



# CÁC KỸ THUẬT NÂNG CAO DML và DDL Triggers

Nguyễn Minh Trung



- DML Trigger
- DDL Trigger
- Tương tác nhiều Trigger
- Thao tác dữ liệu lớn - Bulk



- ☐ DML Trigger
- ☐ DDL Trigger

- **DML Trigger**
- DDL Trigger
- Tương tác nhiều Trigger
- Thao tác dữ liệu lớn - Bulk



# Cơ bản về Trigger – khái niệm

- ☐ DML Trigger
- ☐ DDL Trigger

- Hai kỹ thuật chính đảm bảo các qui tắc thương mại và ràng buộc dữ liệu:
  - các ràng buộc (constraints)
  - các bẫy sự kiện (triggers)
- Trigger
  - trường hợp đặc biệt của thủ tục được buộc chặt tới các sự kiện của bảng (table)
  - một đối tượng cơ sở dữ liệu
    - có thể chứa các mã lệnh T-SQL được thực thi tự động



CANTHO UNIVERSITY

# Cơ bản về Trigger – khái niệm

- ☐ DML Trigger
- ☐ DDL Trigger

- nó chỉ khởi động đáp lại các sự kiện INSERT, UPDATE hoặc DELETE trên table
- các sự kiện ngôn ngữ định nghĩa dữ liệu (DDL) xảy ra trong server hay cơ sở dữ liệu.
- SQL Server bao gồm 3 kiểu trigger:
  - DML triggers: thao tác dữ liệu
  - DDL triggers: định nghĩa dữ liệu
  - logon triggers: đáp lại sự kiện login



# Cơ bản về Trigger – các kiểu trigger

- ☐ DML Trigger
- ☐ DDL Trigger

- SQL Server có hai kiểu giao tác trigger:  
*instead of trigger* và *after trigger*
  - *instead of trigger*
    - xác định các hành động của sự kiện kích hoạt trigger (INSERT, UPDATE, or DELETE) sẽ bị bỏ qua và
    - thay vào đó là các lệnh trong trigger được thực hiện trước.
    - INSTEAD OF trigger được định nghĩa trên đối tượng Table và View với một hoặc nhiều bảng
  - *After trigger*:
    - được thực hiện sau khi các hành động của câu lệnh INSERT, UPDATE hoặc DELETE được thực hiện.
    - từ khóa AFTER giống như từ khóa FOR của các phiên bản trước và
    - AFTER trigger chỉ được xác định trên các đối tượng Tables



CANTHO UNIVERSITY

# Cơ bản về Trigger – các kiểu trigger

- ☐ DML Trigger
- ☐ DDL Trigger

Chức năng	AFTER trigger	INSTEAD OF trigger
Khả năng áp dụng	Tables	Tables and views
Số lượng có thể cho mỗi table hoặc view	Nhiều cho mỗi kích hoạt thao tác (Update, Delete và Insert)	Một cho mỗi kích hoạt (Update, Delete và Insert)
Tham chiếu tác động dây chuyền	Áp dụng không giới hạn	Instead of Update và Delete trigger thì không được phép trên các bảng là các bảng đích của tham chiếu tác động dây chuyền các ràng buộc toàn vẹn
Thi hành	Sau: <ul style="list-style-type: none"><li>• Xử lý ràng buộc</li><li>• Khai báo các hành động tham chiếu</li><li>• Tạo dựng bảng inserted và deleted</li><li>• Các thao tác trigger</li></ul>	Before: <ul style="list-style-type: none"><li>• Xử lý ràng buộc</li></ul> Thay cho <ul style="list-style-type: none"><li>• Các thao tác trigger</li></ul> Sau: <ul style="list-style-type: none"><li>• Tạo dựng bảng inserted và deleted</li></ul>
Thứ tự thi hành	Thi hành đầu tiên và sau cùng được xác định	Không áp dụng
Các tham chiếu cột varchar(max), nvarchar(max), và varbinary(max) trong bảng inserted and deleted	Được phép	Được phép
Các tham chiếu cột text, ntext, and image trong bảng inserted and deleted	Không được phép	Được phép



CANTHO UNIVERSITY

# Cơ bản về Trigger – khi nào sử dụng trigger

- ☐ DML Trigger
- ☐ DDL Trigger

- Trigger hữu ích
  - cho việc kiểm tra các thay đổi dữ liệu hoặc
  - kiểm tra cơ sở dữ liệu cũng như các quy tắc kinh doanh (business rules)
  - các biện pháp bảo đảm tính toàn vẹn dữ liệu như các ràng buộc miền giá trị, khóa chính, khóa ngoại, ..., không thỏa mãn nhu cầu của ứng dụng
  - kiểm tra các thao tác dữ liệu có thỏa với các business rules hay không
    - vì vậy tránh đi những mục nhập sai lầm trong bảng

