Quản Trị Dữ Liệu Với Microsoft SQL Server

Chương: 12

Triggers

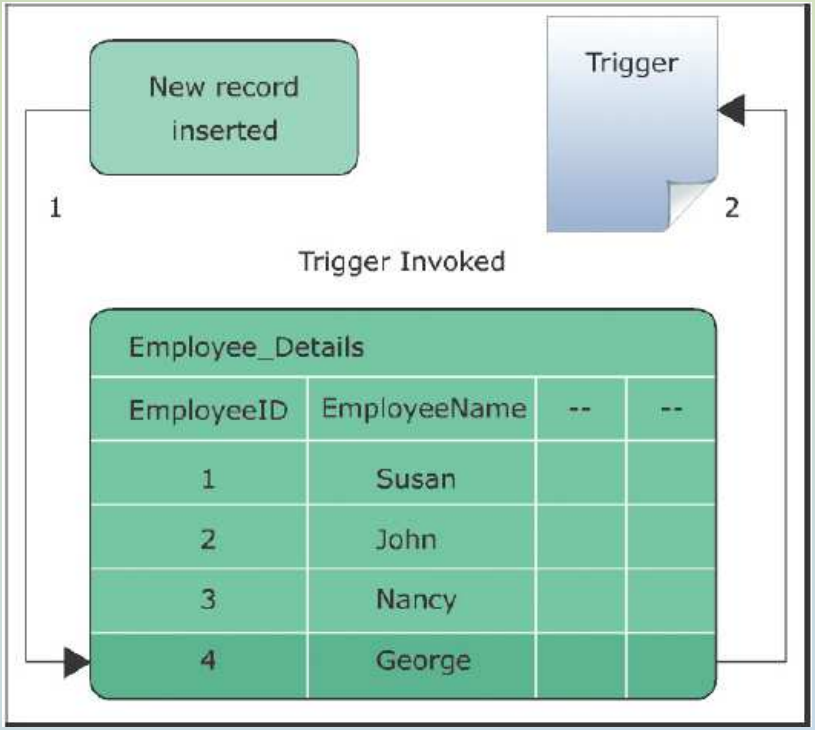
Mục tiêu

- Giải thích trigger
- Giải thích các loại trigger nhau
- Giải thích các bước thủ tục để tạo DML trigger
- Giải thích các bước thủ tục sửa đổi(alter) tạo DML trigger
- Mô tả các trigger lồng nhau(nested)
- Mo tả hàm update
- Giải thích việc xử lý đa dòng trong một phiên
- Giải thích ý nghĩa thực hiện của triggers

SQL Server 2012

Giới thiệu

- Một 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.
  - 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`.



SQL Server 2012

Mục đích của Trigger

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 chuyền(cascading) tới các bảng có liên quan.

- Người dùng có thể sử dụng trigger để thay đổi lan chuyền 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.

Các yếu tố lập trình Transact-SQL

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`.

SQL Server 2012

Các loại Trigger

Trigger trong SQL Server 2012 được phân thành ba loại sau:

Data Manipunation 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. Loggon 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

SQL Server 2012

DDL Trigger so với DML Trigger

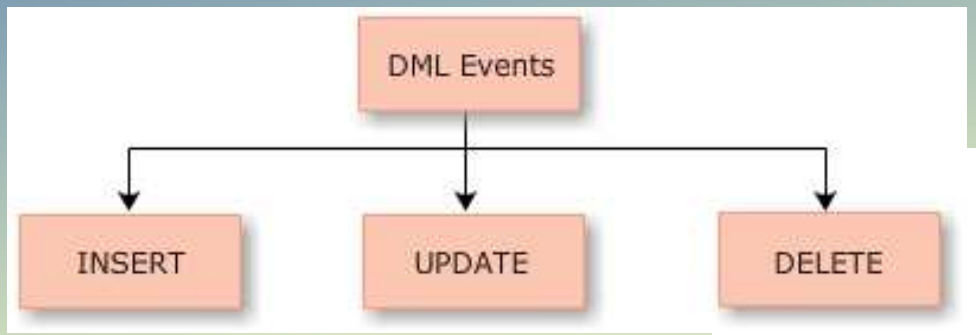
DDL và DML trigger có sự sử dụng và được thực thi khác biệt với các sự kiện csdl khác nhau.

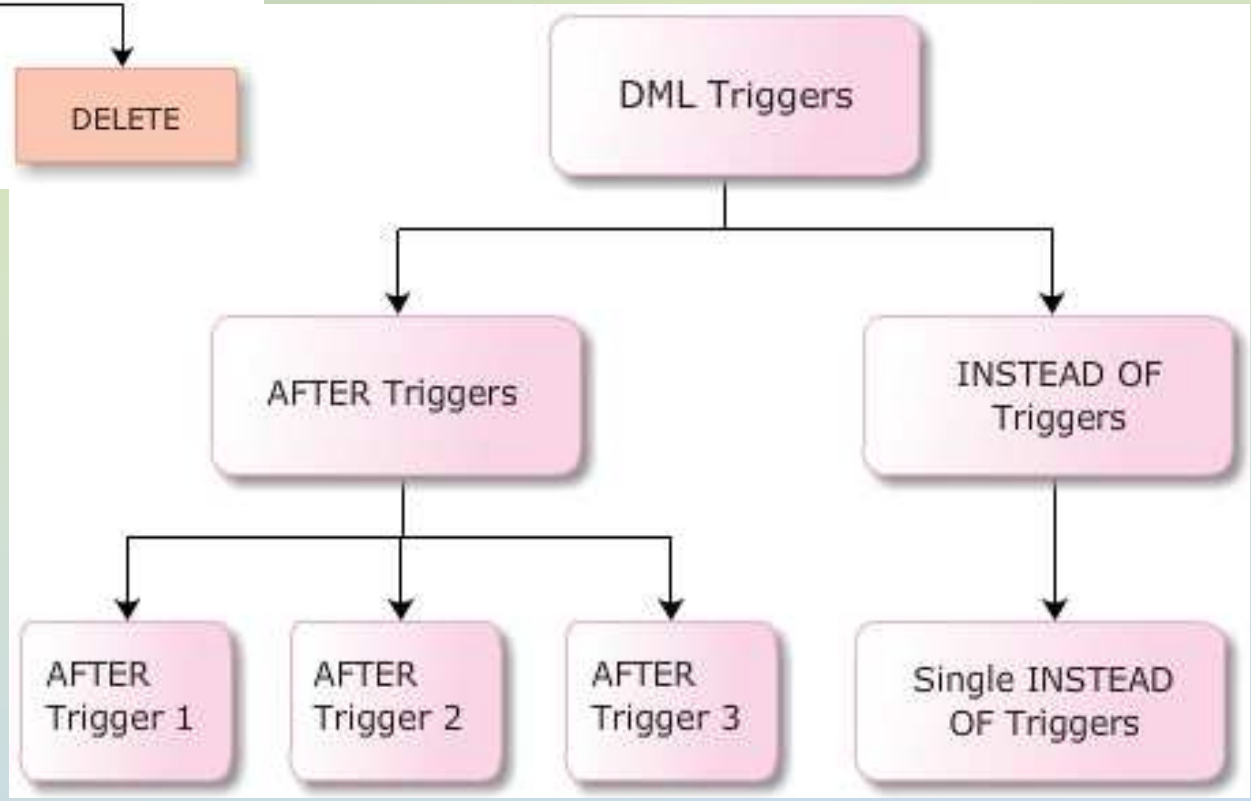
- Bảng dưới đây liệt kê một số khác biệt giữa hai loại trigger:

