عنوان مقاله: آشنایی با Query store | بخش اول
نویسنده مقاله: تیم فنی نیک آموز
تاریخ انتشار: دی ماه ۱۴۰۰
منبع: https://nikamooz.com/query-store-part۰۱

مقدمه

ویژگی Query store یکی از قابلیت های بی نظیری هست که از نسخه SQL SERVER ۲۰۱۶ شاهد ان هستیم . با استفاده از این ویژگی، می‌توانید الگوی‌هایی را کشف کنید که باعث اختلال در عملکرد سیستم می‌شود. اما قبل از این که به صورت جزئی تر وارد این مبحث شویم نیاز هست که برخی از تعاریف را با دقت بررسی کنیم که ایا قابلیت‌هایی فعلی که در نسخه‌های قبل نیز وجود داشته است، چه نقاط قوت و ضعفی نسبت به این ویژگی دارند و چرا مایکروسافت تاکید دارد که استفاده از این قابلیت در برطرف مشکلات سیستم نسبت به سایر روش‌ها از اولویت بالاتری برخوردار هست.

کلیه مطالب و عنوانی که در این مقاله مطالعه خواهید کرد بر اساس کتاب Query Store for SQL Server ۲۰۱۹ بوده و قصد داریم سرفصل های این کتاب را بررسی و تحلیل کرده و از موارد ارائه شده در این کتاب استفاده کنیم . این سرفصل ها در ۱۱ عنوان و مبحث اموزشی طبقه بندی شده که توضیحات هر فصل به شرح زیر است :

- در فصل اول به ساختار Query store و مقایسه روش‌های کلاسیک Tuning پرداخته شده و مزایا و معایب هر بخش توضیح داده شده است.
- در فصل دوم معماری داخلی Query store تشریح شده که متشکل از چه بخش‌هایی است و هر المان دقیقا به چه شکل عمل می‌کند.
- در فصل سوم کتاب، پارامترهایی که برای تنظیمات این قابلیت در نظر گرفته شده تشریح شده است. هر پارامتر بسته به سناریو و بیزینسی که کار می‌کنید ممکن است تغییرات در نحوه کانفیگ ان اعمال شود. لذا سرفصل مجزایی برای این فصل در نظر گرفته شده که Best practiceها را در این زمینه معرفی خواهیم کرد.
- در فصل چهارم کتاب، گزارشات و داشبوردهایی معرفی شده است که پس از فعال‌سازی و تجمیع داده‌ها می‌می توانیم از تحلیل آن‌ها برای بررسی گلوگاه‌های سیستم استفاده کنیم. این گزارشات در طبقه‌بندی‌های مختلف و از جنبه‌های مختلف دیتاهای جمع اوری شده را تحلیل و ارزیابی می‌کند.
- در فصل پنجم کتاب مبحثی تحت عنوان Query Store Catalog Views معرفی شده است. این فصل در واقع DMV هایی را معرفی می‌کند که بخشی از ان مرتبط با تنظیمات اولیه Query store و بخش‌های دیگر مرتبط با گزارشات پیشرفته‌تری هست که در نسخه‌های بعدی Sql server افزوده شده است.

- در فصل ششم کتاب بحثی به عنوان Query Store Use Cases مطرح شده است. این سرفصل بر اساس سناریوهای دنیای واقعی طراحی شده است که واقعا کاربرد Query store را نمایش دهد. به عنوان مثال بحث ارتقا ورژن در محیط‌های عملیاتی که یکی از پر چالش‌ترین کارهای یک DBA هست در این سناریو به صورت مفصل توضیح داده شده است.
- در فصل هفتم بر روی تحلیل‌ها، متریک‌ها و ابزارهای مختلفی که می‌توانیم از داده‌های ان در گزارشات مختلف استفاده کنیم. می‌توانیم فلگ‌های مرتبط را برای اعمال تغییرات بر روی Queryهای مختلف اجرا کنیم.
- در فصل هشتم با یکی از قابلیت‌های بی‌نظیری که در نسخه ۲۰۱۹ به Query store اضافه شده است این موضوع را بررسی و تحلیل می‌کنیم. این قابلیت فوق العاده تحت عنوان Automatic Plan Regession Correction (APRC) شناخته می‌شود. در واقع پلن‌ها را به صورت خودکار بهبود داده و تغییراتی بر روی انها اجرا می‌کند تا کندی‌های مرتبط با Query ها و سربارهای اضافی به شدت کاهش پیدا کند
- در فصل نهم کتاب به موضوع wait statisticsها و اطلاعات ارزشمندی که Query store در تجمیع ان‌ها در اختیار ما قرار می‌دهد می‌پردازیم.
- در نهایت در فصل دهم کتاب به معرفی ابزارهایی می‌پردازیم که می‌توان از داده‌های Query store به شکل پیشرفته‌تر و بهتری برای تحلیل‌ها استفاده کرد. همچنین با ابزارهای Powershell اشنا خواهیم شد که برای config های اولیه و زیر ساخت Query store در نظر گرفته شده است که تنظیمات و بعضی موارد را به راحتی برای ما انجام خواهد داد.

