

القوادر Triggers

عند القيام بإحدى عمليات الإضافة أو الحذف أو التعديل على سجل (أو سجلات) من الجدول، يقوم الجدول في SQL Server بإنتاج إخطار أو إشعار. ونقول حينها أن الجدول قام بإطلاق حدث، يمكنك استخدام هذا الحدث لاتخاذ بعض الإجراءات.

القادر (Trigger) هو عملية يتم تنفيذها وراء الكواليس عند تطبيق حدث على الجدول.

تطبيق عملي: تمهيد لإنشاء القوادر

1- قم بتشغيل SQL Server Management Studio

2- من شريط الأدوات قياسي، انقر على New Query

3- سنقوم بإنشاء قاعدة بيانات جديدة تحتوي على جدول واحد، اكتب ما يلي:

```
-- =====
-- Database:    CeilInn4
-- =====

IF EXISTS(SELECT name FROM sys.databases
          WHERE name = N'CeilInn4b')
DROP DATABASE CeilInn4;
GO
CREATE DATABASE CeilInn4;
GO

USE CeilInn4;
GO

IF OBJECT_ID('Rooms', 'U') IS NOT NULL
    DROP TABLE Rooms
GO

-- =====
-- Database:    CeilInn4
-- Table:      Rooms
-- Description: This table is used to hold information
--              about his room rented for the hotel
-- =====
CREATE TABLE Rooms
(
    RoomNumber nvarchar(10),
    LocationCode nchar(10) default N'Silver Spring',
    RoomType nvarchar(20) default N'Bedroom',
    BedType nvarchar(40) default N'Queen',
    Rate money default 85.95,
    Available bit default 1
)
```

```
);
GO

-- =====
-- Database: CeilInn4
-- Table: DatabaseOperations
-- Description: This table is used to hold information
-- about operations performed on any table
-- of the database. It specifies:
-- a. The type of object on which the action
-- was performed. The types of object can
-- be a table
-- b. The name of the table
-- c. The name of the employee who
-- performed the action.
-- d. The action that was performed. This
-- can be an insert, an update, or a
-- delete operation
-- e. The date/time the action was performed
-- =====
CREATE TABLE DatabaseOperations (
    ObjectType nchar(20),
    ObjectName nvarchar(40),
    EmployeeName nvarchar(50),
    ActionPerformed nvarchar(50),
    TimePerformed datetime2
);
GO
```

4- اضغط F5 لتنفيذ الاستعلام.

إنشاء قاذح (Trigger):

يمكنك إنشاء القوالب باستخدام SQL، سنتحدث عن ذلك لاحقاً. لتتعرف على الهيكل الأساسي لاستعلام إنشاء قاذح، افتح نافذة الاستعلام، ثم من Template Explorer قم بتوسيع البند Triggers، ثم اسحب البند Create T-SQL Trigger وأقلته على إطار الاستعلام لتحصل على ما يلي:

```
-- =====
-- Template generated from Template Explorer using:
-- Create Trigger (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- See additional Create Trigger templates for more
-- examples of different Trigger statements.
--
-- This block of comments will not be included in
-- the definition of the function.
-- =====
SET ANSI_NULLS ON
GO
```

```

SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
CREATE TRIGGER <Schema_Name,
               sysname,
               Schema_Name>.<Trigger_Name,
                           sysname,
                           Trigger_Name>
ON <Schema_Name, sysname, Schema_Name>.<Table_Name, sysname,
Table_Name>
AFTER <Data_Modification_Statements, , INSERT,DELETE,UPDATE>
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here

END
GO

```

تنفيذ القوادح:

خلافا للإجراءات المخزنة؛ لست بحاجة لتنفيذ القادح، لأن نظام التشغيل بنفسه (من خلال الحدث) ومشغل قاعدة البيانات يتوليان القيام بذلك. يتم تنفيذ القادح وراء الكواليس فور إرسال الجدول الحدث المناسب لإطلاق القادح. في الواقع، يتوقف إطلاق الحدث على وقوع تغيير (إضافة، حذف، تعديل) في الكائن من عدم وقوعه.

