**UNION (Transact-SQL)**

**SQL Server 2012**

[Other Versions](javascript:;)

Description: http://i.technet.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [SQL Server 2008 R2](http://technet.microsoft.com/en-us/library/ms180026(d=printer,v=sql.105).aspx)
* [SQL Server 2008](http://technet.microsoft.com/en-us/library/ms180026(d=printer,v=sql.100).aspx)
* [SQL Server 2005](http://technet.microsoft.com/en-us/library/ms180026(d=printer,v=sql.90).aspx)

Combines the results of two or more queries into a single result set that includes all the rows that belong to all queries in the union. The UNION operation is different from using joins that combine columns from two tables.

The following are basic rules for combining the result sets of two queries by using UNION:

* The number and the order of the columns must be the same in all queries.
* The data types must be compatible.

[Transact-SQL Syntax Conventions](http://technet.microsoft.com/en-us/library/ms177563.aspx)

[Syntax](javascript:void(0))

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_7f853951-6464-47b9-ae20-af5719482877');" \o "Copy to clipboard.)

    { <query\_specification> | ( <query\_expression> ) }

  UNION [ ALL ]

  <query\_specification | ( <query\_expression> )

[ UNION [ ALL ] <query\_specification> | ( <query\_expression> )

    [ ...n ] ]

[Arguments](javascript:void(0))

<query\_specification> | ( <query\_expression> )

Is a query specification or query expression that returns data to be combined with the data from another query specification or query expression. The definitions of the columns that are part of a UNION operation do not have to be the same, but they must be compatible through implicit conversion. When data types differ, the resulting data type is determined based on the rules for [data type precedence](http://technet.microsoft.com/en-us/library/ms190309.aspx). When the types are the same but differ in precision, scale, or length, the result is determined based on the same rules for combining expressions. For more information, see [Precision, Scale, and Length (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms190476.aspx).

Columns of the xml data type must be equivalent. All columns must be either typed to an XML schema or untyped. If typed, they must be typed to the same XML schema collection.

UNION

Specifies that multiple result sets are to be combined and returned as a single result set.

ALL

Incorporates all rows into the results. This includes duplicates. If not specified, duplicate rows are removed.

[Examples](javascript:void(0))

**A. Using a simple UNION**

In the following example, the result set includes the contents of the ProductModelID and Name columns of both the ProductModel and Gloves tables.

Transact-SQL

USE AdventureWorks2012;

GO

IF OBJECT\_ID ('dbo.Gloves', 'U') IS NOT NULL

DROP TABLE dbo.Gloves;

GO

-- Create Gloves table.

SELECT ProductModelID, Name

INTO dbo.Gloves

FROM Production.ProductModel

WHERE ProductModelID IN (3, 4);

GO

-- Here is the simple union.

USE AdventureWorks2012;

GO

SELECT ProductModelID, Name

FROM Production.ProductModel

WHERE ProductModelID NOT IN (3, 4)

UNION

SELECT ProductModelID, Name

FROM dbo.Gloves

ORDER BY Name;

GO

**B. Using SELECT INTO with UNION**

In the following example, the INTO clause in the second SELECT statement specifies that the table named ProductResults holds the final result set of the union of the designated columns of the ProductModel and Gloves tables. Note that the Gloves table is created in the first SELECT statement.

Transact-SQL

USE AdventureWorks2012;

GO

IF OBJECT\_ID ('dbo.ProductResults', 'U') IS NOT NULL

DROP TABLE dbo.ProductResults;

GO

IF OBJECT\_ID ('dbo.Gloves', 'U') IS NOT NULL

DROP TABLE dbo.Gloves;

GO

-- Create Gloves table.

SELECT ProductModelID, Name

INTO dbo.Gloves

FROM Production.ProductModel

WHERE ProductModelID IN (3, 4);

GO

USE AdventureWorks2012;

GO

SELECT ProductModelID, Name

INTO dbo.ProductResults

FROM Production.ProductModel

WHERE ProductModelID NOT IN (3, 4)

UNION

SELECT ProductModelID, Name

FROM dbo.Gloves;

GO

SELECT ProductModelID, Name

FROM dbo.ProductResults;

**C. Using UNION of two SELECT statements with ORDER BY**

The order of certain parameters used with the UNION clause is important. The following example shows the incorrect and correct use of UNION in two SELECT statements in which a column is to be renamed in the output.