اما وقت ان رسیده است که کمی در مورد Query Store و مزیت‌های ان صحبت کنیم. در این کتاب با یک عبارت جالب این موضوع عنوان شده است که Query store همانند جعبه سیاه هواپیما هست. در پشت همین جمله تفسیرهای مختلفی وجود دارد. اگر مستندهای مرتبط با صنایع هوایی و بررسی سوانح و سوابق پروازها را دنبال کرده باشید، متوجه خواهید شد که تیم های عملیاتی سوانح هوایی ، بر اساس یک نقشه راه مشخص از زمانی که این ارتباط با برج مراقبت برقرار می‌شود کلیه این ارتباط ها را مورد بررسی قرار می دهد . جالب اینجاست که تمامی وضعیت جانبی هواپیما نسبت به دما، ارتفاع و سایر اطلاعاتی که از سنسورهای مرتبط صادر می‌شود در این جعبه سیاه ذخیره می‌شوند. پس تا این جا متوجه شدیم که جعبه سیاهی که داریم چه قدر ارزشمند هست و چه اطلاعاتی را در اختیار ما قرار می‌دهد. اما جمع اوری این اطلاعات نیاز به یک دانش و سواد ویزه ایی برای تحلیل کلیه این اطلاعات خواهد داشت تا از بروز سوانح مختلف جلوگیری کند. لذا نیاز به داشبوردهایی خواهیم داشت که بتوانیم از اطلاعات فوق دانش مرتبط را استخراج کنیم.

مثال دوم را که می‌توان برای بحث Query store در دنیای واقعی در نظر گرفت بحث ارتباط بین مادر و کودک هست. از بدو تولد، مادر در حال جمع اوری اطلاعاتی هست که از بچه خود به سمت او می‌رسد. مادر این کودک در ذهن خود Baseline یا نقشه راهی را به مرور درست می‌کند که دقیقا چه زمان هایی کودکش نیازمند غذا هست. در چه زمان‌هایی نیاز به خواب داشته یا ممکن است که نیاز به محبت و نوازش مادر داشته باشد. خارج از این زمان‌ها ممکن است کودکش دچار مشکلی شده باشد که از ریتم، اهنگ و زمان گریه کردنش این اطلاعات را به راحتی متوجه می‌شود. پس این Baseline

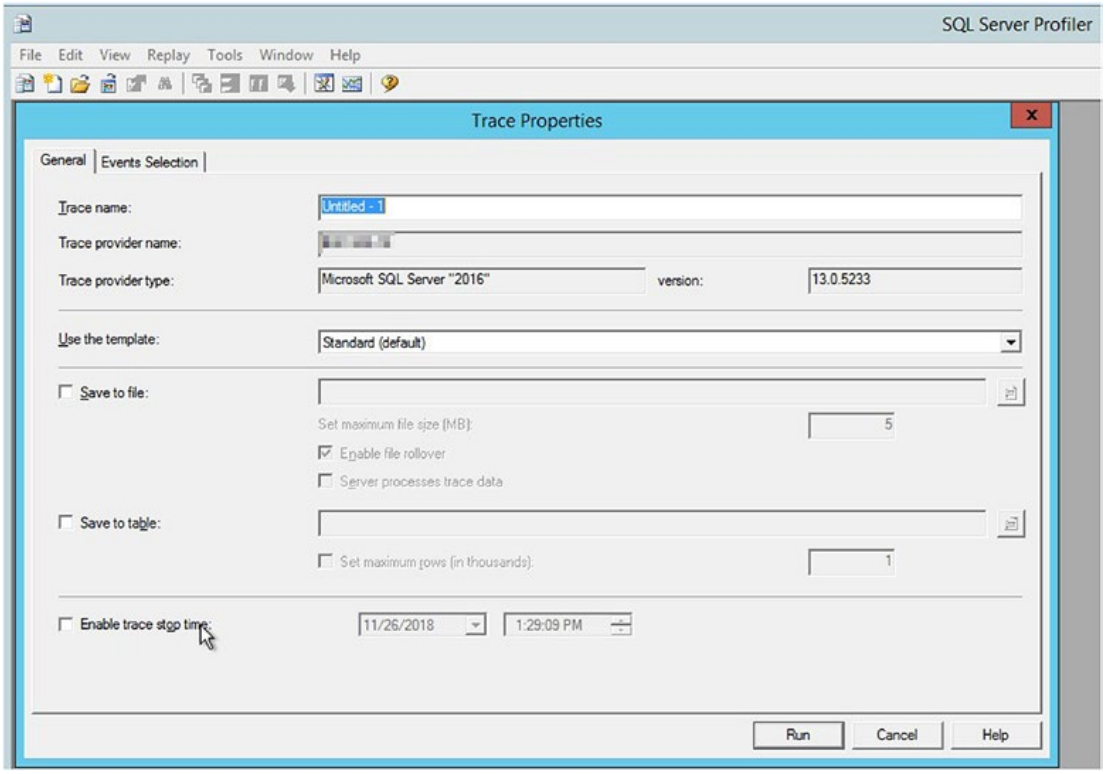
به صورت غریزی در این شخص به وجود می آید و چه بسا برای بچه‌های بعدی اش ، بتواند از دیتاهایی که به دست امده استفاده کند.

