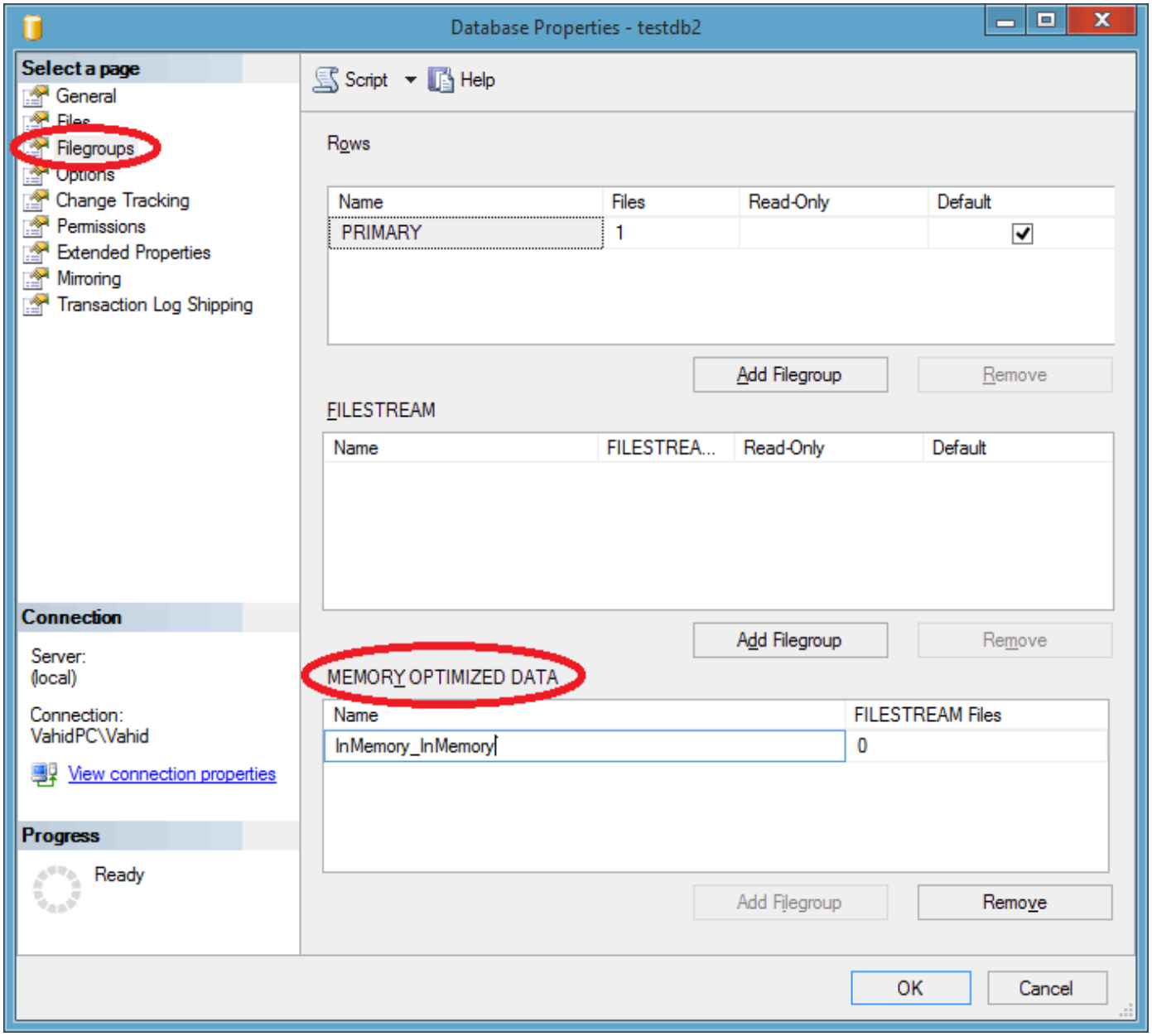
عنوان: **ایجاد جداول بهینه سازی شده برای حافظه در SQL Server 2014**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۱۱ ۲۱:۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, In-Memory OLTP

پس از نگاهی به مفاهیم مقدماتی OLTP درون حافظه‌ای در SQL Server 2014 ، در ادامه به نحوه‌ی انجام تنظیمات خاص جداول بهینه سازی شده برای حافظه خواهیم پرداخت.

