Triggers -- Sql Server

By s.india, 26 Apr 2008

INTRODUCTION

TRIGGERS IN SQL SERVER

BACKGROUND

This article gives a brief introduction about Triggers in Sql Server 2000/2005.

What is a Trigger

A trigger is a special kind of a store procedure that executes in response to certain action on the table like insertion, deletion or updation of data. It is a database object which is bound to a table and is executed automatically. You can't explicitly invoke triggers. The only way to do this is by performing the required action no the table that they are assigned to.

Types Of Triggers

There are three action query types that you use in SQL which are INSERT, UPDATE and DELETE. So, there are three types of triggers and hybrids that come from mixing and matching the events and timings that fire them.

Basically, triggers are classified into two main types:-

(i) After Triggers (For Triggers)
(ii) Instead Of Triggers

(i) After Triggers

These triggers run after an insert, update or delete on a table. They are **not supported for views.**
AFTER TRIGGERS can be classified further into three types as:

(a) AFTER INSERT Trigger.
(b) AFTER UPDATE Trigger.
(c) AFTER DELETE Trigger.

Let's create After triggers. First of all, let's create a table and insert some sample data. Then, on this table, I will be attaching several triggers.

```
CREATE TABLE Employee_Test
(
Emp_ID INT Identity,
Emp_name Varchar(100),
Emp_Sal Decimal (10,2)
)

INSERT INTO Employee_Test VALUES ('Anees',1000);
INSERT INTO Employee_Test VALUES ('Rick',1200);
INSERT INTO Employee_Test VALUES ('John',1100);
INSERT INTO Employee_Test VALUES ('Stephen',1300);
INSERT INTO Employee_Test VALUES ('Maria',1400);
```

I will be creating an AFTER INSERT TRIGGER which will insert the rows inserted into the table into another audit table. The main purpose of this audit table is to record the changes in the main table. This can be thought of as a generic audit trigger.

Now, create the audit table as:-

```
CREATE TABLE Employee_Test_Audit
(
```

```
Emp_ID int,
Emp_name varchar(100),
Emp_Sal decimal (10,2),
Audit_Action varchar(100),
Audit_Timestamp datetime
)
```

(a) AFTRE INSERT Trigger

This trigger is fired after an INSERT on the table. Let's create the trigger as:-

```
CREATE TRIGGER trgAfterInsert ON [dbo].[Employee_Test]
FOR INSERT
AS
declare @empid int;
declare @empname varchar(100);
declare @empsal decimal(10,2);
declare @audit_action varchar(100);

select @empid=i.Emp_ID from inserted i;
select @empname=i.Emp_Name from inserted i;
select @empsal=i.Emp_Sal from inserted i;
set @audit_action='Inserted Record -- After Insert Trigger.';

insert into Employee_Test_Audit
          (Emp_ID,Emp_Name,Emp_Sal,Audit_Action,Audit_Timestamp)
values(@empid,@empname,@empsal,@audit_action,getdate());

PRINT 'AFTER INSERT trigger fired.'
GO
```

The CREATE TRIGGER statement is used to create the trigger. THE ON clause specifies the table name on which the trigger is to be attached. The FOR INSERT specifies that this is an AFTER INSERT trigger. In place of FOR INSERT, AFTER INSERT can be used. Both of them mean the same.
In the trigger body, table named **inserted** has been used. This table is a logical table and contains the row that has been inserted. I have selected the fields from the logical inserted table from the row that has been inserted into different variables, and finally inserted those values into the Audit table.
To see the newly created trigger in action, lets insert a row into the main table as :

```
 insert into Employee_Test values('Chris',1500);
```

Now, a record has been inserted into the Employee_Test table. The AFTER INSERT trigger attached to this table has inserted the record into the Employee_Test_Audit as:-

```
6   Chris  1500.00   Inserted Record -- After Insert Trigger. 2008-04-26 12:00:55.700
```

(b) AFTER UPDATE Trigger

This trigger is fired after an update on the table. Let's create the trigger as:-

```
CREATE TRIGGER trgAfterUpdate ON [dbo].[Employee_Test]
FOR UPDATE
AS
declare @empid int;
declare @empname varchar(100);
declare @empsal decimal(10,2);
declare @audit_action varchar(100);

select @empid=i.Emp_ID from inserted i;
select @empname=i.Emp_Name from inserted i;
select @empsal=i.Emp_Sal from inserted i;

if update(Emp_Name)
  set @audit_action='Updated Record -- After Update Trigger.';
if update(Emp_Sal)
  set @audit_action='Updated Record -- After Update Trigger.';

insert into Employee_Test_Audit(Emp_ID,Emp_Name,Emp_Sal,Audit_Action,Audit_Timestamp)
values(@empid,@empname,@empsal,@audit_action,getdate());

PRINT 'AFTER UPDATE Trigger fired.'
GO
```

