عنوان مقاله: آشنایی با Query store | بخش دوم
نویسنده مقاله: تیم فنی نیک آموز
تاریخ انتشار: دی ماه ۱۴۰۰
منبع: https://nikamooz.com/query-store-part۰۲

مقدمه

در قسمت قبلی **آشنایی با Query store | بخش اول** فصل اول کتاب (Query Store for SQL Server ۲۰۱۹) را با هم بررسی کردیم و راه حل‌ها و روش‌هایی که قبل از استفاده از Query store استفاده می‌شد را مشاهده کردید. هر کدام از روش‌های مطرح شده، مزایا و معایبی نسبت به هم دیگر داشتند که به تفکیک بررسی شد و سپس به این جمع‌بندی رسیدیم که برای آنالیز کوئری‌ها و پلن‌های اجرایی آن‌ها باید بتوانیم بر اساس یک Baseline و نقشه راه مشخصی نسبت به تحلیل آن‌ها اقدام کنیم. اما در این فصل به این می‌پردازیم که مقداری با معماری داخلی Query store آشنا شویم و سپس پارامترهایی که این قابلیت در اختیار ما قرار می‌دهد را بررسی کنیم.

معماری داخلی Query store

همان طور که اشاره شد، ماهیت جمع آوری و ذخیره‌سازی اطلاعاتی که توسط Query store ذخیره می‌شود بر اساس یک Baseline است. اما نکته ایی که هست، در خصوص جمع آوری دیتا و اطلاعات، این هست که این ذخیره و جمع آوری دیتاها باید با یک معماری خوبی طراحی شده باشد که حداقل سربار را بر روی سیستم تحمیل کند که خودش عامل کندی سیستم نباشد. قبل از این که این معماری را تشریح کنیم نیاز هست که بدانیم Query Store در حالت کلی چه اطلاعاتی را در اختیار ما قرار می‌دهد؟ این اطلاعات در چه جداول یا View های سیستمی ذخیره می‌شود و به چه شکل قابل استفاده است؟

به صورت کلی استفاده از Query Store سه مدل از اطلاعات مرتبط با Query را در اختیار ما قرار می دهد که به شرح زیر هستند:

- **اطلاعات مرتبط با Query and Plan Information**
  - در این بخش اطلاعات مرتبط با خود Query و پلن های اجرایی آن که توسط Query Optimizer تولید می شود ذخیره و نگه داری می شود.
- **اطلاعات مرتبط با Query Runtime Information**

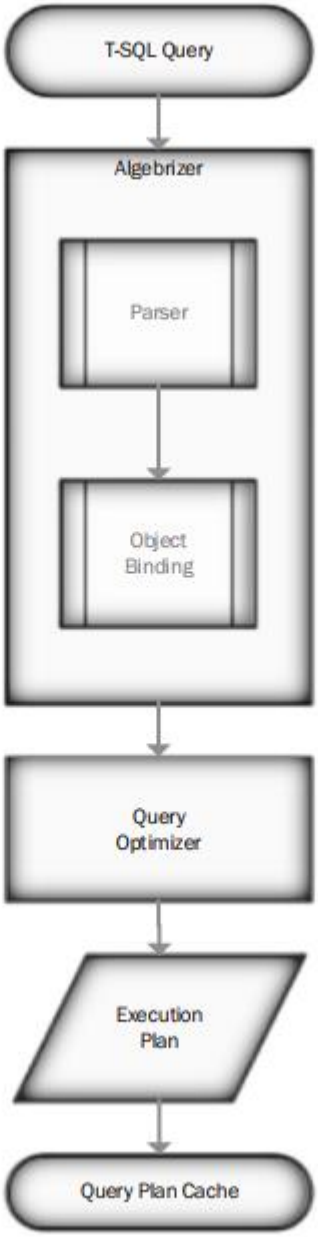
- در این بخش اطلاعات مرتبط با متریک‌ها و زمان اجرای کوئری‌ها مشخص می‌شد. این که کوئری ما چند بار اجرا شده است و در هر اجرا چه مشخصاتی را از خود ثبت کرده است. به صورت کلی این بخش شامل اطلاعاتی هست که کاملا مرتبط با بحث Performance کوئری خواهد بود و می‌توانیم بر اساس این امار و ارقام، به تحلیل‌های جامعی رسید.

- **اطلاعات مرتبط با Query Wait Statistics**
  - در این بخش اطلاعات مرتبط با Wait هایی که هر Query داراست نمایش داده می شود . به صورت کلی بحث Wait ها شامل چند طبقه بندی مختلف هست و می توانیم علل کندی Query ها بر اساس این طبقه بندی دریافت کنیم .

*** نکته: استفاده از Query Store به ازای هر دیتابیس در نظر گرفته می‌شود و تنظیمات کلی ان ممکن است متناسب با Workload و سربار هر دیتابیسی به صورت جداگانه در نظر گرفته شود. این قابلیت باعث می‌شود که کلیه امار و اطلاعات حاصل شده به ازای هر دیتابیس به صورت مجزا طبقه‌بندی و تحلیل گردد .

قابلیت Query Store در ابتدا در نسخه Azure مایکروسافت عملیاتی شد که به صورت تستی قیدبک لازم را بر روی چندین سناریو تست کنن. سپس از نسخه ۲۰۱۶ به بعد شاهد این قابلیت بودیم. در هر نسخه بعد از این ورژن شاهد این هستیم که فیچرهای کامل‌تری به این قابلیت اضافه شده است. یکی از این فیچرها که از نسخه ۲۰۱۷ به بعد به Query Store اضافه شد، قابلیت جمع آوری و رصد کردن Wait Statistics هایی هست که هر کوئری در هر اجرا از خود بر جای می‌گذارد. همچنین قابلیت‌هایی نظیر APRC که یکی از بهترین قابلیت‌ها در Query Store است، نیز در ابتدا در نسخه ۲۰۱۶ وجود نداشت و سپس در نسخه‌های بعدی اضافه گردید. لذا توصیه می‌شود در صورت امکان از اخرین ورژن‌های SQL SERVER برای بهره مندی از کلیه این قابلیت‌ها استفاده شود. در سایت مایکروسافت نیز نکاتی در این خصوص ارایه شده است که باگ‌هایی در نسخه‌های پایین‌تر وجود داشته که باعث اعمال سرباری بر روی حافظه یا سایر منابع سیستم شده است که در ورژن‌های بعدی این موارد به صورت کامل برطرف شده است.

