عنوان مقاله: کوئری‌های مهم در SQL Server | بخش اول
نویسنده مقاله: غلامحسین عبادی
تاریخ انتشار: اردیبهشت ماه ۱۴۰۱
منبع: https://nikamooz.com/Important-Query-part۰۱

مقدمه

شما به عنوان یک برنامه نویس و یا متخصص پایگاه داده باید کوئری‌های مهم و کاربردی را در دسترس داشته باشید تا هنگام وقوع مشکلات ناگهانی و وقوع disaster بلافاصله از آنها جهت حل مشکلات استفاده کنید.نکته حائز اهمیت این است که یک DBA حرفه ای میداند که در این مواقع باید خونسردی خود را حفظ کند. در این آموزش سعی شده تا شما را با چند تا از کوئری های مهم آشنا کنیم.

۱ – کوئری اول (بدست آوردن میزان پیشرفت دستورات بکاپ و ریستور) :

از این کوئری شما می توانید جهت مشاهده میزان پیشرفت Backup و یا مشاهده میزان پیشرفت Restore و همینطور می توانید جهت مشاهده میزان پیشرفت دستورات DBCC (مانند DBCC CheckDB و غیره) استفاده نمایید.

متن این کوئری:

```
SELECT
session_id,
  command,
  s.text,
  start_time,
  percent_complete,
  CAST(((DATEDIFF(s,start_time,GetDate()))/۳۶۰۰) as varchar)
    + ' hour(s), ' + CAST((DATEDIFF(s,start_time,GetDate())%۳۶۰۰)/۶۰ as varchar)
    + 'min, ' + CAST((DATEDIFF(s,start_time,GetDate())%۶۰) as varchar)
    + ' sec' as running_time, CAST((estimated_completion_time/۳۶۰۰۰۰۰) as varchar)
    + ' hour(s), ' + CAST((estimated_completion_time %۳۶۰۰۰۰۰)/۶۰۰۰۰ as varchar)
    + 'min, ' + CAST((estimated_completion_time %۶۰۰۰۰)/۱۰۰۰ as varchar)
    + ' sec' as est_time_to_go, dateadd(second,estimated_completion_time/۱۰۰۰, getdate())
```

```
    as est_completion_time
     FROM sys.dm_exec_requests r
     CROSS APPLY sys.dm_exec_sql_text(r.sql_handle) s
WHERE
 r.command in ('RESTORE DATABASE','BACKUP DATABASE')
 OR r.command like 'DBCC%'
```

لازم به ذکر است که ویو سیستمی sys.dm_exec_requests ، در واقع اطلاعاتی را در مورد تمام در خواست های اجرا شده در اسکیوال سرور به ما نشان می دهد.

تذکر : ما می توانیم به کمک ویوهای سیستمی زیر اطلاعاتی را راجع به درخواست هایی که به سمت اسکیوال سرور ارسال شده اند را ببینیم.

۱. ویو سیستمی sys.dm_exec_requests
۲. ویو سیستمی sys.dm_exec_connections
۳. ویو سیستمی sys.dm_exec_sessions
۴. ویو سیستمی sys.sysprocesses
۵. SP سیستمی sp_who
۶. SP سیستمی sp_who۲

دقت کنید که Session_id های بالای ۵۰ مربوط به کوئری های کاربران می باشد.

تذکر : ویوهای سیستمی از نسخه SQL Server ۲۰۰۵ اضافه شدند.

همچنین لازم به ذکر است که در کوئری فوق از فانگشن سیستمی sys.dm_exec_sql_text جهت مشاهده **متن کوئری** در حال اجرا استفاده شده است.

تذکر: چنانچه بخواهید یک DMV و یک DMF را با هم جوین کنید باید از دستور Cross Apply استفاده کنید.

ستون‌های مربوط به این کوئری:

۱. **Session_id:** شماره سشن
۲. **Command**: نوع عملیات جاری (که میتواند Backup و یا Restore و یا یکی از عملیات DBCC باشد).
۳. **s.text**: متن کوئری در حال اجرا را نشان می دهد.
۴. **start_time**: زمان شروع عملیات را نشان می دهد.
۵. **Percent_complete**: درصد پیشرفت کار را نمایش می دهد
۶. **running_time**: مدت زمان سپری شده جهت انجام عملیات مورد نظر
۷. **est_time_to_go**: زمان تقریبی باقیمانده عملیات جاری
۸. **est_completion_time**: زمان تقریبی اتمام عملیات جاری

