

Este artículo se tradujo de forma manual. Mueva el puntero sobre las frases del artículo para ver el texto original. [Más información.](#)

 Traducción  Original

MERGE (Transact-SQL)

SQL Server 2012 Personas que lo han encontrado útil: 3 de 3

Realiza operaciones de inserción, actualización o eliminación en una tabla de destino según los resultados de una combinación con una tabla de origen. Por ejemplo, puede sincronizar dos tablas insertando, actualizando o eliminando las filas de una tabla según las diferencias que se encuentren en la otra.

[Convenciones de sintaxis de Transact-SQL](#)

Sintaxis

```
[ WITH <common_table_expression> [,...n] ]
MERGE
    [ TOP ( expression ) [ PERCENT ] ]
    [ INTO ] <target_table> [ WITH ( <merge_hint> ) ] [ [ AS ] table_alias ]
    USING <table_source>
    ON <merge_search_condition>
    [ WHEN MATCHED [ AND <clause_search_condition> ]
      THEN <merge_matched> ] [ ...n ]
    [ WHEN NOT MATCHED [ BY TARGET ] [ AND <clause_search_condition> ]
      THEN <merge_not_matched> ]
    [ WHEN NOT MATCHED BY SOURCE [ AND <clause_search_condition> ]
      THEN <merge_matched> ] [ ...n ]
    [ <output_clause> ]
    [ OPTION ( <query_hint> [ ,...n ] ) ]
;

<target_table> ::=
{
    [ database_name . schema_name . | schema_name . ]
    target_table
}

<merge_hint>::=
{
    { [ <table_hint_limited> [ ,...n ] ]
      [ [ , ] INDEX ( index_val [ ,...n ] ) ] }
}

<table_source> ::=
{
    table_or_view_name [ [ AS ] table_alias ] [ <tablesample_clause> ]
    [ WITH ( table_hint [ [ , ]...n ] ) ]
    | rowset_function [ [ AS ] table_alias ]
      [ ( bulk_column_alias [ ,...n ] ) ]
    | user_defined_function [ [ AS ] table_alias ]
    | OPENXML <openxml_clause>
    | derived_table [ AS ] table_alias [ ( column_alias [ ,...n ] ) ]
}
```

```

| <joined_table>
| <pivoted_table>
| <unpivoted_table>
}

<merge_search_condition> ::=
    <search_condition>

<merge_matched> ::=
    { UPDATE SET <set_clause> | DELETE }

<set_clause> ::=
SET
    { column_name = { expression | DEFAULT | NULL }
    | { udt_column_name. { { property_name = expression
                            | field_name = expression }
                            | method_name ( argument [ ,...n ] ) }
    }
    | column_name { .WRITE ( expression , @Offset , @Length ) }
    | @variable = expression
    | @variable = column = expression
    | column_name { += | -= | *= | /= | %= | &= | ^= | |= } expression
    | @variable { += | -= | *= | /= | %= | &= | ^= | |= } expression
    | @variable = column { += | -= | *= | /= | %= | &= | ^= | |= } expression
    } [ ,...n ]

<merge_not_matched> ::=
{
    INSERT [ ( column_list ) ]
        { VALUES ( values_list )
        | DEFAULT VALUES }
}

<clause_search_condition> ::=
    <search_condition>

<search_condition> ::=
    { [ NOT ] <predicate> | ( <search_condition> ) }
    [ { AND | OR } [ NOT ] { <predicate> | ( <search_condition> ) } ]
    [ ,...n ]

<predicate> ::=
    { expression { = | < | > | != | > | > = | ! > | < | < = | ! < } expression
    | string_expression [ NOT ] LIKE string_expression
    [ ESCAPE 'escape_character' ]
    | expression [ NOT ] BETWEEN expression AND expression
    | expression IS [ NOT ] NULL
    | CONTAINS
    ( { column | * } , '< contains_search_condition >' )
    | FREETEXT ( { column | * } , 'freetext_string' )
    | expression [ NOT ] IN ( subquery | expression [ ,...n ] )
    | expression { = | < | > | != | > | > = | ! > | < | < = | ! < }
    { ALL | SOME | ANY } ( subquery )
    | EXISTS ( subquery ) }

<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

WITH <common_table_expression>

