**Intro to Query Store Notes (Enrico van de Laar)**

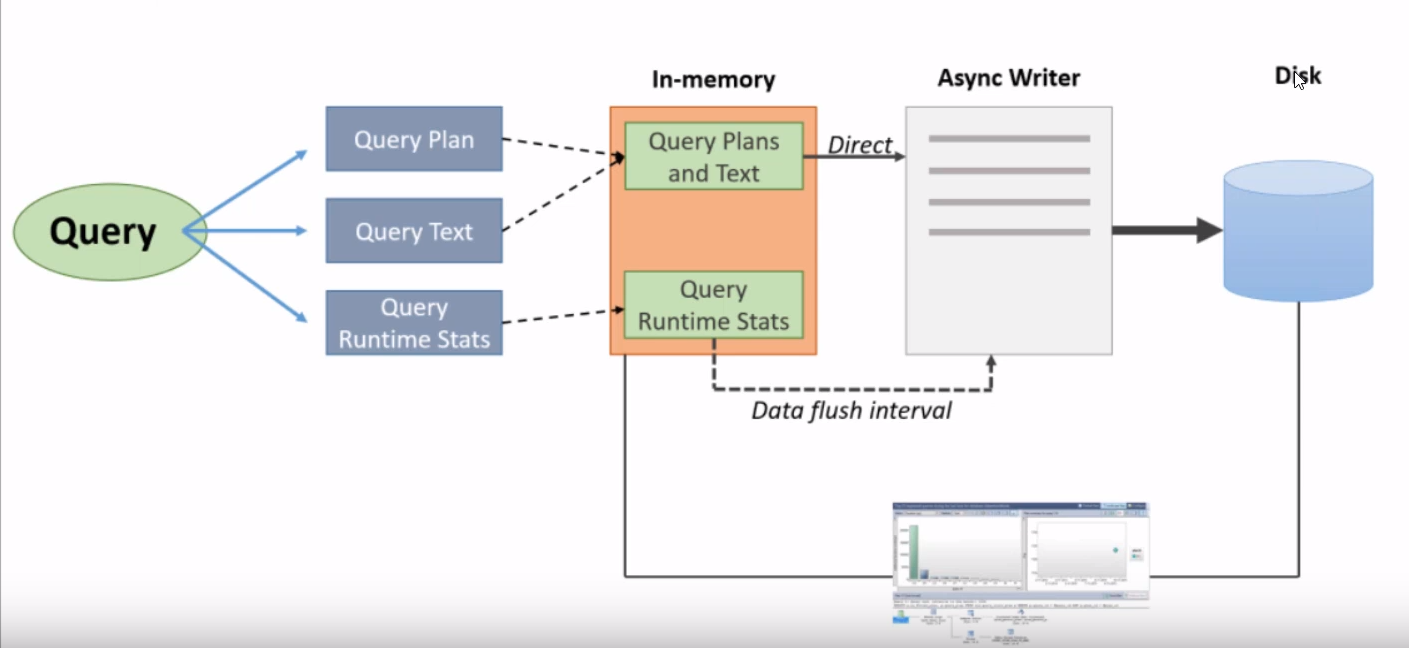
**What is Query Store? –**

* SQL Servers ‘flight-recorder’
* “Captures the performance of queries executed against a particular database”
  + Collects performance information such as duration of query, rows processed etc.
* It isn’t a store where you can buy queries… \*facepalm\*

