THE LECTURE 8

TRIGGERS

TRIGGER

A trigger is a special method of stored procedure and it invokes automatically when an event starts in the database server. DML triggers execute when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

TYPES OF TRIGGERS

Triggers contain three types as follows;

- DML Triggers
- DDL Triggers
- Logon Triggers

DML TRIGGERS

DML stands for Data Manipulation Language. INSERT, UPDATE, and DELETE statements are DML statements. DML triggers get fired whenever data is modified using INSERT, UPDATE, and DELETE events.

DML triggers can be again classified into 2 types

- 1. After triggers (Sometimes called FOR triggers)
- 2. Instead of triggers

AFTER TRIGGER

- After triggers get fired after only with a condition when a modification action occurs. The INSERT, UPDATE and DELETE commands because of an after trigger gets fired after the execution of a complete statement.
- Therefore, we need to use tblEmployee and tblEmployeeAudit tables for further examples as follows;
- SQL Script to create tblEmployee tab

```
CREATE TABLE tblEmployee
(
  Id int Primary Key,
  Name nvarchar(30),
  Salary int,
  Gender nvarchar(10),
  DepartmentId int
)

Insert data into tblEmployee table
Insert into tblEmployee values (1,'John', 5000, 'Male', 3)
Insert into tblEmployee values (2,'Mike', 3400, 'Male', 2)
Insert into tblEmployee values (3,'Pam', 6000, 'Female', 1)
```

AFTER TRIGGERS

- Once a new Employee is added to the table in the database. Now, I want to retrieve the ID, date and time, the new employee is added to the tblEmployeeAudit table.
- The easiest way to get the same result by using an AFTER TRIGGER for INSERT event.
- **AFTER TRIGGER FOR INSERTION**
- Example for AFTER TRIGGER for INSERT event on the tblEmployee table as follows;

```
CREATE TRIGGER tr_tblEMployee_ForInsert
ON tblEmployee
FOR INSERT
AS
BEGIN
 Declare @Id int
 Select @Id = Id from inserted

 insert into tblEmployeeAudit
 values('New employee with Id  = ' + Cast(@Id as nvarchar(5)) + ' is added at ' +
cast(Getdate() as nvarchar(20)))
END
```

AFTER TRIGGERS

- Insert into tblEmployee values (7,'Tan', 2300, 'Female', 3)
- when a row deletes records from the table tblEmployee.
- Example for AFTER TRIGGER for DELETE event a tblEmployee table:

```
CREATE TRIGGER tr_tblEMployee_ForDelete
ON tblEmployee
FOR DELETE
AS
BEGIN
 Declare @Id int
 Select @Id = Id from deleted

 insert into tblEmployeeAudit
 values('An existing employee with Id  = ' + Cast(@Id as nvarchar(5)) + ' is deleted at ' +
Cast(Getdate() as nvarchar(20)))
END
```

AFTER UPDATE TRIGGER

- Triggers work with two organized tables, INSERTED and DELETED. Newly updated data stored in the inserted table and old specific data stored in the deleted table. After triggering for UPDATE event makes use of both inserted and deleted tables.
- Create AFTER UPDATE trigger script:

```
Create trigger tr_tblEmployee_ForUpdate
on tblEmployee
for Update
as
Begin
 Select * from deleted
 Select * from inserted
End
```

INSTEAD OF INSERT TRIGGER

- As we well know that AFTER triggers get fired after the triggering action (INSERT, UPDATE or DELETE events), whereas, INSTEAD OF triggers get fired instead of the triggering action (INSERT, UPDATE or DELETE events).
- In addition, INSTEAD OF Insert triggers makes use for correctly update views that are based on multiple tables.

```
CREATE TABLE tblEmployee
( Id int Primary Key,
Name nvarchar(30),
Gender nvarchar(10),
DepartmentId int)
```

