Statistics

You get to the Statistics page from the Database Overview Page or from the hierarchy tree.

What are Statistics

Statistics sample your data and are use by the query optimizer to determine the best way to execute your query. If statistics are out of date, your queries may not be using the best process to get to data.

The importance of statistics maintenance

With SQL Server there are different ways of accessing tables, for instance a clustered index seek is different from a clustered index scan. The seek takes you right to the rows you need, where the scan goes through the entire clustered index. Even better than those may be a nonclustered index scan, and better yet a nonclustered index seek.

First off let’s cover some terminology.

- Clustered index: an index that is applied to the table itself and rearranges the structure of the table to a b-tree that is used for quick searching, but it does contain all of the table data in the leaf nodes.
- Nonclustered index: a copy of some of the data into a different b-tree structure, ordered in a way that can be useful for searching. The nonclustered index can be much smaller if you only include a subset of the columns from the original table.
- Page or Data Page: All of your data, tables, indexes are made up of pages that are simply an 8K chunk of memory and disk that contains rows. A data page may contain a single row, or it could contain many narrow rows that make up the 8k piece of memory and disk.
- Statistics: data that SQL Server keeps with information about what is contained in your clustered and nonclustered indexes. It contains information the density and distribution of data in that index. Statistics sample your data and are use by the query optimizer to determine the best way to execute your query.

Now image if you have a table with 1 billion rows in it this table would contain many pages. The seek versus scan would make a big difference in the number of pages or rows being looked out to give you your results.

Think of a small table with just a few narrow columns, and only 15 to 20 rows. In that case the indexing may not matter at all because the table is likely only made up of a couple to a few pages. No matter how you scan or seek on that table you are not accessing more than a couple pages.

However there are times that SQL Server makes mistakes and chooses the wrong way to find your data. One example of this if you statistics are out of data and SQL Server things that the data in an index is different from what it actually is. SQL server think from out of date statistics that a table only has a few rows, and then decide to do a table or clustered index scan, when it actually contains many more rows, and should have used an index. This can lead to very challenging performance tuning because your queries may not be performing the way you expect and indexes may not be helping you.

Statistics maintenance is what is needed to keep the statistics up to date. The built in maintenance plans that you can configure from SQL Server Management Studio give you the brute force approach to just rebuild them all, but this can be very resource intensive. There are better approaches which can be scripted by looking at only those tables and indexes that have out of date statistics, or only those table with a significant number of changes since the statistics were last rebuilt, then rebuilding just those.

You can use the Statistics report in Database Health Monitor to check in on your statistics to see how they look. This is one of the first places that I check when doing query performance work.

No Statistics – That is an indication that statistics should be updated.

Old Statistics – Also needs to be updated.

No rows changed – no reason to rebuild any of these statistics.

Looking fine – These are indexes that only have a small number of changes associated with them and don’t need to be rebuilt.

Example: Statistics Gone Wrong

1. Table created in a test environment
2. Add indexes as needed
3. Deploy to a production system (no autostatistics)
4. Table starts out small in production
5. Generate statistics (once)
6. Table grows BIG over the next 4 years, with no new statistics
7. Query optimizer still thinks your table is small, and you get the wrong plan.