**Details**

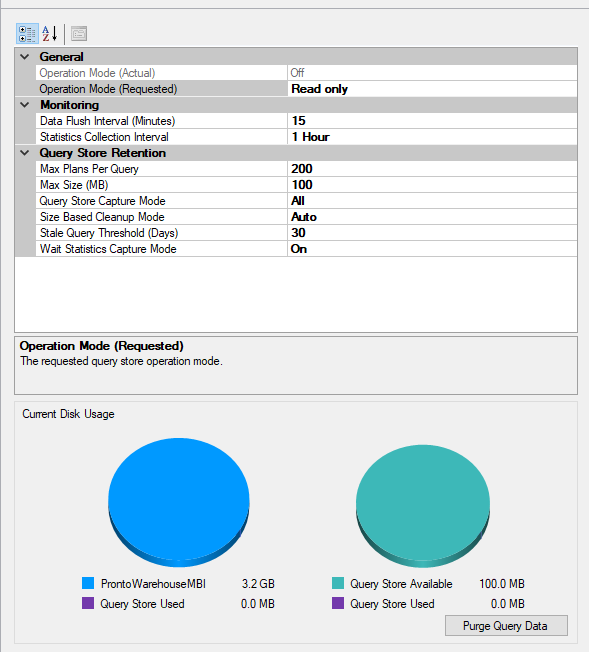
* Configured at a database level
* Stores on disk / storage subsystem (unaffected by restarts / failovers)
* Quickly and easily analyse performance using built-in reports and dmvs **(GH: Can we customise?)**
* Retains query history for as long as you want… 2 years, 3 years… just need enough disk space
* Integrated into SQL Server Engine… grabs information from the query and stores it directly… rather than pulling from the plan cache after query processing is complete
* Available on Standard
* Allows access for developers (everyone!) to determine query performance

**Architecture**



* **Query –** When a query is **executed**, it grabs 3 core pieces of information:
  + **Query Plan –** Basically the **estimated** execution plan
  + **Query Text –** If you have multiple queries, it will store each query separately
  + **Query Runtime Stats –** Collects for each statement
* **In-Memory**
  + **Query Plan** and **Query Text** is buffered in-memory
  + **Plans and Text** and **Runtime Stats** have their own buckets **(GH: Why?)**
* **Async Writer**
  + New or Unique Query Plans and Text (ie. executed for the first time) will directly be passed on to a process called the Async Writer
  + **Async Writer** –
    - a collection of jobs which SQL Server has to write things to disk for
    - “Think of it as a list where SQL server places a bit of a work order like – “ok please write this query plan to disk…”. The async writer will then write it to disk when it’s ready
    - Query Plans and Text will still be in the query store, as they are flushed to disk **practically immediately**… **if SQL Server crashes, it’s unlikely that you will lose this information**
    - The Query Runtime Stats are not directly written to the async writer – they are written at a specific interval which is dependant on a setting called **Data Flush Interval,** where they will eventually be hardened **(plays a huge role in the performance of data store)**
    - If you have a huge Data Flush Interval setting and SQL Server crashes, **you will lose all the runtime stats that haven’t been written to disk yet**
* “Everything we do with the DMV’s and reporting, it just gathers the information together… so it doesn’t matter if the data is in memory or already on disk, it will gather it and it will show you in the built in reports or dmvs – we don’t have write separate queries for on-disk / on-memory tables, everything is combined together”

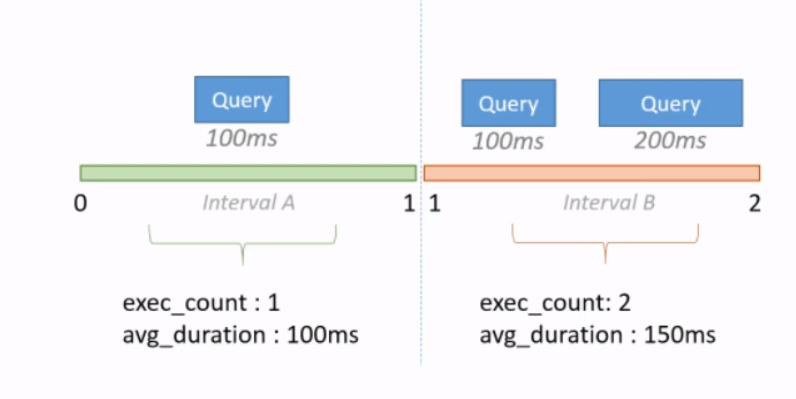
**Database Settings**



* **Statistics Collection Interval –** how long data should be aggregated together (covered later)
* **Max Size (MB)** – Allows the query store data to grow to a maximum of (eg.) 200MB
  + Note – The Query Store is stored in the **PRIMARY file of the user database** (not in a separate database)
* **Query Store Capture Mode –** What type of queries should the Query Store capture?
  + **All / None –** Self explanatory
  + **Auto –** Capture Query based on Resource Consumption
* **Size Based Cleanup Mode –** Once it hits a certain size, it will automatically clean up really old data (runtime stats), or queries that haven’t been executed in **X** number of days (determined by **Stale Query Threshold (Days)**
* **Wait Statistics Capture Mode –** Captures Wait Statistics
* **Purge Query Data –** Empties Query Store

