**بسمه تعالی**



برنامه نویسی بانک اطلاعاتی SQL Server

**نویسنده : محمد حسین فخرآوری**

تا اینجا با روش‌های مختلف جمع آوری اطلاعات آماری مرتبط با کوئری‌های اجرا شده‌ی در SQL Server آشنا شدیم. در این قسمت قصد داریم بررسی کنیم این اطلاعات جمع آوری شده، چه مفاهیمی را در بر دارند و مهم‌ترین‌های آن‌ها کدامند؟

**شاخص‌های مهم بررسی کارآیی کوئری‌ها**

**در ابتدای بررسی هر کوئری، باید 4 شاخص بسیار مهم، مدنظر باشند:**

\* مدت زمان اجرای کوئری: هرچند بررسی مدت زمان اجرای کوئری، شاخص مهمی‌است، اما الزاما حاوی اطلاعات مفیدی در مورد آن کوئری نیست. برای مثال اگر یک کوئری زیاد طول می‌کشد، حتما به معنای وجود مشکلی با آن نیست؛ ممکن است اطلاعات زیادی را واکشی می‌کند یا ممکن است توسط عاملی سد شده‌است. در این موارد هرچند مشکلاتی وجود دارند، اما مستقیما مرتبط با آن کوئری نیستند.

\* میزان مصرف CPU: میزان کاری که باید توسط CPU انجام شود تا کوئری به نتیجه برسد.

\* I/O : در SQL Server می‌توان هم physical I/O و هم logical I/O را بررسی کرد. برای مثال اگر اطلاعات مورد درخواست توسط کوئری هم اکنون در حافظه موجود باشند، نیازی به physical I/O پرهزینه نخواهد بود و در مقابل آن logical I/O کم هزینه‌تر است.

\* میزان مصرف حافظه

در کل هر کدام از این شاخص‌ها اگر دارای مقدار بالایی باشند، بیانگر وجود مشکلی است.

**مروری بر ابزارهای مختلف اندازه‌گیری شاخص‌های کارآیی**

**Management studio**

درون Management studio می‌توان اطلاعات مرتبط با یک کوئری را به صورت زنده مشاهده کرد. البته این اطلاعات صرفا مرتبط با یک کوئری و یا تعدادی مشخص هستند؛ چون باید کوئری را به صورت دستی درون این برنامه اجرا کرد و سپس اطلاعات اجرای کوئری‌ها را دریافت نمود. اطلاعات آماری که توسط آن نیز ارائه می‌شود محدودیت‌هایی دارد. برای مثال مدت زمان اجرای کوئری و یا تعداد رکوردهای تحت تاثیر قرار گرفته شده را می‌توان مشاهده کرد. اما به اندازه‌ی اطلاعات ارائه شده‌ی در یک execution plan کامل نیست. به علاوه بازگشت اطلاعات حاصل از اجرای کوئری‌ها درون این برنامه، سربار خودش را داشته و سبب کند شدن برنامه می‌شود. در آخر اطلاعات ارائه شده‌ی توسط آن‌را نیز باید از قسمت‌های مختلفی جمع آوری و به صورت دستی ذخیره کرد.

**Extended Events**

توسط Extended Events نیز می‌توان همانند Management studio، اطلاعات آماری یک تک کوئری و یا یک batch را جمع آوری کرد؛ اما پس از ایجاد و تنظیم آن، به صورت خودکار اجرا می‌شود. در حین تعریف یک سشن Extended Events می‌توان شاخص‌های خاصی را انتخاب کرد و یا شرط‌های دقیقی را اعمال کرد. خروجی آن نیز به صورت خودکار در یک فایل ذخیره می‌شود.

**Dynamic management objects**

با استفاده از DMO's از نتایج آماری مرتبط با تک کوئری‌ها، به نتایج تجمعی حاصل از اجرای آن‌ها می‌رسیم. این نتایج نیز در plan cache ذخیره می‌شوند. به این معنا که اگر کش، تخلیه با اجرای دستور DBCC FREEPROCCACHEو یا سرور ری‌استارت شود، این اطلاعات از دست خواهند رفت. هدف آن بیشتر رفع اشکال کوئری‌هایی است که هم اکنون در حال اجرا هستند. اگر نیاز به اطلاعات دوره‌ای را داشته باشید، نیاز خواهید داشت تا با تهیه‌ی snapshotهایی از بانک اطلاعاتی، این تاریخچه را تکمیل کنید. به همین جهت Query Store ارائه شده‌است تا نیازی به اینکار نباشد.

