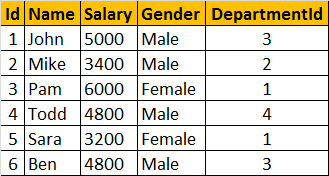
**In SQL server there are 3 types of triggers**  
1. DML triggers   
2. DDL triggers  
3. Logon trigger  
  
**We will discuss about DDL and logon triggers in a later session**. In this video, we will learn about DML triggers.   
  
   
  
   
  
   
  
**In general, a trigger is a special kind of stored procedure** that automatically executes when an event occurs in the database server.  
  
**DML stands for Data Manipulation Language.** INSERT, UPDATE, and DELETE statements are DML statements. DML triggers are fired, when ever data is modified using INSERT, UPDATE, and DELETE events.  
  
**DML triggers can be again classified into 2 types.**  
1. After triggers (Sometimes called as FOR triggers)  
2. Instead of triggers  
  
**After triggers, as the name says, fires after the triggering action**. The INSERT, UPDATE, and DELETE statements, causes an after trigger to fire after the respective statements complete execution.  
  
**On ther hand, as the name says, INSTEAD of triggers, fires instead of the triggering action**. The INSERT, UPDATE, and DELETE statements, can cause an INSTEAD OF trigger to fire INSTEAD OF the respective statement execution.  
  
**We will use tblEmployee and tblEmployeeAudit**tables for our examples  
  
**SQL Script to create tblEmployee table:**  
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)   
  
**tblEmployee**  
   
  
  
**SQL Script to create tblEmployeeAudit table:**  
CREATE TABLE tblEmployeeAudit  
(  
  Id int identity(1,1) primary key,  
  AuditData nvarchar(1000)  
)  
  
**When ever, a new Employee is added**, we want to capture the ID and the date and time, the new employee is added in tblEmployeeAudit table. The easiest way to achieve this, is by having an AFTER TRIGGER for INSERT event.  
  
**Example for AFTER TRIGGER for INSERT event on tblEmployee table:**  
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  
  
**In the trigger, we are getting the id from inserted table.** So, what is this inserted table? INSERTED table, is a special table used by DML triggers. When you add a new row into tblEmployee table, a copy of the row will also be made into inserted table, which only a trigger can access. You cannot access this table outside the context of the trigger. The structure of the inserted table will be identical to the structure of tblEmployee table.  
  
**So, now if we execute the following INSERT statement on tblEmployee.**Immediately, after inserting the row into tblEmployee table, the trigger gets fired (executed automatically), and a row into tblEmployeeAudit, is also inserted.

**Insert into tblEmployee values (7,'Tan', 2300, 'Female', 3)**  
  
**Along, the same lines, let us now capture audit information, when a row is deleted**from the table, tblEmployee.  
**Example for AFTER TRIGGER for DELETE event on 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  
  
**The only difference here is that**, we are specifying, the triggering event as **DELETE**and retrieving the deleted row ID from **DELETED** table. DELETED table, is a special table used by DML triggers. When you delete a row from tblEmployee table, a copy of the deleted row will be made available in DELETED table, which only a trigger can access. Just like INSERTED table, DELETED table cannot be accessed, outside the context of the trigger and, the structure of the DELETED table will be identical to the structure of tblEmployee table.  
  
In the next session, we will talk about AFTER trigger for UPDATE event.