

TRƯỜNG ĐÀO TẠO LẬP TRÌNH VIÊN VÀ QUẢN TRỊ MẠNG QUỐC TẾ BACHKHOA-APTECH

---

# Bài 12

# Trigger

# Tóm tắt

- Giới thiệu Trigger
- Thao tác với Trigger



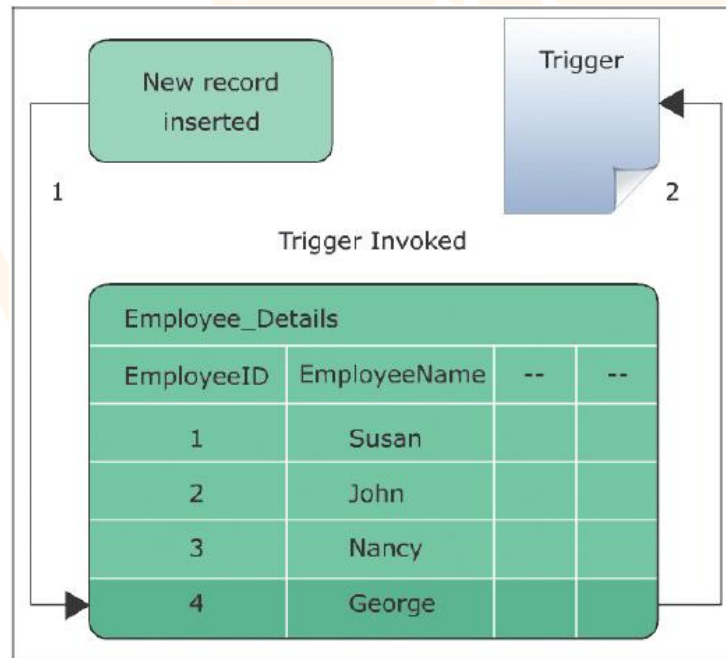
# Giới thiệu

## Trigger

- Là một thủ tục lưu trữ (**stored procedure**) được thực hiện khi cố gắng thực hiện **sửa đổi dữ liệu** trong một bảng **được bảo vệ bởi trigger**.
- **Không thể thực thi** trực tiếp, và cũng **không thể truyền** hoặc nhận **tham số**.
- Được định nghĩa trên các **bảng cụ thể** và các bảng được coi như là bảng trigger.

# Giới thiệu

- Được định nghĩa cho các hành động **INSERT**, **UPDATE**, hoặc **DELETE** trên bảng, nó **tự động** được **kích hoạt** khi các hành động này được thực hiện.
- Được tạo ra bằng câu lệnh **CREATE TRIGGER**.



# Giới thiệu

Trigger có thể **chứa xử lý logic phức tạp** và thường được sử dụng để duy trì tính toàn vẹn dữ liệu mức thấp. Mục đích(use) chính của trigger có thể được phân loại như sau:

- **Thay đổi lan truyền** (cascading) tới các bảng có liên quan.
- Thực thi các **toàn vẹn dữ liệu** phức tạp hơn ràng buộc (constraint) CHECK.
- Định nghĩa **thông báo lỗi** theo ý muốn

# Giới thiệu

## Thay đổi lan truyền(cascading) tới các bảng có liên quan.

- Người dùng có thể sử dụng trigger để thay đổi lan truyền tới các bảng có liên quan.

## Thực thi các toàn vẹn dữ liệu phức tạp hơn ràng buộc(constraint) CHECK.

- Không giống như constraint CHECK, trigger có thể tham chiếu đến các cột trong các bảng khác. Do vậy, có thể được áp dụng để kiểm tra các toàn vẹn dữ liệu phức tạp sau:
  - Kiểm tra các ràng buộc trước khi cập nhật hoặc xóa lan truyền
  - Tạo trigger đa dòng cho các hành động được thực thi trên nhiều dòng
  - Thực thi toàn vẹn tham chiếu giữa các cơ sở dữ liệu

## Định nghĩa thông báo lỗi theo ý muốn

- Được sử dụng để cung cấp các lời giải thích chi tiết hoặc phù hợp hơn trong các tình huống lỗi nhất định.

# Giới thiệu

Các yếu tố lập trình Transact-SQL cho phép thực hiện nhiều thao tác khác nhau mà không thể thực hiện được bằng một câu lệnh đơn.

## Duy trì dữ liệu chưa chuẩn hóa

- Toàn vẹn dữ liệu mức thấp có thể được duy trì trong môi trường csdl chưa được chuẩn hóa (denormalized) bằng trigger.
- Dữ liệu chưa chuẩn hóa thường đề cập đến dữ liệu dư thừa hoặc dữ liệu dẫn suất(derived). Ở đây, trigger được sử dụng để kiểm tra mà không cần kết hợp chính xác.

## So sánh tình trạng dữ liệu trước và sau khi cập nhật.

- Trigger cung cấp tùy chọn để tham chiếu sự thay đổi dữ liệu do các lệnh INSERT, UPDATE, và DELETE.

# Giới thiệu

Trigger được phân thành ba loại sau:

## Data Manipulation Language(DML) Triggers

- Thực thi khi dữ liệu trong một bảng hoặc view được chèn, cập nhật hoặc xóa bởi các lệnh INSERT, UPDATE, DELETE.

## Data Definition Language(DDL) Trigger

- Thực thi khi bảng hoặc view được tạo, xóa, thay đổi bằng các lệnh CREATE, DROP, ALTER.

## Logon Triggers

- Thực thi các thủ tục nội tại để đáp ứng sự kiện LOGON. Logon trigger kích hoạt sau khi giai đoạn chứng thực đăng nhập kết thúc, nhưng trước khi phiên làm việc người dùng thực được thiết lập. Logon trigger không kích hoạt nếu việc chứng thực thất bại