Especifica la vista o el conjunto de resultados temporal indicado, que también se conoce como expresión de tabla común, definido en el ámbito de la instrucción MERGE. El conjunto de resultados se deriva de una consulta simple. La instrucción MERGE hace referencia al conjunto de resultados. Para obtener más información, vea [WITH common_table_expression \(Transact-SQL\)](#).

TOP (*expression*) [PERCENT]

Especifica el número o porcentaje de filas afectadas. *expression* puede ser un número o un porcentaje de filas. Las filas a las que se hace referencia en la expresión TOP no están organizadas en ningún orden. Para obtener más información, vea [TOP \(Transact-SQL\)](#).

La cláusula TOP se aplica después de que se combinen toda la tabla de origen y toda la tabla de destino, y se quiten las filas combinadas que no reúnan las condiciones para las acciones de inserción, actualización o eliminación. La cláusula TOP reduce aún más el número de filas combinadas al valor especificado y se aplican las acciones de inserción, actualización o eliminación a las filas combinadas restantes de una manera desordenada. Es decir, no hay ningún orden en el que las filas se distribuyan entre las acciones definidas en las cláusulas WHEN. Por ejemplo, cuando se especifica TOP (10) afecta a 10 filas; de estas filas, 7 se pueden actualizar y 3 insertar, o se pueden eliminar 1, actualizar 5 e insertar 4, etc.

Dado que la instrucción MERGE realiza un recorrido completo de ambas tablas, de destino y de origen, el rendimiento de E/S puede verse afectado al utilizar la cláusula TOP para modificar una tabla grande mediante la creación de varios lotes. En este escenario, es importante asegurarse de que todos los lotes sucesivos tengan como destino nuevas filas.

database_name

Es el nombre de la base de datos donde se encuentra *target_table*.

schema_name

Es el nombre del esquema al que pertenece *target_table*.

target_table

Es la tabla o la vista con la que las filas de datos de <table_source> se hacen coincidir según la <clause_search_condition>. *target_table* es el destino de las operaciones de inserción, actualización o eliminación que las cláusulas WHEN de la instrucción MERGE especifican.

Si *target_table* es una vista, cualquier acción con ella debe satisfacer las condiciones para actualizar las vistas. Para obtener más información, vea [Modificar datos mediante una vista](#).

target_table no puede ser una tabla remota. *target_table* no puede tener ninguna regla definida.

[AS] *table_alias*

Es un nombre alternativo que se utiliza para hacer referencia a una tabla.

USING <table_source>

Especifica el origen de datos que se hace coincidir con las filas de datos en *target_table* según `<merge_search condition>`. El resultado de esta coincidencia dicta las acciones que tomarán las cláusulas `WHEN` de la instrucción `MERGE`. `<table_source>` puede ser una tabla remota o una tabla derivada que tengan acceso a las tablas remotas.

`<table_source>` puede ser una tabla derivada que use el [constructor con valores de tabla](#) de Transact-SQL para construir una tabla especificando varias filas.

Para obtener más información acerca de la sintaxis y los argumentos de esta cláusula, vea [FROM \(Transact-SQL\)](#).

ON `<merge_search_condition>`

Especifica las condiciones en las que `<table_source>` se combina con *target_table* para determinar dónde coinciden.

Advertencia

