(May 5th, 2026)

If you know someone who would benefit from being an Insider, feel free to forward this PDF to them so they can sign up here.

Note: As an Insider, you can read all prior Insider newsletters here.

Quick Tips for our Insider friends!

Happy May! Since the last newsletter I have been dragged into the modern age finally with the death of my trusty old laptop, a Dell M6800 workhorse I've been using with W7 since… wait for it… 2015! I'd been using a smaller W10 machine to run SQL Server but my beloved Applecross is no more. I'm a big believer in "if it ain't broke, don't fix it", but yes, Kimberly was right – I should have upgraded many moons ago. Goodbye W7, my old friend!

SQLskills News

We've replaced our back-end video streaming system with one that has a lot more end-user functionality, is more robust, and has features like chapters so I can provide indexes into the long Immersion Event daily recordings. The transition has already been made and should be seamless for all viewers. I'll be working through our library to add the chapter indexes over the new few months.

Book Review

Two more history books for you this time…

The first is Barbara W. Tuchman's *The Zimmermann Telegram*. Anyone who knows a bit of history about the USA in WWI knows about the Zimmermann telegram, and how its contents pretty much forced Wilson into the war. As with any episode, there was much more to it than that simple cause-and-effect. Fascinating stuff and highly recommended! Additional from Amazon: "In January 1917, the war in Europe was, at best, a tragic standoff. Britain knew that all was lost unless the United States joined the war, but President Wilson was unshakable in his neutrality. At just this moment, a crack team of British decoders in a quiet office known as Room 40 intercepted a document that would change history. The Zimmermann telegram was a top-secret message to the president of Mexico, inviting him to join Germany and Japan in an invasion of the United States. How Britain managed to inform the American government without revealing that the German codes had been broken makes for an incredible story of espionage and intrigue as only Barbara W. Tuchman could tell it"

The second is Tennent H. Bagley's *Spymaster: Startling Cold War Revelations of a Soviet KGB Chief*. Fascinating memoir of KGB spymaster Sergey Kondrashev, written by his friend and CIA spymaster Bagley, with whom he struck up a friendship in 1994, and was actually on opposing sides in some operations! Banned by Putin's regime, Bagley promised to have this published in the west after Kondrashev's death. Highy recommended! Partial from Amazon: "From the dark days of World War II through the Cold War, Sergey A. Kondrashev was a major player in Russia's notorious KGB espionage apparatus. Rising through its ranks through hard work and keen understanding of how the spy and political games are played, he "handled" American and British defectors, recruited Western operatives as double agents, served as a ranking officer at the East Berlin and Vienna KGB bureaus, and tackled special assignments from the Kremlin."

Ponderings…

What are some of the things that come to mind when you think about performance monitoring? Here are some examples: transaction logs running out of space, queries taking too long or timing out, I/Os taking too long, low memory, 100% CPU, blocking.

What do you do when monitoring flags that one of these has occurred on your production system?

React.

You have to dig in with whatever instrumentation you've built, or using any 3rd-party tool you've bought, to try to figure out what went wrong so you can take corrective action to stop it happening (under a lot of pressure) or prevent it happening again in the future (under not quite so much pressure).

It's a reactive process: problems have to occur before they're noticed, so you don't get a chance to prevent bad things happening to your workload.

A similar thing happens in the stock market: if a stock becomes very strong and the price increases, people buy it. If a stock tanks and the price decreases sharply, people sell it. Hence the phrase "buy high, sell low" to describe reactive behavior that can stop people making money on stocks.

In the stock market, the big wins come from noticing the signs that a stock is going to take off, and buying it as early as possible to maximize gains, and, of course, knowing when it's going to go down again to sell while the price is at or near its peak.

Pro-activity beats reactivity.

I think we should be aiming for the same behavior with our SQL Server performance monitoring.

Instead of building monitoring that alerts us when a problem has occurred, we should build monitoring that can tell when a problem is starting to occur so we can deal with it before it detrimentally affects the workload.

