OUTPUT (cláusula de Transact-SQL)

SQL Server 2012 Personas que lo han encontrado útil: 1 de 1

Devuelve información de las filas afectadas por una instrucción INSERT, UPDATE, DELETE o MERGE, o expresiones basadas en esas filas. Estos resultados se pueden devolver a la aplicación de procesamiento para que los utilice en mensajes de confirmación, archivado y otros requisitos similares de una aplicación. Los resultados también se pueden insertar en una tabla o variable de tabla. Además, puede capturar los resultados de una cláusula OUTPUT en una instrucción anidada INSERT, UPDATE, DELETE o MERGE, e insertar los resultados en una tabla de destino o vista.

Nota
Una instrucción UPDATE, INSERT o DELETE que tenga una cláusula OUTPUT devolverá filas al cliente aunque la instrucción encuentre errores y se revierta. El resultado no se debe usar si se produce algún error al ejecutar la instrucción.

Se utiliza en:

DELETE

INSERT

UPDATE

MERGE

Convenciones de sintaxis de Transact-SQL

Sintaxis

```
<OUTPUT_CLAUSE> ::=
{
    [ OUTPUT <dml_select_list> INTO { @table_variable | output_table } [ ( column_list ) ] ]
    [ OUTPUT <dml_select_list> ]
}
<dml_select_list> ::=
{ <column_name> | scalar_expression } [ [AS] column_alias_identifier ]
    [ ,...n ]

<column_name> ::=
{ DELETED | INSERTED | from_table_name } . { * | column_name }
    | $action
```

Argumentos

@table_variable

@table_variable

Especifica una variable **table** en la que se insertan las filas devueltas en lugar de devolverse al autor de la llamada. Debe declararse *@table_variable* antes de la instrucción INSERT, UPDATE, DELETE o MERGE.

Si no se especifica *column_list*, la variable **table** debe tener el mismo número de columnas que el conjunto de resultados OUTPUT. Las excepciones son las columnas de identidad y calculadas, que deben omitirse. Si se especifica *column_list*, las columnas omitidas deben aceptar valores NULL o tener valores predeterminados asignados.

Para obtener más información acerca de las variables **table**, vea table (Transact-SQL).

output_table

Especifica una tabla en la que se insertan las filas devueltas en lugar de devolverse al autor de la llamada. *output_table* puede ser una tabla temporal.

Si no se especifica *column_list*, la tabla debe tener el mismo número de columnas que el conjunto de resultados OUTPUT. Las excepciones son las columnas de identidad y calculadas. Éstas deben omitirse. Si se especifica *column_list*, las columnas omitidas deben aceptar valores NULL o tener valores predeterminados asignados.

output_table no puede:

- Tener definidos desencadenadores habilitados.
- Participar en alguna de las partes de una restricción FOREIGN KEY.
- Tener restricciones CHECK o reglas habilitadas.

column_list

Es una lista opcional de nombres de columna de la tabla de destino de la cláusula INTO. Es equivalente a la lista de columnas permitida en la instrucción INSERT.

scalar_expression

Es cualquier combinación de símbolos y operadores que se evalúa como un solo valor. No se permiten funciones de agregado en *scalar_expression*.

Cualquier referencia a las columnas de la tabla que se va a modificar debe calificarse con el prefijo INSERTED o DELETED.

column_alias_identifier

Es un nombre alternativo que se utiliza para hacer referencia al nombre de columna.

DELETED

Es un prefijo de columna que especifica el valor eliminado en la operación de actualización o eliminación. Las columnas con prefijo DELETED reflejan el valor antes de que se complete la instrucción UPDATE, DELETE o MERGE.

DELETED no se puede utilizar con la cláusula OUTPUT en la instrucción INSERT.

INSERTED

Es un prefijo de columna que especifica el valor agregado en la operación de inserción o actualización. Las columnas con prefijo INSERTED reflejan el valor después de que se complete la instrucción UPDATE, INSERT o MERGE, pero antes de que se ejecuten los desencadenadores.

INSERTED no se puede utilizar con la cláusula OUTPUT en la instrucción DELETE.

from_table_name

Es un prefijo de columna que especifica una tabla incluida en la cláusula FROM de una instrucción DELETE, UPDATE o MERGE que se utiliza para especificar las filas que se van a actualizar o eliminar.

