**Understanding the Basic of Triggers**

By [Tharindu Dhaneenja](http://www.sql-server-performance.com/author/tharindudhaneenja/)

 A trigger is an operation that is executed when some kind of event occurs to the database. It can be a data or object change.  Listed below are the different types of tiggers:  
  
Types of Triggers

* DML(data manipulation language) triggers (SQL Server 2000- 80.0)
  + AFTER triggers (Only Tables)
  + INSTEAD OF triggers (Tables or Views)
* DDL(data definition language) triggers (SQL Server 2005- 90.0)
* SQLCLR triggers (SQL Server 2005- 90.0)

Rules of Triggers

* cannot create or modify Database objects using triggers
  + For example, cannot perform “CREATE TABLE… or ALTER TABLE” sql statements under the triggers
* cannot perform any administrative tasks
  + For example, cannot perform “BACKUP DATABASE…” task under the triggers
* cannot pass any kind of parameters
* cannot directly call triggers
* WRITETEXT statements do not allow a trigger

Advantage of Triggers  
  
Triggers are useful for auditing data changes or auditing database as well as managing business rules.  Below are some examples:

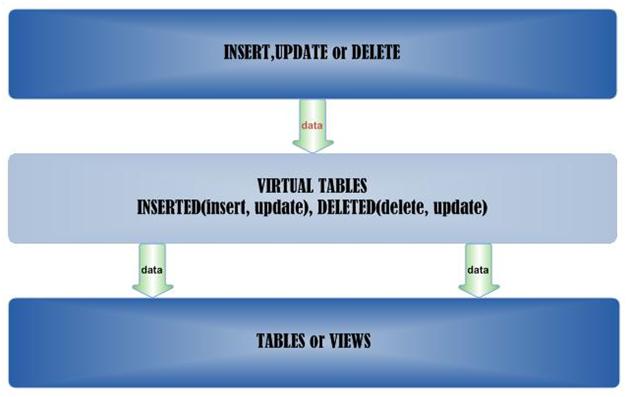
* Triggers can be used to enforce referential integrity (For example you may not be able to apply foreign keys)
* Can access both new values and old values in the database when going to do any insert, update or delete

Disadvantage of Triggers

* Triggers hide database operations.  For example when debugging a stored procedure, it’s possible to not be aware that a trigger is on a table being checked for data changes
* Executing triggers can affect the performance of a bulk import operation

Best Programming Practice

* Do not use triggers unnecessarily, if using triggers use them to resolve a specific situation
* Where possible, replace a trigger operation with a stored procedure or another kind of operation
* Do not write lengthy triggers as they can increase transaction duration; and also reduce the performance of data insert, update and delete operations as the trigger is fired everytime the operation occurs.

How Triggers Work?  
  
  
  
When a trigger is executed SQL Server creates two virtual tables called INSERTED and DELETED. For example when inserting a recode into a table, SQL Server creates a virtual table call INSERTED and loads data into the inserted table then executes the trigger statements and writes the related data pages. One feature of this is joining other tables to the virtual tables.  
  
DML (data manipulation language)  
One Table or View can have multiple triggers. Table triggers are executed whenever table data changes and view triggers only execute when data changes are made through the view.  
  
Syntax:  
Trigger on an INSERT, UPDATE, or DELETE statement to a table or view (DML Trigger)  
CREATE TRIGGER [schema\_name.]trigger\_name   
ON {table | view}   
[WITH <dml\_trigger\_option> [ ,...n ] ]  
{FOR | AFTER | INSTEAD OF}   
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }   
[WITH APPEND]   
[NOT FOR REPLICATION]   
AS { sql\_statement  [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier [ ; ] > }

 <dml\_trigger\_option>::=  
    [ENCRYPTION]  
    [EXECUTE AS Clause]

<method\_specifier> ::=  
    assembly\_name.class\_name.method\_name

AFTER triggers(Only Tables)

An after trigger is a database object that is Bind to a related table and these are executed after the action of the INSERT, UPDATE, or DELETE statement is performed. It’s never executed if a constraint violation occurs. This cannot be defined on views.  
  
Example  
USE [AdventureWorks]  
GO  
–create table  
CREATE TABLE [tblDataAuditLog](  
 [DALID] [int] IDENTITY(1,1) NOT NULL,  
 [comments] [varchar](250) NOT NULL,  
 [recordedDate] [datetime] NOT NULL CONSTRAINT [DF\_tblDataAuditLog\_recordedDate]  DEFAULT (getdate()),  
 [transactionType] [char](1) NOT NULL  
);  
GO  
— Drop the trigger.  
IF OBJECT\_ID (‘Sales.trgDataAuditInsert’,'TR’) IS NOT NULL  
    DROP TRIGGER Sales.trgDataAuditInsert  
GO  
— Create the trigger (INSERT)  
CREATE TRIGGER [Sales].[trgDataAuditInsert]   
  ON [Sales].[Customer]  
  AFTER INSERT   
AS   
 DECLARE @AccountNumber varchar(15)  
 SELECT @AccountNumber=INSERTED.AccountNumber FROM INSERTED  
 INSERT INTO [dbo].[tblDataAuditLog]([comments],[transactionType])  
 SELECT ‘Insert new customer – ‘+ @AccountNumber , ‘I’;  
GO  
— Create the trigger (DELETE)  
CREATE TRIGGER [Sales].[trgDataAuditDelete]   
  ON [Sales].[Customer]  
  AFTER DELETE   
AS   
 DECLARE @AccountNumber varchar(15)  
 SELECT @AccountNumber=DELETED.AccountNumber FROM DELETED  
 INSERT INTO [dbo].[tblDataAuditLog]([comments],[transactionType])  
 SELECT ‘Delete customer – ‘+ @AccountNumber , ‘D’;  
GO  
— Create the trigger (UPDATE)  
CREATE TRIGGER [Sales].[trgDataAuditUpdate]   
  ON [Sales].[Customer]  
  AFTER UPDATE   
AS   
 DECLARE @AccountNumber varchar(15)  
 SELECT @AccountNumber=DELETED.AccountNumber FROM DELETED  
 INSERT INTO [dbo].[tblDataAuditLog]([comments],[transactionType])  
 SELECT ‘Update customer – ‘+ @AccountNumber , ‘U’;  
GO

– Disable trigger  
IF OBJECT\_ID (‘Sales.trgDataAuditInsert’,'TR’) IS NOT NULL  
    DISABLE TRIGGER Sales.trgDataAuditInsert ON Sales.Customer;  
GO

–insert   
INSERT INTO sales.Customer(TerritoryID, CustomerType, ModifiedDate)  
VALUES(1,’S',getdate());  
–update  
UPDATE Sales.Customer  
SET ModifiedDate=getdate()+2  
WHERE CustomerID=(SELECT MAX(CustomerID) FROM Sales.Customer);  
–delete  
DELETE FROM Sales.Customer  
WHERE CustomerID=(SELECT MAX(CustomerID) FROM Sales.Customer);  
GO  
SELECT \* FROM [tblDataAuditLog]  
GO

[ASK A QUESTION](http://www.sql-server-performance.com/forum/)