عنوان: **بررسی Transactions و Locks در SQL Server**
نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۲/۰۳ ۹:۳۰
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, T-SQL, Database, development

مقدمه SQL Server، با هر تقاضا به عنوان یک واحد مستقل رفتار می‌کند. در وضعیت‌های پیچیده ای که فعالیت‌ها توسط مجموعه ای از دستورات SQL انجام می‌شود، به طوری که یا همه باید اجرا شوند یا هیچکدام اجرا نشوند، این روش مناسب نیست. در چنین وضعیت هایی، نه تنها تقاضاهای موجود در یک دنباله به یکدیگر بستگی دارند، بلکه شکست یکی از تقاضاهای موجود در دنباله، به معنای این است که کل تقاضاهای موجود در دنباله باید لغو شوند، و تغییرات حاصل از تقاضاهای اجراشده در آن دنباله خنثی شوند تا بانک اطلاعاتی به حالت قبلی برگردد.

1- تراکنش چیست؟ تراکنش شامل مجموعه ای از یک یا چند دستور SQL است که به عنوان یک واحد عمل می‌کنند. اگر یک دستور SQL در این واحد با موفقیت اجرا نشود، کل آن واحد خنثی می‌شود و داده هایی که در اجرای آن واحد تغییر کرده اند، به حالت اول برگردانده می‌شود. بنابراین تراکنش وقتی موفق است که هر یک از دستورات آن با موفقیت اجرا شوند. برای درک مفهوم تراکنش مثال زیر را در نظر بگیرید: سهامدار A در معامله ای 400 سهم از شرکتی را به سهامدار B می‌فروشد. در این سیستم، معامله وقتی کامل می‌شود که حساب سهامدار A به اندازه 400 بدهکار و حساب سهامدار B همزمان به اندازه 400 بستانکار شود. اگر هر کدام از این مراحل با شکست مواجه شود، معامله انجام نمی‌شود.

2- خواص تراکنش هر تراکنش دارای چهار خاصیت است (معروف به ACID) که به شرح زیر می‌باشند:

2-1- خاصیت یکپارچگی (Atomicity) یکپارچگی به معنای این است که تراکنش باید به عنوان یک واحد منسجم (غیر قابل تفکیک) در نظر گرفته شود. در مثال مربوط به مبادله سهام، یکپارچگی به معنای این است که فروش سهام توسط سهامدار A و خرید آن سهام توسط سهامدار B، مستقل از هم قابل انجام نیستند و برای این که تراکنش کامل شود، هر دو عمل باید با موفقیت انجام شوند.
اجرای یکپارچه، یک عمل "همه یا هیچ" است. در عملیات یکپارچه، اگر هر کدام از دستورات موجود در تراکنش با شکست مواجه شوند، اجرای تمام دستورات قبلی خنثی می‌شود تا به جامعیت بانک اطلاعاتی آسیب نرسد.

2-2- خاصیت سازگاری (Consistency) سازگاری زمانی وجود دارد که هر تراکنش، سیستم را در یک حالت سازگار قرار دهد (چه تراکنش به طور کامل انجام شود و چه در اثر وجود خطایی خنثی گردد). در مثال مبادله سهام، سازگاری به معنای آن است که هر بدهکاری مربوط به حساب فروشنده، موجب همان میزان بستانکاری در حساب خریدار می‌شود.
در SQL Server، سازگاری با راهکار ثبت فایل سابقه انجام می‌گیرد که تمام تغییرات را در بانک اطلاعاتی ذخیره می‌کند و جزییات را برای ترمیم تراکنش ثبت می‌نماید. اگر سیستم در اثنای اجرای تراکنش خراب شود، فرآیند ترمیم SQL Server با استفاده از این اطلاعات، تعیین می‌کند که آیا تراکنش با موفقیت انجام شده است یا خیر، و در صورت عدم موفقیت آن را خنثی می‌کند. خاصیت سازگاری تضمین می‌کند که بانک اطلاعاتی هیچگاه تراکنش‌های ناقص را نشان نمی‌دهد.