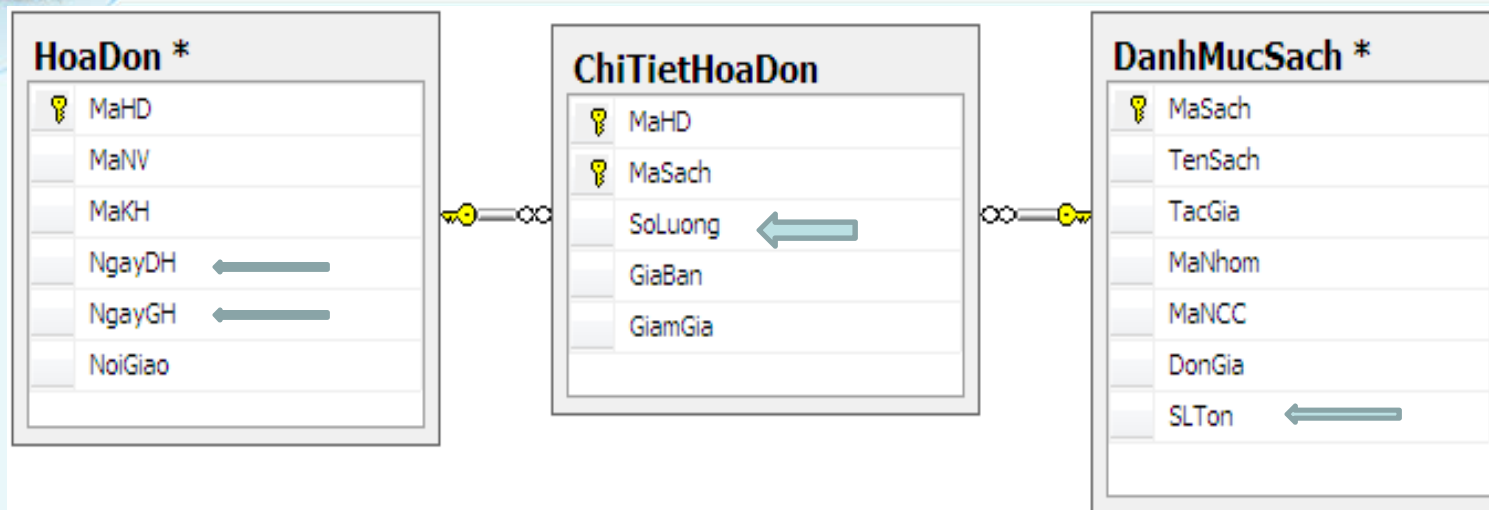




CANTHO UNIVERSITY

# Cơ bản về Trigger – khi nào sử dụng trigger

- ☐ DML Trigger
- ☐ DDL Trigger



- Trong bảng **HoaDon** cần một ràng buộc toàn vẹn liên thuộc tính là trong một hóa đơn **NgayDH** phải nhỏ hơn bằng **NgayGH**
- Trong bảng **ChiTietHoaDon**, nếu có sự thay đổi cột **SoLuong** của một mã sách nào thì cũng cần cập nhật lại cột **SLTon** trong bảng **DanhMucSach**
- Khi một danh mục sách nào đó được bán đi với một số lượng (được thêm vào trong bảng **ChiTietHoaDon**) thì **SLTon** trong **DanhMucSach** phải tự động giảm đi với số lượng tương ứng. Ngoài ra, trước khi muốn bán danh mục sách nào đó, một kiểm tra phải được thực hiện là **SLTon** của danh mục sách đó có đáp ứng với số lượng yêu cầu.



CANTHO UNIVERSITY

# Cơ bản về Trigger – tạo trigger

- ☐ DML Trigger
- ☐ DDL Trigger

Cú pháp như sau:

```
CREATE TRIGGER [owner.]trigger_name  
ON [owner.] table_name  
{ FOR | AFTER | INSTEAD OF }  
{[INSERT][,][UPDATE][,][DELETE]}  
AS  
  
[IF UPDATE(column_name)  
[AND UPDATE(column_name)| OR  
UPDATE(column_name)] ...]  
{ sql_statements }
```



CANTHO UNIVERSITY

# Cơ bản về Trigger – tạo trigger

- ☐ DML Trigger
- ☐ DDL Trigger

Chú ý:

- Câu lệnh DML sử dụng hai bảng đặc biệt
  - INSERTED
  - DELETED
  - SQL Server tự động tạo và quản lý các bảng này
  - Cấu trúc của hai bảng này tương tự cấu trúc của bảng mà DML trigger tác động
  - Dữ liệu trong hai bảng này tùy thuộc vào câu lệnh tác động lên bảng làm kích hoạt trigger;



# Cơ bản về Trigger – tạo trigger

- ☐ DML Trigger
- ☐ DDL Trigger

- Bảng DELETED chứa bản sao
  - các dòng bị ảnh hưởng của các lệnh Delete và Update.
  - bảng INSERTED trong trường hợp này không có dữ liệu bởi lệnh Delete
- Bảng INSERTED chứa bản sao
  - các dòng bị ảnh hưởng bởi các lệnh Insert và Update.
  - bảng DELETED trong trường hợp này không có dữ liệu bởi lệnh Insert
- Khi câu lệnh UPDATE được thực thi trên bảng
  - giống như thao tác Delete theo sau bởi thao tác insert,
  - các dòng dữ liệu cũ được sao chép vào bảng DELETED,
  - còn trong bảng INSERTED sẽ là các dòng sau khi đã được cập nhật



CANTHO UNIVERSITY

# Cơ bản về Trigger – tạo trigger

- ☐ DML Trigger
- ☐ DDL Trigger

Ví dụ: Tạo trigger kiểm tra mã số khách hàng khi thêm vào một khách hàng

```
CREATE Trigger t_Kiemtra
On Khachhang
INSTEAD OF INSERT
AS
BEGIN
    Declare @MSKH int
    Select @MSKH=Count(*)
    From Khachhang KH, Inserted IT
    Where KH.MSKH = IT.MSKH
    If @MSKH > 0
        Print N'Mã số khách hàng đã có trong bảng khách hàng'
    ELSE
        Begin
            Insert into KhachHang
            Select * From Inserted
        End
END
```

```
Insert Into Khachhang
Values('KH005', N'Lê Bá Nhân',
      N'Sao Mai', N'35 Hung Vuong',
      '071.123356','0710.3883922')
```



## Cơ bản về Trigger – Kích hoạt trigger dựa trên sự thay đổi dữ liệu trên cột

- ☐ DML Trigger
- ☐ DDL Trigger

- Mệnh đề IF UPDATE trong trigger được sử dụng
  - chỉ định trigger được kích hoạt và
  - thực hiện những thao tác cụ thể khi việc thay đổi dữ liệu chỉ liên quan đến một số cột nhất định
  - để xác định có câu lệnh INSERT hoặc UPDATE làm ảnh hưởng một cột xác định trong bảng hay không
- Chú ý: hàm COLUMNS\_UPDATED() để kiểm tra các cột trong một bảng được cập nhật bởi
  - Update hoặc
  - Insert



