SQL Programming

Módulo 4 - Operaciones DML

Operaciones DML

Cláusula INSERT

Agrega una o varias filas a una tabla o una vista en SQL Server.

Limitaciones y restricciones

Cuando se insertan valores en tablas remotas y no se especifican todos los valores de todas las columnas, debe identificar las columnas en las que se deben insertar los valores especificados.

Cuando se utiliza TOP con INSERT las filas a las que hace referencia no están organizadas de ninguna manera y la cláusula ORDER BY no se puede especificar directamente en esta instrucción. Si necesita usar TOP para insertar las filas en un orden cronológico significativo, debe utilizar TOP junto con una cláusula ORDER BY que se especifica en una instrucción de subselección. Vea la sección Ejemplos que aparece más adelante en este tema.

Cláusula INSERT

Las consultas INSERT en las que se usa SELECT con ORDER BY para rellenar filas garantizan el modo en que se calculan los valores de identidad, pero no el orden en el que las filas se insertan.

En Almacenamiento de datos paralelos, la cláusula ORDER BY no es válida en VISTAS, CREATE TABLE AS SELECT, INSERT SELECT, funciones insertadas, tablas derivadas, subconsultas ni expresiones de tabla común, salvo que se especifique también TOP.

Para los siguientes ejemplos contamos con la siguiente tabla.

Sintaxis

```
CREATE TABLE dbo.Sectores(

SectorID TINYINT NOT NULL IDENTITY(1,1),
Gerencia VARCHAR(50),
Sector VARCHAR(50) DEFAULT 'Sin Sector'
);
```

Cláusula INSERT

Insertar datos en una tabla con columnas que tienen valores predeterminados

En el ejemplo siguiente se muestra la inserción de filas en una tabla con columnas que generan automáticamente un valor o tienen un valor predeterminado.

La instrucción INSERT inserta filas que contienen valores para algunas de las columnas, pero no para todas. En la última instrucción INSERT, no se especifica ninguna columna y solamente se insertan los valores predeterminados con la cláusula DEFAULT VALUES.

Sintaxis

```
INSERT INTO dbo.Sectores DEFAULT VALUES;

SELECT * FROM dbo.Sectores;
-------------------------------
SectorID    Gerencia    Sector
1            NULL      Sin Sector
```

Cláusula INSERT

Insertar una sola fila de datos

En el siguiente ejemplo se inserta una fila. Dado que los valores para todas las columnas se suministran e incluyen en el mismo orden que las columnas de la tabla, no es necesario especificar los nombres de columna en la lista de columnas .

Sintaxis

```
INSERT INTO dbo.Sectores VALUES ('Finanzas',
'Contaduria');

SELECT * FROM dbo.Sectores;
-----------------------------------
SectorID    Gerencia      Sector
1            NULL       Sin Sector
2            Finanzas   Contaduria
```

Cláusula INSERT

Insertar varias filas de datos

En el siguiente ejemplo se usa el constructor de valores de tabla para insertar dos filas en la tabla dbo.Sectores en una sola instrucción INSERT. Dado que los valores para todas las columnas se suministran e incluyen en el mismo orden que las columnas de la tabla, no es necesario especificar los nombres de columna en la lista de columnas.

Sintaxis

```
INSERT INTO dbo.Sectores VALUES
('Finanzas','Cobranzas'), ('Finanzas','Ventas');

SELECT * FROM dbo.Sectores;
--------------------------------
SectorID    Gerencia     Sector
1            NULL       Sin   Sector
2            Finanzas   Contaduria
3            Finanzas   Cobranzas
4            Finanzas   Ventas
```

Cláusula INSERT

Insertar datos que no están en el mismo orden que las columnas de la tabla

En el siguiente ejemplo se utiliza una lista de columnas para especificar de forma explícita los valores insertados en cada columna.

El orden de las columnas de la tabla Sectores.es, Gerencia, Sector; no obstante, las columnas no se incluyen en dicho orden.