خب تا اینجا متوجه شدیم که مجموعه اطلاعاتی جمع آوری شده توسط Query Store که در این کتاب به اسم Data Set مشخص شده است شامل چه مواردی است. در ادامه معماری اصلی Query Store را تشریح می‌کنیم که این اطلاعات هر کدام در چه مرحله ایی از اجرای Query ها ذخیره می‌شود که کمترین سربار را بر روی سیستم نیز تحمیل کند.
زمانی که یک Query به سمت Engine ارسال می‌شود باید یک مراحلی طی شود به جهت این که پلن اجرای آن ساخته شود. این مراحل مطابق شکل پایین بدین شکل است:



همان طور که در شکل بالا مشاهده می‌کنید، فرایند اجرای هر Query در SQL SERVER مطابق با مدل بالا طی چند بخش به ترتیب اجرا می‌شود. در مرحله اول Parser را می‌بینیم که موظف هست سینتکس Query را از لحاظ گرامر و کلمات کلیدی که مرتبط با دستورات T-Sql هست مورد بررسی قرار دهد. در صورتی که مشکلی در سینتکس Query های نبود، خروجی آن به مرحله بعدی ارسال می‌شود. در مرحله دوم باید Engine از صحت استفاده از جداول یا کلیه Objectهای استفاده شده در Query شما مطمئن شود که بتوانید کلیه رفرنس‌های مرتبط را در دیتابیس پیدا کند. حتما با این مشکل مواجه شدید که پیغام خطایی مشاهده شود که جدولی وجود ندارد یا فیلدهای ان به اشتباه ذکر شده است. این موارد همگی در این بخش کنترل می‌شود. در صورتی که کلیه این Object ها به درستی استفاده شده باشد خروجی این مرحله به سمت Query

Optimizer ارسال می‌شود. در این بخش Query Optimizer موظف هست که با توجه به Object های استفاده شده و سبک Query نویسی شما از ایندکس‌ها و Statistics ها و کلیه مواردی که بتوانید به خروجی مورد نظر برسد استفاده کند. استفاده از این موارد به این دلیل هست که بهینه‌ترین پلن اجرایی ایجاد شده که در سریع‌ترین زمان ممکن بر اساس پلن اجرایی خروجی به کاربر نمایش داده شود. نکته حائز اهمیت این هست که تغییراتی در ورژن ۲۰۱۴ بر روی بحث Execution plan ها اعمال شده است که در عمده سناریوها باعث بهبود در اجرای کوئری‌ها شده است و در مواردی نیز مشاهده شده است که با کندی‌های خاصی همراه بوده است. برای این قسمت می‌توانید این تغییرات و الگوی‌های اصلاح شده در الگوریتم پلن‌های اجرایی را از سایت مایکروسافت مشاهده کنید.

برای دریافت اطلاعات بیشتر در این خصوص می توانید از لینک زیر جزییات مربوط به این تغییر الگوریتم هایی که در پلن های اجرایی اعمال شده است را مطالعه بفرمایید : https://www.sqlshack.com/whats-new-sql-server-۲۰۱۴-cardinality-estimator/

در مرحله بعد زمانی که بهینه‌ترین پلن اجرایی ایجاد شد، نیاز هست که بر اساس نقشه راه مورد نظر، Query ما اجرا شود و در مرحله بعدی این پلن کش شود. کش شدن پلن‌های اجرایی به این دلیل هست که هزینه ساخت یک پلن به عنوان یک سربار برای سیستم در نظر گرفته شده است. به همین دلیل پلن‌های اجرایی کش می‌شود تا در صورتی که مجددا Query مورد نظر ارسال شد بر اساس پلن کش شده خروجی را به کاربر نمایش دهد. البته ذکر این نکته نیز حائز اهمیت است که ممکن است به هر دلیل این پلن‌های اجرایی از حافظه کش خارج شود و دوباره ساخته شود. این پلن‌های اجرایی در محلی به اسم Plan Cache که بخشی از حافظه هست ذخیره و نگه داری می‌شود.

نکته ایی که در این قسمت باید به خاطر داشته باشید این هست که Query Store قابلیت جمع آوری و ذخیره آن دسته از کوئری‌هایی را داراست که به شکل DML یا Data Manipulation Language نوشته می‌شود. این مدل Query ها شامل عملیات‌هایی نظیر

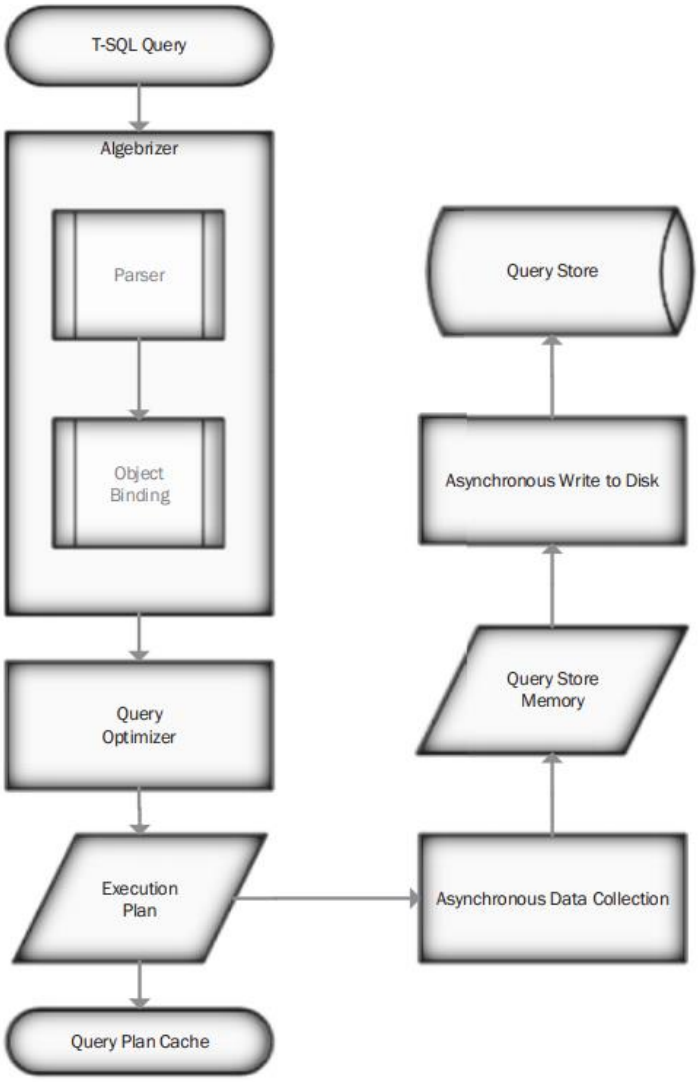
- SELECT
- UPDATE
- DELETE
- INSERT

میباشد. کوئری‌هایی که به شکل Data Definition Language (DDL) نوشته می‌شود این قابلیت را ندارند. نمونه دستوراتی که در زمینه DDL استفاده می شود شامل عملیات زیر هست :