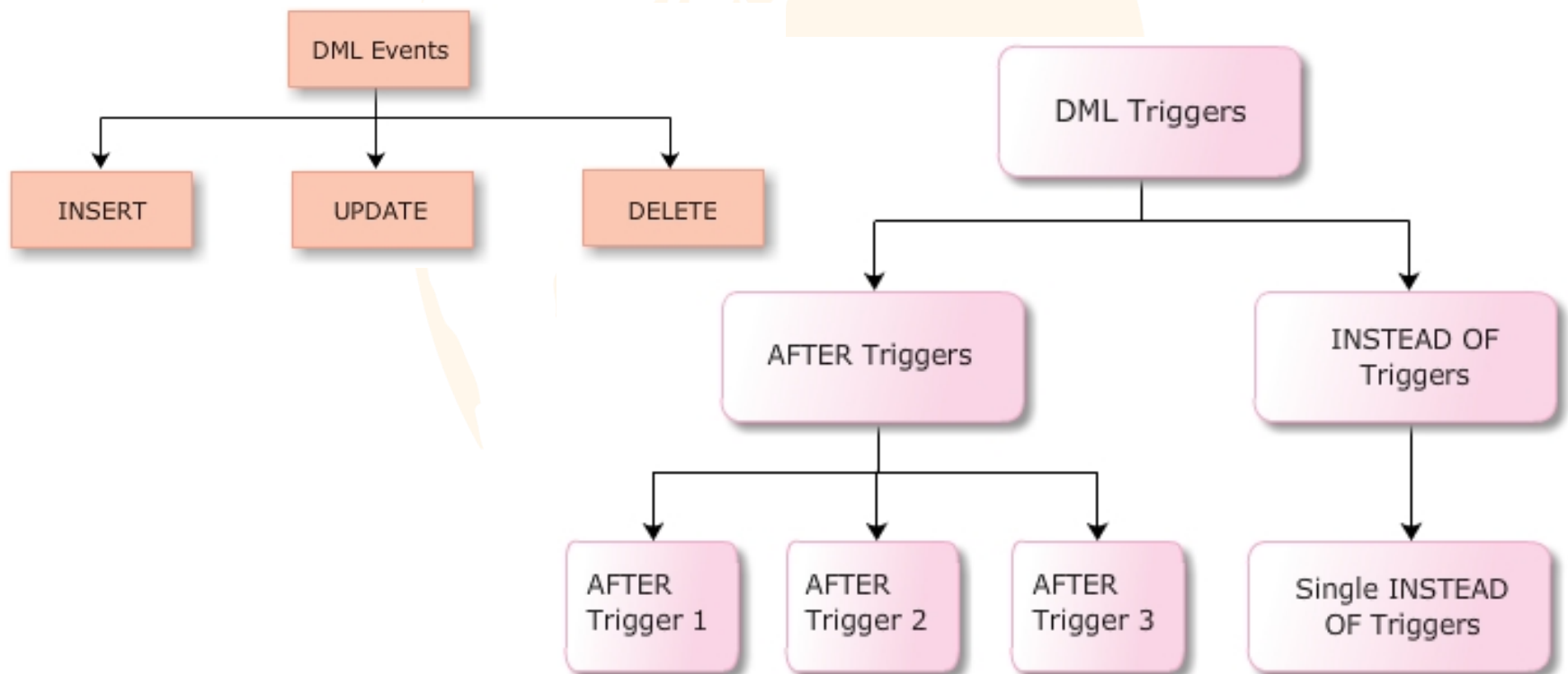
# Giới thiệu

## So sánh 2 loại trigger:

DDL Trigger	DML Trigger
Các trigger DDL thực thi các thủ tục lưu trữ trên câu lệnh CREATE, ALTER, và DROP.	Các trigger DML thực thi trên các câu lệnh INSERT, UPDATE, và DELETE.
Các trigger DDL được sử dụng để kiểm tra và kiểm soát các hoạt động của cơ sở dữ liệu.	Các trigger DML được sử dụng để thực thi các quy tắc nghiệp vụ khi dữ liệu được sửa đổi trong các bảng hoặc view.
Các trigger DDL chỉ hoạt động sau khi bảng hoặc view được sửa đổi.	Các trigger DML thực thi trong khi sửa đổi dữ liệu hoặc sau khi dữ liệu được sửa đổi.
Các trigger DDL được định nghĩa ở mức cơ sở dữ liệu hoặc máy chủ.	Các trigger DML được định nghĩa ở mức cơ sở dữ liệu.

# Giới thiệu

DML trigger được thực thi khi các sự kiện DML xảy ra trong bảng hoặc view. Các sự kiện DML này bao gồm các câu lệnh **INSERT, UPDATE, và DELETE**.



# Giới thiệu

Bảng **Inserted** và bảng **Deleted**:

- Các câu lệnh SQL trong DML trigger có thể sử dụng hai bảng logic đặc biệt là Inserted và Deleted **để sửa đổi dữ liệu** trong csdl.
- Bảng Inserted và Deleted **không phải** là các bảng được duy trì **vật lý trong csdl**, nó được tạo(create) và xóa(drop) bất kỳ khi nào các sự kiện trigger xảy ra.

# Giới thiệu

## Bảng Inserted

- Chứa bản sao của bản ghi dữ liệu được chỉnh sửa bằng thao tác INSERT, UPDATE trên bảng trigger.
- Các thao tác INSERT và UPDATE sẽ thực hiện thêm các bản ghi mới vào cả bảng Inserted và bảng trigger.

## Bảng Deleted

- Chứa bản sao của bản ghi dữ liệu được chỉnh sửa bằng thao tác DELETE , UPDATE trên bảng trigger
- Những thao tác này thực hiện xóa các bản ghi khỏi bảng trigger và chèn chúng sang bảng Deleted.



# DML TRIGGER

# Thao tác

## Insert Triggers

- Trigger INSERT được thực thi khi một **bản ghi mới** được chèn vào bảng
- **Đảm bảo** rằng các **giá trị** được nhập vào **phù hợp** với các ràng buộc được định nghĩa trên bảng đó.
- Lưu một bản sao của các bản ghi vào bảng Inserted và kiểm tra xem **giá trị mới** trong bảng Inserted **phù hợp với các ràng buộc** đã được chỉ ra hay không.

# Thao tác

- **Chèn** dòng trong bảng trigger **nếu bản ghi hợp lệ**, ngược lại, nó sẽ hiển thị một thông ghi báo lỗi.
- Được tạo ra bằng cách sử dụng từ khóa **INSERT** trong các câu lệnh **CREATE TRIGGER** và **ALTER TRIGGER**.

## Cú pháp:

```
CREATE TRIGGER [schema_name.] trigger_name  
ON [schema_name.] table_name [WITH ENCRYPTION]  
{FOR INSERT} AS  
[IF UPDATE (column_name)...] [{AND | OR} UPDATE (column_name)...]  
<sql_statements>
```

# Thao tác

Trong đó:

- `schema_name`: chỉ ra tên của lược đồ mà bảng/trigger thuộc về nó.
- `trigger_name`: chỉ ra tên của trigger.
- `table_name`: chỉ ra bảng mà DML trigger được tạo ra cho nó.
- `WITH ENCRYPTION`: mã hóa phần text (nội dung) của câu lệnh `CREATE TRIGGER`.



# Thao tác

- FOR: chỉ ra rằng DML trigger được thực thi sau khi các thao tác chỉnh sửa hoàn tất (complete).
- INSERT: chỉ ra rằng DML trigger sẽ được gọi bởi các thao tác insert.
- UPDATE: trả về giá trị kiểu Boolean cho biết rằng có sự thay đổi INSERT hay UPDATE nào được thực hiện trên cột được chỉ ra không.
- column\_name: là tên của cột cần kiểm tra có bị UPDATE không.
- sql\_statement: chỉ ra câu lệnh SQL được thực thi trong DML trigger.

