BRENT OZAR
UNLIMITED®

How I Use the First Responder Kit

p1

BRENT OZAR
UNLIMITED

99-05: dev, architect, DBA
05-08: DBA, VM, SAN admin
08-10: MCM, Quest Software
Since: consulting DBA

www.BrentOzar.com
Help@BrentOzar.com

p2

My job: 2-day SQL Critical Care®

Day 1, morning: rapidly assess a single SQL Server, database indexes, queries running against it, team

Day 1 afternoon & day 2 morning: write findings

Day 2 afternoon: deliver findings & training to get the team out of the emergency, quickly

I use the First Responder Kit
to make it all happen, fast.

Live attendees: introduce yourself & your location

	Database Developer	Development DBA	Production DBA
Design tables	Sometimes		
Write queries	Daily	Sometimes	
Deploy changes	Daily	Daily	Sometimes
Tune queries	Sometimes	Daily	Weekly
Monitor performance	Rarely	Daily	Weekly
Troubleshoot outages		Sometimes	Daily
Install & config SQL			Daily
Design & test DR			Sometimes

Live attendees: Slack pro tips

Accidentally close your browser? Want to share screenshots? Lots of pro tips: https://BrentOzar.com/slack

To share code or T-SQL, click the + sign next to where you type text in, and choose "code or text snippet."

To share screenshots, go to https://imgur.com, upload it, and then copy/paste the URL in Slack.

No direct messages please.

Today's agenda

Server-wide health check: sp_Blitz

Performance check: sp_BlitzFirst

Find the queries causing the waits: sp_BlitzCache

Checking indexes that'd help queries: sp_BlitzIndex

Analyzing deadlocks: sp_BlitzLock

What's running now: sp_BlitzWho, sp_WhoIsActive

Restore databases: sp_DatabaseRestore

My demos will be in my lab.

- SQL Server 2022, 4 cores, 30GB RAM
- Stack Overflow database: BrentOzar.com/go/querystack
- Workloads from the Mastering classes run by SQLQueryStress

You repeating that workload is NOT today's goal.

But if you wanna learn more about the technique: https://BrentOzar.com/go/workload

p7

You're going to uncover a lot.

Settings you didn't know about

Metrics you haven't seen before

Problems you didn't know you had

p8

I can’t teach you everything.

I can’t teach you:

- What every warning means
  (but that’s what the URLs are for)
- How to solve every problem
  (even the Mastering classes can’t do that)

But I want to teach you:

- How I investigate problems
- Where I’d go next to dig deeper

p9

So you will have a lot of questions.

And I wanna answer ‘em all!

But at the same time, let’s be fair:

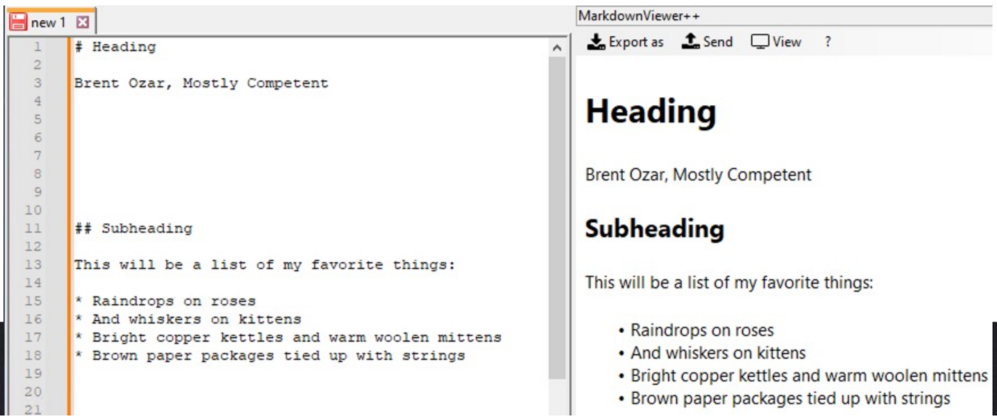
- One server per person
- I may give you a link: read it,
  follow the instructions
- The link may be a paid class
  (mine or someone else’s,
  because I don’t cover everything)

p10

We'll take notes as we go.