پس از اجرای این کوئری مطابق شکل زیر شما می توانید اطلاعات ارزشمند زیر را داشته باشید:

```
SELECT
session_id,
  command,
  s.text,
  start_time,
  percent_complete,
  CAST(((DATEDIFF(s,start_time,GetDate()))/3600) as varchar)
    + ' hour(s), ' + CAST((DATEDIFF(s,start_time,GetDate())%3600)/60 as
    + 'min, ' + CAST((DATEDIFF(s,start_time,GetDate())%60) as varchar)
```

Results | Messages

session_id	command	text	start_time	percent_complete	running_time	est_time_to_go	est_completion_time
78	BACKUP DATABASE	BACKUP DATABASE [AdventureWorksDW2019] TO DISK...	2022-05-03 11:28:21.120	61.43127	0 hour(s), 0min, 3 sec	0 hour(s), 0min, 1 sec	2022-05-03 11:28:25.517

۲ – کوئری سوم (بدست آوردن میزان fragmentation کلیه ایندکس های مربوط به یک دیتابیس) :

به کمک این کوئری می توانیم میزان fragmentation (به معنی متلاشی شده و یا از هم گسیخته شدن) کلیه ایندکس های مربوط به کلیه جداول یک دیتابیس را بدست آوریم. کد مربوط به این کوئری به صورت زیر می باشد.

```
SELECT OBJECT_NAME(ips.OBJECT_ID) as TableName
 ,i.NAME as IndexName
 ,ips.index_id
 ,index_type_desc
 ,avg_fragmentation_in_percent
 ,page_count
FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL, 'SAMPLED') ips
INNER JOIN sys.indexes i ON (ips.object_id = i.object_id)
 AND (ips.index_id = i.index_id)
ORDER BY avg_fragmentation_in_percent DESC
```

پس از اجرای کد فوق میزان فرگمنت مربوط به کلیه ایندکس های یک دیتابیس را مشاهده خواهیم کرد. به شکل زیر دقت کنید. در این کوئری از فانگشن سیستمی sys.dm_db_index_physical_stats() و همچنین از ویو سیستمی sys.index استفاده شده است.

```
SELECT OBJECT_NAME(ips.OBJECT_ID) as TableName
 ,i.NAME as IndexName
 ,ips.index_id
 ,index_type_desc
 ,avg_fragmentation_in_percent
 ,page_count
FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, N
INNER JOIN sys.indexes i ON (ips.object_id = i.object_id)
 AND (ips.index_id = i.index_id)
```

208 %

Results | Messages

	TableName	IndexName	index_id	index_type_desc	avg_fragmentation_in_percent	page_count
25	ProductCostHistory	PK_ProductCostHistory_ProductID_StartDate	1	CLUSTERED INDEX	33.3333333333333	3
26	Product	AK_Product_Name	3	NONCLUSTERED INDEX	25	4
27	DatabaseLog	PK_DatabaseLog_DatabaseLogID	2	NONCLUSTERED INDEX	25	4
28	ProductModelProductDescri...	PK_ProductModelProductDescriptionCulture_ProductMo...	1	CLUSTERED INDEX	25	4
29	BusinessEntityContact	AK_BusinessEntityContact_rowguid	2	NONCLUSTERED INDEX	25	4
30	ProductVendor	PK_ProductVendor_ProductID_BusinessEntityID	1	CLUSTERED INDEX	20	5
31	BusinessEntityContact	PK_BusinessEntityContact_BusinessEntityID_PersonID_...	1	CLUSTERED INDEX	16.6666666666667	6
32	ProductInventory	PK_ProductInventory_ProductID_LocationID	1	CLUSTERED INDEX	14.2857142857143	7

تذکر: به کمک فانگشن سیستمی sys.dm_db_index_physical_stats می توانیم میزان فرگمنت ایندکس ها را بدست آوریم.

ستون های مربوط به کوئری فوق به صورت زیر می باشد :

TableName: نام جدول

IndexName: نام ایندکس موجود در جدول

Index_id: ID مربوط به ایندکس

Index_type_desc: نوع ایندکس

Avg_fragmentation_in_percent: درصد فرگمنت یک ایندکس

Page_count: تعداد صفحات یک ایندکس

تذکر: دقت کنید دوستان روی دیتابیس های عملیاتی واقعی هیچگاه از این کوئری استفاده نکنید چون باعث کندی دیتابیس شما خواهد شد.