Es importante especificar solamente las columnas de la tabla de destino que se utilizan para los propósitos de la coincidencia. Es decir, especifique las columnas de la tabla de destino que se comparan con la correspondiente columna de la tabla de origen. No intente mejorar el rendimiento de las consultas filtrando las filas de la tabla de destino en la cláusula `ON`, según se especifica con `AND NOT target_table.column_x = value`. Si se hace esto, se pueden devolver resultados inesperados e incorrectos.

WHEN MATCHED THEN `<merge_matched>`

Especifica que todas las filas de *target_table* que coinciden con las filas que devuelve `<table_source>` `ON <merge_search_condition>` y que satisfacen alguna condición de búsqueda adicional se actualizan o eliminan según la cláusula `<merge_matched>`.

La instrucción `MERGE` puede tener a lo sumo dos cláusulas `WHEN MATCHED`. Si se especifican dos cláusulas, la primera debe ir acompañada de una cláusula `AND <search_condition>`. Para una fila determinada, la segunda cláusula `WHEN MATCHED` se aplica solamente si no se aplica la primera. Si hay dos cláusulas `WHEN MATCHED`, una debe especificar una acción `UPDATE` y la otra una acción `DELETE`. Si se especifica `UPDATE` en la cláusula `<merge_matched>` y más de una fila de `<table_source>` coincide con una fila en *target_table* según la `<merge_search_condition>`, SQL Server devuelve un error. La instrucción `MERGE` no puede actualizar la misma fila más de una vez, ni actualizar o eliminar la misma fila.

WHEN NOT MATCHED [BY TARGET] THEN `<merge_not_matched>`

Especifica que una fila se inserta en *target_table* para cada fila que devuelve `<table_source>` `ON <merge_search_condition>` que no coincide con una fila de *target_table*, pero satisface una condición de búsqueda adicional, si está presente. La cláusula `<merge_not_matched>` especifica los valores que insertar. La instrucción `MERGE` puede tener solamente una cláusula `WHEN NOT MATCHED`.

WHEN NOT MATCHED BY SOURCE THEN `<merge_matched>`

Especifica que todas las filas de *target_table* que no coinciden con las filas que devuelve `<table_source>` `ON <merge_search_condition>` y que satisfacen alguna condición de búsqueda adicional se actualizan o eliminan según la cláusula `<merge_matched>`.

La instrucción `MERGE` puede tener a lo sumo dos cláusulas `WHEN NOT MATCHED BY SOURCE`. Si se especifican dos cláusulas, la primera debe ir acompañada de una cláusula `AND <clause_search_condition>`. Para una fila determinada, la segunda cláusula `WHEN NOT MATCHED BY SOURCE` se aplica solamente si no se aplica la primera. Si hay dos cláusulas `WHEN NOT MATCHED BY SOURCE`, una debe especificar una acción `UPDATE` y la otra una acción `DELETE`. Solamente se puede hacer referencia a las columnas de la tabla de destino en `<clause_search_condition>`.

Cuando `<table_source>` no devuelve ninguna fila, no se puede tener acceso a las columnas de la tabla de origen. Si la acción de actualización o eliminación especificada en la cláusula `<merge_matched>` hace referencia a las columnas de la tabla de origen, se devuelve el error 207 (nombre de columna no válido). La cláusula `WHEN NOT MATCHED BY SOURCE THEN UPDATE SET TargetTable.Col1 = SourceTable.Col1` puede hacer que la instrucción

genere un error porque **Col1** en la tabla de origen es inaccesible.

AND <clause_search_condition>

Especifica cualquier condición de búsqueda válida. Para obtener más información, vea [Condiciones de búsqueda \(Transact-SQL\)](#).

<table_hint_limited>

Especifica una o más sugerencias de tabla que se aplican en la tabla de destino para cada una de las acciones de inserción, actualización o eliminación que realiza la instrucción MERGE. La palabra clave WITH y los paréntesis son obligatorios.