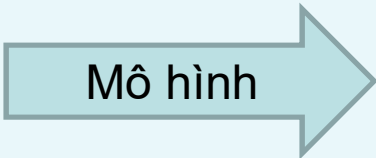
CANTHO UNIVERSITY

## Cơ bản về Trigger – Kích hoạt trigger dựa trên sự thay đổi dữ liệu trên cột

- ☐ DML Trigger
- ☐ DDL Trigger

Ví dụ: Thay đổi cột số lượng trong ChiTietDatHang, khi đó chúng ta cũng cần thay đổi số lượng hàng bên bảng HangHoa. Sự thay đổi này có thể xảy ra trên nhiều hàng của cột số lượng

```
CREATE TRIGGER t_ChiTietDatHang_update_soluong
ON ChiTietDatHang
FOR UPDATE
AS
BEGIN
```



Mô hình

```
    IF UPDATE(soluong)
        UPDATE HangHoa
        SET HangHoa.SoLuongHang = HangHoa.SoLuongHang -
            (SELECT sum(Inserted.Soluong - Deleted.Soluong)
             FROM (Deleted inner join Inserted ON Deleted.SoDonDH = Inserted.SoDonDH
                  AND Deleted.MSHH = Inserted.MSHH)
             WHERE HangHoa.MSHH= deleted.MSHH)
        WHERE HangHoa.MSHH in (select Inserted.MSHH from Inserted)
END
```



CANTHO UNIVERSITY

# Cơ bản về Trigger – ví dụ Trigger

- ☐ DML Trigger
- ☐ DDL Trigger

- Tạo bảng *Employee\_Test*

```
CREATE TABLE Employee_Test
```

```
(  
Emp_ID INT Identity,
```

```
Emp_name Varchar(100),
```

```
Emp_Sal Decimal (10,2)
```

```
)
```

- Chèn dữ liệu vào bảng *Employee\_Test*

```
INSERT INTO Employee_Test VALUES ('Anees',1000);
```

```
INSERT INTO Employee_Test VALUES ('Rick',1200);
```

```
INSERT INTO Employee_Test VALUES ('John',1100);
```

```
INSERT INTO Employee_Test VALUES ('Stephen',1300);
```

```
INSERT INTO Employee_Test VALUES ('Maria',1400);
```





CANTHO UNIVERSITY

# Cơ bản về Trigger – ví dụ Trigger

- ☐ DML Trigger
- ☐ DDL Trigger

- - Tạo bảng *Employee\_Test\_Audit*  

```
CREATE TABLE Employee_Test_Audit  
(  
  Emp_ID int,  
  Emp_name varchar(100),  
  Emp_Sal decimal (10,2),  
  Audit_Action varchar(100),  
  Audit_Timestamp datetime  
)
```



CANTHO UNIVERSITY

# Cơ bản về Trigger – ví dụ Trigger

- ☐ DML Trigger
- ☐ DDL Trigger

## Employee\_Test\_Audit

Emp_ID
Emp_name
Emp_Sal
Audit_Action
Audit_Timestamp

## Employee\_Test

Emp_ID
Emp_name
Emp_Sal



CANTHO UNIVERSITY

## ví dụ Trigger - AFTER INSERT TRIGGER

- ☐ DML Trigger
- ☐ DDL Trigger

Trigger này được kích hoạt sau khi có một lệnh INSERT trên bảng **Employee\_Test** và sau đó chèn dữ liệu vào bảng *Employee\_Test\_Audit*.

```
CREATE TRIGGER trgAfterInsert ON [dbo].[Employee_Test]
FOR INSERT
AS
BEGIN
declare @empid int; declare @empname varchar(100);
declare @empsal decimal(10,2); declare @audit_action varchar(100);
SELECT @empid=i.Emp_ID FROM inserted i;
SELECT @empname=i.Emp_Name FROM inserted i;
SELECT @empsal=i.Emp_Sal FROM inserted i;
set @audit_action='Inserted Record -- After Insert Trigger.';
INSERT into Employee_Test_Audit(Emp_ID,Emp_Name,Emp_Sal,Audit_Action,Audit_Timestamp)
VALUES(@empid,@empname,@empsal,@audit_action,getdate());
PRINT 'AFTER INSERT trigger fired.'
END
```



CANTHO UNIVERSITY

## ví dụ Trigger - AFTER UPDATE TRIGGER

- ☐ DML Trigger
- ☐ DDL Trigger

Trigger này kiểm tra có thay không cột Emp\_Name hoặc cột Emp\_Sal cập nhật dữ liệu trong bảng *Employee\_Test*

```
CREATE TRIGGER trgAfterUpdate ON [dbo].[Employee_Test]
FOR UPDATE
AS
BEGIN
declare @empid int; declare @empname varchar(100);
declare @empsal decimal(10,2); declare @audit_action varchar(100);
SELECT @empid=i.Emp_ID FROM inserted i;
SELECT @empname=i.Emp_Name FROM inserted i;
SELECT @empsal=i.Emp_Sal FROM inserted i;
IF update(Emp_Name) set @audit_action='Updated Record -- After Update Trigger.';
IF update(Emp_Sal) set @audit_action='Updated Record -- After Update Trigger.';
INSERT INTO Employee_Test_Audit(Emp_ID,Emp_Name,Emp_Sal,Audit_Action,Audit_Timestamp)
VALUES(@empid,@empname,@empsal,@audit_action,getdate());
PRINT 'AFTER UPDATE Trigger fired.'
END
```