- CREATE
- DROP
- ALTER

- TRUNCATE removed.
- COMMENT
- RENAME

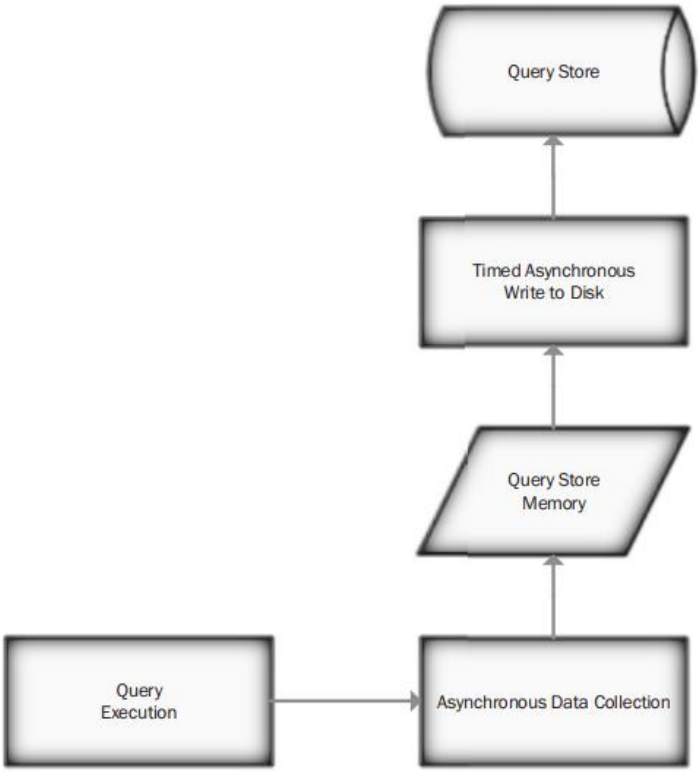
همان طور که توضیح دادیم مراحلی برای اجرای یک کوئری در نظر گرفته شده است. در مرحله بعد میخواهیم نقش Query Store را در این فرایند تشریح کنیم. به شکل زیر دقت کنید:



همان طور که در شکل بالا مشخص هست، زمانی که پلن اجرایی ساخته می‌شود به موازات آن یک فرایندی به شکل آسنکرون اجرا شده است. بدین جهت که اطلاعات کوئری و پلن اجرایی را بتواند از حافظه به سمت دیسک منتقل کند. در مرحله Query Store Memory یک فضای مجزایی از پلن کش در نظر گرفته شده که در این کتاب با نام temporary storage

space مشخص گردیده است. به گفته کتاب، این روش ذخیره‌سازی به شکل آسنکرون کمترین سربار را بر روی سیستم تحمیل می‌کند. در این بخش متوجه شدیم که اطلاعات مرتبط با Query و پلن اجرایی ان به شکل ذخیره می‌شود.

در بخش دوم همان طور که توضیح داده شد باید در کنار پلن‌های اجرایی، مشخصات مرتبط با نحوه اجرا و متریک‌های مرتبط با کوئری در خصوص دفعات اجرا و زمان اجرا و غیره باید ذخیره شود. نحوه ذخیره‌سازی این دو المان که شامل Query Runtime Information و Wait Statistics هستند بسیار ساده تر هست.
زمانی که کوئری شما اجرا می‌شود، مطابق با مرحله قبل مجدد فرایند آسنکرون اجرا شده که اطلاعات مرتبط با Query Runtime Information باید ذخیره شود. به شکل زیر دقت کنید:



شاید در این قسمت سوالی مطرح شود که فرایند ذخیره‌سازی طبق چه زمانی اطلاعات را از حافظه استخراج و به سمت دیسک منتقل می‌کند. برای جواب به این سوال باید گفت که پارامتری در تنظیمات Query Store داریم که مدت زمان رجوع به محل ذخیره‌سازی و جمع آوری دیتا را مشخص می‌کند. این مدت زمان به صورت پیش فرض ۱۵ دقیقه در نظر گرفته شده است. نکته ایی که در اینجا مهم هست، نوع ذخیره‌سازی این اطلاعات بر روی حافظه و دیسک هست. زمانی که این اطلاعات بر روی حافظه هستند در واقع به شکل تجمیع شده یا aggregated در نظر گرفته می‌شود. زمانی که فرایند انتقال به سمت دیسک انجام می‌شود این اطلاعات بر طبق تنظیمات دیگری که در Query Store انجام می‌شود مجددا reaggregated می‌شود و می توانیم به شکل بهتر و با جزییات بیشتری تحلیل‌هایی بر روی آن‌ها انجام دهیم. ملاک تفکیک

و خارج شدن این اطلاعات بر حسب زمان اجرایی ان‌ها به صورت پیش فرض هر ۶۰ دقیقه یکبار در نظر گرفته می‌شود. به همین دلیل می‌توانید در بازه‌های زمانی مشخص فیلترهای لازم را بر روی اطلاعات ذخیره شده در دیسک، اعمال نمایید.

این نکته را در نظر بگیرید که Query Store نسبت به تغییراتی که بر روی کوئری‌ها انجام می‌دهد حساسیت‌های خود را دارا می‌باشد. به عنوان مثال زمانی که عملیات recompile بر روی Query اجرا شود پلن اجرایی جدید در کنار پلن قبلی ذخیره می‌شود که سوابق آن‌ها را بتوانیم مشاهده کنیم. همان طور که مشاهده شد در فرایند آسنکرون، ممکن است اطلاعات از بین برود زیرا طی فواصل زمانی مشخص قرار هست که اطلاعات را از حافظه به سمت دیسک منتقل کنیم. بدین منظور می‌توانیم از sys. sp_query_store_flush_db استفاده کنیم تا کلیه اطلاعات ذخیره شده در حافظه سریعا به دیسک منتقل شود. این احتمال وجود دارد که به دلیل ریستارت شدن یا خاموش شدن سرور یا هر مورد دیگری این اطلاعات از بین برود یا نیاز باشد طی فرایندی ریستارت سرور انجام شود و قبل از ان این اطلاعات را ذخیره کنیم.