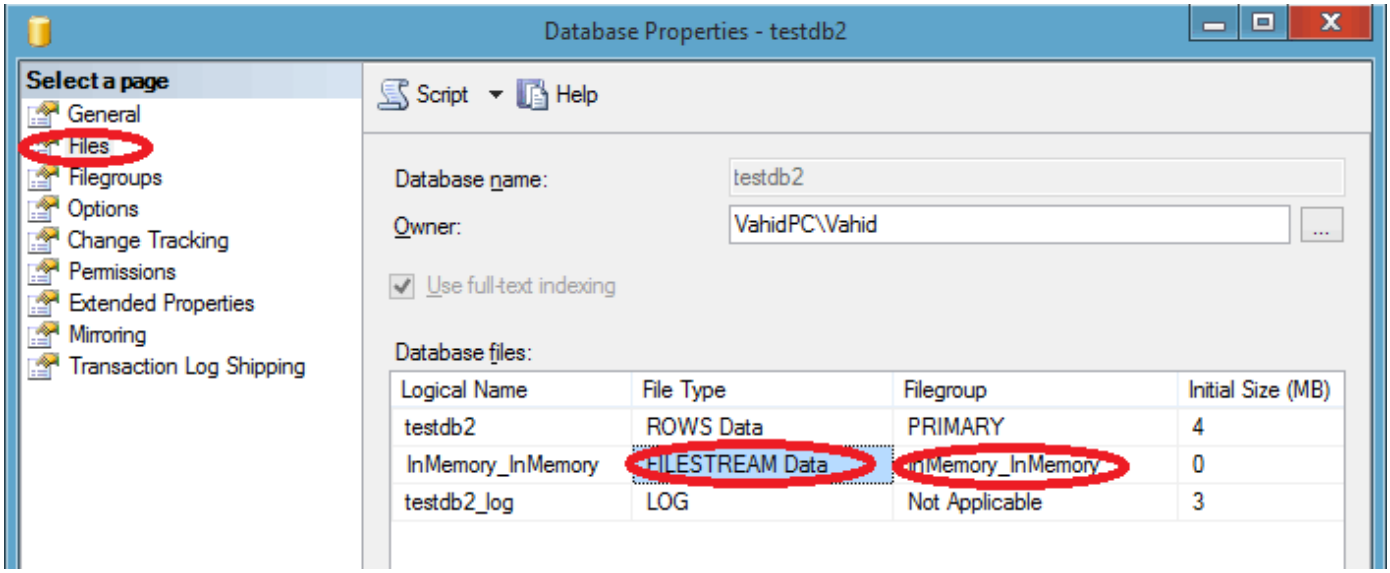
ایجاد یک بانک اطلاعاتی با پشتیبانی از جداول بهینه سازی شده برای حافظه

برای ایجاد جداول بهینه سازی شده برای حافظه، ابتدا نیاز است تا تنظیمات خاصی را به بانک اطلاعاتی آن اعمال کنیم. برای اینکار می‌توان یک بانک اطلاعاتی جدید را به همراه یک filestream filegroup ایجاد کرد که جهت جداول بهینه سازی شده برای حافظه، ضروری است؛ یا اینکه با تغییر یک بانک اطلاعاتی موجود و افزودن filegroup یاد شده نیز می‌توان به این مقصود رسید.
در اینگونه جداول خاص، اطلاعات در حافظه‌ی سیستم ذخیره می‌شوند و برخلاف جداول مبتنی بر دیسک سخت، صفحات اطلاعات وجود نداشته و نیازی نیست تا به کش بافر وارد شوند. برای مقاصد ذخیره سازی نهایی اطلاعات جداول بهینه سازی شده برای حافظه، موتور OLTP درون حافظه‌ای آن، فایل‌های خاصی را به نام checkpoint در یک filestream filegroup ایجاد می‌کند که از آن‌ها جهت ردیابی اطلاعات استفاده خواهد کرد و نحوی ذخیره سازی اطلاعات در آن‌ها از شیوه‌ی با کارآیی بالایی به نام append only mode پیروی می‌کند.
با توجه به متفاوت بودن نحوه‌ی ذخیره سازی نهایی اطلاعات اینگونه جداول و دسترسی به آن‌ها از طریق استریم‌ها، توصیه شده‌است که filestream filegroupهای تهیه شده را در یک SSD یا Solid State Drive قرار دهید.

پس از اینکه بانک اطلاعاتی خود را به روش‌های معمول ایجاد کردید، به برگه‌ی خواص آن در management studio مراجعه کنید. سپس صفحه‌ی file groups را در آن انتخاب کرده و در پایین برگه‌ی آن، در قسمت جدید memory optimized data، بر روی دکمه‌ی Add کلیک کنید. سپس نام دلخواهی را وارد نمائید.



پس از ایجاد یک گروه فایل جدید، به صفحه‌ی `files` خواص بانک اطلاعاتی مراجعه کرده و بر روی دکمه‌ی `Add` کلیک کنید. سپس `File type` این ردیف اضافه شده را از نوع `file stream data` و `file group` آن‌را همان گروه فایلی که پیشتر ایجاد کردیم، تنظیم کنید. در ادامه `logical name` دلخواهی را وارد کرده و در آخر بر روی دکمه‌ی `Ok` کلیک کنید تا تنظیمات مورد نیاز جهت تعاریف جدول بهینه سازی شده برای حافظه به پایان برسد.



