

Session - 12

Triggers

Welcome to the Session, **Triggers**.

This session explains the triggers and different types of triggers. The session also describes the procedure to create and alter DML triggers, nested triggers, update functions, handling multiple rows in a session, and performance implication of triggers.

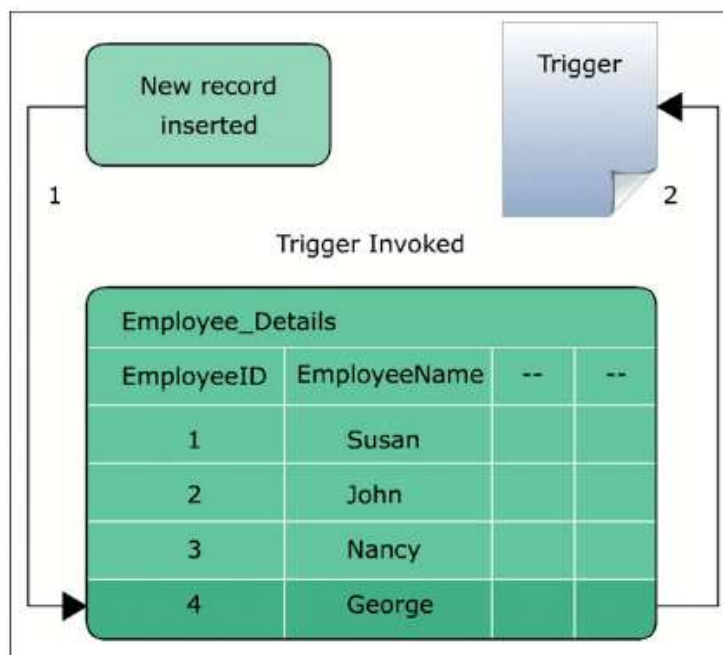
In this Session, you will learn to:

- Explain triggers
- Explain the different types of triggers
- Explain the procedure to create DML triggers
- Explain the procedure to alter DML triggers
- Describe nested triggers
- Describe update functions
- Explain the handling of multiple rows in a session
- Explain the performance implication of triggers

12.1 Giới thiệu

Trigger là một thủ tục lưu trữ được thực hiện khi có một nỗ lực để sửa đổi dữ liệu trong bảng được bảo vệ bằng trigger. Không giống như các thủ tục lưu trữ hệ thống tiêu chuẩn, không thể thực thi các trigger trực tiếp, chúng cũng không truyền hoặc nhận các tham số. Trigger được định nghĩa trên các bảng cụ thể và những bảng này được gọi là bảng trigger.

Nếu trigger được định nghĩa trên hành động INSERT, UPDATE, hoặc DELETE trên một bảng, nó tự hủy khi những hành động này được thử. Sự thực thi tự động này của trigger không thể bị phá vỡ. Trong SQL Server 2012, trigger được tạo ra bằng cách sử dụng câu lệnh CREATE TRIGGER. Hình 12.1 hiển thị ví dụ về trigger.



Hình 12.1: Triggers

12.2 Uses of Triggers

Trigger có thể chứa logic xử lý phức tạp và thường được sử dụng để duy trì tính toàn vẹn dữ liệu ở mức độ thấp. Cách dùng chính của trigger có thể được phân loại như sau:

➤ Phân tầng thay đổi thông qua các bảng liên quan

Người dùng có thể sử dụng một trigger cho các thay đổi tầng thông qua các bảng liên quan. Ví dụ, hãy xem xét bảng **Salary_Details** có **FOREIGN KEY**, **EmpID** tham chiếu tới **PRIMARY KEY**, **EmpID** của bảng **Employee_Details**. Nếu sự kiện cập nhật hoặc xóa xảy ra trong bảng **Employee_Details**, trigger cập nhật hoặc xóa có thể được định nghĩa để xếp tầng những thay đổi đối với bảng **Salary_Details**.

➤ **Bắt buộc tính toàn vẹn dữ liệu phức tạp hơn các ràng buộc CHECK**

Không giống như các hạn chế CHECK, trigger có thể tham khảo các cột trong các bảng khác. Tính năng này có thể được sử dụng để áp dụng các kiểm tra tính toàn vẹn dữ liệu phức tạp. Tính toàn vẹn dữ liệu có thể được thực thi bằng cách:

- Kiểm tra các ràng buộc trước khi xếp tầng các cập nhật hoặc xóa.
- Tạo ra các trigger nhiều hàng cho các hành động được thực hiện trên nhiều hàng.
- Thực thi tính toàn vẹn tham chiếu giữa các cơ sở dữ liệu.

➤ **Định nghĩa các thông báo lỗi tùy chỉnh**

Các thông báo lỗi tùy chỉnh được sử dụng để cung cấp lời giải thích phù hợp hoặc chi tiết hơn trong các tình huống lỗi nhất định. Có thể sử dụng các trigger để gọi các thông báo lỗi tùy chỉnh được xác định trước như vậy khi các điều kiện lỗi có liên quan xảy ra.

➤ **Duy trì dữ liệu phi tiêu chuẩn hóa**

Tính toàn vẹn dữ liệu mức thấp có thể được duy trì trong các môi trường cơ sở dữ liệu khi tiêu chuẩn hóa sử dụng các trigger. Dữ liệu khi tiêu chuẩn hóa thường đề cập đến dữ liệu dư thừa hoặc bất nguồn. Ở đây, các trigger được sử dụng cho các kiểm tra không yêu cầu các so khớp chính xác. Ví dụ, nếu giá trị của năm sẽ được kiểm tra đối với các ngày hoàn tất, trigger có thể được sử dụng để thực hiện kiểm tra.

➤ **So sánh trước và sau các trạng thái của dữ liệu đang được sửa đổi**

Trigger cung cấp tùy chọn này để tham khảo các thay đổi được thực hiện cho dữ liệu bằng các câu lệnh INSERT, UPDATE, và DELETE. Điều này cho phép người dùng tham khảo các hàng bị ảnh hưởng khi các sửa đổi được thực hiện thông qua các trigger.

12.3 Các loại Triggers

Trigger có thể được đặt để thực thi tự động một hành động khi một sự kiện ngôn ngữ xảy ra trong một bảng hoặc khung nhìn. Các sự kiện ngôn ngữ có thể được phân loại là các sự kiện DML và các sự kiện DDL. Trigger liên quan đến các sự kiện DML được gọi là trigger DML, trong khi trigger kết hợp với các sự kiện DDL được gọi là trigger DDL.

Trigger trong SQL Server 2012 có thể được phân thành ba loại cơ bản:

➤ **DML Triggers**

Các trigger DML thực thi khi dữ liệu được chèn, sửa đổi hoặc xóa trong một bảng hoặc khung nhìn sử dụng câu lệnh INSERT, UPDATE, hoặc DELETE.

➤ **DDL Triggers**

Các trigger DDL thực thi khi một bảng hoặc khung nhìn được tạo ra, chỉnh sửa hoặc xóa bằng cách sử dụng câu lệnh CREATE, ALTER, hoặc DROP.

➤ **Logon Triggers**

Các trigger đăng nhập thực thi các thủ tục lưu trữ khi một phiên được thiết lập với một sự kiện LOGON. Những trigger này được gọi sau khi chứng thực đăng nhập được hoàn thành và trước khi phiên thực tế được thiết lập. Các trigger đăng nhập kiểm soát các phiên máy chủ bằng cách hạn chế các thông tin đăng nhập không hợp lệ hoặc hạn chế số lượng phiên.

12.4 Các trigger DDL so với các trigger DML

Các trigger DDL và DML có cách sử dụng khác nhau và được thực thi với các sự kiện cơ sở dữ liệu khác nhau. Bảng 12.1 liệt kê ra những khác biệt giữa các trigger DDL và DML.

DDL Triggers	DML Triggers
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 kinh doanh khi dữ liệu được sửa đổi trong các bảng hoặc khung nhìn.
Các trigger DDL chỉ hoạt động sau khi bảng hoặc khung nhìn 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.