جداول و ساختار Query Store

قبل از این که به صورت کامل وارد ساختار های مرتبط با Query Store شویم نیاز هست که به مفهومی به اسم اشاره کنیم . در Engine دیتابیس Sql Server به ازای مفاهیم مختلف Catalog Views های متعددی وجود دارند . مفهوم Catalog Views نیز به این معنی است که حالت سفارشی سازی شده ایی از اطلاعات هست که در اختیار کاربر قرار داده شود . مایکروسافت برای قابلیت Query Store نیز Catalog Views های مرتبط با آن را در نظر گرفته است که به صورت تخصصی بتوانیم اطلاعات مرتبط با سه بخش بالا که به آنها اشاره شد را به صورت کامل از سیستم استخراج کنیم . در لیست زیر این طبقه بندی را مشاهده می کنید

- Always On Availability Groups Catalog Views (Transact-SQL)
- Azure SQL Database Catalog Views
- Change Tracking Catalog Views (Transact-SQL)
- CLR Assembly Catalog Views (Transact-SQL)
- Data Collector Views (Transact-SQL)
- Data Spaces (Transact-SQL)
- Database Mail Views (Transact-SQL)
- Database Mirroring Witness Catalog Views (Transact-SQL)
- Databases and Files Catalog Views (Transact-SQL)
- Endpoints Catalog Views (Transact-SQL)
- Extended Events Catalog Views (Transact-SQL)
- Extended Properties Catalog Views (Transact-SQL)
- External Operations Catalog Views (Transact-SQL)

- Filestream and FileTable Catalog Views (Transact-SQL)
- Full-Text Search and Semantic Search Catalog Views (Transact-SQL)
- Linked Servers Catalog Views (Transact-SQL)
- Messages (for Errors) Catalog Views (Transact-SQL)
- Object Catalog Views (Transact-SQL)
- Partition Function Catalog Views (Transact-SQL)
- Policy-Based Management Views (Transact-SQL)
- Resource Governor Catalog Views (Transact-SQL)
- Query Store Catalog Views (Transact-SQL)
- Scalar Types Catalog Views (Transact-SQL)
- Schemas Catalog Views (Transact-SQL)
- Security Catalog Views (Transact-SQL)
- Service Broker Catalog Views (Transact-SQL)
- Server-wide Configuration Catalog Views (Transact-SQL)
- Spatial Data Catalog Views
- Azure Synapse Analytics and Parallel Data Warehouse Catalog Views
- Stretch Database Catalog Views (Transact-SQL)
- XML Schemas (XML Type System) Catalog Views (Transact-SQL)

اطلاعات مختلفی که مرتبط با Query Store هستند در Catalog Viewsهای زیر ذخیره می‌شوند:

- sys.database_query_store_options (Transact-SQL)
- sys.query_context_settings (Transact-SQL)
- sys.query_store_plan (Transact-SQL)
- sys.query_store_query (Transact-SQL)
- sys.query_store_query_text (Transact-SQL)
- sys.query_store_wait_stats (Transact-SQL)
- sys.query_store_runtime_stats (Transact-SQL)
- sys.query_store_runtime_stats_interval (Transact-SQL)
- sys.query_store_query_hints (Transact-SQL)

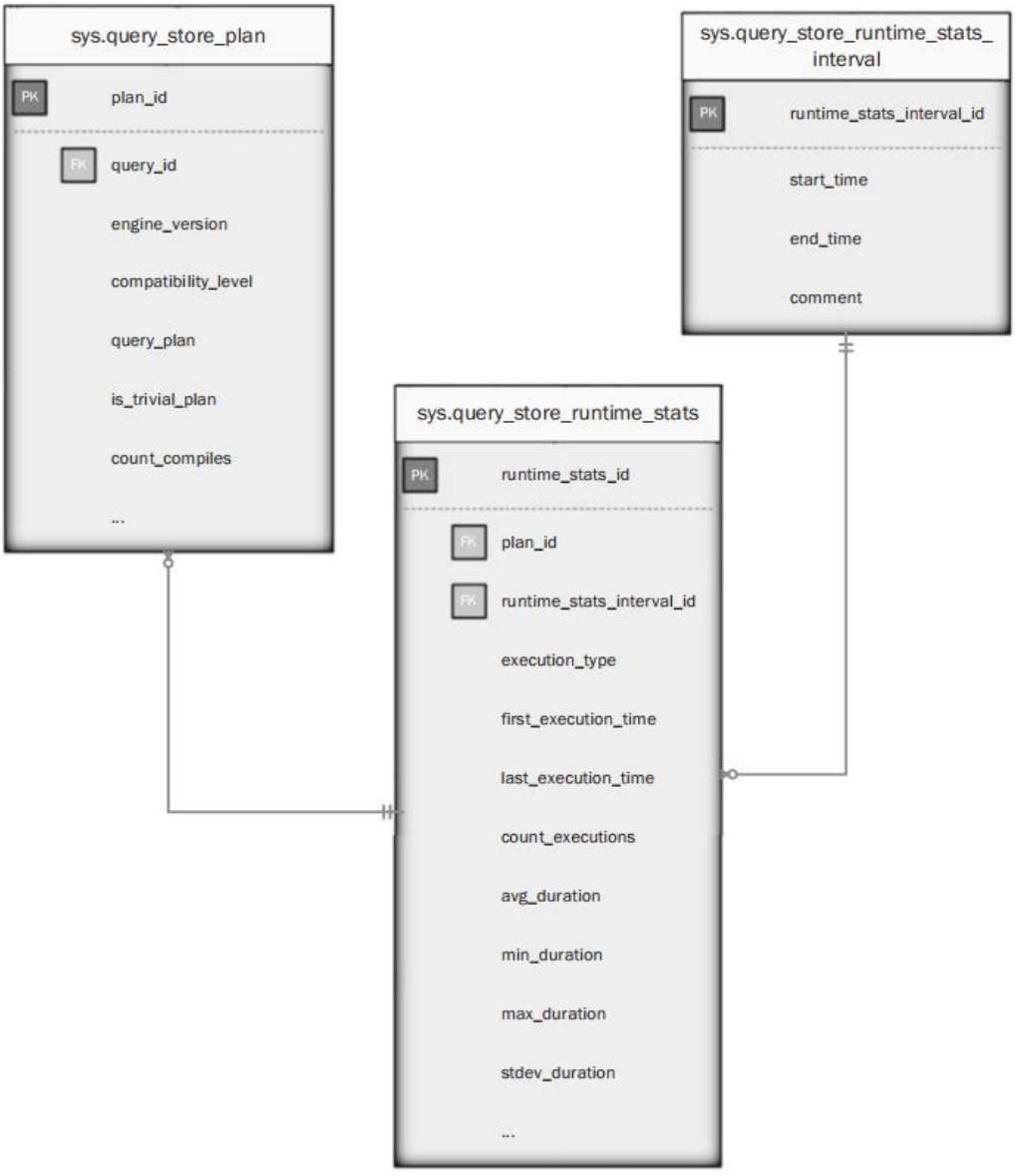
در هر Catalog View ایی که مشاهده می‌کنید اطلاعات مختلفی جهت تحلیل مورد استفاده قرار می‌گیرد. زمانی که هر کدام از Catalog Viewهای بالا را اجرا کنید ستون‌های را مشاهده می‌کنید که جزییات مرتبط با آن بخش به صورت کامل نمایش داده می‌شود. به عنوان مثال در sys. query_store_plan می‌توانیم ستون‌های مختلفی را مشاهده کنیم که یکی از آن‌ها مرتبط با بحث compatibility_level دیتابیس هست که مشخص می‌کند کوئری‌ها در چه مدی در حال اجرا هستند. یا ستونی را مشاهده می‌کنیم که با is_parallel_plan مشخص می‌شود و می‌توانیم به راحتی کلیه پلن‌هایی که به سمت موازی‌سازی هدایت شده‌اند را از سیستم استخراج کنیم. پس حتما به صورت مجزا این Catalog Viewها را به ترتیب اجرا کنید تا با اطلاعات سفارشی شده که در هر بخش است آشنا شوید. همچنین یکی از مهمترین ستون‌هایی که در این کتاب با تاکید زیاد بر روی ان اشاره شده ستونی به اسم count_compiles هست که در دو Catalog Views به sys. query_store_query و sys. query_store_planمی توانید مشاهده کنید. اطلاعات این ستون مشخص می‌کند که به چه دفعاتی Query هایی که نوشته شده است compiled یا recompiled شده است. استفاده از پارامترهایی در کوئری باعث می‌شود که Query به ازای هر اجرا، recompiled شود که این به تنهایی ممکن است سرباری بر روی سیستم اعمال کند و در بعضی از موارد Performance کوئری را نیز بهبود دهد. پس کلیه این پارامترها با جزییات دقیقی از این Catalog Viewها قابل استخراج هست