این مراحل را توسط دو دستور T-SQL ذیل نیز می‌توان سریعتر انجام داد:

```
USE [master]
GO
ALTER DATABASE [testdb2]
      ADD FILEGROUP [InMemory_InMemory] CONTAINS MEMORY_OPTIMIZED_DATA
GO
ALTER DATABASE [testdb2]
      ADD FILE ( NAME = N'InMemory_InMemory', FILENAME =
N'D:\SQL_Data\MSSQL11.MSSQLSERVER\MSSQL\DATA\InMemory_InMemory' )
      TO FILEGROUP [InMemory_InMemory]
GO
```

ساختار گروه فایل بهینه سازی شده برای حافظه

گروه فایل بهینه سازی شده برای حافظه، دارای چندین دربرگیرنده است که هر کدام چندین فایل را در خود جای خواهند داد:
- Root File که در برگیرنده‌ی متادیتای اطلاعات است.
- Data File که شامل ردیف‌های اطلاعات ثبت شده در جداول بهینه سازی شده‌ی برای حافظه هستند. این ردیف‌ها همواره به انتهای data file اضافه می‌شوند و دسترسی به آن‌ها ترتیبی است. کارآیی IO این روش نسبت به روش دسترسی اتفاقی به مراتب بالاتر است. حداکثر اندازه این فایل 128 مگابایت است و پس از آن یک فایل جدید ساخته می‌شود.
- Delta File شامل ردیف‌هایی است که حذف شده‌اند. به ازای هر ردیف، حداقل اطلاعاتی از آن را در خود ذخیره خواهد کرد؛ شامل ID ردیف حذف شده و شماره تراکنش آن. همانطور که پیشتر نیز ذکر شد، این موتور جدید درون حافظه‌ای، برای یافتن راه چاره‌ای جهت به حداقل رسانی قفل گذاری بر روی اطلاعات، چندین نگارش از ردیف‌ها را به همراه timestamp آن‌ها در خود ذخیره می‌کند. به این ترتیب، هر به روز رسانی به همراه یک حذف و سپس ثبت جدید است. به این ترتیب دیگر بانک اطلاعاتی نیازی نخواهد داشت تا به دنبال رکورد موجود برگردد و سپس اطلاعات آن‌را به روز نماید. این موتور جدید فقط اطلاعات به روز شده را در انتهای رکوردهای موجود با فرمت خود ثبت می‌کند.

ایجاد جداول بهینه سازی شده برای حافظه

پس از آماده سازی بانک اطلاعاتی خود و افزودن گروه فایل استریم جدیدی به آن برای ذخیره سازی اطلاعات جداول بهینه سازی شده برای حافظه، اکنون می‌توانیم اینگونه جداول خاص را در کنار سایر جداول متداول موجود، تعریف و استفاده نمائیم:

```
-- It is not a Memory Optimized
CREATE TABLE tblNormal
(
   [CustomerID] int NOT NULL PRIMARY KEY NONCLUSTERED,
```

```
   [Name] nvarchar(250) NOT NULL,
   CustomerSince DATETIME not NULL
      INDEX [ICustomerSince] NONCLUSTERED
)

--  DURABILITY = SCHEMA_AND_DATA
CREATE TABLE tblMemoryOptimized_Schema_And_Data
(
    [CustomerID] INT NOT NULL
PRIMARY KEY NONCLUSTERED HASH WITH (BUCKET_COUNT = 1000000),
    [Name] NVARCHAR(250) NOT NULL,
    [CustomerSince] DATETIME NOT NULL
INDEX [ICustomerSince] NONCLUSTERED
) WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA)


-- DURABILITY = SCHEMA_ONLY
CREATE TABLE tblMemoryOptimized_Schema_Only
(
    [CustomerID] INT NOT NULL
PRIMARY KEY NONCLUSTERED HASH WITH (BUCKET_COUNT = 1000000),
    [Name] NVARCHAR(250) NOT NULL,
    [CustomerSince] DATETIME NOT NULL
INDEX [ICustomerSince] NONCLUSTERED
) WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_ONLY)
```

در اینجا سه جدول را مشاهده می‌کنید که در بانک اطلاعاتی آماده شده در مرحله‌ی قبل، ایجاد خواهند شد. مورد اول یک جدول معمولی است که از آن برای مقایسه سرعت ثبت اطلاعات با سایر جداول ایجاد شده، استفاده خواهد شد.

همانطور که مشخص است، دو جدول بهینه سازی شده برای حافظه، همان سه ستون جدول معمولی مبتنی بر دیسک سخت را دارا هستند؛ اما با این تفاوت‌ها:

- دارای ویژگی **MEMORY_OPTIMIZED = ON** می‌باشند. به این ترتیب اینگونه جداول نسبت به جداول متداول مبتنی به دیسک سخت متمایز خواهند شد.

