(June 12th, 2011)

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 bi-weekly Quick Tips is coming to you from Luxembourg in Europe where we're traveling around for a week between the Connections conference in Karlsruhe, Germany and our Immersion Event in London in a week's time.

Before the conference we headed down to Stuttgart and Liechtenstein and now we're heading towards Bruges, Amsterdam, and Cologne for a couple of nights each – immersing ourselves in the history and architecture of the cities. It's an almost overwhelming amount to take in during the course of a week and very hard to pick places to stop when we have so little time. Lots of pictures will be forthcoming!

The most recent book I've read is Rebecca Stott's ***The Coral Thief***. It's based in Paris in 1815 just after Napoleon has been defeated at Waterloo and is steeped in the blossoming world of natural history amid new theories like transformism and work in Cuvier's museum and laboratory. I love historical novels and this one doesn't disappoint as it's mixed with some unexpected thriller aspects too.

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

Paul's Ponderings

In this newsletter I'd like to discuss some of the trade-offs around schema design which are necessary for better performance.

At several clients recently we've been involved in discussions about how far to normalize a relational model. Third normal form (3NF) is the goal, where all non-key attributes define only the key. As a nice way to remember the primary rules behind 3NF you can use this play on the oath one swears in court "the attributes define the key, the whole key, and nothing but the key, so help me Codd" See this wikipedia link for more info and a formal definition using relational terms.

The problem is that normalizing to 3NF (or beyond – over-normalizing) can often cause performance problems when a query needs to join together many tables to be able to get from one entity to another, based on various query predicates.

For example, we have one client where the schema was written around ten years ago by a C# developer who was not especially SQL Server savvy. Every possible entity is stored in its own table and has an ID. All the attributes of each entity are stored in their own tables, with their own IDs. The relationships of which entities belong to other entities are stored in their own tables. The history of how each attribute has changed for each entity is stored in their own tables, with their own IDs. So a particular entity table may have foreign-key references to many other tables.

I would call this a Gordian-Knot design—because it's so complex that it's very hard to understand.

This is a heavily over-normalized schema where doing almost anything requires joining 10+ tables together. What's worse is that no archiving strategies have been implemented yet and so there are many years of data in the database resulting in joins that have to process huge amounts of data. There are multiple things that have to be done to fix this:

- Remove old data to reduce the data size and automate an archiving strategy
- Before that's done, consider using filtered indexes and query predicates to reduce the data size
- Ultimately de-normalize parts of the schema to allow some of the tables to collapse together to reduce join complexity.

At another client we got involved in schema design discussions between the devs, DBAs, and a data modeling expert. We convinced them to slightly de-normalize their perfectly normalized (from a data modeling perspective) new schema to avoid similar SQL Server performance issues.

And part of the discussion was that it didn't matter how normalized the schema would be because the Entity Framework layer would abstract that all away from the developers.... don't even get me started talking about the performance problems we've seen from taking that approach!

Call to action this time: if you're involved with designing a new schema, make sure you take a step back and look at things from SQL Server's perspective to ensure you don't end up with a schema that's inherently going to lead to poor performance. This is all a big generalization and very much an 'it depends', but do make sure you consider it. Schema *changes* are very hard to make...

I'm really interested to know your views on schema design and performance—feel free to drop me a line, confidentially as always.

Video Demo

One of the issues that we all know about is tempdb contention. I've blogged many times about this and there's a ton of other information out there too on how to reduce page latch contention in tempdb. However, there's one point of contention you can't do anything about so in the demo video (7 mins) this time I'm going to show you how to spot it.

I produced the video in WMV and MOV formats so everyone can watch. You can get the videos:

- For WMV: here
- For MOV: here

I recommend downloading before watching. And you can get the demo code here.

SQLskills Offerings

Our four Immersion Events in Bellevue, Washington in August are filling up (IE1 and IE2 are almost sold-out) and a few hardy souls are even taking all four classes in a row – wow! See here for more details.

We have a new combined SQL Server and failover clustering health check available – give us 8 hours and we'll show you what's going wrong with your system, without needing access to your system. We're fast, efficient, and provide serious ROI on your consulting investment. Let me know if you're interested – special rates for Insiders.

Summary

I hope you've enjoyed this issue - I 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