CANTHO UNIVERSITY

## ví dụ Trigger - AFTER DELETE TRIGGER

- ☐ DML Trigger
- ☐ DDL Trigger

Trigger này xóa dữ liệu trong bảng *Employee\_Test* và chèn vào bảng *Employee\_Test\_Audit*

```
CREATE TRIGGER trgAfterDelete ON [dbo].[Employee_Test]
AFTER DELETE
AS
BEGIN
declare @empid int; declare @empname varchar(100);
declare @empsal decimal(10,2); declare @audit_action varchar(100);
SELECT @empid=d.Emp_ID FROM deleted d;
SELECT @empname=d.Emp_Name FROM deleted d;
SELECT @empsal=d.Emp_Sal FROM deleted d;
SET @audit_action='Deleted -- After Delete Trigger.';
INSERT INTO Employee_Test_Audit(Emp_ID, Emp_Name, Emp_Sal, Audit_Action, Audit_Timestamp)
VALUES(@empid,@empname,@empsal,@audit_action,getdate());
PRINT 'AFTER DELETE TRIGGER fired.'
END
```



CANTHO UNIVERSITY

## ví dụ Trigger - INSTEAD OF TRIGGER

- ☐ DML Trigger
- ☐ DDL Trigger

Ví dụ: trigger sau bỏ qua thao tác xóa

```
CREATE TRIGGER trg_Test_InsteadOfDelete ON [dbo].[Employee_Test]  
INSTEAD OF DELETE
```

```
AS
```

```
BEGIN
```

```
PRINT 'No Record on Employee_Test Deleted -- Instead Of Delete Trigger.'
```

```
END
```



CANTHO UNIVERSITY

## ví dụ Trigger - INSTEAD OF TRIGGER

- ☐ DML Trigger
- ☐ DDL Trigger

Ví dụ: Trigger này sẽ ngăn chặn xóa mẫu tin từ bảng Employee\_Test khi **Emp\_Sal >1200**. Ngược lại, mẫu tin sẽ bị xóa và chèn mẫu tin bị xóa vào bảng Employee\_Test\_Audit

```
CREATE TRIGGER trgInsteadOfDelete ON [dbo].[Employee_Test]
```

```
Instead of DELETE
```

```
AS
```

```
declare @emp_id int; declare @emp_name varchar(100);
```

```
declare @emp_sal int; select @emp_id=d.Emp_ID from deleted d;
```

```
select @emp_name=d.Emp_Name from deleted d;
```

```
select @emp_sal=d.Emp_Sal from deleted d;
```

```
BEGIN
```

```
if(@emp_sal>1200)
```

```
begin RAISERROR('Cannot delete where salary > 1200',16,1); ROLLBACK; end
```

```
else
```

```
begin
```

```
delete from Employee_Test where Emp_ID=@emp_id; COMMIT;
```

```
insert into Employee_Test_Audit(Emp_ID, Emp_Name, Emp_Sal, Audit_Action, Audit_Timestamp)
```

```
values(@emp_id,@emp_name,@emp_sal, 'Deleted -- Instead Of Delete Trigger.', getdate());
```

```
PRINT 'Record Deleted -- Instead Of Delete Trigger.' end
```

```
END
```



## Cơ bản về Trigger – sửa đổi trigger

- ☐ DML Trigger
- ☐ DDL Trigger

- Cú pháp sửa đổi trigger

```
ALTER TRIGGER [owner.]trigger_name
```

```
ON [owner.] table_name
```

```
FOR {[INSERT][,][UPDATE][,][DELETE]}
```

```
AS
```

```
[IF UPDATE(column_name)
```

```
[AND UPDATE(column_name)|OR UPDATE(column_name)]
```

```
...]
```

```
{ sql_statements }
```





## Cơ bản về Trigger – Kích hoạt / tắt và xóa Trigger

- ☐ DML Trigger
- ☐ DDL Trigger

- Cú pháp kích hoạt / tắt trigger  
ALTER TABLE [database\_name.[schema\_name].|  
schema\_name.] table\_name  
  
                  {ENABLE| DISABLE } TRIGGER { ALL |  
trigger\_name [ ,...n ] }
- Cú pháp xóa trigger  
DROP TRIGGER schema\_name.trigger\_name [ ,...n ] [ ; ]



CANTHO UNIVERSITY

## Cơ bản về Trigger – Giới hạn DML Trigger

- ☐ DML Trigger
- ☐ DDL Trigger

- Một số câu lệnh không được phép bên trong một trigger:
  - CREATE, ALTER, hoặc DROP cơ ở dữ liệu
  - RECONFIGURE
  - RESTORE database hoặc log
  - DISK RESIZE
  - DISK INIT



- ☐ DML Trigger
- ☐ **DDL Trigger**

- DML Trigger
- **DDL Trigger**
- Tương tác nhiều Trigger
- Thao tác dữ liệu lớn - Bulk



# DDL Trigger – Khái niệm

- ☐ DML Trigger
- ☐ DDL Trigger

- DDL trigger kích hoạt đáp ứng lại khi các sự kiện diễn ra trên
  - server
  - cơ sở dữ liệu
    - bởi các câu lệnh CREATE, ALTER, DROP, GRANT, DENY, REVOKE
  - còn DDL đáp lại sự thay đổi lược đồ



# DDL Trigger – Khái niệm

- ☐ DML Trigger
- ☐ DDL Trigger

- DDL trigger kích hoạt đáp ứng lại khi các sự kiện diễn ra trên
  - server
  - cơ sở dữ liệu
    - bởi các câu lệnh CREATE, ALTER, DROP, GRANT, DENY, REVOKE
  - còn DDL đáp lại sự thay đổi lược đồ
- DDL trigger được lưu trữ
  - trong CSDL mà DDL trigger được gắn vào.
  - với các Server DDL Trigger theo dõi các thay đổi ở cấp độ Server



CANTHO UNIVERSITY

# DDL Trigger – Tạo DDL trigger

- ☐ DML Trigger
- ☐ DDL Trigger

Cú pháp:

```
CREATE TRIGGER trigger_name  
ON { ALL SERVER | DATABASE }  
[ WITH < ENCRYPTION > ]  
{ FOR | AFTER } { event_type | event_group } [ ,...n ]  
AS  
{ sql_statement [ ; ] [ ,...n ] }
```

Trong đó:

- event\_type: là tên của một sự kiện ngôn ngữ T-SQL sau khi thực hiện gây ra kích hoạt một DDL trigger như **CREATE\_TABLE**, **ALTER\_TABLE**, **DROP\_TABLE**...