تسيير القوادح:

يعتبر القادح غرض (كائن) من قاعدة بيانات، لذلك فهو يحمل اسما ويمكن التعديل عليه، كما يمكن حذفه.

التعديل على القادح:

إذا كان سلوك القادح غير مناسب، يمكنك تغييره، وصيغة التعديل على القوادح كما يلي:

```

ALTER TRIGGER schema_name.trigger_name
ON schema_name.table_name
AFTER , UPDATE>
AS
    statement

```

للحصول على هيكل استعلام لإنشاء قاذح، قم بفتح نافذة الاستعلام، ثم من Templates Explorer، قم بتوسيع البند Triggers، واسحب الخيار Alter ثم اسقطه في إطار الاستعلام:

```
=====
-- Alter T-SQL Trigger Template
=====
USE <database_name, sysname, AdventureWorks>
GO

ALTER TRIGGER <schema_name, sysname, Sales>.<trigger_name, sysname,
uStore>
ON <schema_name, sysname, Sales>.<table_name, sysname, Store>
AFTER <data_modification_statements, , UPDATE>
AS <T-SQL_statement,
,
UPDATE Sales.Store
SET ModifiedDate = GETDATE()
FROM inserted WHERE inserted.CustomerID = Sales.Store.CustomerID>
GO
```

حذف القوالب:

يمكنك حذف قاذح من الجدول باستخدام الصيغة التالية:

```
DROP TRIGGER TriggerName
```

حيث يمثل *TriggerName* اسم القاذح الذي تود حذفه.

قوالب DML:

يدعم SQL Server ثلاثة أنواع من القوالب: DDL، DML، و Logon.

قاذح DML هو إجراء يقوم بأداء أحد العمليات على بيانات الجدول. وهذا يعني أن هذا النوع من القوالب يجب أن يتم إنشاؤه استناداً إلى جدول من قاعدة البيانات.

قوالب DML للإضافة¹

قاذح الإضافة هو قاذح DML يتم تشغيله عند إضافة سجل إلى الجدول. الصيغة العامة

لإنشاء قاذح DML كما يلي:

```
CREATE TRIGGER TriggerName
ON TableName
AFTER/FOR INSERT
AS
TriggerCode
```

¹ DML هي اختصار Data Manipulation Language.

بعد عبارة **CREATE TRIGGER** يأتي اسم القادح بإتباع قواعد تسمية الكائنات في SQL Server، ثم كلمة ON يليها اسم الجدول الذي سيطبق عليه القادح، حيث يجب التأكد من وجود الجدول في قاعدة البيانات.

نتحدث في هذه الحالة عن إنشاء قادح بعد إضافة سجل، لأجل ذلك؛ يمكنك استخدام أحد العبارتين AFTER INSERT أو FOR INSERT.

لكتابة شفرة SQL التي تشكل موضوع عمل القادح، تضاف كلمة AS متبوعة بالتعليمات البرمجية المناسبة.

بعد إنشاء قادح الإضافة، وحالما يحين الوقت المناسب لإطلاق القادح (أي عند وقوع الحدث) سيتم تنفيذ القادح. وحينها يقوم محرك قاعدة البيانات تلقائياً وداخليا بإنشاء جدول مؤقت اسمه **inserted**. يحتوي هذا الجدول على نسخة من السجل (أو السجلات) التي تم إنشاؤها. يمكنك الوصول إلى تلك السجلات إذا لزم الأمر.