SQL SCRIPT TO CREATE TBLDEPARTMENT TABLE:

```
CREATE TABLE tblDepartment
(DeptId int Primary Key,
DeptName nvarchar(20))
```

INSTEAD OF INSERT TRIGGER

Insert data into tblDepartment table

```
Insert into tblDepartment values (1,'IT')
Insert into tblDepartment values (2,'Payroll')
Insert into tblDepartment values (3,'HR')
Insert into tblDepartment values (4,'Admin')
```

Insert data into tblEmployee table

```
Insert into tblEmployee values (1,'John', 'Male', 3)
Insert into tblEmployee values (2,'Mike', 'Male', 2)
Insert into tblEmployee values (3,'Pam', 'Female', 1)
Insert into tblEmployee values (4,'Todd', 'Male', 4)
Insert into tblEmployee values (5,'Sara', 'Female', 1)
Insert into tblEmployee values (6,'Ben', 'Male', 3)
```

CREATING A VIEW

So, we have our two required tables, let's create a view which is based on these two tables, it will fetch the records of Employee Id, Name, Gender and DepartmentName columns. Therefore, the view is based on multiple tables.

SCRIPT TO CREATE A VIEW:

```
Create view vWEmployeeDetails

as

Select Id, Name, Gender, DeptName

from tblEmployee

join tblDepartment

on tblEmployee.DepartmentId = tblDepartment.DeptId
```

Once you execute this line, Select * from vWEmployeeDetails, It will retrieve all the records from the table as follows;

INSERTING INTO A VIEW

- So, let's insert a single row into the view function, vWEmployeeDetails, by running the following query. At this moment, it will throw an error like "View or function vWEmployeeDetails is not updatable because the modification affects multiple base tables."
- Insert into vWEmployeeDetails values (7, 'Valarie', 'Female', 'IT')
- Finally, we inserted a row above into a view which is based on multiple tables, it gives an error by default.

INSTEAD OF INSERT TRIGGER

- Now, let’s have a look into this, how INSTEAD OF TRIGGERS give us help in this condition. Since we are facing an error, when we try to insert a single row into the view function, let’s make an INSTEAD OF INSERT trigger on the view vWEmployeeDetails.

INSTEAD OF INSERT TRIGGER

- **SCRIPT TO CREATE INSTEAD OF INSERT TRIGGER:**

```
Create trigger tr_vWEmployeeDetails_InsteadOfInsert
on vWEmployeeDetails
Instead Of Insert
as
Begin
 Declare @DeptId int

 --Check if there is a valid DepartmentId
 --for the given DepartmentName
 Select @DeptId = DeptId
 from tblDepartment
 join inserted
 on inserted.DeptName = tblDepartment.DeptName

 --If DepartmentId is null throw an error
 --and stop processing
 if(@DeptId is null)
 Begin
  Raiserror('Invalid Department Name. Statement terminated', 16, 1)
  return
 End

 --Finally insert into tblEmployee table
 Insert into tblEmployee(Id, Name, Gender, DepartmentId)
 Select Id, Name, Gender, @DeptId
 from inserted
End
```

INSTEAD OF UPDATE TRIGGER

An INSTEAD OF UPDATE triggers gets fired instead of an update event, that can be on a table or a view function. For example, let's understand, an INSTEAD OF UPDATE trigger, and then when you try to make the update into the row within that view function or table, instead of the UPDATE, in this situation, the trigger get invoked automatically. Instead of update trigger based on multiple tables.

So, let's create both the tables Employee and Department as follows.

SQL SCRIPT TO CREATE TBLEMPLOYEE TABLE:

- CREATE TABLE tblEmployee
- (Id int Primary Key,
- Name nvarchar(30),
- Gender nvarchar(10),
- DepartmentId int)

SQL SCRIPT TO CREATE TBLDEPARTMENT TABLE

- CREATE TABLE tblDepartment
- (DeptId int Primary Key,
- DeptName nvarchar(20)
-)

INSTEAD OF UPDATE TRIGGER

So, we have our required two tables, let's create a view which is based on these two tables, it will fetch the records of Employee Id, Name, Gender and DepartmentName columns which are based on multiple tables.

