# UPDATE STATISTICS (Transact-SQL)

01/04/2018 • 8 minutes to read • Contributors

### In this article

Syntax
Arguments
Remarks
When to Use UPDATE STATISTICS
Limitations and Restrictions
Updating All Statistics with sp_updatestats
Determining the Last Statistics Update
PDW / SQL Data Warehouse
Permissions
Examples
Examples: Azure SQL Data Warehouse and Parallel Data Warehouse
See Also

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

Updates query optimization statistics on a table or indexed view. By default, the query optimizer already updates statistics as necessary to improve the query plan; in some cases you can improve query performance by using UPDATE STATISTICS or the stored procedure sp_updatestats to update statistics more frequently than the default updates.

Updating statistics ensures that queries compile with up-to-date statistics. However, updating statistics causes queries to recompile. We recommend not updating statistics too frequently because there is a performance tradeoff between improving query plans and the time it takes to recompile queries. The specific tradeoffs depend on your application. UPDATE STATISTICS can use tempdb to sort the sample of rows for building statistics.

Transact-SQL Syntax Conventions

FEEDBACK

# Syntax

Copy

```
-- Syntax for SQL Server and Azure SQL Database

UPDATE STATISTICS table_or_indexed_view_name
    [
        {
            { index_or_statistics__name }
          | ( { index_or_statistics_name } [ ,...n ] )
                }
    ]
    [    WITH
        [
            FULLSCAN
              [ [ , ] PERSIST_SAMPLE_PERCENT = { ON | OFF } ]
            | SAMPLE number { PERCENT | ROWS }
              [ [ , ] PERSIST_SAMPLE_PERCENT = { ON | OFF } ]
            | RESAMPLE
              [ ON PARTITIONS ( { <partition_number> | <range> } [, …n] ) ]
            | <update_stats_stream_option> [ ,...n ]
        ]
        [ [ , ] [ ALL | COLUMNS | INDEX ]
        [ [ , ] NORECOMPUTE ]
        [ [ , ] INCREMENTAL = { ON | OFF } ]
        [ [ , ] MAXDOP = max_degree_of_parallelism ]
    ] ;

<update_stats_stream_option> ::=
    [ STATS_STREAM = stats_stream ]
    [ ROWCOUNT = numeric_constant ]
    [ PAGECOUNT = numeric_contant ]
```

Copy

```
-- Syntax for Azure SQL Data Warehouse and Parallel Data Warehouse

UPDATE STATISTICS schema_name . ] table_name
    [ ( { statistics_name | index_name } ) ]
    [ WITH
       {
              FULLSCAN
            | SAMPLE number PERCENT
            | RESAMPLE
        }
    ]
[;]
```

# Arguments

*table_or_indexed_view_name*
Is the name of the table or indexed view that contains the statistics object.

*index_or_statistics_name*
Is the name of the index to update statistics on or name of the statistics to update. If *index_or_statistics_name* is not specified, the query optimizer updates all statistics for the table or indexed view. This includes statistics created using the CREATE STATISTICS statement, single-column statistics created when AUTO_CREATE_STATISTICS is on, and statistics created for indexes.

For more information about AUTO_CREATE_STATISTICS, see ALTER DATABASE SET Options (Transact-SQL). To view all indexes for a table or view, you can use sp_helpindex.

FULLSCAN
Compute statistics by scanning all rows in the table or indexed view. FULLSCAN and SAMPLE 100 PERCENT have the same results. FULLSCAN cannot be used with the SAMPLE option.

SAMPLE *number* { PERCENT | ROWS }
Specifies the approximate percentage or number of rows in the table or indexed view for the query optimizer to use when it updates statistics. For PERCENT, *number* can be from 0 through 100 and for ROWS, *number* can be from 0 to the total number of rows. The actual percentage or number of rows the query optimizer samples might not match the percentage or number specified. For example, the query optimizer scans all rows on a data page.

SAMPLE is useful for special cases in which the query plan, based on default sampling, is not optimal. In most situations, it is not necessary to specify SAMPLE because the query optimizer uses sampling and determines the statistically significant sample size by default, as required to create high-quality query plans.

Starting with SQL Server 2016, sampling of data to build statistics is done in parallel, when using compatibility level 130, to improve the performance of statistics collection. The query optimizer will use parallel sample statistics, whenever a table size exceeds a certain threshold.

SAMPLE cannot be used with the FULLSCAN option. When neither SAMPLE nor FULLSCAN is specified, the query optimizer uses sampled data and computes the sample size by default.

We recommend against specifying 0 PERCENT or 0 ROWS. When 0 PERCENT or ROWS is specified, the statistics object is updated but does not contain statistics data.

For most workloads, a full scan is not required, and default sampling is adequate.
However, certain workloads that are sensitive to widely varying data distributions may require an increased sample size, or even a full scan.
For more information, see the CSS SQL Escalation Services blog.

RESAMPLE
Update each statistic using its most recent sample rate.

Using RESAMPLE can result in a full-table scan. For example, statistics for indexes use a full-table scan for their sample rate. When none of the sample options (SAMPLE, FULLSCAN, RESAMPLE) are specified, the query optimizer samples the data and computes the sample size by default.

