



Instructor:

- Mr. Nazeef Ul Haq (Lab)

Learning Objectives:

- Understanding of Stored Procedure and Views.

Helping Material:

Views

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

Types of views in SQL Server

There are the following two types of views:

1. User-Defined Views
2. System-Defined Views

First we discuss the User-Defined Views.

User Define Views: First we create two tables. First create a Employee_Details table for the basic info of an employee.

1. `CREATE TABLE [dbo].[Employee_Details]`
2. `(`
3. `[Emp_Id] [int] IDENTITY(1,1) NOT NULL,`
4. `[Emp_Name] [nvarchar](50) NOT NULL,`
5. `[Emp_City] [nvarchar](50) NOT NULL,`
6. `[Emp_Salary] [int] NOT NULL,`
7. `CONSTRAINT [PK_Employee_Details] PRIMARY KEY CLUSTERED`
8. `(`
9. `[Emp_Id] ASC`
10. `)`
11. `WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]`
12. `)`
13. `ON [PRIMARY]`
- 14.
15. `GO`

Now insert some data into the table as in the following:

1. `Insert Into Employee_Details Values('Pankaj','Alwar',25000)`
2. `Insert Into Employee_Details Values('Rahul','Jaipur',26000)`
3. `Insert Into Employee_Details Values('Rajan','Delhi',27000)`
4. `Insert Into Employee_Details Values('Sandeep','Alwar',28000)`
5. `Insert Into Employee_Details Values('Sanjeev','Jaipur',32000)`
6. `Insert Into Employee_Details Values('Narendra','Alwar',34000)`
7. `Insert Into Employee_Details Values('Neeraj','Delhi',29000)`
8. `Insert Into Employee_Details Values('Div','Jaipur',25000)`
9. `Insert Into Employee_Details Values('Tanuj','Alwar',22000)`
10. `Insert Into Employee_Details Values('Nitin','Jaipur',20000)`

Now the table Employee_Detail will look as in the following.

1. `Select * from Employee_Details`

Results		Messages		
	Emp_Id	Emp_Name	Emp_City	Emp_Salary
1	1	Pankaj	Alwar	25000
2	2	Rahul	Jaipur	26000
3	3	Rajan	Delhi	27000
4	4	Sandeep	Alwar	28000
5	5	Sanjeev	Jaipur	32000
6	6	Narendra	Alwar	34000
7	7	Neeraj	Delhi	29000
8	8	Div	Jaipur	25000
9	9	Tanuj	Alwar	22000
10	10	Nitin	Jaipur	20000

We create another table named **Employee_Contact**.

1. `CREATE TABLE [dbo].[Employee_Contact]`
2. `(`
3. `[Emp_Id] [int] NOT NULL,`
4. `[MobileNo] [nvarchar](50) NOT NULL`
5. `) ON [PRIMARY]`
- 6.
7. `GO`
- 8.
9. `ALTER TABLE [dbo].[Employee_Contact] WITH CHECK ADD CONSTRAINT [FK_Employee_Contact_Employee_Details] FOREIGN KEY([Emp_Id])`
10. `REFERENCES [dbo].[Employee_Details] ([Emp_Id])`
11. `GO`
- 12.
13. `ALTER TABLE [dbo].[Employee_Contact] CHECK CONSTRAINT [FK_Employee_Contact_Employee_Details]`
14. `GO`

Insert some values into the **Employee_Contact** table as in the following:

1. `Insert Into Employee_Contact Values(1,'9813220191')`

2. `Insert Into Employee_Contact Values(2,'9813220192')`
3. `Insert Into Employee_Contact Values(3,'9813220193')`
4. `Insert Into Employee_Contact Values(4,'9813220194')`
5. `Insert Into Employee_Contact Values(5,'9813220195')`
6. `Insert Into Employee_Contact Values(6,'9813220196')`
7. `Insert Into Employee_Contact Values(7,'9813220197')`
8. `Insert Into Employee_Contact Values(8,'9813220198')`
9. `Insert Into Employee_Contact Values(9,'9813220199')`
10. `Insert Into Employee_Contact Values(10,'9813220135')`

Now the table Employee_Contact will look as in the following:

1. `select * from Employee_Contact`

Results		Messages
	Emp_Id	MobileNo
1	1	9813220191
2	2	9813220192
3	3	9813220193
4	4	9813220194
5	5	9813220195
6	6	9813220196
7	7	9813220197
8	8	9813220198
9	9	9813220199
10	10	9813220135

Now we start a detailed discussion of User Defined Views (UDVs).

