WHILE (Transact-SQL)

THIS TOPIC APPLIES TO: ✓ SQL Server (starting with 2008) ✓ Azure SQL Database ✓ Azure SQL Data Warehouse ✓ Parallel Data Warehouse

Sets a condition for the repeated execution of an SQL statement or statement block. The statements are executed repeatedly as long as the specified condition is true. The execution of statements in the WHILE loop can be controlled from inside the loop with the BREAK and CONTINUE keywords.

Transact-SQL Syntax Conventions

Syntax

```
WHILE Boolean_expression
     { sql_statement | statement_block | BREAK | CONTINUE }
```

```
-- Azure SQL Data Warehouse and Parallel Data Warehouse
WHILE Boolean_expression
     { sql_statement | statement_block | BREAK }
```

Arguments

Boolean_expression
Is an expression that returns **TRUE** or **FALSE**. If the Boolean expression contains a SELECT statement, the SELECT statement must be enclosed in parentheses.

{*sql_statement* | *statement_block*}
Is any Transact-SQL statement or statement grouping as defined with a statement block. To define a statement block, use the control-of-flow keywords BEGIN and END.

BREAK
Causes an exit from the innermost WHILE loop. Any statements that appear after the END keyword, marking the end of the loop, are executed.

CONTINUE
Causes the WHILE loop to restart, ignoring any statements after the CONTINUE keyword.

Remarks

If two or more WHILE loops are nested, the inner BREAK exits to the next outermost loop. All the statements after the end of the inner loop run first, and then the next outermost loop restarts.

Examples

A. Using BREAK and CONTINUE with nested IF...ELSE and WHILE

In the following example, if the average list price of a product is less than `$300`, the `WHILE` loop doubles the prices and then selects the maximum price. If the maximum price is less than or equal to `$500`, the `WHILE` loop restarts and doubles the prices again. This loop continues doubling the prices until the maximum price is greater than `$500`, and then exits the `WHILE` loop and prints a message.

```
USE AdventureWorks2012;
GO
WHILE (SELECT AVG(ListPrice) FROM Production.Product) < $300
BEGIN
   UPDATE Production.Product
      SET ListPrice = ListPrice * 2
   SELECT MAX(ListPrice) FROM Production.Product
   IF (SELECT MAX(ListPrice) FROM Production.Product) > $500
      BREAK
   ELSE
      CONTINUE
END
PRINT 'Too much for the market to bear';
```

B. Using WHILE in a cursor

The following example uses `@@FETCH_STATUS` to control cursor activities in a `WHILE` loop.

```
DECLARE Employee_Cursor CURSOR FOR
SELECT EmployeeID, Title
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle = 'Marketing Specialist';
OPEN Employee_Cursor;
FETCH NEXT FROM Employee_Cursor;
WHILE @@FETCH_STATUS = 0
   BEGIN
      FETCH NEXT FROM Employee_Cursor;
   END;
CLOSE Employee_Cursor;
DEALLOCATE Employee_Cursor;
```

```
GO
```

Examples: Azure SQL Data Warehouse Public Preview and Parallel Data Warehouse

C: Simple While Loop

In the following example, if the average list price of a product is less than `$300`, the `WHILE` loop doubles the prices and then selects the maximum price. If the maximum price is less than or equal to `$500`, the `WHILE` loop restarts and doubles the prices again. This loop continues doubling the prices until the maximum price is greater than `$500`, and then exits the `WHILE` loop.

```
-- Uses AdventureWorks

WHILE ( SELECT AVG(ListPrice) FROM dbo.DimProduct) < $300
BEGIN
    UPDATE dbo.DimProduct
        SET ListPrice = ListPrice * 2;
    SELECT MAX ( ListPrice) FROM dbo.DimProduct
    IF ( SELECT MAX (ListPrice) FROM dbo.DimProduct) > $500
        BREAK;
END
```

See Also

ALTER TRIGGER (Transact-SQL)
Control-of-Flow Language (Transact-SQL)
CREATE TRIGGER (Transact-SQL)
Cursors (Transact-SQL)
SELECT (Transact-SQL)

Community Additions