TRIGGERS
References

• SQL : The Complete Reference (Second Edition)
  JAMES R. GROFF AND PAUL N. WEINBERG
Triggers

- A trigger is a special set of stored procedure code whose activation is caused by modifications to the database contents.
- Unlike stored procedures created with a CREATE PROCEDURE statement, a trigger is not activated by a CALL or EXECUTE statement. Instead, the trigger is associated with a database table.
- When the data in the table is changed (by an INSERT, DELETE, or UPDATE statement), the trigger is fired, which means that the DBMS executes the SQL statements that make up the body of the trigger.
- Triggers can be used to cause automatic updates of information within a database.
• Triggers are normally used in two areas: creating audit records and reflecting changes to crucial business tables, and validating changes against a set of business rules coded in SQL.

• Triggers can be assigned to tables or views. There are two types of triggers,
  • **INSTEAD OF** and **FOR | AFTER**

• After triggers fires after the triggering action. INSERT, DELETE, UPDATE action cause to fire the trigger after execution of the respective statement

• INSTEAD of triggers fires instead of the triggering action. INSERT, DELETE, UPDATE action cause to fire the trigger instead of the execution of the respective statement
WHY USE TRIGGERS?

• **Improve data integrity**
  
  • When an action is performed on data, it is possible to check if the manipulation of the data follows the underlying business rules. If not so, it can avoid erroneous entries to the table.

  • **Ex:** when a request is made to withdraw cash from an account, the stored procedure will create a record on the client's statement table for the withdrawal, and the trigger will automatically reduce the balance as required.

  • The trigger may also be the point at which a check is made on the client's balance to verify that there is enough balance to allow the withdrawal.
Process A updates Table A

Trigger fires on Table A change. Updates Table B

Trigger fires on Table B change. Updates Table A
Advantages and Disadvantages

Auditing changes.

A trigger can detect and disallow specific updates and changes that should not be permitted in the database.

Cascaded operations. A trigger can detect an operation within the database (such as deletion of a customer or salesperson) and automatically cascade the impact throughout the database (such as adjusting account balances or sales targets).
Advantages and Disadvantages

Enforce interrelationships. A trigger can enforce more complex interrelationships among the data in a database than those that can be expressed by simple referential integrity constraints or check constraints, such as those that require a sequence of SQL statements or IF...THEN...ELSE processing.

Stored procedure invocation. A trigger can call one or more stored procedures or even invoke actions outside the DBMS itself through external procedure calls in response to database updates.
DML Triggers

DML triggers are invoked when any DML commands like INSERT, DELETE, and UPDATE happen on the data of a table and or view.

AFTER triggers

fire after the data modification statement completes but before the statement's work is committed to the databases.

Instead of Triggers

INSTEAD OF Triggers fire instead of the operation that fires the trigger, so if you define an INSTEAD OF trigger on a table for the Insert/Delete/update operation, they try to delete/update rows, they will not actually get insert/deleted /updated.
CREATE TRIGGER

Create trigger Tr_Name-----|Trigger name
On employee ------Name of the table that trigger created on
For Operation -----Operation type(insert/update/delete)
AS
Begin

--------------------------(Body)

END;
Ex: Insert operation (After)

• Create a trigger to get an audit record of Added Emp ID and date when insert records to the employee table

<table>
<thead>
<tr>
<th>EmpID</th>
<th>AuditData</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>New Employee=8 Added on Feb 23 2018 7:17PM</td>
</tr>
</tbody>
</table>

Employee table

<table>
<thead>
<tr>
<th>EmpID</th>
<th>EmpName</th>
<th>Gender</th>
<th>DepartmentId</th>
<th>JoinedDate</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>Male</td>
<td>3</td>
<td>2018-02-01</td>
<td>25000</td>
</tr>
<tr>
<td>2</td>
<td>Smith</td>
<td>Female</td>
<td>4</td>
<td>2017-05-01</td>
<td>35000</td>
</tr>
<tr>
<td>3</td>
<td>Sheron</td>
<td>Male</td>
<td>2</td>
<td>2015-02-01</td>
<td>45000</td>
</tr>
<tr>
<td>4</td>
<td>Dian</td>
<td>Female</td>
<td>1</td>
<td>2018-02-01</td>
<td>50000</td>
</tr>
<tr>
<td>5</td>
<td>Rosshell</td>
<td>Male</td>
<td>3</td>
<td>2015-02-01</td>
<td>10000</td>
</tr>
<tr>
<td>6</td>
<td>Kat</td>
<td>Male</td>
<td>2</td>
<td>2014-02-01</td>
<td>37000</td>
</tr>
</tbody>
</table>

