [dbo].[CarLOG_INSERT] ON [dbo].[CarLog]
INSTEAD OF INSERT
AS
BEGIN
    IF @@NESTLEVEL = 1
        PRINT('DATA CANNOT BE INSERTED DIRECTLY IN CarLog TABLE')
    ELSE
        BEGIN
            DECLARE @car_id INT, @car_name VARCHAR(50)

            SELECT @car_id = INSERTED.CarId, @car_name = INSERTED.CarName
            FROM INSERTED
            INSERT INTO CarLog
            VALUES(@car_id, @car_name)
        END
END
```

# Lệnh hiệu chỉnh và xóa trigger

- Hiệu chỉnh

**ALTER TRIGGER Trigger\_Name**

...

- Xóa trigger:

**DROP TRIGGER Trigger\_Name**

- Vô hiệu hóa tất cả các trigger

**ALTER TABLE table\_name**

**DISABLE TRIGGER ALL**

# Liệt kê các trigger

- Liệt kê các trigger đã định nghĩa trên table/view
  - Sử dụng SSMS
  - Dùng lệnh

```
SELECT t.name AS TableName, tr.name AS TriggerName
FROM sys.triggers tr INNER JOIN sys.tables t
ON t.object_id = tr.parent_id
WHERE t.name in ('Table_name')
```

# Các lưu ý khi tạo và sử dụng trigger

<b>For/After</b>	<b>Instead of</b>
- Chỉ áp dụng cho table	- Áp dụng cho table, view Không cho phép áp dụng với các View có lựa chọn With Check Option
- 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 : + Xử lý ràng buộc + Thực hiện xong hành động I/U/D phát sinh trigger	- Thi hành trước khi: + Xử lý ràng buộc + Thay thế hành động phát sinh trigger
	- Không xây dựng được trên table có áp dụng FK cascade D/U cho cùng một action
Table inserted và deleted Không cho phép column có kiểu dữ liệu text, ntext, image	Table inserted và deleted Cho phép column có kiểu dữ liệu text, ntext, image

# Các lưu ý khi tạo và sử dụng trigger

- Một trigger Không thể định nghĩa trên nhiều hơn 1 table/view
- Không gọi trực tiếp hoặc truyền nhận tham số đối với trigger
- Trigger được định nghĩa trên 1 table cụ thể, nhưng không thể tạo trigger trên temporary hay system table.
- Được sử dụng hầu hết các phát biểu T\_SQL trong trigger All CREATE, ALTER, DROP, GRANT, REVOKE, DENY ,LOAD, RESTORE ,RECONFIGURE TRUNCATE TABLE ,UPDATE STATISTICS, SELECT INTO
- Không nên dùng quá nhiều trigger trong một table



# Các lưu ý khi tạo và sử dụng trigger

- Trigger không thể chứa các lệnh

- ▶ ALTER DATABASE
- ▶ CREATE DATABASE
- ▶ DISK RESIZE
- ▶ DROP DATABASE
- ▶ LOAD DATABASE and LOAD LOG
- ▶ RECONFIGURE
- ▶ RESTORE DATABASE and RESTORE LOG

# Tóm tắt