SQL Server BREAK statement overview

In the previous tutorial, you have learned how to use the [WHILE](https://www.sqlservertutorial.net/sql-server-stored-procedures/sql-server-while/) statement to create a loop. To exit the current iteration of a loop, you use the BREAK statement.

The following illustrates the typical syntax of the BREAK statement:

WHILE Boolean\_expression

BEGIN

-- statements

IF condition

BREAK;

-- other statements

END

Code language: SQL (Structured Query Language) (sql)

In this syntax, the BREAK statement exit the WHILE loop immediately once the condition  specified in the [IF](https://www.sqlservertutorial.net/sql-server-stored-procedures/sql-server-if-else/) statement is met. All the statements between the BREAK and END keywords are skipped.

Suppose we have a WHILE loop nested inside another WHILE loop:

WHILE Boolean\_expression1

BEGIN

-- statement

WHILE Boolean\_expression2

BEGIN

IF condition

BREAK;

END

END

Code language: SQL (Structured Query Language) (sql)

In this case, the BREAK statement only exits the innermost loop in the WHILE statement.

Note that the BREAK statement can be used only inside the WHILE loop. The IF statement is often used with the BREAK statement but it is not required.

SQL Server BREAK statement example

The following example illustrates how to use the BREAK statement:

DECLARE @counter INT = 0;

WHILE @counter <= 5

BEGIN

SET @counter = @counter + 1;

IF @counter = 4

BREAK;

PRINT @counter;

END

Code language: SQL (Structured Query Language) (sql)

Output:

1

2

3

In this example:

First, we [declared a variable](https://www.sqlservertutorial.net/sql-server-stored-procedures/variables/) named @counter and set its value to zero.

Then, we used the WHILE statement to increases the @counter by one in each iteration and print out the @counter‘s value as long as the value of the @counter is less than or equal to five.

Inside the loop, we also checked if the value of @counter equals four, then we exited the loop. In the fourth iteration, the value of the counter reached 4, then the loop is terminated. Also, the PRINT statement after the BREAK statement was skipped.

## Introduction to the SQL Server CONTINUE statement

The CONTINUE statement stops the current iteration of the loop and starts the new one. The following illustrates the syntax of the CONTINUE statement:

WHILE Boolean\_expression

BEGIN

-- code to be executed

IF condition

CONTINUE;

-- code will be skipped if the condition is met

END

Code language: SQL (Structured Query Language) (sql)

In this syntax, the current iteration of the loop is stopped once the condition evaluates to TRUE. The next iteration of the loop will continue until the Boolean\_expression evaluates to FALSE.

Similar to the [BREAK](https://www.sqlservertutorial.net/sql-server-stored-procedures/sql-server-break/) statement, the CONTINUE statement is often used in conjunction with an [IF](https://www.sqlservertutorial.net/sql-server-stored-procedures/sql-server-if-else/) statement. Note that this is not mandatory though.

## SQL Server CONTINUE example

The following example illustrates how the CONTINUE statement works.

DECLARE @counter INT = 0;

WHILE @counter <= 5

BEGIN

SET @counter = @counter + 1;

IF @counter = 3

CONTINUE;

PRINT @counter;

END

Code language: SQL (Structured Query Language) (sql)

Here is the output:

1

2

4

5

Code language: SQL (Structured Query Language) (sql)

In this example:

* First, we [declared a variable](https://www.sqlservertutorial.net/sql-server-stored-procedures/variables/) named @counter and set its value to zero.
* Then, the [WHILE](https://www.sqlservertutorial.net/sql-server-stored-procedures/sql-server-while/) loop started. Inside the WHILE loop, we increased the counter by one in each iteration. If the @counter was three, we skipped printing out the value using the CONTINUE statement. That’s why in the output, you do not see the number three is showing up.