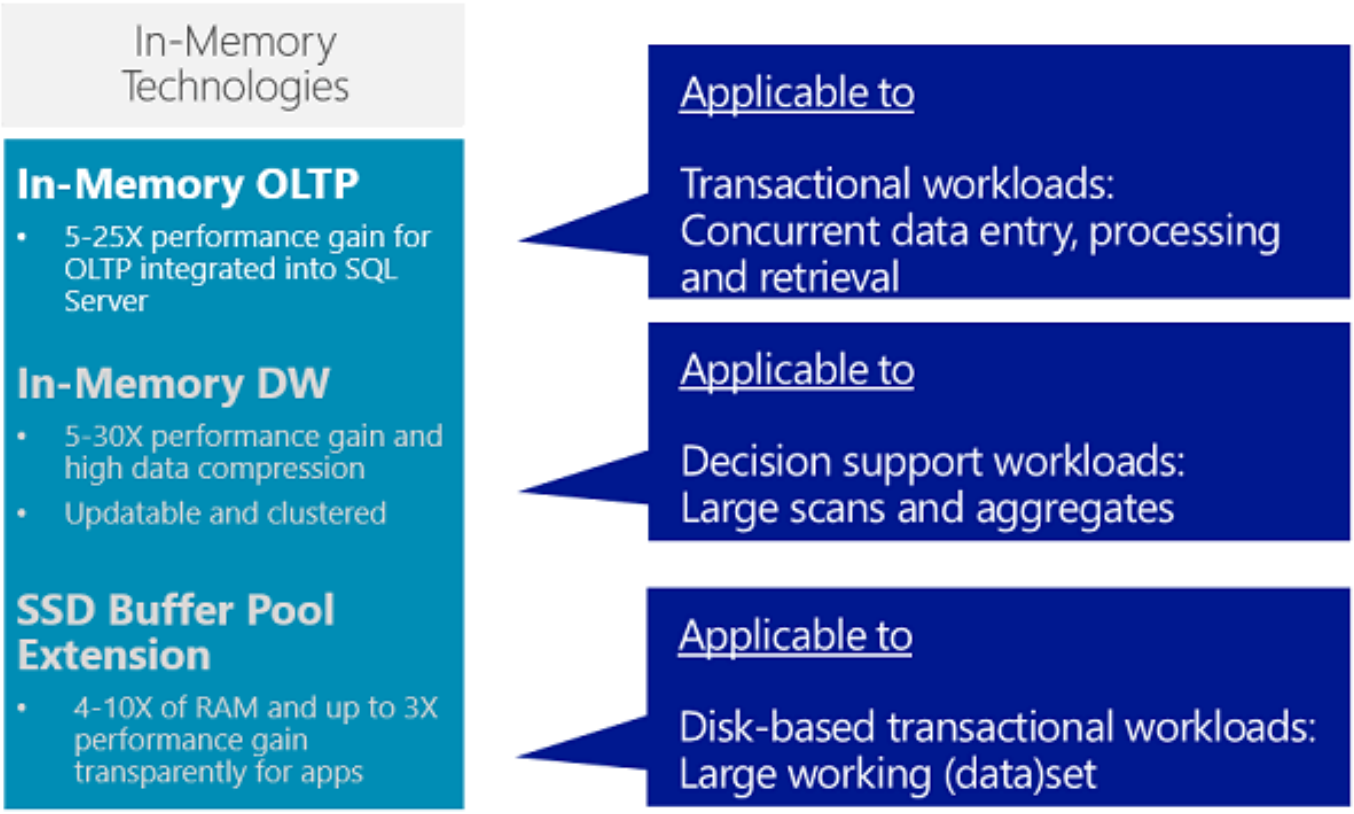
عنوان: **معرفی OLTP درون حافظه‌ای در SQL Server 2014**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۱۱ ۰:۵۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, In-Memory OLTP

OLTP درون حافظه‌ای، مهم‌ترین ویژگی جدید SQL Server 2014 است. موتور بانک اطلاعاتی disk based اس کیوال سرور، حدود 15 تا 20 سال قبل تهیه شده‌است و موتور جدید درون حافظه‌ای OLTP آن، بزرگترین بازنویسی این سیستم از زمان ارائه‌ی آن می‌باشد و شروع این پروژه به 5 سال قبل بر می‌گردد. علت تهیه‌ی آن نیز به نیازهای بالای پردازش‌های همزمان مصرف کنندگان این محصول در سال‌های اخیر، نسبت به 15 سال قبل مرتبط است. با استفاده از امکانات OLTP درون حافظه‌ای، امکان داشتن جداول معمولی disk based و جداول جدید memory optimized با هم در یک بانک اطلاعاتی میسر است؛ به همراه مهیا بودن تمام زیرساخت‌هایی مانند تهیه بک آپ، بازیابی آن‌ها، امنیت و غیره برای آن‌ها.