Audit Table
Alter trigger Chk_Insert_Emp
On Employee
For Insert ---- Operation type(insert/update/delete)
AS
Begin

Declare @Id int
Select @Id=EmpID from inserted

Insert into Audit_Emp (EmpID,AuditData) values
(@ID,'New Employee'+cast(@Id As nvarchar(50))+'Added on'+CAST(GETDATE () As nvarchar(50)))

END;
## Delete Operation (After)

<table>
<thead>
<tr>
<th>EmpID</th>
<th>EmpName</th>
<th>Gender</th>
<th>DepartmentId</th>
<th>JoinedDate</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>Male</td>
<td>3</td>
<td>2018-02-01</td>
<td>25000</td>
</tr>
<tr>
<td>2</td>
<td>Smith</td>
<td>Female</td>
<td>4</td>
<td>2017-05-01</td>
<td>35000</td>
</tr>
<tr>
<td>3</td>
<td>Sheron</td>
<td>Male</td>
<td>2</td>
<td>2015-02-01</td>
<td>45000</td>
</tr>
<tr>
<td>4</td>
<td>Dian</td>
<td>Female</td>
<td>1</td>
<td>2018-02-01</td>
<td>50000</td>
</tr>
<tr>
<td>5</td>
<td>Roshell</td>
<td>Male</td>
<td>3</td>
<td>2015-02-01</td>
<td>10000</td>
</tr>
<tr>
<td>6</td>
<td>Kat</td>
<td>Male</td>
<td>2</td>
<td>2014-02-01</td>
<td>37000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EmpID</th>
<th>AuditData</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Existing Employee deleted on Feb 23 2018 7:40PM</td>
</tr>
</tbody>
</table>
Create trigger Chk_Delete_Emp
On Employee
For Delete ----Operation type(insert/update/delete)
AS
Begin

Declare @Id int
Select @Id=EmpID from deleted

Insert into Audit_Emp (EmpID,AuditData) values
(@Id,'Existing Employee ='+cast(@Id As nvarchar(50))+'Deleted on'+CAST(GETDATE () As nvarchar(50)))

END;
Instead of Triggers

INSTEAD OF Triggers fire instead of the operation that fires the trigger, so if you define an INSTEAD OF trigger on a table for the Insert/Delete /update operation, they try to delete/update rows, they will not actually get insert/deleted /updated
<table>
<thead>
<tr>
<th>EmpID</th>
<th>EmpName</th>
<th>Gender</th>
<th>DepartmentId</th>
<th>JoinedDate</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>Male</td>
<td>3</td>
<td>2018-02-01</td>
<td>25000</td>
</tr>
<tr>
<td>2</td>
<td>Smith</td>
<td>Female</td>
<td>4</td>
<td>2017-05-01</td>
<td>35000</td>
</tr>
<tr>
<td>3</td>
<td>Sheron</td>
<td>Male</td>
<td>2</td>
<td>2015-02-01</td>
<td>45000</td>
</tr>
<tr>
<td>4</td>
<td>Dian</td>
<td>Female</td>
<td>1</td>
<td>2018-02-01</td>
<td>50000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DepID</th>
<th>Depname</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IT</td>
</tr>
<tr>
<td>2</td>
<td>Payroll</td>
</tr>
</tbody>
</table>

```
Insert into Vw_Empdep values(5,'shan',33000,'HR');
```

Msg 4405, Level 16, State 1, Line 1
View or function 'Vw_Empdep' is not updatable because the modification affects multiple base tables.
Create trigger Tri_Vw_EmpDep
on Vw_EmpDep
Instead of Insert
as
Begin
Select * from inserted
Select * from deleted
End;

Insert into Vw_Empdep values(5, 'shan', 33000, 'HR');
END