DDL Triggers	DML Triggers
DDL triggers execute stored procedures on CREATE, ALTER, and DROP statements.	DML triggers execute on INSERT, UPDATE, and DELETE statements.
DDL triggers are used to check and control database operations.	DML triggers are used to enforce business rules when data is modified in tables or views.
DDL triggers operate only after the table or a view is modified.	DML triggers execute either while modifying the data or after the data is modified.
DDL triggers are defined at either the database or the server level.	DML triggers are defined at the database level.

Tạo DML Trigger

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`.





Sử dụng 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

- 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.

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.

SQL Server 2012

Insert Triggers 1-4

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.

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.

Insert Triggers 2-4

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>
```

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`.
- `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.

Insert Triggers 3-4

- `UPDATE`: trả về giá trị kiểu Booean 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.

➢ Đoạn code sau cho thấy cách tạo một `INSERT` trigger trên bảng có tên **`Account_Transactions`**:

```
CREATE TRIGGER CheckWithdrawal_Amount
ON Account_Transactions
FOR INSERT
AS
   IF (SELECT Withdrawal From inserted) > 80000
   BEGIN
       PRINT 'Withdrawal amount cannot exceed 80000'
       ROLLBACK TRANSACTION
   END
```

Insert Triggers 4-4

- Đoạn code sau chèn một bản ghi và hiển thị thông điệp lỗi khi lượng Withdrawal vượt quá 80000:

```
INSERT INTO Account_Transactions
(TransactionID, EmployeeID,
   CustomerID,TransactionTypeID,TransactionDate,
TransactionNumber,Deposit,Withdrawal)
VALUES
(1008,'E08','C08','T08','05/02/12','TN08', 300000,90000)
```

Kết quả:

```
Withdrawal amount cannot exceed 80000.
```

SQL Server 2012

Update Triggers 1-5

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.

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.

Update Triggers 2-5

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.
- `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 Triggers 3-5

- `UPDATE`: trả về giá trị kiểu Booean 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.

➢ Đoạn code sau cho thấy cách tạo một `UPDATE` trigger trên bảng có tên **`EmployeeDetails`**:

```
CREATE TRIGGER CheckBirthDate ON EmployeeDetails
FOR UPDATE
AS
IF (SELECT BirthDate From inserted) > getDate()
   BEGIN
        PRINT 'Date of birth cannot be greater than today's date'
        ROLLBACK
   END
```

Update Triggers 4-5

- Đoạn code dưới đây cập nhật các bản ghi và hiển thị thông báo lỗi khi ngày sinh chỉ ra không hợp lệ:

```
UPDATE EmployeeDetails
SET BirthDate='2015/06/02'
WHERE EmployeeID='E06')
```

Kết quả:

```
Date of birth cannot be greater than today's date.
```

Tạo Update Triggers

- Được tạo ở mức cột hoặc mức bảng.
- Trigger ở mức cột thực thi khi các cập nhật được thực hiện trong cột được chỉ ra.
- Trigger ở mức bảng thực thi khi các cập nhật được thực hiện bất kỳ đâu trong toàn bộ bảng.
- Hàm `UPDATE()` được sử dụng để chỉ ra cột khi tạo một `UPDATE` trigger cho mức cột.

Update Triggers 5-5

- Đoạn code dưới đây tạo một `UPDATE` trigger mức cột trên cột **`EmployeeID`** của bảng **`EmployeeDetails`**:

```
CREATE TRIGGER Check_EmployeeID ON EmployeeDetails
FOR UPDATE
AS
IF UPDATE(EmployeeID)
BEGIN
   PRINT 'You cannot modify the ID of an employee'
   ROLLBACK TRANSACTION
END
```

- Đoạn code dưới đây làm cho update trigger được kích hoạt :

```
UPDATE EmployeeDetails
SET EmployeeID='E12'
WHERE EmployeeID='E04'
```

SQL Server 2012

Delete Triggers 1-3

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 dung 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`.

Delete Triggers 2-3

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.

Delete Triggers 3-3

- Đoạn code sau đây cho thấy cách tạo một `DELETE` trigger trên bảng **`Account_Transactions`**:

```
CREATE TRIGGER CheckTransactions
ON Account_Transactions
FOR DELETE
AS
IF 'T01' IN (SELECT TransactionID FROM deleted)
BEGIN
PRINT 'Users cannot delete the transactions.'
ROLLBACK TRANSACTION
END
```

- Đoạn code sau đây xóa các bản ghi có Deposit là 5000 khỏi bảng **`Account_Transactions`** và hiển thị thông điệp lỗi:

```
DELETE FROM Account_Transactions
WHERE Deposit= 50000
```

Kết quả:

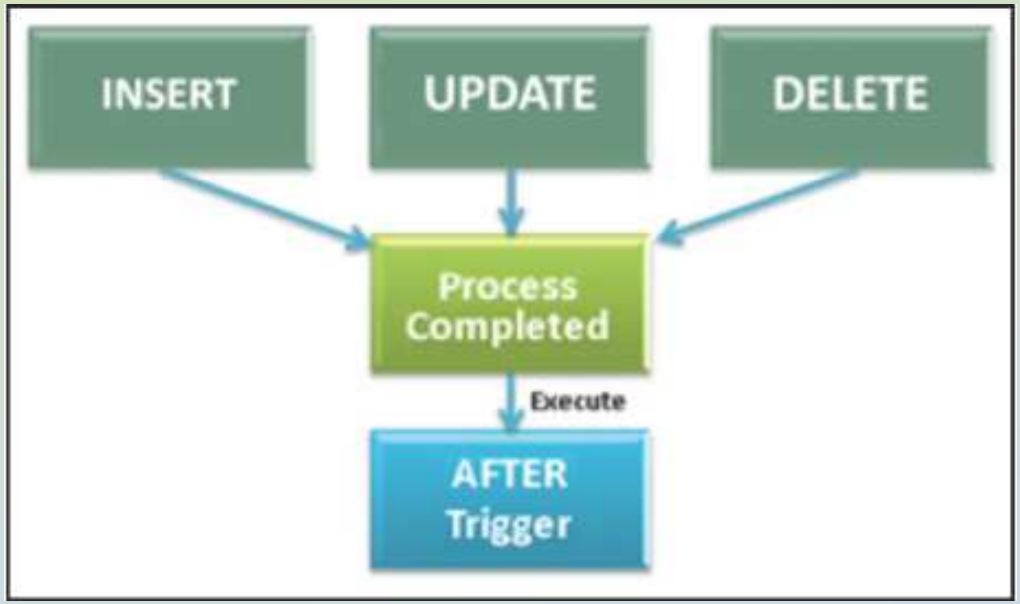
```
Users cannot delete the transactions.
```

AFTER Triggers 1-3

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.



AFTER Triggers 2-3

Cú pháp:

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

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.

SQL Server 2012

AFTER Triggers 3-3

- Đoạn code sau đây cho thấy cách tạo một `AFTER DELETE` trigger trên bảng **`EmployeeDetails`**:

```
CREATE TRIGGER Employee_Deletion
ON EmployeeDetails
AFTER DELETE
AS
BEGIN
DECLARE @num nchar;
SELECT @num = COUNT(*) FROM deleted
PRINT 'No. of employees deleted = ' + @num
END
```

- Đoạn code sau xóa các bản ghi khỏi bảng **`EmployeeDetails`** và hiển thị thông điệp lỗi:

```
DELETE FROM EmployeeDetails WHERE EmployeeID='E07'
```

Kết quả:

```
No. of employees deleted = 0.
```

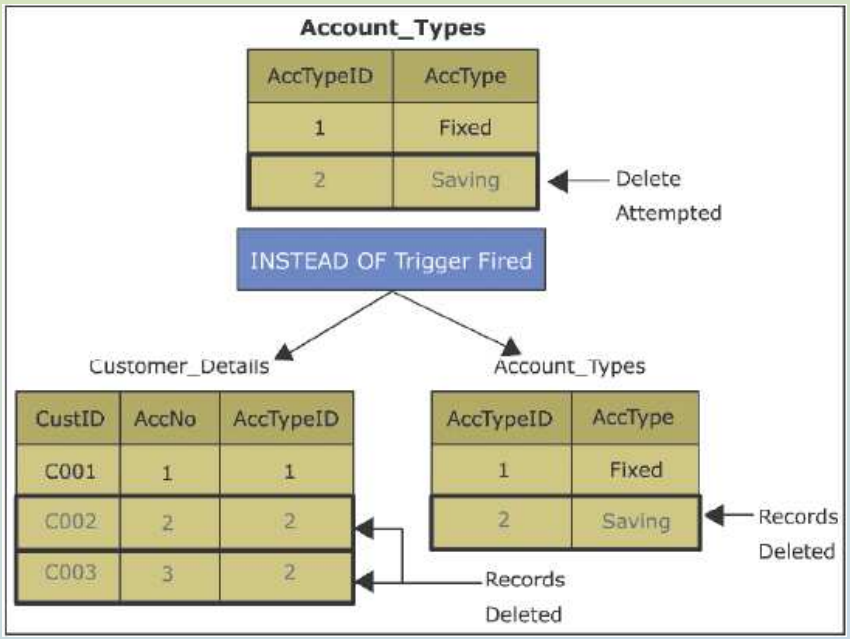
SQL Server 2012

INSTEAD OF Triggers 1-3

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.



INSTEAD OF Triggers 2-3

Cú pháp:

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

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 tho tác chỉnh sửa. Các trigger này không được nghĩa trên các view có thể cập nhật bằng tùy chọn `WITH CHECK OPTION`.

INSTEAD OF Triggers 3-3

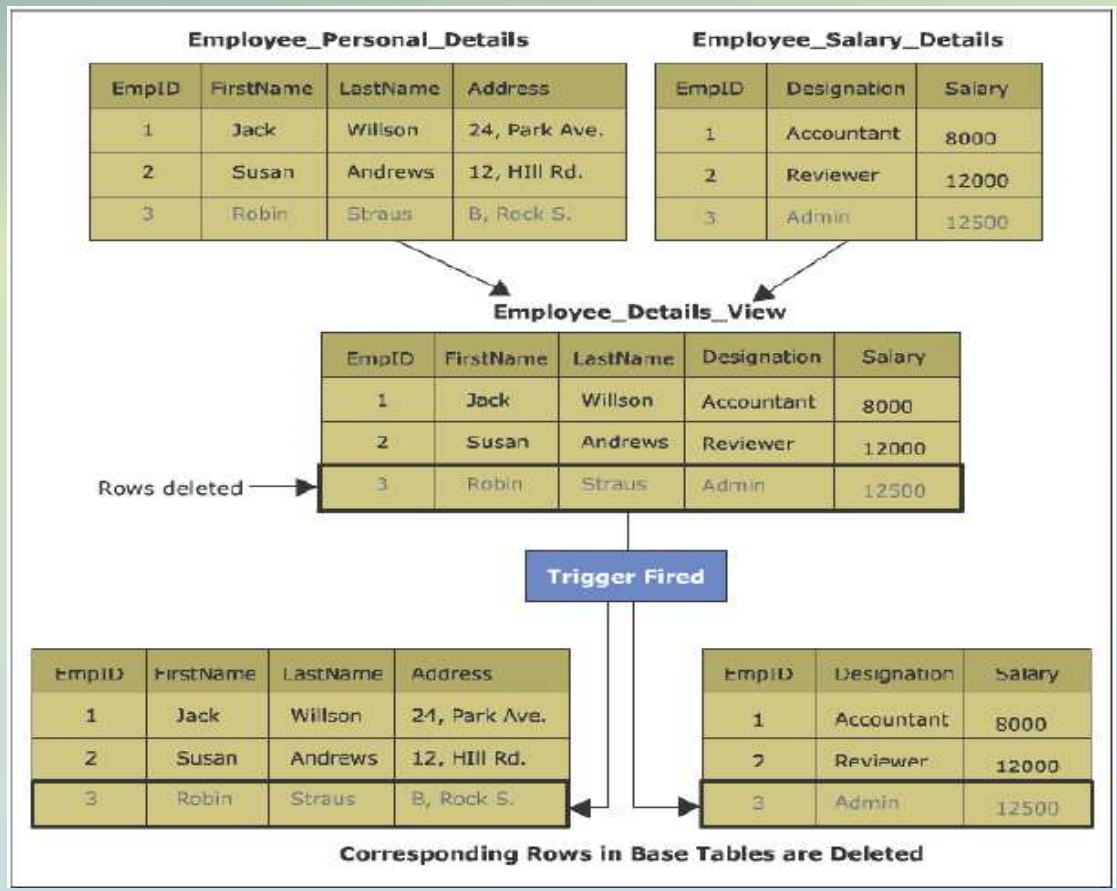
- Đoạn code dưới đây tạo `INSTEAD OF DELETE` trigger trên bảng **`Account_Transactions`**:

```
CREATE TRIGGER Delete_AccType ON Account_Transactions
INSTEAD OF DELETE
AS
BEGIN
   DELETE FROM EmployeeDetails WHERE EmployeeID IN
   (SELECT TransactionTypeID FROM deleted)
   DELETE FROM Account_Transactions WHERE TransactionTypeID
   IN  (SELECT TransactionTypeID FROM deleted)
END
```

Sử dụng INSTEAD OF Triggers với Views 1-3

Có thể được chỉ ra cho các bảng cũng như view và cung cấp một phạm vi rộng hơn và các kiểu cập nhậtmà người dùng có thể thực hiện đối với view.

Mỗi bảng hoặc view bị giới hạn chỉ có một `INSTEAD OF` trigger cho mỗi hành động `INSERT`, `UPDATE`, hoặc `DELETE` và không thể sử dụng với view có mệnh đề `WITH CHECK OPTION`.



Sử dụng INSTEAD OF Triggers với Views 2-3

- Đoạn code dưới đây tạo bảng có tên **Employee_Personal_Details**:

```
CREATE TABLE Employee_Personal_Details
(
   EmpID int NOT NULL,
   FirstName varchar(30) NOT NULL,
   LastName varchar(30) NOT NULL,
   Address varchar(30)
)
```

- Đoạn code dưới đây tạo bảng có tên **Employee_Salary_Details**:

```
CREATE TABLE Employee_Salary_Details
(
   EmpID int NOT NULL,
   Designation varchar(30),
   Salary int NOT NULL
)
```

Sử dụng INSTEAD OF Triggers với Views 3-3

- Đoạn code dưới đây tạo view từ các bảng có tên **Employee_Personal_Details** và **Employee_Salary_Details**:

```
CREATE VIEW Employee_Details_View
AS
SELECT e1.EmpID, FirstName, LastName, Designation, Salary
FROM Employee_Personal_Details e1
JOIN Employee_Salary_Details e2
ON e1.EmpID = e2.EmpID
```

- Đoạn code dưới đây tạo `INSTEAD OF DELETE` trigger trên view **Delete_Employees** :