Sintaxis

```
INSERT INTO dbo.Sectores (Sector, Gerencia) VALUES
('Legales', 'Finanzas');

SELECT * FROM dbo.Sectores;
------------------------------
SectorID    Gerencia      Sector
1            NULL       Sin Sector
2            Finanzas   Contaduria
3            Finanzas   Cobranzas
4            Finanzas   Ventas
5            Finanzas   Legales
```

Cláusula INSERT

Insertar datos en una tabla con una columna de identidad

En el siguiente ejemplo se muestran los distintos métodos para insertar datos en una columna de identidad.

Las dos primeras instrucciones INSERT permiten generar valores de identidad para las filas nuevas.

La tercera instrucción INSERT invalida la propiedad IDENTITY de la columna con la instrucción SET IDENTITY_INSERT e inserta un valor explícito en la columna de identidad.

Sintaxis

```
INSERT INTO dbo.Sectores (Gerencia, Sector) VALUES ('Finanzas', 'Ventas');
SET IDENTITY_INSERT dbo.Sectores ON;
INSERT INTO dbo.Sectores (SectorID, Gerencia, Sector) VALUES
(100, 'Recursos Humanos', DEFAULT);

SELECT * FROM  dbo.Sectores;

SET IDENTITY_INSERT dbo.Sectores OFF;
--------------------------------------------
SectorID       Gerencia                    Sector
1               NULL                Sin Sector
2               Finanzas                   Contaduria
3               Finanzas                   Cobranzas
4               Finanzas                   Ventas
5               Finanzas                   Legales
6               Produccion              Manufactura
7               Finanzas                   Ventas
100             Recursos Humanos       Sin Sector
```

Cláusula INSERT

Insertar datos en una tabla con el resultado de una consulta

En el siguiente ejemplo se inserta una nueva Gerencia y Sector proveniente de la tabla HumanResources.Department.

Sintaxis

```
INSERT INTO dbo.Sectores (Gerencia, Sector)

SELECT Name, GroupName FROM HumanResources.Department WHERE
DepartmentID = 1;

SELECT * FROM dbo.Sectores
-------------------------------------------------
SectorID     Gerencia                Sector
1            NULL                   Sin Sector
2            Finanzas                Contaduria
3            Finanzas                Cobranzas
4            Finanzas                Ventas
5            Finanzas                Legales
6            Produccion              Manufactura
7            Finanzas                Legales
100          Recursos Humanos         Sin Sector
101          Engineering             Research and Development
```

Cláusula UPDATE

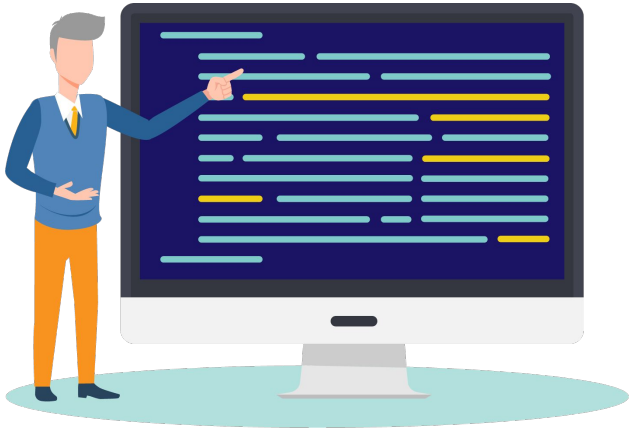
Cambia los datos de una tabla o vista.

Usar una instrucción UPDATE simple

En el ejemplo siguiente se actualiza un solo valor de columna para todas las filas de la tabla Person.Address.

Sintaxis

```
UPDATE Person.Address
SET ModifiedDate = GETDATE();
```



Cláusula UPDATE

Especificar un valor calculado

En los siguientes ejemplos se usan valores calculados en una instrucción UPDATE.

En el ejemplo se duplica el valor de la columna ListPrice para todas las filas de la tabla Product.

