Why Is My DBA So Grumpy?

WHEN I USE NOLOCK AND OTHER HINTS

Rick Lowe

rick@data-flowe.com

DataFLowe

http://dataflowe.wordpress.com/

Microsoft CERTIFIED Master SQL Server® 2008

Microsoft CERTIFIED Solutions Master Charter - Data Platform

Beginning at the End

- Hints should be rare
- Rare is not the same as _never_. Most hints exist for a reason.
- Best added as a response to verified issues
- IMO, should always be accompanied by a comment

What Is NOLOCK

- Tells SQL Server we care about a fast response more than we care about a consistent response
- Specifically, suppresses the shared locks for queries
- Ignored by insert/update/delete statements (fortunately)
- NOT a “turbo button”
- More like “running with scissors” mode

How Blocking Usually Happens

- Writers block other writers
- Writers block readers
- Readers block writers

Why NOLOCK Is Tempting

- READ COMMITTED is the default
- Readers take shared locks
- Writers take exclusive locks
- Shared and exclusive locks not compatible

Contributing Factors

- Past issues cleared up with NOLOCK
- Belief NOLOCK means … _NO_ lock
- Bad code samples / misinformation on internet
- Lock escalation concerns
- Belief dirty reads are usually OK

Philosophical NOLOCK Issues

- Usually not necessary
- Possible to read uncommitted data
- If persisted (e.g. ETL), corruption can spread

Practical NOLOCK Issues

- Possible to miss records as they move
- Possible to double-count records as they move
- Unpredictable results when joining

Demo 1: NOLOCK

Eliminating NOLOCK

- If hints don't actually prevent blocking, simply remove
- If blocking exists, consider optimistic concurrency (row versioning)
  - Read Committed Snapshot Isolation (RCSI) makes row versioning the default behavior rather than blocking for the DB
  - If RCSI seems risky, consider enabling snapshot isolation level. If snapshot is active, row versioning can be requested as needed.

About Optimistic Concurrency

- SQL Server 2005 or later
- Once enabled, all writes save version (uses TempDB)
- Need a little headroom in TempDB
- Internal version tags will increase index fragmenation

ROWLOCK

- Mostly harmless, more annoying than problematic
- Sometimes seen due to concern about lock escalation
  - In cases where lock escalation is driven by amount of memory used by locks, may delay lock escalation
  - With modern hardware, not likely to make a difference
- Can force use of row locks rather than page locks
  - If MSSQL is using page lock is there a more interesting problem?
  - Is it a heap? Adding clustered index may be good idea anyway

RECOMPILE

- Legitimate uses do exist
- “I think this is going to be hairy” isn’t necessarily one of them
- When used correctly, great way to handle parameter sensitivity

RECOMPILE Side Effects

- Interferes with plan cache
- Increases compilation (CPU use)
- Did I mention … interferes with plan cache (query history)
- Less problematic at query level than stored procedure level

OPTIMIZE FOR

- Like RECOMPILE, helps with parameter sensitivity
- Plan will be cached (reused), but assumptions about parameters are changed
- Overuse is less problematic than overuse of RECOMPILE … but parameter sniffing is usually your friend
- Plan is likely to be fair for most values, not optimal
- Possible no single plan will work for every set of parameters
- If optimizing for specific value, data distribution may change

Thank You