**بسمه تعالی**



برنامه نویسی بانک اطلاعاتی SQL Server

**نویسنده : محمد حسین فخرآوری**

1. **Transaction Isolation Levels چیست؟**
2. **آیا Nolock هیچ وقت باعث ایجاد Blockها نمی شود؟**
3. **آیا Read Committed ، نمی تواند Lockها را نگه دارد؟**

**Transaction Isolation Levels چیست؟**

آیا تا به حال به مشکلات سمت SQL Server فکر کرده اید؟؟؟ آیا تا به حال به مشکل سرعت در سمت دیتابیس برخورد کرده اید ؟؟!! خوب پس تا کنون با رفتارهای Locking و Blocking مواجه شده اید. زمانی که شما یک Query بد دارید شاید به دلیل Index گذاری نادرست روی عملکرد کل دیتابیس شما تاثیر خواهد گذاشت.

آیا سعی کرده اید که یک Transaction جدید  بدون Commit کردن بنویسید؟؟؟!!! تمام جدول شما Lock خواهد شد و دیتابیس شما درست در جلو چشمانتان نابود خواهد شد پس پیشنهاد من ، هرگز امتحانش نکنید .

زمانی که در مورد Locking و Blocking در دیتابیس های Relational صحبت می کنیم به ناچار مجبوریم در مورد Transaction Isolation Levelهای متفاوت نیز صحبت کنیم.SQL Server دو نوع مدل همزمانی را پشتیبانی می کند ، اولی بدبین قدیمی و دومی خوشبین جدید.

امروز می خواهم روی مدل قدیمی تمرکز کنم.در مدل همزمانی بدبین قدیمی SQL Server از ۴ Transaction Isolation Level متفاوت پشتیبانی می کند:

Read Uncommitted  
Read Committed  
Repeatable Read  
Serializable

در کل روی SQL چند نوع Transaction می توانیم داشته باشیم ، Transactionهایی که فقط اطلاعات را می خوانند

(SELECT Queries) و همچنین Transactionهایی که روی اطلاعات

تغییر انجام می دهند (INSERT , UPDATE , DELETE , MERGE) . هر دفعه که شما اطلاعات را می خوانید SQL  از Shared Lock – S استفاده می کند و هر دفعه که شما در اطلاعات تغییری ایجاد می کنید SQL  از Exclusive Locks – X استفاده می کند. هر دو نوع Lock با همدیگر ناسازگار می باشند و این بدان معناست که آن هایی که اطلاعات را می خوانند (Reader) آن هایی که اطلاعات رو تغییر می دهند (Writer) ، لاک می کنند و بالعکس.

حال با کمک Transaction Isolation Levels می توان مدت زمانی که Readerها S Lockها را نگه می دارند را تحت کنترل درآورد. Writerها همیشه از X Lockها استفاده می کنند و نمی توان آن ها را تحت تاثیر قرار داد.SQL Server به صورت پیش فرض از Isolation Level ، Read Committed استفاده می کند و این بدان معناست که SQL Server زمانی که شما اطلاعات را می خوانید از Lock S استفاده می کند و وقتی رکورد خوانده می شود Lock S رها می شود. وقتی رکوردها را سطر به سط می خوانید این Lock S سطر به سطر گذاشته شده و برداشته می شود.

حال هر زمان که بخواهید که Readerها S Lock ایجاد نکنند می توانید از دستور Read Uncommitted استفاده کنید.Read Uncommitted بدان معناست که شما می توانید اطلاعات کثیف یعنی اطلاعاتی که هنوز Commit نشده اند را بخوانید. خوب این خیلی سریع می باشد ، کسی نمی تواند شما را Block کند ولی از طرفی دیگر بسیار خطرناک می باشد زیرا اطلاعاتی را می خوانید که هنوز Commit نشده اند. یک لحظه به این فکر کنید که اطلاعات Transaction ی را خوانده اید که به هر دلیل Commit آن دچار مشکل شده و Rollback شده است ، حال شما اطلاعاتی در دست دارید که اصلا در دیتابیس شما وجود ندارند.