Transact-SQL

USE AdventureWorks2012;

GO

IF OBJECT\_ID ('dbo.Gloves', 'U') IS NOT NULL

DROP TABLE dbo.Gloves;

GO

-- Create Gloves table.

SELECT ProductModelID, Name

INTO dbo.Gloves

FROM Production.ProductModel

WHERE ProductModelID IN (3, 4);

GO

/\* INCORRECT \*/

USE AdventureWorks2012;

GO

SELECT ProductModelID, Name

FROM Production.ProductModel

WHERE ProductModelID NOT IN (3, 4)

ORDER BY Name

UNION

SELECT ProductModelID, Name

FROM dbo.Gloves;

GO

/\* CORRECT \*/

USE AdventureWorks2012;

GO

SELECT ProductModelID, Name

FROM Production.ProductModel

WHERE ProductModelID NOT IN (3, 4)

UNION

SELECT ProductModelID, Name

FROM dbo.Gloves

ORDER BY Name;

GO

**D. Using UNION of three SELECT statements to show the effects of ALL and parentheses**

The following examples use UNION to combine the results of three tables that all have the same 5 rows of data. The first example uses UNION ALL to show the duplicated records, and returns all 15 rows. The second example uses UNION without ALL to eliminate the duplicate rows from the combined results of the three SELECT statements, and returns 5 rows.

The third example uses ALL with the first UNION and parentheses enclose the second UNION that is not using ALL. The second UNION is processed first because it is in parentheses, and returns 5 rows because the ALL option is not used and the duplicates are removed. These 5 rows are combined with the results of the first SELECT by using the UNION ALL keywords. This does not remove the duplicates between the two sets of 5 rows. The final result has 10 rows.

Transact-SQL

USE AdventureWorks2012;

GO

IF OBJECT\_ID ('dbo.EmployeeOne', 'U') IS NOT NULL

DROP TABLE dbo.EmployeeOne;

GO

IF OBJECT\_ID ('dbo.EmployeeTwo', 'U') IS NOT NULL

DROP TABLE dbo.EmployeeTwo;

GO

IF OBJECT\_ID ('dbo.EmployeeThree', 'U') IS NOT NULL

DROP TABLE dbo.EmployeeThree;

GO

SELECT pp.LastName, pp.FirstName, e.JobTitle

INTO dbo.EmployeeOne

FROM Person.Person AS pp JOIN HumanResources.Employee AS e

ON e.BusinessEntityID = pp.BusinessEntityID

WHERE LastName = 'Johnson';

GO

SELECT pp.LastName, pp.FirstName, e.JobTitle

INTO dbo.EmployeeTwo

FROM Person.Person AS pp JOIN HumanResources.Employee AS e

ON e.BusinessEntityID = pp.BusinessEntityID

WHERE LastName = 'Johnson';

GO

SELECT pp.LastName, pp.FirstName, e.JobTitle

INTO dbo.EmployeeThree

FROM Person.Person AS pp JOIN HumanResources.Employee AS e

ON e.BusinessEntityID = pp.BusinessEntityID

WHERE LastName = 'Johnson';

GO

-- Union ALL

SELECT LastName, FirstName, JobTitle

FROM dbo.EmployeeOne

UNION ALL

SELECT LastName, FirstName ,JobTitle

FROM dbo.EmployeeTwo

UNION ALL

SELECT LastName, FirstName,JobTitle

FROM dbo.EmployeeThree;

GO

SELECT LastName, FirstName,JobTitle

FROM dbo.EmployeeOne

UNION

SELECT LastName, FirstName, JobTitle

FROM dbo.EmployeeTwo

UNION

SELECT LastName, FirstName, JobTitle

FROM dbo.EmployeeThree;

GO

SELECT LastName, FirstName,JobTitle

FROM dbo.EmployeeOne

UNION ALL

(

SELECT LastName, FirstName, JobTitle

FROM dbo.EmployeeTwo

UNION

SELECT LastName, FirstName, JobTitle

FROM dbo.EmployeeThree

);

GO