Views in SQL Server

by Steve Manik
17 November 2005

View

A view is a virtual table that consists of columns from one or more tables. Though it is similar to a table, it is stored in the database. It is a query stored as an object. Hence, a view is an object that derives its data from one or more tables. These tables are referred to as base or underlying tables.

Once you have defined a view, you can reference it like any other table in a database.

A view serves as a security mechanism. This ensures that users are able to retrieve and modify only the data seen by them. Users cannot see or access the remaining data in the underlying tables. A view also serves as a mechanism to simplify query execution. Complex queries can be stored in the form as a view, and data from the view can be extracted using simple queries.

Example

Consider the Publishers table below. If you want users to see only two columns in the table, you can create a view called vwPublishers that will refer to the Publishers table and the two columns required. You can grant Permissions to users to use the view and revoke Permissions from the base Publishers table. This way, users will be able to view only the two columns referred to by the view. They will not be able to query on the Publishers table.

Publishers

PubId	PubName	City	State	Country
0736	New Moon Books	Boston	MA	USA
0877	Binnet & Hardly	Washington	DC	USA
1389	Algodata Infosystems	Berkeley	CA	USA
1622	Five Lakes Publishing	Chicago	IL	USA

VW Publishers

PubId	PubName
0736	New Moon Books
0877	Binnet & Hardly
1389	Algodata Infosystems
1622	Five Lakes Publishing

Views ensure the security of data by restricting access to the following data:

- Specific rows of the tables.
- Specific columns of the tables.
- Specific rows and columns of the tables.
- Rows fetched by using joins.
- Statistical summary of data in a given tables.
- Subsets of another view or a subset of views and tables.

Some common examples of views are:

- A subset of rows or columns of a base table.
- A union of two or more tables.
- A join of two or more tables.
- A statistical summary of base tables.
- A subset of another view, or some combination of views and base table.

Creating Views

A view can be created by using the CREATE VIEW statement.

Syntax

```
CREATE VIEW view_name
[(column_name[,column_name]….)]
[WITH ENCRYPTION]
AS select_statement [WITH CHECK OPTION]
```

Where:

view_name specifies the name of the view and must follow the rules for identifiers.

column_name specifies the name of the column to be used in view. If the column_name option is not specified, then the view is created with the same columns as specified in the select_statement.

WITH ENCRYPTION encrypts the text for the view in the syscomments table.

AS specifies the actions that will be performed by the view.

select_statement specifies the SELECT Statement that defines a view. The view may use the data contained in other views and tables.

WITH CHECK OPTION forces the data modification statements to fulfill the criteria given in the SELECT statement defining the view. It also ensures that the data is visible after the modifications are made permanent.

The restrictions imposed on views are as follows:

- A view can be created only in the current database.
- The name of a view must follow the rules for identifiers and must not be the same as that of the base table.
- A view can be created only if there is a SELECT permission on its base table.
- A SELECT INTO statement cannot be used in view declaration statement.
- A trigger or an index cannot be defined on a view.
- The CREATE VIEW statement cannot be combined with other SQL statements in a single batch.

Example

```
CREATE VIEW vwCustomer
AS
SELECT CustomerId, Company Name, Phone
FROM Customers
```

Creates a view called vwCustomer. Note that the view is a query stored as an object. The data is derived from the columns of the base table Customers.

You use the view by querying the view like a table.

```
SELECT *FROM vwCUSTOMER
```

The output of the SELECT statement is:

CustomerId	Company Name	Phone
ALFKI	Alfreds Futterkiste	030-0074321
ANTON	Antonio Moreno Taqueria	(5)555-3932

(91 rows affected)

Getting Information on a View

SQL Server stores information on the view in the following system tables:

- SYSOBJECTS — stores the name of the view.
- SYSCOLUMNS — stores the names of the columns defined in the view.
- SYSDEPENDS — stores information on the view dependencies.
- SYSCOMMENTS — stores the text of the view definition.

There are also certain system-stored procedures that help retrieve information on views. The sp_help system-stored procedure displays view-related information. It displays the view definition, provided the name of the view is given as its parameter.

Example

```
Sp_helptext vwCustomer
```

Displays the definition of the vwCustomer view.