# Thao tác

Ví dụ: Tạo trigger kiểm soát việc nhập dữ liệu (INSERT) vào bảng GIAO\_DICH\_KHACH\_HANG không vượt quá 5000 đồng và thông báo:

```
-- Tạo 1 trigger: yêu cầu khi thực hiện rút tiền không vượt quá 5000
CREATE TRIGGER KiemTra_SoTienRut ON giao_dich_khachhang
FOR INSERT -- Có các hành động: INSERT | UPDATE | DELETE
AS -- định nghĩa thân của trigger
-- IF - Kiểm tra những điều kiện ràng buộc nghiệp vụ để đảm bảo tính
đúng đắn của dữ liệu
IF (SELECT so_tien_rut FROM inserted) > 5000
BEGIN -- bắt đầu nhóm lệnh nghiệp vụ
PRINT N'Số tiền rút không được vượt quá 5000'
ROLLBACK TRANSACTION
END -- kết thúc các lệnh nghiệp vụ
```

# Thao tác

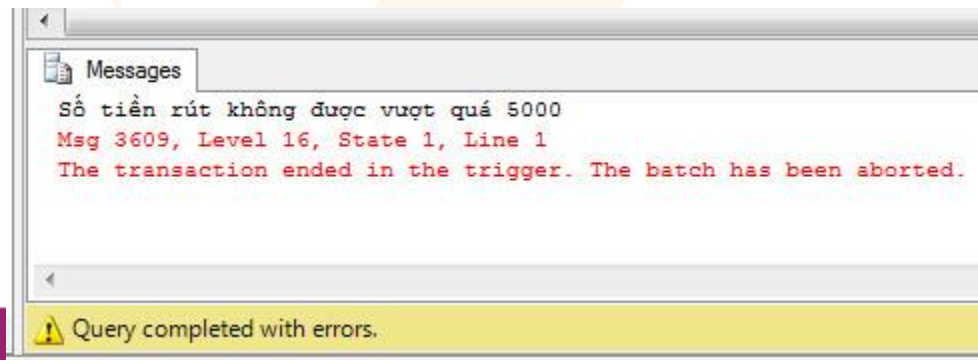
Thực thi việc insert dữ liệu vào bảng với dữ liệu số tiền rút nhỏ và lớn hơn 5000.

-- Thực thi thêm dữ liệu sau khi đã có TRIGGER

```
INSERT INTO giao_dich_khachhang(ten_kh, so_tien_rut) VALUES ('minhvt',  
3333)
```



```
INSERT INTO giao_dich_khachhang(ten_kh, so_tien_rut) VALUES ('minhvt',  
5555)
```



## Update Triggers

- Sao chép các bản ghi gốc vào bảng Deleted và các bản ghi mới vào bảng Inserted **khi một bản ghi được cập nhật.**
- **Đánh giá bản ghi mới** để xác định xem các giá trị có **phù hợp với các ràng buộc** được chỉ ra trong bảng trigger không.

# Thao tác

- Sao chép bản ghi từ bảng Inserted vào bảng trigger được cung cấp các bản ghi hợp lệ.
- **Hiển thị thông báo lỗi** nếu các giá trị mới là không hợp lệ và bản sao các bản ghi từ các bảng Deleted trở lại vào bảng trigger.
- Được tạo ra bằng cách sử dụng từ khóa **UPDATE** trong lệnh **CREATE TRIGGER** và **ALTER TRIGGER**.

# Thao tác

## Cú pháp:

```
CREATE TRIGGER [schema_name.] trigger_name  
ON [schema_name.] table_name [WITH ENCRYPTION]  
{FOR UPDATE} AS [IF UPDATE (column_name)...] [{AND | OR} UPDATE  
    (column_name)...]  
<sql_statements>
```

Trong đó:

- `schema_name`: chỉ ra tên của lược đồ mà bảng/trigger thuộc về nó.
- `trigger_name`: chỉ ra tên của trigger.
- `table_name`: chỉ ra bảng mà DML trigger được tạo ra cho nó.

# Thao tác

- **WITH ENCRYPTION:** mã hóa phần text (nội dung) của câu lệnh CREATE TRIGGER.
- **FOR:** chỉ ra rằng DML trigger được thực thi sau khi các thao tác chỉnh sửa hoàn tất (complete).
- **INSERT:** chỉ ra rằng DML trigger sẽ được gọi bởi các thao tác insert.
- **UPDATE:** trả về giá trị kiểu Boolean cho biết rằng có sự thay đổi INSERT hay UPDATE nào được thực hiện trên cột được chỉ ra không.
- **column\_name:** là tên của cột cần kiểm tra có bị UPDATE không.
- **sql\_statement:** chỉ ra câu lệnh SQL được thực thi trong DML trigger.

# Thao tác

Ví dụ: Tạo trigger kiểm soát ngày rút tiền trong bảng trên không được là ngày trong tương lai.

-- Tạo Trigger khi thực hiện cập nhật dữ liệu

```
CREATE TRIGGER kiểmtra_capnhat_gd ON giao_dich_khachhang
```

```
FOR UPDATE AS
```

```
IF (SELECT ngay_gd FROM inserted) > getDate()
```

```
BEGIN
```

```
PRINT N'Không thể cập nhật ngày giao dịch trong tương lai'
```

```
ROLLBACK TRANSACTION -- Không cho phép nếu là ngày trong tương  
lai
```

```
END
```

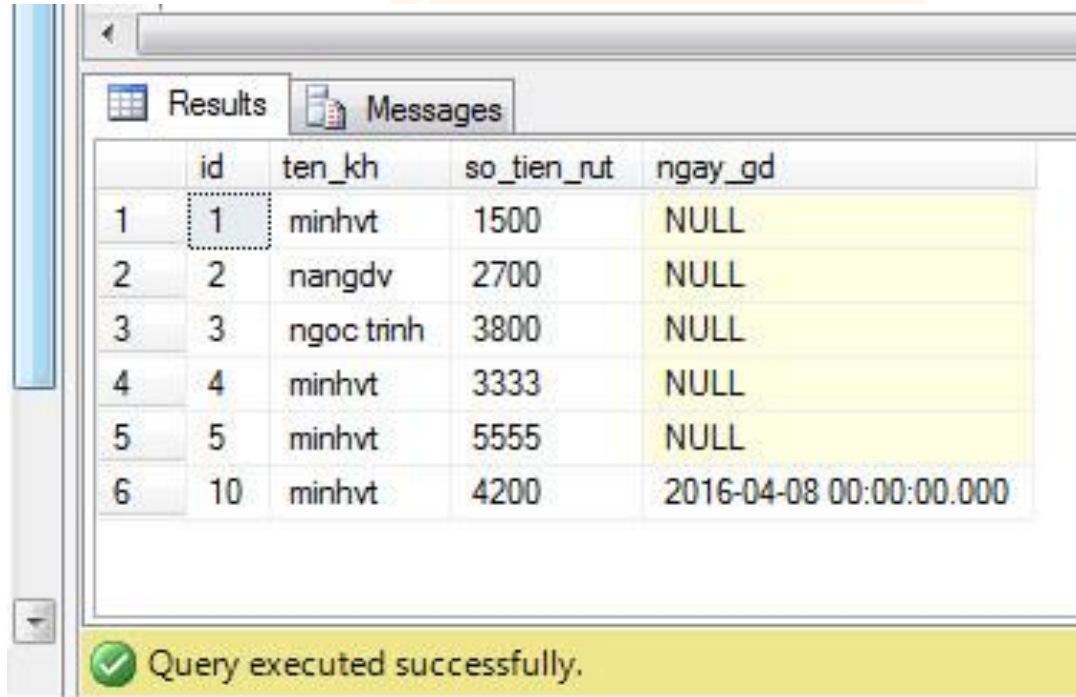


# Thao tác

Thêm dữ liệu:

-- Thêm dữ liệu

```
INSERT INTO giao_dich_khachhang(ten_kh, so_tien_rut,  
ngay_gd) VALUES ('minhvt', 4200, '2016-04-08') -- Insert  
vẫn được
```



	id	ten_kh	so_tien_rut	ngay_gd
1	1	minhvt	1500	NULL
2	2	nangdv	2700	NULL
3	3	ngoc trinh	3800	NULL
4	4	minhvt	3333	NULL
5	5	minhvt	5555	NULL
6	10	minhvt	4200	2016-04-08 00:00:00.000

Query executed successfully.

# Thao tác

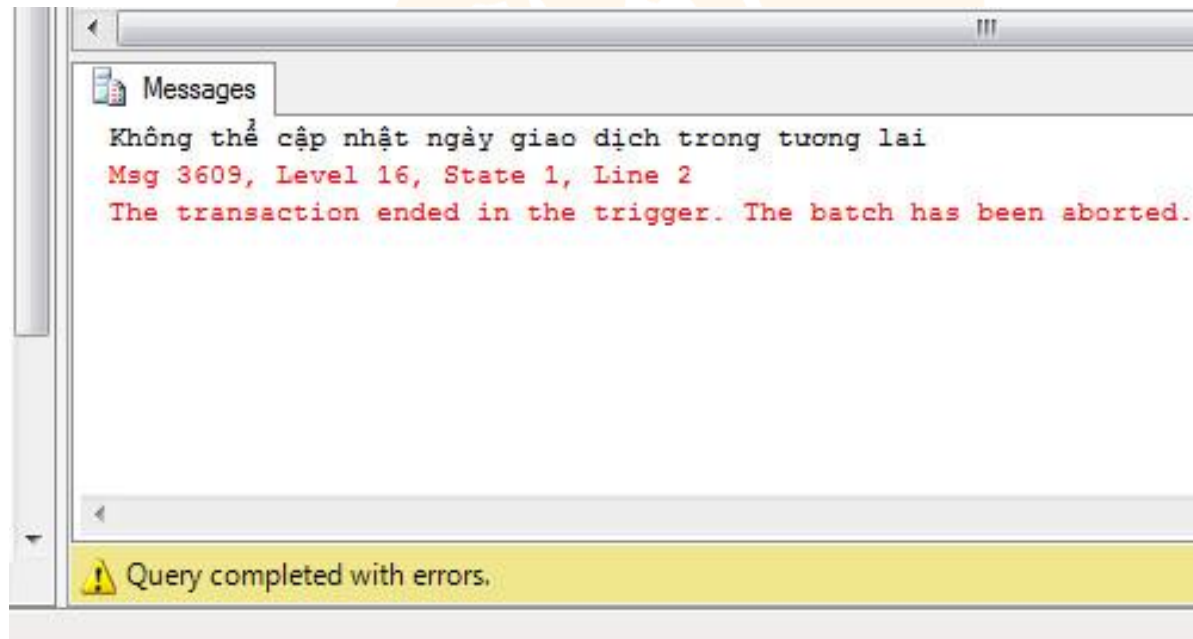
Nhưng khi CỖ Ý thực hiện cập nhật:

```
-- Lệnh cập nhật bị từ chối
```

```
UPDATE giao_dich_khachhang SET
```

```
giao_dich_khachhang.ngay_gd = '2016-04-10'
```

```
WHERE giao_dich_khachhang.id = 10
```



# Thao tác

## Delete Triggers

- Có thể được tạo ra để **giới hạn người dùng xóa** các bản ghi cụ thể trong bảng.
- Sẽ xảy ra các vấn đề sau khi người dùng cố gắng xóa các bản ghi:
  - ✓ Bảng ghi được xóa khỏi bảng trigger và chèn vào bảng Deleted.
  - ✓ Nó **kiểm tra các ràng buộc** với các bảng ghi xóa.
  - ✓ Nếu các bản ghi xóa vi phạm ràng buộc, DELETE trigger **hiển thị thông điệp lỗi**.
  - ✓ Bảng ghi bị xóa được lưu trữ trong bảng Deleted được **copy quay lại bảng trigger**.
- Được tạo bằng cách dùng từ khóa **DELETE** trong các câu lệnh **CREATE TRIGGER**.

# Thao tác

## Cú pháp:

```
CREATE TRIGGER <trigger_name>  
ON <table_name>  
[WITH ENCRYPTION]  
FOR DELETE  
AS <sql_statement>
```

Trong đó:

DELETE: chỉ ra DML trigger này sẽ được gọi bởi các thao tác xóa.

# Thao tác

Ví dụ: Tạo trigger cấm xóa giao dịch có ID bằng 10

-- Trigger cho hành động xóa

CREATE TRIGGER kiểmtra\_xoa

ON giao\_dich\_khachhang

FOR DELETE

AS

IF 10 IN (SELECT id FROM deleted)

BEGIN

PRINT N'Không thể xóa giao dịch số 10'