Si la tabla que se va a modificar se especifica también en la cláusula FROM, cualquier referencia a las columnas de esa tabla deben calificarse con el prefijo INSERTED o DELETED.

*

Especifica que todas las columnas afectadas por la acción de eliminación, inserción o actualización se devuelvan en el orden en que se encuentran en la tabla.

Por ejemplo, `OUTPUT DELETED.*` en la siguiente instrucción DELETE devuelve todas las columnas eliminadas de la tabla `ShoppingCartItem`:

```
DELETE Sales.ShoppingCartItem
    OUTPUT DELETED.*;
```

column_name

Es una referencia explícita a una columna. Cualquier referencia a la tabla que se va a modificar debe certificarse correctamente mediante el prefijo INSERTED o DELETED, según corresponda; por ejemplo: INSERTED**.***column_name*.

$action

Solo está disponible para la instrucción MERGE. Especifica una columna de tipo **nvarchar(10)** en la cláusula OUTPUT de una instrucción MERGE que devuelve uno de estos tres valores por cada fila: 'INSERT', 'UPDATE' o 'DELETE', según la acción realizada en dicha fila.

Comentarios

La cláusula OUTPUT <dml_select_list> y la cláusula OUTPUT <dml_select_list> INTO { **@***table_variable* | *output_table* } pueden definirse en una sola instrucción INSERT, UPDATE, DELETE o MERGE.

Nota
A menos que se indique lo contrario, las referencias a la cláusula OUTPUT se refieren tanto a la cláusula OUTPUT como a la cláusula OUTPUT INTO.

La cláusula OUTPUT puede ser útil para recuperar el valor de las columnas de identidad o calculadas después de una operación con INSERT o UPDATE.

Cuando se incluye una columna calculada en <dml_select_list>, la columna correspondiente de la tabla de salida o variable de tabla no es una columna calculada. Los valores de la nueva columna son los que se calcularon en el momento en que se ejecutó la instrucción.

No se garantiza que coincidan el orden en que se aplican los cambios en la tabla y el orden en que se insertan las filas en la tabla de salida o variable de tabla.

Si se modifican parámetros o variables como parte de una instrucción UPDATE, la cláusula OUTPUT siempre devuelve el valor del parámetro o la variable tal como se encontraba antes de ejecutar la instrucción, en lugar de devolver el valor modificado.

OUTPUT se puede utilizar con una instrucción UPDATE o DELETE en un cursor que utilice la sintaxis WHERE CURRENT OF.

La cláusula OUTPUT no se admite en las siguientes instrucciones:

- Instrucciones DML que hacen referencia a vistas locales con particiones, vistas distribuidas con particiones o

- Instrucciones DML que hacen referencia a vistas locales con particiones, vistas distribuidas con particiones o tablas remotas.
- Instrucciones INSERT que contienen una instrucción EXECUTE.
- Los predicados de texto completo no están permitidos en la cláusula OUTPUT cuando el nivel de compatibilidad de la base de datos está establecido en 100.
- La cláusula OUTPUT INTO no se puede utilizar para realizar inserciones en vistas o en una función de conjunto de filas.
- No se puede crear una función definida por el usuario si contiene una cláusula OUTPUT INTO que tiene una tabla como destino.

Para evitar el comportamiento no determinista, la cláusula OUTPUT no puede contener las referencias siguientes:

- Subconsultas o funciones definidas por el usuario que obtienen acceso a datos de usuario o del sistema, o que se asume que obtienen dicho acceso. Se supone que las funciones definidas por el usuario realizan el acceso a los datos si no están enlazadas a un esquema.
- Una columna de una vista o función insertada con valores de tabla si la columna se define mediante uno de los métodos siguientes:
  - Una subconsulta.
  - Una función definida por el usuario que obtiene acceso a datos de usuario o del sistema, o que se asume que obtiene dicho acceso.
  - Una columna calculada que contiene una función definida por el usuario que obtiene acceso a datos de usuario o del sistema en su definición.