رفع مشکل مربوط به fragmentation به صورت موقت:

مشکل Fragmentation را می توان توسط دو دستور Rebuild و یا Reorganize کردن ایندکس ها برطرف نمود. این کار را بعضی از افراد به کمک Maintenance Plan انجام می دهند و بعضی دیگر به کمک اسکریپت هایی که آماده کرده اند (این اسکریپت ها را معمولا به صورت Job تنظیم می نمایند)، انجام می دهند. چنانچه میزان fragmentation ایندکس شما بین ۱۰ تا ۳۰ درصد بود از دستور Reorganize جهت برطرف کردن fragmentation ایندکس ها استفاده کنید و چنانچه میزان fragmentation ایندکس ها شما بالاتر از ۳۰ درصد بود در این صورت از دستور Rebuild جهت بر طرف کردن fragmentation ایندکس ها استفاده کنید. ولی یادتان باشد این فرمول همیشه درست نیست و بهتر است همیشه خودتان ایندکس ها را با توجه بهQuery هایتان بررسی کنید و به عنوان یک DBA بهترین تصمیم را بگیرید.

همچنین شما می توانید با تنظیمات آپشن مربوط به Fill Factor میزان Fragmentation را تقریبا بر طرف نمایید. منظور از Fill Factor میزان فضای رزروی که می توانیم برای Page هایی که در قسمت Leaf Level ایندکس ها قرار دارند، در نظر گرفت. به مثال زیر دقت کنید.

```
Alter Index IX_tblKala_Des ON tblKala Rebuild with (online=on, FillFactor=۹۰)
```

همچنین لازم به ذکر است که عدد مربوط به fillfactor را باید با سعی و خطا به دست آورد. به طور مثال یک بار با عدد ۹۰ تست می کنیم ولی اگر ببینیم که همچنان فرگمنت داریم عدد را به ۸۵ کاهش می دهیم و اگر باز هم ببینیم که فرگمنت داریم این عدد را به ۸۰ کاهش می دهیم. دقت کنید که با تنظیم این آپشن میزان IO شما افزایش خواهد یافت که باعث کاهش Performance خواهد شد. بهتر است مشکل را اساسی حل کنید مثلا به Data Type ها مراجعه کرده و آنها را درست کنید تا برای همیشه از شر مشکل مربوط به Fragmentation خلاص شوید.

تذکر: دستورات DML (Insert , Delete , Update)، خصوصا دستور Update می تواند باعث ایجاد مشکل مربوط به fragmentation شود.

تذکر: Fragmentation می تواند باعث افزایش IO و همچنین باعث افزایش ترافیک شبکه و نهایتا باعث کاهش Performance و کاهش سرعت کوئری های شما شود.

تذکر مهم: هیچگاه روی فیلدهای incremental و همچنین روی فیلدهای کلید اصلی آپشن Fillfactor را تنظیم نکنید.

تذکر مهم: در سیستم های عملیات و Production دقت کنید که حتما و حتما آپشن online=on را هنگام Rebuild ایندکس ها استفاده نمایید.

انواع Fragmentation

۱ – Internal Fragmentation: در اثر وجود فضای غیرقابل استفاده در یک Page ایجاد می شود. این نوع فرگمنت می تواند به طور مثال از اثر Delete کردن رکوردها ایجاد شود.

۲ – External Fragmentation: در این نوع فرگمنت ترتیب منطقی صفحات با ترتیب فیزیکی صفحات یکی نیستند.

۳ – File Fragmentation: این نوع فرگمنت زمانی رخ می دهد که قسمت های مختلف یک فایل (دیتا فایل) در اثر وجود فایل های دیگر(مثلا فایلها اکسل و یا ورد و یا فایل های بکاپ و یا فایل های دیگر) سر راه خود ، در قسمت های مختلف دیسک ذخیره شوند.

مشکل مربوط به File Fragmentation را می توانیم به کمک Backup و سپس Restore حل نماییم. همچنین اگر می خواهید مشکل File Fragmentation را حل نمایید کافیست اندازه اولیه دیتا فایل دیتابیس خود (initial size) را به اندازه کافی بزرگ بگیرید. مثلا می توانید اندازه اولیه دیتا فایل خود را به اندازه آرشیو سالیانه ، در همان ابتدا در نظر بگیرید.

۳ – کوئری سوم: بدست آوردن تعداد کانکشن های مربوط به هر لاگین روی هر دیتابیس

شما به کمک کوئری زیر می توانید تعداد کانکشن های مربوط به لاگین ، روی هر دیتابیس را مشاهده نمایید.

```
SELECT
DB_NAME(dbid) AS DBName,
COUNT(dbid) AS NumberOfConnections,
loginame
FROM    sys.sysprocesses
GROUP BY dbid, loginame
ORDER BY DB_NAME(dbid)
```