**Query Store**

Query Store کار ذخیره سازی متن plan و آمار تجمعی مرتبط با آن‌را به صورت خودکار انجام می‌دهد و آن‌را درون بانک اطلاعاتی کاربر ذخیره می‌کند. به همین جهت با خالی شدن کش، برخلاف DMO's، اطلاعات آن حذف نمی‌شود.

**مثالی از روش‌های مختلف جمع آوری اطلاعات آماری حاصل از اجرای کوئری‌ها در SQL Server**

در ادامه قصد داریم با مثالی، خلاصه‌ای را از سه قسمتی که تاکنون بررسی کردیم، ارائه دهیم. برای این منظور ابتدا رویه‌ی ذخیره شده‌ی زیر را ایجاد می‌کنیم:

|  |
| --- |
| DROP PROCEDURE IF EXISTS [Application].[usp\_GetPersonInfo];  GO  CREATE PROCEDURE [Application].[usp\_GetPersonInfo]  (@PersonID INT)  AS  SELECT  [p].[FullName],  [p].[EmailAddress],  [c].[FormalName]  FROM [Application].[People] [p]  LEFT OUTER JOIN [Application].[Countries] [c]  ON [p].[PersonID] = [c].[LastEditedBy]  WHERE [p].[PersonID] = @PersonID;  GO |

کار آن دریافت اطلاعات یک کاربر بر اساس ID او می‌باشد.

سپس یک سشن Extended event را با نام QueryPerf ایجاد می‌کنیم:

|  |
| --- |
| IF EXISTS (SELECT \* FROM sys.server\_event\_sessions WHERE [name] = 'QueryPerf')  BEGIN  DROP EVENT SESSION [QueryPerf] ON SERVER;  END  GO  CREATE EVENT SESSION [QueryPerf]  ON SERVER  ADD EVENT sqlserver.sp\_statement\_completed(  WHERE ([duration]>(1000))),  ADD EVENT sqlserver.sql\_statement\_completed(  WHERE ([duration]>(1000))),  ADD EVENT sqlserver.query\_post\_execution\_showplan  ADD TARGET package0.event\_file(  SET filename=N'C:\Temp\QueryPerf\test.xel',max\_file\_size=(256))  WITH (  MAX\_MEMORY=16384 KB,EVENT\_RETENTION\_MODE=ALLOW\_SINGLE\_EVENT\_LOSS,  MAX\_DISPATCH\_LATENCY=5 SECONDS,MAX\_EVENT\_SIZE=0 KB,  MEMORY\_PARTITION\_MODE=NONE,TRACK\_CAUSALITY=ON,STARTUP\_STATE=OFF);  GO |

این سشن به رخ‌دادهای sql\_statement\_completed، sp\_statement\_completed و query\_post\_execution\_showplan، اگر طول مدت آن کوئری بیش از 1 میلی ثانیه باشد، واکنش نشان می‌دهد. نتیجه‌ی نهایی را نیز در پوشه‌ی C:\Temp\QueryPerf ذخیره می‌کند (این پوشه را باید به صورت دستی ایجاد کنید).

در ادامه Query Store را نیز بر روی بانک اطلاعاتی فعال کرده و همچنین اگر اطلاعاتی از پیش در آن وجود دارند، پاک می‌شود.

سپس هر آنچه را که در plan cache نیز وجود دارد، حذف می‌کنیم:

|  |
| --- |
| DBCC FREEPROCCACHE;  GO |

اکنون سشن QueryPerf را که پیشتر ایجاد کردیم، آغاز می‌کنیم:

|  |
| --- |
| ALTER EVENT SESSION [QueryPerf]  ON SERVER  STATE = START;  GO |

نتیجه‌ی آن‌را در قسمت management->extended events، با سبز شدن آیکن QueryPerf می‌توانید مشاهده کنید.