```
CREATE TRIGGER Delete_Employees
ON Employee_Details_View
INSTEAD OF DELETE
AS
BEGIN
DELETE FROM Employee_Salary_Details WHERE EmpID IN
(SELECT EmpID FROM deleted)
DELETE FROM Employee_Personal_Details WHERE EmpID IN
(SELECT EmpID FROM deleted)
```

- Đoạn code dưới đây xóa dòng từ view **Employee_Details_View** :

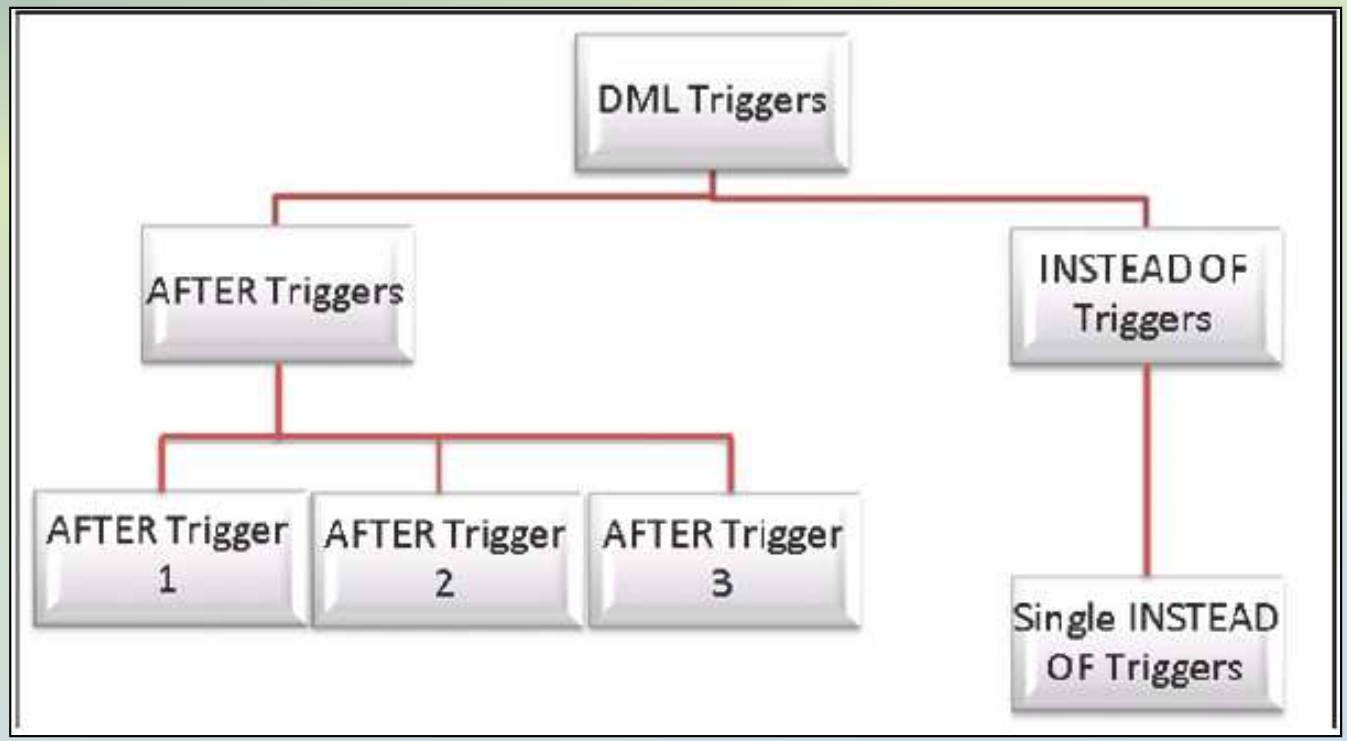
```
DELETE FROM Employee_Details _View WHERE EmpID='3'
```

SQL Server 2012

Làm việc với DML Triggers 1-3

Người dùng có thể tạo nhiều `AFTER` trigger cho cùng một hành động, nhưng chúng phải có tên nhau.

Một `AFTER` trigger có thể gồm có một số các câu lệnh SQL thực hiện các chức năng khác nhau.

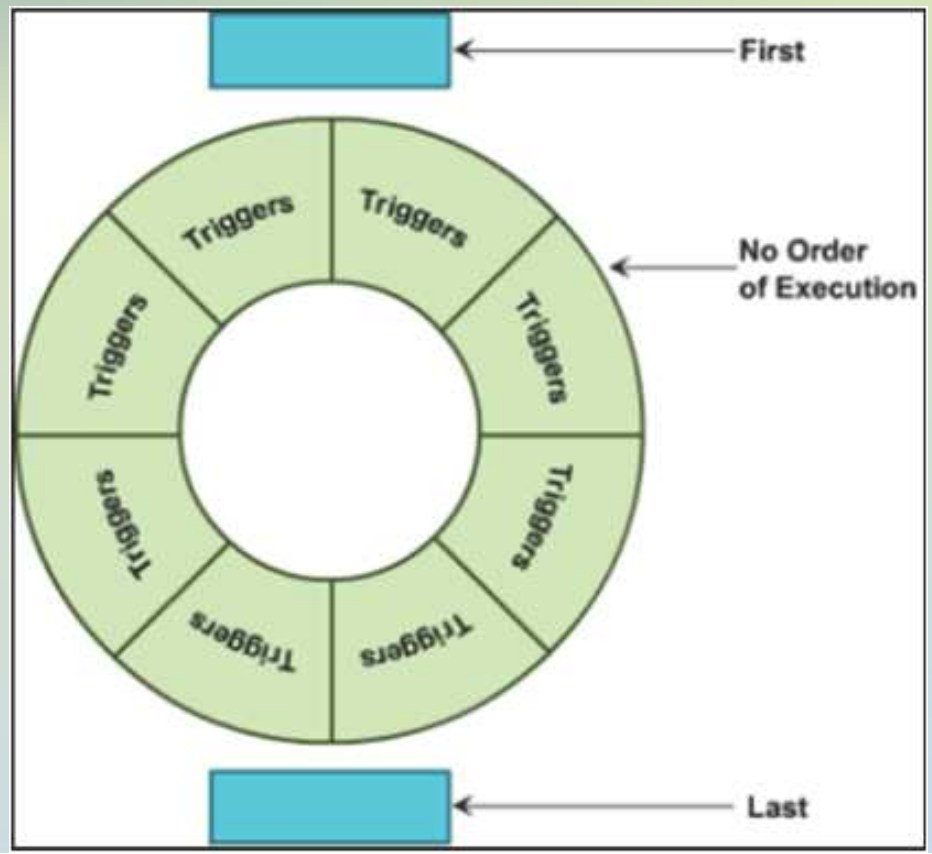


Làm việc với DML Triggers 2-3

- Thứ tự thực thi của DML Triggers

SQL Server 2012 cho phép người dùng chỉ ra thứ tự `AFTER` trigger nào được thực hiện trước và trigger nào thực hiện cuối cùng.

Tuy nhiên không thể có các trigger trên cùng bảng có cùng thứ tự đầu hoặc cuối.



Làm việc với DML Triggers 3-3

Cú pháp:

```
sp_settriggerorder [ @triggername = ] '[ triggerschema. ]
   triggername' , [ @order = ] 'value' , [ @stmttype = ]
   'statement_type'
```

Trong đó,

`[ triggerschema.] triggername`: là tên của DML hoặc DDL trigger và lược đồ mà trigger thuộc về nó và thứ tự của trigger cần được chỉ ra.

`value`: chỉ ra thứ tự thực thi của trigger như là `FIRST`, `LAST`, hoặc `NONE`. Nếu chỉ ra `FIRST` thì sau này trigger được kích hoạt đầu tiên.

`statement_type`: chỉ ra loại câu lệnh SQL (`INSERT`, `UPDATE`, hoặc `DELETE`) sẽ gọi DML trigger.

- Đoạn code sau thực thi trigger **`Employee_Deletion`** được định nghĩa trên bảng khi thao tác **`DELETE`** được thực thi:

```
EXEC sp_settriggerorder @triggername = 'Employee_Deletion ',
   @order = 'FIRST', @stmttype = 'DELETE
```

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 2012 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`.

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.

- Đoạn code dưới đây xem định nghĩa của trigger **`Employee_Deletion`**:

```
sp_helptext 'Employee_Deletion'
```

SQL Server 2012

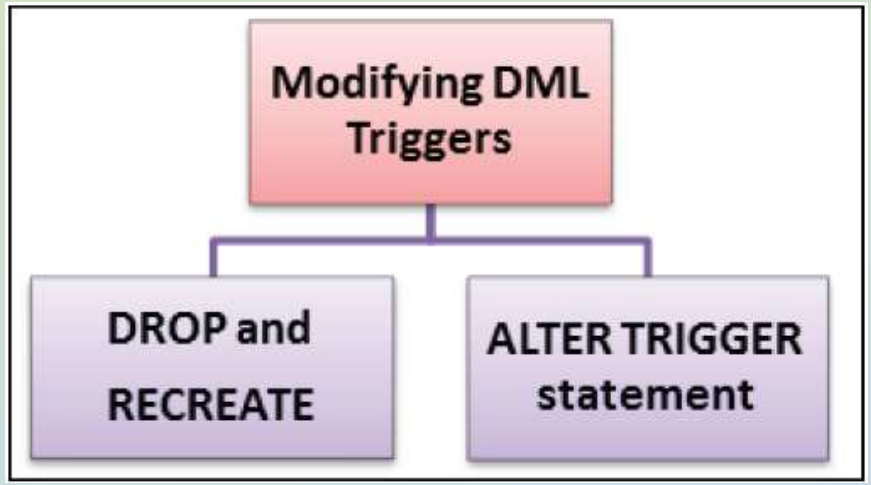
Sửa đổi định nghĩa của DML Triggers 1-3

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:

- Xóa và tạo lại trigger với các thông số mới.
- Thay đổi thông số bằng câu lệnh `ALTER TRIGGER`.

DML trigger có thể được mã hóa để ẩn đi phần định nghĩa của nó.



Sửa đổi định nghĩa của DML Triggers 2-3

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.

Sửa đổi định nghĩa của DML Triggers 3-3

- Đoạn code sau sửa đổi trigger **CheckEmployeeID** được tạo trên bảng **EmployeeDetails** bằng tùy chọn `WITH ENCRYPTION`:

```
ALTER TRIGGER CheckEmployeeID ON EmployeeDetails
WITH ENCRYPTION
FOR INSERT
AS
IF 'E01' IN (SELECT EmployeeID FROM inserted)
BEGIN
   PRINT 'User cannot insert the customers of Austria'
   ROLLBACK TRANSACTION
END
```

Kết quả

```
The text for object CheckEmployeeID is encrypted.
```

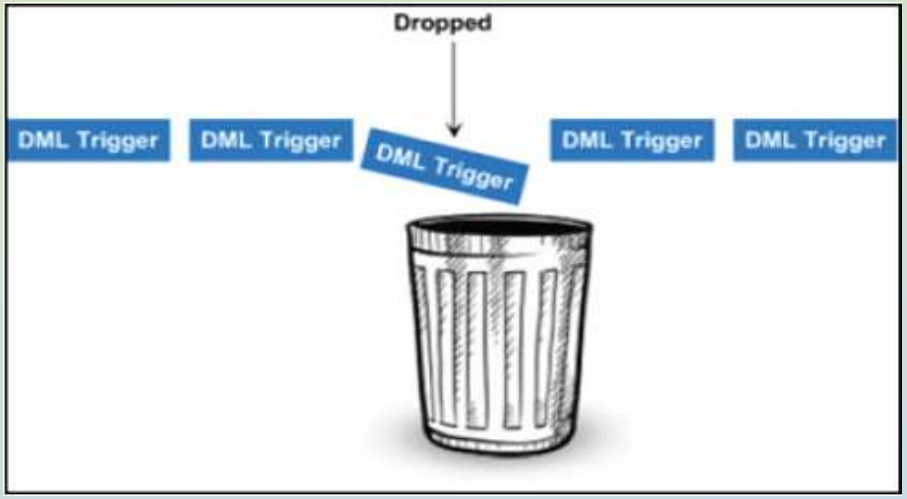
Xóa DML Triggers 1-2

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).



Xóa DML Triggers 2-2

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.

- Đoạn code sau đây xóa trigger **`CheckEmployeeID`** được tạo trên bảng **`EmployeeDetails`** :

```
DROP TRIGGER CheckEmployeeID
```

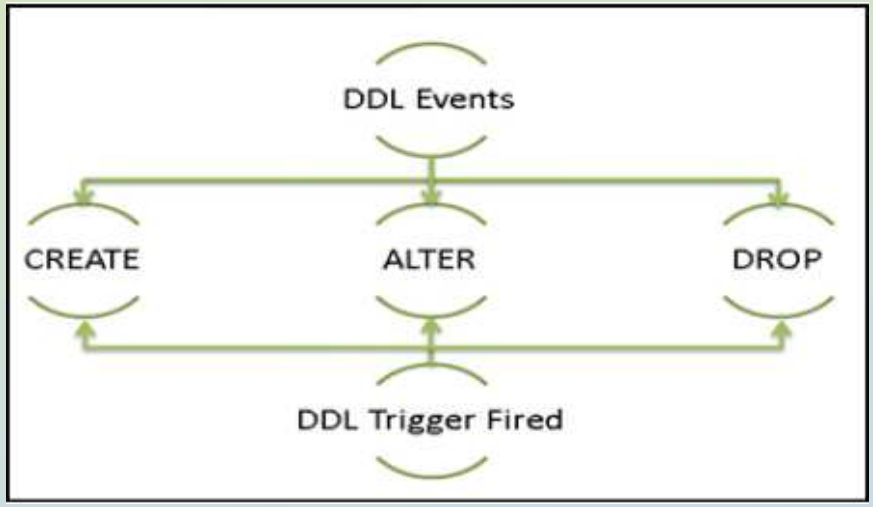
DDL Triggers 1-2

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.



DDL Triggers 2-2

Cú pháp:

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

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.

- Đoạn code dưới đây tạo một DDL trigger cho việc xóa và thây đổi bảng:

```
CREATE TRIGGER Secure
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS
PRINT 'You must disable Trigger "Secure" to drop or alter tables!'
ROLLBACK
```

SQL Server 2012

Phạm vi của DDL Triggers 1-2

DDL trigger được gọi bởi các lệnh SQL thực thi trong csdl hiện hành hoặc server hiện hành.

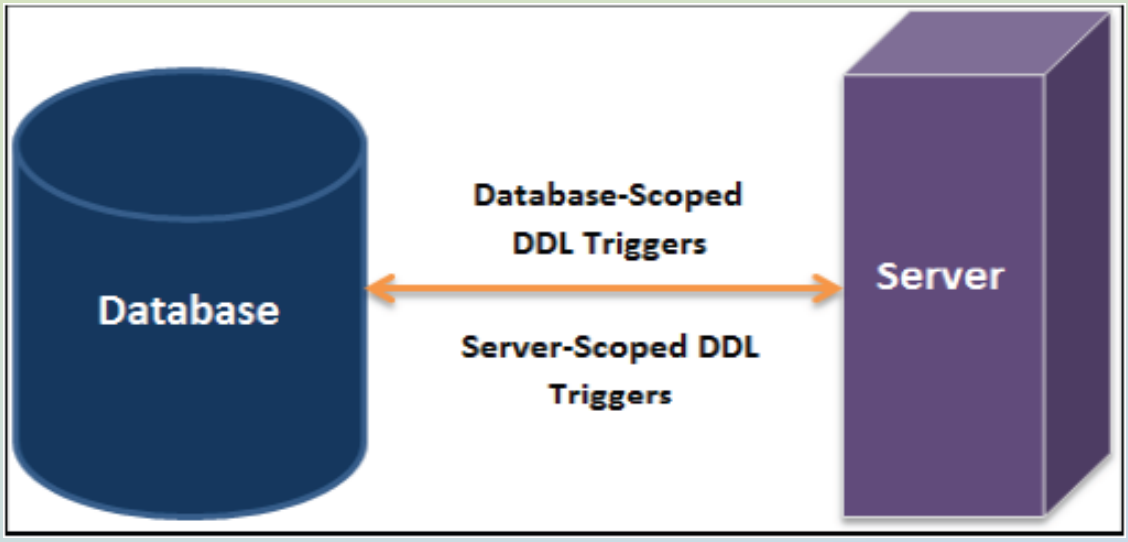
DDL trigger được tạo cho câu lệnh `CREATE LOGIN` thực thi trên sự kiện `CREATE LOGIN` trong server.

Phạm vi của DDL trigger phụ thuộc vào trigger thực thi cho các sự kiện csdl hay các sự kiện server.

Phạm vi của DDL Triggers 2-2

DDL trigger được phân thành hai loại như sau:

- DDL Triggers phạm vi csdl(Database-Scoped):
  - được gọi bởi các sự kiện sửa đổi lược đồ csdl.
  - lưu trữ các trigger trong csdl và thực thi trên sự kiện DDL, ngoại trừ các sự kiện đó có liên quan đến các bảng tạm.
- DDL Triggers phạm vi Server(Server-Scoped):
  - được gọi bởi các sự kiện DDL ở mức server.
  - được lưu trữ trong csdl master.



SQL Server 2012

Phạm vi của DDL Triggers

- The database-scoped statements are:

CREATE_APPLICATION_ROLE	ALTER_APPLICATION_ROLE	DROP_APPLICATION_ROLE
CREATE_ASSEMBLY	ALTER_ASSEMBLY	DROP_ASSEMBLY
ALTER_AUTHORIZATION_DATABASE		
CREATE_CERTIFICATE	ALTER_CERTIFICATE	DROP_CERTIFICATE
CREATE_CONTRACT	DROP_CONTRACT	
GRANT_DATABASE	DENY_DATABASE	REVOKE_DATABASE
CREATE_EVENT_NOTIFICATION	DROP_EVENT_NOTIFICATION	
CREATE_FUNCTION	ALTER_FUNCTION	DROP_FUNCTION
CREATE_INDEX	ALTER_INDEX	DROP_INDEX
CREATE_MESSAGE_TYPE	ALTER_MESSAGE_TYPE	DROP_MESSAGE_TYPE
CREATE_PARTITION_FUNCTION	ALTER_PARTITION_FUNCTION	DROP_PARTITION_FUNCTION
CREATE_PARTITION_SCHEME	ALTER_PARTITION_SCHEME	DROP_PARTITION_SCHEME
CREATE_PROCEDURE	ALTER_PROCEDURE	DROP_PROCEDURE
CREATE_QUEUE	ALTER_QUEUE	DROP_QUEUE
CREATE_REMOTE_SERVICE_BINDING	ALTER_REMOTE_SERVICE_BINDING	DROP_REMOTE_SERVICE_BINDING
CREATE_ROLE	ALTER_ROLE	DROP_ROLE
CREATE_ROUTE	ALTER_ROUTE	DROP_ROUTE
CREATE_SCHEMA	ALTER_SCHEMA	DROP_SCHEMA
CREATE_SERVICE	ALTER_SERVICE	DROP_SERVICE
CREATE_STATISTICS	DROP_STATISTICS	UPDATE_STATISTICS
CREATE_SYNONYM	DROP_SYNONYM	
CREATE_TABLE	ALTER_TABLE	DROP_TABLE
CREATE_TRIGGER	ALTER_TRIGGER	DROP_TRIGGER
CREATE_TYPE	DROP_TYPE	
CREATE_USER	ALTER_USER	DROP_USER
CREATE_VIEW	ALTER_VIEW	DROP_VIEW
CREATE_XML_SCHEMA_COLLECTION	ALTER_XML_SCHEMA_COLLECTION	DROP_XML_SCHEMA_COLLECTION

SQL Server 2012

Triggers lồng (Nested Triggers) 1-3

Cả hai DDL và DML triggers được lồng khi một trigger thực trhi một hành động mà khởi tạo một trigger khác.

DDL và DML trigger có thể lồng tới 32 mức.

Trigger lồng có thể được sử dụng để thực hiện các chức năng như lưu trữ dự phòng các dòng bị ảnh hưởng bởi các hành động trước đó.

Transact-SQL trigger thục thi các code được quản lý qua việc tham chiếu chương trình con CLR(CLR routine), tổng hợp, hoặc kiểu, mà được xem như là một mức so với 32 mức lồng nhau.

Người dùng có thể vô hiệu(disable) trigger lồng, bằng cách thiết lập tùy chọn nested triggers của thủ tục `sp_configure` là 0 để tắt.

SQL Server 2012

Triggers lồng (Nested Triggers) 2-3

- Đoạn code sau đây tạo tạo một `AFTER DELETE` trigger có tên **`Employee_Deletion`** trên bảng **`Employee_Personal_Details`** :