  Cuando SQL Server detecta este tipo de columna en la cláusula OUTPUT, se produce el error 4186. Para obtener más información, vea MSSQLSERVER_4186.

Insertar datos devueltos de una cláusula OUTPUT en una tabla

Al capturar los resultados de una cláusula OUTPUT en una instrucción INSERT, UPDATE, DELETE o MERGE anidada e insertarlos en una tabla de destino, tenga presente la información siguiente:

- Toda la operación es atómica. Se ejecutarán la instrucción INSERT y la instrucción DML anidada que contiene la cláusula OUTPUT, o bien se producirá un error en toda la instrucción.
- Las restricciones siguientes se aplican al destino de la instrucción INSERT externa:
  - El destino no puede ser una expresión de tabla común, vista o tabla remota.
  - El destino no puede tener una restricción FOREIGN KEY, ni ser objeto de referencia por una restricción FOREIGN KEY.
  - No se pueden definir desencadenadores en el destino.
  - El destino no puede participar en la replicación de mezcla ni en las suscripciones actualizables para la replicación transaccional.
- Las restricciones siguientes se aplican a la instrucción DML anidada:
  - El destino no puede ser una tabla remota ni una vista con particiones.

- El propio origen no puede contener una cláusula <dml_table_source>.

- La cláusula OUTPUT INTO no se admite en instrucciones INSERT que contengan una cláusula <dml_table_source>.
- @@ROWCOUNT devuelve las filas insertadas únicamente por la instrucción INSERT externa.
- @@IDENTITY, SCOPE_IDENTITY e IDENT_CURRENT devuelven los valores de identidad generados solo por la instrucción DML anidada, y no los generados por la instrucción INSERT externa.
- Las notificaciones de consulta tratan la instrucción como una entidad única, y el tipo de cualquier mensaje creado es el del DML anidado, aunque el cambio significativo provenga de la propia instrucción INSERT externa.
- En la cláusula <dml_table_source>, las cláusulas SELECT y WHERE no pueden incluir subconsultas, funciones de agregado, funciones de categoría, predicados de texto completo, funciones definidas por el usuario que realicen accesos a datos, ni la función TEXTPTR.

Desencadenadores

Las columnas devueltas de OUTPUT reflejan los datos tal como estaban después de completarse la instrucción INSERT, UPDATE o DELETE, pero antes de ejecutarse los desencadenadores.

En el caso de los desencadenadores INSTEAD OF, los resultados devueltos se generan como si la operación de INSERT, UPDATE o DELETE se hubiese producido realmente, aunque no se produzcan modificaciones como resultado de la operación del desencadenador. Si se utiliza una instrucción que incluye una cláusula OUTPUT en el cuerpo de un desencadenador, deben utilizarse alias de tabla para hacer referencia a las tablas inserted y deleted del desencadenador con el fin de evitar la duplicación de las referencias a columnas con las tablas INSERTED y DELETED asociadas a OUTPUT.

Si la cláusula OUTPUT se especifica sin especificar también la palabra clave INTO, el destino de la operación DML no puede tener definido ningún desencadenador habilitado para la acción DML dada. Por ejemplo, si se define la cláusula OUTPUT en una instrucción UPDATE, la tabla de destino no puede tener desencadenadores UPDATE habilitados.

Si se establece la opción sp_configure de disallow results from triggers, una cláusula OUTPUT sin una cláusula INTO hará que la instrucción genere un error cuando se invoque desde un desencadenador.

Tipos de datos

La cláusula OUTPUT admite los tipos de datos de objetos grandes: **nvarchar(max)**, **varchar(max)**, **varbinary(max)**, **text**, **ntext**, **image** y **xml**. Cuando se utiliza la cláusula .WRITE en la instrucción UPDATE para modificar una columna de tipo **nvarchar(max)**, **varchar(max)** o **varbinary(max)**, se devuelven las imágenes anterior y posterior completas de los valores si se hace referencia a ellas. La función TEXTPTR( ) no puede aparecer como parte de una expresión en una columna de tipo **text**, **ntext** o **image** en la cláusula OUTPUT.