در Read Committed شما می توانید از Non Repeatable Reads استفاده کنید ، چون وقتی اطلاعات را دوبار در Transaction می خوانید باز هم کسی می توانداطلاعات را تغییر داده باشد. اگر می خواهید از Non Repeatable Read جلوگیری کنید می توانید از Repeatable Read استفاده کنید. در Repeatable Read کاری که SQL Server انجام می دهد این است که S Lockها را تا Commit شدن و یا Rollback شدن Transaction نگهداری می کند که این بدان معناست که کسی نمی تواند اطلاعات خوانده شده ی شما را تغییر دهد.

در هر مبحثی که تا به حال در مورد Isolation Levelها داشتیم کل صحبت بر سر رکوردهایی بود که ممکن است به دلیل Commit نشدن تغییر کند و یا اصلا وجود نداشته باشد ، به این رکوردها Phantom Records گویند. اگر می خواهید از شر این رکوردها لاص شوید می توانید Isolation Levelی به نام Serilizable استفاده کنید که استفاده از آن معمولا ممنوع می باشد چون در این حالت SQL Server برای حذف این نوع رکورد ها از Lockی به نام Key Range Locking بهره می برد و این یعنی شما یک دسته از اطلاعات را Lock می کنید در نتیجه هیچ Transaction دیگری نمی تواند در آن دسته از اطلاعات Insertی انجام دهد .

خوب با توجه به این بحث متوجه شدیم هرچقدر Isolation Level شما سخت گیرانه تر باشد در کار دیتابیس شما خلل بیشتری ایجاد می کند ، همیشه از Read Uncommitted استفاده کردن و یا بالعکس همیشه از Serializable استفاده کردن راه حل عاقلانه ای نمی باشد و بهتر است استفاده از این ها به صورت علاقلانه و در سناریوهای مختلف متفاوت باشد.

**آیا Nolock هیچ وقت باعث ایجاد Blockها نمی شود؟**

همانطور که قبل تر هم گفتم استفاده از Nolock کار شما را سریع تر می کند چون Select Query شما نمی تواند Block شود ولی متاسفانه Nolock هم زمانی که از عبارات DLL عباراتی مانند Alter Table استفاده می شود ، می تواند خودش عامل ایجاد Lock باشد.

وقتی شما از عبارت Alter Table استفاده می کنید این یعنی اینکه می خواهید ساختار جدول (Schema) را تغییر دهید ، در همچین مواقعی SQL Server از نوع خاصی از Lock به نام Schema Modification Lock (Sch-M) روی آن جدول خاص بهره می برد. حال زمانی که شما از یک دستور Select ساده روی همان جدول استفاده می کنید SQL Server قبل آن باید Physical Execution Plan را برای انتخاب بهترین و سریع ترین راه برای اجرای Query ببیند در نتیجه می بایست از یک Lock به نام Schema Stability Lock (Sch-S) بهره ببرد ، این Lock  همانطور که از اسمش پیداست بدین معناست که در طول اجرا کردن یک Select Query ساده می بایست ساختار یا همان Schema جدول پایدار و یا Stable باشد.

خوب با کمی دقت متوجه می شویم که این دو Lock با هم در تناقض می باشند ، یکی به دنبال پایدار بودن ساختار جدول می باشد ولی دیگر به دنبال تغییر ساختار جدول می باشد و اینچنین می شود که دستور Nolock شما می تواند خودش هم Lock ایجاد کند. حالا به این مثال دقت کنید ، من می خواهم جدولی بسازم و چند رکورد در آن Insert کنم :

|  |
| --- |
| CREATE TABLE TestTable  (  Column1 INT,  Column2 INT,  Column3 INT  )  GO  DECLARE @i INT = 0  WHILE (@i < 10000)  BEGIN  INSERT INTO TestTable VALUES (@i, @i + 1, @i + 2)  SET @i += 1  END  GO |

