عنوان مقاله: تاثیر ایندکس های Computed-Column روی دستور DBCC CHECKDB

نویسنده مقاله: تورج عزیزی

تاریخ انتشار: ۲ اردیبهشت ۱۴۰۳

منبع: https://nikamooz.com/impact-of-computed-column-indexes-on-the-dbcc-checkdb-command/

DBCC CHECKDB یکی از دستورات SQL Server است که برای بررسی صحت و یکپارچگی پایگاه داده استفاده می‌شود. این دستور، یکپارچگی منطقی و فیزیکی تمامی اشیای موجود در پایگاه داده مربوطه را موردبررسی قرار می‌دهد. درصورت تشخیص هرگونه مشکل، DBCC CHECKDB همچنین می‌تواند گزینه‌هایی را برای بهبود به شما پیشنهاد دهد. ممکن است بنا به دلایلی همچون گسترده‌بودن دیتابیس‌ها، پیچیدگی Schema، محدودیت سخت‌افزاری و سایر موارد، اجرای دستور DBCC CHECKDB به کندی انجام شود. در این مقاله، به بررسی تأثیر وجود ایندکس Non-clustered روی Computed Column بر زمان اجرای دستور DBCC CHECKDB بررسی شده است. برای آن که بهترین نتایج از مطالعه این مطلب حاصل شود، لازم است دانش کافی از مفاهیمی همچون DBCC CHECKDB ، Non-clustered Index، Thread، Latch و Hashing داشته باشید.

چالش در شرایط زیر اتفاق می‌افتد:

یک ایندکس Non-clustered داریم که یک Computed Column به‌عنوان بخشی از کلید ایندکس یا به‌عنوان یکی از ستون‌های INCLUDED وجود دارد. این موضوع بر زمان اجرای دستورات DBCC CHECKTABLE DBCC CHECKTABLE ، DBCC CHECKDB و DBCC CHECKFILEGROUP تأثیر می‌گذارد.

به‌عنوان پیش‌زمینه، گزیده‌ای از کتاب SQL Server 2012 Internals را در ادامه قرار داده‌ایم. این کتاب درمورد چگونگی بررسی ایندکس‌های Non-Clustered برای تضمین ثبات دیتا (Consistency) در دیتابیس است.

DBCC CHECKDB در هنگام اجرا، آزمایشی روی ایندکس‌های Non-Clustered انجام می‌دهد که الگوریتم cross-checks نامیده می‌شوند. این الگوریتم صحت موارد زیر را بررسی می‌کند:

- هر رکورد موجود در یک ایندکس Non-clustered (Filtered یا NonFiltered) باید یک رکورد معادل (Map) در جدول base داشته باشد (یعنی Heap Clustered Table).
- هر رکورد موجود در جدول base باید دقیقاً به یک رکورد در ایندکس Non-Clustered فیلترنشده و همچنین، دقیقاً به یک رکورد به ازای هر ایندکس فیلترشده (به شرطی که دارای شرایط فیلتر باشد) Map شود.

مکانیزم انجام این آزمایش به‌طور مؤثری در هر release از SQL Server 7.0 تاکنون تغییر کرده است «به این معنی که در هر نسخه جدید، کارآمدتر شده است. در SQL Server 2012 دو جدول Hash به ازای هر پارتیشن از ایندکس Non-clustered ایجاد می‌شود.» یک جدول Hash برای رکوردهای موجود (Actual Records) در آن پارتیشن

بخصوص و یک جدول Hash به ازای رکوردهایی که باید در این پارتیشن وجود داشته باشند (Should-Exist Records) (که از روی جدول base ساخته می‌شود.)

زمانی که یک رکورد در ایندکس Non-Clustered پردازش می‌شود، تمام ستون‌های تشکیل‌دهنده کلید آن به یک مقدار BIGINT ، هَش (Hash) می‌شوند، علاوه‌بر ستون‌های کلید، موارد زیر نیز به‌طور جداگانه به مقدار BIGINT ، HASH خواهند شد:

- لینک فیزیکی یا منطقی که به جدول base ارجاع دارد و با RID جدول base شناخته می‌شود.
- تمامی ستون‌های Included «حتی مقادیر LOB و FILESTREAM» همگی به یک مقدار BIGINT ، HASH می‌شوند.
- دو مقدار Hash باهم جمع می‌شوند و یک مقدار نهایی به ازای هر رکورد از ایندکس Non-Clustered به نام Master Hash Value تولید می‌کنند.