Colas

OUTPUT se puede utilizar en aplicaciones que utilizan tablas como colas, o para contener conjuntos de resultados intermedios. Dicho de otro modo, la aplicación agrega o quita filas de la tabla constantemente. En el ejemplo siguiente se utiliza la cláusula OUTPUT en una instrucción DELETE para devolver la fila eliminada a la aplicación que realiza la llamada.

Transact-SQL

```
USE AdventureWorks2012;
GO
DELETE TOP(1) dbo.DatabaseLog WITH (READPAST)
OUTPUT deleted.*
WHERE DatabaseLogID = 7;
GO
```

En este ejemplo, se quita una fila de una tabla utilizada como cola y se devuelven los valores eliminados a la aplicación de procesamiento en una única acción. También se puede implementar otro tipo de semántica, como utilizar una tabla para implementar una pila. No obstante, SQL Server no garantiza el orden en que las instrucciones DML procesan y devuelven las filas por medio de la cláusula OUTPUT. Es la aplicación la que debe incluir una cláusula WHERE que garantice la semántica deseada, o reconocer que, si hay varias filas aptas para la operación DML, no se garantiza el orden. En el ejemplo siguiente se utiliza una subconsulta y se supone que la unicidad es una característica de la columna `DatabaseLogID` para implementar la semántica de ordenación deseada.

Transact-SQL

```
USE tempdb;
GO

CREATE TABLE dbo.table1
(
    id INT,
    employee VARCHAR(32)
)
go

INSERT INTO dbo.table1 VALUES
      (1, 'Fred')
     ,(2, 'Tom')
     ,(3, 'Sally')
     ,(4, 'Alice');
GO

DECLARE @MyTableVar TABLE
(
    id INT,
    employee VARCHAR(32)
);

PRINT 'table1, before delete'
SELECT * FROM dbo.table1;

DELETE FROM dbo.table1
OUTPUT DELETED.* INTO @MyTableVar
WHERE id = 4 OR id = 2;

PRINT 'table1, after delete'
SELECT * FROM dbo.table1;

PRINT '@MyTableVar, after delete'
SELECT * FROM @MyTableVar;

DROP TABLE dbo.table1;

--Results
--table1, before delete
--id          employee
------------- ------------------------------
--1           Fred
--2           Tom
--3           Sally
--4           Alice
--
--table1, after delete
```

```
--id          employee
------------- ------------------------------
--1           Fred
--3           Sally
--@MyTableVar, after delete
--id          employee
------------- ------------------------------
--2           Tom
--4           Alice
```

Nota

Use la sugerencia de tabla READPAST en las instrucciones UPDATE y DELETE si el escenario permite que varias aplicaciones realicen una lectura destructiva de una tabla. De esta forma se impide que surjan problemas de bloqueo si otra aplicación ya está leyendo el primer registro de la tabla que reúne los requisitos.

Permisos

Se requieren permisos SELECT en las columnas recuperadas a través de <dml_select_list> o utilizadas en <scalar_expression>.

Se requieren permisos INSERT en las tablas especificadas en <output_table>.

Ejemplos

A.Utilizar OUTPUT INTO con una instrucción INSERT simple

En el siguiente ejemplo se inserta una fila en la tabla `ScrapReason` y se utiliza la cláusula `OUTPUT` para devolver los resultados de la instrucción a la variable `table@MyTableVar`. Como la columna `ScrapReasonID` se ha definido con una propiedad IDENTITY, no se especifica ningún valor en la instrucción `INSERT` de esa columna. No obstante, tenga en cuenta que el valor generado por el Motor de base de datos para esa columna se devuelve en la cláusula `OUTPUT` de la columna `inserted.ScrapReasonID`.

Transact-SQL

```
USE AdventureWorks2012;
GO
DECLARE @MyTableVar table( NewScrapReasonID smallint,
                           Name varchar(50),
                           ModifiedDate datetime);
INSERT Production.ScrapReason
    OUTPUT INSERTED.ScrapReasonID, INSERTED.Name, INSERTED.ModifiedDate
        INTO @MyTableVar
VALUES (N'Operator error', GETDATE());

--Display the result set of the table variable.
SELECT NewScrapReasonID, Name, ModifiedDate FROM @MyTableVar;
--Display the result set of the table.
SELECT ScrapReasonID, Name, ModifiedDate
FROM Production.ScrapReason;
GO
```

B.Usar OUTPUT con una instrucción DELETE

En el ejemplo siguiente se eliminan todas las filas de la tabla `ShoppingCartItem`. La cláusula `OUTPUT deleted.*` especifica que se devuelvan a la aplicación que realiza la llamada los resultados de la instrucción `DELETE`, es decir, todas las columnas de las filas eliminadas. La instrucción `SELECT` posterior comprueba los resultados de la operación de eliminación en la tabla `ShoppingCartItem`.