2-3- خاصیت تفکیک (Isolation) تفکیک موجب می‌شود هر تراکنش در فضای خودش و جدا از سایر تراکنش‌های دیگری که در سیستم انجام می‌گیرد، اجرا شود و نتایج هر تراکنش فقط در صورت کامل شدن آن قابل مشاهده است. اگر چندین تراکنش همزمان در سیستم در حال اجرا باشند، اصل تفکیک تضمین می‌کند که اثرات یک تراکنش تا کامل شدن آن، قابل مشاهده نیست. در مثال مربوط به مبادله سهام، اصل تفکیک به معنای این است که تراکنش بین دو سهامدار، مستقل از تمام تراکنش‌های دیگری است که در سیستم به مبادله سهام می‌پردازند و اثر آن وقتی برای افراد قابل مشاهده است که آن تراکنش کامل شده باشد. این اصل در مواردی که سیستم همزمان از چندین کاربر پشتیبانی می‌کند، مفید است.

2-4- پایداری (Durability) پایداری به معنای این است که تغییرات حاصل از نهایی شدن تراکنش، حتی در صورت خرابی سیستم نیز پایدار می‌ماند. اغلب سیستم‌های مدیریت بانک اطلاعاتی رابطه ای، از طریق ثبت تمام فعالیت‌های تغییر دهنده‌ی داده‌ها در بانک اطلاعاتی، پایداری را تضمین می‌کنند. در صورت خرابی سیستم یا رسانه ذخیره سازی داده ها، سیستم قادر است آخرین

بهنگام سازی موفق را هنگام راه اندازی مجدد، بازیابی کند. در مثال مربوط به مبادله سهام، پایداری به معنای این است که وقتی انتقال سهام از سهامدار A به B با موفقیت انجام گردید، حتی اگر سیستم بعداً خراب شد، باید نتیجه‌ی آن را منعکس سازد.

3- مشکلات همزمانی(Concurrency Effects):

3-1- Dirty Read: زمانی روی می‌دهد که تراکنشی رکوردی را می‌خواند، که بخشی از تراکنشی است که هنوز تکمیل نشده است، اگر آن تراکنش Rollback شود اطلاعاتی از بانک اطلاعاتی دارید که هرگز روی نداده است.
اگر سطح جداسازی تراکنش (پیش فرض) Read Committed باشد، این مشکل بوجود نمی‌آید.

3-2- Non-Repeatable Read: زمانی ایجاد می‌شود که رکوردی را دو بار در یک تراکنش می‌خوانید و در این اثنا یک تراکنش مجزای دیگر داده‌ها را تغییر می‌دهد. برای پیشگیری از این مسئله باید سطح جداسازی تراکنش برابر با Repeatable Read یا Serializable باشد.

3-3- Phantoms: با رکوردهای مرموزی سروکار داریم که گویی تحت تاثیر عبارات Update و Delete صادر شده قرار نگرفته اند. به طور خلاصه شخصی عبارت Insert را درست در زمانی که Update مان در حال اجرا بوده انجام داده است، و با توجه به اینکه ردیف جدیدی بوده و قفلی وجود نداشته، به خوبی انجام شده است. تنها چاره این مشکل تنظیم سطح Serializable است و در این صورت بهنگام رسانی‌های جداول نباید درون بخش Where قرار گیرد، در غیر این صورت Lock خواهند شد.

3-4- Lost Update: زمانی روی می‌دهد که یک Update به طور موفقیت آمیزی در بانک اطلاعاتی نوشته می‌شود، اما به طور اتفاقی توسط تراکنش دیگری بازنویسی می‌شود. راه حل این مشکل بستگی به کد شما دارد و بایست به نحوی تشخیص دهید، بین زمانی که داده‌ها را می‌خوانید و زمانی که می‌خواهید آنرا بهنگام کنید، اتصال دیگری رکورد شما را بهنگام کرده است.

4- منابع قابل قفل شدن 6 منبع قابل قفل شدن برای SQL Server وجود دارد و آن‌ها سلسله مراتبی را تشکیل می‌دهند. هر چه سطح قفل بالاتر باشد، Granularity کمتری دارد. در ترتیب آبشاری Granularity عبارتند از:

- **Database:** کل بانک اطلاعاتی قفل شده است، معمولاً طی تغییرات Schema بانک اطلاعاتی روی می‌دهد.
- **Table:** کل جدول قفل شده است، شامل همه اشیای مرتبط با جدول.
- **Extent:** کل Extent (متشکل از هشت Page) قفل شده است.
- **Page:** همه داده‌ها یا کلیدهای Index در آن Page قفل شده اند.
- **Key:** قفلی در کلید مشخصی یا مجموعه کلید هایی Index وجود دارد. ممکن است سایر کلیدها در همان Index Page تحت تاثیر قرار نگیرند.
- **Row or Row Identifier (RID):** هر چند قفل از لحاظ فنی در Row Identifier قرار می‌گیرد ولی اساساً کل ردیف را قفل می‌کند.