No se permiten NOLOCK ni READUNCOMMITTED. Para obtener más información acerca de las sugerencias de tabla, vea [Sugerencias de tabla \(Transact-SQL\)](#).

Especificar la sugerencia TABLOCK en una tabla que es el destino de una instrucción INSERT tiene el mismo efecto que especificar la sugerencia TABLOCKX. Se realiza un bloqueo exclusivo en la tabla. Cuando se especifica FORCESEEK, se aplica a la instancia implícita de la tabla de destino combinada con la tabla de origen.

Advertencia

Si se especifica READPAST con WHEN NOT MATCHED [BY TARGET] THEN INSERT, pueden producirse operaciones INSERT que infrinjan las restricciones UNIQUE.

INDEX (index_val [,...n])

Especifica el nombre o identificador de uno o más índices de la tabla de destino para realizar una combinación implícita con la tabla de origen. Para obtener más información, vea [Sugerencias de tabla \(Transact-SQL\)](#).

<output_clause>

Devuelve una fila para cada fila de *target_table* que se actualiza, inserta o elimina, en ningún orden en concreto.

\$action se puede especificar en la cláusula de salida. **\$action** es una columna de tipo **nvarchar(10)** que devuelve uno de estos tres valores por cada fila: 'INSERT', 'UPDATE' o 'DELETE', según la acción realizada en dicha fila. Para obtener más información acerca de los argumentos de esta cláusula, vea [OUTPUT \(cláusula de Transact-SQL\)](#).

OPTION (<query_hint> [,...n])

Especifica que se utilizan las sugerencias del optimizador para personalizar el modo en que el motor de base de datos procesa la instrucción. Para obtener más información, vea [Sugerencias de consulta \(Transact-SQL\)](#).

<merge_matched>

Especifica la acción de actualización o eliminación que se aplica a todas las filas de *target_table* que no coinciden con las filas que devuelve <table_source> ON <merge_search_condition>, y que satisfacen cualquier condición de búsqueda adicional.

UPDATE SET <set_clause>

Especifica la lista de nombres de columna o de variable que se van a actualizar en la tabla de destino y los valores con los que se actualizan.

Para obtener más información acerca de los argumentos de esta cláusula, vea [UPDATE \(Transact-SQL\)](#). No se puede establecer una variable con el mismo valor que una columna.

DELETE

Especifica que las filas coincidentes de las filas de *target_table* se eliminan.

<merge_not_matched>

Especifica los valores que insertar en la tabla de destino.

(column list)

Es una lista de una o varias columnas de la tabla de destino en la que insertar los datos. Las columnas se deben especificar como un nombre de una sola parte o, de lo contrario, se producirá un error en la instrucción MERGE. *column_list* se debe agregar entre paréntesis y delimitarse mediante comas.

VALUES (*values_list*)

Es una lista separada por comas de constantes, variables o expresiones que devuelve los valores que se insertarán en la tabla de destino. Las expresiones no pueden contener una instrucción EXECUTE.

DEFAULT VALUES

Hace que la fila insertada contenga los valores predeterminados definidos para cada columna.

Para obtener más información sobre esta cláusula, vea [INSERT \(Transact-SQL\)](#).

<search condition>

Especifica las condiciones de búsqueda utilizadas para especificar <merge_search_condition> o <clause_search_condition>. Para obtener más información acerca de los argumentos de esta cláusula, vea [Condiciones de búsqueda \(Transact-SQL\)](#).

Comentarios

Al menos se debe especificar una de las tres cláusulas MATCHED, pero se pueden especificar en cualquier orden. Una variable no puede actualizarse más de una vez en la misma cláusula MATCHED.