همان طور که توضیح دادیم ما در Query Store سه بخش اطلاعات را ذخیره می کردیم:

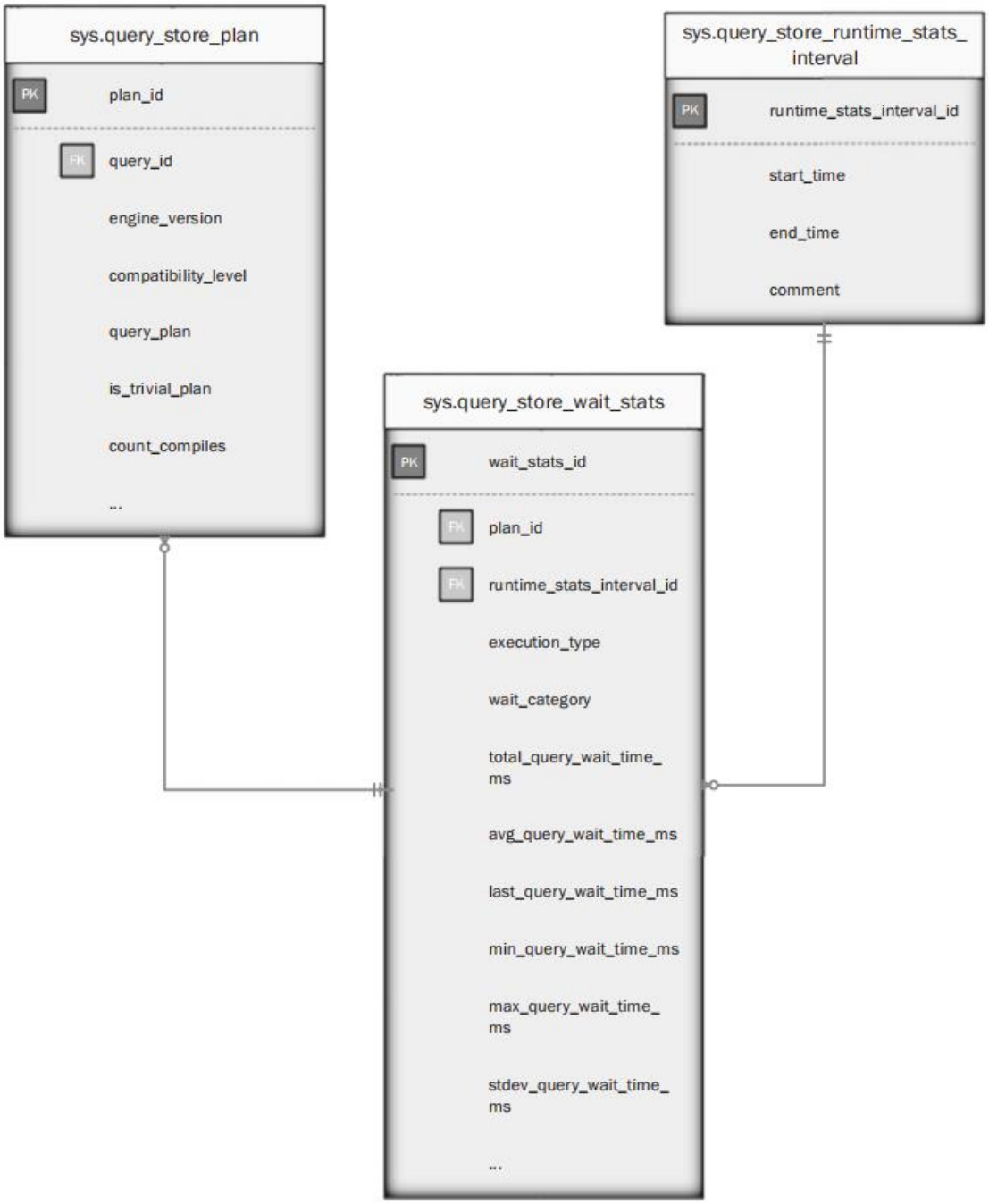
- اطلاعات مرتبط با Query and Plan Information
- اطلاعات مرتبط با Query Runtime Information
- اطلاعات مرتبط با Query Wait Statistics

در قسمت Query and Plan Information شاهد Catalog Views های زیر هستیم :

در قسمت Runtime Information شاهد Catalog Views های زیر هستیم:



در قسمت Wait Statistics نیز شاهد Catalog Views های زیر هستیم:



تا اینجا با تفکیک Data Set ها، جداول و Catalog Views های مرتبط با Query Store آشنا شده شدیم. در قسمت بعد میخواهیم تنظیمات مرتبط با نحوه کانفیگ کردن Query Store و پارامترهای آن را بیشتر مورد بررسی قرار دهیم. نکته ایی که در کتاب بدان شده است در این خصوص است که کلیه این پارامترها در ابتدا به همان صورت پیش فرضی که در در نظر گرفته شده است بررسی می‌شود سپس در مرحله بعد با توجه به سناریوها و بیزینس‌های مختلف و حالت‌هایی که ممکن

است به صورت سیستمی در پارامترها تغییراتی ایجاد شود در قالب Best Practices در نظر گرفته شده است. لذا در ابتدا هر کدام از این پارامترها را تشریح می‌کنیم و در مرحله بعد نکات لازم در خصوص نوع پارمترهای اولویت دار در سیستم را به شما توضیح خواهیم داد.

این نکته را نیز در نظر داشته باشید که کلیه ورژن‌هایی که قابلیت Query Store را پشتیانی می‌کنند، این امکان در آن‌ها به صورت پیش فرض غیر فعال هست. پس در مرحله اول باید نسبت به فعال‌سازی Query Store در ان‌ها اقدام کنیم.