5- تسریع قفل (Lock Escalation) و تاثیرات قفل روی عملکرد اگر تعداد آیتم‌های قفل شده کم باشد نگهداری سطح بهتری از Granularity (مثلاً RID به جای Page) معنی دار است. هرچند با افزایش تعداد آیتم‌های قفل شده، سربار مرتبط با نگهداری آن قفل‌ها در واقع باعث کاهش عملکرد می‌شود، و می‌تواند باعث شود قفل به مدت طولانی‌تری در محل باشد(هر چه قفل به مدت طولانی‌تری در محل باشد، احتمال این که شخصی آن رکورد خاص را بخواهد بیشتر است).
هنگامی که تعداد قفل نگهداری شده به آستانه خاصی برسد آن گاه قفل به بالاترین سطح بعدی افزایش می‌یابد و قفل‌های سطح پایین‌تر نباید به شدت مدیریت شوند (آزاد کردن منابع و کمک به سرعت در مجادله).
توجه شود که تسریع مبتنی بر تعداد قفل هاست و نه تعداد کاربران.

6- حالات قفل (Lock Modes): همانطور که دامنه وسیعی از منابع برای قفل شدن وجود دارد، دامنه ای از حالات قفل نیز وجود دارد.

6-1- Shared Locks (S): زمانی استفاده می‌شود، که فقط باید داده‌ها را بخوانید، یعنی هیچ تغییری ایجاد نخواهید کرد. Shared Lock با سایر Shared Lockهای دیگر سازگار است، البته قفل‌های دیگری هستند که با Shared Lock سازگار نیستند. یکی از کارهایی که Shared Lock انجام می‌دهد، ممانعت از انجام Dirty Read از طرف کاربران است.

6-2- Exclusive Locks (X): این قفل‌ها با هیچ قفل دیگری سازگار نیستند. اگر قفل دیگری وجود داشته باشد، نمی‌توان به Exclusive Lock دست یافت و همچنین در حالی که Exclusive Lock فعال باشد، به هر قفل جدیدی از هر شکل اجازه ایجاد شدن در منبع را نمی‌دهند.
این قفل از اینکه دو نفر همزمان به حذف کردن، بهنگام رسانی و یا هر کار دیگری مبادرت ورزند، پیشگیری می‌کند.

6-3- Update Locks (U): این قفل ها نوعی پیوند میان Shared Locks و Exclusive Locks هستند.
برای انجام Update باید بخش Where را (در صورت وجود) تایید اعتبار کنید، تا دریابید فقط چه ردیف هایی را می‌خواهید بهنگام رسانی کنید. این بدان معنی است که فقط به Shared Lock نیاز دارید، تا زمانی که واقعاً بهنگام رسانی فیزیکی را انجام دهید. در زمان بهنگام سازی فیزیکی نیاز به Exclusive Lock دارید.
Update Lock نشان دهنده این واقعیت است که دو مرحله مجزا در بهنگام رسانی وجود دارد، Shared Lock ای دارید که در حال تبدیل شدن به Exclusive Lock است. Update Lock تمامی Update Lockهای دیگر را از تولید شدن باز می‌دارند، و همچنین فقط با Shared Lock و Intent Shared Lockها سازگار هستند.

6-4- Intent Locks: با سلسله مراتب شی سر و کار دارد. بدون Intent Lock، اشیای سطح بالاتر نمی‌دانند چه قفلی را در سطح پایین‌تر داشته اید. این قفل‌ها کارایی را افزایش می‌دهند و 3 نوع هستند:

6-4-1- Intent Shared Lock (IS: Shared Lock) در نقطه پایین‌تری در سلسله مراتب، تولید شده یا در شرف تولید است. این نوع قفل تنها به Table و Page اعمال می‌شود.

6-4-2- Intent Exclusive Lock (IX): همانند Intent Shared Lock است اما در شرف قرار گرفتن در آیتم سطح پایین‌تر است.