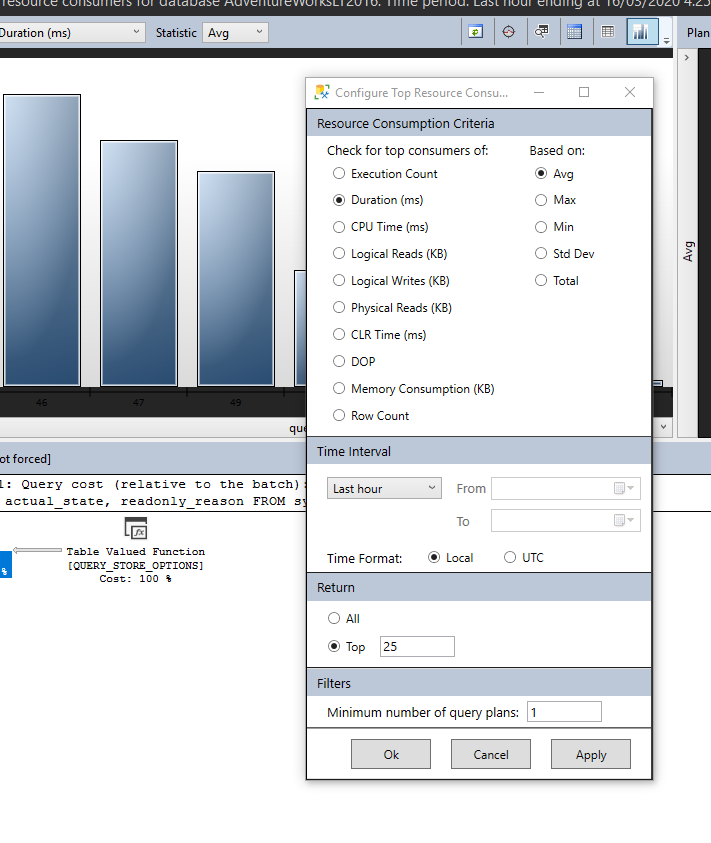
**Query Performance Metrics**

* Records and aggregates query runtime stats using **time intervals** (default – 1 hour)
  + Remember that you are not looking at a single query execution, but rather an aggregation of multiple executions of the same query
* The Time Interval depends on **Statistics Collection Interval** settings… it plays a huge role in that the lower you set your interval, the more granular the data is
  + The lower the Interval – more granularity, but more storage cost
  + The higher the Interval – less storage overhead, less granularity
* **Understanding query performance metrics**



* + When multiple queries take place in the intervals same interval, the **exec\_count** will display the total number of queries that took place, which will affect the **avg\_duration** of all queries taking place

**Analysing data inside the Query Store -** two methods –

* + **Built in reporting** - Provides additional, easy access, features like plan comparison and forcing. Some say the reports are not fully featured, but new reports are added after each release of SSMS
  + **Top Resource Consuming Queries -** 
  + Able to configure against multiple metrics, using various aggregates, time interval, number of queries, and set a minimum number of query plans
* **DMVs** – programmatically accessing QS data
  + can be used to overcome the perceived limitations of the built-in reports

