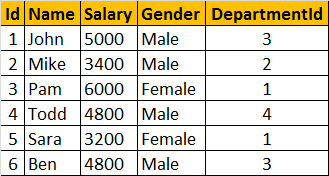
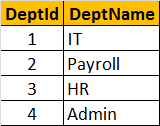
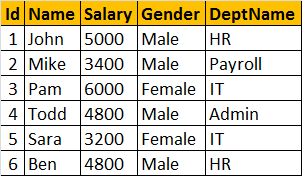
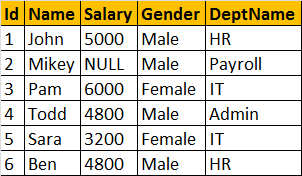
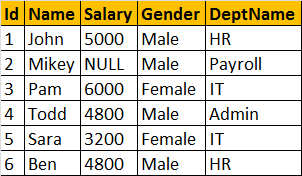
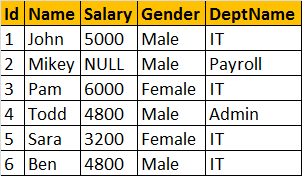
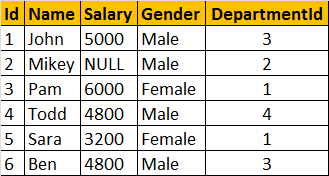
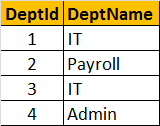
**Let's understand views with an example**. We will base all our examples on **tblEmployee** and **tblDepartment** tables.   
  
  
**SQL Script to create tblEmployee table:**  
CREATE TABLE tblEmployee  
(  
  Id int Primary Key,  
  Name nvarchar(30),  
  Salary int,  
  Gender nvarchar(10),  
  DepartmentId 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', 5000, 'Male', 3)  
Insert into tblEmployee values (2,'Mike', 3400, 'Male', 2)  
Insert into tblEmployee values (3,'Pam', 6000, 'Female', 1)  
Insert into tblEmployee values (4,'Todd', 4800, 'Male', 4)  
Insert into tblEmployee values (5,'Sara', 3200, 'Female', 1)  
Insert into tblEmployee values (6,'Ben', 4800, 'Male', 3)  
  
**At this point Employees and Departments table should look like this.**  
Employees Table:  
  
  
Departments Table:  
  
  
**Now, let's write a Query which returns the output as shown below:**  
  
  
  
**To get the expected output**, we need to join **tblEmployees** table with **tblDepartments** table. [If you are new to joins, please click here to view the video on Joins in SQL Server.](http://csharp-video-tutorials.blogspot.com/2012/08/joins-in-sql-server-part-12.html)   
Select Id, Name, Salary, Gender, DeptName  
from tblEmployee  
join tblDepartment  
on tblEmployee.DepartmentId = tblDepartment.DeptId  
  
**Now let's create a view, using the JOINS query, we have just written.**  
Create View vWEmployeesByDepartment  
as  
Select Id, Name, Salary, Gender, DeptName  
from tblEmployee  
join tblDepartment  
on tblEmployee.DepartmentId = tblDepartment.DeptId  
  
**To select data from the view**, SELECT statement can be used the way, we use it with a table.  
SELECT \* from vWEmployeesByDepartment  
  
**When this query is executed**, the database engine actually retrieves the data from the underlying base tables, **tblEmployees and tblDepartments**. The View itself, doesnot store any data by default. However, we can change this default behaviour, which we will talk about in a later session. So, this is the reason, a view is considered, as just, a stored query or a virtual table.  
  
**Advantages of using views:**  
1. Views can be used to reduce the **complexity of the database schema**, for non IT users. The sample view, **vWEmployeesByDepartment**, hides the complexity of joins. Non-IT users, finds it easy to query the view, rather than writing complex joins.  
  
2. Views can be used as a mechanism to implement **row and column level security**.  
**Row Level Security:**  
For example, I want an end user, to have access only to IT Department employees. If I grant him access to the underlying tblEmployees and tblDepartments tables, he will be able to see, every department employees. To achieve this, I can create a view, which returns only IT Department employees, and grant the user access to the view and not to the underlying table.  
  
**View that returns only IT department employees:**  
Create View vWITDepartment\_Employees  
as  
Select Id, Name, Salary, Gender, DeptName  
from tblEmployee  
join tblDepartment  
on tblEmployee.DepartmentId = tblDepartment.DeptId  
where tblDepartment.DeptName = 'IT'  
  
**Column Level Security:**  
Salary is confidential information and I want to prevent access to that column. To achieve this, we can create a view, which excludes the Salary column, and then grant the end user access to this views, rather than the base tables.  
  