The AFTER UPDATE Trigger is created in which the updated record is inserted into the audit table. There is **no logical table updated like the logical table inserted.** We can obtain the updated value of a field from the **update(column_name)** function. In our trigger, we have used, **if update(Emp_Name)** to check if the column Emp_Name has been updated. We have similarly checked the column Emp_Sal for an update.
Let's update a record column and see what happens.

```
 update Employee_Test set Emp_Sal=1550 where Emp_ID=6
```

This inserts the row into the audit table as:-

```
6  Chris  1550.00  Updated Record -- After Update Trigger.   2008-04-26 12:38:11.843
```

(c) AFTER DELETE Trigger

This trigger is fired after a delete on the table. Let's create the trigger as:-

```
CREATE TRIGGER trgAfterDelete ON [dbo].[Employee_Test]
AFTER DELETE
AS
declare @empid int;
declare @empname varchar(100);
declare @empsal decimal(10,2);
declare @audit_action varchar(100);

select @empid=d.Emp_ID from deleted d;
select @empname=d.Emp_Name from deleted d;
select @empsal=d.Emp_Sal from deleted d;
set @audit_action='Deleted -- After Delete Trigger.';

insert into Employee_Test_Audit
(Emp_ID,Emp_Name,Emp_Sal,Audit_Action,Audit_Timestamp)
values(@empid,@empname,@empsal,@audit_action,getdate());

PRINT 'AFTER DELETE TRIGGER fired.'
GO
```

In this trigger, the deleted record's data is picked from the **logical deleted table** and inserted into the audit table.
Let's fire a delete on the main table.
A record has been inserted into the audit table as:-

```
 6  Chris 1550.00  Deleted -- After Delete Trigger.  2008-04-26 12:52:13.867
```

All the triggers can be enabled/disabled on the table using the statement

```
ALTER TABLE Employee_Test {ENABLE|DISBALE} TRIGGER ALL
```

Specific Triggers can be enabled or disabled as :-

```
ALTER TABLE Employee_Test DISABLE TRIGGER trgAfterDelete
```

This disables the After Delete Trigger named trgAfterDelete on the specified table.

(ii) Instead Of Triggers

These can be used as an interceptor for anything that anyonr tried to do on our table or view. If you define an *Instead Of trigger* on a table for the Delete operation, they try to delete rows, and they will not actually get deleted (unless you issue another delete instruction from within the trigger)
INSTEAD OF TRIGGERS can be classified further into three types as:-

(a) INSTEAD OF INSERT Trigger.
(b) INSTEAD OF UPDATE Trigger.
(c) INSTEAD OF DELETE Trigger.

(a) Let's create an Instead Of Delete Trigger as:-

```
CREATE TRIGGER trgInsteadOfDelete ON [dbo].[Employee_Test]
INSTEAD OF DELETE
AS
declare @emp_id int;
declare @emp_name varchar(100);
declare @emp_sal int;

select @emp_id=d.Emp_ID from deleted d;
select @emp_name=d.Emp_Name from deleted d;
select @emp_sal=d.Emp_Sal from deleted d;

BEGIN
  if(@emp_sal>1200)
  begin
   RAISERROR('Cannot delete where salary > 1200',16,1);
   ROLLBACK;
  end
  else
  begin
   delete from Employee_Test where Emp_ID=@emp_id;
   COMMIT;
   insert into Employee_Test_Audit(Emp_ID,Emp_Name,Emp_Sal,Audit_Action,Audit_Timestamp)
   values(@emp_id,@emp_name,@emp_sal,'Deleted -- Instead Of Delete Trigger.',getdate());
   PRINT 'Record Deleted -- Instead Of Delete Trigger.'
  end
END
GO
```

This trigger will prevent the deletion of records from the table where Emp_Sal > 1200. If such a record is deleted, the Instead Of Trigger will rollback the transaction, otherwise the transaction will be committed.
Now, let's try to delete a record with the Emp_Sal >1200 as:-

```
delete from Employee_Test where Emp_ID=4
```

This will print an error message as defined in the RAISE ERROR statement as:-

```
Server: Msg 50000, Level 16, State 1, Procedure trgInsteadOfDelete, Line 15
Cannot delete where salary > 1200
```

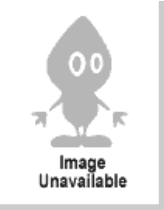
And this record will not be deleted.
In a similar way, you can code Instead of Insert and Instead Of Update triggers on your tables.

CONCLUSION

In this article, I took a brief introduction of triggers, explained the various kinds of triggers – After Triggers and Instead Of Triggers along with their variants and explained how each of them works. I hope you will get a clear understanding about the Triggers in Sql Server and their usage.

License

About the Author



s.india
Software Developer (Senior)
India

No Biography provided

Comments and Discussions