آیا جداول بهینه سازی شده‌ی برای حافظه، همان DBCC PINTABLE منسوخ شده هستند؟

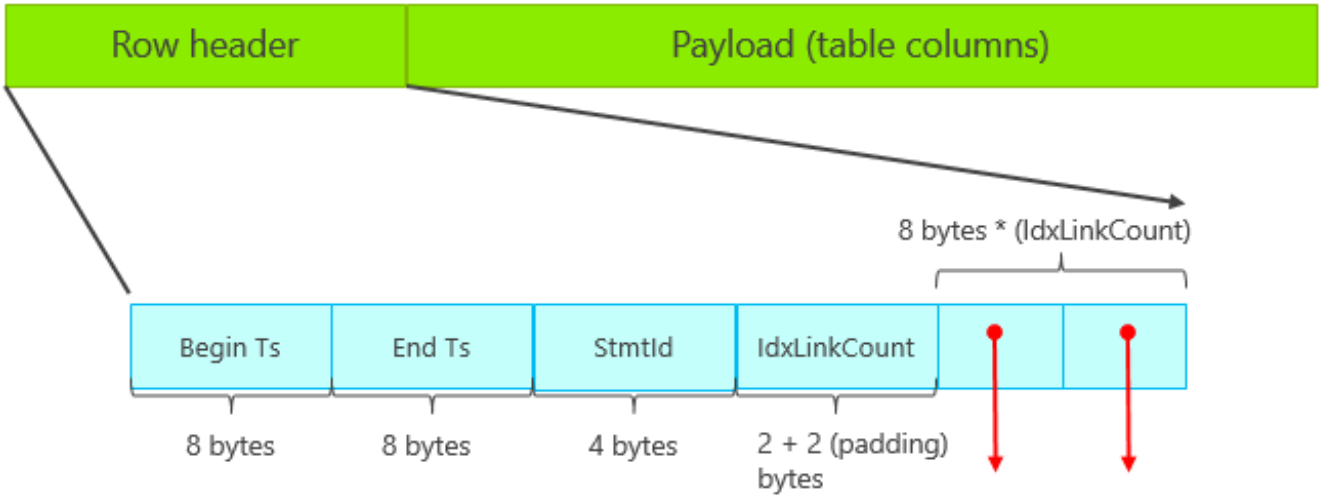
در نگارش‌های قدیمی‌تر اس کیوال سرور، دستوری وجود داشت به نام DBCC PINTABLE که سبب ثابت نگه داشتن صفحات جداول مبتنی بر دیسک یک دیتابیس، در حافظه می‌شد. به این ترتیب تمام خواندن‌های مرتبط با آن جدول، از حافظه صورت می‌گرفت. مشکل این روش که سبب منسوخ شدن آن گردید، اثرات جانبی آن بود؛ مانند خوانده شدن صفحات جدیدتر (با توجه به اینکه ساختار پردازشی و موتور بانک اطلاعاتی تغییری نکرده بود) و نیاز به حافظه‌ی بیشتر تا حدی که کل کش بافر سیستم را پر می‌کرد و امکان انجام سایر امور آن مختل می‌شدند. همچنین اولین ارجاعی به یک جدول، سبب قرار گرفتن کل آن در حافظه می‌گشت. به علاوه ساختار این سیستم نیز همانند روش مبتنی بر دیسک، بر اساس همان روش‌های قفل گذاری، ذخیره سازی اطلاعات و تهیه ایندکس‌های متداول بود.

اما جداول بهینه سازی شده‌ی برای حافظه، از یک موتور کاملا جدید استفاده می‌کنند؛ با ساختار جدیدی برای ذخیره سازی اطلاعات و تهیه ایندکس‌ها. دسترسی به اطلاعات آن‌ها شامل قفل گذاری‌های متداول نیست و در آن حداقل زمان دسترسی به

اطلاعات درنظر گرفته شده‌است. همچنین در آن‌ها data pages یا index pages و کش بافر نیز وجود ندارد.