Table 12.1: Differences between DDL and DML Triggers

12.5 Tạo DML Triggers

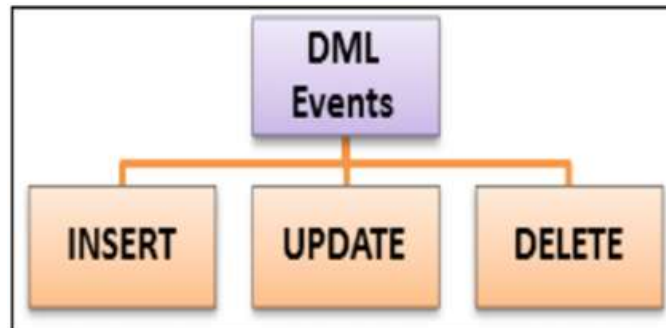
Các trigger DML được thực thi khi sự kiện DML xảy ra trong các bảng hoặc khung nhìn. Những sự kiện DML này bao gồm các câu lệnh INSERT, UPDATE, và DELETE. Các trigger DML có thể thực thi khi hoàn thành các sự kiện DML hoặc ở vị trí của các sự kiện DML.

Các trigger DML bắt buộc tính toàn vẹn tham chiếu bằng cách xếp tầng các thay đổi tới các bảng có liên quan khi một hàng được sửa đổi. Các trigger DML có thể thực hiện nhiều hành động cho mỗi câu lệnh sửa đổi.

Các trigger DML thuộc ba loại chính:

- **INSERT trigger**
- **UPDATE trigger**
- **DELETE trigger**

Hình 12.2 hiển thị các loại trigger DML.



Hình 12.2: DML Triggers

12.5.1 Giới thiệu về các bảng đã chèn và xóa

Những câu lệnh SQL trong các trigger DML sử dụng hai loại bảng đặc biệt để sửa đổi dữ liệu trong cơ sở dữ liệu. Khi dữ liệu được chèn, cập nhật, hoặc xóa, SQL Server 2012 tạo ra và quản lý những bảng này một cách tự động. Các bảng tạm thời lưu trữ dữ liệu ban đầu cũng như dữ liệu đã sửa đổi. Những bảng này như sau:

➤ **Bảng đã chèn**

Bảng Inserted chứa bản sao các bản ghi được sửa đổi với hoạt động INSERT và UPDATE trên bảng trigger. Bảng trigger là bảng trên đó trigger được định nghĩa. Hoạt động INSERT và UPDATE chèn các bản ghi mới vào bảng Inserted và Trigger.

➤ **Bảng đã xóa**

Bảng Deleted chứa bản sao các bản ghi được sửa đổi với hoạt động DELETE và UPDATE trên bảng trigger. Bảng trigger là bảng trên đó trigger được định nghĩa. Những hoạt động này xóa các bản ghi từ bảng trigger và chèn chúng vào bảng Deleted.

Ghi chú - Bảng Inserted và Deleted về khía cạnh vật lý không thể giữ hiện diện trong cơ sở dữ liệu. Chúng được tạo ra và thả bỏ bất cứ khi nào bất kỳ sự kiện kích hoạt nào xảy ra

12.5.2 Insert Triggers

Trigger INSERT được thực thi khi một bản ghi mới được chèn vào bảng. Trigger INSERT đảm bảo rằng giá trị đang được nhập vào phù hợp với các ràng buộc được định nghĩa trên bảng đó.

Khi người dùng chèn một bản ghi vào bảng đó, trigger INSERT lưu một bản sao của bản ghi đó vào bảng Inserted. Sau đó nó sẽ kiểm tra xem giá trị mới trong bảng Inserted phù hợp với các ràng buộc đã chỉ định.

Nếu bạn ghi hợp lệ, trigger INSERT chèn hàng này vào bảng trigger, nếu không nó sẽ hiển thị thông báo lỗi.

Trigger INSERT được tạo ra sử dụng từ khóa INSERT trong câu lệnh **CREATE TRIGGER** và **ALTER TRIGGER**.

Sau đây là cú pháp cho việc tạo ra trigger INSERT.

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 có chứa bảng/trigger đó. **trigger_name:** chỉ ra tên của trigger.

table_name: chỉ ra bảng mà trên đó trigger DML được tạo ra.

WITH ENCRYPTION: mã hóa văn bản của câu lệnh **CREATE TRIGGER**.

FOR: chỉ ra rằng trigger DML thực thi sau khi các hoạt động sửa đổi được hoàn tất.

INSERT: chỉ ra rằng trigger DML này sẽ được gọi bởi các hoạt động chèn.

UPDATE: Trả về giá trị Boolean cho biết xem lần thử **INSERT** hoặc **UPDATE** đã được thực hiện hay chưa trên một cột cụ thể.

column_name: Là tên của cột để kiểm tra về hành động **UPDATE**.

AND: Kết hợp hai biểu thức Boolean và trả về **TRUE** khi cả hai biểu thức là **TRUE**.

OR: Kết hợp hai biểu thức Boolean và trả về **TRUE** nếu ít nhất một biểu thức là **TRUE**.

sql_statement: chỉ ra các câu lệnh SQL được thực thi trong trigger DML.

Code Snippet 1 tạo ra trigger INSERT trên bảng có tên là **Account_Transactions**. Khi một bản ghi mới được đưa vào, và nếu số tiền rút vượt quá **80000**, trigger chèn sẽ hiển thị thông báo lỗi và quay lui giao tác sử dụng câu lệnh ROLLBACK TRANSACTION.

Giao tác là tập hợp một hoặc nhiều câu lệnh được xử lý như một đơn vị công việc. Giao tác có thể thành công hoặc thất bại thành một khối, do đó tất cả các câu lệnh trong đó cùng thành công hay thất bại. Câu lệnh ROLLBACK TRANSACTION hủy bỏ hoặc quay lui một giao tác.

Code Snippet 1:

```
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
```

Code Snippet 2 chèn một bản ghi nơi **Số tiền rút** vượt quá **80000**. Điều này làm cho trigger INSERT hiển thị thông báo lỗi và quay lui giao tác đó.

Code Snippet 2:

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

Thông báo lỗi sau được hiển thị như được chỉ ra bằng câu lệnh PRINT: Số tiền rút không thể vượt quá 80000.

12.5.3 Update Triggers

Trigger UPDATE sao chép bản ghi gốc vào bảng Deleted và bản ghi mới vào bảng Inserted khi bản ghi được cập nhật. Sau đó nó đánh giá bản ghi mới để xác định xem các giá trị này có phù hợp với các ràng buộc đã chỉ định trong bảng trigger hay không.

Nếu các giá trị mới là hợp lệ, bản ghi từ bảng Inserted được sao chép vào bảng trigger. Tuy nhiên, nếu các giá trị mới là không hợp lệ, thông báo lỗi được hiển thị. Ngoài ra, bản ghi gốc được sao chép từ bảng Deleted trở lại vào bảng trigger.

Trigger UPDATE được tạo ra sử dụng từ khóa UPDATE trong câu lệnh **CREATE TRIGGER** và **ALTER TRIGGER**. Sau đây là cú pháp cho việc tạo ra trigger UPDATE ở mức bảng.

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,

FOR UPDATE: chỉ ra rằng trigger DML này sẽ được gọi sau các hoạt động cập nhật.

Code Snippet 3 tạo ra trigger UPDATE ở mức bảng trên bảng **EmployeeDetails**. Khi một bản ghi được sửa đổi, trigger UPDATE được kích hoạt. Nó sẽ kiểm tra xem ngày tháng năm sinh có lớn hơn ngày hôm nay không. Nó sẽ hiển thị thông báo lỗi cho các giá trị không hợp lệ và quay lui hoạt động sửa đổi bằng cách sử dụng câu lệnh **ROLLBACK TRANSACTION**.

Code Snippet 3:

```
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
```


Code Snippet 4 cập nhật một bản ghi trong đó ngày tháng năm sinh không hợp lệ được chỉ định. Điều này làm cho trigger cập nhật hiển thị thông báo lỗi và quay lui giao tác đó.

Code Snippet 4:

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

Thông báo lỗi sau được hiển thị như được chỉ ra bằng câu lệnh **PRINT**:

Ngày tháng năm sinh không thể lớn hơn ngày hôm nay.

➤ **Tạo các Update Triggers**

Những trigger UPDATE được tạo ra ở mức cột hoặc ở mức bảng. Những trigger ở mức cột thực thi khi các lần cập nhật được thực hiện trong cột chỉ định. Những trigger ở mức bảng thực thi khi các lần cập nhật được thực hiện bất cứ đâu trong toàn bộ bảng.

Để tạo ra một trigger UPDATE ở mức cột, hàm UPDATE() được sử dụng để chỉ ra cột này.