6-4-3- Shared With Intent Exclusive (SIX: Shared Lock) در پایین سلسله مراتب شی تولید شده یا در شرف تولید است اما Intent Lock قصد اصلاح داده‌ها را دارد بنابراین در نقطه مشخصی تبدیل به Intent Exclusive Lock می‌شود.

6-5- Schema Locks: به دو شکل هستند:

6-5-1- Schema Modification Lock (Sch-M): تغییر Schema به شی اعمال شده است. هیچ پرس و جویی یا سایر عبارت‌های Create، Alter و Drop نمی‌توانند در مورد این شی در مدت قفل Sch-M اجرا شوند. با همه حالات قفل ناسازگار است.

6-5-2- Schema Stability Lock (Sch-S): بسیار شبیه به Shared Lock است، هدف اصلی این قفل پیشگیری از Sch-M است وقتی که قبلاً قفل هایی برای سایر پرس و جو-ها (یا عبارت‌های Create، Alter و Drop) در شی فعال شده اند. این قفل با تمامی انواع دیگر قفل سازگار است به جز با Sch-M.

6-6- Bulk Update Locks (BU): این قفل‌ها بارگذاری موازی داده‌ها را امکان پذیر می‌کنند، یعنی جدول در مورد هر فعالیت نرمال (عبارات T-SQL) قفل می‌شود، اما چندین عمل bcp یا Bulk Insert را می‌توان در همان زمان انجام داد. این قفل فقط با Sch-S و سایر قفل هایBU سازگار است.

7- سطوح جداسازی (Isolation Level):
7-1- Read Committed (وضعیت پیش فرض): با Read Committed همه Shared Lockهای ایجاد شده، به محض اینکه عبارت ایجاد کننده آنها تکمیل شود، به طور خودکار آزاد می‌شوند. به طور خلاصه قفل‌های مرتبط با عبارت Select به محض تکمیل عبارت Select آزاد می‌شوند و SQL Server منتظر پایان تراکنش نمی‌ماند. اگر تراکنش پرس و جویی را انجام می‌دهد که داده‌ها را اصلاح می‌کند (Insert، Delete و Update) قفل‌ها برای مدت تراکنش نگه داشته می‌شوند.
با این سطح پیش فرض، می‌توانید مطمئن شوید جامعیت کافی برای پیشگیری از Dirty Read دارید، اما همچنان Phantoms و Non-Repeatable Read می‌تواند روی دهد.

7-2- Read Uncommitted: خطرناک‌ترین گزینه از میان تمامی گزینه‌ها است، اما بالاترین عملکرد را به لحاظ سرعت دارد. در واقع با این تنظیم سطح تجربه همه مسائل متعدد هم زمانی مانند Dirty Read امکان پذیر است. در واقع با تنظیم این سطح به SQL Server اعلام می‌کنیم هیچ قفلی را تنظیم نکرده و به هیچ قفلی اعتنا نکند، بنابراین هیچ تراکنش دیگری را مسدود نمی‌کنیم.
می‌توانید همین اثر Read Uncommitted را با اضافه کردن نکته بهینه ساز **NOLOCK** در پرس و جوها بدست آورید.

7-3- Repeatable Read: سطح جداسازی را تا حدودی افزایش می‌دهد و سطح اضافی محافظت همزمانی را با پیشگیری از Dirty Read و همچنین Non-Repeatable Read فراهم می‌کند.
پیشگیری از Non-Repeatable Read بسیار مفید است اما حتی نگه داشتن Shared Lock تا زمان پایان تراکنش می‌تواند دسترسی کاربران به اشیا را مسدود کند، بنابراین به بهره وری لطمه وارد می‌کند.
نکته بهینه ساز برای این سطح **REPEATEABLEREAD** است.

7-4- Serializable: این سطح از تمام مسائل هم زمانی پیشگیری می‌کند به جز برای Lost Update.
این تنظیم سطح به واقع بالاترین سطح آنچه را که سازگاری نامیده می‌شود، برای پایگاه داده فراهم می‌کند. در واقع فرآیند بهنگام رسانی برای کاربران مختلف به طور یکسان عمل می‌کند به گونه ای که اگر همه کاربران یک تراکنش را در یک زمان اجرا می‌کردند، این گونه می‌شد « پردازش امور به طور سریالی».
با استفاده از نکته بهینه ساز **SERIALIZABLE** یا **HOLDLOCK** در پرس و جو شبیه سازی می‌شود.

