**Using a simple CREATE VIEW**

The following example creates a view by using a simple SELECT statement. A simple view is helpful when a combination of columns is queried frequently. The data from this view comes from the HumanResources.Employee and Person.Contact tables of the AdventureWorks database. The data provides name and hire date information for the employees of Adventure Works Cycles. The view could be created for the person in charge of tracking work anniversaries but without giving this person access to all the data in these tables.

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/copycode.gifCopy Code |
| USE AdventureWorks ;  GO  IF OBJECT\_ID ('hiredate\_view', 'V') IS NOT NULL  DROP VIEW hiredate\_view ;  GO  CREATE VIEW hiredate\_view  AS  SELECT c.FirstName, c.LastName, e.EmployeeID, e.HireDate  FROM HumanResources.Employee e JOIN Person.Contact c on e.ContactID = c.ContactID ;  GO | |

**B. Using WITH ENCRYPTION**

The following example uses the WITH ENCRYPTION option and shows computed columns, renamed columns, and multiple columns.

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/copycode.gifCopy Code |
| USE AdventureWorks ;  GO  IF OBJECT\_ID ('Purchasing.PurchaseOrderReject', 'V') IS NOT NULL  DROP VIEW Purchasing.PurchaseOrderReject ;  GO  CREATE VIEW Purchasing.PurchaseOrderReject  WITH ENCRYPTION  AS  SELECT PurchaseOrderID, ReceivedQty, RejectedQty,  RejectedQty / ReceivedQty AS RejectRatio, DueDate  FROM Purchasing.PurchaseOrderDetail  WHERE RejectedQty / ReceivedQty > 0  AND DueDate > CONVERT(DATETIME,'20010630',101) ;  GO | |

**C. Using WITH CHECK OPTION**

The following example shows a view named SeattleOnly that references five tables and allows for data modifications to apply only to employees who live in Seattle.

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/copycode.gifCopy Code |
| USE AdventureWorks ;  GO  IF OBJECT\_ID ('dbo.SeattleOnly', 'V') IS NOT NULL  DROP VIEW dbo.SeattleOnly ;  GO  CREATE VIEW dbo.SeattleOnly  AS  SELECT c.LastName, c.FirstName, a.City, s.StateProvinceCode  FROM Person.Contact AS c  JOIN HumanResources.Employee AS e ON c.ContactID = e.ContactID  JOIN HumanResources.EmployeeAddress AS ea ON e.EmployeeID = ea.EmployeeID  JOIN Person.Address AS a ON ea.AddressID = a.AddressID  JOIN Person.StateProvince AS s ON a.StateProvinceID = s.StateProvinceID  WHERE a.City = 'Seattle'  WITH CHECK OPTION ;  GO | |

**D. Using built-in functions within a view**

The following example shows a view definition that includes a built-in function. When you use functions, you must specify a column name for the derived column.

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/copycode.gifCopy Code |
| USE AdventureWorks ;  GO  IF OBJECT\_ID ('Sales.SalesPersonPerform', 'V') IS NOT NULL  DROP VIEW Sales.SalesPersonPerform ;  GO  CREATE VIEW Sales.SalesPersonPerform  AS  SELECT TOP 100 SalesPersonID, SUM(TotalDue) AS TotalSales  FROM Sales.SalesOrderHeader  WHERE OrderDate > CONVERT(DATETIME,'20001231',101)  GROUP BY SalesPersonID;  GO | |

**E. Using partitioned data**

The following example uses tables named SUPPLY1, SUPPLY2, SUPPLY3, and SUPPLY4. These tables correspond to the supplier tables from four offices, located in different countries/regions.

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/copycode.gifCopy Code |
| --Create the tables and insert the values.  CREATE TABLE dbo.SUPPLY1 (  supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 1 and 150),  supplier CHAR(50)  );  CREATE TABLE dbo.SUPPLY2 (  supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 151 and 300),  supplier CHAR(50)  );  CREATE TABLE dbo.SUPPLY3 (  supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 301 and 450),  supplier CHAR(50)  );  CREATE TABLE dbo.SUPPLY4 (  supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 451 and 600),  supplier CHAR(50)  );  GO  INSERT dbo.SUPPLY1 VALUES ('1', 'CaliforniaCorp');  INSERT dbo.SUPPLY1 VALUES ('5', 'BraziliaLtd');  INSERT dbo.SUPPLY2 VALUES ('231', 'FarEast');  INSERT dbo.SUPPLY2 VALUES ('280', 'NZ');  INSERT dbo.SUPPLY3 VALUES ('321', 'EuroGroup');  INSERT dbo.SUPPLY3 VALUES ('442', 'UKArchip');  INSERT dbo.SUPPLY4 VALUES ('475', 'India');  INSERT dbo.SUPPLY4 VALUES ('521', 'Afrique');  GO  --Create the view that combines all supplier tables.  CREATE VIEW all\_supplier\_view  WITH SCHEMABINDING  AS  SELECT supplyID, supplier  FROM dbo.SUPPLY1  UNION ALL  SELECT supplyID, supplier  FROM dbo.SUPPLY2  UNION ALL  SELECT supplyID, supplier  FROM dbo.SUPPLY3  UNION ALL  SELECT supplyID, supplier  FROM dbo.SUPPLY4; | |

Modifies a previously created view. This includes an indexed view. ALTER VIEW does not affect dependent stored procedures or triggers and does not change permissions. For more information about the parameters used in the ALTER VIEW statement, see CREATE VIEW (Transact-SQL).

