بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

memory-optimized tables

مدرس : مهندس افشین رفوآ

دوره آموزش MVC

اصلاح جداول بهینه سازی شده بر اساس حافظه (memory-optimized tables)

در SQL Server 2016 Community Technology Preview 2 (چرخه ی ارائه ی نرم افزار) شما می توانید به کمک دستور ALTER TABLE، عملیات ALTER بر روی جدول memory-optimized پیاده کنید. Database application (برنامه ای برای وارد کردن و بازیابی اطلاعات از پایگاه داده ی کامپیوتری) می تواند به کار خود ادامه داده و تا زمانی که فرایند اصلاح پایان نیافته تمامی عملیاتی که سعی دارند به جدول دسترسی داشته باشند، کاملا مسدود می شوند.

در ویرایش قبلی SQL Server، شما می بایست چندین مرحله را به طور دستی کامل می کردید تا جداول memory-optimized بروز رسانی شوند.

پیش از راه اندازی عملیات ALTER TABLE، سربار کاری (workload) باید متوقف گردد. هر تراکنش کاربری که قبل از شروع اجرای دستور ALTER TABLE راه اندازی شده و به جدول دسترسی داشته باشد، ممکن است باعث شود ALTER TABLE با کد خطا 41325 از کار بیافتد.

ساختار نگارشی ALTER TABLE

ساختار نگارشی (syntax) ALTER TABLE به منظور اعمال تغییرات به table schema و نیز افزودن، حذف کردن و باز سازی اندیس ها بکار می ورد. اندیس ها در واقع جزئی از تعریف جدول محسوب می شوند. ساختار نگارشی ALTER TABLE ... ADD/DROP/ALTER INDEX تنها ویژه ی جداول بهینه سازی بر

اساس حافظه طراحی شده و قابل استفاده می باشد. ساختارهای نگارشی CREATE INDEX, DROP INDEX و ALTER INDEX (بدون مشخص کردن دستور ALTER TABLE) برای اندیس های ایجاد شده بر روی جداول memory-optimized قابل استفاده نمی باشد.

نمونه ی زیر ساختار نگارشی سه دستور ADD/DROP/ALTER INDEX را نمایش می دهد.

```
| ADD
  {
    <column_definition>
   | <table_constraint>
   | <table_index>
  } [ ,...n ]
| DROP
  {
    [ CONSTRAINT ]
    {
      constraint_name
    } [ ,...n ]
    | COLUMN
    {
      column_name
    } [ ,...n ]
    | INDEX
    {
      index_name
    } [ ,...n ]
  } [ ,...n ]
| ALTER INDEX index_name
  {
    REBUILD WITH ( <rebuild_index_option> )
  }
}
```

رویه ی زیر مراحل لازم برای بروز رسانی جدول را به صورت خلاصه بیان می کند. در این مثال، طی بروز رسانی یک اندیس نیز می شود. این رویه اسم جدول را حفظ کرده و به دو عملیات رونوشت برداری (copy) نیاز دارد: یکبار داده های کپی شده را در جدول موقتی و بار دیگر در جدول جدید جای گذاری می کند. تغییر bucket_count یک اندیس، افزودن و یا حذف یک ستون نیز به همین شکل صورت می گیرد.

نوع تغییراتی که زیر فهرست شده، کاملا پشتیبانی می شود:

1. تغییر bucket_count

2. افزودن/حذف کردن یک اندیس
3. تغییر دادن، افزودن/حذف کردن یک ستون
4. افزودن/حذف کردن یک محدودیت (constraint)

Schema-bound dependency (وابستگی متصل به schema)

Stored procedure های ترجمه شده به DLL دارای یک نوع وابستگی متصل به schema (schema-bound dependency) به جداول memory-optimized که به آن ها دسترسی دارند، هستند. schema-bound dependency عبارت است از یک رابطه بین دو موجودیت که تا زمانی که موجودیت ارجاع دهنده (referencing entity) همچنان وجود دارد، مانع از حذف و یا تغییر موجودیت ارجاع داده شده (referenced entity) می شود.