برای فعال سازی Query Store در یک دیتابیس مشخص می توانید از اسکریپت زیر استفاده کنید:

```
ALTER DATABASE [<Database Name>] SET QUERY_STORE=ON;
```

در صورتی که بخواهید این قابلیت را بر روی چند دیتابیس اجرا کنید می توانید از اسکریپت زیر استفاده کنید:

```
DECLARE @SQL NVARCHAR(MAX) = N'';
SELECT @SQL += REPLACE(
        N'ALTER DATABASE [{{DBNAME}}] SET QUERY_STORE=ON ',
        '{{DBName}}',
        [name]
    )
FROM   sys.databases
WHERE  state_desc = 'ONLINE'
    AND [name] NOT IN ('master', 'tempdb')
ORDER BY
    [name];
EXEC (@SQL);
```

استفاده از Query Store برای دیتابیس های سیستمی Master و Tempdb غیر قابل انجام است . همچنین برای روی دیتابیس MSDB می توانید این قابلیت را فعال کنید و همانند دیتابیس های غیر سیستمی Query ها و امار این دیتابیس را نیز رصد کنید . پارامتر هایی که در خصوص Query Store بررسی می کنیم در شکل زیر آورده شده است :

Table 3-1. Configuration options for Query Store

Configuration Name	Options	Description	Default
OPERATION_MODE	OFF READ_WRITE READ_ONLY	Mode of operation for the Query Store	OFF (SQL 2016 and 2016) READ_WRITE for Azure SQL Database
CLEANUP_POLICY(STALE_QUERY_THRESHOLD_DAYS)	BIGINT	Dictates CLEAN_UP policy (0 is never)	30 days (or 7 for Azure SQL Database Basic Edition)
DATA_FLUSH_INTERVAL_SECONDS	BIGINT	Flush interval frequency for buffered Query Store data	900 (15 minutes)
MAX_STORAGE_SIZE_MB	BIGINT	The maximum size of Query Store in MB	100 (SQL Server 2016 and 2017), 1000 (SQL Server 2019), (1024 for SQL Azure Database Premium and 10 for SQL Azure Database Basic Edition)
INTERVAL_LENGTH_MINUTES	1, 5, 10, 15, 30, 60, or 1440	Aggregation interval for statistics	60
SIZE_BASED_CLEANUP_MODE	AUTO OFF	Attempt to clean up if approaching capacity	AUTO
QUERY_STORE_CAPTURE_MODE	AUTO ALL CUSTOM NONE	Query capturing behavior	AUTO (SQL Server 2016 and 2017), ALL (SQL Server 2019) ALL (Azure SQL Database)
MAX_PLANS_PER_QUERY	INT	How many distinct plans to keep per query	200. Not available in SQL Server 2016
WAIT_STATISTICS_CAPTURE_MODE	ON OFF	Specifies rather to capture wait statistics for queries	ON in SQL Server 2017 and 2019 and Azure SQL Database. Not available in SQL Server 2016

❖ پارامتر OPERATION_MODE

این پارامتر مشخص می‌کند که Query Store دیتاهای جمع آوری شده را فقط تحلیل کند یا دیتاهایی که به سیستم وارد می‌شوند را نیز به امار قبلی اضافه کرده و تحلیل آن را نیز انجام دهد. به عبارت دیگر می‌توان گفت که در حالت فقط خواندنی یا read-only تنظیم شود یا در حالت خواندن – نوشتن یا read and write mode تنظیمات ان انجام شود. زمانی که بر روی حالت فقط خواندنی در نظر گرفته شود فقط اطلاعات جمع آوری شده از قبل در Query Store وجود دارد و اطلاعات جدیدی Capture نمی‌شود.

نحوه فعال سازی این پارامتر با استفاده از اسکریپت زیر قابل انجام است:

```
ALTER DATABASE [<Database Name>] SET QUERY_STORE ( OPERATION_MODE = READ_WRITE );
```

در خصوص این پارامتر نکته ایی وجود دارد که در این کتاب بدین شکل به آن اشاره شده است. تحت شرایط خاصی ممکن است که OPERATION_MODE با وجود این که بر روی حالت خواندن – نوشتن در نظر گرفته شده است، به حالت فقط خواندنی تغییر وضعیت دهد. عواملی مختلفی ممکن است باعث بروز همچین سناریو ایی شود. به عنوان مثال فضای کافی برای درج دیتا جدید وجود ندارد و مدت زمان حذف دیتاهای قدیمی نیز هنوز نرسیده است. لذا باید حالتی باشد که فقط اطلاعات موجود وجود داشته باشند و اطلاعات جدیدی به سیستم وارد نشود. در sys. database_query_store_options ستونی وجود دارد به اسم readonly_reason که علت این سوییچ شدن پارامتر به حالت فقط خواندنی را به شما نمایش می‌دهد. به صورت پیش فرض، زمانی که تنظیمات Query Store را در حالت GUI انجام می‌دهید این تنظیم بر روی READ_WRITE. در نظر گرفته شده است

❖ پارامتر CLEANUP_POLICY (STALE_QUERY_THRESHOLD_DAYS)

این پارامتر تعداد روزهایی رو مشخص می‌کند که اطلاعات قدیمی نیاز به ذخیره شدن دارند و بعد از ان باید از Query Store حذف شوند. به صورت پیش فرض مقدار این پارامتر بر روی عدد ۳۰ در نظر گرفته شده است.
نحوه فعال سازی این پارامتر با استفاده از اسکریپت زیر قابل انجام است:

```
ALTER DATABASE [<Database Name>] SET QUERY_STORE ( CLEANUP_POLICY = (
STALE_QUERY_THRESHOLD_DAYS = <Value> ) )
```

❖ پارامتر DATA_FLUSH_INTERVAL_SECONDS

این پارامتر مدت زمان مورد نیاز برای انتقال دیتا از حافظه به دیسک را در نظر میگیرد. با توجه به فرایند آسنکرون که کمی قبل‌تر توضیح داده شد، برای انتقال اطلاعات از حافظه به سمت دیسک، باید زمانی مشخص می‌شد که در بازه‌های مختلف، اطلاعات برای تحلیل و ذخیره‌سازی، منتقل شود. با استفاده از این پارامتر می‌توانید این مدت زمان را مشخص کنید.

نکته » تنظیم زمانی این پارامتر از دو جنبه حائز اهمیت است:

۱. مدت زمان در بازه های زمانی کوتاه
- در این حالت سربار I/O به سیستم تحمیل می شود و ممکن است در بعضی از سناریو ها با کندی های همراه باشد

۲. مدت زمان در بازه های زمانی طولانی
- در این حالت ممکن است به هر دلیل اطلاعات موجود در حافظه با ریست شدن یا خرابی سرور از بین برود .