در ادامه چون می‌خواهیم نتایج آماری را در management studio نیز مشاهده کنیم، ابتدا جمع آوری شاخص‌های آماری را در یک پنجره‌ی جدید new query، فعال می‌کنیم:

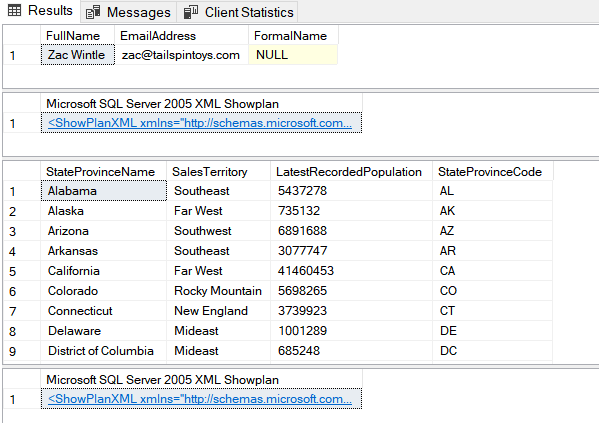
|  |
| --- |
| SET STATISTICS IO ON;  GO  SET STATISTICS TIME ON;  GO  SET STATISTICS XML ON;  GO |

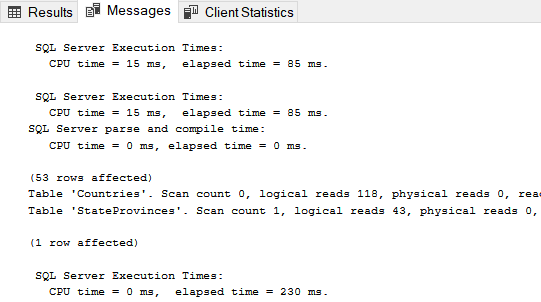
همچنین در منوی Query، گزینه‌ی Include client statistics را نیز انتخاب می‌کنیم تا مشخص شود که آیا عملیات insert/update/delete انجام شده‌است. چه تعداد ردیف تحت تاثیر اجرای این کوئری قرار گرفته‌اند. چه تعداد تراکنش انجام شده‌است. همچنین اطلاعات آماری شبکه و زمان نیز ارائه شوند.

پس از این تنظیمات، اکنون نوبت به اجرای کوئری‌های زیر رسیده‌است که یکی پارامتری است و دیگری AdHoc:

|  |
| --- |
| EXECUTE [Application].[usp\_GetPersonInfo] 1234;  GO  SELECT  [s].[StateProvinceName],  [s].[SalesTerritory],  [s].[LatestRecordedPopulation],  [s].[StateProvinceCode]  FROM [Application].[Countries] [c]  JOIN [Application].[StateProvinces] [s]  ON [s].[CountryID] = [c].[CountryID]  WHERE [c].[CountryName] = 'United States';  GO |

با اجرای آن، در management studio، برگه‌های messages و client statistics ظاهر می‌شوند که هر کدام اینبار اطلاعات آماری اجرای این کوئری را به همراه دارند. همچنین در قسمت results، امکان مشاهده‌ی query plan، به علت فعال بودن اطلاعات آماری XML، وجود دارد.

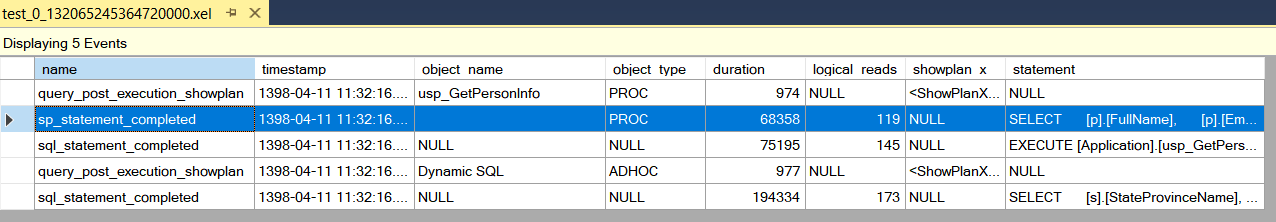




سپس سشن QueryPerf را متوقف و حذف می‌کنیم:

|  |
| --- |
| ALTER EVENT SESSION [QueryPerf]  ON SERVER  STATE = STOP;  GO  DROP EVENT SESSION [QueryPerf] ON SERVER;  GO |