- دارای ویژگی **DURABILITY** بوده و توسط مقدار SCHEMA_AND_DATA آن مشخص می‌کنیم که آیا قرار است اطلاعات و ساختار جدول، ذخیره شوند یا تنها قرار است ساختار جدول ذخیره گردد (حالت SCHEMA_ONLY).

- بر روی ستون Id آن‌ها یک hash index ایجاد شده‌است که وجود آن ضروری است و در کل بیش از 8 ایندکس را نمی‌توان تعریف کرد.

برخلاف ایندکس‌های B-tree جداول مبتنی بر سخت دیسک، ایندکس‌های جداول بهینه سازی شده برای حافظه، اطلاعات را تکرار نمی‌کنند. این‌ها صرفا اشاره‌گرهایی هستند به ردیف‌های اصلی اطلاعات. به این معنا که این ایندکس‌ها لاگ نشده و همچنین بر روی سخت دیسک ذخیره نمی‌شوند. کار بازسازی مجدد آن‌ها در اولین بار بازیابی بانک اطلاعاتی و آغاز آن به صورت خودکار انجام می‌شود. به همین جهت مباحثی مانند index fragmentation و نگهداری ایندکس‌ها دیگر در اینجا معنا پیدا نمی‌کنند.

دو نوع ایندکس را در اینجا می‌توان تعریف کرد. اولین آن‌ها hash index است و دومین آن‌ها range index. هش ایندکس‌ها برای حالاتی که در کوئری‌ها از عملگر تساوی استفاده می‌شود بسیار مناسب هستند. برای عملگرهای مقایسه‌ای از ایندکس‌های بازه‌ای استفاده می‌شود.

همچنین باید دقت داشت که پس از ایجاد ایندکس‌ها، دیگر امکان تغییر آن‌ها و یا تغییر ساختار جدول ایجاد شده نیست.

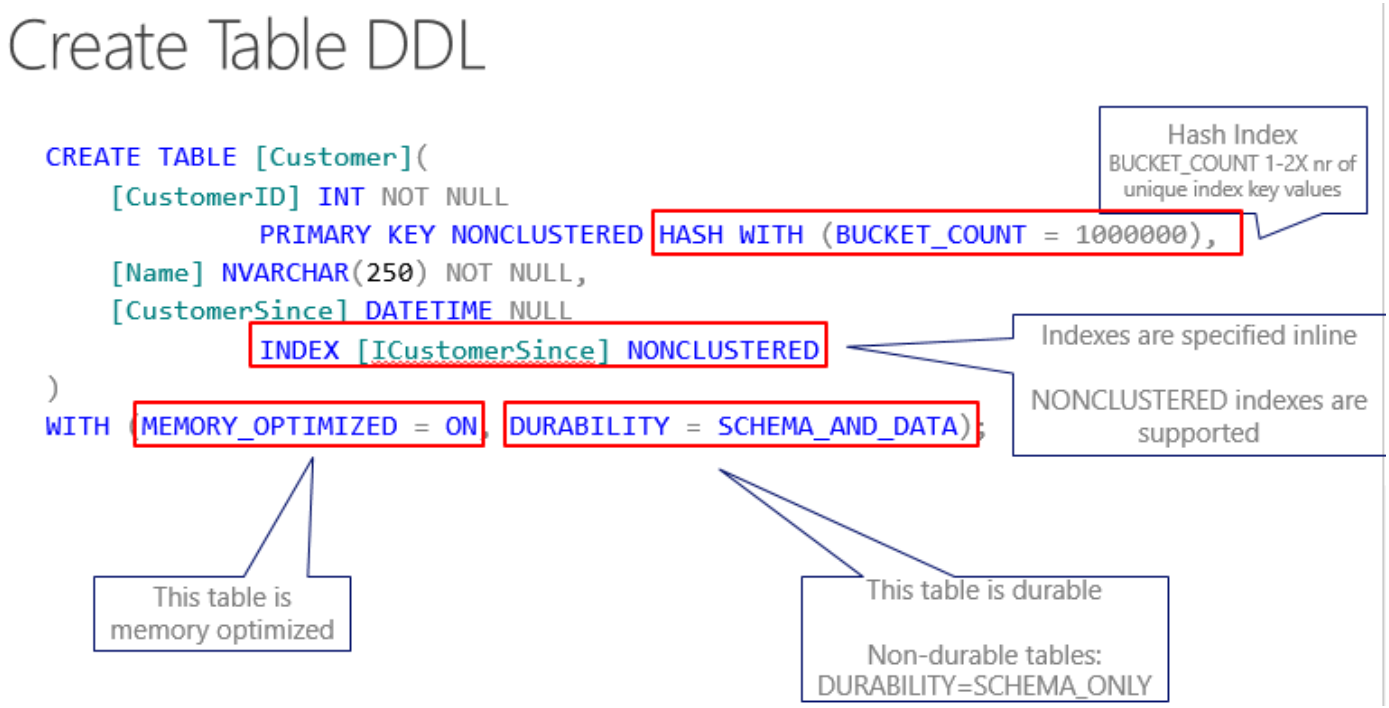
همچنین ایندکس‌های تعریف شده در جداول بهینه سازی شده برای حافظه، تنها بر روی ستون‌هایی غیرنال پذیر از نوع BIN2 collation مانند int و datetime قابل تعریف هستند. برای مثال اگر سعی کنیم بر روی ستون Name ایندکسی را تعریف کنیم، به این خطا خواهیم رسید:

```
Indexes on character columns that do not use a *_BIN2 collation are not supported with indexes on
memory optimized tables.
```