SCRIPT TO CREATE THE VIEW:

- Create view vWEmployeeDetails
- as
- Select Id, Name, Gender, DeptName
- from tblEmployee
- join tblDepartment
- on tblEmployee.DepartmentId = tblDepartment.DeptId

Once you execute this line, Select * from vWEmployeeDetails, It will retrieve all the records from the table as follows

INSTEAD OF UPDATE TRIGGER

Id	Name	Gender	DeptName
1	John	Male	HR
2	Mike	Male	Payroll
3	Pam	Female	IT
4	Todd	Male	Admin
5	Sara	Female	IT
6	Ben	Male	HR

INSTEAD OF UPDATE TRIGGER

In above example, when we inserted a single row into the view table and got an error statement like- ‘View or function vWEmployeeDetails is not updatable because the modification affects multiple base tables.‘

So, let’s quickly update the view function, in addition, it affects both the tables, and if we face the same error statement. Then, the UPDATE command changes its column “Name” from the table tblEmployee and column “DeptName” from the table tblDepartment.

So, when we run this query, we face the same error.

- Update vWEmployeeDetails
- set Name = ‘Johny’, DeptName = ‘IT’
- where Id = 1

So, let’s do some change in the department of John from HR to IT. The UPDATE query runs only one table and that is the tblDepartment table. So, the query may succeed. But, there is a condition before executing the query, please make note that employees name JOHN and BEN both are in HR department

INSTEAD OF UPDATE TRIGGER

- Update vWEmployeeDetails
- set DeptName = ‘IT’
- where Id = 1

After execution of the query, now select all data records from the view function, and note that BEN’sDeptName has also changed to IT. We also change JOHN’s DeptName. So, the UPDATE command did not work as we expected. Why it happened, because of the UPDATE query command, updated column name DeptName from HR to the IT, in the tblDepartment table. For an update, we need to change the DeptId of JOHN from 3 to 1.

INSTEAD OF UPDATE TRIGGER

- **INCORRECTLY UPDATED VIEW**

Id	Name	Gender	DeptName
1	John	Male	IT
2	Mike	Male	Payroll
3	Pam	Female	IT
4	Todd	Male	Admin
5	Sara	Female	IT
6	Ben	Male	IT

Record with Id = 3, has the DeptName changed from 'HR' to 'IT'

DeptId	DeptName
1	IT
2	Payroll
3	IT
4	Admin

We should have actually updated, JOHN's DepartmentId from 3 to 1

Id	Name	Gender	DepartmentId
1	John	Male	3
2	Mike	Male	2
3	Pam	Female	1
4	Todd	Male	4
5	Sara	Female	1
6	Ben	Male	3

INSTEAD OF UPDATE TRIGGER

- **SCRIPT TO CREATE INSTEAD OF UPDATE TRIGGER:**

```
Create Trigger tr_vWEmployeeDetails_InsteadOfUpdate
on vWEmployeeDetails
instead of update
as
Begin
 -- if EmployeeId is updated
 if(Update(Id))
 Begin
  Raiserror('Id cannot be changed', 16, 1)
  Return
 End

 -- If DeptName is updated
 if(Update(DeptName))
 Begin
  Declare @DeptId int

  Select @DeptId = DeptId
  from tblDepartment
  join inserted
  on inserted.DeptName = tblDepartment.DeptName

  if(@DeptId is NULL )
  Begin
   Raiserror('Invalid Department Name', 16, 1)
   Return
  End

  Update tblEmployee set DepartmentId = @DeptId
  from inserted
  join tblEmployee
  on tblEmployee.Id = inserted.id
 End

 -- If gender is updated
 if(Update(Gender))
 Begin
  Update tblEmployee set Gender = inserted.Gender
  from inserted
  join tblEmployee
  on tblEmployee.Id = inserted.id
 End

 -- If Name is updated
 if(Update(Name))
 Begin
  Update tblEmployee set Name = inserted.Name
  from inserted
  join tblEmployee
  on tblEmployee.Id = inserted.id
 End
End
```