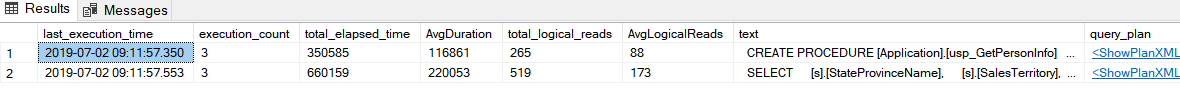
فایل خروجی با پسوند xel آن را که در پوشه‌ی C:\Temp\QueryPerf ذخیره شده‌است، می‌توان در management studio مشاهده کرد. البته در ابتدای نمایش آن، صرفا دو ستون name و timestamp را نمایش می‌دهد که می‌توان با انتخاب هر ردیف آن و سپس انتخاب و کلیک راست بر روی ردیف‌های details آن، گزینه‌ی Show Column in table را انتخاب کرد تا شاخص مدنظر، در ستون‌های گزارش نیز ظاهر شود.



اگر بخواهیم از عملیات صورت گرفته توسط DMO's کوئری بگیریم:

|  |
| --- |
| SELECT  [qs].[last\_execution\_time],  [qs].[execution\_count],  [qs].[total\_elapsed\_time],  [qs].[total\_elapsed\_time]/[qs].[execution\_count] [AvgDuration],  [qs].[total\_logical\_reads],  [qs].[total\_logical\_reads]/[qs].[execution\_count] [AvgLogicalReads],  [t].[text],  [p].[query\_plan]  FROM sys.dm\_exec\_query\_stats [qs]  CROSS APPLY sys.dm\_exec\_sql\_text([qs].sql\_handle) [t]  CROSS APPLY sys.dm\_exec\_query\_plan([qs].[plan\_handle]) [p]  WHERE [t].[text] LIKE '%Countries%';  GO |

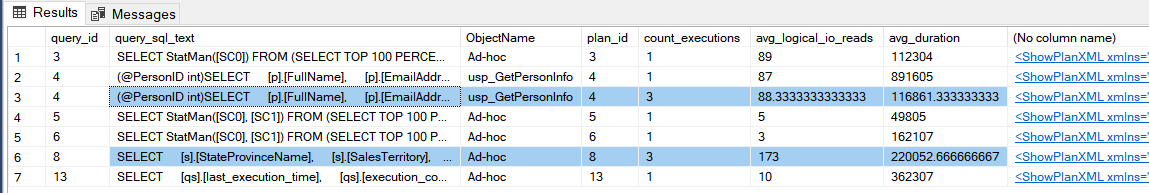
که در آن تنها ردیف‌هایی که متن کوئری آن‌ها حاوی Countries است، فیلتر شده، به یک چنین خروجی خواهیم رسید:



همانطور که مشاهده می‌کنید، شاخص‌های چهارگانه‌ای که در ابتدای بحث معرفی شدند، در مورد کوئری پارامتری نوشته شده، وضعیت بسیار بهتری نسبت به کوئری AdHoc دوم دارند.

از Query Store هم می‌توان به صورت زیر کوئری گرفت (علاوه بر قسمت رابط کاربری Query Store که ذیل اشیاء مرتبط با بانک اطلاعاتی WideWorldImporters در management studio قابل مشاهده‌است.

|  |
| --- |
| SELECT [qsq].[query\_id], [qst].[query\_sql\_text],  CASE WHEN [qsq].[object\_id]=0 THEN N'Ad-hoc' ELSE OBJECT\_NAME([qsq].[object\_id])END AS [ObjectName],  [qsp].[plan\_id], [rs].[count\_executions], [rs].[avg\_logical\_io\_reads], [rs].[avg\_duration],  TRY\_CONVERT(XML, [qsp].[query\_plan]), [rs].[last\_execution\_time], (DATEADD(MINUTE, -(DATEDIFF(MINUTE, GETDATE(), GETUTCDATE())),  [rs].[last\_execution\_time])) AS [LocalLastExecutionTime]  FROM [sys].[query\_store\_query] [qsq]  JOIN [sys].[query\_store\_query\_text] [qst] ON [qsq].[query\_text\_id]=[qst].[query\_text\_id]  JOIN [sys].[query\_store\_plan] [qsp] ON [qsq].[query\_id]=[qsp].[query\_id]  JOIN [sys].[query\_store\_runtime\_stats] [rs] ON [qsp].[plan\_id]=[rs].[plan\_id]  WHERE [qst].[query\_sql\_text] LIKE '%Countries%';  GO |

s