Code Snippet 5 tạo ra trigger UPDATE ở mức cột trên cột **EmployeeID** của bảng **EmployeeDetails**. Khi **EmployeeID** được sửa đổi, trigger UPDATE được kích hoạt và thông báo lỗi được hiển thị. Hoạt động sửa đổi được quay lui bằng cách sử dụng câu lệnh **ROLLBACK TRANSACTION**.

Code Snippet 5:

```
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
```

Code Snippet 6 cập nhật một bản ghi trong đó giá trị trong cột **EmployeeID** đang được sửa đổi. Điều này làm cho trigger cập nhật hoạt động. Trigger cập nhật hiển thị thông báo lỗi và quay lại giao tác đó.

Code Snippet 6:

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

12.5.4 Delete Triggers

Có thể tạo ra trigger DELETE để hạn chế người dùng không xóa một bản ghi cụ thể trong bảng. Điều sau đây sẽ xảy ra nếu người dùng cố gắng xóa bản ghi:

- Bản ghi bị xóa khỏi bảng trigger và chèn vào bảng Deleted.
- Nó được kiểm tra về các ràng buộc đối với việc xóa.
- Nếu có ràng buộc trên bản ghi để ngăn chặn việc xóa, trigger **DELETE** sẽ hiển thị thông báo lỗi.
- Bản ghi đã xóa được lưu trữ trong bảng Deleted được sao chép ngược lại bảng trigger.

Trigger DELETE được tạo ra bằng cách sử dụng từ khóa **DELETE** trong câu lệnh **CREATE TRIGGER**. Sau đây là cú pháp cho việc tạo ra một trigger **DELETE**.

Cú pháp:

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

trong đó,

DELETE: chỉ ra rằng trigger DML này sẽ được gọi bởi các hoạt động xóa.

Code Snippet 7 tạo ra một trigger DELETE trên bảng **Account_Transactions**. Nếu bản ghi của một **TransactionID** bị xóa, trigger DELETE được kích hoạt và thông báo lỗi được hiển thị. Hoạt động xóa được quay lui sử dụng câu lệnh **ROLLBACK TRANSACTION**.

Code Snippet 7:

```
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
```

Code Snippet thử xóa các bản ghi từ bảng **Account_Transactions** trong đó **Deposit** là 50000.

Code Snippet 8:

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

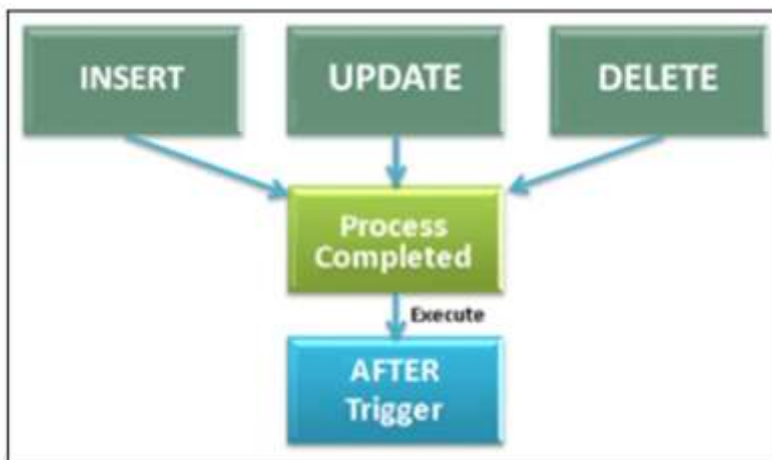
Thông báo lỗi được hiển thị như được chỉ ra bằng câu lệnh PRINT.

Người dùng không thể xóa những giao tác này.

12.6 Các Triggers After

Trigger AFTER được thực thi khi hoàn thành các hoạt động INSERT, UPDATE, hoặc DELETE. Trigger AFTER có thể được tạo ra chỉ trên các bảng. Bảng có thể có nhiều trigger AFTER được định nghĩa cho mỗi hoạt động INSERT, UPDATE, và DELETE. Nếu nhiều trigger AFTER được tạo ra trên cùng một bảng, người dùng phải định nghĩa thứ tự theo đó những trigger này phải được thực thi. Trigger AFTER được thực thi khi kiểm tra ràng buộc trong bảng đã hoàn tất. Ngoài ra, trigger được thực thi sau khi các bảng Inserted và Deleted được tạo ra.

Hình 12.3 hiển thị các loại trigger AFTER.



Hình 12.3: AFTER Triggers

Sau đây là cú pháp cho việc tạo ra trigger AFTER.

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 rằng trigger DML thực thi sau khi các hoạt động sửa đổi được hoàn tất.

{ [INSERT] [,] [UPDATE] [,] [DELETE] } : chỉ ra những hoạt động gọi trigger DML.

Code Snippet 9 tạo ra một trigger **AFTER DELETE** trên bảng **EmployeeDetails**. Nếu có bất kỳ bản ghi nhân viên nào bị xóa khỏi bảng, trigger **AFTER DELETE** được kích hoạt. Trigger sẽ hiển thị số lượng bản ghi nhân viên đã xóa khỏi bảng này.

Code Snippet 9:

```
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
```

Code Snippet 10 xóa một bản ghi khỏi bảng **EmployeeDetails**.

Code Snippet 10:

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

Thông báo lỗi sau được hiển thị:

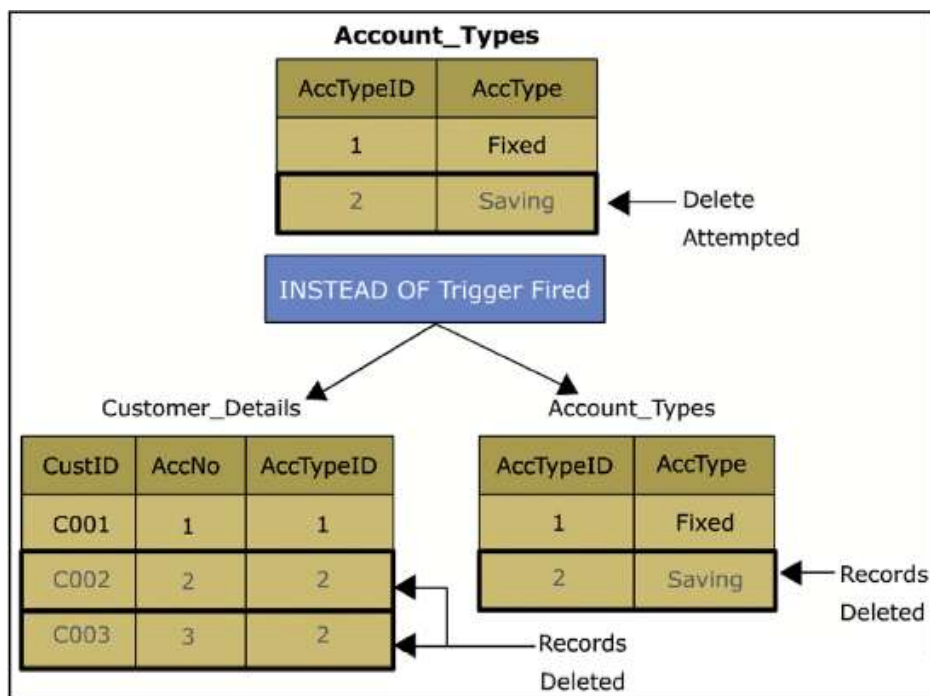
Số lượng nhân viên đã xóa = 0

12.7 Các trigger **INSTEAD OF**

Trigger **INSTEAD OF** được thực thi thay cho các hoạt động **INSERT**, **UPDATE**, hoặc **DELETE**. Các trigger **INSTEAD OF** có thể được tạo ra trên các bảng cũng như khung nhìn. Bảng hoặc khung nhìn có thể chỉ có một trigger **INSTEAD OF** được định nghĩa cho mỗi hoạt động **INSERT**, **UPDATE**, và **DELETE**.

Các trigger **INSTEAD OF** được thực thi trước khi các lần kiểm tra ràng buộc được thực hiện trên bảng này. Những trigger này được thực thi sau khi việc tạo ra các bảng Inserted và Deleted. Các trigger **INSTEAD OF** tăng sự đa dạng của các loại cập nhật mà người dùng có thể thực hiện đối với khung nhìn.

