Table of contents

TOP (Transact-SQL)

Article

01/13/2023

11 minutes to read

18 contributors Feedback

In this article

Applies to: SQL Server Azure SQL Database Azure SQL Managed Instance Azure Synapse Analytics Analytics Platform System (PDW)

Limits the rows returned in a query result set to a specified number of rows or percentage of rows in SQL Server. When you use TOP with the ORDER BY clause, the result set is limited to the first N number of ordered rows. Otherwise, TOP returns the first N number of rows in an undefined order. Use this clause to specify the number of rows returned from a SELECT statement. Or, use TOP to specify the rows affected by an INSERT, UPDATE, MERGE, or DELETE statement.

Transact-SQL syntax conventions

Syntax

Following is the syntax for SQL Server and Azure SQL Database:

[TOP (expression) [PERCENT] [WITH TIES]]

Following is syntax for Azure Synapse Analytics and Analytics Platform System (PDW):

[TOP (expression) [WITH TIES]]

Note To view Transact-SQL syntax for SQL Server 2014 and earlier, see Previous versions documentation.

Arguments

expression

The numeric expression that specifies the number of rows to be returned. expression is implicitly converted to a float value if you specify PERCENT. Otherwise, expression is converted to bigint.

PERCENT

P

Indicates that the query returns only the first expression percent of rows from the result set. Fractional values are rounded up to the next integer value.

WITH TIES

Returns two or more rows that tie for last place in the limited results set. You must use this argument with the ORDER BY clause. WITH TIES might cause more rows to be returned than the value specified in expression. For example, if expression is set to 5 but two additional rows match the values of the ORDER BY columns in row 5, the result set will contain seven rows.

You can specify the TOP clause with the WITH TIES argument only in SELECT statements, and only if you've also specified the ORDER BY clause. The returned order of tying records is arbitrary. ORDER BY doesn't affect this rule.

Best Practices

In a SELECT statement, always use an ORDER BY clause with the TOP clause. Because, it's the only way to predictably indicate which rows are affected by TOP.

Use OFFSET and FETCH in the ORDER BY clause instead of the TOP clause to implement a query paging solution. A paging solution (that is, sending chunks or "pages" of data to the client) is easier to implement using OFFSET and FETCH clauses. For more information, see ORDER BY Clause (Transact-SQL).

Use TOP (or OFFSET and FETCH) instead of SET ROWCOUNT to limit the number of rows returned. These methods are preferred over using SET ROWCOUNT for the following reasons:

As a part of a SELECT statement, the query optimizer can consider the value of expression in the TOP or FETCH clauses during query optimization. Because you use SET ROWCOUNT outside of a statement that runs a query, its value can't be considered in a query plan.

Compatibility Support

For backward compatibility, the parentheses are optional in SELECT statements if the expression is an integer constant. We recommend that you always use parentheses for TOP in SELECT statements. Doing so provides consistency with its required use in INSERT, UPDATE, MERGE, and DELETE statements.

Interoperability

The TOP expression doesn't affect statements that might run because of a trigger. The inserted and deleted tables in the triggers return only the rows that are truly affected by the INSERT, UPDATE, MERGE, or DELETE statements. For example, if an INSERT TRIGGER fires as the result of an INSERT statement that used a TOP clause.

SQL Server allows for updating rows through views. Because you can include the TOP clause in the view definition, certain rows may disappear from the view if the rows no longer meet the requirements of the TOP expression due to an update.

When specified in the MERGE statement, the TOP clause applies after the entire source table and the entire target table are joined. And, the joined rows that don't qualify for an insert, update, or delete action are removed. The TOP clause further reduces the number of joined rows to the specified value and the insert, update, or delete actions apply to the remaining joined rows in an unordered way. That is, there's no order in which the rows are distributed among the actions defined in the WHEN clauses. For example, if specifying TOP (10) affects 10 rows, of these rows, seven may be updated and three inserted. Or, one may be deleted, five updated, and four inserted, and so on. Because the MERGE statement does a full table scan of both the source and target tables, I/O performance can be affected when you use the TOP clause to modify a large table by creating multiple batches. In this scenario, it's important to ensure that all successive batches target new rows.

Use caution when you're specifying the TOP clause in a query that contains a UNION, UNION ALL, EXCEPT, or INTERSECT operator. It's possible to write a query that returns unexpected results because the order in which the TOP and ORDER BY clauses are logically processed isn't always intuitive when these operators are used in a select operation. For example, given the following table and data, assume that you want to return the least expensive red car and the least expensive blue car. That is, the red sedan and the blue van.

CREATE TABLE dbo.Cars(Model VARCHAR(15), Price MONEY, Color VARCHAR(10)); INSERT dbo.Cars VALUES ('sedan', 10000,

'red'), ('convertible', 15000, 'blue'), ('coupe', 20000, 'red'), ('van', 8000, 'blue');

To achieve these results, you might write the following query.