لذا این پارامتر باید با توجه به دو نکته ایی که در متن بالا بدان اشاره شد تنظیم شد .
نحوه فعال سازی این پارامتر با استفاده از اسکریپت زیر قابل انجام است:

```
ALTER DATABASE [<Database Name>] SET QUERY_STORE ( DATA_FLUSH_INTERVAL_
SECONDS = <Value> );
```

به صورت پیش فرض مدت زمان آن بر روی ۹۰۰ ثانیه در نظر گرفته شده است. عددی که در این قسمت وارد می‌شود نیز بر حسب ثانیه در نظر گرفته می‌شود.

❖ پارامتر MAX_STORAGE_SIZE_MB

این پارامتر میزان فضای ذخیره‌سازی برای Query Store و اطلاعات آن را مشخص می‌کند. به صورت پیش فرض بر روی ۱۰۰۰ مگابایت در نظر گرفته شده و واحد اصلی ان نیز بر حسب مگابایت هست. در صورتی که این فضا به حداکثر خود برسد تغییری در وضعیت ذخیره‌سازی دیتا در پارامتر OPERATION_MOD شاهد خواهیم بود.

نحوه فعال سازی این پارامتر با استفاده از اسکریپت زیر قابل انجام است:

```
ALTER DATABASE [<Database Name>] SET QUERY_STORE ( MAX_STORAGE_SIZE_MB =
<Value> );
```

❖ پارامتر INTERVAL_LENGTH_MINUTES

این پارامتر مشخص می کند که در چه بازه های زمانی دیتا از حالت aggregated که در حافظه موجود است ، به حالت Re-aggregated تبدیل و برای نمایش در چارت ها و داشبورد های تحلیل Query store اماده سازی شود . تنظیم این پارامتر فقط می تواند بر روی اعداد ۱, ۵, ۱۰, ۱۴, ۶۰, ۱۴۴۰ انجام شود که به صورت پیش فرض بر روی ۶۰ در نظر گرفته شده است

نحوه فعال سازی این پارامتر با استفاده از اسکریپت زیر قابل انجام است:

```
ALTER DATABASE [<Database Name>] SET QUERY_STORE ( INTERVAL_LENGTH_MINUTES
= <Value> );
```

❖ پارامتر QUERY_STORE_CAPTURE_MODE

به صورت پیش فرض مقدار این پارامتر برابر است با ALL. در این حالت کلیه مواردی که در بخش DataSet به آن اشاره شد ذخیره می‌شود. در صورتی که برای مقدار None تنظیم شود فقط اطلاعات مرتبط با Runtime Query ها جمع آوری می‌شود. همچنین در صورتی که بر روی مقدار AUTO تنظیم شود درخواست‌هایی که منابع قابل توجهی را استفاده می‌کند را ذخیره نمی‌کند.

نحوه فعال سازی این پارامتر با استفاده از اسکریپت زیر قابل انجام است:

```
ALTER DATABASE [<Database Name>] SET QUERY_STORE ( QUERY_STORE_CAPTURE_MODE
= [<Value>] );
```

❖ پارامتر MAX_PLANS_PER_QUERY

این پارامتر مشخص می‌کند که به ازای هر Query چند پلن اجرایی توسط Query Store ذخیره شود. به صورت پیش فرض برای هر کوئری ۲۰۰ پلن اجرایی ذخیره می‌شود. نکته‌ای که مهم هست، تعداد پلن‌ها در صورت تغییر دادن این مقدار است. ممکن هست که فضای بیشتری بر روی دیسک نیاز باشد که پلن‌های اضافی ذخیره شود. پیشنهاد این کتاب نیز بر روی همین عدد هست. همچنین در نسخه SQL SERVER ۲۰۱۶ این قابلیت نیست پس باید به ورژنی که استفاده می‌کنید دقت کنید.

به صورت کلی برای این که کل پلن های اجرایی یک کوئری را مشاهده کنیم می توانیم از کوئری زیر استفاده کنیم:

```
SELECT query_hash,
    COUNT(DISTINCT query_plan_hash) distinct_plans
FROM   sys.dm_exec_query_stats
GROUP BY
    query_hash
ORDER BY
    distinct_plans DESC;
```

همان طور که در شکل پایین ملاحظه می کنید به ازای هر Query Hash ایی که داریم تعداد پلن های اجرایی ان را نمایش می دهد.

```
SELECT query_hash,
       COUNT(DISTINCT query_plan_hash) distinct_plans
FROM   sys.dm_exec_query_stats
GROUP BY
       query_hash
ORDER BY
       distinct_plans DESC;
```

132 %

Results | Messages

	query_hash	distinct_plans
1	0x1E0BB7E280669FF9	4
2	0x03222F716E4AD5EF	4
3	0x25CD3566292AE736	4
4	0x4A5D98C71CBD4BF6	4
5	0x7DA70F4574D2E28D	4
6	0xFFCEA355FBFC28D7	4
7	0x94376DBEDDE8EE35	3
8	0xB44D9485B3B7A005	3
9	0xDAC2842BD871BC68	3
10	0xE6977637E3E86092	3
11	0xED300550ADED823E	3
12	0x5AF1581363E257FF	3
13	0x069EDD3350AF7893	3
14	0x4ED774DFD0AAA529	3
15	0x7F0E349E867CA588	3
16	0x80E61CED948E5F9C	3
17	0x826201C4AEF25313	3
18	0x6B2670BDAD77D614	3
19	0x44B0C8F3244C2DEC	3
20	0x24819C19C40A7361	3
21	0x24C59CD26A21ED84	3
22	0x22C7B8EA78612F18	2

نحوه فعال سازی این پارامتر با استفاده از اسکریپت زیر قابل انجام است:

```
ALTER DATABASE [<Database Name>] SET QUERY_STORE ( MAX_PLANS_PER_QUERY =
<Value> );
```

پارامتر WAIT_STATISTICS_CAPTURE_MODE

با استفاده از این پارامتر می توانیم کلیه Wait های مرتبط با کوئری هایی که جمع آوری می شود را نیز ذخیره کنیم . همان طور که گفته شد این قابلیت در ورژن SQL SERVER ۲۰۱۶ موجود نیست و از ورژن ها بعد از ان می توانید استفاده کنید .