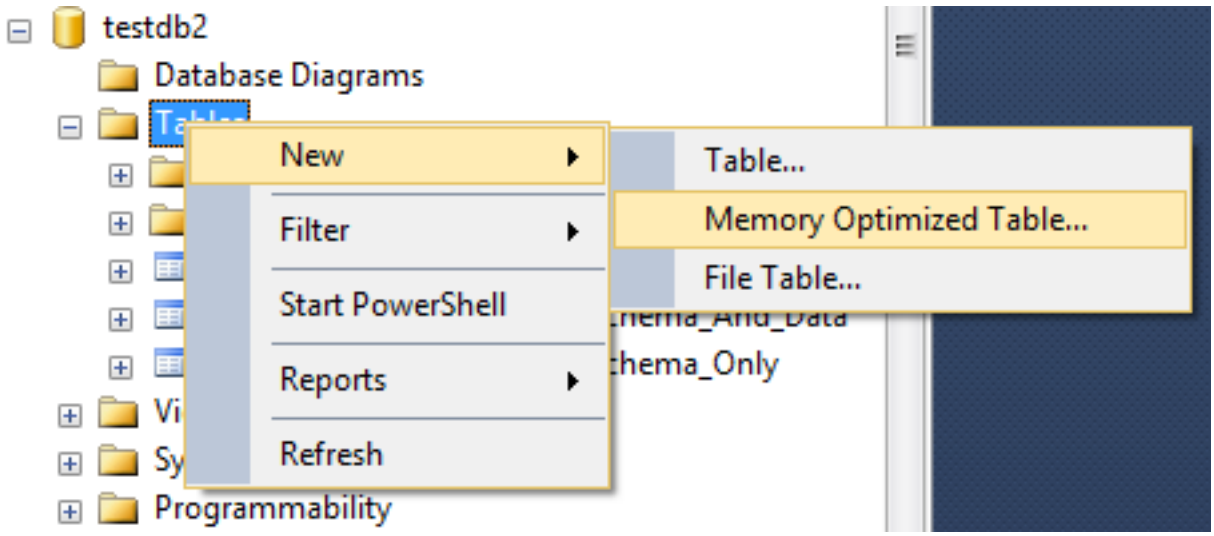
- در حین تعریف هش ایندکس‌ها، مقدار BUCKET_COUNT نیز باید تنظیم شود. هر bucket توسط مقداری که حاصل هش کردن یک ستون است مشخص می‌شود. کلیدهای منحصربفرد دارای هش‌های یکسان در bucketهای یکسانی ذخیره می‌شوند. به همین جهت توصیه شده‌است که حداقل مقدار bucket تعیین شده در اینجا مساوی یا بیشتر از مقدار تعداد کلیدهای منحصربفرد یک جدول باشد؛ مقدار پیش فرض 2 برابر توسط مایکروسافت توصیه شده‌است.

- نوع‌های قابل تعریف ستون‌ها نیز در اینجا به موارد ذیل محدود هستند و جمع طول آن‌ها از 8060 نباید بیشتر شود:

```
bit, tinyint, smallint, int, bigint, money, smallmoney, float, real, datetime, smalldatetime,
datetime2,
date, time, numberic, decimal, char(n),  varchar(n) ,nchar(n),  nvarchar(n), sysname, binary(n),
varbinary(n), and Uniqueidentifier
```



همچنین در management studio، گزینه‌ی جدید memory optimized table -> new نیز اضافه شده‌است و انتخاب آن سبب می‌شود تا قالب T-SQL ایی برای تهیه این نوع جداول، به صورت خودکار تولید گردد.



البته این گزینه تنها برای بانک‌های اطلاعاتی که دارای گروه فایل استریم مخصوص جداول بهینه سازی شده برای حافظه هستند،

فعال می‌باشد.