My preferred file format: Markdown

What you type: What gets rendered:



```
# Heading

Brent Ozar, Mostly Competent







## Subheading

This will be a list of my favorite things:

* Raindrops on roses
* And whiskers on kittens
* Bright copper kettles and warm woolen mittens
* Brown paper packages tied up with strings
```

Heading

Brent Ozar, Mostly Competent

Subheading

This will be a list of my favorite things:

- Raindrops on roses
- And whiskers on kittens
- Bright copper kettles and warm woolen mittens
- Brown paper packages tied up with strings

p11

My favorite Markdown apps

Windows: Notepad++ with MarkdownViewer plugin

Mac: Typora.io app, $15

I've tried, but haven't embraced Visual Studio Code:

- Cross-platform (Win, Mac, Linux, SQL, Postgres)
- Can run queries, integrate w/source control
- Can take notes in Markdown

p12

Let’s get started!

p13

sp_Blitz

What it does: overall health check

When to run it:

- When taking over a new-to-you SQL Server
- Before signing off that a server is ready for prod
- When you come back from vacation
- ~~On a scheduled basis~~

Azure SQL DB: unsupported due to ms_foreachdb
(most of the other FRK scripts work in Azure SQL DB)

p15

Most useful parameters

@CheckServerInfo = 1
adds low-priority FYI rows like data size, server size

@CheckUserDatabaseObjects = 0
if you don't control the user db contents and you don't want to see any problems with 'em

@OutputType = 'markdown'
if you want to paste the results into an email or a StackOverflow question

p16

Writing the output to tables

@OutputDatabaseName = ‘DBAtools’,

@OutputSchemaName = ‘dbo’,

@OutputTableName = ‘Blitz’

Most of the First Responder Kit scripts
offer these same parameters.

p17

sp_BlitzFirst

sp_BlitzFirst

What it does: performance health check

Who it's for:

- End users & power users: run w/o parameters, checks performance as of right now
- For database pros: lots of parameters

p19

Parameters for pros

@ExpertMode:

- 1: returns lots more diagnostic data
- 2: same as above, but without sp_BlitzWho before & after

@Seconds = 60: lets you take a longer sample

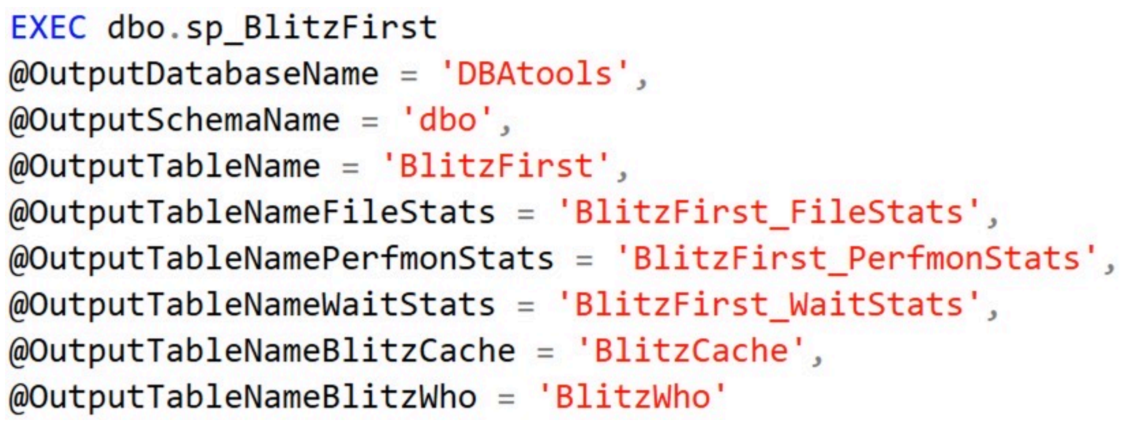
@SinceStartup = 1: since the server started up

Logged to table: lets you go back over time

p20

Logging to table

Put this in an Agent job, run it every 15 minutes



```
EXEC dbo.sp_BlitzFirst
@OutputDatabaseName = 'DBAtools',
@OutputSchemaName = 'dbo',
@OutputTableName = 'BlitzFirst',
@OutputTableNameFileStats = 'BlitzFirst_FileStats',
@OutputTableNamePerfmonStats = 'BlitzFirst_PerfmonStats',
@OutputTableNameWaitStats = 'BlitzFirst_WaitStats',
@OutputTableNameBlitzCache = 'BlitzCache',
@OutputTableNameBlitzWho = 'BlitzWho'
```

Special parameters for today

```
sp_BlitzFirst @SinceStartup = 1, @OutputType = 'Top10’
```

Take a screenshot of the full result set

Post it on https://imgur.com

Put that link in Slack (don’t upload the photo into Slack)

Wait stats cheat sheet

CX%: queries going parallel to do a lot of work.
sp_BlitzCache @SortOrder = ‘reads’ or ‘cpu’

PAGEIOLATCH: reading uncached data pages from disk.
sp_BlitzCache @SortOrder = ‘reads’

SOS_SCHEDULER_YIELD: CPU pressure.
sp_BlitzCache @SortOrder = ‘cpu’

WRITELOG, HADR_SYNC_COMMIT: logging changes.
sp_BlitzCache @SortOrder = ‘writes’

Much more: Mastering Server Tuning class

sp_BlitzCache

sp_BlitzCache

What it does: show your top most resource-intensive queries, analyze them for performance issues

When to run it: after finding your SQL Server's top bottleneck (like CPU, reads, blocking), when you want to find the queries causing the bottleneck

Most important parameter: @SortOrder

@SortOrder common options

Reads: for PAGEIOLATCH waits

CPU: for SOS_SCHEDULER_YIELD waits

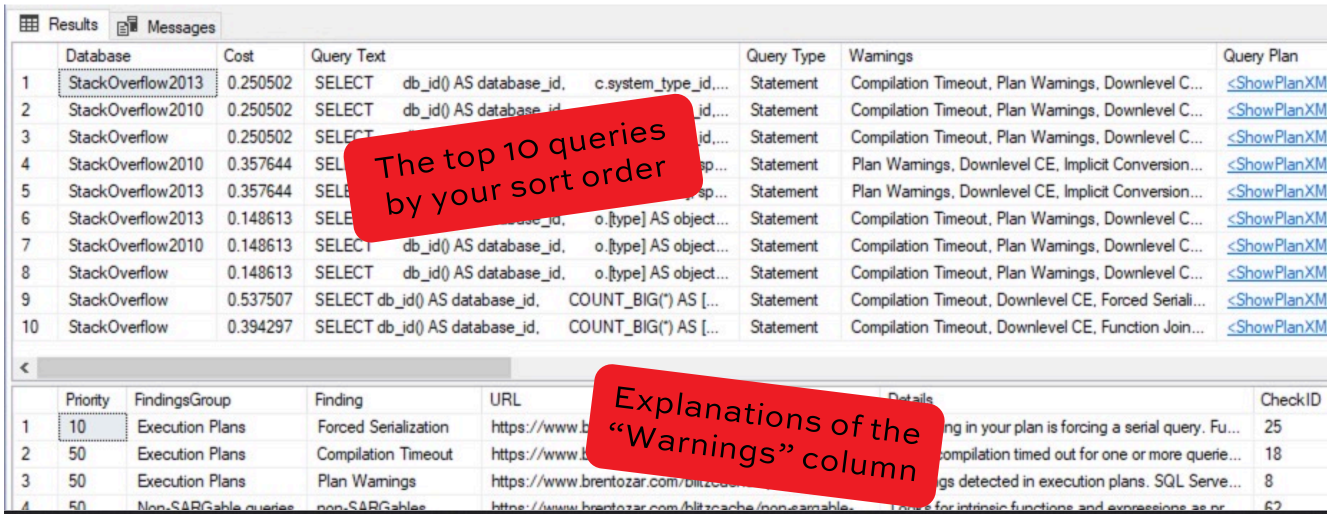
Writes: for WRITELOG, HADR_SYNC_COMMIT

Avg duration: for rarely run, but long-running queries

Executions: for queries the app should cache instead

Memory grant: queries flushing the cache

Returns 2 result sets



Results
Messages
The top 10 queries by your sort order
Explanations of the "Warnings" column

	Database	Cost	Query Text	Query Type	Warnings	Query Plan
1	StackOverflow2013	0.250502	SELECT db_id() AS database_id, c.system_type_id,...	Statement	Compilation Timeout, Plan Warnings, Downlevel C...	<ShowPlanXM
2	StackOverflow2010	0.250502	SELECT db_id() AS database_id ... id,...	Statement	Compilation Timeout, Plan Warnings, Downlevel C...	<ShowPlanXM
3	StackOverflow	0.250502	SELECT ... id,...	Statement	Compilation Timeout, Plan Warnings, Downlevel C...	<ShowPlanXM
4	StackOverflow2010	0.357644	SEL ... sp...	Statement	Plan Warnings, Downlevel CE, Implicit Conversion...	<ShowPlanXM
5	StackOverflow2013	0.357644	SELE ... sp...	Statement	Plan Warnings, Downlevel CE, Implicit Conversion...	<ShowPlanXM
6	StackOverflow2013	0.148613	SELE ... id, o.[type] AS object...	Statement	Compilation Timeout, Plan Warnings, Downlevel C...	<ShowPlanXM
7	StackOverflow2010	0.148613	SELECT db_id() AS database_id, o.[type] AS object...	Statement	Compilation Timeout, Plan Warnings, Downlevel C...	<ShowPlanXM
8	StackOverflow	0.148613	SELECT db_id() AS database_id, o.[type] AS object...	Statement	Compilation Timeout, Plan Warnings, Downlevel C...	<ShowPlanXM
9	StackOverflow	0.537507	SELECT db_id() AS database_id, COUNT_BIG(*) AS [...	Statement	Compilation Timeout, Downlevel CE, Forced Seriali...	<ShowPlanXM
10	StackOverflow	0.394297	SELECT db_id() AS database_id, COUNT_BIG(*) AS [...	Statement	Compilation Timeout, Downlevel CE, Function Join...	<ShowPlanXM

	Priority	FindingsGroup	Finding	URL	Details	CheckID
1	10	Execution Plans	Forced Serialization	https://www.b	ng in your plan is forcing a serial query. Fu...	25
2	50	Execution Plans	Compilation Timeout	https://www.b	compilation timed out for one or more querie...	18
3	50	Execution Plans	Plan Warnings	https://www.brentozar.com/blitzca	gs detected in execution plans. SQL Serve...	8
4	50	Non-SARGable queries	non-SARGables	https://www.brentozar.com/blitzcache/non-sargable-	... for intrinsic functions and expressions as pr	62

But be careful...

Always check the bottom "Warnings" result set first.

Look for Priority 1 warnings.

They indicate you can't trust the top results, usually due to memory pressure or unparameterized queries.

Read the URL – it's a big deal.
You have to fix that first, or you're wasting time.

Cool parameters

@ExportToExcel = 1: returns a result set that's copy/paste friendly, doesn't include plans. Useful for sending the top 10 list to your developers.

@MinutesBack = 15: lets you only show queries that have run recently. Basically filters out overnight jobs.

@ExpertMode = 1: I actually never use this.

p29

Can show last actual query plans

SQL Server 2019 & newer can track these.

ALTER DATABASE SCOPED CONFIGURATION SET LAST_QUERY_PLAN_STATS = ON;

Does have overhead: only turn it on when you're actively tuning the server, and turn it back off after.

p30

sp_BlitzIndex

What it does: analyzes design issues with indexes.

When to run it: when you're in charge of index tuning, and you want to spend a few hours working on them.

This is not a quick-fix script: the results require work on your part to parse & combine.

@GetAllDatabases = 1: loops through all user databases and runs sp_BlitzIndex in each, combining the results into one result set.

@Mode Parameter

@Mode = 0 (default): returns prioritized findings based on the D.E.A.T.H. Method in my index classes.

@Mode = 2: inventory of existing indexes, good for copy/pasting into Excel to work offline. Has a @SortOrder parameter for things like rows, size, reads, writes, lock time.

@Mode = 3: inventory of missing indexes. On 2019+, check out the Sample Query Plan column.

@Mode = 4: like Mode 0, but returns more findings (that tend to be harder to fix.)

p33

Outputting those modes to table

In the cloud, you may want to schedule this.

Cloud services can fail over at any time.

Index metadata is only kept in memory.

When you wanna do index analysis, you may not have any metadata to work with if the server failed over.

Consider logging sp_BlitzIndex to table on Friday afternoons, after you've had a week of activity.

p34

Another mode: table-level details

sp_BlitzIndex @TableName = ‘Users’

sp_BlitzIndex @DatabaseName = ‘StackOverflow’, @SchemaName = ‘dbo’, @TableName = ‘Users’

1. Existing indexes
2. Missing indexes
3. Columns in the table, datatypes, collations
4. Foreign keys
5. Statistics
6. Columnstore visualization

p35

How I use it, bonus round

sp_BlitzIndex @Mode = 2, @SortOrder = ‘rows’

sp_BlitzIndex @Mode = 2, @SortOrder = ‘size’

sp_BlitzIndex @Mode = 2, @SortOrder = ‘writes’

And ask, “Hey, do we really need this stuff?”

p36

sp_BlitzLock

sp_BlitzLock

What it does: analyzes recent deadlocks, groups them together by table, query, app, login.