مثال سومی که می‌توان برای این موضوع مطرح کرد، بحث دوربین‌های مداربسته ایی هست که امروزه در اطراف خود مشاهده می‌کنیم. شاید زمانی که وارد ساختمان ایی بشوید متوجه می‌شوید از دوربین‌های مختلف در جهات مختلف استفاده شده است تا انالیز فیلم‌های ضبط شده با دقت بیشتری بررسی شود. سپس این قابلیت در اختیار شما قرار دارد که بسته به زمان ضبط ویدیو، با سرعت کم و زیاد ویدیو تحلیل شود یا بر روی یک نقطه خاص زوم شود که جزییات چهره‌ها و محیط پیرامون خود را بتوانیم دقیق‌تر تحلیل کنیم.

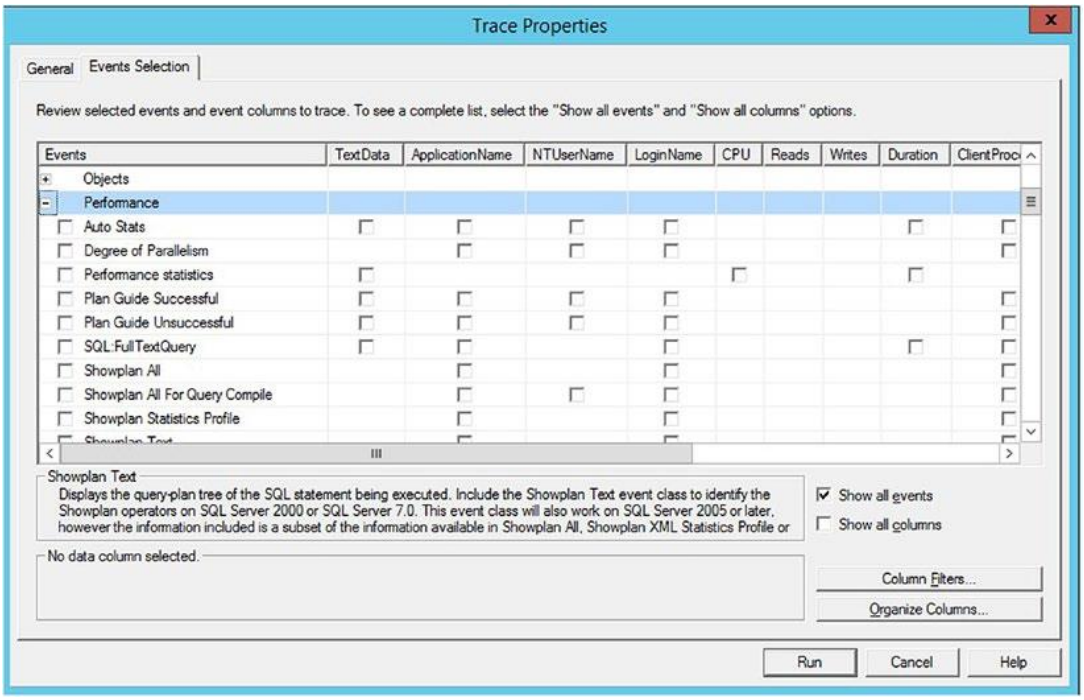
پس با راه اندازی چنین قابلیت بی‌نظیری در واقع مانند مادری بر سر کودک خود که همان بیزینس خومان هست مستقیما نظارت می‌کنیم. باید این کودک را به صورت مداوم تحت نظارت داشته باشیم و عملکردهای روتین آن را مرتبا زیر نظر گرفته و از نحوه کارکرد ان الگوهایی به دست بیاریم که در تحقق این هدف به ما کمک کند. در بخش اول این کتاب مفهوم Baseline معرفی شده است که یکی از اصلی‌ترین وظایف هر DBA ایی هست که آن را پیاده‌سازی کند. اما اگر بخواهیم یک تعریف مشخص‌تری نسبت به موضوع Baseline داشته باشیم، می‌توانیم به این شکل بیان کنیم که در واقع متریک‌ها و پارامترهایی را به مرور زمان جمع اوری کنیم و بر اساس ان به الگوهایی دست یابیم . این متریک‌ها در زمینه اطلاعات پردازشی Query ها نظیر CPU، Memory، Disk و میزان خواندن / نوشتن و زمان اجرای هر کوئری است. این اطلاعات باید در بازه‌های زمانی مشخص به تفکیک از هم مجزا شود. به عنوان مثال در سیستم‌های عملیاتی، زمان‌هایی وجود دارد که کلیه تراکنش‌ها در تایم‌های پیک و غیر پیک اتفاق می‌افند. لذا باید مشخص کنیم که این روند دقیقا چه زمان‌هایی در اوج خود است و چه زمان‌هایی خارج از این بازه هستیم و باید نحوه تنظیمات سیستم به چه صورت باشد.
در قسمت بعد، می‌خواهیم در مورد روش‌هایی صحبت کنیم که در نسخه‌های قبل از ورژن ۲۰۱۶ یا حتی بعد از آن هم وجود داشته یا دارد و این قابلیت را با این روش‌ها مقایسه و ارزیابی کنیم.