Sintaxis

```
UPDATE Production.Product
SET ListPrice = ListPrice * 2;
```

Actualizar varias columnas

En el siguiente ejemplo se actualizan los valores de las columnas Bonus, CommissionPct y SalesQuota para todas las filas de la tabla SalesPerson.

Sintaxis

```
UPDATE Sales.SalesPerson
SET Bonus = 6000, CommissionPct = 1.10, SalesQuota = NULL;
```

Cláusula UPDATE

Usar la cláusula WHERE

En el ejemplo siguiente se utiliza la cláusula WHERE para especificar las filas que se van a actualizar.

La instrucción actualiza el valor de la columna Color de la tabla Production.Product para todas las filas con un valor existente de 'Red' en la columna Color y con un valor que comience por 'Road-250' en la columna Name.

Sintaxis

```
UPDATE Production.Product
SET Color = 'Metallic Red'
WHERE Name LIKE N'Road-250%' AND Color = 'Red';
```

Cláusula UPDATE

Actualizar las filas con valores DEFAULT

En el siguiente ejemplo se establece la columna CostRate en su valor predeterminado (0.00) para todas las filas que tengan un valor de CostRate mayor que 20.00.

Sintaxis

```
UPDATE Production.Location
SET CostRate = DEFAULT
WHERE CostRate > 20.00;
```

Usar la cláusula TOP

En los siguientes ejemplos use la cláusula TOP para limitar el número de filas que se modifican en una instrucción UPDATE. Cuando se usa una cláusula TOP (n) con UPDATE, la operación de actualización se realiza en una selección aleatoria de un número de filas n. En el ejemplo siguiente se actualiza un 25 por ciento la columna VacationHours en 10 filas aleatorias de la tabla Employee.

Sintaxis

```
UPDATE TOP (10) HumanResources.Employee
SET VacationHours = VacationHours * 1.25;
```

Cláusula UPDATE

Si debe usar TOP para aplicar actualizaciones por orden cronológico, debe utilizarse junto con ORDER BY en una instrucción de subselección. En el siguiente ejemplo se actualizan las horas de vacaciones de los 10 empleados cuyas fechas de alta son más antiguas.

Sintaxis

```
UPDATE HumanResources.Employee
SET VacationHours = VacationHours + 8
FROM (SELECT TOP 10 BusinessEntityID FROM
HumanResources.Employee
     ORDER BY HireDate ASC) AS th
WHERE HumanResources.Employee.BusinessEntityID =
th.BusinessEntityID;
```

Para los siguientes ejemplos contamos con la siguiente tabla.

Sintaxis

```
CREATE TABLE dbo.SectoresNuevo(

Sector VARCHAR(50),
SectorNuevo VARCHAR(50)
);


INSERT INTO dbo.SectoresNuevo (Sector, SectorNuevo)
VALUES ('Contaduria', 'Tesoreria');
```

Cláusula UPDATE

Actualizar las filas con valores de otra tabla

En el siguiente ejemplo modifica el valor de la columna sector de la tabla sectores tomando el valor del campo SectorNuevo de la tabla Sectores nuevo.

Sintaxis

```
UPDATE s
       SET Sector=sn.SectorNuevo
FROM
       Sectores s
       INNER JOIN SectoresNuevo sn
       ON S.Sector=sn.Sector

SELECT * FROM dbo.Sectores;
-----------------------------------------
SectorID     Gerencia                Sector
1            NULL                   Sin Sector
2            Finanzas                Tesoreria
3            Finanzas               Cobranzas
4            Finanzas               Ventas
5            Finanzas               Legales
6            Produccion             Manufactura
7            Finanzas               Legales
100          Recursos Humanos        Sin Sector
101          Engineering            Research and Development
```

Cláusula DELETE

Quita una o varias filas de una tabla o vista de SQL Server

Procedimientos recomendados

Para eliminar todas las filas de una tabla, use TRUNCATE TABLE. TRUNCATE TABLE es más rápido que DELETE y utiliza menos recursos de los registros de transacciones y de sistema. TRUNCATE TABLE tiene restricciones; por ejemplo, la tabla no puede participar en la replicación.