زمانی که یک رکورد در جدول base پردازش می‌شود، الگوریتم زیر به ازای هر رکورد موجود در ایندکس Non-Clustered که باید به ازای این رکورد داده وجود داشته باشد، اجرا می‌شود. این موضوع با درنظرگرفتن شرط فیلتر برای filtered non-clustered index ها انجام خواهد شد:

۱. رکورد موجود در ایندکس Non-Clustered را در حافظه ایجاد کرده و مجدد کلید RID جدول base و ستون‌های Included را در کنار سایر فیلدها قرار دهید.

۲. تمامی مقادیر ستون‌ها را به یک مقدار BIGINT ، Hash کنید.

۳. مقدار حاصل را به مقدار Master Hash Value رکوردی اضافه کنید که باید وجود داشته باشد.

فرضیه‌ای وجود دارد که الگوریتم cross-checks بر آن استوار است: آن فرضیه این است که اگر خرابی وجود نداشته باشد، باید مقدار Master Hash Value به‌دست آمده از رکوردهای موجود و مقدار Master Hash Value به‌دست آمده از رکوردهایی که باید وجود داشته باشند، به‌طور دقیق مطابقت داشته باشند.

زمانی که روی یک ستون Computed ایندکس Non-Clustered تعریف شده باشد یا عضوی از ستون‌های این ایندکس باشد، باید مقدار ستون Computed برمبنای فرمول تعریف شده محاسبه شود. برای انجام این کار، مکانیزمی به نام Expression Evaluator ایجاد می‌شود. این مکانیزم توسط کد Query Processor اجرا می‌شود و رفتار آن خارج از کنترل DBCC CHECKDB است. مانعی در این‌جا وجود دارد؛ Thread ای که Expression Evaluator را اجرا می‌کند، باید یک Exclusive Latch را تا انتهای اجرا در اختیار داشته باشد. همین موضوع باعث بروز Bottleneck و افت شدید در Performance می‌شود.

سناریوی تست

یک آزمایش روی سیستمی که SQL Server 2012 SP1 CU3 نصب‌شده با مشخصات سخت‌افزاری زیر اجرا شد:

- Dell R720
- ۶۴GB of memory
- ۲E5-2670 CPUs with 8 physical cores and hyperthreading enabled

دیتابیس تستی AdventureWorks با اندازه ۵۰۰ GB وجود دارد که روی ۸ data file روی دو درایو Fusion-io 320GB توزیع‌شده و tempdb و لاگ آن روی دو درایو Fusion-io 320GB دیگر ایجاد شده است. این Config برای حذف‌wait ناشی از IO درنظر گرفته شده است.

چندین تست اولیه DBCC CHECKDB با تنظیمات Parallel نامحدود اجرا کرده‌ایم که در حدود ۳۴۰ دقیقه زمان اجرا برده است. این زمان اجرا، بسیار کند به‌نظر می‌رسد و باعث شد تا به اجرای تست دیگری بپردازیم و به خروجی sys.dm_os_waiting_tasks توجه کنیم. با انجام این کار، مشخص شد هربار نیمی از Thread ها، منتظر به‌دست آوردن DBCC_OBJECT_METADATA latch بوده‌اند:

```
WaitType        Wait_S   Resource_S  Signal_S  WaitCount  Percentage
AvgWait_S  AvgRes_S  AvgSig_S
--------------- ---------- ----------- --------- ---------- ----------- --
-------- --------- ---------
CXPACKET        684482.45  682667.13   1815.32   60294236   34.90
۰.۰۱۱۴     ۰.۰۱۱۳  ۰.۰۰۰۰
OLEDB           659325.08  659325.08   0.00      207661462  33.62
0.0032    ۰.۰۰۳۲  ۰.۰۰۰۰
LATCH_EX        615168.39  605357.28   9811.11   798224634  31.37
۰.۰۰۰۸     ۰.۰۰۰۸  ۰.۰۰۰۰

LatchClass               Wait_S   WaitCount  Percentage  AvgWait_S
------------------------ ---------- ---------- ----------- ----------
DBCC_OBJECT_METADATA   ۶۱۱۷۶۸.۰۰  ۷۶۴۸۴۵۶۳۶  ۹۹.۱۶      ۰.۰۰۰۸
```