Create SQL VIEW in SQL Server

1. `CREATE VIEW view_name AS`
2. `SELECT columns`
3. `FROM tables`
4. `WHERE conditions;`

Let us create some views.

Method 1: We can select all columns of a table. The following example demonstrates that:

1. `Create View Employee_View1`
2. `as`
3. `select * from Employee_Details`

Method 2: We can select specific columns of a table. The following example demonstrates that:

1. `Create View Employee_View2`
2. `as`
3. `select Emp_Id,Emp_Name,Emp_City from Employee_Details`

Method 3: We can select columns from a table with specific conditions. The following example demonstrates that:

1. `Create View Employee_View3`
2. `as`
3. `select * from Employee_Details where Emp_Id>3`

Method 4: We can create a view that will hold the columns of different tables. The following example demonstrates that:

1. **Create View** Employee_View4
2. **as**
3. **select** Employee_Details.Emp_Id,Employee_Details.Emp_Name,Employee_Details.Emp_Salary,Employee_Contact.MobileNo **from** Employee_Details
4. **Left Outer Join**
5. **Employee_Contact**
6. **on**
7. **Employee_Details .Emp_Id= Employee_Contact.Emp_Id**
8. **Where** Employee_Details.Emp_Id>2

Retrieve Data From View in SQL Server

This SQL CREATE VIEW example would create a virtual table based on the result set of the select statement. Now we can retrieve data from a view as follows:

1. **Select *** **from** Employee_View4
- 2.
3. **Select** Emp_Id,Emp_Name,Emp_Salary **from** Employee_View4

The preceding query shows that we can select all the columns or some specific columns from a view.

Dropping a View in SQL Server

We can use the Drop command to drop a view. For example, to drop the view Employee_View3, we can use the following statement.

1. **Drop View** Employee_View1

Renaming the View in SQL Server

We can use the sp_rename system procedure to rename a view. The syntax of the sp_rename command is given below:

1. **Sp_Rename** OldViewName , NewViewName

Example

1. **Sp_Rename** Employee_View4 , Employee_ViewNew

In the preceding example, we rename the view **Employee_View1** as **Employee_ViewNew**.

Getting Information about a view: We can retrieve all the information of a view using the Sp_Helptext system Stored Procedure. Let us see an example.

1. **Sp_Helptext** Employee_View4

Output

Results		Messages
Text		
1	Create View Employee_View4	
2	as	
3	select Employee_Details.Emp_Id,Employee_Details.E...	
4	Left Outer Join	
5	Employee_Contact	
6	on	
7	Employee_Details .Emp_Id= Employee_Contact.Emp_Id	
8	Where Employee_Details.Emp_Id>2	

Alter View in SQL Server

We can alter the schema or structure of a view. In other words, we can add or remove some columns or change some conditions that are applied in a predefined view. Let us see an example.

1. **Alter View** Employee_View4
2. **as**
3. **select** Employee_Details.Emp_Id,Employee_Details.Emp_Name,Employee_Details.Emp_Salary,Employee_Contact.MobileNo **from** Employee_Details
4. **Left Outer Join**
5. **Employee_Contact**
6. **on**
7. **Employee_Details .Emp_Id= Employee_Contact.Emp_Id**
8. **Where** Employee_Details.Emp_Id>5 and Employee_Details.Emp_City='Alwar'

Refreshing a View in SQL Server

Let us consider the scenario now by adding a new column to the table Employee_Details and examine the effect. We will first create a view.

1. **Create View** Employee_View1
2. **as**
3. **Select *** **from** Employee_Details
- 4.
5. Now **add** a **column** in Employee_Details **table**
- 6.
7. **Alter Table** Employee_Details **Add** MY_sal nvarchar(50)


Now retrieve the data from the table and view and you will receive the following output:

1. **Select *** **from** Employee_Details
2. **Select *** **from** Employee_View1

Output

Results					
Messages					
	Emp_Id	Emp_Name	Emp_City	Emp_Salary	MY_sal
1	1	Pankaj	Alwar	25000	NULL
2	2	Rahul	Jaipur	26000	NULL
3	3	Rajan	Delhi	27000	NULL
4	4	Sandeep	Alwar	28000	NULL
5	5	Sanjeev	Jaipur	32000	NULL
6	6	Narendra	Alwar	34000	NULL

	Emp_Id	Emp_Name	Emp_City	Emp_Salary
1	1	Pankaj	Alwar	25000
2	2	Rahul	Jaipur	26000
3	3	Rajan	Delhi	27000
4	4	Sandeep	Alwar	28000
5	5	Sanjeev	Jaipur	32000
6	6	Narendra	Alwar	34000

 Query executed successfully.

