

Advanced T-SQL and Troubleshooting

Module 4: Optimizing Actions

Bob Beauchemin
BobB@SQLskills.com



Overview

- **INSERTing Data**
 - INSERT...SELECT, INSERT...EXECUTE, INSERT INTO
- **UPDATEing Data**
 - Update using a JOIN or subquery
- **DELETEing Data**
 - Bulk delete
- **MERGE**
- **TOP, ROWCOUNT and Actions**
- **Output from Action Statements**



2

© SQLskills, All rights reserved.
<http://www.SQLskills.com>



Row Constructors

- Use VALUES clause to construct a set of rows
- Insert multiple rows based on values in a single
- INSERT statement
- SQL 2006 standard compatible

```
DECLARE @t TABLE (id int, name varchar(20));  
INSERT INTO @t VALUES  
    (1, 'Fred'), (2, 'Jim'), (3, 'Sue');
```

Using the INSERT...SELECT Statement

- All rows that satisfy the SELECT statement are inserted
 - Verify that the table that receives new row exists
 - Must ensure that data types are compatible
 - Determine whether default values exist or whether null values are allowed
 - Derived columns, constants, and defaults can be used as input

INSERT...SELECT

```
USE northwind
INSERT customers
SELECT substring (firstname, 1, 3) +
      substring (lastname, 1, 2) AS name
      , lastname, firstname, title
      , address, city, region, postalcode
      , country, homephone, NULL
FROM employees
```

Using INSERT...EXECUTE

- **Allows population from stored procedure**
 - ❑ Number and data type of columns must match
 - ❑ Procedure must return a single rowset
 - ❑ Procedure can be on remote server
 - ❑ Limitations for (N)TEXT
 - ❑ SELECT statement in stored procedure cannot contain CTE
 - ❑ Useful for storing information from some DBCC commands

SELECT INTO

- **SELECT INTO creates a table from resultset**
 - Used for producing intermediate temp table for reshaping data then SELECTing
 - Must have CREATE TABLE privilege, except when creating temp table
 - Can be used to create an empty copy an empty copy of existing table
 - Does not require non-logged mode after SQL Server 7.0

Updating Rows Based on Other Tables

- **Using the UPDATE Statement**
 - Updates a row once
 - Updates columns or variable names that follow the SET keyword
- **Using Joins**
 - Uses the FROM clause
 - Some issues when using this...more later
- **Using Subqueries**
 - Returns a single value for each row

Indirect updates

```
-- UPDATE from a correlated subquery
UPDATE titles
SET YTD_SALES = (SELECT ISNULL(SUM(qty),0)
                 FROM sales AS s
                 WHERE s.title_id = t.title_id)
FROM titles as t
```

```
-- UPDATE from a JOIN
UPDATE od
SET Discount = Discount + 0.02
FROM [Order Details] AS od
JOIN Products AS p ON od.productid = p.productid
WHERE p.SupplierID = 5
```

Deleting Rows

- **Can delete rows using additional tables**
 - Similar to UPDATE
 - Using Additional FROM Clause
 - First FROM clause indicates table to modify
 - Second FROM clause specifies restricting criteria for the DELETE statement
- **Specifying Conditions in the WHERE Clause**
 - Subqueries determine which rows to delete
- **TRUNCATE TABLE is fastest way to delete all rows in a table**
 - But its not logged

MERGE Statement

- **Multiple set operations in a single SQL statement**
- **Uses multiple sets as input**
 - MERGE target USING source ON ...
- **Operations can be INSERT, UPDATE, DELETE**
- **Operations based on**
 - WHEN MATCHED
 - WHEN NOT MATCHED [ON TARGET]
 - WHEN NOT MATCHED ON SOURCE
- **ANSI SQL 2006 compliant - with extensions**

More on MERGE

- **MERGE statement can reference a \$action column**
 - Used when MERGE used with OUTPUT clause
- **Multiple WHEN clauses possible**
 - For MATCHED and NOT MATCHED ON SOURCE
 - Only one can be an update
 - Only one WHEN clause for NOT MATCHED ON TARGET
- **MERGE can be used with any table source**
- **A MERGE statement causes triggers to be fired once**
- **Rows affected includes total rows affected by all clauses**

MERGE Performance

- **MERGE statement is transactional**
 - No explicit transaction required
- **One Pass Through Tables**
 - At most a full outer join
 - Matching rows = when matched
 - Left-outer join rows = when target not matched
 - Right-outer join rows = when source not matched

MERGE and Determinism

- **UPDATE using a JOIN is non-deterministic**
 - If more than one row in source matches ON clause, either/any row can be used for the UPDATE
- **MERGE is deterministic**
 - If more than one row in source matches ON clause,
 - its an error

TOP With INSERT,UPDATE,DELETE

- This is meant to replace SET ROWCOUNT
 - Cannot UPDATE/DELETE the higher/lowest because no ORDER BY in these statements
 - Allow batching with optimization

```
DECLARE @batch_size AS INT
SET @batch_size = 100

WHILE (1=1)
BEGIN
    UPDATE TOP(@batch_size) licenses SET expired = 1
        WHERE expire_date < GETDATE()
            AND expired = 0
    IF @@rowcount < @batch_size BREAK
END
GO
```

DML with Results (OUTPUT Clause)

- New OUTPUT clause which facilitates the return of data by referencing inserted and deleted tables
 - In INSERT statements, refer to INSERTED to retrieve columns from the newly inserted rows
 - In DELETE statements, refer to DELETED to retrieve columns from the deleted rows
 - In UPDATE and MERGE statements, refer to DELETED to return the old image of the updated rows, and to INSERTED to return their new image

DML with Results (OUTPUT Clause)

- **OUTPUT DELETED.*** in the DELETE statement below, returns all columns deleted from the ShoppingCartItem table to the table variable

```
DECLARE @MyTableVar TABLE (...)  
  
-- OK for table or variable  
-- must name columns if column list  
DELETE Sales.ShoppingCartItem  
OUTPUT DELETED.* INTO @MyTableVar  
WHERE ProductID = 862
```

DML with Results (OUTPUT Clause)

- **Restrictions and Requirements**
 - Not supported in any DML statements referencing local or distributed partitioned views, or remote tables
 - Columns returned from OUTPUT reflect the data as it is once the INSERT, UPDATE, or DELETE statement has completed, but before triggers are executed
 - OUTPUT can be used with an UPDATE or DELETE statement within a cursor. (no guarantee on sort order)
 - If parameters or variables are modified in an UPDATE statement, the OUTPUT clause always returns the pre-change value
 - E.g don't use @@IDENTITY, use INSERTED

Review

- INSERT...SELECT and INSERT...EXECUTE
- UPDATE by SELECTing from another table
- DELETE by SELECTing from another table
- MERGE statement
- TOP clause and action statements
- OUTPUT from action statements

Questions?