Hình 12.4 hiển thị ví dụ về các trigger INSTEAD OF.



Hình 12.4: INSTEAD OF Triggers

Trong ví dụ được trình bày trong hình 12.4, trigger INSTEAD OF DELETE trên bảng **Account_Types** được tạo ra. Nếu có bất kỳ bản ghi nào trong bảng **Account_Types** bị xóa, những bản ghi tương ứng trong **Customer_Details** cũng sẽ được gỡ bỏ. Do đó, thay vì làm việc chỉ trên một bảng, ở đây, trigger đảm bảo rằng hoạt động xóa được thực hiện trên cả hai bảng.

Ghi chú - Người dùng không thể tạo các trigger **INSTEAD OF** cho các hoạt động xóa hoặc cập nhật trên các bảng có các tùy chọn tằng **ON DELETE** và tằng **ON UPDATE** đã chọn.

The following is the Cú pháp for creating an INSTEAD OF trigger.

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 khung nhìn mà trên đó trigger DML được tạo ra.

INSTEAD OF: chỉ ra rằng trigger DML thực thi thay cho các hoạt động sửa đổi. Những trigger này không được định nghĩa trên các khung nhìn có thể cập nhật sử dụng **WITH CHECK OPTION**.

Code Snippet 11 tạo ra trigger **INSTEAD OF DELETE** trên bảng **Account_Transactions**. Nếu có bất kỳ bản ghi nào trong bảng **Account_Transactions** bị xóa, những bản ghi tương ứng trong bảng **EmployeeDetails** sẽ được gỡ bỏ.

Code Snippet 11:

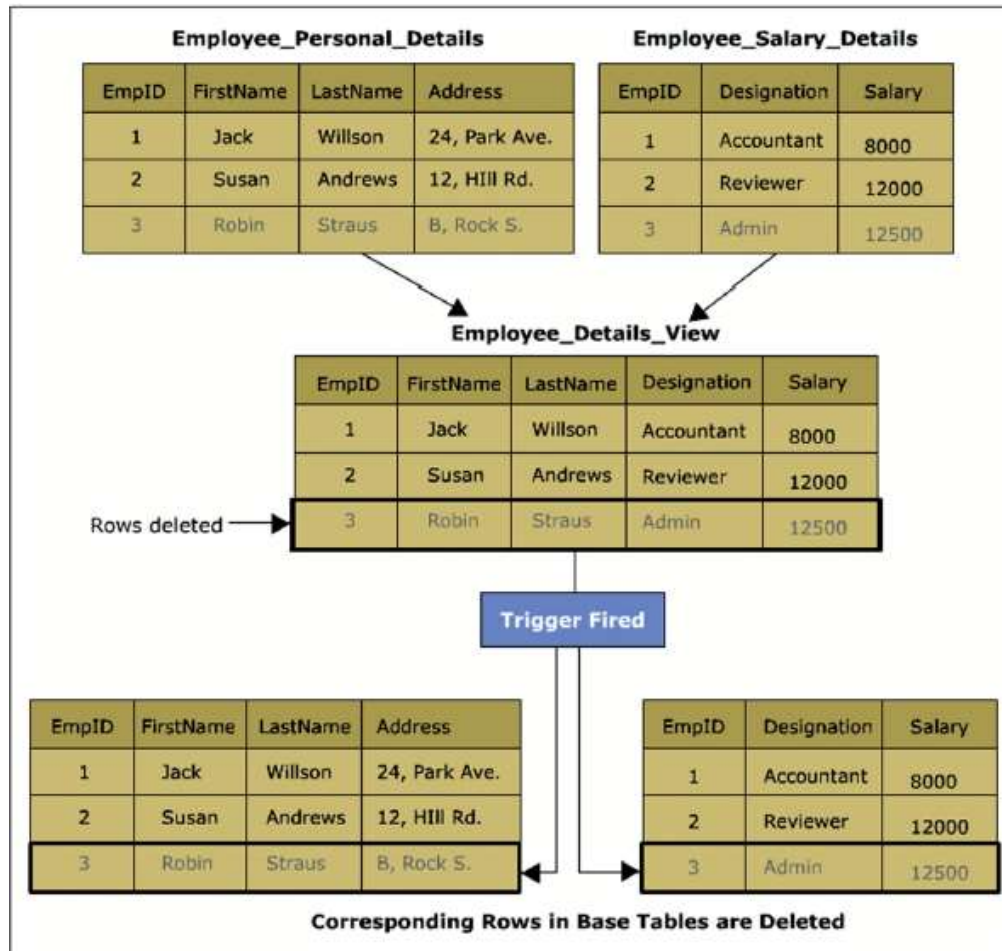
```
CREATE TRIGGER Delete_AccType
ON Account_Transactions
INSTEAD OF DELETE
AS
BEGIN
DELETE FROM EmployeeDetails WHERE EmployeeID IN
(SELECT TransactionTypeID FROM deleted)
DELETE FROM Account_Transactions WHERE TransactionTypeID IN
(SELECT TransactionTypeID FROM deleted)
END
```

12.7.1 Sử dụng các trigger **INSTEAD OF** với các khung nhìn

Các trigger **INSTEAD OF** có thể được chỉ ra trên các bảng cũng như khung nhìn. Trigger này thực thi thay vì hành động kích hoạt ban đầu. Các trigger **INSTEAD OF** cung cấp phạm vi rộng hơn và các loại cập nhật mà người dùng có thể thực hiện đối với khung nhìn. Mỗi bảng hoặc khung nhìn được giới hạn chỉ cho một trigger **INSTEAD OF** cho mỗi hành động kích hoạt (**INSERT**, **UPDATE**, hoặc **DELETE**).

Người dùng không thể tạo ra một trigger **INSTEAD OF** trên các khung nhìn có mệnh đề **WITH CHECK OPTION** đã định nghĩa.

Hình 12.5 hiển thị ví dụ về việc sử dụng các trigger **INSTEAD OF** với các khung nhìn.



Hình 12.5:Using **INSTEAD OF** Triggers with Views

Code Snippet 12 tạo ra một bảng có tên là **Employee_Personal_Details**.

Code Snippet 12:

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

Code Snippet 13 tạo ra một bảng có tên là **Employee_Salary_Details**.

Code Snippet 13:

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

Code Snippet 14 tạo ra một khung nhìn **Employee_Details_View** sử dụng các cột từ bảng **Employee_Personal_Details** và **Employee_Salary_Details** bằng cách nối hai bảng trên cột **EmpID**.

Code Snippet 14:

```
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
```

Code Snippet 15 tạo ra trigger INSTEAD OF DELETE **Delete_Employees** trên khung nhìn **Employee_Details_View**. Khi một hàng được xóa khỏi khung nhìn, trigger được kích hoạt. Nó xóa những bản ghi tương ứng khỏi những bảng cơ sở của khung nhìn có tên là **Employee_Personal_Details** và **Employee_Salary_Details**.

Code Snippet 15:

```
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)
```

Code Snippet 16 xóa một hàng khỏi khung nhìn **Employee_Details_View** trong đó **EmpID='2'**.

Code Snippet 16:

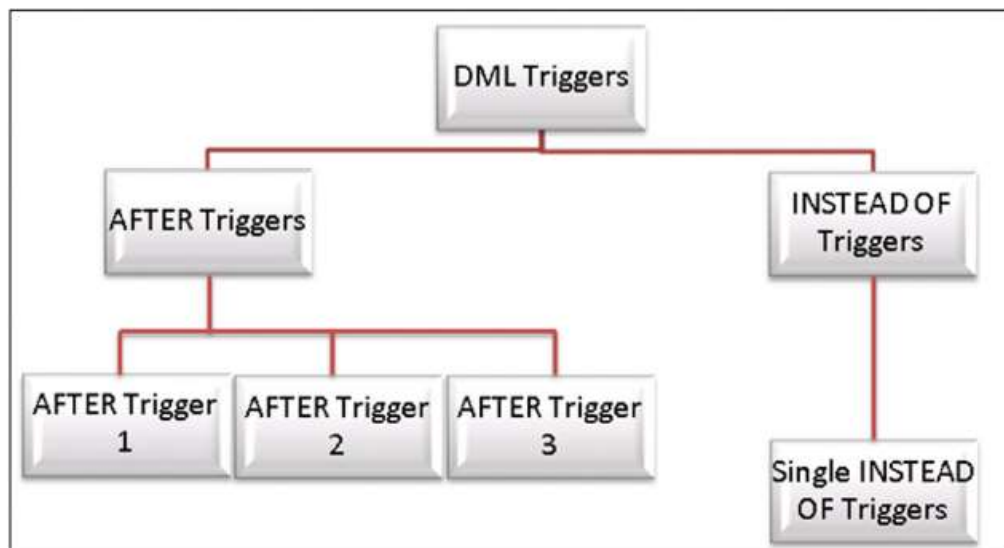
```
DELETE FROM Employee_Details_View WHERE EmpID='3'
```

12.8 Làm việc với các trigger DML

SQL Server 2012 cho phép người dùng tạo ra nhiều trigger AFTER cho mỗi hành động kích hoạt (như là **UPDATE**, **INSERT**, và **DELETE**) trên một bảng. Tuy nhiên, người dùng có thể tạo ra chỉ một trigger INSTEAD OF cho mỗi hành động kích hoạt trên một bảng.