Cualquier acción de inserción, actualización o eliminación especificada en la tabla de destino por la instrucción MERGE está limitada por las restricciones definidas en ella, incluidas las restricciones de integridad referencial en cascada. Si IGNORE_DUP_KEY se establece en ON para algún índice único de la tabla de destino, MERGE omite este valor.

La instrucción MERGE requiere un punto y coma (;) como terminador. Se genera el error 10713 cuando una instrucción MERGE se ejecuta sin el terminador.

Cuando se utiliza después de MERGE, @@ROWCOUNT ([Transact-SQL](#)) devuelve el número total de filas insertadas, actualizadas y eliminadas al cliente.

MERGE es una palabra clave totalmente reservada cuando el nivel de compatibilidad de la base de datos se establece en 100. La instrucción MERGE también está disponible en los niveles de compatibilidad 90 y 100 de la base de datos; sin embargo, la palabra clave no se reserva completamente cuando el nivel de compatibilidad se establece en 90.

La instrucción **MERGE** no se debe usar cuando se emplea la replicación de actualización en cola. **MERGE** y el desencadenador de actualización en cola no son compatibles. Reemplace la instrucción **MERGE** con una instrucción de inserción o de actualización.

Implementación de desencadenadores

Para cada acción de inserción, actualización o eliminación especificada en la instrucción MERGE, SQL Server activa los desencadenadores AFTER correspondientes definidos en la tabla de destino, pero no garantiza qué acción activará los desencadenadores primero o último. Los desencadenadores definidos para la misma acción cumplen el orden que especifique. Para obtener más información sobre cómo establecer el orden de activación de los desencadenadores, vea [Especificar el primer y el último desencadenador](#).

Si la tabla de destino tiene habilitado un desencadenador INSTEAD OF definido en ella para una acción de inserción, actualización o eliminación realizada por una instrucción MERGE, debe tener habilitado un desencadenador INSTEAD OF para todas las acciones especificadas en la instrucción MERGE.

Si hay desencadenadores INSTEAD OF UPDATE o INSTEAD OF DELETE definidos en *target_table*, las operaciones de actualización o eliminación no se realizan. En su lugar, se activan los desencadenadores y las tablas **inserted** y **deleted**

se rellenan en consecuencia.

Si hay definidos desencadenadores `INSTED OF INSERT` en *target_table*, la operación de inserción no se realiza. En su lugar, se activan los desencadenadores y la tabla **inserted** se rellena en consecuencia.

Permisos

Requiere el permiso `SELECT` en la tabla de origen y los permisos `INSERT`, `UPDATE` o `DELETE` en la tabla de destino. Para obtener información adicional, consulte la sección Permisos de los temas [SELECT](#), [INSERT](#), [UPDATE](#) o [DELETE](#).

Ejemplos

A. Usar MERGE para realizar operaciones INSERT y UPDATE en una tabla en una sola instrucción

Un escenario común es la actualización de una o varias columnas de una tabla si una fila coincidente existe, o la inserción de datos como una fila nueva si no existe ninguna fila coincidente. Normalmente, para hacer esto se pasan los parámetros a un procedimiento almacenado que contiene las instrucciones `INSERT` y `UPDATE` adecuadas. Con la instrucción `MERGE` puede realizar ambas tareas en una sola instrucción. En el ejemplo siguiente se muestra un procedimiento almacenado que contiene una instrucción `INSERT` y una instrucción `UPDATE`. A continuación, el procedimiento se modifica para realizar las operaciones equivalentes utilizando una sola instrucción `MERGE`.

Transact-SQL