ROLLBACK TRANSACTION

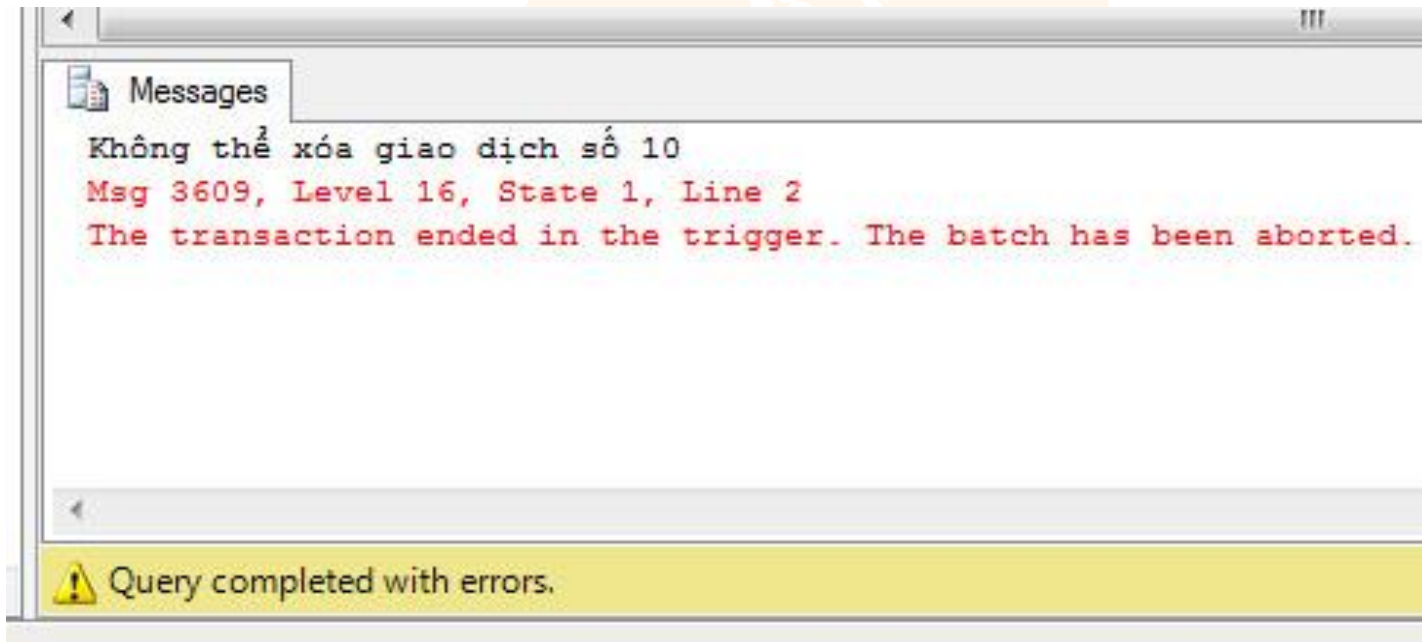
END

# Thao tác

Cố ý xóa:

-- Lệnh xóa

```
DELETE giao_dich_khachhang  
WHERE giao_dich_khachhang.id > 0
```

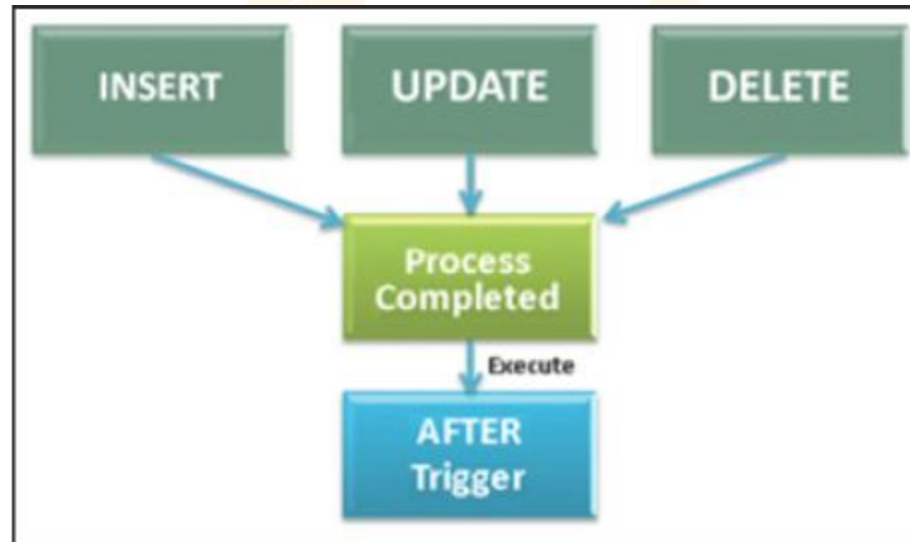


# Thao tác

## AFTER Triggers

- Được **thực thi sau khi các thao tác** INSERT, UPDATE, hoặc DELETE hoàn tất và chỉ có thể tạo cho các bảng.
- Một bảng có thể định nghĩa nhiều AFTER trigger cho mỗi thao tác INSERT, UPDATE, và DELETE và người dùng phải **định nghĩa thứ tự** thực thi cho các trigger.
- **Được thực thi sau khi** việc **kiểm tra ràng buộc** (constraint) trong bảng hoàn tất và trigger cũng được thực thi sau khi bảng Inserted và Deleted được tạo.

# Thao tác



## Cú pháp:

```
CREATE TRIGGER <trigger_name>  
ON <table_name>  
[WITH ENCRYPTION]  
{ FOR | AFTER }  
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }  
AS <sql_statement>
```



# Thao tác

Trong đó:

- FOR | AFTER: chỉ ra DML trigger thực thi sau khi các thao chỉnh sửa hoàn tất.
- { [ INSERT ] , [ UPDATE ] [ , ] [ DELETE ] }: chỉ ra thao tác mà DML trigger sẽ được gọi.

# Thao tác

Ví dụ: Tạo trigger khi thực hiện xóa bản ghi thì tính tổng số bản ghi đã xóa và hiển thị ra màn hình.

-- Tạo Trigger After: là những trigger được thực hiện sau khi CHUYỂN  
ĐÃ RỒI

```
CREATE TRIGGER hienthi_thongbao_xoa ON giao_dich_khachhang  
AFTER DELETE
```

```
AS
```

```
BEGIN
```

```
DECLARE @sobanghi int;
```

```
SELECT @sobanghi = COUNT(*) FROM deleted -- Lấy tất cả bản ghi trong  
bảng deleted của trigger
```

```
PRINT N'Số bản ghi đã xóa là ' + CONVERT(varchar, @sobanghi)
```

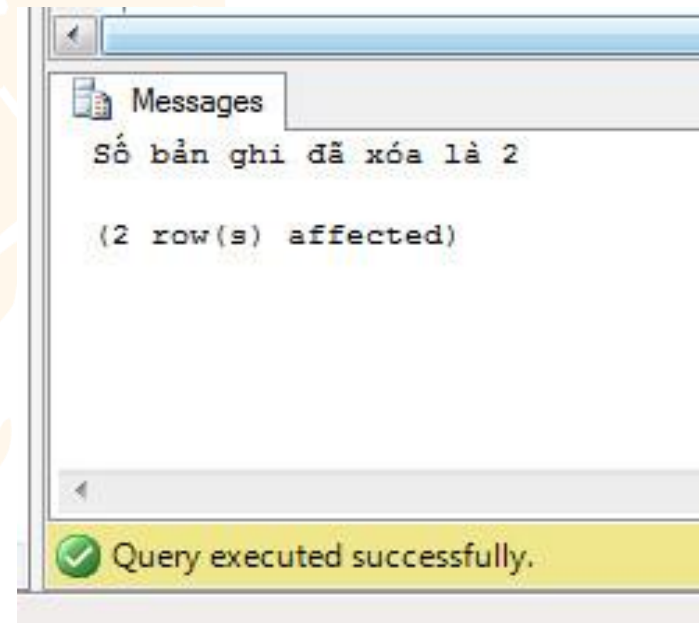
```
END
```

# Thao tác

Thực hiện lệnh xóa:

-- Lệnh xóa

```
DELETE giao_dich_khachhang WHERE  
giao_dich_khachhang.id > 0 AND  
giao_dich_khachhang.id < 3
```