Khi người dùng có nhiều trigger AFTER trên một hành động kích hoạt, tất cả những trigger này phải có một tên khác. Trigger AFTER có thể bao gồm một số câu lệnh SQL thực hiện các hàm khác nhau.

Hình 12.6 hiển thị những loại trigger DML.



Hình 12.6: Types of DML Triggers

Ghi chú - Một trigger AFTER đơn lẻ có thể được gọi bởi nhiều hơn một hành động kích hoạt

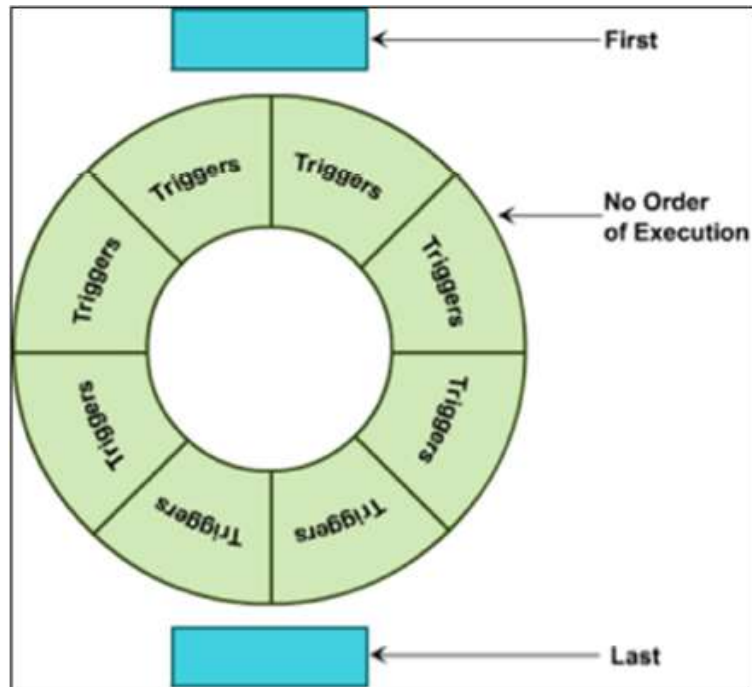
➤ **Thứ tự thực thi các trigger DML**

SQL Server 2012 cho phép người dùng chỉ ra trigger AFTER nào sẽ được thực thi đầu tiên và cái nào sẽ được thực thi cuối cùng. Tất cả các trigger AFTER đã gọi giữa những trigger đầu tiên và cuối cùng không có thứ tự thực thi nhất định.

Tất cả các hành động gây ra có trigger đầu tiên và cuối cùng được định nghĩa cho chúng. Tuy nhiên, không có hai hành động kích hoạt trên một bảng có thể có cùng các trigger đầu tiên và cuối cùng.

Người dùng có thể sử dụng thủ tục lưu trữ `sp_settriggerorder` để định nghĩa thứ tự của các trigger **DML AFTER**.

Hình 12.7 hiển thị thứ tự thực thi của các trigger DML.



Hình 12.7: Execution Order of DML Triggers

Sau đây là cú pháp để chỉ ra thứ tự thực thi nhiều trigger AFTER DML.

Cú pháp:

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

trong đó,

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

value: chỉ ra thứ tự thực hiện của trigger như FIRST, LAST, hoặc NONE. Nếu FIRST được chỉ ra, sau đó trigger được khai hỏa đầu tiên. Nếu LAST được chỉ ra, trigger được khai hỏa cuối cùng. Nếu NONE được chỉ ra, thứ tự khai hỏa của trigger là không xác định.

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

Code Snippet 17 đầu tiên thực thi trigger **Employee_Deletion** đã định nghĩa trên bảng này khi hoạt động DELETE được thực hiện trên cột **Withdrawal** của bảng này.

Code Snippet 17:

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

12.8.1 Xem định nghĩa về các trigger DML

Định nghĩa trigger bao gồm tên trigger, bảng trên đó trigger được tạo ra, các hành động kích hoạt, và các câu lệnh SQL được thực thi. SQL Server 2012 cung cấp thủ tục lưu trữ sp_helptext để lấy các định nghĩa trigger.

Tên trigger DML phải được chỉ ra như là tham số khi thực thi sp_helptext. Hình 12.8 hiển thị định nghĩa các trigger DML.

Ghi chú - Không thể xem định nghĩa trigger nếu định nghĩa được mã hóa

Sau đây là cú pháp cho việc xem trigger DML.

Cú pháp:

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

trong đó,

DML_trigger_name: chỉ ra tên của trigger DML định nghĩa của nó sẽ được hiển thị.

Code Snippet 18 hiển thị những định nghĩa của trigger **Employee_Deletion**, đã tạo ra trên bảng này.

Code Snippet 18:

```
sp_helptext 'Employee_Deletion'
```

12.8.2 Sửa đổi định nghĩa về các trigger DML

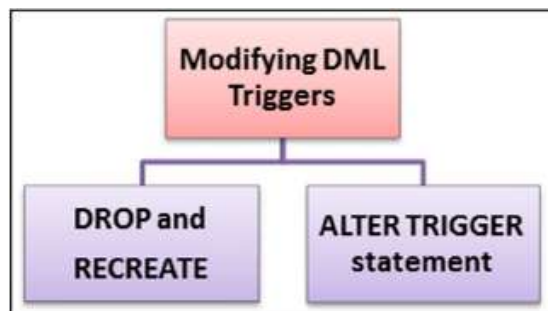
Các tham số trigger được định nghĩa vào thời điểm tạo ra trigger. Những tham số này bao gồm loại hành động kích hoạt để gọi trigger và các câu lệnh SQL được thực thi.

Nếu người dùng muốn sửa đổi bất kỳ tham số nào cho trigger DML, người dùng có thể làm như vậy trong bất kỳ một trong hai cách:

- Thả bỏ và tái tạo trigger với những tham số mới.
- Thay đổi những tham số này bằng cách sử dụng câu lệnh **ALTER TRIGGER**.

Nếu đối tượng tham khảo trigger DML được đổi tên, trigger phải được sửa đổi để phản ánh sự thay đổi trong tên đối tượng.

Ghi chú - Trigger DML có thể được mã hóa để ẩn định nghĩa của nó



Hình 12.8: Modifying DML Triggers

Sau đây là cú pháp cho việc sửa đổi trigger DML.

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 các định nghĩa trigger DML không được hiển thị.

FOR | AFTER : chỉ ra rằng trigger DML thực thi sau khi các hoạt động sửa đổi được hoàn tất.

INSTEAD OF : chỉ ra rằng trigger DML thực thi thay cho các hoạt động sửa đổi.

Code Snippet 19 thay đổi trigger **CheckEmployeeID** đã tạo ra trên bảng **EmployeeDetails** bằng cách sử dụng tùy chọn **WITH ENCRYPTION**.