حال بعد از آن با باز کردن یک Transaction می خواهیم ساختار یک جدول را تغییر دهیم

|  |
| --- |
| BEGIN TRANSACTION  -- DDL statements require a Sch-M lock on the objects that are modified.  -- In this case, the table "TestTable" gets a Sch-M lock (Schema modification lock)!  ALTER TABLE TestTable ADD Column4 INT |

همانطور که می بینید Transaction هنوز Commit نشده و در جریان می باشد ، حال بیایید یک Session دیگر باز کنیم

|  |
| --- |
| -- The statement is now blocking, even with the NOLOCK query hint!  -- SQL Server has to compile the query, and requests a Sch-S lock (Schema Stability lock).  -- This lock is incompatible with the Sch-M lock!  SELECT \* FROM TestTable WITH (NOLOCK) |

در این حالت SQL Server به خاطر اینکه دستور شما Block یک Transaction دیگر شده است نتیجه ای نخواهد داشت حتی با استفاده از Nolock. برای یافتن ریشه این مشکل می توانید از DMV sys.dm\_tran\_locks برای هر دو Session استفاده کنید و می بینید که Sch-M Lock چیزی است که Sch-S Lock را Block کرده است.

در نتیجه تا اینجا متوجه شدیم که Nolock و یا حتی Read Uncommitted سریع و بدون دغدغه اجرا شدن Query شما را گارانتی نمی کنند.

* SELECT \* FROM sys.dm\_tran\_locks

**آیا Read Committed ، نمی تواند Lockها را نگه دارد؟**

همان طور که قبلا هم صحبت کردیم Read Committed یک S Lock روی رکوردی که دارید اطلاعاتش را می خوانید ایجاد می کند و پس از خوانده شدن آن رکورد Lock نیز برداشته می شود در نتیجه این Lock تا زمانی باقی خواهد ماند که رکورد خوندنش تمام نشده است. اما این حالت استثنا نیز دارد زمانی که شما در Execution Plan نیاز به مرتب سازی اطلاعات دارید و این بدان معناست که SQL Server می بایست یک کپی از اطلاعات شما ایجاد کند.

بعد از اینکه اطلاعات مورد نیاز کپی شد دیگر نیازی به واکشی اطلاعات از جدول و یا Index وجود ندارد. ولی کپی کردن اطلاعات زمانی که رکوردهای کمی در اختیار داریم ساده به نظر می رسد ، جدولی را در نظر بگیرید که ستونی از نوع VARCHAR(Max) دارد ، این بدان معناست که در آن ستون می توان تا ۲ گیگابایت اطلاعات را ذخیره نمود و خوب کپی کردن این حجم از اطلاعات حافضه و TempDb شما را از بین خواهد برد.

برای جلوگیری از مشکل اینکه بقیه Session ها توسط این کپی Block نشوند ، SQL Server این S Lock را تا پاین نگه می دارد تا در این مدت کسی نتواند اطلاعات را تغییر دهد.در نتیجه Transaction مانند زمانی که از Isolation Level Repeatable Read استفاده می شود رفتار می کند که این حالت عملکرد دیتابیس شما را تحت تاثیر قرار می دهد و در این حالت SQL  اطلاعات دست نخورده را در اختیار شما قرار خواهد داد با اینکه از عبارت Read Committed استفاده کرده بودید.

هر آنچه مورد بررسی قرار گرفت بیانگر این حقیقت است که Transaction Isolation Levelها هم گاهی اوقات می توانند بر خلاف معنای خود عمل کنند و باید سناریوهای مختلف را با دقت بررسی کنید تا کاملا متوجه بشوید ، در این مسیر DMVها بسیار کارآمد خواهند بود و کمک زیادی به شما خواهند کرد.