روش اول : استفاده از SQL Server Profiler

یکی از روش‌های جمع اوری اطلاعات بر اساس رخدادهایی که در SQL Server وجود دارد، استفاده از Profiler هست. در Profiler رخدادها یا Event ها در دسته‌بندی‌های مختلف تعریف و طبقه‌بندی شده است که برای جمع اوری اطلاعات بر اساس هر رخداد می‌توانیم به متن Query هایی که اجرا می‌شود دست پیدا کنیم. این برنامه فیلترهایی در اختیار کاربر می‌گذارد که صرفا بر روی ورودی‌های مشخص شده این فیلترها اعمال کند.



اما نکته ایی که وجود دارد این است که این قابلیت در ورژن‌های اخیر مطابق با گفته‌های سایت مایکروسافت، منسوخ شده است و استفاده از Extended Events ها، جایگزین این روش شده است. برای جمع اوری اطلاعات توسط این ابزار، طی یک بازه زمانی خاص، اطلاعات جمع اوری شده و توسط نرم افزاری به اسم Database Engine Tuning Advisor دیتاهای مرتبط، تحلیل و بررسی می‌شود. علت منسوخ شدن این روش در واقع به خاطر سربار زیادی هست که بر روی سرور تحمیل می‌شود و همچنین وابستگی به یک ابزار مجزا برای تحلیل و انالیز اطلاعات هست. به همین دلیل متوجه خواهید شد که ما این ترکیب دو نرم‌افزار را در یک قابلیت بی‌نظیرتر همراه با سربارهای خیلی کمنری در Query Store داریم. لازم به ذکر هست که تحلیل‌های Database Engine Tuning Advisor بر اساس پیشنهاد دادن ایندکس‌ها و Statistics هاست اما نکته ایی که باید به ان توجه داشته باشید این هست که دقیقا درچه بازه ایی شما دارید این اطلاعات را از سیستم جمع اوری می‌کنید؟ یعنی ممکن است که سربار Workload سیستم عملیاتی شما در زمان‌های مختلف، متفاوت بوده و اعمال تغییرات اصلا به نفع سیستم نباشد.



همان طور که در شکل بالا مشاهده می‌کنید، طبقه‌بندی‌های رخدادها نسبت به هر دسته مشخص شده است. به عنوان مثال در گروه مرتبط با Performance شاهد رخدادهای مختلفی هستیم که در شرایط و سناریوهای مختلفی رخ خواهد داد. مثلا اگر نسبت به کامپایل و ساخت پلن‌های اجرایی و نحوه ساخت‌ها اشنایی داشته باشید در فازهای مختلف درخت‌واره‌هایی تشکیل می‌شود که خروجی آن‌ها در سایر مراحل استفاده می‌شود که در نهایت منجر به ساخت پلن می‌شود. ساخت پلن یک هزینه زیادی به سیستم تحمیل می‌کند. این که بتوانیم مدت زمانی ساخت پلن‌های اجرایی و کامپایل ان‌ها را از سیستم استخراج کنیم بسیار می‌تواند اهمیت داشته باشد. یا به عنوان مثال، می‌خواهیم زمانی که که ایندکس‌ها به روزرسانی می‌شود را از سیستم استخراج کنیم که چه زمانی و با چه تعداد به روزرسانی برای یک ایندکس مشخص این اتفاق می‌افتد.

روش دوم : استفاده از Server-Side Traces

در این روش به جای این که نرم‌افزار Profiler را مستقیما اجرا کنید کلیه فیلترها و رخدادهایی که تنظیم کردیم را ایجاد کرده سپس از سیستم خروجی گرفته و اسکریپت مورد نظر را بر روی سرور اجرا می‌کنید. به این روش تریس سرور ساید اصطلاحا گفته می‌شد که بر مبنای یک سری Store procedure های سیستمی این Event ها ساخته شده و شروع به ذخیره‌سازی دیتا می‌کند. اطلاعات بیشتر را می‌توانید در لینک زیر مشاهده کنید:

https://docs.microsoft.com/en-us/sql/relationaldatabases/system-stored-procedures/sp-trace-seteventtransact-sql?view=sql-server-۲۰۱۷

اما اگر بخواهیم معایب دو روش Server-Side Traces و SQL Server Profiler را نسبت به Query store تشریح کنیم در واقع یک موضوع مهم خواهیم رسید و ان تجمیع و گروه‌بندی دیتاهای جمع آوری شده است که در دو روش فوق چنین قابلیتی وجود ندارد ولی در در Query store کلیه این اطلاعات بر اساس تنظیمات اولیه، به صورت مرتب جمع اوری و سپس تجمیع می‌شد. همچنین موضوع مهمی که در این کتاب شاهد ان هستیم بحث سرباری هست که این دو روش نسبت به Query store به سیستم اعمال می‌کنند.