CANTHO UNIVERSITY

# DDL Trigger – Tạo DDL trigger

- ☐ DML Trigger
- ☒ DDL Trigger

- Ví dụ: Câu lệnh dưới đây xây dựng một trigger được kích hoạt khi xảy ra các sự kiện ở cấp độ CSDL. Trigger này sẽ ngăn chặn các lệnh DROP TABLE và ALTER TABLE.

Create Trigger t\_Safety

On Database

For **ALTER\_TABLE**, DROP\_TABLE

As

Begin

Print N'Phải xóa trigger t\_Safety trước khi ALTER hay DROP bảng'

Rollback tran

End



CANTHO UNIVERSITY

# DDL Trigger – Tắt/kích hoạt và xóa trigger

- ☐ DML Trigger
- ☒ DDL Trigger

- Tắt/ kích hoạt Trigger

DISABLE TRIGGER tên\_trigger

ON { tên\_đối\_tượng | DATABASE | SERVER }

- Xóa Trigger

DROP TRIGGER trigger\_name [ ,...n ]

ON { DATABASE | ALL SERVER }





- ☐ DML Trigger
- ☐ **DDL Trigger**
- ☐ **Tương tác nhiều Trigger**

- DML Trigger
- DDL Trigger
- **Tương tác nhiều Trigger**
- Thao tác dữ liệu lớn - Bulk



CANTHO UNIVERSITY

# Tương tác nhiều Trigger – khái niệm

- ☐ DML Trigger
- ☐ DDL Trigger

- Nếu không có một kế hoạch rõ ràng, một cơ sở dữ liệu với nhiều Trigger có thể
  - gây ra không tổ chức và
  - cực kỳ khó khăn để khắc phục sự cố
- SQL Server 6.5,
  - mỗi sự kiện trigger chỉ có thể một trigger và
  - một trigger chỉ áp dụng tới một sự kiện trigger
- SQL Server 7.0,
  - SQL Server cho phép nhiều AFTER trigger trên sự kiện bảng và
  - một trigger có thể áp dụng tới nhiều hơn một sự kiện



Version SQL



CANTHO UNIVERSITY

# Tương tác nhiều Trigger – trigger lồng nhau

- ☐ DML Trigger
- ☐ DDL Trigger

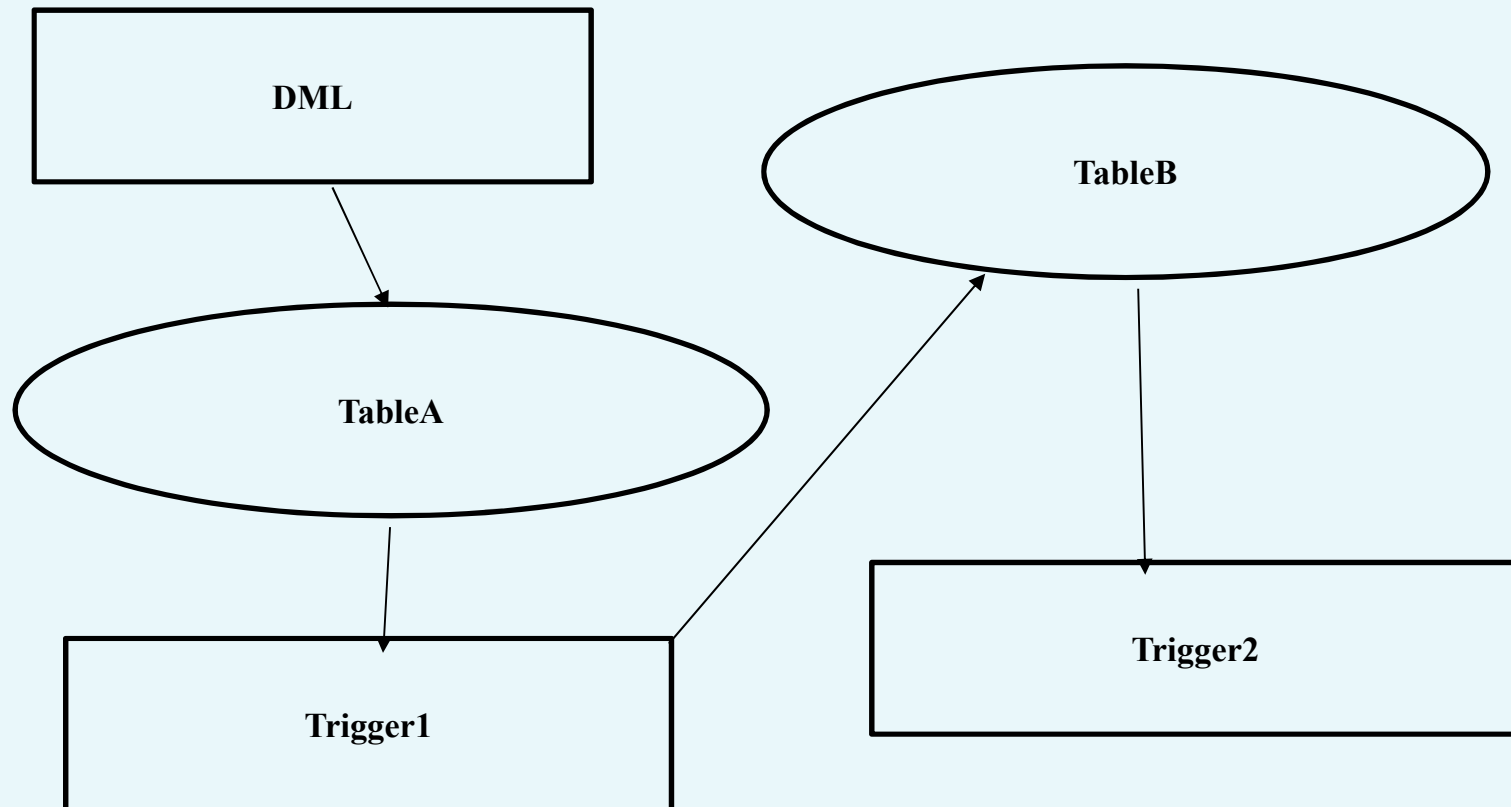
- Một Trigger được gọi là lồng nhau khi
  - trigger thực hiện một hành động mà hành động này khởi xướng một Trigger khác
  - ví dụ, chèn một mẫu tin vào trong một bảng TableA
    - gây nên trigger chèn mẫu tin của TableA được kích hoạt.
    - trigger chèn mẫu tin của bảng TableA tiếp tục cập nhật một mẫu tin trong TableB,
    - gây ra Trigger cập nhật của TableB kích hoạt
- Trigger lồng nhau thường được sử dụng
  - cho trường hợp các thay đổi tham chiếu tác động dây truyền qua các bảng quan hệ logic phức tạp
- Cấp độ lồng tối đa của các trigger không vượt quá 32 cấp và có thể dùng hàm hệ thống @@NESTLEVEL để biết cấp độ hiện hành của Trigger