```
USE AdventureWorks2012;
GO
CREATE PROCEDURE dbo.InsertUnitMeasure
    @UnitMeasureCode nchar(3),
    @Name nvarchar(25)
AS
BEGIN
    SET NOCOUNT ON;
    -- Update the row if it exists.
    UPDATE Production.UnitMeasure
        SET Name = @Name
        WHERE UnitMeasureCode = @UnitMeasureCode
    -- Insert the row if the UPDATE statement failed.
    IF (@@ROWCOUNT = 0 )
    BEGIN
        INSERT INTO Production.UnitMeasure (UnitMeasureCode, Name)
        VALUES (@UnitMeasureCode, @Name)
    END
END;
GO
-- Test the procedure and return the results.
EXEC InsertUnitMeasure @UnitMeasureCode = 'ABC', @Name = 'Test Value';
SELECT UnitMeasureCode, Name FROM Production.UnitMeasure
WHERE UnitMeasureCode = 'ABC';
GO

-- Rewrite the procedure to perform the same operations using the MERGE statement.
-- Create a temporary table to hold the updated or inserted values from the OUTPUT clause.
CREATE TABLE #MyTempTable
    (ExistingCode nchar(3),
    ExistingName nvarchar(50),
    ExistingDate datetime)
```

```

        existingdate datetime,
        ActionTaken nvarchar(10),
        NewCode nchar(3),
        NewName nvarchar(50),
        NewDate datetime
    );
GO
ALTER PROCEDURE dbo.InsertUnitMeasure
    @UnitMeasureCode nchar(3),
    @Name nvarchar(25)
AS
BEGIN
    SET NOCOUNT ON;

    MERGE Production.UnitMeasure AS target
    USING (SELECT @UnitMeasureCode, @Name) AS source (UnitMeasureCode, Name)
    ON (target.UnitMeasureCode = source.UnitMeasureCode)
    WHEN MATCHED THEN
        UPDATE SET Name = source.Name
    WHEN NOT MATCHED THEN
        INSERT (UnitMeasureCode, Name)
        VALUES (source.UnitMeasureCode, source.Name)
        OUTPUT deleted.*, $action, inserted.* INTO #MyTempTable;

END;
GO
-- Test the procedure and return the results.
EXEC InsertUnitMeasure @UnitMeasureCode = 'ABC', @Name = 'New Test Value';
EXEC InsertUnitMeasure @UnitMeasureCode = 'XYZ', @Name = 'Test Value';
EXEC InsertUnitMeasure @UnitMeasureCode = 'ABC', @Name = 'Another Test Value';

SELECT * FROM #MyTempTable;
-- Cleanup
DELETE FROM Production.UnitMeasure WHERE UnitMeasureCode IN ('ABC','XYZ');
DROP TABLE #MyTempTable;
GO

```

B. Usar MERGE para realizar operaciones UPDATE y DELETE en una tabla en una sola instrucción

En el siguiente ejemplo se usa MERGE para actualizar diariamente la tabla **ProductInventory** de la base de datos de ejemplo **AdventureWorks**, en función de los pedidos procesados en la tabla **SalesOrderDetail**. La columna **Quantity** de la tabla **ProductInventory** se actualiza restando el número de pedidos realizados cada día para cada producto de la tabla **SalesOrderDetail**. Si el número de pedidos de un producto baja el nivel de inventario del mismo hasta 0 o un valor menor, la fila correspondiente a ese producto se elimina de la tabla **ProductInventory**.

Transact-SQL

```

USE AdventureWorks2012;
GO
IF OBJECT_ID (N'Production.usp_UpdateInventory', N'P') IS NOT NULL DROP PROCEDURE Production.u:
GO
CREATE PROCEDURE Production.usp_UpdateInventory
    @OrderDate datetime
AS
MERGE Production.ProductInventory AS target
USING (SELECT ProductID, SUM(OrderQty) FROM Sales.SalesOrderDetail AS sod
        JOIN Sales.SalesOrderHeader AS soh
        ON sod.SalesOrderID = soh.SalesOrderID
        AND soh.OrderDate = @OrderDate
        GROUP BY ProductID) AS source (ProductID, OrderQty)

```



```

ON (target.ProductID = source.ProductID)
WHEN MATCHED AND target.Quantity - source.OrderQty <= 0
    THEN DELETE
WHEN MATCHED
    THEN UPDATE SET target.Quantity = target.Quantity - source.OrderQty,
                    target.ModifiedDate = GETDATE()
OUTPUT $action, Inserted.ProductID, Inserted.Quantity, Inserted.ModifiedDate, Deleted.ProductID,
        Deleted.Quantity, Deleted.ModifiedDate;
GO

