عنوان: **ماندگاری با تاخیر در SQL Server 2014**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۱۴ ۱۳:۳۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, Performance, In-Memory OLTP

به صورت پیش فرض SQL Server از روش write-ahead log - WAL استفاده می‌کند. به این معنا که کلیه تغییرات، پیش از commit نهایی باید در لاگ فایل آن نوشته شوند. این مساله با تعداد بالای تراکنش‌ها تا حدودی بر روی سرعت سیستم می‌تواند تاثیرگذار باشد. برای بهبود این وضعیت، در SQL Server 2014 قابلیتی به نام delayed_durability اضافه شده‌است که با فعال سازی آن، کلیه اعمال مرتبط با لاگ‌های تراکنش‌ها به صورت غیرهمزمان انجام می‌شوند. به این ترتیب تراکنش‌ها زودتر از معمول به پایان خواهد رسید؛ با این فرض که نوشته شدن تغییرات در لاگ فایل‌ها، در آینده‌ای محتمل انجام خواهند شد. این مساله به معنای فدا کردن D در ACID است (Atomicity, Consistency, Isolation, Durability). البته باید دقت داشت که رسیدن به ACID کامل هزینه‌بر است و شاید خیلی از اوقات تمام اجزای آن نیازی نباشند یا حتی بتوان با اندکی تخفیف آن‌ها را اعمال کرد؛ مانند D به تاخیر افتاده.

برای اینکار SQL Server از یک بافر 60 کیلوبایتی برای ذخیره سازی اطلاعات لاگ‌هایی که قرار است به صورت غیرهمزمان با تراکنش‌ها نوشته شوند، استفاده می‌کند. هر زمان که این 60KB پر شد، آن‌را flush کرده و ثبت خواهد نمود. به این ترتیب به دو مزیت خواهیم رسید:

- پردازش تراکنش‌ها بدون منتظر شدن جهت commit نهایی در دیسک سخت ادامه خواهند یافت. صبر کمتر به معنای امکان پردازش تراکنش‌های بیشتری در یک سیستم پر ترافیک است.
- با توجه به بافری که از آن صحبت شد، اینبار اعمال Write به صورت یک سری batch اعمال می‌شوند که کارآیی و سرعت بیشتری نسبت به حالت تکی دارند.