ثبت اطلاعات در جداول معمولی و بهینه سازی شده برای حافظه و مقایسه کارآیی آن‌ها

در مثال زیر، 100 هزار رکورد را در سه جدولی که پیشتر ایجاد کردیم، ثبت کرده و سپس مدت زمان اجرای هر کدام از مجموعه عملیات را بر حسب میلی ثانیه بررسی می‌کنیم:

```
set statistics time off
SET STATISTICS IO Off
set nocount on
go
-----------------------------

Print 'insert into tblNormal'

DECLARE @start datetime = getdate()
declare @insertCount int = 100000
declare @startId int = 1
declare @customerID int = @startId

while @customerID < @startId + @insertCount
begin
    insert into tblNormal values (@customerID, 'Test', '2013-01-01T00:00:00')
    set @customerID +=1
end

Print DATEDIFF(ms,@start,getdate());
go
-----------------------------

Print 'insert into tblMemoryOptimized_Schema_And_Data'

DECLARE @start datetime = getdate()
declare @insertCount int = 100000
declare @startId int = 1
declare @customerID int = @startId

while @customerID < @startId + @insertCount
begin
    insert into tblMemoryOptimized_Schema_And_Data values (@customerID, 'Test', '2013-01-01T00:00:00')
    set @customerID +=1
end
Print DATEDIFF(ms,@start,getdate());
Go
-----------------------------

Print 'insert into tblMemoryOptimized_Schema_Only'

DECLARE @start datetime = getdate()
declare @insertCount int = 100000
declare @startId int = 1
declare @customerID int = @startId

while @customerID < @startId + @insertCount
begin
    insert into tblMemoryOptimized_Schema_Only values (@customerID, 'Test', '2013-01-01T00:00:00')
    set @customerID +=1
end
Print DATEDIFF(ms,@start,getdate());

Go
```

با این خروجی تقریبی که بر اساس توانمندی‌های سخت افزاری سیستم می‌تواند متفاوت باشد:

```
insert into tblNormal
36423

insert into tblMemoryOptimized_Schema_And_Data
30516

insert into tblMemoryOptimized_Schema_Only
3176
```

و برای حالت select خواهیم داشت:

```
set nocount on
print 'tblNormal'
set statistics time on
select count(CustomerID) from tblNormal
set statistics time off
go
print 'tblMemoryOptimized_Schema_And_Data'
set statistics time on
select count(CustomerID) from tblMemoryOptimized_Schema_And_Data
set statistics time off
go
print 'tblMemoryOptimized_Schema_Only'
set statistics time on
select count(CustomerID) from tblMemoryOptimized_Schema_Only
set statistics time off
go
```

با این خروجی

```
tblNormal
 SQL Server Execution Times:
CPU time = 46 ms,  elapsed time = 52 ms.

tblMemoryOptimized_Schema_And_Data
 SQL Server Execution Times:
CPU time = 32 ms,  elapsed time = 33 ms.

tblMemoryOptimized_Schema_Only
 SQL Server Execution Times:
CPU time = 31 ms,  elapsed time = 30 ms.
```

تاثیر جداول بهینه سازی شده برای حافظه را در 350K inserts بهتر می‌توان با نمونه‌های متداول مبتنی بر دیسک مقایسه کرد.

برای مطالعه بیشتر

Getting started with SQL Server 2014 In-Memory OLTP

Introduction to SQL Server 2014 CTP1 Memory-Optimized Tables

Overcoming storage speed limitations with Memory-Optimized Tables for SQL Server

Memory-optimized Table – Day 1 Test

Memory-Optimized Tables – Insert Test

Memory Optimized Table – Insert Test ...Again

نظرات خوانندگان

نویسنده: علی رضایی
تاریخ: ۱۳۹۳/۰۳/۱۲ ۱۹:۲۲

با سلام و تشکر فراوان جهت این آموزش.
میشه لطفاً بررسی کنید چرا نتیجه نهایی برای من متفاوت شده، طوری که زمان جستجو برای جدول نرمال کمتره!
راستی زمان ورود اطلاعات (100 رکورد) شبیه به زمان‌های مثال شما است و من از یک هارد SSD Corsair استفاده میکنم.

```
set nocount on
print 'tblNormal'
set statistics time on
select count(CustomerID) from tblNormal
set statistics time off
go
print 'tblMemoryOptimized_Schema_And_Data'
set statistics time on
select count(CustomerID) from tblMemoryOptimized_Schema_And_Data
set statistics time off
go
print 'tblMemoryOptimized_Schema_Only'
set statistics time on
select count(CustomerID) from tblMemoryOptimized_Schema_Only
set statistics time off
go
```