CANTHO UNIVERSITY

# Tương tác nhiều Trigger – trigger lồng nhau

- ☐ DML Trigger
- ☐ DDL Trigger



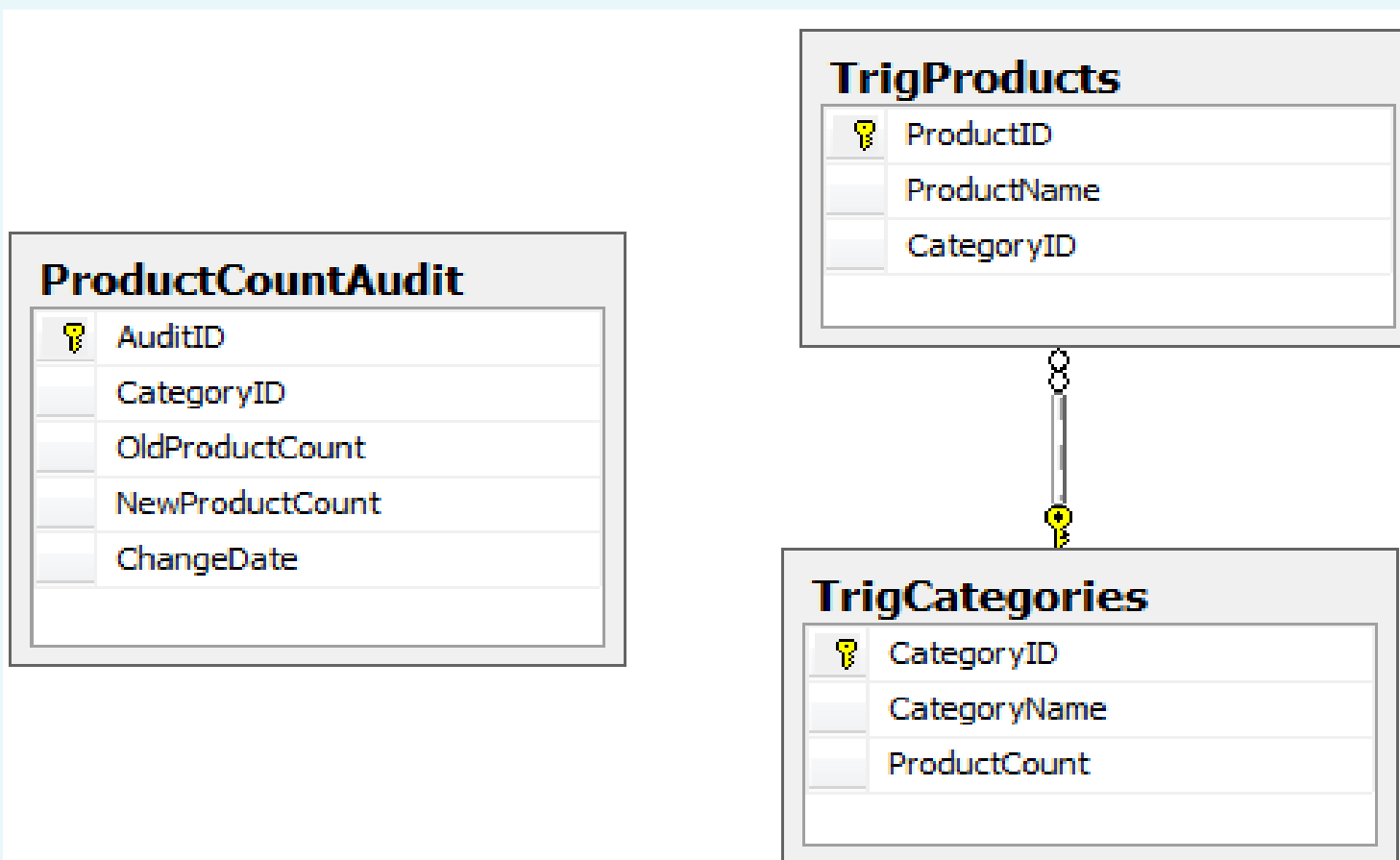


CANTHO UNIVERSITY

# Tương tác nhiều Trigger – trigger lồng nhau

- ☐ DML Trigger
- ☐ DDL Trigger

- Ví dụ: trigger lồng nhau, giả sử có lược đồ sau:





CANTHO UNIVERSITY

# Tương tác nhiều Trigger – trigger lồng nhau

- ☐ DML Trigger
- ☐ DDL Trigger

- Tạo một trigger **trg\_TrigProduct\_Insert** trên bảng **TrigProducts** để cập nhật trường **ProductCount** của bảng **TrigCategories** mỗi khi có một mẫu tin chèn vào bảng **TrigProducts**

```
CREATE TRIGGER trg_TrigProduct_Insert
ON TrigProducts
AFTER INSERT
AS
SET NOCOUNT ON
UPDATE c
SET ProductCount = ProductCount +
(SELECT COUNT(*)
FROM Inserted
WHERE CategoryID = c.CategoryID)
FROM TrigCategories c JOIN Inserted i
ON c.CategoryID = i.CategoryID
```

- Tạo **trg\_Categories\_ProductCountAudit** được tạo trên bảng **TrigCategories** để kiểm tra sự thay đổi của trường **ProductCount** trong bảng **TrigCategories**

```
CREATE TRIGGER
trg_Categories_ProductCountAudit
ON TrigCategories
FOR UPDATE
AS
IF UPDATE (ProductCount)
BEGIN
INSERT INTO ProductCountAudit
(CategoryID, OldProductCount, NewProductCount)
SELECT i.CategoryID, d.ProductCount,
i.ProductCount
FROM Deleted d JOIN Inserted i
ON d.CategoryID = i.CategoryID;
END
```

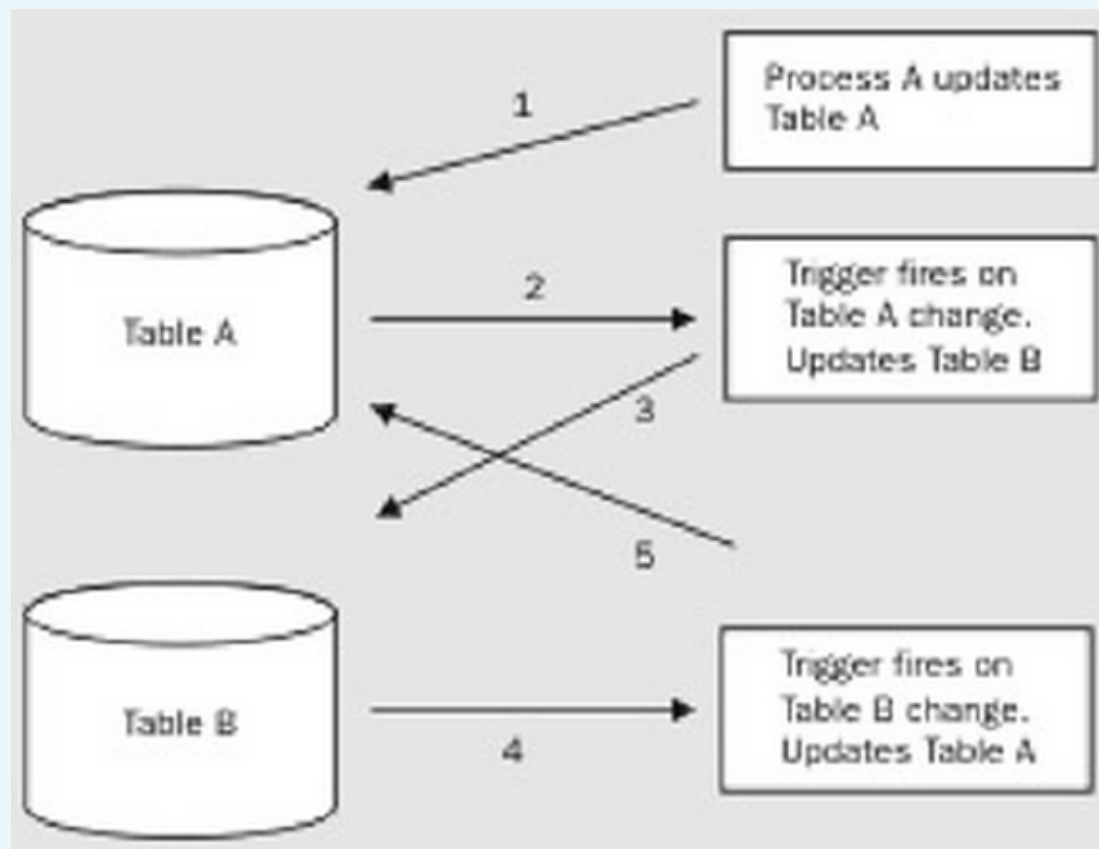


CANTHO UNIVERSITY

# Tương tác nhiều Trigger – trigger lồng nhau

- ☐ DML Trigger
- ☐ DDL Trigger

- **Chú ý:** trigger lồng nhau có thể dẫn đến vòng lặp vô tận,
  - khi đó cấp độ lồng của trigger vượt quá 32 và
  - kết thúc cuối cùng là một lỗi được tạo bởi SQL Server.