```
CREATE TRIGGER Employee_Deletion
ON Employee_Personal_Details
AFTER DELETE
AS
BEGIN
   PRINT 'Deletion will affect Employee_Salary_Details table'
   DELETE FROM Employee_Salary_Details WHERE EmpID IN
   (SELECT EmpID FROM deleted)
END
```

Khi một bản ghi được xóa khỏi bảng **`Employee_Personal_Details`**, trigger **`Employee_Deletion`** được kích hoạt và một thông báo được hiển thị. Bản ghi của nhân viên cúng được xóa khỏi bảng **Employee_Personal_Details**.

Triggers lồng (Nested Triggers) 3-3

- Đoạn code sau đây tạo một trigger `AFTER DELETE` có tên **`Deletion Confirmation`** trên bảng **`Employee_Personal_Details`**:

```
CREATE TRIGGER Deletion_Confirmation
ON Employee_Salary_Details
AFTER DELETE
AS
BEGIN
PRINT 'Employeedetailssuccessfullydeletedfromemployee
Salary_Detailstable'
END
```

Khi một bản ghi được xóa khỏi bảng **Employee_Salary_Details**, trigger **Deletion_Confirmation** được kích hoạt. Trigger này in ra thông điệp xác nhận số bản ghi bị xóa.

Qua đó, cho thấy trigger **Employee_Personal_Details** và **Employee_Deletion** là được lồng nhau.

UPDATE() 1-2

Trả về giá trị Boolean xác định xem cột của view hoặc của bảng có bị thao tác bởi hành động `UPDATE` hoặc `INSERT` hay không

Có thể được sử dụng bất cứ nơi nào bên trong thân của một Transact-SQL `UPDATE` hoặc `INSERT` trigger để kiểm tra xem trigger nên thực hiện một số hành động.

Cú pháp:

```
UPDATE ( column )
```

Trong đó,

`column`: là tên của cột để kiểm tra xem có `INSERT` hay `UPDATE`.

UPDATE 2-2

- Đoạn code sau đây tạo một trigger `Accounting` trên bảng **`Account_Transactions`** để cập nhật cột **`TransactionID`** hoặc **`EmployeeID`**:

```
CREATE TRIGGER Accounting
ON Account_Transactions
AFTER UPDATE
AS
IF ( UPDATE (TransactionID) OR UPDATE (EmployeeID) )
BEGIN
   RAISERROR (50009, 16, 10)
END;
GO
```

Xử lý đa dòng trong một phiên (Session)

Khi một người dùng viết mã cho DML trigger, sau đó câu lệnh gây ra trigger được kích hoạt sẽ là một câu lệnh đơn.

Câu lệnh đơn sẽ tác động nhiều dòng dữ liệu, thay vì một dòng duy nhất.

Khi các chức năng của một DML trigger có liên quan đến việc tự động tính lại tổng các giá của một bảng và lưu trữ kết quả trong bảng khác, thì việc cân nhắc đa dòng là rất quan trọng.

- Đoạn code sau đây lưu trữ tổng (running total) cho thao tác insert duy nhất một dòng chèn:

```
USE AdventureWorks2012;
GO
CREATE TRIGGER PODetails
ON Purchasing.PurchaseOrderDetail
AFTER INSERT AS
UPDATE PurchaseOrderHeader
SET SubTotal = SubTotal + LineTotal
FROM inserted
WHERE PurchaseOrderHeader.PurchaseOrderID = inserted.PurchaseOrderID;
```

Xử lý đa dòng trong một phiên (Session)

- Đoạn code sau đây lưu trữ tổng (running total) cho thao tác insert duy nhất một dòng chèn:

```
USE AdventureWorks2012;
GO
CREATE TRIGGER PODetailsMultiple
ON Purchasing.PurchaseOrderDetail
AFTER INSERT
AS
UPDATEPurchasing.PurchaseOrderHeader
SET SubTotal = SubTotal +
         (SELECT Sum(LineTotal)
         FROM inserted
         WHEREP urchaseOrderHeader.PurchaseOrderID
         = inserted.PurchaseOrderID)
WHERE PurchaseOrderHeader. PurchaseOrderID IN
         (SELECT PurchaseOrderID FROM inserted) ;
```

Trong đoạn code này, các tổng con(subtotal) được tính toán và lưu trữ cho thao tác chèn vào nhiều dòng hoặc duy nhất một dòng.

SQL Server 2012

Tóm tắt

- 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.
- 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.

SQL Server 2012

Hệ quả hiệu suất của Trigger

Trigger thực thi không tốn chi phí, đúng hơn là chúng đáp ứng khá nhanh.

Many performance issues can occur because of the logic present inside the trigger.

Một nguyên tắc hay sẽ giữ logic đơn giản bên trong trigger và tránh sử dụng cursor trong khi đang thực hiện các câu lệnh với bảng khác và các bài toán khác nhau là nguyên nhân gây ra hiệu suất chậm lại.