Module 7

Using DML to Modify Data

Zyead Ahmed.
Aspiring To Learning Data Engineer.

Module Overview

- Adding Data to Tables
- Modifying and Removing Data
- Generating Numbers

Lesson 1: Adding Data to Tables

- Using INSERT to Add Data
- Using INSERT with SELECT and EXEC
- Using SELECT INTO
- Demonstration: Inserting Data Into Tables

Using INSERT to Add Data

- The INSERT...VALUES statement inserts a single row by default

```
INSERT INTO Sales.OrderDetails(
orderid, productid, unitprice, qty, discount)
VALUES(12000,39,18,2,0.05);
```

- Table and row constructors add multi-row capability to INSERT...VALUES

```
INSERT INTO Sales.OrderDetails(
orderid, productid, unitprice, qty, discount)
VALUES
        (12001,39,18,2,0.05),
        (12002,39,18,5,0.10);
```

Using INSERT with DEFAULT Constraints

- DEFAULT constraints are used to assign a value to a column when none is specified in INSERT statement
- Defaults defined in CREATE TABLE statement
- INSERT statement can omit columns which have defaults defined
  - Applies to IDENTITY and NULL too
- VALUES clause can use DEFAULT keyword
  - If no default constraint assigned, NULL inserted

```
INSERT INTO Sales.OrderDetails
   (orderid, productid, unitprice, qty, discount)
VALUES (11077, 72, 34.80, DEFAULT, DEFAULT);
```

Using INSERT with SELECT and EXEC

- INSERT...SELECT is used to insert the result set of a query into an existing table

```
INSERT INTO Sales.OrderHist(
        orderid,custid,empid,orderdate)
SELECT orderid,custid,empid,orderdate
FROM Sales.Orders
WHERE orderdate < '20080101';
```

- INSERT...EXEC is used to insert the result of a stored procedure or dynamic SQL expression into an existing table

```
INSERT INTO dbo.T1 (productid, productname, unitprice)
EXEC Production.ProdsByCategory
        @numrows = 5, @catid=1;
```

Using SELECT INTO

- SELECT...INTO is similar to INSERT...SELECT but SELECT...INTO creates a new table each time the statement is executed
- Copies column names, data types, and nullability
- Does not copy constraints or indexes

```
SELECT orderid, custid, empid, orderdate, shippeddate
INTO Sales.OrderArchive
FROM Sales.Orders
WHERE orderdate < '20080101';
```

Demonstration: Inserting Data Into Tables

In this demonstration, you will see how to:

- Insert rows into tables

Lesson 2: Modifying and Removing Data

- Using UPDATE to Modify Data
- Using MERGE to Modify Data
- Using DELETE to Remove Data
- Using TRUNCATE TABLE to Remove Data
- Demonstration: Modifying and Removing Data From Tables

Using UPDATE to Modify Data

- Updates all rows in a table or view
  - Set can be filtered with a WHERE clause
  - Set can be defined with a JOIN clause
- Only columns specified in the SET clause are modified

```
UPDATE Production.Products
   SET unitprice = (unitprice * 1.04)
WHERE categoryid =  1 AND discontinued = 0;
```

Using MERGE to Modify Data

- MERGE modifies data based on a condition
  - When the source matches the target
  - When the source has no match in the target
  - When the target has no match in the source

```
MERGE INTO schema_name.table_name AS TargetTbl
    USING (SELECT <select_list>) AS SourceTbl
    ON (TargetTbl.col1 = SourceTbl.col1)
    WHEN MATCHED THEN
        UPDATE SET col2 = SourceTbl.col2
WHEN NOT MATCHED THEN
    INSERT (<column_list>)
    VALUES (<value_list>);
```

Using DELETE to Remove Data

- DELETE without a WHERE clause deletes all rows

```
DELETE FROM dbo.Nums;
```

- Use a WHERE clause to delete specific rows

```
DELETE FROM Sales.OrderDetails
WHERE orderid = 10248;
```

Using TRUNCATE TABLE to Remove Data

- TRUNCATE TABLE clears the entire table
  - Storage physically deallocated, rows not individually removed
- Minimally logged
  - Can be rolled back if TRUNCATE issued within a transaction
- TRUNCATE TABLE will fail if the table is referenced by a foreign key constraint in another table

```
TRUNCATE TABLE dbo.Nums;
```

Demonstration: Modifying and Removing Data From Tables

In this demonstration, you will see how to:

- Update and delete data in a table

Lesson 3: Generating Numbers

- Using IDENTITY
- Using Sequences

Using IDENTITY

- IDENTITY property of a column generates sequential numbers automatically for insertion into a table
  - Can specify optional seed and increment values
- Only one column in a table may have IDENTITY property defined
  - IDENTITY column omitted in INSERT statements
- Functions provided to return last generated values

```
CREATE TABLE Production.Products(
productid int IDENTITY(1,1) NOT NULL,
productname nvarchar(40) NOT NULL,
categoryid int NOT NULL,
unitprice money NOT NULL)
```

Using Sequences

- Sequence objects added in SQL Server 2012
- Independent objects in database
  - More flexible than the IDENTITY property
  - Can be used as default value for a column
- Manage with CREATE/ALTER/DROP statements
- Retrieve value with the NEXT VALUE FOR clause

```
-- Define a sequence
CREATE SEQUENCE dbo.InvoiceSeq AS INT START WITH 1
INCREMENT BY 1;
-- Retrieve next available value from sequence
SELECT NEXT VALUE FOR dbo.InvoiceSeq;
```

Lab: Using DML to Modify Data

- Exercise 1: Inserting Records with DML
- Exercise 2: Update and Delete Records Using DML

Logon Information
Virtual machine: **20461C-MIA-SQL**
User name: **ADVENTUREWORKS\Student**
Password: **Pa$$w0rd**

Estimated Time: 30 Minutes

Lab Scenario

- You are a database developer for Adventure Works and need to create DML statements to update data in the database to support the website development team. The team need T-SQL statements that they can use to carry out updates to data, based on actions performed on the website. You will supply DML statements that they can modify to their specific requirements.

Module Review and Takeaways

- Review Question(s)