100 %

Results Messages

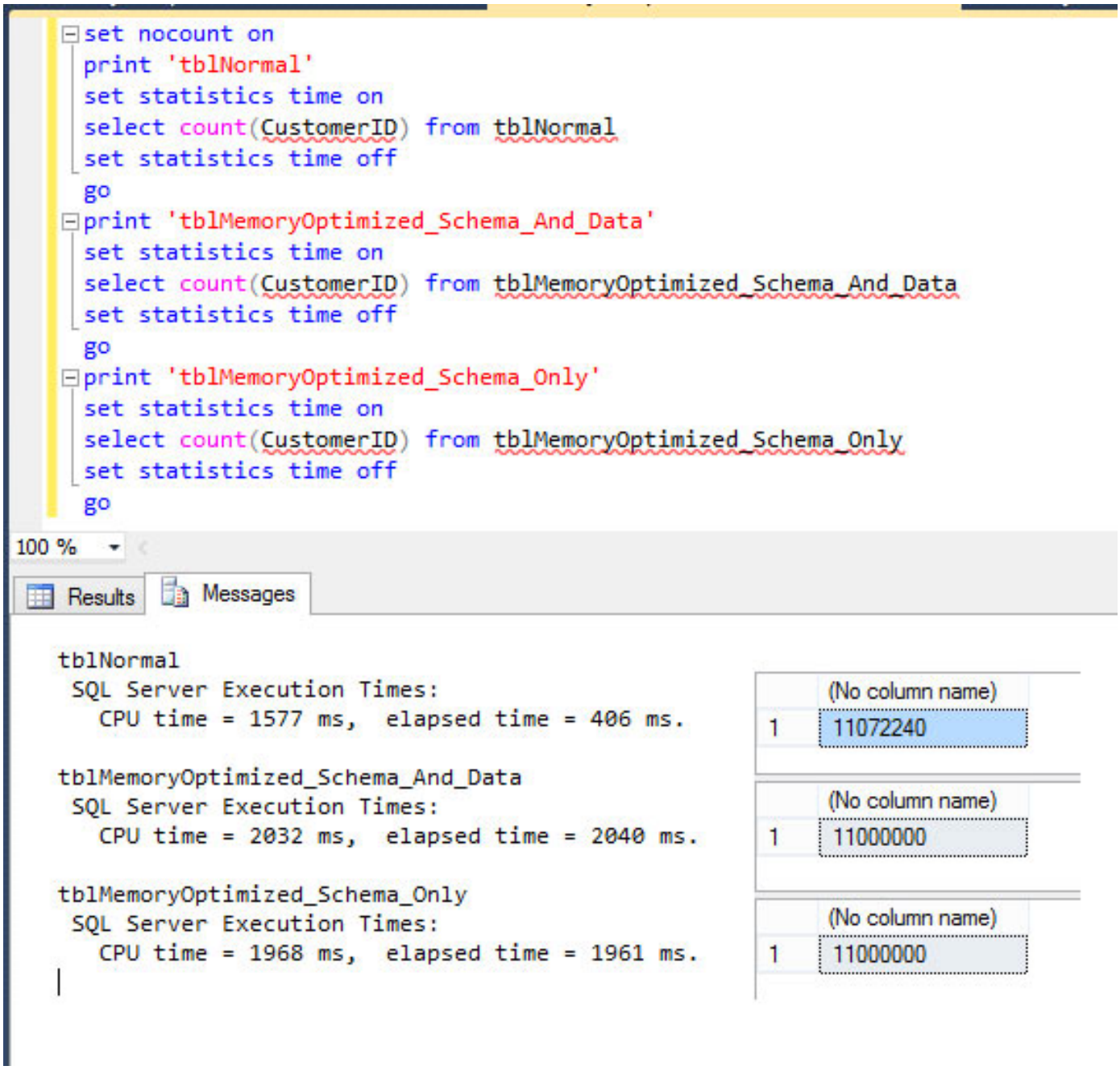
```
tblNormal

 SQL Server Execution Times:
   CPU time = 16 ms,  elapsed time = 6 ms.
tblMemoryOptimized_Schema_And_Data

 SQL Server Execution Times:
   CPU time = 15 ms,  elapsed time = 18 ms.
tblMemoryOptimized_Schema_Only

 SQL Server Execution Times:
   CPU time = 16 ms,  elapsed time = 17 ms.
```

ویرایش:
تست جدید با بیش از 11 میلیون رکورد و خاموش کردن فایروال کومودو و خاموش کردن windows defender



نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۱۳ ۰:۳۷

- بستگی دارد. چقدر سیستم شما RAM دارد. مشخصات CPU آن چیست و خیلی از مسایل جانبی دیگر (مانند تحت نظر بودن پوشه‌های فایل استریم‌ها توسط آنتی ویروس یا خیر).
- هدف اصلی از این تحولات، کارهای همزمان و بررسی تفاوت بهبود در کارهای چند ریسمانی و چند کاربری است. مثال فوق یک مثال تک ریسمانی است.
- یک آزمایش را باید چندبار تکرار کرد و بعد میانگین گرفت. برای تکرار هم نیاز است کش‌های سیستم را هربار حذف کنید:

```
set nocount on
CHECKPOINT
DBCC FREEPROCCACHE()
DBCC DROPCLEANBUFFERS
```

- بعد از پایان Insert تعداد ردیف‌های زیاد در یک جدول درون حافظه‌ای باید اطلاعات آماری آن‌را دستی به روز کرد (^).

```
UPDATE STATISTICS tblNormal WITH FULLSCAN, NORECOMPUTE;
UPDATE STATISTICS tblMemoryOptimized_Schema_And_Data WITH FULLSCAN, NORECOMPUTE;
UPDATE STATISTICS tblMemoryOptimized_Schema_Only WITH FULLSCAN, NORECOMPUTE;
```

این دستورات باید قبل از اجرای سه کوئری آخر قرار گیرند.