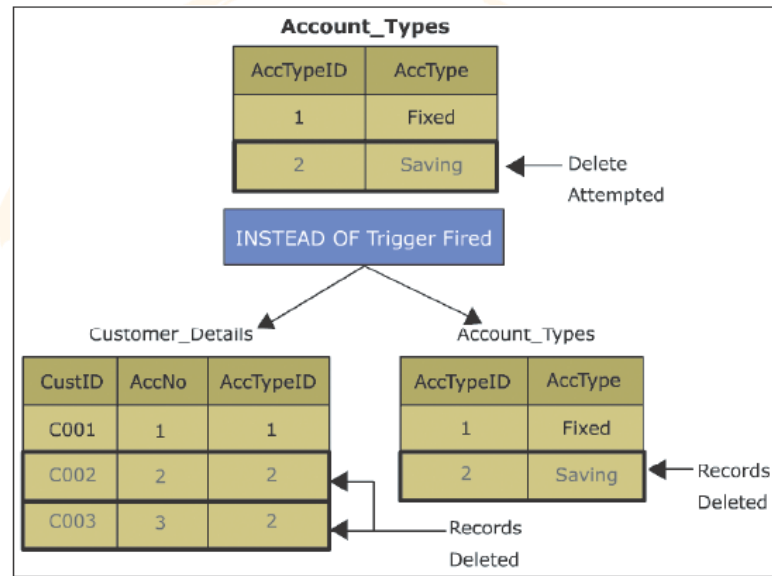


# Thao tác

## INSTEAD OF Triggers

- Được thực thi thay cho các thao tác INSERT, UPDATE, hoặc DELETE.
- Có thể tạo trên **bảng** cũng như trên **view**, và chỉ có thể định nghĩa một INSTEAD OF trigger cho mỗi thao tác INSERT, UPDATE, and DELETE.
- **Được thực hiện trước khi** việc **kiểm tra các ràng buộc** (constraint) được thực thi trên bảng và sau khi các bảng Inserted và Deleted được tạo ra.

# Thao tác



## Cú pháp:

```
CREATE TRIGGER <trigger_name> ON { <table_name> | <view_name> }  
{ FOR | AFTER | INSTEAD OF }  
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }  
AS <sql_statement>
```

# Thao tác

Trong đó:

- `view_name`: chỉ ra tên của view mà DML trigger được tạo cho nó.
- **INSTEAD OF**: chỉ ra DML trigger thực thi thay cho các thao tác chỉnh sửa. Các trigger này không được nghĩa trên các view có thể cập nhật bằng tùy chọn **WITH CHECK OPTION**.

# Thao tác

Ví dụ: Viết trigger để thực hiện xóa dữ liệu ở bảng NhanVien khi xóa PhongBan

-- Tạo instead of trigger, nó sẽ được thực thi trước khi các ràng buộc được kiểm tra

```
CREATE TRIGGER trg_delete_pb ON PhongBan
```

```
INSTEAD OF DELETE AS
```

```
BEGIN
```

```
DELETE NhanVien WHERE NhanVien.id_pb IN (SELECT id_pb FROM deleted)
```

```
END
```

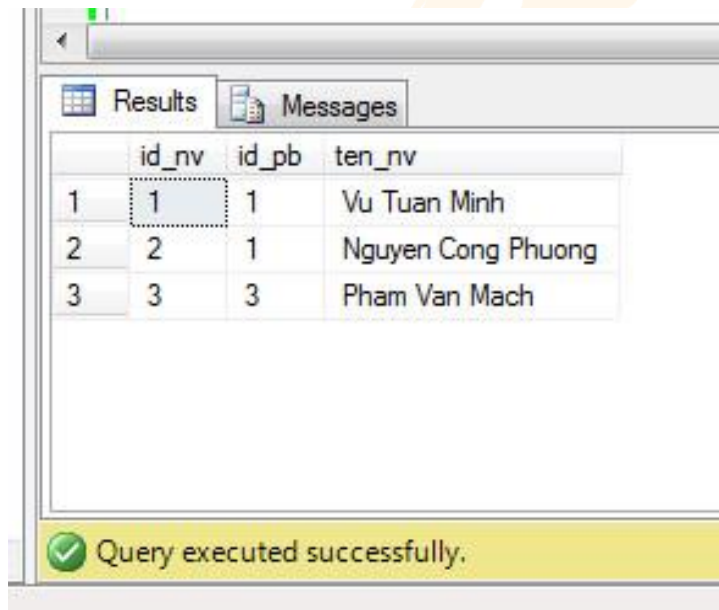
-- Thực hiện xóa dữ liệu có ràng buộc

```
DELETE PhongBan
```

```
WHERE id_pb = 1
```

# Thao tác

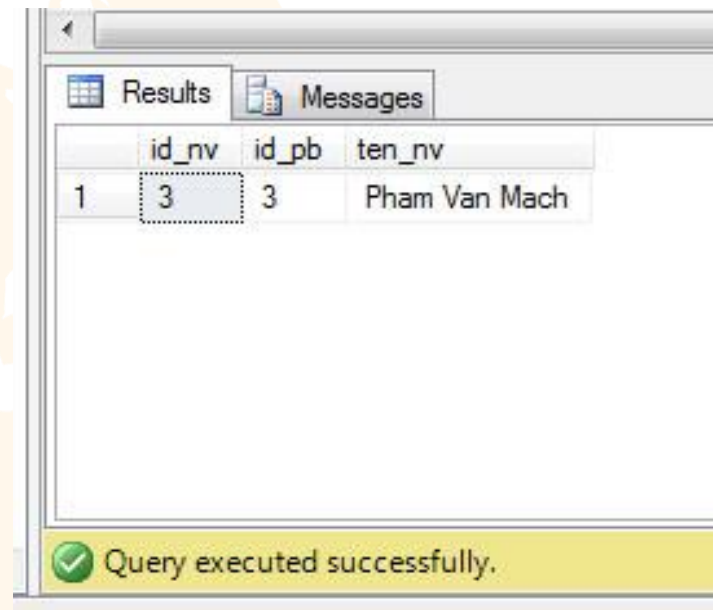
## Bảng NhanVien:



	id_nv	id_pb	ten_nv
1	1	1	Vu Tuan Minh
2	2	1	Nguyen Cong Phuong
3	3	3	Pham Van Mach

Query executed successfully.

Trước khi xóa



	id_nv	id_pb	ten_nv
1	3	3	Pham Van Mach

Query executed successfully.

Sau khi xóa



## Xem định nghĩa của DML Trigger

- Định nghĩa của một trigger gồm có tên trigger, bảng mà trigger được tạo trên đó, hành động làm trigger được gọi, và các câu lệnh SQL được thực thi.
- SQL Server 2014 cung cấp **thủ tục sp\_helptext** để lấy các định nghĩa của trigger.
- Tên DML trigger phải được chỉ ra trong phần tham số của thủ tục khi thực thi sp\_helptext.