Note

If a view is created with the WITH ENCRYPTION option, it cannot view the sp_helptext system-stored procedure.

Altering Views

You can modify a view without dropping it. This ensures that the permission on the view is also not lost. You can modify a view without affecting its dependent objects, such as triggers and stored procedures.

You modify a view using the ALTER VIEW statement.

Syntax

```
ALTER VIEW view _name [column_ name)]
[WITH ENCRYPTION]
AS select_statement
[WITH CHECK OPTION]
```

Where:

view_name is the view to be altered.

column_name specifies the name of the column to be used in the view. If the column_name option is not specified then the view is created with the same column as specified in the select_statement.

WITH ENCRYPTION encrypts the text for the view in the SYSCOMMENTS tables.

AS specifies the action that will be performed by the view.

select_statement specifies the SELECT statement that defines a view. The view could use other views and tables.

WITH CHECK OPTION forces the data modification statements to follow the criteria given in the SELECT statement definition.

Example

```
ALTER VIEW vwCustomer
AS
SELECT CustomerId, Company Name, Phone, Fax
FROM Customers
```

Alters the vwCustomers view to add the Fax column of the Customers table.

When you query on the view using

```
SELECT *FROM vwCustomer
```

You see the following output:

CutomerId	Company Name	Phone, Fax
ALFKI	Alfreds Futterkiste	030-0074321, 030-0076545
ANTON	Antonio Moreno Taqueria	(5)555-3932 NULL
AROUT	Around the Horn	(171)555-7788, (171)555-6750

(991 rows affected)

Note

If you define a view, a SELECT* statement, and then the structure of the underlying tables by adding columns, the new columns do not appear in the view. When all columns are selected in a CREATE VIEW statement, the column list is interpreted only when you first create a view. To see the new columns in the view, you must alter the view.

Dropping Views

You can drop a view from a database by using the DROP VIEW statement. When a view is dropped, it has no effect on the underlying tables. Dropping a view removes its definition and all the permissions assigned to it. Furthermore, if you query any views that reference a dropped view, you receive an error message. However, dropping a table that references a view does not drop the view automatically. You must drop it explicitly.

Syntax

```
DROP VIEW view_name
```

Where:

view_name is the name of the view to be dropped.

You can drop multiple views with a single DROP VIEW statement. The names of the views that need to be dropped are separated by commas in the DROP VIEW statement.

Note

If a view is defined with the select *statement and the base table is altered by adding a few columns, the new columns do not get included in the view. The asterisk(*) symbol is interpreted and expanded only when the view is created. In order to access the new columns via the view, it is necessary to drop the view and recreate it.

If the underlying object is dropped, the views based on the object becomes inactive and any attempt to query data from the view will result in an error message.

Renaming Views

You can rename a view without having to drop it. This ensures that the permissions on the view are not lost.

The guidelines for renaming a view are as follows:

- The view must be in the current database.
- The new name for the view must be followed by the rules for identifiers.
- A view can be renamed only by its owner.
- A view can also be renamed by the owner of the database.
- A view can be renamed by using the sp_rename system stored procedure.

Syntax

```
Sp_rename_old_viewname, new_viewname
```

Where:

old_viewname is the view that needs to be renamed.

new_viewname is the new name of the view.

Example

```
Sp_rename vwCutomers vwCustomerDetails
```

Renames vwCutomers to vwCustomerDetails.

Manipulating Data Using Views

You can modify data by using a view in only one of its base tables even though a view may be derived from multiple underlying tables. For example, a view vwNew that is derived from two tables, table and table2, can be used to modify either table or table2 in a single statement. A single data modification statement that affected both the underlying tables is not permitted.

You cannot modify the following of Columns using a view:

- Columns that are based on computed values.
- Columns that are based on built_in_function like numeric and string functions.
- Columns that are based on row aggregate functions.

Consider a situation in which a table contains a few Columns that have been defined as NOT NULL, but the view derived from the table does not contain any of these NOT NULL columns. During an INSERT operation, there may be situation where:

All the NOT NULL columns in the base tables are defined with default values.

In the first case, the INSERT operation will be successful because the default values are supplied for the NOT NULL columns. In the second case, the INSERT operation will fail because default values are not supplied for the NOT NULL columns.