INSTEAD OF DELETE TRIGGER

- An INSTEAD OF DELETE trigger gets fired in the state of the DELETE event on a table or a view. Let's understand with an example. Let's assume, an INSTEAD OF DELETE trigger on a view or a table, and when you try to update a single row from that view or table, in the state of a real DELETE event, then the trigger automatically gets fired.
- INSTEAD OF DELETE TRIGGERS only used to delete data in terms of records from a view or a table, which is based on multiple tables.

INSTEAD OF DELETE TRIGGER

Let’s create two tables Employee and Department.

SQL SCRIPT TO CREATE TBLEMPLOYEE TABLE:

- CREATE TABLE tblEmployee
- (Id int Primary Key,
- Name nvarchar(30),
- Gender nvarchar(10),
- DepartmentId int)

SQL SCRIPT TO CREATE TBLDEPARTMENT TABLE

- CREATE TABLE tblDepartment
- (DeptId int Primary Key,
- DeptName nvarchar(20)
-)

INSTEAD OF DELETE TRIGGER

Since we now have the required tables, let's create a view based on these tables.

And, this view will return Employee Id, Name, Gender and DepartmentName columns. So, the view is based on multiple tables.

SCRIPT TO CREATE THE VIEW:

- Create view vWEmployeeDetails
- as
- Select Id, Name, Gender, DeptName
- from tblEmployee
- join tblDepartment
- on tblEmployee.DepartmentId = tblDepartment.DeptId

INSTEAD OF DELETE TRIGGER

- Once you execute this line, Select * from vWEmployeeDetails, It will retrieve all the records from the table as follows;
- In above example, when we inserted a single row into the view table and got an error statement like- ‘View or function vWEmployeeDetails is not updatable because the modification affects multiple base tables.‘
- Although, when we tried to update a view which is based on multiple tables, we faced the same error. To get the error, it will affect both the base tables. If the update query affects only one base table, we do not get the error, but the UPDATE query does not work properly if the “DeptName” column gets updated.
- Now, let’s try to delete a row from the view, and we get the same error.
- Delete from vWEmployeeDetails where Id = 1

Id	Name	Gender	DeptName
1	John	Male	HR
2	Mike	Male	Payroll
3	Pam	Female	IT
4	Todd	Male	Admin
5	Sara	Female	IT
6	Ben	Male	HR

INSTEAD OF DELETE TRIGGER

SCRIPT TO CREATE INSTEAD OF DELETE TRIGGER:

```
Create Trigger tr_vWEmployeeDetails_InsteadOfDelete
on vWEmployeeDetails
instead of delete
as
Begin
Delete tblEmployee
from tblEmployee
join deleted
on tblEmployee.Id = deleted.Id
```

INSTEAD OF DELETE TRIGGER

The trigger tr_vWEmployeeDetails_InsteadOfDelete applicable in DELETED table. But Deleted table contains all the rows that we tried to DELETE from the view. So, we join the DELETED table with table tblEmployee to delete the unwanted rows. In such cases, Joins are much faster than the subqueries.

When you execute the following DELETE command, the row gets DELETED as expected from tblEmployee table

- Delete from vWEmployeeDetails where Id = 1
- A small difference among the triggers as given below

Trigger	INSERTED or DELETED?
Instead of Insert	DELETED table is always empty and the INSERTED table contains the newly inserted data.
Instead of Delete	INSERTED table is always empty and the DELETED table contains the rows deleted
Instead of Update	DELETED table contains OLD data (before update), and inserted table contains NEW data(Updated data)