SELECT TOP(1) Model, Color, Price FROM dbo.Cars WHERE Color = 'red' UNION ALL SELECT TOP(1) Model, Color, Price FROM dbo.Cars WHERE Color = 'blue' ORDER BY Price ASC; GO

Following is the result set.

Model Color Price ------------- ---------- ------- sedan red 10000.00 convertible blue 15000.00

The unexpected results are returned because the TOP clause logically runs before the ORDER BY clause, which sorts the results of the operator (UNION ALL in this case). So, the previous query returns any one red car and any one blue car and then orders the result of that union by the price. The following example shows the correct method of writing this query to achieve the desired result.

SELECT Model, Color, Price FROM (SELECT TOP(1) Model, Color, Price FROM dbo.Cars WHERE Color = 'red' ORDER BY Price ASC) AS a UNION ALL SELECT Model, Color, Price FROM (SELECT TOP(1) Model, Color, Price FROM dbo.Cars WHERE Color = 'blue' ORDER BY Price ASC) AS b; GO

By using TOP and ORDER BY in a subselect operation, you ensure that the results of the ORDER BY clause are applied to the TOP clause and not to sorting the result of the UNION operation.

Here is the result set.

Model Color Price ------------- ---------- ------- sedan red 10000.00 van blue 8000.00

Limitations and Restrictions

When you use TOP with INSERT, UPDATE, MERGE, or DELETE, the referenced rows aren't arranged in any order. And, you can't directly specify the ORDER BY clause in these statements. If you need to use TOP to insert, delete, or modify

rows in a meaningful chronological order, use TOP with an ORDER BY clause specified in a subselect statement. See the following Examples section in this article.

You can't use TOP in an UPDATE and DELETE statements on partitioned views.

You can't combine TOP with OFFSET and FETCH in the same query expression (in the same query scope). For more information, see ORDER BY Clause (Transact-SQL).

Examples

Basic syntax

Examples in this section demonstrate the basic functionality of the ORDER BY clause using the minimum required syntax.

A. Using TOP with a constant value

The following examples use a constant value to specify the number of employees that are returned in the query result set. In the first example, the first 10 undefined rows are returned because an ORDER BY clause isn't used. In the second example, an ORDER BY clause is used to return the top 10 recently hired employees.

USE AdventureWorks2012; GO -- Select the first 10 random employees. SELECT TOP(10)JobTitle, HireDate FROM HumanResources.Employee; GO -- Select the first 10 employees hired most recently. SELECT TOP(10)JobTitle, HireDate FROM HumanResources.Employee ORDER BY HireDate DESC; GO

B. Using TOP with a variable

The following example uses a variable to specify the number of employees that are returned in the query result set.

USE AdventureWorks2012; GO DECLARE @p AS INT = 10; SELECT TOP(@p)JobTitle, HireDate, VacationHours FROM HumanResources.Employee ORDER BY VacationHours DESC; GO

C. Specifying a percentage

The following example uses PERCENT to specify the number of employees that are returned in the query result set. There are 290 employees in the HumanResources.Employee table. Because five percent of 290 is a fractional value, the value is rounded up to the next whole number.

USE AdventureWorks2012; GO SELECT TOP(5)PERCENT JobTitle, HireDate FROM HumanResources.Employee ORDER BY HireDate DESC; GO

Including tie values

A. Using WITH TIES to include rows that match the values in the last row

The following example gets the top 10 percent of all employees with the highest salary and returns them in descending order according to their salary. Specifying WITH TIES ensures that employees with salaries equal to the lowest salary returned (the last row) are also included in the result set, even if it exceeds 10 percent of employees.

USE AdventureWorks2012; GO SELECT TOP(10) PERCENT WITH TIES pp.FirstName, pp.LastName, e.JobTitle, e.Gender, r.Rate FROM Person.Person AS pp INNER JOIN HumanResources.Employee AS e ON pp.BusinessEntityID = e.BusinessEntityID INNER JOIN HumanResources.EmployeePayHistory AS r ON r.BusinessEntityID = e.BusinessEntityID ORDER BY Rate DESC; GO

Limiting the rows affected by DELETE, INSERT, or UPDATE

A. Using TOP to limit the number of rows deleted

When you use a TOP (n) clause with DELETE, the delete operation is done on an undefined selection of n number of rows. That is, the DELETE statement chooses any (n) number of rows that meet the criteria defined in the WHERE clause. The following example deletes 20 rows from the PurchaseOrderDetail table that have due dates earlier than July 1, 2002.