تطبيق عملي: إنشاء قادح DML

1- في نافذة استعلام جديدة اكتب الاستعلام الآتي لإنشاء القادح:

```
USE CeilInn4;
GO

-- =====
-- Database: CeilInn4
-- DML Trigger:RecordInsertion
-- Description:This trigger updates the DatabaseOperations
-- by letting it know that a new record was
-- added to the Rooms table. The trigger
-- also specifies the name of the employee
-- who performed the operation and the time
-- this occurred
-- =====
CREATE TRIGGER RecordInsertion
ON Rooms
AFTER INSERT
AS
BEGIN
    INSERT INTO DatabaseOperations
    VALUES (N'Table', N'Rooms', SUSER_SNAME(),
            N'Created a new record', GETDATE())
END
GO
```

2- اضغط F5 للتنفيذ.

5- في مستكشف الكائنات، انقر بالزر الأيمن على Databases ثم اختر Refresh

6- قم بتوسيع البند Databases ثم CeilInn4

7- من قاعدة البيانات CeilInn4 ثم افتح Tables

8- من أجل إطلاق حدث "إدخال بيانات" سنقوم بإضافة بعض السجلات، بالزر الأيمن

انقر على Rooms ثم اختر Edit Top 200 Rows

RoomNumber	LocationCode	RoomType	BedType	Rate	Available
104	SLSP				
105	SLSP		King	95.50	True
106	SLSP		King	95.50	True
107	SLSP				True
108	SLSP		King	95.50	
109	SLSP				True
110	SLSP	Conference		450.00	True

9- أغلق الجدول

10- من مستكشف الكائنات، انقر بالزر الأيمن على الجدول DatabaseOperations وانقر

على Select Top 1000 rows لمشاهدة السجلات.

11- أغلق الجدول DatabaseOperations.

قواعد DML للحذف:

يمكنك إنشاء قاذح يتم تطبيقه عند حذف سجل من الجدول، باستخدام الصيغة التالية:

```
CREATE TRIGGER TriggerName
ON TableName
AFTER/FOR DELETE
AS
TriggerCode
```

في هذه الحالة، تستخدم الصيغة AFTER DELETE أو FOR DELETE. بقية المعاملات تطرقنا إليها في الحالة السابقة.

عند حذف أحد السجلات من الجدول، يقوم محرك قاعدة البيانات بإنشاء جدول مؤقت يسمى **deleted**. يحتوي هذا الجدول على نسخ من السجلات التي تم حذفها. يمكنك، إذا لزم الأمر، الوصول إلى هذا الجدول لمعرفة المزيد عن تلك السجلات.

تطبيق عملي: قادح DML للحذف

1- في نافذة استعلام جديدة، اكتب الآتي لإنشاء قادح حذف:

```
USE CeilInn4;
GO

-- =====
-- Database: CeilInn4
-- DML Trigger:RecordDeletion
-- Description:This trigger adds a new record to the
-- DatabaseOperations when an existing record
-- of the Rooms table has been deleted.
-- =====
CREATE TRIGGER RecordDeletion
ON Rooms
AFTER DELETE
AS
BEGIN
    INSERT INTO DatabaseOperations
    VALUES (N'Table', N'Rooms', SUSER_SNAME(),
            N'Deleted a room', GETDATE())
END
GO
```

2- اضغط F5 لتنفيذ الاستعلام

3- لحذف سجل من الجدول Rooms، انقر عليه من متصفح الكائنات، ثم اختر Edit
Top 200 Rows

4- بالزر الأيمن انقر على طرف السجل 106 واختر Delete ثم Yes للتأكيد.

5- أغلق الجدول

6- من متصفح الكائنات، انقر بالزر الأيمن على الجدول DatabaseOperations واختر
Select Top 1000 rows لمعاينة سجلاته

7- إغلاق DatabaseOperations الجدول.

خصائص قوادح DML:

يمكنك الذهاب أبعد من ذلك في إنشاء قوادح، يمكنك إنشاء العديد من القوادح (طبعا، بأسماء مختلفة) التي تقوم بنفس الإجراءات على الجدول. فمثلا يمكنك إنشاء عدة قوادح إضافة (أو حذف أو تحديث) التي تعمل على نفس الجدول، و تستهدف نفس الإجراءات.

تشمل القواعد العديد من الخصائص الإضافية.

القواعد وقيود إدخال البيانات:

رأينا في درس سابق أن لمساعدة المستخدم لإدخال البيانات، يمكنك منع إدخال بيانات إلى أحد الحقول أو إجبار المستخدم على إدخال قيم إليها. وذلك بتفعيل خاصية **NOT NULL** للحقل المحدد. إذا لم يتم المستخدم أثناء إدخال البيانات بتوفير قيمة لهذا الحقل، فإن السجل لا يمكن إنشاؤه. عند إنشاء قاع **DML** لإضافة بيانات، لذلك يجب مراعاة إمكانية عدم إدخال أي قيمة، وإلا فلن يتم تشغيل القاع.

رأينا أيضا ضمن هذه السلسلة إمكانية إنشاء قيد لإدخال البيانات (**Check Constraint**) على الجدول، للتأكد من مطابقة السجل المضاف (أو المعدل) لشروط معينة، وإذا لم تتحقق الشروط المطبقة على الحقل، فلن يتم إنشاء (أو تعديل) السجل. إذا كان القاع يقوم بإضافة أو تعديل بيانات دون مراعاة قيود الإدخال، فسيفشل في ذلك.

من حدود قيود إدخال البيانات هو أنها لا تطبق إلا على الجدول الذي أنشأت عليه، بينما يمكن لقاع **DML** أن يقوم بأداء قيد لإدخال البيانات على أكثر من جدول واحد، وهذا ما يوفر ميزة زائدة على قيود الإدخال الاعتيادية.

رأينا عند دراسة البيانات وعلاقات التكامل المرجعي (**Referential Integrity**)، أنه عند تحرير سجل من الجدول الأصل، فسيحصل التغيير أيضا على الجدول الابن، وهذا يدل على أن تطبيق التكامل المرجعي يحصل على أكثر من جدول. يقوم قاع **DML** عند إطلاقه، بفحص قواعد التكامل المرجعي، ويتوقف حال مخالفته أحد هذه القواعد.

القواعد البديلة (**Instead of Trigger**):

ليكن الجدول التالي في قاعدة بيانات:

```
CREATE DATABASE SmallBusiness;
GO

USE SmallBusiness;
GO

CREATE TABLE Customers
```



```
(
    CustomerID int identity(1, 1) primary key not null,
    AccountNumber nchar(10),
    FullName nvarchar(50)
);
GO
```

```
CREATE TABLE DatabaseOperations (
    EmployeeName nvarchar(50),
    ActionPerformed nvarchar(50),
    TimePerformed datetime2
);
GO
```

من خلال ما رأيناه سابقاً، عندما يقوم المستخدم بإدخال بيانات إلى جدول (أو كائن View)، يقوم هذا الأخير بإطلاق حدث فور إنشاء السجل. رأينا أن قوادم DML تسمح بإنشاء إخطار لذلك الحدث، فمثلاً يمكنك استغلال ذلك لملء جدول (Log) لتعقب التغييرات.

افتراضياً، عند القيام بإضافة أو حذف أو تعديل سجل، فإن التغيير يحصل فوراً ويطبق على الجدول، وبدلاً من قبول التغييرات يمكنك إلغاؤها. يمكنك أيضاً استخدام قادم DML من إعداد مذكورة لحفظ التغييرات، ويتم ذلك بإنشاء نوع آخر من قوادم DML وهو القوادم البديلة (Instead of Trigger).

عند تنفيذ قادم AFTER/FOR على أي جدول بعد حدث طرأ عليه، قد ترغب بفعل شيء ما قبل هذا الحدث. مثلاً، قد تحتاج إلى منع المستخدم من إضافة (أو حذف أو تغيير) بيانات في الجدول، وطبعاً، ينبغي الاهتمام بـ طبيعة عمل القادح، طريقة وحيدة تسمح لك القيام بذلك هي استخدام القوادم البديلة.

إنشاء القوادم البديلة:

عند تطبيق قادم AFTER، يمكن تطبيق قادم بديل على جدول أو كائن View. الصيغة:

```
CREATE TRIGGER TriggerName
ON TableOrViewName
INSTEAD OF INSERT/UPDATE/DELETE
AS
    TriggerCode
```

يبدأ الاستعلام بعبارة CREATE TRIGGER متبوعة باسم القادح، بعد ذلك اكتب ON يليها اسم جدول أو الكائن View التي سيؤثر عليها القادح.

الجديد هنا عبارة INSTEAD OF، يليها نوع العمليات الحاصلة. إذا كنت تريد:

- النقاط حدث إنشاء سجل، استخدم المعامل **INSERT**

- النقاط حدث تعديل سجل، استخدم المعامل **UPDATE**

- النقاط حدث حذف سجل، استخدم المعامل **DELETE**

لتحرير استعلام SQL مضمون عمل القادح، أضف كلمة AS ثم اكتب التعليمات البرمجية.

عند استخدام التعبير **INSTEAD OF**، يتم تطبيق القادح عندما يفتح الجدول (أو كائن **View**) ولكن قبل حصول تغيرات الحدث. والفرق بينه وبين قادح **AFTER**، أنه يمكنك تنفيذ بعض الإجراءات قبل أن يحصل التغيير في الجدول أو الكائن **View**، هذا يعني أيضا أنه إذا كانت وظيفة القادح إنشاء سجل جديد، فالسجل في هذه الحالة غير موجود، وهو ما يعني أنك لا تستطيع النقاط هذا السجل. وهنا يمكنك أيضا منع إنشاء السجل الذي سيتم إنشاؤه.

مثال، القادح الآتي يقوم بإضافة سجل جديد إلى الجدول عوض عملية الإضافة إليه:

```
USE SmallBusiness;
GO

CREATE TRIGGER CreateCustomer
ON Customers
INSTEAD OF INSERT
AS
BEGIN
    INSERT INTO DatabaseOperations
    VALUES (USER_SNAME(),
            N'Attempt to create new record', GETDATE())
END
GO
```

إذا كنت ترغب في الحصول على نسخة من السجل الذي تمت إضافته إلى الجدول،

يمكنك الولوج إليه من خلال السجل (الجدول) النظامي **inserted** (لقوادح الإضافة **INSERT**

أو التحديث **UPDATE**) أو السجل **deleted** (لقوادح الحذف). مثال:

```
USE SmallBusiness;
GO

DROP TRIGGER CreateCustomer;
GO

CREATE TRIGGER CreateCustomer
ON Customers
INSTEAD OF INSERT
AS
BEGIN
```

```
INSERT INTO Customers
SELECT AccountNumber, FullName FROM inserted
END
GO
```

تطبيق عملي: إنشاء قوالب بديلة

1- في نافذة استعلام، اكتب ما يلي لإنشاء قوالب إضافية وآخر للتحديث:

```
USE CeilInn4;
GO

-- =====
-- Database: CeilInn4
-- View: Logistics
-- Description: This view retrieves the list of rooms
-- of this hotel
-- =====
CREATE VIEW Logistics
AS
    SELECT RoomNumber, LocationCode, RoomType,
           BedType, Rate, Available
    FROM Rooms;
GO

-- =====
-- Database: CeilInn4
-- DML Trigger: AttemptedRecordInsertion
-- Description: This trigger acts on a table to update the
-- DatabaseOperations to let it know that an
-- attempt was made to create a new room
-- =====
CREATE TRIGGER AttemptedRecordInsertion
ON Rooms
INSTEAD OF INSERT
AS
BEGIN
    INSERT INTO DatabaseOperations
    VALUES (N'Table', N'Rooms', SUSER_SNAME(),
            N'Attempted to create a new record', GETDATE())
END
GO

-- =====
-- Database: CeilInn4
-- DML Trigger: AttemptedRecordUpdate
-- Description: This trigger acts on a view to update the
-- DatabaseOperations to let it know that an
-- attempt was made to edit a record of
-- the Rooms table
-- =====
CREATE TRIGGER AttemptedRecordUpdate
ON Logistics
INSTEAD OF UPDATE
AS
BEGIN
    INSERT INTO DatabaseOperations
    VALUES (N'View', N'Logistics', SUSER_SNAME(),
            N'Attempted to change a room's information',
            GETDATE())
```

END
GO

2- اضغط F5 لتنفيذه

3- من متصفح الكائنات وتحت CeilInn4، انقر بالزر الأيمن على Rooms، وانقر Edit Top 200 Rows

4- قم بإضافة السجلات التالية:

RoomNumber	LocationCode	RoomType	BedType	Rate	Available
104	LRL		King	95.50	
112	SLSP		King	95.50	True

5- لاحظ إشعار التنبيه على رؤوس السجلات التي تم تغييرها في الجدول. أغلق الجدول

6- من متصفح الكائنات، تحت Views، انقر بالزر الأيمن على الجدول Logistics واختار Edit Top 200 Rows

7- قم بتعديل السجلات الآتية:

RoomNumber	LocationCode	RoomType	BedType	Rate	Available
108	LRL	Conference	Delete	425.75	

8- لاحظ إشعار التنبيه على رؤوس السجلات لكائن View. اضغط مفتاح Esc

9- أغلق الجدول

10- من متصفح الكائنات، انقر بالزر الأيمن على الجدول DatabaseOperations ثم اختر Select Top 1000 rows لمعاينة سجلاته

11- أغلق الجدول.

خصائص القواعد البديلة:

هناك عدة فروق بين قواعد AFTER/FOR و INSTEAD OF. مثلاً:

- قواعد INSTEAD OF UPDATE و INSTEAD OF DELETE لا يمكنها التأثير على الجداول التي تحوي حقولاً معلمة بخاصية DELETE أو UNIQUE.
- لا يمكنك إنشاء أكثر من قاعد بديل (INSTEAD OF) لكل جدول.

قواعد DDL:

رأينا في الدروس الأولى، أن إنشاء قاعدة بيانات يستخدم تعليمات Data Definition Language (DDL). ورأينا فيما بعد أمثلة أخرى على ما تنطوي عليه أوامر DDL بما فيها إنشاء الجداول، تقوم كل من هذه التعليمات بإطلاق حدث DDL.

قواعد DDL هي القواعد التي يتم تنفيذها عندما تقوم أوامر DDL معينة بإطلاق حدث. ويشمل ذلك إنشاء أو تعديل أو حذف كائن وليس سجلاته، وهذا هو الفرق الأساسي بين هذا النوع من القواعد وبين قواعد DML التي يتم تنفيذها عند وقوع حدث متعلق بالسجلات.

تمنحك قواعد DDL فرصة القيام ببعض الأعمال الإدارية استجابة لبعض الأحداث، على سبيل المثال، يمكنك الحصول على إشعار أو إخطار من أي شخص تلقائياً عبر البريد الإلكتروني، بأن (وأي) كائناً تم إنشاؤه. أو يمكنك استخدام قاعد DDL لإلغاء هذه العملية.