We don't get the results we expected because the schema of the view is already defined. So when we add a new column into the table it will not change the schema of the view and the view will contain the previous schema. For removing this problem we use the system-defined Stored Procedure `sp_refreshview`.

sp_refreshview is a system-level Stored Procedure that refreshes the metadata of any view once you edit the schema of the table. Let's execute the following:

1. `Exec sp_refreshview Employee_View1`
2. `Select * from Employee_Details`
3. `Select * from Employee_View1`

Output

Results					
Messages					
	Emp_Id	Emp_Name	Emp_City	Emp_Salary	MY_sal
1	1	Pankaj	Alwar	25000	NULL
2	2	Rahul	Jaipur	26000	NULL
3	3	Rajan	Delhi	27000	NULL
4	4	Sandeep	Alwar	28000	NULL
5	5	Sanjeev	Jaipur	32000	NULL

	Emp_Id	Emp_Name	Emp_City	Emp_Salary	MY_sal
1	1	Pankaj	Alwar	25000	NULL
2	2	Rahul	Jaipur	26000	NULL
3	3	Rajan	Delhi	27000	NULL
4	4	Sandeep	Alwar	28000	NULL
5	5	Sanjeev	Jaipur	32000	NULL
6	6	Narendra	Alwar	34000	NULL
7	7	Neeraj	Delhi	29000	NULL

SchemaBinding a VIEW

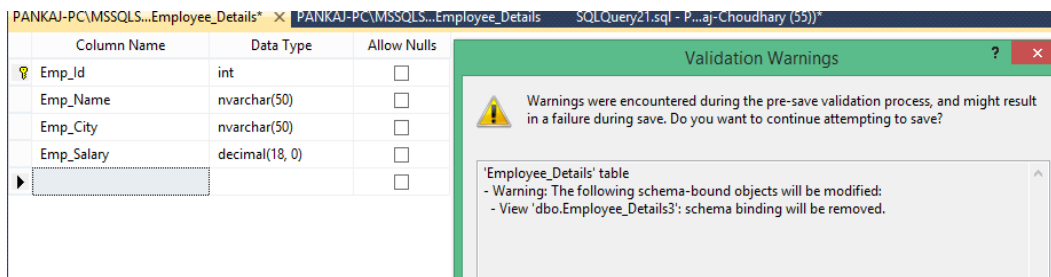
In the previous example, we saw that if we add a new column into the table then we must refresh the view.

Such a way if we change the data type of any column in a table then we should refresh the view. If we want to prevent any type of change in a base table then we can use the concept of SCHEMABINDING. It will lock the tables being referred to by the view and restrict all kinds of changes that may change the table schema (no Alter command).

We can't specify "Select * from tablename" with the query. We need to specify all the column names for reference.

1. **Create View** Employee_Details3
2. **with** SCHEMABINDING
3. **as**
4. **select** Emp_Id,Emp_Name,Emp_Salary,Emp_City **from** DBO.Employee_Details

In the preceding example, we create a view using Schemabinding. Now we try to change the datatype of Emp_Salary from int to Decimal in the Base Table.



We find that we cannot change the data type because we used the SCHEMABINDING that prevents any type of change in the base table.

Encrypt a view in SQL Server

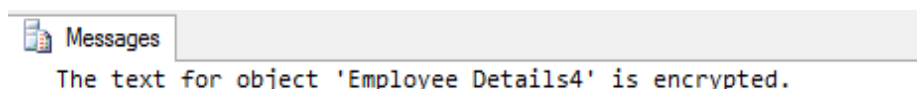
The "WITH ENCRYPTION" option can encrypt any views. That means it will not be visible via SP_HELPTEXT. This option encrypts the definition. This option encrypts the definition of the view. Users will not be able to see the definition of the view after it is created. This is the main advantage of the view where we can make it secure.

1. **Create View** Employee_Details4
2. **with** Encryption
3. **as**
4. **select** Emp_Id,Emp_Name,Emp_Salary,Emp_City **from** DBO.Employee_Details

Now we try to retrieve the definition of the view.

1. **Exec** sp_helptext 'Employee_Details4'

Output



Check Option: The use of the Check Option in a view is to ensure that all the Update and Insert commands must satisfy the condition in the view definition.