# Thao tác

## Cú pháp:

```
sp_helptext '<DML_trigger_name>'
```

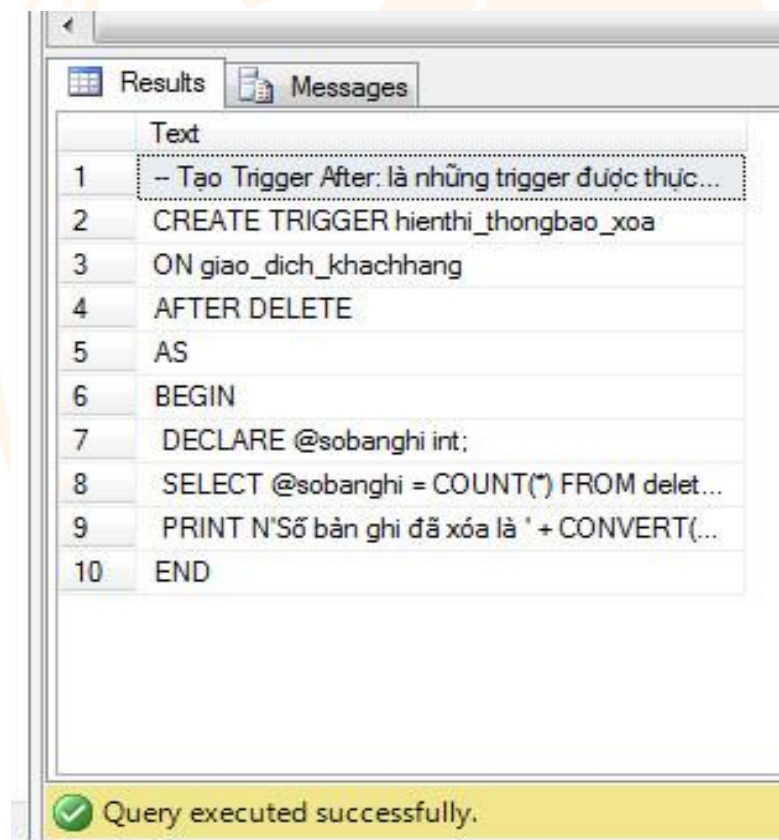
Trong đó:

- DML\_trigger\_name: chỉ ra tên của DML trigger mà phần định nghĩa của nó sẽ được hiển thị.

# Thao tác

-- Xem cú pháp tạo trigger

EXEC sp\_helptext 'hienthi\_thongbao\_xoa'



## Sửa đổi định nghĩa của DML Triggers

- Các thông số của trigger được xác định tại thời điểm tạo bao gồm loại hành động để gọi kích hoạt trigger và các câu lệnh SQL được thực thi.
- Người dùng có thể chỉnh sửa thông số bất kỳ cho DML trigger bằng một trong hai cách:
  1. **Xóa và tạo lại** trigger với các thông số mới.
  2. Thay đổi thông số bằng câu lệnh **ALTER TRIGGER**.
- DML trigger có thể **được mã hóa** để ẩn đi phần định nghĩa của nó.

# Thao tác

## Cú pháp:

```
ALTER TRIGGER <trigger_name> ON  
    { <table_name> | <view_name> }  
[WITH ENCRYPTION]  
{ FOR | AFTER | INSTEAD OF }  
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }  
AS <sql_statement>
```

Trong đó:

- WITH ENCRYPTION: chỉ ra rằng định nghĩa của DML trigger không được hiển thị (khi dùng sp\_helptext,...để xem).
- FOR | AFTER: chỉ ra rằng DML trigger thực thi sau khi các thao tác chỉnh sửa được hoàn tất.
- INSTEAD OF: chỉ ra rằng DML trigger thực thi thay cho (in place of) các thao tác chỉnh sửa.

# Thao tác

Ví dụ:

```
-- Sửa TRIGGER
```

```
ALTER TRIGGER hienthi_thongbao_xoa ON giao_dich_khachhang
```

```
AFTER DELETE AS
```

```
BEGIN
```

```
DECLARE @sobanghi int;
```

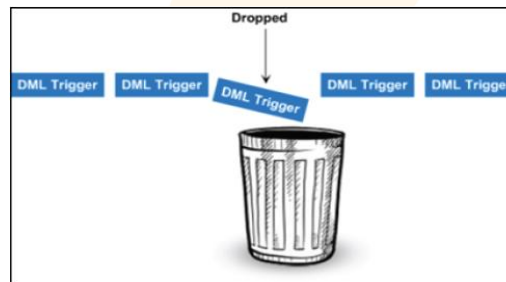
```
SELECT @sobanghi = COUNT(*) FROM deleted -- Lấy tất cả bản ghi  
trong bảng deleted của trigger
```

```
PRINT N'Records has been deleted ' + CONVERT(varchar,  
@sobanghi)
```

```
END
```

## Xóa DML Triggers

- Trigger có thể xóa được bằng câu lệnh DROP TRIGGER.
- Cũng có thể sử dụng một câu lệnh DROP TRIGGER để xóa nhiều trigger.
- Khi một bảng bị xóa, tất cả các trigger định nghĩa trên bảng đó cũng bị xóa.
- Khi DML trigger bị xóa khỏi bảng, thông tin về trigger cũng bị xóa khỏi view danh mục(catalog views).



# Thao tác

## Cú pháp:

```
DROP TRIGGER <DML_trigger_name> [ ,...n ]
```

Trong đó:

- DML\_trigger\_name: chỉ ra tên của DML trigger muốn xóa.
- [ ,...n ]: chỉ ra một danh sách tên các DML triggers có thể bị xóa.