Transact-SQL

```
USE AdventureWorks2012;
GO
DELETE Sales.ShoppingCartItem
OUTPUT DELETED.*
WHERE ShoppingCartID = 20621;

--Verify the rows in the table matching the WHERE clause have been deleted.
SELECT COUNT(*) AS [Rows in Table] FROM Sales.ShoppingCartItem WHERE ShoppingCartID = 20621;
GO
```

C.Usar OUTPUT INTO con una instrucción UPDATE

En el ejemplo siguiente se actualiza un 25 por ciento la columna `VacationHours` de las 10 primeras filas de la tabla `Employee`. La cláusula `OUTPUT` devuelve el valor de `VacationHours` antes de aplicar la instrucción `UPDATE` en la columna `deleted.VacationHours`, y el valor actualizado de la columna `inserted.VacationHours` en la variable `table@MyTableVar`.

A continuación, dos instrucciones `SELECT` devuelven los valores de `@MyTableVar` y los resultados de la operación de actualización en la tabla `Employee`.

Transact-SQL

```
USE AdventureWorks2012;
GO
DECLARE @MyTableVar table(
    EmpID int NOT NULL,
    OldVacationHours int,
    NewVacationHours int,
    ModifiedDate datetime);
UPDATE TOP (10) HumanResources.Employee
SET VacationHours = VacationHours * 1.25,
    ModifiedDate = GETDATE()
OUTPUT inserted.BusinessEntityID,
       deleted.VacationHours,
       inserted.VacationHours,
       inserted.ModifiedDate
INTO @MyTableVar;
--Display the result set of the table variable.
SELECT EmpID, OldVacationHours, NewVacationHours, ModifiedDate
FROM @MyTableVar;
GO
--Display the result set of the table.
SELECT TOP (10) BusinessEntityID, VacationHours, ModifiedDate
FROM HumanResources.Employee;
GO
```

D.Usar OUTPUT INTO para devolver una expresión

El ejemplo siguiente, que se basa en el ejemplo C, define una expresión en la cláusula `OUTPUT` como la diferencia entre el

El ejemplo siguiente, que se basa en el ejemplo C, define una expresión en la cláusula `OUTPUT` como la diferencia entre el valor actualizado de `VacationHours` y el valor de `VacationHours` antes de aplicar la actualización. El valor de esta expresión se devuelve a la variable `table@MyTableVar` en la columna `VacationHoursDifference`.

Transact-SQL

```
USE AdventureWorks2012;
GO
DECLARE @MyTableVar table(
    EmpID int NOT NULL,
    OldVacationHours int,
    NewVacationHours int,
    VacationHoursDifference int,
    ModifiedDate datetime);
UPDATE TOP (10) HumanResources.Employee
SET VacationHours = VacationHours * 1.25,
    ModifiedDate = GETDATE()
OUTPUT inserted.BusinessEntityID,
       deleted.VacationHours,
       inserted.VacationHours,
       inserted.VacationHours - deleted.VacationHours,
       inserted.ModifiedDate
INTO @MyTableVar;
--Display the result set of the table variable.
SELECT EmpID, OldVacationHours, NewVacationHours,
    VacationHoursDifference, ModifiedDate
FROM @MyTableVar;
GO
SELECT TOP (10) BusinessEntityID, VacationHours, ModifiedDate
FROM HumanResources.Employee;
GO
```

E.Usar OUTPUT INTO con from_table_name en una instrucción UPDATE

En el ejemplo siguiente se actualiza la columna `ScrapReasonID` de la tabla `WorkOrder` para todas las órdenes de trabajo que tengan especificados `ProductID` y `ScrapReasonID`. La cláusula `OUTPUT INTO` devuelve los valores de la tabla que se actualiza (`WorkOrder`) y de la tabla `Product`. La tabla `Product` se utiliza en la cláusula `FROM` para especificar las filas que se van a actualizar. Dado que la tabla `WorkOrder` tiene definido un desencadenador `AFTER UPDATE`, se requiere la palabra clave `INTO`.