نحوه فعال سازی این پارامتر با استفاده از اسکریپت زیر قابل انجام است:

```
ALTER DATABASE [<Database Name>] SET QUERY_STORE ( WAIT_STATISTICS_CAPTURE_
MODE = <Value> );
```

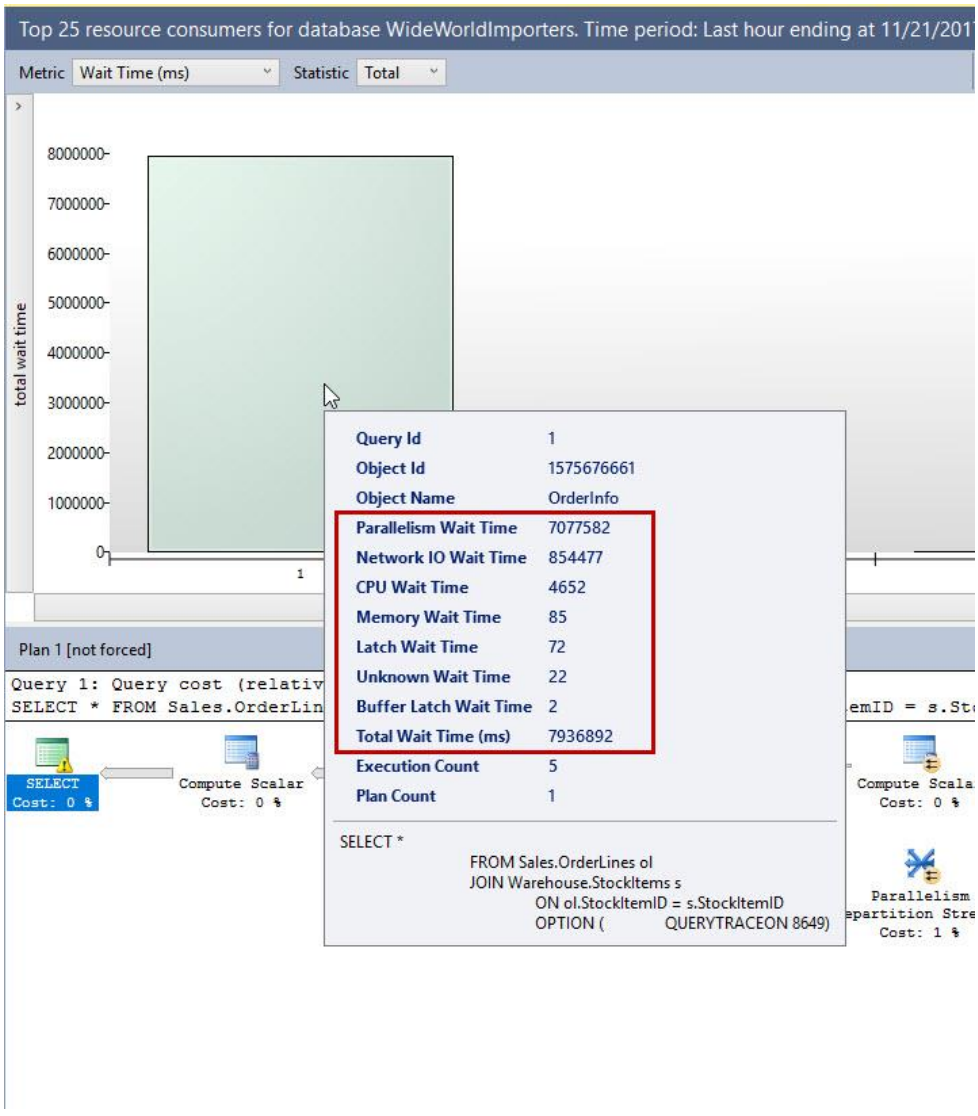
مبحث Wait ها در Query Store شامل بخش‌های مختلف می‌شود. این که در واقع چه نوع Wait Type هایی در این بخش پوشش داده می‌شود و مباحث فنی‌تر ان که در مقالات بعدی به بررسی آن خواهیم پرداخت. همچنین می‌توانید با استفاده از اسکریپت زیر کلیات مرتبط با مبحث Wait ها را از Catalog Views هایی که معرفی شد مشاهده نمایید:

```
SELECT TOP(۲۵)
    [ws].[wait_category_desc],
    [ws].[avg_query_wait_time_ms],
    [ws].[total_query_wait_time_ms],
    [ws].[plan_id],
    [qt].[query_sql_text],
    [rsi].[start_time],
    [rsi].[end_time]
FROM   [sys].[query_store_query_text] [qt]
    JOIN [sys].[query_store_query] [q]
      ON  [qt].[query_text_id] = [q].[query_text_id]
    JOIN [sys].[query_store_plan] [qp]
      ON  [q].[query_id] = [qp].[query_id]
    JOIN [sys].[query_store_runtime_stats] [rs]
      ON  [qp].[plan_id] = [rs].[plan_id]
    JOIN [sys].[query_store_runtime_stats_interval] [rsi]
      ON  [rs].[runtime_stats_interval_id] = [rsi].[runtime_stats_interval_id]
    JOIN [sys].[query_store_wait_stats] [ws]
      ON  [ws].[runtime_stats_interval_id] = [rs].[runtime_stats_interval_id]
      AND [ws].[plan_id] = [qp].[plan_id]
WHERE  [rsi].[end_time] > DATEADD(MINUTE, -۵, GETUTCDATE())
```

```
    AND [ws].[execution_type] = 0
ORDER BY
    [ws].[avg_query_wait_time_ms] DESC;
```

	wait_category_desc	avg_query_wait_time_ms	total_query_wait_time_ms	plan_id	query_id	query_sql_text	start_time	end_time
1	Parallelism	1422782.625	11382261	1	1	SELECT * FROM Sales.OrderLine...	2017-11-21 23:50:00.0000000 +00:00	2017-11-22 00:00:00.0000000 +00:00
2	Network IO	172690	1381520	1	1	SELECT * FROM Sales.OrderLine...	2017-11-21 23:50:00.0000000 +00:00	2017-11-22 00:00:00.0000000 +00:00
3	CPU	91.5	732	1	1	SELECT * FROM Sales.OrderLine...	2017-11-21 23:50:00.0000000 +00:00	2017-11-22 00:00:00.0000000 +00:00
4	Latch	17.375	139	1	1	SELECT * FROM Sales.OrderLine...	2017-11-21 23:50:00.0000000 +00:00	2017-11-22 00:00:00.0000000 +00:00
5	Memory	14.5	116	1	1	SELECT * FROM Sales.OrderLine...	2017-11-21 23:50:00.0000000 +00:00	2017-11-22 00:00:00.0000000 +00:00
6	Memory	2	2	13	51	(@interval_end_time datetimeoffset(7...	2017-11-21 23:50:00.0000000 +00:00	2017-11-22 00:00:00.0000000 +00:00

همچنین اطلاعات تکمیلی هر پلن اجرایی در خصوص Wait ها را می توانید با کلیک بر روی هر چارت مشاهده کنید :



کلیه پارامتر های اشاره شده نیز قابل تنظیم از طریق GUI نیز هستند. همان طور که در عکس پایین مشاهده می کنید مسیر دسترسی برای تغییر این پارامتر ها بدین شکل هست: