In this article I will explain with simple examples, how to write Insert, Update and Delete Triggers in SQL Server.

This tutorial is applicable for all versions of SQL Server i.e. 2005, 2008, 2012, 2014, etc.

Database

I have made use of the following table Customers with the schema as follows.

MUDASSAR-PC\SQL2...- dbo.Customers

Column Name	Data Type	Allow Nulls
CustomerId	int	
Name	varchar(100)	
Country	varchar(50)	

I have already inserted few records in the table.

MUDASSAR-PC\SQL2...- dbo.Customers

CustomerId	Name	Country
1	John Hammond	United States
2	Mudassar Khan	India
3	Suzanne Mathe...	France
4	Robert Schidner	Russia
NULL	NULL	NULL

Below is the CustomerLogs table which will be used to log the Trigger actions.

MUDASSAR-PC\SQL...bo.CustomerLogs

Column Name	Data Type	Allow Nulls
LogId	int	
CustomerId	int	
ACTION	varchar(50)	

> **Note**: You can download the database table SQL by clicking the download link below.
>
> Download SQL file

Triggers

Triggers are database operations which are automatically performed when an action such as Insert, Update or Delete is performed on a Table or a View in database.

Triggers are associated with the Table or View directly i.e. each table has its own Triggers.

Types of Triggers

There are two types of Triggers. After and Instead of Triggers.

After Triggers

These triggers are executed after an action such as Insert, Update or Delete is performed.

Instead of Triggers

These triggers are executed instead of any of the Insert, Update or Delete operations. For example, let's say you write an Instead of Trigger for Delete operation, then whenever a Delete is performed the Trigger will be executed first and if the Trigger deletes record then only the record will be deleted.

After Triggers

Now I will explain you with examples the After Triggers for Insert, Update and Delete operations.

Insert Trigger

Below is an example of an After Insert Trigger. Whenever a row is inserted in the Customers Table, the following trigger will be executed. The newly inserted record is available in the INSERTED table.

The following Trigger is fetching the CustomerId of the inserted record and the fetched value is inserted in the CustomerLogs table.

```
CREATE TRIGGER [dbo].[Customer_INSERT]
       ON [dbo].[Customers]
AFTER INSERT
AS
BEGIN
       SET NOCOUNT ON;

       DECLARE @CustomerId INT

       SELECT @CustomerId = INSERTED.CustomerId
       FROM INSERTED

       INSERT INTO CustomerLogs
       VALUES(@CustomerId, 'Inserted')
END
```

Update Trigger

Below is an example of an After Update Trigger. Whenever a row is updated in the Customers Table, the following trigger will be executed. The updated record is available in the INSERTED table.

The following Trigger is fetching the CustomerId of the updated record. In order to find which column is updated, you will need to use UPDATE function and pass the Column name of the Table to it.

The UPDATE function will return TRUE for a Column if its value was updated else it will return false.

Finally based on which column of the record has been updated a record (containing the CustomerId and the appropriate action) is inserted in the CustomerLogs table.

```
CREATE TRIGGER [dbo].[Customer_UPDATE]
       ON [dbo].[Customers]
AFTER UPDATE
AS
BEGIN
       SET NOCOUNT ON;

       DECLARE @CustomerId INT
       DECLARE @Action VARCHAR(50)
```

```
	SELECT @CustomerId = INSERTED.CustomerId
	FROM INSERTED

	IF UPDATE(Name)
	BEGIN
		SET @Action = 'Updated Name'
	END

	IF UPDATE(Country)
	BEGIN
		SET @Action = 'Updated Country'
	END

	INSERT INTO CustomerLogs
	VALUES(@CustomerId, @Action)
END
```

Delete Trigger

Below is an example of an After Delete Trigger. Whenever a row is delete in the Customers Table, the following trigger will be executed. The deleted record is available in the DELETED table.

The following Trigger is fetching the CustomerId of the deleted record and the fetched value is inserted in the CustomerLogs table.

```
CREATE TRIGGER [dbo].[Customer_DELETE]
	ON [dbo].[Customers]
AFTER DELETE
AS
BEGIN
	SET NOCOUNT ON;

	DECLARE @CustomerId INT

	SELECT @CustomerId = DELETED.CustomerId
	FROM DELETED

	INSERT INTO CustomerLogs
	VALUES(@CustomerId, 'Deleted')
END
```

The following screenshot displays the Log table after the above Triggers were executed.

MUDASSAR-PC\SQL...bo.CustomerLogs

LogId	CustomerId	ACTION
1	5	Inserted
2	5	Updated Name
3	5	Updated Country
4	5	Deleted
5	5	InsteadOf Delete
NULL	NULL	NULL

Instead Of Triggers

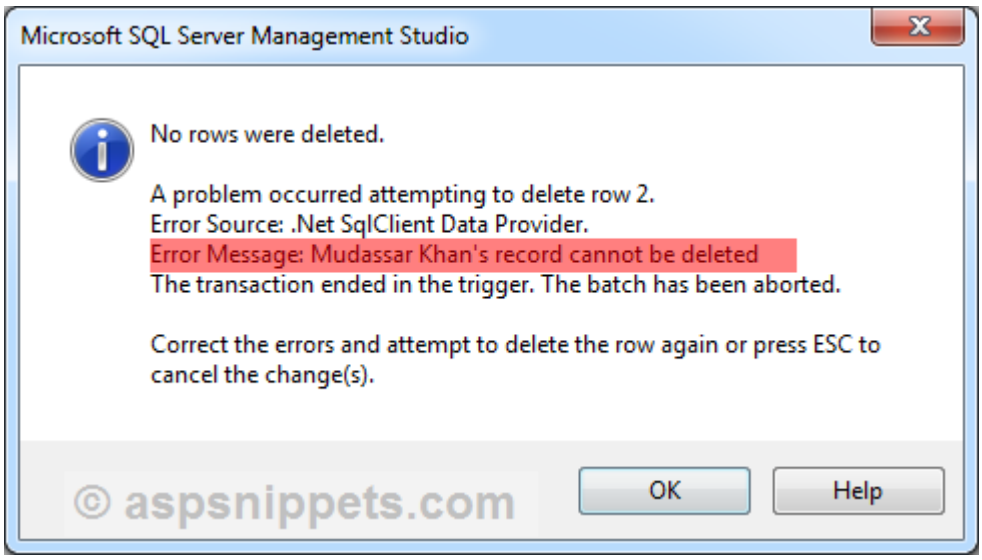
Below is an example of an Instead Of Delete Trigger. Whenever anyone tries to delete a row from the Customers table the following trigger is executed.

Inside the Trigger, I have added a condition that if record has CustomerId value 2 then such a record must not be deleted and an error must be raised. Also a record is inserted in the CustomerLogs table.

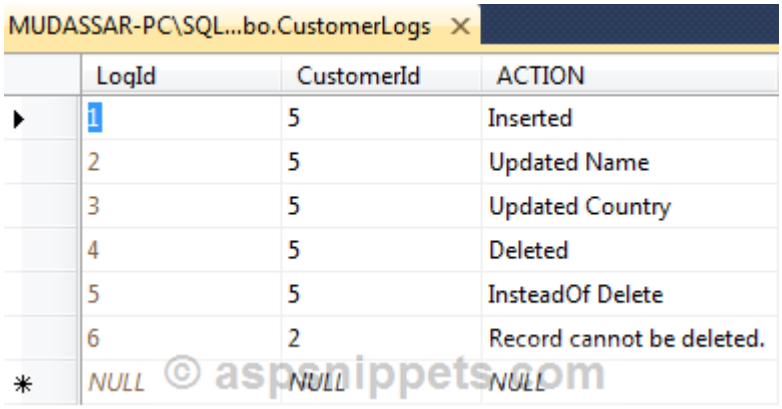
If the CustomerId value is not 2 then a delete query is executed which deletes the record permanently and a record is inserted in the CustomerLogs table.

```
CREATE TRIGGER [dbo].[Customer_InsteadOfDELETE]
       ON [dbo].[Customers]
INSTEAD OF DELETE
AS
BEGIN
       SET NOCOUNT ON;

       DECLARE @CustomerId INT

       SELECT @CustomerId = DELETED.CustomerId
       FROM DELETED

       IF @CustomerId = 2
       BEGIN
              RAISERROR('Mudassar Khan''s record cannot be deleted',16 ,1)
              ROLLBACK
              INSERT INTO CustomerLogs
              VALUES(@CustomerId, 'Record cannot be deleted.')
       END
       ELSE
       BEGIN
              DELETE FROM Customers
              WHERE CustomerId = @CustomerId

              INSERT INTO CustomerLogs
              VALUES(@CustomerId, 'Instead Of Delete')
       END
END
```

The following error message shown when record with CustomerId 2 is deleted.



The following screenshot displays the Log table after the Instead Of Trigger is executed.



MUDASSAR-PC\SQL...bo.CustomerLogs

LogId	CustomerId	ACTION
1	5	Inserted
2	5	Updated Name
3	5	Updated Country
4	5	Deleted
5	5	InsteadOf Delete
6	2	Record cannot be deleted.
NULL	NULL	NULL

To test the Customer Insert triggers, execute the following code:

```
USE [test]
GO

INSERT INTO Customers(Name,Country) VALUES('Frank Furter', 'USA');
INSERT INTO Customers(Name,Country) VALUES('Scooby Doo', 'USA');
INSERT INTO Customers(Name,Country) VALUES('Victor Frankenstein', 'Transylvannia');
INSERT INTO Customers(Name,Country) VALUES('Freddie Flintstone', 'Bedrock USA');

GO
```

There is an insert trigger on the customer table. The trigger you created will add the customer information to the CustomerLog table. Right click on that table and select the option to view 1000 rows

For deletes, there are 2 triggers. The instead of trigger will run first. It will either delete the record or prohibit deletion if you try to delete customer id 2. The trigger will add notations to the customer log indicating whether the record was deleted or not. After that runs, the "after trigger" will run and it will retrieve the deleted record from a temp deleted record table and insert it into the log. For customerId 5 below, you will see 2 entries in the log. For customer id, you will only see 1 entry indicating it was not deleted.

To test the Customer Delete trigger, execute the following code:

```
USE [test]
GO
DELETE FROM Customers WHERE CustomerId=5;
GO
DELETE FROM Customers WHERE CustomerId=2;   --  this will create an error because it is prohibited in the trigger
GO
```

The results should show customer id 5 is removed and customer id 2 remains because the trigger won't allow it to be deleted.

For updates, there is only 1 trigger it checks to see if you updated the name or country and adds a log to the customer log
To test the trigger, enter the code below and then check the customer log

```
USE [test]
GO
UPDATE Customers  SET [Name] = 'Daphne Blake' WHERE CustomerId=1;
UPDATE Customers SET [Country] ='France' WHERE CustomerId=2;
GO
```

NOTE: To see the triggers after you create them, expand the tables (you will see a triggers folder) Expand that folder and then double click a trigger and the code will display