PERSIST_SAMPLE_PERCENT = { ON | OFF }
When **ON**, the statistics will retain the set sampling percentage for subsequent updates that do not explicitly specify a sampling percentage. When **OFF**, statistics sampling percentage will get reset to default sampling in subsequent updates that do not explicitly specify a sampling percentage. The default is **OFF**.

> **Note**
>
> If AUTO_UPDATE_STATISTICS is executed, it uses the persisted sampling percentage if available, or use default sampling percentage if not. RESAMPLE behavior is not affected by this option.

> **Tip**
>
> DBCC SHOW_STATISTICS and sys.dm_db_stats_properties expose the persisted sample percent value for the selected statistic.

**Applies to**: SQL Server 2016 (starting with SQL Server 2016 SP1 CU4) through SQL Server 2017 (starting with SQL Server 2017 CU1).

ON PARTITIONS ( { <partition_number> | <range> } [, ...n] ) ] Forces the leaf-level statistics covering the partitions specified in the ON PARTITIONS clause to be recomputed, and then merged to build the global statistics. WITH RESAMPLE is required because partition statistics built with different sample rates cannot be merged together.

**Applies to**: SQL Server 2014 through SQL Server 2017

ALL | COLUMNS | INDEX
Update all existing statistics, statistics created on one or more columns, or statistics created for indexes. If none of the options are specified, the UPDATE STATISTICS statement updates all statistics on the table or indexed view.

NORECOMPUTE
Disable the automatic statistics update option, AUTO_UPDATE_STATISTICS, for the specified statistics. If this option is specified, the query optimizer completes this statistics update and disables future updates.

To re-enable the AUTO_UPDATE_STATISTICS option behavior, run UPDATE STATISTICS again without the NORECOMPUTE option or run **sp_autostats**.

> **Warning**

> Using this option can produce suboptimal query plans. We recommend using this option sparingly, and then only by a qualified system administrator.

For more information about the AUTO_STATISTICS_UPDATE option, see [ALTER DATABASE SET Options (Transact-SQL)](#).

INCREMENTAL = { ON | OFF }
When **ON**, the statistics are recreated as per partition statistics. When **OFF**, the statistics tree is dropped and SQL Server re-computes the statistics. The default is **OFF**.

If per partition statistics are not supported an error is generated. Incremental stats are not supported for following statistics types:

- Statistics created with indexes that are not partition-aligned with the base table.
- Statistics created on Always On readable secondary databases.
- Statistics created on read-only databases.
- Statistics created on filtered indexes.
- Statistics created on views.
- Statistics created on internal tables.
- Statistics created with spatial indexes or XML indexes.

**Applies to**: SQL Server 2014 through SQL Server 2017

MAXDOP = *max_degree_of_parallelism*
**Applies to**: SQL Server (Starting with SQL Server 2017 CU3).

Overrides the **max degree of parallelism** configuration option for the duration of the statistic operation. For more information, see [Configure the max degree of parallelism Server Configuration Option](#). Use MAXDOP to limit the number of processors used in a parallel plan execution. The maximum is 64 processors.

*max_degree_of_parallelism* can be:

1
Suppresses parallel plan generation.

>1
Restricts the maximum number of processors used in a parallel statistic operation to the specified number or fewer based on the current system workload.

0 (default)
Uses the actual number of processors or fewer based on the current system workload.

<update_stats_stream_option> Identified for informational purposes only. Not supported. Future compatibility is not guaranteed.

# Remarks

## When to Use UPDATE STATISTICS

For more information about when to use UPDATE STATISTICS, see Statistics.

## Limitations and Restrictions

- Updating statistics is not supported on external tables. To update statistics on an external table, drop and re-create the statistics.
- The MAXDOP option is not compatible with STATS_STREAM, ROWCOUNT and PAGECOUNT options.

## Updating All Statistics with sp_updatestats

For information about how to update statistics for all user-defined and internal tables in the database, see the stored procedure sp_updatestats (Transact-SQL). For example, the following command calls sp_updatestats to update all statistics for the database.

SQL Copy

```
EXEC sp_updatestats;
```

## Determining the Last Statistics Update

To determine when statistics were last updated, use the STATS_DATE function.

## PDW / SQL Data Warehouse

The following syntax is not supported by PDW / SQL Data Warehouse

SQL Copy

```
update statistics t1 (a,b);
```

SQL Copy

```
update statistics t1 (a) with sample 10 rows;
```

SQL Copy

```
update statistics t1 (a) with NORECOMPUTE;
```

SQL Copy

```
update statistics t1 (a) with INCREMENTAL=ON;
```

SQL Copy

```
update statistics t1 (a) with stats_stream = 0x01;
```

# Permissions

Requires ALTER permission on the table or view.

# Examples

## A. Update all statistics on a table

The following example updates the statistics for all indexes on the `SalesOrderDetail` table.

SQL Copy

```
USE AdventureWorks2012;
GO
UPDATE STATISTICS Sales.SalesOrderDetail;
GO
```

## B. Update the statistics for an index

The following example updates the statistics for the `AK_SalesOrderDetail_rowguid` index of the `SalesOrderDetail` table.

SQL Copy

```
USE AdventureWorks2012;
GO
UPDATE STATISTICS Sales.SalesOrderDetail AK_SalesOrderDetail_rowguid;
GO
```

## C. Update statistics by using 50 percent sampling