10 | Programming with Transact-SQL

Graeme Malcolm | Senior Content Developer, Microsoft
Geoff Allix | Principal Technologist, Content Master

Module Overview

- Batches
- Comments
- Variables
- Conditional Branching
- Loops
- Stored Procedures

Batches

- Batches are sets of commands sent to SQL Server as a unit
- Batches determine variable scope, name resolution
- To separate statements into batches, use a separator:
  - SQL Server tools use the GO keyword
  - GO is not a T-SQL command!
  - GO [count] executes the batch the specified number of times

```
SELECT * FROM Production.Product;
SELECT * FROM Sales.Customer;
GO
```

Comments

- Marks T-SQL code as a comment:
  - For a block, enclose it between **/*** and ***/** characters

```
/*
        This is a block
    of commented code
*/
```

  - For inline text, precede the comments with **--**

```
--   This line of text will be ignored
```

- T-SQL Editors typically color-code comments, as shown above

Variables

- Variables are objects that allow storage of a value for use later in the same batch
- Variables are defined with the DECLARE keyword
  - Variables can be declared and initialized in the same statement
- Variables are always local to the batch in which they're declared and go out of scope when the batch ends

```
--Declare and initialize variables
DECLARE @color nvarchar(15)='Black', @size nvarchar(5)='L';
--Use variables in statements
SELECT *
  FROM Production.Product
  WHERE Color=@color and Size=@size;
GO
```

DEMO

Using Variables

Conditional Branching

- IF…ELSE uses a predicate to determine the flow of the code
  - The code in the IF block is executed if the predicate evaluates to TRUE
  - The code in the ELSE block is executed if the predicate evaluates to FALSE or UNKNOWN

```
IF @color IS NULL
     SELECT * FROM Production.Product;
ELSE
     SELECT * FROM Production.Product
      WHERE Color = @color;
```

DEMO

Using IF…ELSE

Looping

- WHILE enables code to execute in a loop
- Statements in the WHILE block repeat as the predicate evaluates to TRUE
- The loop ends when the predicate evaluates to FALSE or UNKNOWN
- Execution can be altered by BREAK or CONTINUE

```
DECLARE @custid AS INT = 1, @lname AS NVARCHAR(20);
WHILE @custid <=5
   BEGIN
      SELECT @lname = lastname FROM Sales.Customer
      WHERE customerid = @custid;
      PRINT @lname;
      SET @custid += 1;
   END;
```

DEMO

Demo: Using WHILE

Stored Procedures

Similar to Function??

- Database objects that encapsulate Transact-SQL code
- Can be parameterized
  - Input parameters
  - Output parameters

Default Value

```
CREATE PROCEDURE SalesLT.GetProductsByCategory (@CategoryID INT = NULL)
AS
IF @CategoryID IS NULL
    SELECT ProductID, Name, Color, Size, ListPrice
    FROM SalesLT.Product
ELSE
    SELECT ProductID, Name, Color, Size, ListPrice
    FROM SalesLT.Product
    WHERE ProductCategoryID = @CategoryID;
```

- Executed with the EXECUTE command

```
EXECUTE SalesLT.GetProductsByCategory 6;
```

DEMO

Creating a Stored Procedure

Programming with Transact-SQL

- Batches
- Comments
- Variables
- Conditional Branching
- Loops
- Stored Procedure

- Lab: Programming with Transact-SQL

Microsoft