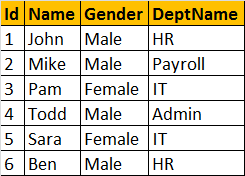
**Suggested SQL Server Videos before watching this Video**  
[Part 39 - Views](http://csharp-video-tutorials.blogspot.com/2012/09/views-in-sql-server-part-39.html)  
[Part 40 - Updateable Views](http://csharp-video-tutorials.blogspot.com/2012/09/updateable-views-part-40.html)  
[Part 43 - DML triggers](http://csharp-video-tutorials.blogspot.com/2012/09/dml-triggers-part-43.html)  
[Part 44 - DML After Update Trigger](http://csharp-video-tutorials.blogspot.com/2012/09/after-update-trigger-part-44.html)   
  
   
  
   
  
   
  
**In this video we will learn about, INSTEAD OF triggers**, specifically INSTEAD OF INSERT trigger. We know that, AFTER triggers are fired after the triggering event(INSERT, UPDATE or DELETE events), where as, INSTEAD OF triggers are fired instead of the triggering event(INSERT, UPDATE or DELETE events). In general, INSTEAD OF triggers are usually used to correctly update views that are based on multiple tables.   
  
**We will base our demos on Employee and Department** tables. So, first, let's create these 2 tables.  
  
**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)  
)  
  
**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)  
  
**Since, we now have the required tables**, let's create a view based on these tables. The view should return Employee Id, Name, Gender and DepartmentName columns. So, the view is obviously 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  
  
**When you execute**, **Select \* from vWEmployeeDetails**, the data from the view, should be as shown below  
   
  
**Now, let's try to insert a row into the view, vWEmployeeDetails**, by executing the following query. At this point, an error will be raised stating 'View or function vWEmployeeDetails is not updatable because the modification affects multiple base tables.'  
Insert into vWEmployeeDetails values(7, 'Valarie', 'Female', 'IT')

**So, inserting a row into a view that is based on multipe tables**, raises an error by default. Now, let's understand, how INSTEAD OF TRIGGERS can help us in this situation. Since, we are getting an error, when we are trying to insert a row into the view, let's create an INSTEAD OF INSERT trigger on the view **vWEmployeeDetails.**  
  
**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  
  
**Now, let's execute the insert query:**  
Insert into vWEmployeeDetails values(7, 'Valarie', 'Female', 'IT')  
  
**The instead of trigger correctly inserts**, the record into tblEmployee table. Since, we are inserting a row, the **inserted** table, contains the newly added row, where as the **deleted** table will be empty.   
  
**In the trigger, we used Raiserror() function**, to raise a custom error, when the DepartmentName provided in the insert query, doesnot exist. We are passing 3 parameters to the Raiserror() method. The first parameter is the error message, the second parameter is the severity level. Severity level 16, indicates general errors that can be corrected by the user. The final parameter is the state. We will talk about Raiserror() and exception handling in sql server, in a later video session.