Transact-SQL

```
USE AdventureWorks2012;
GO
DECLARE @MyTestVar table (
    OldScrapReasonID int NOT NULL,
    NewScrapReasonID int NOT NULL,
    WorkOrderID int NOT NULL,
    ProductID int NOT NULL,
    ProductName nvarchar(50)NOT NULL);
UPDATE Production.WorkOrder
SET ScrapReasonID = 4
OUTPUT deleted.ScrapReasonID,
       inserted.ScrapReasonID,
       inserted.WorkOrderID,
       inserted.ProductID,
       p.Name
    INTO @MyTestVar
FROM Production.WorkOrder AS wo
```

```
FROM Production.WorkOrder AS wo
    INNER JOIN Production.Product AS p
    ON wo.ProductID = p.ProductID
    AND wo.ScrapReasonID= 16
    AND p.ProductID = 733;
SELECT OldScrapReasonID, NewScrapReasonID, WorkOrderID,
    ProductID, ProductName
FROM @MyTestVar;
GO
```

F.Usar OUTPUT INTO con from_table_name en una instrucción DELETE

En el ejemplo siguiente se eliminan las filas de la tabla `ProductProductPhoto` según los criterios de búsqueda definidos en la cláusula `FROM` de la instrucción `DELETE`. La cláusula `OUTPUT` devuelve columnas de la tabla que se elimina (`deleted.ProductID`, `deleted.ProductPhotoID`) y de la tabla `Product`. La tabla se utiliza en la cláusula `FROM` para especificar las filas que se van a eliminar.

Transact-SQL

```
USE AdventureWorks2012;
GO
DECLARE @MyTableVar table (
    ProductID int NOT NULL,
    ProductName nvarchar(50)NOT NULL,
    ProductModelID int NOT NULL,
    PhotoID int NOT NULL);

DELETE Production.ProductProductPhoto
OUTPUT DELETED.ProductID,
       p.Name,
       p.ProductModelID,
       DELETED.ProductPhotoID
    INTO @MyTableVar
FROM Production.ProductProductPhoto AS ph
JOIN Production.Product as p
    ON ph.ProductID = p.ProductID
    WHERE p.ProductModelID BETWEEN 120 and 130;

--Display the results of the table variable.
SELECT ProductID, ProductName, ProductModelID, PhotoID
FROM @MyTableVar
ORDER BY ProductModelID;
GO
```

G.Usar OUTPUT INTO con un tipo de datos de objetos grandes

En el ejemplo siguiente se actualiza un valor parcial de `DocumentSummary`, una columna de tipo `nvarchar(max)` de la tabla `Production.Document`, utilizando la cláusula `.WRITE`. La palabra `components` se sustituye por la palabra `features` al especificar la palabra sustituta, la ubicación inicial (desplazamiento) de la palabra que se va a sustituir en los datos existentes y el número de caracteres que se va a sustituir (longitud). En el ejemplo se utiliza la cláusula `OUTPUT` para devolver las imágenes anterior y posterior de la columna `DocumentSummary` en la variable `table@MyTableVar`. Observe que se devuelven las imágenes anterior y posterior completas de la columna `DocumentSummary`.