نحوه‌ی ذخیره سازی و مدیریت اطلاعات جداول بهینه سازی شده برای حافظه

جداول بهینه سازی شده برای حافظه، فرمت ردیف‌های کاملا جدیدی را نیز به همراه دارند و جهت قرارگرفتن در حافظه ودسترسی سریع به آن‌ها بهینه سازی شده‌اند. برخلاف جداول مبتنی بر دیسک سخت که اطلاعات آن‌ها در یک سری صفحات خاص به نام‌های data or index pages ذخیره می‌شوند، اینگونه جداول، دارای ظروف مبتنی بر صفحه نیستند و از مفهوم چند نگارشی برای ذخیره سازی اطلاعات استفاده می‌کنند؛ به این معنا که ردیف‌ها به ازای هر تغییری، دارای یک نگارش جدید خواهند بود و بلافاصله در همان نگارش اصلی به روز رسانی نمی‌شوند.
در اینجا هر ردیف دارای یک timestamp شروع و یک timestamp پایان است. timestamp شروع بیانگر تراکنشی است که ردیف را ثبت کرده و timestamp پایان برای مشخص سازی تراکنشی بکار می‌رود که ردیف را حذف کرده است. اگر timestamp پایان، دارای مقدار بی‌نهایت باشد، به این معنا است که ردیف متناظر با آن هنوز حذف نشده‌است. به روز رسانی یک ردیف در اینجا، ترکیبی است از حذف یک ردیف موجود و ثبت ردیفی جدید. برای یک عملیات فقط خواندنی، تنها نگارش‌هایی که timestamp معتبری داشته باشند، قابل مشاهده خواهند بود و از مابقی صرفنظر می‌گردد.
در OLTP درون حافظه‌ای که از روش چندنگارشی همزمانی استفاده می‌کند، برای یک ردیف مشخص، ممکن است چندین نگارش وجود داشته باشند؛ بسته به تعداد باری که یک رکورد به روز رسانی شده‌است. در اینجا یک سیستم garbage collection همیشه فعال، نگارش‌هایی را که توسط هیچ تراکنشی مورد استفاده قرار نمی‌گیرند، به صورت خودکار حذف می‌کند؛ تا مشکل کمبود حافظه رخ ندهد.



آیا می‌توان به کارآیی جداول بهینه سازی شده برای حافظه با همان روش متداول مبتنی بر دیسک اما با بکارگیری حافظه‌ی بیشتر و استفاده از یک SSD RAID رسید؟

خیر! حتی اگر کل بانک اطلاعاتی مبتنی بر دیسک را در حافظه قرار دهید به کارآیی روش جداول بهینه سازی شده‌ی برای حافظه نخواهید رسید. زیرا در آن هنوز مفاهیمی مانند data pages و index pages به همراه یک buffer pool پیچیده وجود دارند. در روش‌های مبتنی بر دیسک، ردیف‌ها از طریق page id و row offset آن‌ها قابل دسترسی می‌شوند. اما در جداول بهینه سازی شده‌ی برای حافظه، ردیف‌های جداول با یک B-tree خاص به نام Bw-Tree در دسترس هستند.

میزان حافظه‌ی مورد نیاز برای جداول بهینه سازی شده‌ی برای حافظه

باید درنظر داشت که تمام جداول بهینه سازی شده‌ی برای حافظه، به صورت کامل در حافظه ذخیره خواهند شد. بنابراین بدیهی

است که نیاز به مقدار کافی حافظه در اینجا ضروری است. توصیه صورت گرفته، داشتن حافظه‌ای به میزان دو برابر اندازه‌ی اطلاعات است. البته در اینجا چون با یک سیستم هیبرید سر و کار داریم، حافظه‌ی کافی جهت کار buffer pool مختص به جداول مبتنی بر دیسک را نیز باید درنظر داشت.
همچنین اگر به اندازه‌ی کافی حافظه در سیستم تعبیه نشود، شاهد شکست مداوم تراکنش‌ها خواهید بود. به علاوه امکان بازیابی و restore جداول را نیز از دست خواهید داد.
البته لازم به ذکر است که اگر کل بانک اطلاعاتی شما چند ترابایت است، نیازی نیست به همین اندازه یا بیشتر حافظه تهیه کنید. فقط باید به اندازه‌ی جداولی که قرار است جهت قرار گرفتن در حافظه بهینه سازی شوند، حافظه تهیه کنید که حداکثر آن 256 گیگابایت است.