- Một thủ tục A cập nhật bảng TableA
- Trigger trên bảng A được kích hoạt
- Trigger được định nghĩa trên TableA cập nhật Table B
- Bảng TableB có một trigger kích hoạt
- Trigger trên bảng TableB cập nhật TableA



CANTHO UNIVERSITY

# Tương tác nhiều Trigger – trigger đệ qui

- ☐ DML Trigger
- ☐ DDL Trigger

- Một Trigger đệ qui
  - một trigger có thể chính nó gây ra một kích hoạt
  - Có hai loại đệ qui
    - đệ qui trực tiếp (direct recursion) hoặc
    - đệ qui không trực tiếp (indirect recursion)





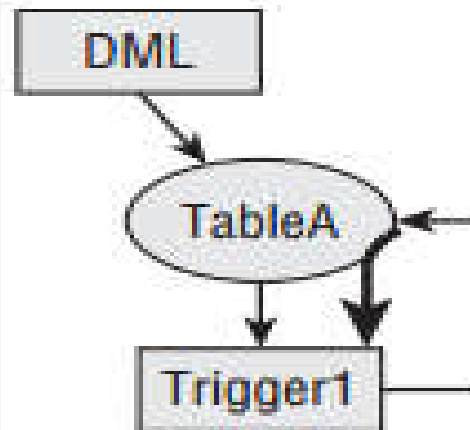
CANTHO UNIVERSITY

# Tương tác nhiều Trigger – trigger đệ qui

- ☐ DML Trigger
- ☐ DDL Trigger

- Một Trigger đệ qui trực tiếp
  - xảy ra khi một trigger kích hoạt và thực hiện một hành động
  - mà hành động này gây ra chính trigger đó kích hoạt lại

## Recursive Triggers





CANTHO UNIVERSITY

# Tương tác nhiều Trigger – trigger đệ qui

- ☐ DML Trigger
- ☐ DDL Trigger

- **Đệ** qui không trực tiếp (indirect recursion)
  - xảy ra khi một trigger kích hoạt và thực hiện một hành động
  - mà hành động này gây ra trigger khác (có thể trên cùng bảng hoặc khác bảng) kích hoạt
  - trigger thứ hai này thực hiện một hành động mà hành động này gây ra trigger ban đầu (original trigger) kích hoạt lại
- Mặc nhiên đệ qui trực tiếp (direct recursion) là tắt (disabled)

**ALTER DATABASE DatabaseName SET RECURSIVE\_TRIGGERS ON | OFF**




CANTHO UNIVERSITY

# Tương tác nhiều Trigger – trigger đệ qui

- ☐ DML Trigger
- ☐ DDL Trigger

- Tạo bảng Product

```
CREATE TABLE [dbo].[Product] (  
    [ProductID] uniqueidentifier Default NEWID() PRIMARY KEY,  
    [ProductName] [nvarchar](50) NOT NULL,  
    [Code] char(15) NOT NULL,  
    Created SmallDateTime NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    Modified SmallDateTime NOT NULL DEFAULT CURRENT_TIMESTAMP)
```

Product	
	ProductID
	ProductName
	Code
	Created
	Modified

Tạo một Trigger đệ qui Trig\_ModifiedDate trên bảng Product. Trigger này ghi ngày và thời gian hiện tại lên **cột modified** cho bất kỳ hàng nào được cập nhật trên bảng Product



CANTHO UNIVERSITY

# Tương tác nhiều Trigger – trigger đệ qui

- ☐ DML Trigger
- ☐ DDL Trigger

```
CREATE TRIGGER Trig_ModifiedDate ON dbo.Product
AFTER UPDATE
AS
BEGIN
IF @@ROWCOUNT = 0
    RETURN;
If Trigger_NestLevel() > 1
    RETURN;
SET NOCOUNT ON;
PRINT  N'Số lần: ' + CAST(TRIGGER_NESTLEVEL() AS NVARCHAR(20));
If (UPDATE(Created) or UPDATE(Modified))
    Begin
        Raiserror('Update failed.', 16, 1)
        ROLLBACK;    Return
    End
UPDATE Product ----- Update the Modified date
SET Modified = CURRENT_TIMESTAMP
WHERE EXISTS (SELECT * FROM Inserted AS I WHERE i.ProductID = Product.ProductID);
END
```



- DML Trigger
- DDL Trigger
- Tương tác nhiều Trigger
- **Thao tác dữ liệu lớn - Bulk**



CANTHO UNIVERSITY

- **Thao tác dữ liệu lớn – Bulk**

**Xem tài liệu trong sách (trang 70)**



# Cám ơn






### SQL Server Release History

Version	Year	Release Name	Codename
1.0 (OS/2)	1989	SQL Server 1.0 (16bit)	-
1.1 (OS/2)	1991	SQL Server 1.1 (16bit)	-
4.21 (WinNT)	1993	SQL Server 4.21	SQLNT
6.0	1995	SQL Server 6.0	SQL95
6.5	1996	SQL Server 6.5	Hydra
7.0	1998	SQL Server 7.0	Sphinx
-	1999	SQL Server 7.0 OLAP Tools	Palato mania
8.0	2000	SQL Server 2000	Shiloh
8.0	2003	SQL Server 2000 64-bit Edition	Liberty
9.0	2005	SQL Server 2005	Yukon
10.0	2008	SQL Server 2008	Katmai
10.25	2010	SQL Azure DB	CloudDB
10.5	2010	SQL Server 2008 R2	Kilimanjaro (aka KJ)
11.0	2012	SQL Server 2012	Denali







ChiTietDatHang	
	SoDonDH
	MSHH
	SoLuong
	GiaDatHang
	GiamGia

HangHoa	
	Column Name
	MSHH
	TenHH
	QuyCach
	Gia
	SoLuongHang
	GhiChu

KhachHang	
	MSKH
	HoTenKH
	TenCongTy
	DiaChi
	SoDienThoai
	SoFax

DatHang		
	Column Name	Da
	SoDonDH	nvarchar
	MSKH	nvarchar
	MSNV	nvarchar
	NgayDH	datetime
	NgayGH	datetime

NhanVien	
	MSNV
	HoTenNV
	ChucVu
	DiaChi
	SoDienThoai