اندکی تاریخچه

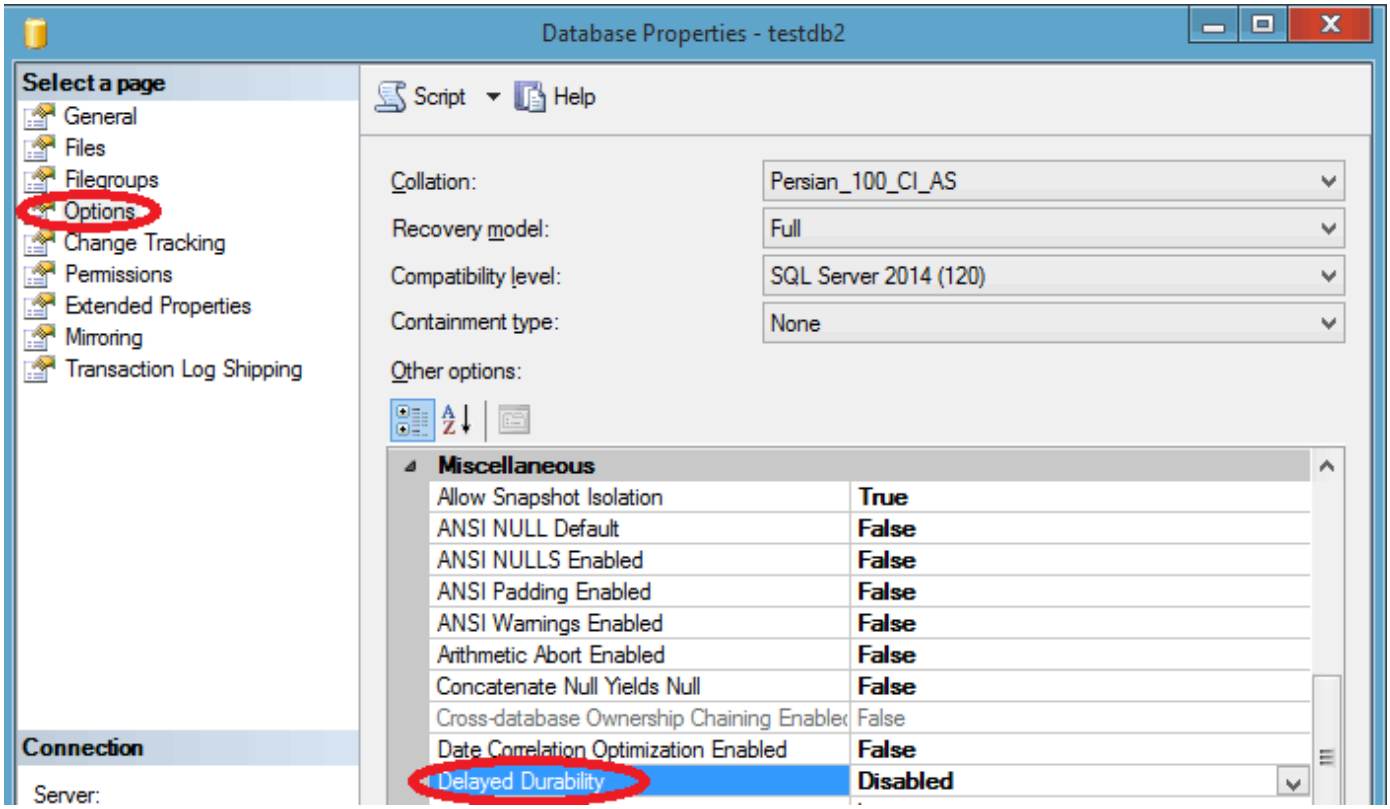
ایده یک چنین عملی 28 سال قبل توسط Hal Berenson ارائه شده‌است! اوراکل آن‌را در سال 2006 تحت عنوان Asynchronous Commit پیاده سازی کرد و مایکروسافت در سال 2014 آن‌را ارائه داده‌است.

فعال سازی ماندگاری غیرهمزمان در SQL Server

فعال سازی این قابلیت در سطح بانک اطلاعاتی، در سطح یک تراکنش مشخص و یا در سطح رویه‌های ذخیره شده کامپایل شده مخصوص OLTP درون حافظه‌ای، میسر است.

برای فعال سازی ماندگاری با تاخیر در سطح یک دیتابیس، خواهیم داشت:

```
ALTER DATABASE dbname SET DELAYED_DURABILITY = DISABLED | ALLOWED | FORCED;
```



در اینجا اگر ALLOWED را انتخاب کنید، به این معنا است که لاگ کلیه تراکنش‌های مرتبط با این بانک اطلاعاتی به صورت غیرهمزمان نوشته می‌شوند. حالت FORCED نیز دقیقا به همین معنا است با این تفاوت که اگر حالت ALLOWED انتخاب شود، تراکنش‌های ماندگار (آن‌هایی که به صورت دستی DELAYED_DURABILITY را غیرفعال کرده‌اند)، سبب flush کلیه تراکنش‌هایی با ماندگاری به تاخیر افتاده خواهند شد و سپس اجرا می‌شوند. در حالت Forced تنظیم دسترسی DELAYED_DURABILITY = OFF در سطح تراکنش‌ها تاثیری نخواهد داشت؛ اما در حالت ALLOWED این مساله به صورت دستی در سطح یک تراکنش قابل لغو است.
البته باید توجه داشت، صرفنظر از این تنظیمات، یک سری از تراکنش‌ها همیشه ماندگار هستند و بدون تاخیر؛ مانند تراکنش‌های سیستمی، تراکنش‌های بین دو یا چند بانک اطلاعاتی و کلیه تراکنش‌هایی که با FileTable، Change Data Capture و Change Tracking سر و کار دارند.