چه برنامه‌هایی بهتر است از امکانات OLTP درون حافظه‌ای SQL Server 2014 استفاده کنند؟

- برنامه‌هایی که در آن‌ها تعداد زیادی تراکنش کوتاه مدت وجود دارد به همراه درجه‌ی بالایی از تراکنش‌های همزمان توسط تعداد زیادی کاربر.
- اطلاعاتی که توسط برنامه زیاد مورد استفاده قرار می‌گیرند را نیز می‌توان در جداول بهینه سازی شده جهت حافظه قرار داد.
- زمانیکه نیاز به اعمال دارای write بسیار سریع و با تعداد زیاد است. چون در جداول بهینه سازی شده‌ی برای حافظه، صفحات داده‌ها و ایندکس‌ها وجود ندارند، نسبت به حالت مبتنی بر دیسک، بسیار سریعتر هستند. در روش‌های متداول، برای نوشتن اطلاعات در یک صفحه، مباحث همزمانی و قفل‌گذاری آن‌را باید در نظر داشت. در صورتیکه در روش بهینه سازی شده‌ی برای حافظه، به صورت پیش فرض از حالتی همانند snapshot isolation و همزمانی مبتنی بر نگارش‌های مختلف رکورد استفاده می‌شود.
- تنظیم و بهینه سازی جداولی با تعداد Read بالا. برای مثال، جداول پایه سیستم که اطلاعات تعاریف محصولات در آن قرار دارند. این نوع جداول عموما با تعداد Readهای بالا و تعداد Write کم شناخته می‌شوند. چون طراحی جداول مبتنی بر حافظه از hash tables و اشاره‌گرهایی برای دسترسی به رکوردهای موجود استفاده می‌کند، اعمال Read آن نیز بسیار سریعتر از حالت معمول هستند.
- مناسب جهت کارهای data warehouse و ETL Staging Table. در جداول مبتنی بر حافظه امکان عدم ذخیره سازی اطلاعات بر روی دیسک سخت نیز پیش بینی شده‌است. در این حالت فقط اطلاعات ساختار جدول، ذخیره‌ی نهایی می‌گردد و اگر سرور نیز ری استارت گردد، مجددا می‌تواند اطلاعات خود را از منابع اصلی data warehouse تامین کند.

محدودیت‌های جداول بهینه سازی شده‌ی برای حافظه در SQL Server 2014

- تغیر اسکیما و ساختار جداول بهینه سازی شده‌ی برای حافظه مجاز نیست. به بیان دیگر دستور ALTER TABLE برای اینگونه جداول کاربردی ندارد. این مورد جهت ایندکس‌ها نیز صادق است. همان زمانیکه جدول ایجاد می‌شود، باید ایندکس آن نیز تعریف گردد و پس از آن این امکان وجود ندارد.
تنها راه تغییر اسکیمای اینگونه جداول، Drop و سپس ایجاد مجدد آن‌ها است.
البته باید درنظر داشت که SQL Server 2014، اولین نگارش این فناوری را ارائه داده‌است و در نگارش‌های بعدی آن، بسیاری از این محدودیت‌ها قرار است که برطرف شوند.
- جداول بهینه سازی شده‌ی برای حافظه حتما باید دارای یک ایندکس باشند. البته اگر یک primary key را برای آن‌ها تعریف نمائید، کفایت می‌کند.
- از unique indexها پشتیبانی نمی‌کند، مگر اینکه از نوع primary key باشد.
- حداکثر 8 ایندکس را می‌توان بر روی اینگونه جداول تعریف کرد.
- امکان تعریف ستون identity در آن وجود ندارد. اما می‌توان از قابلیت sequence برای رسیدن به آن استفاده کرد.
- DML triggers را پشتیبانی نمی‌کند.
- کلیدهای خارجی و قیود را پشتیبانی نمی‌کند.
- حداکثر اندازه‌ی یک ردیف آن 8060 بایت است. بنابراین از نوع‌های داده‌ای max دار و XML پشتیبانی نمی‌کند.
این مورد در حین ایجاد جدول بررسی شده و اگر اندازه‌ی ردیف محاسبه‌ی شده‌ی آن توسط SQL Server 2014 بیش از 8060 بایت باشد، جدول را ایجاد نخواهد کرد.