روش سوم : استفاده از Extended Event

معرفی این روش به این دلیل نسبت به دو روش قبلی ارجحیت داشته که یک نسخه بسیار سبک‌تری برای جمع آوری دیتاها در اختیار شما قرار خواهد داد. در این روش هم مطابق با روش استفاده از Profiler، شاهد یک سری رخدادها در گروه‌بندی‌های مختلف با فیلترهای مختلف هستیم که باعث می‌شود رخدادهای کلیه ان موضوع به صورت مجزا ضبط و تهیه شود. در صورتی که Query زیر را اجرا نمایید شاهد ان خواهید بود که در سه طبقه‌بندی مرتبط با ('Stored Procedures', 'TSQL', 'Performance') شاهد این Event ها خواهید بود :

```
USE MASTER;

SELECT DISTINCT
    tb.trace_event_id,
    te.[name]       AS 'Event Class',
    em.package_name  AS 'Package',
    em.xe_event_name  AS 'XEvent Name',
    tca.[name]       AS 'Profiler Category'
FROM  (
      sys.trace_events te
      LEFT OUTER JOIN sys.trace_xe_event_map em
      ON te.trace_event_id =
      em.trace_event_id
    )
    LEFT OUTER JOIN sys.trace_event_bindings tb
       ON  em.trace_event_id = tb.trace_event_id
    INNER JOIN sys.trace_categories tca
       ON  tca.category_id = te.category_id
WHERE  tb.trace_event_id IS NOT NULL
    AND tca.[name] IN ('Stored Procedures', 'TSQL', 'Performance')
ORDER BY tb.trace_event_id
```

125 %

Results | Messages

	trace_event_id	Event Class	Package	XEvent Name	Profiler Category
1	10	RPC:Completed	sqlserver	rpc_completed	Stored Procedures
2	11	RPC:Starting	sqlserver	rpc_starting	Stored Procedures
3	12	SQL:BatchCompleted	sqlserver	sql_batch_completed	TSQL
4	13	SQL:BatchStarting	sqlserver	sql_batch_starting	TSQL
5	28	Degree of Parallelism	sqlserver	degree_of_parallelism	Performance
6	34	SP:CacheMiss	sqlserver	sp_cache_miss	Stored Procedures
7	35	SP:CacheInsert	sqlserver	sp_cache_insert	Stored Procedures
8	36	SP:CacheRemove	sqlserver	sp_cache_remove	Stored Procedures
9	37	SP:Recompile	sqlserver	sql_statement_recompile	Stored Procedures
10	38	SP:CacheHit	sqlserver	sp_cache_hit	Stored Procedures
11	40	SQL:StmtStarting	sqlserver	sql_statement_starting	TSQL
12	41	SQL:StmtCompleted	sqlserver	sql_statement_completed	TSQL
13	42	SP:Starting	sqlserver	module_start	Stored Procedures
14	43	SP:Completed	sqlserver	module_end	Stored Procedures
15	44	SP:StmtStarting	sqlserver	sp_statement_starting	Stored Procedures
16	45	SP:StmtCompleted	sqlserver	sp_statement_completed	Stored Procedures
17	58	Auto Stats	sqlserver	auto_stats	Performance
18	68	Showplan Text (Unencoded)	sqlserver	query_pre_execution_showplan	Performance
19	71	Prepare SQL	sqlserver	prepare_sql	TSQL
20	72	Exec Prepared SQL	sqlserver	exec_prepared_sql	TSQL
21	73	Unprepare SQL	sqlserver	unprepare_sql	TSQL
22	96	Showplan Text	sqlserver	query_pre_execution_showplan	Performance
23	97	Showplan All	sqlserver	query_pre_execution_showplan	Performance
24	98	Showplan Statistics Profile	sqlserver	query_post_execution_showplan	Performance
25	100	RPC Output Parameter	sqlserver	rpc_completed	Stored Procedures
26	122	Showplan XML	sqlserver	query_pre_execution_showplan	Performance
27	146	Showplan XML Statistics Profile	sqlserver	query_post_execution_showplan	Performance
28	165	Performance statistics	sqlserver	query_cache_removal_statistics	Performance
29	165	Performance statistics	sqlserver	query_pre_execution_showplan	Performance
30	165	Performance statistics	sqlserver	uncached_sql_batch_statistics	Performance
31	166	SQL:StmtRecompile	sqlserver	sql_statement_recompile	TSQL
32	168	Showplan XML For Query Co...	sqlserver	query_post_compilation_show...	Performance
33	169	Showplan All For Query Compile	sqlserver	query_post_compilation_show...	Performance
34	198	XQuery Static Type	sqlserver	xquery_static_type	TSQL
35	217	Plan Guide Successful	sqlserver	plan_guide_successful	Performance
36	218	Plan Guide Unsuccessful	sqlserver	plan_guide_unsuccessful	Performance

Query executed successfully.

باز در این روش، شاهد این هستیم که محدودیت‌هایی که در روش‌های قبلی وجود داشت اینجا نیز شاهد ان هستیم و همچنین باید در یک فواصل زمانی خاص این جمع اوری دیتا انجام شود و ممکن است منجر به بزرگ شدن فایل نهایی شود. لذا در Query store علاوه بر قابلیت‌های تجمیع دیتا که در هیچکدام از روش‌های بالا وجود نداشت را خواهیم داشت به علاوه این که می‌توانیم کنترلی بر روی ذخیره فایل‌ها نیز داشته باشیم و با تنظیمات و پارامترهای مرتبط با آن به نحوه موثرتری ذخیره ان‌ها را انجام دهیم.