این خروجی منطقی به‌نظر نمی‌رسید؛ زیرا نمی‌توان باور کرد که چنین Bottleneck شدیدی (تقریباً ۱ ms در هر latch wait و ۸۰۰ میلیون wait!) وجود داشته باشد. با اجرای کوئری زیر، به جستجوی ایندکس‌های Non-Clustered دارای Computed Column می‌پردازیم:

```
SELECT
    [s].[name],
    [o].[name],
    [i].[name],
    [co].[name],
    [ic].*
FROM sys.columns [co]
JOIN sys.index_columns [ic]
    ON [ic].[object_id] = [co].[object_id]
    AND [ic].[column_id] = [co].[column_id]
JOIN sys.indexes [i]
    ON [i].[object_id] = [ic].[object_id]
    AND [i].[index_id] = [ic].[index_id]
JOIN sys.objects [o]
    ON [i].[object_id] = [o].[object_id]
JOIN sys.schemas [s]
    ON [o].[schema_id] = [s].[schema_id]
WHERE [co].[is_computed] = 1;
GO
```

۶ ایندکس Non-Clustered دارای Computed Column در برخی جداول بزرگ دیتابیس یافت شده است. پس از غیرفعال‌کردن این ایندکس‌ها، مجدداً تست‌ها را اجرا کرده و نتیجه آن به شکل زیر است:

- ۱۷-۱۸ دقیقه در هربار اجرا

Bottleneck ناشی از Expression Evaluator منجربه اجرای ۲۰ برابر کندتر DBCC CHECKDB شده بود. جزئیات wait و latch قابل مشاهده است:

```
WaitType        Wait_S   Resource_S  Signal_S  WaitCount  Percentage
AvgWait_S  AvgRes_S  AvgSig_S
--------------- ---------- ---------- --------- ---------- ---------- --
-------- --------- ---------
CXPACKET        33064.56   29331.74  ۳۷۳۲.۸۳   ۴۲۰۳۴۵۳۳   ۴۸.۱۴
۰.۰۰۰۸   ۰.۰۰۰۷  ۰.۰۰۰۱
OLEDB           28883.78   28883.78    0.00    ۱۷۳۹۴۰۱۵۴  ۴۲.۰۵      ۰.۰۰۰۲
۰.۰۰۰۲  ۰.۰۰۰۰
LATCH_EX        5021.20  ۴۶۰۵.۵۰     ۴۱۵.۷۰  ۳۰۳۴۰۶۵۹   ۷.۳۱       ۰.۰۰۰۲
۰.۰۰۰۲  ۰.۰۰۰۰

LatchClass               Wait_S   WaitCount  Percentage  AvgWait_S
------------------------ ---------- ---------- ----------- ----------
DBCC_CHECK_AGGREGATE     5039.36  ۳۰۲۶۷۴۵۱   ۹۸.۸۲      ۰.۰۰۰۲
```

راهی بجز غیرفعال‌کردن ایندکس‌های Non-Clustered روی Computed Column ها در هنگام اجرای DBCC CHECKDB و Rebuild آن‌ها پس از اتمام اجرا وجود ندارد. البته این روش، راهکار مناسبی نیست. در نسخه SQL Server 2016 به بعد، DBCC CHECKDB از این آزمایش‌ها برای چنین ایندکس‌هایی صرف نظر می‌کند، مگر اینکه از گزینه WITH EXTENDED_LOGICAL_CHECKS استفاده کرده باشید.

جمع بندی

در این مقاله، به بررسی تأثیرات ایندکس‌های Computed Column پرداختیم. دستور DBCC CHECKDB در SQL Server به‌منظور بررسی یکپارچگی منطقی و فیزیکی داده‌ها و ساختارهای داده‌ای مورد استفاده قرار می‌گیرد. ایندکس‌هایی که برروی Computed Column تعریف می‌شوند، می‌توانند تأثیر قابل توجهی بر عملکرد این دستور داشته باشند. در انتها نیز پیشنهاد می‌کنیم برای یادگیری بیشتر مباحث، به **آموزش SQL Server** مراجعه نمایید.