Using I/O latencies as an example, the typical alerts fire when I/O latency crosses a threshold, say 20ms. I think it would be better for the monitoring to know what the I/O pattern is (including variations in the workload based on time/day/position in the business cycle) and then fire an alert when the I/O latency starts to deviate from the norm for a sustained period. This way the problem can be investigated far sooner than waiting until the I/O latency has hit a critical point where the workload is already suffering badly.

This is where I think performance monitoring should go in future, and it isn't a trivial thing to do, especially considering all the facets of performance monitoring. But think of the possibilities: knowing a performance problem is going to happen before it actually does!

Call to action: The next time you're implementing some monitoring, consider if there's a way to easily change what you're implementing so you get advance notice of a problem, rather than notice that a problem has occurred.

Another thing you can do is take a holistic look at your overall SQL Server instances, and a great way to do that is with something like our comprehensive health check!

The Curious Case of…

This section of the newsletter explains recent problems we've helped with on client systems or been asked about online; they might be something you're experiencing too.

This is an oldie, but is very similar to a problem we saw recently, so running it again – enjoy!

Jonathan was working with a client recently who was having slowly degrading performance in their entire workload. The server had 64GB of memory installed, with 53GB configured for max server memory, following generally-recommended practices. Looking at the wait statistics, he saw that nearly every query was driving physical reads, which was overloading the I/O subsystem and causing long *PAGEIOLATCH_SH* waits. The page life expectancy on the server was in 100-200s, showing signs of buffer pool churn and contention on the system.

The next step was to check the contents of the buffer pool and see what was using the memory internally in SQL Server, using a script based on code from here. Jonathan was surprised to see that the distribution database (the server was a replication Distributor as well as serving other roles) was using nearly 44GB of the buffer pool space with the user databases consuming less than 6GB of buffer pool.

He then checked the replication settings and found that someone had set the Distributor to have a retention period of 72 days instead of 72 hours! This was for a single subscriber for reporting that was in the same data center and was only ever offline for routine patching every quarter.

Changing the distribution retention back to hours from days and running the *Distribution Clean Up: Distribution* job on the server reduced the size of the *MSrepl_commands* table (which is scanned by the Distribution Agents for subscriptions) and allowed the user databases to return to normal buffer pool usage.

It's always interesting to see how changing a setting in one area (albeit accidentally) can have such a profound effect on the overall health of a SQL Server instance. Make sure that you understand all the ramifications of any configuration change that you make.

#TBT

(Turn Back Time…) Blog posts we've published since the previous newsletter plus some older resources we've referred to recently that you may find useful.

The theme for the TBT this time is backups:

- My Pluralsight course: *SQL Server: Understanding and Performing Backups*
- My TechNet Magazine article from 2009: *Understanding SQL Server Backups*
- My TechNet Magazine article from 2009: *Recovering from Disasters Using Backups*
- Backup blog posts from our accidental DBA series
- Blog post: *New script: How much of the database has changed since the last full backup?*
- Blog post: *Importance of having the right backups*
- Blog post: *Importance of validating backups*
- My blog post category on *Backup/Restore*

And some of my recent blog posts:

- *SQL101: Application data consistency checking*

I hope you find these useful and interesting!

Video Demo

In this video Jonathan demonstrates a feature enhancement to Extended Events in SQL Server 2025 called Time Based Event Sessions which allows a new session level option to specify the MAX_DURATION an event session is allowed to run before automatically stopping the collection.

The video is about 8 minutes long and you can get it here.

And the demo code is here.

Enjoy!

SQLskills Training

We have no plans for live, public classes in 2026, but we've already released the first of many new courses coming over the next year! And of course, all our recorded courses from the last few years are still as relevant as ever.

To help your manager understand the importance of focused, technical training, we've also added a few items to help you justify spending your training dollars with us:

- Letter to your manager explaining why SQLskills training is worthwhile
- Community blog posts about our classes

You can get all the details in our shop.

Summary

I hope you've enjoyed this issue – I really enjoy putting these together. If there's anything else you're interested in, I'd love to hear from you - drop me a line.

Thanks,
Paul