Let us see an Example:

```
1. GO
2.
3. Create view [dbo].[Employee_Details7]
4. as
5. select * from Employee_Details
6. where Emp_Salary>30000
7.
8. GO
```

In the preceding example, we create a view that contains all the data for which Emp_Salry > 30000 but we can insert the data for a salary less than 30000 as follows.

```
1. Insert Into Employee_Details7 values ('ram','mumbai',25000,'Pan')
```

For preventing this problem we can use the Check Option property such as:

```
1. GO
2.
3. Create view [dbo].[Employee_Details7]
4. as
5. select * from Employee_Details
6. where Emp_Salary>30000
7. with Check Option
8. GO
```

Now if we try to execute the preceding query then it will throw an error such as:

```
1. Insert Into Employee_Details7 values ('ram','mumbai',25000,'Pan')
```

Output

Msg 550, Level 16, State 1, Line 1

The attempted insert or update failed because the target view either specifies WITH CHECK OPTION or spans a view that specifies WITH CHECK OPTION and one or more rows resulting from the operation did not qualify under the CHECK OPTION constraint.

The statement has been terminated.

DML Query In View

In a view we can implement many types of DML query like insert, update and delete. But for a successful implementation of a DML query we should use some conditions like:

1. View should not contain multiple tables
2. View should not contain set function.
3. View should not use the Distinct keyword
4. View should not contain Group By, having clauses
5. View should not contain Sub query
6. View should not use Set Operators
7. All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

If we use the preceding conditions then we can implement a DML Query in the view without any problem. Let us see an example.

1. `select * from Employee_Details7`

Output

	Emp_Id	Emp_Name	Emp_City	Emp_Salary
1	5	Sanjeev	Jaipur	32000
2	6	Narendra	Alwar	34000

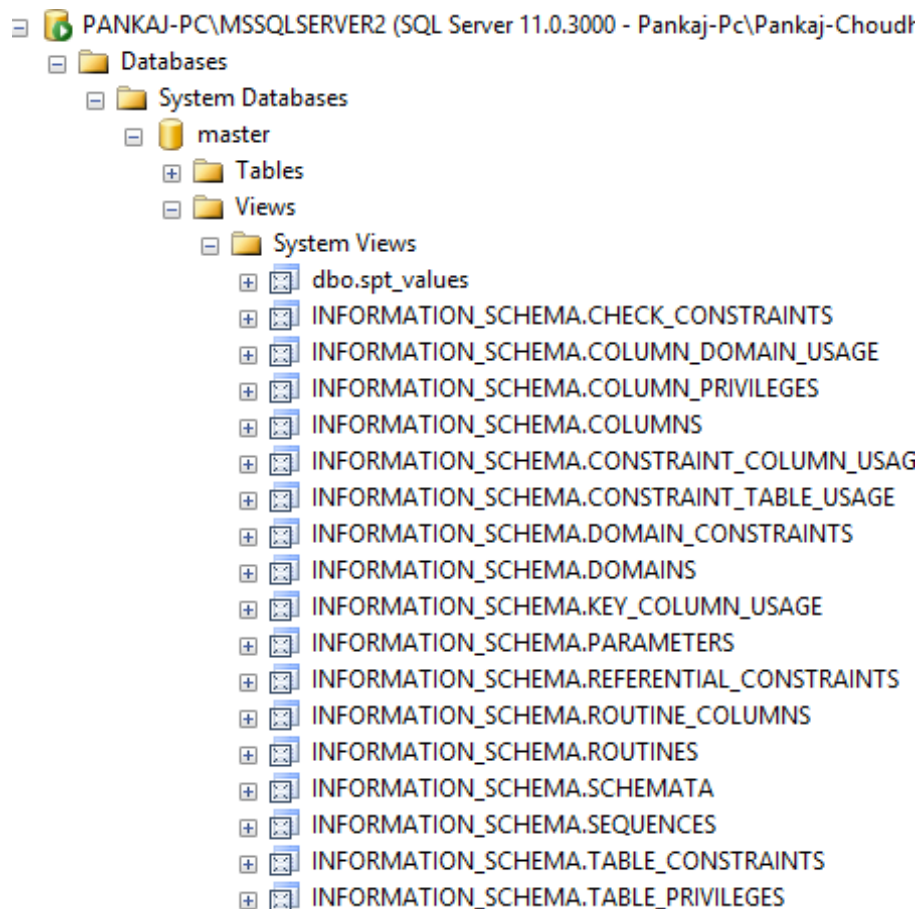
Now we implement a DML Query as in the following:

1. `Insert Into Employee_Details7 values ('ram','mumbai',35000)`
2. `Update Employee_Details7 set Emp_Name='Raju' where Emp_id=5`
3. `delete from Employee_Details7 where Emp_Id=6`
4. `select * from Employee_Details7`

Output

Results		Messages		
	Emp_Id	Emp_Name	Emp_City	Emp_Salary
1	5	Raju	Jaipur	32000
2	16	ram	mumbai	35000

System Define Views: SQL Server also contains various predefined databases like Tempdb, Master, temp. Each database has their own properties and responsibility. Master data is a template database for all other user-defined databases. A Master database contains many Predefine_View that work as templates for other databases and tables. Master databases contain nearly 230 predefined views.



