عنوان: **بالا بردن سرعت DbContext هنگام ثبت داده های زیاد**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۳/۰۳ ۲۲:۵
آدرس: www.dotnettips.info
برچسب‌ها: Entity framework, EF Code First, DetectChanged

تشریح مسئله : شاید شما هم هنگام ثبت، ویرایش و حتی حذف داده‌های زیاد در Code First متوجه کاهش چشمگیر کارایی پروژه خود شده باشید.(برای مثال ثبت 5000 داده یا بیشتر به صورت هم زمان).برای رفع مشکل بالا چه باید کرد؟

نکته : آشنایی اولیه با مفاهیم EF CodeFirst برای درک بهتر مفاهیم الزامی است.

EntityFramework Code First هنگام کار با Poco Entities برای اینکه مشخص شود که چه داده هایی باید به دیتابیس ارسال شود مکانیزمی به نام Detect Changed معرفی کرده است که وظیفه آن بررسی تفاوت‌های بین مقادیر خواص CurrentValue و OriginalValue هر Entity است که باعث افت چشمگیر سرعت هنگام اجرای عملیات CRUD می‌شود.
هنگامی که از یک Entity کوئری گرفته می‌شود یا از دستور Attach برای یک Entity استفاده می‌کنیم مقادیر مورد نظر در حافظه ذخیره می‌شوند. استفاده از هر کدام دستورات زیر DbContext را مجبور به فراخوانی الگوریتم Automatic Detect Changed می‌کند.

DbSet.Find
DbSet.Local
DbSet.Remove
DbSet.Add
DbSet.Attach
DbContext.SaveChanges
DbContext.GetValidationErrors
DbContext.Entry
DbChangeTracker.Entries

البته Code First امکانی را فراهم کرده است که هنگام پیاده سازی عملیات CRUD اگر تعداد داده‌های شرکت کننده زیاد است برای رفع مشکل کاهش سرعت بهتر است این رفتار را غیر فعال کنیم . به صورت زیر:

```
using (var context = new BookContext())
{
    try
    {
        context.Configuration.AutoDetectChangesEnabled = false;

        foreach (var book in aLotOfBooks)
        {
            context.Books.Add(book);
        }
    }
    finally
    {
        context.Configuration.AutoDetectChangesEnabled = true;
    }
```

در پایان هم وضعیت را به حالت قبل بر می‌گردانیم.
در مورد کاهش مصرف حافظه EF CodeFirst هنگام واکشی داده‌های زیاد هم می‌تونید از این مقاله استفاده کنید.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۰۳ ۲۳:۹

ممنون.
- نکته دیگری که به شدت روی سرعت bulk insert حین کار با ORMها (فرقی نمی‌کنه؛ تمامشون) تاثیر داره، استراتژی انتخاب نوع primary key است. زمانیکه کلید اصلی از نوع auto generated توسط بانک اطلاعاتی باشه، ORM بعد از insert سعی می‌کنه این Id رو به مصرف کننده برگردونه (چون برنامه نقشی در تعیین اون نداره). یعنی عملا با یک insert و بلافاصله با یک select مواجه خواهیم بود. البته این مورد هم باید اضافه بشه که ORMها برای فرآیندها و تراکنش‌هایی کوتاه طراحی شدن و این مساله در حالت متداول کار با ORMها اهمیتی نداره و اصلا به چشم نمیاد.
همین مساله سرعت insert رو «فقط» در حالت فراخوانی با تعداد بالا در یک حلقه برای مثال به شدت پایین میاره و اگر مثلا کلید اصلی توسط خود برنامه مدیریت بشه (مثلا از نوع Guid تولید شده در برنامه باشه)، سرعت bulk insert به شدت بالا میره چون به select بعد از insert نیازی نخواهد بود.
- در حالت bulk insert اگر شخص مطمئن هست که اطلاعات ارسالی توسط او اعتبارسنجی شدن، بهتره تنظیم context.Configuration.ValidateOnSaveEnabled = false رو هم انجام بده. این مورد اعتبارسنجی در حین ذخیره سازی رو غیرفعال می‌کنه.
- همچنین شخصی در اینجا در مورد تعداد بار فراخوانی متد SaveChanges در یک حلقه، تحقیقی رو انجام داده که جالب است.

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۲/۰۳/۰۳ ۲۳:۳۸

ممنون از توضیحات تکمیلی.
تحقیق مورد نظر رو هم بررسی کردم. در نوع خودش جالب بود.

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۲/۰۳/۰۴ ۱۲:۲

سلام ... خیلی ممنون بابت مطلب مفیدتون ...
در چه مواردی نباید از این روش استفاده کرد؟!

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۲/۰۳/۰۴ ۱۲:۴۹

در کل هر زمان که قصد انجام Bulk Insert رو ندارید این رفتار را غیر فعال نکنید.(به صورت پیش فرض فعال است)
البته بهتره که هر زمان در عملیات Bulk Insert تعداد رکوردهای مورد نظر خیلی زیاد بود به ازای یک تعداد مشخص از Entity ها(برای مثال 1000) یک بار DbContext رو SaveChanged کرده و اونو Dispose کنید و دوباره یک Instance جدید از DbContext بسازید و ادامه کار(دلیل دوباره ساختن DbContext هم اینه که DbContext ، بعد از دستور SaveChanged دیتای مورد نظر رو در دیتابیس ذخیره می‌کنهه ولی فقط State هر Entity رو به Unchaged تغییر میده و خود Entity رو Detach نمی‌کنه که این خود باعث افزایش ObjectGraph موجود در DbContext می‌شود و در نتیجه کاهش کارایی).
در ضمن می‌تونید با فراخوانی دستور DetectChanged مستقیما DbContext رو مجبور به بررسی وضعیت خواص CurrentValue و OriginalValue هر Entity بکنید.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۳/۰۵ ۹:۲۶

کتابخانه extended ef هم به نظر در این مورد جالب است.