EXECUTE Production.usp_UpdateInventory '20030501'

```

C. Usar MERGE para realizar operaciones INSERT y UPDATE en una tabla de destino mediante una tabla de origen derivada

En el ejemplo siguiente se usa MERGE para modificar la tabla **SalesReason**, actualizando o insertando las filas. Cuando el valor de **NewName** de la tabla de origen coincide con un valor de la columna **Name** de la tabla de destino, (**SalesReason**), la columna **ReasonType** se actualiza en la tabla de destino. Cuando el valor de **NewName** no coincide, la fila del origen se inserta en la tabla de destino. La tabla de origen es una tabla derivada que usa la característica de constructor con valores de tabla de Transact-SQL para especificar varias filas en la tabla de origen. Para obtener más información acerca de cómo usar el constructor de valores de tabla en una tabla derivada, vea [Constructor con valores de tabla \(Transact-SQL\)](#). El ejemplo también muestra cómo almacenar los resultados de la cláusula OUTPUT en una variable de tabla y, a continuación, resumir los resultados de la instrucción MERGE realizando una sencilla operación SELECT que devuelve el recuento de las filas insertadas y actualizadas.

Transact-SQL

```

USE AdventureWorks2012;
GO
-- Create a temporary table variable to hold the output actions.
DECLARE @SummaryOfChanges TABLE(Change VARCHAR(20));

MERGE INTO Sales.SalesReason AS Target
USING (VALUES ('Recommendation','Other'), ('Review', 'Marketing'), ('Internet', 'Promotion'))
    AS Source (NewName, NewReasonType)
ON Target.Name = Source.NewName
WHEN MATCHED THEN
    UPDATE SET ReasonType = Source.NewReasonType
WHEN NOT MATCHED BY TARGET THEN
    INSERT (Name, ReasonType) VALUES (NewName, NewReasonType)
OUTPUT $action INTO @SummaryOfChanges;

-- Query the results of the table variable.
SELECT Change, COUNT(*) AS CountPerChange
FROM @SummaryOfChanges
GROUP BY Change;


```

D. Insertar los resultados de la instrucción MERGE en otra tabla

En el ejemplo siguiente se capturan los datos devueltos por la cláusula OUTPUT de una instrucción MERGE y se insertan en otra tabla. La instrucción MERGE actualiza la columna **Quantity** de la tabla **ProductInventory**, en función de los pedidos procesados en la tabla **SalesOrderDetail**. En el ejemplo se capturan las filas actualizadas y se insertan en otra tabla que se usa para realizar el seguimiento de los cambios del inventario.

Transact-SQL

```
USE AdventureWorks2012;
GO
CREATE TABLE Production.UpdatedInventory
    (ProductID INT NOT NULL, LocationID int, NewQty int, PreviousQty int,
    CONSTRAINT PK_Inventory PRIMARY KEY CLUSTERED (ProductID, LocationID));
GO
INSERT INTO Production.UpdatedInventory
SELECT ProductID, LocationID, NewQty, PreviousQty
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 BETWEEN '20030701' AND '20030731'
        GROUP BY ProductID) AS src (ProductID, OrderQty)
    ON pi.ProductID = src.ProductID
    WHEN MATCHED AND pi.Quantity - src.OrderQty >= 0
        THEN UPDATE SET pi.Quantity = pi.Quantity - src.OrderQty
    WHEN MATCHED AND pi.Quantity - src.OrderQty <= 0
        THEN DELETE
    OUTPUT $action, Inserted.ProductID, Inserted.LocationID, Inserted.Quantity AS NewQty, Deleted.ProductID, Deleted.LocationID, Deleted.Quantity AS PreviousQty
    AS Changes (Action, ProductID, LocationID, NewQty, PreviousQty) WHERE Action = 'UPDATE';
GO
```



Vea también

Adiciones de comunidad