These predefined views are very useful to us. Mainly we divide system views into the following two parts.

1. Information Schema
2. Catalog View

Information schema: There are nearly 21 Information Schemas in the System. These are used for displaying the most physical information of a database, such as table and columns. An Information Schema starts from INFORMATION_SCHEMA.[View Name]. Let us see an example.

1. `select * from INFORMATION_SCHEMA.VIEW_TABLE_USAGE`
2. `where TABLE_NAME='Employee_Details'`

Output

Results		Messages				
	VIEW_CATALOG	VIEW_SCHEMA	VIEW_NAME	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME
1	Views_Practice	dbo	Employee_Details7	Views_Practice	dbo	Employee_Details

This Information_Schema returns the details of all the views used by table **Employee_Details**.

1. `select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS`
2. `where TABLE_NAME='Employee_Details'`

Output

Results		Messages							
	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE	IS_DEFERRABLE	INITIALLY_DEFERRED
1	Views_Practice	dbo	PK_Employee_Details	Views_Practice	dbo	Employee_Details	PRIMARY KEY	NO	NO

This **Information_Schema** returns the information about the constraints of a table.

Catalog View: Catalog Views are categorized into various groups also. These are used to show the self-describing information of a database. These start with “**sys**”.

1. `select * from sys.all_views`

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date	is_ms_shipped	is_published	is_schema_published	is_replicated	has_permissions
1	TABLE_PRIVILEGES	-1072372588	NULL	3	0	V	VIEW	2012-02-10 20:33:15.110	2012-02-10 20:33:15.160	1	0	0	0	0
2	dm_os_hosts	-1061705188	NULL	4	0	V	VIEW	2012-02-10 20:26:04.163	2012-02-10 20:26:04.267	1	0	0	0	0
3	dm_os_memory_brokers	-1055124494	NULL	4	0	V	VIEW	2012-02-10 20:26:12.260	2012-02-10 20:26:12.347	1	0	0	0	0
4	openkeys	-1047118026	NULL	4	0	V	VIEW	2012-02-10 20:18:21.130	2012-02-10 20:18:21.240	1	0	0	0	0
5	dm_os_memory_allocations	-1045383193	NULL	4	0	V	VIEW	2012-02-10 20:26:05.267	2012-02-10 20:26:05.357	1	0	0	0	0
6	dm_os_memory_operations	-1040075447	NULL	4	0	V	VIEW	2012-02-10 20:26:28.203	2012-02-10 20:26:28.297	1	0	0	0	0
7	dm_os_session_object_columns	-1039848107	NULL	4	0	V	VIEW	2012-02-10 20:26:18.587	2012-02-10 20:26:18.680	1	0	0	0	0
8	dm_os_loaded_modules	-1035533931	NULL	4	0	V	VIEW	2012-02-10 20:26:04.937	2012-02-10 20:26:05.030	1	0	0	0	0

This query provides information to all types of views using a database.

1. `select * from sys.databases`

	name	database_id	source_database_id	owner_sid	create_date
1	master	1	NULL	0x01	2003-04-08 09:13:36.390
2	tempdb	2	NULL	0x01	2015-03-26 16:35:33.413
3	model	3	NULL	0x01	2003-04-08 09:13:36.390
4	msdb	4	NULL	0x01	2012-02-10 21:02:17.770
5	ReportServer\$MSSQLSERVER2	5	NULL	0x01050000000000000515000000487074533D22CFEF911FFB...	2015-02-16 16:10:39.220
6	ReportServer\$MSSQLSERVER2TempDB	6	NULL	0x01050000000000000515000000487074533D22CFEF911FFB...	2015-02-16 16:10:40.330
7	practice	7	NULL	0x01050000000000000515000000487074533D22CFEF911FFB...	2015-02-16 17:14:34.260
8	portal	8	NULL	0x01050000000000000515000000487074533D22CFEF911FFB...	2015-03-04 11:24:19.927

This query will provide the information about all the databases defined by the system, including user-defined and system-defined database.

What to Submit:

You are required to submit the following files.

- Run all the things in any table of Northwind database and create a pdf file similar like this.