7-5- Snapshot: جدترین سطح جداسازی است که در نسخه 2005 اضافه شد، که شبیه ترکیبی از Read Committed و Read Uncommitted است. به طور پیش فرض در دسترس نیست، در صورتی در دسترس است که گزینه ALLOW_SNAPSHOT_ISOLATION برای بانک اطلاعاتی فعال شده باشد.(برای هر بانک اطلاعاتی موجود در تراکنش)
Snapshot مشابه Read Uncommitted هیچ قفلی ایجاد نمی‌کند. تفاوت اصلی آن‌ها در این است که تغییرات صورت گرفته در بانک اطلاعاتی را در زمان‌های متفاوت تشخیص می‌دهند. هر تغییر در بانک اطلاعاتی بدون توجه به زمان یا Commit شدن آن، توسط پرس و جو هایی که سطح جداسازی Read Uncommitted را اجرا می‌کنند، دیده می‌شود. با Snapshot فقط تغییراتی که قبل از شروع تراکنش، Commit شده اند، مشاهده می‌شود.
از شروع تراکنش Snapshot، تمامی داده‌ها دقیقاً مشاهده می‌شوند، زیرا در شروع تراکنش Commit شده اند.
نکته: در حالی که Snapshot توجهی به قفل‌ها و تنظیمات آنها ندارد، یک حالت خاص وجود دارد. چنانچه هنگام انجام Snapshot یک عمل Rollback (بازیافت) بانک اطلاعاتی در جریان باشد، تراکنش Snapshot قفل‌های خاصی را برای عمل کردن به عنوان یک مکان نگهدار و سپس انتظار برای تکمیل Rollback تنظیم می‌کند. به محض تکمیل Rollback، قفل حذف شده و Snapshot به طور طبیعی به جلو حرکت خواهد کرد.

Isolation level	Dirty read	Non-Repeatable read	Phantom
Read uncommitted	Yes	Yes	Yes
Read committed	No	Yes	Yes
Repeatable read	No	No	Yes
Snapshot	No	No	No
Serializable	No	No	No

نظرات خوانندگان

نویسنده: محمد
تاریخ: ۱۳۹۳/۰۲/۰۵ ۱۲:۳۷

سلام ، خسته نباشید. ممنون از زحماتی که برای این مقاله کشیدید.
من یه سوال داشتم درباره تراکنش (Transaction) : من توی پروژه ام از sqlTransaction و isolation سریالایز استفاده کردم در این بلاک، 9 جدول من مورد عملیات درج و یا ویرایش قرار می‌گیره (و البته دو وب سرویس نیز صدا زده می‌شه) و معمولا 2 یا 3 ثانیه برای ثبت این تراکنش زمان صرف میشه . تا اینجا مشکلی نیست ولی اگر چند کاربر مثلا 5 نفر این عملیات انجام بدن در بعضی از مواقع (به ندرت) به ترتیب عملیات برای هر نفر انجام میشه (همین توقع از سریالایز می‌ره). ولی در اکثر مواقع خطا deadlock می‌ده و یا خطا timeout و همچنین جاهای مختلف برنامه که به یکی از این 9 جدول select می‌شه به انها هم همین خطاها رو می‌ده . لطفا در صورت امکان راه حلی برای مشکل من بگین . ممنون

نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۲/۰۵ ۱۵:۴۵

با سلام و سپاس از محبت تان، چند نکته به ذهن من میرسد، که شاید راهگشا باشد:
- تعداد دستورات درون بلاک Transaction تا جایی که میسر است، کمینه باشد.
- عملیات مربوط به وب سرویس، خارج از Transaction انجام گیرد. (به عنوان یک نکته در کار با Transactionباید عملیات Input و Output خارج از Transaction انجام گیرد.)
- طبیعتا هر چه درجه Isolation بالاتر باشد، Lock سختگیرانه‌تر برخورد می‌کند.
- برای مشکل بن بست، اگر می‌توانید ترتیب در اختیار گرفتن جداول را (پس از شناسائی محل بن بست) تغییر دهید و یا از جداول Temp استفاده کنید.
- چنانچه در قسمت هایی از برنامه همانطور که اشاره کردید فقط نیاز به خواندن مقادیر جداول دارید، از بهینه ساز پرس و جوی nolock استفاده کنید. برای مثال:

```
select * from tbl with (NOLOCK)
```