جدول زیر تاثیر عملیات ALTER بر روی یک شی schema-bound در زمان کامپایل را تشریح می کند.

عملیات ALTER	در صورتی که شی schema-bound در حال انجام کارهای زیر باشد، پیام خطا باز می گرداند:
ALTER TABLE ... ADD COLUMN	در حال اجرای عملیات درج (بدون قرار دادن ستون در لیست ستون ها) در جدول مورد نظر باشد.
ALTER TABLE ... DROP COLUMN	در حال اجرای عملیات select/update بر روی ستون جدول باشد. و یا در حال اجرای عملیات درج (با لحاظ کردن ستون مورد نظر در لیست ستون ها) در جدول باشد. و یا در حال اجرای عملیات درج در جدول (با گنجاندن ستون در لیست ستون ها) باشد.
ALTER TABLE ... ALTER COLUMN	در حال اجرای عملیات select/update بر روی ستون مورد نظر باشد یا در حال اجرای عملیات insert (بدون قرار دادن ستون مورد نظر در column list) داخل جدول باشد. یا در حال اجرای عملیات درج (با قرار دادن ستون در لیست ستون ها) داخل جدول مورد نظر باشد.

در حال مشخص کردن یک محدودیت کلید اصلی (primary key constraint) در index hint باشد و constraint ای که حذف می شود محدودیت کلید اصلی باشد.	ALTER TABLE ... DROP CONSTRAINT
در حال تعریف اندیس در یک index hint باشد.	ALTER TABLE ... DROP INDEX

مثال

مثال زیر hash index bucket count موجود را تغییر می دهد. این کار باعث می شود hash index جاری با bucket count جدید باز سازی شود. این در حالی است که دیگر خواص (property های) hash index به حالت قبلی خود باقی می مانند.

```
ALTER TABLE Sales.SalesOrderDetail_inmem
    ALTER INDEX imPK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID
        REBUILD WITH (BUCKET_COUNT=67108864)
GO
```

در زیر مثال دیگری را مشاهده می کنید. این مثال یک ستون که محدودیت (constraint) NOT NULL به آن اعمال شده و دارای یک تعریف DEFAULT می باشد و نیز با بهره گیری از WITH VALUES مقادیری را ویژه ی هر سطر موجود در جدول فراهم می نماید، اضافه می کند. اگر WITH VALUES بکار گرفته نشود، هر سطر دارای مقدار NULL در ستون جدید خواهد بود.

```
ALTER TABLE Sales.SalesOrderDetail_inmem
    ADD Comment NVARCHAR(100) NOT NULL DEFAULT N'' WITH VALUES
GO
```

نمونه ی زیر یک محدودیت کلید اصلی (primary key constraint) به ستون موجود اعمال می کند.

```
CREATE TABLE dbo.UserSession (
  SessionId int not null,
  UserId int not null,
  CreatedDate datetime2 not null,
  ShoppingCartId int,
  index ix_UserId nonclustered hash (UserId) with (bucket_count=400000)
)
WITH (MEMORY_OPTIMIZED=ON, DURABILITY=SCHEMA_ONLY)
GO

ALTER TABLE dbo.UserSession
    ADD CONSTRAINT PK_UserSession PRIMARY KEY NONCLUSTERED (SessionId)
GO
```

مثال زیر یک اندیس را حذف می کند.

```
ALTER TABLE Sales.SalesOrderDetail_inmem
    DROP INDEX ix_ModifiedDate
GO
```

نمونه ی زیر یک بر خلاف مثال قبلی یک اندیس اضافه می کند.

```
ALTER TABLE Sales.SalesOrderDetail_inmem
    ADD INDEX ix_ModifiedDate (ModifiedDate)
GO
```

مثال زیر چندین ستون افزوده، سپس یک اندیس به آن اضافه و تعدادی محدودیت به آن اعمال می کند.

```
ALTER TABLE Sales.SalesOrderDetail_inmem
    ADD   CustomerID int NOT NULL DEFAULT -1 WITH VALUES,
        ShipMethodID int NOT NULL DEFAULT -1 WITH VALUES,
        INDEX ix_Customer (CustomerID)
GO
```