**Avoid using NOLOCK on SQL Server UPDATE and DELETE statements**

By: [Aaron Bertrand](https://www.mssqltips.com/sqlserverauthor/49/aaron-bertrand/)

FROM: <https://www.mssqltips.com/sqlservertip/3172/avoid-using-nolock-on-sql-server-update-and-delete-statements/?utm_source=dailynewsletter&utm_medium=email&utm_content=headline&utm_campaign=20181026>

**Problem**

Occasionally I come across UPDATE and DELETE statements where the target SQL Server table is referenced with the [NOLOCK hint](https://www.mssqltips.com/sqlservertip/2470/understanding-the-sql-server-nolock-hint/). Does this hint help or hurt performance in this case? Should I keep using this hint, or should I remove it from these DML statements?

**Solution**

There are definitely a couple of issues you should be aware of when using NOLOCK against the target table of a DML statement. To be clear, the pattern I'm talking about is this one:

UPDATE t1 SET t1.x = something

FROM dbo.t1 WITH (NOLOCK)

INNER JOIN ...;

-- or

DELETE t1 FROM dbo.t1 WITH (NOLOCK)

WHERE EXISTS (SELECT ...);

-- or

;WITH x AS (SELECT x,y FROM dbo.t1 WITH (NOLOCK))

UPDATE x SET ...;

Of course, you can't even try using NOLOCK with [MERGE](https://www.mssqltips.com/sqlservertip/1704/using-merge-in-sql-server-to-insert-update-and-delete-at-the-same-time/); SQL Server immediately complains with the following error:

Msg 1065, Level 15, State 1  
The NOLOCK and READUNCOMMITTED lock hints are not allowed for target tables of INSERT, UPDATE, DELETE or MERGE statements.

And in spite of the wording of the message, the same error is not raised in any of the other patterns I demonstrated above.

So what are the problems?

If SQL Server doesn't raise an error message, then this must be okay, right? This depends on your definition of "okay" and, more importantly, how future-proof you want your code to be. I'll briefly explain three reasons these NOLOCK hints shouldn't be in your DML statements:

**The hint is a no-op**

NOLOCK has been ignored in this scenario since SQL Server 2005. So code that uses this hint is implying some sort of non-blocking behavior that doesn't actually happen (and, frankly, isn't possible, when you think about it).

I ran some tests on SQL Server 2008 and confirmed that the locking behavior (at least according to [sys.dm\_tran\_locks](https://www.mssqltips.com/sqlservertip/1968/understanding-sql-server-locking/)) is identical, with or without the hint. And yes, other statements were still blocked while the update was running (unless they were [SELECTs](https://www.mssqltips.com/sqlservertutorial/10/select-command-for-sql-server/) which also used NOLOCK, allowing for dirty reads in either scenario - again, this was true with or without the hint on the UPDATE).

**The syntax is deprecated**

In the SQL Server Books Online topic, [Deprecated Database Engine Features in SQL Server 2005](http://technet.microsoft.com/en-us/library/ms143729(v=sql.90).aspx), you can find the following deprecation notice:

*"Specifying NOLOCK or READUNCOMMITTED in the FROM clause of an UPDATE or DELETE statement when applied to the target table of the statement."*

You can also find the following quote in the topic [Table Hint (Transact-SQL)](http://msdn.microsoft.com/en-us/library/ms187373(v=sql.90).aspx):

*"Support for use of the READUNCOMMITTED and NOLOCK hints in the FROM clause that apply to the target table of an UPDATE or DELETE statement will be removed in a future version of SQL Server. Avoid using these hints in this context in new development work, and plan to modify applications that currently use them."*

This means that, while your code does not raise any errors today, it may start to fail in a future version of SQL Server. Nobody wants to carry forward code that has the potential to break through no other change than an upgrade.

**Index corruption can occur**

Yes, you read that right: using this hint in a DML statement can corrupt an index in certain versions of SQL Server 2008, 2008 R2, and 2012. This problem is described in the following Knowledge Base article:

* + [KB #2878968 : FIX: Nonclustered index corruption may occur when you run a complex UPDATE statement together with a NOLOCK hint against a table in SQL Server 2008, SQL Server 2008 R2 or SQL Server 2012](http://support.microsoft.com/kb/2878968)

There is a fix available, but currently only through a cumulative update (no base service pack or release level among these three versions has the fix). So if you are in a shop that waits for full service packs, you may be vulnerable to this issue until SQL Server 2012 Service Pack 2 is released (and I am fairly confident that will happen long before a new service pack is released for 2008 or 2008 R2, which hits end of mainstream support later this year). For those people, the only way to be sure you are immune from the issue is to remove NOLOCK from the target of any UPDATE statements.

There is another related issue, if you have a view that is comprised of SELECT statements with NOLOCK, and you try to perform DML against the view:

* + [KB #2248999 : FIX: An index corruption occurs when you update a view in Microsoft SQL Server 2008 or in Microsoft SQL Server 2008 R2](http://support.microsoft.com/kb/2248999)

This one has a smaller vector and does not affect SQL Server 2012, but still could be a concern in your environment.

Conclusion

I hope that I have convinced you that NOLOCK has no place as a hint against the target table in a DML statement. As a broad first attempt to identify this pattern in your code, I'll offer this query, with the disclaimer that it is very likely to produce false positives, won't necessarily identify UPDATE statements that are built with dynamic SQL, and obviously won't find any offending statements in external applications.

SELECT s.name, o.name, m.definition

FROM sys.schemas AS s

INNER JOIN sys.objects AS o

ON s.[schema\_id] = o.[schema\_id]

INNER JOIN sys.sql\_modules AS m

ON o.[object\_id] = m.[object\_id]

WHERE UPPER(m.[definition]) LIKE N'%UPDATE%SET%FROM%NOLOCK%'

OR UPPER(m.[definition]) LIKE N'%DELETE%FROM%NOLOCK%'

OR UPPER(m.[definition]) LIKE N'%WITH%(%SELECT%NOLOCK%)%DELETE%'

OR UPPER(m.[definition]) LIKE N'%WITH%(%SELECT%NOLOCK%)%UPDATE%';

**Next Steps**

* Identify and correct any code in your codebase where you are using the NOLOCK hint against the target of an UPDATE or DELETE statement.
* Consider investigating the possibility of using Snapshot or Read Committed Snapshot Isolation instead of NOLOCK for the read portion of your workload.
* Read the following tips and other resources:
  + [Understanding the SQL Server NOLOCK hint](https://www.mssqltips.com/sqlservertip/2470/understanding-the-sql-server-nolock-hint/)
  + [Execute SQL Server SELECT command without locking](https://www.mssqltips.com/sqlservertip/1646/execute-sql-server-select-command-without-locking/)
  + [Remove some SQL Server blocking issues with the NOLOCK hint](https://www.mssqltips.com/sqlservertip/1232/remove-some-sql-server-blocking-issues-with-the-nolock-hint/)
  + [Kendra Little's guide to implementing Snapshot or Read Committed Snapshot Isolation](http://www.brentozar.com/archive/2013/01/implementing-snapshot-or-read-committed-snapshot-isolation-in-sql-server-a-guide/)

Last Update: 2014-02-24