Code:

```
-- Xóa bỏ TRIGGER
```

```
DROP TRIGGER hienthi_thongbao_xoa
```





# DDL TRIGGER

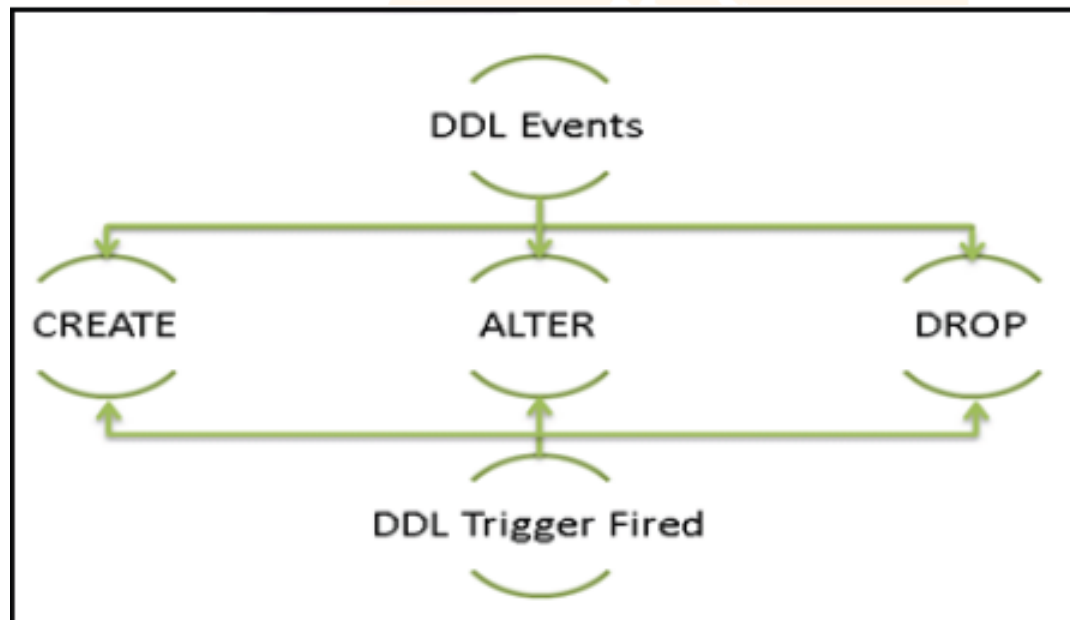
# Thao tác

- DDL triggers thực thi thủ tục lưu khi các sự kiện DDL như các câu lệnh **CREATE, ALTER, và DROP** xảy ra trong csdl hoặc server.
- DDL triggers chỉ có thể hoạt động khi hoàn thành các sự kiện DDL.
- DDL triggers được sử dụng để **ngăn sự chỉnh sửa** trong lược đồ csdl. Một lược đồ là một tập hợp của các đối tượng như tables, views, và vv trong một csdl.
- DDL triggers có thể **gọi một sự kiện hoặc hiển thị một thông điệp** khi có sự sửa đổi trên lược đồ và cũng được định nghĩa ở mức csdl và mức server.

# Thao tác

## Cú pháp:

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



# Thao tác

Trong đó:

- ALL SERVER: chỉ ra rằng DDL trigger thực thi khi sự kiện DDL xảy ra trong server hiện hành.
- DATABASE: chỉ ra rằng DDL trigger thực thi khi sự kiện DDL xảy ra trong csdl hiện hành.
- event\_type: chỉ ra tên của sự kiện DDL sẽ gọi DDL trigger.

# Thao tác

Ví dụ: Viết lệnh tạo Trigger nhóm DDL cấm không được thực hiện thao tác tạo bảng trên csdl BKShop.

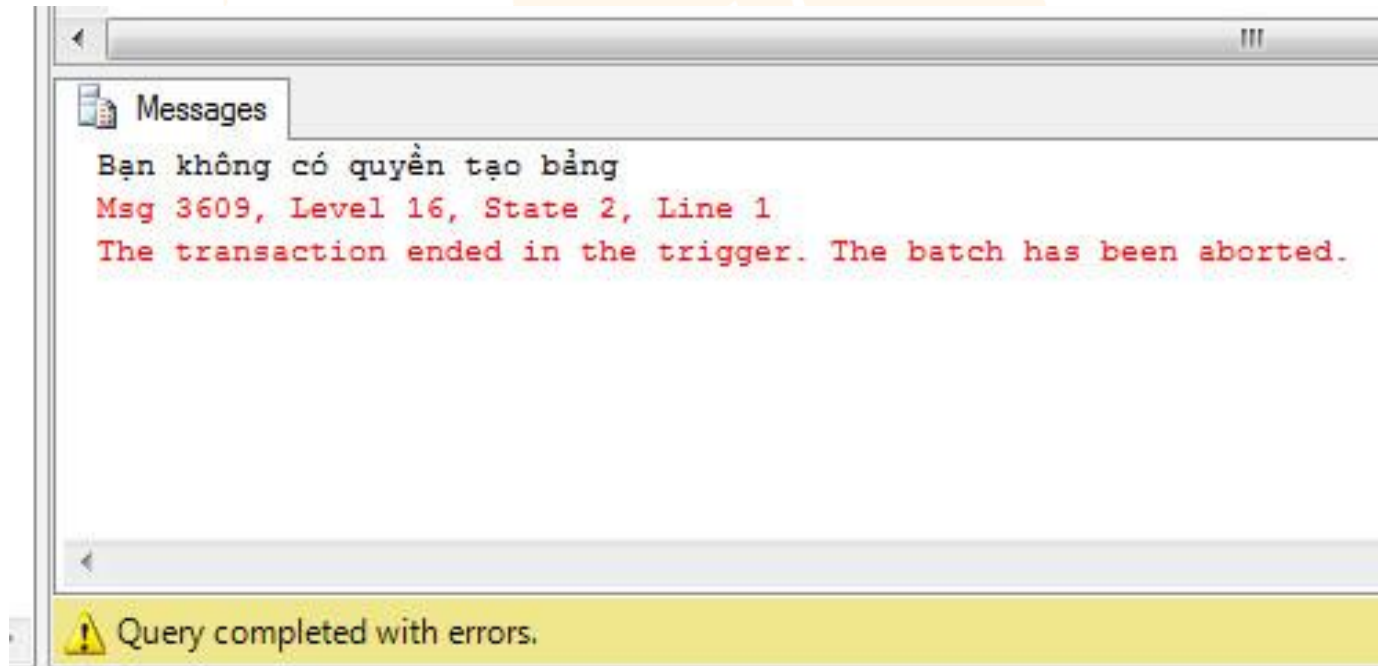
-- Cú pháp tạo trigger

```
CREATE TRIGGER tg_createtb  
ON DATABASE  
FOR CREATE_TABLE, ALTER_TABLE  
AS  
print N'Bạn không có quyền tạo bảng';  
ROLLBACK;
```

# Thao tác

Cố ý tạo và...:

```
CREATE TABLE tblTest(  
id int,  
name varchar(128)  
)
```



# Tóm tắt bài học

- Trigger là một thủ tục lưu trữ (**stored procedure**) được thực hiện khi cố gắng thực hiện sửa đổi dữ liệu trong một bảng được bảo vệ bởi trigger.
- Logon trigger thực thi thủ tục lưu khi một phiên làm việc (session) được thiết lập với sự kiện LOGON.
- **DML trigger** được thực thi khi các sự kiện DML trigger xảy ra trên bảng hoặc view.
- **INSERT trigger** được thực thi khi một bản ghi mới được chèn vào bảng.

# Tóm tắt bài học

- **UPDATE trigger** sao chép bản ghi gốc vào bảng Deleted table và bản ghi mới vào bảng Inserted khi các bản ghi được cập nhật.
- **DELETE trigger** có thể được tạo để giới hạn người dùng khỏi việc xóa các bản ghi cụ thể trong bảng.
- **AFTER trigger** được thực thi sau khi các thao tác INSERT, UPDATE, hoặc DELETE hoàn tất.



## TRƯỜNG ĐÀO TẠO LẬP TRÌNH VIÊN VÀ QUẢN TRỊ MẠNG QUỐC TẾ BACHKHOA-APTECH

---

# Thank for watching!

