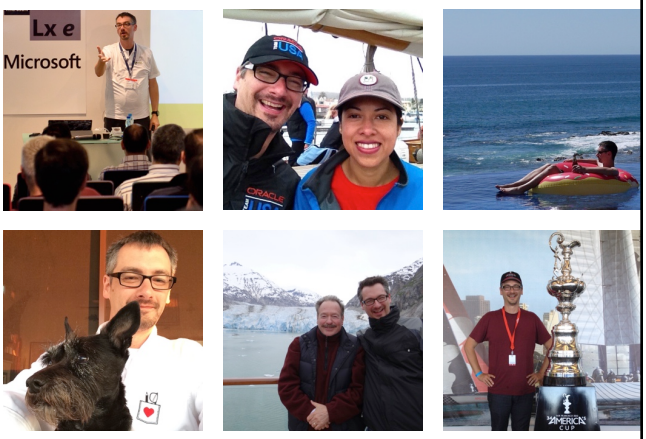
Watch Brent Tune Queries

www.BrentOzar.com
sp_Blitz – sp_BlitzFirst
email newsletter – videos
SQL Critical Care®



BrentOzar.com /go/tunequeries

Be Creepy.

Blitz first for obvious problems

End user requirements gathering

Capture query metrics

Read the metrics and plan

Experiment with the query cost

Execution plan review

Parallelism opportunities

Index improvements

Blitz the box first

sp_Blitz® for SQL Server
setting that's been changed
and influences the plan

sp_BlitzIndex® looking for
disabled indexes, heaps,
obvious missing indexes

End user requirements

Define your finish line:
how much time can we spend?

Can we run it less often,
or run it somewhere else?

Get the business purpose
of the query output

Find out if it's machine-generated,
inline, dynamic, or stored proc

Capture query metrics

Make sure the query doesn't write data

Run the query with your SSMS tuning settings on

Make sure you've got the right query and the right plan

Make sure it's not a parameter sniffing problem

Start a separate window to compare before/after

Read metrics, plan

Identify the logical reads, CPU time, duration, cost

What's the biggest problem: reads, CPU, or duration?

Does the query's duration/ reads/CPU match up with the amount of work it's doing?

Experiment with query cost

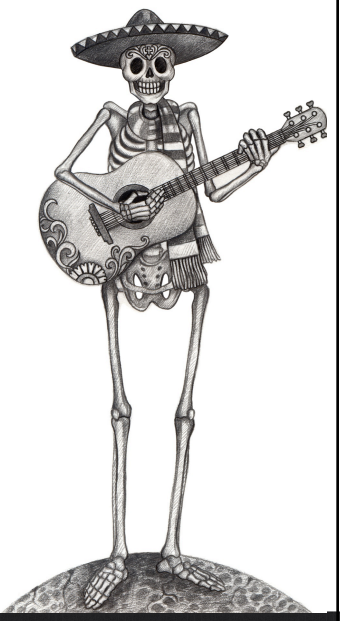
Remove the ORDER BY

Change the list of fields in the SELECT to just be SELECT 1

Switch table variables to temp tables

For any non-INNER joins, make sure they really need to be something else, and if we need the data.

Did the cost change take you to the finish line? Start asking tough questions.



Execution plan review

Look at plan's properties for Optimization Level and Reason for Early Termination.

Look at the top right operator's estimated vs actual row counts.

If est vs actual is off, why?

- SARGability issues
- No indexes or stats
- Stats out of date
- Functions

Estimates still off? Try splitting query into multiple queries, temp tables.

Parallelism opportunities

Is the query going parallel, and if so, is it benefitting?

If not, does the query exceed server's Cost Threshold for Parallelism?

Does it have a long duration (>1 second) that's matched by identical CPU time?

Are there any parallelism inhibitors? BrentOzar.com/go/serialudf

Index improvements

Key lookups? Can we widen an existing index into a covering index, or add one?

Any unmatched index warnings?

Is there an indexed view involved that SQL should use, but isn't?

Be Creepy.

Blitz first for obvious problems

End user requirements gathering

Capture query metrics

Read the metrics and plan

Experiment with the query cost

Execution plan review

Parallelism opportunities

Index improvements

BrentOzar.com/go/tunequeries

Full script downloads

More examples with video walkthroughs

Links to the tools I use and the Stack Overflow DB