Topic link iconTransact-SQL Syntax Conventions

# ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/collapse.gifSyntax

|  |
| --- |
|  |
| ALTER VIEW [ *schema\_name* . ] *view\_name* [ **(** *column* [ **,**...*n* ] **)** ]  [ WITH <view\_attribute> [ **,**...n ] ]  AS *select\_statement*  [ WITH CHECK OPTION ] [ ; ]  <view\_attribute> ::=  {  [ ENCRYPTION ]  [ SCHEMABINDING ]  [ VIEW\_METADATA ]  } |

# ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/collapse.gifArguments

*schema\_name*

Is the name of the schema to which the view belongs.

*view\_name*

Is the view to change.

*column*

Is the name of one or more columns, separated by commas, that are to be part of the specified view.

|  |
| --- |
| **ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/note.gifImportant:** |
| Column permissions are maintained only when columns have the same name before and after ALTER VIEW is performed. |
| **ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/note.gifNote:** |
| In the columns for the view, the permissions for a column name apply across a CREATE VIEW or ALTER VIEW statement, regardless of the source of the underlying data. For example, if permissions are granted on the **SalesOrderID** column in a CREATE VIEW statement, an ALTER VIEW statement can rename the **SalesOrderID** column, such as to **OrderRef**, and still have the permissions associated with the view using **SalesOrderID**. |

ENCRYPTION

Encrypts the entries in sys.syscomments that contain the text of the ALTER VIEW statement. WITH ENCRYPTION prevents the view from being published as part of SQL Server replication.

SCHEMABINDING

Binds the view to the schema of the underlying table or tables. When SCHEMABINDING is specified, the base tables cannot be modified in a way that would affect the view definition. The view definition itself must first be modified or dropped to remove dependencies on the table to be modified. When you use SCHEMABINDING, the *select\_statement* must include the two-part names (*schema***.***object*) of tables, views, or user-defined functions that are referenced. All referenced objects must be in the same database.

Views or tables that participate in a view created with the SCHEMABINDING clause cannot be dropped, unless that view is dropped or changed so that it no longer has schema binding. Otherwise, the Database Engine raises an error. Also, executing ALTER TABLE statements on tables that participate in views that have schema binding fail if these statements affect the view definition.

SCHEMABINDING cannot be specified if the view contains alias data type columns.

VIEW\_METADATA

Specifies that the instance of SQL Server will return to the DB-Library, ODBC, and OLE DB APIs the metadata information about the view, instead of the base table or tables, when browse-mode metadata is being requested for a query that references the view. Browse-mode metadata is additional metadata that the instance of Database Engine returns to the client-side DB-Library, ODBC, and OLE DB APIs. This metadata enables the client-side APIs to implement updatable client-side cursors. Browse-mode metadata includes information about the base table that the columns in the result set belong to.

For views created with VIEW\_METADATA, the browse-mode metadata returns the view name and not the base table names when it describes columns from the view in the result set.

When a view is created by using WITH VIEW\_METADATA, all its columns, except a **timestamp** column, are updatable if the view has INSERT or UPDATE INSTEAD OF triggers. For more information, see the Remarks section in CREATE VIEW (Transact-SQL).

AS

Are the actions the view is to take.

*select\_statement*

Is the SELECT statement that defines the view.

WITH CHECK OPTION

Forces all data modification statements that are executed against the view to follow the criteria set within *select\_statement*.

# ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/collapse.gifRemarks

For more information about ALTER VIEW, see Remarks in CREATE VIEW (Transact-SQL).

|  |
| --- |
| **ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/note.gifNote:** |
| If the previous view definition was created by using WITH ENCRYPTION or CHECK OPTION, these options are enabled only if they are included in ALTER VIEW. |

If a view currently used is modified by using ALTER VIEW, the Database Engine takes an exclusive schema lock on the view. When the lock is granted, and there are no active users of the view, the Database Engine deletes all copies of the view from the procedure cache. Existing plans referencing the view remain in the cache but are recompiled when invoked.

ALTER VIEW can be applied to indexed views; however, ALTER VIEW unconditionally drops all indexes on the view.

# ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/collapse.gifPermissions

To execute ALTER VIEW, at a minimum, ALTER permission on OBJECT is required.

# ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/collapse.gifExamples

The following example creates a view that contains all employees and their hire dates called EmployeeHireDate. Permissions are granted to the view, but requirements are changed to select employees whose hire dates fall before a certain date. Then, ALTER VIEW is used to replace the view.

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/copycode.gifCopy Code |
| USE AdventureWorks ;  GO  CREATE VIEW HumanResources.EmployeeHireDate  AS  SELECT c.FirstName, c.LastName, e.HireDate  FROM HumanResources.Employee AS e JOIN Person.Contact AS c  ON e.ContactID = c.ContactID ;  GO | |

The view must be changed to include only the employees that were hired before 1997. If ALTER VIEW is not used, but instead the view is dropped and re-created, the previously used GRANT statement and any other statements that deal with permissions pertaining to this view must be re-entered.

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/local/copycode.gifCopy Code |
| ALTER VIEW HumanResources.EmployeeHireDate  AS  SELECT c.FirstName, c.LastName, e.HireDate  FROM HumanResources.Employee AS e JOIN Person.Contact AS c  ON e.ContactID = c.ContactID  WHERE HireDate < CONVERT(DATETIME,'19980101',101) ;  GO | |