Code Snippet 19:

```
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
```

Bây giờ, nếu người dùng cố gắng xem định nghĩa của trigger **CheckEmployeeID** bằng cách sử dụng thủ tục lưu trữ `sp_helptext`, thông báo lỗi sau đây được hiển thị:

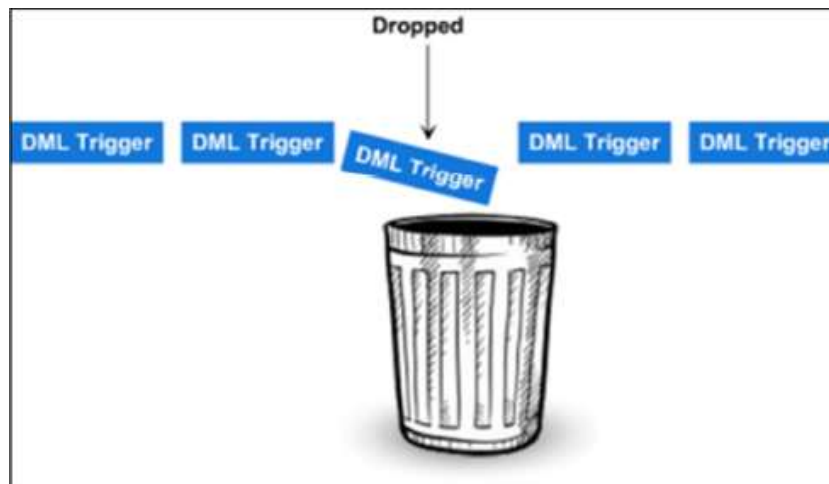
Văn bản cho đối tượng `CheckEmployeeID` được mã hóa

12.8.3 Thả bỏ các trigger DML

SQL Server 2012 cung cấp tùy chọn thả bỏ một trigger DML đã tạo ra trên bảng nếu trigger không còn cần thiết. Có thể thả bỏ trigger bằng cách sử dụng câu lệnh **DROP TRIGGER**. Cũng có thể thả bỏ nhiều trigger bằng cách sử dụng một câu lệnh trigger thả bỏ đơn lẻ.

Khi một bảng bị thả bỏ, tất cả trigger đã định nghĩa trên bảng đó cũng được thả bỏ.

Hình 12.9 minh họa khái niệm về các trigger DML đã thả bỏ.



Hình 12.9: Dropping DML Triggers

Ghi chú - Khi trigger DML bị xóa khỏi bảng này, thông tin về trigger cũng bị loại bỏ khỏi các khung nhìn danh mục.

Sau đây là cú pháp cho việc thả bỏ các trigger DML.

Cú pháp:

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

trong đó,

DML_trigger_name: chỉ ra tên của trigger DML sẽ được thả bỏ

[,...n] : chỉ ra rằng nhiều trigger DML có thể được thả bỏ.

Code Snippet 20 thả bỏ trigger **CheckEmployeeID** đã tạo ra trên bảng **EmployeeDetails**.

Code Snippet 20:

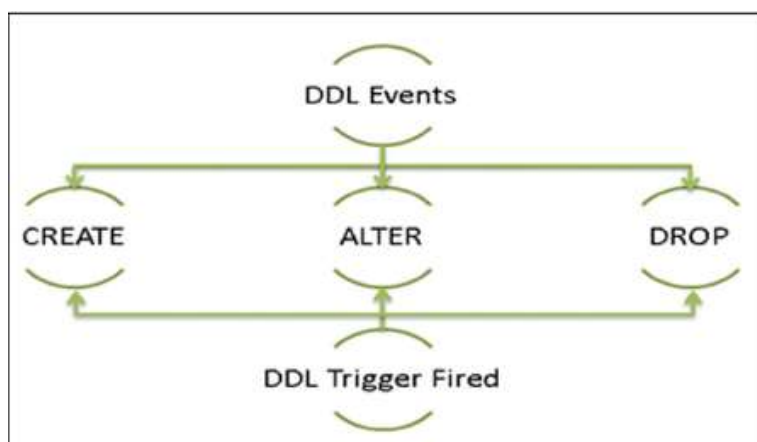
```
DROP TRIGGER CheckEmployeeID
```

12.9 DDL Triggers

Trigger DDL thực thi thủ tục lưu trữ khi các sự kiện DDL như là các câu lệnh CREATE, ALTER, và DROP xảy ra trong cơ sở dữ liệu hoặc máy chủ này. Các trigger DDL có thể chỉ hoạt động khi hoàn thành các sự kiện DDL.

Có thể sử dụng các trigger DDL để ngăn chặn những thay đổi trong lược đồ cơ sở dữ liệu. Lược đồ là một tập hợp các đối tượng như bảng, dạng xem, và văn bản trong một cơ sở dữ liệu.

Các trigger DDL có thể gọi một sự kiện hoặc hiển thị một thông báo dựa trên những sửa đổi đã thử trên lược đồ này. Các trigger DDL được định nghĩa ở mức cơ sở dữ liệu hoặc ở mức máy chủ. Hình 12.10 hiển thị các loại trigger DDL.



Hình 12.10: DDL Triggers

Sau đây là cú pháp cho việc thả bỏ các trigger DDL.

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 trigger DDL thực thi khi các sự kiện DDL xảy ra trong máy chủ hiện tại .

DATABASE: chỉ ra rằng trigger DDL thực thi khi các sự kiện DDL xảy ra trong cơ sở dữ liệu hiện tại.

event_type: chỉ ra tên của sự kiện DDL để gọi trigger DDL.

Code Snippet 21 tạo ra một trigger DDL để thả bỏ và thay đổi một bảng.

Code Snippet 21:

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

Trong đoạn mã này, trigger DDL được tạo ra cho câu lệnh **DROP TABLE** và **ALTER TABLE**.

12.9.1 Phạm vi của các trigger DDL

Các trigger DDL được gọi bằng các câu lệnh SQL đã thực thi trong cơ sở dữ liệu hiện tại hoặc trên máy chủ hiện tại. Ví dụ, trigger DDL đã tạo ra cho câu lệnh CREATE TABLE thực thi trên sự kiện CREATE TABLE trong cơ sở dữ liệu. Trigger DDL đã tạo ra cho câu lệnh CREATE LOGIN thực thi trên sự kiện CREATE LOGIN trong máy chủ.

Phạm vi của trigger DDL phụ thuộc vào việc trigger thực thi cho các sự kiện cơ sở dữ liệu hay các sự kiện máy chủ. Theo đó, các trigger DDL được phân thành hai loại như sau:

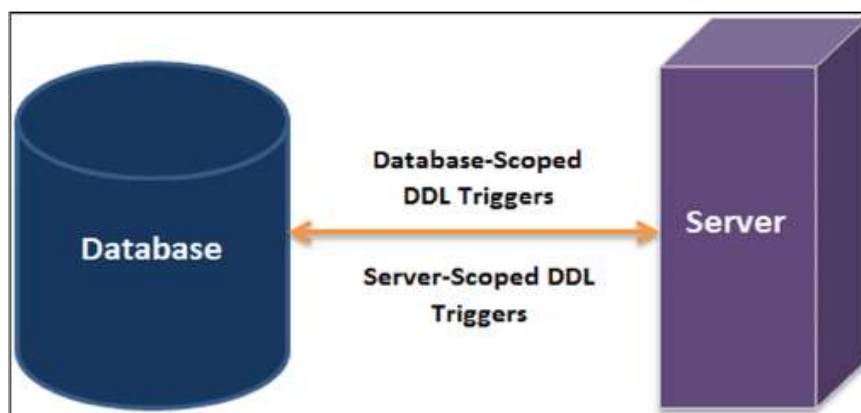
➤ **Các trigger DDL có phạm vi cơ sở dữ liệu**

Các trigger DDL có phạm vi cơ sở dữ liệu được gọi bằng những sự kiện thay đổi lược đồ cơ sở dữ liệu. Những trigger này được lưu trữ trong cơ sở dữ liệu và thực thi trên các sự kiện DDL, ngoại trừ những cái liên quan đến các bảng tạm thời.

➤ **Các trigger DDL có phạm vi máy chủ**

Các trigger DDL có phạm vi máy chủ được gọi bằng các sự kiện DDL ở mức máy chủ. Những trigger này được lưu trữ trong cơ sở dữ liệu master.

Hình 12.11 hiển thị phạm vi của các trigger DDL.



Hình 12.11: Scope of DDL Triggers

12.10 Các trigger lồng nhau

Cả trigger DDL và DML được lồng nhau khi trigger thực hiện một hành động khởi xướng trigger khác. Có thể lồng các trigger DDL và DML lên đến 32 mức. Giả sử nếu một trigger sửa đổi một bảng trên đó có một trigger khác, trigger thứ hai được khởi xướng, sau đó gọi một trigger thứ ba, và vân vân.