- تعداد bucket count هم مهم است. در اینجا فرمولی برای محاسبه آن ارائه شده:

```
Select  POWER(
    2,
    CEILING( LOG( COUNT( 0)) / LOG( 2)))
    AS 'BUCKET_COUNT'
FROM
  (SELECT DISTINCT CustomerId
      FROM tblMemoryOptimized_Schema_And_Data) T
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۱۴ ۰:۵۹

ارزش واقعی جداول درون حافظه‌ای را باید با اعمال تراکنش‌های همزمان و بررسی میزان پاسخگویی سیستم بررسی کرد و نه صرفا با یک آزمایش ساده تک ریسمانی. برای این منظور برنامه‌ای به نام ostress.exe توسط مایکروسافت تهیه شده‌است که امکان انجام یک چنین آزمایشاتی را میسر می‌کند. برای دریافت آن به آدرس‌های ذیل مراجعه کنید:

RML Utilities X64

RML Utilities X86

که نهایتا در این مسیر C:\Program Files\Microsoft Corporation\RMLUtils نصب خواهد شد.

سپس در خط فرمان این سه دستور را امتحان کنید:

```
-- Insert 10000 records using 20 threads, Repeat Execution 3 times

-- disk-based
"C:\Program Files\Microsoft Corporation\RMLUtils\ostress.exe" -n20 -r3 -S. -E -dTestdb2 -q -Q"set
statistics time off; SET STATISTICS IO Off; set nocount on; DECLARE @start datetime = getdate();
declare @insertCount int = 10000; declare @startId int = 1; while @startId < @insertCount begin insert
into tblNormal values ('Test', '2013-01-01T00:00:00') set @startId +=1 end; Print
DATEDIFF(ms,@start,getdate());" -oc:\temp\output

-- memory-optimized, tblMemoryOptimized_Schema_And_Data
"C:\Program Files\Microsoft Corporation\RMLUtils\ostress.exe" -n20 -r3 -S. -E -dTestdb2 -q -Q"set
statistics time off; SET STATISTICS IO Off; set nocount on; DECLARE @start datetime = getdate();
declare @insertCount int = 10000; declare @startId int = 1; while @startId < @insertCount begin insert
into tblMemoryOptimized_Schema_And_Data values ('Test', '2013-01-01T00:00:00') set @startId +=1 end;
Print DATEDIFF(ms,@start,getdate());" -oc:\temp\output

-- memory-optimized, tblMemoryOptimized_Schema_Only
"C:\Program Files\Microsoft Corporation\RMLUtils\ostress.exe" -n20 -r3 -S. -E -dTestdb2 -q -Q"set
statistics time off; SET STATISTICS IO Off; set nocount on; DECLARE @start datetime = getdate();
declare @insertCount int = 10000; declare @startId int = 1;  while @startId <  @insertCount begin
insert into tblMemoryOptimized_Schema_Only values ('Test', '2013-01-01T00:00:00') set @startId +=1 end;
Print DATEDIFF(ms,@start,getdate());" -oc:\temp\output
```

زمانیکه را که در پایان کار نمایش می‌دهد، مبنای واقعی مقایسه است.

البته برای اجرای این دستورات نیاز است فیلد CustomerID را identity تعریف کنید (در هر سه جدول مطلب جاری).

```
-- It is not Memory Optimized
CREATE TABLE tblNormal
(
       [CustomerID] int identity NOT NULL PRIMARY KEY NONCLUSTERED,
       [Name] nvarchar(250) NOT NULL,
   CustomerSince DATETIME not NULL
      INDEX [ICustomerSince] NONCLUSTERED
)

--  DURABILITY = SCHEMA_AND_DATA
CREATE TABLE tblMemoryOptimized_Schema_And_Data
(
    [CustomerID] INT  identity NOT NULL
PRIMARY KEY NONCLUSTERED HASH WITH (BUCKET_COUNT = 131072),
    [Name] NVARCHAR(250) NOT NULL,
    [CustomerSince] DATETIME NOT NULL
INDEX [ICustomerSince] NONCLUSTERED
) WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA)


-- DURABILITY = SCHEMA_ONLY
CREATE TABLE tblMemoryOptimized_Schema_Only
(
    [CustomerID] INT  identity NOT NULL
PRIMARY KEY NONCLUSTERED HASH WITH (BUCKET_COUNT = 131072),
    [Name] NVARCHAR(250) NOT NULL,
    [CustomerSince] DATETIME NOT NULL
INDEX [ICustomerSince] NONCLUSTERED
) WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_ONLY)
```