(May 12th, 2014)

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

Quick Tips for our Insider friends!

Hey Insiders!

This newsletter is coming to you from Chicago where as it hits your inbox, Jonathan's teaching the first module of our IE3 course on high availability and disaster recovery. We've had a great time over the last two weeks with 72 people attending the first four classes here.

We're working hard on new Pluralsight courses and over the next few months you can expect to see courses from Glenn on DMVs, from Jonathan on Change Data Capture and Service Broker, from Joe on query tuning patterns, from Erin on DBCC, from me on fragmentation, and from Kimberly on stored procedure performance.

We've published the pre- and post-conference workshop line-up for SQLintersection in Las Vegas in November and the show is open for registration – see here for details. We'll have the full list of sessions up in a week or two. I'm really looking forward to presenting a full-day workshop on using waits and latches for performance tuning!

Two books to discuss this time: the first is George R. R. Martin's *A Game of Thrones*. It's an excellent start to the series of five books! We've only seen the first season of the HBO show so far, so I'm looking forward to starting the second book as soon as we get home from Chicago and moving into unknown story territory. Considering the complexity of Martin's world and how interwoven the story-lines are, I'm glad there are four more long books to go through. Highly recommended!

The second is Iain Banks' *The Quarry*. This is the last book that my favorite author of all time wrote before he died last year. It's quite excellent. The dying character is irascible, bitter, and filled with justifiable invective against elements of the world and I have to imagine this is semi-autobiographical. It's a great story and a fitting final novel. I think I'll have to go back and re-read Banks' non-sci-fi books (already read the sci-fi ones several times each). Thanks Iain – RIP.

Please let us know if you liked what you read/saw here and/or have any suggestions for future Quick Tips.

Note: you can get all the prior Insider newsletters here.

Paul's Ponderings

Every day in our lives, most of us seek to minimize risk. Examples include assessing traffic to choose the safest point to pull out into the road, choosing flights in a multi-flight trip to avoid flight delays causing a missed connection, and delaying rocket launches based on the probability of bad weather occurring.

Ok – we don't all do the last one, but you get the idea. It's a natural human tendency to want to limit our risk of possible disaster and negative outcomes.

Here's an example of forced risk aversion. A few years ago I was performing a health check for a client of a client. Our client writes casino-management software and their client was a casino – we'll call them X. I identified a performance issue that could be mitigated through the creation of several nonclustered indexes and some code changes. Our client said that wouldn't be possible for 18 months because the Gaming Commission in the state where X resides doesn't allow software changes until they've been extensively investigated and tested, to avoid any risk of instability or fraud.

The only other solution I could think of was to max out the server memory and drop in some SSDs, to make the server cache as much of the workload as possible and store the rest on the fastest possible I/O subsystem. This would be expensive to the client, but X was a casino, so I figured they could afford it. Problem solved.

That was a case where there was no choice, but often when dealing with SQL Server performance problems that require code changes – and sometimes extensive code changes – I find that people don't stop to consider less risky alternatives.

Changing code is inherently risky. Most code is not documented, even to a minimal standard of 'what does each block do and why'. It's quite horrifying to see really, especially as a former developer and development manager at DEC and Microsoft where code complexity demanded rigorous documentation (and even then, it was often inadequate).

My point is that allowing developers to make changes to a mostly undocumented and sometimes poorly-understood code base should give you sleepless nights. The risks involved include instability, loss of functionality, introducing security flaws, introducing other performance problems, and a host of other potential side-effects. So what are you to do?

Consider the alternatives:

1. Can you avoid/lessen the performance problem by buying more memory for the server and storing more of the workload in the buffer pool?
2. Can you avoid/lessen the performance problem by boosting the speed of the I/O subsystem – most likely by going to solid-state storage? (And don't forget the Buffer Pool Extension feature in SQL Server 2014 that's an intermediate step between memory and the I/O subsystem.)

3. Can you upgrade your server to one that has newer processors with faster per-core performance?
4. Can you create a small number of nonclustered indexes that will provide a net gain in performance, even considering the added runtime maintenance, regular index maintenance, and increased storage required for them?

The first three of these cost money, but so does making a code change. You're paying the salary of developers, testers, project managers, and others involved in making the change. New hardware is a one-time capital expense.

Numbers 1, 2, and 4 are also things that can be done within days, providing almost immediate relief of a performance problem, whereas changing code *properly* takes a lot of time.

Don't get me wrong – I'm not saying that you can use hardware to completely remove the need to change code, or that by default you should throw hardware at a performance problem. As your workload increases, you'll likely hit the point where you *have* to change the code to fix the problem, but you can often delay that point, giving you time to properly design, prototype, test, and implement the right fix in a robust, less risky way.

Call to action: The next time you're considering changing existing code to solve a performance problem, ask yourself if there's a less risky, and possibly cheaper, short-term alternative. And try to avoid the view that capital expense is a "worse" cost than making a code change. If the code change involves people who you pay money, and carries a risk of causing more problems that will take more people who you pay money to fix, plus annoy clients and possible lose business, which solution is cheaper in the long run? I don't want you to throw hardware at performance problems by default – I just want you to consider the alternatives to making code changes.

I'm curious to hear your thoughts on minimizing risk, so please feel free to drop me a line, always treated confidentially, of course.

Video Demo

In this newsletter I'm sharing with you a demo video from my Pluralsight course on SQL Server: Logging, Recovery, and the Transaction Log. One of the things some people find interesting is learning the internals of how certain operations work, and one such operation is a page split. Page splits are what causes fragmentation, and they are very expensive in terms of the amount of transaction log that is generated. This demo shows you what's going under the covers when a page split happens and why so much transaction log is generated.

The video is just over five minutes long and you can get it in WMV format here.

You can get the demo code here.

Enjoy!

SQLskills Offerings

Please know that all of our classes will run and their dates will not change. Additionally, most of our public training courses will be held in the first half of this year. We will add a few more classes in the second half of the year, but not all that many (specifically, one each of IE1, IE2, IEHW, and IE0 in Chicago and two IE1 deliveries in Australia). Please plan accordingly. And, if you're in Australia and want direct notice when our classes are added, please email training@SQLskills.com.

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

- Letter to your boss explaining why SQLskills training is worthwhile
- Community blog posts about our classes
- Immersion Event FAQ

2014 Immersion Events

Chicago, IL

- May 19-23, 2014: **IE4**: Immersion Event on Security, PowerShell, and Developer Support
- May 19-21, 2014: **IE0**: Immersion Event for the Accidental/Junior DBA

Bellevue, WA

- June 9-13, 2014: **IE1**: Immersion Event on Internals and Performance
- June 16-20, 2014: **IE2**: Immersion Event on Performance Tuning

See here for the main Immersion Event Calendar page that allows you to drill through to each class for more details and registration links.

Fall SQLintersection

As mentioned above, we've released the workshop line-up for our Fall SQLintersection event and it's open for registration. This year our Fall SQLintersection conference will be the week of November 10th in Las Vegas. We'll post the session line-up in a week or two once it's finalized. We hope to see you there!

Summary

We hope you've enjoyed this issue - we really enjoy putting these together.

If there is anything else you're interested in, we'd love to hear from you - drop us a line.

Thanks,
Paul and Kimberly

Paul@SQLskills.com and Kimberly@SQLskills.com