روش چهارم: استفاده از Dynamic Management Views (DMVs)

مبحث DMV ها یکی از جذاب‌ترین مباحث موجود برای تحلیل و انالیز اطلاعات از بخش‌های مختلف سیستم هست. توابع و View هایی وجود دارند که با ترکیب ان‌ها با یکدیگر به نتایج فوق العاده ایی می‌توان رسید و از جنبه‌های مختلف می‌توانیم مشکلات سیستم را استخراج کنیم. به عنوان مثال نمونه ایی از این موارد را می‌توانید در لیست زیر مشاهده کنید:

- sys. dm_exec_query_stats
- sys. dm_exec_requests
- sys. dm_exec_cursors
- sys. dm_exec_xml_handles
- sys. dm_exec_query_memory_grants
- sys. dm_exec_connections

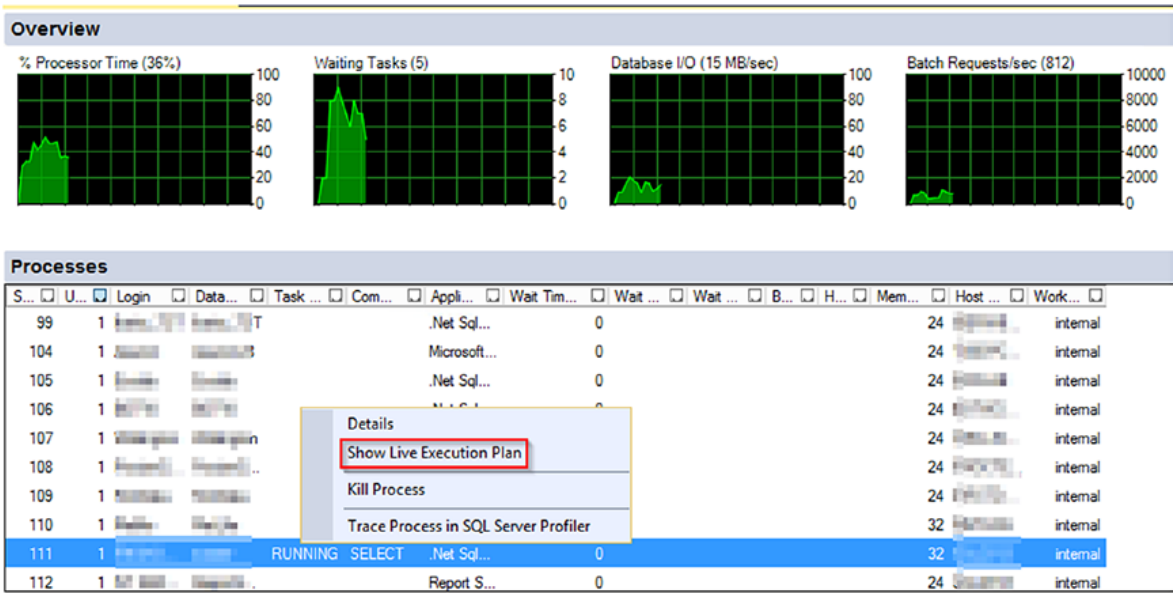
تحلیل هر کدام از این View ها بسته به سناریوهای مختلف و ستون‌هایی که به شما نمایش داده می‌شود متفاوت خواهد بود. در مقالات مجزایی به این بحث خواهیم پرداخت که در چه زمان‌هایی از چه DMV هایی و برای چه کاری استفاده کنیم. در صورتی که علاقه‌مند به مطالعه تخصصی در این حوزه بودید می‌توانید از کتاب SQL Server DMVs in Action: Better Queries with Dynamic Management Views استفاده کنید.

اما در این قسمت به صورت تخصصی بر روی موضوع Query store و مقایسه این روش با روش‌های دیگر می‌پردازیم. استفاده از بحث DMV ها در تحلیل اطلاعات Query ها و پلن‌های اجرایی دو عیب بزرگ نسبت به بحث Query store دارد. موضوع اول این هست که اطلاعاتی که توسط DMV ها نمایش داده می‌شد بر روی دیسک ذخیره نمی‌شد و مستقیم در حافظه هست. لذا با ریستارت شدن سرور کلیه این اطلاعات پاک می‌شود و اصلا مبنا و ملاک قطعی برای تحلیل‌ها نیست چون ممکن است که به هر دلیل نظیر قطعی برق یا سایر موارد دیگر این اتفاق رخ دهد. موضوع دوم هم مرتبط با این فرایند داخلی در Sql server هست.در صورتی که Engine متوجه شود که پلن‌های اجرایی وجود دارد که بلااستفاده تشخیص داده شده باشد، ان‌ها را از حافظه خارج می‌کند. لذا ما اطلاعات این تیپ پلن‌ها را از دست خواهیم داد و هیچ تحلیلی نمیتوانیم بر روی اناها انجام دهیم و Baseline ایی در این حالت وجود نخواهد داشت.