در سطح تراکنش‌های می‌توان نوشت:

```
COMMIT TRANSACTION WITH (DELAYED_DURABILITY = ON);
```

و یا در رویه‌های ذخیره شده کامپایل شده مخصوص OLTP درون حافظه‌ای خواهیم داشت:

```
BEGIN ATOMIC WITH (DELAYED_DURABILITY = ON, ...)
```

سؤال: آیا فعال سازی DELAYED_DURABILITY بر روی مباحث locking و isolation levels تاثیر دارند؟
پاسخ: خیر. کلیه تنظیمات قفل گذاری‌ها همانند قبل و بر اساس isolation levels تعیین شده، رخ خواهند داد. تنها تفاوت در اینجا است که با فعال سازی DELAYED_DURABILITY، کار commit بدون صبر کردن برای پایان نوشته شدن اطلاعات در لاگ سیستم صورت می‌گیرد. به این ترتیب قفل‌های انجام شده زودتر آزاد خواهند شد.

سؤال: میزان از دست دادن اطلاعات احتمالی در این روش چقدر است؟
در صورتیکه سرور کرش کند یا ری‌استارت شود، حداکثر به اندازه‌ی 60KB اطلاعات را از دست خواهید داد (اندازه‌ی بافری که برای اینکار درنظر گرفته شده‌است). البته عنوان شده‌است که اگر ری‌استارت یا خاموشی سرور، از پیش تعیین شده باشد، ابتدا

کلیه لاگ‌های flush نشده، ذخیره شده و سپس ادامه‌ی کار صورت خواهد گرفت؛ ولی زیاد به آن اطمینان نکنید. اما همواره با فراخوانی sys.sp_flush_log، می‌توان به صورت دستی بافر لاگ‌های سیستم را flush کرد.

یک آزمایش

در ادامه قصد داریم یک جدول جدید را در بانک اطلاعاتی آزمایشی testdb2 ایجاد کنیم. سپس یکبار تنظیم DELAYED_DURABILITY = FORCED را انجام داده و 10 هزار رکورد را ثبت می‌کنیم و بار دیگر DELAYED_DURABILITY = DISABLED را تنظیم کرده و همین عملیات را تکرار خواهیم کرد:

```
CREATE TABLE tblData(
    ID INT IDENTITY(1, 1),
    Data1 VARCHAR(50),
    Data2 INT
);
CREATE CLUSTERED INDEX PK_tblData ON tblData(ID);
CREATE NONCLUSTERED INDEX IX_tblData_Data2 ON tblData(Data2);

-------------------------

alter database testdb2 SET DELAYED_DURABILITY = FORCED;

-------------------------

SET NOCOUNT ON
Print 'DELAYED_DURABILITY = FORCED'
DECLARE @counter AS INT = 0
DECLARE @start datetime = getdate()
WHILE (@counter < 10000)
BEGIN
      INSERT INTO tblData (Data1, Data2) VALUES('My Data', @counter)
      SET @counter += 1
END
Print DATEDIFF(ms,@start,getdate());
GO

-------------------------

alter database testdb2 SET DELAYED_DURABILITY = DISABLED;
truncate table tblData;
-------------------------

SET NOCOUNT ON
Print 'DELAYED_DURABILITY = DISABLED'
DECLARE @counter AS INT = 0
DECLARE @start datetime = getdate()
WHILE (@counter < 10000)
BEGIN
      INSERT INTO tblData (Data1, Data2) VALUES('My Data', @counter)
      SET @counter += 1
END
Print DATEDIFF(ms,@start,getdate());
GO

-----------------------
```

با این خروجی:

```
DELAYED_DURABILITY = FORCED
666
DELAYED_DURABILITY = DISABLED
2883
```

در این آزمایش، سرعت insertها در حالت DELAYED_DURABILITY = FORCED حدود 4 برابر است نسبت به حالت معمولی.

برای مطالعه بیشتر

Control Transaction Durability

SQL Server 2014 Delayed Durability/Lazy Commit

Delayed Durability in SQL Server 2014 – Part 1

Is In-Memory OLTP Always a silver bullet for achieving better transactional speed

Delayed Durability in SQL Server 2014