Nếu các trigger lồng nhau được cho phép, khi đó những trigger này theo thứ tự bắt đầu một vòng lặp vô hạn. Điều này sẽ vượt quá mức lồng nhau và trigger sẽ chấm dứt.

Các trigger lồng nhau có thể được sử dụng để thực hiện các hàm như lưu trữ bản sao lưu của những hàng bị ảnh hưởng bằng những hành động trước đó.

Trigger Transact-SQL thực thi mã có quản lý thông qua tham khảo một thường trình CLR, tổng hợp, hoặc loại, tham chiếu số đếm làm một mức đối với giới hạn xếp lồng 32 cấp. Các phương pháp được gọi từ bên trong mã có quản lý không được đếm đối với giới hạn này.

Người dùng có thể vô hiệu hóa các trigger lồng nhau, bằng cách đặt tùy chọn trigger lồng nhau của `sp_configure` thành 0 hoặc off. Cấu hình mặc định được cho phép cho các trigger lồng nhau. Nếu tùy chọn trigger lồng nhau là off, khi đó trigger đệ quy bị vô hiệu hóa, không phân biệt thiết lập trigger đệ quy được đặt bằng cách sử dụng **ALTER DATABASE**.

Code Snippet 22 tạo ra một trigger **AFTER DELETE** có tên là **Employee_Deletion** trên bảng **Employee_Personal_Details**.

CodeSnippet22:

```
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 tin nhắn được hiển thị. Ngoài ra, bản ghi của nhân viên được xóa khỏi bảng **Employee_Salary_Details**.

Code Snippet 23 tạo ra một trigger **AFTER DELETE Deletion_Confirmation** trên bảng **Employee_Salary_Details**.

CodeSnippet23:

```
CREATE TRIGGER Deletion_Confirmation
ON Employee_Salary_Details
AFTER DELETE
AS
BEGIN
PRINT 'Employee details successfully deleted from Employee_Salary_Details table'
END
DELETE FROM Employee_Personal_Details WHERE EmpID=1
```

Khi bản ghi được xóa khỏi bảng **Employee_Salary_Details**, trigger **Deletion_Confirmation** được kích hoạt. Trigger này in thông báo xác nhận về bản ghi đang bị xóa.

Như vậy, các trigger **Employee_Deletion** và **Deletion_Confirmation** được thấy sẽ được lồng nhau.

12.11 UPDATE()

Hàm UPDATE () trả về một giá trị Boolean chỉ ra liệu hành động UPDATE hoặc INSERT đã được thực hiện trên một khung nhìn hoặc cột cụ thể của một bảng.

Hàm UPDATE () có thể được sử dụng bất cứ nơi nào bên trong phần thân của một trigger UPDATE hoặc INSERT Transact-SQL để kiểm tra xem trigger nên thực thi một số hành động hay không.

Sau đây là cú pháp cho UPDATE ().

Cú pháp:

```
UPDATE ( column )
```

trong đó

column: là tên của cột để kiểm tra về hành động **INSERT** hoặc **UPDATE**.

Code Snippet 24 tạo ra trigger **Accounting** trên bảng **Account_Transactions** để cập nhật các cột **TransactionID** hoặc **EmployeeID**.

Code Snippet 24:

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

12.12 Xử lý nhiều hàng trong một phiên

Khi người dùng viết mã cho một trigger DML, khi đó câu lệnh này làm cho trigger kích hỏa sẽ là câu lệnh đơn lẻ. Câu lệnh đơn lẻ này sẽ ảnh hưởng đến nhiều hàng dữ liệu, thay vì một hàng đơn lẻ. Đây là một hành vi phổ biến cho các trigger DELETE và UPDATE bởi những câu lệnh này thường ảnh hưởng đến nhiều hàng. Hành vi cho các trigger INSERT ít phổ biến hơn bởi câu lệnh INSERT cơ bản chỉ thêm một hàng.

Khi chức năng của một trigger DML tự động kéo theo việc tính toán lại các giá trị tóm tắt của một bảng và lưu trữ kết quả vào bảng khác, sau đó cân nhắc nhiều hàng là rất quan trọng.

Code Snippet 25 lưu trữ tổng đang chạy cho một lần chèn đơn hàng.

Code Snippet 25:

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

Trong đoạn mã này, tổng phụ được tính toán và lưu trữ cho một hoạt động chèn đơn hàng. Code Snippet 26 lưu trữ tổng đang chạy cho một lần chèn đa hàng hoặc đơn hàng.

Code Snippet 26:

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

Trong đoạn mã này, tổng phụ được tính toán và lưu trữ cho hoạt động chèn đa hàng hoặc đơn hàng.

12.13 Ảnh hưởng hiệu suất của các trigger

Trong thực tế, trigger không mang chi phí, mà là chúng khá nhạy. Tuy nhiên, nhiều vấn đề hiệu suất có thể xảy ra do logic hiện tại bên trong trigger. Giả sử một trigger tạo ra một con trỏ và các vòng qua nhiều hàng, sau đó sẽ có một sự suy giảm trong quá trình này.

Tương tự như vậy, hãy xem xét rằng trigger thực thi các câu lệnh SQL khác nhau đối với các bảng khác tách biệt khỏi các bảng Inserted và Deleted. Điều này một lần nữa sẽ dẫn đến sự suy giảm tốc độ của các câu lệnh SQL nằm trong trigger này.

Một nguyên tắc tốt sẽ giữ cho logic đơn giản trong những trigger và tránh sử dụng các con trỏ trong khi thực thi các câu lệnh đối với bảng khác và các tác vụ khác làm cho hiệu suất chậm lại.

12.14 Kiểm tra tiến độ của bạn

1. Câu nào sau đây về các trigger trong SQL Server 2012 là đúng?

a.	Trigger lấy thông tin từ các bảng của cùng CSDL, cũng như CSDL khác.	c.	Các trigger DML thực thi trên các câu lệnh INSERT, UPDATE, và DELETE.
b.	Các trigger DDL chỉ hoạt động sau khi một bảng hoặc khung nhìn được sửa đổi.	d.	Các trigger DDL thực thi trong khi sửa đổi dữ liệu hoặc sau khi dữ liệu được sửa đổi.
(A)	a, b, c	(C)	a, b, d
(B)	b, c, d	(D)	a, c, d

2. So khớp các loại trigger DML trong SQL Server 2012 đối với các mô tả tương ứng của chúng.

	Description		DML Trigger
a.	Thực thi khi người dùng thay thế một bản ghi hiện có bằng một giá trị mới.	1.	INSERT
b.	Thực thi khi hoàn thành các hoạt động sửa đổi.	2.	UPDATE
c.	Thực thi thay cho các hoạt động sửa đổi.	3.	DELETE
d.	Thực hiện khi người dùng thêm một bản ghi lên một bảng.	4.	AFTER
e.	Thực hiện khi người dùng loại bỏ một bản ghi khỏi một bảng.	5.	INSTEAD OF
(A)	a-1,b-4, c-2,d-3, e-5	(C)	a-2, b-4,c-3, d-5,e-1
(B)	a-2,b-4, c-5,d-1, e-3	(D)	a-1, b-2,c-3, d-4, e-5

3. Câu nào sau đây về các trigger trong SQL Server 2012 là đúng?

a.	Các trigger DML có thể thực hiện nhiều hành động cho mỗi câu lệnh sửa đổi.	c.	Các trigger UPDATE không sử dụng bảng Deleted để cập nhật các bản ghi trong một bảng.
b.	Các bảng Inserted và Deleted được tạo ra bởi SQL Server 2012 khi một bảng mới được tạo ra trong cơ sở dữ liệu.	d.	Các trigger đã xóa không sử dụng bảng Inserted để xóa các bản ghi trong một bảng.
(A)	a, b	(C)	a, d
(B)	c, d	(D)	b, d

4. Câu nào sau đây về làm việc với các trigger DML của SQL Server 2012 là đúng?

a.	Mỗi hành động kích hoạt không có thể có nhiều trigger AFTER.	c.	Định nghĩa trigger DML có thể được sửa đổi bằng cách thả bỏ và tái tạo trigger đó.
b.	Hai hành động kích hoạt trên một bảng Định nghĩa trigger DML có thể có cùng các trigger đầu tiên và cuối cùng.	d.	Định nghĩa trigger DML có thể được xem bằng cách sử dụng thủ tục lưu trữ sp_helptext.
(A)	a, c	(C)	b, d
(B)	b, c	(D)	c, d

5. Các trigger_____có thể được sử dụng để thực hiện các hàm như lưu trữ bản sao lưu của những hàng bị ảnh hưởng bằng những hành động trước đó.

(A)	Lồng	(C)	DDL
(B)	DML	(D)	INSTEAD OF

12.14.1 Đáp án

1.	A
2.	B
3.	C
4.	D
5.	A

Tóm tắt

- Trigger là một thủ tục lưu trữ được thực hiện khi có một nỗ lực để sửa đổi dữ liệu trong bảng được bảo vệ bằng trigger.
- Các trigger đăng nhập thực thi các thủ tục lưu trữ khi một phiên được thiết lập với một sự kiện LOGON.
- Các trigger DML được thực thi khi sự kiện DML xảy ra trong các bảng hoặc khung nhìn.
- Trigger INSERT được thực thi khi một bản ghi mới được chèn vào bảng.
- Trigger UPDATE sao chép bản ghi gốc vào bảng Deleted và bản ghi mới vào bảng Inserted khi bản ghi được cập nhật.
- Có thể tạo ra trigger **DELETE** để hạn chế người dùng không xoá một bản ghi cụ thể trong bảng.
- Trigger **AFTER** được thực thi khi hoàn thành các hoạt động **INSERT**, **UPDATE**, hoặc **DELETE**.



1. **Galaxy Airlines** là một dịch vụ hãng hàng không vừa được tung ra, hoạt động các chuyến bay đến và đi từ các thành phố khác nhau trên toàn châu Âu. Công ty duy trì dữ liệu liên quan đến các giao tác hàng ngày liên quan đến các dịch vụ bay trong cơ sở dữ liệu SQL Server 2012. Để cho phép hoạt động hiệu quả và nhanh hơn, **Galaxy Airlines** đã quyết định kết hợp sử dụng các trigger trong các ứng dụng cơ sở dữ liệu của họ. Danh sách chi tiết các hoạt động phải được thực hiện được liệt kê như sau:

- a. Tạo ra những bảng sau đây trong cơ sở dữ liệu **GalaxyAirlines**. Bảng 12.2 liệt kê bảng **Flight**.

Tên trường	Loại DL	Key Field	Mô tả
Aircraft code	varchar(10)	Primary key	Lưu trữ mã máy bay
Type	varchar(6)		Mô tả loại máy bay
Source	varchar(20)		Lưu trữ tên của thành phố là nơi máy bay sẽ khởi hành
Destination	varchar(20)		Lưu trữ tên của thành phố là nơi máy bay sẽ đến
Dep_time	varchar(10)		Lưu trữ thời gian khởi hành
Journey_hrs	int		Lưu trữ số giờ hành trình

Table 12.2: Flight Table

Table 12.3 liệt kê bảng **Flight_Details**.

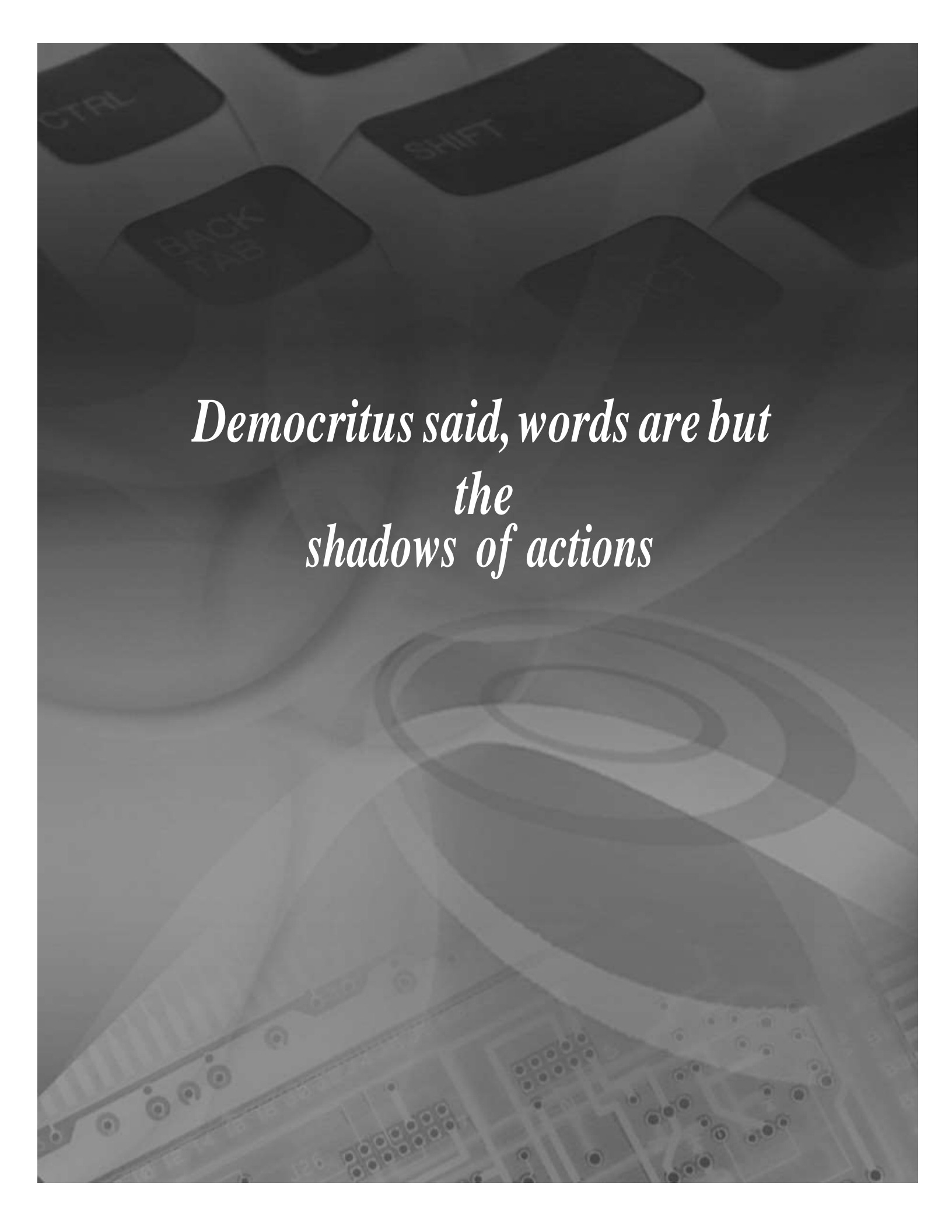
Tên trường	Loại DL	Key Field	Mô tả
Class_Code	varchar(10)	Primary key	Lưu trữ hạng, gồm ưu tiên, thương gia hoặc phổ thông
Aircraft code	varchar(10)	Foreign key	Lưu trữ mã máy bay
Fare	money		Lưu trữ số tiền vé
Seats	int		Lưu trữ tổng số chỗ ngồi trên chuyến bay

Table 12.3: Flight_Details Table

- b. Viết các câu lệnh để tạo ra trigger **trgCheckSeats** sẽ kích hoạt bất cứ khi nào một hàng mới được đưa vào bảng **Flight_Details**. Giới hạn tối đa số ghế một chuyến bay có thể chứa là **150**. Trigger nên kiểm tra giá trị của số ghế đang được chèn vào. Nếu là hơn 150, hoạt động INSERT không được cho phép để thành công.



- c. Chèn ít nhất là năm bản ghi vào mỗi bảng.
- d. Viết các câu lệnh để tạo ra trigger **UpdateValid** sẽ kích hoạt mỗi lần hàng được cập nhật vào bảng **Flight_Details**. Trigger nên xác định xem cột **Seats** có hiện diện trong danh sách các cột đang được cập nhật hay không. Nếu có, hoạt động UPDATE sẽ không thành công vì cột **Seats** được định nghĩa là một hằng số và không thể thay đổi được.
- e. Write Viết các câu lệnh để tạo ra một trigger DDL **ProhibitDelete** sẽ kích hoạt bất cứ khi nào người dùng đang cố gắng xóa một bảng khỏi cơ sở dữ liệu **Galaxy Airlines**. Trigger không phải cho phép người dùng thực hiện ca lần xóa và phải hiển thị thông báo "You are not allowed to delete tables in this database" (Bạn không được phép xóa các bảng trong cơ sở dữ liệu này).



*Democritus said, words are but
the
shadows of actions*