**View that returns all columns except Salary column:**  
Create View vWEmployeesNonConfidentialData  
as  
Select Id, Name, Gender, DeptName  
from tblEmployee  
join tblDepartment  
on tblEmployee.DepartmentId = tblDepartment.DeptId  
  
3. Views can be used to present **only aggregated data** and **hide detailed data**.  
  
**View that returns summarized data**, Total number of employees by Department.  
Create View vWEmployeesCountByDepartment  
as  
Select DeptName, COUNT(Id) as TotalEmployees  
from tblEmployee  
join tblDepartment  
on tblEmployee.DepartmentId = tblDepartment.DeptId  
Group By DeptName  
  
To look at view definition - sp\_helptext vWName  
To modify a view - ALTER VIEW statement   
To Drop a view - DROP VIEW vWName

### Updateable Views

**Create Table tblEmployee Script:**  
CREATE TABLE tblEmployee  
(  
  Id int Primary Key,  
  Name nvarchar(30),  
  Salary int,  
  Gender nvarchar(10),  
  DepartmentId int  
)  
  
**Script to insert data:**  
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)  
Insert into tblEmployee values (4,'Todd', 4800, 'Male', 4)  
Insert into tblEmployee values (5,'Sara', 3200, 'Female', 1)  
Insert into tblEmployee values (6,'Ben', 4800, 'Male', 3)  
  
**Let's create a view**, which returns all the columns from the tblEmployees table, except Salary column.  
Create view vWEmployeesDataExceptSalary  
as  
Select Id, Name, Gender, DepartmentId  
from tblEmployee  
  
**Select data from the view**: A view does not store any data. So, when this query is executed, the database engine actually retrieves data, from the underlying tblEmployee base table.  
Select \* from vWEmployeesDataExceptSalary  
  
**Is it possible to Insert, Update and delete rows**, from the underlying tblEmployees table, using view vWEmployeesDataExceptSalary?  
**Yes**, SQL server views are updateable.  
  
**The following query updates, Name column from Mike to Mikey**. Though, we are updating the view, SQL server, correctly updates the base table tblEmployee. To verify, execute, SELECT statement, on tblEmployee table.  
Update vWEmployeesDataExceptSalary   
Set Name = 'Mikey' Where Id = 2  
  
**Along the same lines**, it is also possible to insert and delete rows from the base table using views.  
Delete from vWEmployeesDataExceptSalary where Id = 2  
Insert into vWEmployeesDataExceptSalary values (2, 'Mikey', 'Male', 2)  
  
**Now, let us see, what happens if our view is based on multiple base tables**. For this purpose, let's create tblDepartment table and populate with some sample data.  
**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')  
  
**Create a view which joins tblEmployee and tblDepartment tables**, and return the result as shown below.  
  
  
**View that joins tblEmployee and tblDepartment**  
Create view vwEmployeeDetailsByDepartment  
as  
Select Id, Name, Salary, Gender, DeptName  
from tblEmployee  
join tblDepartment  
on tblEmployee.DepartmentId = tblDepartment.DeptId  
  
**Select Data from view vwEmployeeDetailsByDepartment**  
Select \* from vwEmployeeDetailsByDepartment  
  
**vwEmployeeDetailsByDepartment Data:**  
  
  
  
**Now, let's update, John's department, from HR to IT**. At the moment, there are 2 employees (Ben, and John) in the HR department.  
Update vwEmployeeDetailsByDepartment   
set DeptName='IT' where Name = 'John'  
  
**Now, Select data from the view vwEmployeeDetailsByDepartment:**  
  
  
**Notice, that Ben's department is also changed to IT**. To understand the reasons for incorrect UPDATE, select Data from tblDepartment and tblEmployee base tables.  
  
**tblEmployee Table**  
  
  
**tblDepartment**  
  
  
  
**The UPDATE statement, updated DeptName from HR to IT in tblDepartment table**, instead of upadting **DepartmentId** column in **tblEmployee** table. So, the conclusion - If a view is based on multiple tables, and if you update the view, it may not update the underlying base tables correctly. To correctly update a view, that is based on multiple table, INSTEAD OF triggers are used.

**Indexed views in sql server**