إنشاء قاعد DDL:

يمكنك إنشاء قاعد DDL باستخدام الصيغة الأساسية التالية:

```
CREATE TRIGGER TriggerName
ON DATABASE/ALL SERVER
FOR/AFTER WhatEvent
AS
    TriggerCode
```

البداية بعبارة CREATE TRIGGER يليها اسم القاعد وفق قواعد العامة لتسمية الأغراض، ثم كلمة ON متبوعة:

- بكلمة DATABASE لتطبيق القاعد على قاعدة البيانات الحالية، وسيتم تنفيذ أوامر القاعد على قاعدة البيانات المحددة.
 - بعبارة ALL SERVER إذا كنت تود تطبيق القاعد على الخادم. في هذه الحالة، سيتم تنفيذ القاعد عندما يقع الحدث المقصود في أي جزء من أجزاء الخادم.
- بعد تحديد الكائن (الخادم بأكمله أو قاعدة البيانات الحالية) الذي سيطبق عليه القاعد، أضف كلمة FOR أو AFTER متبوعة بالحدث الذي سينفذ القاعد لأجله. وكما ذكرنا سابقاً،

فإن الأحداث هنا هي أوامر **DDL**. لتحديد أحد هذه الأحداث، استخدام صيغة الحدث مع الفصل بين الكلمات بالرمز **"_"**. مثلاً، إذا كنت تريد أن يتم تنفيذ القادح عند إرسال استعلام إنشاء الجدول (**CREATE TABLE**)، حدد الحدث **CREATE_TABLE**.

بعد ذلك، اكتب كلمة **AS** متبوعة بالتعليمات البرمجية التي سينفذها القادح.

في هذا المثال يقوم القادح عند إنشاء جدول جديد، بإضافة ملاحظة في الجدول:

```
USE SmallBusiness;
GO

CREATE TRIGGER LogNewTableCreation
ON DATABASE
FOR CREATE_TABLE
AS
BEGIN
    INSERT INTO DatabaseOperations
    VALUES (USER_SNAME(),
            N'A new table was created', GETDATE())
END
GO
```

كلما تم إنشاء جدول جديد في قاعدة البيانات الحالية، يتم تنفيذ القادح، لإضافة اسم المستخدم الذي أنشأ الجدول وتاريخ ووقت إنشائه، وعبارة نصية قصيرة، ثم تخزين هذه البيانات في جدول **DatabaseOperations**.

كما أوضحنا في قوادح **DML**، يمكنك إدارة (تعديل أو حذف) قوادح **DDL** بنفس الطريقة. ويتم ذلك باستخدام نفس المواصفات التي رأيناها مع قوادح **DML**.


```
CREATE TABLE [dbo].[student](
    [stid] [int] NOT NULL,
    [stname] [nvarchar](50) NULL,
    CONSTRAINT [PK_student] PRIMARY KEY CLUSTERED
(
    [stid] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

```
CREATE TABLE [dbo].[student_log](
    [stid] [int] NULL,
    [stname] [nvarchar](50) NULL,
    [action] [nvarchar](50) NULL,
    [username] [nvarchar](50) NULL,
    [modificationdate] [datetime] NULL
) ON [PRIMARY]

GO
```

```
GO
/***** Object: Trigger [dbo].[update_history_on_student]    Script Date: 11/4/2022
10:16:29 AM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[update_history_on_student] ON [dbo].[student]
for UPDATE,delete,insert
AS
if exists(SELECT * from inserted) and exists (SELECT * from deleted)
begin
insert into student_log
SELECT
deleted.stid,
    deleted.stname, 'BEFORE UPDATE', SYSTEM_USER

, GETDATE()
FROM deleted
Join inserted
    On inserted.stid  = deleted.stid

insert into student_log
SELECT
    stid,stname,
    'After UPDATE', SYSTEM_USER, GETDATE()
```

```
FROM inserted
end
If exists(select * from deleted) and not exists(Select * from inserted)
begin
insert into student_log
SELECT
    deleted.stid,
    deleted.stname, 'deleted', SYSTEM_USER
, GETDATE()
FROM deleted
end

If not exists(select * from deleted) and exists(Select * from inserted)
begin
insert into student_log
SELECT
    stid,
    stname, 'inserted', SYSTEM_USER
, GETDATE()
FROM inserted
end
```