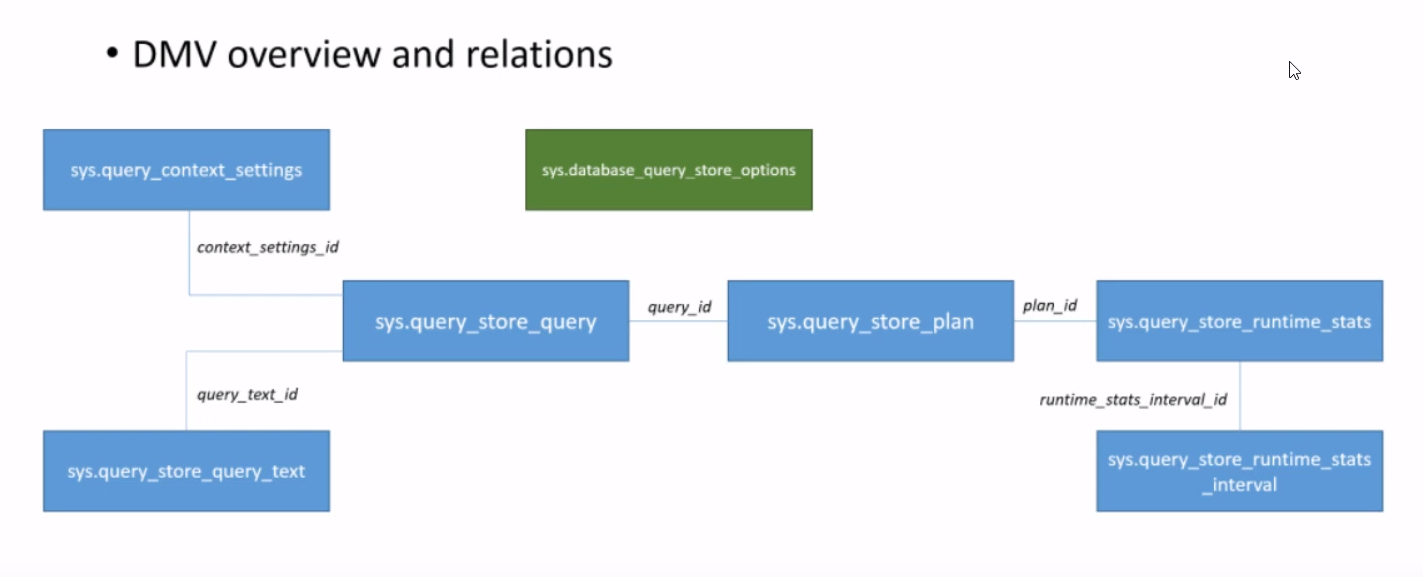
**Query Store Metadata (Querying the Query Store)**

-- Config Settings

SELECT \* FROM sys.database\_query\_store\_options

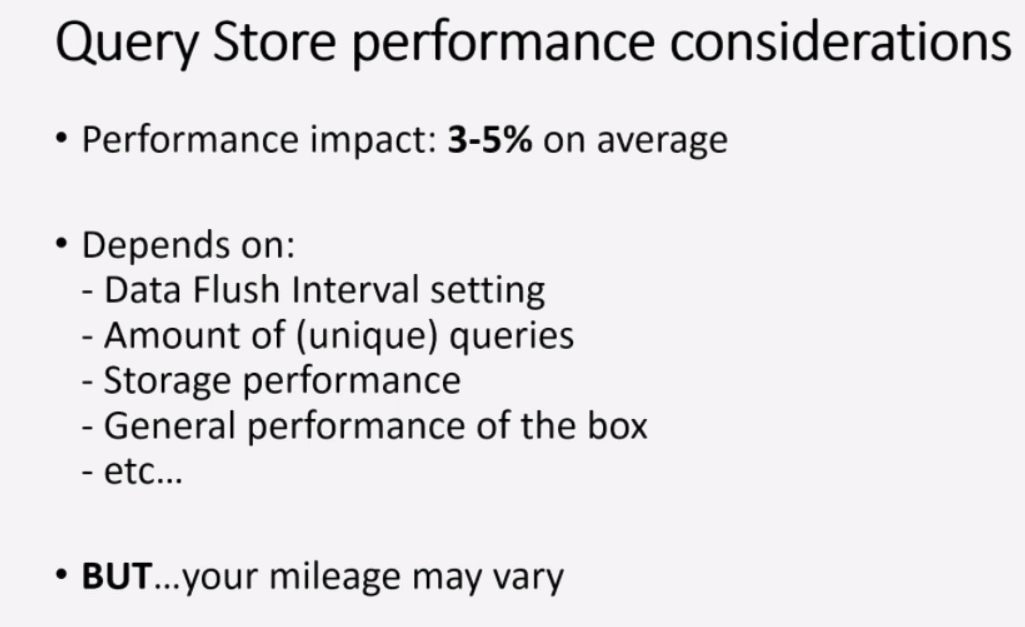
-- Runtime Stats - gateway to all query performance. Records metrics that record how the query was executed.