اگر سرور را ری استارت کنیم، چه اتفاقی برای اطلاعات جداول بهینه سازی شده‌ی برای حافظه رخ می‌دهد؟

حالت DURABILTY انتخاب شده‌ی در حین ایجاد جدول بهینه سازی شده‌ی برای حافظه، تعیین کننده‌ای این مساله است. اگر SCHEMA_ONLY انتخاب شده باشد، کل اطلاعات شما با ری استارت سرور از دست خواهد رفت؛ البته اطلاعات ساختار جدول حفظ خواهد گردید. اگر حالت SCHEMA_AND_DATA انتخاب شود، اطلاعات شما پس از ری‌استارت سرور نیز در دسترس خواهد بود. این اطلاعات به صورت خودکار از لاگ تراکنش‌ها بازیابی شده و مجددا در حافظه قرار می‌گیرند.
حالت SCHEMA_ONLY برای مصارف برنامه‌های data warehouse بیشتر کاربرد دارد. جایی که اطلاعات قرار است از منابع داده‌ی مختلفی تامین شوند.

برای مطالعه بیشتر

SQL Server 2014: NoSQL Speeds with Relational Capabilities
SQL Server 2014 In-Memory OLTP Architecture and Data Storage
Overview of Applications, Indexes and Limitations for SQL Server 2014 In-Memory OLTP Tables
Microsoft SQL Server 2014: In-Memory OLTP Overview
SQL Server in Memory OLTP for Database Developers
Exploring In-memory OLTP Engine (Hekaton) in SQL Server 2014 CTP1

نظرات خوانندگان

نویسنده: فرید طاهری
تاریخ: ۱۳۹۳/۰۳/۱۱ ۱۰:۳

جناب نصیری با تشکر از مقاله مفیدتون لازم می‌دونم در جهت تکمیل مباحث به چند نکته اشاره کنم

1- « *اگر حالت SCHEMA_AND_DATA انتخاب شود، اطلاعات شما پس از ری‌استارت سرور نیز در دسترس خواهد بود. این اطلاعات به صورت خودکار از لاگ تراکنش‌ها بازیابی شده و مجددا در حافظه قرار می‌گیرند* ».

بازیابی اطلاعات مربوط به تراکنش‌هایی که به ازای In Memory OLTP است بوسیله Data File + Delta File و Log File می‌باشد. در صورتیکه Schema_AND_Data را به ازای این نوع جداول فعال کنید داده‌های شما در Data File و داده‌های حذف شده در Delta File ثبت می‌گردد. مکانیزم Log File برای In Memory OLTP همچنان مانند جداول Disk base وجود دارد اما با بهینه سازی مناسب مانند ثبت Log Record کمتر به ازای عملیات کاربران و...

2- در جایی دیگر در متن اشاره شده که In Memory OLTP اجازه استفاده از Identity را به کاربر نمی‌دهد باید اشاره کنم که این موضوع برای نسخه CTP بوده است در نسخه RTM این قابلیت وجود دارد . لازم می‌دانم اشاره کنم که در Books Online جایی گفته شده که امکان استفاده وجود ندارد و در جایی هم گفته شده وجود دارد .

به مثال زیر دقت کنید

```
CREATE TABLE test(
[ID] BIGINT IDENTITY(1,1) NOT NULL PRIMARY KEY NONCLUSTERED HASH WITH (BUCKET_COUNT=10000),
N1 NVARCHAR(100),
N2 NVARCHAR(100),
N3 NVARCHAR(100)
) WITH (MEMORY_OPTIMIZED=ON,DURABILITY = SCHEMA_AND_DATA)
GO
```

این مثال در SQL Server 2014 RTM Edition قابل اجرا است اما یکسری محدودیت داریم . مثلاً مقدار شروع و گام افزایش باید 1 باشد .
باز هم از مطالب خوب شما متشکرم مقاله مفیدی بود .

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۳/۲۸ ۹:۳۵

ممنون بابت مطلب
البته فکر کنم unique identifier الان پشتیبانی میشه و این مسئله مربوط به زمانی بود که نسخه CTP ارائه شده بود و بعد از ارائه نسخه نهایی این مشکل برطرف شد.
البته یک مسئله ای که من دیدم این هست که توی EF برای پیاده سازی این جداول از طریق کوئری عمل می‌کنن و اینطوری دوباره به سمت کوری نویسی و خارج شدن از شی گرایی میشیم. اگر متدی یا خصوصیتی بود که بتونیم جدول رو oltp معرفی کنیم بسیار خوب میشد.
متاسفانه محدودیت هاش هم خیلی زیاده.