Transact-SQL

```
USE AdventureWorks2012;
GO
DECLARE @MyTableVar table (
```

```
    SummaryBefore nvarchar(max),
    SummaryAfter nvarchar(max));
UPDATE Production.Document
SET DocumentSummary .WRITE (N'features',28,10)
OUTPUT deleted.DocumentSummary,
       inserted.DocumentSummary
    INTO @MyTableVar
WHERE Title = N'Front Reflector Bracket Installation';
SELECT SummaryBefore, SummaryAfter
FROM @MyTableVar;
GO
```

H.Usar OUTPUT en un desencadenador INSTEAD OF

En el ejemplo siguiente se utiliza la cláusula `OUTPUT` en un desencadenador para devolver los resultados de la operación del desencadenador. En primer lugar se crea una vista en la tabla `ScrapReason` y, después, en la vista se define un desencadenador `INSTEAD OF INSERT` que permite al usuario modificar únicamente la columna `Name` de la tabla base. Puesto que la columna `ScrapReasonID` es una columna `IDENTITY` de la tabla base, el desencadenador omite el valor suministrado por el usuario. Esto permite que el Motor de base de datos genere automáticamente el valor correcto. Asimismo, se omite el valor suministrado por el usuario para `ModifiedDate`, que se establece en la fecha actual. La cláusula `OUTPUT` devuelve los valores reales insertados en la tabla `ScrapReason`.

Transact-SQL

```
USE AdventureWorks2012;
GO
IF OBJECT_ID('dbo.vw_ScrapReason','V') IS NOT NULL
    DROP VIEW dbo.vw_ScrapReason;
GO
CREATE VIEW dbo.vw_ScrapReason
AS (SELECT ScrapReasonID, Name, ModifiedDate
    FROM Production.ScrapReason);
GO
CREATE TRIGGER dbo.io_ScrapReason
    ON dbo.vw_ScrapReason
INSTEAD OF INSERT
AS
BEGIN
--ScrapReasonID is not specified in the list of columns to be inserted
--because it is an IDENTITY column.
    INSERT INTO Production.ScrapReason (Name, ModifiedDate)
        OUTPUT INSERTED.ScrapReasonID, INSERTED.Name,
               INSERTED.ModifiedDate
    SELECT Name, getdate()
    FROM inserted;
END
GO
INSERT vw_ScrapReason (ScrapReasonID, Name, ModifiedDate)
VALUES (99, N'My scrap reason','20030404');
GO
```

Éste es el conjunto de resultados generado el 12 de abril de 2004 (`'2004-04-12'`). Tenga en cuenta que las columnas `ScrapReasonIDActual` y `ModifiedDate` reflejan los valores generados en la operación del desencadenador en lugar de los valores suministrados en la instrucción `INSERT`.

```
ScrapReasonID  Name            ModifiedDate
```

```
------------  --------------- ----------------------

17            My scrap reason  2004-04-12 16:23:33.050
```

I.Usar OUTPUT INTO con columnas de identidad y calculadas

En el ejemplo siguiente se crea la tabla `EmployeeSales` y, después, se insertan en ella varias filas utilizando una instrucción `INSERT` con una instrucción `SELECT` para recuperar los datos de las tablas de origen. La tabla `EmployeeSales` contiene una columna de identidad (`EmployeeID`) y una columna calculada (`ProjectedSales`). Puesto que Motor de base de datos de SQL Server genera estos valores durante la operación de inserción, ninguna de estas columnas se puede definir en `@MyTableVar`.

Transact-SQL

```
USE AdventureWorks2012 ;
GO
IF OBJECT_ID ('dbo.EmployeeSales', 'U') IS NOT NULL
    DROP TABLE dbo.EmployeeSales;
GO
CREATE TABLE dbo.EmployeeSales
( EmployeeID   int IDENTITY (1,5)NOT NULL,
  LastName     nvarchar(20) NOT NULL,
  FirstName    nvarchar(20) NOT NULL,
  CurrentSales money NOT NULL,
  ProjectedSales AS CurrentSales * 1.10
);
GO
DECLARE @MyTableVar table(
  LastName     nvarchar(20) NOT NULL,
  FirstName    nvarchar(20) NOT NULL,
  CurrentSales money NOT NULL
  );

INSERT INTO dbo.EmployeeSales (LastName, FirstName, CurrentSales)
  OUTPUT INSERTED.LastName,
         INSERTED.FirstName,
         INSERTED.CurrentSales
  INTO @MyTableVar
    SELECT c.LastName, c.FirstName, sp.SalesYTD
    FROM Sales.SalesPerson AS sp
    INNER JOIN Person.Person AS c
        ON sp.BusinessEntityID = c.BusinessEntityID
    WHERE sp.BusinessEntityID LIKE '2%'
    ORDER BY c.LastName, c.FirstName;

SELECT LastName, FirstName, CurrentSales
FROM @MyTableVar;
GO
SELECT EmployeeID, LastName, FirstName, CurrentSales, ProjectedSales
FROM dbo.EmployeeSales;
GO
```