SELECT \* FROM sys.query\_store\_runtime\_stats



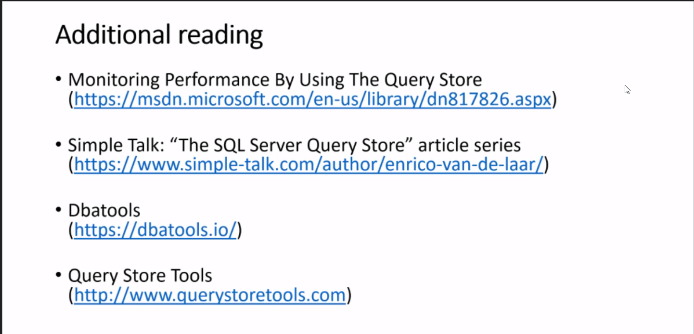
**Forcing**

* Very easy to configure, saves a lot of time to troubleshoot, however….
* If you force the plan, it keeps the forced plan **FOREVER!** This setting persists until you unforce the plan manually!



**Measuring Query Store Performance**

* Perfmon
* Wait Statistics (QDS\_)
* Extended Events



**QnA**

* + **How come it uses Estimated, rather than Actual? –** Actual would return a new query plan for each execution, estimated only returns one
  + **What happens with log shipping / Always On?** – Query store would be Read Only?
  + Using sys.query\_store\_plan, we can keep track if a plan is forced or not (**is\_forced\_plan**)
  + Turning off the Query store will NOT clear the current data cache – only purge items does this
  + Currently, Query store can ONLY be configured to use the primary group

**SQL Server Query Store – Overview (2017,** <https://www.sqlshack.com/sql-server-query-store-overview/>)

* Common scenarios where the SQL Server Query Store feature can be useful:
  + Find the most expensive queries for CPU, I/O, Memory etc.
  + Get full history of query executions
  + Get information about query regressions (a new execution plan generated by query engine is worse than the older one)
  + Quickly find performance regression and fixing it by forcing the previous query plan whose performances are much better than a newly generated plan
  + Determine how many times a query was executed in the given range of time

**SQL Day 2019 – The Query Store and Query Tuning** (Grant Fritchey - <https://www.youtube.com/watch?v=TlanVeGci8s>)

* Can be an ‘undo button’… why?
  + **Use Case:** Cardinality Estimator from SQL 2014
    - **Situation:** SQL Server 2014 introduced a change to the cardinality estimator, which assigned a select number of queries sub-optimal plans, with no means to undo the problem
    - **Action:** Query Store (introduced in SQL Server 2016) gives us an undo button!
    - **Result:**
    - **Summary:**
* Captures **all** query plans by default…
  + With cache, queries with recompile hints will NOT enter cache
  + Whereas Query store will capture it
* Captures runtime statistics at an **aggregated** view, across a specified interval (default: 60 mins)
  + How long did the query take?
  + How much cpu did it use?
  + How many writes did it have
  + Wait Stats per query (2017 onwards)
* Not a replacement for extended events, which captures absolute detail at the lowest grain
* Grain of the **Query Store Data**
  + **Query** (sys.query\_store\_query) consists of:
    - Query Text (sys.query\_store\_query\_text)
    - Query Plan (sys.query\_store\_query\_plan)
  + **Runtime Views**
    - Runtime Stats (sys.query\_store\_runtime\_stats)
    - Runtime Stats Interval (sys.query\_store\_runtime\_Stats\_interval)
    - Wait Stats (sys.query\_store\_wait\_stats)
* **In Memory OLTP**…
  + Exactly the same… except
  + Not Persisted
  + Use **sys.sp\_xtp\_control\_query\_exec\_stats**
  + Plan forcing can fail
  + No Memory Grants
  + Need to recompile to capture metrics

**Commands:**

* **sys.sp\_query\_store\_flush\_db** – force contents to disk (rather than waiting for 15 minute interval)
* **sys.fn\_stmt\_sql\_handle\_from\_sql\_stmt -**

**Top Resource Consumer Report**

* Note that the Query Plans in the bottom pane are **“Estimated Plans”**
* Quote from Grant Fitchey: “All plans are estimated, it’s just that some plans have runtime statistics”

**Summary**

* Understand the information and mechanisms behind the Query Store
* Learn how to work with an maintain the Query Store
* Discover new opportunities and mechanisms for dealing with query performance tuning issues