لذا برای این حالت نیاز هست که به صورت مرتب این اطلاعات در یک دیتابیس مجزا ذخیره شود که بتوانیم این محدودیت را دور بزنیم و دیتاها بر روی دیسک ذخیره کنیم. یعنی یک کار اضافی! در صورتی که زمانی قابلیت‌های Query store را تشریح کنیم متوجه می‌شویم که یکی از اهداف طراحی این قابلیت این بوده که از صرف کارهای تکراری خودداری شود.

روش پنجم : Lightweight Show Plan Trace Flag

از نسخه SQL Server ۲۰۱۴ایی بودیم که در زمان اجرا کوئری‌ها بتونیم پلن‌های اجرایی ان را مشاهده کنیم. این Trace Flag به شماره ۷۴۱۲ این قابلیت را در اختیار ما قرار میدهد که بتوانیم به این پلن‌های اجرایی دسترسی داشته باشیم. به مرور در نسخه‌های بالاتر نیز، سربار این عملیات کاهش پیدا کرد و تغییراتی به همراه داشته است. اخیرا در ورژن ۲۰۱۹ به صورت پیش فرض فعال هست و به راحتی می‌توانید از طریق DMV ایی به نام sys.dm_exec_query_statistics_xml اخرین وضعیت پلن‌های اجرایی را نیز مشاهده کنیم. همچنین با استفاده از Activity monitor این کار نیز قابل انجام هست. فعال بودن این FLAG بر روی سرور به صورت حدودی، تقریبا ۲٪ سربار بر روی CPU اعمال خواهد کرد. همان طور که در تصویر پایین نیز مشاهده می‌کنید به این طریق دسترسی به پلن اجرایی در حال اجرا به این روش نیز قابل دسترس هست:



همان طور که توضیح داده شد این روش نمیتواند جایگزین مناسبی برای بحث Baseline باشد. چرا که به صورت موقت و لحظه ایی می‌توانیم اخرین فعالیت‌ها بر روی سرور مشاهده کنیم.

روش ششم: استفاده از sys. dm_exec_query_plan_stats

در SQL Server ۲۰۱۹ تابعی معرفی شد به اسم بالا که در واقع با استفاده از ان می توانید اخرین پلن های ذخیره شده برای کوئری ها را مشاهده کنید . برای مشاهده این پلن های اجرایی می توانید از طریق کوئری زیر اقدام کنید :

```
SELECT er.session_id,
    er.start_time,
    er.status,
    er.command,
    st.text,
    qp.query_plan       AS cached_plan,
    qps.query_plan      AS last_actual_exec_plan
FROM  sys.dm_exec_requests  AS er
    OUTER APPLY sys.dm_exec_query_plan(er.plan_handle) qp
OUTER APPLY sys.dm_exec_sql_text(er.sql_handle) st
OUTER APPLY sys.dm_exec_query_plan_stats(er.plan_handle) qps
WHERE  session_id > ۵۰
    AND STATUS IN ('running', 'suspended');
```

همان طور که توضیح داده شد این روش نمیتواند جایگزین مناسبی برای بحث Baseline باشد. نکته ایی که در کوئری بالا وجود دارد این هست که session_id > ۵۰ معمولا برای کوئری‌هایی هست که از سمت برنامه یا کاربر به سمت Engine ارسال می‌شود. لذا ملاک تصمیم گیری بر روی این مدل از پلن‌های اجرایی را می‌توانید با استفاده از این شرط در کوئری خود برقرار کنید. یکی از دلایلی که ضعف این روش را در برابر Query Store نمایش می‌دهد، این هست که تا زمانی که پلن‌های اجرایی شما در حافظه وجود دارند، شما می‌توانید از این ویو سیستمی خروجی دریافت کنید. لذا با ریستارت شدن سرور یا هر دلیل دیگری که ممکن است پلن اجرایی از حافظه خارج شود این اطلاعات از دسترس خارج می‌شود. در این صورت باید یک Baseline دستی درست شود که کلیه این اطلاعات را دربازه‌های زمانی خاص ذخیره و نگهداری شود. در صورتی که Query store کلیه این فرایندها را برای ما به راحتی انجام می‌دهد.

در این جا لازم هست که یک نکته بسیار مهم اشاره شود. ذخیره و نگه داری پلن‌های اجرایی به همراه اطلاعات کلی آن‌ها ممکن است در DMV ها یا DMF های مختلفی قرارداشته باشد. به عنوان مثال برای بررسی کلیه سشن‌های متصل به SQL Server و درخواست‌هایی که به سمت Engine ارسال شده است باید از sys. dm_exec_requests و sys.