When to run it:

- When sp_Blitz warns you about a high number of deadlocks per day
- When users complain about deadlock errors
- ~~When you're just curious~~

p38

Where the data comes from

The built-in system health Extended Events trace

Good news: on by default in SQL Server

Bad news:

- Amount of history is small, but you can easily extend it: https://www.mssqltips.com/sqlservertip/6456/improve-sql-server-extended-events-systemhealth-session/
- Not on by default in Azure SQL DB anymore

To work around those, you can create your own XE deadlock trace or output sp_BlitzLock to table regularly.

p39

How I use it

Run it with no parameters

Jump to the bottom result set

Identify the top 1-3 tables involved in most deadlocks, tune their indexes (Mastering Index Tuning class)

Identify the top 1-3 queries, tune them, make them more SARGable, make their transactions short & sweet, change their isolation level

p40

sp_BlitzWho

What it does: lists currently running queries, sorted from longest running to shortest

Who it's for: DBAs, developers, sysadmins

When to run it:

- When people are complaining SQL Server is slow
- When you want to find who's using TempDB (if it's session-based usage)

p42

Parameters

@ExpertMode = 1: adds memory grant columns, Resource Governor workload groups

@OutputDB/Schema/Table: useful for setting up a break-glass emergency job ahead of time for support

Similar script: sp_WhoIsActive. Useful when the server is so slow that sp_BlitzWho won’t respond.

p43

sp_BlitzBackups

sp_BlitzBackups

What it does: analyze your msdb backup tables

- Recovery Point Objective (RPO) Worst Case, Minutes: how much data you would have lost if the SQL Server went down at the worst possible time
- Recovery Time Objective (RTO) Worst Case, Minutes: how long your restores might take if the SQL Server went down at the worst possible time

p45

RPO worst case example

Full backups run every night at midnight

Log backups run every 15 minutes

But for some reason, the SQL Server restarted or the Agent job failed at 9:14, so no log backups at 9:15AM

RPO worst case:

- Our server goes down at 9:29AM
- Right before the 9:30AM log backup jobs
- We'd lose 29 minutes of data (9AM to 9:29AM)

p46

RTO worst case example

Full backups run every night at midnight, take 4 hours

Log backups run every 15 minutes, take 1 minute each

RTO worst case:

- Our server goes down at 11:59PM
- We have to restore the last full, plus the entire day's log backups
- The full backups took 4 hours
- The 95 log backups took 95 minutes total (1.5h)
- RTO worst case: 5.5 hours

p47

How I use it

Client: "We take backups every 15 minutes"

Me: "OK, let's check… hmm, looks like:

- Some databases are in simple recovery model
- We've missed a couple of backup jobs this week
- Our full restores would take a day"

Me: "Do you still feel like you're on top of this?"

p48

sp_DatabaseRestore

What it does:

- Read folders with Ola Hallengren backup files
- Generate restore scripts (and optionally run 'em)

When to use it:

- Refresh dev/test/QA servers with recent prod backups
- Seed AGs, log shipped secondaries
- Test backups by doing a restore, CHECKDB

p50

Common parameters

@Database = source db name

@RestoreDatabaseName = if you want to restore it under a different name

@BackupPathFull, Diff, Log: where the backups are

@TestRestore = 1: drop db when the restore finishes

@RunCHECKDB = 1

@RunRecovery = 1: otherwise more logs can be added

@Execute = ‘N’: if you just want to generate scripts

p51

BRENT OZAR
UNLIMITED®

Whew. We covered a lot today.

Let’s recap a few things for folks who need to leave early, and then we’ll write up our findings for our teammates.

p52

The overall process

1. Do a server-wide health check with sp_Blitz
2. Do a performance check with sp_BlitzFirst (and set it up to log data to tables every 15 min)
3. Find the queries causing your top waits by using sp_BlitzCache’s @SortOrder parameter
4. Check sp_BlitzIndex @GetAllDatabases = 1 to see if index changes can help a lot, fast

p53

Now, paint the big picture.

Your teammates need to see:

- The priority 1-50 sp_Blitz findings that scare you
- Prioritized list of your top wait types
- A list of the top 3-4 queries causing those waits
- A rough idea of what tables need index help first

p54

You can do this!

For private help for anything after the class, email Help@BrentOzar.com with:

- A note that you were in this class
- sp_Blitz @CheckServerInfo = 1
- sp_BlitzFirst @SinceStartup= 1

p55

Thanks, and have fun conquering your servers!

p56