Use la función @@ROWCOUNT para devolver el número de filas eliminadas a la aplicación cliente. Para más información.

Utilizar DELETE sin la cláusula WHERE

En el ejemplo siguiente se eliminan todas las filas de la tabla Sales.SalesPerson porque no se utiliza una cláusula WHERE para limitar el número de filas eliminadas.

Sintaxis

```
DELETE FROM Sales.SalesPersonQuotaHistory;
```

Cláusula DELETE

Usar la cláusula WHERE para eliminar un conjunto de filas

En el ejemplo siguiente se eliminan todas las filas de la tabla ProductCostHistory de la base de datos AdventureWorks2012 en las que el valor de la columna StandardCost es superior a 1000.00.

Sintaxis

```
DELETE FROM Production.ProductCostHistory
WHERE StandardCost > 1000.00;
```

En el siguiente ejemplo se muestra una cláusula WHERE más compleja. La cláusula WHERE define dos condiciones que deben cumplirse para determinar las filas que se van a eliminar.

El valor de la columna StandardCost debe estar comprendido entre 12.00 y 14.00 y el valor de la columna EndDate debe ser NULL.

Sintaxis

```
DELETE Production.ProductCostHistory
WHERE StandardCost BETWEEN 12.00 AND 14.00 AND
EndDate IS NULL;
```

Cláusula DELETE

Usar la cláusula Join para eliminar un conjunto de filas.

En el ejemplo siguiente se eliminan todas las filas donde el sector exista en las entidades Sectores y SectorNuevo.

Sintaxis

```
DELETE s

FROM dbo.Sectores s
INNER JOIN dbo.SectoresNuevo sn
ON S.Sector=sn.Sector;

SELECT * FROM dbo.Sectores;
-----------------------------------
SectorID     Gerencia                Sector
1              NULL           Sin Sector
3              Finanzas              Cobranzas
4              Finanzas              Ventas
5              Finanzas              Legales
6              Produccion      Manufactura
7              Finanzas              Legales
100            Recursos Humanos    Sin Sector
101            Engineering     Research and Development
```

TRUNCATE TABLE

Quita todas las filas de una tabla o las particiones especificadas de una tabla, sin registrar las eliminaciones individuales de filas.

TRUNCATE TABLE es similar a la instrucción DELETE sin una cláusula WHERE; no obstante, TRUNCATE TABLE es más rápida y utiliza menos recursos de registros de transacciones y de sistema.

TRUNCATE TABLE

Restricciones

- No puede utilizar TRUNCATE TABLE en las siguientes tablas:
- Tablas a las que se hace referencia mediante una restricción FOREIGN KEY. (Puede truncar una tabla que tenga una clave externa que haga referencia a sí misma).
- Tablas que participan en una vista indizada.
- Tablas que se publican mediante replicación transaccional o replicación de mezcla.
- En el caso de las tablas con una o más de estas características, utilice la instrucción DELETE.
- TRUNCATE TABLE no puede activar un desencadenador porque la operación no registra eliminaciones de filas individuales. Para obtener más información, En Almacenamiento de datos SQL de Azure y Almacenamiento de datos paralelos:
- TRUNCATE TABLE no se permite dentro de la instrucción EXPLAIN.
- TRUNCATE TABLE no se puede ejecutar dentro de una transacción.

TRUNCATE TABLE

Truncar tablas de gran tamaño

Microsoft SQL Server ofrece la posibilidad de quitar o truncar las tablas con más de 128 extensiones sin mantener bloqueos simultáneos en todas las extensiones necesarias para la eliminación.

Truncar una tabla

En el siguiente ejemplo se quitan todos los datos de la tabla Sectores. Se incluyen instrucciones SELECT antes y después de la instrucción TRUNCATE TABLE para comparar los resultados.

Sintaxis

```
TRUNCATE TABLE dbo.Sectores;
GO
SELECT COUNT(*) AS CantidadRegistros

FROM dbo.Sectores;
------------------------
CantidadRegistros
0
```

¡Muchas gracias!

¡Sigamos trabajando!