dm_exec_sessions استفاده شود. همچنین برای بررسی این که کوئری‌های ارسال شده به سمت سرور چه مقدار نیاز به فضای حافظه دارند یا فضای حافظه در اختیار گرفتند، می‌توانیم از sys. dm_exec_query_memory_grants استفاده کنیم. پس نکته در این جاست که DMV های مختلف از جنبه‌های مختلف اطلاعات تکمیلی و بسیار خوبی به ما می‌دهند. اما برای رسیدن به یک دیتای ارزشمند و قابل تحلیل باید موارد مورد نیاز را از تک به تک این DMV ها استخراج کنیم. لذا ممکن است که برای تحلیل‌های مختلف نیازمند ایجاد چندین کوئری پیچیده باشیم که از ابعاد گوناگون مشکلات سیستم را استخراج کنیم. این روش در معرفی فصل اول کتاب هم به این شکل بوده است که صرفا هدف اشنایی با کلیه قابلیت‌ها و توابعی هست که می‌تواند به ما کمک کند در صورت نیاز Baseline سفارشی خود را ایجاد کرده و از ان استفاده کنیم.

نگاه دقیق‌تر به بحث Query store

همان طور که در روش‌های قبلی توضیح داده شد، هر روش نسبت به سایر راهکارهای ارایه شده مزایا و معایبی به همراه داشت. اما Query Store این بازی را به شکل دیگری تغییر داده است. مهمترین مزیت راه اندازی قابلیت Query store در سرورهای عملیاتی، ذخیره و نگهداری کلیه اطلاعات مرتبط با پلن‌های اجرایی و اطلاعات ارزشمند دیگری هست که بر روی دیسک ذخیره می‌شود نه بر روی حافظه. همین عامل باعث می‌شود که تحلیل‌ها را بتوانیم در هر زمان در دسترس داشته باشیم. مزیت دوم پارمترهای کنترلی این قابلیت هست که بر چه اساسی قرار است این تنظیمات انجام شود. لذا راه اندازی این قابلیت بر روی هر سناریو ایی عملا بدون مشکل خواهد بود و بهترین نتایج را در برخواهد داشت.

در لیست زیر به اماری که می توانیم از Query Store دریافت کنیم خواهیم رسید :

- Execution count
- Duration
- CPU time
- Logical reads
- Logical writes
- Physical reads
- CLR time
- DOP
- Memory consumption
- Row count

در خصوص گزینه‌های بالا، Query store اطلاعات مختلف از ماکزیمم، مینیموم، متوسط و مقایر استاندارد هر یک از این ایتم‌ها را در اختیار ما قرار خواهد داد. به همین دلیل با وجود این اطلاعات می‌توانیم در سناریوها و موارد مختلفی از این قابلیت به خوبی استفاده کنیم. این سناریوها عبارت‌اند از :

۱. مشاهده و اصلاح کوئری هایی که با پسرفت در اجرا همراه بودند
۲. مشاهده و اصلاح کوئری هایی که بالاترین منابع را از سیستم برای اجرا دریافت کردند .
۳. تست های A/B
۴. پایداری و استقرار صحیح در هنگام ارتقا ورژن SQL SERVER بلااخص از نسخه هایی پایین تر به نسخه های بالاتر (با توجه به تغییرات الگوریتم ها در نسخه ۲۰۱۴ ، عملیات ارتقا سیستم با کندی های زیادی همراه بوده است .)
۵. مشاهده و تشخیص Ad Hoc Query هایی که سربار زیادی بر روی سیستم اعمال می کند

جمع‌بندی:

کلیه مواردی که در قسمت بالا توضیح داده شد، در مقاله مجزایی به تفصیل کلیه نکات مرتبط با آن‌ها مطرح می‌شود. همچنین مایکروسافت نیز بر استفاده از این قابلیت تاکید زیادی دارد که Performance سیستم‌های عملیاتی را بتوانیم با این روش‌ها بهبود دهیم. در این مقاله سعی بر این بود که قبل از ورود به این مبحث، با کلیه روش‌هایی که برای پیدا کردن کوئری‌های مشکل دار در سرور داریم، اشنا شویم، سپس در مرحله بعد بتوانیم مزایا و معایب هر کدام را با یکدیگر مقایسه کنیم. همان طور که مشاهده کردید Query Store قابلیتی هست که از هر روش، مزیت‌های ان را پیاده‌سازی کرده و تجمیع آن‌ها در قالب یک Baseline مشخص باعث بهبود در عملکرد و تحلیل‌های شما خواهد شد. در قسمت های بعدی با معماری این قابلیت بیشتر آشنا می شویم و گزارشات و تنظیمات مختص به این قابلیت را به با جزییات بیشتری توضیح خواهیم داد .

منابع مقاله:

https://www.sqlskills.com/blogs/erin/query-store-performance-overhead/
https://git.ir/pluralsight-solving-real-world-problems-with-sql-server-۲۰۱۶-query-store/
https://www.pluralsight.com/courses/solving-real-world-problems-sql-server-۲۰۱۶-query-store
https://www.sqlskills.com/blogs/erin/query-store-settings/
https://www.sqlskills.com/blogs/erin/query-store-examples-stories-from-customers/
https://docs.microsoft.com/en-us/sql/relational-databases/performance/best-practice-with-the-query-store?view=sql-server-ver۱۵
https://docs.microsoft.com/en-us/sql/relational-databases/performance/query-store-usage-scenarios?view=sql-server-ver۱۵