J.Usar OUTPUT y OUTPUT INTO en una sola instrucción

En el ejemplo siguiente se eliminan las filas de la tabla `ProductProductPhoto` según los criterios de búsqueda definidos en la cláusula `FROM` de la instrucción `DELETE`. La cláusula `OUTPUT INTO` devuelve las columnas de la tabla que se elimina (`deleted.ProductID`, `deleted.ProductPhotoID`) y columnas de la tabla `Product` a la variable `table@MyTableVar`. La tabla `Product` se utiliza en la cláusula `FROM` para especificar las filas que se van a eliminar. La cláusula `OUTPUT` devuelve las columnas `deleted.ProductID` y `deleted.ProductPhotoID`, y la fecha y hora de eliminación de la fila de la tabla

ProductProductPhoto a la aplicación que realiza la llamada.

Transact-SQL

```
USE AdventureWorks2012;
GO
DECLARE @MyTableVar table (
    ProductID int NOT NULL,
    ProductName nvarchar(50)NOT NULL,
    ProductModelID int NOT NULL,
    PhotoID int NOT NULL);

DELETE Production.ProductProductPhoto
OUTPUT DELETED.ProductID,
       p.Name,
       p.ProductModelID,
       DELETED.ProductPhotoID
    INTO @MyTableVar
OUTPUT DELETED.ProductID, DELETED.ProductPhotoID, GETDATE() AS DeletedDate
FROM Production.ProductProductPhoto AS ph
JOIN Production.Product as p
    ON ph.ProductID = p.ProductID
WHERE p.ProductID BETWEEN 800 and 810;

--Display the results of the table variable.
SELECT ProductID, ProductName, PhotoID, ProductModelID
FROM @MyTableVar;
GO
```

K.Insertar los datos devueltos por una cláusula OUTPUT

El ejemplo siguiente captura datos devueltos por la cláusula OUTPUT de una instrucción MERGE y los inserta en otra tabla. La instrucción MERGE actualiza diariamente la columna Quantity de la tabla ProductInventory en función de los pedidos procesados en la tabla SalesOrderDetail. También elimina las filas correspondientes a los productos cuyas existencias están en el valor 0 o por debajo de este valor. En el ejemplo, se capturan las filas que se eliminan y se insertan en otra tabla, ZeroInventory, que realiza el seguimiento de los productos sin existencias.

Transact-SQL

```
USE AdventureWorks2012;
GO
IF OBJECT_ID(N'Production.ZeroInventory', N'U') IS NOT NULL
    DROP TABLE Production.ZeroInventory;
GO
--Create ZeroInventory table.
CREATE TABLE Production.ZeroInventory (DeletedProductID int, RemovedOnDate DateTime);
GO

INSERT INTO Production.ZeroInventory (DeletedProductID, RemovedOnDate)
SELECT ProductID, GETDATE()
FROM
(   MERGE Production.ProductInventory AS pi
    USING (SELECT ProductID, SUM(OrderQty) FROM Sales.SalesOrderDetail AS sod
           JOIN Sales.SalesOrderHeader AS soh
           ON sod.SalesOrderID = soh.SalesOrderID
           AND soh.OrderDate = '20070401'
           GROUP BY ProductID) AS src (ProductID, OrderQty)
    ON (pi.ProductID = src.ProductID)
```

```
    WHEN MATCHED AND pi.Quantity - src.OrderQty <= 0
        THEN DELETE
    WHEN MATCHED
        THEN UPDATE SET pi.Quantity = pi.Quantity - src.OrderQty
    OUTPUT $action, deleted.ProductID) AS Changes (Action, ProductID)
WHERE Action = 'DELETE';
IF @@ROWCOUNT = 0
PRINT 'Warning: No rows were inserted';
GO
SELECT DeletedProductID, RemovedOnDate FROM Production.ZeroInventory;
```

Vea también

Adiciones de comunidad