Jes Schultz Borland says:

## [Are “bad” statistics the reason my query is slow?](http://brentozar.us2.list-manage1.com/track/click?u=9082566fb63d87be35c0662bc&id=7b42bc7b84&e=b137e91a2b)

How often are “bad” statistics to blame for a “slow” or “bad” query? Chances are, not nearly as often as you think.

### What are “bad” statistics?

Most often, when people say statistics are “bad”, what they mean is “out-of-date”. But what does that mean?

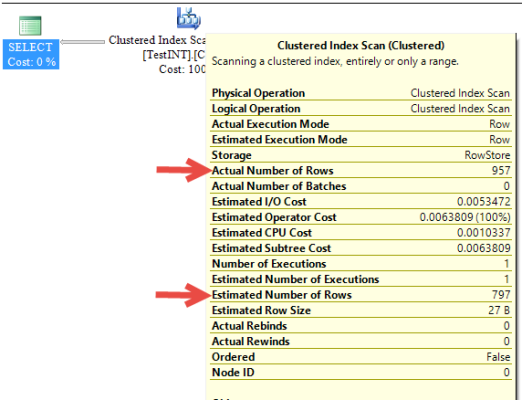
Let’s compare two tables – dbo.OrderStatuses and dbo.Orders. OrderStatuses is a 25-row table that contains a master list of all possible order statuses. Only one or two changes occur per year. Orders is a 2-million-plus-row table that contains information about every order placed in our system. There are, on average, 10,000 new orders placed per day (currently).

Let’s say that statistics were last updated on dbo.OrderStatuses 90 days ago, and last updated on dbo.Orders 9 days ago. Are either of those “out of date”? The date the statistics were last updated isn’t as important as how many changes have occurred since the last update.

Rougly, all tables with more than 500 rows will need a number of changes equal to 20% of the rows plus 500. ([The formula for updating statistics is documented here](http://brentozar.us2.list-manage.com/track/click?u=9082566fb63d87be35c0662bc&id=109840fb38&e=b137e91a2b).) This formula applies to tables of 1,000 rows, 560,000 rows, or even 2,000,000 rows. Larger tables require more changes to automatically update – thus, large tables are more susceptible to “bad” statistics.

### How do I tell if statistics are my problem?

Or, as the question was asked in a different way, is there a way to tell if statistics are my problem without looking at the query plan? When asked that way, I would answer, “Not easily.”

I most commonly find tables that have statistics problems by looking at a query’s actual execution plan and comparing the estimated and actual rows returned. [](http://brentozar.us2.list-manage.com/track/click?u=9082566fb63d87be35c0662bc&id=4020866f8d&e=b137e91a2b)

Finding out-of-date stats in other ways would require me to do some digging. I’d have to find the number of rows in an index, calculate the change threshold, and use sys.dm\_db\_stats\_properties modification\_counter to determine if it’s close to the threshold. Then I’d have to decide if I should manually update the stats or not.

I prefer to approach the problem on a query-by-query basis instead.

### What can you do to combat this?

Leave the default options to auto-create and auto-update stats turned on.

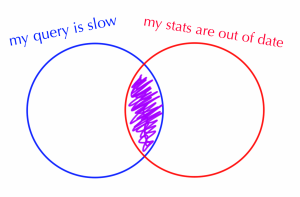
Run [Ola Hallengren’s](http://brentozar.us2.list-manage1.com/track/click?u=9082566fb63d87be35c0662bc&id=26b09eacaf&e=b137e91a2b) index maintenance script – or, his entire set of scripts. Ola’s index maintenance script has the parameter @UpdateStatistics, which allows you to choose index stats, column stats, or all. You can also choose to set @OnlyModifiedStatistics to true, so stats are only modified if rows have changed since the last stats update. This is a great way to include stats maintenance along with index maintenance in your regular routine.

If you have large tables which have frequent updates, but automatic stats updates and a weekly stats job aren’t enough to keep up, it’s acceptable to create a SQL Server Agent Job that updates the stats on that table on a regular basis – perhaps once a day.

In SQL Server 2008 R2 SP1, Microsoft introduced [trace flag 2371](http://brentozar.us2.list-manage.com/track/click?u=9082566fb63d87be35c0662bc&id=f490a1c8ed&e=b137e91a2b). With this trace flag enabled, rather than a set 20% change threshold, the more rows you have in a table, the lower the change percentage gets for an auto update. What is the formula? That’s not published yet, but one example is given in the KB – “For example, if the trace flag is activated, update statistics will be triggered on a table with 1 billion rows when 1 million changes occur.” Be careful with this option, though, and test it thoroughly before putting it in production. Frequent stats updates can cause frequent query recompilations – queries can take a bit longer to run, and CPU will be used more.

### What if statistics aren’t my problem?

The chances of a poor query performance being caused by out-of-date statistics is very low.

[](http://brentozar.us2.list-manage.com/track/click?u=9082566fb63d87be35c0662bc&id=05d54ba607&e=b137e91a2b)

There are so many things that factor into query performance that you should look at first. Is the query slow only for you, or for one set of values? [The problem could be parameter sniffing](http://brentozar.us2.list-manage1.com/track/click?u=9082566fb63d87be35c0662bc&id=16812b3e8c&e=b137e91a2b). Evaluate the overall health of your SQL Server by checking your wait statistics, or using a tool like [sp\_AskBrent](http://brentozar.us2.list-manage.com/track/click?u=9082566fb63d87be35c0662bc&id=d08c1d39b2&e=b137e91a2b) to see what is happening in your server right now. Is your server suffering from blocking? Review the query execution plan – are there obvious issues such as large scans, index seeks combined with key lookups, large sorts, or other common problems? And last but not least, do you have baselines to compare to?

Remember: bad, or out-of-date, statistics, can cause problems, but it’s rare. A poorly-performing query is always worth tuning!

Want to learn more about statistics? Check out my video [Statistics: Beyond Performance](http://brentozar.us2.list-manage.com/track/click?u=9082566fb63d87be35c0662bc&id=19ced8e707&e=b137e91a2b). Need performance tuning help? Check out [Developer’s Guide to SQL Server Performance](http://brentozar.us2.list-manage1.com/track/click?u=9082566fb63d87be35c0662bc&id=688a5f47f3&e=b137e91a2b)!

**Kendra says**: I completely agree with Jes– I was once a “statistics blamer” and thought it was the source of all my slow query problems. It turns out that there was just a lot I didn’t understand.

*Wanna join Brent Ozar Unlimited?* [*We're hiring DBAs.*](http://brentozar.us2.list-manage1.com/track/click?u=9082566fb63d87be35c0662bc&id=49db0ce638&e=b137e91a2b)