جدول های موقت و متغیرهای جدول

همانطور که query های خود را نمایش دادید، دریافتید که به دلایل اجرا ممکن است مجبور باشید نتایج متوسط را در یک ساختار موقت ذخیره کنید. SQL Server جدول های موقت سراسری و محلی ،به همراه متغیرهایی برای ذخیره ی نتایج متوسط، ارائه می کند.

توضیحات

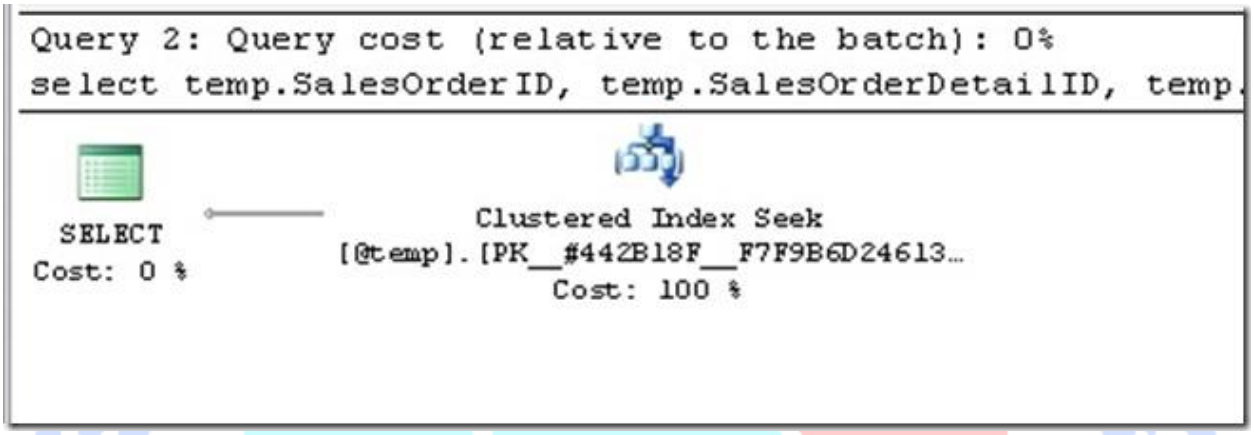
هیچ پاسخ درستی در مقابل این پرسش وجود ندارد که کدامیک از این انواع بهترین نوع برای استفاده می باشد، چرا که بستگی به عملکرد دارد. درک تفاوت های بین آنها قبل از تصمیم گیری در مورد استفاده ی هر کدام، بسیار مهم است. جدول های محلی موقت (به عنوان مثال #temp)، برای اتصال شما مکانی هستند و اگر مورد استفاده قرار نگیرند، پس از مدتی از بین می روند. آنها مانند جدول های منظم کار می کنند. می توانند ایندکس شوند، محدودیت هایی به آنها افزوده شود و استاتیک ها را در خود داشته باشند. جدول های موقت سراسری (برای مثال ##temp) در اتصالات مشابه جدول های محلی کار می کنند و اگر مدتی مورد استفاده قرار نگیرند از بین می روند. جدول های موقت مجددا کامپایل نیز می شوند.

در مقابل متغیرهای جدول با عنوان ایجاد شده های متضاد اعلام می شوند. به هرحال متغیرها به طریقی با جدول های موقت متفاوت می باشند: این متغیرها از طریق CREATE INDEX ایندکس نمی شوند، با استفاده از منطق SELECT/INTO ایجاد نمی شوند، کوتاه نمی شوند و در خود استاتیک ندارند. آنها به ایندکس ها اجازه می دهند تا از طریق محدودیت های PRIMARY KEY و UNIQUE ایجاد شوند که در تعریف متغیر اعلام می شوند و این ها می توانند به وسیله ی query ها استفاده شوند. این حقیقت که جدول در خود استاتیک ندارد، بسیار مهم است زیرا که این امر می تواند به طور بالقوه منجر به query های بهینه ی کمتری در هنگام کار با مجموعه های بزرگ داده شود. این امر در نهایت بر روی تصمیم گیری شما در مورد استفاده از کدامیک تاثیر خواهد داشت. در اینجا هیچ مطلقی وجود ندارد و در انتها آزمایش نشان می دهد که کدام نوع مفیدتر می باشد.

در اینجا مثالی از داده ی انتخاب شده از یک متغیر جدول با ایندکس های اختصاص داده شده از طریق کلید اولیه و محدودیت های خاص را مشاهده می کنید:

```
declare @temp table(ProductID int, SalesOrderID int, SalesOrderDetailID int, OrderQty smallint,
        primary key clustered (SalesOrderID,SalesOrderDetailID),
        unique nonclustered (ProductID,SalesOrderID,SalesOrderDetailID))
insert into @temp (ProductID, SalesOrderID, SalesOrderDetailID, OrderQty)
select ProductID, SalesOrderID, SalesOrderDetailID, OrderQty
from Sales.SalesOrderDetail
select temp.SalesOrderID, temp.SalesOrderDetailID, temp.ProductID, temp.OrderQty
from @temp as temp
where temp.SalesOrderID = 43661
go
```

طرح اجرایی در این مورد یک clustered index ارائه می دهد. همانطور که query ها پیچیده تر می شوند و مقدار داده در پردازش افزایش می یابد، به خاطر نبودن استاتیک ها ممکن است متغیر جدول ضعیف عمل کند ویا ممکن است ضعیف عمل نکند.



اینکه متغیرهای جدول فقط در حافظه کار می کنند، یک داستان متداول بود. جدول های متغیر و متغیرهای جدول در tempdb معرفی شدها اند و SQL Server سعی خواهد کرد که آنها را در حافظه نگاه دارد. به هرحال فشار حافظه روی سرور می تواند باعث شود محتوای هر دو وارد دیسک شوند.