USE AdventureWorks2012; GO DELETE TOP (20) FROM Purchasing.PurchaseOrderDetail WHERE DueDate

If you want to use TOP to delete rows in a meaningful chronological order, use TOP with ORDER BY in a subselect statement. The following query deletes the 10 rows of the PurchaseOrderDetail table that have the earliest due dates. To ensure that only 10 rows are deleted, the column specified in the subselect statement (PurchaseOrderID) is the primary key of the table. Using a nonkey column in the subselect statement may result in the deletion of more than 10 rows if the specified column contains duplicate values.

USE AdventureWorks2012; GO DELETE FROM Purchasing.PurchaseOrderDetail WHERE PurchaseOrderDetailID IN (SELECT TOP 10 PurchaseOrderDetailID FROM Purchasing.PurchaseOrderDetail ORDER BY DueDate ASC); GO

B. Using TOP to limit the number of rows inserted

The following example creates the table EmployeeSales and inserts the name and year-to-date sales data for the top five employees from the table HumanResources.Employee . The INSERT statement chooses any five rows returned by the SELECT statement that meet the criteria defined in the WHERE clause. The OUTPUT clause displays the rows that are inserted into the EmployeeSales table. Notice that the ORDER BY clause in the SELECT statement isn't used to determine the top five employees.

USE AdventureWorks2012 ; GO IF OBJECT_ID ('dbo.EmployeeSales', 'U') IS NOT NULL DROP TABLE dbo.EmployeeSales; GO CREATE TABLE dbo.EmployeeSales (EmployeeID NVARCHAR(11) NOT NULL, LastName NVARCHAR(20) NOT NULL, FirstName NVARCHAR(20) NOT NULL, YearlySales MONEY NOT NULL); GO INSERT TOP(5)INTO dbo.EmployeeSales OUTPUT inserted.EmployeeID, inserted.FirstName, inserted.LastName, inserted.YearlySales SELECT sp.BusinessEntityID, c.LastName, c.FirstName, sp.SalesYTD FROM Sales.SalesPerson AS sp INNER JOIN Person.Person AS c ON sp.BusinessEntityID = c.BusinessEntityID WHERE sp.SalesYTD > 250000.00 ORDER BY sp.SalesYTD DESC; GO

If you want to use TOP to insert rows in a meaningful chronological order, use TOP with ORDER BY in a subselect statement. The following example show how to do this. The OUTPUT clause displays the rows that are inserted into the EmployeeSales table. Notice that the top five employees are now inserted based on the results of the ORDER BY clause instead of undefined rows.

INSERT INTO dbo.EmployeeSales OUTPUT inserted.EmployeeID, inserted.FirstName, inserted.LastName, inserted.YearlySales SELECT TOP (5) sp.BusinessEntityID, c.LastName, c.FirstName, sp.SalesYTD FROM Sales.SalesPerson AS sp INNER JOIN

Person.Person AS c ON sp.BusinessEntityID = c.BusinessEntityID WHERE sp.SalesYTD > 250000.00 ORDER BY sp.SalesYTD DESC; GO

The following example uses the TOP clause to update rows in a table. When you use a TOP (n) clause with UPDATE, the update operation runs on an undefined number of rows. That is, the UPDATE statement chooses any (n) number of rows that meet the criteria defined in the WHERE clause. The following example assigns 10 customers from one salesperson to another.

USE AdventureWorks2012; UPDATE TOP (10) Sales.Store SET SalesPersonID = 276 WHERE SalesPersonID = 275; GO

If you have to use TOP to apply updates in a meaningful chronology, you must use TOP together with ORDER BY in a subselect statement. The following example updates the vacation hours of the 10 employees with the earliest hire dates.

UPDATE HumanResources.Employee SET VacationHours = VacationHours + 8 FROM (SELECT TOP 10 BusinessEntityID FROM HumanResources.Employee ORDER BY HireDate ASC) AS th WHERE HumanResources.Employee.BusinessEntityID = th.BusinessEntityID; GO

Examples: Azure Synapse Analytics and Analytics Platform System (PDW)

The following example returns the top 31 rows that match the query criteria. The ORDER BY clause ensures that the 31 returned rows are the first 31 rows based on an alphabetical ordering of the LastName column.

Using TOP without specifying ties.

SELECT TOP (31) FirstName, LastName FROM DimEmployee ORDER BY LastName;

Result: 31 rows are returned.

Using TOP, specifying WITH TIES.

```
SELECT TOP (31) WITH TIES FirstName, LastName FROM DimEmployee ORDER BY LastName;
```

Result: 33 rows are returned, because three employees named Brown tie for the 31st row.

See Also

SELECT (Transact-SQL)

INSERT (Transact-SQL)

UPDATE (Transact-SQL)

DELETE (Transact-SQL)

ORDER BY Clause (Transact-SQL)

SET ROWCOUNT (Transact-SQL)

MERGE (Transact-SQL)

Reference

[Regulating Human Research: IRBs from Peer Review to Compliance Bureaucracy]

[Journal of the Rosacea Research & Development Institute: Volume 1 Number 1, 2010]